UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Desarrollo de agente inteligente y *dashboard* para contribuir al control de Leishmaniasis Cutánea en Guatemala

Trabajo de graduación en la modalidad de Megaproyecto presentado por Diego Leonel Sevilla de León, José Alejandro Tejada Flores, José Pablo Cifuentes Sánchez y Oscar Esteban Juárez Paz para optar por el grado académico de Licenciados en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2021

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Desarrollo de agente inteligente y *dashboard* para contribuir al control de Leishmaniasis Cutánea en Guatemala

Trabajo de graduación en la modalidad de Megaproyecto presentado por Diego Leonel Sevilla de León, José Alejandro Tejada Flores, José Pablo Cifuentes Sánchez y Oscar Esteban Juárez Paz para optar por el grado académico de Licenciados en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2021

Vo.Bo:

(f) 

Ing. Tomás Galvez

Tribunal examinador:

(f) 

MSc. Douglas Leonel Barrios González

Fecha de aprobación: Guatemala 13 de diciembre 2021

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

La Leishmaniasis cutánea es sin dudar la forma clínica más frecuentemente observada en Guatemala, siendo tan notoria y específica en regiones selváticas que se cataloga como endémica. El objetivo principal del Proyecto es desarrollar e implementar una solución para el estudio, catalogación y monitoreo frecuente de la enfermedad en regiones del norte de Guatemala.

Para cumplir con el objetivo se diseñaron, implementaron y probaron dos soluciones entrelazadas e interdependientes: una aplicación móvil con un agente inteligente capaz de detectar Leishmaniasis cutánea y un portal de datos estadísticos e históricos de la enfermedad para su uso y análisis. La aplicación ayudaría a combatir el impacto de la enfermedad en épocas estacionales, con fácil implementación y de gran utilidad para los agentes de campo. Por otro lado, el portal informativo es un espacio donde se pueden actualizar y mostrar los registros históricos para su cuantificación y análisis por investigadores relacionados con el combate a la enfermedad. Ambas soluciones fueron trabajadas desde sus cimientos en cuatro módulos principales: *Frontend*, *Backend*, Infraestructura e inteligencia artificial. Cada módulo obtuvo resultados y desarrolló avances de forma independiente, pero a su vez conectados al resto.

El módulo de inteligencia artificial fue el primero en ser desarrollado, el agente inteligente es el encargado de analizar y tomar decisiones sobre lesiones de Leishmaniasis e identificarlas adecuadamente. El resultado fue que el agente fue capaz de diagnosticar Leishmaniasis cutánea en un teléfono móvil con especificaciones asequibles. El agente validó un conjunto de imágenes obtenidas para entrenarlo con una precisión del 90.95% usando el modelo InceptionV3, aunque se necesitarían más imágenes para mejores resultados.

En cuanto al módulo de *Backend* e Infraestructura, estos son las columnas en las cuales se basan los servicios y conexiones de aplicaciones. Se creó un diseño de base de datos adecuado, se usó Node.js para satisfacer las necesidades que requieren los sitios WEB de APIS modernos y se montó la infraestructura en Amazon Web Services. Además, se hizo uso de otros servicios clave como AWS S3, Route 53, Amazon CloudFront, y demás. Esto resultó en autentificación de usuarios en el portal, acceso restringido, seguridad robusta y manejo de errores en ambas soluciones.

Finalmente, en el módulo de *Frontend*, que es el encargado de diseñar e implementar la interfaz con la que el usuario interactúa, se hizo uso de patrones de toma de retroalimentación cuantitativos y cualitativos. Un 45% de personas indicaron que el tablero de Leishmaniasis fue de su agrado, un 56% respondió favorablemente ante las funcionalidades atómicas y un 67% de persones indican que sí usarían el tablero. Finalmente, las pruebas de usabilidad arrojaron resultados positivos en interacción y manejo de flujos para los usuarios.

I. INTRODUCCIÓN

Los efectos de las enfermedades pueden variar dependiendo de muchos factores, entre ellos, la región o lugar de origen del cual provengan. La Leishmaniasis cutánea, es una de ellas. El objetivo del presente trabajo es desarrollar e implementar una solución para el estudio, la catalogación y el constante monitoreo de la enfermedad proveniente de la región norte de Guatemala.

La Leishmaniasis es causada por un parásito del género Leishmania, el cual cuenta con más de veinte especies distintas y se transmiten a los animales y humanos a través de la picadura del parásito. Dado que la enfermedad es endémica en Guatemala y posee etapas de alto y bajo contagio, el estudio y control de la Leishmaniasis es constante. Es en este punto donde se propone una implementación estratégica y confiable para mantener la información sobre la enfermedad lo más actualizada posible. Una aplicación que ayude a combatir el impacto de una enfermedad estacional, fácil de implementar y de utilidad para los agentes de campo.

Partiendo de las necesidades actuales, portales informativos o espacios públicos en donde se pueda mantener un registro actualizado de la cantidad de casos por región son herramientas que ayudan a tomar decisiones acorde a la información recabada. Es así donde nace la idea de crear un portal cuya presentación, cuantificación y visualización adecuada de los datos proveen un valor alto a investigadores o agentes de salud para darle una mejor atención a la población. Además del portal, la introducción de la inteligencia artificial en ramas más allá de la computación clásica se ha convertido en algo mucho más grande. Respecto a la medicina, Giovanny Briganti describe en su artículo Artificial Intelligence in Medicine: Today and Tomorrow que es debido a ciertas razones. Entre ellas que la AI revoluciona tecnologías médicas para hacerlas entendibles fácilmente y lidiar con problemas complejos en áreas donde hay muchísima información, pero poca teoría. Otra razón es que la AI orientada a medicina cumple con un criterio llamado 4P (Predictiva, preventiva, personalizada y Participativa) que influye en el paciente. Finalmente, el auge de los smartphones da acceso a las personas de mejor manera. Esas razones afirman desarrollo de un agente inteligente para la detección de Leishmaniasis cutánea y su posterior conexión con el portal de visualización de forma efectiva.

II. OBJETIVOS

A. Generales

1. Desarrollar un agente inteligente en apoyo al diagnóstico de la Leishmaniasis cutánea en regiones endémicas de Guatemala.
2. Desarrollar un sistema para resumir y visualizar gráficamente los datos epidemiológicos sobre la Leishmaniasis cutánea en Guatemala.

B. Específicos

1. Desarrollar un agente inteligente para la identificación de úlceras de Leishmaniasis cutánea en apoyo al diagnóstico diferencial.
2. Validar el agente inteligente desarrollado para la identificación de úlceras de Leishmaniasis cutánea.
3. Implementar un panel de control que permita a autoridades del subprograma de Leishmaniasis a tomar decisiones basadas en datos.
4. Desarrollar una aplicación móvil que integre información epidemiológica y el reconocimiento de lesiones para predecir casos sospechosos de Leishmaniasis cutánea, en apoyo al diagnóstico de los técnicos de salud.
5. Desarrollar y validar un agente inteligente para la identificación de úlceras de Leishmaniasis cutánea en apoyo al diagnóstico diferencial.
6. Proporcionar información que aporte valor de la situación actual de Leishmaniasis a autoridades del Subprograma de Leishmaniasis.

III. JUSTIFICACIÓN

La Leishmaniasis cutánea es la forma clínica más frecuentemente observada en Guatemala, siendo El Petén y Alta Verapaz los departamentos que reportan el mayor número de casos anualmente. La enfermedad también se presenta en otros departamentos como Quiché, Izabal, Baja Verapaz y Huehuetenango (Copeland *et al*, 1990 y Arana *et al*, 2000). Clínicamente, la Leishmaniasis cutánea puede ir desde una úlcera pequeña y única, hasta úlceras múltiples de gran tamaño (World Health Organization, 1990).

Durante el 2000-2007 un 77% de los casos en Guatemala (4837/5709) fueron confirmados en personas con una edad mayor de 10 años. Un total del 67% de los casos fueron hombres. Los factores asociados a la enfermedad fueron:

- La transmisión se dio en áreas boscosas.
- La ocurrencia es marcada en el género masculino mayores de 10 años (la mayoría trabajadores jornaleros y de áreas boscosas).

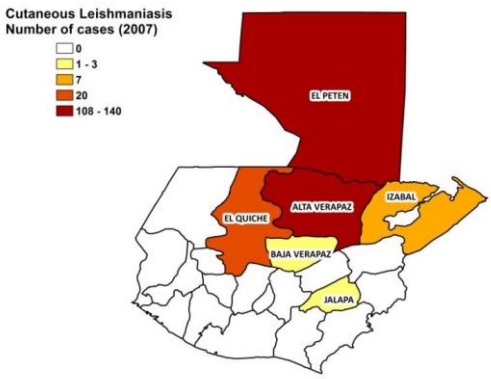
En el año 2008, se reportaron incidencias con un total de 0.1 casos por cada 100,000 habitantes (Arana,1998). En el país de Guatemala, la Leishmaniasis cutánea (abreviada CL para propósitos de este documento), está causada por dos especies del parásito:

- *Leishmania-L. braziliensis (CL)*
- *Leishmania-L. mexicana (ML)*

La especie L. Braziliensis evoluciona de forma rápida. No se cura sin terapia específica y a menudo responde rápidamente a los tratamientos. La especie L. Mexicana en cambio, crece de forma más lenta, frecuentemente se cura sin tratamiento o terapia especializada pero no responde bien a los tratamientos (Herwaldt *et al*, 1992). El primer caso de Leishmaniasis visceral fue reportado en 1949 en un niño de dos años. La incidencia de esta enfermedad en Guatemala es baja. De 2000-2007 fueron diagnosticados 67 casos. De cualquier forma, la población de área endémica es la que está en mayor riesgo.

A. Mapas y tendencias

Ilustración 1:Número de casos e incidencia de Leishmaniasis Cutánea en Guatemala.



(Tomado de Herwaldt *et al.*, 1992)

B. Vigilancia, diagnóstico y tratamiento

1. Vigilancia

La tasa de incidencia de la Leishmaniasis cutánea (CL) en Guatemala ha aumentado de 3,14 por 100.000 habitantes en 2011 a 6,24 por 100.000 habitantes en 2019, (Ilustración 2), debido a una combinación de mejor vigilancia activa, cambios climáticos y actividades ocupacionales que implican un contacto continuo con los bosques, entre otros. Desde 2003 se ha implementado un programa nacional de control, pero los esfuerzos continuos han generado resultados desiguales para las diferentes comunidades endémicas (Pérez *et al*, 2020)

Las acciones a nivel nacional para el control de la CL incluyen la vigilancia activa y pasiva, el diagnóstico y el tratamiento ofrecidos sin costo por el Ministerio de Salud Pública y Asistencia Social (MSPAS) y actividades de sensibilización. Se exploran las barreras y los facilitadores del control de LC en Guatemala según lo experimentado y percibido por los usuarios y proveedores de servicios con el fin de informar estrategias basadas en evidencia para fortalecer el control de LC en el país (MSPAS, 2018).

2. Diagnóstico

La Leishmaniasis cutánea se caracteriza por la aparición de una o más lesiones en zonas descubiertas del cuerpo, los lugares más frecuentes son cara, cuello, brazos y piernas. En el lugar de inoculación, aparece una pápula que usualmente crece y se convierte en una úlcera indolora, la cual es usualmente redondeada, con una superficie de apariencia granulosa y limpia cuando no existe infección por bacterias y recubierta de costra, con bordes que característicamente cubren toda la úlcera los cuales son elevados e indurados. (Soto, C. *et al*,2018) Para diagnosticar la enfermedad se usan por lo general dos métodos:

- CL: confirmación vía microscopio
- Laboratorios

3. Diagnóstico de laboratorio

Se realizan cuatro frotes para obtener material del borde de la lesión. (Para más información consultar el Manual de Procedimientos de la Unidad de Diagnóstico del Laboratorio Nacional de Salud). Ante la sospecha clínica o epidemiológica de Leishmaniasis, es necesario visualizar el parásito para corroborar el diagnóstico por métodos de laboratorio. (Soto, C. *et al*,2018)

4. Tratamiento

- Conducta/tratamiento
- Proceda según Protocolo de Vigilancia Epidemiológica
- Realice frote directo de la manera siguiente
- Rotule dos láminas porta objetos en uno de los extremos con el número de registro del laboratorio.
- Escoja la lesión más nueva y activa, esto es la lesión con los bordes más elevados.

- La lesión debe estar libre de infección. Si estuviera infectada, debe tratarse primero la infección y después tomar la muestra.
- Retire la costra de la superficie, si estuviera presente.
- Elija el borde más activo de la lesión y realice el raspado con una hoja de bisturí. Hágalo de tal manera que no sangre, si esto sucede limpie la sangre con una gasa estéril.
- Se deben tomar dos láminas y hacer dos extendidos uno de cada uno de los bordes opuestos de la lesión.
- Deje secar las muestras a temperatura ambiente, de preferencia fijar la lámina con alcohol absoluto.
- Envíe la muestra inmediatamente al laboratorio del nivel local. Si esto no fuera posible, refiera el frote al Laboratorio Nacional de Salud, acompañado de la ficha clínica.
- Realice una prueba de embarazo a mujeres en edad fértil, antes de iniciar el tratamiento.
- El esquema de tratamiento debe ser administrado de acuerdo con la tabla siguiente:

(Soto, C. *et al*,2018)

Ilustración 2: Esquema de dosificación de antimoniato de meglumine.

KILOGRAMOS	DOSIS DIARIA (ml)	KILOGRAMOS	DOSIS DIARIA (ml)
10	2.5	36	8.9
11	2.7	37	9.1
12	3.0	38	9.4
13	3.2	39	9.6
14	3.5	40	9.9
15	3.7	41	10.1
16	4.0	42	10.4
17	4.2	43	10.6
18	4.4	44	10.9
19	4.7	45	11.1
20	4.9	46	11.4
21	5.2	47	11.6
22	5.4	48	11.9
23	5.7	49	12.1
24	5.9	50	12.3
25	6.2	51	12.6
26	6.4	52	12.8
27	6.7	53	13.1
28	6.9	54	13.3
29	7.2	55	13.6
30	7.4	56	13.8
31	7.7	57	14.1
32	7.9	58	14.3
33	8.1	59	14.6
34	8.4	60	14.8
35	8.6	Mayor de 60	15.0

(Soto, C. *et al*,2018)

IV. MARCO TEÓRICO

A. Leishmaniasis y sus variantes

1. ¿Qué es Leishmaniasis?

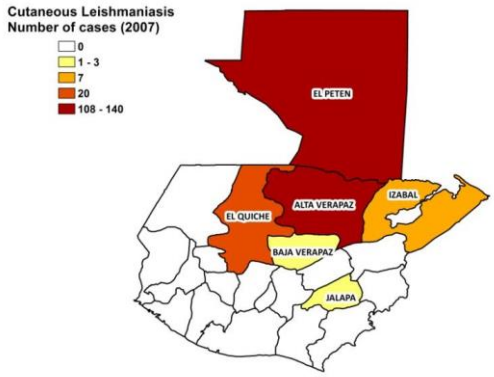
La Leishmaniasis cutánea es una enfermedad que se caracteriza por la formación de úlceras en la piel, las cuales permanecen por largo tiempo si la persona no se trata (WHO, 1990). Es la forma clínica de Leishmaniasis más frecuentemente observada en Guatemala, siendo El Petén y Alta Verapaz los departamentos que reportan el mayor número de casos anualmente. La enfermedad también se presenta en otros departamentos como Quiché, Izabal, Baja Verapaz y Huehuetenango (Copeland *et al*, 1990 y Arana *et al*, 2000). Clínicamente, la Leishmaniasis cutánea puede ir desde una úlcera pequeña y única, hasta úlceras múltiples de gran tamaño (WHO, 1990).

En el año 2008, se reportaron incidencias de 0.1 casos por cada 100,000 habitantes (Arana,1998), pero más recientemente, la incidencia ha aumentado a 6.24 casos por cada 100,000 habitantes (MSPAS, 2019). En Guatemala, la Leishmaniasis cutánea (abreviada LC para propósitos de este documento), está causada por dos especies del parásito (Herwaldt *et al.*, 1992):

- *Leishmania-L. braziliensis* (Lb)
- *Leishmania-L. mexicana* (Lm)

La enfermedad ocasionada por la especie L. braziliensis evoluciona de forma rápida. No se cura sin terapia específica y a menudo responde rápidamente a los tratamientos. La especie L. mexicana en cambio, ocasiona una enfermedad que evoluciona de forma más lenta, frecuentemente se cura sin tratamiento o terapia especializada pero no responde bien a los tratamientos (Herwaldt *et al*, 1992). El primer caso de Leishmaniasis visceral fue reportado en 1949 en un niño de dos años. La incidencia de esta enfermedad en Guatemala es baja. De 2000-2007 fueron diagnosticados 67 casos. En la siguiente ilustración se muestra un mapa de calor de Guatemala, en donde se puede observar las áreas con mayor riesgo.

Ilustración 3: Número de casos e incidencias de Leishmaniasis Cutánea en Guatemala.



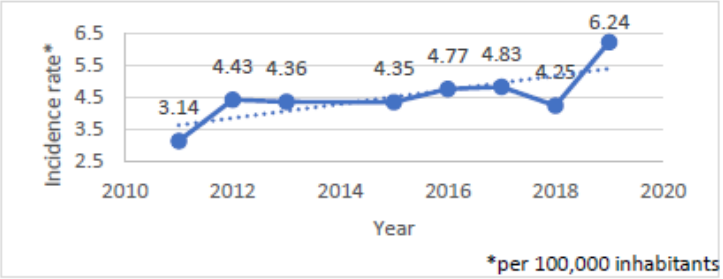
(Herwaldt *et al.*, 1992)

2. Vigilancia

La tasa de incidencia de la Leishmaniasis cutánea (CL) en Guatemala ha aumentado de 3,14 por 100.000 habitantes en 2011 a 6,24 por 100.000 habitantes en 2019, (Ilustración 2), debido a una combinación de mejor vigilancia activa, cambios climáticos y actividades ocupacionales que implican un contacto continuo con los bosques, entre otros. Desde 2003 se ha implementado un programa nacional de control, pero los esfuerzos continuos han generado resultados desiguales para las diferentes comunidades endémicas (Pérez *et al*, 2020).

Las acciones a nivel nacional para el control de la CL incluyen la vigilancia activa y pasiva, el diagnóstico y el tratamiento ofrecidos sin costo por el Ministerio de Salud Pública y Asistencia Social (MSPAS) y actividades de sensibilización. Se exploran las barreras y los facilitadores del control de LC en Guatemala según lo experimentado y percibido por los usuarios y proveedores de servicios con el fin de informar estrategias basadas en evidencia para fortalecer el control de LC en el país (MSPAS, 2018).

Ilustración 4: Incidencias de Leishmaniasis Cutánea (2011-2019).



(MSPAS, 2019)

3. Diagnóstico

La Leishmaniasis cutánea se caracteriza por la aparición de una o más lesiones en zonas descubiertas del cuerpo, los lugares más frecuentes son cara, cuello, brazos y piernas. En el lugar de inoculación, aparece una pápula que usualmente crece y se convierte en una úlcera indolora, la cual es usualmente redondeada, con una superficie de apariencia granulosa y limpia cuando no existe infección por bacterias y recubierta de costra, con bordes que característicamente cubren toda la úlcera los cuales son elevados e indurados. (MSPAS ,2018). Para diagnosticar la enfermedad se usan por lo general dos métodos:

- CL: confirmación vía microscopio
- Laboratorios.

Se realizan cuatro frotes para obtener material del borde de la lesión. Ante la sospecha clínica o epidemiológica de Leishmaniasis, es necesario visualizar el parásito para corroborar el diagnóstico por métodos de laboratorio. (MSPAS, 2018).

4. Tratamiento

- Conducta/tratamiento
- Proceda según Protocolo de Vigilancia Epidemiológica
- Realice frote directo de la manera siguiente
- Rotule dos láminas porta objetos en uno de los extremos con el número de registro del laboratorio.
- Escoja la lesión más nueva y activa, esto es la lesión con los bordes más elevados.
- La lesión debe estar libre de infección. Si estuviera infectada, debe tratarse primero la infección y después tomar la muestra.
- Retire la costra de la superficie, si estuviera presente.
- Elija el borde más activo de la lesión y realice el raspado con una hoja de bisturí. Hágalo de tal manera que no sangre, si esto sucede limpie la sangre con una gasa estéril.
- Se deben tomar dos láminas y hacer dos extendidos uno de cada uno de los bordes opuestos de la lesión.
- Deje secar las muestras a temperatura ambiente, de preferencia fijar la lámina con alcohol absoluto.
- Envíe la muestra inmediatamente al laboratorio del nivel local. Si esto no fuera posible, refiera el frote al Laboratorio Nacional de Salud, acompañado de la ficha clínica.
- Realice una prueba de embarazo a mujeres en edad fértil, antes de iniciar el tratamiento.
- El esquema de tratamiento debe ser administrado de acuerdo con la tabla siguiente:

(Soto, C. *et al*,2018)

Ilustración 5: Esquema de dosificación de antimoniato de meglumine.

KILOGRAMOS	DOSIS DIARIA (ml)	KILOGRAMOS	DOSIS DIARIA (ml)
10	2.5	36	8.9
11	2.7	37	9.1
12	3.0	38	9.4
13	3.2	39	9.6
14	3.5	40	9.9
15	3.7	41	10.1
16	4.0	42	10.4
17	4.2	43	10.6
18	4.4	44	10.9
19	4.7	45	11.1
20	4.9	46	11.4
21	5.2	47	11.6
22	5.4	48	11.9
23	5.7	49	12.1
24	5.9	50	12.3
25	6.2	51	12.6
26	6.4	52	12.8
27	6.7	53	13.1
28	6.9	54	13.3
29	7.2	55	13.6
30	7.4	56	13.8
31	7.7	57	14.1
32	7.9	58	14.3
33	8.1	59	14.6
34	8.4	60	14.8
35	8.6	Mayor de 60	15.0

(Soto, C. *et al*,2018)

Si después de dos meses de terminado el esquema de tratamiento no hay una reducción del 50% del diámetro de la lesión, se debe administrar un segundo ciclo de tratamiento. Si no hay cura al terminar este tratamiento, se debe referir a los hospitales Roosevelt o San Juan de Dios. (MSPAS ,2018).

5. Enfermedades con síntomas similares

Las infecciones tanto bacterianas como por hongos son frecuentemente diagnósticos diferenciales de la Leishmaniasis cutánea. Esto quiere decir que son enfermedades que tienden a confundirse con Leishmaniasis. Algunas de estas infecciones son las úlceras bacterianas piógenas, ectima, impétigo y otras (Ver Anexo 1 para el listado completo) (OPS, 2020).

B. Ciencia de datos e inteligencia artificial

1. ¿Qué es ciencia de datos?

La ciencia de datos es un enfoque multidisciplinario para extraer conocimientos de datos recopilados. Se puede resumir el ciclo de vida de la ciencia de datos en las siguientes etapas:

- Captura de datos: en esta etapa se recopilan los datos sin procesar de todas las fuentes relevantes a través de casi cualquier método, desde la recolección manual, hasta captura de datos de sistemas y dispositivos en tiempo real. Esta etapa parece no ser tan importante, pero hay que tomar en consideración que sin datos para analizar no importa las habilidades matemáticas o los algoritmos que disponga, no tendrá material con el que trabajar (IBM, 2020 y Muller *et al*, 2020).
- Análisis: una vez que se disponga de datos con los que trabajar y se comprenda las complejidades de esos datos, ya se puede comenzar a realizar un análisis sobre ellos. Estos análisis pueden ser estadísticos, predictivos, regresión, aprendizaje automático, algoritmos de aprendizaje profundo, entre otros. Todo esto para poder extraer información de los datos preparados (IBM, 2020 y Muller *et al*, 2020).
- Presentación: la mayoría de las personas no comprenden bien los números. No pueden ver los patrones que ve el científico de datos. Proporcionar una presentación gráfica de estos patrones es importante para ayudar a otros a visualizar lo que significan los números y cómo aplicarlos de manera significativa. Más importante aún, la presentación debe contar una historia específica para que el impacto de los datos no se pierda (Muller *et al*, 2020).

2. ¿Qué es un panel de control?

Como se explicó anteriormente, esta es una forma de presentar de manera gráfica los resultados de los datos recolectados. Si bien se puede usar de muchas formas distintas, su intención principal es proporcionar información valiosa a todo tipo de profesionales interesados en monitorear el desempeño del negocio o proyecto analizado, crear informes,

establecer estimaciones y objetivos para el trabajo futuro (Wexler *et al*, 2017). Otros beneficios de un panel de control son:

- La capacidad de identificar tendencias.
- Una forma sencilla de medir la eficiencia.
- Los medios para generar informes detallados con un solo clic.
- La capacidad de tomar decisiones en base a evidencias.
- Visibilidad total de todos los sistemas, campañas y acciones.
- Identificación rápida de valores atípicos y correlaciones de datos.

(Wexler *et al*, 2017)

3. Herramientas para realizar un panel de control

Existen diversidad de plataformas disponibles para realizar un panel de control, la mayoría de ellas de pago y aunque ofrecen un período de prueba sin ningún costo, este no pasa de treinta días. Por ello, se consideraron herramientas que dispongan de un plan gratis para que el proyecto se pueda implementar sin ningún problema.

a. Google Data Studio

Actualmente Google ofrece esta herramienta completamente gratuita, sin embargo, existen costos adicionales involucrados cuando se utilizan conectores de datos de terceros, por ejemplo, Facebook o LinkedIn (Hill, 2020).

Si hablamos de las limitantes de esta plataforma, la mayoría de ellas están relacionadas con el tamaño de los datos y no con las capacidades de visualización o transformación. Para las filas procesadas, generalmente soportan entre un millón y dos millones (Hill, 2020).

En cuanto a la cantidad de elementos que soporta, la herramienta permitirá hasta cincuenta elementos en una página. Este es un límite que un informe bien diseñado nunca debe alcanzar (Hill, 2020).

b. Microsoft Power BI

Esta plataforma actualmente se divide tres partes: Power Bi Desktop la cual es una aplicación de escritorio de Windows, Power Bi Service que es el servicio de software en la nube y las aplicaciones móviles de Power Bi (Microsoft, 2021).

Las restricciones de los distintos planes de Power Bi no están ligadas tanto a funcionalidad sino a colaboración y distribución, a continuación, se explican las tres opciones de precios junto con sus limitantes. Power BI Desktop por otro lado permite a un usuario consumir contenido en un área de trabajo asignada a una capacidad de Power Bi Premium, así como también, acceder a algunas de las características del servicio Power Bi para su propio contenido personal. Actualmente esta opción es la única gratuita (Microsoft, 2021).

4. ¿Qué es inteligencia artificial?

En las últimas décadas han surgido distintas definiciones de inteligencia artificial (IA), se presentarán las más simples y populares. John McCarthy, quien recibió el Premio Turing en 1971 por sus importantes contribuciones en este campo, ofrece la siguiente definición en (McCarthy, 2004): "Es la ciencia y la ingeniería de hacer máquinas inteligentes, especialmente programas de computadora inteligentes. Está relacionado con la tarea similar de usar computadoras para comprender la inteligencia humana, pero la IA no tiene que limitarse a métodos que son biológicamente observables".

En su forma más simple, la inteligencia artificial es un campo que combina la informática y conjuntos de datos sólidos para permitir la resolución de problemas, imitando las capacidades y toma de decisiones de la mente humana. También comprende subcampos de aprendizaje automático y aprendizaje profundo. Estas disciplinas están compuestas por algoritmos de IA que buscan crear sistemas que hacen predicciones o clasificaciones basadas en datos de entrada (IBM Cloud Education, 2020).

5. Aprendizaje automático

Es un subgrupo de la inteligencia artificial, la cual se basa en crear sistemas que puedan aprender de forma automática. Esto quiere decir que son capaces de identificar patrones complejos dentro de grandes conjuntos de datos sin la necesidad de interferencia humana. Sus algoritmos se pueden dividir en dos grupos:

a. Aprendizaje supervisado

En este tipo de aprendizaje, la máquina aprende por medio del ejemplo. Los algoritmos son entrenados por medio de preguntas, las cuales son características, como por ejemplo una fotografía y respuestas, las cuales son las etiquetas de las características. Por lo tanto, cuando el algoritmo tiene la respuesta a cierta pregunta, analiza ese conocimiento para respuestas futuras (McCarthy, 2004).

b. Aprendizaje no supervisado

A diferencia del tipo de aprendizaje anterior, en el aprendizaje no supervisado solo se le proporcionan las características, ya que el objetivo es extraer conocimientos de conjuntos de datos donde no existe una etiqueta previamente (McCarthy, 2004).

Este tipo de aprendizaje permite realizar tareas de procesamiento más complejas. Sin embargo, puede ser más impredecible puesto que infiere patrones sin referencia a resultados conocidos o etiquetados (McCarthy, 2004).

6. Redes neuronales

Inicialmente se describe una red neuronal artificial (ANN) a partir de sus similitudes con el funcionamiento del cerebro humano y posteriormente se dará una definición con perspectiva de aplicación.

Una red neuronal artificial en términos simples es un modelo computacional inspirado biológicamente, que consiste en elementos de procesamiento los cuales llamamos neuronas y conexiones entre ellos con coeficientes o pesos ligados a las conexiones. Estas conexiones constituyen la estructura neuronal y adjuntas a estas estructuras se encuentras los algoritmos de entrenamiento, optimización y recuperación (Shanmuganathan *et al.*, 2016).

Otra definición un poco más técnica dice que una red neuronal es una colección de técnicas matemáticas que se pueden utilizar para el procesamiento, la previsión y la agrupación de señales. Estas técnicas son de regresión paralelas no lineales, de múltiples capas. El modelado de redes neuronales lo podemos imaginar cómo ajustar una línea, plano o hiperplano a través de un conjunto de puntos de datos. Se puede ajustar una línea, plano o hiperplano a través de cualquier conjunto de datos para poder definir la relación que puede existir entre lo que el usuario elija datos de entrada y salida (Shanmuganathan *et al.*, 2016).

7. Redes convolucionales

Una red neuronal convolucional (CNN) es una red neuronal de aprendizaje profundo diseñada para procesar matrices estructuradas de datos como imágenes. Se utilizan ampliamente en el campo de visión por computadora.

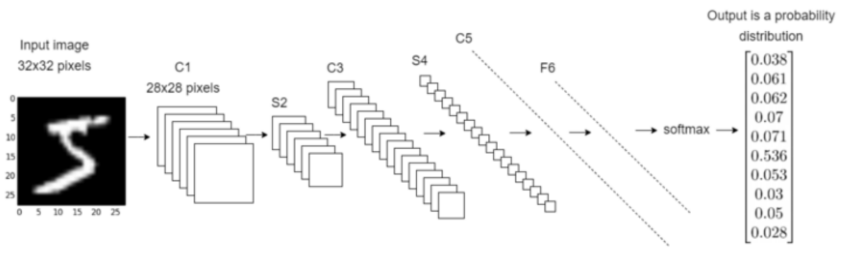
Son muy buenas para detectar patrones en la imagen de entrada, como líneas, degradados, círculos o incluso formas más complejas como caras o carros. Esta propiedad hace que las CNN sean muy poderosas para la visión por computadora. A diferencia de los algoritmos utilizados en este campo anteriormente, las CNN pueden operar directamente en una imagen sin ningún procesamiento previo.

a. Diseño de una red neuronal convolucional

Una red neuronal convolucional es creada apilando muchas capas ocultas, una encima de la otra en secuencia. Es este diseño secuencial el que le permite que las CNN aprendan características jerárquicas. Las capas ocultas son normalmente capas convolucionales, seguidas de capas de activación, algunas seguidas de capas de capas agrupadas (pooling layers) (Wood, 2020).

Se utilizará la red neuronal convolucional LeNet-5 publicada por Yann LeCun en 1998 para explicar a grandes rasgos los principios básicos de este tipo de redes.

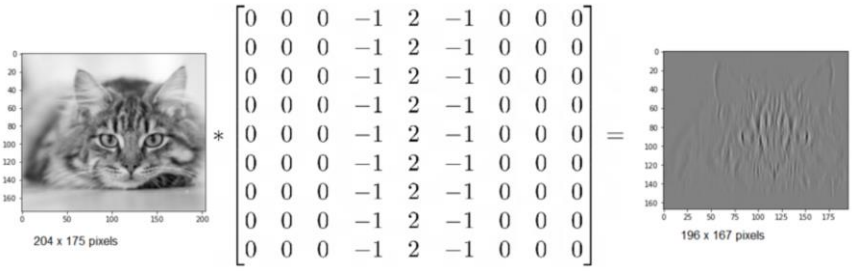
Ilustración 6: Estructura de la red neuronal convolucional LeNet-5.



LeNet toma una imagen de entrada de un dígito escrito a mano de un tamaño de 32x32 píxeles. Cada capa de la Ilustración 7 va seguida de una función de activación tanh, excepto la última ya que tiene una función de activación softmax (Wood, 2020).

La primera capa convolucional representada como C1 en la Ilustración anterior consiste en seis núcleos convolucionales que van recorriendo y filtrando la imagen de entrada. Estas generan seis imágenes de tamaño 28x28. Esta primera capa convolucional identifica características básicas como bordes rectos y esquinas (Wood, 2020).

Ilustración 7: Filtro de una convolución de ejemplo.



Para dar una idea del resultado de una convolución, en la Ilustración 8 se muestra como una fotografía de un gato es multiplicada por una matriz, la cual es una especie de filtro. Como resultado tenemos una imagen que resalta algunas líneas verticales de la imagen original. De esta forma, las capas convolucionales resaltan ciertas características de las imágenes con el fin de detectar patrones (Wood, 2020).

La segunda capa, representada como S2, es una capa de agrupación promedio (average pooling layer) lo que significa que cada cuadrado de cuatro píxeles en la salida de la capa anterior C1 se promedia a un solo píxel. Esto hace que las seis imágenes de 28x28 se reduzcan en un factor de 2, produciendo seis imágenes de salida de 14x14 (Wood, 2020).

La tercera capa, representada como C3 es una capa convolucional muy similar a C1, por lo que no se explicará de nuevo a profundidad. La diferencia es que produce 16 imágenes de tamaño 10x10 ya que es formada por 16 núcleos convolucionales. La cuarta capa, representada como S4 es la segunda capa de agrupación promedio, de igual forma con funcionamiento muy similar al explicado previamente (Wood, 2020).

La quinta capa, representada como C5 es una capa convolucional. En este caso está completamente conectada con 120 salidas. Esta capa transforma los 400 nodos (5x5x16) que provienen de S4 a 120 nodos de la siguiente capa F6. En este punto, la salida ya no es una imagen sino una matriz de una dimensión (Wood, 2020).

La última capa consiste en los resultados luego de que la función softmax transforme la salida de F6 en una distribución de probabilidad. Contiene 10 neuronas, cada una con un valor que representa la probabilidad de que la imagen ingresada corresponda a ese dígito (Wood, 2020).

8. Sesgo y varianza

Cada modelo de aprendizaje profundo viene con su sesgo y variación, a continuación, definiremos cada uno de estos términos, mencionaremos los problemas que generan, así como también posibles soluciones.

a. Sesgo

El sesgo es la diferencia entre la predicción media del modelo y el valor correcto que se está tratando de predecir. Un modelo con alto sesgo es aquel que presta poca atención a los datos de entrenamiento y simplifica demasiado el modelo. Siempre conduce a un error alto tanto en los datos de entrenamiento como prueba (Dube, 2021).

b. Varianza

La varianza es la variabilidad de la predicción del modelo para un conjunto de datos dado. Un modelo con alta varianza es aquel que presta mucha atención a los datos de entrenamiento y no generaliza sobre los datos que no ha visto antes. Como resultado, estos modelos funcionan muy bien en los datos de entrenamiento, pero en el caso de los datos de prueba, presentan altas tasas de error (Dube, 2021).

c. Interpretación de sesgo y varianza

Como regla general, podemos estimar el sesgo con el error de entrenamiento, mientras que la varianza la podemos estimar con la diferencia entre el error de entrenamiento y el error de prueba (Dube, 2021).

Al observar estas dos estimaciones, se puede caracterizar rápidamente el modelo con el que se está trabajando. Un sesgo alto y una varianza baja indica un ajuste insuficiente, y un sesgo bajo y una varianza alta indica un ajuste excesivo. El sesgo bajo y la varianza baja es un escenario ideal. Un sesgo alto y una gran variación significa serios problemas (Dube, 2021).

Tabla 1: Resumen de interpretación sesgo y varianza.

	Bajo sesgo	Alto sesgo
Baja varianza	Ideal	Desajuste
Alta varianza	Sobreajuste	Indeseable

d. Soluciones de problemas de sesgo y varianza

Un alto sesgo y una falta de ajuste implica que el modelo de aprendizaje profundo está poco calificado para resolver el problema para el cuál fue entrenado; para corregir un alto sesgo, se debe hacer el modelo más calificado, en otras palabras, aumentar su capacidad agregando más neuronas y capas. Otra posibilidad es disminuir la regularización a riesgo de aumentar la varianza. La regularización es una restricción sobre la capacidad total, por lo que disminuir la restricción aumentará su capacidad (Dube, 2021).

La alta variación y el sobreajuste significa que el modelo de aprendizaje profundo está sobrecalificado para resolver el problema para el cuál fue entrenado. Para corregirlo, se debe crear mayor desafío para el modelo. Esto se puede lograr proporcionando más datos para que el modelo tenga que trabajar más duro y utilice la capacidad sobrante no utilizada para aprender de él. Otra posible solución es incrementar la regularización para incrementar la restricción de la capacidad del modelo, sin embargo, esto puede provocar un alto sesgo. También se puede reducir el tamaño del modelo, pero es recomendable primero intentar con otras soluciones recomendadas anteriormente (Dube, 2021).

El sobreajuste también se puede reducir mediante una parada anticipada. Esto significa detener el entrenamiento tan pronto como el error en el conjunto de desarrollo comience a aumentar. Otra posible solución es usar una técnica llamada deserción, la cual no utiliza neuronas al azar durante el entrenamiento para reducir las correlaciones entre ellas y presionarlas hacia el aprendizaje de características especializadas sólidas (Dube, 2021).

e. Error de Bayes

Hay que tomar en cuenta que probablemente nunca se podrá reducir el error de prueba a cero. La mejor tasa de error posible se denomina tasa de error de Bayes. Este es un error teóricamente irreducible del que nunca podemos deshacernos, incluso construyendo el mejor modelo. Esto se debe a que siempre existe alguna fuente de incertidumbre (Dube, 2021).

Si conocemos perfectamente todos los escenarios y no hay ruido aleatorio, entonces el error de Bayes se reducirá a cero. Este error inevitable frecuentemente se puede medir estudiando a expertos humanos en el problema que estamos intentando resolver. Si los expertos no pueden lograr el error cero, un modelo de inteligencia artificial, teóricamente tampoco podrá hacerlo (Dube, 2021).

9. Transferencia de aprendizaje

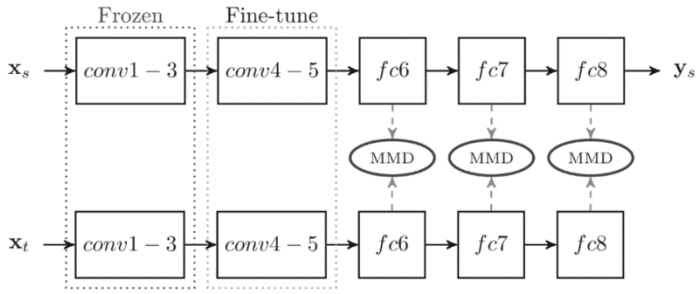
En el contexto del aprendizaje profundo, existen principalmente dos enfoques para el aprendizaje por transferencia, el aprendizaje por transferencia basado en modelos y el aprendizaje por transferencia basado en características. Generalmente se utilizan simultáneamente en un modelo de aprendizaje por transferencia profunda (Yang, *et al.*, 2020).

El aprendizaje de transferencia basado en modelos a través del uso compartido de parámetros y el ajuste fino es el método más popular. Esto se debe a que los parámetros de una red neuronal profunda son transferibles, ya que son adecuados para múltiples dominios. Esta capacidad de generalización de los parámetros se denomina transferibilidad (Yang, *et al.*, 2020).

El uso compartido de parámetros supone que los parámetros son altamente transferibles y copia directamente los parámetros de la red origen a la red destino. El método de ajuste fino asume que los parámetros en la red de origen son útiles, pero necesitan ser entrenados con los datos de destino para adaptarse mejor al dominio de destino. Los modelos de aprendizaje por transferencia basados en características aprenden

un espacio de características común que comparten los dominios de origen y destino (Yang, *et al.*, 2020).

Ilustración 8: Ejemplo de red aplicando aprendizaje por transferencia profunda.



Un ejemplo típico se muestra en la Ilustración anterior en donde las primeras tres capas se copian de una red de origen. Esto corresponde al uso compartido de parámetros en el aprendizaje por transferencia basado en modelos y también al intercambio de características en el aprendizaje por transferencia basado en características. Las siguientes dos capas se inicializan con parámetros de la red de origen y se ajustan durante el proceso de entrenamiento. Finalmente, las últimas tres capas son específicas del dominio y se aprenden en función de los datos de destino (Yang, *et al.*, 2020).

a. Aumentación de datos

La aumentación de datos genera múltiples versiones de imágenes ligeramente diferentes de cada imagen en el conjunto de entrenamiento original. El aumento de datos puede usar distintas técnicas como voltear la imagen en varias direcciones, rotar la imagen dentro de un rango de ángulos, entre muchas otras (Lee *et al.*, 2020).

Esto se puede implementar ya sea en línea o fuera de línea y una operación de aumento se puede especificar de forma aleatoria o mediante incrementos fijos. Para el aumento fuera de línea, las versiones aumentadas de las imágenes se generan previamente y se mezclan con los datos originales en un conjunto de entrenamiento grande, posteriormente, se agrupan aleatoriamente como mini lotes para el entrenamiento (Lee *et al.*, 2020).

Para el aumento en línea, las diversas técnicas de aumento generalmente se implementan como parte del proceso con probabilidad y rango seleccionado previamente por el usuario. El conjunto de entrenamiento se ingresa en mini lotes, pero cada imagen de un lote es aumentada (Lee *et al.*, 2020).

El aumento de datos introduce variaciones o fluctuaciones en los datos originales, esto reduce el riesgo de sobreajuste a un conjunto de entrenamiento pequeño y mejora la generalización. Sin embargo, es importante considerar que esta técnica no equivale a tener un conjunto de imágenes de entrenamiento independientes. Las imágenes aumentadas no aportan mucho conocimiento nuevo a la red en comparación con las nuevas imágenes independientes (Lee *et al.*, 2020).

10. Librerías para crear modelos de inteligencia artificial

a. TensorFlow

TensorFlow se desarrolló en Google Brain y luego se convirtió en un proyecto de código abierto. Agrupa una gran cantidad de modelos y algoritmos de aprendizaje automático y profundo. Utiliza Python para proporcionar una API frontal, mientras ejecuta todo en C++ con alto rendimiento (TensorFlow, 2021).

b. Scikit-Learn

Es un estándar antiguo del mundo de la ciencia de datos y puede ser bueno ejecutar bocetos rápidos de modelos de inteligencia artificial para ver si un modelo puede tener alguna interpretación. Esta librería tiene excelentes herramientas de preprocesamiento de datos, por lo que normalmente se usa para preprocesar datos o para hacer bocetos y posteriormente se traslada las ideas a otras librerías como TensorFlow o Pytorch (scikit-learn, 2021).

c. Pytorch

Fue desarrollado por FAIR, Facebook AI Research. Luego a principios de 2018, el equipo de FAIR fusionó Caffe2 (otra librería de inteligencia artificial) con PyTorch. Es el principal competidor de TensorFlow. Usualmente cuando los ingenieros deciden usar una plataforma de inteligencia artificial, la elección se reduce a "¿Usamos TensorFlow o PyTorch?" (PyTorch, 2021).

11. Parámetros e hiperparámetros

Estos son dos conceptos que a simple vista parecen similares, pero en realidad son completamente diferentes. La única similitud es que ambos son conjuntos de valores que se tienen que afinar al momento de crear un modelo (Nicosia *et al*, 2019).

- Parámetros: son las variables que se estiman durante el proceso de entrenamiento con los conjuntos de datos. Estos valores no se ingresan manualmente, sino que son obtenidos (Nicosia *et al*, 2019).
- Hiperparámetros: son los valores de las configuraciones utilizadas durante el proceso de entrenamiento. Estos no se obtienen de los datos de entrenamiento, por lo que el científico de datos los suele ingresar o utilizar un algoritmo que de alguna forma encuentre la mejor combinación (Nicosia *et al*, 2019). Algunos ejemplos de hiperparámetros son:
    - Velocidad de aprendizaje: determina el tamaño de los saltos en cada iteración mientras se desplaza hacia un mínimo de una función de error (Nicosia *et al*, 2019).
    - Droput Rate: es una forma simple de prevenir el sobre ajuste en redes neuronales. Esta técnica consiste en seleccionar neuronas de forma aleatoria e ignorarlas durante el entrenamiento. (Nicosia *et al*, 2019).

12. Optimización de hiperparámetros

El aprendizaje automático ha logrado muchos éxitos en una amplia gama de áreas, la mayoría de las veces, estos resultados se basan en gran medida en la elección de los valores correctos para muchos hiperparámetros. A continuación, se mencionan algunas técnicas para poder determinar los hiperparámetros de una red:

a. Búsqueda aleatoria

Es un procedimiento en el que se determina la mejor solución para el modelo construido mediante combinaciones aleatorias de hiperparámetros. Al elegir el hiperparámetro al azar, la búsqueda aleatoria elimina la evaluación exhaustiva de todas las combinaciones de diferentes hiperparámetros (Prakash *et al*, 2021).

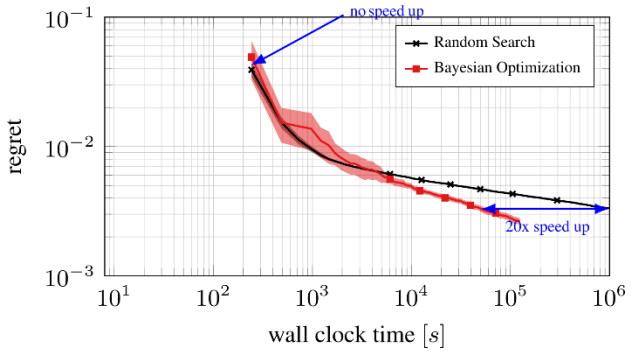
b. Optimización Bayesiana

Utiliza un modelo probabilístico para modelar la función objetivo en función del conjunto de puntos de datos ya observados. Utiliza una función de adquisición basada en el modelo actual para identificar nuevas configuraciones prometedoras. El proceso estándar es el siguiente:

- Seleccionar el punto que maximice la función adquisición.
- Evaluar la función objetivo en este punto.
- Agregar la nueva observación a los datos.
- Reajustar el modelo.

La optimización Bayesiana necesita tiempo para construir un buen modelo, por lo que se comporta de manera similar a la búsqueda aleatoria al principio como se puede observar en la siguiente Ilustración. Sin embargo, con el aumento de las iteraciones, el modelo recopila cada vez más información sobre el espacio de búsqueda, lo que genera grandes ventajas sobre la búsqueda aleatoria (Biedenkapp *et al*, 2018).

Ilustración 9: Comparación de optimización bayesiana y búsqueda aleatoria.



(Biedenkapp *et al*, 2018)

c. Hiperbanda

Es una estrategia que hace uso de aproximaciones económicas de evaluar la función objetivo en presupuestos más pequeños. Llama repetidamente a una función para identificar la mejor de n configuraciones seleccionadas aleatoriamente. Es decir, evalúa estas n

configuraciones con un presupuesto pequeño, mantiene la mejor mitad y duplica su presupuesto. En la siguiente Ilustración podemos observar el comportamiento de la selección y exploración del espacio (Biedenkapp *et al*, 2018).

d. BOHB

Este algoritmo combina la optimización Bayesiana e hiperbanda para combinar ambas ventajas en una. BOHB se basa en el algoritmo de hiperbanda para determinar cuántas configuraciones evaluar con qué presupuesto, pero reemplaza la selección aleatoria de configuraciones al comienzo de cada iteración por una búsqueda basada en modelos utilizando el algoritmo de optimización Bayesiana (Biedenkapp *et al*, 2018).

C. Aplicaciones móviles

1. Aplicaciones nativas

Las aplicaciones móviles o nativas son desarrolladas específicamente para cada sistema operativo, es decir, es específico para cada plataforma. Una de las ventajas de estas aplicaciones es que aprovechan las funcionalidades del dispositivo, como la identificación de uso de del Wifi, ubicación, cámara, entre otros (SOLBYTE, 2020).

2. Aplicación web

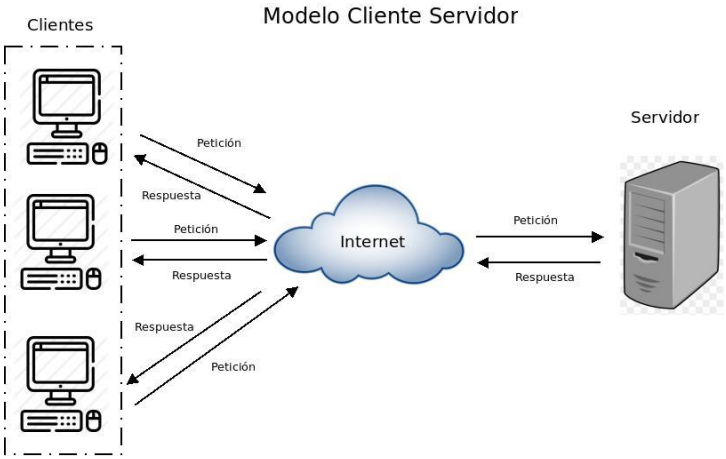
A pesar de ser parecidas a las páginas web, estas son compatibles con cualquier sistema operativo, se adapta y no tendrá que ser desarrollada la misma aplicación para plataformas distintas. Utilizan el navegador disponible en el dispositivo, sin embargo, muchas no funcionan sin una conexión a internet (SOLBYTE, 2020).

D. Modelo cliente-servidor

Se le llama una arquitectura cliente servidor, ya que este consiste en dos partes: el cliente, en dónde se encuentra la interfaz de la aplicación; y el servidor: encargado de controlar el acceso a los datos. En este sistema, la aplicación del cliente utilizará procedimientos específicos para acceder o interactuar con los datos, los cuales están almacenados en el servidor de base de datos (Vega, 2019).

Este modelo se utiliza en varios servicios y protocolos de internet, por lo que lo adecuado es que el servidor sea una máquina con mucha capacidad, hardware potente y software específico que actúa de depósito de datos y funcione como un sistema gestor de base de datos o aplicaciones. El cliente por otra parte suele consistir en estaciones de trabajo que solicitan varios servicios al servidor, el cual tiene el rol de dar la respuesta demandada el cliente (Schiaffarino, 2019).

Ilustración 10: Esquema del modelo cliente servidor.



(Schiaffarino, 2019)

1. Protocolo HTTP

Proveniente de las siglas en inglés: Hypertext Transfer Protocol, es un protocolo en la capa de aplicación para la transmisión de documentos hipermedia. Fue diseñado principalmente para la comunicación entre los navegadores y los servidores web, pero puede ser utilizado para otros propósitos. Sigue el modelo cliente-servidor, en donde se establece una conexión haciendo una petición a un servidor y se espera una respuesta de este. Este protocolo no maneja estados, lo que significa que no guarda ningún dato entre consultas (Mozilla, 2021).

a. Códigos de estado de respuesta de HTTP

Los códigos de estado de respuesta HTTP dan a entender si una solicitud fue completada con éxito. Las respuestas se agrupan en cinco clases diferentes.:

- Respuestas informativas, con código entre 100 – 199
- Respuestas satisfactorias, código entre 200 – 299
- Redirecciones, código entre 300 – 399
- Errores de parte del cliente, código entre 400 – 499
- Errores de los servidores, código entre 500 – 599

b. Control de acceso HTTPP (CORS)

El intercambio de Recursos de Origen Cruzado (CORS) es un mecanismo que utiliza la cabecera de HTTP de forma adicional para permitir que un usuario obtenga permiso para acceder a recursos seleccionados desde un servidor. El usuario creó la solicitud HTTP de origen cruzado cuando solicita un recurso desde un dominio distinto, un protocolo o un puerto diferente al del documento que lo generó. Esta práctica se emplea por razones de seguridad, dado que los exploradores modernos manejan los componentes sobre el intercambio de datos del lado del cliente.

c. JSON WEB TOKEN

El intercambio de Recursos de Origen Cruzado (CORS) es un mecanismo que utiliza la cabecera de HTTP de forma adicional para permitir que un usuario obtenga permiso para acceder a recursos seleccionados desde un servidor. El usuario creó la solicitud HTTP de origen cruzado cuando solicita un recurso desde un dominio distinto, un protocolo o un puerto diferente al del documento que lo generó. Esta práctica se emplea por razones de seguridad, dado que los exploradores modernos manejan los componentes sobre el intercambio de datos del lado del cliente.

