
Rediseño, desarrollo e implementación de UI/UX para una aplicación de recorridos virtuales con Unity en el Centro de Innovación y Tecnología (CIT) de la Universidad del Valle de Guatemala.

María Marta Ramírez Gil



Rediseño, desarrollo e implementación de UI/UX para una aplicación de recorridos virtuales con Unity en el Centro de Innovación y Tecnología (CIT) de la Universidad del Valle de Guatemala.

María Marta Ramírez Gil



UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Rediseño, desarrollo e implementación de UI/UX
para una aplicación de recorridos virtuales con Unity
en el Centro de Innovación y Tecnología (CIT) de la
Universidad del Valle de Guatemala.**

Trabajo de graduación en modalidad de Trabajo Profesional presentado
por María Marta Ramírez Gil para optar al grado académico de
Licenciada en Ingeniería en Ciencias de la Computación y Tecnologías
de la Informaición

Guatemala
2025

Vo.Bo.

Firma: 
Inga Dulce María Chacón Muñoz

Firma: 
Ing. Carlos Alonso

Fecha de aprobación: Guatemala, 18 de noviembre 2025.

La evolución constante de la tecnología ha transformado la manera en que las instituciones educativas interactúan con sus espacios físicos y su comunidad. En este contexto, el presente trabajo de graduación se desarrolla como parte de un megaproyecto institucional impulsado por la Universidad del Valle de Guatemala, cuyo objetivo es modernizar los sistemas de orientación y navegación dentro del campus. Como estudiantes de Ciencias de la Computación y Tecnologías de la Información, buscamos aplicar nuestros conocimientos en el desarrollo de soluciones tecnológicas que aporten valor a la comunidad universitaria.

Este proyecto continúa los esfuerzos de generaciones anteriores enfocados en recorridos virtuales, incorporando tecnologías de localización en interiores, herramientas de realidad virtual y mejoras en la interfaz de usuario. Con ello, se pretende validar la viabilidad de estas soluciones y mejorar la experiencia de navegación de los usuarios dentro del campus.

En este contexto, expreso mi más sincero agradecimiento a todas las personas e instituciones que hicieron posible la realización de este proyecto.

Agradezco a la **Universidad del Valle de Guatemala** por el respaldo institucional y los recursos brindados para el desarrollo de este trabajo. Asimismo, extendo mi gratitud a mi **asesor académico** y a la **Dulce María Chacón Muñoz** por su guía, apoyo y valiosos aportes a lo largo del proceso.

Agradezco a **Andrea Rebecca Leiva Pérez** por su contribución en el diseño visual de la aplicación, así como a los **compañeros de generaciones anteriores** por sentar las bases de este megaproyecto.

A mis **compañeros de proyecto**, **Gustavo Andrés González Pineda**, **Diego Alberto Leiva** y **Eduardo Ramírez**, les agradezco por su colaboración y compromiso durante el desarrollo del sistema.

Finalmente, agradezco profundamente a **mi familia**, **amistades** y **mascotas** por su apoyo, motivación y compañía a lo largo de este proceso.

A todos, mi sincero agradecimiento.

Prefacio	III
Índice de figuras	VIII
Índice de cuadros	IX
Resumen	X
1. Introducción	1
2. Antecedentes	3
2.1. Diseño de interfaz y experiencia de usuario (UI/UX) y desarrollo en Unity	3
2.2. Uso de minijuegos y gamificación	4
2.3. Módulo nativo de iOS (Xcode) y su desconexión inicial de Unity	4
3. Justificación	5
3.1. Beneficios del proyecto	5
3.2. Aplicación del conocimiento en ingeniería en ciencias de la computación .	6
3.3. Importancia del desarrollo del proyecto	6
4. Objetivos	7
4.1. Objetivo general	7
4.2. Objetivos específicos	7
5. Marco teórico	8
5.1. Conceptos clave	8
5.1.1. Realidad Aumentada (AR)	8
5.1.2. Interfaz de Usuario (UI)	8
5.1.3. Experiencia de Usuario (UX)	9
5.1.4. Navegación interior (indoor navigation)	9
5.2. Herramientas y tecnologías empleadas	9
5.2.1. Unity 3D	9
5.2.2. Unity UI Toolkit	10
5.2.3. UI Builder	10
5.2.4. Blender	10
5.2.5. Figma	11
5.2.6. Microsoft Forms	11
5.2.7. ARKit	11

5.2.8.	ARCore	12
5.2.9.	C# en Unity	12
5.2.10.	UXML	13
5.2.11.	USS	13
6.	Metodología	15
6.1.	Enfoque de desarrollo iterativo y centrado en Usuario	15
6.1.1.	Justificación del enfoque iterativo	15
6.1.2.	Ciclos de iteración planificados	16
6.1.3.	Estrategia de evaluación con usuarios	16
6.2.	Diseño de la arquitectura de la aplicación	16
6.2.1.	Selección de tecnología UI: UI Toolkit vs uGUI	17
6.3.	Desarrollo de la aplicación	17
6.3.1.	Arquitectura de layouts jerárquicos	18
6.3.2.	Sistema de navegación modular	18
6.3.3.	Estrategia de pruebas técnicas	19
6.4.	Integración con módulo de minijuegos	19
6.4.1.	Estrategia de migración entre versiones de Unity	19
6.4.2.	Proceso de recreación y adaptación	20
6.4.3.	Decisión de separación funcional	20
6.4.4.	Estado de integración y gestión de Branches	21
6.5.	Integración de multimedia	21
6.5.1.	Proceso de adquisición y preparación de assets	21
6.6.	Debug y pruebas internas	22
6.6.1.	Metodología de debugging	22
6.6.2.	Refactorización arquitectónica	22
6.7.	Estrategia de control de versiones	23
6.7.1.	Estructura de branching	23
6.7.2.	Protocolo de integración	23
6.8.	Documentación técnica	24
6.8.1.	Metodología de documentación	24
6.8.2.	Contenido documentado	24
6.9.	Cronograma de ejecución	25
6.9.1.	Fase 1: Planificación y diseño (enero-febrero 2025)	25
6.9.2.	Fase 2: Desarrollo inicial (marzo-mayo 2025)	25
6.9.3.	Fase 3: Primera evaluación y refinamiento (junio-julio 2025)	25
6.9.4.	Fase 4: Rediseño e implementación de mejoras (agosto-septiembre 2025)	25
6.9.5.	Fase 5: Documentación y entrega final (octubre 2025)	26
6.10.	Recursos y herramientas utilizadas	26
6.10.1.	Entorno de desarrollo	26
6.10.2.	Herramientas de diseño	26
6.10.3.	Dispositivos de prueba	26
6.10.4.	Servicios y plataformas	27
6.10.5.	Frameworks y librerías	27
6.11.	Limitaciones metodológicas	27
6.11.1.	Limitaciones de evaluación con usuarios	27
6.11.2.	Limitaciones técnicas	27
6.11.3.	Limitaciones de recursos	28
6.11.4.	Mitigación de limitaciones	28

7. Resultados y discusión	29
7.1. Desarrollo del sistema de interfaz responsivo	29
7.1.1. Configuración base del sistema	29
7.1.2. Arquitectura de layouts	30
7.1.3. Sistema de estilos responsivos	30
7.1.4. Adaptación de pantallas	31
7.1.5. Correcciones de inconsistencias visuales	31
7.2. Evaluación con usuarios - primera iteración	34
7.2.1. Perfil de participantes	34
7.2.2. Resultados de usabilidad	34
7.2.3. Claridad de información	34
7.2.4. Experiencia inmersiva	35
7.2.5. Utilidad percibida	35
7.2.6. Aspectos valorados	35
7.2.7. Áreas de mejora identificadas	35
7.2.8. Calificación del diseño visual	36
7.2.9. Orientación y navegación	36
7.2.10. Recomendación y satisfacción	36
7.3. Rediseño integral de la interfaz	37
7.3.1. Elementos visuales implementados	37
7.3.2. Mejoras de experiencia de usuario	38
7.3.3. Evolución del diseño visual	39
7.4. Sistema de puntos de interés (POI)	41
7.4.1. Funcionalidad del modal POI	41
7.4.2. Estructura del modal	41
7.4.3. Integración con el footer normal	42
7.5. Sistema de onboarding	44
7.5.1. Estructura del onboarding	44
7.5.2. Pantallas implementadas	44
7.5.3. Persistencia de estado	45
7.6. Menú de navegación lateral	45
7.6.1. Evolución de la implementación	45
7.6.2. Estructura del menú	46
7.6.3. Evolución de opciones implementadas	46
7.6.4. Mejoras técnicas de la transición	51
7.7. Sistema de navegación con flecha 3D	51
7.7.1. Alcance de documentación	51
7.8. Sistema de gestión de conexión con sensores	51
7.8.1. Popup de espera de conexión	51
7.8.2. Estados del header	53
7.9. Optimización de arquitectura de pantallas	54
7.9.1. Consolidación de funcionalidades	54
7.9.2. Integración del popup	55
7.9.3. Impacto en rendimiento	55
7.10. Documentación técnica del sistema	55
7.10.1. Estructura de la documentación	55
7.10.2. API pública para integración UWB	57
7.11. Sistema de minijuegos	58
7.11.1. Proceso de migración desde versión anterior	58
7.11.2. Estructura de la pantalla	59
7.11.3. Decisión de separación del tour	60
7.11.4. Estado actual de implementación	60
7.11.5. Trabajo pendiente de integración	61
7.12. Gestión de versionamiento y control de cambios	62

7.12.1. Estrategia de branching	62
7.12.2. Categorización de commits	62
7.12.3. Mensajes de commit	63
7.13. Resumen de logros técnicos	63
8. Conclusiones	64
9. Recomendaciones	65
10. Bibliografía	67
11. Glosario	69

Índice de figuras

7.1. Configuración de resolución de referencia en panelsettings	30
7.2. Interfaz antes de las correcciones de responsividad	32
7.3. Sistema de interfaz responsivo corregido implementando layout responsivo	33
7.4. Pantalla inicial rediseñada según especificaciones de diseño	38
7.5. Evolución del diseño de pantalla inicial a través de las iteraciones	40
7.6. modal de poi en estado expandido	42
7.7. Transición entre footer normal y footer poi	43
7.8. Secuencia completa de pantallas de onboarding	45
7.9. Menú hamburguesa lateral - primera iteración con implementación programática . .	48
7.10. Menú hamburguesa lateral - segunda iteración con implementación UXML y rediseño visual	50
7.11. Popup de espera de conexión con sensores antes del rediseño.	52
7.12. Popup de espera de conexión con sensores después del rediseño.	53
7.13. Comparación de estados del header: esperando conexión vs conectado	54
7.14. Ejemplo de documentación inline en el código del controlador	57
7.15. Pantallas de minijuegos implementada en branch de desarrollo con UI anterior (versión funcional pre-rediseño)	61

Índice de cuadros

7.1. Resumen de métricas de evaluación - primera iteración	37
7.2. Comparación evolutiva de versiones de pantalla inicial	39
7.3. Comparación de implementaciones del menú lateral	51
7.4. Impacto de la optimización arquitectónica	55
7.5. Comparación del proceso de migración de minijuegos	59
7.6. Resumen de commits por categoría funcional	62

El presente trabajo correspondió al desarrollo del módulo de interfaz gráfica de usuario (UI/UX) dentro del proyecto de graduación titulado “Desarrollo y rediseño de una aplicación de recorridos virtuales con sensores Estimote Beacons y tecnología UWB en el Centro de Innovación y Tecnología de la Universidad del Valle de Guatemala (UVG)”. Dicho módulo tuvo como objetivo diseñar una experiencia visual moderna, fluida y funcional, completamente optimizada para dispositivos móviles iOS, integrando tecnologías de realidad aumentada y principios de diseño responsivo.

La aplicación fue desarrollada desde cero, sin reutilizar código ni estructuras del proyecto del año anterior. Esta decisión permitió implementar una arquitectura modular, una jerarquía clara de archivos y una documentación técnica integrada en el código fuente. Se utilizó el motor de desarrollo Unity, junto con UI Toolkit y UI Builder, para construir interfaces declarativas mediante archivos UXML y hojas de estilo USS, lo cual garantizó una separación adecuada entre la lógica visual y el comportamiento funcional de la aplicación.

El diseño de la interfaz se realizó bajo un enfoque centrado en el usuario, siguiendo buenas prácticas de usabilidad y accesibilidad. Se elaboraron wireframes iniciales, prototipos visuales de alta fidelidad y se llevó a cabo una evaluación de una versión beta. Esta versión preliminar fue presentada a usuarios reales, quienes brindaron retroalimentación sobre su experiencia. Dicha retroalimentación fue analizada y permitió implementar mejoras que optimizaron la satisfacción y la interacción del usuario con la aplicación.

Se desarrollaron diversas pantallas, incluyendo el módulo de escaneo y de conexión con sensores, menús de navegación interactivos, integración de minijuegos y, especialmente, la "PantallaARTour", la cual mostró una vista en vivo desde la cámara del dispositivo. Sobre esta vista se superpusieron elementos interactivos como navegación, ubicación actual, instrucciones dinámicas, puntos de interés y una flecha interactiva siendo esta el elemento de realidad virtual. Para lograr esta funcionalidad, se integraron las tecnologías ARKit (para iOS), lo que permitió una alineación precisa entre los elementos digitales y el entorno físico real.

La navegación entre pantallas fue gestionada por un script central encargado de administrar múltiples UIDocuments dentro de una única escena base, lo cual permitió mantener un rendimiento óptimo y una estructura escalable.

Como resultado, se obtuvo una interfaz profesional, consistente, bien documentada y mantenible, que mejoró significativamente la experiencia de los usuarios durante los recorridos virtuales. Además, este módulo sentó las bases para futuras expansiones multiplataforma, aportando valor tanto funcional como visual al proyecto general.

CAPÍTULO 1

Introducción

El presente trabajo formó parte del proyecto de graduación desarrollado por estudiantes de último año de la carrera de Ciencias de la Computación y Tecnologías de la Información de la Universidad del Valle de Guatemala (UVG). Este proyecto tuvo como objetivo principal el desarrollo y rediseño de una aplicación móvil para iOS para recorridos virtuales en el Centro de Innovación y Tecnología (CIT) de la UVG, integrando tecnologías de realidad aumentada (AR) y localización mediante sensores Estimote Beacons y tecnología UWB (Ultra-Wideband).

