UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Aplicación de asistencia a conductores de transporte terrestre pesado mediante la detección temprana de sueño por medio de *Machine Learning*: aplicación móvil y modelo de preguntas y respuestas

Trabajo de graduación presentado por Oscar André Paredez Urizar para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Aplicación de asistencia a conductores de transporte terrestre pesado mediante la detección temprana de sueño por medio de *Machine Learning*: aplicación móvil y modelo de preguntas y respuestas

Trabajo de graduación presentado por Oscar André Paredez Urizar para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

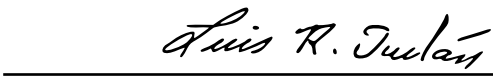
2023

Vo.Bo.:

(f) 

M.A. Luis Roberto Furlán Collver

Tribunal Examinador:

(f) 

M.A. Luis Roberto Furlán Collver

(f) 

M.Sc. Douglas Leonel Barrios Gonzalez

(f) 

M.Sc. Kareen Anasilvia Salazar Morales

Fecha de aprobación: Guatemala, 6 de diciembre de 2023.

Índice

Resumen

La fatiga en pilotos de transporte pesado constituye un serio problema en términos de seguridad vial, ya que puede incrementar las posibilidades de accidentes y comprometer la integridad de conductores, pasajeros y demás usuarios de la carretera. El proyecto de graduación tiene como objetivo desarrollar una aplicación móvil especializada en la recolección de información sobre un piloto de transporte pesado que permitirá alertar al mismo sobre situaciones de fatiga y somnolencia. La aplicación se compone de tres módulos fundamentales: una interfaz móvil, un simulador de copiloto interactivo y un módulo complementario que transmite alertas de detección de fatiga a la aplicación. La aplicación busca mejorar la seguridad vial al mantener a los pilotos alerta y entretenidos, sin ser un distractor, durante sus trayectos, evitando situaciones de riesgo asociadas a la fatiga.

Para abordar este desafío, se llevó a cabo una investigación en áreas clave como la interacción humano-computador, ergonomía cognitiva, experiencia del usuario y usabilidad. Esta labor ha sido fundamental en la creación de una aplicación para dispositivos móviles Android que sea de uso fácil e intuitivo.

La aplicación cuenta con una serie de funcionalidades como:

- el cambio automático de modo diurno y modo nocturno,
- el cambio automático del brillo de la pantalla,
- la activación y desactivación de los parámetros mencionados anteriormente
- una guía auditiva que proporciona instrucciones durante todo el período de navegación,
- la recolección de fotografías del rostro de los pilotos para ser analizadas y determinar si el piloto va alerta o no,
- un simulador de copiloto que recita monólogos relacionados con las categorías de interés seleccionadas por el usuario previo a iniciar el viaje.

La aplicación se conecta a un servidor local que fue desarrollado como un módulo complementario. Dicho servidor procesa señales de sensores, fotografías faciales y ritmo cardíaco,

para determinar el estado de somnolencia del piloto, una vez el viaje ha sido iniciado. El servidor transmite a la aplicación el resultado del análisis para que esta proceda con alertas.

Abstract

Fatigue in heavy transport pilots is a serious problem in terms of road safety, as it can increase the likelihood of accidents and compromise the integrity of drivers, passengers, and other road users. The graduation project aims to develop a specialized mobile application for collecting information about a heavy transport pilot that will alert them to situations of fatigue and drowsiness. The application consists of three fundamental modules: a mobile interface, an interactive co-pilot simulator, and a complementary module that transmits fatigue detection alerts to the application. The application seeks to improve road safety by keeping pilots alert and entertained, without being a distraction, during their journeys, avoiding risk situations associated with fatigue.

To address this challenge, research was conducted in key areas such as human-computer interaction, cognitive ergonomics, user experience, and usability. This work has been fundamental in creating an Android mobile application that is easy and intuitive to use.

The application has a series of functionalities such as:

- the automatic switch between day mode and night mode,
- the automatic adjustment of screen brightness,
- the activation and deactivation of the two parameters mentioned above,
- an auditory guide that provides instructions throughout the navigation period,
- the collection of photographs of the pilots' faces to be analyzed to determine if the pilot is alert or not,
- a co-pilot simulator that recites monologues related to the categories of interest selected by the user before starting the trip.

The application connects to a local server that was developed as a complementary module. This server processes sensor signals, facial photographs, and heart rate to determine the state of drowsiness of the pilot, once the journey has been initiated. The server transmits the result of the analysis to the application so that it can proceed with alerts.

CAPÍTULO 1

Introducción

El proyecto consiste en desarrollar una aplicación para dispositivos móviles Android que contribuya a reducir los riesgos asociados a la fatiga en pilotos durante sus trayectos. Para desarrollar este producto, el enfoque principal se centró en una estética sobria y en una disposición estratégica de las interacciones que facilitan su uso. Estos detalles han sido considerados para mejorar significativamente la experiencia de los pilotos, elevando la calidad y la accesibilidad de la aplicación en su rutina diaria. Se han implementado características específicas para enriquecer la experiencia del usuario, tomando en cuenta el entorno al que se enfrentan como pilotos de transporte pesado. Esto abarca la disponibilidad de modos diurno y nocturno para una visibilidad óptima, una paleta de colores que se adapta a ambos modos, y un cambio de brillo automático que previene la fatiga visual. Los pilotos tienen la flexibilidad de activar o desactivar estas funciones según sus preferencias. También se ha incluido una asistencia por voz en todo el flujo de la aplicación. Adicionalmente, la aplicación tiene una herramienta capaz de tomar muestras faciales, las cuales pueden ser utilizadas para detectar rasgos de somnolencia. Finalmente, se han incorporado botones de fácil acceso y tamaño adecuado para una interacción sin complicaciones, junto a flujos de navegación intuitivos y bien guiados, asegurando así una experiencia fluida y eficaz.

El simulador de copiloto es un componente clave de la aplicación. Se inicia solicitando al piloto sus temas de interés favoritos, como por ejemplo ciencia, deportes, historia, entre otros. A través de interacciones periódicas utilizando el altavoz del dispositivo móvil, el simulador proporciona información del interés del piloto, correspondientes a las categorías seleccionadas inicialmente. Esta función tiene como objetivo principal asegurar que el piloto no se sienta solitario durante el viaje y, al mismo tiempo, proporcionar entretenimiento y compañía durante el trayecto.

La aplicación integra dos módulos adicionales que detectan la somnolencia observando su rostro y su ritmo cardíaco. Estos módulos utilizan técnicas de aprendizaje de máquina para identificar signos de cansancio. En caso de detectar algún indicio, la aplicación emite alertas en forma de voz con el fin de notificar al piloto y fomentarlo a tomar acciones preventivas para mantenerse alerta y seguro en el camino. Para la integración de los módulos mencio-

nados, se desarrolló un servidor complementario que procesa fotografías y ritmo cardíaco, encargándose de enviar señales a la aplicación en caso de que detecte signos de somnolencia a través de estos indicadores.

CAPÍTULO 2

Justificación

La necesidad de abordar la fatiga en los conductores de vehículos de transporte pesado es una prioridad en el ámbito de la seguridad vial. Las largas jornadas y extensas rutas son características habituales de esta profesión, lo que incrementa el riesgo de agotamiento y, por ende, compromete la seguridad no solo del conductor sino también de otros usuarios de la vía. Ante esta problemática, el proyecto que se presenta propone una solución tecnológica destinada a detectar señales tempranas de fatiga en los conductores.

La aplicación móvil diseñada para este fin responde a la necesidad crítica de mantener a los pilotos de transporte pesado alerta durante sus desplazamientos. Se ha prestado especial atención al desarrollo de una interfaz gráfica sencilla e intuitiva, con el fin de facilitar el uso de la misma, minimizar las distracciones y asegurar que la atención del conductor se mantenga en la carretera. Este enfoque se alinea con el objetivo de fortalecer la concentración y la vigilancia, elementos clave para prevenir incidentes y accidentes en el transporte de carga pesada.

La importancia de este proyecto radica en su potencial para preservar la integridad física de los conductores y promover un entorno de tránsito más seguro. Al buscar reducir la incidencia de accidentes causados por la fatiga, la aplicación no solo busca proteger a los conductores, sino que también apunta a tener un impacto positivo en la seguridad general de las carreteras. La implementación de esta herramienta tecnológica puede llegar a representar un paso adelante en la protección de la vida y el bienestar de todos los actores involucrados en el sector del transporte por carretera.

CAPÍTULO 3

Objetivos

3.1. Objetivo general

Diseñar una interfaz móvil intuitiva y funcional capaz de recolectar información sobre pilotos, que permitirá alertarlos sobre situaciones de fatiga y somnolencia. La aplicación debe exhibir un diseño minimalista que facilita una comprensión inmediata, complementado con interacciones sencillas. Asimismo, la aplicación debe de ser capaz de adaptarse a diferentes ambientes de iluminación.

3.2. Objetivos específicos

- Aplicar técnicas y conceptos de ergonomía cognitiva e interacción humano-computador para simplificar la experiencia del usuario, logrando una interfaz limpia, minimalista y fácil de utilizar.
- Realizar una toma de muestras faciales y de ritmo cardíaco al piloto que puedan ser utilizadas para detectar estados de somnolencia.
- Fomentar un sentido de compañía para los pilotos a través de un simulador de copiloto que permita conversar sobre diferentes temas de interés para los mismos.

CAPÍTULO 4

Marco teórico

La tecnología puede desempeñar un papel crucial en el mantenimiento de la atención de los pilotos durante sus viajes por carretera. Por ejemplo, la implementación de sistemas de alerta que monitorean la atención del piloto y lo avisan en caso de distracción puede reducir significativamente el riesgo de accidentes. Estas alertas no solo ayudan a dirigir la atención del piloto de vuelta a la tarea principal de conducir, sino que también pueden contribuir a crear conciencia sobre la importancia de la atención plena al volante.

Además, utilizar herramientas tecnológicas puede registrar datos sobre el comportamiento del piloto, como sus movimientos oculares y el uso de dispositivos, para proporcionar retroalimentación sobre sus hábitos de conducción y niveles de distracción. Esta información no solo puede mejorar la comprensión de los pilotos sobre los riesgos asociados con la conducción distraída, sino que también puede fomentar prácticas más seguras al volante. En resumen, la tecnología ofrece herramientas que pueden mantener la atención de los pilotos en la carretera y mejorar su conciencia sobre la importancia de estar plenamente concentrados mientras conducen [1].

Es crucial tomar precauciones al implementar medidas tecnológicas diseñadas para mantener a los conductores alerta durante sus viajes. Aunque estas medidas tienen como objetivo mejorar la seguridad en la carretera, algunas de ellas pueden resultar contraproducentes al generar distracciones que aumentan el riesgo de resultados no deseados. Por ejemplo, el uso del teléfono celular mientras se conduce debe ser manejado con precaución para evitar cualquier tipo de distracción. La interacción con la pantalla del teléfono puede provocar una especie de ceguera por falta de atención, lo que impide que los conductores perciban información crucial en su entorno y compromete su capacidad para reaccionar ante situaciones de peligro.

Por lo tanto, es esencial que los conductores se concentren exclusivamente en la tarea principal: conducir. Aunque hay conductores capaces de realizar múltiples tareas simultáneamente, conocidos como *supertaskers*, constituyen una minoría en la población. Por ende, no resulta práctico implementar tecnologías que exijan una interacción constante entre los conductores y sus dispositivos. Al limitar esta interacción, se permite que los conductores en-

foquen completamente su atención en la conducción, reduciendo así el riesgo de distracciones y aumentando la seguridad en el camino [2] .

Un análisis más detallado revela que la sobrecarga de áreas cerebrales como la corteza frontal durante el viaje puede tener consecuencias significativas en la capacidad de los conductores para mantener la atención y reaccionar ante estímulos externos. Estudios han demostrado que la distracción causada por la interacción con dispositivos móviles durante la conducción puede afectar negativamente el rendimiento del conductor y aumentar el riesgo de accidentes. Por lo tanto, es fundamental adoptar medidas que minimicen estas distracciones y promuevan una conducción segura y enfocada [3].

Para la creación de la aplicación, es crucial escoger tecnologías y lenguajes específicamente adaptados al mundo móvil. Se ha seleccionado React Native como la plataforma de elección. Este *framework* de Javascript, reconocido mundialmente por su robustez en el ámbito del desarrollo móvil, proporciona un balance idóneo entre rapidez de implementación y versatilidad. Esta plataforma, caracterizada por permitir una única base de código adaptable a múltiples sistemas operativos, consolida y agiliza el ciclo de desarrollo[4].

El *framework* de React Native cuenta con dos entornos de desarrollo. Uno de ellos es React Native CLI, el cual es el seleccionado para formar parte de la base de este proyecto. El motivo por el cual se ve este entorno como el ideal para este proyecto es por el control total y la configuración personalizada que éste permite. Las dependencias nativas de Android pueden ser manipuladas directamente, lo cual puede ser de gran utilidad debido a que se desean implementar herramientas que puedan llegar a requerir manipulaciones y configuraciones manuales, como lo son las librerías de cámara, *GPS*, texto a voz, etc. Adicionalmente, React Native CLI no impone restricciones en cuanto a compatibilidad con librerías, como puede llegar a ocurrir con Expo, el otro entorno existente. Finalmente, React Native CLI permite que las aplicaciones creadas con este entorno sean más pequeñas en tamaño, ya que solamente se incluyen las dependencias estrictamente necesarias [4].

Adicionalmente, React Native resalta por su capacidad de integración con los recursos nativos del dispositivo. El acceso directo a herramientas como la cámara y el *GPS* es invaluable dado el enfoque de este proyecto. Esta interacción ininterrumpida con el hardware del dispositivo, fortalece las funcionalidades de la aplicación y garantiza una experiencia de usuario excepcional[4].