Los JWTs pueden separarse en tres partes: encabezado, carga útil y firma de la siguiente manera:

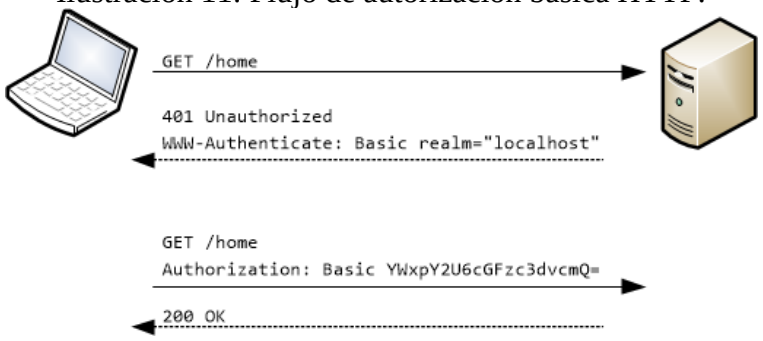
- Encabezado: la información en el encabezado describe el algoritmo utilizado para generar la firma
- Carga útil: información de utilidad dentro de la autenticación JWT es almacenada en esta parte. Las consultas normalmente llevan consigo métodos de autenticación. Por ejemplo, los permisos de usuario admin, moderator y user. Si el campo admin es "true" entonces es posible realizar acciones en el sistema de acuerdo con los permisos de ese usuario específico.
- Firma: la firma se deriva del encabezado y la carga útil. Los pasos para la creación de la firma son los siguientes:
    - base64UrlEncode(header) + "." + base64UrlEncode(payload)
    - hash_value = hash([base64UrlEncode(header) + "." + base64UrlEncode(payload)], secret-key)
    - Signature = base64UrlEncode(hash_value)

Debido a que la *llave-secreta* (secret-key) es conocida únicamente por el servidor, solo el servidor puede emitir nuevos tokens con una firma válida. Los usuarios no pueden falsificar tokens, ya que producir una firma válida para un token requiere del conocimiento de la *llave-secreta*.

JWT tiene aplicaciones en varios mecanismos de autenticación. Por lo general, son enviados en el encabezado de **Authorization** cuando un usuario envía una solicitud al cliente.

HTTP brinda una forma para controlar el acceso y la autenticación de un usuario, la más común forma de autenticación es la denominada "Basic". Este marco puede ser usado por un servidor, para revisar la solicitud de un cliente y así brindar información de autenticación. El flujo es el siguiente: El cliente hace una petición al servidor, el servidor responde con un status 401 (sin autorización) y responde con instrucciones para autorizarse con un encabezado de respuesta que contiene al menos una revisión. Una vez el cliente incluya en el encabezado la solicitud *Authorization* con sus credenciales, el servidor verificará que éstas sean correctas y así enviar la información solicitada. En la Ilustración a continuación se muestra el flujo visual para este marco de autenticación (Mozilla, 2021).

Ilustración 11: Flujo de autorización básica HTTP.



(Microsoft, 2021)

En la Ilustración 15 puede observarse el siguiente flujo:

- El servidor responde con código 401 (Unauthorized) y provee información sobre cómo realizar la autorización con el encabezado WWW-Authenticate.
- Con esta respuesta, el cliente entonces puede adjuntar el encabezado *Authorization* a su consulta para proveer las credenciales.
- De esta forma las consultas se realizarán siempre enviando las credenciales para poder retribuir la información deseada.

E. Sistemas y tecnologías de desarrollo para backend

1. Bases de datos

a. Ventajas de las bases de datos

- Control sobre redundancia de datos: en estos sistemas, todos los ficheros están integrados, lo que garantiza que no se almacenarán varias copias de los mismos datos. La redundancia no se podrá eliminar redundancia completamente, dado que en ocasiones es necesaria para modelar las relaciones entre los datos (Vega, 2019).
- Consistencia de datos: si se controla adecuadamente la redundancia de datos, se reduce el riesgo que existan inconsistencias. Si un dato está almacenado una vez, cualquier actualización se realizará solo una vez, y estará disponible para cualquier usuario que quiera acceder a él inmediatamente (Vega, 2019).
- Compartición de datos: en el caso de archivos o ficheros, estos pertenecen a las personas o departamentos que los utilizan. En una base de datos, los datos pertenecen a una sola empresa o negocio, pero pueden ser compartidos por otros usuarios autorizados (Vega, 2019).
- Integridad y seguridad en los datos: la integridad de los datos hace referencia a la validez y consistencia del almacenamiento de estos. Es posible inferir que, al manejar datos, se deben aplicar series de instrucciones o reglas que se deben seguir. En cuanto a la seguridad de los datos, es la protección de la base de datos frente a usuarios no autorizados. Se deben tomar medidas de seguridad adecuadas para protegerlos y así evitar vulnerabilidades sobre los datos (Vega, 2019).

b. Bases de datos relacional

Una base de datos relacional quiere que decir que los datos almacenados cumplen con el modelo relacional y es el más utilizado en la actualidad cuando se trata de implementación de bases de datos. El diferenciador es que permites establecer interconexiones (relacionales) entre los datos y, con el uso de estas conexiones, poder relacionar los datos de las tablas (AWS, 2021).

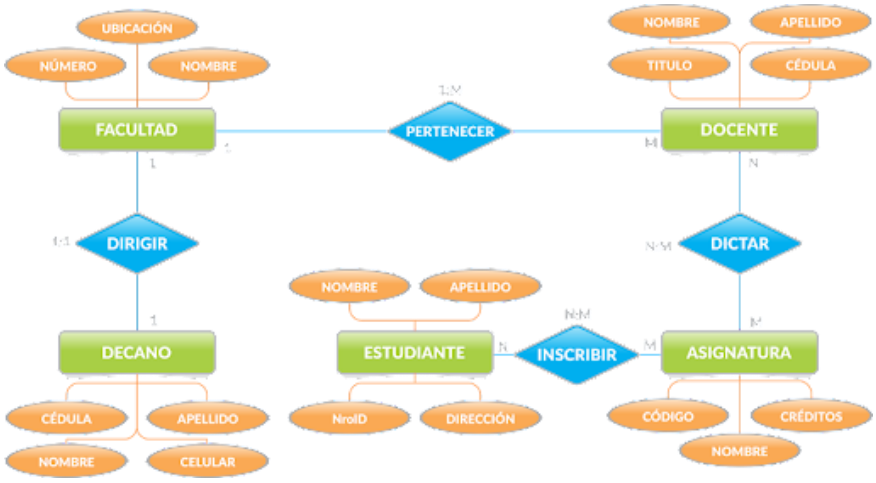
c. Base de datos no relacional

Una base de datos no relacional está diseñada para modelos de datos específicos, con esquemas flexibles. Son reconocidas por su facilidad de desarrollo, funcionalidad y rendimiento a escala. Estas bases de datos están diseñadas para múltiples patrones de acceso a datos que incluyen aplicaciones de baja latencia, con el fin de hacer análisis sobre datos semiestructurados (AWS, 2021).

d. Modelo entidad relación

Un modelo entidad relación es un tipo de diagrama o flujo que permite ver cómo las "entidades", es decir, datos relacionados a personas, objetos o conceptos, se relacionan entre sí dentro de un sistema. Se usan comúnmente para diseñar datos relacionales en los campos de ingeniería de software, educación, investigación e incluso en sistemas de información empresarial. Estos modelos también poseen su simbología, entre ellos rectángulos, óvalos, líneas de conexión, y otros elementos para representar la interconexión de las entidades, sus relaciones y atributos (ESIC BUSINESS & MARKETING SCHOOL, 2018).

Ilustración 12: Ejemplo de modelo entidad relación.



(ESIC BUSINESS & MARKETING SCHOOL, 2018)

2. Interfaz de programación de aplicaciones

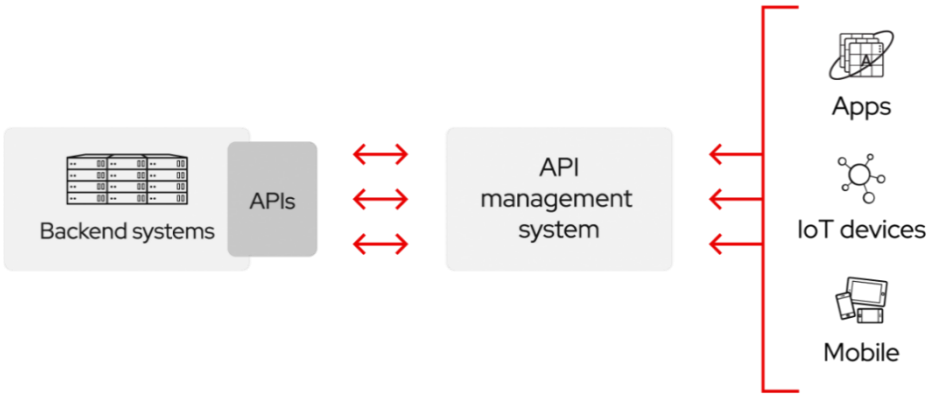
El término API es una abreviatura del inglés Application Programming Interface. Un API es un conjunto de comandos y funciones informáticos que permite a los desarrolladores crear uno o varios programas para sistemas operativos u otras aplicaciones con un objetivo funcional. Uno de los motivos por las cuales existen, es que permiten la simplificación del trabajo a un creador de programas, pues no tendrá que escribir código desde un inicio. El informático usará funciones predefinidas para realizar acciones sobre una tecnología, plataforma, sistema operativo o incluso con otro programa (ABC, 2015).

Existen múltiples diseños de arquitectura de software al desarrollar un API, cada uno con sus ventajas y desventajas. Todos los acercamientos son soluciones viables, que han funcionado a través de los años y tienen mucho soporte en un sinfín de aplicaciones. Entre todos comparten el hecho que permiten el intercambio de datos, mientras mantienen la seguridad en cada consulta y el control del usuario (ABC, 2015).

Las API permiten que los productos y/o servicios se comuniquen con otros, sin la necesidad de entender cómo estos otros servicios están implementados. Se puede interpretar un API como un contrato, en el cual cada parte envía una solicitud remota con una estructura, y esta estructura será determinará cómo responderá el software de otra parte. Las API son un mediador para conectar una infraestructura a través del desarrollo de aplicaciones nativas de la nube, aunque también permiten compartir datos con otros usuarios y clientes (Red Hat Inc, 2021).

Cabe mencionar que las API deben ser diseñadas de acuerdo con los estándares web. Las API web, por lo general, usan HTTP para solicitar mensajes y así dar una definición de la estructura de los mensajes de respuesta. Es común que estas respuestas tomen la forma de un archivo XML o JSON, formatos que presentan una forma de manejar datos en aplicaciones (Red Hat Inc, 2021).

Ilustración 13: Estructura de una API remota.



(Red Hat Inc, 2021)

En la Ilustración 17 se puede observarse una implementación general de un API en donde existen distintos dispositivos que envían información hacia el Backend del sistema en cuestión. Las APIs están diseñadas para interactuar a través de una red de

comunicaciones. Al ser remotas quiere decir que la información requerida es externa al sistema que hará la consulta. La mayoría de las implementaciones están basadas en tecnologías y estándares web y se da por sentado que un API web es remota.

Las APIs web utilizan HTTP para las consultas y proveen una definición de la estructura de respuesta a dichas consultas. Los mensajes de respuesta normalmente están en formato XML o JSON, ya que dichos formatos son de fácil manipulación para las aplicaciones.

3. Servicios web

Según IBM, los servicios web son aplicaciones modulares autocontenidas que puede describir, publicar, localizar e invocar a través de una red. Son también aplicaciones que ayudan a mejorar la flexibilidad de procesos u acciones mediante la integración de aplicaciones que, de no ser por ellos, no se logra una comunicación. Estas implementan una arquitectura orientada a servicio, que por sus siglas en inglés Service Oriented Arquitecture, son también conocidas como SOA. Dan soporte al compartir recursos y datos de manera flexible y con un estándar preestablecido; es importante organizar los servicios para así apoyar su reutilización dinámica y automatización de procesos (IBM, 2021).

a. SOAP

Los servicios SOAP, por sus siglas en inglés Simple Object Access Protocol, son servicios web que se caracterizan por el intercambio de datos XML (Extensible Markup Language). Funciona por varios medios de transferencia de datos u archivos mediante un tipado fuerte y un formato pesado tanto en tamaño como en procesamiento (Bora & Bezboruah, 2015).

b. AJAX

Significa JavaScript asíncrono y XML, por sus siglas en inglés, *Asynchronous JavaScript and XML*. Es un conjunto de técnicas de desarrollo web que permiten que las aplicaciones web funcionen de forma asíncrona, procesando cualquier solicitud al servidor en segundo plano. Tiene la capacidad de enviar y obtener datos del servidor sin necesidad de volver a cargar toda la página web. Gestiona el contenido dinámico y permite la interacción dinámica del usuario (Beneventi, 2021).

c. REST API

Un REST API, por sus siglas en inglés, Representational State Transfer, traducido al español: Transferencia de Estados Representacionales. Esta es una forma simple y estandarizada de arquitectura de software que la industria hoy en día utiliza y conoce con gran confianza. La comunicación se da entre el cliente y el servidor mediante un servicio web RESTful. Esta arquitectura es escalable y sin estados, lo que le permite una adaptación muy fácil al crecimiento de complejidad del servicio. La creación de una tecnología como esta permite un rendimiento muy alto en las aplicaciones gracias al uso de alojamiento en memoria por parte del dispositivo. Esto último es vital para el proyecto dado que el usuario final no siempre tendrá una conexión activa y estable al internet (IBM Cloud Education, 2021).

d. RESTfull

Para que una API se considere RESTful, se deben cumplir los siguientes criterios:

- Arquitectura cliente-servidor, la cual está compuesta por uno o más usuarios, servidos y recursos cuyo medio de transporte se realiza a través de servicios HTTP.
- Comunicación entre el cliente y el servidor sin estado (stateless), esto indica que no se almacena información de los clientes en las consultas al servidor, y que cada consulta es independiente sin relación al resto de consultas.
- Optimización de interacciones entre el cliente y el servidor almacenando datos en caché.
- Uso de un sistema de capas entre las interacciones cliente-servidor, participando en la recuperación de información solicitada.
- Código según se solicite, lo que indica que se puede enviar código ejecutable del servidor al cliente lo cual amplía las acciones que el cliente puede ejecutar. Este criterio es opcional.
- Interfaz uniforme entre elementos, de forma que la información se presente de forma estandarizada.
- Recursos solicitados deben ser identificables de la representación que se envían al cliente.
- El cliente podrá utilizar los recursos a través de la representación recibida.
- Los mensajes que se envíen al cliente deben ser autodescriptivos, con el fin de poder interpretar qué hacer con la información.
- Debe contener hipertexto o hipermedios, es decir, se puede utilizar hipervínculos para acceder a recursos.

(Red Hat Inc, 2020)

e. Caché

Recurso que cuenta la unidad central de procesamiento, con la capacidad de almacenar datos temporales y recientes en una especie de búfer o memoria auxiliar. Esta puede darse tanto en hardware como software, y a pesar de ser limitada de tamaño, permite que los datos almacenados en esta memoria sean accedidos de forma rápida. Tanto los servidores como los navegadores pueden contar con una, permiten optimización de consultas y datos en ambientes de producción (Editorial Etecé, 2021).

f. GraphQL

Es un lenguaje a base de querys para APIs. Esta arquitectura fue diseñada para proveer una gran flexibilidad y eficiencia a la hora de resolver problemas pequeños al intentar hacer consultas desde una aplicación hacia un servidor. Es, hasta la fecha de creado este documento, el diseño más nuevo de comunicación cliente-servidor y resuelve muchos de los problemas que REST introduce, como las sobre consultas y los beneficios de utilizar un esquema (GraphQL, 2021).

4. Backend

En palabras informales, backend (también conocido como Back-End) es la parte del desarrollo que se encarga de la lógica en la que una aplicación o página web funciona. Consiste principalmente es un conjunto de acciones que pasan dentro de una aplicación, las cuales el usuario final no podrá ver (de allí la palabra "detrás de"). Por ejemplo, la publicación de datos a un servidor por medio de transferencia de datos es una acción que el usuario ejecuta, más no un proceso que este mismo usuario pueda ver cómo se realiza (Parashuraman, 2021).

Un desarrollador de backend está encargado de: Implementar una arquitectura que compromete servidores, bases de datos y sistemas operativos; Manejar consultas de clientes con APIs; manejo de base de datos; crear lógicas de negocio que corran la aplicación; solución de problemas de rendimiento o cuello de botella; crear código reutilizable y escalable para la aplicación (Parashuraman, 2021).

a. JavasScript

JavaScript es un lenguaje de programación o de secuencias de comandos que permite implementar funciones complejas en servicios web. Es utilizado principalmente para el desarrollo de sitios web, no requiere de compilación y los navegadores lo usan para interpretar el código. Puede ser ejecutado sin la necesidad de instalar otro programa, es un lenguaje basado en acciones (Mozilla, 2021).

b. TypeScript

TypeScript es un superset de JavaScript, es decir, agrega sintaxis adicional a JavaScript para un desarrollo de aplicaciones más estandarizado. Es una solución creada por Microsoft para aplicaciones de gan escala. Este superset es la solución a muchos problemas de JavaScript, puesto que está pensado para el desarrollo de aplicaciones robustas, implementando características que permiten desarrollar herramientas más avanzadas en el desarrollo de aplicaciones (Hernández, 2018).

Una de las características principales es su tipado estático, es decir, las variables tendrán un tipo de dato y los valores solo se podrán asignar a variables del tipo de dato correspondiente. También posee soporte para interfaces, conversión de tipos, argumentos con tipo de dato, tipo de retorno para funciones y entre otras características que agregan valor al desarrollar. Entre ese valor, está el autocompletado de código y la detección temprana de posibles errores al momento de compilar la aplicación (Hernández, 2018).

c. JSON

Su nombre corresponde a las sigas JavaScript Object Notation y su traducción al español, Notación de Objetos de JavaScript. Este es un formato de intercambio de datos ligero, que es fácil de leer y escribir para los programadores y simple de interpretar y generar para las máquinas. Este formato de texto es independiente de lenguaje lo que hace este formato de intercambio de datos ideal para usar con API REST o AJAX (Barrera, 2021).

d. Middleware

El término middleware se refiere a un sistema de software que ofrece servicios y funciones comunes para las aplicaciones. Se encarga de tareas de gestión de datos, servicios de aplicaciones, mensajería, autenticación y gestión de API. Además, actúa como conductor entre las aplicaciones, los datos y los usuarios. Abarca desde servidores web, hasta sistemas de autenticación y herramientas de mensajería. En el ámbito del desarrollo de aplicaciones, es útil para ejecutar aplicaciones a escala y en la nube (Red Hat Inc, 2021).

e. Framework

Junto con el lenguaje de programación seleccionado, frameworks son una parte integral del desarrollo web. Permiten organizar el código de una forma estructurada y mejorar la eficiencia reduciendo código repetitivo. Contiene un conjunto de herramientas que garantiza una reducción de errores y aumento de productividad. Dado que un framework es probado, construido y optimizado por múltiples ingenieros de software con experiencia, son versátiles, robustos y eficientes. Usar un framework para desarrollar aplicaciones permite el enfoque en la funcionalidad a alto nivel de la aplicación; esto es puesto que cualquier funcionalidad a bajo nivel es manejada por el framework como tal (Vijay, 2021).

f. Node JS

Node.js es un ambiente en tiempo de ejecución que permite a los desarrolladores de software ejecutar aplicaciones tanto en Frontend como en backend de aplicaciones web usando JavaScript. Node.js tiene APIs que soportan peticiones HTTP, sistema de archivos y otras características del lado del servidor que APIs de Frontend proveen un soporte limitado. La habilidad de Node.js de manejar múltiples peticiones de forma simultánea y proveer respuestas hace una solución ideal para aplicaciones web enfocadas en el cliente (Clever Solution, 2020).

g. Express

Express es un framework que provee una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Pose muchos métodos de programa de utilidad HTTP y middleware, por lo que la creación de una API sólida es rápida y sencilla. En cuanto a rendimiento, proporciona una capa de características de aplicación web básicas, que no ocultan las características de Node.js (OpenJS Foundation, 2021). Algunas de las funcionalidades principales de Express son:

- Implementación de funciones de respuesta a consultas HTTP
- Definición de tablas de enrutamiento para acciones dependiendo de los métodos HTTP utilizados y la URL en cuestión
- Renderización dinámica basada en argumentos

h. Object Relational Mapping

Traducido al español, mapeo objeto-relacional es un modelo de programación que permite mapear estructuras de una base de datos relacional sobre una estructura lógica de

entidades con el objetivo de hacer más simple y acelerar el desarrollo de aplicaciones. Las estructuras de base de datos quedan vinculadas de forma que las acciones de crear, leer, editar o eliminar sobre la base de datos real se hacen de forma indirecta. Una de las ventajas es que el desarrollador no tendrá que generar código manual de SQL para hacer consultar y gestionar la base de datos (Deloitte, 2021).

Sequelize

Sequelize es un ORM (Object Relational Mapping) para Node.js que permite manipular bases de datos desde el middleware o cualquier otro lenguaje de desarrollo. Permite convertir tables de una base de datos en entidades desde un lenguaje de programación orientado a objetos, lo cual agiliza el acceso a los datos (Garcia, 2018).

5. Herramientas útiles para servicios Backend

a. PostMan

Postman es una plataforma API dirigida a desarrolladores web para crear y usar APIs. Permite realizar peticiones HTTP a cualquier API y provee múltiples herramientas para hacer pruebas sobre una API, como lo es el diseño, realización de pruebas, documentación de endpoints, simulación, etc. Es útil también para hacer pruebas, dado que permite comprobar el correcto funcionamiento del sistema luego de haberlo programado (Postman, Inc, 2021).

b. SWAGGER

Swagger es una herramienta útil para describir, producir, consumir y visualizar APIs RESTful. Engancha en sistema de documentación con el código descrito en el servidor y así sincronizarlos con cualquier cambio. De esta forma se documentan los servicios al mismo tiempo que se crea la implementación de cada uno. Proporciona una interfaz visual donde se pueden probar las consultas al API programadas, además de ver la documentación con métodos, parámetros y modelos JSON (Rodríguez, 2013).

c. Logging, debugging y pruebas

- Logging: es una referencia al registro de información útil y errores que permiten identificar y dar seguimiento a problemas en un sistema, de la forma más eficiente posible.
- Pruebas: es una parte crucial en el proceso de desarrollo de software. En variadas ocasiones las empresas de software cuentan con departamentos completos dedicados a las pruebas. De forma general, están relacionadas al buen funcionamiento del sistema.
- Debugging: es el proceso de localización y reparo de fallas de software. Un debugger es una herramienta que permite al desarrollador encontrar posibles errores en el código fuente de una aplicación en cuestión. (Colin J., 2014)

d. WINSTON

Winston es una librería simple y universal con soporte para múltiples transportes, es decir, un lugar de almacenamiento para los registros. La herramienta puede ser

configurada para crear registros de diferentes tipos, desde depuración, interacciones con bases de datos, y otros registros útiles para monitorear la salud de cualquier sistema, además de poder validar qué fue lo que pasó justo antes que un sistema fallara. Entre los distintos niveles para creación de registros se encuentran:

- Error: errores de un sistema considerados críticos
- Info: información del sistema
- Warn: advertencia sobre posibles errores
- Debug: información útil para depurar el sistema. (Vicente, 2017)

Cada uno hace referencia a conceptos clave para el mantenimiento de un sistema y facilita la clasificación de registros de información dependiendo del contexto.

6. Garantizar calidad en el sistema

Uno de los aspectos más importantes del desarrollo de software, es garantizar que cualquier porción de código nuevo integrado al sistema, realmente aporte funcionalidad y garantice que no quiebre cualquier otro aspecto ya programado. Por ello, es importante realizar una serie de pruebas y mantener conceptos cercanos al desarrollo.

a. Pruebas unitarias

Las pruebas unitarias es un tipo de pruebas de software, en donde unidades individuales o componentes de software son probadas. El propósito es validar que cada unidad de código de software se ejecute como esperado. Las pruebas unitarias se realizan durante el desarrollo (la fase de codificación) de una aplicación por los desarrolladores. Se aísla la sección de código a desarrollo y verifica su exactitud (Hamilton, 2021).

- Ventajas
    - Permite a los desarrolladores refactorizar el código en una fecha más tardía, y asegurar que el módulo aún funciona correctamente. El procedimiento es escribir casos para todas las funciones y métodos de forma que cuando algún cambio introduzca una falla, este podrá ser identificado y corregido rápidamente.
    - Dado que es un desarrollo modular, se pueden probar ciertas partes del proyecto sin necesidad de esperar a que otras partes sean terminadas.

      (Hamilton, 2021)
- Desventajas
    - No se puede esperar a que logre identificar cada error en el programa. No es posible evaluar todos los caminos de ejecución, aún en los programas más triviales.
    - Las pruebas unitarias, por su propia naturaleza, se centran en una unidad de código. Por lo tanto, no puede detectar errores de integración o errores generales a nivel de sistema.
- Jest
    - Jest es una librería útil para realizar pruebas unitarias en lenguajes JavaScript. Fue desarrollado por Facebook, es fácil de instalar y configurar, además esta optimizado para actuar rápidamente cuando se realiza algún

cambio sobre los archivos base. Corre las pruebas en paralelo de forma distribuido con el fin de maximizar el rendimiento de la aplicación (Facebook Inc, 2017).

b. Pruebas de punto a punto

Las pruebas de punto a punto, también conocidas en inglés como end-to-end testing es un tipo de prueba en donde se simula el flujo de la aplicación de inicio hasta final, esperando que se comporte de una forma esperado. También es conocido como test funcional. Un ejemplo de este tipo de prueba se da cuando se prueba un endpoint o una ruta que involucra todo lo que necesita el controlador para funcionar, tal como conexión a base de datos, dependencias, entre otros (Orie, 2019).

c. Pruebas de carga

Las pruebas de carga analizan el comportamiento del sistema bajo condiciones de trabajo regulares, y consiste en solamente probar o simular la carga de trabajo. Estas pruebas no tienen como objetivo romper el sistema (Hamilton, 2021).

d. Pruebas de estrés

Las pruebas de estrés son un tipo de prueba de software que verifica la estabilidad y confiabilidad de una aplicación o sistema. El objetivo de las pruebas es medir el software en cuanto a robustez y capacidades de manejo de errores en condiciones de carga extremadamente pesadas; garantizando así que el software no se bloquee en situaciones de crisis. Incluso prueba más allá de los puntos operativos normales y evalúa cómo funciona el software en condiciones extremas (Hamilton, 2021).

F. Infraestructura y tecnologías de la nube

Infraestructura es el nombre con el que puede llamarse al conjunto de dispositivos y/o aplicaciones necesarias para el funcionamiento de un sistema, en este caso englobando el campo de las tecnologías de la información.

Puede componerse de varios elementos como:

- Servidores: los hay de distintos tipos dependiendo de las necesidades de la o las empresas gestionando la construcción del sistema.
- Almacenamiento: existen diferentes soluciones de almacenamiento que pueden aplicarse al sistema, de nuevo, dependiendo mucho de los requerimientos e implicaciones.
- Redes: permite la comunicación interna de los diferentes módulos y componentes que conforman al sistema, así como también la comunicación con agentes externos al ambiente de desarrollo.

1. ¿Qué es computación en la nube?

Computación en la nube es un modelo para habilitar la disponibilidad, conveniencia y demanda de acceso a un conjunto de servicios tecnológicos proporcionados. Entre ellos es posible mencionar: servidores, almacenamiento, aplicaciones y otros. Estos servicios

pueden ser provisionados de manera rápida y conveniente, tomando en cuenta una fácil administración y una interacción minimalista con el sistema. (Golden, 2015) Este modelo se compone de cinco características esenciales:

- Servicios por demanda: un consumidor puede proveer recursos computacionales como tiempo de servidores o almacenamiento de forma automática cuando sea necesario a sus aplicaciones.
- Amplio acceso de red: los recursos están disponibles en la red, promoviendo el fácil acceso a dispositivos móviles, laptops, estaciones de trabajo y más.
- Conjunto de recursos: los recursos están distribuidos en distintas locaciones que el cliente no conoce dependiendo del centro de datos, región, entre otros. Sin embargo, el cliente puede elegir dichos recursos y utilizarlos para su conveniencia en un mismo lugar.
- Elasticidad rápida: capacidad de escalabilidad en ocasiones automáticamente y medida por demanda.
- Medición de servicios: los sistemas de la nube controlan y optimizan de forma automática la administración de recursos dependiendo del tipo de servicio. Cabe mencionar almacenamiento, ancho de banda, procesamiento, cuentas de usuario activas, entre otros.

2. Amazon Web Services – AWS

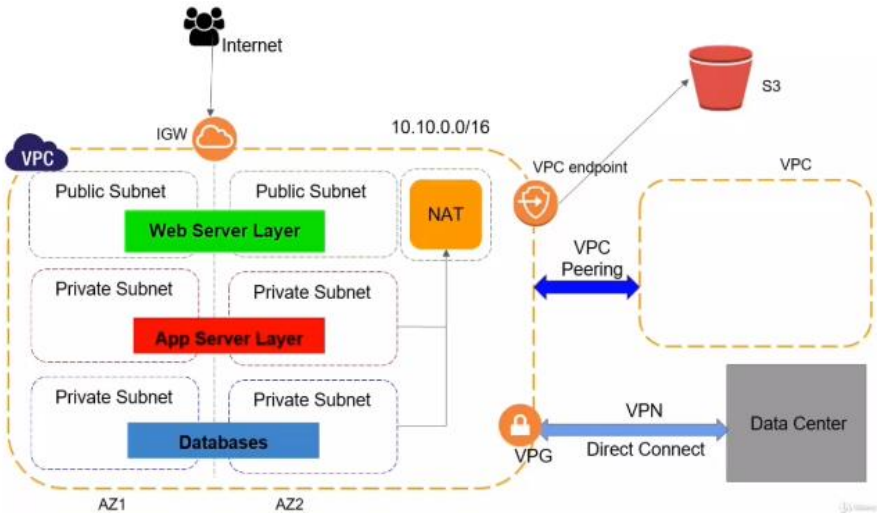
Como una solución factible para construir la base del sistema en proceso de desarrollo, se consideraron los servicios brindados por Amazon Web Services (AWS). Amazon AWS es un proveedor que brinda desde tecnologías de infraestructura como cómputo, almacenamiento y bases de datos, hasta tecnologías emergentes como el aprendizaje automático e inteligencia artificial y análisis e internet de las cosas (IoT).

a. ¿Por qué usar Amazon AWS?

- Facilidad de uso: AWS está diseñado para permitir que los desarrolladores de software puedan hospedar de una forma rápida y segura sus aplicaciones. (Amazon Web Services Inc., 2021)
- Flexibilidad: AWS permite la selección de distintos servicios según sea necesario, así como el o los lenguajes de programación a utilizar, el sistema operativo, la base de datos y más.
- Costo efectivo: los servicios de AWS son pagados según su consumo. Además, con la capa gratuita de uso permite utilizar hasta 85% de los servicios que brinda AWS sin costo alguno. Esta oferta varía entre los servicios en los que aplica, así como sus limitaciones.
- Escalabilidad: la solución de informática escalable en AWS permite implementar y operar un entorno de varios usuarios para flujos de trabajo con muchos recursos informáticos.
- Seguro: AWS cuenta con hardware distribuido alrededor del mundo. Esto quiere decir que los servidores, bases de datos y otros servicios de computación y almacenamiento pueden configurarse para estar distribuidos en distintas regiones físicas.

3. Convención de infraestructura en AWS

Ilustración 14: Implementación convencional de infraestructura en AWS.



(Gagan G.,2021)

De forma general, en la Ilustración 18 se observa una red virtual privada que contiene subredes internas públicas y privadas conectadas entre sí, compartiendo a los servicios internos de AWS utilizados en el sistema como bases de datos y servidores. Se cuenta con un gateway a internet que permite al sistema interactuar con el mundo exterior, así como endpoints para el consumo de los servicios prestados por agentes o personas externas al sistema.

El buen diseño planificado de infraestructura para un proyecto personal o ente empresarial es indispensable para asegurar el buen funcionamiento del sistema en cuestión. Es la base de la búsqueda de resiliencia ante posibles fallas y fiabilidad de los servicios prestados, que contendrán la aplicación con la que interactúa el usuario final.

4. Amazon EC2

Computación de la Nube Elástica, por sus siglas en ingles Elastic Compute Cloud (EC2), es un servicio web que permite provisionar de forma segura servidores en AWS. Es posible ajustar la capacidad y el número de servidores virtuales para acoplarse a las necesidades mientras cambian. (Mishra, 2017)

Con Amazon EC2 es posible configurar y poner en producción un servidor virtual en tan solo minutos. Solamente se financia por hora y es posible eliminar dicho servidor en caso de que ya no sea necesario en solo minutos.

Conceptos clave:

- Tipos de instancia: dictan la combinación precisa de CPU, RAM, Almacenamiento y Red para la creación de un servidor virtual.

- Grupos de seguridad (Security Groups): toman el rol de firewalls para cada instancia.
- Modelos de precio: por demanda, reservado y spot. En este caso se usará por demanda.
- Almacenamiento de datos
- Ciclo de vida de una instancia

(Mishra, 2017)

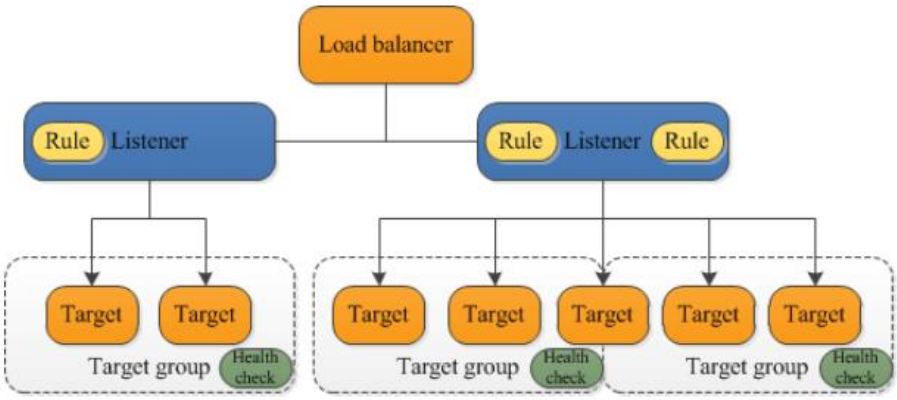
a. ¿Qué es un servidor?

Un servidor es una computadora con recursos especialmente orientados a las telecomunicaciones para la transmisión de informaciónn que es consultada por los clientes. Entre estos clientes cabe destacar otras computadoras, dispositivos móviles, impresoras, entre otros.

b. Balanceador de carga de aplicación - ALB

Según Tankariya y Parmar (2019) un Balanceador de Carga de Aplicación (Aplicación Load Balancer) es un servicio recientemente introducido por AWS, el cual permite enrutar tráfico basado en el contenido de una aplicación, que puede estar distribuida en distintos servicios y microservicios como una aplicación hospedada en EC2.

Ilustración 15: Componentes básicos de un balanceador de carga de aplicación.



(Amazon Web Services, 2021)

En la Ilustración 19 se pueden observarse algunos componentes clave del proceso de balanceo de carga. El balanceador de carga actúa como el punto de comunicación con el cliente, distribuyendo la información recibida. Esta distribución se hace a partir de reglas donde se especifica a que objetivo (Target) debe redirigirse el tráfico entrante. Reiterando, estos objetivos pueden estar representados por instancias EC2.

5. Amazon VPC

Amazon VPC quiere decir Nube Virtual Privada por sus siglas en inglés (Virtual Private Cloud). Según Wadia (2016) este servicio es una parte lógica aislada de la nube

proporcionada por AWS. Permite la creación y administración de topologías de red y subredes permitiendo la creación de instancias dentro de cada una. Es importante mencionar que a través de este servicio es posible decidir que instancias serán públicas o accedidas a través de internet y que instancias serán privadas. Esto es realizado por medio del uso de firewalls de red que hacen posible el control de tráfico circulante a través de cada instancia. Cada subred puede tener una configuración y firewall distintos que permiten el paso de cierto tipo de tráfico a través de la red. Algunos conceptos importantes de Amazon VPC a tomar en cuenta son:

- Subredes: representan contenedores lógicos con un rango especificado de IPs para asignar a los dispositivos virtuales contenidos dentro, como instancias EC2. Es posible crear subredes públicas y privadas. Las subredes públicas pueden ser accedidas desde internet mientras que las redes privadas solo pueden ser accedidas de acuerdo
- Grupos de seguridad (Security Groups): los grupos de seguridad son básicamente firewalls que filtran el tráfico entrante y saliente de acuerdo con las reglas especificadas. Los grupos de seguridad trabajan a nivel de instancias individuales.
- Lista de Control de Accesos de Red (Network ACLs): al igual que los grupos de seguridad filtran el tráfico entrante y saliente a través de reglas especificadas, pero las ACLs trabajan a nivel de subred. Por tal motivo, agregan otra capa de seguridad a las instancias del sistema.
- Tablas de Enrutamiento (Routing Tables): las tablas de ruteo son conjuntos de reglas para direccionar el tráfico circulante en las subredes de una VPC. Es posible utilizar una misma tabla de enrutamiento para varias subredes a la vez.
- Gateways de Internet (Internet Gateways): son dispositivos virtuales que proveen internet a las subredes públicas de una VPC. En una VPC personalizada es necesario hacer referencia a este dispositivo desde la tabla de enrutamiento de cada subred para obtener conectividad.
- Instancias NAT: solución para proveer internet a subredes privadas. Una instancia NAT sirve como un puente de una sola vía que permite consultas desde una subred privada hacia el exterior. Cabe destacar que una instancia NAT no permite consultas exteriores a la subred privada, siguiendo así la línea de seguridad deseada.

a. ¿Qué es una red de comunicación?

En el campo de las tecnologías de la información, es un conjunto de dispositivos interconectados entre sí a través de un medio, que intercambian información y comparten recursos. Esto sucede a través de tecnologías específicas que proveen de protocolos y facilidades en general, necesarias para el intercambio de información entre los usuarios de la red recurrente. Un usuario de red siempre hará referencia a un dispositivo funcional, ya sea operado por una persona o automatizado para cumplir algún tipo de trabajo.

6. Amazon RDS

Es un servicio de manejo de bases de datos que usa SQL para consultas. Permite crear bases de datos asociadas y administradas por AWS. AWS se encarga del manejo automático de respaldos, parcheo del sistema operativo, recuperación de información, protección a fallas, entre otros.

Entre los motores de bases de datos que se manejan en RDS están:

- PostgreSQL
- MySQL
- MariaDB
- AuroraDB

En este caso se utiliza MySQL debido a su compatibilidad con las tecnologías utilizadas en el módulo de Backend.

a. ¿Qué es una base de datos?

Según Oracle (2021) una base de datos es una colección de datos o información estructurada que normalmente esta almacenada de forma digital en un sistema tecnológico. Usualmente una base de datos es controlada por un sistema de administración de base de datos (Data Base Management System DBMS). A este conjunto de términos junto con las aplicaciones asociadas con ellos, se les conoce de forma general como una base de datos.

Hoy en día la información en las bases de datos más comunes es modelada como filas y columnas en una serie de tablas para asegurar un procesamiento y consultas eficientes. Las bases de datos relacionales utilizan el lenguaje de consultas estructurado (Structured Query Language - SQL) para escribir y consultar información. Este paradigma de almacenamiento de información sigue siendo ampliamente utilizado en la actualidad, sin embargo, han surgido nuevas ideas que pueden utilizarse como complemento a la idea original.

7. Amazon S3

S3 es un servicio de la nube de Amazon para el almacenamiento de objetos. Es altamente escalable y es fácil obtener acceso a través de la red de internet. Es posible utilizar S3 para almacenar y retribuir información virtualmente ilimitada, en cualquier momento desde cualquier lugar. Su nombre proviene de Almacenamiento Simple en la Nube (Simple Storage Service) y está diseñado para el albergue de documentos, imágenes, videos, archivos de logs, audio, archivos comprimidos y prácticamente cualquier extensión de archivo. Algunos conceptos importantes en S3:

- Bucket: es una unidad lógica en S3, similar a un directorio. Es un contenedor que puede almacenar todo tipo de objetos y también otros directorios. Debe ser creado con un nombre único y existe un límite de cien buckets por cuenta en AWS. Los objetos dentro de un bucket pueden ser accedidos por medio de la URL asociada y el identificador del bucket. Por ejemplo, si se almacena el archivo pdf archivo1 dentro del directorio archivos en el bucket con nombre prueba1 entonces se podría acceder a dicho archivo con la siguiente url: http://prueba1.s3.us-east-2.amazonaws.com/archivos/archivo1.pdf
- Sitio Web Estático: en las propiedades de cada bucket es posible habilitar la opción para que el bucket se utilice como el hospedaje para un sitio web estático. En esta funcionalidad básicamente se almacena el exportable generado (directorio) por alguna aplicación para el desarrollo de sitios web estáticos en el bucket configurado.

Luego, es posible observar el sitio web en un explorador por medio de la URL del bucket, como http://mi_sitio_estatico.s3-website.us-east-2.amazonaws.com.

a. ¿Por qué hospedar un sitio web en S3?

Gran parte de los sitios web hoy en día se han convertido en sitios web estáticos, lo cual significa que están basados en HTML, CSS y JavaScript puramente, sin ejecución de código del lado de un servidor como tal. Debido a que el modelo para desarrollo y conexión del sistema que se utilizará es el Modelo Vista Controlador, entonces las conexiones internas con la infraestructura son manejadas por el Backend. Por tal motivo, no hay razón para hospedar el sitio web en un servidor de forma tradicional.

Al usar esta configuración, es posible hospedar un sitio web por una cantidad pequeña de dinero al mes de forma sencilla, contando con las propiedades de alta escalabilidad características de S3.

8. Amazon CloudWatch

CloudWatch es un servicio de AWS utilizado para el monitoreo de varios servicios y aplicaciones alojados en la arquitectura en cuestión. Puede ser utilizado para obtener un numero de métricas de distintos servicios, lo cual permite un seguimiento de métricas y la inicialización de acciones dependiendo de puntos limite que sean configurados en dichas mediciones. (Tankariya & Parmar, 2019) Es posible configurar alarmas basadas en estos puntos límites y por ende ejecutar procesos de reinicio de instancias, almacenamiento de datos y más.

Por ejemplo, si el uso de CPU de un servidor supera el 85% de uso, es posible tomar acciones al respecto, como reiniciar el servidor o crear otra instancia que lo reemplace.

Algunos de los elementos que conforman este servicio son:

- Contenedores de métricas (Namespaces)
- Métricas
- Dimensiones
- Estadísticos
- Alarmas

Es importante hacer mención que un portal gráfico que permite la visualización de métricas relacionadas a los servicios utilizados en un sistema es de suma importancia. Un administrador puede fácilmente hacerse una idea del funcionamiento general de las aplicaciones respaldadas por la infraestructura al consultar dicha información.

9. Amazon Route 53 y CloudFront

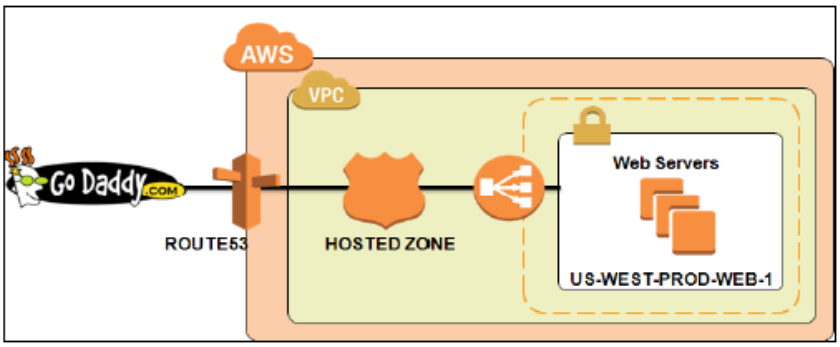
a. Route 53

Amazon Route 53 es un Servicio de Nombre de Dominios (Domain Name Service – DNS) altamente escalable y responsable del enrutamiento de usuarios hacia aplicaciones basadas en el internet. Este servicio trabaja como cualquier DNS, pero a una escala mucho

mayor, ya que se encarga de traducir nombres como www.leishmaniasis.com tanto a IPs de instancias como a endpoints de un balanceador de carga elástico o de Amazon S3. El verdadero poder de Route 53 reside en la capacidad que tiene de enrutar el tráfico de forma inteligente lo cual es conseguido con el uso de health checks y flujos de tráfico basados en rutas. (Wadia, 2016) Los chequeos de salud (health checks) son básicamente consultas realizadas de forma automática por AWS en los servicios configurados. Son útiles para monitorear el estado del funcionamiento de un servicio específico.

Recientemente Route 53 incorporó la funcionalidad para registrar dominios y de esta forma permitir al administrador tener control de varios nombres de dominios sin recurrir a proveedores externos a AWS.

Ilustración 16: Representación gráfica de funcionamiento básico de Route 53.



En la Ilustración 20 se pueden observarse varios componentes importantes en el uso de Route 53. En primer lugar, una zona de hospedaje (Hosted Zone) no es nada más que un contenedor que guarda información para el enrutamiento de tráfico de la aplicación actualmente manejada puede ser pública o privada dependiendo del contexto de implementación. Esta zona de hospedaje debe ser creada para un dominio especifico como www.leishmaniasis.com y luego realizar la configuración de registros (records) para que el servicio DNS comprenda como distribuir el tráfico entrante entre los servicios AWS que estén contemplados, en este caso un balanceador de carga.

b. CloudFront

Según Wadia (2016) Amazon CloudFront es un servicio proporcionado por AWS especialmente diseñado para la distribución y entrega de contenido alrededor del mundo. Es una alternativa a S3 al momento de la distribución de contenido web de forma geográfica. Existen varios procedimientos a tomar en cuenta para el uso correcto de este servicio, varios de ellos están englobados en dos pasos generales, los cuales son:

- Servidor de origen (origin server): la configuración de un servidor de origen es clave en el proceso, ya que es de este lugar donde CloudFront toma los archivos o el contenido para la respectiva distribución. Un servidor de origen puede estar representado por un bucket de S3 o hasta una instancia de EC2 contenida en una VPC. Los objetos almacenados en el servidor de origen pueden ser prácticamente cualquier objeto que pueda ser servido sobre el protocolo HTTP, como una página web.

- Distribución de Cloudfront (Cloudfront Distribution): la creación de una distribución de CoudFront describe cual servidor de origen utilizar cuando un usuario consulta una imagen u objeto multimedia en la aplicación o sitio web en cuestión. En este paso la distribución de CloudFront proporcionara una nueva URL que tiene como objetivo el servidor de origen recurrente, por ejemplo: http://112233.cloudfront.net/imagen.jpg.

Una vez concluidos estos pasos CloudFront envía esta distribución a distintas ubicaciones de borde (edge locations) que representan pequeños centros de datos donde se almacenan copias de memoria temporal (caché) de los objetos y los mantienen listos para su distribución alrededor del mundo. De esta forma, cuando un usuario accede o consulta un objeto CloudFront realizará una revisión de cache y, en todo caso, consultará el centro de datos más cercano para retribuir el objeto en cuestión.

Este acercamiento a la entrega de objetos multimedia es muy útil al momento del manejo de grandes cantidades de datos que son consultados por usuarios a grandes tazas de frecuencia. Además, en un sitio web muy concurrido, puede mejorar el rendimiento y aumentar la disponibilidad de los servicios.

Cabe mencionar que la aplicación pensada para el proyecto no tendrá que soportar grandes cantidades de consultas, ya que el número de usuarios de la aplicación es limitado y se encuentra entre dos a tres decenas. De cualquier manera, CloudFront presenta funcionalidades interesantes abriendo pauta para investigación y desarrollo de nuevo aprendizaje.

c. ¿Cómo se relacionan CloudFront y Route53?

Es posible crear una relación concisa entre ambos servicios al usar la URL generada en la distribución de CloudFront por un nombre de dominio personalizado y creado en Route53. Además, unir ambos servicios habilita la funcionalidad de monitorear el estado del o los servidores de origen por medio de la configuración de health checks y de esta forma enrutar las consultas a los servidores que estén saludables y estables.

10. AWS Lambda

AWS lambda es un servicio de computación serverless, manejado por eventos. El principal propósito de AWS Lambda es proveer una capa abstracta para hospedar funciones de aplicaciones y automatizar servicios secundarios en la nube. AWS lambda permite subir porciones de código fuente, las cuales son denominadas en conjunto como función lambda. Lambda tiene soporte para distintos lenguajes de programación, entre ellos están Java, Node.js, C#, Python y más.

Las funciones lambda son normalmente ejecutadas en base a un evento de algún servicio AWS. Estos eventos pueden ser un objeto nuevo ingresado a S3 o nueva información de un usuario ingresada a una base de datos en RDS, entre varios otros. Estas funciones tienen soporte en una gran variedad de eventos para la variedad de servicios AWS ofrecidos. Además de funcionar para eventos específicos, también es posible

calendarizar su ejecución o ser llamadas por medio de un API dependiendo del contexto de uso.

11. ¿Qué es un certificado seguro?

Los certificados son archivos que proporcionan seguridad en la comunicación de datos y aseguran la identidad de las entidades compartiendo dicha información. Amazon cuenta con un servicio proveedor de estos certificados llamado AWS Certificate Manager (ACM). Este servicio hace posible provisionar, administrar e implementar certificados públicos o privados SSL (Secure Socket Layer) y TLS (Transport Layer Security). ACM centraliza la administración de los certificados y simplifica el proceso manual tradicional de adquirir, subir y renovar estos archivos especiales antes mencionados. (Zillo Neto *et al.*, 2021)

Dentro de una implementación en la nube, ACM puede asignar de forma directa certificados para Balanceadores de Carga, Distribuciones de CloudFront, APIs implementadas y más.

a. ¿Qué es un certificado SSL?

SSL quiere decir Capa de Encajes Segura (Secure Socket Layer) y, en términos simples, es la tecnología estándar para establecer una conexión segura en internet, salvaguardando cualquier tipo de información sensible enviada entre dos sistemas, previniendo que cualquier tercero intercepte los datos para modificarlos o realizar cualquier acción maliciosa con ellos. Los dos sistemas pueden representar a un servidor y un cliente o a un servidor y otro servidor en comunicación. (Digisert, 2021)

La información transferida entre los sistemas siempre será imposible de leer al emplear los certificados. Se utilizan algoritmos de cifrado para codificar los datos en tránsito, evitando que puedan ser leídos cuando se envían a través de la conexión. Esta información puede ser cualquier información confidencial o personal, que puede incluir números de tarjetas de crédito y otros tipos de información financiera, nombres y direcciones.

TLS o Seguridad de Capa de Transporte (Transport Layer Security), según Digisert (2021), es únicamente una versión actualizada y más segura que la versión SSL. Normalmente se hace referencia a los certificados como SSL, ya que es la forma más común de llamarlos, de cualquier manera, al adquirir un certificado SSL se obtienen las opciones de encriptación ofrecidas por TLS.

Cabe mencionar que el acrónimo HTTPS (Protocolo de Transferencia de Hipertexto Seguro) aparece en la URL cuando un sitio web está asegurado con un certificado SSL.

12. Seguridad de los servicios

Según Davies (2011) el programador promedio no comprende la seguridad, hasta el momento en que, por sí mismo, se adentra en ella. Es importante que la seguridad no sea

tratada como una caja negra y se comprendan los procedimientos y las funciones relacionadas a ello en un sistema o un módulo específico del sistema.

Existen varios puntos a tomar en cuenta al momento de pensar en la implementación de seguridad en un sistema en la nube. AWS cuenta con varias soluciones para este objetivo, íntimamente relacionado con las tecnologías utilizadas en la aplicación que se hospedara en la infraestructura.

a. Infraestructura segura

AWS permite crear y personalizar redes virtuales privadas (VPCs) en donde es posible asegurar la infraestructura en cuestión con múltiples opciones como Security Groups, Network Access Control Lists, Tablas de Enrutamiento y más. AWS recomienda tomar en cuenta siempre dichas opciones para contar con varias capas de seguridad en cada dispositivo virtual que se utilice dentro de la o las subredes creadas.

b. Seguridad en balanceo de carga elástica

Los balanceadores de carga tienen soporte para encriptación cifrada de extremo a extremo, a través del protocolo TLS para redes que utilizan conexiones HTTPS. Realizar este proceso es sencillo, ya que únicamente debe crearse un certificado seguro en ACM y luego relacionarlo al balanceador de carga en cuestión. De esta forma cualquier consulta realizada desde el exterior a los servicios por detrás del balanceador de carga estará encriptada y se hará de forma segura.

13. Interacción de Backend con infraestructura

Es interesante hacer una retrospección sobre los componentes que conforman un sistema tecnológico, darle un significado a cada uno y, por último, relacionarlos para construir el todo que se busca.

La infraestructura de un sistema es la parte más interna del mismo, por así decirlo. Se le llama infraestructura al conjunto de recursos de hardware y software necesario para que cualquier sistema funcione. Estos recursos permiten conectividad, comunicaciones y gestión interna de los servicios proporcionados. El Backend sirve como un puente entre el FRONTEND y la infraestructura utilizada. Esto debido a que el Backend debe establecer conexión con los recursos de almacenamiento en la infraestructura para proporcionar al sitio o página web las funcionalidades que busca el cliente final. Resumiendo:

- Un sitio web únicamente desarrollado en FRONTEND, sería atractivo para el usuario, pero su funcionalidad sería nula.
- Un sitio web únicamente desarrollado en Backend, no tendría forma de interactuar con el usuario para utilizar los recursos retribuidos desde la infraestructura.
- La infraestructura por sí sola no representa una aplicación, pero sí representa la base y el sitio donde pueden resguardarse los componentes que conforman dicha aplicación tecnológica.