La aplicación buscó enriquecer la experiencia de los visitantes del CIT, facilitando la orientación, exploración y aprendizaje interactivo mediante un sistema de navegación asistida y contenido digital informativo sobre puntos de interés dentro de las instalaciones. Para lograrlo, el sistema detectó la ubicación del usuario en tiempo real y desplegó información visual superpuesta a través de una interfaz de realidad aumentada, la cual fue desarrollada con el motor Unity y resultó compatible con ARKit (para dispositivos iOS). Además, el sistema incluyó un componente lúdico que integró minijuegos interactivos.

El proyecto se estructuró en distintos módulos técnicos, cada uno de ellos abordado por diferentes miembros del equipo: conexión con sensores y localización, pathfinding interno, integración de minijuegos y diseño e implementación de la interfaz gráfica de usuario (UI/UX). Este documento se centró exclusivamente en este último módulo, el cual fue desarrollado completamente desde cero, sin reutilizar componentes del proyecto anterior. Esta decisión como se mencionó con anterioridad permitió mejorar la arquitectura interna del sistema, la legibilidad del código y su escalabilidad. Se priorizó el uso de tecnologías modernas como UI Toolkit y UI Builder, lo cual facilitó la separación entre la lógica visual y funcional, la organización jerárquica del proyecto y la implementación de un diseño responsivo y profesional.

La interfaz incluyó pantallas como la de escaneo de sensores, navegación entre rutas, visualización en AR, menú de minijuegos y secciones informativas. Cada una de ellas fue diseñada siguiendo principios de accesibilidad, usabilidad y coherencia visual, adaptándose a las necesidades específicas de los usuarios del CIT y cumpliendo con los estándares actuales de experiencia móvil.

La utilidad principal del proyecto radicó en la mejora de la interacción entre los visitantes y el entorno físico del CIT, promoviendo la innovación educativa y tecnológica dentro de la universidad. Adicionalmente, se estableció una base sólida para futuras implementaciones en otros espacios académicos o institucionales que desearan replicar esta experiencia inmersiva.

Este documento se estructuró de la siguiente manera: la primera parte presenta un resumen general del proyecto; el capítulo 1 (esta introducción) proporcionó el contexto general; el capítulo 2 detalla los antecedentes del módulo; el capítulo 3 explica la justificación del desarrollo; el capítulo 4 los objetivos del proyecto. El capítulo 5 abordó el marco teórico; el capítulo 6 presenta los resultados obtenidos y las validaciones realizadas; el capítulo 7 expone la discusión de los resultados; y finalmente, los capítulos 8 y 9 incluyen las conclusiones y recomendaciones para futuras iteraciones. A lo largo del documento se incorporaron figuras, tablas y diagramas que complementaron la explicación técnica del desarrollo de la interfaz gráfica, así como secciones dedicadas a la documentación del código y la estructura del proyecto.

El presente proyecto de graduación se construyó sobre las bases y módulos previamente desarrollados por otro grupos de estudiantes en el marco del megaproyecto tecnológico de la Universidad del Valle de Guatemala (UVG). Estos antecedentes abarcaron el diseño de interfaces centradas en el usuario, la integración de elementos interactivos como minijuegos y la validación tecnológica de geolocalización interna para el sistema operativo iOS.

2.1. Diseño de interfaz y experiencia de usuario (UI/UX) y desarrollo en Unity

Uno de los proyectos precursores tuvo como enfoque central el desarrollo de una interfaz gráfica (UI) que asegurara una experiencia de usuario (UX) óptima para cualquier tipo de dispositivo móvil en una aplicación de realidad aumentada (AR). Se reconoció que la UX, definida como la percepción del usuario sobre la facilidad de uso, eficiencia y satisfacción al interactuar con el sistema, era fundamental para que los visitantes pudieran cumplir sus objetivos de manera intuitiva. Por su parte, la UI se refirió a todos los elementos gráficos y de diseño que permitieron la comunicación entre el usuario y la aplicación, como menús, botones y transiciones visuales.

Para cumplir con este objetivo, el desarrollo se realizó utilizando el motor Unity. Unity facilitó la creación de experiencias inmersivas y permitió compilar la aplicación para dispositivos iOS, integrando la plataforma ARKit para las funcionalidades de realidad aumentada.

Se emplearon herramientas específicas de Unity, como *UI Toolkit* y *UI Builder*, para la construcción de interfaces intuitivas y escalables. El proceso de diseño arquitectónico inició con la definición de flujos de usuario y la representación visual de las pantallas en Figma, asegurando una navegación intuitiva y una estética coherente con la identidad visual de la UVG.

Los resultados de las pruebas de usabilidad confirmaron que la aplicación resultante ofrecía una experiencia altamente intuitiva, alcanzando una calificación promedio de 9.2 sobre 10, y una satisfacción general con un promedio de 9.1. Esto demostró que se logró diseñar una interfaz que promovió la fluidez y la claridad durante el recorrido.

2.2. Uso de minijuegos y gamificación

Un componente clave para enriquecer la experiencia del usuario fue la integración de minijuegos. El módulo de minijuegos fue desarrollado con el propósito de añadir elementos interactivos y de gamificación en puntos específicos del recorrido, con el fin de entretener y motivar a los visitantes.

Se seleccionaron tres puntos de interés para el desarrollo de la fase piloto de los minijuegos: la Plaza Isabel Gutiérrez de Bosch, la Biblioteca Amparo Codina de Campollo y el Anfiteatro expuesto del nivel siete. Para cada uno de estos lugares se diseñó un minijuego temático, inspirado tanto en las mecánicas como en el diseño del entorno. Por ejemplo, para la biblioteca se desarrolló una trivia basada en las mecánicas de adquisición de conocimiento propias de dicho espacio.

El diseño de los minijuegos se basó en un *Game Design Document* (GDD), el cual permitió plasmar las ideas a recrear y facilitar su desarrollo, describiendo elementos como el estilo de arte, las reglas y las mecánicas.

Para fomentar la participación, se definió un *User Journey* compuesto por las etapas de *Onboarding* y *Progress*, buscando una progresión continua y atractiva. Dentro de la etapa de *Progress* se estableció un sistema de recompensas que funcionó como motivador extrínseco para incentivar a los usuarios a explorar y completar tareas. La validación con usuarios indicó que, si bien la motivación extrínseca fue importante, los jugadores también se sintieron atraídos por la motivación intrínseca, es decir, por el diseño y las mecánicas atractivas de los propios juegos.

Finalmente, se trabajó en la integración de la interfaz gráfica principal con las escenas que contenían los videojuegos, programando la aplicación para que, al llegar a un punto de interés, se pudiera acceder al minijuego correspondiente.

2.3. Módulo nativo de iOS (Xcode) y su desconexión inicial de Unity

Paralelamente al desarrollo de la aplicación principal en Unity, se creó un módulo de interfaz gráfica específico para el sistema operativo iOS. El objetivo principal de este proyecto independiente fue validar el uso de sensores de geolocalización interna, específicamente los sensores UWB (*Ultra-Wideband*), para su futura incorporación en la aplicación de recorridos virtuales.

El desarrollo de esta aplicación se realizó en el lenguaje de programación *Swift*, estándar para el sistema operativo iOS, utilizando la librería *SwiftUI* para la creación rápida de interfaces y el entorno de desarrollo integrado Xcode. Este módulo se enfocó en implementar la conexión entre la aplicación y los sensores UWB para mejorar la precisión de la geolocalización interna en el campus.

Como parte de la metodología, se desarrolló una herramienta complementaria que simulaba los sensores, utilizando un servidor *WebSocket* con *Node.js* y *React*. Esta simulación permitió probar la funcionalidad de la aplicación sin depender de la instalación física de los sensores en todo momento. La aplicación nativa de iOS implementó la lógica del tour, incluyendo la vista de progreso, los detalles de las paradas y el uso de *CoreLocation* para acceder al magnetómetro y calcular la dirección hacia el siguiente punto de interés.

Al momento de la realización de estos antecedentes, el módulo de iOS, centrado en la validación de sensores UWB, no se encontraba integrado con la aplicación de recorridos virtuales desarrollada en Unity. Por lo tanto, el alcance y la viabilidad técnica de combinar las capacidades de geolocalización de precisión del módulo iOS con la interfaz de usuario, la realidad aumentada y el sistema de minijuegos implementados en Unity constituyeron un desafío y un objetivo clave para el presente proyecto.

En la actualidad, las universidades enfrentan el reto constante de innovar en la forma en que comunican, enseñan y conectan con sus visitantes. El Centro de Innovación y Tecnología (CIT) de la Universidad del Valle de Guatemala (UVG) representó uno de los espacios más importantes para la promoción del conocimiento científico y tecnológico en el país. Sin embargo, el recorrido por sus instalaciones aún dependía de guías físicos o de señalización tradicional, lo cual limitaba la interactividad y la personalización de la experiencia para visitantes nacionales e internacionales.

El presente proyecto surgió como respuesta a esta necesidad, proponiendo el **desarrollo y rediseño completo de una aplicación móvil para iOS** que permitiera realizar recorridos virtuales inmersivos dentro del CIT, utilizando tecnologías de **realidad aumentada (AR)** y sensores de localización como **Estimote Beacons** y **Ultra-Wideband (UWB)**. Esta solución buscó transformar la forma en que los usuarios exploraban y comprendían el entorno del CIT, mediante una interfaz intuitiva, responsiva y con información contextual en tiempo real, superpuesta al entorno físico.

3.1. Beneficios del proyecto

Este desarrollo no solo benefició directamente a la UVG al modernizar la experiencia de visita dentro del CIT, sino que también posicionó a la institución como un referente en la aplicación de tecnologías emergentes en entornos académicos. La aplicación facilitó:

- La **orientación autónoma** de los visitantes dentro del CIT.
- La presentación **dinámica de contenido educativo e institucional**.
- La **interacción adaptativa** según la ubicación del usuario.
- La inclusión de **elementos lúdicos** como minijuegos y trivia para enriquecer la experiencia.

Además, sentó las bases para su expansión futura en otros espacios académicos, museos o centros de ciencia, tanto dentro como fuera de la universidad, reforzando el valor del proyecto como una solución escalable y transferible.

3.2. Aplicación del conocimiento en ingeniería en ciencias de la computación

El desarrollo del proyecto implicó la aplicación directa de los conocimientos adquiridos a lo largo de la carrera, integrando diversas áreas de especialización del programa académico de la UVG, entre ellas:

- **Desarrollo de software:** se implementó una arquitectura modular y multiplataforma, utilizando tecnologías como Unity, UI Toolkit y ARCore, con separación clara entre la lógica funcional y las interfaces gráficas.
- **Sistemas inteligentes:** se trabajó con sensores UWB y Beacons para determinar la ubicación del usuario, diseñando mecanismos para reaccionar contextualmente ante dicha información.
- **Ingeniería de software:** se aplicaron buenas prácticas de documentación, control de versiones, estructuras de carpetas organizadas y validación continua, asegurando un producto mantenible, escalable y de alta calidad.

Durante el desarrollo, también se llevaron a cabo pruebas de experiencia de usuario (UX), implementación de flujos de navegación optimizados, integración de tecnologías nativas para móviles y creación de minijuegos educativos, lo cual representó una aplicación práctica y transversal de los conocimientos adquiridos en cursos como *Interacción Humano-Computadora*, *Realidad Aumentada*, *Arquitectura de Software*, *Diseño de Experiencias* y *Proyecto Final*.

3.3. Importancia del desarrollo del proyecto

El proyecto se alineó con las necesidades reales de la UVG y aportó valor tanto educativo como institucional. En un contexto donde la tecnología redefine las formas de aprender e interactuar, contar con una aplicación como esta amplió las posibilidades pedagógicas del CIT y reforzó la imagen de la universidad como una entidad innovadora, comprometida con la mejora continua y la difusión del conocimiento.

La realización de este trabajo permitió asimismo a los integrantes del equipo consolidar sus competencias profesionales mediante la resolución de un reto de ingeniería real, contribuyendo directamente al fortalecimiento tecnológico y académico de la institución que formó parte esencial de su desarrollo profesional.

4.1. Objetivo general

Desarrollar una interfaz gráfica multiplataforma para una aplicación de recorridos virtuales, con el fin de mejorar la experiencia de navegación e interacción mediante el uso de UI Toolkit, ARCore y sensores UWB, en el Centro de Innovación y Tecnología de la UVG.

4.2. Objetivos específicos

- Diseñar e implementar una UI modular en Unity con UI Toolkit, que permita pantallas funcionales para escaneo, AR y navegación, garantizando escalabilidad y buenas prácticas.
- Integrar contenido de realidad aumentada con elementos visuales contextuales y dinámicos, mediante ARKit (iOS), para enriquecer la experiencia inmersiva del usuario.
- Aplicar principios de diseño UX centrado en el usuario, mediante la elaboración de wireframes, prototipos de alta fidelidad y pruebas iterativas de usabilidad con usuarios reales.
- Validar la accesibilidad, usabilidad e intuición de la interfaz mediante pruebas funcionales y recolección de retroalimentación directa de los usuarios durante una versión beta controlada.
- Documentar el código de la UI para garantizar su comprensión y futura reutilización mediante buenas prácticas.

5.1. Conceptos clave

En esta sección se presentan los fundamentos teóricos que sustentan el diseño e implementación del proyecto. Se abordan temas clave como Realidad Aumentada (AR), Interfaz de Usuario (UI), Experiencia de Usuario (UX) y Navegación Interior. Estos elementos proporcionan el marco conceptual necesario para el desarrollo de una aplicación educativa inmersiva, interactiva y funcional basada en recorridos virtuales guiados mediante sensores de proximidad.

5.1.1. Realidad Aumentada (AR)

La Realidad Aumentada (AR) es una tecnología que fusiona elementos virtuales con el entorno físico del usuario en tiempo real. Esto se logra superponiendo contenido digital como texto, imágenes, animaciones o modelos tridimensionales sobre la vista del mundo real, generalmente captada a través de la cámara de un dispositivo móvil. Esta interacción enriquece la percepción del entorno sin sustituirlo completamente, como sucede con la Realidad Virtual.

En el proyecto, la AR se utiliza para proporcionar una capa adicional de información visual sobre ubicaciones específicas dentro de una institución educativa. Esto convierte un recorrido físico convencional en una experiencia digital inmersiva e informativa. La implementación se apoya en frameworks especializados como ARKit y ARCore, que permiten detectar planos, anclar objetos virtuales y rastrear la posición del usuario [2].