Una interfaz de usuario adecuadamente diseñada es esencial para garantizar la efectividad y seguridad de cualquier aplicación, pero más aún cuando está destinada a pilotos de transporte pesado. La claridad y el minimalismo van más allá de la estética de la aplicación. Es necesario tomar en cuenta estos aspectos por temas de funcionalidad y seguridad. Dentro del escenario cambiante y con los inevitables riesgos que se corren en la carretera, es fundamental contar con botones de tamaño amplio y fácilmente identificables, ya que esto permite una interacción ágil y certera, reduciendo las distracciones visuales. Al momento de elegir las paletas de colores, tanto para un modo diurno como para un modo nocturno es necesario buscar que los textos y las interacciones sean legibles y entendibles en diferentes ambientes lumínicos. Asimismo, es importante que transmitan comodidad a la vista, tanto individualmente como en grupo. Mientras que durante el día se pueden emplear colores de textos y botones más oscuros sobre fondos claros para reducir el deslumbramiento, por la noche es vital adoptar colores suaves que no sean deslumbrantes para el ojo humano, evitando así la

fatiga ocular y posibles distracciones[5].

La conducción, en sí misma, ya demanda una atención plena. Por ello, funciones como el modo nocturno automático o el ajuste adaptable del brillo son un imperativo. Librar al conductor de ajustes manuales no sólo le permite centrarse en el camino, sino que añade un nivel extra de seguridad a su trayecto. Por si fuera necesario, igualmente la aplicación es lo suficientemente flexible como para deshabilitar estas funcionalidades manualmente.

El control por voz es otro elemento innovador que puede marcar la diferencia en términos de seguridad y comodidad. Puede ofrecer direcciones claras y anticipadas con el fin de facilitar el entendimiento de las pantallas. Es claro que, cuando se diseña una herramienta de este calibre, hay que apostar por un diseño que se alinee con sólidos principios de interacción, practicidad y simplicidad, buscando que, al final del día, el usuario cuente con una herramienta que le resulte intuitiva y confiable[6].

El desarrollo de aplicaciones móviles de alto rendimiento es un área que se encuentra muy de moda y está siendo altamente demandada. Cada técnica y estrategia adoptada para alcanzar niveles de rendimiento altos puede marcar la diferencia entre una experiencia de usuario óptima y una mediocre. Asimismo, ofrecer una aplicación que se mantiene fluida a lo largo de sus interacciones y navegaciones significa satisfacción por parte del usuario. En medio de este panorama, la memorización en React Native, una de las técnicas más utilizadas para este propósito en específico, tiene aplicaciones prácticas más allá del mero almacenamiento de resultados de cálculos; se extiende al ámbito de la renderización de componentes. En este contexto, cuando un componente se memoriza, React Native conserva el resultado renderizado del componente hasta que sus propiedades o estados cambien. Una vez que estos cambian, el componente se vuelve a memorizar y, por ende, se vuelve a renderizar. Este proceso es vital, especialmente para componentes con renderizados computacionalmente intensivos, ya que evita renderizados innecesarios y conserva el estado previo del componente. De esta manera, la memorización no solo contribuye a la optimización del tiempo de ejecución, sino que también minimiza el consumo de recursos, una preocupación esencial en el mundo de la movilidad donde la optimización de recursos es crucial[7].

El módulo de toma de muestras faciales es una herramienta que tiene enfoque en poder proveer dichas fotografías para detectar estado de somnolencia. En este proceso, se captura una serie de fotografías durante diez segundos del rostro del piloto. Dicho proceso requiere de una buena iluminación de ambiente para que la cámara del dispositivo sea capaz de captar correctamente el rostro del usuario.

El uso de la cámara de un dispositivo móvil se ha convertido en una herramienta esencial para diversas aplicaciones, y en el contexto de la detección de somnolencia, su relevancia es incuestionable. El acceso a estas cámaras, el cual es bastante accesible gracias a las herramientas que React Native CLI proporciona, nos brinda la oportunidad de capturar retratos en tiempo real del usuario en una variedad de estados. Una ventaja adicional que React Native aporta es la posibilidad de ajustar la calidad con la que se desea que la cámara capture y reconozca, permitiendo una adaptabilidad según las necesidades del análisis facial. Esta habilidad de poder capturar imágenes del conductor en diferentes etapas de su jornada es fundamental para establecer parámetros precisos de alerta, especialmente en entornos de manejo prolongado donde la fatiga puede ser gradual y sutil[8].

El simulador de copiloto representa una estrategia innovadora para mantener a los pilotos alerta y entretenidos durante sus viajes. Se busca que la interacción que esta funcionalidad realice sea de tal manera que no sea para nada invasiva, buscando que su interacción sea en intervalos prudentes de tiempo. Se busca que el usuario perciba esta herramienta como una funcionalidad agradable y atractiva auditivamente.

Permitir al piloto seleccionar sus temas favoritos para posteriormente escuchar datos curiosos sobre los mismos hacen que su atención se mantenga despierta. Escuchar información sobre temas que representan interés para el oyente representa una mayor actividad cerebral en las áreas asociadas con la atención y la memoria. Por lo tanto, esto colabora a mantener al usuario motivado y enfocado [9]. Esta efectividad igualmente está respaldada por evidencia científica que sugiere que la sensación de contar con la compañía de un copiloto puede contribuir significativamente a que el piloto se mantenga activo durante la conducción[10]. Buscar reducir lo más posible esa sensación de soledad que puede llegar a sentir el piloto durante su viaje es de suma importancia para mantener al piloto activo mentalmente.

Incluir datos relacionados con el viaje se presenta como un elemento de interés para cualquier piloto, especialmente cuando éste se dedica a realizar viajes largos. El cálculo del tiempo consiste en manejar un contador que incrementa por uno cada segundo. Dicho contador se pausa cuando el usuario pausa el viaje, y se reanuda cuando éste reanuda su viaje. Por otra parte, para llevar a cabo un cálculo preciso de la distancia que el piloto ha recorrido en su travesía, se necesitan dos puntos geográficos, el punto anterior y el punto actual. Se necesita saber la latitud y la longitud de cada punto geográfico. Existen varios métodos para calcular distancia, como lo son las fórmulas de geometría esférica, ya que la Tierra es una esfera. Un método común para realizar este cálculo es la fórmula del haverseno. Este método consiste en iniciar convirtiendo las coordenadas de latitud y longitud de ambos puntos de grados a radianes. A continuación se calculan las diferencias entre las coordenadas correspondientes. El siguiente paso es aplicar la fórmula del haverseno[11].

1. $a = sin^2(\frac{\Delta lat}{2}) + cos(lat_1) \cdot cos(lat_2) \cdot sin^2(\frac{\Delta long}{2})$
2. $c = 2 \cdot atan2(\sqrt{a}, \sqrt{1-a})$
3. $d = R \cdot c$

Donde:

- $\Delta lat = lat_2 - lat_1$
- $\Delta long = long_2 - long_1$
- lat_1, $long_1$ *son las coordenadas de latitud y longitud del primer punto.*
- lat_2, $long_2$ *son las coordenadas de latitud y longitud del primer punto.*
- R *es el radio de la Tierra.*
- d *es la distancia entre dos puntos a lo largo de la superficie de la esfera.*

Figura 1: Fórmula del haverseno para calcular distancia entre dos puntos[11]

En cuanto al módulo complementario capaz de emitir alertas de detección de somnolen-

cia, se optó por la utilización de Pydroid 3. Esta aplicación disponible para Android es un entorno de desarrollo integrado para Python. Esta aplicación permite a los usuarios desarrollar y ejecutar código Python, desde una simple impresión de un "hola mundo"hasta correr un servidor local. En este caso, se combinó la capacidad de Pydroid 3 para correr código Python con una herramienta llamada Flask. Esta herramienta es un micro framework para desarrollo web en Python. Se destaca por ser ligero y modular. Flask proporciona la capacidad de levantar un servidor integrado, que puede ser utilizado para probar aplicaciones localmente. Este servidor puede proveer rutas para definir puntos de conexión y comunicación. Esta capacidad de correr un servidor local dentro del dispositivo Android permite la fácil comunicación entre dicho servidor y la aplicación. Es decir, la facilidad de la aplicación de recibir señales por parte del servidor es beneficiosa[12].

CAPÍTULO 5

Metodología

Se contempló una metodología estructurada en cinco etapas para la planificación, diseño, desarrollo y pruebas de la aplicación móvil.

En la primera etapa, se llevó a cabo una revisión exhaustiva de la literatura científica y técnicas relacionadas con la detección de somnolencia en conductores. Se profundizó en los conceptos de diseño de interfaces amigables para aplicaciones móviles, considerando aspectos clave como la interacción humano-computador, ergonomía cognitiva y experiencia de usuario. También se investigó sobre métodos en los cuales aplicaciones móviles pueden tener contacto con modelos de aprendizaje de máquina, para que ambas soluciones tecnológicas interactúen entre sí. Esta revisión bibliográfica permitió establecer una base sólida de conocimientos y buenas prácticas esenciales para el desarrollo del proyecto. Además, se exploraron estudios previos que hayan integrado técnicas similares en aplicaciones prácticas, para identificar potenciales desafíos y soluciones comprobadas.

En la segunda etapa, se procedió con la configuración del ambiente de desarrollo para aplicaciones React Native. Primero se instaló Homebrew, que es un manejador de paquetes para computadoras Apple. Posteriormente se instalaron las siguientes dependencias:

- Node, el cual es un entorno de ejecución para JavaScript, con el comando `brew install node`,
- Watchman, un servicio de monitoreo de archivos, con el comando `brew install watchman`,
- Java Development Kit (JDK), un conjunto de herramientas de software necesarias para desarrollar aplicaciones en Java, con los comandos `brew tap homebrew/cask-versions` y `brew install -cask zulu17`,
- Android Studio y los paquetes que Android Studio indica dentro de su instalador. La descarga se hizo desde su web oficial.

Posteriormente, se actualizó la variable de entorno `JAVA_HOME` con la ruta donde JDK se instaló. Se configuraron las variables de entorno en el archivo `~/.zshrc` que Android Studio requiere de la siguiente manera:

- `export ANDROID_HOME=$HOME/Library/Android/sdk`
- `export PATH=$PATH:$ANDROID_HOME/emulator`
- `export PATH=$PATH:$ANDROID_HOME/platform-tools`

A continuación, se inicializó el proyecto utilizando React Native y React Native CLI como el entorno de desarrollo con el comando npx react-native init driverdrowsinessdetector. Se conectó el dispositivo Android a la computadora y se levantó la aplicación para asegurar que todo estuviera funcionando correctamente. Se optó por utilizar el gestionador de paquetes llamado npm. Primero se corrió npm start, comando que arranca la aplicación, seguido de correr la aplicación desde Android Studio para lanzar la aplicación al dispositivo Android. Se instalaron las primeras librerías para arrancar con el desarrollo, y se fueron instalando más librerías conforme fueron necesarias. Se instalaron las librerías utilizando el comando npm install nombre-de-la-librería.

```
"dependencies": {
  "@adrianso/react-native-device-brightness": "^1.2.7",
  "@react-native-community/geolocation": "^3.1.0",
  "@react-native-community/hooks": "^3.0.0",
  "@react-native-masked-view/masked-view": "^0.2.9",
  "@react-navigation/native": "^6.1.7",
  "@react-navigation/stack": "^6.3.17",
  "geolib": "^3.3.4",
  "react": "18.2.0",
  "react-native": "0.72.4",
  "react-native-camera": "^4.2.1",
  "react-native-gesture-handler": "^2.13.4",
  "react-native-permissions": "^3.9.2",
  "react-native-safe-area-context": "^4.7.4",
  "react-native-screens": "^3.25.0",
  "react-native-sensor-ambient-light": "^0.1.2",
  "react-native-tts": "file:./react-native-tts",
  "react-native-vector-icons": "^10.0.0"
}
```

Figura 2: Lista de librerías utilizadas para el desarrollo de la aplicación

```
<Stack.Navigator
  screenOptions={{
    headerShown: false,
    cardStyle: {backgroundColor: theme.background}
  }}
>
  <Stack.Screen name="Home" component={HomeScreen} />
  <Stack.Screen name="FaceCalibration" component={FaceCalibrationScreen} />
  <Stack.Screen name="SelectCategories" component={SelectCategoriesScreen} />
  <Stack.Screen name="Trip" component={TripScreen} />
  <Stack.Screen name="TripDone" component={TripDoneScreen} />
  <Stack.Screen name="Settings" component={SettingsScreen} />
</Stack.Navigator>
```

Figura 3: configuración de navegación de la aplicación

Posteriormente se procedió con el diseño e implementación de la aplicación móvil. Se utilizaron las librerías de react-native-screens y react-navigation, junto con sus extensiones para establecer el patrón de navegación de la aplicación. El código a continuación representa la configuración de la navegación de la aplicación, estableciendo un navegador de pila de react-navigation que organiza y permite la transición entre las pantallas, cada una asociada con su respectivo componente:

Se trabajaron las pantallas conforme al flujo que la aplicación tendrá, iniciando por la pantalla de inicio. En la pantalla de inicio se enfocó en proveer al usuario con dos simples acciones, lo más claras y accesibles posible. Se colocó un botón que ocupa la mitad superior de la pantalla, con la opción de iniciar un viaje. En la mitad inferior de la pantalla, se colocó un botón con opción a navegar hacia ajustes. Los textos que incluyen los botones fueron colocados en tipografías grandes, acompañados de un ícono que va de acuerdo a la acción a realizar.