G. Sistemas y tecnologías web

1. Dashboard web

Un dashboard es una herramienta de gestión de la información que monitoriza, analiza y muestra de manera visual los indicadores clave de desempeño (KPI), métricas y datos fundamentales para hacer un seguimiento del estado de una empresa, un departamento, una campaña o un proceso específico. (Ortiz, 2021).

2. Framework de programación

Según Muente: "Framework es una estructura previa que se puede aprovechar para desarrollar un proyecto. El Framework es una especie de plantilla, un esquema conceptual, que simplifica la elaboración de una tarea, ya que solo es necesario complementarlo de acuerdo con lo que se quiere realizar" (Muente, 2021)

En sectores de programación, se usan los frameworks para no "inventar" la rueda de nuevo. Esto significa que son bases de programación ya hechas para que desarrolladores no tengan que iniciar de nuevo, sino usar las bondades y los componentes de cada framework para su beneficio. Esto, reserva tiempo y ayuda a ser más productivo.
Muchos frameworks son sostenidos por la comunidad de programadores, mientras que otros son de pago. Los frameworks que son relevantes para el desarrollo de objetivos son los siguientes.

a. React native

React Native es un framework JavaScript para crear aplicaciones reales nativas para iOS y Android, basado en la librearía de JavaScript React para la creación de componentes visuales, cambiando el propósito de estos para, en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas, en este caso iOS y Android. Es decir, en lugar de desarrollar una aplicación web híbrida o en HTML5, lo que se obtiene al final como resultado es una aplicación real nativa. (Campillo, 2020)

b. React.JS

React está basado en un paradigma llamado programación orientada a componentes en el que cada componente es una pieza con la que el usuario puede interactuar. Estas piezas se crean usando una sintaxis llamada JSX permitiendo escribir HTML (y opcionalmente CSS) dentro de objetos JavaScript. Estos componentes son reutilizables y se combinan para crear componentes mayores hasta configurar una web completa. Esta es la forma de tener HTML con toda la funcionalidad de JavaScript y el estilo gráfico de CSS centralizado y listo para ser abstraído y usado en cualquier otro proyecto. (Campillo, 2020)

H. UX/UI y términos de diseño

3. UX/UI

UX (por sus siglas en inglés User eXperience) o en español Experiencia de Usuario, es aquello que una persona percibe al interactuar con un producto o servicio. Se logra un buen UX al enfocarse en diseñar productos útiles, usables y deseables, lo cual influye en que el usuario se sienta satisfecho, feliz y encantado. (Worthington, 2021).

UI (por sus siglas en inglés User Interface) o en español Interfaz del Usuario, es la vista que permite a un usuario interactuar de manera efectiva con un sistema. Es la suma de una arquitectura de información + elementos visuales + patrones de interacción. (Worthington, 2021).

Según Worthington la diferencia entre UX y UI es: "Una forma simple de pensar en esto es decir que el UI es orientado por el diseño y el UX es orientado por el usuario"

La diferencia a nivel de oración no es mucha, pero en cuestiones de practicidad y de otros valores, es bastante alta. Es muy difícil que exista un rol de una persona que cumpla con ambos rubros: tanto UX como UI. La razón es que UI maneja el diseño visual que es tangible y físico. Para lograr la "tangibilidad" se usa de programas de software como Photoshop e Illustrator. Mientras, áreas como UX se encargan de hacer o de diseñar algo más proposicional. Intenta crear una experiencia que en ocasiones NO existe, de manera que tienen que mapear, planear e imaginar cómo es esa experiencia para lograr hacerla "tangible".

4. Moodboard

Según Seoane un MoodBoard es: "una herramienta creativa que consiste en una visualización rápida de imágenes y palabras en un mismo soporte, a modo de lluvia de inputs que nos ayuden a preparar el cerebro para la fase de ideación de un proyecto, de ahí lo de inspiración."

Un moodboard como se citó anteriormente, sienta las bases tanto imaginativas como especulativas de un proyecto. Al ser creado en una etapa tan temprana, esta herramienta apoya ideas y valores que van a ser la fundación del proyecto o solución entera. También es una buena forma de iterar y de tomar soluciones rápidamente. Un moodboard bien hecho e iterado servirá como guía para el futuro y la creación de otras fases.

5. Convenciones y términos de diseño UX/UI

a. Pre design

Es una etapa donde se planea todo el proyecto. Dará como salida un documento técnico con preguntas sobre las necesidades del proyecto, etc. (Worthington, 2021). Para esta etapa es importante plantearse preguntas objetivas al iniciar la etapa, para conocer la solución, por ejemplo:

- ¿Cuáles son las metas de la solución?
- ¿Cuáles son las limitaciones?
- ¿En qué plataforma estará la solución; donde va a vivir?
- ¿Cuál es la audiencia?

La razón es que el diseño y el UX será el experto en términos del contenido que se dará. Luego, se establecen convenciones que se tomarán en todo el diseño de la solución.

b. Convenciones de diseño

Lo ideal sería cubrirlas todas, pero por cuestiones de popularidad, época y cultura, queda a criterio del diseñador cuáles quedan fuera (Worthington, 2021). Entre las convenciones que se toman en consideración y entre las más usadas están las siguientes:

- Conocimiento del mundo real: se basa en modelos analógicos. Por ejemplo, se comprende la funcionalidad de los botones digitales en una pantalla porque se ha experimentado los botones analógicos en el mundo real: en calculadoras, controles remotos, ascensores, etc.
- Comportamiento aprendido: sí el conocimiento del mundo real se basa en modelos analógicos, entonces quizás pueda decirse que el comportamiento aprendido se basa en modelos digitales. Por ejemplo, deslizar el dedo hacia la izquierda o hacia la derecha es una convención digital que ahora es bastante común entre muchas aplicaciones. Es una acción que los usuarios han aprendido al usar teléfonos inteligentes.
- Causa y efecto: se trata de acción y reacción. Cuando un usuario presiona un botón y sucede algo, comprende rápidamente la funcionalidad del botón. Por ejemplo, presionando "Enviar" para enviar un mensaje de correo electrónico, o "Enviar" para enviar un formulario.
- Consistencia: consiste en aplicar una lógica sistemática a la forma en que las interfaces se ven y funcionan para que sigan siendo familiares para el usuario. Por ejemplo, en el teclado de un teléfono, donde todos los botones numerados tienen el mismo aspecto, puede anticipar que presionar el número “8” tendrá más o menos la misma función que presionar el número “1”.
- Interconexión: consiste en garantizar que la acción y la reacción estén estrechamente vinculadas. La interacción debe ser fácil (con el menor número de pasos posible) y los resultados deben ser inmediatos.
- Intuición: nos han condicionado que para interactuar con una interfaz necesitamos tocarla de alguna manera. Si bien el sistema puede no ser obvio de inmediato, la solución en general se descubre rápidamente.
- Regocijo: es la recompensa para el usuario. Una interfaz debería funcionar como estaba prevista, pero hay otras formas de reafirmarse como una experiencia positiva.
- Hacer/deshacer: le da al usuario la tranquilidad de que sus acciones son reversibles. Una buena interfaz de usuario permitirá al usuario cambiar de opinión después de realizar una acción, o al menos notificar al usuario antes de dar un paso crucial. Puede ver algunos ejemplos de esto en el servicio de correo web de Google, Gmail, desde notificarle si posiblemente olvidó adjuntar un documento a un correo electrónico saliente, hasta darle un breve período de tiempo para "deshacer" un mensaje enviado antes. deja su bandeja de salida.

c. Look and feel

Puede traducirse el Look como el estilo y el Feel como el humor/intención de la solución (Worthington, 2021). Esta etapa es de pre lanzamiento, pregasto de recursos, pre-ajuste de elementos. En esta etapa el interés es imaginar cómo será la solución. Para ayudar a enfocar este camino, se realizará un MoodBoard, que es una recopilación de colores, fotografías y tipografías. Su objetivo además de enfocar, es reflejar a la audiencia qué es la aplicación y dónde va a ser desplegada.

En esta etapa también se contemplan los elementos formales de la interfaz de usuario. Estos elementos son la base del resto de la aplicación. Eventualmente, se usarán para prototipar. Los elementos formales se dividen en pasivos y activos.

Se realiza también en esta etapa una Pattern Library, que es una colección de elementos de la interfaz de usuario, incluyendo botones, íconos, tipografía, animaciones entre botones, estados, etc. Debe ser lo más detallado posible para usar como base el MoodBoard.

V. METODOLOGÍA

A. Inteligencia artificial para reconocimiento de lesiones de Leishmaniasis

1. Recolección de imágenes

Para iniciar el proceso de construcción del modelo de inteligencia artificial, lo primero que se llevó a cabo, fue la recopilación de imágenes, tanto de casos positivos de Leishmaniasis cutánea como de enfermedades con síntomas similares (se tomaron como casos negativos). Esto con el fin de hacer un modelo más robusto. Cabe mencionar que las imágenes que se utilizaron para la construcción del modelo de inteligencia artificial no están ligadas a información personal identificable del paciente del cual se obtuvieron.

Para esta etapa se contó con distintas fuentes de imágenes, la primera fue la extracción de imágenes del pdf de "Atlas interactivo de Leishmaniasis en las Américas: aspectos clínicos y diagnósticos diferenciales" (OPS, 2020), el cual está disponible de forma gratuita en el Internet. También se contó con el apoyo de trabajadores del Programa de Vectores del MSPAS, quienes compartieron fotos de las lesiones que atendieron en el campo, etiquetándolas únicamente como positivas o negativas, sin incluir identificadores personales de los pacientes.

Otra fuente de datos fueron las imágenes recolectadas por el Centro de Estudios en Salud de la Universidad del Valle de Guatemala a lo largo de los años para proyectos de investigación o actividades de diagnóstico, de las cuales tampoco se obtuvo información privada identificable.

Finalmente, para complementar las imágenes de casos negativos se utilizó la lista de enfermedades mencionadas en el Anexo 1. Estas enfermedades se buscaron utilizando las siguientes herramientas:

- Motor de búsqueda de Google
- Motor de búsqueda de Microsoft (Bing)
- Motor de búsqueda de Science Source (agencia de fotografía)

2. Partición del conjunto de datos

Posterior a la recolección de las imágenes, se mezclaron las imágenes de las distintas fuentes y separaron en las siguientes categorías: Entrenamiento, Desarrollo y Prueba. Siempre tomando en cuenta que las fotografías del mismo paciente deben estar en la misma categoría, esto debido a que el modelo podría recordar características específicas de cada individuo y engañarnos con un rendimiento alto.

Aproximadamente el set de datos total es de tres mil imágenes positivas y tres mil imágenes negativas, por lo que se decidió dividir las categorías de la siguiente forma: Entrenamiento 60%, Desarrollo 20% y Prueba 20%.

3. Selección de librería para desarrollar el modelo

Luego de la investigación realizada nos quedamos con las tres mejores librerías TensorFlow, Scikit-Learn y PyTorch. Descartamos Scikit-Learn debido a que identificamos que era una librería usada para hacer bocetos no para desarrollo de modelos profundos. Por lo que la decisión se resumió a TensorFlow o PyTorch.

Ambas librerías tienen características muy similares, muy completas y utilizadas por grandes compañías. En cuanto a la documentación de estas, TensorFlow posee una documentación mucho más extensa, así como también en el caso de la comunidad detrás de las librerías es mucho más amplia nuevamente TensorFlow. Por estas razones se decidió usar este módulo para llevar a cabo los modelos de este proyecto.

4. Determinación de rendimiento humano

Como se mencionó anteriormente el rendimiento humano da una idea de la complejidad del problema que se está intentando resolver y sirve de orientación para comparar los resultados obtenidos con el modelo entrenado.

Para encontrar la eficiencia humana se realizó un examen por medio de Google Docs. En ella se encontraban 30 imágenes de lesiones, 15 positas a Leishmaniasis cutánea y el resto enfermedades con síntomas similares (casos negativos). Estas imágenes se seleccionaron de forma aleatoria, y se mostraron al profesional encuestado en orden aleatorio también.

5. Desarrollo del modelo

Para el desarrollo del modelo se utilizaron redes neuronales convolucionales. Debido a que no se dispone de gran cantidad de imágenes se utilizó la técnica de aprendizaje transferido y aumentación de datos. Se realizó varias iteraciones para optimizar los modelos y poder mejorar el rendimiento progresivo. Para este proceso de optimización se utilizó un algoritmo de optimización de hiperparámetros junto con el análisis de Sesgo/Varianza y análisis de error.

A continuación, se explicará brevemente en qué consistieron estos análisis para encontrar los mejores hiper-parámetros y arquitecturas del modelo para resolver este problema en específico.

6. Optimización de hiperparámetros

Como vimos en secciones anteriores, el mejor algoritmo encontrado para optimizar hiperparámetros es BOHB el cual une las mejores cualidades de otros algoritmos de optimización. Los creadores de este algoritmo ofrecen una implementación en Python la cual se utilizó, el nombre de la librería es HpBandSter.

Los hiperparámetros que se configuraron para que se optimizaran son los siguientes:
Aumentación de datos:

- **featurewise_center:** Valor booleano, en el caso de ser verdadero, establece la media sobre el conjunto de datos en 0.
- **samplewise_center:** Valor booleano, en el caso de ser verdadero, establece la media de cada muestra en 0.
- **featurewise_std_normalization:** Valor booleano, en el caso de ser verdadero, establece la desviación estándar del conjunto de datos en 1. Esto se logra dividiendo los datos por la desviación estándar.
- **samplewise_std_normalization:** Valor booleano, en el caso de ser verdadero, establece la desviación estándar de cada muestra en 1.
- **zca_whitening:** Valor numérico para definir el epsilon para el algoritmo de blanqueamiento ZCA. Este es un método de preprocesamiento de imágenes. Una imagen sin este preprocesamiento es redundante, ya que los valores de los píxeles adyacentes están altamente correlacionados. Lo que busca el blanqueamiento es hacer menos correlativos los píxeles adyacentes (El-Amir *et al*, 2020).
- **rotation_range:** Valor numérico para definir el rango de grados para rotaciones aleatorias.
- **width_shift_range:** Valor numérico para definir fracción del ancho del cuál se desplazará la imagen.
- **height_shift_range:** Valor numérico para definir fracción del alto del cuál se desplazará la imagen.
- **brightness_range:** Rango numérico para elegir un valor de cambio de brillo de la imagen.
- **horizontal_flip:** Valor booleano. En el caso de ser verdadero, las entradas se voltean horizontalmente de forma aleatoria.
- **vertical_flip:** Valor booleano. En el caso de ser verdadero, las entradas se voltean verticalmente de forma aleatoria.
- **shear_range:** Valor numérico. Indica la intensidad de deformación de la imagen.
- **zoom_range:** Valor numérico. Indica el rango para acercar la imagen de forma aleatoria.

(TensorFlow, 2021)

Relacionados con el modelo:

- Velocidad de aprendizaje.
- *Droput Rate.*
- Cantidad de neuronas en una capa previa a la salida.

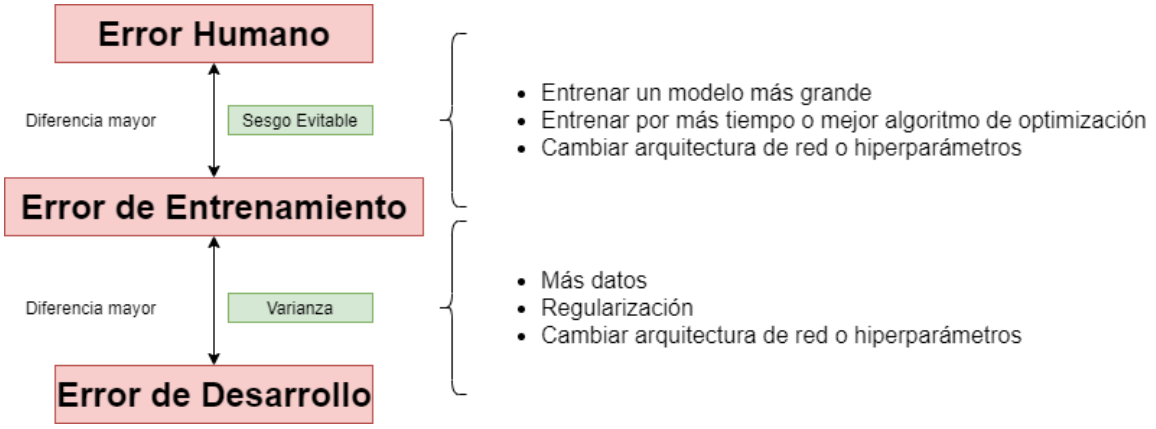
7. Análisis sesgo/varianza

Este análisis se realizó luego de cada iteración de entrenamiento del modelo. Es necesario considerar tres tipos de errores: el error humano, se refiere al porcentaje de fallo de un especialista al clasificar imágenes de lesiones de Leishmaniasis cutánea, el error de entrenamiento se refiere al porcentaje de fallo del modelo al clasificar imágenes dentro del conjunto de entrenamiento y finalmente el de desarrollo, se refiere al porcentaje de error del modelo al clasificar imágenes dentro del conjunto de desarrollo.

Si la diferencia entre el error humano y el error de entrenamiento es más grande que la diferencia entre el error de entrenamiento y el error de desarrollo, entonces se dice que se tiene "Sesgo evitable" y el siguiente paso sería entrenar un modelo con más capas ocultas, con más tiempo, con un mejor algoritmo de optimización, cambiar la arquitectura de la red o los hiper-parámetros.

Si la diferencia entre el error de entrenamiento y error de desarrollo es más grande que la diferencia entre el error humano y el error de entrenamiento, entonces se dice que se tiene Varianza y el siguiente paso sería conseguir más datos, implementar regularizaciones, cambiar arquitectura de la red o los hiper-parámetros.

Ilustración 17: Resumen del análisis sesgo/varianza.



Nota. del lado derecho del gráfico, se muestran los pasos a seguir para corregir cada tipo de error con el fin de intentar disminuirlos y así obtener un mejor rendimiento en el modelo que se está evaluando.
(Fuente: elaboración propia)

8. Análisis de error

Si el porcentaje de error entre el humano y el modelo es mayor a 20% se realiza un análisis de error, en donde se determinan los casos en los que el modelo está cometiendo errores al clasificar una imagen, junto con el porcentaje del total en que cada error se presenta, y así se determina cuál es el problema para revisarlo y obtener un mejor desempeño. A continuación, se muestra un cuadro de ejemplo:

Tabla 2: Ejemplo de análisis de error

Identificador de imagen	Imágenes borrosas	Etiqueta incorrecta
…	…	…
99	✓	
100		✓
% del total	7%	61%

Nota. En esta tabla se muestra cómo se llevaría el conteo de fallos de un modelo.

Como se puede observar en el ejemplo anterior, el 7% de los fallos proviene de imágenes borrosas y un 61% a etiquetas incorrectas del conjunto de imágenes de entrenamiento. Posterior a la elaboración del cuadro, se determinará cuál es el problema con mayor porcentaje del total y con el menor costo para proceder a corregirlo.

9. Compresión de modelo

Luego de las iteraciones que fueron necesarias para la construcción del modelo final, se evaluó el rendimiento utilizando el conjunto de imágenes de prueba. Se tomó en cuenta el porcentaje de falsos positivos, falsos negativos y precisión. Tras analizar estos resultados y determinar que estaba listo para implementarlo en la aplicación móvil, se comprimió el modelo utilizando TensorFlow. Básicamente este proceso convierte los valores de los pesos del modelo de un número con punto decimal a número entero, esto con el fin de hacer las operaciones más sencillas y que el teléfono sea capaz de procesar una imagen (TensorFlow, 2021).

B. Gráficas de panel de control

1. Generación del conjunto de datos

Como se puede observar en el Anexo 2 el Ministerio de Salud llena formularios a mano en donde se registra información valiosa. Debido a que esta información no está de forma digital se desarrolló un programa para generar un conjunto de datos con el que se pudiera trabajar. El objetivo era preparar datos que permitieran realizar gráficas e implementar un portal en donde se puedan ingresar los datos que actualmente se llenan a mano y a su vez se pueda visualizar y analizar la información.

2. Selección de plataforma para desarrollar las gráficas

Tras investigar plataformas para desarrollar gráficas se decidió investigar dos alternativas: Microsoft Power BI y Google Data Studio.

Por lo tanto, esta plataforma debía de disponer la opción de embeber los tableros en una página web. Ambas plataformas brindaban la opción, pero únicamente la herramienta de Google la ofrecía de forma gratuita, por lo que esa fue la herramienta que finalmente se seleccionó.

3. Selección de gráficas

En el Anexo 3 se puede observar la priorización de los datos por parte de las partes interesadas. Esto lo analizó el equipo de *backend* y finalmente cuando se tenía la arquitectura de la base de datos y se tenía claro que información se guardaría se prosiguió con el análisis de cómo mostrar los resultados para que cualquier persona pudiera sacar conclusiones.

El proceso para decidir cómo organizar las gráficas fue primero definir una finalidad u objetivo que debía cumplir esa pantalla y posteriormente a partir de los datos que se almacenan proponer gráficos o indicadores que se alinearan con la finalidad de la pantalla.

Nuevamente, con retroalimentación de las partes interesadas se decidió desarrollar una pantalla de inicio y tres secciones adicionales de gráficas. A continuación, una breve explicación de cada una de las partes:

a. Página inicio

Finalidad: poder sacar conclusiones rápidas de las tres pestañas de gráficos, nada muy complejo de analizar.

Indicadores:

- Cantidad de casos del año actual del portal.
- Casos por sexo en el año.
- Departamento con más casos en el año actual.
- Rendimiento promedio de pruebas realizadas con el modelo.
- Cantidad de casos del año actual de la aplicación.
- Mapa de calor de Guatemala con número de casos del año actual.

b. Gráficos personas

Finalidad: poder analizar características de las personas con lesiones de Leishmaniasis cutánea. Podría ser útil para ver qué características comparten los casos positivos.

Indicadores:

- Cantidad de lesiones promedio por paciente.
- Diámetro de lesiones promedio por paciente.
- Cantidad de casos por rango de edades.
- Proporción de casos por prueba de diagnóstico
- Proporción de casos según condición de egreso.
- Distribución de casos según el sexo.
- Distribución de casos según procedencia.
- Distribución de casos según grupo étnico y lingüístico.
- Distribución de casos según ubicaciones de las lesiones por paciente.

c. Gráficos históricos

Finalidad: mostrar el cambio de las variables a lo largo del tiempo.

Indicadores:

- Número de muertes y letalidad a lo largo del tiempo
- Cantidad de casos a lo largo del tiempo.

Filtros:

- Departamento.
- Sexo.
- Tiempo (poder seleccionar rango de tiempo).

d. Gráficos geográficos

Finalidad: poder analizar toda la información respecto al lugar geográfico.

Indicadores:

- Mapa de casos por departamento.
- Comparación de casos según región.
- Tabla con los siguientes datos:
- Departamento.
- Distrito.
- Edad y género.
- Gráfico de barras de casos por departamento
- Distribución de casos según procedencia.
- Distribución de casos según grupo étnico y lingüístico.
- Distribución de casos según ubicaciones de las lesiones por paciente.

Filtros:

- Sexo.
- Tiempo (Poder seleccionar rango de tiempo).

C. Backend

1. Análisis de datos existentes

Durante las fases de planeación del proyecto, se organizó múltiples reuniones con el personal encargado de la comunidad objetivo, es decir, personal de la Universidad del Valle de Guatemala y sus contactos del Ministerio de Salud Pública y Asistencia Social. Para entrar en un contexto más profundo del problema y la solución que se ofrecerá, estas personas brindaron su flujo actual de cómo manejan la situación de Leishmaniasis en las comunidades, adjuntando también formularios, fichas médicas y otros datos que todo el personal utiliza. Esto resultó útil ya que da lugar al análisis superficial de datos y así utilizarlos las para el desarrollo de la arquitectura del problema.

Parte de este análisis fue realizado contestando las siguientes preguntas: ¿Qué datos tenemos? ¿Qué datos son obtenidos a diario y cuáles son constantes? ¿Cómo podemos organizar y relacionar las diferentes secciones de datos? Entre otras preguntas que conllevan a tomar decisiones efectivas sobre el diseño del *backend* y la base de datos. También se tomó como referencia los datos que utiliza el MSPAS, en conjunto con los formularios que los técnicos de salud llenan con el fin de automatizar o facilitar el proceso.

Tabla 3: Datos del paciente almacenados en registros de caso de Leishmaniasis.

REGISTRO DE CASOS DE LEISHMANIASIS												
Datos del Paciente						Tipo de búsqueda		Procedencia				
Fecha de registro d/m/a	Número correlativo anual	Sexo (M/F)	Edad (meses y años)	Distrito de salud	Ocupación	Activa	Pasiva	Urbana	Rural	Grupo étnico y lingüístico	Migrante: Indicar de donde proviene	
información indispensable												
Es importante, pero no indispensable												

(Fuente: Ministerio de Salud Pública y Asistencia Social)

Tabla 4: Características clínicas en un registro de caso de Leishmaniasis.

Características clínicas							
LC*	LCA**	LMC***	LV****	Cantidad y diámetro de las lesiones	Ubicación de las lesiones	Tiempo de evolución (meses o años)	Existencia de lesiones cicatrizales (Indique dónde)

(Fuente: Ministerio de Salud Pública y Asistencia Social)

Tabla 5: Diagnóstico y tratamiento en el registro de un caso de Leishmaniasis.

Prueba de diagnóstico			Tratamiento		Condición de egreso				
					Vivo				
Frote Directo o biopsia	Diagnóstico Clínico o nexo epidemiológico	PCR*	Medicamento (antimoniato de meglumine)	Termoterapia	Alta curada	Alta con mejoría clínica	Alta no curada	Abandono	Fallecido

(Fuente: Ministerio de Salud Pública y Asistencia Social)

Tabla 6: Ficha clínica para el registro de un caso de Leishmaniasis.

LEISHMANIASIS CUTÁNEA Y MUCOCUTÁNEA

A. DATOS GENERALES

Fecha: ___/___/___Nombre completo _____

Fecha de nacimiento:–_____ DPI_____

Edad: _____Sexo: M ☐ F☐ Ocupación: _____

Grupo étnico: _____ Grupo lingüístico: _____

Procedencia: Urbana _____ Rural _____ Migrante_____

Dirección completa (o puntos de referencia):_____

Departamento: _____ Municipio: _____

Aldea:_____ Caserío:_____

Jefe de casa o persona responsable (en caso de menores de edad)_____

Teléfono: _____

¿A viajado o permanecido en Peten, Huehuetenango, Quiché, Alta Verapaz, Baja Verapaz, Izabal, El Progreso? Sí ☐ No ☐

Si ha viajado a una zona que no sea las anteriores, indicar a donde viajó:_____

¿Ha tenido Leishmaniasis anteriormente? Sí ☐ No ☐ No sabe ☐

Indique cuándo (mes – año) _____

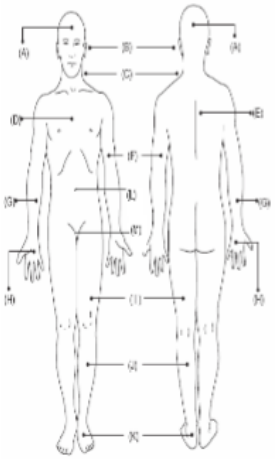
¿Reside en alguna zona endémica de leishmaniasis? Sí ☐ No ☐

Peso: _____lb _____kg

C. PACIENTE SOSPECHOSO DE LEISHMANIASIS CUTÁNEA O MUCOCUTÁNEA

DATOS CLINICOS

Dibuje la localización de las lesiones en el diagrama de abajo



Localización de la(s) Lesión(es)	Cantidad	Diámetro (cm)
A) Cara	☐	_____
B) Orejas	☐	_____
C) Cuello	☐	_____
D) Tórax anterior	☐	_____
E) Tórax posterior	☐	_____
F) Brazos	☐	_____
G) Antebrazos	☐	_____
H) Manos	☐	_____
I) Muslos	☐	_____
J) Piernas	☐	_____
k) Pies	☐	_____
L) Abdomen	☐	_____
M) Genitales	☐	_____

OTROS _____

(Fuente: Ministerio de Salud Pública y Asistencia Social)

a. Abstracción de los datos a entidades

Parte del procedimiento de un diseñador de bases de datos, es poder abstraer la información usada o almacenada por la población objetivo en entidades o tablas de bases de datos. Por ello, se creó una serie de entidades las cuales se tomó como base para poder así crear las tablas que conforman la base de datos. Cabe mencionar que no todos los datos que presentan en los cuadros y figuras son indispensables por lo que no todas las propiedades de una entidad serán obligatorias. Las entidades creadas, junto con sus propiedades son las siguientes:

- Paciente: no posee datos con los que se pueda identificar a una persona, está compuesto de:
    - Género
    - Edad
    - Ocupación
    - Procedencia (de dónde proviene el paciente)
    - Diagnóstico (realizado dentro de la clínica)
- Procedencia: Se refiere a dónde proviene un paciente.
    - Municipio
    - Grupo étnico
    - Departamento
    - Área (urbana, rural o migrante)
- Diagnóstico: El diagnóstico realizado hacia el paciente
    - Fecha
    - Distrito de salud
    - Prueba de diagnóstico (tipo de prueba que se le realiza)
    - Tratamiento
    - Egreso
    - Una o varias características clínicas
    - Una o varias lesiones cicatrízales
- Característica clínica
    - Parte del cuerpo
    - Diámetro de la lesión
    - Tiempo de evolución
    - Cantidad
- Lesión cicatrizal
    - Parte del cuerpo

2. Diseño de la base de datos

Tras múltiples reuniones con la comunidad objetivo y el intercambio de archivos para conocer a detalle los datos con los que las personas trabajan, se realizó un esquema de base de datos capaz de almacenar eficientemente aquella información de valor que se utilizará a lo largo del proyecto. Este es el primer paso para determinar los requisitos del proyecto, ya que, para crear una base de datos, se debe planificar el tipo de información que se quiere almacenar en la misma, se debe tomar en cuenta la información disponible y la que se necesita recopilar.

La planificación de la estructura de la base de datos, en especial para modelos relacionales, es importante para la gestión efectiva del sistema. Con el uso de la información proporcionada por los agentes de MSPAS, se definió a detalle los datos que componen el sistema en base a tablas como: nombre, tipo de campo, tamaño de los datos, cómo se relacionan entre sí, etc. Se optó por usar una base de datos relacional, ya que los datos sí poseen una relación que se puede aprovechar y el sistema a crear no tiene como objetivo ser flexible cuando al almacenamiento de datos se refiere. Una de las mejores formas de visualización de la arquitectura creada es un diagrama entidad relación, lo que permitió completar esta fase del proyecto.

3. Creación de la base de datos

Para la base de datos, se eligió un modelo relacional para esta. La principal razón reside en la sencillez, puesto que permite manejar grandes cantidades de datos con puntos de relación entre sí, para una gestión segura, conforme a las normas programas y estándares bien definidos. Posterior a la creación del diseño base de datos, se programaron múltiples archivos con el fin de replicar este diseño dentro de la base de datos real. Estos scripts contenían código para crear la base de datos, crear las tablas con sus respectivos campos, valores por defecto, llaves foráneas, entre otros. También se programaron scripts para poblar algunas tablas, por ejemplo, las tablas de departamentos, municipios, centros de salud, etc. deberán tener valores por defecto para luego obtener su id y estandarizar el tipo de dato.

Cabe mencionar que se utilizó el motor de base de datos MariaDB, el cual es derivado de MySQL. Estos scripts poseen la sintaxis necesaria para que cualquier otro motor de base de datos SQL pueda correrlos y así tener el mismo resultado en la base de datos. El fin de estandarizarlos, es para que tanto cualquier desarrollador pueda montar la infraestructura fácilmente, y para que servidores (ya sean en la nube o locales) puedan ejecutarlos en cualquier momento para obtener una base de datos lista para el almacenamiento correcto de datos.

4. *Framework* por utilizar y librerías de apoyo

Se decidió proseguir con esta etapa bajo el *framework* de desarrollo Express. Este *framework* ha dado mucho de qué hablar en los últimos años, puesto que es extremadamente ligero, flexible y proporciona características para aplicaciones web y móviles. También tiene un alto rendimiento y se implementó bajo la arquitectura REST.

Para la interacción con la base de datos, se utilizó la librería Sequelize. Posee compatibilidad con múltiples motores de base de datos y brinda la capacidad de diseñar un modelo idéntico al diseñado en la base de datos con el fin de garantizar transacciones sólidas, seguras y con relaciones existentes desde la base de datos. Además, para garantizar un código limpio, tipado y libre de errores, se utilizó el superset de TypeScript dentro de Express. TypeScript hará que el código sea tipado, fácil de escalar y detectará errores de forma temprano para que el desarrollo sea mucho más fluido y estandarizado.

El modelo MVC fue elegido. Esta arquitectura se eligió puesto que es una de las más fáciles de escalar, en el caso que sea necesario. Las clases, objetos e interfaces son independientes entre ellos por lo que crear nuevas acciones o funcionalidades dentro del sistema y la aplicación resultará en una tarea muy sencilla. Además, Express es un *framework* muy flexible y permite realizar una gestión de archivos adecuada que resulta muy útil para completar esta tarea.

- **Modelos:** en la infraestructura creada, los modelos se realizaron con interfaces de TypeScript. Esto permite asignar una propiedad a cada entidad, junto con el tipo de

dato con el que se estará trabajando cada una. Además, se pueden predefinir funciones u otros atributos a utilizar.

- **Controlador:** para cada modelo (ya sea de base de datos o desde la interfaz creada) se creó un controlador, el cual permitirá ejecutar acciones interactuando con base de datos y realizando procedimientos internos. También existen controladores que utilizan múltiples modelos de bases de datos, por ejemplo, el formulario. Se tendrá que crear una entidad procedencia, paciente, diagnóstico y característica clínica en una sola consulta.
- **Vista:** En el caso de la vista, será manejada desde *frontend*. Cabe mencionar que no todos los modelos poseen una vista, en muchos casos varias entidades fueron acopladas a una sola vista con el fin de ahorrar espacio y consultas innecesarias hacia el servidor.

5. Configuración de *cors*

CORS es uno de los estándares de seguridad web más importante, puesto que protege al usuario de transacciones externas y de dominio cruzado. Bajo este concepto, se configuró el servidor programado en Express de forma que la política del mismo dominio coincidiera con aquella en donde estuviera montada y funcional el dashboard para control de Leishmaniasis. Esta configuración, una vez utilizada, estará presente en todos los encabezados en donde el servidor responda una consulta de algún cliente. Si el navegador del cliente no detecta que la respuesta viene del servidor del mismo origen, el navegador bloqueará los datos de la consulta.

a. Programación de modelos dentro del sistema

Con el uso de la librería Sequelize se programaron los modelos existentes de la base de datos ya creada. Esto también implica que, tanto el modelo programado como la tabla de base de datos deben tener nombres de tablas, propiedades y tipos de datos idénticos, lo cual proveerá una abstracción completa de la base de datos dentro del backend y facilitará tanto las transacciones como la interacción con los datos. En resumen, cada modelo representa una tabla en la base de datos.

b. Programación de controladores para cada modelo

Para cada modelo existente en la base de datos, se creó una ruta para cada una de las acciones de este. Esto siempre tomando en cuenta los estándares REST para un API bien definido. Entre las acciones programadas, se encuentran obtener (GET), crear (POST), modificar (PUT) y eliminar (DELETE) cada entidad. Para cada acción, se consideró enviar un estatus HTTP adecuado, es decir, respectivo a cada acción:

- GET retorna un código 200 (OK), la cual posee la información de la entidad consultada por el servidor.
- POST retorna un código 201 (Created), el cual indica que la entidad fue creada con éxito. También se adjunta la entidad en el cuerpo del mensaje.
- PUT retorna un código 204 (No Content), lo que indica que la solicitud ha sido ejecutada con éxito, y no hay contenido adicional que enviar.

- DELETE retorna un código 204 (No Content).

También se consideró que la ruta para el manejo de las entidades es importante. Esta debe ser fácil de entender y sugestiva para el desarrollador. Por ejemplo, en el caso que se quiera crear un diagnóstico, se programó la siguiente ruta: POST: http://<<url-servidor>>/api/diagnostico. El API espera que, dentro del cuerpo de la solicitud, el usuario envíe los parámetros necesarios para crear la entidad en base de datos y así retornar un código 201.

En el caso que se quiera obtener un diagnóstico, se programó la ruta: GET: http://<<url-servidor>>/api/diagnostico/<id>. Esto le indica al servidor que el usuario está solicitando la entidad Diagnóstico con el identificador especificado en la ruta.

c. Manejo y control de errores

Es importante considerar que el usuario puede solicitar una entidad con id inexistente, o bien intentar insertar una entidad en base de datos con un parámetro inadecuado. Por ello, se deben controlar este tipo de errores. Se implementó múltiples condiciones dependiendo de la consulta que el usuario desea ejecutar, con el fin de controlar la mayor cantidad de errores posibles y así regresar una respuesta con los detalles de lo que falló. Para esta parte, también es importante mantener los estándares de status HTTP los cuales algunos de ellos son:

- **Status 400:** solicitud errónea. También considerado como un error del cliente, en donde puede haber un error de sintaxis de petición malformada, solicitud inválida, entre otros.
- **Status 404:** no existente. Se programó para los casos en donde el usuario hace referencia a un elemento que no existe. Por ejemplo, al intentar obtener una entidad con un id que no existe sobre la base de datos.
- **Status 500:** error interno del servidor. Este error puede darse por una excepción causada en tiempo de ejecución dentro del servidor. Esto puede darse por una condición inesperada que impidió que se terminara la solicitud del usuario. Se devuelve cuando ningún otro código de error simboliza qué salió mal.

En todas las ocasiones de error, se devuelve un objeto tipo JSON el cuál posee propiedades descriptivas de qué fue lo que salió mal. Esto con el fin que el frontend pueda mostrarle al cliente de forma técnica qué sucedió y por qué su solicitud no pudo ser completada. Esto está representado de la siguiente forma:

```
{
    msg: string,
    details: string,
}
```

d. Pruebas iniciales del sistema

Llegado este punto del desarrollo del REST API, es prudente realizar una o varias pruebas sobre los controladores creados. Para ello, se utilizó la aplicación Postman que permite realizar consultas hacia un URL específico, con datos u propiedades programables para así crear un payload totalmente configurado. Además, Postman proporciona de manera muy visual los tiempos de respuesta, encabezados recibidos del servidor, código de status, entre otros. Se realizó pruebas solamente sobre las funcionalidades básicas de una entidad, como lo es el método GET y POST.

6. Pruebas de calidad

Es importante garantizar la calidad en cada servicio que ofrece el servidor, por ello también es importante programar una serie de pruebas las cuales tendrán que ser aprobadas cada vez que un cambio es publicado al gestor de versiones en Github.

a. Pruebas unitarias

Las pruebas unitarias fueron creadas utilizando la herramienta JEST, esta fue elegida puesto que es una librería ligera y fácil de integrar con JavaScript; posee también su servicio relacionado en TypeScript. También posee integrado un ejecutor de pruebas, así como una librería de aserción y simulación. Primero, se configuró un simulador de ORM que permite hacer consultas e inserciones a una base de datos de forma programada, siempre bajo una base de datos falsa que sirve como simulador. Luego, se programó una serie de pruebas para cada controlador. El fin de esto es tener una cobertura promedio mínima del 70%.

b. Pruebas de integración continua

Otro de los comportamientos esperados, es saber que cada desarrollador está haciendo un trabajo que se integra adecuadamente a lo previamente trabajado. Por ello, se configuró un servicio de integración continua en el gestor de versiones llamado Travis CI. Se eligió esta herramienta puesto que es un proyecto de código abierto, gratuito, flexible y se aloja en Github.

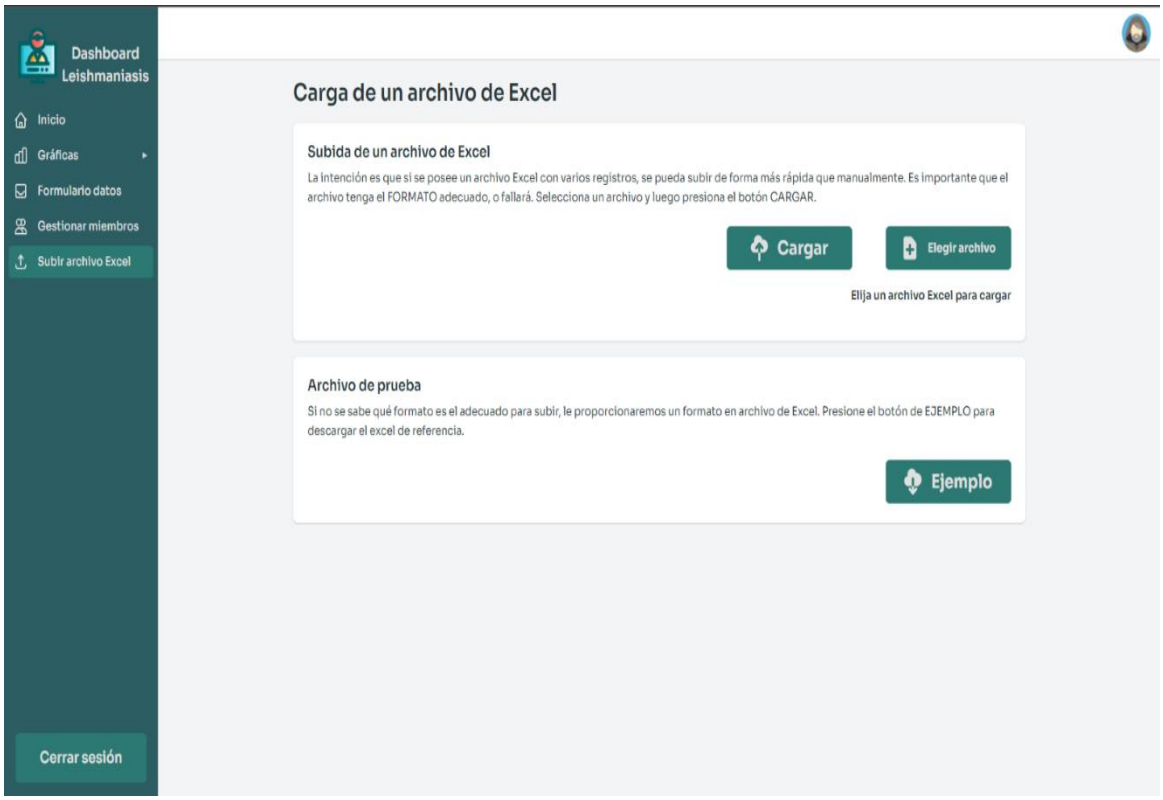
7. Servicio para la inserción de formularios

Uno de los servicios también consiste en que el usuario pueda actualizar el portal web (dashboard) mediante un archivo de Excel en donde contenga datos o registros de la enfermedad reportada. El fin de este servicio es poder alimentar el portal con datos que ya se poseen, pero que no se les está dando ningún eso; esto permite el estudio de estos.

a. Plantilla del documento

Se creó un archivo Excel sin datos, pero con el encabezado adecuado para que los técnicos de salud sepan qué datos ingresar en qué columnas. También se programó cada celda con el fin de solo poder ingresar datos preestablecidos, evitando así realizar una gran cantidad de validaciones por parte del servidor. Este documento también está disponible para su descarga mediante una consulta de tipo GET, la cual el dashboard provee de forma fácil.

Ilustración 18: Sección del portal web para carga de documento con datos.



Como se puede observar en la Ilustración anterior, el portal posee una opción para descargar la plantilla previamente creada. Esto garantizará que el usuario siempre tenga esta plantilla a la mano en caso de que quiera migrar sus datos de un documento a otro, o bien empezar a llenar datos de nuevo.

b. Configuración del servicio

Al igual que el resto de los servicios del *backend*, se creó un *endpoint* para que el servidor pudiese recibir un archivo Excel y así procesarlo. Se utilizó la librería xlsx, esta fue elegida puesto que permite una conversión sencilla de cada fila dentro del documento a un objeto JSON previamente programado, con el fin de poder acceder a las propiedades fácilmente y validar si cada fila posee todas las que son necesarias.

c. Inserción de datos en la base de datos

Una vez hecha la conversión del documento a un objeto JSON, se iteró por cada set de datos ingresado. Se utilizaron los datos necesarios para crear cada una de las entidades necesarias en la base de datos. Dado que algunas entidades dependen del identificador de otras, se programó un proceso secuencial por cada set de datos; no obstante, cada set de datos fue iterado asíncronamente para acelerar el proceso de inserción en base de datos. El resultado de esto se puede interpretar como una lectura paralela de datos, en donde cada fila del documento Excel se lee al mismo tiempo.

D. Infraestructura

1. Planeación general previa

Algunos requerimientos no técnicos:

- El tipo de información almacenada es información impersonal, cuantitativa y cualitativa.
- Se acordó que el sistema manejaría un número aproximado de diez usuarios.

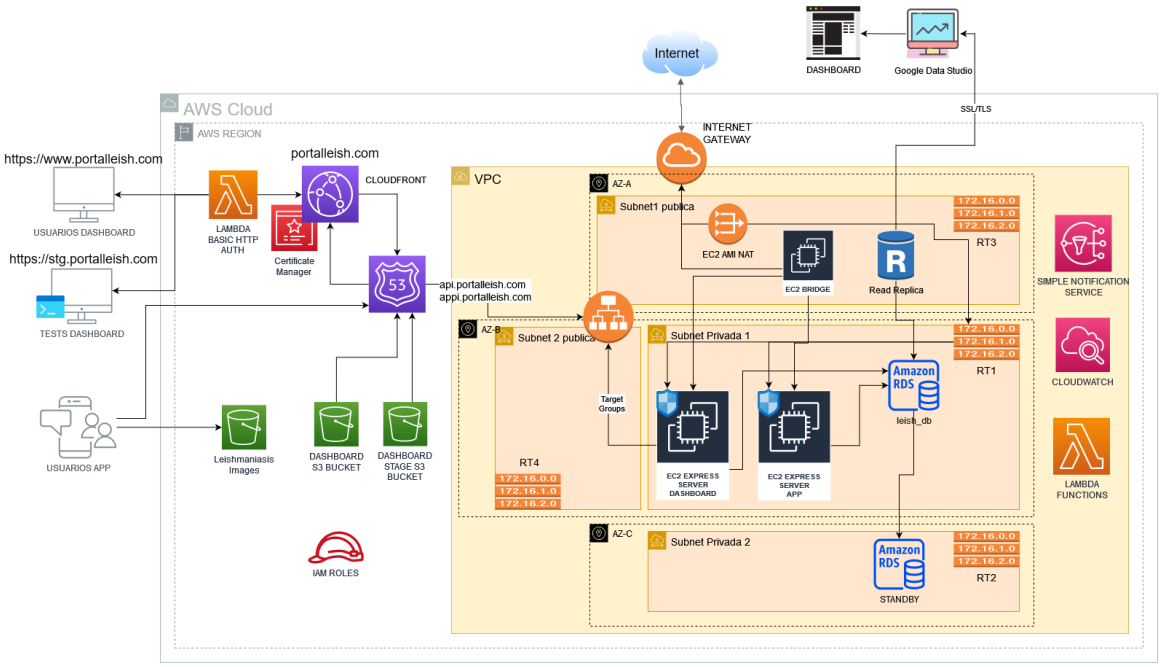
A continuación, un breve resumen de algunos pasos generales tomados en cuenta al principio del desarrollo:

- Creación de cuenta de AWS dedicada al proyecto tomando en cuenta los siguientes aspectos:
    - Presupuesto
    - Alternativas
    - Posibles inconvenientes que puedan presentarse con el uso de ciertas tecnologías como Node.js.
    - Otras alternativas
    - Inconvenientes con otras tecnologías relacionadas.
    - Presupuesto de creación de red virtual privada.
    - Prototipo de prueba en servicios de AWS.
    - Prototipos funcionales.
    - Gestión de puntos del sistema.
- Creación de red virtual privada y subredes para gestionar servidores y conexión a bases de datos.
- Prototipos de prueba funcionales con servicios AWS para un acercamiento a la implementación en producción. Estarán estrechamente relacionados a los modelos de Backend a utilizar.
- Evaluación del hosting del portal web digital a desarrollar en el transcurso del proyecto. Esto involucra la configuración de subdominios para distintos ambientes de pruebas, certificados seguros, entre otros.
- Gestionar varios puntos importantes del sistema a implementar como los niveles de permisos de usuarios, la seguridad de la información, entre otros.
- Prototipos funcionales en el ambiente de producción.

La aplicación diseñada es de tipo Sin Estado (Stateless Application), debido al uso pensado que le darían los usuarios al sistema en el ambiente de producción. Una aplicación sin estado es aquella que no guarda conocimiento o referencias relacionadas a procesos o transacciones pasadas. Cada transacción realizada por un usuario se realiza desde cero, por así decirlo, sin necesidad de conocer datos anteriormente ingresados por el usuario en cuestión. Para los casos de uso del sistema en desarrollo, como consultas de información en general, no es necesario almacenar los estados anteriores de la aplicación como se haría en una aplicación con estado (Stateful Application) que mantendría seguimiento de las preferencias y configuraciones hechas por el usuario del sistema. (Red Hat Inc., 2021)

2. Arquitectura general de infraestructura para aplicación

Ilustración 19: Arquitectura general AWS para aplicación.



En la Ilustración anterior pueden observarse varios servicios de AWS relacionados. La mayoría de los servicios están resguardados en una red virtual privada, con varias subredes públicas y privadas para los diferentes requerimientos de cada servicio. Este diagrama representa la vista general de la infraestructura del sistema que se llevó a cabo durante todo el proceso de desarrollo del proyecto.

a. Descripción de servicios y partes de arquitectura general

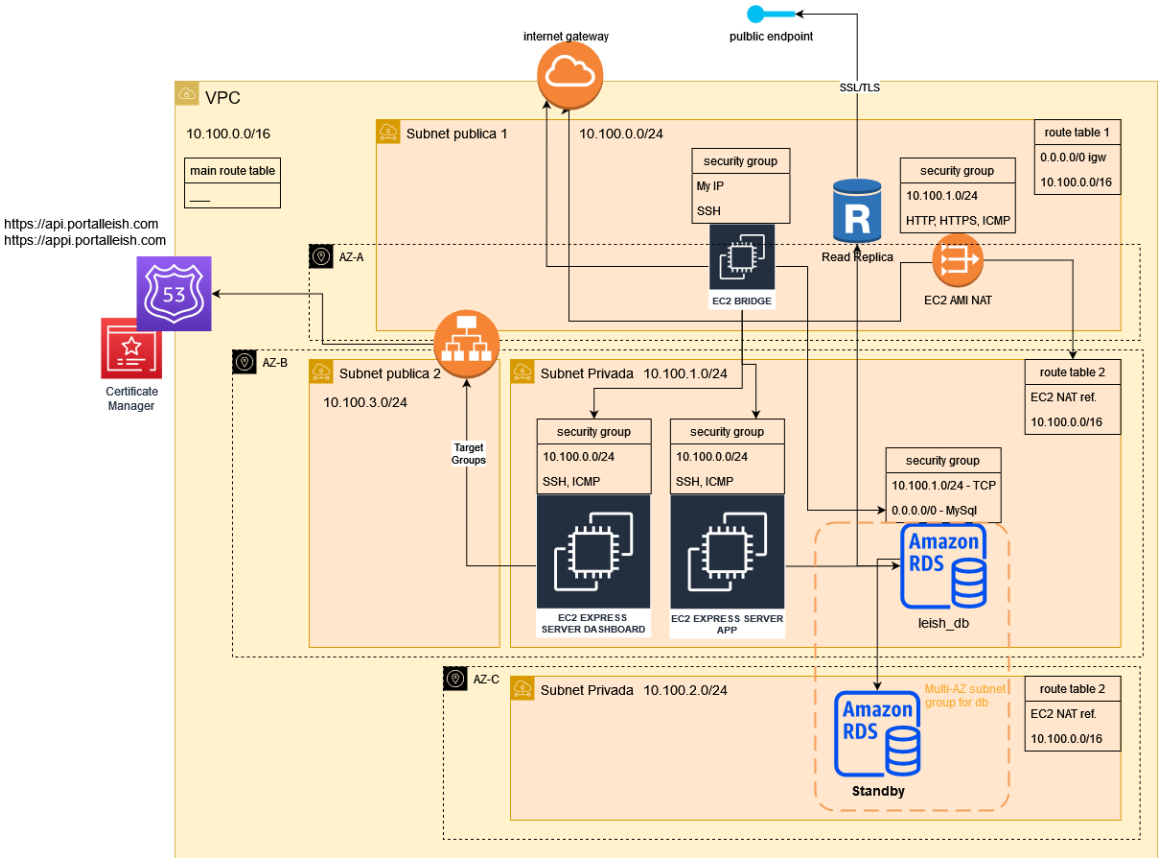
- **La subred pública no. 1** contiene una instancia o servidor EC2 que fue configurada como una NAT Gateway. Una NAT Gateway proporciona acceso de internet a recursos y servicios privados que no tienen IPs públicas, sin exponer dichos servicios a las conexiones de internet entrantes.
- **La subred pública no.** 2 fue creada únicamente por requerimiento de 2 subredes públicas en la implementación del balanceador de carga que comparte las subredes públicas 1 y 2.
- **En la subred privada no. 1** existen 2 servidores de aplicación representados por instancias EC2.
    - El primer servidor contiene el Backend del Portal Web Estadístico.
    - El segundo servidor contiene el Backend de la Aplicación con el agente inteligente para el reconocimiento de imágenes.
    - Ambos servidores estan expuestos de forma segura al público por medio un Balanceador de Carga de Aplicación implementado en las subredes públicas 1 y 2. Este ALB está expuesto con el subdominio api.portalleish.com

proporcionado por Route 53. El proceso fue realizado de forma segura asociando un certificado SSL/TLS a cada uno, gestionados por ACM.
    - El portal web en S3, puede acceder a la información de la base de datos por medio de un sistema de manejo de usuarios. Por otro lado, la aplicación, accede a la información de la base de datos por medio de autenticación básica HTTP. Ambas consideraciones de seguridad están implementadas en el respectivo Backend asociado.
- En la subred privada no. 1 también se resguarda la base de datos principal en RDS qué contiene la información relacionada al portal web y la aplicación.
- En la subred privada no. 2 se resguardará la base de datos de respaldo para la base de datos principal.
- Las imágenes recolectadas desde la aplicación son almacenadas en AWS S3 (Simple Storage Service). Este servicio es asequible y no representa un costo significativo en el proyecto. Este recopilatorio representa un histórico de imágenes, que puede ser de utilidad para futuras investigaciones.
- El portal web estará compuesto de varias partes. Las cuales son:
    - S3 para el hospedaje del sitio web.
    - Route 53 para la adquisición de un dominio y manejo de subdominios.
    - ACM para certificados SSL/TLS.
    - Cloudfront para distribuciones, redireccionamientos, autenticación básica con Lambda Edge y relaciones entre servicios como S3 y Route53.
    - Se cuenta con un ambiente para pruebas (staging) y otro para producción.
        - El primero, contiene la versión más nueva/estable de la aplicación y puede ser accedido por los usuarios del portal. Subdominio: www.portalleish.com.
        - El segundo, representa una copia del ambiente de producción y solo puede ser accedido por los desarrolladores. Funciona para realizar pruebas o reparaciones que deban realizarse al portal. Una vez probada la funcionalidad se procede a pasar estos cambios a producción actualizando el bucket de producción y limpiando cache en la distribución de cloudfront. Subdominio: stg.portalleish.com.
- Se configuraron varias alarmas que alertan de posibles disparos en el uso del CPU o la Memoria RAM en los servidores y en la base de datos, así como otros problemas varios qué pueden presentarse en producción. Para este tema se utilizó CloudWatch y SMS.
    - Amazon CloudWatch es un sistema de monitoreo y métricas desarrollado para el mantenimiento y seguimiento de los servicios en uso.
    - Amazon SMS es un servicio simple de notificaciones desarrollado para el envío de mensajes entre aplicaciones e interesados.
- Las buenas prácticas dictan que toda arquitectura desarrollada en AWS debe de estar distribuida en distintas Zonas de Disponibilidad (Availability Zones, AZ) por lo cual la arquitectura descrita anteriormente está distribuida en 3 AZ distintas.
- Un punto que se consideró luego de la implementación fue la calendarización de funcionamiento, principalmente, de los servidores de aplicación. Realizar esta tarea fue posible por medio de la configuración de funciones lambda, para reducir los costos netos mensuales y anuales en el uso de EC2. Cabe mencionar que este

proceso también podría haberse realizado con la base de datos principal en RDS, sin embargo, al estar asociada con una base de datos de solo lectura (la réplica) AWS no permite detener o apagar las instancias. Esto imposibilita la calendarización de funcionamiento para RDS en el sistema y de esa manera reducir un poco los costos netos relacionados.