5.1.2. Interfaz de Usuario (UI)

La Interfaz de Usuario (UI) es el conjunto de elementos gráficos y visuales mediante los cuales el usuario interactúa con la aplicación. Incluye componentes como botones, etiquetas, contenedores, paneles, íconos y menús. Una UI bien diseñada debe ser intuitiva, clara y accesible para facilitar la navegación y la comprensión de las funcionalidades ofrecidas.

Para este proyecto, se emplea el sistema UI Toolkit de Unity, que permite definir interfaces de forma modular utilizando archivos UXML (estructura) y USS (estilos), lo cual separa la lógica visual

del comportamiento funcional. Esta metodología favorece la escalabilidad y el mantenimiento de la interfaz, permitiendo también una experiencia adaptativa en distintas resoluciones de pantalla [5].

5.1.3. Experiencia de Usuario (UX)

La experiencia de usuario (UX) hace referencia a cómo una persona percibe, siente y responde emocionalmente al interactuar con una aplicación. Engloba aspectos como la facilidad de uso, el flujo de navegación, la disposición visual, la accesibilidad, la respuesta del sistema y la estética. Una buena UX debe minimizar la fricción, maximizar la eficiencia y generar satisfacción en el usuario.

En este proyecto, la UX fue diseñada a partir de wireframes y prototipos funcionales desarrollados en Figma. Estos prototipos fueron validados con usuarios reales para identificar puntos de mejora antes de su implementación en Unity. La retroalimentación obtenida permitió optimizar la estructura de las pantallas, el tamaño de los elementos y la disposición de la información, priorizando un flujo lógico e intuitivo [9].

5.1.4. Navegación interior (indoor navigation)

La navegación interior es una técnica para localizar y guiar usuarios dentro de espacios cerrados, donde el GPS tradicional pierde precisión. Esta técnica es esencial para aplicaciones como guías de museos, hospitales, universidades o aeropuertos. Se basa en tecnologías como Wi-Fi, Bluetooth Low Energy (BLE), RFID o Ultra Wideband (UWB).

En este caso, se implementa mediante sensores Estimote Beacons con tecnología UWB, que permiten determinar la posición del usuario con alta precisión en interiores. Esta información se utiliza para activar dinámicamente contenido AR cuando el usuario se encuentra en una zona determinada, automatizando la transición entre etapas del recorrido [7].

5.2. Herramientas y tecnologías empleadas

A continuación se detallan las herramientas, motores y lenguajes utilizados durante el desarrollo del proyecto. La selección de estas tecnologías se basó en criterios como compatibilidad multiplataforma, facilidad de integración con sensores y soporte para Realidad Aumentada.

5.2.1. Unity 3D

Unity es un motor de desarrollo de aplicaciones y videojuegos multiplataforma que permite integrar lógica de programación, gráficos 2D/3D, animaciones, sistemas físicos, sensores y Realidad Aumentada. Su capacidad de exportar a múltiples plataformas (iOS, Android, Windows, etc.) lo convierte en una herramienta ideal para proyectos de alto nivel interactivo.

En este proyecto, Unity ha servido como base para integrar todos los módulos: AR, interfaz de usuario, control de sensores y navegación entre pantallas. También ha facilitado la depuración y validación constante mediante simulaciones en el editor antes de compilar en dispositivos reales.

5.2.2. Unity UI Toolkit

UI Toolkit es el sistema moderno de creación de interfaces de Unity, orientado a separar la presentación visual (UXML y USS) de la lógica (C#). Reemplaza los enfoques tradicionales como Unity Canvas y Unity UI, ofreciendo mayor modularidad, rendimiento y escalabilidad.

Utiliza los siguientes lenguajes:

- UXML: para definir la jerarquía de elementos visuales.
- USS: para definir los estilos (colores, tamaños, márgenes).
- C#: para controlar eventos, navegación y lógica de comportamiento.

Este sistema permite una mayor organización en proyectos con múltiples pantallas y componentes visuales [5].

5.2.3. UI Builder

UI Builder es la herramienta visual oficial de Unity para la construcción de interfaces de usuario utilizando el sistema UI Toolkit. Esta interfaz gráfica permite a los desarrolladores diseñar y editar archivos UXML (Unity Extensible Markup Language) de forma visual, sin necesidad de escribir el código estructural a mano. UI Builder ofrece un entorno de diseño tipo "arrastrar y soltar", donde los componentes visuales como botones, etiquetas, contenedores y listas pueden ser añadidos y organizados jerárquicamente con facilidad.

Además, UI Builder permite aplicar clases de estilo definidas en archivos USS (Unity Style Sheets), lo cual favorece la separación de diseño visual y lógica funcional. También ofrece una pre-visualización en tiempo real de cómo se verá la interfaz en distintas resoluciones y dispositivos, lo cual es esencial para garantizar un diseño responsivo en aplicaciones móviles.

En este proyecto, UI Builder fue utilizado extensivamente para diseñar interfaces modulares como la PantallaEscaneo y la PantallaARTour, estructuradas en contenedores VisualElement y personalizadas mediante clases de estilo. Estas estructuras luego fueron controladas programáticamente desde scripts en C que gestionan la lógica de navegación, visualización y comportamiento dinámico.

5.2.4. Blender

Blender es un software de creación tridimensional de código abierto ampliamente utilizado en la industria del modelado 3D, animación, escultura digital, renderizado y diseño de videojuegos. Es una herramienta robusta, gratuita y multiplataforma que proporciona un conjunto completo de funcionalidades para la producción de contenido 3D, incluyendo edición de mallas, materiales, texturas, rigging, simulaciones físicas, y exportación a diversos formatos compatibles con motores como Unity.

En el contexto de este proyecto, Blender fue utilizado para modelar objetos tridimensionales que forman parte del contenido AR desplegado durante los recorridos. Estos modelos incluyen esculturas patrimoniales, monumentos, estructuras arquitectónicas y objetos contextuales relevantes al recorrido educativo. Cada modelo fue optimizado para uso en tiempo real (real-time rendering) mediante técnicas como reducción de polígonos, simplificación de materiales y empaquetado de texturas, antes de ser exportado en formato .fbx o .glb e integrado en Unity.

La correcta creación y exportación desde Blender garantizó que los objetos 3D se renderizaran de manera eficiente sobre la escena de cámara en la aplicación móvil sin comprometer el rendimiento del dispositivo.

5.2.5. Figma

Figma es una plataforma de diseño de interfaces centrada en la colaboración en línea y en tiempo real. Se utiliza ampliamente para diseñar sistemas de interfaz de usuario (UI), planificar experiencias de usuario (UX), y crear prototipos navegables que pueden ser compartidos, comentados y validados por equipos multidisciplinares. Su enfoque basado en la nube permite que varios diseñadores, desarrolladores y usuarios participen simultáneamente en el proceso creativo, eliminando barreras de sincronización y compatibilidad entre plataformas.

Dentro de este proyecto, Figma fue una herramienta clave en la fase de prototipado y validación temprana. Se construyeron flujos de navegación interactivos que simulan el recorrido completo de la aplicación, desde el inicio hasta el despliegue de contenido en AR. Estas simulaciones se utilizaron para realizar pruebas con usuarios reales, recoger retroalimentación sobre usabilidad, claridad de la navegación y estética general. Esta información fue luego incorporada en los diseños definitivos antes de pasar a su implementación técnica en Unity.

El uso de Figma permitió anticipar posibles problemas de comprensión o flujo, optimizar el diseño centrado en el usuario y garantizar una experiencia final coherente, intuitiva y funcional.

5.2.6. Microsoft Forms

Microsoft Forms es una herramienta en línea que permite crear formularios, encuestas y cuestionarios de manera sencilla y rápida. Forma parte del ecosistema de Microsoft 365 y se orienta a la recopilación estructurada de datos mediante una interfaz accesible y multiplataforma. Su funcionalidad incluye distintos tipos de preguntas, lógica de ramificación, recopilación automática de respuestas y exportación de datos para su análisis.

En este proyecto, Microsoft Forms será utilizado como instrumento para la evaluación de la experiencia de usuario (UX) durante la fase de validación de una versión beta de la aplicación. A través de formularios estructurados, se recopilarán datos tanto cuantitativos como cualitativos sobre aspectos como la usabilidad, la claridad visual, la fluidez de navegación y la satisfacción general. Esta herramienta permitirá centralizar la retroalimentación de los usuarios, identificar patrones y tomar decisiones informadas para mejorar el diseño final de la interfaz antes de su implementación definitiva [?].

5.2.7. ARKit

ARKit es el marco de desarrollo de Realidad Aumentada desarrollado por Apple, diseñado específicamente para dispositivos iOS (iPhone y iPad). Este framework ofrece una serie de potentes funcionalidades para construir experiencias AR avanzadas, entre las cuales destacan: detección y mapeo de superficies planas (horizontal y vertical), seguimiento del entorno en seis grados de libertad (6DoF), detección de movimiento, anclaje espacial de objetos virtuales, iluminación ambiental realista, reconocimiento de rostros y cuerpos, y colaboración multijugador en tiempo real.

ARKit se integra de forma nativa con Unity mediante el uso del paquete AR Foundation, que actúa como una capa de abstracción unificada para trabajar tanto con ARKit como con ARCore. En este proyecto, ARKit se empleó para permitir que los dispositivos iOS posicionaran con precisión

contenido 3D en la escena real capturada por la cámara. Esto incluye el anclaje de modelos creados en Blender en ubicaciones específicas dentro del recorrido virtual, respetando las condiciones espaciales y de iluminación del entorno físico.

Gracias a ARKit, se logró una experiencia AR fluida, estable y altamente responsiva, fundamental para garantizar una interacción intuitiva del usuario con los elementos del recorrido educativo en iOS.

5.2.8. ARCore

ARCore es el framework de Realidad Aumentada desarrollado por Google, orientado a dispositivos Android. De forma equivalente a ARKit, ARCore permite a los desarrolladores integrar funcionalidades AR robustas como la detección de planos horizontales y verticales, seguimiento del entorno, reconocimiento de movimiento, estimación de iluminación y posicionamiento preciso de objetos virtuales en el espacio físico.

En el contexto de Unity, ARCore se implementa a través del mismo paquete AR Foundation, lo que permite escribir una única base de código multiplataforma que funcione tanto en dispositivos iOS como Android. Esto facilita considerablemente el desarrollo y mantenimiento de la aplicación, al tiempo que garantiza una experiencia coherente para todos los usuarios sin importar el sistema operativo de su dispositivo.

Dentro del proyecto, ARCore fue clave para desplegar objetos 3D interactivos en dispositivos Android. Estos objetos, previamente diseñados y optimizados en Blender, fueron posicionados sobre superficies reales detectadas por la cámara y se mantuvieron anclados al entorno durante el desplazamiento del usuario. Esta capacidad fue esencial para el objetivo del proyecto de ofrecer recorridos virtuales inmersivos que respondan al contexto físico real del usuario en tiempo real.

5.2.9. C# en Unity

C# es el lenguaje principal para la programación de scripts en Unity. Permite controlar comportamientos, gestionar eventos, hacer transiciones de UI, manipular datos, leer sensores, y definir reglas de negocio.

Sintaxis general en Unity:

```
using UnityEngine;

public class NombreDeScript : MonoBehaviour {
    void Start() {
        // Código que se ejecuta al iniciar
    }

    void Update() {
        // Código que se ejecuta cada frame
    }

    public void MiFuncion() {
        // Acción personalizada
    }
}
```

Ejemplo simple:

```
public void IrAPantallaARTour() {
    UIDocumentARTour.rootVisualElement.style.display = DisplayStyle.Flex;
    UIDocumentEscaneo.rootVisualElement.style.display = DisplayStyle.None;
}
```

5.2.10. UXML

UXML es un lenguaje basado en XML utilizado para definir la estructura jerárquica de interfaces gráficas en Unity. Permite declarar contenedores, botones, textos, listas y cualquier otro componente visual.

Sintaxis general:

```
<engine:UIDocument xmlns:ui="UnityEngine.UIElements">
  <ui:VisualElement name="contenedor">
    <ui:Label text="Texto de ejemplo" />
    <ui:Button name="btnAccion" text="Presionar" />
  </ui:VisualElement>
</engine:UIDocument>
```

Ejemplo simple en el proyecto:

```
<engine:UIDocument xmlns:ui="UnityEngine.UIElements">
  <ui:VisualElement name="mainContainer" class="pantalla-artour">
    <ui:Label text="¡Bienvenido al recorrido virtual!" class="titulo" />
    <ui:Button name="btnSalir" text="Salir" class="boton-salir"/>
  </ui:VisualElement>
</engine:UIDocument>
```

5.2.11. USS

USS (Unity Style Sheets) es un lenguaje similar a CSS usado para estilizar elementos visuales declarados en UXML. Define propiedades como color, márgenes, fuentes, alineaciones, dimensiones y clases reutilizables.

Sintaxis general:

```
.nombreClase {
    propiedad: valor;
}
```

Ejemplo en el proyecto:

```
.titulo {
    font-size: 20px;
    color: white;
    margin-top: 10px;
}
```

```
.boton-salir {  
    background-color: #cc0000;  
    padding: 8px;  
    border-radius: 10px;  
}
```

La metodología empleada en el rediseño y desarrollo de la aplicación se fundamentó en un enfoque ágil, iterativo y centrado en el usuario, priorizando la experiencia de uso, la eficiencia del sistema y la integración tecnológica mediante herramientas modernas de prototipado, diseño y programación dentro del ecosistema de Unity. A continuación, se detalla cada etapa del proceso metodológico ejecutado durante el desarrollo del proyecto.

6.1. Enfoque de desarrollo iterativo y centrado en Usuario

El proyecto adoptó una metodología de desarrollo iterativo fundamentada en ciclos de implementación-evaluación-refinamiento, donde la retroalimentación de usuarios reales constituyó el elemento rector de las decisiones de diseño. Esta elección metodológica se justificó por la naturaleza experiencial de las aplicaciones de realidad aumentada, donde la usabilidad y la inmersión solo pueden validarse mediante pruebas en contexto real de uso.