Para la pantalla de ajustes, se colocaron nuevamente dos botones grandes, donde nuevamente cada uno ocupa la mitad de la pantalla verticalmente. El botón de arriba brinda la opción de activar y desactivar el cambio de modo automático. Es decir, si se desactiva, la aplicación ya no cambia de modo diurno a nocturno automáticamente, y viceversa. El botón de abajo brinda la opción de activar y desactivar el cambio de brillo de la pantalla automático. Cabe recalcar que para poder realizar tanto el cambio de modo diurno a nocturno, y viceversa, automático, como el cambio de brillo automático, se empleó una librería llamada react-native-sensor-ambient-light, la cual utiliza el sensor de iluminación del dispositivo Android para determinar la luz de ambiente que recibe el dispositivo. Igualmente, para establecer el valor de brillo de la pantalla se utilizó la librería de @adrianso/react-native-device-brightness para tener acceso a esta definir dicho valor del dispositivo móvil. En base a dicho nivel de iluminación que detecta el sensor del dispositivo, se determinó un umbral óptimo para determinar qué es mucha iluminación en el ambiente y qué es poca iluminación en el ambiente. Este valor de iluminación lo recibe la aplicación constantemente. Cuando el valor de iluminación es menor al umbral, se establece el modo oscuro automático y se baja

el brillo de la pantalla. Cuando el valor de iluminación en el ambiente es mayor al umbral establecido, se activa el modo diurno automático y se establece el brillo de la pantalla al máximo.

```
useEffect(() => {
  if (Platform.OS === 'android') {
    startUpdateLightSensor();

    let previousValue = null;

    // Suscribirse al cambio de iluminación en el ambiente detectado por el sensor
    const subscription = DeviceEventEmitter.addListener(
      'AmbientLightSensor',
      (data) => {
        const lighting = Math.round(data.lightValue);

        if (
          (automaticToggleDarkMode || automaticBrightnessChange) &&
          (previousValue === null || Math.abs(lighting - previousValue) > 3)
        ) {
          if (automaticToggleDarkMode) {
            // Activar el modo oscuro si la iluminación es mejor a 10 (umbral establecido)
            setShouldActivateDarkMode(lighting < 10);
          }
          previousValue = lighting;
          if (automaticBrightnessChange) {
            if (lighting < 10) {
              // Bajar el brillo de la pantalla si la iluminación es mejor a 10 (umbral establecido)
            setShouldActivateDarkMode(lighting < 10);
              DeviceBrightness.setBrightnessLevel(0.1);
            } else {
              // Establecer el brillo máximo a la pantalla
              DeviceBrightness.setBrightnessLevel(1);
            }
          }
        }
      },
    );

    return () => {
      subscription.remove();
      stopUpdateLightSensor();
    };
  }
}, [automaticToggleDarkMode, automaticBrightnessChange]);


// Definir si se renderizará la aplicación con modo diurno o nocturno en base al valor de la línea 19
const theme = shouldActivateDarkMode ? dark : light;
```

Figura 4: Código responsable de alternar entre el modo diurno y nocturno, y de cambiar el brillo de la aplicación

Para la pantalla de calibración facial, a la que se accede tras presionar el botón de iniciar viaje en la pantalla de inicio, se desarrolló una solicitud de acceso a la cámara del dispositivo si nunca antes se había dado. Se utilizó la librería de react-native-permissions para gestionar las solicitudes de permisos directamente en el dispositivo Android. Al aceptar los permisos, se trabajó una visualización de la cámara para que el usuario se asegure que el dispositivo

se encuentra correctamente posicionado antes de iniciar con la toma de muestras. Se colocó un botón en la parte de abajo para que el usuario pueda iniciar con dicha toma de muestras manualmente, y se colocó un indicador del tiempo restante para la toma de muestras.

```
import { request, PERMISSIONS, RESULTS } from 'react-native-permissions';
...
useEffect(() => {
  const requestPermission = async () => {
    let granted = false;
    if (Platform.OS === 'android') {
      await request(PERMISSIONS.ANDROID.CAMERA) === RESULTS.GRANTED
    }
  };

  requestPermission();
}, []);
```

Figura 5: Código responsable de solicitar permisos de acceso a la cámara del dispositivo

```
const capturePicture = async () => {
  if (cameraRef.current && pictureCount < 5) {
    const options = { quality: 1, base64: true };
    const data = await cameraRef.current.takePictureAsync(options);
    setPictureCount(prev => prev + 1);
  }
};
```

Figura 6: Toma de las cinco fotografías faciales

Para la pantalla de selección de categorías, a la que se navega automáticamente al finalizar la toma de muestras faciales, se desarrolló una lista de categorías, acompañadas de un ícono que las representa, para que el usuario pueda escoger al menos cinco en base a sus preferencias. En la parte inferior de la pantalla se implementó un botón que inicialmente se encuentra inhabilitado. Al finalizar la selección, se activa dicho botón que permite al usuario navegar a la pantalla del viaje.

Para la pantalla de viaje, fue necesario desarrollar una solicitud de permisos de localización del dispositivo, puesto que se necesita tener acceso al GPS para calcular la distancia recorrida. Esta gestión se trabajó igualmente con la librería de react-native-permissions. Posteriormente, se trabajó en una interfaz sencilla, donde en la parte superior se tiene información del viaje, como tiempo transcurrido, distancia recorrida y cantidad de alertas emitidas que indican detección de somnolencia.

```
import { check, PERMISSIONS, request, RESULTS } from 'react-native-permissions';
...
const requestLocationPermission = async () => {
    if (Platform.OS === 'android') {
        const result = await request(PERMISSIONS.ANDROID.ACCESS_FINE_LOCATION);
        return result === RESULTS.GRANTED;
    }
    return false;
};

useEffect(() => {
    const requestPermission = async () => {
        let granted = false;
        if (Platform.OS === 'android') {
            granted = await requestLocationPermission();
            if (granted)
                setLocationPermissionsDenied(false);
            else {
                setActivateLocationPermissionsToast(true)
                setLocationPermissionsDenied(true);
            }
        }
    }
    requestPermission();
}, []);
```

Figura 7: Código responsable de solicitar permisos de ubicación precisa

Para la pantalla en mención, fue necesario implementar una manera de calcular el tiempo transcurrido del viaje y la distancia recorrida por el piloto.

Para calcular el tiempo transcurrido de un viaje, se realizó un contador que incrementa por una unidad durante cada segundo que pasa. Dicho contador se pausa si el viaje es pausado, y se reanuda si el viaje se reanuda.

Para calcular la distancia recorrida por el piloto, es necesario realizar un procedimiento estructurado. Al iniciar el viaje, se obtendrá la latitud y longitud del punto en el que el piloto se encuentra por medio de la librería @react-native-community/geolocation. Posteriormente, en cada cambio detectado de punto geográfico por el dispositivo, se procederá a actualizar los valores de latitud y longitud. A continuación, se utilizará la librería de geolib para facilitar el cálculo de la distancia entre dos puntos dadas sus latitudes y longitudes. Esta librería provee funciones que convierten las coordenadas de latitud y longitud de grados a radianes, y luego aplica la fórmula del haverseno para encontrar la distancia entre dos puntos. Si el cambio de distancia entre el punto anterior y el punto actual supera los cincuenta metros, se actualizará el valor de la distancia recorrida. Este umbral de cincuenta metros se establece para evitar actualizaciones constantes por cada metro recorrido. Por lo tanto, el último paso será convertir dicha distancia en metros a kilómetros para ser mostrado en pantalla.

```javascript
import Geolocation from '@react-native-community/geolocation';
import { getPreciseDistance } from 'geolib';
...
// Esta función maneja el cambio de ubicación
const handleLocationChange = (newLocation) => {
    // Verifica si el viaje ha comenzado y si hay una ubicación previa
    if (isTripStarted && previousLocation) {
        const { latitude: lat1, longitude: lon1 } = previousLocation;
        const { latitude: lat2, longitude: lon2 } = newLocation;
        // Calcula la distancia precisa entre la ubicación previa y la nueva
        const distance = getPreciseDistance({ latitude: lat1, longitude: lon1 }, { latitude: lat2, longitude: lon2 });

        // Verifica si la distancia recorrida desde la última actualización es al menos de 50 metros
        console.log("prev", totalDistance, distance)
        if (distance >= 50) {
            // Convierte metros a kilómetros y actualiza la distancia total
            setTotalDistance(prevDistance => prevDistance + (distance / 1000));
            // Actualiza la ubicación previa con la nueva ubicación
            setPreviousLocation(newLocation);
        }
    } else {
        // Si no ha comenzado el viaje o no hay ubicación previa, actualiza la ubicación previa con la nueva
        setPreviousLocation(newLocation);
    }
};

// Efecto para manejar la ubicación cuando los permisos están concedidos y el viaje ha comenzado
useEffect(() => {
    if (!locationPermissionsDenied && isTripStarted) {
        // Inicia el seguimiento de la posición con alta precisión y un filtro de distanci
        const watchId = Geolocation.watchPosition(
            position => {
                const { latitude, longitude } = position.coords;
                const newLocation = { latitude, longitude };
                if (previousLocation) {
                    const { latitude: lat1, longitude: lon1 } = previousLocation;
                    const { latitude: lat2, longitude: lon2 } = newLocation;
                    const distance = getPreciseDistance({ latitude: lat1, longitude: lon1 }, { latitude: lat2, longitude: lon2 });

                    // Si la distancia es mayor o igual a 50 metros, maneja el cambio de ubicación
                    if (distance >= 50) {
                        handleLocationChange(newLocation);
                        setPreviousLocation(newLocation);
                    }
                } else {
                    // Si no hay ubicación previa, simplemente maneja el cambio de ubicación
                    handleLocationChange(newLocation);
                    setPreviousLocation(newLocation);
                }
            },
            error => {
                console.error(error);
            },
            { enableHighAccuracy: true, distanceFilter: 10 }
        );
        return () => {
            Geolocation.clearWatch(watchId);
        };
    }
}, [previousLocation, isTripStarted]); // Dependencias del efecto
```

Figura 8: Código responsable de llevar el control de la distancia recorrida

En la parte de en medio se implementaron cuatro botones, los cuales permiten iniciar, pausar, reanudar y finalizar el viaje. Inicialmente se presenta el botón de iniciar viaje, lo que permite la iniciación del contador de tiempo transcurrido y distancia recorrida. Una vez iniciado el viaje, desaparece el botón de iniciar viaje y se renderizan los botones de pausar viaje, el cual pausa los contadores, y el que permite finalizar el viaje. Si se presiona el botón de pausar viaje, dicho botón desaparece y se muestra el botón de reanudar viaje. Finalmente, se desarrolló la funcionalidad de que si el botón de finalizar viaje se presiona, la aplicación automáticamente redirige a la pantalla de fin de viaje. Adicionalmente, se integró la conexión con el servidor que emite señales en caso de detectar somnolencia. Dicha conexión

se establece mientras un viaje se encuentra en curso. En dado caso se recibe una alerta de somnolencia, se desarrolló un ícono de alerta arriba de los botones, indicando que se detectó fatiga en el piloto. Igualmente, se implementó la funcionalidad de emitir un mensaje de voz con cada alerta recibida, para ser un máximo de tres alertas. Después de las tres alertas, la aplicación deja activo el ícono de alerta e indica que las tres alertas de somnolencia ya fueron lanzadas. Finalmente, en la parte inferior derecha se colocó una pequeña visualización de lo que el piloto puede observar en base a la cámara de su dispositivo.

```
const takePhoto = async () => {
    if (cameraRef) {
        const options = { quality: 0.5, base64: false };
        const data = await cameraRef.current.takePictureAsync(options);
        return data.uri;
    }
};

const fetchData = async () => {
    try {
        const imageUri = await takePhoto();
        const formData = new FormData();
        formData.append('image', {
            uri: imageUri,
            name: 'photo.jpg',
            type: 'image/jpeg',
        });

        // Realizar consulta tipo POST para enviar la fotografía al servidor
        const response = await fetch('http://127.0.0.1:5000/predict', {
            method: 'POST',
            headers: {
                'Content-Type': 'multipart/form-data',
            },
            body: formData,
        });

        const data = await response.json();

        if (!response.ok) {
            throw new Error('Network response was not ok');
        }

        if (response.status === 200) {
            // Si el servidor detecta que los ojos de la fotografía están cerrados, suma 1 al contador de alertas
            if (data.message === "closed") {
                setClosedEyesCounter(prevCounter => prevCounter + 1);
            // Si el servidor no detecta los ojos, suma 1 al contador de fotografías incorrectas y alerta esto
            } else if (data.message === "not_recognized") {
                setEyesNotFoundCounter(prevCounter => prevCounter + 1);
            }
        }

    } catch (error) {
        console.error('Error:', error);
    }
};
```

Figura 9: Petición al servidor con la fotografía facial adjunta

Finalmente, se trabajó en la pantalla de viaje finalizado. En dicha pantalla se implementó una vista en donde se listan las métricas obtenidas del viaje, seguido de un botón que redirige a la pantalla de inicio.

Es importante mencionar que para todas las pantallas se consideró mantener la consistencia en cuanto a los colores y diseños utilizados, con el fin de darle identidad a la aplicación y brindar una mejor experiencia de usuario. Es decir, la cantidad de colores y diseños de vistas que utiliza la aplicación son limitados. Igualmente, se colocó guía por voz en todas las pantallas, en donde al entrar a cualquiera de ellas, se mencionan las posibles interacciones que se pueden realizar.

Además, se implementaron todas las técnicas necesarias para mantener altos estándares de rendimiento en la aplicación. Esto se logrará gracias a la técnica de memorización de componentes y funciones que requieran altos niveles de computabilidad, con el fin de prevenir renderizados innecesarios a lo largo de todas las pantallas del producto final. En virtud de que el dispositivo móvil estará sometido a una carga considerable de tareas, que serán detalladas a continuación, resulta imperativo implementar estrategias de alto rendimiento en la aplicación.

```
// Volver a renderizar el componente CategoryCard solamente si sus propiedades cambian
const propsAreEqual = (prevProps: any, nextProps: any): boolean => {
    return (
        prevProps.name === nextProps.name &&
        prevProps.icon === nextProps.icon);
};

export default React.memo(CategoryCard, propsAreEqual);
```

Figura 10: Ejemplo de memorización en la aplicación

La tercera etapa fue el desarrollo de un modelo de preguntas y respuestas. La idea que se tenía era utilizar la pantalla de selección de categorías, ya que se quería tener una serie de preguntas para cada una de las categorías, así como su respuesta. Periódicamente, en la pantalla de viaje, la aplicación lanzaría dichas preguntas, y las mismas debieran ser respondidas por el piloto por medio de voz. Se generaron preguntas cuyas respuestas fueran de solamente una palabra. Al recibir la respuesta, la aplicación convertiría dicho audio a texto, y evaluaría si la respuesta brindada por el usuario era la correcta con respecto a la respuesta que se tenía almacenada correspondiente a esta pregunta. Cabe mencionar que tanto la librería que se iba a utilizar para convertir la pregunta de texto a voz, como la librería que iba a servir para convertir la respuesta del usuario de voz a texto aplican herramientas de aprendizaje de máquina para ser capaces de cumplir con su función. Es decir, dichas librerías cuentan con modelos entrenados capaces de analizar los textos para convertirlos en audio y los audios para convertirlos a texto.