3. Arquitectura de redes y comunicaciones internas en red virtual privada AWS

Ilustración 20: Arquitectura general AWS para aplicación.



Como se muestran en la Ilustración 31, se definieron 4 subredes, 2 públicas y 2 privadas. El enrutamiento entre dominios sin clases (CIDR - Classless Inter-Domain Routing) permite gran flexibilidad en la división de rangos de direcciones IP para las subredes administradas.

Ilustración 21: Subredes públicas y privadas en consola de VPC.

a. Configuración de rangos ip para la arquitectura

Como se observa en la Ilustración 32 se crearon 4 subredes con bloques de 256 IPs disponibles para utilizar. De forma práctica, se cuenta con 254 IPs disponibles sin tomar en cuenta la primera y última IP, utilizadas para la dirección de Gateway y Broadcast.

- Rango de VPC (vpc-central-leish): 10.100.0.0/16 (65536 IPs disponibles)
- Subred pública no. 1 (leish-public-1): 10.100.0.0/24
- Subred pública no. 2 (leish-public-2): 10.100.3.0/24
- Subred privada no. 1 (leish-private-1): 10.100.1.0/24
- Subred privada no. 2 (leish-private-2): 10.100.2.0/24

Cabe mencionar que la mayoría de estas direcciones IP no son utilizadas, solamente se utilizan las necesarias.

4. Hospedaje de portal web estadístico

Para que los usuarios del portal web lograran interactuar con el sistema era importante que accedieran a él de alguna forma fácil y conveniente. Como el modelo de desarrollo utilizado para el proyecto es el de Modelo Vista Controlador, por naturaleza, el Backend está separado del FRONTEND, lo cual permite libertad al momento de decidir en donde hospedar el sitio web como tal.

Por este motivo se tomó la decisión de utilizar S3 configurado como un sitio web estático para hospedar la aplicación web. Luego de cargar el sitio web a S3, se procedió a hacer la configuración respectiva en CloudFront para redireccionamientos seguros con certificados SSL/TLS, manejo de cache del contenido, asociación con Route 53, entre otros.

Ilustración 22: Distribución de CloudFront para sitio web en S3.

Como se observa en la Ilustración anterior, la distribución de CloudFront tiene como origen el bucket donde se hospeda el sitio web y solo permite accesos a su contenido a través de HTTPS, usando un certificado seguro gestionado por ACM.

Ilustración 23: Política de seguridad relacionada a CloudFront en S3.

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned

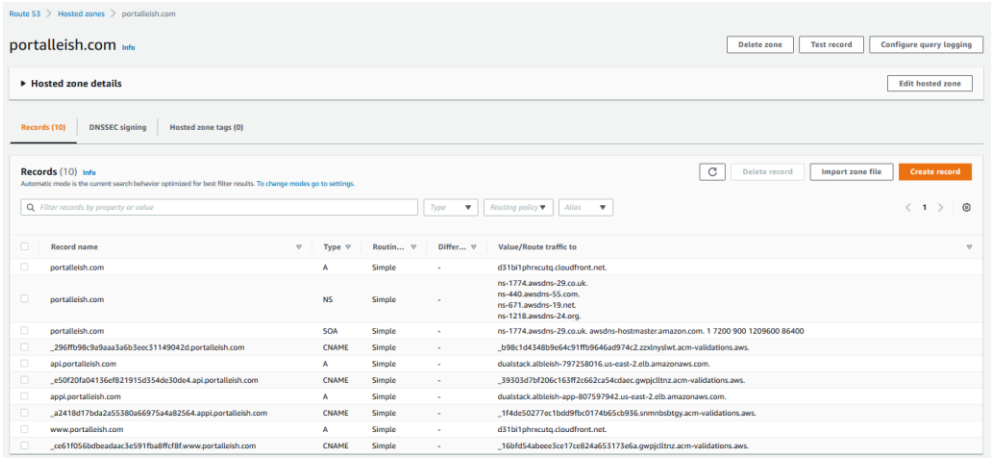
```
{
  "Version": "2008-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity EM41XZZMHM60G"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::portalleish-1/*"
    }
  ]
}
```

En la Ilustración anterior se observa la política de seguridad necesaria para que CloudFront pueda hacer uso del contenido en el bucket y distribuirlo. Cabe mencionar que, por buenas prácticas, CloudFront se configuró para restringir cualquier acceso al endpoint del bucket de forma directa.

a. Relaciones entre cloudfront, acm y route53

Se adquirió el dominio portalleish.com para reemplazar el endpoint de la distribución de CloudFront en Route53.

Ilustración 24: Records en route 53 - Hosted Zone (portalleish.com).



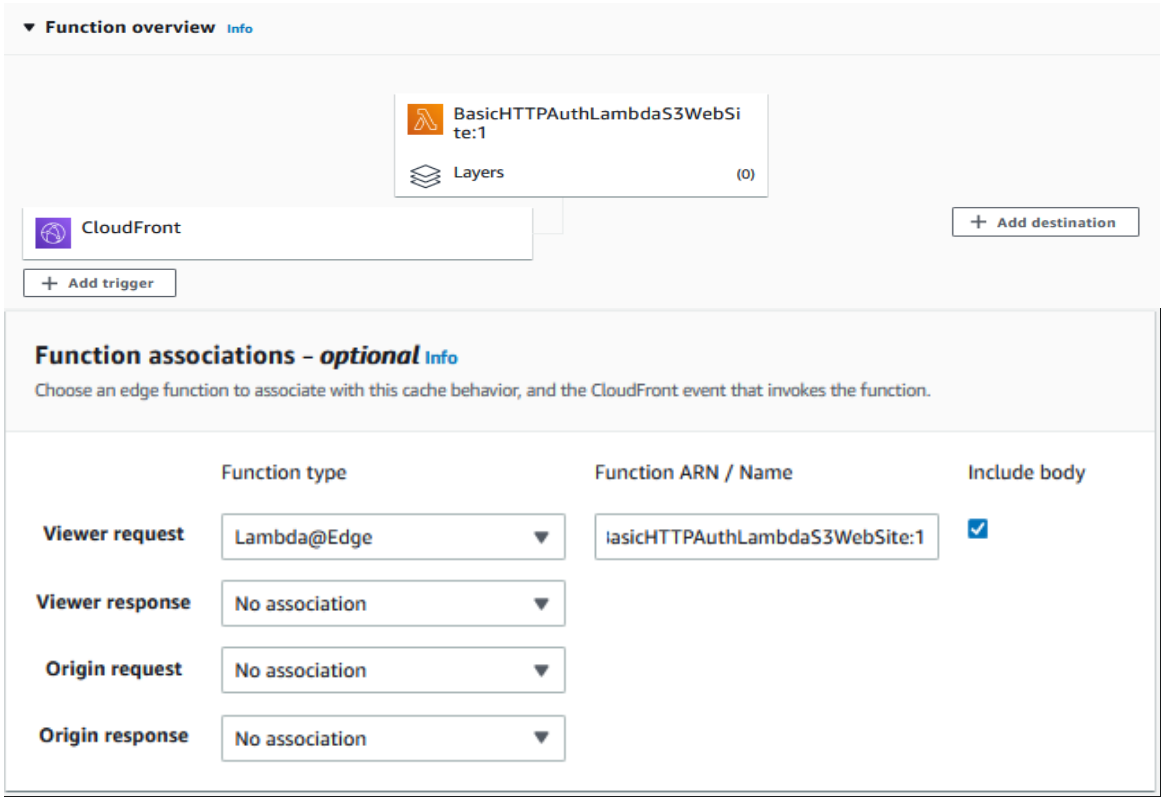
En la Ilustración anterior pueden observarse varios registros, cada uno con diferentes objetivos.

- Los primeros dos registros de tipo NS y SOA son creados de forma automática por Certifícate Manager al hacer la asociación con el dominio que se quiere asegurar.
- Los registros de tipo CNAME apuntan a los identificadores de los certificados gestionados en ACM.
- Los registros de tipo A hacen referencia al subdominio que se quiere asociar con los servicios AWS en cuestión. Por ejemplo, portalleish.com y www.portalleish.com apuntan ambos a la distribución de CloudFront que maneja el contenido del bucket.
  Los registros api.portalleish.com y appi.portalleish.com, fueron creados para apuntar a los balanceadores de carga que exponen los servidores privados con el Backend de la aplicación y el Portal Web. Solo se utilizó el primero con un balanceador de carga.

b. Autenticación http básica para portal web con lambda@edge

El portal web estadístico no debía estar expuesto a ningún contacto con el público general o cualquier tipo de indexación realizada por entidades como Google para sus búsquedas. Por tal motivo, se decidió agregar una capa extra de seguridad al portal web representada por una función lambda con trigger en cada consulta realizada a la distribución de CloudFront que tenía de origen el bucket de S3 con el portal web.

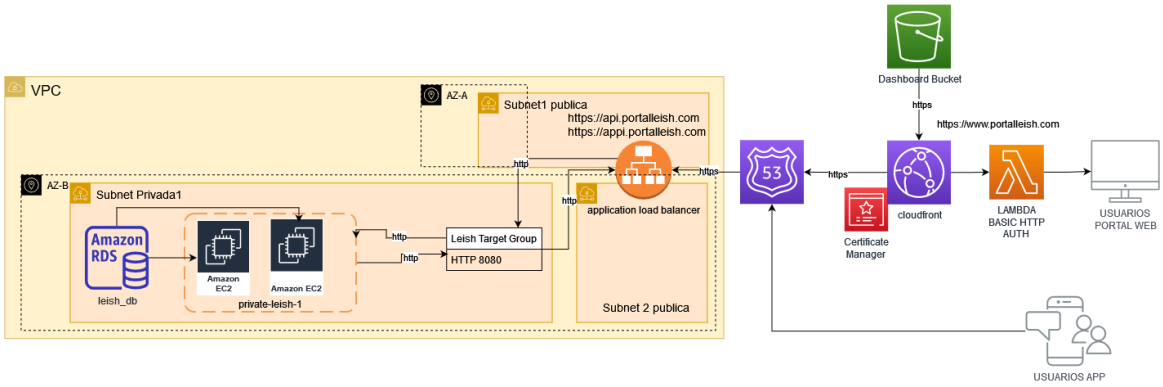
Ilustración 25: Asociación de Lambda y CloudFront para autenticación HTTP básica.



En la Ilustración anterior se observa la relación creada entre una función lambda y la distribución de CloudFront usada para acceder al contenido del bucket donde se hospeda el portal web. Esta función está escrita en JavaScript para el manejo de autenticación HTTP y es accionada cada vez que se realiza una consulta al sitio web (se coloca en el explorador https://www.portalleish.com). Si esta persona no conoce la contraseña y usuario para acceder, se le responderá con un código 401 no logrando acceder al contenido.

5. Exposición de servidores privados con backend de aplicación y portal web

Ilustración 26: Exposición de servidores privados - REST API.



El Backend de una aplicación es la parte del sistema que se encarga de conectar el FRONTEND y la infraestructura que yace por detrás, esto por medio de la configuración de protocolos y buenas prácticas para la seguridad de la información.

Como se muestra en la Ilustración 40 para esta implementación se utilizaron dos Balanceadores de Carga de Aplicación (Aplication Load Balancer) que utilizan el protocolo HTTP para exponer servicios contenidos en subredes privadas. Esta funcionalidad se logra por medio de la configuración de Grupos de Objetivo (Target Groups) que apuntan hacia los servicios privados que quieren exponerse de forma pública.

Los objetivos del target group utilizado son los servidores que contienen el Backend del portal web y la aplicación. Estos objetivos pueden tener varios estados entre los cuales están: healthy, unhealthy, unused, entre otros. Estos estados dependen de que existan servicios corriendo o ejecutándose en los puertos especificados, como Express que corre en el puerto 8080 estableciendo conexión con la base de datos en RDS. Una vez expuestos los servicios, se configuraron dos subdominios, api.portalleish.com y appi.portalleish.com, a través de Route 53 para el balanceador de carga. Solo se utilizó el primero.

Cabe mencionar que, se decidió usar dos balanceadores de carga para separar los servicios de ambos Backend. Esto debido a que algunas rutas como el inicio de sesión presentaban problemas cuando ambos servidores se encontraban en funcionamiento debido a algunos métodos HTTP como OPTIONS y POST.

Ilustración 27: Grupo de objetivo de servidores de aplicación para ALB.

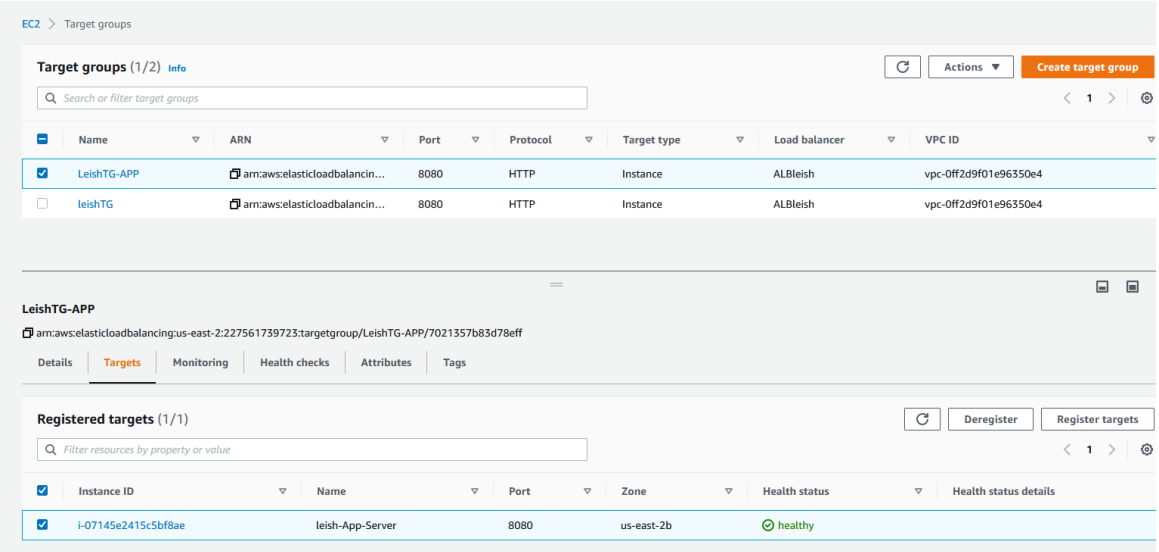
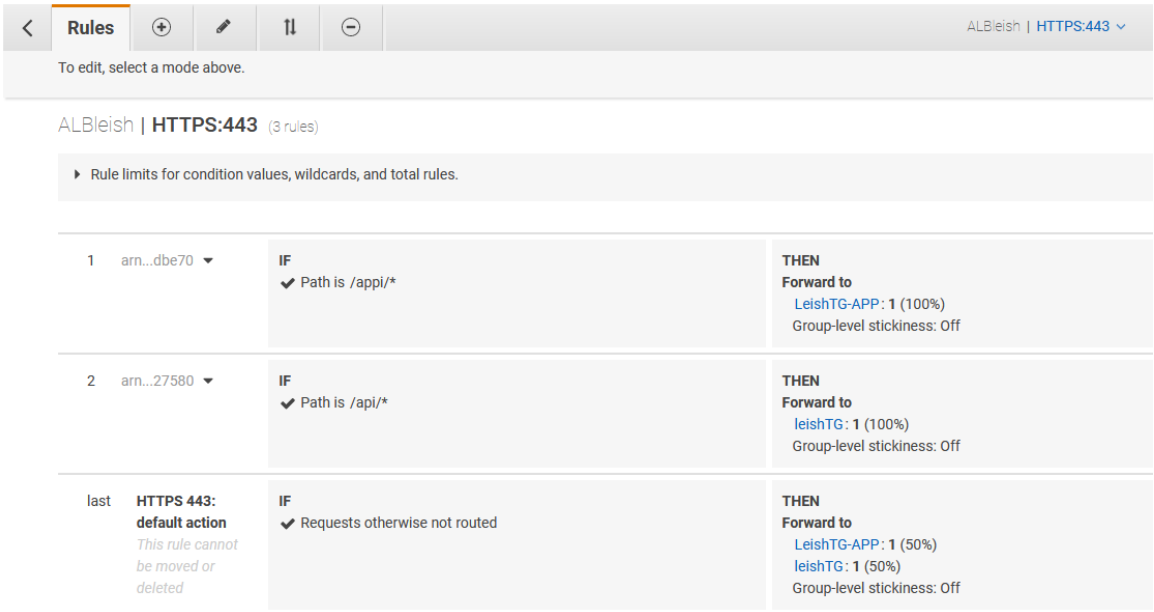


Ilustración 28: Reglas de redireccionamiento de ALB.



Como se muestra en las Ilustraciones anteriores los grupos de objetivo creados apuntan cada uno a un servidor de aplicación, y están asociados al balanceador de carga (ALBleish) con reglas para un redireccionamiento correcto. Cabe mencionar que los health checks estan configurados para realizarse cada 30 segundos y el estado de los registros únicamente cambiará a saludable luego de 5 health checks consecutivos exitosos.

El balanceador de carga cuenta con un certificado SSL/TLS generado en ACM para la circulación de tráfico HTTPS entrante. Es importante destacar que el balanceador de carga solo permite consultas realizadas con HTTPS gracias a la configuración de seguridad. Luego de que el tráfico pasa por este punto se vuelve HTTP dentro de la VPC, ya que el manejo de certificados de forma interna es innecesario para este caso de uso. De esta forma a través de https://api.portalleish.com es posible acceder a las rutas configuradas en ambos Backend para obtener o ingresar información de acuerdo con las reglas especificadas. La configuración del subdominio para ambos endpoint es manejada por Route 53.

a. Autenticación http básica en express

La autenticación HTTP básica es una capa de seguridad que oculta cualquier servicio o portal de intern*et al* público general. Al igual que en el portal web, se decidió hacer la respectiva implementación de esta capa de seguridad para evitar cualquier contacto con el sistema y, además, evitar indexaciones de búsqueda.

Existe variedad de paquetes de instalación que simplifican la implementación de esta capa de seguridad en Node.js con Express. Para este caso se utilizó la versión 1.2.0 del paquete “express-basic-auth” de npm.

La configuración de este paquete se realizó en ambos Backend, aplicación y portal web. De este modo, siempre que se haga alguna solicitud de consulta o inserción a las API implementadas, será necesario incluir un header de Authorization con el usuario y la contraseña definidos en el Backend de cada servidor.

b. Manejo de usuarios y jwt en express

JWT (JSON Web Token) consiste en una forma de comprobar la identidad de los usuarios que interactúan y consultan información a un servidor, de forma general.

Existen varios paquetes relacionados a la generación y firma de tokens con este flujo, que facilitan su implementación en un ambiente de desarrollo como Express. Se utilizó la versión 8.5.1 del paquete “jsonwebtoken” de npm.

Esta implementación fue realizada únicamente en el Backend del portal web, ya que era necesaria una administración de usuarios con ciertos permisos para acceder a ciertas partes con distintas funcionalidades del portal. En la aplicación no fue necesario, ya que no existe un manejo de usuarios como tal, cualquier usuario puede utilizar todas las funcionalidades de la aplicación sin restricción alguna.

La implementación consistía en la siguiente estructura:

auth
- signJWT
- verifyJWT
- verifyLogin
- verifyRegister

controllers
- auth.controller

El directorio *auth* contiene varios archivos importantes:

- **signJWT**: este archivo contiene una función donde se firma la consulta a enviar, se selecciona el algoritmo de encriptación y se hacen otras configuraciones relacionadas al proceso digital de autorización JWT.
- **verifyJWT**: este archivo contiene una función que realiza la verificación de la firma. La firma contenida en la consulta entrante debe ser válida de acuerdo con la llave secreta única del servidor, en este caso del Backend del portal web.
- **verifyLogin:** contiene dos funciones para verificación de existencia de correo en la base de datos y validación de la consulta proveniente.
- **verifyRegister:** son funciones que verifican que los datos provenientes de un usuario sean correctos y así ingresen con un formato establecido a la base de datos en caso no hayan entrado con errores luego de la verificación en FRONTEND. Por ejemplo: el nombre de usuario, el correo, los roles que tendrá ese usuario, entre otros.

De esta forma, cada ruta del API que quiere ser resguardada con ciertos permisos de usuario tiene de por medio las funciones que validan el token o los permisos del usuario que realiza la petición en cuestión.

Los roles creados para usuarios son los siguientes:

- **Administrador**: el usuario administrador tiene acceso a todas las funcionalidades que puede proporcionar el sistema. Puede registrar, modificar y borrar usuarios en la base de datos. Puede visualizar ciertas pantallas solo para administradores y más. Esto del lado del FRONTEND, en cuanto al Backend, siendo breve, un administrador tendrá acceso a cualquier ruta que este estipulada en el FRONTEND para acceso o inserción de información.
- **Usuario**: posee la menor cantidad de permisos definidas.

El sistema se desarrolló para manejar tres roles de usuario en donde también se incluía un moderador que hubiera poseído más permisos que un usuario normal, pero menos privilegios que un administrador. Sin embargo, el caso de uso del portal web no ameritó el uso de este tercer rol por lo cual simplemente quedo descartado del lado del FRONTEND.

En el FRONTEND se adjunta el header x-access-token a la consulta para enviar el token firmado del usuario. No se utilizó el header Authorization como sucede de forma convencional, ya que este fue usado para la autenticación HTTP básica.

c. Módulo de logs con winston

Los registros de información de un sistema son una parte esencial para comprobar el funcionamiento correcto del mismo en el tiempo. En un ambiente local se suele probar el sistema por un periodo corto de tiempo en donde se pueden observar impresiones en consola que informan sobre el estado del sistema. Es importante tomar en cuenta que en un ambiente de producción el sistema estará ejecutando sus funciones de forma continua y, en

el caso ideal, no debería haber un técnico encargado de observar que sucede con el sistema durante todo el transcurso del día.

Por esta razón es de suma importancia tener un registro de lo que sucede con el sistema en el tiempo. De esta manera, al momento de presentarse un inconveniente o algún error será posible hacer un rastreo de este de forma efectiva basándose en los registros en el momento de ocurrencia del error.

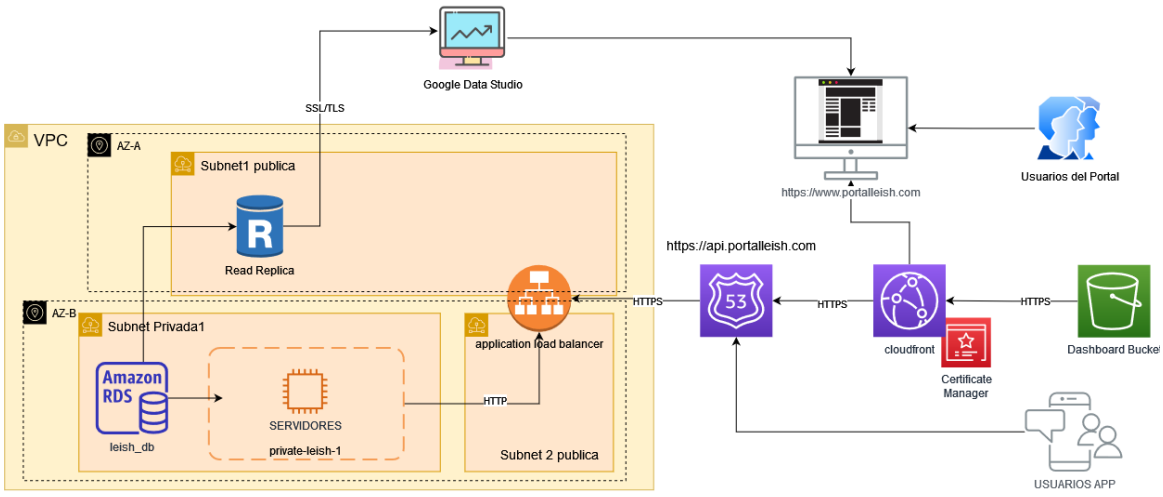
Winston permite hacer una separación de logs por categoría, entre ellos estan info, error, debug y warn.

Los registros de logs del sistema se planearon de la siguiente forma:

- **info**: aquí se almacena cualquier informacion útil del sistema, como un registro exitoso, una nueva sesión de usuario, una impresión en consola personalizada para alguna consulta, entre varios otros.
- **debug**: aquí se almacena la información de registros a la base de datos. Está relacionado a Sequelize, la herramienta de conexión a la base de datos de MySQL en Node.js.
- **error**: aquí se almacenan errores de Sequelize, Express, Node.js y en general cualquier error que pueda existir en la ejecución del programa.

d. Interacción con Google Data Studio

Ilustración 29: Interacción con Google Data Studio para visualización de gráficos en portal web.



En la ilustración anterior puede observarse la creación de una réplica o copia (de solo lectura) de la base de datos principal en la red pública no. 1 para poder establecer conexión con MySQL de Google Data Studio. Google Data Studio cuenta con distintas herramientas para generar gráficas basadas en las tablas existentes en la base de datos y por esta razón se decidió utilizar dicha herramienta.

Un punto importante que es necesario mencionar es que el tráfico entre la base de datos de réplica y Google no pudo ser cifrado con certificados SSL/TLS. El problema fue que RDS, como tal, solo genera un certificado (Self-Signed Certificate) mientras que para la conexión a Google se solicita un certificado de cliente, servidor y la llave privada. Investigando se encontró una posible solución para lograr la conexión encriptada entre las dos entidades, pero estaba relacionada a PostgreSQL.

Cambiar de motor de base de datos significaría un conjunto de varios cambios como crear una nueva base de datos con PostgreSQL, hacer cambios al modelo en ambos Backend, modificar la configuración de conexiones en la infraestructura, entre otros.

Se generaron certificados para cliente, servidor y una llave privada desde Linux, utilizando OpenSSL especificando varios datos relacionados a la localidad y organización con el propósito de utilizar dichos certificados en la conexión. Los pasos para la configuración de conexiones encriptadas en MySQL están estipulados en su documentación. Sin embargo, reiterando, RDS es administrado por AWS y no es posible modificar el Server-Side Startup para Conexiones Encriptadas. Esta configuración es preestablecida por AWS para utilizar únicamente un Self-Signed Certificate.

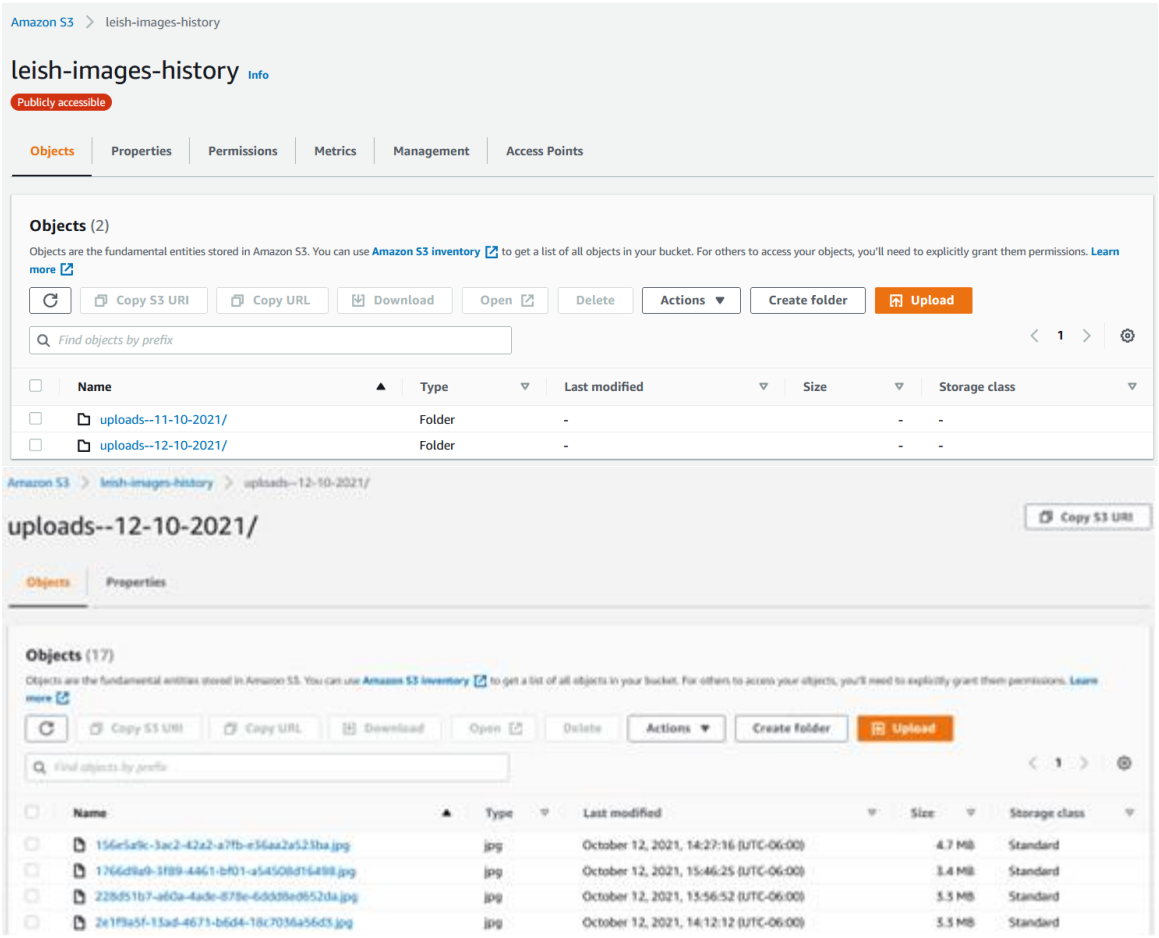
De una u otra forma, para acceder a la información de la base de datos siempre es necesario conocer un usuario y una contraseña creados internamente desde MySQL, con ciertos permisos para acceder a la información. También es necesario conocer el *hostname* de la base de datos, el cual es totalmente confidencial. Por lo dicho anteriormente es bastante improbable que un tercero pueda acceder a la información de la base de datos.

Es importante mencionar que se habla de una medida de seguridad extra para el intercambio de datos entre ambas entidades. No es un tema indispensable que comprometa la funcionalidad del sistema de alguna forma en particular.

6. Almacenamiento de imágenes en s3

Se realizó la configuración respectiva para que las imágenes tomadas con la aplicación fueran almacenadas en directorios de S3, y así crear un histórico con el tiempo. Cabe mencionar que el almacenamiento de las imágenes tomadas por los usuarios de la aplicación con el agente inteligente era un aspecto importante, pero no esencial.

Ilustración 30: Histórico en S3 de imágenes capturadas con la aplicación.



Cabe mencionar que el almacenamiento de las imágenes tomadas por los usuarios de la aplicación con el agente inteligente era un aspecto importante, pero no esencial. En la Ilustración anterior se observan dos directorios creados en fechas distintas. Cada directorio almacena las imágenes tomadas por los usuarios con la aplicación, un día específico.

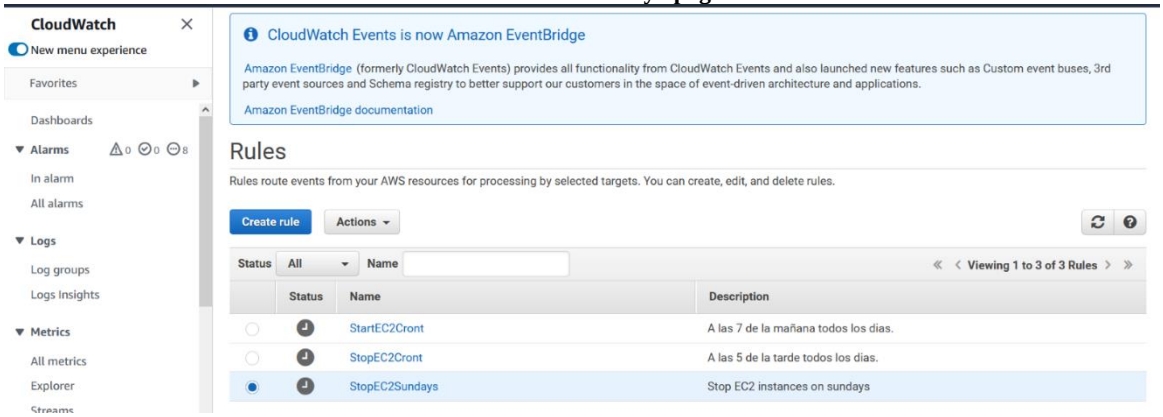
Cabe destacar que esta funcionalidad solo se configuró para la carga de formulario simple, no fue realizado para la carga en conglomerado. Esto para evitar complejidades de almacenamiento en el dispositivo del usuario.

7. Calendarizaciones y monitoreo

La calendarización de ciertos procesos en un sistema de producción es necesaria para mantener controlado el funcionamiento continuo de ciertas aplicaciones, además de posibles costos innecesarios y otros aspectos relacionados.

Desde un comienzo se realizó un presupuesto estimado para el costo de la aplicación ejecutando sus funciones en un ambiente de producción. Por tal motivo se decidió calendarizar el inicio y apagado de las instancias con los servidores de aplicación.

Ilustración 31: Calendarización de inicio y apagado de instancias EC2.



En la ilustración anterior se puede observar que cada instancia se inicia a las siete de la mañana y se apaga a las cinco de la tarde cada día. Además, los domingos hay otra tarea que apaga las instancias para no consumir recursos de presupuesto en un día inactivo para el sistema.

Cabe destacar que este proceso estaba pensado también para la base de datos de RDS, pero no fue posible realizarlo debido a la base de datos de réplica creada para la conexión con Google Data Studio. Esto debido a que AWS no permite detener instancias en RDS que tengan una réplica asociada.

E. Frontend

1. UX/UI de aplicación móvil y dashboard

Se contempló desarrollar dos soluciones como objetivo del proyecto:

- La creación de una aplicación móvil
- Un dashboard para información sobre Leishmaniasis cutánea.

Para alcanzar dichos objetivos se siguieron los pasos siguientes.

a. UX/UI

Para el Dashboard y la aplicación móvil aplican ciertos criterios y patrones de diseño. Aunque ambas al final sean de distintas plataformas o se utilicen en distintos lugares, compartirán similitudes en cuanto al UX/UI. Para elegir patrones, colores y formas, fue necesario un proceso de selección.

Primero, se realizó una etapa llamada *Pre-Design*, que permite conocer el proyecto o solución de mejor forma. Luego, se elaboró el *Look and Feel* para mostrar estados y estilos. En esta etapa se obtuvo como resultado un *Mood Board* y una *Pattern Library*.

El último paso es el *Slice 'n' Dice* que consiste en iterar todos los elementos construidos hasta ese momento, mediante prototipos o *mockups*. Eventualmente fueron realizadas dos iteraciones para lograr un contenido y una interfaz lo más orientada posible al usuario.

b. Pre-design

Una vez establecidas las convenciones (listadas en el *Pre-Design* del marco teórico) y conociendo que el contenido estará orientado al usuario, se elaboró un documento con el siguiente contenido:

- Listado de preguntas a responder.
- Convenciones de interacción con el usuario.
- Un breve resumen de qué es la aplicación, en dónde se alojará, a quién está dirigida y su propósito.
- Casos de uso

A continuación, surgió la necesidad de conocer los intereses y necesidades del usuario. Para esto, se realizaron dos encuestas (ver Anexo 1 y 2).

- La primera encuesta es una serie de preguntas orientadas a los trabajadores de salud, usuarios finales de la aplicación móvil.
- La segunda es una encuesta orientada a conocer a los usuarios del dashboard o de la aplicación móvil.

Para elaborarlas, se usó el software de pago *Typeform* (2021, USA) y se siguieron los siguientes pasos:

- Se elaboró una lista de preguntas relacionadas a las intenciones, gustos y objetivos de los usuarios (ver anexos).
- Una vez elaborada la lista de preguntas se usó el software de *Typeform* (2021, USA) y se programaron en su plataforma.
- Al estar terminada la encuesta, se publicó y se generó un enlace para compartir con los *Stakeholders*.
- Se realizó la prueba piloto de la encuesta con diez personas con un perfil similar a los usuarios finales del sistema y se ajustó la encuesta con base a sus comentarios y observaciones.
- El *link* de la encuesta ajustada se compartió con los trabajadores de salud, coordinadores de distrito y posibles usuarios del sistema, con la colaboración del encargado del Subprograma de Leishmaniasis del MSPAS. De un total de veinte colaboradores, solamente cinco contestaron.

Luego de realizar todos los pasos antes mencionados se empezó con la etapa de *Look and Feel.*

c. Look and feel

Etapa donde se realizó el Moodboard.

Se planeó basarse en distintos patrones de diseño y tomar ejemplos de otras aplicaciones relacionadas que pudieron haber sido útiles. Por ejemplo:

- Bancos
- Apps ampliamente utilizadas (Facebook, Whatsapp, etc.)
- Apps de formularios
- Aplicaciones de encuestas

Para desarrollar el *Moodboard* se hizo lo siguiente:

- Se eligió Frontify como herramienta web para crear tableros virtuales
- Se creó lo siguiente en el canvas del tablero:
    - Una fotografía de ejemplo de las posibles tipografías que tendría la aplicación. Es común que sea todo el abecedario, palabras aleatorias o lorem ipsum.
    - Una paleta de cuatro colores. Estos cuatro colores debían combinar. Para ello se usó una página web llamada https://colorhunt.co y también la Paleta de Adobe.
    - Imágenes relacionadas a los colores. Se tomaron imágenes que representan esos colores y se colocaron en forma de collage.

Este procedimiento se llevó a cabo con cuatro paletas de colores, para crear variedad y que los *Stakeholders* puedan elegir la que más se identifique con ellos.

Luego de realizar el *MoodBoard*, se realizó la transición de elementos formales pasivos hacia activos. Para lograr esto, se tomaron en cuenta las convenciones mencionadas en el marco teórico relacionadas al desarrollo de una interfaz: botones, composición y estructura, estados y cambios.

Para realizar la *Pattern Library* se siguieron los siguientes pasos.

- Se desarrollaron escenarios escritos que describen las acciones del usuario. Por ejemplo, para una aplicación que aplica filtro un escenario se describiría de la siguiente forma:
    - Abra la aplicación.
    - Presione el botón de abajo con el ícono de filtros.
    - Sostenga en alto el teléfono.
    - Escanee su rostro.
    - Aplique el filtro y luego guarde la foto con el botón de abajo a la derecha con el ícono de disco alterado.

Describir los pasos a seguir en la aplicación ayudó a crear una Pattern library tomando en cuenta todos los posibles caminos o estados que tendrá la aplicación. Todo eso siguiendo los escenarios escritos.

- Una vez definida la interacción, se usó un software de prototipado de íconos o de estilos *Adobe Illustrator* (Versión 2020, USA). Crear la *Pattern Library* fue un punto de apoyo ideal para el resto del diseño Luego, basados en iconos sencillos, se

compraron iconos de pago en la web *iconscout* que combinaran con la paleta de colores hecha y que estuvieran relacionados a medicina y salud.

- Luego, usando el tablero creado en *Frontify* se cargaron todos los iconos, tipografías, imágenes, colores, etc. Los recursos subidos a la web de *Frontify* estuvieron disponibles a los miembros del equipo de forma gratuita para su uso y consulta.

Una vez terminada, se obtuvo el conjunto de elementos de la interfaz de usuario necesarios para continuar con la siguiente fase.

d. Slince 'n' dice

El último paso fue agrupar todos los elementos que se crearon en la *Pattern library*, el *MoodBoard* y el resto de los escenarios y crear prototipos interactivos para que se pudiera obtener retroalimentación. Estos fueron creados con Adobe XD (Versión 2021, USA).

Los prototipos tanto de la aplicación móvil como del dashboard fueron realizados en esta plataforma y se solicitó la retroalimentación de los posibles usuarios o personas ajenas al proyecto para mejorar la solución.

Para realizar los prototipos, se realizó lo siguiente:

- Se elaboró la pantalla inicial o home.
- A partir de esa pantalla, se escogió un flujo o camino. El flujo puede ser llenar un formulario, tomar una fotografía, enviar un mensaje, etc.
- Se diagramaron los colores, tipografías e imágenes hechas en la *Pattern Library*.
- Se ajustó cada pantalla con elementos gráficos.
- Se prototipó para que pueda ser interactivo.
- Se realizó una prueba para probar su funcionamiento.

Se planeó iterar hasta dos veces los prototipos de cada solución. De esa forma se garantizaba que el producto estuviera orientado al usuario, fuera consistente y que supliera la necesidad de este.

Los prototipos fueron sometidos a revisión con los Stakeholders en la siguiente reunión que se tuvo. El proceso de toma de retroalimentación fue el siguiente:

- Se citó a una videollamada grupal, con miembros del equipo y los Stakeholders.
- Un total de cuatro Stakeholders asistieron a la toma de retroalimentación.
- Se les mostró los mockups interactivos hechos en Adobe XD, mostrándoles la paleta de colores, el flujo de las transiciones, la iconografía, los Moodboard, las sugerencias de funciones tanto de la aplicación como del dashboard de Leishmaniasis.
- Dieron sus comentarios, plantearon dudas y dejaron interrogantes.

Luego de ese proceso, se iteraron de nuevo, continuando con el proceso anterior. Cada solución (App y Dashboard de Leishmaniasis) fue sometido a dos iteraciones. Siempre variando cantidad de *Stakeholders* presentes. Debido a que es un proyecto relacionado con la salud, no siempre podían buscarse nuevos *Stakeholders* que supieran de la rama de salud de Leishmaniasis y que a la vez no hubieran visto ninguna solución. Por esa razón, debieron repetirse en ocasiones algunos *Stakeholders*.

2. Aplicación móvil

a. Primer prototipo y trabajo inicial

Primero, se maquetó cómo sería la interfaz. Luego se revisaron los mockups y prototipos hechos en Adobe XD (que ya habían sido iterados dos veces) para su posterior uso como guía en la creación de componentes. Se creó un repositorio en GitHub dedicado a esta solución. Para desarrollar en esta tecnología fue necesario un teléfono Android y una computadora. Fue necesario instalar dependencias, librerías y dar la base tanto de código como metodológica de React Native para trabajar sobre ella. Una vez terminada la base se inició a "clonar" los componentes de diseño a componentes orientados a programación. Iniciando con botones, fondos de pantalla, transiciones, navegación entre pantallas, tarjetas de uso, importación de librerías y otros.

Cuando se tuvo un prototipo avanzado y con funcionalidad se pidió retroalimentación a los Stakeholders, así como a otros externos al proyecto.

Esto fue necesario debido a que, al ser una etapa temprana, aún se estaba a tiempo de cambiar la interfaz y mejorarla para que cumpliera el objetivo lo mejor posible. El objetivo específico de la App era: *"Desarrollar una aplicación móvil que integre información epidemiológica y el reconocimiento de lesiones para predecir casos sospechosos de Leishmaniasis cutánea, en apoyo al diagnóstico de los técnicos de salud."*

Se consideró adecuado obtener retroalimentación desde etapas tempranas para lograr un correcto apoyo a los técnicos de salud.

Ilustración 32: Prototipo visual de App Leishmaniasis.

Ilustración 33: Prototipo visual de App Leishmaniasis.

b. Prototipo final y ajustes de retroalimentación

La retroalimentación obtenida por los *Stakeholders* del primer prototipo se puede resumir en: mejoras en rendimiento, cambio de tamaños y algunos flujos confusos relacionadas al UI de la App.

En el segundo prototipo, se condensó todas las sugerencias y se corrigieron errores, se realizaron ajustes de UI y otros.

En ese prototipo se incluyeron algunas mejoras:

- Se incluyeron animaciones de espera entre pantallas.
- Hubo pruebas internas de conexión al *backend*, para analizar lesiones, guardar casos y probar el modo *offline*.
- Se integró una versión preliminar del modelo de detección, resultando en una exitosa inclusión de este.

En el segundo prototipo, la aplicación estuvo un 80% lista, solamente en espera de mejoras para el modelo de detección de Leishmaniasis y que este fuera más certero.

Los resultados detallados de la toma de retroalimentación pueden consultarse en la sección de resultados, en el apartado de App Leishmaniasis.

c. Integración agente inteligente aplicación

Este fue sin lugar a duda el mayor reto a nivel tanto técnico como visual relacionada con la aplicación. Puso en tela de juicio todos los conocimientos relacionados con el modelo como en *framework* de React Native.

Para integrar el modelo con la aplicación se siguieron los siguientes pasos:

- Primero, el modelo se entrenó, se hizo la matriz de confusión y se probaron hiperparámetros para verificar que tuviera un rendimiento adecuado. Esto fue realizado por otro compañero en el módulo de inteligencia artificial.
- El modelo se exportó en varios archivos:
    - Un JSON con la información del modelo, las capas y otros parámetros de Tensorflow
    - Un conglomerado de binarios que poseen la información entrenada del modelo.
- Los archivos se colocaron en el directorio de archivos en la aplicación móvil.
- Se instalaron librerías de React Native relacionadas con Tensorflow.
- Se tomaron algunos cursos y tutoriales en internet para poder entender la lógica de importación, manejo y predicción del modelo con la aplicación que teníamos en ese momento.
- Luego, se tradujo la imagen tomada a binario, se le transfirió al modelo que eventualmente predijo.

Este proceso fue realizado en muchas ocasiones. La razón de esto fue que no se tomaron en cuenta algunas convenciones, versiones contrarias de librerías que cambian y funciones que no servían en situaciones requeridas. La integración tomó más tiempo del esperado, con cerca de treinta o cuarenta iteraciones. Estas, divididas entre fallos en la lógica de una librería, un modelo con hiperparámetros mejorables, o una fotografía no traducida a binario de forma correcta.

3. *Dashboard* Leishmaniasis

a. Primer prototipo y base de código

Primero se tomaron las maquetas o mockups de las animaciones, transiciones, pantallas y patrones de diseño que se tenían de Adobe XD. Al igual que con React Native, fue necesario sentar una base de código que incluía las siguientes herramientas.

- Base de código de React.js
- *Linter* para mantener el código saludable y fácil de leer
- Un *Prettier* para que el código se formateara automáticamente
- Integración de otras librerías de CSS y UI como Tailwind.css, Craco, etc.

Luego, para la parte de la programación, se siguieron estos pasos:

- Tomar maquetas, mockups, y otros patrones de diseño disponibles para saber la cantidad de pantallas, flujos y trabajo por hacer.
- Segmentar las pantallas en componentes, estos a su vez segmentarlos por funcionalidad. Por ejemplo, la pantalla de *login* posee botones de *login*, campos de entrada de texto, etc.
- Implementar cada componente con el *framework* React.js y luego unirlos según cada pantalla.
- Armar los flujos con componentes.
- Conectar gráficos y otras funcionalidades al *backend* y monitorear rendimiento.
- Hacer pruebas y otros factores funcionales para verificar el comportamiento de todos los componentes.

b. Prototipo final y toma de retroalimentación

Se contempló en la iteración final, la toma de retroalimentación tanto a nivel de opinión, como a nivel estadístico.

Para esto, se elaboró una encuesta en Typeform. La encuesta reflejaba el estado actual del prototipo. El principal problema con una interfaz es cuantificarla, ya que es algo visualmente subjetivo. Se solventó este problema agregando en la encuesta barras de indicadores de qué tanto gustó el diseño, así como sugerencias de mejora, etc.

Los resultados de la encuesta se encuentran tabulados en la sección de resultados.

Ilustración 34: Prototipo de Dashboard Leishmaniasis.

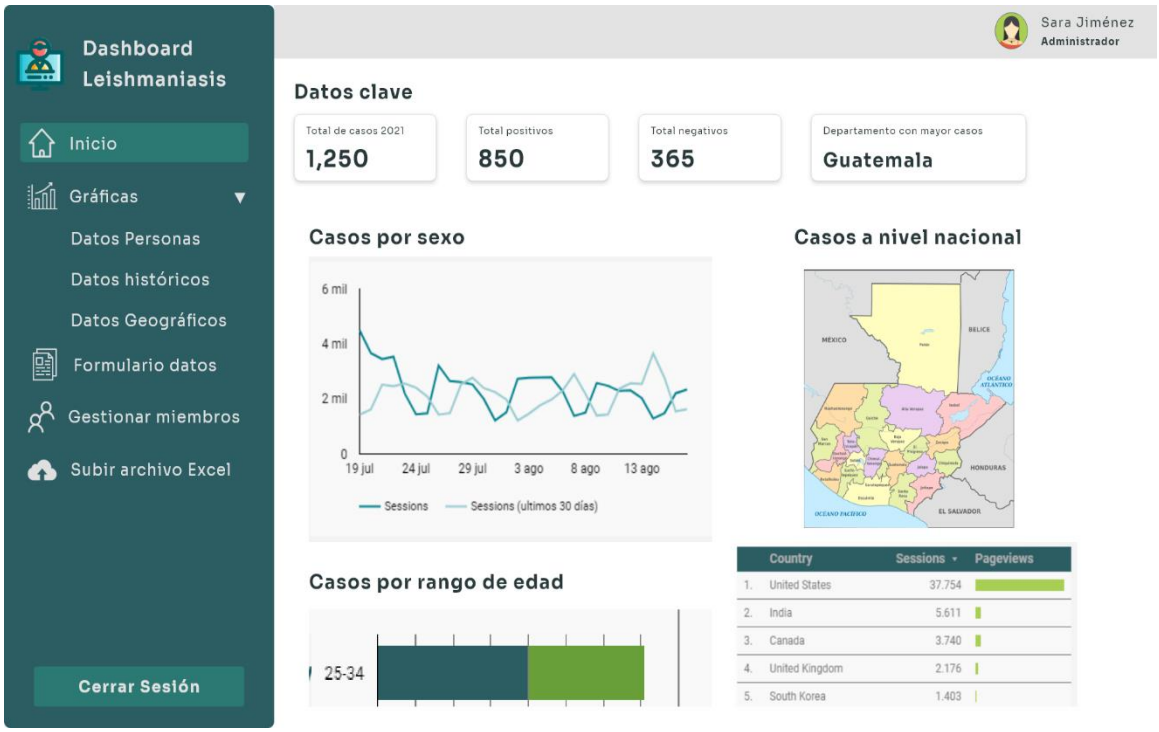


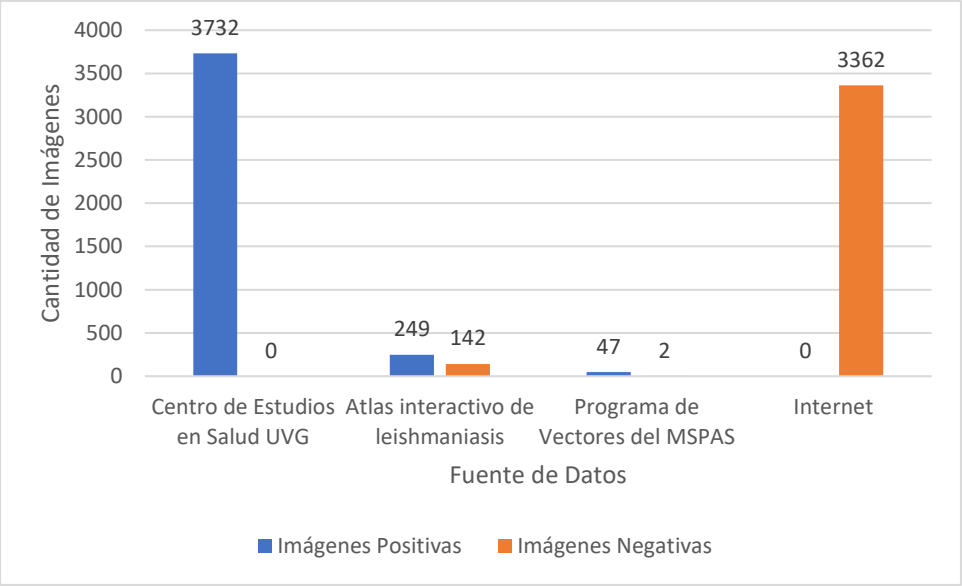
Ilustración 35: Prototipo visual de Dashboard Leishmaniasis.