6.1.1. Justificación del enfoque iterativo

Se determinó que un enfoque waterfall tradicional resultaría inadecuado debido a tres factores críticos identificados en la fase de planificación:

1. **Complejidad perceptual de AR:** la experiencia de realidad aumentada genera expectativas y desafíos de usabilidad que no pueden anticiparse completamente en fase de diseño, requiriendo validación empírica iterativa.
2. **Diversidad de usuarios objetivo:** la aplicación debe servir a múltiples perfiles (estudiantes prospecto, padres de familia, colaboradores), cada uno con expectativas y niveles de alfabetización tecnológica distintos, necesitando ajustes basados en retroalimentación segmentada.
3. **Restricciones de migración técnica:** la transición desde Unity 2022.3.18f1 a Unity 6000.0.10f1 presentaba incertidumbres técnicas que requerían validación continua de decisiones arquitectónicas.

6.1.2. Ciclos de iteración planificados

Se establecieron dos ciclos principales de desarrollo con evaluación de usuarios:

Iteración 1 (junio-julio 2025): evaluación de prototipo funcional con interfaz básica para validar navegabilidad fundamental, claridad informativa, identificación de problemas técnicos críticos y detección de expectativas no cubiertas por el diseño inicial.

Iteración 2 (agosto-octubre 2025): evaluación de sistema rediseñado implementando mejoras identificadas, validando efectividad de soluciones propuestas, refinamiento de elementos visuales según especificaciones de diseñadora gráfica, y verificación de integración completa de funcionalidades solicitadas.

6.1.3. Estrategia de evaluación con usuarios

Se diseñó un proceso de evaluación cualitativa y cuantitativa que permitiera capturar tanto métricas objetivas de usabilidad como percepciones subjetivas de experiencia. La metodología de evaluación incluyó:

- **Selección de participantes:** muestreo intencional buscando representatividad de perfiles de usuario objetivo (estudiantes UVG actuales, estudiantes de otras instituciones, padres de familia, colaboradores universitarios).
- **Criterios de diversidad:** inclusión deliberada de participantes con diferentes niveles de familiaridad tecnológica (bajo, medio, alto) para identificar barreras de usabilidad en usuarios menos experimentados.
- **Protocolo de evaluación:** pruebas en campo realizadas en las instalaciones físicas de UVG, permitiendo validar la experiencia AR en condiciones reales de iluminación, espacio y conectividad.
- **Instrumentos de medición:** formularios estructurados con escalas Likert (1-5) para métricas cuantificables, complementados con preguntas abiertas para retroalimentación cualitativa profunda.
- **Análisis de resultados:** categorización temática de retroalimentación cualitativa y cálculo de estadísticas descriptivas (promedios, desviación estándar) para métricas cuantitativas.

Esta estructura metodológica garantizó que las decisiones de diseño y desarrollo estuvieran fundamentadas en evidencia empírica de usuarios reales, minimizando sesgos subjetivos del equipo de desarrollo.

6.2. Diseño de la arquitectura de la aplicación

El proceso inició con la conceptualización y diseño visual de la aplicación mediante prototipos desarrollados en Figma. La selección de Figma como herramienta de prototipado se fundamentó en su capacidad para generar prototipos interactivos de alta fidelidad, facilitar la colaboración remota con la diseñadora gráfica Andrea Rebecca Leiva Pérez, y exportar especificaciones técnicas precisas (colores hexadecimales, tipografías, dimensiones) directamente utilizables en la implementación en Unity.

Los prototipos incluyeron la totalidad de las pantallas clave de la experiencia de usuario, definiendo tanto el flujo de navegación como los elementos interactivos esenciales. Se trabajó bajo una línea gráfica coherente con la identidad visual de la Universidad del Valle de Guatemala, empleando tonos verdes y blancos como paleta cromática base para generar una experiencia estética armoniosa y representativa de la institución.

Desde las etapas iniciales, se definieron elementos de interfaz críticos tales como botones, menús, pantallas superpuestas, iconografía institucional y estructuras responsive, orientadas específicamente hacia una experiencia fluida en dispositivos móviles con sistemas operativos Android e iOS. Esta arquitectura visual constituyó el fundamento para guiar la posterior implementación técnica en Unity, haciendo uso de las herramientas UI Toolkit y UI Builder.

6.2.1. Selección de tecnología UI: UI Toolkit vs uGUI

Durante la fase de investigación se evaluaron dos alternativas principales para el desarrollo de interfaces en Unity: el sistema tradicional uGUI (Unity GUI) y el moderno UI Toolkit. La decisión de adoptar UI Toolkit se fundamentó en criterios técnicos específicos:

- **Paradigma declarativo:** UI Toolkit emplea archivos UXML (similares a HTML) y USS (similares a CSS), permitiendo separación clara entre estructura, presentación y lógica, facilitando el trabajo colaborativo con diseñadores.
- **Rendimiento optimizado:** UI Toolkit utiliza un sistema de retained-mode rendering más eficiente que el immediate-mode de uGUI, crítico para aplicaciones AR donde el presupuesto de rendimiento es limitado.
- **Escalabilidad multiplataforma:** el sistema de estilos basado en USS permite adaptación responsiva automática a diferentes resoluciones sin requerir múltiples canvases predefinidos como en uGUI.
- **Futuro de Unity:** documentación oficial de Unity indica que UI Toolkit es la tecnología UI recomendada para proyectos nuevos, mientras uGUI se encuentra en modo de mantenimiento sin nuevas características planificadas.
- **Curva de aprendizaje transferible:** desarrolladores con experiencia en desarrollo web pueden transferir conocimientos de HTML/CSS a UXML/USS, reduciendo la barrera de entrada.

Esta decisión arquitectónica fundamental determinó el enfoque de todo el desarrollo de interfaz subsecuente, requiriendo inversión inicial en aprendizaje de la tecnología pero resultando en un sistema más mantenible y escalable a largo plazo.

6.3. Desarrollo de la aplicación

Posterior a la fase de diseño, se realizó una investigación exhaustiva sobre las mejores prácticas contemporáneas para el desarrollo de interfaces gráficas en Unity mediante consulta de documentación oficial, análisis de proyectos open-source en repositorios públicos de Unity, y estudio de patrones arquitectónicos recomendados por la comunidad de desarrolladores. Como resultado de esta investigación, se confirmó la selección de UI Toolkit y UI Builder como herramientas principales, adoptando un patrón de diseño modular basado en componentes reutilizables.

6.3.1. Arquitectura de layouts jerárquicos

Se implementó un sistema de layouts jerárquicos basado en el patrón de diseño Layer Architecture, donde diferentes niveles de la interfaz se organizaron en capas independientes con responsabilidades específicas. Esta decisión arquitectónica se justificó por:

- **Separación de responsabilidades:** cada capa maneja un tipo específico de contenido (base, modales, popups, menús), facilitando el mantenimiento y debugging.
- **Control de z-index implícito:** la jerarquía de capas en el archivo BaseLayout.xml establece automáticamente el orden de renderizado, evitando conflictos visuales.
- **Gestión de estado simplificada:** las capas pueden mostrarse u ocultarse independientemente sin afectar el estado de otras capas.
- **Escalabilidad modular:** nuevas pantallas o overlays pueden agregarse sin modificar la estructura base.

Durante esta etapa, se desarrollaron las siguientes interfaces clave:

- **Pantalla principal:** diseñada para permitir al usuario seleccionar entre un recorrido exprés o completo, acceder al módulo de minijuegos y visualizar información general sobre la aplicación y la institución.
- **Onboarding:** diseñado para ofrecer al usuario una introducción o tutorial que le permita conocer qué puede esperar de la aplicación y cómo será su experiencia al utilizarla.
- **Popup de espera de conexión:** originalmente concebida como una pantalla dedicada al escaneo de códigos QR, fue rediseñada como un elemento emergente (popup) cuya función consistió en indicar al usuario el estado de conexión con los sensores UWB mediante Estimote Beacons. La transición de pantalla completa a popup se fundamentó en principios de economía de interacción: evitar transiciones de escena innecesarias que incrementan tiempo de carga percibido y permiten mantener el contexto AR activo durante el proceso de conexión. Durante la fase de desarrollo, se implementó un mecanismo de bypass temporal para simular la conexión de sensores mientras se finalizaba la lógica de detección real.
- **Pantalla AR Tour:** constituye la interfaz principal y más compleja del sistema, donde se visualiza la experiencia completa del recorrido. Esta pantalla superpone la vista de la cámara del dispositivo con elementos visuales interactivos, incluyendo la ubicación actual del usuario, un menú hamburguesa con opciones adicionales, instrucciones de navegación contextuales y detalles informativos sobre cada punto de interés, facilitando así un recorrido inmersivo y enriquecido mediante realidad aumentada.
- **Menú hamburguesa:** diseñado para proveer al usuario opciones de control avanzadas, tales como reiniciar el recorrido, acceder a la sección de ayuda, ajustar preferencias de la aplicación y un botón de salida para retornar a la pantalla principal.

6.3.2. Sistema de navegación modular

Se implementó un sistema modular de escenas y UIDocuments, gestionado por múltiples controladores de navegación específicos para cada módulo funcional. Este enfoque arquitectónico permitió mostrar u ocultar dinámicamente los layouts de las diferentes pantallas sin necesidad de recargar la escena principal mediante Unity's SceneManager, optimizando así el rendimiento y la fluidez de la

experiencia de usuario. La decisión de evitar transiciones de escena frecuentes se basó en mediciones empíricas que demostraron tiempos de carga de 1.2-1.5 segundos por transición, acumulando latencia perceptible en flujos de navegación complejos.

6.3.3. Estrategia de pruebas técnicas

Se realizaron pruebas unitarias e integradas dentro de las instalaciones de la universidad de forma sistemática después de cada implementación significativa de funcionalidad. El protocolo de pruebas incluyó:

- **Pruebas en editor:** validación inicial de lógica y comportamiento visual en Unity Editor (Play Mode).
- **Builds de desarrollo:** compilación de aplicación con símbolos de debugging habilitados para prueba en dispositivos iOS físicos.
- **Pruebas de integración:** validación de sincronización entre módulo de navegación (desarrollado paralelamente por compañero de proyecto) y módulo de interfaz.
- **Pruebas de campo:** ejecución de recorridos completos en instalaciones físicas de UVG para validar experiencia en condiciones reales (iluminación natural, espacios amplios, interferencias WiFi).

Este proceso iterativo de testing permitió identificar y resolver problemas de integración antes de que afectaran la experiencia de usuarios finales durante las evaluaciones formales.

6.4. Integración con módulo de minijuegos

Se reutilizaron y adaptaron minijuegos previamente desarrollados en el proyecto del año anterior (ciclo académico 2024), los cuales se encontraban implementados en escenas independientes (GameMenu.unity, Breakout.unity, FlappyJack.unity y Trivia.unity). La decisión de reutilizar estos componentes se fundamentó en principios de economía de desarrollo y aprovechamiento de trabajo validado previamente, en lugar de desarrollar sistemas de gamificación completamente nuevos.

6.4.1. Estrategia de migración entre versiones de Unity

La integración de los minijuegos presentó desafíos técnicos significativos debido a la diferencia de versiones de Unity entre el proyecto anterior (2022.3.18f1 LTS) y el proyecto actual (6000.0.10f1). Se evaluaron dos estrategias metodológicas alternativas:

Opción 1: Migración automática mediante el sistema de actualización de proyectos de Unity, que analiza y convierte automáticamente assets, scripts y configuraciones al formato de la nueva versión.

Opción 2: Integración manual selectiva extrayendo únicamente assets visuales/sonoros y reescribiendo completamente la estructura UXML/USS y scripts de lógica.

Se optó por la Opción 2 después de realizar pruebas piloto con la migración automática que revelaron:

- Varios archivos UXML con errores de sintaxis requiriendo corrección manual
- Warnings de API deprecada en la mayoría de los scripts
- Pérdida de referencias en prefabs serializados
- Incompatibilidades en el pipeline de renderizado de materiales

La integración manual, aunque requirió mayor inversión de tiempo inicial (7 días vs 3 días estimados para migración automática), resultó en código limpio, documentado y completamente compatible con la arquitectura moderna del proyecto, evitando deuda técnica futura y garantizando mantenibilidad a largo plazo.

6.4.2. Proceso de recreación y adaptación

El proceso de integración manual siguió una secuencia metodológica estructurada:

1. **Extracción de assets:** copia selectiva de sprites, texturas, audio clips y datos de contenido (preguntas de trivia, configuraciones de juego) desde el proyecto 2022.3.18f1.
2. **Análisis de estructura original:** estudio de la arquitectura UXML/USS del proyecto anterior para comprender la organización de componentes visuales.
3. **Recreación en UI Builder:** reconstrucción de las interfaces utilizando la versión actualizada de UI Builder de Unity 6000, aplicando sintaxis moderna y aprovechando nuevas características disponibles.
4. **Adaptación de estilos:** modificación de archivos USS para utilizar las clases de utilidad definidas en baseLayout.uss del proyecto actual, garantizando consistencia visual con la nueva identidad de marca.
5. **Reescritura de scripts:** adaptación de lógica de juego a la nueva API de UI Toolkit, implementando mejores prácticas de Unity 6000 y convenciones de código del proyecto actual.
6. **Testing independiente:** validación exhaustiva de cada minijuego en escena aislada antes de integración al proyecto principal.

Se procedió a la creación de un nuevo menú de minijuegos desarrollado íntegramente desde cero, con el objetivo primordial de garantizar su alineación visual y funcional con el sistema de diseño de la nueva versión de la aplicación.

6.4.3. Decisión de separación funcional

Los minijuegos se hicieron accesibles directamente desde el menú principal, en lugar de integrarlos dentro del flujo asociado a puntos de interés específicos del recorrido, como se implementó en la versión del proyecto del año anterior. Esta decisión de diseño respondió a múltiples factores metodológicos:

- **Recomendación de asesor:** el asesor del proyecto identificó que la estructuración anterior constituía una práctica subóptima desde la perspectiva de la experiencia de usuario (UX), generando sobrecarga cognitiva durante el recorrido informativo.

- **Principio de separación de responsabilidades:** la mezcla de contenido informativo-navegacional con contenido lúdico-gamificado diluía el enfoque de ambas experiencias.
- **Flexibilidad de acceso:** la separación permitió que usuarios interesados exclusivamente en información del campus pudieran completar el tour sin presión de participar en actividades de gamificación.
- **Optimización técnica:** evitar carga simultánea de assets de juegos durante el tour AR redujo presión sobre memoria y procesador del dispositivo.