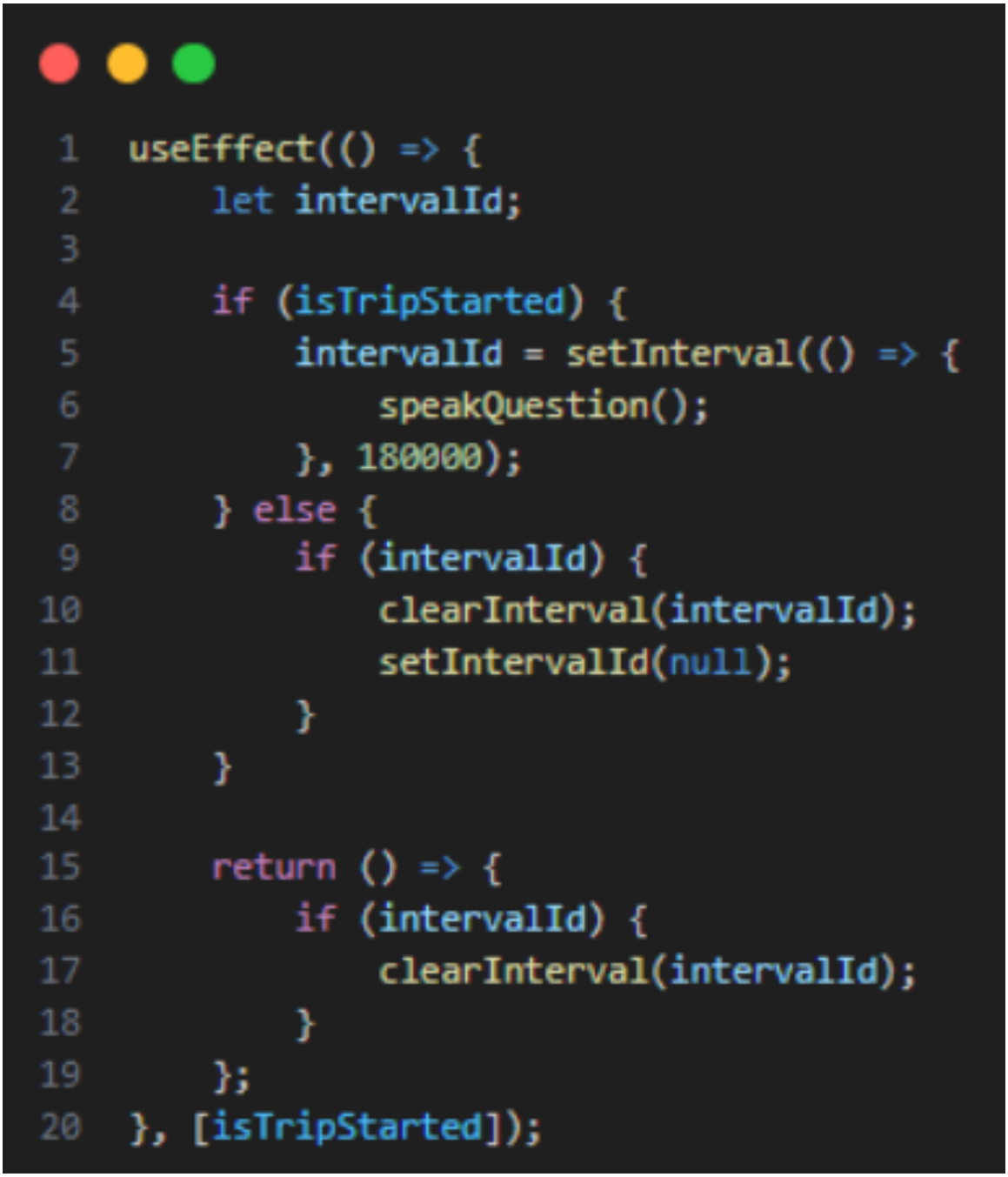
Conforme se fue desarrollando dicha solución, se asimiló que esta no era la ideal por dos motivos. El primer motivo es que el esfuerzo cognitivo, como es el caso de responder a preguntas, puede aumentar la somnolencia[13]. En otras palabras, puede que la funcionalidad anterior sí llegara a reducir el sentido de soledad dentro de una cabina de un vehículo. Sin embargo, esta no iba a colaborar en el esfuerzo de brindar una herramienta que fuera capaz de intervenir en momentos adecuados para mantener alerta y concentrado al piloto. El segundo motivo fue por cuestión de usabilidad. Mezclar la recitación de preguntas, con la aplicación

detectando respuestas, con las alertas por voz indicando somnolencia podía llegar a generar muchos temas simultáneamente para el piloto. Esto podría llegar a abrumar al usuario y, considerando que al mismo tiempo pudiera llegar a escuchar una alerta de somnolencia por medio de voz, fue un factor que definitivamente no se quería que sucediera.

Por el contrario, buscando brindar un sentido de compañía sutil y para nada intrusivo, se decidió optar por cambiar ligeramente el concepto del simulador de copiloto. El enfoque elegido permite al piloto elegir entre una lista variada de categorías antes de iniciar su viaje, solicitando seleccionar al menos cinco de ellas. A través de las categorías seleccionadas, se determina cuáles son los temas que captan mejor la atención y el interés del piloto, garantizando así una experiencia individualizada y enfocada en sus preferencias. El simulador de copiloto recita monólogos de entre quince y treinta segundos de duración sobre datos interesantes correspondientes a los temas seleccionados por el usuario. Esta acción sucede cada tres minutos, buscando brindar una herramienta que no se sienta como intrusiva e insistente, sino que al contrario, active al usuario. Se empleó la librería de react-native-tts para incorporar la funcionalidad de la locución de los monólogos.

```
const getRandomIntroduction = () => {
    const randomIndex = Math.floor(Math.random() * introductions.length);
    return introductions[randomIndex];
};

const getRandomPrompt = () => {
    const randomIndex = Math.floor(Math.random() * categories.length);

    const selectedCategoryId = categories[randomIndex];
    const selectedCategory = data.find(category => category.name === selectedCategoryId);

    if (!selectedCategory) {
        return null;
    }
    const prompts = selectedCategory.prompts;
    const randomPromptIndex = Math.floor(Math.random() * prompts.length);
    const randomPrompt = prompts[randomPromptIndex];
    const categoryName = selectedCategory.name;

    // Devolver el prompt aleatorio con el nombre de la categoría
    return `${categoryName}: ${randomPrompt}`;
}

const speakQuestion = async () => {
    if (shouldSpeak) {
        const introduction = getRandomIntroduction();
        const prompt = getRandomPrompt();
        const fullPrompt = introduction + "  " + prompt;

        // Iniciar la librería TTS para hablar
        Tts.getInitStatus().then(() => {
            Tts.stop(); // Detener cualquier TTS anterior
            setIsSpeaking(true) // Establecer el estado de habla a verdadero
            Tts.speak(fullPrompt); // Hablar el prompt

            // Escuchar cuando la locución ha terminado
            const finishListener = (event) => {
                console.log("finish", event);
                setIsSpeaking(false) // Establecer el estado de habla a falso
                Tts.removeEventListener('tts-finish', finishListener); // Remover el listener
            }

            // Añadir el listener para el evento de finalización de TTS
            Tts.addEventListener('tts-finish', finishListener);

        });
    }
};
```

Figura 11: Selección de monólogo y locución del mismo



```
useEffect(() => {
    let intervalId;

    if (isTripStarted) {
        intervalId = setInterval(() => {
            speakQuestion();
        }, 180000);
    } else {
        if (intervalId) {
            clearInterval(intervalId);
            setIntervalId(null);
        }
    }

    return () => {
        if (intervalId) {
            clearInterval(intervalId);
        }
    };
}, [isTripStarted]);
```

Figura 12: Código encargado de ejecutar la figura anterior cada tres minutos

Esta medida, aparte de buscar mantener al piloto alerta y comprometido durante el trayecto, también tiene como objetivo reducir la sensación de aislamiento y soledad que algunos conductores pueden experimentar durante largos períodos al volante. Además, este enfoque ayuda a crear un ambiente más amigable y relajado en la cabina, facilitando que el piloto se sienta más en control y menos abrumado[14].

La manipulación y adaptación de la biblioteca encargada de la transcripción de texto

a voz, react-native-tts, surgió como otro desafío de gran relevancia en la tercera fase. Fue necesario ajustar el comportamiento de esta librería en lo que respecta a la detección de inicialización y finalización de la transmisión vocal, permitiendo así ejecutar acciones específicas dentro de la aplicación en respuesta a estos eventos. Afrontar esta modificación fue un reto de baja complejidad, sin embargo, se tornó absolutamente necesario para lograr la integración planificada de la aplicación. La adaptación de esta herramienta resultó ser indispensable en el proceso de desarrollo, asegurando una experiencia de usuario óptima y sin contratiempos en la versión final de la aplicación. Fue necesario descargar toda la carpeta que representa la librería de react-native-tts, realizar los cambios a continuación, incluirla en la ruta del proyecto y hacer referencia a dicha carpeta dentro de la lista de librerías presentadas anteriormente para que el proyecto fuera capaz de reconocer esta implementación. Se referenció la librería de la siguiente manera: react-native-tts": "file:./react-native-tts". En vez de colocar la versión de la librería como valor del par clave-valor tal y como se hizo con el resto de librerías, se colocó la ruta de la carpeta correspondiente.

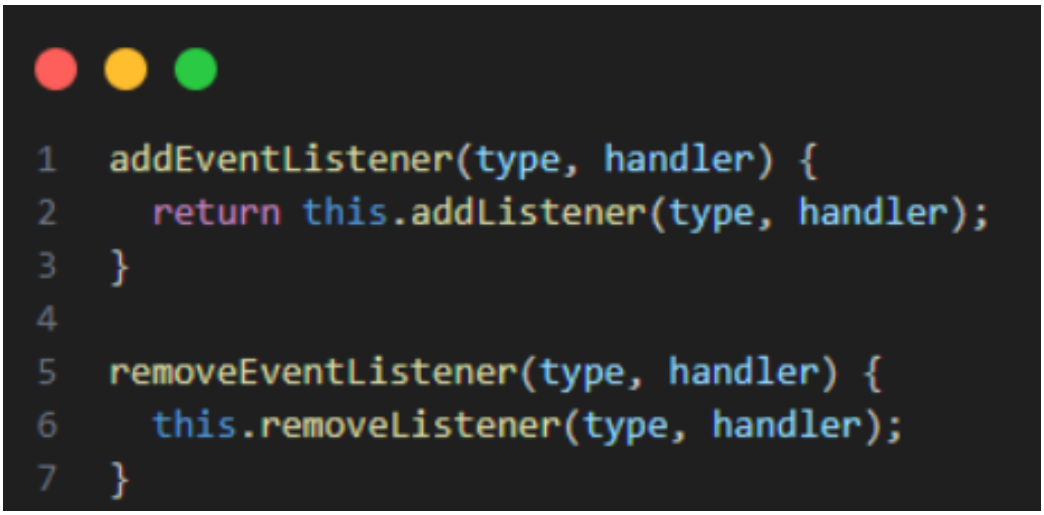


Figura 13: Código implementado por la librería para detectar cuando la voz inicia y finaliza la recitación

Figura 14: Código modificado para detectar cuando la voz inicia y finaliza la recitación

La cuarta fase inició con el enfoque puesto en incorporar modelos de aprendizaje de máquina dentro de la aplicación, proceso necesario para comunicar a la aplicación con los detectores de somnolencia, y por lo tanto, para que la aplicación pudiera recibir señales indicando fatiga. Se tomaron los archivos en formato .bin y .json que se pueden generar al exportar un modelo de aprendizaje de máquina desarrollado con la tecnología Tensorflow, y mediante la documentación oficial de dicha tecnología se siguió una serie de pasos. Se instalaron las dependencias y librerías necesarias para importar los modelos. Posteriormente se realizaron configuraciones específicas para que la aplicación fuera capaz de reconocer dichos archivos y tratarlos en cualquier componente de la aplicación. Tras realizar dichos esfuerzos, no se logró que la aplicación reconociera algún modelo. Según lo retroalimentado por otros usuarios en internet, el problema se debía a un conflicto de versiones entre ciertas dependencias del proyecto con las librerías recién instaladas. Tratar de cambiar versiones para arreglar esto podía implicar romper la aplicación, por lo que se optó por buscar otra solución.

Tras no ser exitosa la implementación original, se buscó una solución alternativa. Se realizó una investigación de formas de comunicar modelos de aprendizaje de máquina, y se encontró con la opción de establecer un servidor. Siempre se tuvo la idea de que no se podía depender de conexión a internet en ningún momento, ya que los pilotos de transporte pesado a menudo pueden verse en situación de falta de señal. Por lo tanto, se optó por realizar una instalación de una herramienta conocida como Pydroid 3 para ejecutar un servidor dentro de un dispositivo Android mediante el uso de un micro *framework* conocido como Flask. En pocas palabras, se instaló la aplicación de Pydroid 3, pagando una membresía para poder obtener la misma. Posteriormente, se instaló Flask en dicha aplicación. Así, se dejó listo el entorno para la configuración del servidor y los demás ajustes necesarios para emitir señales de detección de somnolencia a la aplicación. De esta manera, se logró encontrar una solución que igualmente no dependiera de señal de internet. En resumen, se logró emitir señales de detección de somnolencia a la aplicación. Además, se trabajó en que la aplicación sea capaz de emitir al servidor las imágenes recolectadas en la toma de muestras faciales y las imágenes que se envían continuamente durante el viaje para detectar signos de fatiga en tiempo real.

Es importante mencionar también que inicialmente se quería aprovechar la pantalla de toma de muestras faciales para realizar una toma de muestras del ritmo cardíaco del usuario. Sin embargo, esta implementación también representó muchos bloqueos durante el intento de desarrollo, ya que dicha toma de muestras debía de ser con un reloj inteligente en específico, lo cual redujo la posibilidad de encontrar documentación específica para el dispositivo en mención. Por lo tanto, se decidió aprovechar el servidor local para que este fuera capaz de conectarse al reloj, procedimiento que fue sencillo en comparación con la idea inicial.

En la quinta fase se enfocó en realizar pruebas con usuarios. Se buscó obtener varios puntos de vista relacionados con las herramientas que brinda la aplicación, las paletas de colores utilizadas, el posicionamiento de las interacciones, el flujo de navegación, entre otros. Al recopilar esta información, se pudo realizar un análisis detallado que guió las mejoras y optimizaciones necesarias. Con esto se buscó que el producto final cumpla con todas las exigencias operativas que los conductores puedan llegar a tener.