VI. RESULTADOS

A. Inteligencia artificial para reconocimiento de lesiones de Leishmaniasis

1. Recolección de imágenes

Ilustración 36: Cantidad de imágenes recolectadas por fuente de datos



2. Partición del conjunto de datos

Tabla 7: Distribución de imágenes en los distintos conjuntos clasificadas por fuente de datos.

	Entrenamiento		Desarrollo		Prueba	
Fuente de datos	Positivo	Negativo	Positivo	Negativo	Positivo	Negativo
Centro de Estudios en Salud UVG	2240	0	746	0	746	0
Atlas interactivo de Leishmaniasis	149	85	50	29	50	28
Programa de vectores del MSPAS	28	0	9	0	10	2
Internet	0	2332	0	515	0	515
Total	2417	2417	805	544	806	545

3. Eficiencia humana

Ilustración 37: Resultados de pruebas realizadas a personas sin experiencia.

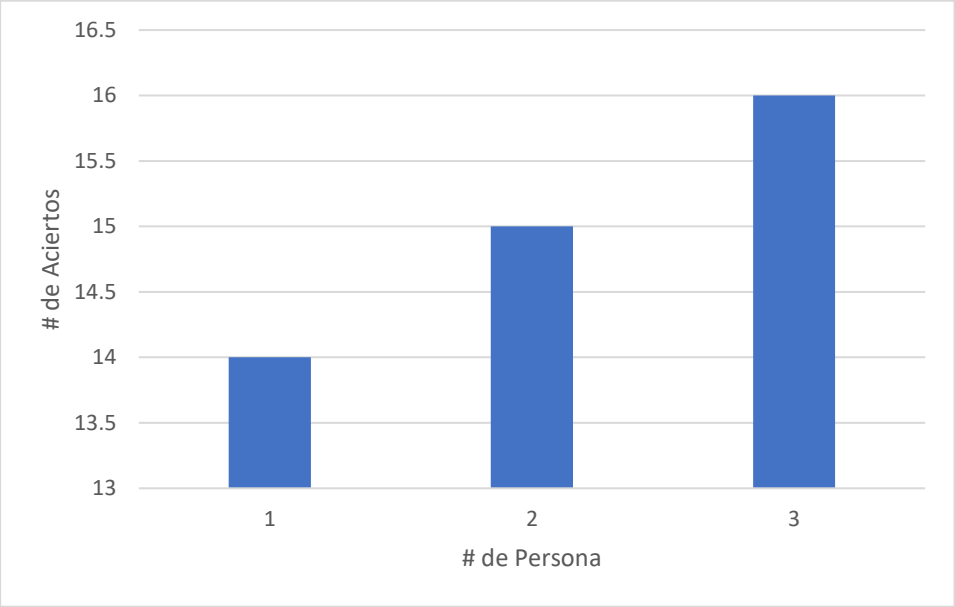


Ilustración 38: Resultados de pruebas realizadas a personas con experiencia.

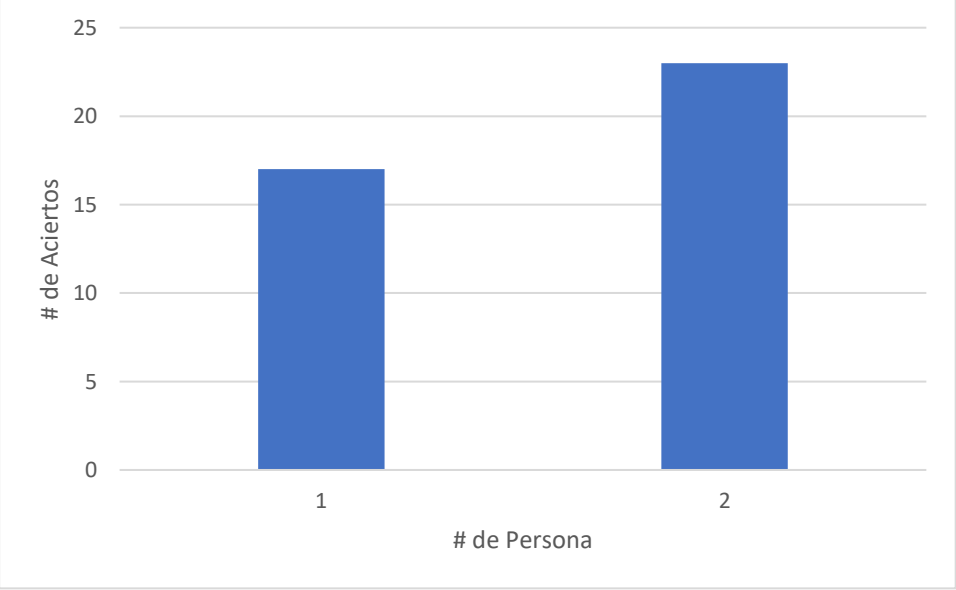


Ilustración 39: Resultados del modelo en la aplicación móvil.

Tabla 8: Resultados de modelo separando incorrectamente las imágenes.

Categoría	Precisión
Entrenamiento	99%
Desarrollo	98%
Prueba	37%

Tabla 9: Hiperparámetros de preprocesamiento y aumentación de datos.

Parámetro	Valor
rescale	1/255
featurewise_std_normalization	True
samplewise_std_normalization	True
rotation_range	40
width_shift_range	0.025
height_shift_range	0.010
horizontal_flip	True
vertical_flip	True
shear_range	0.017
zoom_range	0.075

Tabla 10: Arquitectura de modelo pequeño final.

Tipo de capa	Dimensión de Salida
Conv2D	None, 298, 298, 16
MaxPooling2D	None, 149, 149, 16
Conv2D	None, 147, 147, 32
MaxPooling2D	None, 73, 73, 32
Conv2D	None, 71, 71, 64
MaxPooling2D	None, 35, 35, 64
Conv2D	None, 33, 33, 64
MaxPooling2D	None, 16, 16, 64
Conv2D	None, 14, 14, 64
MaxPooling2D	None, 7, 7, 64
Flatten	None, 3136
Dense	None, 512
Dense	None, 1

Tabla 11: Comparación de cantidad de parámetros entre modelo InceptionV3 y modelo pequeño.

Descripción	Modelo Inception V3	Modelo Pequeño
Parámetros entrenables	32,026,535	1,704,097
Parámetros no entrenables	8,975,264	0
Total de parámetros	41,001,799	1,704,097

Tabla 12: Resultados de modelo final InceptionV3.

Categoría	Precisión
Entrenamiento	96.44%
Desarrollo	93.32%
Prueba	92.59%

Tabla 13: Resultados de modelo pequeño final.

Categoría	Precisión
Entrenamiento	94.43%
Desarrollo	91.54%
Prueba	91.41%

Tabla 14: Comparación de modelos utilizando distintas métricas.

	Exactitud	Precisión	Exhaustividad	Especificidad
Modelo pequeño	91.19%	90.95%	94.10%	87.22%
Modelo Inception V3	93.71%	97.01%	92.32%	95.77%

Tabla 15: Requerimientos mínimos para ejecutar el modelo en dispositivos móviles.

Descripción	Requerimiento mínimo
Memoria RAM	2GB
Versión de Android	5
Memoria de almacenamiento	200MB

B. Gráficas del panel de control

Tabla 16: Distribución de indicadores en panel de control.

Descripción	Cantidad			
	Tablero de inicio	Tablero de estadísticas de personas	Tablero de estadísticas históricas	Tablero de estadísticas geográficas
Indicadores Clave	4	1	0	0
Filtros	1	1	3	3
Gráficos	2	7	3	5

Tabla 17: Resultados de prueba de usabilidad.

Flujo	Tiempo promedio
Identificar los departamentos con más casos en el año actual.	28s
Visualizar estadísticas de personas del último mes.	1m 14s
Identificar cantidad de casos en rango de edad mayores a 70 años.	7s
Visualizar gráficos históricos únicamente del departamento de Zacapa y sexo masculino.	1m 11s
Visualizar cantidad de casos de los municipios de Retalhuleu.	39s

Ilustración 40: Tablero de inicio del panel de control.

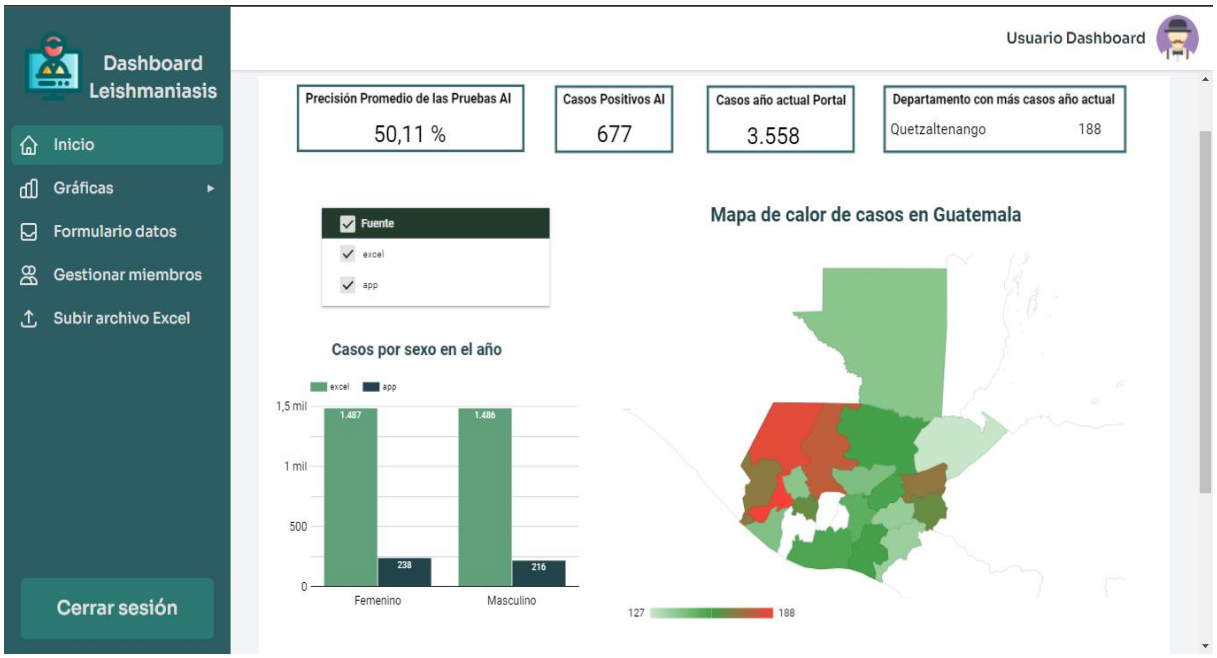


Ilustración 41: Tablero de estadísticas de personas del panel de control.

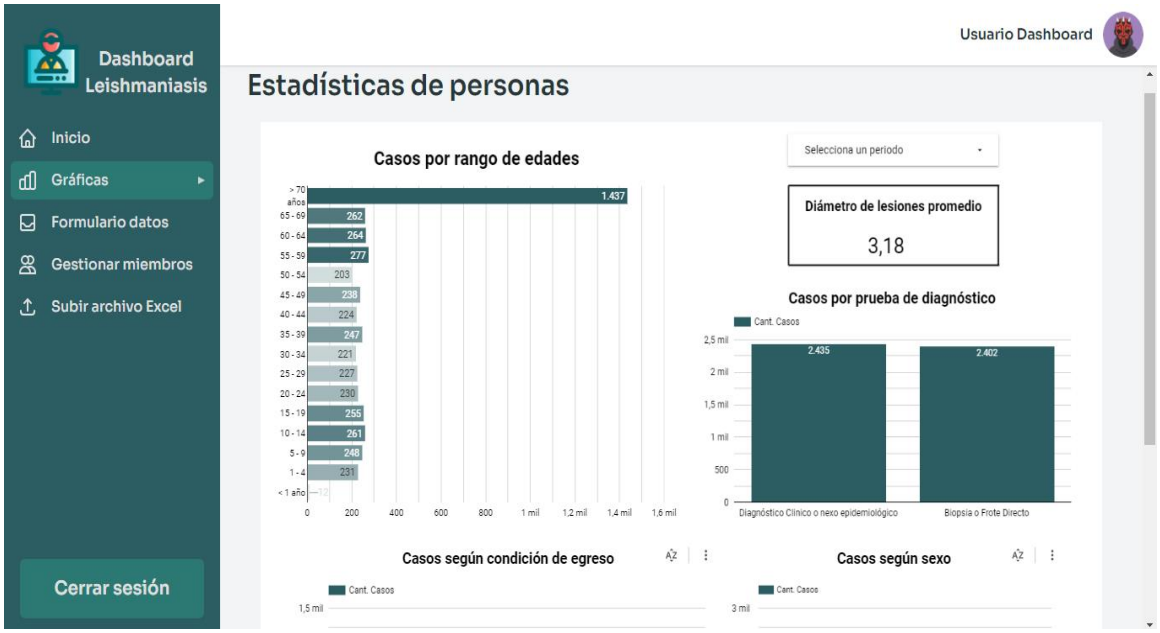
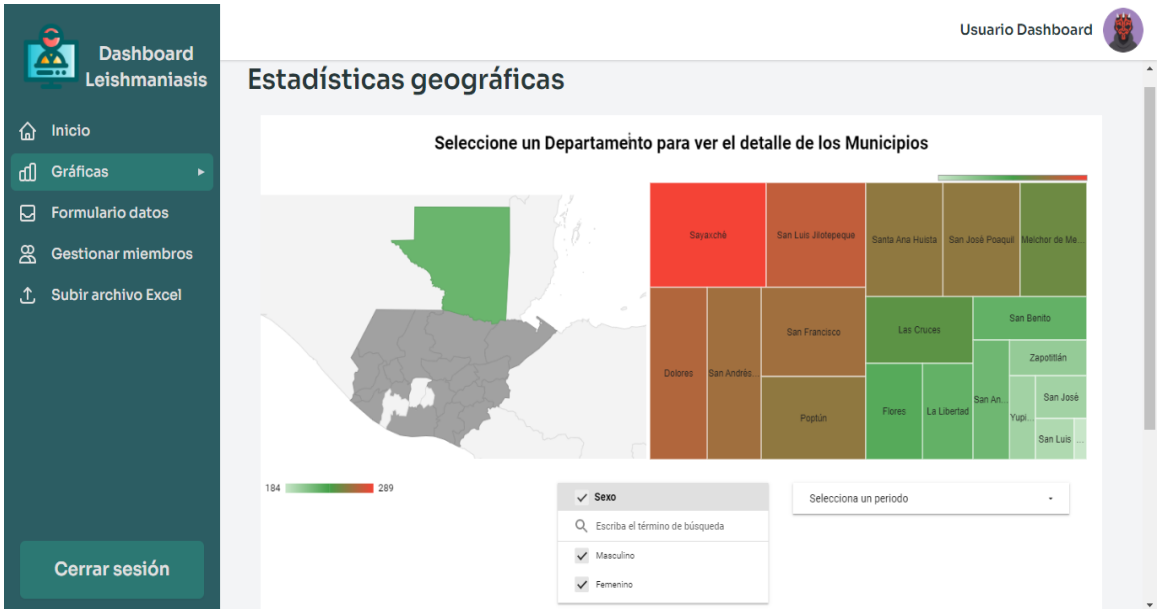


Ilustración 42: Estadísticas históricas del panel de control.

Ilustración 43: Estadísticas geográficas del panel de control.



C. Consultas con API

Se realizaron consultas de forma independientes al sistema tanto de forma local, como ya montado en el servidor. Se utilizó la herramienta Postman para realizar estas pruebas, ya que también provee tiempos de respuesta, seccionándolo en cada procedimiento que se hace entre el cliente y el servidor.

Ilustración 44: Resultado de consulta con CORS configurado.

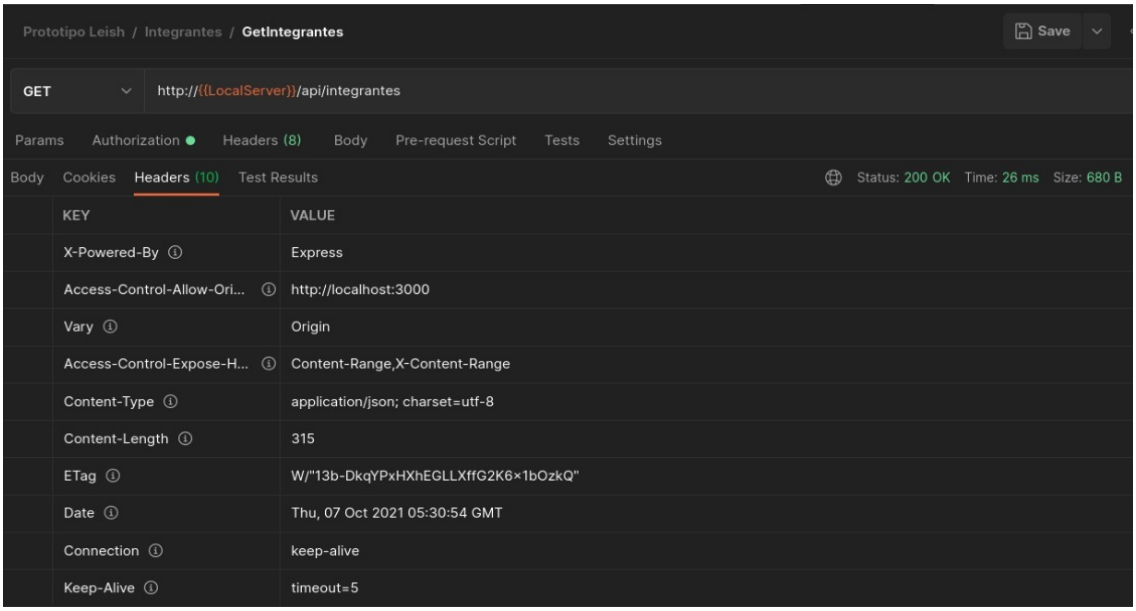


Ilustración 45: Consulta tipo GET hacia todas las entidades de un tipo.

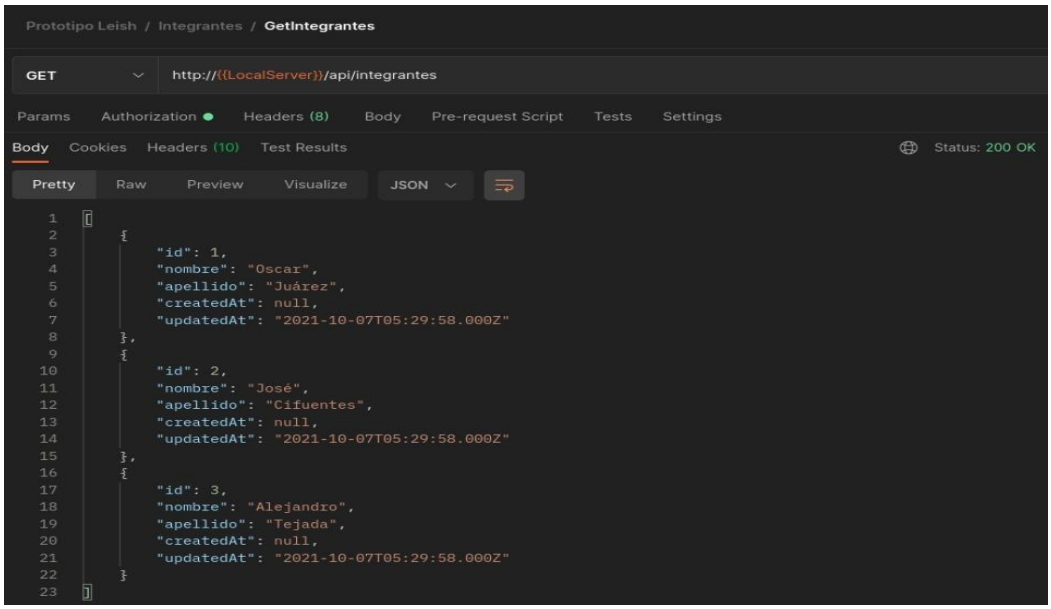


Ilustración 46: Consulta tipo POST de la entidad integrantes.

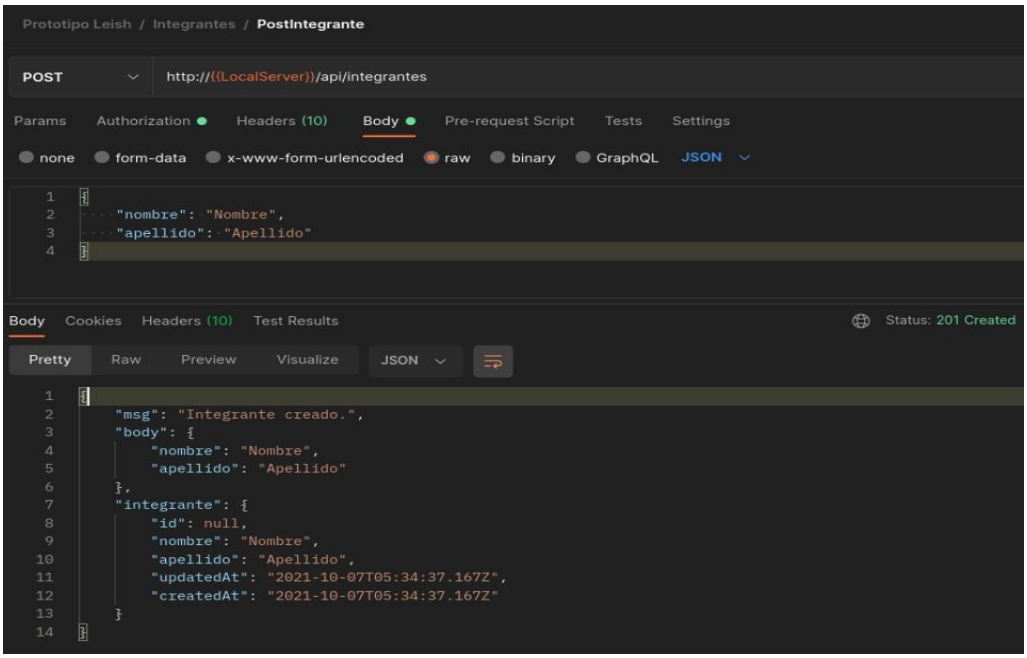


Ilustración 47: Resultado en base de datos luego de consulta tipo POST.

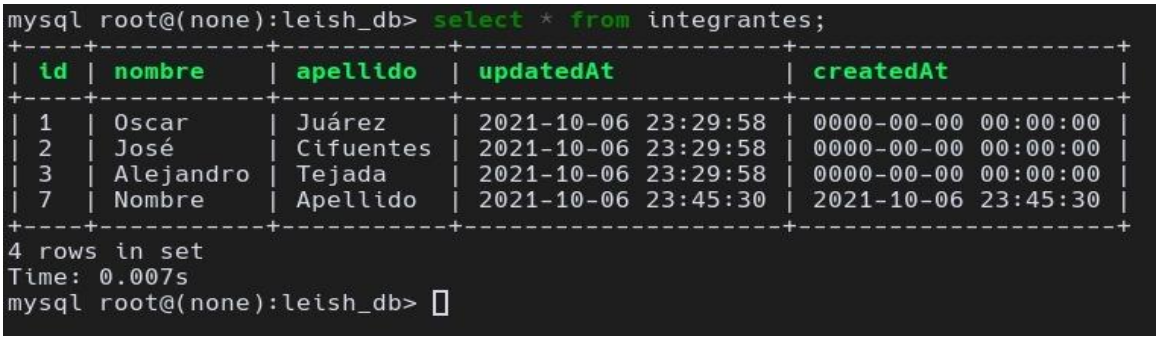


Ilustración 48: Consulta tipo PUT hacia la entidad integrante.

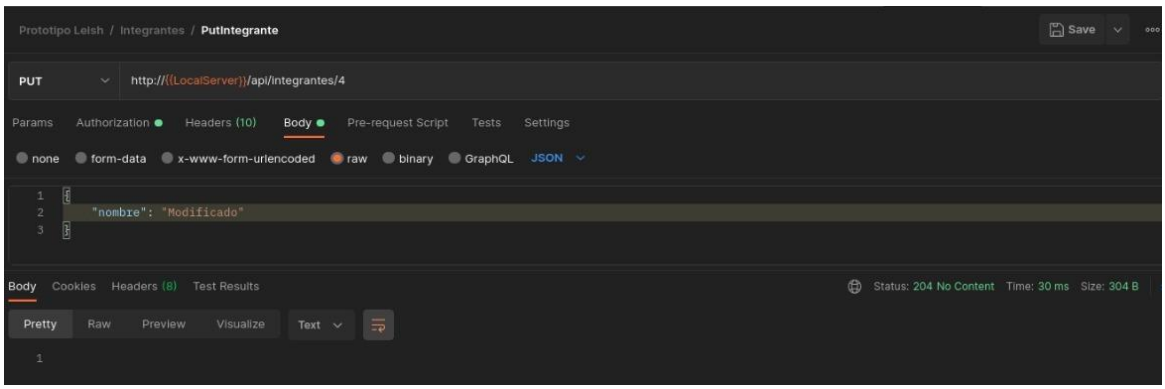


Ilustración 49: Resultado en base de datos luego de consulta tipo PUT.

```
mysql root@(none):leish_db> select * from integrantes;
+----+------------+-----------+---------------------+---------------------+
| id | nombre     | apellido  | updatedAt           | createdAt           |
+----+------------+-----------+---------------------+---------------------+
| 1  | Oscar      | Juárez    | 2021-10-06 23:29:58 | 0000-00-00 00:00:00 |
| 2  | José       | Cifuentes | 2021-10-06 23:29:58 | 0000-00-00 00:00:00 |
| 3  | Alejandro  | Tejada    | 2021-10-06 23:29:58 | 0000-00-00 00:00:00 |
| 7  | Modificado | Apellido  | 2021-10-06 23:46:14 | 2021-10-06 23:45:30 |
+----+------------+-----------+---------------------+---------------------+
4 rows in set
Time: 0.007s
mysql root@(none):leish_db>
```

Ilustración 50: Consulta tipo GET a servidor online.

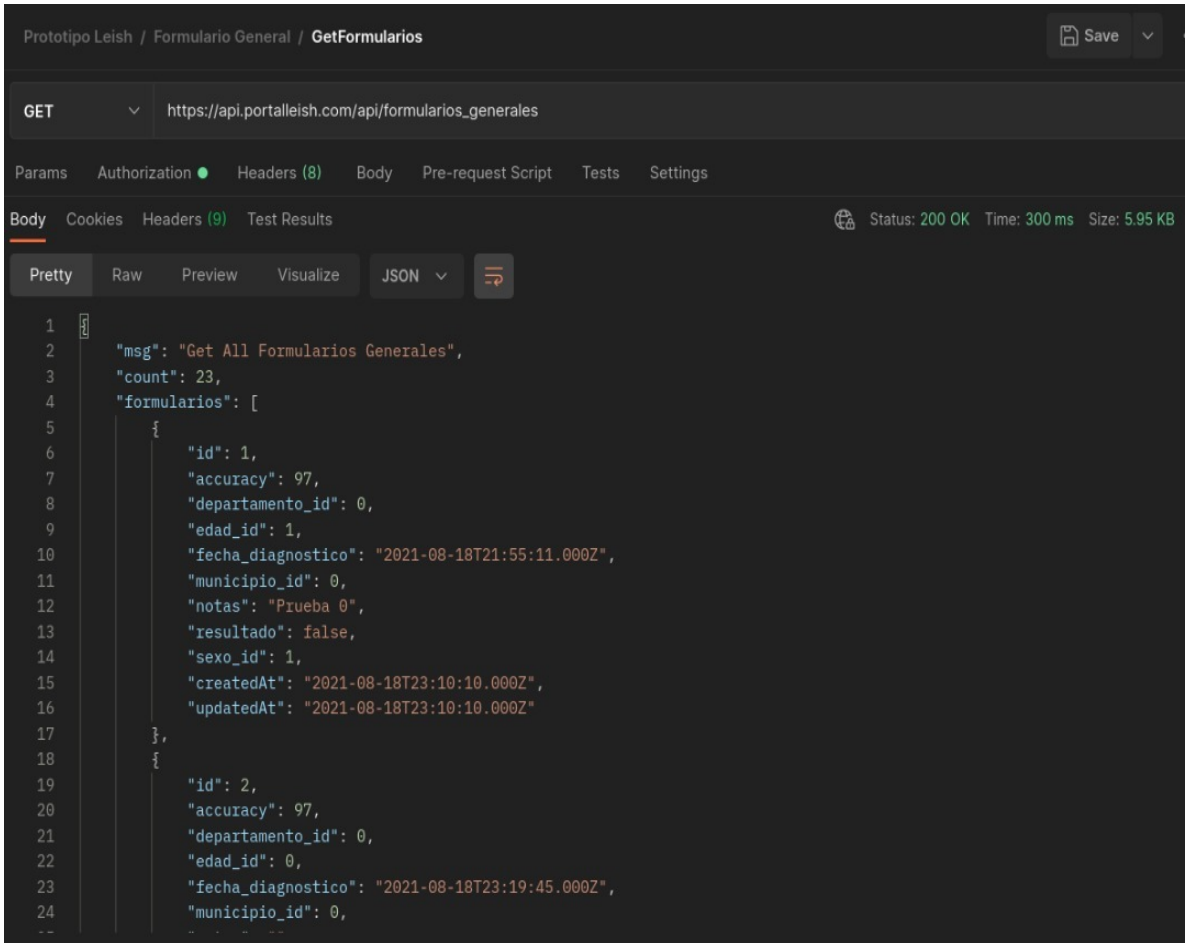


Ilustración 51: Consulta tipo POST a servidor online.

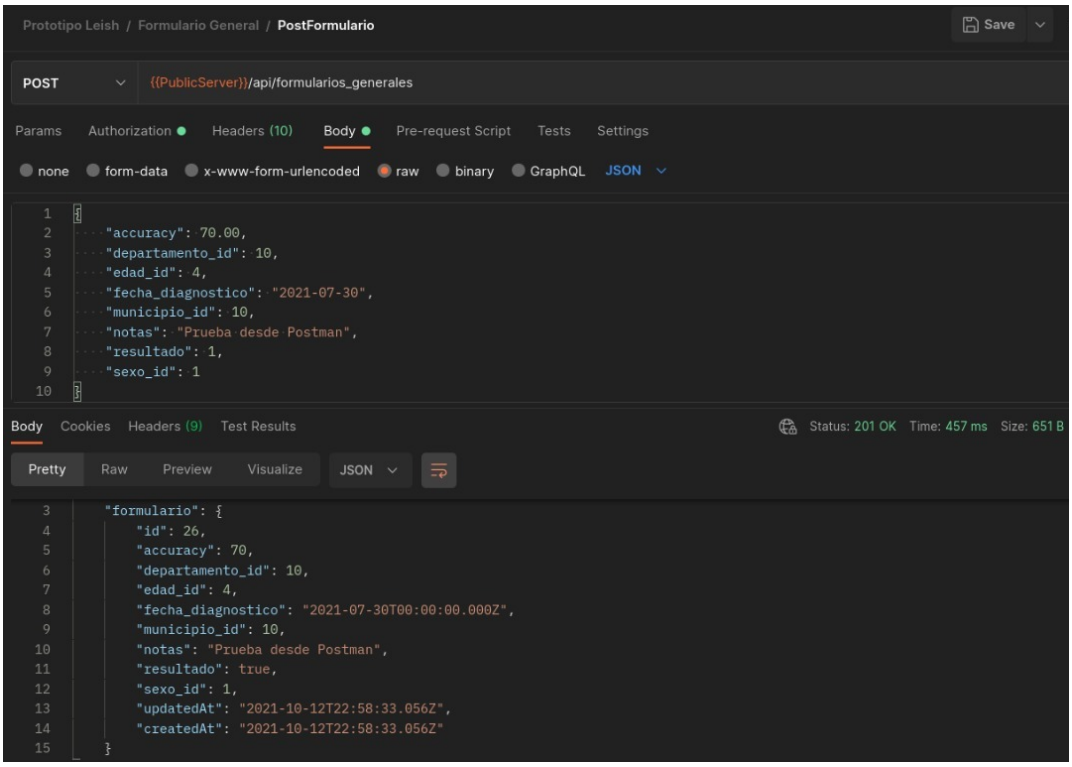
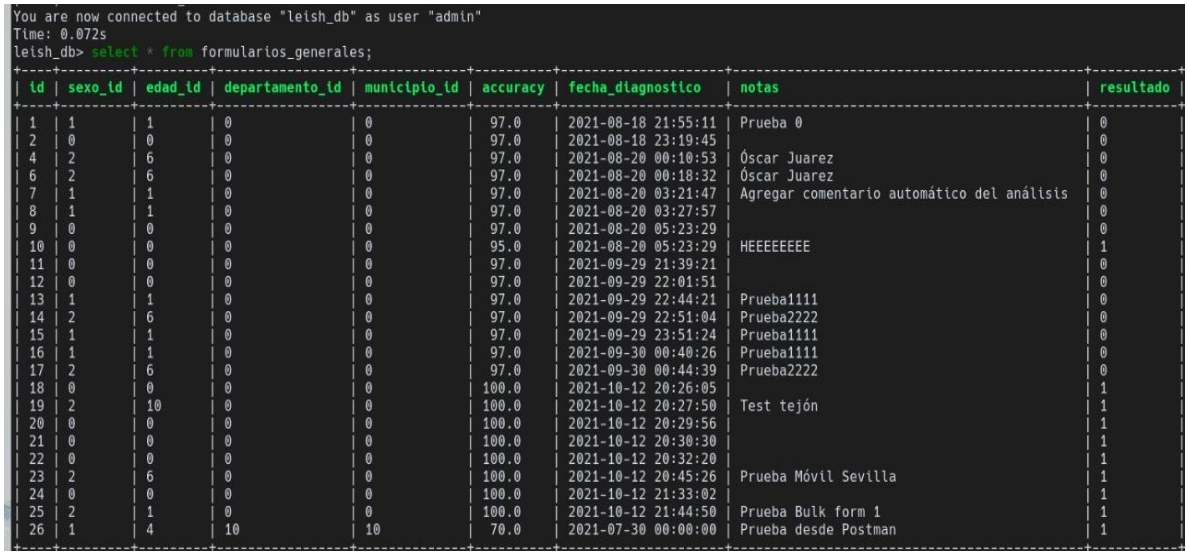


Ilustración 52: Base de datos del servidor luego de consulta tipo POST.



```
You are now connected to database "leish_db" as user "admin"
Time: 0.072s
leish_db> select * from formularios_generales;
```

id	sexo_id	edad_id	departamento_id	municipio_id	accuracy	fecha_diagnostico	notas	resultado
1	1	1	0	0	97.0	2021-08-18 21:55:11	Prueba 0	0
2	0	0	0	0	97.0	2021-08-18 23:19:45		0
4	2	6	0	0	97.0	2021-08-20 00:10:53	Óscar Juarez	0
6	2	6	0	0	97.0	2021-08-20 00:18:32	Óscar Juarez	0
7	1	1	0	0	97.0	2021-08-20 03:21:47	Agregar comentario automático del análisis	0
8	1	1	0	0	97.0	2021-08-20 03:27:57		0
9	0	0	0	0	97.0	2021-08-20 05:23:29		0
10	0	0	0	0	95.0	2021-08-20 05:23:29	HEEEEEEEE	1
11	0	0	0	0	97.0	2021-09-29 21:39:21		0
12	0	0	0	0	97.0	2021-09-29 22:01:51		0
13	1	1	0	0	97.0	2021-09-29 22:44:21	Prueba1111	0
14	2	6	0	0	97.0	2021-09-29 22:51:04	Prueba2222	0
15	1	1	0	0	97.0	2021-09-29 23:51:24	Prueba1111	0
16	1	1	0	0	97.0	2021-09-30 00:40:26	Prueba1111	0
17	2	6	0	0	97.0	2021-09-30 00:44:39	Prueba2222	0
18	0	0	0	0	100.0	2021-10-12 20:26:05		1
19	2	10	0	0	100.0	2021-10-12 20:27:50	Test tejón	1
20	0	0	0	0	100.0	2021-10-12 20:29:56		1
21	0	0	0	0	100.0	2021-10-12 20:30:30		1
22	0	0	0	0	100.0	2021-10-12 20:32:20		1
23	2	6	0	0	100.0	2021-10-12 20:45:26	Prueba Móvil Sevilla	1
24	0	0	0	0	100.0	2021-10-12 21:33:02		1
25	2	1	0	0	100.0	2021-10-12 21:44:50	Prueba Bulk form 1	1
26	1	4	10	10	70.0	2021-07-30 00:00:00	Prueba desde Postman	1

Ilustración 53: Consulta tipo POST de cien formularios.

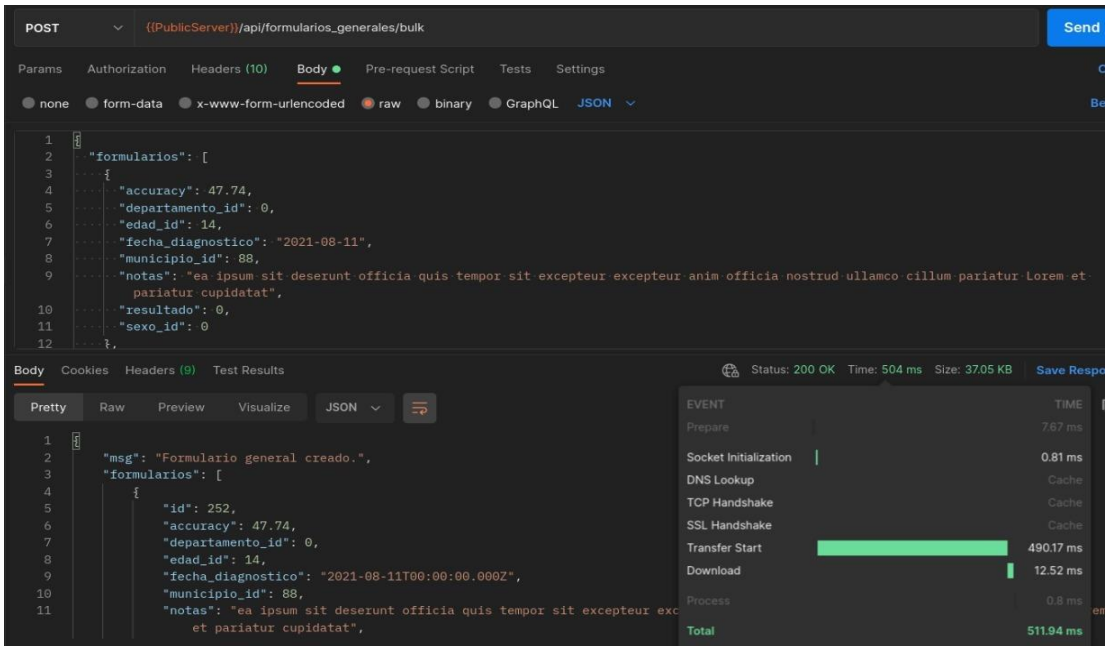
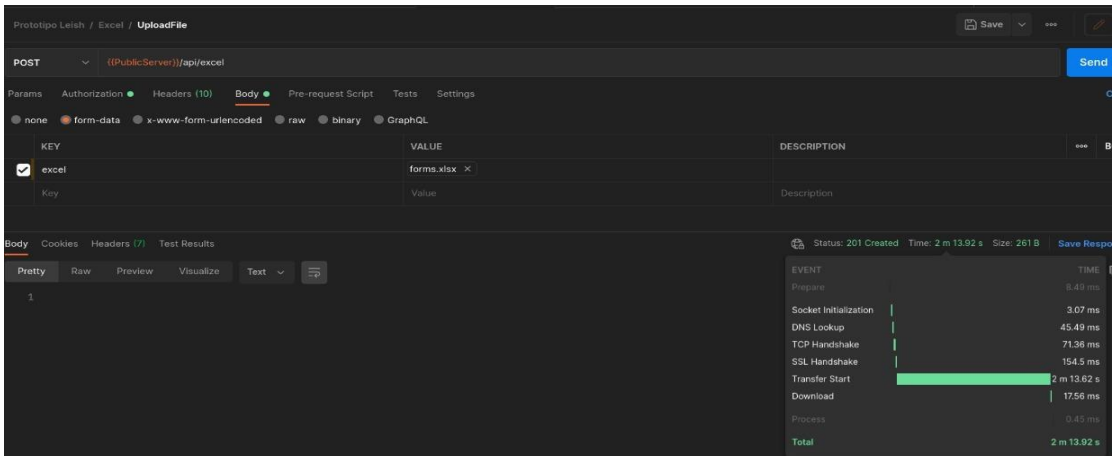


Ilustración 54: Registros actualizados en bases de datos.

id	sexo_id	edad_id	departamento_id	municipio_id	accuracy	fecha_diagnostico	notas
1	1	6	4	49	37.71	2021-11-09 15:09:07	id ea dolore non ea laborum esse al
2	<null>	<null>	<null>	<null>	67.55	2021-11-03 19:50:13	<null>
3	1	15	14	194	48.71	2021-11-03 11:16:25	mollit pariatur ex proident aliquip
4	0	9	11	145	65.54	2021-11-08 21:32:31	cillum labore et et officia officia
5	0	13	2	21	65.35	2021-11-08 18:53:55	irure incididunt incididunt tempor
6	0	8	18	265	81.82	2021-11-03 23:47:52	quis sunt velit tempor reprehenderi
7	1	10	14	186	40.04	2021-11-14 04:43:36	magna pariatur laboris aute duis
8	1	5	3	27	84.18	2021-11-08 23:16:52	laboris mollit adipisicing ad id te
9	0	9	18	267	45.14	2021-11-04 06:03:59	ut excepteur ut velit esse laborum
10	1	6	1	5	35.28	2021-11-09 09:59:54	occaecat ea tempor nulla nisi non e
11	0	2	19	288	62.31	2021-11-07 10:24:23	irure enim quis fugiat amet et sit
12	0	7	12	156	64.84	2021-11-03 04:40:40	ipsum id Lorem enim eu ea sunt
13	1	12	1	8	4.63	2021-11-05 17:30:59	commodo aute cillum cillum minim ma
14	0	2	14	200	9.55	2021-11-10 22:52:55	esse elit laborum nisi do deserunt
15	0	6	17	234	93.85	2021-11-12 17:58:52	et nulla deserunt non minim et aliq
16	0	16	6	66	55.71	2021-11-13 16:31:29	fugiat
17	1	15	8	90	0.78	2021-11-03 15:39:11	sit exercitation reprehenderit cons
18	1	16	21	315	2.96	2021-11-03 17:09:22	ullamco reprehenderit excepteur aut
19	0	8	11	143	10.89	2021-11-01 09:53:27	tempor tempor magna mollit ullamco
20	0	6	12	157	97.53	2021-11-09 00:03:37	reprehenderit exercitation dolor do
21	1	4	14	206	77.38	2021-11-10 09:48:05	laborum enim sint nulla voluptate
22	<null>	<null>	<null>	<null>	90.81	2021-11-10 01:11:33	<null>
23	0	8	1	10	78.02	2021-11-12 10:45:07	proident
24	0	11	12	155	26.45	2021-11-13 17:04:19	incididunt qui exercitation esse es
25	<null>	<null>	<null>	<null>	42.28	2021-11-11 18:26:41	<null>
26	0	11	13	171	99.17	2021-11-01 05:05:00	sit ut et eiusmod mollit aute sint
27	1	3	14	191	34.79	2021-11-07 11:36:24	in cillum sit nulla reprehenderit n
28	0	4	17	233	11.86	2021-11-11 01:45:36	velit magna voluptate proident qui
29	0	9	3	26	92.11	2021-11-12 11:26:10	ex cillum
30	<null>	<null>	<null>	<null>	95.95	2021-11-01 10:29:24	<null>
31	<null>	<null>	<null>	<null>	26.62	2021-11-09 03:30:05	<null>
32	0	10	2	18	82.77	2021-11-01 10:14:47	do minim aliquip ipsum dolor Lorem
33	<null>	<null>	<null>	<null>	47.08	2021-11-08 09:16:49	<null>
34	1	1	14	196	78.45	2021-11-07 23:35:13	aliquip mollit duis duis magna ea e
35	1	4	5	53	6.79	2021-11-01 01:34:02	esse adipisicing exercitation esse
36	1	11	20	307	4.14	2021-11-14 14:50:39	consectetur minim dolore magna temp
37	<null>	<null>	<null>	<null>	13.66	2021-11-01 18:03:19	<null>
38	0	16	8	104	69.46	2021-11-08 20:01:46	sunt commodo ipsum dolor dolore ess
39	0	13	11	141	88.52	2021-11-10 11:48:30	reprehenderit nostrud esse veniam m
40	1	7	7	75	81.75	2021-11-04 18:32:53	proident cillum nulla officia exerc
41	1	10	14	186	64.62	2021-11-13 18:42:14	irure ad ut dolor labore id nulla e
42	0	10	11	139	73.05	2021-11-05 17:52:32	dolore adipisicing deserunt laborum
43	0	9	16	228	62.01	2021-11-02 11:48:23	occaecat id tempor reprehenderit oc
44	1	2	3	27	55.92	2021-11-06 03:44:51	mollit ex amet non consectetur labo
45	1	6	1	7	9.56	2021-11-06 07:32:23	dolor qui pariatur deserunt dolor q
46	0	1	14	197	78.64	2021-11-04 22:42:07	cillum eiusmod non excepteur
47	1	8	5	58	14.55	2021-11-01 07:59:31	excepteur excepteur veniam excepteu
48	1	8	1	3	18.08	2021-11-08 19:26:20	ex dolor dolore dolor qui ea cupida
49	1	9	14	196	95.32	2021-11-05 15:57:49	dolor occaecat commodo laboris nost
50	1	7	17	254	78.85	2021-11-09 20:45:13	deserunt consequat in culpa ut ea a

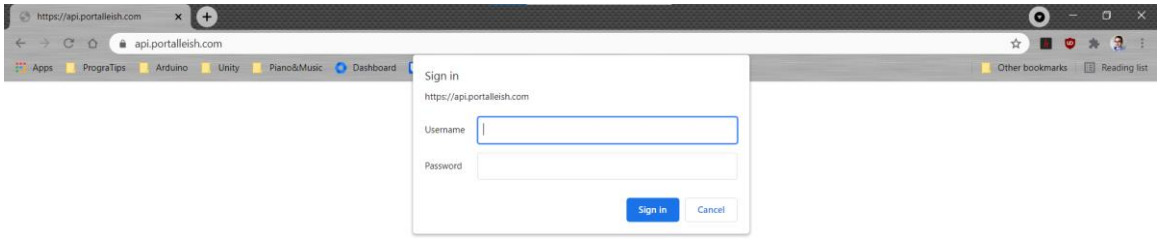
Ilustración 55: Consulta tipo POST de formulario Excel con 10,000 datos.



D. Infraestructura

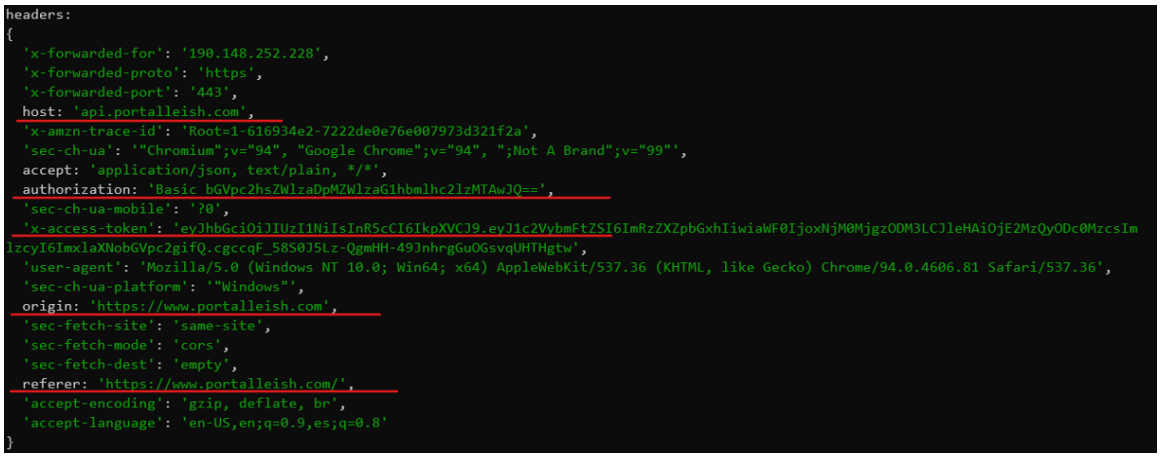
1. API

Ilustración 56: Autenticación HTTP básica para API.



En esta parte se puede visualizar el desplegable para la autenticación básica http del API desarrollada desde Google Chrome.

Ilustración 57: Impresión de encabezados.



```
headers:
{
  'x-forwarded-for': '190.148.252.228',
  'x-forwarded-proto': 'https',
  'x-forwarded-port': '443',
  host: 'api.portalleish.com',
  'x-amzn-trace-id': 'Root=1-616934e2-7222de0e76e007973d321f2a',
  'sec-ch-ua': '"Chromium";v="94", "Google Chrome";v="94", ";Not A Brand";v="99"',
  accept: 'application/json, text/plain, */*',
  authorization: 'Basic bGVpc2hsZWlzaDpMZWlzaG1hbmlhc2lzMTAwJQ==',
  'sec-ch-ua-mobile': '?0',
  'x-access-token': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImRzZXZpbGxhIiwiaWF0IjoxNjM0MjgzODM3LCJleHAiOjE2MzQyODc0MzcsIm
lzcyI6ImxlaXNobGVpc2gifQ.cgccqF_58S0J5Lz-QgmHH-49JnhrgGuOGsvqUHTHgtw',
  'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.81 Safari/537.36',
  'sec-ch-ua-platform': '"Windows"',
  origin: 'https://www.portalleish.com',
  'sec-fetch-site': 'same-site',
  'sec-fetch-mode': 'cors',
  'sec-fetch-dest': 'empty',
  referer: 'https://www.portalleish.com/',
  'accept-encoding': 'gzip, deflate, br',
  'accept-language': 'en-US,en;q=0.9,es;q=0.8'
}
```

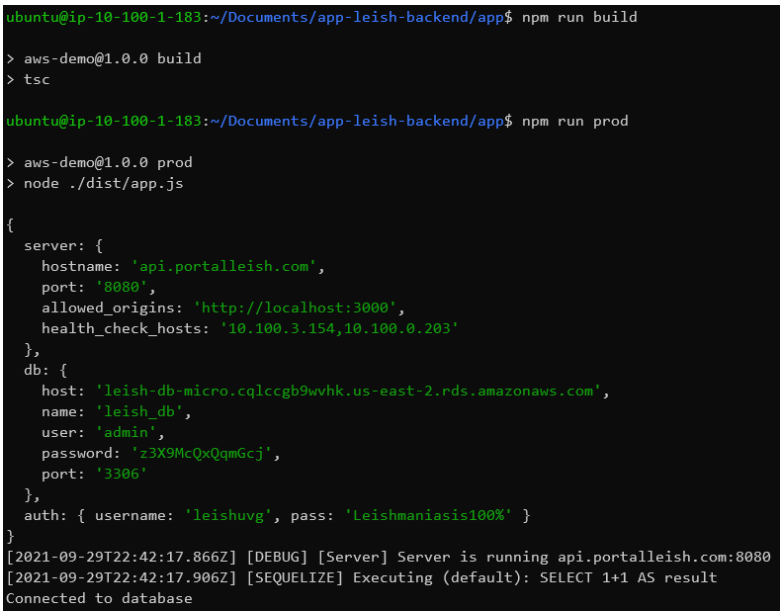
Aquí se observan los encabezados de la consulta realizada por el portal web en el dominio de www.portalleish.com, al API o Backend del portal web en el dominio api.portalleish.com.