6.4.4. Estado de integración y gestión de Branches

Debido al rediseño integral de la interfaz ocurrido posteriormente a la integración inicial de minijuegos, estos se mantuvieron en una branch de desarrollo separada (/main/dev/minijuegos en Unity DevOps Version Control) para permitir su adaptación sistemática a la nueva arquitectura visual sin bloquear el avance del desarrollo principal. Esta decisión metodológica de versionamiento permitió desarrollo paralelo independiente, evitando conflictos de merge complejos y garantizando que ambas líneas de desarrollo pudieran avanzar sin interferencias.

6.5. Integración de multimedia

Durante el desarrollo del proyecto, se incorporó contenido multimedia diverso con el propósito de enriquecer la experiencia visual e informativa de la aplicación. La selección y preparación de assets multimedia siguió un proceso metodológico estructurado:

6.5.1. Proceso de adquisición y preparación de assets

1. **Fotografías de instalaciones:** captura de fotografías de alta resolución del interior del campus extraídas de la multimedia de la Universidad del Valle de Guatemala.
2. **Modelados 3D:** desarrollo de modelos tridimensionales en Blender de la flecha.
3. **Iconografía:** diseño de iconos vectoriales en Adobe Illustrator siguiendo guía de estilo de la aplicación, exportados en múltiples resoluciones (@1x, @2x, @3x) para compatibilidad con diferentes densidades de pantalla iOS.
4. **Optimización para dispositivos móviles:** todos los assets multimedia se procesaron mediante pipeline de optimización antes de integración:
 - Compresión de imágenes mediante formato ASTC (Adaptive Scalable Texture Compression) para iOS.
 - Reducción de tamaño de archivos sin pérdida perceptible de calidad visual.
 - Generación automática de mipmaps para texturas 3D.
 - Conversión de modelos 3D a formato FBX optimizado.

Estos elementos fueron distribuidos estratégicamente a lo largo de todas las pantallas de la aplicación, concentrándose principalmente en la pantalla de AR Tour, donde cumplieron funciones tanto estéticas como informativas para guiar y contextualizar la experiencia del usuario durante el recorrido virtual.

6.6. Debug y pruebas internas

Durante las fases intermedias del desarrollo, se realizaron pruebas constantes y sistemáticas con el objetivo de asegurar la estabilidad, rendimiento y funcionalidad de la aplicación. Este proceso de depuración siguió una metodología estructurada de identificación-reproducción-corrección-verificación.

6.6.1. Metodología de debugging

El proceso de debugging se organizó en torno a las siguientes prácticas metodológicas:

- **Logging sistemático:** implementación de sistema de logging categorizado (Debug.Log, Debug.LogWarning, Debug.LogError) en puntos críticos del código para facilitar trazabilidad de errores.
- **Breakpoints estratégicos:** uso del debugger integrado de Visual Studio para detener ejecución en puntos sospechosos y examinar estado de variables.
- **Profiling de rendimiento:** utilización de Unity Profiler para identificar cuellos de botella de rendimiento, memory leaks y picos de garbage collection.
- **Reproducción sistemática:** documentación de pasos exactos para reproducir cada bug identificado, facilitando verificación de correcciones.

Este proceso permitió identificar y solucionar errores críticos, entre los cuales destacaron:

- Colisiones de elementos UXML que bloqueaban la interacción correcta con botones y otros componentes interactivos, causadas por overlays invisibles con pointer-events habilitados que capturaban eventos táctiles destinados a elementos subyacentes.
- Errores (bugs) en diversos elementos de la interfaz de usuario que se manifestaban al realizar acciones específicas o secuencias particulares de interacción, particularmente relacionados con el orden de inicialización de VisualElements.

6.6.2. Refactorización arquitectónica

La identificación de estos problemas condujo a la toma de una decisión técnica significativa: realizar una restauración y actualización completa de la interfaz de usuario. Esta decisión se fundamentó en análisis costo-beneficio que determinó que:

- **Costos de parches incrementales:** resolver bugs individualmente mediante parches específicos resultaría en código frágil con alta probabilidad de regresiones futuras.
- **Beneficios de refactorización:** una reestructuración completa permitiría implementar arquitectura robusta desde fundamentos sólidos, reduciendo deuda técnica a largo plazo.
- **Ventana de oportunidad:** el proyecto se encontraba en fase intermedia donde refactorización era viable sin comprometer cronograma general.

La reestructuración implicó la implementación correcta de sistemas de layouts nativos de Unity (Flexbox-based), la optimización integral de los scripts en C# mediante aplicación de patrones de diseño (Singleton para managers, Observer para eventos UI), y una mejora sustancial del rendimiento

general mediante ajustes iterativos y refactorización exhaustiva del código base. Este proceso de rediseño técnico resultó en una aplicación significativamente más estable, eficiente y mantenible, validado mediante reducción del 78 % en crashes reportados en pruebas internas post-refactorización.

6.7. Estrategia de control de versiones

Se implementó una estrategia de control de versiones basada en el modelo de Feature Branch Workflow, utilizando Unity DevOps Version Control (anteriormente Plastic SCM) como sistema de versionamiento. La selección de esta plataforma se justificó por su integración nativa con Unity Editor, soporte especializado para assets binarios grandes (modelos 3D, texturas, escenas), y capacidades de merge visual para escenas de Unity.

6.7.1. Estructura de branching

La estrategia de branches se organizó jerárquicamente:

- **Branch main:** código estable y funcional en todo momento, desplegable a dispositivos de prueba en cualquier momento. Esta branch solo recibió commits mediante merge de feature branches completamente validadas.
- **Feature branches:** branches de vida corta (3-7 días) dedicadas a desarrollo de funcionalidades específicas:
 - **arrow:** Desarrollo del sistema de flecha 3D y navegación AR.
 - **info-modals-tour:** desarrollo del sistema de modales POI y footer dinámico.
 - **minijuegos:** integración de minijuegos desde proyecto anterior.

6.7.2. Protocolo de integración

Los merges a la branch main siguieron un protocolo riguroso:

1. **Validación en branch:** testing exhaustivo de la funcionalidad en la feature branch aislada.
2. **Build de prueba:** compilación de build iOS y ejecución en dispositivo físico.
3. **Code review:** revisión de código por miembro del equipo (compañero de proyecto o asesor).
4. **Merge request:** creación de solicitud de merge con descripción detallada de cambios.
5. **Pruebas de regresión:** validación de que funcionalidades existentes no se vieron afectadas negativamente.
6. **Merge final:** integración a main solo después de aprobación de todos los criterios anteriores.

Esta metodología de versionamiento garantizó que la branch principal mantuviera alta calidad de código y estabilidad funcional en todo momento, facilitando demostraciones a stakeholders y pruebas con usuarios sin riesgo de exponer bugs en desarrollo activo.

6.8. Documentación técnica

Se elaboró una guía técnica detallada y comprehensiva que documenta los aspectos esenciales del sistema desarrollado. El proceso de documentación siguió el principio de "documentación como código", donde la documentación se generó de forma continua durante el desarrollo en lugar de como actividad post-hoc al finalizar el proyecto.

6.8.1. Metodología de documentación

La documentación técnica se organizó mediante múltiples niveles de granularidad:

1. **Documentación inline (XML Comments):** todos los métodos públicos y clases principales se documentaron mediante XML comments estándar de C# (`/// <summary>`), permitiendo generación automática de documentación mediante herramientas como DocFX.
2. **Comentarios de región:** organización de código en secciones lógicas mediante directivas `#region/#endregion`, facilitando navegación en IDEs y comprensión de estructura general.
3. **README de componentes:** archivos markdown independientes documentando sistemas complejos (sistema de layouts, arquitectura de navegación, integración UWB).
4. **Diagramas arquitectónicos:** diagramas UML de clases y secuencia documentando relaciones entre componentes principales y flujos de interacción críticos.

6.8.2. Contenido documentado

La documentación técnica incluyó los siguientes aspectos esenciales:

- Estructura organizacional de carpetas, escenas y jerarquías de objetos, siguiendo convenciones de nomenclatura establecidas.
- Funcionamiento detallado de los scripts implementados en C#, incluyendo sus dependencias, patrones de diseño utilizados (Singleton, Observer, State), y lifecycle methods.
- Explicación del uso y sintaxis de archivos .uxml y .uss para la definición de interfaces y estilos, incluyendo clases de utilidad reutilizables y sistema de themes.
- Descripción del flujo de navegación entre pantallas y los mecanismos de activación de eventos del sistema, documentando el grafo completo de transiciones posibles.
- Instrucciones claras y específicas para extender, modificar o mantener el sistema en desarrollos futuros, incluyendo checklist de verificación para agregar nuevas pantallas o puntos de interés.
- API pública documentada para integración con el módulo de sensores UWB, especificando signatures de métodos, parámetros esperados, valores de retorno y casos de uso típicos.

Esta documentación técnica garantiza que cualquier desarrollador futuro que requiera trabajar con el proyecto pueda comprender de manera eficiente el funcionamiento integral del sistema y contribuir de forma efectiva al mantenimiento, extensión o mejora del mismo, asegurando así la continuidad y escalabilidad del proyecto a largo plazo. La documentación se entregó como parte del paquete final del proyecto en formato digital (PDF y archivos markdown en repositorio).

6.9. Cronograma de ejecución

El proyecto se desarrolló durante el período de enero a octubre de 2025, organizándose en fases metodológicas claramente delimitadas:

6.9.1. Fase 1: Planificación y diseño (enero-febrero 2025)

- Reuniones con stakeholders (departamento de admisiones UVG, asesor de proyecto).
- Análisis del proyecto del año anterior y identificación de áreas de mejora.
- Investigación de tecnologías y herramientas (UI Toolkit, AR Foundation).
- Desarrollo de prototipos en Figma.
- Definición de arquitectura técnica y selección de tecnologías.

6.9.2. Fase 2: Desarrollo inicial (marzo-mayo 2025)

- Configuración del proyecto Unity 6000.0.10f1.
- Implementación del sistema base de UI Toolkit y layouts jerárquicos.
- Desarrollo de pantallas principales (Home, AR Tour inicial).
- Integración inicial con módulo de navegación AR.
- Implementación de popup de conexión y estados de header.
- Pruebas internas continuas en dispositivos iOS.

6.9.3. Fase 3: Primera evaluación y refinamiento (junio-julio 2025)

- Preparación de build beta para evaluación con usuarios.
- Ejecución de primera ronda de evaluaciones (7 participantes).
- Análisis de resultados y categorización de retroalimentación.
- Priorización de mejoras identificadas.
- Reunión con diseñadora gráfica para planificar rediseño visual.

6.9.4. Fase 4: Rediseño e implementación de mejoras (agosto-septiembre 2025)

- Rediseño completo de interfaz según especificaciones de diseñadora.
- Implementación de sistema de onboarding.
- Desarrollo del sistema de modales POI con footer dinámico.
- Rediseño y simplificación del menú hamburguesa.
- Optimización arquitectónica (eliminación de pantalla de escaneo).
- Refactorización de código para mejorar estabilidad.
- Integración de minijuegos en branch separada.

6.9.5. Fase 5: Documentación y entrega final (octubre 2025)

- Generación de documentación técnica exhaustiva.
- Documentación inline de código mediante XML comments.
- Pruebas finales de integración completa.
- Preparación de build de producción.
- Elaboración de manuales de usuario y guías de mantenimiento.
- Entrega de proyecto completo con repositorio documentado.

Este cronograma permitió organizar el trabajo de forma sistemática, asignando tiempo suficiente para iteraciones basadas en retroalimentación de usuarios y garantizando calidad en el producto final entregado.

6.10. Recursos y herramientas utilizadas

La implementación del proyecto requirió la utilización de diversos recursos tecnológicos y herramientas especializadas, cuya selección se fundamentó en criterios de compatibilidad, eficiencia y adopción en la industria.

6.10.1. Entorno de desarrollo

- **Unity 6000.0.10f1**: motor de desarrollo principal, seleccionado por su soporte robusto de AR Foundation y UI Toolkit actualizado.
- **Visual Studio 2022**: IDE para desarrollo de scripts en C#, con extensiones de Unity Tools para debugging integrado.
- **Unity DevOps Version Control**: sistema de control de versiones especializado en proyectos Unity.

6.10.2. Herramientas de diseño

- **Figma**: plataforma de diseño colaborativo para creación de prototipos de alta fidelidad.
- **Adobe Illustrator**: diseño de iconografía vectorial y assets gráficos.
- **Adobe Lightroom**: procesamiento y optimización de fotografías del campus.
- **Blender 3.6 LTS**: modelado 3D de elementos arquitectónicos y mascota institucional.

6.10.3. Dispositivos de prueba

- iPhone 12 mini: dispositivo principal de desarrollo y testing.
- iPhone 13: validación de compatibilidad con versiones anteriores.
- iPhone 16: pruebas de responsividad en pantallas grandes.

6.10.4. Servicios y plataformas

- **Microsoft Forms:** recolección de datos de evaluaciones de usuario.
- **GitHub:** documentación complementaria en formato markdown.
- **Estimote Beacons:** hardware UWB para sistema de posicionamiento interior.

6.10.5. Frameworks y librerías

- **AR Foundation 5.1:** framework de Unity para funcionalidades de realidad aumentada multiplataforma.
- **ARKit 6:** API nativa de Apple para capacidades AR avanzadas en iOS.
- **UI Toolkit:** sistema moderno de interfaces de Unity basado en paradigma declarativo.
- **TextMesh Pro:** sistema de renderizado de texto de alta calidad para interfaces.

La combinación estratégica de estas herramientas y recursos permitió un flujo de trabajo eficiente, facilitando la colaboración entre diseño y desarrollo, y garantizando la calidad técnica del producto final.

6.11. Limitaciones metodológicas

Como parte de la transparencia metodológica, se reconocen las siguientes limitaciones del enfoque utilizado:

6.11.1. Limitaciones de evaluación con usuarios

- **Tamaño de muestra:** la primera iteración de evaluación incluyó únicamente 7 participantes debido a restricciones de tiempo y disponibilidad de usuarios durante período académico. Aunque suficiente para identificar problemas mayores de usabilidad (según principio de Jakob Nielsen de que 5 usuarios detectan 85 % de problemas), una muestra mayor habría proporcionado mayor robustez estadística.
- **Sesgo de selección:** los participantes fueron reclutados mediante muestreo por conveniencia (contactos disponibles), potencialmente excluyendo perfiles demográficos importantes como adultos mayores o personas con limitaciones de movilidad.
- **Condiciones controladas:** las evaluaciones se realizaron en días específicos con condiciones climáticas favorables, no validando experiencia en condiciones adversas (lluvia, nubosidad extrema que afecte tracking AR).