CAPÍTULO 6

Resultados

El proceso de creación de la aplicación ha culminado en un producto final que combina una interfaz minimalista, intuitiva y de alto rendimiento, adaptable a cualquier tipo de iluminación de ambiente, con flujos sencillos capaces de recolectar información del conductor. Asimismo el producto es capaz de ofrecer un viaje agradable que colabore con reducir la fatiga mientras el piloto se encuentra conduciendo.

Al abrir la aplicación, con lo primero que se encuentran los usuarios es con el control guiado por voz. Esta herramienta básicamente brinda instrucciones de cómo interactuar en cada una de las pantallas. En la pantalla de inicio, los usuarios se encuentran con dos opciones: iniciar un viaje o acceder a la sección de ajustes. En el apartado de configuración, los usuarios tienen la capacidad de activar o desactivar el modo nocturno automático y el cambio de brillo de la pantalla automático, otorgándoles el control para personalizar la aplicación de acuerdo a sus preferencias de visualización. Es relevante destacar que, como se ha mencionado previamente, la aplicación está dotada de la capacidad de cambiar entre modo diurno y nocturno automáticamente y modificar el brillo de la pantalla del dispositivo por sí sola cuando detecta condiciones de baja iluminación en el entorno.

Cuando un piloto decide comenzar su viaje, se le solicitan permisos de acceso a la cámara del dispositivo solamente la primera vez que accede a esta pantalla. Posteriormente es dirigido hacia la pantalla de toma de muestras faciales. En esta pantalla, se le pide al usuario colocar su rostro dentro del círculo que se muestra. Esto se hace para captar muestras precisas. Al presionar el botón de iniciar la calibración, se le toman cinco fotografías del rostro del conductor durante diez segundos. El usuario es capaz de ir viendo los momentos en donde se le toman las fotografías. Es mandatorio tener una buena iluminación en el ambiente para obtener fotografías útiles.

Una vez completada la etapa anterior, el usuario es automáticamente guiado hacia la pantalla de selección de categorías de interés, donde se le anima a elegir al menos cinco áreas temáticas. Estas selecciones determinan los monólogos que el simulador de copiloto presentará una vez el viaje ha sido iniciado. De esta manera, se busca que el contenido recitado sea atractivo para la capacidad auditiva del conductor, fomentando así una mayor

atención. Además, se establece una conexión más cercana entre el piloto y el simulador de copiloto, haciendo la experiencia más agradable y menos monótona.

Una vez seleccionadas las categorías y presionado el botón de continuar, antes de acceder a la pantalla del viaje, se le solicitan permisos de ubicación al dispositivo del usuario. Luego, al entrar a la pantalla dedicada al viaje, se proporciona información esencial del viaje, incluyendo el tiempo transcurrido, la distancia recorrida y la cantidad de advertencias que se le han dado al conductor por presentar síntomas de somnolencia, las cuales son recibidas desde el servidor. Además, se incluye un indicador de sonido que señala cuando el simulador de copiloto se encuentra hablando. Una vez iniciado el viaje, la aplicación captura automáticamente el rostro del piloto y envía la fotografía al servidor cada dos segundos, todo esto operando en segundo plano. Mediante la constante recepción de estas fotografías, el servidor es capaz de interpretar las mismas para saber cuándo emitir una alerta a la aplicación. Si se detecta fatiga, el simulador de copiloto alerta al piloto mediante una notificación de voz y se incrementa en uno el contador de alertas emitidas.

También, por cada detección de fatiga que el servidor emita, la aplicación inmediatamente ofrece una llamada de atención por medio de voz. La primera advertencia le pregunta al usuario "Hola, ¿Todo bien? ¿Necesitas un descanso?". La segunda, con un tono más serio, le dice "¡Oye! Estamos detectando que te estás relajando demasiado. Por favor, toma un descanso ahora."Finalmente, la tercera llamada de atención directamente lo motiva a detener su curso de la siguiente manera: "¡Detente ahora! Estamos detectando que te estás quedando dormido. Tu bienestar es crucial. Tómate un descanso inmediatamente."

En esta misma pantalla, se ofrece la opción de iniciar el viaje, y una vez iniciado, sólo se puede pausar, reanudar o finalizar. Una vez finalizado el viaje, se navega hacia una pantalla en donde se pueden visualizar las métricas del viaje recientemente finalizado, incluyendo la distancia total recorrida, el tiempo total transcurrido y la cantidad total de llamadas de atención que se le dieron al piloto por habérsele detectado fatiga.

Con el propósito de asegurar que la aplicación cumple con los requisitos de los usuarios finales, se desarrolló un conjunto de pruebas de usabilidad. Estas involucran a cinco usuarios estratégicamente seleccionados. Se buscó evaluar la calidad de la experiencia de usuario y la interacción humano-computador. Asimismo, se buscó identificar áreas de oportunidad de mejora para aplicarles refinamiento en caso de ser necesario, con el fin de optimizar la experiencia del usuario.

Profundizando en las pruebas de usabilidad, estas se dividieron en dos. La primera consistió en una serie de cinco tareas que los usuarios debían completar. Estas tareas, cuidadosamente seleccionadas, fueron: desactivación del modo nocturno automático, toma correcta de muestras faciales, selección de cinco categorías específicas, iniciar y pausar un viaje, y finalizar un viaje. Se seleccionó aleatoriamente si debían hacer la prueba en el modo diurno o nocturno de la aplicación, y se midió el tiempo que les tomó completar la tarea, la cantidad de errores que cometieron durante la ejecución de la tarea, los puntos de satisfacción del 1 al 10 que les generó la experiencia y comentarios generales sobre ella.

Según los resultados presentados a continuación, se confirma que la aplicación es de agrado para los usuarios, ya que cumple con muchos de los estándares esperados por los mismos. Las respuestas obtenidas de las pruebas de usabilidad confirman que la aplicación

brinda una experiencia agradable para el usuario debido a su diseño minimalista e intuitivo, la guía por voz que brinda en todas las pantallas del producto y el flujo sencillo de navegación que ofrece. El tiempo requerido para completar cada tarea siendo muy similar entre todos los usuarios y la nula cantidad de errores cometidos durante el desarrollo de las tareas indica que los flujos de la aplicación son concisos y no presentan mayor dificultad para comprender cómo navegar en la aplicación. Los altos puntos de satisfacción que los usuarios dieron durante las cinco tareas indican que la experiencia de usuario es agradable y sencilla. En resumen, los comentarios sugieren una experiencia de usuario completa y positiva en términos de usabilidad y diseño del producto.

Usuario	**Género**	**Edad**	**Opinión del Usuario**
1	Femenino	28	Vivo a 1 hora y media de mi trabajo y madrugo para llegar a tiempo. Me despierto a las 4 de la mañana y la mayoría de veces no ha salido el sol cuando salgo de mi casa.
2	Masculino	22	Durante el tráfico acostumbro a cabecear, por lo que me serviría llevar un app como ésta para ir viendo los datos de mi viaje por pura curiosidad.
3	Masculino	22	Acostumbro a realizar muchos viajes en carro cuando viajo a Estados Unidos. Mi familia tiende a dormirse por lo que esta app podría serme de utilidad.
4	Masculino	24	Debido a mi trabajo viajo mucho en distintos horarios del día y en carretera de ida y vuelta a Xela.
5	Masculino	23	Soy una persona que padece de sueño especialmente cuando manejo mi automóvil. Tanto en la madrugada que me dirijo al trabajo como en la noche que salgo de la universidad, padezco de esto.

Cuadro 1: Información de los usuarios que realizaron pruebas de usabilidad

Usuario	**Modo de la aplicación**	**Tiempo transcurrido (seg)**	**Errores durante la tarea**	**Puntuación de satisfacción (1-10)**	**Comentarios generales**
1	Diurno	6	0	8	Considero que es bastante sencillo acceder a los ajustes
2	Nocturna	4	0	10	Flujo sencillo e intuitivo
3	Diurno	5	0	10	Sencillo y directo
4	Nocturno	6	0	10	Me gusta que los botones sean grandes ya que facilita las interacciones
5	Diurno	7	0	10	Flujo sencillo

Cuadro 2: Tarea 1 de pruebas de usabilidad - Desactivación del modo nocturno automático

Usuario	**Modo de la aplicación**	**Tiempo transcurrido (seg)**	**Errores durante la tarea**	**Puntuación de satisfacción (1-10)**	**Comentarios generales**
1	Diurno	29	0	8	Creo que si fuera automática sería mejor, pero no tengo problemas con que sea manual.
2	Nocturna	20	0	10	Fue un proceso y sencillo, no tuve ningún problema.
3	Diurno	18	0	10	Directo a lo que es, no me complicó nada.
4	Nocturno	21	0	10	Me gusta la animación, se siente moderno.
5	Diurno	24	0	10	Flujo sencillo y fácil de seguir.

Cuadro 3: Tarea 2 de pruebas de usabilidad - Tomar muestras faciales

Usuario	**Modo de la aplicación**	**Tiempo transcurrido (seg)**	**Errores durante la tarea**	**Puntuación de satisfacción (1-10)**	**Comentarios generales**
1	Nocturno	31	0	9	Poner los dibujos más grandes
2	Nocturno	27	0	10	Es sencillo elegir las categorías
3	Nocturno	33	0	8	No sabía que se podía desplazar hacia abajo en la lista de categorías
4	Diurno	34	0	9	No sabía que habían más categorías en la parte de abajo
5	Nocturno	32	0	10	Es sencillo seleccionar las categorías

Cuadro 4: Tarea 3 de pruebas de usabilidad - Seleccionar las Categorías Deporte, Tecnología, Culinaria, Cine y Viajes

Usuario	**Modo de la aplicación**	**Tiempo transcurrido (seg)**	**Errores durante la tarea**	**Puntuación de satisfacción (1-10)**	**Comentarios generales**
1	Nocturno	46	0	10	Pense que ya habia seleccionado mis 5 categorias pero solo llevaba 4. Colocar un contador seria ideal.
2	Diurno	37	0	10	Resaltar más el tiempo y la distancia.
3	Diurno	36	0	10	Sencillo porque solamente se requiere presionar el botón de iniciar viaje,
4	Diurno	38	0	9	Me gusta la interfaz sencilla de la pantalla de viaje.
5	Diurno	37	0	10	Todo muy intuitivo porque las acciones que el usuario puede realizar son limitadas.

Cuadro 5: Tarea 4 de pruebas de usabilidad - Iniciar un viaje y pausarlo a los diez segundos de viaje transcurrido

Usuario	**Modo de la aplicación**	**Tiempo transcurrido (seg)**	**Errores durante la tarea**	**Puntuación de satisfacción (1-10)**	**Comentarios generales**
1	Nocturno	37	1	8	El botón de finalizar debería ser más rojo.
2	Diurno	40	0	10	Todo muy intuitivo.
3	Nocturno	27	0	10	Sencillo iniciar y finalizar un viaje.
4	Diurno	35	0	9	Todo el flujo es sencillo, pero no entiendo para que sirve la previsualización de la cámara en la última pantalla.
5	Nocturno	29	0	10	Todo el flujo de la aplicación es muy intuitivo. Con la guía de la distancia es muy fácil saber cuanta distancia recorrí.

Cuadro 6: Tarea 5 de pruebas de usabilidad - Finalizar un viaje después de 10 segundos y evaluar la interfaz de finalización de viaje

La segunda prueba de usabilidad consistió en que los cinco usuarios debían interactuar con el simulador de copiloto durante 15 minutos. El objetivo de esta prueba fue evaluar la percepción y satisfacción de los usuarios con respecto al simulador de copiloto implementado. Se buscaba garantizar que los usuarios no percibieran el simulador de copiloto como algo intrusivo y distractivo. Por el contrario, se quería confirmar que esta funcionalidad se visualiza como una herramienta de soporte y que realmente pudiera llegar a ser capaz de reactivar las capacidades cognitivas del usuario. A través de los resultados obtenidos, se puede observar que el simulador de copiloto se percibe como una herramienta útil que ayuda a mantener alerta a los usuarios debido a que los monólogos que éste recita son de agrado para el mismo. En resumen, se ratifica que esta funcionalidad no solo es una herramienta que puede llegar a ser de utilidad para los pilotos, sino que también enriquece la experiencia global del usuario y causa agrado en los mismos.

Usuario	1
¿Cómo calificarías la utilidad del simulador de copiloto en una escala del 1 al 10?	8
¿Cómo calificarías, en una escala del 1 al 10, la variedad de temas y conversaciones iniciadas por el simulador de copiloto? (Donde 1 es muy poca variedad y 10 es una variedad excelente)	10
¿El simulador de copiloto te ayudó a mantenerte alerta y concentrado durante el viaje?	Sí
¿Sentiste que los monólogos proporcionadas por el simulador de copiloto se ajustaban bien a tus intereses?	Sí
¿El simulador de copiloto hizo el viaje más agradable?	Sí
¿Sentiste que el simulador de copiloto intervino en los momentos adecuados o fue demasiado intrusivo?	No fue nada intrusivo
¿Hubo algún momento en el que deseaste que el simulador de copiloto estuviera en silencio?	Sí
¿Te resultó fácil y cómodo seleccionar tus categorías preferidas antes de iniciar el viaje?	Sí
¿Cada cuánto tiempo te gustaría que el simulador de copiloto te hablara durante el viaje?	3 minutos
¿Tienes alguna sugerencia para mejorar el simulador de copiloto?	Ninguno

Cuadro 7: Respuestas del usuario no. 1 al cuestionario sobre el simulador de copiloto

Usuario	2
¿Cómo calificarías la utilidad del simulador de copiloto en una escala del 1 al 10?	10
¿Cómo calificarías, en una escala del 1 al 10, la variedad de temas y conversaciones iniciadas por el simulador de copiloto? (Donde 1 es muy poca variedad y 10 es una variedad excelente)	10
¿El simulador de copiloto te ayudó a mantenerte alerta y concentrado durante el viaje?	Sí
¿Sentiste que los monólogos proporcionadas por el simulador de copiloto se ajustaban bien a tus intereses?	Sí
¿El simulador de copiloto hizo el viaje más agradable?	Sí
¿Sentiste que el simulador de copiloto intervino en los momentos adecuados o fue demasiado intrusivo?	Sí, intervino en los momentos adecuados
¿Hubo algún momento en el que deseaste que el simulador de copiloto estuviera en silencio?	No
¿Te resultó fácil y cómodo seleccionar tus categorías preferidas antes de iniciar el viaje?	Sí
¿Cada cuánto tiempo te gustaría que el simulador de copiloto te hablara durante el viaje?	1:30 - 2:00 minutos
¿Tienes alguna sugerencia para mejorar el simulador de copiloto?	Tal vez poder modificar ese intervalo para que intervenga el copiloto