2. Aplicación con agente inteligente

La aplicación se diseñó para poder cargar formularios de dos formas distintas, de forma simple y en conglomerado (bulk). En la forma simple se carga únicamente un formulario a la vez, mientras que en bulk, se almacenan varios formularios en la memoria del dispositivo y se cargan todos en una misma consulta.

a. Inicio de servidor con backend de la aplicación

Ilustración 58: Inicio de servidor con backend de la aplicación.



En esta parte se observa el inicio de Express en el servidor con el Backend de la aplicación. Se observa la configuración del servidor en el objeto **server** y la configuración de la base de datos en el objeto **db**. Además, en el objeto **auth**, están incluidos las credenciales de la autenticación HTTP básica.

b. Registro de formulario simple

Ilustración 59: Flujo de prueba de formulario simple en aplicación.

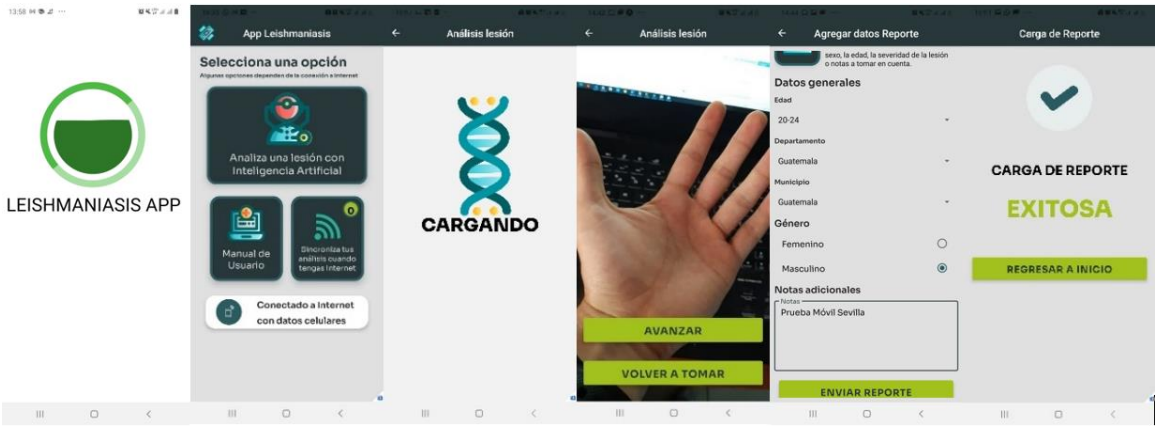
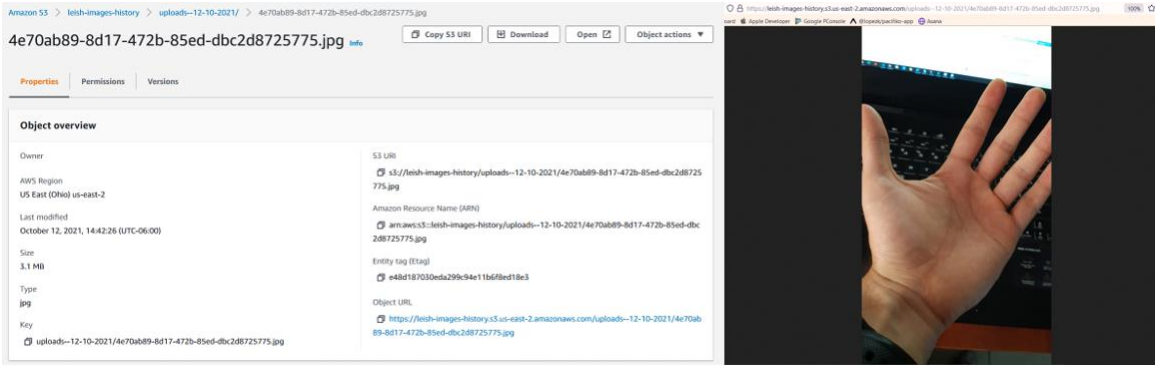


Ilustración 60: Almacenamiento de la imagen en S3.



En la imagen izquierda se pueden observar las propiedades de la imagen en S3 y, al lado derecho, la imagen desplegada en el explorador.

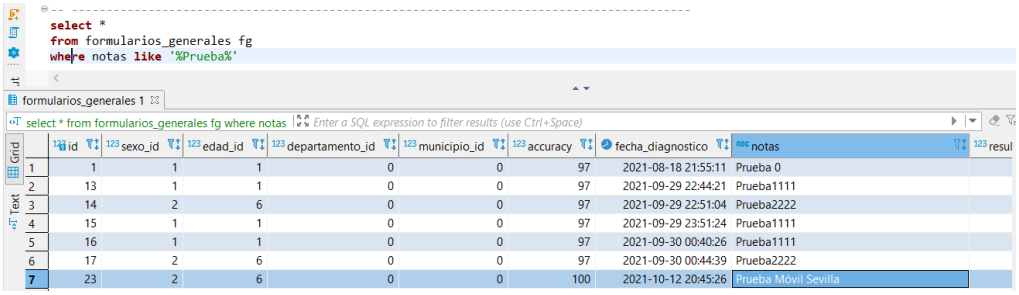
Ilustración 61: Logging de base de datos (Sequelize) – registro de formulario simple.

```
ubuntu@ip-10-100-1-183:~/Documents/app-leish-backend/app$ tail -f logs/sequelize.log
{"message":"[2021-10-12T20:45:25.407Z] [SEQUELIZE] Executing (default): INSERT INTO `formularios_generales` (`id`,`accuracy`,`departamento_id`,`edad_id`,`fecha_
diagnostico`,`municipio_id`,`notas`,`resultado`,`sexo_id`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-12T20:45:25.418Z] [INFO] [FORMULARIO] METHOD: [POST] - URL: [/] - IP: [::ffff:10.100.3.154]","level":"info"}
{"message":"{\"accuracy\":100,\"departamento_id\":\"Guatemala\",\"edad_id\":6,\"fecha_diagnostico\":\"2021-10-12T20:45:26.070Z\",\"municipio_id\":\"Guatemala\",
\"notas\":\"Prueba Móvil Sevilla\",\"resultado\":true,\"sexo_id\":2}","level":"info"}
```

Ilustración 62: Logging de acceso (Inserción) – registro de formulario simple.

```
ubuntu@ip-10-100-1-183:~/Documents/app-leish-backend/app$ tail -f logs/access.log
{"message":"[2021-10-12T20:45:25.418Z] [INFO] [FORMULARIO] METHOD: [POST] - URL: [/] - IP: [::ffff:10.100.3.154]","level":"info"}
{"message":"{\"accuracy\":100,\"departamento_id\":\"Guatemala\",\"edad_id\":6,\"fecha_diagnostico\":\"2021-10-12T20:45:26.070Z\",\"municipio_id\":\"Guatemala\",
\"notas\":\"Prueba Móvil Sevilla\",\"resultado\":true,\"sexo_id\":2}","level":"info"}
```

Ilustración 63: Resultados en base de datos – registro de formulario simple.



```
select *
from formularios_generales fg
where notas like '%Prueba%'
```

formularios_generales 1

select * from formularios_generales fg where notas — Enter a SQL expression to filter results (use Ctrl+Space)

	id	sexo_id	edad_id	departamento_id	municipio_id	accuracy	fecha_diagnostico	notas	resul
1	1	1	1	0	0	97	2021-08-18 21:55:11	Prueba 0	
2	13	1	1	0	0	97	2021-09-29 22:44:21	Prueba1111	
3	14	2	6	0	0	97	2021-09-29 22:51:04	Prueba2222	
4	15	1	1	0	0	97	2021-09-29 23:51:24	Prueba1111	
5	16	1	1	0	0	97	2021-09-30 00:40:26	Prueba1111	
6	17	2	6	0	0	97	2021-09-30 00:44:39	Prueba2222	
7	23	2	6	0	0	100	2021-10-12 20:45:26	Prueba Móvil Sevilla	

c. Registro de formulario en conglomerado

Ilustración 64: Flujo de prueba de conglomerado de formularios en aplicación.

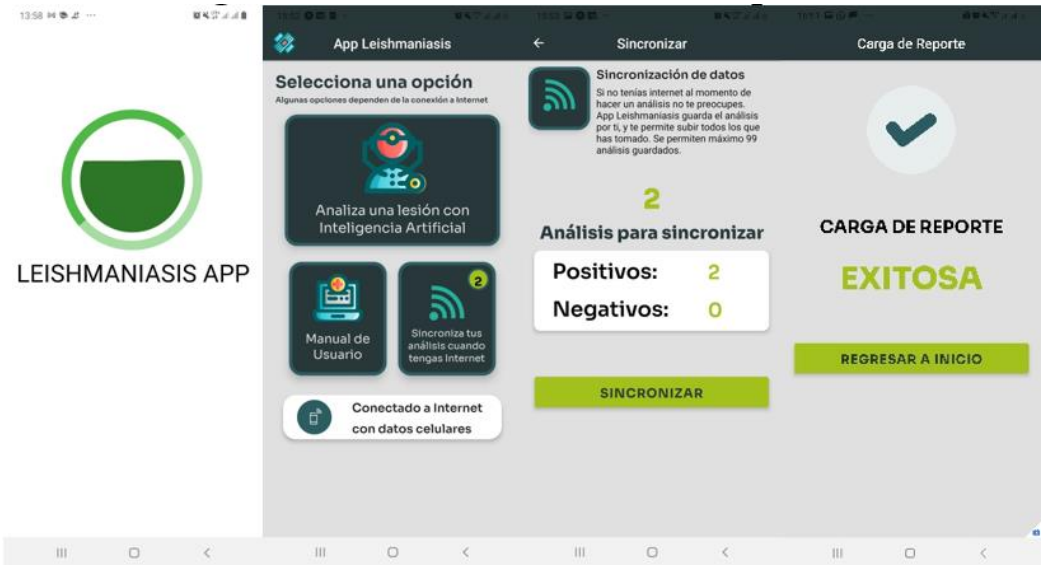
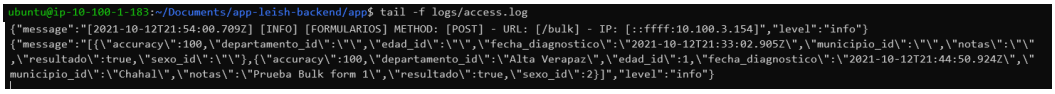
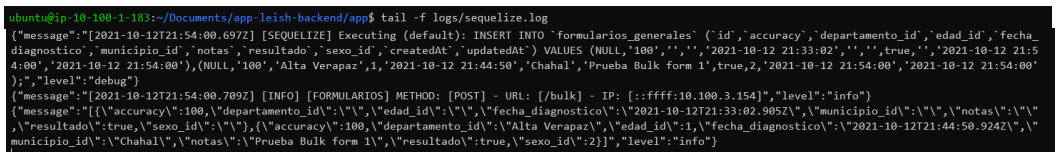


Ilustración 65: Logging de acceso (Inserción) – formulario en conglomerado.



```
ubuntu@ip-10-100-1-183:~/Documents/app-leish-backend/app$ tail -f logs/access.log
{"message":"[2021-10-12T21:54:00.709Z] [INFO] [FORMULARIOS] METHOD: [POST] - URL: [/bulk] - IP: [::ffff:10.100.3.154]","level":"info"}
{"message":"[{\"accuracy\":100,\"departamento_id\":\"\",\"edad_id\":\"\",\"fecha_diagnostico\":\"2021-10-12T21:33:02.905Z\",\"municipio_id\":\"\",\"notas\":\"
,\"resultado\":true,\"sexo_id\":\"\"},{\"accuracy\":100,\"departamento_id\":\"Alta Verapaz\",\"edad_id\":1,\"fecha_diagnostico\":\"2021-10-12T21:44:50.924Z\",\"
municipio_id\":\"Chahal\",\"notas\":\"Prueba Bulk form 1\",\"resultado\":true,\"sexo_id\":2}]","level":"info"}
```

Ilustración 66: Logs de base de datos (Sequelize) – formulario en conglomerado.



```
ubuntu@ip-10-100-1-183:~/Documents/app-leish-backend/app$ tail -f logs/sequelize.log
{"message":"[2021-10-12T21:54:00.697Z] [SEQUELIZE] Executing (default): INSERT INTO `formularios_generales` (`id`,`accuracy`,`departamento_id`,`edad_id`,`fecha_
diagnostico`,`municipio_id`,`notas`,`resultado`,`sexo_id`,`createdAt`,`updatedAt`) VALUES (NULL,'100','','','2021-10-12 21:33:02','','',true,'','2021-10-12 21:5
4:00','2021-10-12 21:54:00'),(NULL,'100','Alta Verapaz',1,'2021-10-12 21:44:50','Chahal','Prueba Bulk form 1',true,2,'2021-10-12 21:54:00','2021-10-12 21:54:00'
);","level":"debug"}
{"message":"[2021-10-12T21:54:00.709Z] [INFO] [FORMULARIOS] METHOD: [POST] - URL: [/bulk] - IP: [::ffff:10.100.3.154]","level":"info"}
{"message":"[{\"accuracy\":100,\"departamento_id\":\"\",\"edad_id\":\"\",\"fecha_diagnostico\":\"2021-10-12T21:33:02.905Z\",\"municipio_id\":\"\",\"notas\":\"
,\"resultado\":true,\"sexo_id\":\"\"},{\"accuracy\":100,\"departamento_id\":\"Alta Verapaz\",\"edad_id\":1,\"fecha_diagnostico\":\"2021-10-12T21:44:50.924Z\",\"
municipio_id\":\"Chahal\",\"notas\":\"Prueba Bulk form 1\",\"resultado\":true,\"sexo_id\":2}]","level":"info"}
```

Ilustración 67: Resultado en base de datos – formulario en conglomerado.

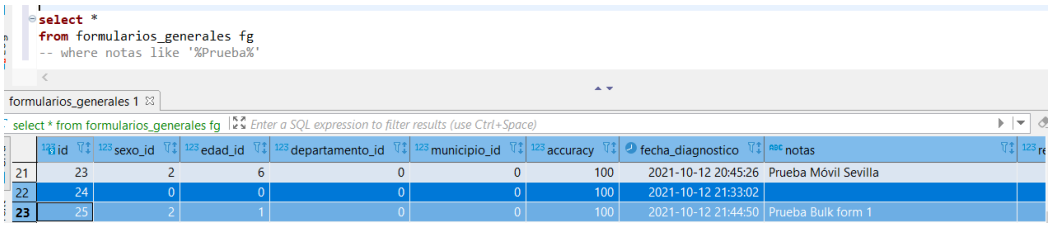
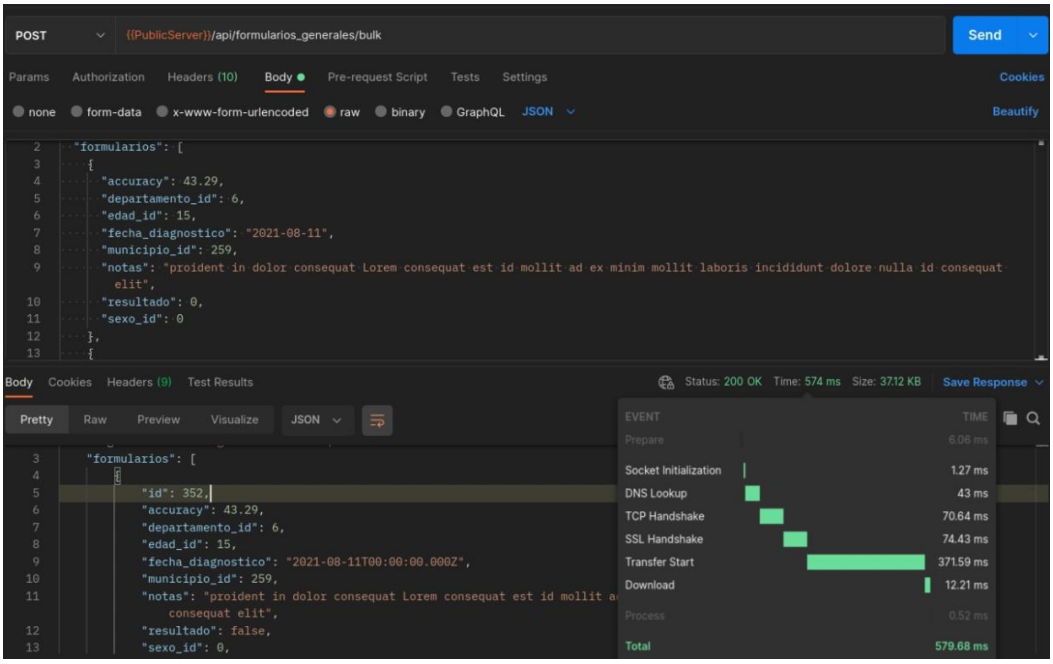


Ilustración 68: Tiempo de registro de cien formularios desde Postman.

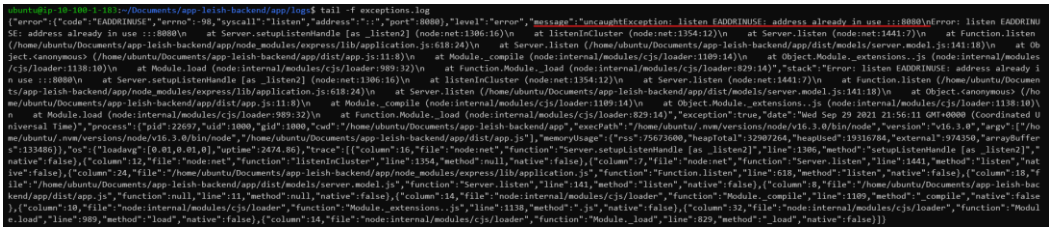


En la ilustración anterior se puede observar la carga de cien formularios en el servidor de la aplicación. El tiempo de carga fue de 574 ms como puede notarse.

d. Logging de errores y excepciones en servidor de aplicación

Las excepciones y errores manejados en los servidores con Express son casos en los que el sistema no puede manejar una solicitud.

Ilustración 69: Logging de excepciones servidor de aplicación.



A través del comando ***tail -f archivo.log***, puede observarse el registro de un error causado al intentar iniciar el servidor en el puerto 8080. Este error sucedió porque el Backend del servidor en Express ya estaba ejecutándose en dicho puerto.

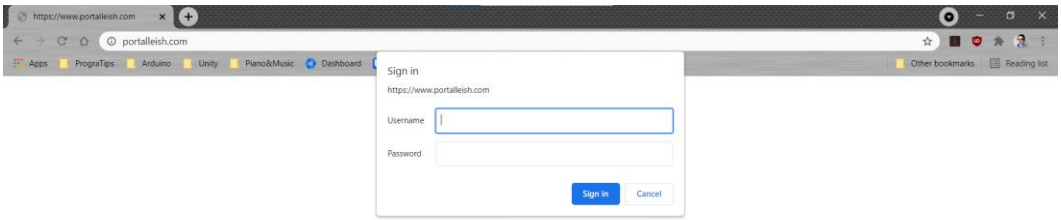
Ilustración 70: Logging de errores servidor de aplicación.

```
ubuntu@ip-10-100-1-183:~/Documents/app-leish-backend/app$ tail -f logs/error.log
{"message":"[2021-09-30T03:08:12.498Z] [ERROR] [FORMULARIOS] Error, por favor contacte al personal.obj: SequelizeConnectionError: connect ETIMEDOUT","level":"er
ror"}
{"message":"[2021-10-04T06:14:03.628Z] [ERROR] [DB-CONNECTION] SequelizeConnectionError: connect ETIMEDOUT","level":"error"}
```

En este caso se observa un error relacionado a la conexión con la base de datos en RDS. Este error nos indica que la base de datos no estaba levantada al momento de levantar el servicio de Express.

3. Portal web estadístico

Ilustración 71: Autenticación HTTP básica para portal web estadístico en S3.



En esta parte se observa el desplegable de autenticación HTTP básica para el sitio web configurado con una función lambda.

a. Inicio de servidor con backend del portal web

Ilustración 72: Inicio de servidor con backend del portal web.

```
{
  db: {
    host: 'leish-db-micro.cqlccgb9wvhk.us-east-2.rds.amazonaws.com',
    name: 'leish_db',
    user: 'admin',
    password: 'z3X9McQxQqmGcj',
    port: '3306'
  },
  server: {
    hostname: 'localhost',
    port: '8080',
    allowed_origins: 'https://www.portalleish.com',
    token: {
      expireTime: 3600,
      issuer: 'leishleish',
      secret: '8202d6f19be98614e3d3db13b494ee1b510961a99f966bc04080d6e1cc801c895351bd58a49b3a3029093f8e44649077bf430d8c9c5a4ae6bfb50c639009c542'
    },
    httpauth: { username: 'leishleish', pass: 'Leishmaniasis100%' },
    user_roles: { roles: [Object], roles_: [Object] }
  }
}
[2021-10-15T3:43:54] [DEBUG] [Server] Server is running localhost:8080
[2021-10-15T3:43:54] [SEQUELIZE] Executing (default): SELECT 1+1 AS result
Connected to database
[2021-10-15T3:43:54] [DEBUG] [Server] METHOD: [GET] - URL: [/] - IP: [::ffff:10.100.0.221]
[2021-10-15T3:43:54] [DEBUG] [Server] METHOD: [GET] - URL: [/] - STATUS: [401] - IP: [::ffff:10.100.0.221]
[2021-10-15T3:43:54] [DEBUG] [Server] METHOD: [GET] - URL: [/] - IP: [::ffff:10.100.3.154]
[2021-10-15T3:43:54] [DEBUG] [Server] METHOD: [GET] - URL: [/] - STATUS: [401] - IP: [::ffff:10.100.3.154]
```

En esta imagen se puede observar el inicio de Express en el servidor con el Backend del portal web. Se observan varios objetos como **db**, **server**, **httpauth** y **user_roles**. Todos con las configuraciones correspondientes para que el servidor realice las validaciones

correspondientes de acuerdo con las conexiones a base de datos, accesos a la información, verificaciones, entre otros.

b. Manejo de usuarios

Ilustración 73: Inicio de sesión.

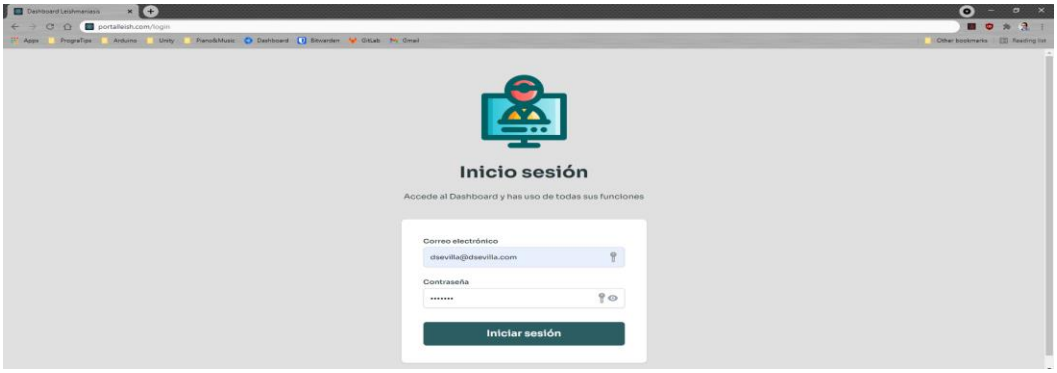


Ilustración 74: Logging de acceso (Inserción) – inicio de sesión.

Ilustración 75: Logs de Sequelize – inicio de sesión.

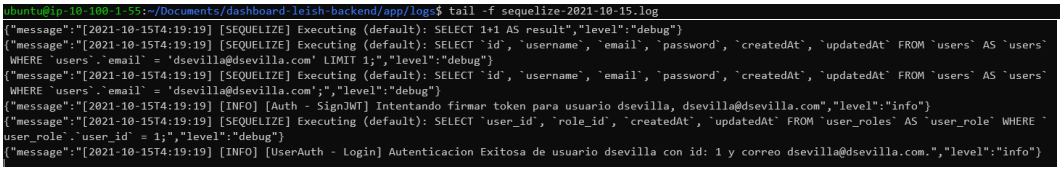
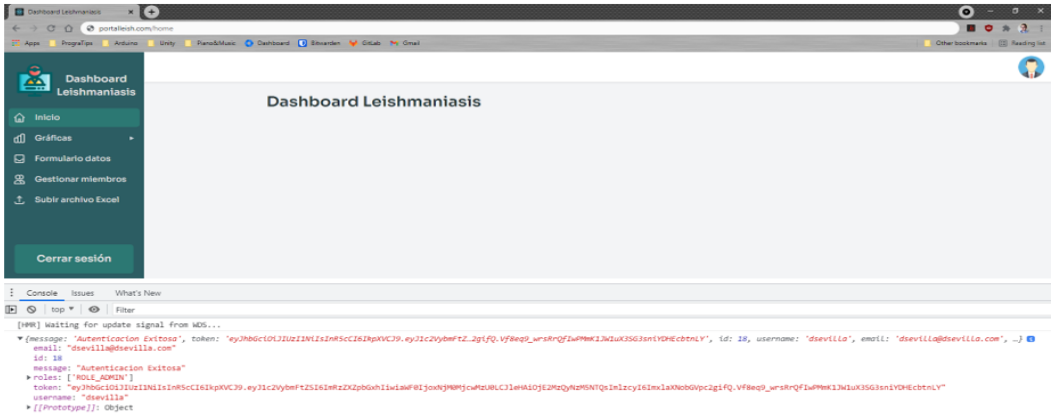


Ilustración 76: Redirección desde la página de inicio de sesión.



En esta imagen podemos observar al usuario dsevilla con rol de administrador autenticado exitosamente. Se observa su token correspondiente.

Ilustración 77: Registrar nuevo usuario.

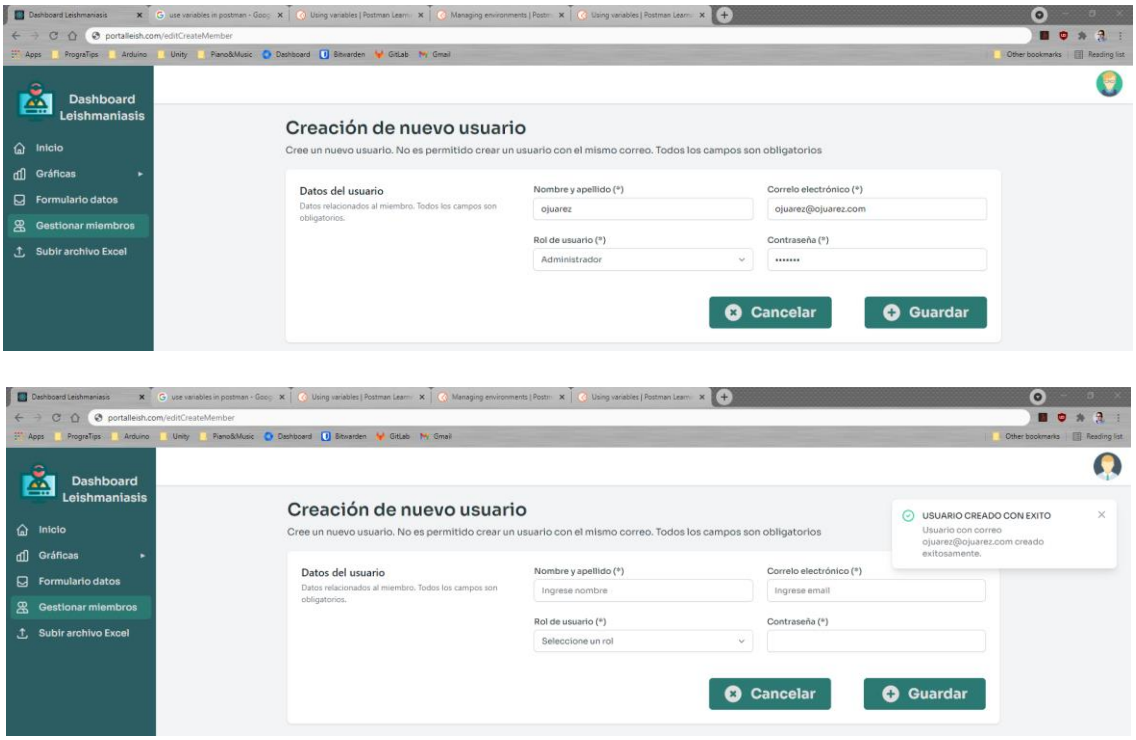


Ilustración 78: Logging de acceso (Inserción) – registro de usuario.

```
ubuntu@ip-10-100-1-55:~/Documents/dashboard-leish-backend/app$ tail -f logs/access-2021-10-15.log
{"message":"[2021-10-15T4:34:40] [INFO] [Auth - verifyJWT] Validating Token","level":"info"}
{"message":"[2021-10-15T4:34:40] [INFO] [Auth - verifyJWT] [object Object]","level":"info"}
{"message":"[2021-10-15T4:34:40] [INFO] [UserAuth - Register] Usuario con id 2 y correo ojuarez@ojuarez.com registrado","level":"info"}
```

Ilustración 79: Logging de base de datos (Sequelize) – registro de usuario.

```
ubuntu@ip-10-100-1-55:~/Documents/dashboard-leish-backend/app$ tail -f logs/sequelize-2021-10-15.log
{"message":"[2021-10-15T4:34:40] [INFO] [Auth - verifyJWT] [object Object]","level":"info"}
{"message":"[2021-10-15T4:34:40] [SEQUELIZE] Executing (default): SELECT `id`, `username`, `email`, `password`, `createdAt`, `updatedAt` FROM `users` AS `users`
 WHERE `users`.`username` = 'ojuarez' LIMIT 1;","level":"debug"}
{"message":"[2021-10-15T4:34:40] [SEQUELIZE] Executing (default): SELECT `id`, `username`, `email`, `password`, `createdAt`, `updatedAt` FROM `users` AS `users`
 WHERE `users`.`email` = 'ojuarez@ojuarez.com' LIMIT 1;","level":"debug"}
{"message":"[2021-10-15T4:34:40] [SEQUELIZE] Executing (default): SELECT `id`, `username`, `email`, `password`, `createdAt`, `updatedAt` FROM `users` AS `users`
 WHERE `users`.`email` = 'ojuarez@ojuarez.com';","level":"debug"}
{"message":"[2021-10-15T4:34:40] [SEQUELIZE] Executing (default): INSERT INTO `users` (`id`,`username`,`email`,`password`,`createdAt`,`updatedAt`) VALUES (DEFAU
LT,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T4:34:40] [INFO] [UserAuth - Register] Usuario con id 2 y correo ojuarez@ojuarez.com registrado","level":"info"}
{"message":"[2021-10-15T4:34:40] [SEQUELIZE] Executing (default): INSERT INTO `user_roles` (`user_id`,`role_id`,`createdAt`,`updatedAt`) VALUES (?,?,?,?);","lev
el":"debug"}
```

Ilustración 80: Visualización de nuevo usuario en tablero.

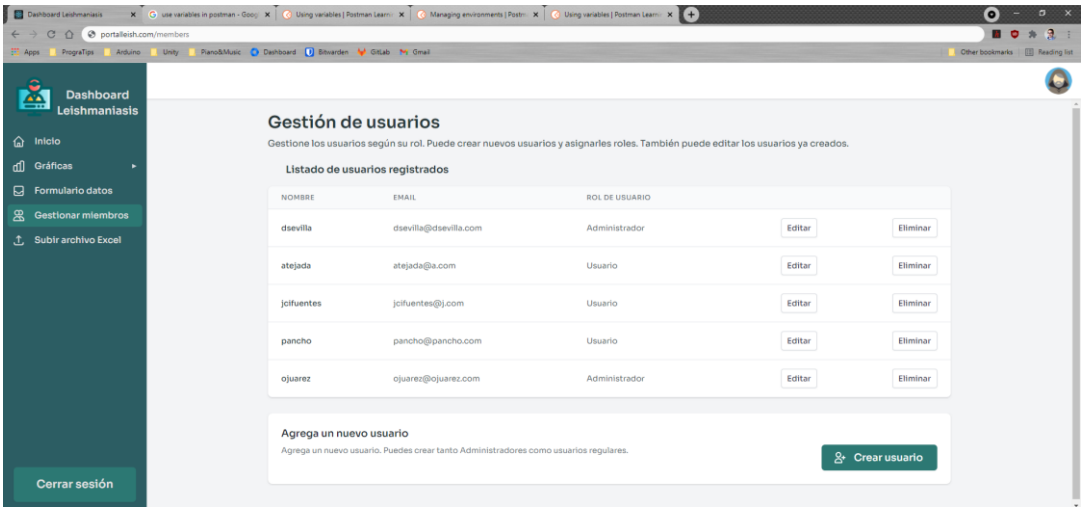
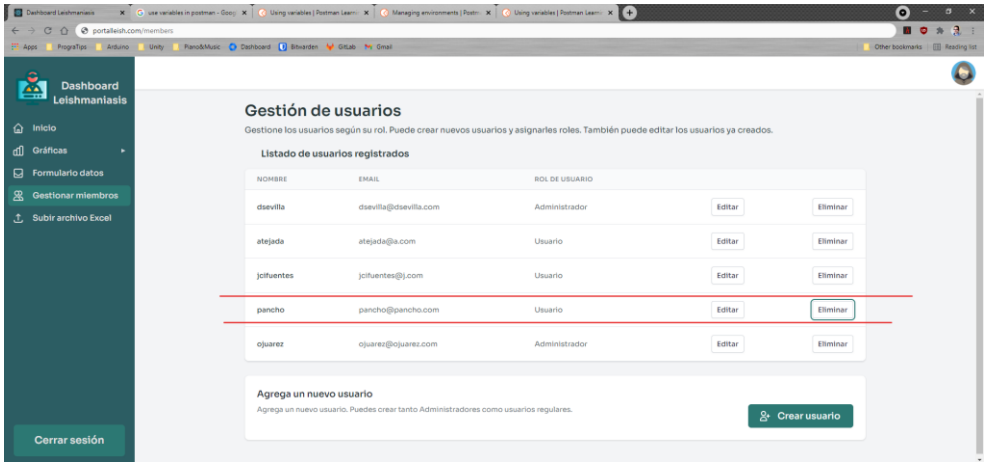


Ilustración 81: Eliminar usuario.



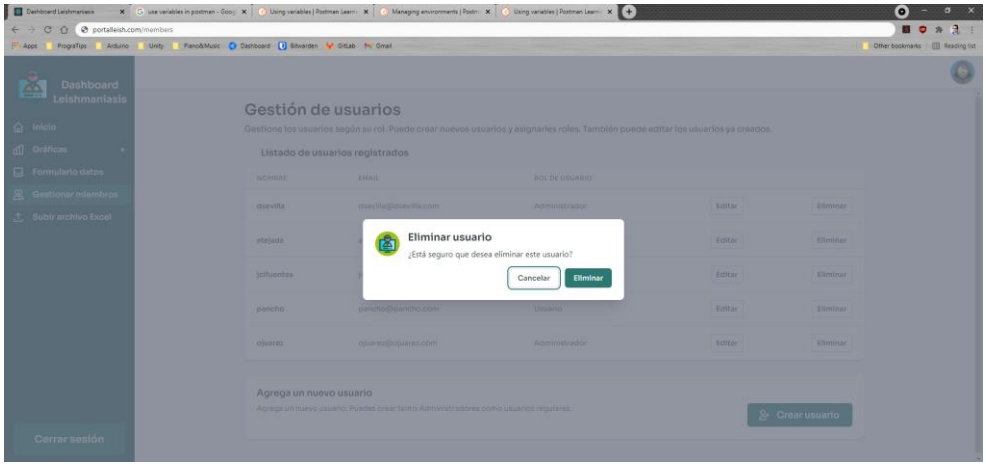


Ilustración 82: Logging de acceso (Inserción) – eliminar usuario.

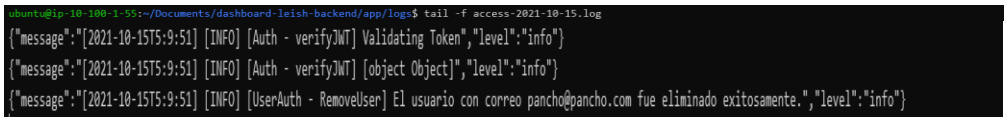


Ilustración 83: Logging de base de datos (Sequelize) – eliminar usuario.

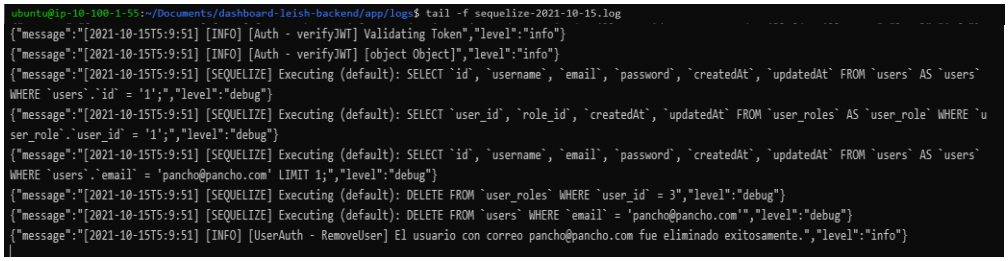


Ilustración 84: Usuario eliminado en tablero.

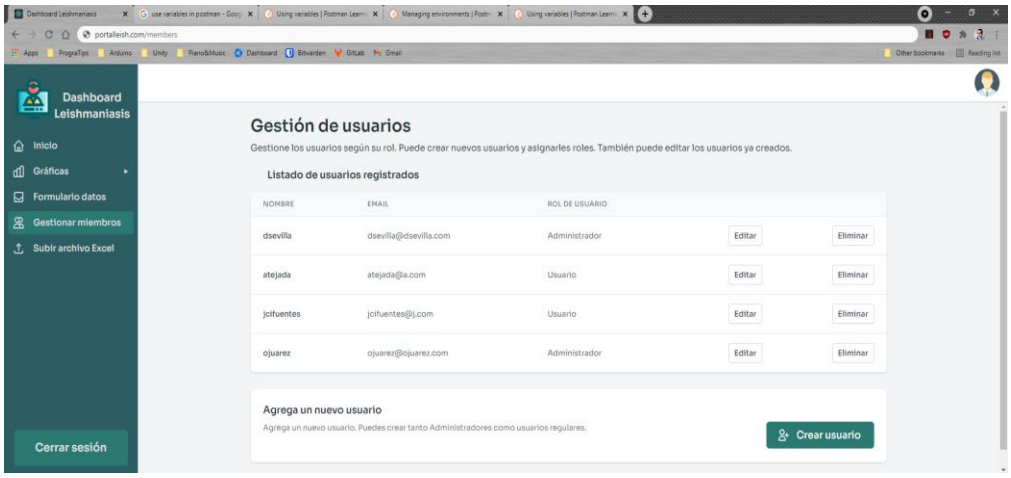


Ilustración 85: Modificar usuario.

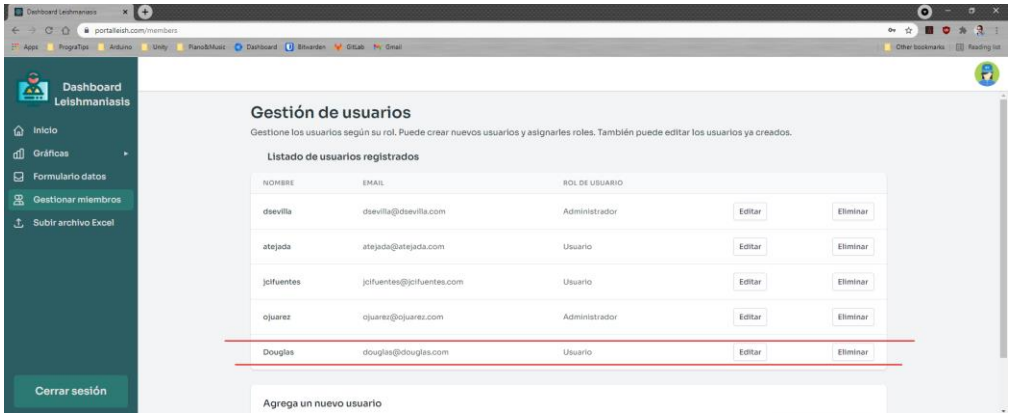
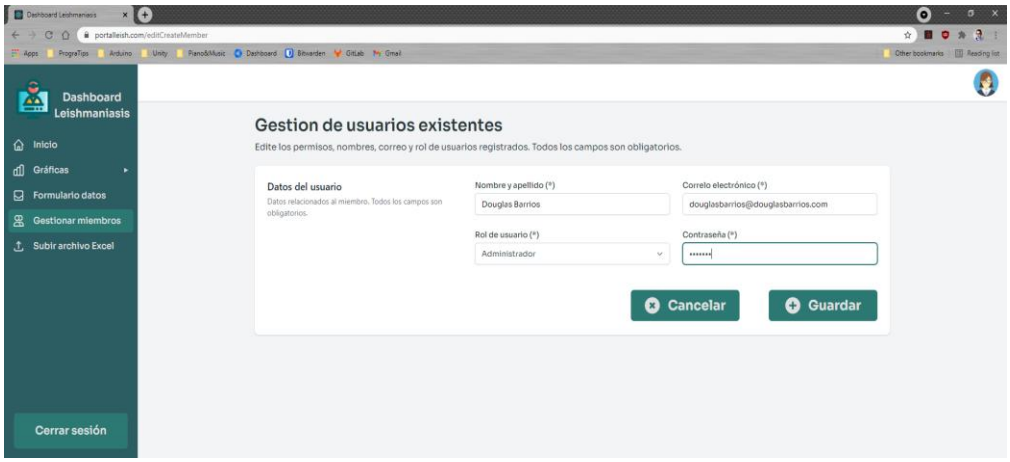


Ilustración 86: Edición de usuarios.



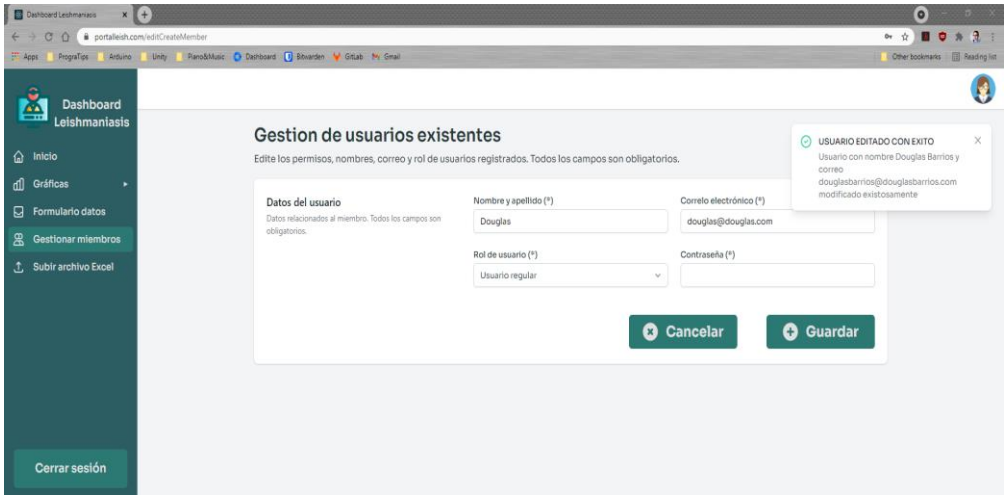


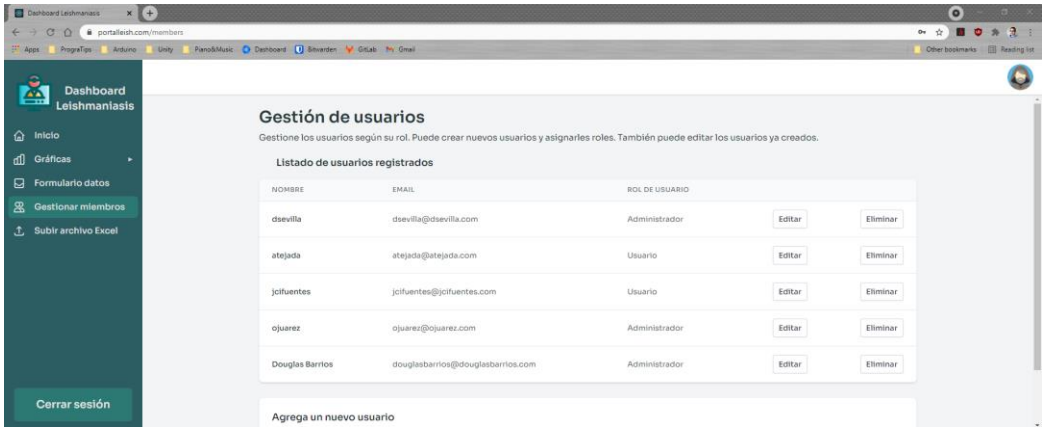
Ilustración 87: Logging de acceso (Inserción) – modificar usuario.

```
ubuntu@ip-10-100-1-55:~/Documents/dashboard-leish-backend/app/logs$ tail -f access-2021-10-15.log
{"message":"[2021-10-15T5:9:51] [INFO] [Auth - verifyJWT] Validating Token","level":"info"}
{"message":"[2021-10-15T5:9:51] [INFO] [Auth - verifyJWT] [object Object]","level":"info"}
{"message":"[2021-10-15T5:9:51] [INFO] [UserAuth - RemoveUser] Usuario con nombre Douglas Barrios y correo douglasbarrios@douglasbarrios.com modificado existosa
mente","level":"info"}
```

Ilustración 88: Logging de base de datos (Sequelize) – modificar usuario.

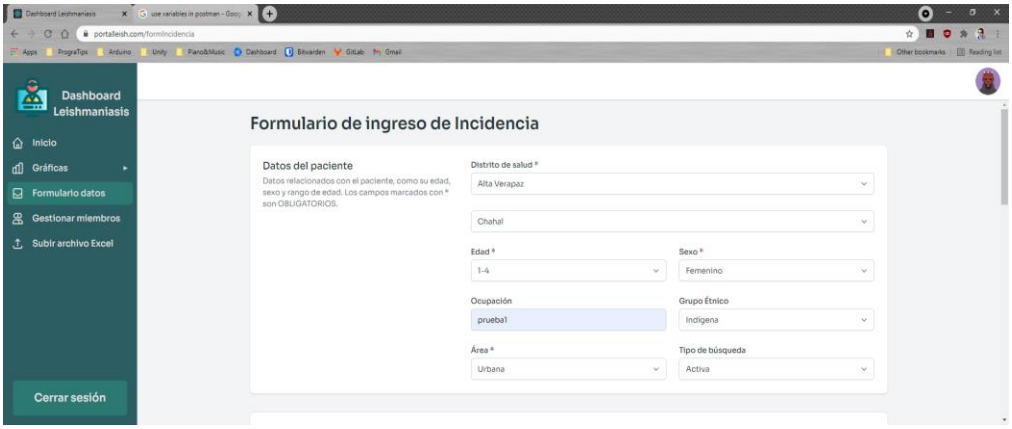
```
ubuntu@ip-10-100-1-55:~/Documents/dashboard-leish-backend/app/logs$ tail -f sequelize-2021-10-15.log
{"message":"[2021-10-15T5:9:51] [INFO] [Auth - verifyJWT] Validating Token","level":"info"}
{"message":"[2021-10-15T5:9:51] [INFO] [Auth - verifyJWT] [object Object]","level":"info"}
{"message":"[2021-10-15T5:9:51] [SEQUELIZE] Executing (default): SELECT `id`, `username`, `email`, `password`, `createdAt`, `updatedAt` FROM `users` AS `users`
WHERE `users`.`id` = '1';","level":"debug"}
{"message":"[2021-10-15T5:9:51] [SEQUELIZE] Executing (default): SELECT `user_id`, `role_id`, `createdAt`, `updatedAt` FROM `user_roles` AS `user_role` WHERE `u
ser_role`.`user_id` = '1';","level":"debug"}
{"message":"[2021-10-15T5:9:51] [SEQUELIZE] Executing (default): SELECT `id`, `username`, `email`, `password`, `createdAt`, `updatedAt` FROM `users` AS `users`
WHERE `users`.`id` = 8;","level":"debug"}
{"message":"[2021-10-15T5:9:51] [SEQUELIZE] Executing (default): DELETE FROM `user_roles` WHERE `user_id` = 8","level":"debug"}
{"message":"[2021-10-15T5:9:51] [SEQUELIZE] Executing (default): INSERT INTO `user_roles` (`user_id`,`role_id`,`createdAt`,`updatedAt`) VALUES (?,?,?,?);","leve
l":"debug"}
{"message":"[2021-10-15T5:9:51] [SEQUELIZE] Executing (default): UPDATE `users` SET `username`=?,`email`=?,`password`=?,`updatedAt`=? WHERE `id` = ?","level":"d
ebug"}
{"message":"[2021-10-15T5:9:51] [INFO] [UserAuth - RemoveUser] Usuario con nombre Douglas Barrios y correo douglasbarrios@douglasbarrios.com modificado existosa
mente","level":"info"}
```

Ilustración 89: Visualización de usuario modificado en tablero.



c. Carga de formularios

Ilustración 90: Carga de formulario.



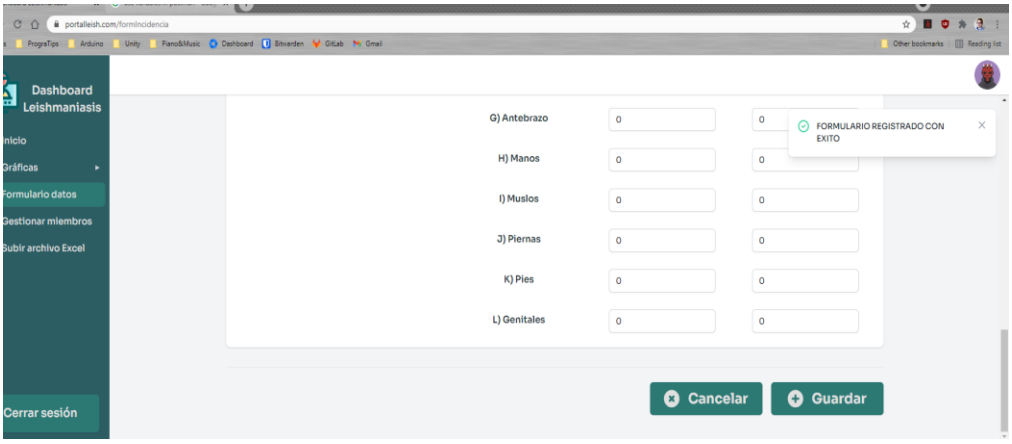


Ilustración 91: Logs de acceso (Inserción) – carga de formularios.

```
ubuntu@ip-10-100-1-55:~/Documents/dashboard-leish-backend/app/logs$ tail -f access-2021-10-15.log
{"message":"[2021-10-15T17:43:56] [INFO] [Auth - verifyJWT] Validating Token","level":"info"}
{"message":"[2021-10-15T17:43:56] [INFO] [Auth - verifyJWT] [object Object]","level":"info"}
```

Ilustración 92: Logs de base de datos (Sequelize) – carga de formularios.

```
ubuntu@ip-10-100-1-55:~/Documents/dashboard-leish-backend/app/logs$ tail -f sequelize-2021-10-15.log
{"message":"[2021-10-15T17:43:56] [INFO] [Auth - verifyJWT] Validating Token","level":"info"}
{"message":"[2021-10-15T17:43:56] [INFO] [Auth - verifyJWT] [object Object]","level":"info"}
{"message":"[2021-10-15T17:43:56] [SEQUELIZE] Executing (a1ef66f0-ab3a-4357-8477-db44b859a6ca): START TRANSACTION;","level":"debug"}
{"message":"[2021-10-15T17:43:56] [SEQUELIZE] Executing (a1ef66f0-ab3a-4357-8477-db44b859a6ca): INSERT INTO `procedencias` (`id`,`area_id`,`departamento_id`,`mu
nicipio_id`,`grupo_etnico_id`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T17:43:56] [SEQUELIZE] Executing (default): SELECT `id`, `distrito`, `departamento_id`, `municipio_id`, `createdAt`, `updatedAt` FROM `di
stritos_salud` AS `distritos_salud` WHERE `distritos_salud`.`municipio_id` = 1 LIMIT 1;","level":"debug"}
{"message":"[2021-10-15T17:43:56] [SEQUELIZE] Executing (a1ef66f0-ab3a-4357-8477-db44b859a6ca): INSERT INTO `diagnosticos` (`id`,`fecha`,`prueba_diagnostico_id`
,`tratamiento_id`,`egreso_id`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T17:43:56] [SEQUELIZE] Executing (a1ef66f0-ab3a-4357-8477-db44b859a6ca): INSERT INTO `caracteristicas_clinicas` (`id`,`diagnostico_id`,`p
arte_cuerpo_id`,`diametro_lesion`,`tiempo_evolucion`,`cantidad`,`createdAt`,`updatedAt`) VALUES (NULL,6,1,'1',1,1,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(
NULL,6,2,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(NULL,6,3,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(NULL,6,4,'0',1,0,'2021-10-15 17:48
:19','2021-10-15 17:48:19'),(NULL,6,5,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(NULL,6,6,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(NULL,
6,7,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(NULL,6,8,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(NULL,6,9,'0',1,0,'2021-10-15 17:48:19',
'2021-10-15 17:48:19'),(NULL,6,10,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(NULL,6,11,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19'),(NULL,6,1
2,'0',1,0,'2021-10-15 17:48:19','2021-10-15 17:48:19');","level":"debug"}
{"message":"[2021-10-15T17:43:56] [SEQUELIZE] Executing (a1ef66f0-ab3a-4357-8477-db44b859a6ca): INSERT INTO `lesiones_cicatrizales` (`id`,`diagnostico_id`,`part
e_cuerpo_id`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T17:43:56] [SEQUELIZE] Executing (a1ef66f0-ab3a-4357-8477-db44b859a6ca): INSERT INTO `pacientes` (`id`,`sexo_id`,`edad_id`,`procedencia_i
d`,`diagnostico_id`,`ocupacion`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T17:43:56] [SEQUELIZE] Executing (a1ef66f0-ab3a-4357-8477-db44b859a6ca): COMMIT;","level":"debug"}
```

d. Carga y descarga de archivos XLSX

Ilustración 93: Carga de archivo.