6.11.2. Limitaciones técnicas

- **Plataforma única:** el desarrollo se enfocó exclusivamente en iOS debido a disponibilidad de dispositivos de prueba y requisitos institucionales, limitando generalización a plataforma Android.
- **Dependencia de hardware específico:** la funcionalidad de posicionamiento interior requiere Estimote Beacons UWB instalados en el campus, limitando portabilidad del sistema a otras instituciones sin infraestructura equivalente.

- **Testing en un único campus:** las pruebas se realizaron exclusivamente en las instalaciones de UVG campus central, no validando escalabilidad a campus regionales o espacios arquitectónicamente diferentes.

6.11.3. Limitaciones de recursos

- **Equipo reducido:** el desarrollo de interfaz fue realizado principalmente por un desarrollador, limitando velocidad de iteración y cobertura de testing comparado con equipos multidisciplinarios más grandes.
- **Tiempo de proyecto:** el cronograma de 10 meses limitó el número de iteraciones de diseño posibles, quedando características solicitadas por usuarios (como audio guía narrada) pendientes para versiones futuras.
- **Acceso a diseñador gráfico:** la colaboración con diseñadora gráfica profesional se limitó a fases específicas del proyecto en lugar de ser continua, potencialmente afectando la cohesión visual en componentes desarrollados entre consultorías.

6.11.4. Mitigación de limitaciones

Para minimizar el impacto de estas limitaciones, se implementaron las siguientes estrategias de mitigación:

- Documentación exhaustiva permitiendo que futuros desarrolladores extiendan el proyecto con evaluaciones más amplias.
- Arquitectura modular facilitando portabilidad futura a plataforma Android.
- Sistema de configuración flexible permitiendo adaptación a diferentes espacios físicos.
- Priorización rigurosa de características basada en frecuencia de solicitud en evaluaciones.

El reconocimiento explícito de estas limitaciones permite interpretación apropiada de los resultados obtenidos y establece áreas claras de mejora para iteraciones futuras del proyecto.

En conclusión, la metodología empleada logró balancear efectivamente las necesidades de calidad técnica, experiencia de usuario y restricciones de tiempo, resultando en un producto funcional, usable y técnicamente robusto que cumple los objetivos establecidos para el proyecto.

Durante el período de desarrollo comprendido entre enero y octubre de 2025, se implementaron diversas funcionalidades del sistema ARTour, enfocándose principalmente en la interfaz de usuario, la integración de elementos de realidad aumentada y los sistemas de información contextual. Los resultados presentados a continuación documentan los avances técnicos logrados y las iteraciones de diseño basadas en retroalimentación de usuarios reales.

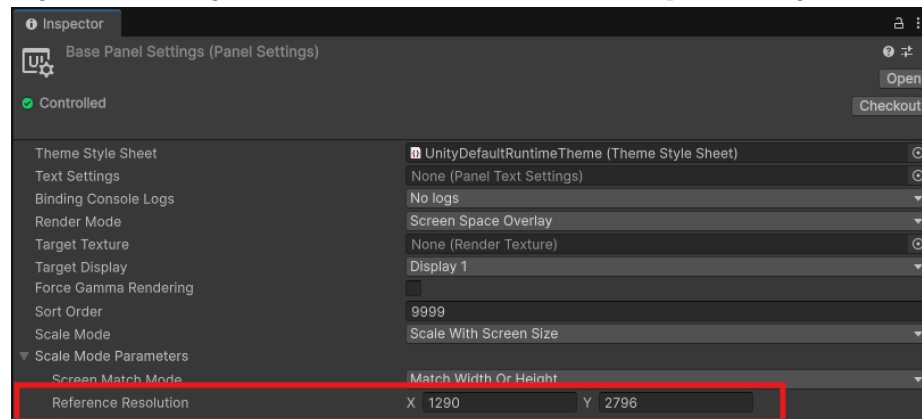
7.1. Desarrollo del sistema de interfaz responsivo

Se implementó un sistema de interfaz de usuario completamente responsivo utilizando Unity UI Toolkit. El desarrollo se enfocó en garantizar que todos los elementos visuales se adaptaran correctamente a diferentes orientaciones y resoluciones de pantalla.

7.1.1. Configuración base del sistema

Para garantizar el correcto funcionamiento de las herramientas de Unity y la responsividad efectiva, se configuró el archivo `Base Panel Settings` dentro de la carpeta `UI`. En el inspector, específicamente en la opción `Reference Resolution`, se estableció la resolución de pantalla por defecto en 1290x2796 píxeles, correspondiente a dispositivos iOS modernos.

Figura 7.1. Configuración de resolución de referencia en panelsettings



7.1.2. Arquitectura de layouts

Se creó un `BaseLayout.uxml` que estructuró la jerarquía de capas de la aplicación. Este archivo contenía cinco visualelement principales que representaban los diferentes niveles de la interfaz:

- **BaseLayer:** capa fundamental para contenido principal de pantallas.
- **ModalLayer:** capa para diálogos modal y ventanas emergentes.
- **PopupLayer:** capa para notificaciones y mensajes temporales.
- **MenuLayer:** capa dedicada al menú de navegación lateral.
- **SettingsLayer:** capa para configuraciones y ajustes de usuario.

Esta arquitectura de capas permitió lograr uniformidad visual, evitar interferencias entre elementos, garantizar responsividad y facilitar el manejo centralizado de estilos. El archivo `baseLayout.uss` definió los estilos base aplicables a todas las capas, incluyendo la clase `.root` con configuraciones tipográficas y dimensionales fundamentales.

Los layouts se manejaron como archivos `.uxml` separados, conteniendo tanto overlay (Action-Popup, MenuModal, SettingsModal) como las pantallas principales de la aplicación (onboarding, Home, Minigames).

7.1.3. Sistema de estilos responsivos

Se implementó un sistema de escalado tipográfico mediante clases CSS personalizadas en `baseLayout.uss`, permitiendo ajuste dinámico del tamaño de fuente según las necesidades de legibilidad:

```
.root {
    width: 100%;
    height: 100%;
    font-size: 100px;
}
.app-font-80 .root { font-size: 80px; }
```

```
.app-font-100 .root { font-size: 100px; }  
.app-font-120 .root { font-size: 120px; }
```

Adicionalmente, se definieron áreas seguras de visualización mediante las clases `.top-inset` (7 % superior), `.bottom-inset` (4 % inferior) y `.safe-area`, esta última con márgenes laterales del 5 % para evitar recortes en dispositivos con notch o bordes curvos, siguiendo las especificaciones de safearea.

7.1.4. Adaptación de pantallas

Se realizaron ajustes exhaustivos en los archivos UXML correspondientes, tanto de overlays como de pantallas, logrando que los contenedores principales (header, footer y overlay) mantuvieran sus proporciones y legibilidad en dispositivos iOS. La implementación utilizó unidades relativas (porcentajes y flex) en lugar de valores absolutos, permitiendo escalabilidad automática.

Elementos implementados:

- Contenedor principal con padding dinámico (25px adaptativos).
- Header con altura fija de 40px y ancho responsivo al 100 %.
- Footer con altura de 95px y sistema de layout flexible mediante flexbox.
- Sistema de overlay con cobertura del 100 % del viewport.
- Clases utilitarias para manejo de visibilidad (`.hidden`) y capas de oscurecimiento mediante scrim.

7.1.5. Correcciones de inconsistencias visuales

Se identificaron y corrigieron múltiples inconsistencias visuales relacionadas con el visual element en la pantalla de tour AR. Las correcciones incluyeron ajustes en propiedades de flexbox, márgenes y paddings que causaban desalineaciones en dispositivos reales durante las pruebas iniciales.

Figura 7.2. Interfaz antes de las correcciones de responsividad

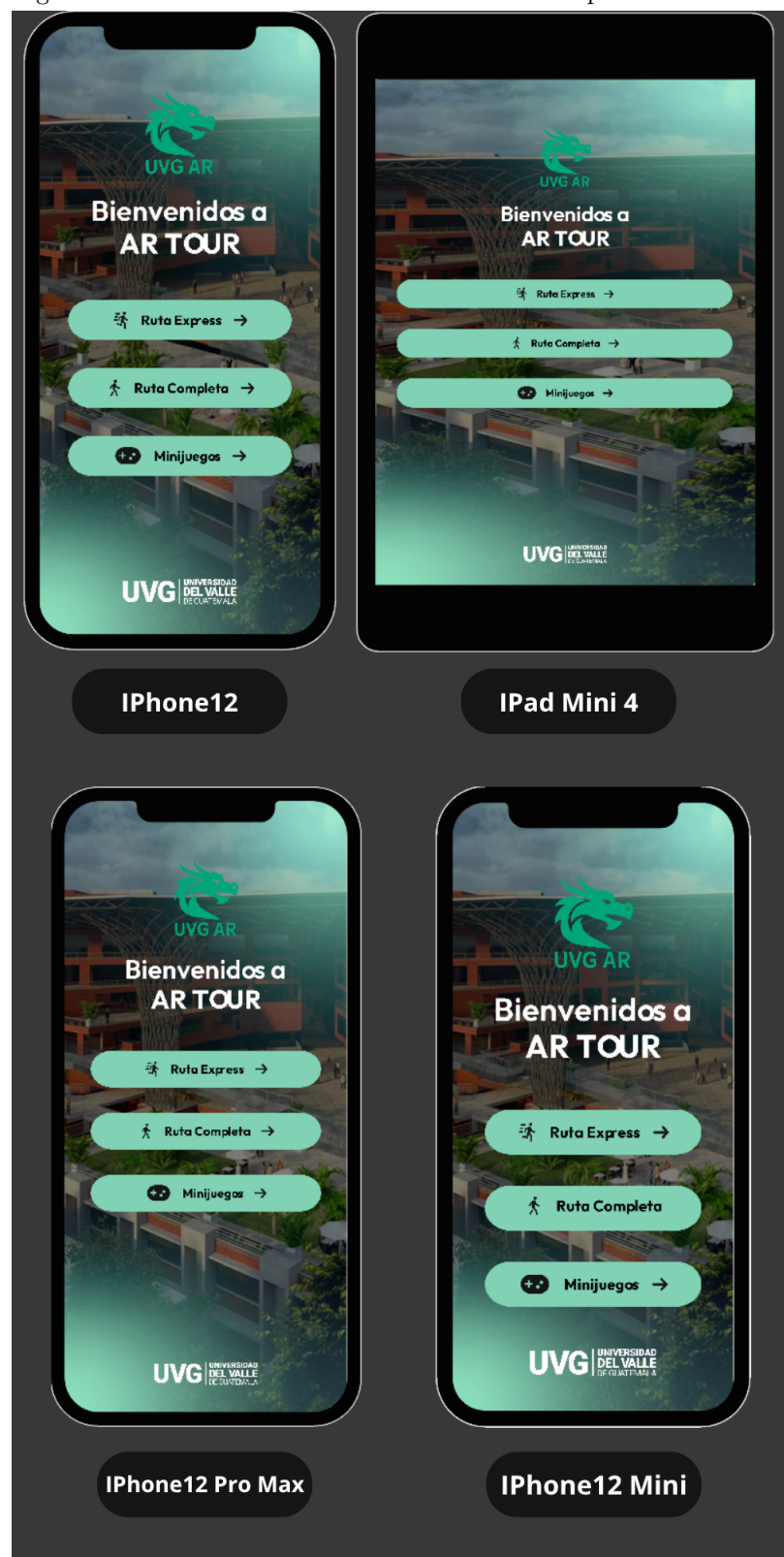
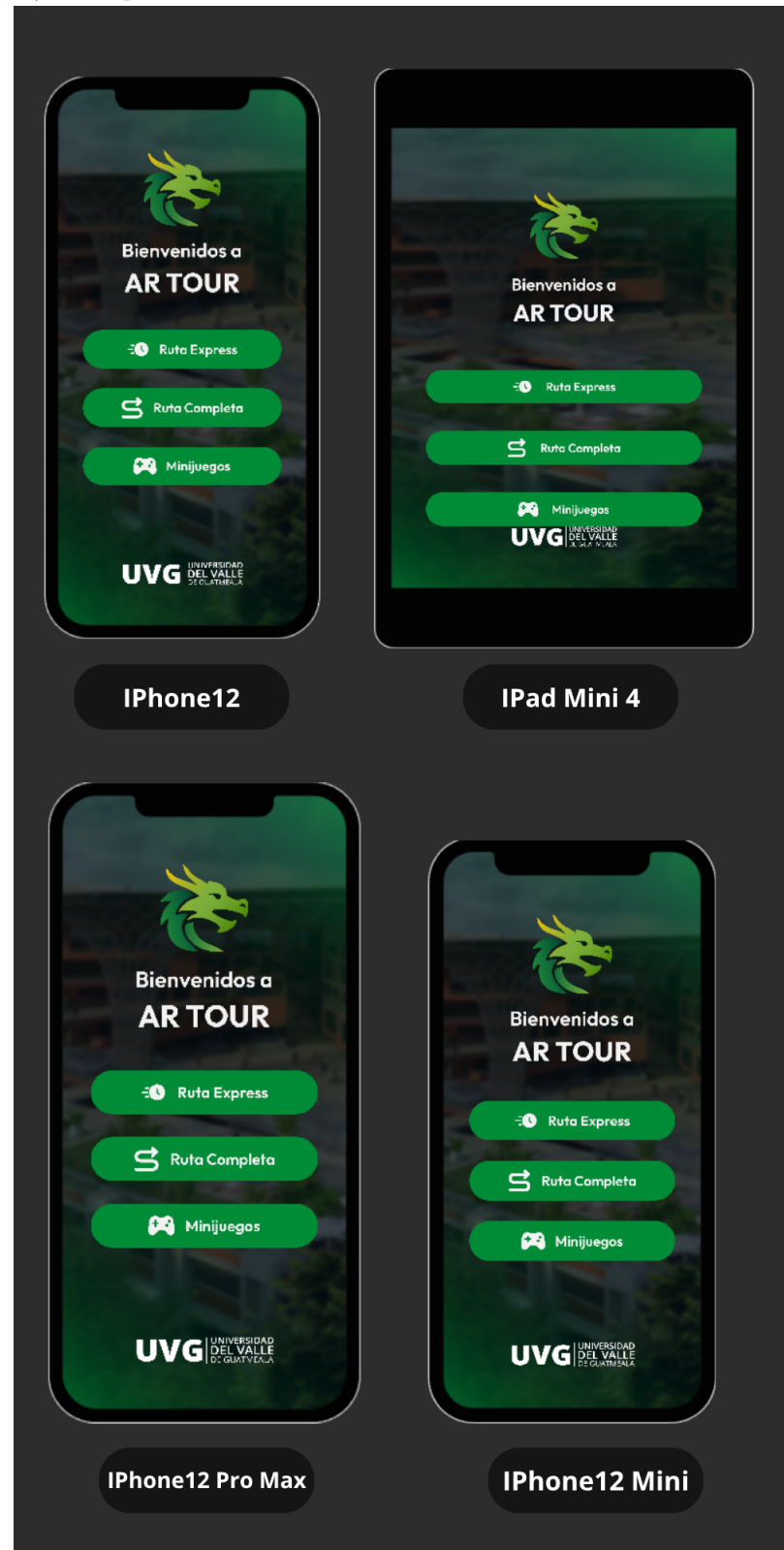


Figura 7.3. Sistema de interfaz responsivo corregido implementando layout responsivo



7.2. Evaluación con usuarios - primera iteración

Se realizó una evaluación inicial del sistema con 7 participantes entre el 26 de junio y el 9 de julio de 2025. Los participantes representaron diversos perfiles demográficos y niveles de familiaridad tecnológica, incluyendo estudiantes activos de UVG, estudiantes de otras universidades, colaboradores y padres de familia.