Cuadro 8: Respuestas del usuario no. 2 al cuestionario sobre el simulador de copiloto

Usuario	3
¿Cómo calificarías la utilidad del simulador de copiloto en una escala del 1 al 10?	10
¿Cómo calificarías, en una escala del 1 al 10, la variedad de temas y conversaciones iniciadas por el simulador de copiloto? (Donde 1 es muy poca variedad y 10 es una variedad excelente)	10
¿El simulador de copiloto te ayudó a mantenerte alerta y concentrado durante el viaje?	Sí
¿Sentiste que los monólogos proporcionadas por el simulador de copiloto se ajustaban bien a tus intereses?	Sí
¿El simulador de copiloto hizo el viaje más agradable?	Sí
¿Sentiste que el simulador de copiloto intervino en los momentos adecuados o fue demasiado intrusivo?	Sí
¿Hubo algún momento en el que deseaste que el simulador de copiloto estuviera en silencio?	No
¿Te resultó fácil y cómodo seleccionar tus categorías preferidas antes de iniciar el viaje?	Sí, aunque no sabía que habían más categorías al darle scroll down
¿Cada cuánto tiempo te gustaría que el simulador de copiloto te hablara durante el viaje?	3 minutos
¿Tienes alguna sugerencia para mejorar el simulador de copiloto?	Mejorar la barra de scroll en la selección de categorías

Cuadro 9: Respuestas del usuario no. 3 al cuestionario sobre el simulador de copiloto

Usuario	4
¿Cómo calificarías la utilidad del simulador de copiloto en una escala del 1 al 10?	9
¿Cómo calificarías, en una escala del 1 al 10, la variedad de temas y conversaciones iniciadas por el simulador de copiloto? (Donde 1 es muy poca variedad y 10 es una variedad excelente)	10
¿El simulador de copiloto te ayudó a mantenerte alerta y concentrado durante el viaje?	Sí
¿Sentiste que los monólogos proporcionadas por el simulador de copiloto se ajustaban bien a tus intereses?	Sí
¿El simulador de copiloto hizo el viaje más agradable?	Sí
¿Sentiste que el simulador de copiloto intervino en los momentos adecuados o fue demasiado intrusivo?	Sí intervino en momentos prudentes
¿Hubo algún momento en el que deseaste que el simulador de copiloto estuviera en silencio?	Sí
¿Te resultó fácil y cómodo seleccionar tus categorías preferidas antes de iniciar el viaje?	Sí
¿Cada cuánto tiempo te gustaría que el simulador de copiloto te hablara durante el viaje?	5 minutos
¿Tienes alguna sugerencia para mejorar el simulador de copiloto?	Agregar opción de modificar la voz que habla

Cuadro 10: Respuestas del usuario no. 4 al cuestionario sobre el simulador de copiloto

Usuario	5
¿Cómo calificarías la utilidad del simulador de copiloto en una escala del 1 al 10?	9
¿Cómo calificarías, en una escala del 1 al 10, la variedad de temas y conversaciones iniciadas por el simulador de copiloto? (Donde 1 es muy poca variedad y 10 es una variedad excelente)	10
¿El simulador de copiloto te ayudó a mantenerte alerta y concentrado durante el viaje?	Sí porque cada vez que hablaba me hacía concentrarme en lo que estaba escuchando, lo que activaba mi cerebro indirectamente
¿Sentiste que los monólogos proporcionadas por el simulador de copiloto se ajustaban bien a tus intereses?	Sí porque justamente habló sobre temas que yo le indiqué
¿El simulador de copiloto hizo el viaje más agradable?	Sí porque escuchar datos curiosos sobre temas que me interesan siempre es agradable
¿Sentiste que el simulador de copiloto intervino en los momentos adecuados o fue demasiado intrusivo?	Sí intervino oportunamente
¿Hubo algún momento en el que deseaste que el simulador de copiloto estuviera en silencio?	Sí, creo que habla muy seguido y a veces puede chocar con la música de mi carro
¿Te resultó fácil y cómodo seleccionar tus categorías preferidas antes de iniciar el viaje?	Sí
¿Cada cuánto tiempo te gustaría que el simulador de copiloto te hablara durante el viaje?	5 minutos
¿Tienes alguna sugerencia para mejorar el simulador de copiloto?	Ninguna

Cuadro 11: Respuestas del usuario no. 5 al cuestionario sobre el simulador de copiloto

A continuación se presentan capturas de pantalla que muestran todo el flujo de la aplicación, tanto en modo diurno como en modo nocturno. Estos resultados incluyen los cambios sugeridos por los usuarios en las pruebas de usabilidad que se consideraron oportunos para mejorar el producto final.

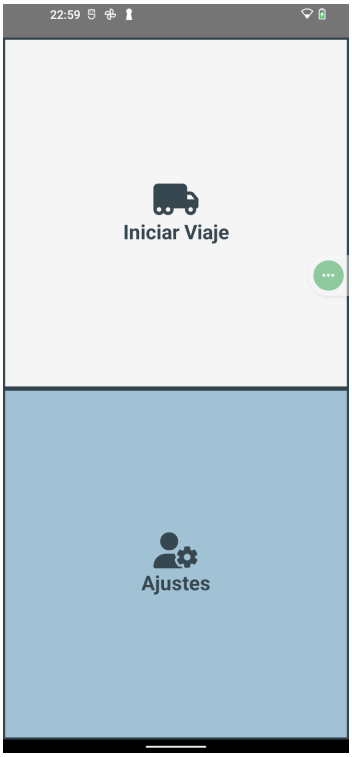


Figura 15: Pantalla de inicio - Modo diurno

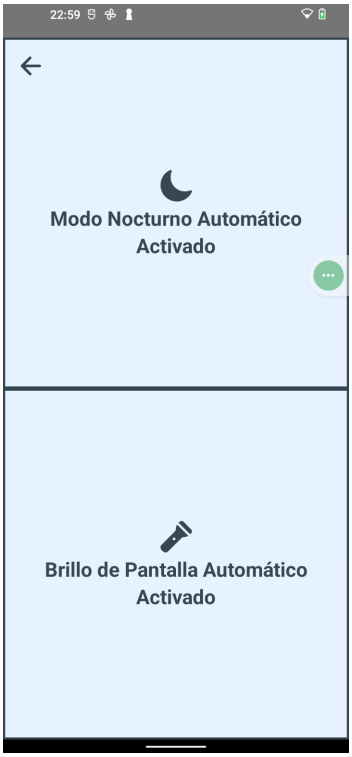


Figura 16: Pantalla de ajustes - Modo diurno

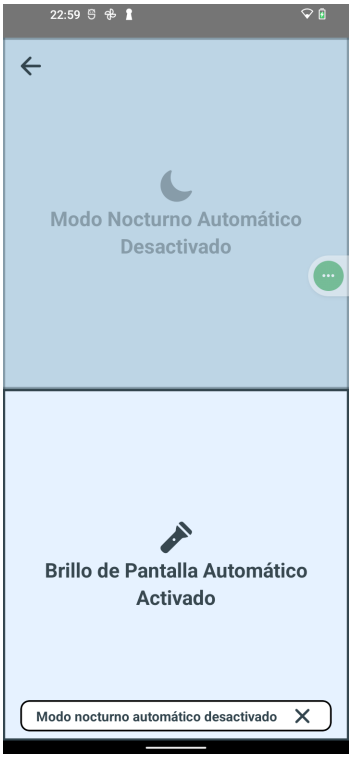


Figura 17: Pantalla de ajustes - Modo nocturno automático desactivado - Modo diurno

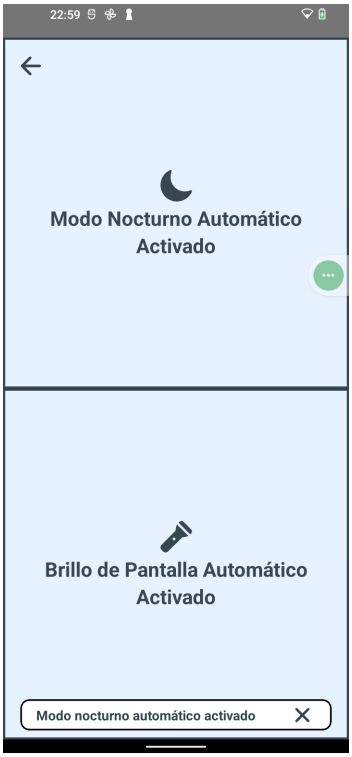


Figura 18: Pantalla de ajustes - Modo nocturno automático activado - Modo diurno

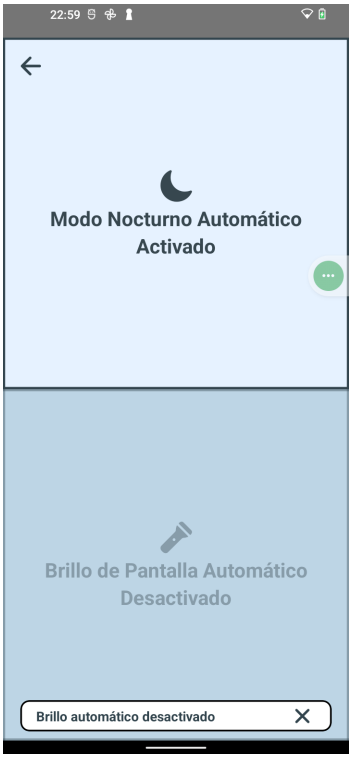


Figura 19: Pantalla de ajustes - Cambio de brillo automático desactivado - Modo diurno

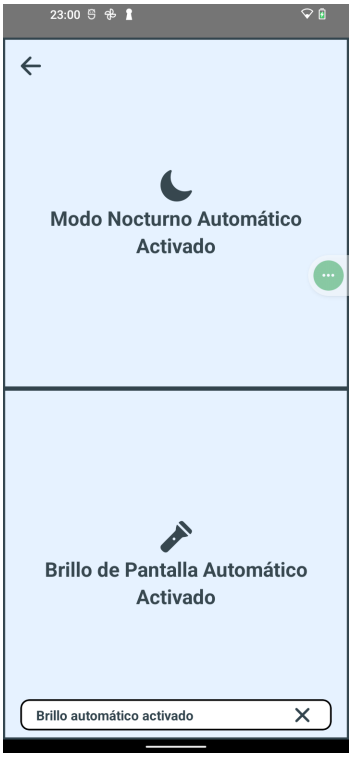


Figura 20: Pantalla de ajustes - Cambio de brillo automático activado - Modo diurno

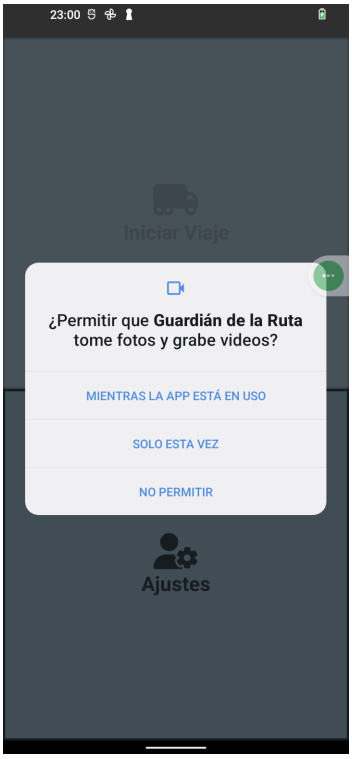


Figura 21: Pantalla de solicitud de permisos de cámara - Modo diurno

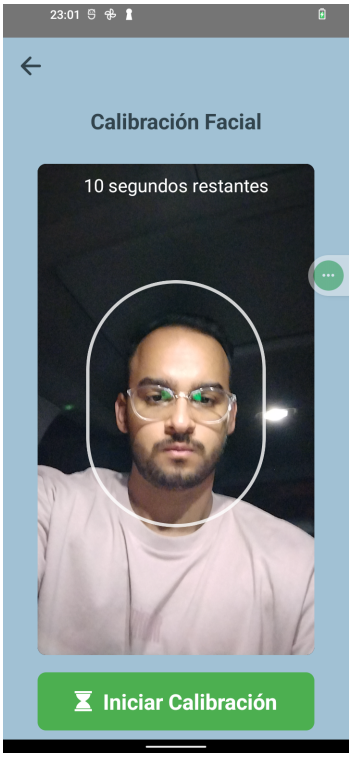


Figura 22: Pantalla de calibración facial - Modo diurno

Figura 23: Pantalla de calibración facial - Iniciada - Modo diurno

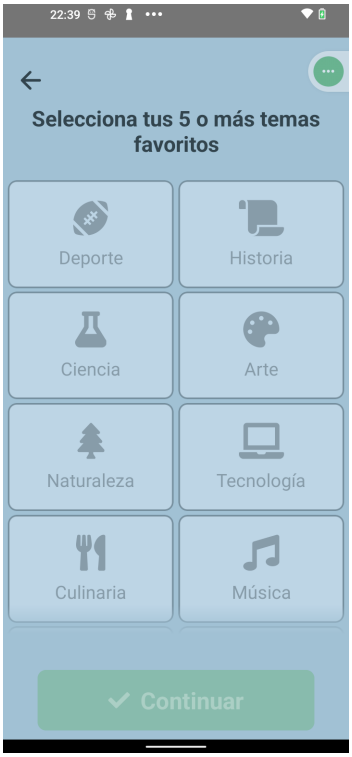


Figura 24: Pantalla de selección de categorías - Ninguna seleccionada - Modo diurno

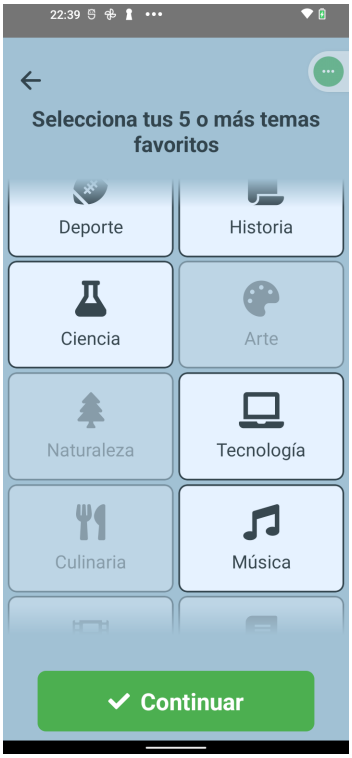


Figura 25: Pantalla de Selección de categorías - Algunas seleccionadas - Modo diurno