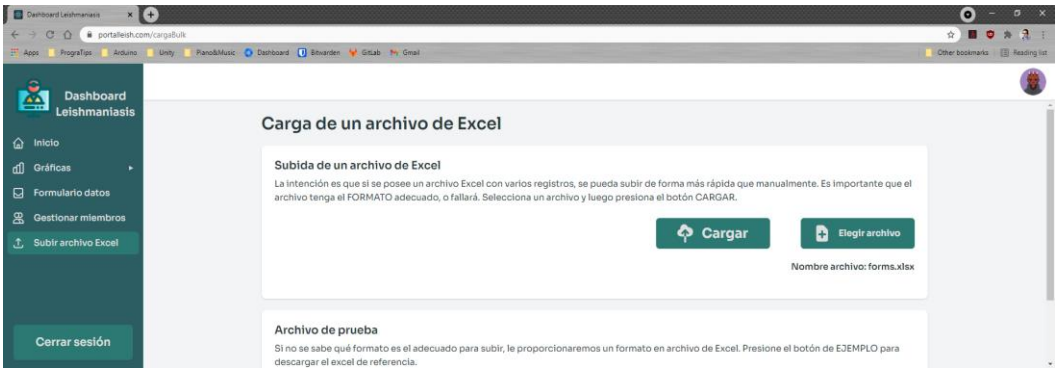


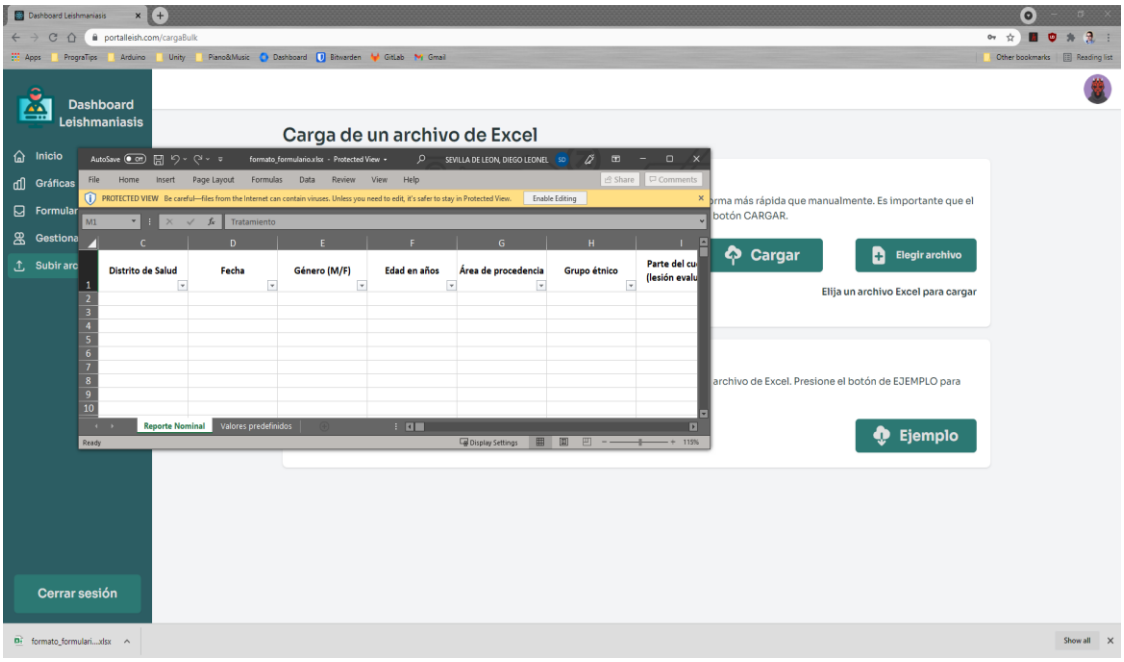
Ilustración 94: Logging de acceso (Inserción) – carga y descarga de archivos xlsx.

```
ubuntu@ip-10-100-1-55:~/Documents/dashboard-leish-backend/app/logs$ tail -f access-2021-10-15.log
{"message":"[2021-10-15T7:41:13] [INFO] [Auth - verifyJWT] Validating Token","level":"info"}
{"message":"[2021-10-15T7:41:13] [INFO] [Auth - verifyJWT] [object Object]","level":"info"}
```

Ilustración 95: Logging de base de datos (Sequelize) – carga y descarga de archivos xlsx.

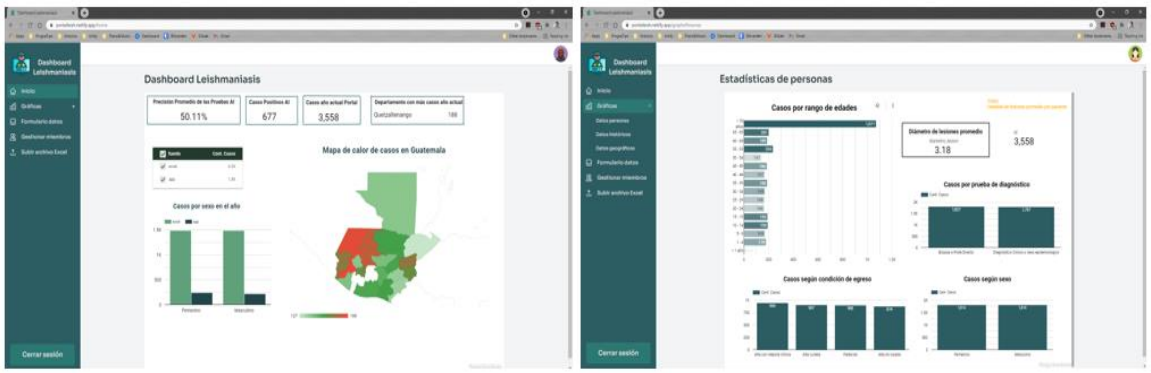
```
ubuntu@ip-10-100-1-55:~/Documents/dashboard-leish-backend/app/logs$ tail -f sequelize-2021-10-15.log
{"message":"[2021-10-15T7:41:13] [SEQUELIZE] Executing (default): SELECT `id`, `sexo`, `createdAt`, `updatedAt` FROM `sexo` AS `sexo` WHERE `sexo`.`sexo` LIKE '
%femenino%' LIMIT 1;","level":"debug"}
{"message":"[2021-10-15T7:41:13] [SEQUELIZE] Executing (76a6a2b9-7657-42bd-8356-4c692ca4e932): INSERT INTO `pacientes` (`id`,`sexo_id`,`edad_id`,`procedencia_id
`,`diagnostico_id`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T7:41:13] [SEQUELIZE] Executing (default): SELECT `id`, `sexo`, `createdAt`, `updatedAt` FROM `sexo` AS `sexo` WHERE `sexo`.`sexo` LIKE '
%masculino%' LIMIT 1;","level":"debug"}
{"message":"[2021-10-15T7:41:13] [SEQUELIZE] Executing (76a6a2b9-7657-42bd-8356-4c692ca4e932): INSERT INTO `pacientes` (`id`,`sexo_id`,`edad_id`,`procedencia_id
`,`diagnostico_id`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T7:41:13] [SEQUELIZE] Executing (default): SELECT `id`, `sexo`, `createdAt`, `updatedAt` FROM `sexo` AS `sexo` WHERE `sexo`.`sexo` LIKE '
%femenino%' LIMIT 1;","level":"debug"}
{"message":"[2021-10-15T7:41:13] [SEQUELIZE] Executing (76a6a2b9-7657-42bd-8356-4c692ca4e932): INSERT INTO `pacientes` (`id`,`sexo_id`,`edad_id`,`procedencia_id
`,`diagnostico_id`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T7:41:13] [SEQUELIZE] Executing (76a6a2b9-7657-42bd-8356-4c692ca4e932): INSERT INTO `pacientes` (`id`,`sexo_id`,`edad_id`,`procedencia_id
`,`diagnostico_id`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?);","level":"debug"}
{"message":"[2021-10-15T7:41:13] [SEQUELIZE] Executing (76a6a2b9-7657-42bd-8356-4c692ca4e932): COMMIT;","level":"debug"}
```

Ilustración 96: Descarga de archivo.



e. Visualización de gráficos – Google Data Studio

Ilustración 97: Visualización de gráficos – Google Data Studio.



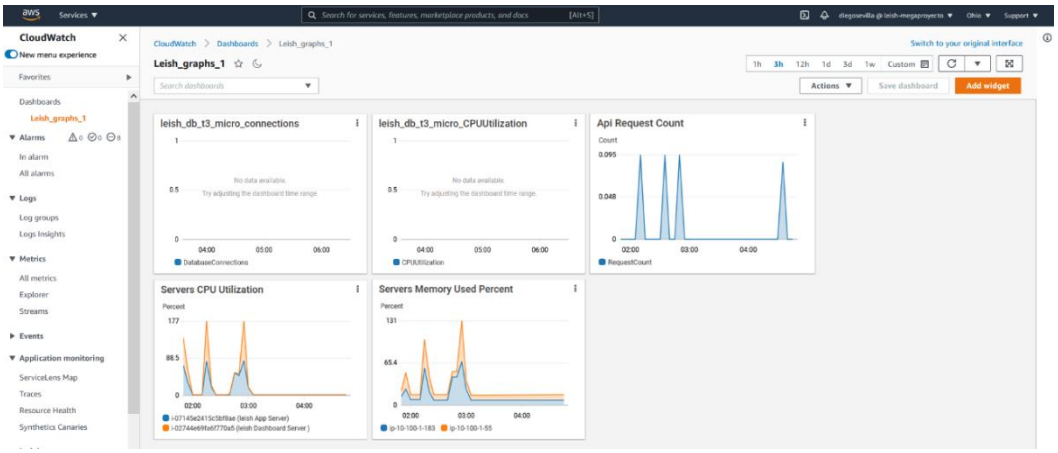
El portal web cuenta con 4 vistas para el despliegue de gráficos relacionados a datos demográficos del Leishmaniasis. Las vistas son las siguientes:

- **Inicio:** se muestra un resumen estadístico con casos positivos, departamento con más casos, mapa de calor, entre otros.
- **Datos personas:** se muestran estadísticas de personas como casos por rango de edades, sexo, el diámetro promedio de las lesiones, entre otros.
- **Datos históricos:** se muestran estadísticas históricas como conteos de casos según sexo, departamento, la letalidad a lo largo del tiempo, entre otros.
- **Datos geográficos:** se muestran estadísticas geográficas de la enfermedad por departamento y municipios.

4. Monitoreo de los servicios

a. Portal de métricas varias en CloudWatch

Ilustración 98: Pruebas de memoria y uso de CPU en Servidores.



Se cuenta con dos gráficas importantes, una para medición de uso de memoria y otra para medición del uso del CPU. La tercera gráfica, *API Request Count*, únicamente muestra consultas realizadas al API en el tiempo. Se usó una utilidad de linux llamada Stress, que básicamente genera procesos para exigir un uso del 100% en el CPU y la memoria dependiendo de lo que se especifique en la línea de comandos.

- *Servers CPU Utilization:* muestra tres pruebas cortas de un minuto donde se exige el CPU al 100%.
- *Memory Used Percent:* muestra tres pruebas cortas de un minuto donde se exige la memoria RAM al 100%.

Ambas pruebas se realizaron para verificar el funcionamiento de alarmas configuradas en CloudWatch.

Ilustración 99: Alarmas en correo configurado.

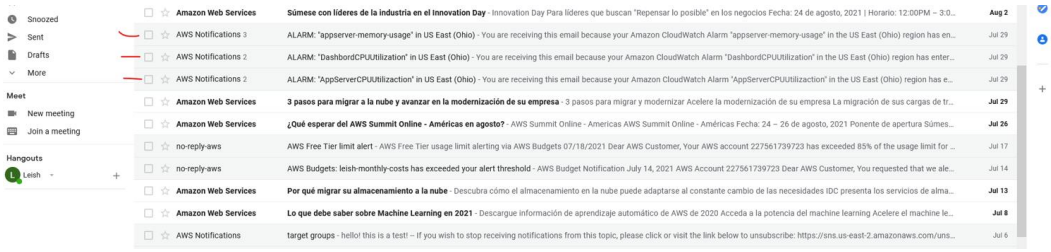
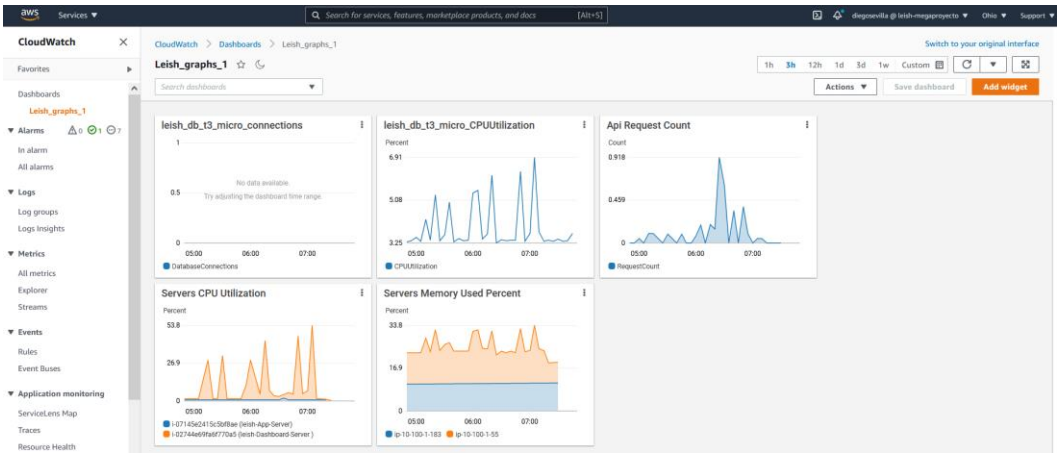


Ilustración 100: Carga masiva de formularios y archivos XLSX.



En esta Ilustración se pueden observar cuatro gráficas. Estas visualizaciones fueron generadas a partir de pruebas en el API del portal web, al cargar registros generados aleatoriamente que serán insertados en producción al cargar archivos de Excel. Las gráficas mostradas en el tablero digital son:

- *Leish_db_t3_micro_CPUUtilization:* muestra el porcentaje de CPU utilizado de la instancia en RDS.
- *Api Request Count:* muestra las consultas al API realizadas en el tiempo.
- *Servers CPU Utilization:* muestra el porcentaje de uso de CPU de ambos servidores.
- *Servers Memory Used Percent:* muestra el porcentaje de uso de memoria RAM de ambos servidores.

5. Calendarizaciones y monitoreo

a. Inicio y apagado de instancias EC2

Ilustración 101: Inicio del servidor de la aplicación por función lambda en CloudWatch.

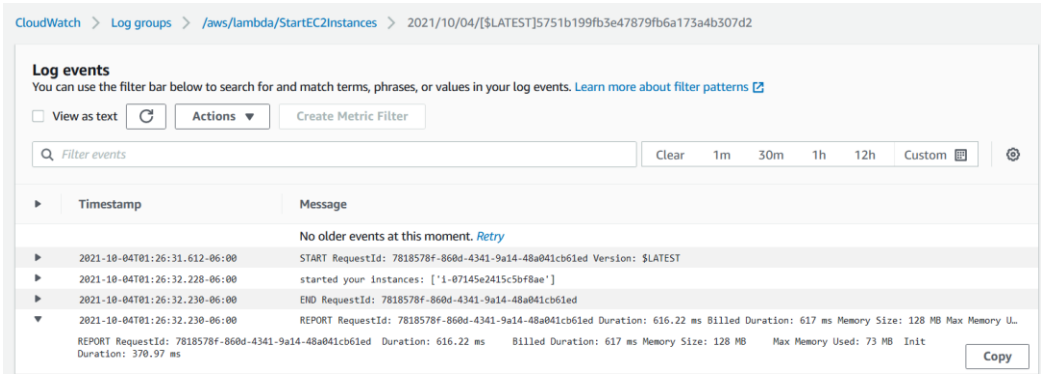
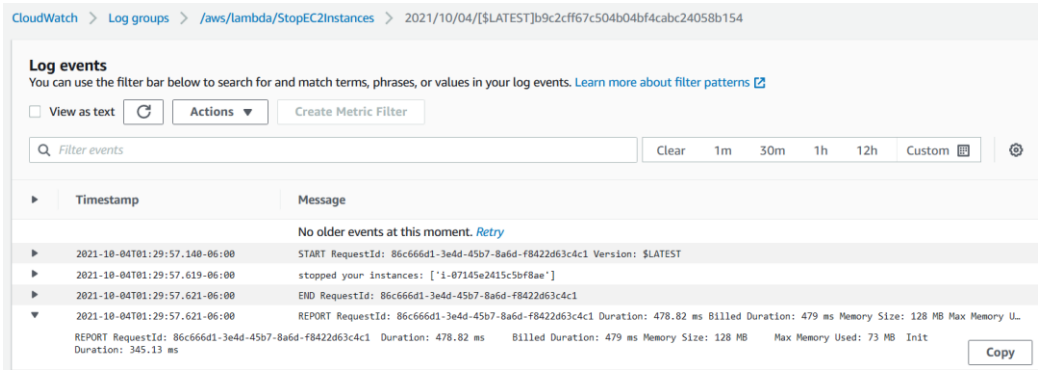


Ilustración 102: Apagado del servidor de la aplicación por función lambda en CloudWatch.

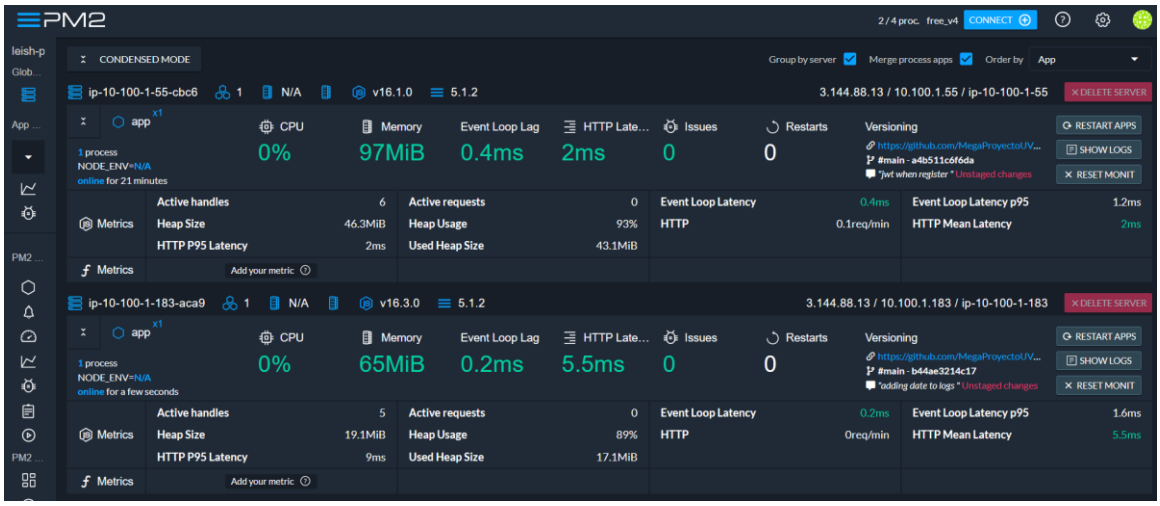


b. Inicio automatizado de Express en instancias EC2

Se generaron scripts de inicio para asegurar el inicio de las aplicaciones al momento de reiniciar las instancias. Esta tarea se realizó con el paquete pm2 para mantenimiento de ambientes de producción. En esencia, estos scripts se ejecutarían el momento que una instancia fuera encendida, dando inicio al API desarrollado con Node.js Express.

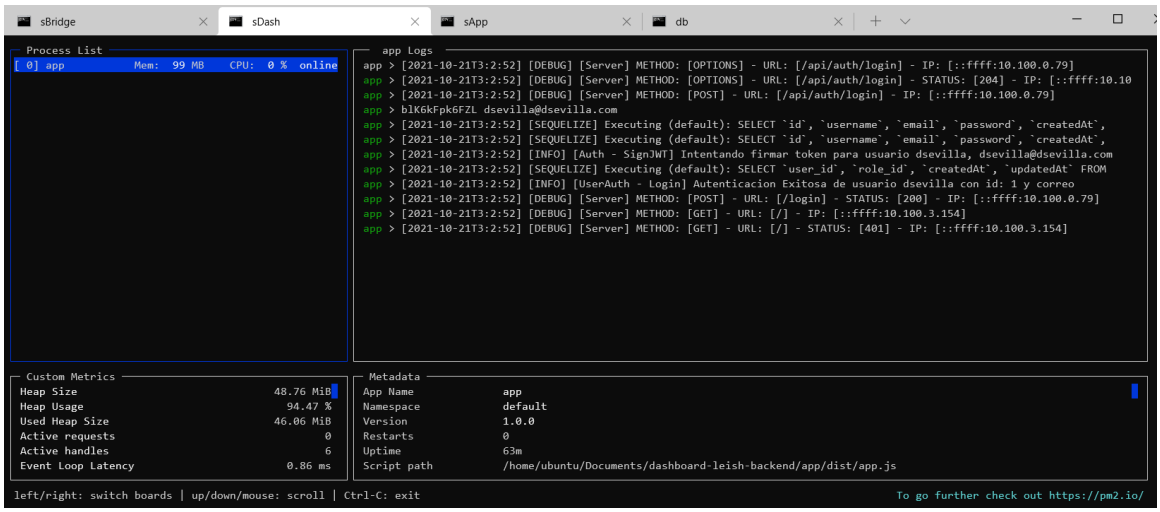
c. Monitoreo de Express desde PM2

Ilustración 103: Portal web de PM2.



En esta parte se observa la visualización de los servidores de aplicación monitoreados por el servicio de monitoreo de PM2 desde un explorador. El servicio funciona en tiempo real, permite ver el Logging de la aplicación, errores que puedan presentarse, el tiempo que las aplicaciones llevan ejecutándose, entre otros. Cuenta también con un servicio de notificaciones por correo automático, que alerta sobre errores durante la ejecución. También proporciona información sobre el sistema operativo, como la memoria y el CPU utilizados por el proceso de Node.js.

Ilustración 104: Login visto en consola de PM2 en servidor de portal web.



6. Tiempo de respuesta de servicios

Tabla 18: Tiempos de respuesta de endpoints.

Servicio	Endpoint	Descripción	Tiempo de respuesta
Aplicación	api/formularios_generales	Carga de formulario simple	< 200 ms ≈ 0.5s
	api/formularios_generales/bulk	Carga de formularios en bulk	573 ms $\approx$ 1s
Portal Web	api/formulario	Carga de formulario	$1 - 2\ s$
	api/excel	Carga de archivo xlsx	$1 - 2\ s \sim variable$
	api/excel/downloadFile	Descarga de archivo xlsx	$0.5 - 1\ s$
	api/auth/login	Inicio de sesión de usuario	$1 - 2\ s$
	api/auth/register	Registro de usuario	$1 - 2\ s$
	api/users/all_users	Consulta de los usuarios	$1 - 2\ s$
	api/users/remove_user	Eliminar un usuario	$1 - 2\ s$
	api/users/edit_user	Editar un usuario	$1 - 2\ s$

En la Tabla 18 se aprecian los tiempos de respuesta de cada uno de los endpoints proporcionados por los sistemas de Backend desarrollados para el proyecto.

En el caso de la aplicación se tienen dos endpoints principales, uno para la carga de un solo formulario y otro para la carga de formularios en conglomerado. Los tiempos en milisegundos son datos proporcionados por la aplicación de Postman. En la práctica los tiempos pueden tardar un poco mas ya sea por el rendimiento del dispositivo o el framework de desarrollo utilizado para ejecutar las consultas HTTP en internet. Por tal motivo los tiempos se redondean a 0.5 segundos y 1 segundo.

En el caso del portal web se contaban con ocho endpoints. El tiempo promedio de respuesta para todos los endpoints fue de aproximadamente un segundo, exceptuando la carga del archivo xlsx. El archivo xlsx podía contar con n cantidad de filas por lo cual el tiempo para su carga a la base de datos era variable.

E. Frontend

1. App Leishmaniasis

Los resultados presentados a continuación son la toma de retroalimentación de los prototipos realizados con la aplicación durante su desarrollo. Por qué algunos son de carácter cualitativo y su relevancia se discutirán posteriormente.

a. Entrevista #1: Rosa Tejada

La entrevista fue realizada a un Stakeholder que estaba ligado al proyecto, pero que no sabía de su desarrollo. La retroalimentación, dudas y sugerencias se resumen en la siguiente minuta:

- "¿Qué hace el proyecto pues? Ok, ya me recordé. Esta bonito. Recuerdo que me habías contado de ese proyecto hace unas semanas. ¿Si lograron terminarlo? Veo que hay tres botones. Me gustan los dibujitos."
- Yo: ¿qué piensas de los colores?
- Ella: Siempre te ha gustado el verde, pero entiendo que si es verde hay muchas cosas que deben hacer ellos con ese color, ¿verdad?
- Yo: Sí. Bastante. ¿Entiendes para qué son los botones? ¿te confunden?
- Ella: Entiendo este que dice analizar, es el más grande. Pero los de abajo me confunde el que dice: Sincronizar. Tal vez el ícono.
- Yo: ok vale. ¿Ese que dice Manual de usuario le entiendes?
- Ella: Sí, es como un manual que traen los electrodomésticos.
- Yo: Sí, justo eso. ¿Por qué no probas los botones, navegas y todo?
- Ella: claro
- *En el proceso, navegó por todas las pantallas. Las preguntas más relevantes fueron.*
- Ella: ¿Qué hago acá? (¿pantalla de analizar lesión?
- Yo: ahí te indica el porcentaje de la lesión y si es negativo o no.
- Ella: Sí, estos botones me gustan el color, pero el texto quizás muy grande.
- Yo: ok. Presiona alguno
- Ella: *presiona el botón de agregar datos a una lesión mediante un formulario.*
- Yo: vale que piensas de este formulario
- Ella: esta simple, son apenas unos cuantos cuadritos. ¿están todos los departamentos y municipios acá?
- Yo: intenta
- Ella: *Intenta ver si están todos.*
- Yo: dale sigue
- Ella: *Presiona el botón de sincronizar y sale la animación de cargando*
- Ella: Me gusta la animación.
- Yo: Claro, le puse mucho esfuerzo.

b. Entrevista #2: Josselyn Rodas

La entrevista fue realizada a un Stakeholder que no estaba ligado de ninguna manera al proyecto. La retroalimentación, dudas y sugerencias se resumen en la siguiente minuta.

- Yo: Presento el proyecto, ventajas, lo que estamos buscando, que haremos tanto una app como un portal para los datos.
- Ella: Vale, entiendo.
- Yo: Muy bien, esta es la app. Le mando un link de Adobe XD donde puede probar el flujo del prototipo.
- Ella: me gusta el color.
- Yo: gracias. Navega un poco por la aplicación, si algo no te hace sentido, dime.
- Ella: Navega unos dos minutos. Completa varios flujos.
- Yo: ¿Cómo la ves?
- Ella: Veo que no es mucho. Entiendo que no es tan sencillo como se ve, algo normal, pero no veo que sea mucho.
- Yo: Justo, tienes razón. El valor se lo damos mediante otras cosas. Como el dashboard o el que tome bien la lesión.
- Ella: ¿Qué es un dashboard?
- Yo: Procedo a explicarle sobre el Dashboard/tablero
- Ella: Los íconos me gustan. ¿Cuáles les pondrás en la aplicación final?
- Yo: estoy decidiendo. ¿Pero te gustaría verlo luego?
- Ella: ok
- Yo: gracias por tu tiempo

Tabla 19: Retroalimentación de App Leishmaniasis

No.	Pantalla/Flujo o función	Objetivo de enfoque	Retroalimentación obtenida
1	Página de inicio	Objetivo específico (**a)**	Buenos colores, iconos entendibles. Mejorar quitar las tarjetas de internet en vez de dos solamente una. También más altura para los botones.
2	Tarjetas de funcionalidad	Objetivo específico (**a)**	Más altura
3	Manual de usuario	Objetivo específico (**a)**	Más preguntas, más secciones que puedan tener dudas. Cubrir preguntas generales, aquellas relacionadas con el análisis, dividirlas en temas.
4	UI de pantalla de inicio	Objetivo específico (**a)**	Bonitos colores.
5	Navegación general entre pantallas	Objetivo específico (**a)**	Fluido. No se trabó, en algunas ocasiones parecía trabarse, pero era solamente la percepción visual de la persona.

c. Prueba de usabilidad

Una prueba de usabilidad fue realizada para conocer si la aplicación seria de utilidad para los usuarios finales. Debido al carácter cualitativo de la prueba, está redactado narrando los hechos desde la perspectiva del entrevistador.

Entrevistador: Alejandro Tejada

Resultados prueba de usabilidad

Introducción

1. Para dar contexto:
    a. Hola, estamos trabajando en un proyecto relacionad con Leishmaniasis Cutánea en Guatemala, como bien sabrás, es una enfermedad endémica. Estamos desarrollando una app que detecte Leishmaniasis cutánea y un tablero donde esa información pueda ser vista.
    b. Estamos desarrollando esta entrevista para obtener información cualitativa y saber si será útil para ti en el futuro, ¡gracias!
2. Antes que entremos de lleno en las pruebas:
    a. Creo que deberíamos conocernos unos minutos, ¿qué te parece? (Ambos se presentan, nombres, edades, tiempo trabajando, etc.)
3. Se inicia el marco de la prueba o setup

Marco de prueba

1. Por favor, abre el siguiente link donde está el prototipo de la app (*se le proporciona el link del prototipo visual de la App hecho en adobe XD)* y espera ahí en la primera pantalla.
2. ¿Podrías compartirme tu pantalla por favor?
    a. Siéntete libre de hacerlo cuando quieras.
3. Gracias por compartir, ahora te diré lo que haremos. Tengo algunas peticiones que hacerte, unas pequeñas tareas que tendrás que hacer siguiendo el diseño, mantendré silencio en algunas partes para NO interferir en tus interacciones, pero no significa que no puedas decir en voz alta los pasos que seguiste, eso ayudará a que sepamos si te será de utilidad.
4. Muy bien, ¡Empecemos!

Test o prueba

1. **Contexto**:
    a. *Eres un evaluador del MSPAS o un usuario final con la aplicación instalada en tu teléfono. Estas en el interior del país, y ante ti hay una fila de personas intentando ser analizadas por posible Leishmaniasis cutánea. Ahora, es tu turno de tomar análisis, ver resultados y comentarles a ellos que pasó.*
2. **Tareas para realizar:**

a. **¿Cómo tomarías una lesión de una persona que viene en la fila?**

Flujo	Notas/complicaciones/dudas	Tiempo por flujo
Analizar una lesión presionando el botón de análisis de lesión.	• Se le dijo al usuario que el modelo tardaría un tiempo en cargar, por eso, aunado al tiempo del flujo, se le sumó la carga del modelo y la carga a la nube del resultado. • Les pareció bien que el botón tuviera grandes proporciones, de esa forma lo miraban bastante bien. • La carga de reportes le pareció bastante rápida • Surgió la duda de qué pasaría si no tenían internet, se le explicó el modo offline	70 segundos.

b. **¿Si tuvieras alguna duda de cómo usar la aplicación qué harías?**

Flujo	Notas/complicaciones/dudas	Tiempo por flujo
Ver el manual de usuario con preguntas y respuestas a dudas generales.	• El usuario no pudo detectar rápidamente la opción de ayuda, estaba a la derecha, se planea moverla a la izquierda o recalcar con texto que sí es la real. • Quiso saber si habría forma de agregar más preguntas luego • Dio algunas sugerencias de preguntas.	45 segundos.

c. **¿Qué pasa si no tienes internet?**

Flujo	Notas/complicaciones/dudas	Tiempo por flujo
Flujo de modo sin internet	• Se confundió un poco cuándo observó que se colocaba gris el botón de sincronizar casos, cuando lo presionó, la notificación le indicó que no estaba disponible por falta de internet. El usuario asintió, comentó que la notificación le aclaró que no podría usarla hasta que tuviera redes. • El usuario prendió su Wifi, y la notificación se removió, le agradó.	2 minutos y 30 segundos (apagar internet, volver abrir app, etc.).

d. **¿Cómo sincronizarías los casos que has tomado sin internet?**

Flujo	Notas/complicaciones/dudas	Tiempo por flujo
Sincronizar casos	• El usuario no comprendía la razón de sincronizar. • Se le tuvo que explicar que era para alimentar los datos del tablero, realmente la aplicación no los necesitaba, pero cuando entendió que aportaría, le pareció útil.	50 segundos.

2. Dashboard Leishmaniasis

Primero, se presentará la retroalimentación obtenida en un primer prototipo. Luego, la tabulación de resultados de la encuesta al prototipo final.

Tabla 20: Resultados de retroalimentación prototipo #1 de Dashboard Leishmaniasis

No.	Pantalla/Flujo o Función	Objetivo de enfoque	Retroalimentación obtenida
1	Página de inicio de sesión	Objetivos específicos (**b)** y (**c)**	El color de los entregables es bueno, continúa con los colores de la aplicación y se nota que son los mismos. Colocar alertas en el inicio de sesión para saber si algo salió mal o no
2	Formulario de incidencia	Objetivos específicos (**b)** y (**c)**	Cambiar campos de género y de etnicidad para que cuadren con la base de datos.
3	Subir archivos Excel	Objetivos específicos (**b)** y (**c)**	Ponernos de acuerdo en el formato que se usará para la carga de casos, etc.
4	Sección de gráficos	Objetivos específicos (**b)** y (**c)**	Se acordaron cierta cantidad de gráficos, además se va a dividir la sección de reportes en tres secciones para que sea más sencillo usarlas.
5	Gestión de usuario	Objetivos específicos (**b)** y (**c)**	Agregar la opción de eliminar un usuario si se es administrador.

La encuesta realizada en Typeform del prototipo final, arrojó los siguientes resultados.

a. Resultados encuesta final

Ilustración 105: Resultados de la pregunta #1 de la encuesta de Dashboard Leishmaniasis.

Ilustración 106: Resultados de la pregunta #5 de la encuesta de Dashboard Leishmaniasis.

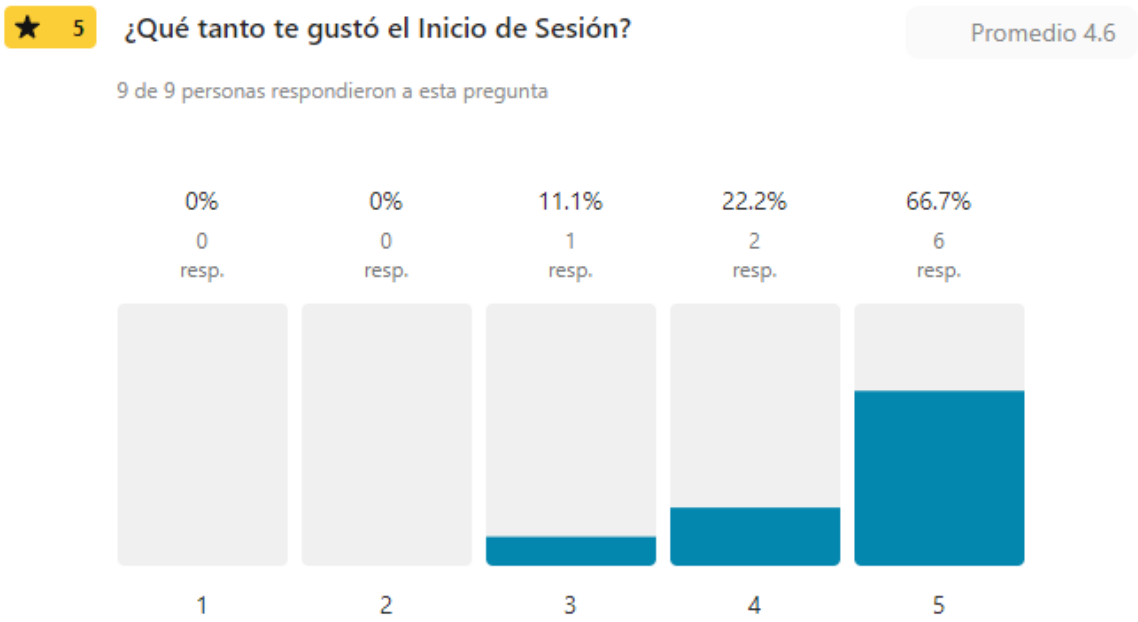


Ilustración 107: Resultados de la pregunta #6 de la encuesta de Dashboard Leishmaniasis.

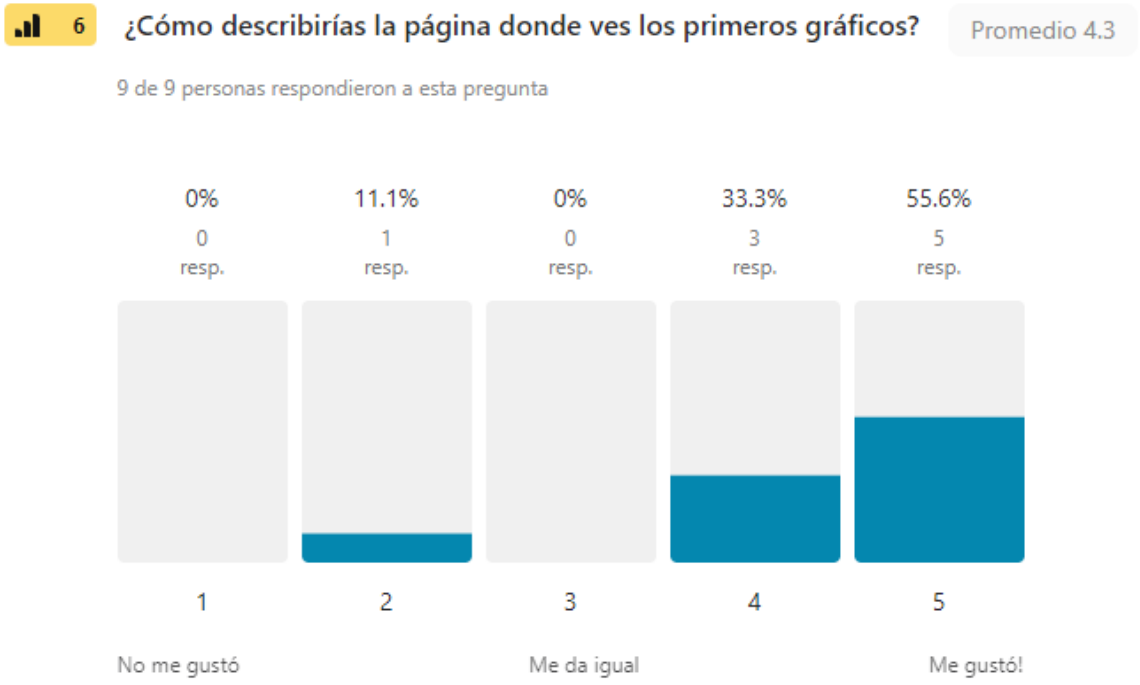


Ilustración 108: Resultados de la pregunta #7 de la encuesta de Dashboard Leishmaniasis.

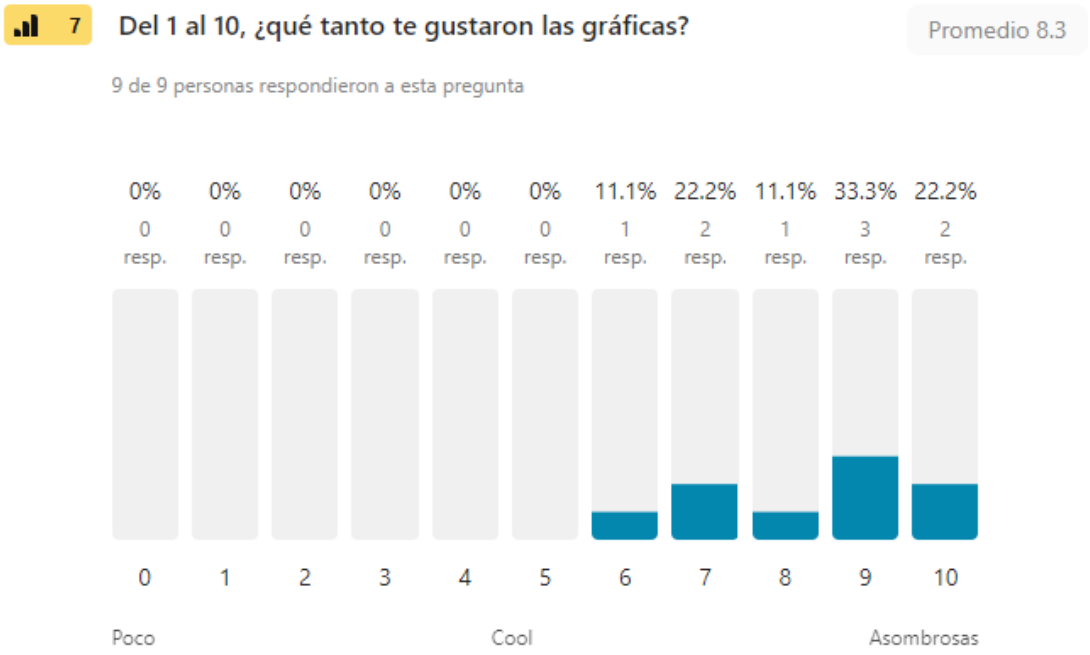
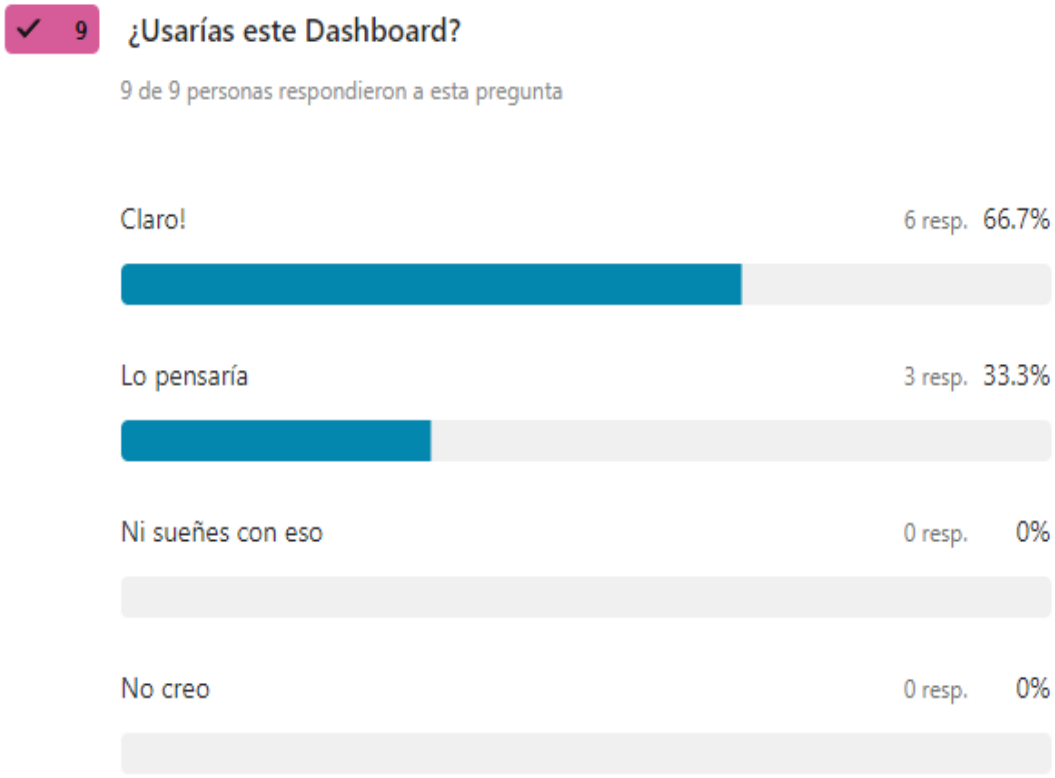


Ilustración 109: Resultados de la pregunta #9 de la encuesta de Dashboard Leishmaniasis.



Un campo final de sugerencias y comentarios fue colocado al final de la encuesta. Las sugerencias obtenidas se presentan a continuación:

- *"Utilizar un tipo de letra distinto, quizás más formal o sencilla."*
- *"Me parece que los tonos están muy bien proporcionados"*
- *"La paleta de colores me parece bien."*
- *"Creo que una interfaz más limpia y tener enfoque o algo que me indique antes de las opciones y en cuál página estoy actualmente creo que puede ser algo que ayudaría y tal vez que no sea tan "grande" o que llene tanto la pantalla la interfaz en general."*
- *"Tener las gráficas en la mejor calidad posible"*

b. Prueba de usabilidad de Dashboard Leishmaniasis

Una prueba de usabilidad fue realizada para conocer si el tablero seria de utilidad para los usuarios finales. Debido al carácter cualitativo de la prueba, está redactado narrando los hechos desde la perspectiva del entrevistador.

Entrevistador: Alejandro Tejada

Resultado de prueba de usabilidad

Introducción

1. Para dar contexto:
    a. Hola, estamos trabajando en un proyecto relacionad con Leishmaniasis Cutánea en Guatemala, como bien sabrás, es una enfermedad endémica. Estamos desarrollando una app que detecte Leishmaniasis cutánea y un tablero donde esa información pueda ser vista. Esta entrevista es sobre el tablero de información.
    b. Estamos desarrollando esta entrevista para obtener información cualitativa y saber si será útil para ti en el futuro, ¡gracias!
2. Antes que entremos de lleno en las pruebas:
    a. Creo que deberíamos conocernos unos minutos, ¿qué te parece? (Ambos se presentan, nombres, edades, tiempo trabajando, etc.)
3. Se inicia el marco de la prueba o setup

Marco de prueba

1. Por favor, abre el siguiente link donde está el prototipo del tablero (*se le proporciona el link del prototipo visual del tablero hecho en React y automatizado con Netlify)* y espera ahí en la primera pantalla de inicio de sesión.
2. ¿Podrías compartirme tu pantalla por favor?
    a. Siéntete libre de hacerlo cuando quieras
3. Gracias por compartir, ahora te diré lo que haremos. Tengo algunas peticiones que hacerte, unas pequeñas tareas que tendrás que hacer siguiendo el diseño, mantendré silencio en algunas partes para NO interferir en tus interacciones, pero no significa que no puedas decir en voz alta los pasos que seguiste, eso ayudará a que sepamos si te será de utilidad.
4. Muy bien, ¡Empecemos!

Test o prueba

1. **Contexto**:
    a. *Eres un colaborador del MSPAS o un usuario final. Necesitas ver la información relacionada a la enfermedad del último año. También, te piden administrar los accesos de ciertos usuarios, verificar otros gráficos, y subir un par de formularios de forma manual.*
2. **Tareas para realizar:**

a. **¿Cómo inicias sesión?**

Flujo	Notas/complicaciones/dudas	Tiempo por flujo
Inicio se sesión	• El usuario ingresó los datos. La razón de la tardanza fue que metió un correo que no era válido, recibió una notificación antes de poder intentar iniciar la sesión. Además, metió una contraseña muy corta, la retroalimentación de sus errores se hizo luego y los corrigió.	1 minuto y 10 segundos.

b. **¿Cómo encontrarías gráficas relacionadas a Guatemala o algún rango de edades?**

Flujo	Notas/complicaciones/dudas	Tiempo por flujo
Gráficos de personas, geográficos e históricos.	• Al usuario le pareció interesante que, en la primera pantalla al iniciar sesión, se vean datos. Como los datos pedidos en la pregunta no están ahí, los buscó. Al inicio, no comprendió donde estaban, pero cuando descubrió que una pestaña contenía más opciones. • Encontró los gráficos históricos. En cada sección, preguntó porque tenían esos nombres. Se le explicó que se intentó agrupar por secciones e información relacionada.	5 minutos (tomando en cuenta todos los gráficos).

c. **¿Qué harías si quieres ingresar un formulario manualmente?**

Flujo	Notas/complicaciones/dudas	Tiempo por flujo
Ingreso de lesión de forma manual	• Ingresó a la opción de formulario rápido, le sorprendió lo grande del formulario, pero entendió que esas formas siempre han sido grandes y tediosas de llenar. • Algunos datos eran obligatorios, no comprendió al principio cuales, pero cuando leyó la descripción supo que los que tenían un asterisco lo era. Se le olvidó uno, pero hay seguridad y corrigió su error	3 minutos (llenando datos y viendo cuales eran obligatorios)

VII. ANÁLISIS DE RESULTADOS

1. Inteligencia artificial para reconocimiento de lesiones de Leishmaniasis

Gran parte del tiempo de la implementación de este módulo se invirtió en la recolección de imágenes, debido a que es una de las fases más importantes al momento de desarrollar un modelo de inteligencia artificial. En la Ilustración 50 se puede observar el total de imágenes recolectadas por fuente de datos, sumando un total de 4,028 imágenes de Leishmaniasis y 3,506 imágenes de enfermedades similares a ella.

Algo importante a resaltar es que idealmente un agente inteligente, se debe de entrenar y validar con material lo más similar a lo esperado cuando esté en producción. En nuestro caso, serían imágenes capturadas directamente desde un dispositivo móvil. De las distintas fuentes de donde se recolectaron fotografías, únicamente las provenientes del Programa de Vectores del MSPAS podemos asegurar que fueron tomadas con dispositivos móviles. Lastimosamente la cantidad de fotografías de esta fuente son insignificantes comparadas con el total recolectado.

Encontrar la eficiencia humana es de suma importancia para tener un valor de referencia al momento de analizar los resultados de los modelos desarrollados. En este caso, se determinó tanto la eficiencia de un humano sin experiencia identificando Leishmaniasis cutánea como también con experiencia previa. La eficiencia de un humano sin experiencia previa esperada es del 50% debido a que como es un problema binario, si toma decisiones al azar, fácilmente podría obtener la mitad de las respuestas correctas.

En la Ilustración 51 se pueden observar los resultados de los compañeros del grupo de desarrollo del proyecto al intentar identificar Leishmaniasis cutánea a partir de fotografías. Por temas de permisos y privacidad, la eficiencia humana solo se pudo determinar con miembros del equipo. Se puede comprobar que el promedio de sus resultados da exactamente una eficiencia del 50% lo que coincide con el valor esperado.

En la Ilustración 52 se puede observar los resultados de los dos profesionales que se encuentran apoyando el desarrollo de este proyecto, ellos son parte del equipo. El promedio de la eficiencia humana teniendo experiencia previa es del 66.67%, esto indica que incluso para un profesional, se torna complicado identificar Leishmaniasis cutánea disponiendo solamente de la información visual a través de una fotografía.

En la Tabla 8, se observan los resultados finales de un modelo en el cual se dividió el conjunto de imágenes incorrectamente. Es decir, se tomaron imágenes distintas del mismo paciente en múltiples categorías de datos. Se puede notar que la precisión de la categoría de entrenamiento y desarrollo es sobresaliente, supera la eficiencia humana con experiencia previa. Esto sugiere que los resultados pueden estar alterados. Esta advertencia se confirma con la precisión de la categoría de prueba, no solo es muy distante de la precisión de las otras dos categorías, sino que está por debajo de la precisión humana sin experiencia previa.

El motivo de esta precisión engañosa es que, al tener imágenes del mismo paciente en múltiples categorías, el modelo en lugar de generalizar características propias de la

lesión, se memoriza aspectos de los pacientes. Razón por la cual, al procesar la categoría de prueba, la cual contiene imágenes independientes de la categoría de entrenamiento y desarrollo, la precisión es muy baja.

Tomando las buenas prácticas mencionadas anteriormente, en la Tabla 7, se resume la distribución de imágenes utilizadas para desarrollar los modelos que se discutirán más adelante. Así también en la Tabla 9, se muestran los mejores hiperparámetros de preprocesamiento y aumentación de datos, estos se determinaron utilizando el algoritmo BOHB. Algo interesante es que independientemente de la arquitectura de la red, luego de la exploración del algoritmo BOHB, se llegaba a resultados aproximadamente iguales.

Se entrenaron dos distintas arquitecturas de red, InceptionV3 y un modelo más pequeño el cual se describe la arquitectura en la Tabla 10. Para dar una idea de la diferencia de tamaños de estas dos redes, se puede observar la Tabla 11, en donde InceptionV3 supera al modelo pequeño por aproximadamente 40 millones de parámetros. De la Tabla 12 a la 14 se reflejan los resultados de ambos modelos. En promedio, el modelo de InceptionV3 es superior. Sin embargo, por la dimensión de esta red no es posible ejecutarse desde un teléfono móvil.