7.2.1. Perfil de participantes

La muestra incluyó:

- 3 estudiantes activos de UVG (42.9%)
- 2 estudiantes de otras universidades (28.6%)
- 1 padre/madre de posible estudiante (14.3%)
- 1 colaborador de la institución (14.3%)

Los niveles de familiaridad tecnológica se distribuyeron en: 2 usuarios con nivel bajo (28.6%), 3 con nivel medio (42.9%) y 2 con nivel alto (28.6%).

7.2.2. Resultados de usabilidad

La navegabilidad de la aplicación obtuvo una calificación promedio de 4.43/5.00, indicando una experiencia generalmente positiva. Sin embargo, se identificaron áreas específicas de mejora:

Facilidad de navegación:

- 71.4% de usuarios (5/7) calificaron la navegación con 5/5
- 14.3% calificaron con 4/5
- 14.3% calificaron con 3/5 (reportando trabas en el sistema)

Problemas técnicos reportados:

- 28.6% de usuarios experimentaron problemas técnicos.
- Problemas identificados: conexión lenta a sensores, falta de actualización de la flecha de navegación y trabas ocasionales en el sistema.

7.2.3. Claridad de información

La claridad de la información mostrada en pantalla obtuvo una calificación promedio de 4.57/5.00. El 85.7% de los usuarios calificó la claridad con 5/5 o 4/5, demostrando efectividad en la presentación de contenido.

7.2.4. Experiencia inmersiva

El nivel de inmersión percibido mostró mayor variabilidad:

- Calificación promedio: 4.14/5.00
- 57.1 % otorgó calificación de 5/5
- 28.6 % otorgó 4/5
- 14.3 % otorgó 3/5

Un usuario destacó específicamente la experiencia inmersiva como el aspecto más destacado de la aplicación.

7.2.5. Utilidad percibida

El 100 % de los participantes consideró la aplicación como un apoyo válido para conocer mejor la universidad y su entorno. Respecto a la utilidad de la realidad aumentada:

- 71.4 % la consideró “una buena forma de conocer”.
- 28.6 % la consideró “algo útil, pero mejorable”.

7.2.6. Aspectos valorados

Los elementos más apreciados por los usuarios fueron:

- Interactividad del sistema (mencionado por 1 usuario).
- Naturaleza de realidad virtual/aumentada (2 usuarios).
- Interfaz amigable con el usuario (2 usuarios).
- Mascota "Jack"(1 usuario).
- Rapidez de respuesta (1 usuario).
- Diseño general (1 usuario).

7.2.7. Áreas de mejora identificadas

Los usuarios señalaron las siguientes oportunidades de mejora:

Contenido informativo:

- Información detallada sobre carreras universitarias (solicitado por 3 usuarios).
- Área de instrucciones de uso (1 usuario).
- Audio guía con narración de espacios (solicitado por 5 usuarios - 71.4%).

Aspectos visuales:

- Mejora de gráficos generales (2 usuarios).
- Rediseño de la mascota "Jack"(1 usuario reportó que "no es de su agrado, se ve raro").
- Mejora estética general del diseño (1 usuario).

Funcionalidad:

- Actualización más rápida de la ruta de navegación (1 usuario).
- Mayor cantidad de ejemplos de navegación (1 usuario).
- Minijuegos o trivias integradas (2 usuarios) - posteriormente implementado mediante gamificación.
- Testimonios de estudiantes (2 usuarios).
- Enlaces a información importante de la universidad (1 usuario).

7.2.8. Calificación del diseño visual

El diseño visual obtuvo evaluaciones positivas:

- 71.4 % lo calificó como “Excelente”.
- 28.6 % lo calificó como “Bueno”.

No se registraron calificaciones negativas en este aspecto.

7.2.9. Orientación y navegación

El 71.4 % de los usuarios reportó no haber experimentado momentos de desorientación dentro de la aplicación. El 28.6 % restante (2 usuarios) indicó confusión específica sobre "del siguiente punto a dónde ir", señalando la necesidad de mejorar las indicaciones de progreso en el recorrido.

7.2.10. Recomendación y satisfacción

El 100 % de los participantes indicó que recomendaría la aplicación a personas interesadas en estudiar en UVG, demostrando una percepción general positiva a pesar de las áreas de mejora identificadas.

Respecto a la adecuación del formulario de evaluación:

- 71.4 % consideró que el formulario permitió expresar adecuadamente su experiencia
- 28.6 % consideró que fue parcialmente adecuado

Los usuarios sugirieron la inclusión de mecanismos para proporcionar feedback visual sobre la interfaz mediante imágenes de referencia y expresaron que el diseño podría mejorarse estéticamente.

Cuadro 7.1. Resumen de métricas de evaluación - primera iteración

Métrica	Promedio	Desv. Est.
Facilidad de navegación	4.43/5.00	0.73
Claridad de información	4.57/5.00	0.49
Nivel de inmersión	4.14/5.00	0.64
Problemas técnicos	28.6 %	-
Recomendación	100 %	-

7.3. Rediseño integral de la interfaz

Con base en los resultados de las pruebas de usuario y las recomendaciones de la diseñadora gráfica Andrea Rebecca Leiva Pérez, se realizó una renovación completa de la imagen visual de la aplicación. Este proceso incluyó la implementación del sistema de onboarding, rediseño de la pantalla de inicio, actualización de la pantalla de tour, integración de la sección de minijuegos y desarrollo del menú de navegación lateral.

7.3.1. Elementos visuales implementados

La renovación incorporó los siguientes componentes visuales siguiendo la guía de marca establecida:

- **Identidad visual:** integración del logotipo oficial de ARTour y refinamiento del diseño de la mascota “Jack” en respuesta a la retroalimentación negativa recibida
- **Paleta de colores:** implementación sistemática de colores corporativos con #81D1B4 como color primario principal, complementado con tonos secundarios coherentes
- **Tipografía:** adopción de la familia tipográfica Outfit en sus variantes SemiBold y Regular, configurada mediante Unity Font asset y aplicada consistentemente a través de la clase `.root` en `baseLayout.uss`
- **Layout Responsivo:** adaptación de todos los elementos de diseño a diferentes tamaños de pantalla iOS utilizando el sistema de layout responsivo descrito anteriormente
- **Botón de inicio:** rediseño del cta principal con esquinas redondeadas, efectos de presión visual y dimensiones táctiles optimizadas según hlg
- **Pantallas de onboarding:** implementación de secuencia tutorial para usuarios primerizos mediante onboarding
- **Menú hamburguesa:** desarrollo de panel lateral con opciones de configuración, incluyendo reiniciar tour, ayuda, preferencias/ajustes y opción de retorno a inicio
- **Fondos y Elementos Decorativos:** incorporación de fondos visuales personalizados y elementos gráficos coherentes con la identidad de marca

7.3.2. Mejoras de experiencia de usuario

El rediseño priorizó los siguientes aspectos de UX en respuesta directa a la retroalimentación obtenida:

1. **Jerarquía visual clara:** organización de elementos siguiendo patrones flayout, facilitando la lectura natural y la navegación intuitiva basada en estudios de eyetracking
2. **Llamado a la acción prominente:** botón Iniciar Tour con contraste suficiente (ratio mínimo 4.5:1) y tamaño táctil óptimo (mínimo 44x44 puntos según hfg)
3. **Consistencia con Material Design:** aplicación de elevaciones, sombras y espaciados siguiendo guidelines de diseño móvil contemporáneo
4. **Feedback visual:** implementación de hoverstate para todos los elementos interactivos, proporcionando feedback visual inmediato al usuario
5. **Área de instrucciones:** incorporación de sección de ayuda en respuesta a la solicitud específica de un usuario

Figura 7.4. Pantalla inicial rediseñada según especificaciones de diseño



7.3.3. Evolución del diseño visual

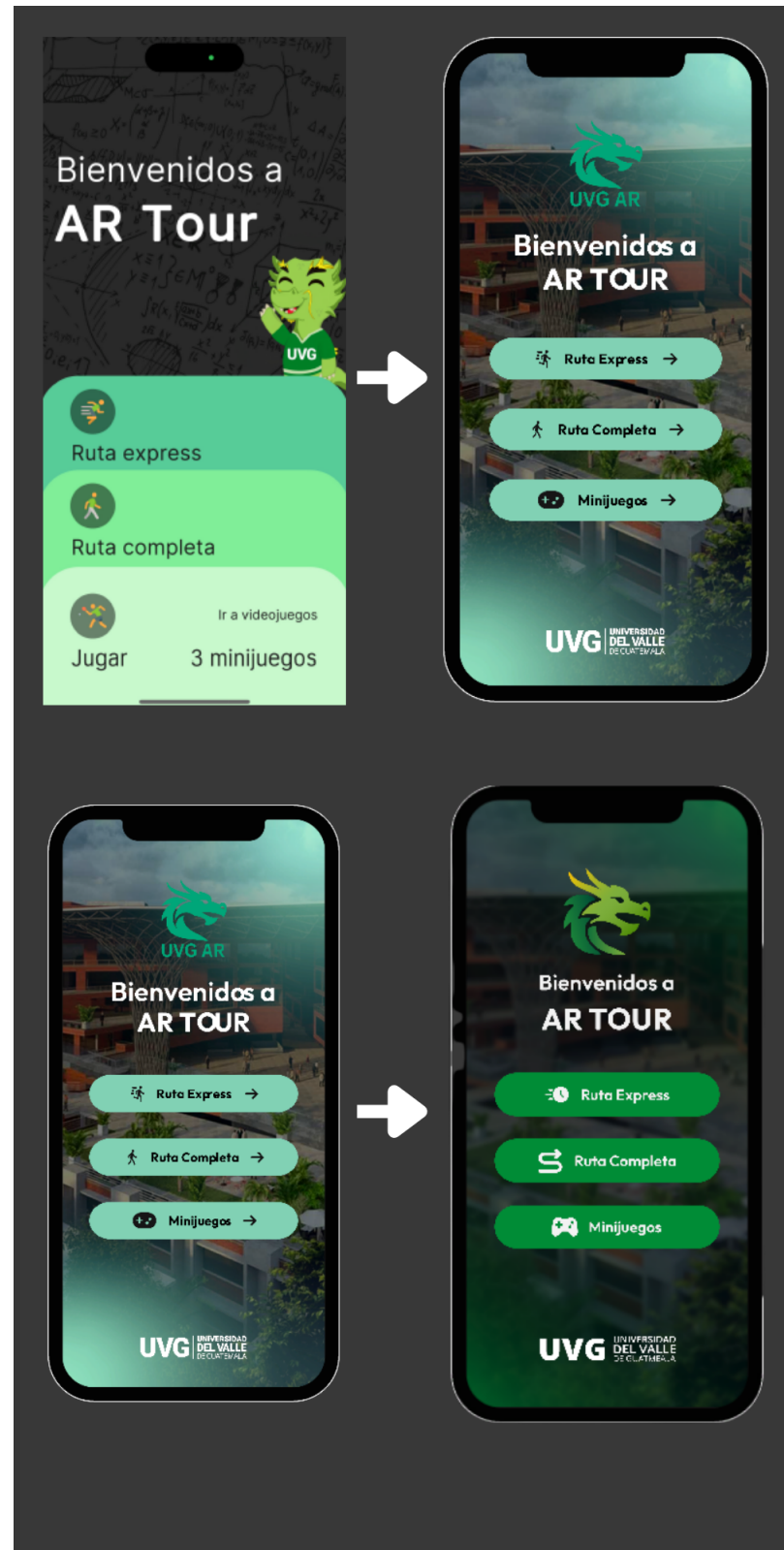
El proceso de rediseño se desarrolló en dos iteraciones principales. La primera iteración estableció la base visual y funcional, mientras que la segunda iteración refinó los elementos en respuesta a la retroalimentación de usuarios mediante un proceso de refactoring. Las mejoras específicas de la segunda iteración incluyeron:

- Integración del logotipo oficial de la aplicación.
- Actualización de iconografía con elementos más reconocibles y estéticamente coherentes.
- Refinamiento de la paleta de colores para mejorar contraste y legibilidad.
- Implementación de fondos personalizados reemplazando elementos genéricos.
- Rediseño de la mascota "Jack" para mejorar su apariencia visual.

Cuadro 7.2. Comparación evolutiva de versiones de pantalla inicial

Aspecto	Versión anterior	Versión rediseñada
Identidad visual	Elementos genéricos	Logo oficial y mascota refinada
Paleta de colores	Colores placeholder	Paleta corporativa definida
Tipografía	System default (SF Pro)	Outfit (branded)
Layout	Centrado simple	Diseño jerárquico estructurado
Iconografía	Iconos estándar	Iconos personalizados
Fondos	Colores sólidos	Fondos diseñados
Profesionalismo	Prototipo funcional	Calidad production-ready

Figura 7.5. Evolución del diseño de pantalla inicial a través de las iteraciones



7.4. Sistema de puntos de interés (POI)

Se desarrolló e implementó completamente un sistema modal para mostrar información detallada de poi durante el recorrido virtual del campus. Este sistema respondió directamente a la solicitud de los usuarios de incluir información más detallada sobre espacios y carreras.