Figura 26: Pantalla de solicitud de permisos de cámara - Modo diurno

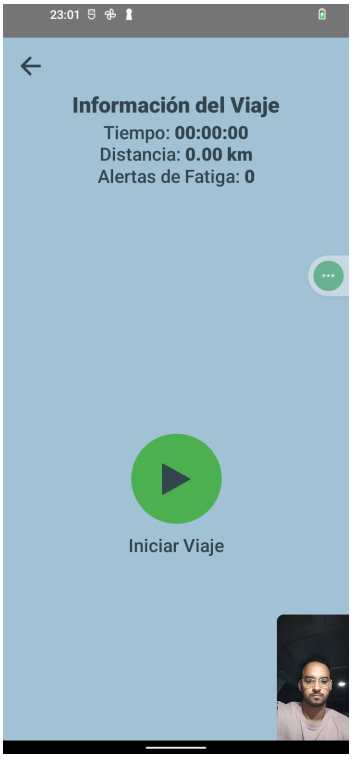


Figura 27: Pantalla de viaje - Viaje no iniciado - Modo diurno

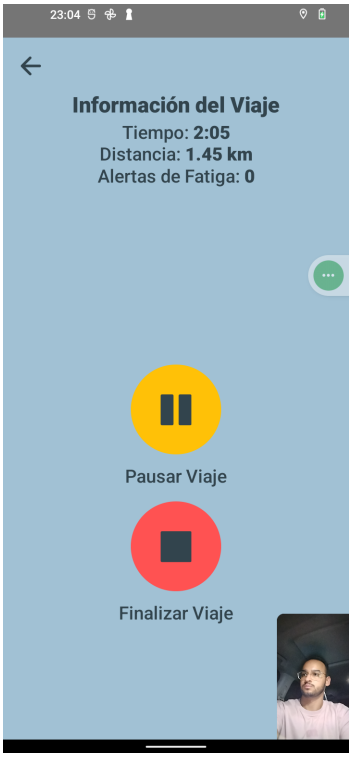


Figura 28: Pantalla de viaje - Viaje en curso - Modo diurno

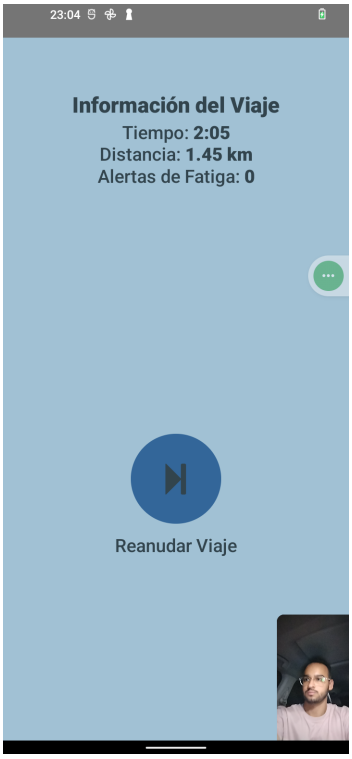


Figura 29: Pantalla de viaje - Viaje pausado - Modo diurno

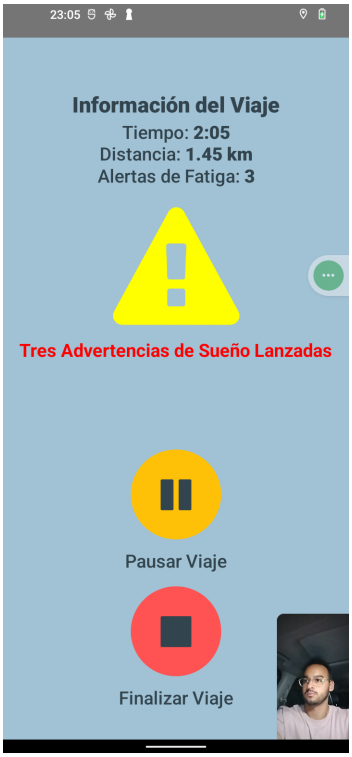


Figura 30: Pantalla de viaje - Viaje en curso - Tres advertencias de sueño lanzadas - Modo diurno

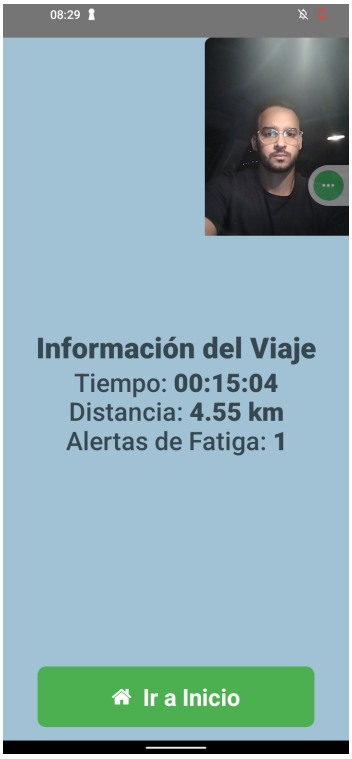


Figura 31: Pantalla de viaje finalizado - Modo diurno

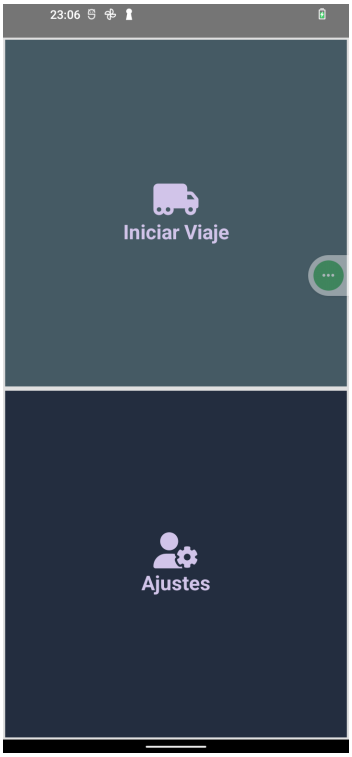


Figura 32: Pantalla de inicio - Modo nocturno

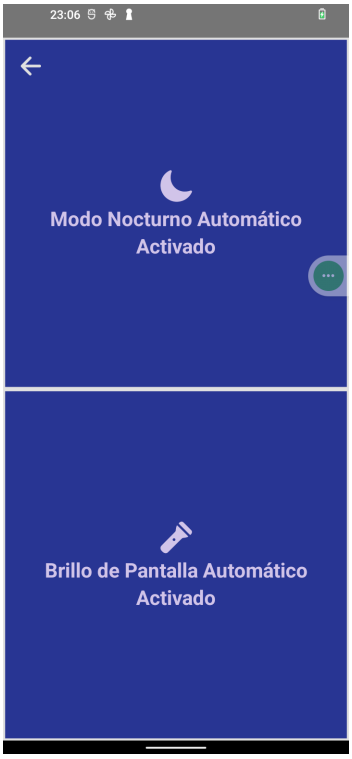


Figura 33: Pantalla de ajustes - Modo nocturno

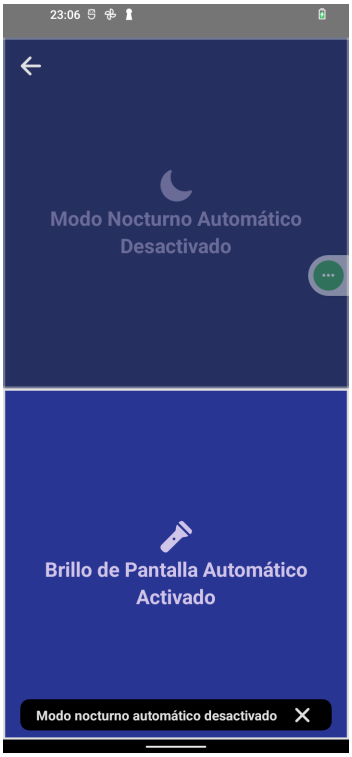


Figura 34: Pantalla de ajustes - Modo nocturno automático desactivado - Modo nocturno

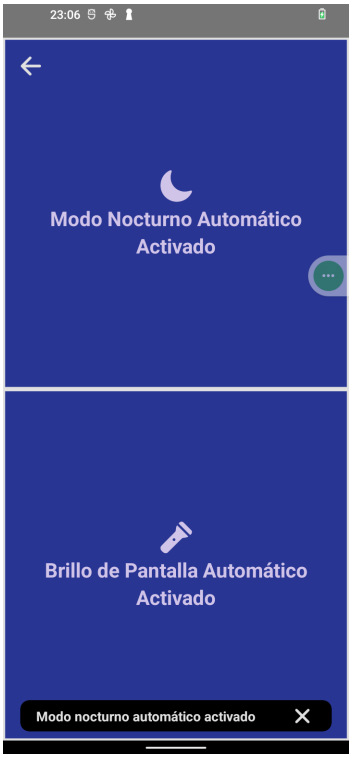


Figura 35: Pantalla de ajustes - Modo nocturno automático activado - Modo nocturno

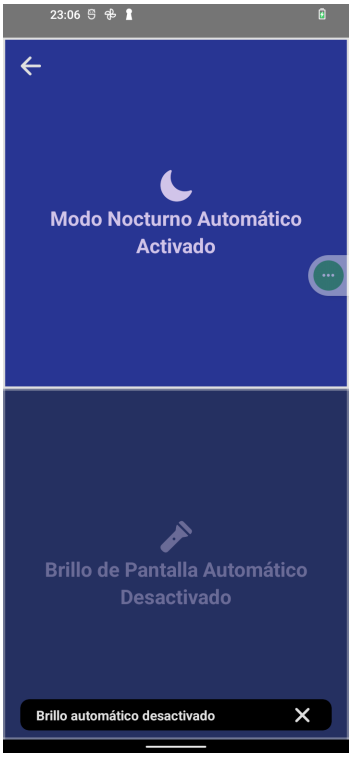


Figura 36: Pantalla de ajustes - Cambio de brillo automático desactivado - Modo nocturno

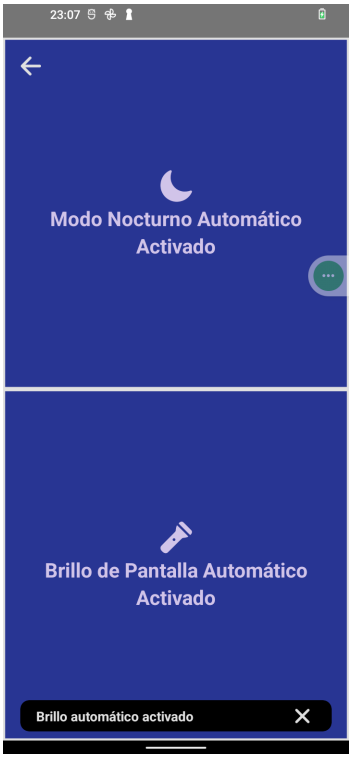


Figura 37: Pantalla de ajustes - Cambio de brillo automático activado - Modo nocturno

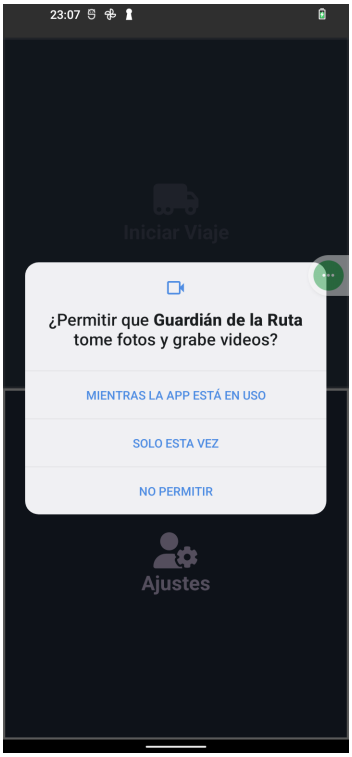


Figura 38: Pantalla de solicitud de permisos de cámara - Modo nocturno

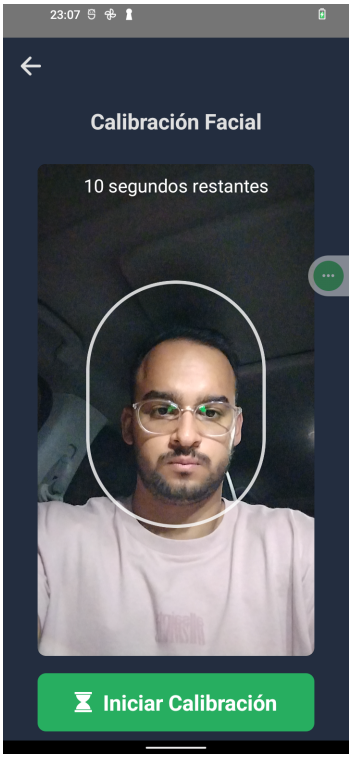


Figura 39: Pantalla de calibración facial - Modo nocturno

Figura 40: Pantalla de calibración facial - Iniciada - Modo nocturno

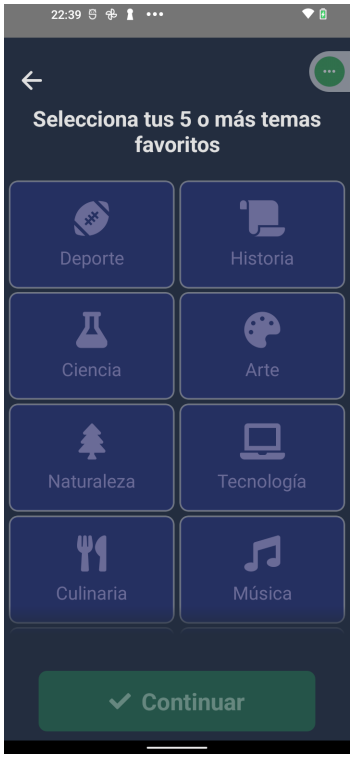


Figura 41: Pantalla de selección de categorías - Ninguna seleccionada - Modo nocturno