Se realizaron distintas pruebas y así poder determinar que los requerimientos mínimos para ejecutar el modelo pequeño en dispositivos móviles son los descritos en la Tabla 15. Analizando los recursos utilizados durante el uso de la aplicación, se determinó que en promedio el dispositivo móvil utiliza menos de 2GB de memoria RAM. Cabe aclarar que la aplicación no utiliza 2GB de memoria RAM, sino que este consumo incluye tanto operaciones fundamentales del teléfono como el uso de la aplicación.

En la Ilustración 53, se puede observar los resultados del modelo reflejados en la aplicación móvil. Debido a que los usuarios finales poseen un nivel académico bajo, se acordó con las partes interesadas mantener los resultados de forma simple, por lo que únicamente se despliega si fue negativo o positivo el diagnóstico y un porcentaje el cual representa qué tan seguro está el modelo de dicho resultado.

La versión de Android mínima para ejecutar la aplicación móvil es la Android 5. Actualmente la mayoría de las aplicaciones tienen este requerimiento mínimo puesto que es una versión de Android lanzada hace 7 años y ya está descontinuada. Finalmente, la aplicación pesa 132MB. Sin embargo, se determinó como requerimiento mínimo 200MB de almacenamiento libre para que el dispositivo móvil no presente ninguna complicación.

2. Gráficas del panel de control

Como se mencionó previamente, se desarrollaron cuatro tableros para poder analizar los datos en distintas perspectivas desde el panel de control. En la Tabla 12 se describe la cantidad de indicadores clave, filtros y gráficos de cada tablero. Se puede notar, que el tablero con más indicadores clave es el tablero de inicio, debido a que la intención de este tablero es brindarle información resumida al usuario y así poder sacar conclusiones rápidas. En la Ilustración 54 se muestra una imagen del resultado final de este tablero.

Otro aspecto importante que podemos notar en cada tablero, se tiene al menos un filtro; esto ayudará al usuario a concentrarse en los datos que le interesen y poder realizar un mejor análisis de la información. De la Ilustración 55 a la 57, se muestran fotografías del producto final de cada tablero. Se puede observar que los tableros son bastante simples e interactivos, lo anteriormente mencionado se validó con ayuda de pruebas de usabilidad.

En la Tabla 17, se puede observar los resultados de la prueba de usabilidad, en donde se resumen todas las interacciones de los distintos tableros en 5 flujos. Se puede notar que el flujo 2 y 4 son los que mayor tiempo promedio poseen, esto se debe a que son flujos donde se requiere interacción con los filtros de los tableros. Si bien es cierto, el tiempo promedio supera el minuto en ambos flujos, este tiempo es aceptable puesto que los participantes no estaban familiarizados con el panel de control. Respecto a los tiempos promedio del resto de flujos, se puede notar que los participantes no se demoraron mucho en interactuar con las gráficas, lo cual valida que nuestro diseño es intuitivo para el usuario.

3. Backend

En la sección A de los resultados, se puede notar cómo se realizaron pruebas con *Postman* sobre los diferentes *endpoints* programados, las Ilustraciones muestran tanto la respuesta del servidor, como el código de estatus respectivo y el tiempo que tardó la ejecución completa. En la Ilustración 26 se puede mostrar también la correcta configuración de seguridad CORS, permitiendo así al navegador del usuario final interactuar con el servidor. También se adjuntó fotografías de la base de datos en alguna de estas ocasiones, demostrando así su adecuada interacción con los datos. Estos pasos son de suma importancia, puesto que permiten realmente probar el sistema creado hasta el momento y que algunas de las rutas básicas también funcionen adecuadamente.

Las Ilustraciones 32, 33, 35 y 47 muestran también la interacción de Postman con el servidor ya subido en la nube, junto con el uso del dominio adquirido por el equipo. Para consultas simples, como lo son las Ilustraciones 32 y 33 se puede notar que el tiempo de respuesta es muy bajo, siendo esta no mayor a 1000 ms. Las Ilustraciones 35 y 39 por otra parte, denotan un tiempo de respuesta mucha mayor dado que la cantidad de datos utilizados son mayores y por ende el servidor necesitará de más capacidad y tiempo de procesamiento.

Las Ilustraciones 35 y 36 también denotan pruebas más pudientes, puesto que se generó data aleatoria con el fin de probar el tiempo de respuesta del sistema al ejecutar este tipo de transacciones. Ambas resultaron exitosas y con un tiempo de ejecución y respuesta considerablemente bajo por parte del servidor. Con la evidencia mostrada, cabe mencionar que el diseño de base de datos y diseño del ambiente de programación son de suma importancia puesto que son la base de la infraestructura del sistema creado y se logró exitosamente crear un conjunto de tecnologías que funcionan adecuadamente entre sí.

En cuanto a la sección B de los resultados, se muestran las tecnologías integradas de cada uno de los módulos del proyecto los cuales son: *Frontend, Backend*, Agente Inteligente e Infraestructura en la nube. El flujo de la aplicación permite la inserción de formularios a base de datos de forma práctica y amigable para el usuario. Además, el

servidor es capaz de informar al usuario acerca del estado de sus consultas, si estas fueron exitosas o no. También cabe notar es que los registros de interacción entre el servidor y la base de datos muestra información de alta utilidad al momento de depurar la aplicación y encontrar errores inesperados.

En cuanto al portal web estadístico, *backend* logró exitosamente cumplir con los servicios esperados por parte del usuario. Este tuvo la capacidad de brindar un documento Excel plantilla, con el objetivo que los usuarios puedan cargar los datos que desean almacenar de forma cómoda. El sistema también permite ingresar un formulario individual, dando retroalimentación al usuario acerca del estado de los datos cuando desea guardar el formulario. Cada una de las interacciones con el servidor cumple con tres objetivos: la interacción con el sistema de forma segura (gracias a la autenticación); ciclo de interacción completo con los datos del usuario y la base de datos; registros por parte del servidor para conocer en todo momento las acciones ejecutadas y así poder depurar posibles errores.

Finalmente, se garantizó la calidad en el sistema creando múltiples pruebas unitarias como la mostrada en la Ilustración 50, esto permitirá que el código desarrollado por parte del equipo siempre se encuentre en funcionamiento, reduciendo así la cantidad de errores inesperados en donde el usuario final pueda salir afectado.

4. Infraestructura

El diseño temprano de la arquitectura fue un punto clave y de suma importancia para tener un panorama más claro del objetivo a alcanzar con los servicios de la nube utilizados en el proyecto. El uso de instancias EC2 para el *backend* de la aplicación fue apropiado por el rendimiento mostrado en las pruebas y el bajo costo, hablando de temas presupuestarios. El uso de dos balanceadores de carga con grupos de objetivo separados proporcionó una solución a los problemas de enrutamiento en los *endpoints* generados con Express en ambos *backend*, logrando el funcionamiento correcto de la aplicación y el portal web en simultaneo.

En cuanto al almacenamiento de datos, RDS es una opción confiable y fácil de administrar, debido a que este proceso es realizado, en su mayoría, por AWS de forma automática. Sin embargo, en el transcurso del proyecto, se encontraron alternativas que hubieran representado opciones más accesibles en precio y con mayor flexibilidad de administración para los casos de uso del sistema que se desarrolló.

Continuando, cabe destacar que, el hospedaje del portal web estadístico en Amazon S3 fue una opción sencilla de implementar, acorde a las necesidades de uso y de costo insignificante en el presupuesto del proyecto en conjunto.

Pudo notarse en la sección del API y portal web estadístico que la autenticación básica HTTP cumple con el objetivo de privar al público general de acceder o poder visualizar los servicios prestados, por no contar con el usuario y la contraseña necesarios para acceder.

En la sección de la aplicación, los registros de ambos tipos de carga de formulario fueron exitosos. El tiempo de respuesta para la carga de un formulario simple, oscilaba entre 1 y 2 segundos como máximo. Por otro lado, la respuesta para un total de 10 formularios en conglomerado oscilaba entre 3 y 4 segundos, lo cual daba pauta a un uso rápido y cómodo de la aplicación durante el trabajo de campo. El tiempo de carga de 100 formularios, desde el ambiente de Postman, tomó 574 ms. Esto significa que, para la capacidad máxima de 99 formularios desde la aplicación, un tiempo estimado seria de 1 o 2 segundos con buena conexión a internet. Cabe mencionar que se utilizaron dispositivos móviles conectados a wifi 3G / LTE con una velocidad promedio de 10 Mbps.

La carga de las imágenes se realizó en paralelo a la carga de los formularios y fue exitosa. Se configuró la creación de directorios por fecha para tener un histórico de las fotografías tomadas.

Esta primera etapa de pruebas con la aplicación fue de suma importancia para promover el uso de un módulo dedicado a Logging en el sistema. Esto claro, para poder dar seguimiento a posibles errores en el ambiente de producción, sin la necesidad de supervisión técnica constante. En varias ocasiones se tuvieron problemas que no lograban ser identificados rápidamente, debido a la falta de este módulo.

Continuando con las funcionalidades del portal web estadístico, pudo observarse que el manejo de usuarios fue exitoso en cada una de las pruebas. Cabe mencionar que, como el sistema es privado, siempre habrá al menos un usuario default registrado de tipo administrador (creado desde *backend*), con el cual se puedan crear, modificar y eliminar otros usuarios destinados a la utilización del portal web.

La carga de formularios fue exitosa de igual manera, con un tiempo de respuesta bastante rápido, entre 1 y 2 segundos. En cuanto a la carga de los archivos xlsx, el tiempo de respuesta variaba dependiendo del tamaño del archivo, ya que cada columna era un nuevo registro por insertar del lado del *backend*. No se presentaron retrasos en las inserciones más haya de esta especificación.

Se hicieron pruebas de demanda media alta con un número de cinco usuarios, cargando formularios, archivos, editando permisos de usuarios y visualizando los gráficos. Las pruebas fueron exitosas, no se presentaron inconvenientes.

La visualización de las gráficas fue posible gracias a la conexión que se estableció con la base de datos de réplica. Como pudo observarse se contaban con las vistas correspondientes desplegando graficas provenientes de Google Data Studio, en donde se construyeron las gráficas y se acoplaron a las necesidades del usuario del sistema.

5. Frontend

La principal metodología de toma de resultados fue la entrevista cualitativa y el HCI. Esto es importante de destacar, puesto que delimita el alcance de las entrevistas y de los factores que intervienen en una correcta interacción de toma de retroalimentación.

a. Aplicación móvil

Tomar lo anteriormente dicho en cuenta mejora la perspectiva de lo que se esperaba de parte del usuario. Por ejemplo, nótese las entrevistas #1 y #2 en la sección de resultados de la aplicación móvil. Estas describen una toma de retroalimentación hablada mediante videoconferencia, no basada en estadísticas o números. El entrevistado en cuestión tiene carta libre para navegar por la solución y el que realiza la entrevista, tan solo lo guía y hace preguntas clave. Los resultados se miden en niveles de satisfacción. ¿Es objetiva una retroalimentación u opinión de esa manera? ¿Está sesgada la misma?

Si, está sesgada puesto que el entrevistador puede orientar las respuestas si así lo desea, ya que el entrevistado puede no comprender del todo. ¿Por qué se eligió este tipo? La razón primordial es una característica clave de la entrevista cualitativa: la flexibilidad y sinceridad.

No es posible garantizar que el entrevistador sea imparcial, pero sí es posible entender mejor al entrevistado mediante esta técnica. Al ser flexible, permite adaptarse a las situaciones del tema, sujeto o tiempo. Habiendo esclarecido eso, las entrevistas #1 y #2 toman otro enfoque: no es solo lo que dicen sino las expresiones qué realizan. Claro, hay en juego otros factores ambientales (como la situación social que se discute luego, la disposición de la persona, si está cansada o si no le atrae) pero es solo a través de la entrevista cualitativa que se puede concluir si el resultado de la aplicación móvil es satisfactorio para los *Stakeholders*.

Algunos puntos sobresalientes de los resultados los da la entrevistada Rosa Tejada cuando dice: “Si, es como esos manuales que traen los electrodomésticos”. Ella se refería al manual de usuario que la aplicación posee. Este símil de funcionalidades es importante, y es una convención que debe tratarse siempre de cumplir: hacer referencias en funcionalidades a cosas que sean lógicas o que estén relacionadas con los usuarios finales. Otro punto es cuando menciona el color verde de la aplicación. El verde fue una decisión de diseño basada en las convenciones mencionadas en el marco teórico y en la toma de retroalimentación con *Stakeholders*. Pero lo importante no es eso, sino destacar que la entrevistada hace un símil entre el verde y otros términos como: naturaleza, salud, bosque, frescura, salud, atención. El hecho que los colores sean asociados a un sentimiento, recuerdo o cualidad no es nuevo. Es algo intrínseco en cada ser humano. Es esa cualidad en todo ser humano de conectar el color con algo que influye en los diseñadores de interfaces: usar colores y variantes de estos que evoquen algo en el usuario. En el caso de Rosa, ella lo relacionó sin el entrevistador mencionárselo, confirmando que la decisión de tomar esos colores fue cuando más acertada.

Si bien, hay factores positivos en el diseño y manejo de la aplicación, también hay puntos de mejora que no se tomaron en cuenta. Casi al final de la entrevista, el entrevistador le menciona a Josselyn que la aplicación estaría relacionada con el Dashboard de Leishmaniasis. Eso sacó de enfoque al usuario que no sabía de su existencia, y lo confundió. Si bien al explicarle qué era, ella comprendió, su desenfoque hizo que adoptara otra visión del prototipo: el porqué de este. Si bien, puede que antes de esa declaración ella pensara como un organismo entero la app, al contarle que existía más, pudo haberse sesgado su

opinión de que había más y de que quizás no fue de su agrado ese y lo demás sí. Es este punto también una flaqueza de la entrevista cualitativa: una vez sesgada, no es posible encaminar de nuevo al sujeto. Es esa flaqueza lo que sugiere que siempre se tomen usuarios diferentes para toma de retroalimentación cualitativa. Al ser nuevos, no conocen el progreso anterior y pueden valorar más objetivamente el prototipo o funcionalidad final.

La prueba de usabilidad del usuario final arroja resultados cuan lo menos interesantes. En ella, se describe como un usuario recorre los flujos, pero dándolo en forma de tareas. L aprueba tiene como objetivo entender si el diseño tiene fallas y no necesariamente la opinión del usuario, ¿por qué? Porque si el usuario comete un error, no es su culpa, sino del diseño. La prueba enfoca la mirada en el factor de usabilidad. Al tomar tiempos, se puede hacer una idea de qué tanto le costó al usuario, qué tan frustrado pudo sentirse, etc. El usuario final entrevistado no tuvo muchos inconvenientes, el flujo de analizar una lesión fue el más largo, pero también en donde menos dudas surgieron. Le pareció interesante el factor de la rapidez de la carga, los colores y finalmente, comentó que era algo que le sería de utilidad.

Finalmente, la Tabla 20 recoge pedidos puntuales de los *Stakeholders*. Estos deben tomarse con precaución y con cuidado. El proceso sugerido para saber cuáles adaptar o no proviene de la palabra clave valor. El valor se crea cuando al usuario final esa funcionalidad o atributo le sirve, le ayuda o lo aventaja. Si la funcionalidad crea valor para el usuario final (y no un *Stakeholder* externo que no usará el producto), es cuando se debe implementar. Cualquier retroalimentación es valiosa, pero el arte consiste en saber cuál tomar e implementar y cuál descartar, pero usarla para experiencia y aprendizaje del diseñador (su servidor).

b. *Dashboard* Leishmaniasis

El enfoque adoptado para el diseño e implementación de la aplicación es bueno. ¿Pero hubiera funcionado igual, peor o mejor un enfoque cuantitativo?

Esa pregunta fue el timonel para la toma de retroalimentación del Dashboard Leishmaniasis. Para propósitos de facilidad, de ahora en adelante se usará el término DL para referirse al Dashboard Leishmaniasis.

Primero, se realizó un análisis cualitativo del prototipo inicial del DL. Este fue desarrollado a nivel de programación no paralelamente a la aplicación, sino posterior. Por lo cual, la toma de retroalimentación no fue tan extensa y personal como en la aplicación, pero no les resta calidad a los resultados obtenidos. Primero, la Tabla 19 describe los resultados obtenidos. Es interesante notar que para el DL hay muchos más cambios que para la aplicación. ¿por qué?

Pensar en el DL y la aplicación como un organismo vivo: ambos forman parte del otro. Ese concebimiento de la idea fue planteada al inicio de la idea del proyecto por el grupo. Se quería algo que estuviera unido intrínsecamente. Decisiones posteriores tanto de diseño como a nivel de infraestructura apoyan esta visión. Pero, orientado al módulo de *Frontend* es importante destacar que se compartieron muchos factores de diseño: íconos,

colores, lenguaje de la aplicación, textos y convenciones. Esta relación permitió compactar trabajo y reutilizar prototipos e ideas. Ese aprovechamiento de ideas se transmite en más funcionalidades que fueron aprovechadas. Por mencionar algunas: formularios de lesiones, gráficos de lesiones y resúmenes, carga de casos en formato Excel, manejo de usuarios, etc. Esas funcionalidades crean valor para los usuarios finales, y es esa la razón por la cual en el primer prototipo hay tantas peticiones, cambios y opiniones. Puede suponerse entonces, que entre mayor valor el cliente piense que le da, mayor serán la cantidad de cambios que pidan o que piensen que pueden beneficiarle.

Quisiera discutir el punto anterior con mayor profundidad. ¿qué hace que un usuario se "emocione" o sienta interés por funcionalidades? La respuesta está en la expectativa del usuario y la relación con el valor que le encuentra. El usuario puede entrar en un estado de euforia cuando se da cuenta del beneficio de cierta funcionalidad para él. Esa es posiblemente la razón por la cual, en entrevistas o reuniones para toma de requerimientos, los usuarios no dejan de pedir cosas, quieren más y más al notar la tangibilidad del proyecto. Esa expectativa debe manejarse con cuidado. Hay un dicho que dice que un cliente nunca sabe lo que quiere, y otro que el cliente siempre tiene la razón. Nótese como en la Tabla 20 las cinco tuplas de retroalimentación son sugerencias o peticiones bien definidas y no ambiguas. En ese punto, se puede decir que el "cliente" si sabe lo quiere. Pero, cuando se habla de otras funcionalidades posibles, el usuario abandona su emoción anterior y la sustituye por una nueva: ya sea mejor o peor. Se decidió llamar a ese abandono de una idea que gustaba por una nueva (mejor o peor) a pesar de la emoción que se tenía como tangibilidad eufórica. Ese momento de la conversación es donde existe el arte de tomar requerimientos: el balance entre lo que el cliente NO sabe y lo que SÍ cree saber. Un buen diseñador debe tomar en cuenta estos factores y estar atento a las señales cualitativas. ¿Por qué es importante? De esa forma, cuando los requerimientos sean tomados y estos se transformen en funcionalidades, el valor que estas den al usuario sea real, bien basado y fundamentalmente sólido. La toma de requerimientos y de retroalimentación de la Tabla 20 muestra esa euforia, y se trató por todos los medios posibles de hallar el balance antes mencionado. Para lograr un balance, es posible incluso recurrir a otros métodos, como colocar un límite de funcionalidades, un límite de horas, un punto de discusión intermedio o aclarar desde un inicio el alcance del proyecto.

Las Ilustraciones 10 a la 14 son los resultados graficados de una encuesta realizada en Typeform para el prototipo final del DL. La #10 trata de ahondar en la primera impresión de los usuarios. 45% dijo que le gustó "bastante". Un 34%, "mucho" y un 22% "regular". Este tipo de toma de opinión es interesante: es un punto intermedio entre lo cualitativo y cuantitativo. La toma final del DL se enfocó en eso debido a que se quería denotar si había una diferencia grande entre un enfoque puramente cualitativo o semi cuantitativo. Se nota que las palabras usadas son ambiguas, basadas en la jerga local de Guatemala. ¿Qué se extrae de este resultado? Ningún voto fue para "no me gustó" y para "no mucho", por tanto, se concluye que la primera impresión es positiva. Hay un término en psicología que dice que la primera impresión se da en los veinte segundos de conocer a una persona. En terminología web y de diseño esa brecha es cortísima: dos segundos. Los tiempos de carga influyen claro, pero un buen diseño gusta al espectador en los primeros segundos. ¿Qué puede influir? Para eso aclaremos lo primero que el usuario ve: formas y colores. De nuevo, los colores son importantes porque denotan el humor que tendrá el usuario o la

predisposición a gustarle. Dados los resultados se respalda que la primera impresión fue buena.

Las Ilustraciones 11, 12 y 13 describen no algo general como una opinión, sino está orientado a funcionalidades específicas. Esto, con el fin de atomizar la encuesta. Gracias a esto, se puede saber si, aunque generalmente fue de agrado el DL, una funcionalidad quizás no combina con el diseño, o quizás no tiene sentido para el usuario. La #12 termina con un 56% de encuestados respondiendo favorablemente, y con el 11% respondiendo desfavorablemente. Finalmente, las Ilustraciones 14 y 15 vuelven a recoger la opinión general del DL. ¿por qué repetir la pregunta? Porque si se toma la misma pregunta al inicio o al final, es posible saber si las opiniones del usuario cambiaron al moverse por el prototipo. Puede que al inicio les guste, pero al navegar no. La Ilustración 14 termina con un 67% de personas respondiendo que efectivamente, que si usarían el DL. Un 33% contesto que lo pensaría. Dos tercios de los encuestas no es una mala señal. La Ilustración 15 muestra un puntaje en la escala de 1-10. Se aprecia que se forma una especie de histograma. Buena señal, ya que ningún dato bajó de un 6, por tanto, podría decirse que el promedio es con 9 puntos de 10, respaldado por un 45% de encuestados. La prueba de usabilidad enmarca aún más los resultados anteriores, ya que el usuario muestra en sus interacciones en los flujos cronometrados que le parece útil la forma en que están organizados los gráficos, los nombres adecuados y la terminología o jerga interna que usan.

Aspectos los cuales vale la pena mencionar y que abarcan ambos proyectos son relacionados a la situación social, y como estos influyen de maneras que antes no se hubiera imaginado. La pandemia del COVID-19 trajo el confinamiento a todo el mundo. En el área de diseño de interfaces, UI, UX y otras relacionadas, el prototipado e iteraciones con usuarios son puntos clave de los proyectos. Por lo general, esta toma de requerimientos y de prototipos a nivel de papel eran en persona, interactuando, midiendo expresiones y sacando conclusiones en el momento. Al estar recluidos, se ha recurrido a otro tipo de métodos para esto. Los diseñadores usan plataformas para toma de retroalimentación (en el caso de este proyecto fueron encuestas, videoconferencias y algunas presenciales) que antes ni siquiera era concebible. Sin embargo, estos acercamientos han demostrado ser igual de efectivos. Lastimosamente, no podemos abandonar el sentido social que por naturaleza tiene el ser humano. Al no poder interactuar apropiadamente, muchos proyectos han partido mal ya que no se tiene una buena fundación. Eso fue tomado en cuenta en este proyecto hasta lo posible. ¿cómo? Tratando de tomar requerimientos lo más pronto posible, iterando en varias ocasiones los mismos, manteniendo una comunicación constante con el usuario final, y mantener una relación sana entre usuario y desarrolladores. ¿Qué tanto afecta una situación social a un proyecto? A nivel de un proyecto intermedio, fue un impedimento, pero no un bloqueo total. Es probable que para proyectos más grandes en compañías con mayor personal si sea una fuente de bloqueos constantes, por lo cual, tomar acciones o iniciativas tempranas parece ser una buena solución

VIII. CONCLUSIONES

A. Inteligencia artificial

1. Se desarrolló un agente inteligente capaz de diagnosticar Leishmaniasis cutánea con requerimiento mínimo en teléfono móvil de 2GB de memoria RAM, versión de Android 5 y 200MB de memoria de almacenamiento.
2. Se validó el agente inteligente desarrollado con el conjunto de imágenes obtenidas a lo largo del proyecto obteniendo una precisión de 90.95%, sin embargo, la cantidad de imágenes obtenidas con un teléfono móvil, no fueron las suficientes para asegurar que el rendimiento del modelo sea efectivo con imágenes tomadas directamente desde la aplicación móvil.
3. El modelo InceptionV3 supera por 7% en precisión al modelo pequeño, debido a la cantidad de parámetros que contiene, superando al modelo pequeño en 40 millones, siendo imposible ejecutarse desde un teléfono móvil.
4. Se desarrolló un panel de control intuitivo con la capacidad de mostrar los datos desde las perspectivas de personas, históricos y geográficos, para autoridades del subprograma de Leishmaniasis y así poder tomar decisiones basadas en datos.
5. Al realizarse pruebas de usabilidad, los flujos que presentaron mayor dificultad para el usuario fueron los que requerían filtros para el análisis de los datos, con un promedio de tiempo de un minuto.

B. Backend

1. La creación de un diseño de base de datos considerando los datos que utilizan los usuarios finales, junto con su retroalimentación, es fundamental para crear un sistema confiable y de almacenamiento ordenado de información.
2. El uso de Node.js junto con sus herramientas como Express, provee librerías y utilerías que satisfacen las necesidades que conlleva crear un API tanto para tecnologías web, como para aplicaciones nativas.
3. La interfaz REST para conectar varios sistemas basados en el protocolo HTTP cumplió adecuadamente la realización de servicios propuestos.
4. El diseño Modelo, Vista, Controlador demostró ser una estrategia de alta calidad dado que la programación de entidades, sus controladores y su interacción con base de datos fue exitosa.
5. La infraestructura creada en Amazon *Web Services* es altamente viable para proyectos escalables relacionados con el modelo cliente – servidor.

C. Infraestructura

1. El diseño temprano de la arquitectura de infraestructura fue de suma importancia para tener un panorama más claro de los servicios en conjunto que se utilizaron y de esta manera proporcionar la funcionalidad al sistema como un todo.
2. Contar con un módulo de *logging* en el *backend* es esencial para la detección de errores en producción. Entre ellos, el fallo de inserción de registros a base de datos y/o errores de conexión a la infraestructura.

3. El hospedaje y distribución de un sitio web estático en internet, es posible a través de tres servicios clave: AWS S3, *Route* 53 y Amazon *CloudFront*, los cuales cumplen con minimizar la administración y el costo de operación del portal.
4. La arquitectura construida cumple con los requerimientos del sistema en cuestión; manejo de usuarios, carga y descarga de archivos xlsx y carga de formularios.
5. La información y los servicios proporcionados por las aplicaciones están restringidos al público general y a indexaciones de búsqueda, por la utilización de Autenticación HTTP Básica.
6. La información y los servicios proporcionados por las aplicaciones no pueden ser accedidos por agentes externos gracias a la autenticación JWT.
7. El sistema desarrollado está pensado para funcionar por sí solo durante un periodo de seis meses, con el presupuesto necesario involucrado. Este periodo podría extenderse, al obviar posibles actualizaciones en los servicios utilizados.

D. Frontend

1. Una toma de retroalimentación cualitativa sobre una cuantitativa tiene la característica sobresaliente de ser flexible y sincera. Esto, la convierte en un valioso recurso en proyectos de diseño de interfaces o experiencias de usuario.
2. El poder de una entrevista cualitativa se da cuando entiende que es de los pocos métodos de obtención de opinión que puede ser objetivo en situaciones donde los factores ambientales son fuertes, como la pandemia del COVID-19.
3. En el diseño de interfaces y de proyectos similares, la relación de funcionalidades de una interfaz debe estar lo más apegada a cosas de la vida real. De esa forma, el usuario las entiende sin siquiera analizarlo.
4. Si una funcionalidad o cambio nuevo provee valor al usuario, ese el criterio más alto que puede tenerse para saber si implementa o no.
5. Un análisis semi cuantitativo tiene la peculiaridad de proveer una experiencia similar al puramente cualitativo, con la salvedad que recoge más funcionalidades que posiblemente no se implementen.
6. Hay una relación intrínseca entre la expectativa del usuario y la relación del valor de una funcionalidad, esta guía el proceso de toma de requerimientos.
7. La tangibilidad eufórica es el punto ideal de toma de requerimientos, y al que se debería aspirar en un proceso formal de diseño de interfaces con el usuario.
8. Un 45% de personas indicaron que el DL fue de su agrado, un 56% respondió favorablemente ante las funcionalidades atómicas y un 67% de personas indican que sí usarían el dashboard.
9. Hubo una recepción positiva a la pregunta de la escala entre 1-10 sobre el DL en general. Un histograma muestra que 45% de los encuestados lo puntean con un 9 de 10, con una incertidumbre de 1 punto.
10. Las pruebas de usabilidad, arrojaron resultados positivos, con los usuarios finales asegurando que les sería de provecho ya que toma en cuenta lenguaje, funciones y otras cosas que usan seglarmente.
11. La pandemia del COVID-19 afectó el proceso de UX/UI, sin embargo, es posible sortear estos impedimentos sociales mediante el uso correcto de formularios, entrevistas, una buena comunicación y otros recursos.

IX. RECOMENDACIONES

A. Inteligencia artificial

1. Si se desea utilizar modelos desarrollados con TensorFlow en dispositivos móviles, procurar desarrollar aplicaciones nativas para poder utilizar TensorFlow, el cuál dispone de optimizaciones propias para cada dispositivo móvil.
2. Seguir recolectando imágenes tanto de casos positivos de Leishmaniasis cutánea como de enfermedades con síntomas similares tomadas directamente con teléfonos móviles. Esto ayudará a validar el modelo existente y mejorar los resultados, logrando un mejor impacto al apoyar del personal del Ministerio de Salud en el campo.

B. Backend

1. El diseño y análisis temprano de los datos con los que el usuario final interactúa es de suma importancia, puesto que permitirá un diseño de base de datos más preciso para garantizar el correcto manejo de los datos.
2. Un ambiente de programación para un sistema como este debe ser creado analizando las herramientas que el equipo considere adecuadas, no dejando de lado que las herramientas deben ser fáciles de integrar entre sí.
3. En el caso que se manejen datos personales que puedan perjudicar la vida de los pacientes y su información personal, es esencial manejar datos encriptados en la base de datos y garantizar un flujo seguro al manejar los datos.
4. Utilizar un ORM como Sequelize en los servicios de *backend* proveen un alto valor por parte de los desarrolladores puesto que estandariza las interacciones con bases de datos y permite una interacción más automatizada.

C. Infraestructura

1. Se recomienda cambiar el motor de base de datos en RDS de MySQL a PostgreSQL y así lograr encriptar la conexión con Google Data Studio, para casos de uso similares. Otra alternativa para lograr la conexión cifrada de los datos utilizando SSL/TLS con Google Data Studio, es la administración de una base de datos en EC2.
2. Utilizar un volumen de EC2 como base de datos permitiría tener más control sobre las conexiones realizadas a la información y, además, permitiría reducir los costos al eliminar procesos automáticos realizados por RDS. Sin embargo, cabe destacar que, al seguir esta recomendación se agregarían otras complejidades de administración como: manejo de respaldos, gestión del servidor de base de datos, antivirus del sistema, entre otros.
3. Al momento de utilizar Balanceadores de Carga de Aplicación para APIs en 2 servidores distintos, es importante utilizar *endpoints* diferenciados, es decir, con nombres de rutas distintos para cada uno. ¿Por qué? Porque de esta forma se pueden configurar reglas de redireccionamiento de las consultas al api en el balanceador de carga. Sino se realiza de esta forma, entonces no se aprovecha el funcionamiento

del balanceador de cargo y se tendrían problemas de redireccionamiento de las consultas HTTP.

4. Es importante verificar los subdominios manejados por CloudFront en las reglas de CORS de los servidores. Una diferencia como portalleish.com y www.portalleish.com puede ocasionar problemas en el uso del portal web.

D. Frontend

1. En una entrevista cualitativa, una vez sesgado al entrevistado no es posible obtener más retroalimentación objetiva, por tanto, se recomienda en iteraciones cambiar *Stakeholder* en cada iteración de un proyecto.
2. Para una toma de requerimientos efectiva, no basta con tener un listado de preguntas y de funcionalidades. También es necesario una buena comunicación con los *Stakeholders*, mantener en vista el alcance del proyecto y colocarse una fecha de entregable.
3. Para evitar que el distanciamiento social provoque fallas en las expectativas del *Stakeholder* y del equipo de desarrollo es indispensable la comunicación constante, aunque fuera corta pero constante. De esa forma, se cierran brechas en materia de malentendidos.

X. BIBLIOGRAFÍA

Akintunde, C. (2021, 17 abril). Basic Introduction to User Experience and User Interface Design. Medium. https://bootcamp.uxdesign.cc/basic-introduction-to-user-experience-and-user-interface-design-f0aae08a2b44

Arana BA (1998). Toward a better understanding of cutaneous leishmaniasis in Guatemala. Ph.D. Thesis, Liverpool School of Tropical Medicine.

Arana BA, Rizzo NR, Navin TR, Klein RE, Kroeger A (2000). Cutaneous leishmaniasis in Guatemala: People's knowledge, concepts and practices. Annals of Tropical Medicine & Parasitology 94: 779-786.

Babich, N. (2019, 7 October). The 4 Golden Rules of UI Design | Adobe XD. Ideas. https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/

Biedenkapp, A. y Hutter, F. (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale. https://www.automl.org/blog_bohb/ Consultada el 21 de octubre de 2021.

Briganti, G. (2020, 5 febrero). Artificial Intelligence in Medicine: Today and Tomorrow. Frontiers. Recuperado 18 de octubre de 2021, de https://www.frontiersin.org/articles/10.3389/fmed.2020.00027/full#B4

Campillo, A. (2020, 28 enero). ¿Qué es React y para qué sirve? Drauta. https://www.drauta.com/que-es-react-y-para-que-sirve

Copeland HW, Arana BA, and Navin TR (1990). Comparison of active and passive case detection of cutaneous leishmaniasis in Guatemala. Am J. Trop Med Hyg 43:257-259.

Copeland HW, Arana BA, and Navin TR (1990). Comparison of active and passive case detection of cutaneous leishmaniasis in Guatemala. Am J. Trop Med Hyg 43:257-259.

Dube, Simant. 2021. An Intuitive Exploration of Artificial Intelligence: Theory and Applications of Deep Learning. Palo Alto: Springer. Págs. 107-108.

El-Amir, H. y Hamdy, M. (2020). Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow. Egypt: Apress. 274 págs.

Gomis, R. (2020, 20 agosto). ¿Están interesados los médicos en aplicar la inteligencia artificial? ¿Es útil? Salud con Ciencia. https://cienciasdelasalud.blogs.uoc.edu/inteligencia-artificial-en-medicina/

Herwaldt BL, Arana BA, Navin TR (1992). The natural history of cutaneous leishmaniasis, Guatemala. Journal of Infectious Diseases 165:518-527.

Hill, C. (2020). Google Data Studio pricing, usage cost and limits. https://analyticshelp.io/blog/google-data-studio-pricing-cost-limits/ Consultada el 20 de octubre de 2021.

IBM Cloud Education. 2020. Artificial Intelligence (AI). https://www.ibm.com/cloud/learn/what-is-artificial-intelligence Consultada el 22 de agosto del 2021.

IBM. (2020). Data Science. https://www.ibm.com/cloud/learn/data-science-introduction Consultada el 20 de octubre de 2021.

Kamel, M.N., Geraghty, E.M. 2020. Geographical tracking and mapping of coronavirus disease COVID-19/severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) epidemic and associated events around the world: how 21st century GIS technologies are supporting the global fight against outbreaks and epidemics. Int J Health Geogr. https://doi.org/10.1186/s12942-020-00202-8

Lee, Gobert y Fujita, Hiroshi. 2020. Deep Learning in Medical Image Analysis: Challenges and Applications. Australia: Springer. 15 págs.

Leishmaniasis (2018) Centro de Estudios de Salud. Extraído el 20 de febrero del 2021 de https://www.ces.uvg.edu.gt/page/leishmaniasis/

McCarthy, John. 2004. WHAT IS ARTIFICIAL INTELLIGENCE?. https://homes.di.unimi.it/borghese/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatisai.pdf Consultada el 22 de agosto del 2021.

Microsoft. (2021). Precios de Power BI: Análisis para todo tipo de organizaciones. https://powerbi.microsoft.com/es-es/pricing/ Consultada el 21 de octubre de 2021.

MSPAS. (2018) Protocolos de Vigilancia Epidemiológica Enfermedades Vectoriales de Origen Parasitario. Ministerio de Salud Pública y Asistencia Social. http://epidemiologia.mspas.gob.gt/files/Publicaciones%202018/Protocolos/Enfermedades%20Vectoriales%20de%20Origen%20Parasitario.pdf Consultada el 17 de octubre del 2021.

MSPAS. (2019) Situación de la leishmaniasis en Guatemala, año 2018-2019. Guatemala

Muente, G. (2021, 12 febrero). Guía completa del Framework: qué es, cuáles tipos existen y por qué es importante en Internet. Rock Content - ES. https://rockcontent.com/es/blog/framework/

Muller, J. y Massaron, L. (2020). Data Science Programming All-in-One For Dummies. Canada: Wiley Brand. 11 págs.

Nicosia, G., Pardalos, P., *et al*. (2019). Machine Learning, Optimization, and Data Science: 4th International Conference, LOD 2018. Switzerland: Springer. 277 págs.

Nielsen, J. (2020, 15 November). 10 Usability Heuristics for User Interface Design. Nielsen Norman Group. https://www.nngroup.com/articles/ten-usability-heuristics/

OPS. (2020). Atlas interactivo de leishmaniasis en las Américas: aspectos clínicos y diagnósticos diferenciales. https://www.paho.org/es/documentos/atlas-interactivo-leishmaniasis-americas-aspectos-clinicos-diagnosticos-diferenciales Consultada el 20 de octubre de 2021.

Organización Panamericana de la Salud (2020). Atlas interactivo de leishmaniasis en las Américas: aspectos clínicos y diagnóstico diferencial. Extraído el 26 de marzo del 2021 de https://iris.paho.org/handle/10665.2/52645

Ortiz, D. (2020, 29 junio). ¿Qué es un dashboard y para qué se usa? (2021). Cyberclick. https://www.cyberclick.es/numerical-blog/que-es-un-dashboard

Padilla E (1982). Contribución al estudio de la leishmaniasis forestal americana en Guatemala. Thesis, Facultad de Medicina y Cirugía e Institutos Anexos. Universidad San Carlos de Guatemala.

Pérez, I. *et al* (2020) Strengthening Cutaneous Leishmaniasis control in Guatemala: policy recommendations* Policy Brief. Ministerio de salud pública y asistencia social, UVG, Centro de estudios de salud - CES. Extraido de: https://www.paho.org/en/documents/policy-brief-strengthening-cutaneous-leishmaniasis-control-guatemala-policy Consultada el 17 de octubre del 2021.

Pérez, I. *et al* (2020) Strengthening Cutaneous Leishmaniasis control in Guatemala: policy recommendations* Policy Brief. Ministerio de salud pública y asistencia social, UVG, Centro de estudios de salud - CES. Extraido de: https://www.paho.org/en/documents/policy-brief-strengthening-cutaneous-leishmaniasis-control-guatemala-policy

Prakash, O., Ranjan, A., *et al*. (2021). Computational Intelligence and Healthcare Informatics. USA:Wiley. 152 págs.

PyTorch. (2021). FROM RESEARCH TO PRODUCTION. https://pytorch.org/ Consultada el 23 de octubre de 2021.

Scikit-learn. (2021). Machine Learning in Python. https://scikit-learn.org/stable/ Consultada el 23 de octubre de 2021.

Seoane, M. S. (2020, 29 septiembre). Qué es y para qué sirve un moodboard Consultora de innovación y formación | Design Thinking en España. https://designthinking.gal/que-es-y-para-que-sirve-un-moodboard/

Shanmuganathan, Subana y Samarasinghe, Sandhya. 2016. Artificial Neural Network Modelling. Suiza: Springer. 4 págs.

TensorFlow. (2021). Model optimization. https://www.tensorflow.org/lite/performance/model_optimization Consultada el 24 de octubre de 2021.

TensorFlow. (2021). tf.keras.preprocessing.image.ImageDataGenerator. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator Consultada el 24 de octubre de 2021.

TensorFlow. (2021). Una plataforma de aprendizaje automático de código abierto de extremo a extremo. https://www.tensorflow.org/ Consultada el 23 de octubre de 2021.

Wexler, S., Shaffer, J. y Gotgreave, A. (2017). The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios. Canada: Wiley. XIV págs.

Wood, Thomas. 2020. Convolutional Neural Network. https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network Consultada el 22 de agosto del 2021.

Worthington M. (2021) Visual Elements of User Interface Design. Extraído de: https://www.coursera.org/learn/visual-elements-user-interface-design

Yang, Qiang, *et al*. 2020. Transfer Learning. Nueva York: Cambridge University Press. 227 págs.

XI. ANEXOS

ANEXO 1: LISTADO DE INFECCIONES BACTERIANAS Y POR HONGOS.

1. Infecciones
2. Úlceras bacterianas piógenas
3. Ectima
4. Impétigo
5. Forúnculos
6. Sífilis primaria
7. Micobacterias no tuberculosas
8. Tuberculosis cutánea
9. Escrofulodermia
10. Lupus vulgar
11. Lepra
12. Cromomicosis
13. Esporotricosis
14. Histoplasmosis
15. Lobomicosis
16. Paracoccidioidomicosis
17. Enfermedades inflamatorias y reactivas
18. Úlceras venosas
19. Úlceras arteriales
20. Úlceras vasculares mixtas
21. Infecciones en úlceras vasculares
22. Enfermedades vasculares
23. Úlcera diabética
24. Lupus discoide
25. Psoriasis
26. Necrosis cutánea
27. Anemia de células falciformes
28. Úlceras traumáticas
29. Picaduras de insectos
30. Pioderma gangrenoso
31. Sarcoidosis cutánea
32. Granuloma facial
33. Úlcera banal
34. Tumores malignos
35. Carcinoma basocelular
36. Carcinoma escamocelular
37. Queratoacantoma
38. Linfoma cutáneo
39. Linfoma cutáneo
40. Linfocitoma cutis
41. Hemangioma

42. Sarcoma de Kaposi

Para una breve descripción de cada una de las enfermedades junto con imágenes de ejemplo consulte (OPS, 2020).

ANEXO 2: FICHA CLÍNICA LEISHMANIASIS CUTÁNEA MINISTERIO DE SALUD PÚBLICA Y ASISTENCIA SOCIAL.

Ilustración 110: Ficha clínica Leishmaniasis cutánea parte 1.

MINISTERIO DE SALUD PÚBLICA Y ASISTENCIA SOCIAL
PROGRAMA DE ENFERMEDADES TRANSMITIDAS POR VECTORES
SUBPROGRAMA DE LEISHMANIASIS

FICHA CLÍNICA
LEISHMANIASIS CUTÁNEA Y MUCOCUTÁNEA

A. DATOS GENERALES

Fecha: ___/___/___Nombre completo _____

Fecha de nacimiento:--_____ DPI_____

Edad: _____Sexo: M ☐ F☐ Ocupación: _____

Grupo étnico: _____ Grupo lingüístico: _____

Procedencia: Urbana _____ Rural _____ Migrante_____

Dirección completa (o puntos de referencia):_____

Departamento: _____ Municipio: _____

Aldea:_____ Caserío:_____

Jefe de casa o persona responsable (en caso de menores de edad)_____

Teléfono: _____

¿A viajado o permanecido en Peten, Huehuetenango, Quiché, Alta Verapaz, Baja Verapaz, Izabal, El Progreso?
No ☐
Si ha viajado a una zona que no sea las anteriores, indicar a donde viajó:_____

¿Ha tenido Leishmaniasis anteriormente? Sí ☐ No ☐ No sabe ☐

Indique cuándo (mes – año) _____

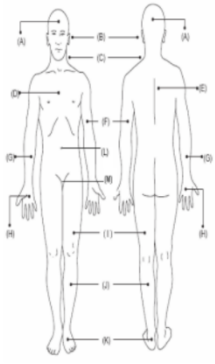
¿Reside en alguna zona endémica de leishmaniasis? Sí ☐ No ☐

Peso: _____lb _____kg

C. PACIENTE SOSPECHOSO DE LEISHMANIASIS CUTÁNEA O MUCOCUTÁNEA

DATOS CLINICOS

Dibuje la localización de las lesiones en el diagrama de abajo



Localización de la(s) Lesión(es)	Cantidad	Diámetro (cm)
A) Cara	☐	_____
B) Orejas	☐	_____
C) Cuello	☐	_____
D) Tórax anterior	☐	_____
E) Tórax posterior	☐	_____
F) Brazos	☐	_____
G) Antebrazos	☐	_____
H) Manos	☐	_____
I) Muslos	☐	_____
J) Piernas	☐	_____
k) Pies	☐	_____
L) Abdomen	☐	_____
M) Genitales	☐	_____
OTROS		

Problemas de salud actuales o recientes
_____

_____

Apariencia de las lesiones: _____

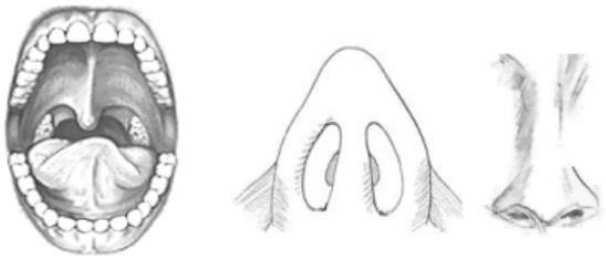
Ilustración 111: Ficha clínica Leishmaniasis cutánea parte 2.

Tiempo de padecer la lesión: _____
¿Cumple con criterios para aplicar termoterapia?

Si _____ No _____

¿Tiene cicatrices anteriores?:Si_____ ¿Dónde?_____No _____

Si tiene lesiones sospechosas en la nariz o boca, señálelas en el diagrama de abajo:



Tratamientos empíricos realizados:

Hierbas o remedios caseros	Si ☐	No ☐
Ácidos	Si ☐	No ☐
Pintura de uñas	Si ☐	No ☐
Quemarse localmente	Si ☐	No ☐
Antibióticos	Si ☐	No ☐

Otros: _____

No sabe _____

DATOS DE LABORATORIO PARA LEISHMANIASIS CUTÁNEA ULCERADA, ATÍPICA Y MUCOCUTÁNEA.

Fecha del diagnóstico (día/mes/año): ___/___/___

FROTIS DIRECTO	PCR	BIOPSIA	CLINICA O NEXO EPIDEMIOLÓGICO	OTRO

Resultado de la evaluación diagnóstica: Positivo para Leishmaniasis cutánea Si ☐ No ☐

Si la respuesta es si, indique:

Leishmaniasis cutánea ulcerada:	Si ☐	No ☐
Leishmaniasis cutánea atípica:	Si ☐	No ☐
Leishmaniasis mucocutánea:	Si ☐	No ☐

Tratamiento administrado y/o procedimientos realizados:
_____
_____

OBSERVACIONES_____
_____
_____

Nombre y cargo de la persona que informa: _____

NOTA: original quedará archivada en el servicio en que se estudió el caso, copias de esta ficha enviarlas a: Dirección de área de salud y al subprograma de Leishmaniasis.

GOBIERNO de GUATEMALA
DR. ALEJANDRO GIAMMATTEI
MINISTERIO DE SALUD PÚBLICA Y ASISTENCIA SOCIAL

ANEXO 3: DATOS A GUARDAR DE FICHA CLÍNICA LEISHMANIASIS CUTÁNEA.

Tabla 21: Información propuesta por parte del ministerio de salud para almacenar.

REGISTRO DE CASOS DE LEISHMANIASIS	
Fecha de registro d/m/a	
Número correlativo anual	
Sexo (M/F)	
Edad (meses y años)	
Distrito de salud	
Ocupación	
Tipo de búsqueda	Activa
	Pasiva
Procedencia	Urbana
	Rural
	Grupo étnico y lingüístico
	Migrante: Indicar de donde proviene
Características clínicas	LC*
	LCA**
	LMC***
	LV****
	Cantidad y diámetro de las lesiones
	Ubicación de las lesiones
	Tiempo de evolución (meses o años)
	Existencia de lesiones cicatrízales (Indique dónde)
Prueba de diagnóstico	Frote Directo o biopsia
	Diagnóstico Clínico o nexo epidemiológico
	PCR*
Tratamiento	Medicamento (antimoniato de meglumine)
	Termoterapia
Condición de egreso	Alta curada
	Alta con mejoría clínica
	Alta no curada
	Abandono
	Fallecido

Información de color verde información indispensable, de color amarillo información importante, pero no indispensable.

ANEXO 4: PREGUNTAS DE ENCUESTA 1: PREGUNTAS EN GENERAL A CUALQUIER COLABORADOR O FUNCIONARIO DEL MINISTERIO DE SALUD.

Información de color verde información indispensable, de color amarillo información importante, pero no indispensable.

LINK ENCUESTA: https://mt8jdfehdsn.typeform.com/to/FUDnAXnt

En las preguntas donde se pide escoger un color u alguna otra opción, existen imágenes de apoyo visual.

Al iniciar la encuesta se le muestra lo siguiente al encuestado:

Somos estudiantes de Ingeniería en Ciencias de la Computación de la Universidad del Valle de Guatemala (UVG). Estamos realizando un proyecto en colaboración con el Centro de Estudios en Salud de la UVG y el Ministerio de Salud Pública y Asistencia Social. El objetivo del proyecto es desarrollar un agente inteligente para el apoyo al diagnóstico de la Leishmaniasis cutánea en regiones endémicas de Guatemala y un sistema para resumir y visualizar gráficamente los datos epidemiológicos sobre la Leishmaniasis cutánea en Guatemala.
Queremos pedir su ayuda respondiendo una encuesta por una sola vez; completarla le tomará 4 minutos.
Usted es libre de decidir si completa o no la encuesta; no habrá consecuencias para usted si se decide no participar. Tampoco habrá beneficios directos para usted por participar, así como tampoco riesgos, ya que no recolectaremos su información personal.

Las preguntas son (todas las preguntas tendrán opciones de respuesta):

1. ¿Qué palabra o frase opinas que representa mejor a la organización de la que eres parte?
2. ¿Qué colores piensas que representan mejor a la organización y a el trabajo que haces?
3. ¿Cuál es tu color favorito?
4. ¿Qué tan cómodo te sientes usando una computadora?
5. ¿Qué paleta o conjunto de colores de los que se muestra te gusta más?
6. El tablero de Leishmaniasis busca informar a usuarios autorizados sobre los casos de Leishmaniasis cutánea registrados. Mostraría gráficas, filtros y algunas otras opciones. * ¿Lo usarías? *
7. ¿Por qué si lo usarías?
8. ¿Por qué no lo usarías?
9. ¿Cuáles íconos de los que se muestran te gustan más?"
10. Por último, ¿alguna sugerencia o comentario para la implementación del Tablero de Leishmaniasis o la aplicación móvil para identificación de lesiones de Leishmaniasis?

ANEXO 5: PREGUNTAS DE ENCUESTA 2: PREGUNTAS ORIENTADAS A LOS COLABORADORES DEL O FUNCIONARIOS DEL MINISTERIO DE SALUD QUE UTILICEN LA APP DE LEISHMANIASIS

En las preguntas donde se pide escoger un color u alguna otra opción, existen imágenes de apoyo visual.
LINK ENCUESTA: https://mt8jdfehdsn.typeform.com/to/O199Sxh0

Al iniciar la encuesta se le informa lo siguiente al encuestado:
Somos estudiantes de Ingeniería en Ciencias de la Computación de la Universidad del Valle de Guatemala (UVG). Estamos realizando un proyecto en colaboración con el Centro de Estudios en Salud de la UVG y el Ministerio de Salud Pública y Asistencia Social. El objetivo del proyecto es desarrollar un agente inteligente para el apoyo al diagnóstico de la Leishmaniasis cutánea en regiones endémicas de Guatemala y un sistema para resumir y visualizar gráficamente los datos epidemiológicos sobre la Leishmaniasis cutánea en Guatemala.

Queremos pedir su ayuda respondiendo una encuesta por una sola vez; completarla le tomará 4 minutos.

Usted es libre de decidir si completa o no la encuesta; no habrá consecuencias para usted si se decide no participar. Tampoco habrá beneficios directos para usted por participar, así como tampoco riesgos, ya que no recolectaremos su información personal.

Las preguntas son (todas las preguntas tendrán opciones de respuesta):

1. ¿Qué modelo y marca de celular posees?
    a. Por ejemplo: Samsung Galaxy A10
2. ¿Qué sistema operativo tiene tu teléfono?
3. ¿Hay señal de internet en la región en la que estás asignado?
4. En la escala de 1 a 5, siendo 1 muy mala y 5 muy buena. ¿Qué tan buena es la señal de internet que tienes cuando estás diagnosticando?
5. ¿Pagas tú el plan de datos de tu teléfono móvil?
6. En caso de que la respuesta anterior sea no, desplegará la pregunta: ¿Quién paga tu plan de datos?
7. ¿Tu plan de datos tiene Whatsapp, Facebook e Instagram ilimitado?
8. ¿Cuántos Gigas de internet tienes disponibles cada mes?
9. ¿Tienes acceso a Wifi en algún momento?
10. ¿Qué tan frecuente tienes acceso a Wifi?
11. ¿Estarías dispuesto a que la aplicación use entre 1% y 3% de tu plan móvil para registrar los resultados del análisis en un servidor remoto?