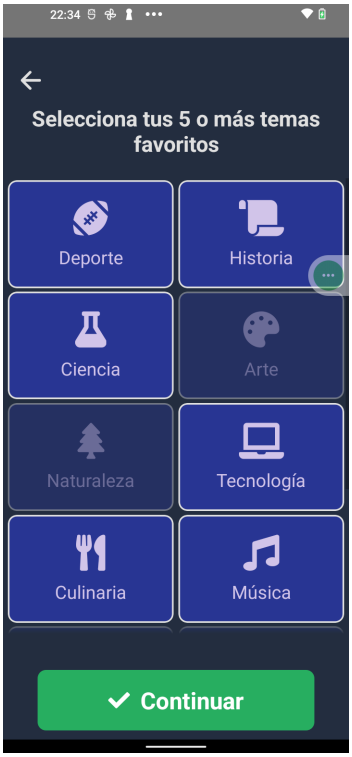


Figura 42: Pantalla de selección de categorías - Algunas seleccionadas - Modo nocturno

Figura 43: Pantalla de solicitud de permisos de cámara - Modo nocturno

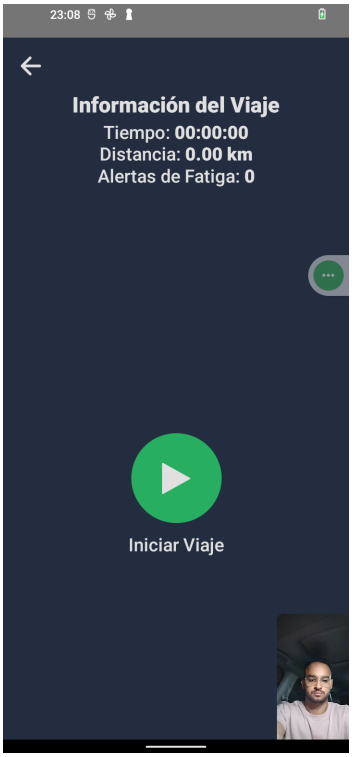


Figura 44: Pantalla de viaje - Viaje no iniciado - Modo nocturno

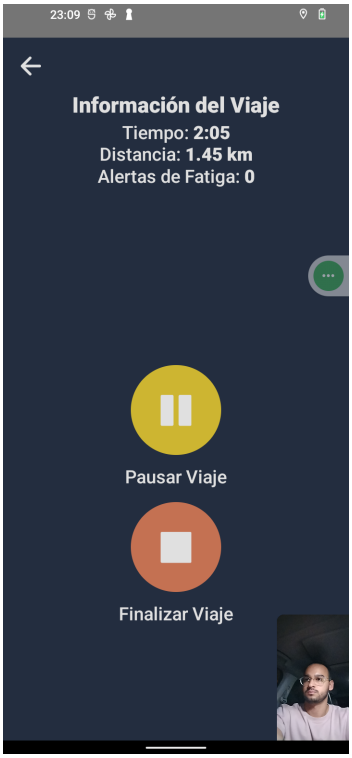


Figura 45: Pantalla de viaje - Viaje en curso - Modo nocturno

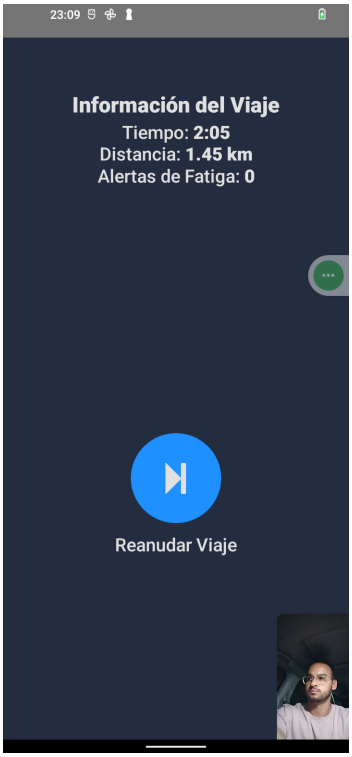


Figura 46: Pantalla de viaje - Viaje pausado - Modo nocturno

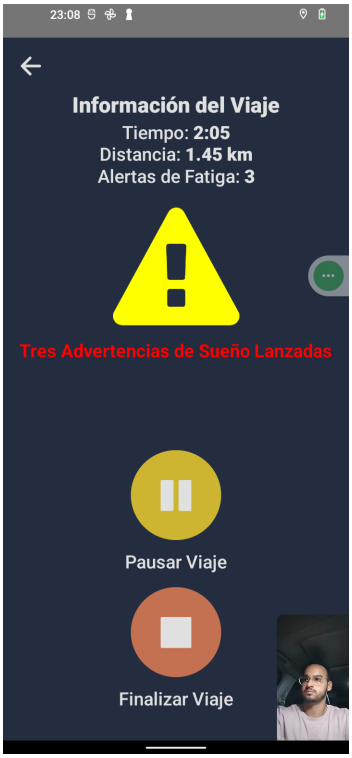


Figura 47: Pantalla de viaje - Viaje en curso - Tres advertencias de sueño lanzadas - Modo nocturno

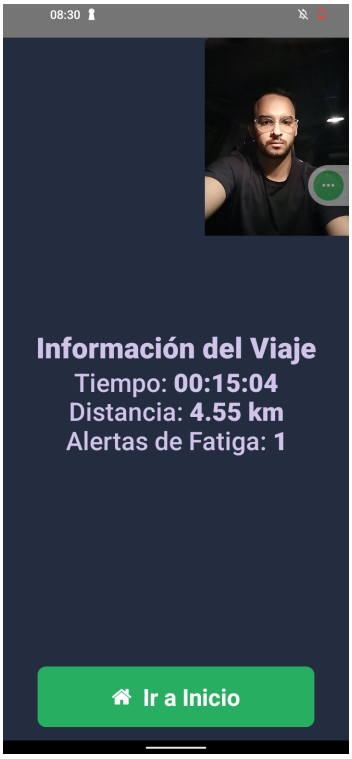


Figura 48: Pantalla de viaje finalizado - Modo nocturno

CAPÍTULO 7

Conclusiones

Se ha logrado diseñar una interfaz móvil orientada específicamente a pilotos, que no sólo es intuitiva y funcional, sino que también se centra en la recolección eficaz de datos para ser utilizados como métodos de detección de fatiga y alertarlos con base en eso.

Con la aplicación de técnicas y conceptos de ergonomía cognitiva e interacción humano-computador, se ha logrado conseguir una experiencia de usuario ideal. Se ha puesto el debido enfoque en el diseño de la interfaz de usuario, asegurándose de que se ajuste a las necesidades que los conductores de transporte pesado presentan dada su situación. El producto final ofrece una interfaz limpia y agradable en la utilización y visualización, permitiendo una navegación más fluida y eficiente.

El objetivo de que la aplicación fuera capaz de tomar tanto muestras faciales como de ritmo cardíaco para poder ser utilizados para detectar somnolencia se alcanzó parcialmente. La capacidad de la aplicación de tomar muestras faciales del piloto se logró con éxito. Sin embargo, el segundo punto se optó por ser desarrollado de otra manera. A pesar de ello, se asegura una medición precisa y eficaz de ambos parámetros.

La funcionalidad del simulador de copiloto interactivo ha agregado un valor importante para el producto. Más allá de buscar mantener alerta al conductor, esta herramienta logra reducir la sensación de soledad y aislamiento dentro de la cabina del vehículo que puede surgir durante largos periodos de viaje. Se alcanzó el objetivo de que el piloto se sienta acompañado, lo que conlleva a mayor concentración y un entretenimiento eficiente y cero distractivo. Adicionalmente, se logró crear un ambiente amigable que facilita que el piloto se sienta en control y activado mentalmente.

CAPÍTULO 8

Recomendaciones

React Native, el *framework* utilizado para desarrollar Guardián de la Ruta, proporciona dos entornos para el desarrollo de aplicaciones: React Native CLI y Expo. A pesar de que la aplicación fue desarrollada exitosamente utilizando React Native CLI, optar por Expo pudo haber resultado en una experiencia de desarrollo un poco más fluida y eficiente. Expo se caracteriza principalmente por ofrecer un entorno completamente preconfigurado y listo para utilizar.

Asimismo, Expo facilita enormemente la integración de diversidad de librerías y funcionalidades, como la de la cámara, el texto a voz y la librería de ubicación. Estas son librerías que son controladas por los propios creadores de Expo, ya que tienen un catálogo de ellas que proveen todas estas funcionalidades nativas en conjunto. Esto quiere decir que es menos probable que alguna de las librerías presente fallas con el proyecto, ya que todo es manejado por un solo ecosistema, como lo es Expo. Este enfoque reduce la posibilidad de enfrentar problemas de compatibilidad o configuración incorrecta, como sucedió con la librería de texto a voz del lado de React Native CLI y que tocó modificar manualmente para alcanzar los resultados deseados.

Finalmente, otro punto a destacar de Expo es que este entorno facilita el proceso de actualización al permitir actualizaciones por vía aérea, lo que significa que los usuarios pueden recibir las nuevas versiones que se vayan lanzando de la aplicación sin tener que hacerlo manualmente desde la tienda. Esto puede ser ventajoso en el momento en el que se desee implementar nuevas funcionalidades, cambiar temas de experiencia de usuario o mejorar cualquier aspecto de la aplicación en general.

Por lo tanto, aunque ambas rutas tienen sus pros, elegir Expo para este proyecto, por los puntos mencionados previamente, pudo haber resultado en una experiencia de desarrollo un poco más amigable, un tiempo de implementación más rápido y una integración más sencilla de las librerías utilizadas para la aplicación.

Es importante prestar atención a los más mínimos detalles del producto ya que estos pueden agregar valor en la percepción del usuario y en la usabilidad general de la aplicación.

Por eso, se recomienda diseñar cuidadosamente un ícono que represente de manera clara y atractiva el propósito de la aplicación. Se recomienda integrar un ícono minimalista y memorable, para mantener la sencillez y para que los usuarios puedan identificar fácilmente la aplicación en la pantalla de inicio de sus dispositivos. Se exhorta a trabajar en un ícono con un diseño innovador y simple.

Asimismo, otro detalle que puede tener un impacto positivo en la aplicación es diseñar meticulosamente una tipografía que sea minimalista y fácil de leer. Esto debido a que el tipo de fuente juega un papel crucial en la experiencia del usuario, e integrar una fuente desarrollada específicamente para esta aplicación no solo agregará identidad a la misma, si no que también se traduce en una mayor tasa de aceptación y satisfacción del usuario. La integración de estos dos elementos puede ser clave para establecer una presencia fuerte y reconocible en el mercado.

El último punto a recomendar es mantener actualizadas todas las librerías posibles de la aplicación. Esto puede contribuir a varios aspectos claves, como lo son la seguridad y la estabilidad de las librerías, y también a contribuir a mejorar el rendimiento general del producto. Las actualizaciones constantes pueden contribuir a correcciones de errores, cerradas inesperadas de la aplicación y también en la eficiencia en la que procesan la información. Asimismo, utilizar versiones antiguas de las librerías pueden ocasionar problemas de incompatibilidad con nuevas versiones de sistema operativo o con otras librerías de la aplicación. En resumen, mantener las librerías que la aplicación utiliza actualizadas es una buena práctica que ayuda a la estabilidad, seguridad y rendimiento de la aplicación.

CAPÍTULO 9

Bibliografía

[1] D. L. Strayer y F. A. Drews, "Cell-phone induced inattention blindness," *Current Directions in Psychological Science*, vol. 16, págs. 128-131, 2007.

[2] D. L. Strayer, F. A. Drews y W. A. Johnston, "Cell phone induced failures of visual attention during simulated driving," *Journal of Experimental Psychology: Applied*, vol. 9, págs. 23-52, 2003.

[3] D. L. Strayer y W. A. Johnston, "Driven to distraction: Dual-task studies of simulated driving and conversing on a cellular phone," *Psychological Science*, vol. 12, págs. 462-466, 2001.

[4] W. Danielsson, "React Native application development," *Linköpings universitet, Swedia*, vol. 10, n.º 4, pág. 10, 2016.

[5] C. Mullins, "Responsive, mobile app, mobile first: untangling the UX design web in practical experience," en *Proceedings of the 33rd Annual International Conference on the Design of Communication*, 2015, págs. 1-6.

[6] D. Saffer, *Designing for interaction: Creating innovative applications and devices*. New Riders, 2005.

[7] N. Hansson y T. Vidhall, "Effects on performance and usability for cross-platform application development using React Native," 2016.

[8] M. A. Assari y M. Rahmati, "Driver drowsiness detection using face expression recognition," en *2011 IEEE international conference on signal and image processing applications (ICSIPA)*, IEEE, 2011, págs. 337-341.

[9] M. Naveh-Benjamin e Y. Naveh-Benjamin, "The impact of interest on attention and memory during conversations," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 34, n.º 6, págs. 1469-1486, 2008. DOI: 10.1037/a0012950.

[10] J. F. May y C. L. Baldwin, "Driver fatigue: The importance of identifying causal factors of fatigue when considering detection and countermeasure technologies," *Transportation research part F: traffic psychology and behaviour*, vol. 12, n.º 3, págs. 218-224, 2009.

[11] D. A. Prasetya, P. T. Nguyen, R. Faizullin, I. Iswanto y E. F. Armay, "Resolving the shortest path problem using the haversine algorithm," *Journal of Critical Reviews*, vol. 7, n.º 1, págs. 62-64, 2020.

[12] A. Rodríguez-Martínez, J. Sánchez-Miranda y J. Fernández-Rodríguez, "Desarrollo de un sistema de detección de somnolencia para conductores basado en Python y Flask," *Revista de Ingeniería y Ciencia*, vol. 28, n.º 2, págs. 155-166, 2022. DOI: 10.22201/fca.28078956e.2022.2.129.

[13] A. González-Morales, I. Serrano-García y A. Fernández-Liria, "Efecto del esfuerzo cognitivo en la somnolencia," *Revista de Neurología*, vol. 57, n.º 5, págs. 327-333, 2013. DOI: 10.33588/rn.3072.

[14] M. Martínez-Martínez y A. Fernández-Liria, "La compañía como factor de seguridad en la conducción," *Revista de Neurología*, vol. 64, n.º 1, págs. 25-31, 2017. DOI: 10.33588/rn.3224.