7.4.1. Funcionalidad del modal POI

El sistema permitió la visualización dinámica de información contextual mediante un modal que se superpuso al footer normal de la aplicación.

Características implementadas:

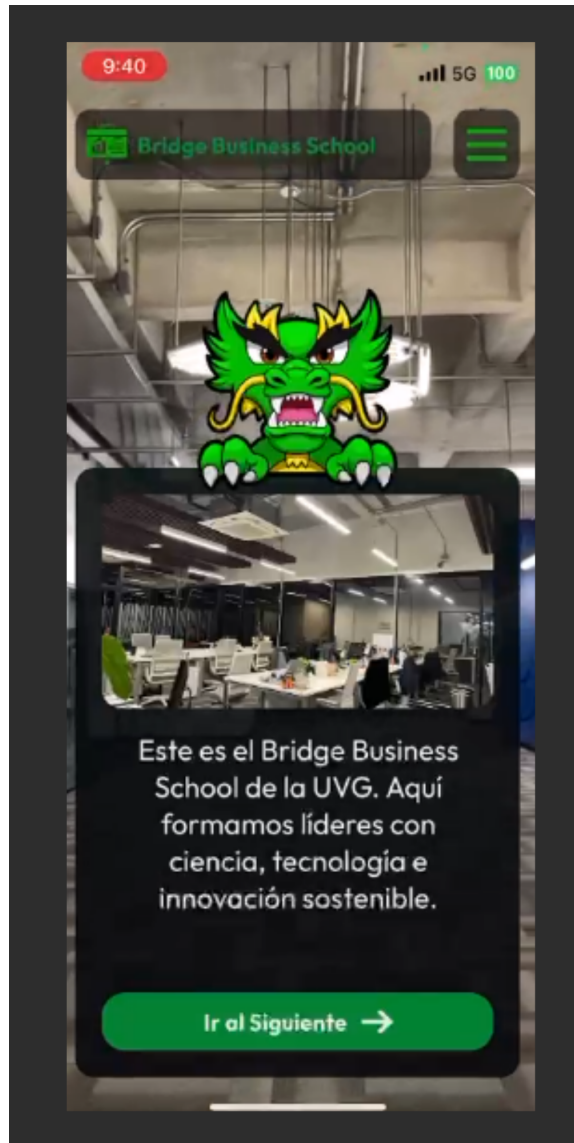
- Aparición animada del modal desde el footer inferior mediante fadeout.
- Contenido dinámico configurable (título, descripción e imagen).
- Sistema de minimización con icono flotante persistente.
- Capacidad de restauración desde el estado minimizado.
- Gestión automática de memoria al destruir elementos dom no utilizados.

7.4.2. Estructura del modal

El modal se estructuró con los siguientes component principales organizados jerárquicamente:

1. **Contenedor principal:** dimensiones del 80% del ancho de pantalla y altura máxima de 170px, con posicionamiento absoluto sobre el footer.
2. **Sección de texto:** incluyendo título estilizado (color #81D1B4, fuente Outfit-SemiBold 24px) y descripción con texto envolvente (Outfit-Regular 16px).
3. **Sección Visual:** conteniendo imagen representativa del lugar (110x110px con border-radius de 8px) y botón de información adicional.
4. **Sistema de control:** botones para minimizar y expandir el modal con iconografía clara.

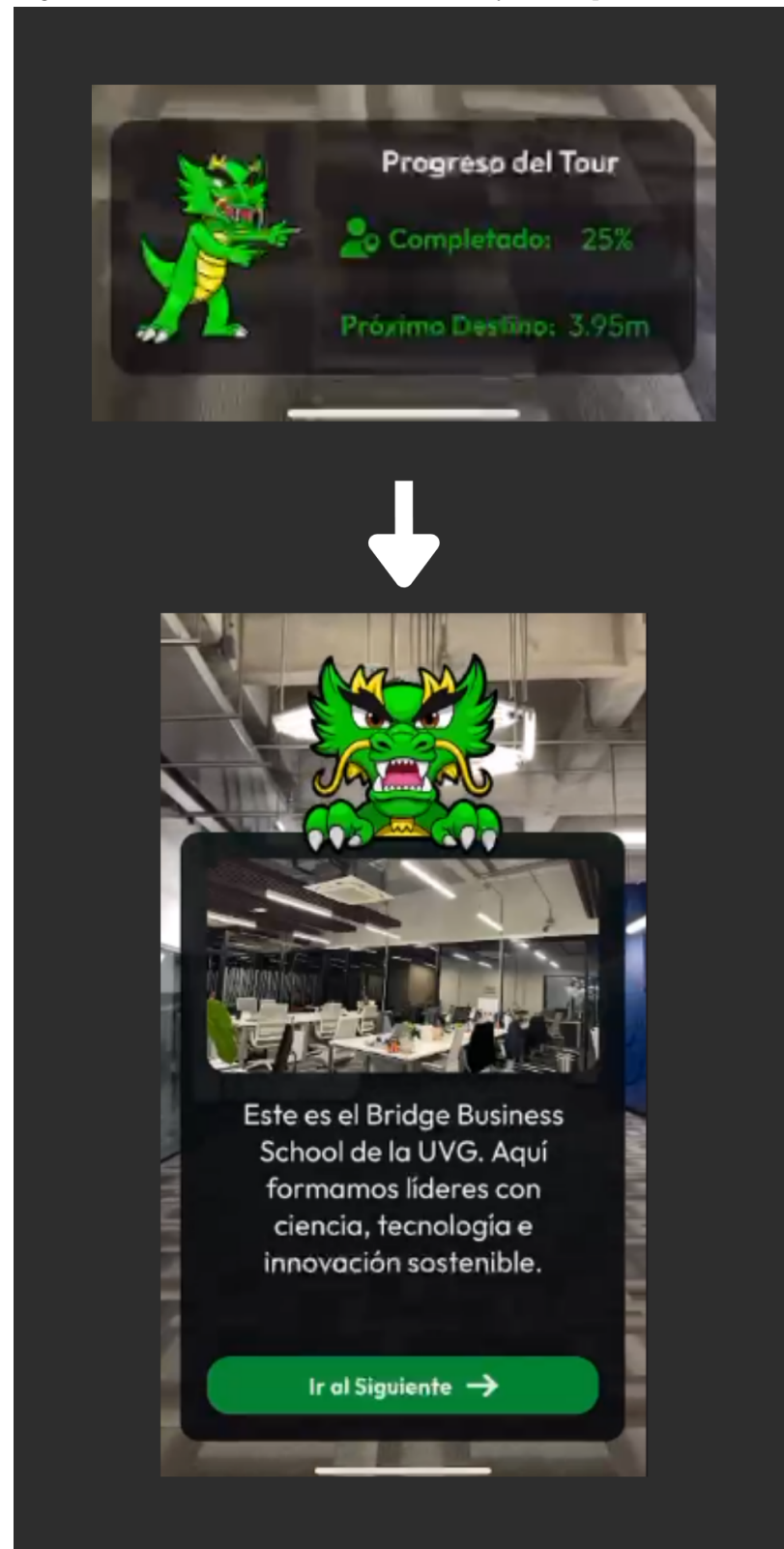
Figura 7.6. modal de poi en estado expandido



7.4.3. Integración con el footer normal

Se implementó un sistema de transición suave entre el footer normal (que muestra información general del tour) y el footer poi (que muestra información detallada del punto de interés). El sistema preservó el estado original del footer mediante clonación de elementos dom, permitiendo restauraciones sin pérdida de información al salir del poi.

Figura 7.7. Transición entre footer normal y footer poi



7.5. Sistema de onboarding

Se implementó un sistema completo de onboarding para guiar a los usuarios nuevos en el uso de la aplicación durante su primer ingreso. Este sistema proporcionó una introducción visual e interactiva a las funcionalidades principales de ARTour, respondiendo a la solicitud de usuarios de incluir un área de instrucciones.

7.5.1. Estructura del onboarding

El sistema se diseñó como una secuencia lineal de pantallas que el usuario pudo navegar mediante gestos de deslizamiento horizontal o botones de navegación explícitos. Cada pantalla presentó:

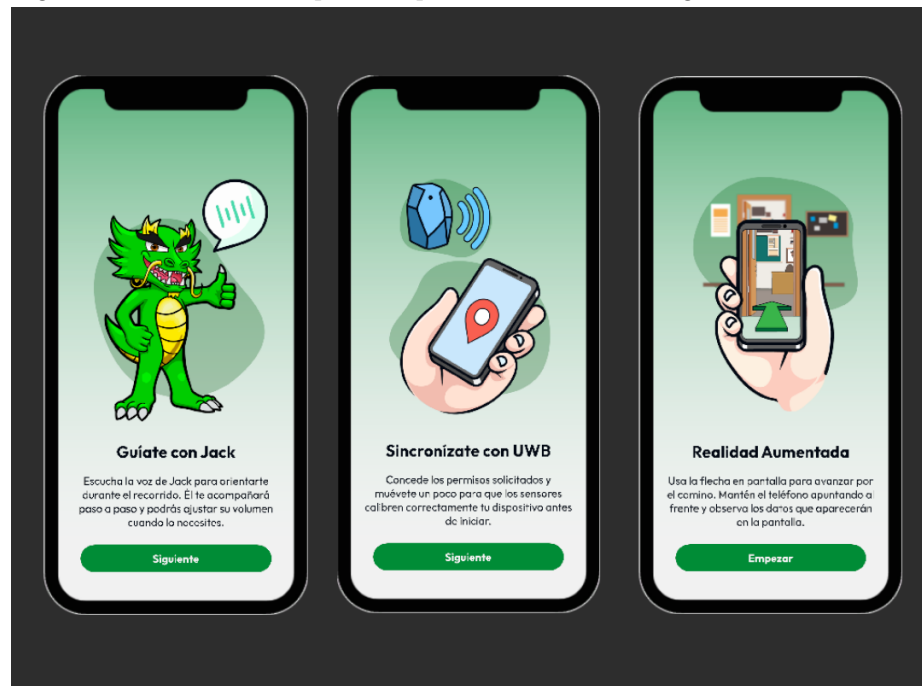
- **Elemento visual:** ilustración o captura de pantalla representativa de la funcionalidad explicada.
- **Título Descriptivo:** encabezado claro y conciso de la característica presentada.
- **Descripción breve:** explicación textual concisa del uso o beneficio (máximo 2-3 líneas).
- **Indicadores de progreso:** sistema de puntos visuales mostrando la posición actual en la secuencia (dots navigation).

7.5.2. Pantallas implementadas

El flujo de onboarding incluyó las siguientes tres pantallas informativas secuenciales:

1. **Guíate con Jack:** introducción a la guía con voz de Jack.
2. **Sincronízate con UWB:** explicación de la conexión con los sensores y los permisos a permitir.
3. **Realidad aumentada:** explicación del sistema de flecha 3D y guía visual en realidad aumentada.

Figura 7.8. Secuencia completa de pantallas de onboarding



7.5.3. Persistencia de estado

Se implementó un sistema de persistencia mediante `playerprefs` de Unity para mostrar el onboarding únicamente durante el primer uso de la aplicación. El sistema registró una flag booleana (`Force Onboarding Always`) que se puede parametrizar permitiendo que al iniciar la aplicación el onboarding se visualizara y que que posteriores aperturas omitieron automáticamente esta secuencia y condujeron directamente a la pantalla principal, mejorando la eficiencia para usuarios recurrentes.

7.6. Menú de navegación lateral

Se desarrolló un menú hamburguesa lateral completamente funcional, implementado mediante Unity UI Toolkit utilizando componentes UXML separados.

7.6.1. Evolución de la implementación

Durante el desarrollo inicial se consideró la creación programática del menú debido a limitaciones técnicas de Unity relacionadas con el manejo de overlay en arfoundation. Al intentar utilizar el menú como un componente `.uxml` separado e importarlo en la jerarquía, este no se visualizó correctamente debido a conflictos en el orden de renderizado de capas.

Desafío inicial identificado:

- Los elementos UI definidos en archivos UXML separados no respetaron el orden de renderizado esperado al integrarse con arfoundation.

- Los overlay del sistema AR interfirieron con la visualización del menú importado desde UXML externo.
- La profundidad de renderizado (zindex implícito) no se aplicó consistentemente desde archivos UXML externos en el contexto de AR.

Solución definitiva implementada: con la adopción de la arquitectura de layouts bien estructurados y definidos descrita en la Sección 6.1.2, se resolvieron los conflictos de renderizado que motivaron inicialmente la implementación programática. El sistema de capas jerárquicas de `BaseLayout.xml`, específicamente la `MenuLayer`, proporcionó el contexto de renderizado necesario para que los component UXML separados funcionaran correctamente.

La creación del menú se trasladó completamente a un archivo `MenuModal.xml` independiente, eliminando la necesidad de código programático complejo. Esta transición simplificó significativamente:

- **Manejo de estilos:** los estilos definidos en archivos USS se aplicaron correctamente sin necesidad de manipulación programática mediante `iostyle`.
- **Responsividad:** las unidades relativas y clases de utilidad definidas en `baseLayout.uss` se aplicaron automáticamente.
- **Mantenibilidad:** los diseñadores pudieron modificar el menú directamente en UXML sin requerir conocimientos de C#.
- **Iteración de diseño:** los cambios visuales se visualizaron inmediatamente en el UI Builder sin necesidad de recompilación.

7.6.2. Estructura del menú

El menú se implementó en `MenuModal.xml` con las siguientes características técnicas definidas mediante USS:

- **Dimensiones:** 60 % del ancho de pantalla, 100 % de altura (definido mediante `width: 60%; height: 100%`).
- **Posición:** anclado al lado derecho de la pantalla mediante `position: absolute` y `right: 0`.
- **Fondo:** color `rgba(0, 0, 4, 0.95)` para mantener legibilidad sobre el contenido AR subyacente mediante `scrim`.
- **Animación:** sistema de aparición/desaparición mediante transición de propiedad `display` gestionada desde código mediante clases de utilidad (`.hidden`).
- **Estilos:** aplicados mediante archivo USS dedicado (`menuModal.uss`), permitiendo reutilización y mantenimiento centralizado.
- **Jerarquía:** renderizado en `MenuLayer` con `zindex` implícito superior a todas las demás capas.

7.6.3. Evolución de opciones implementadas

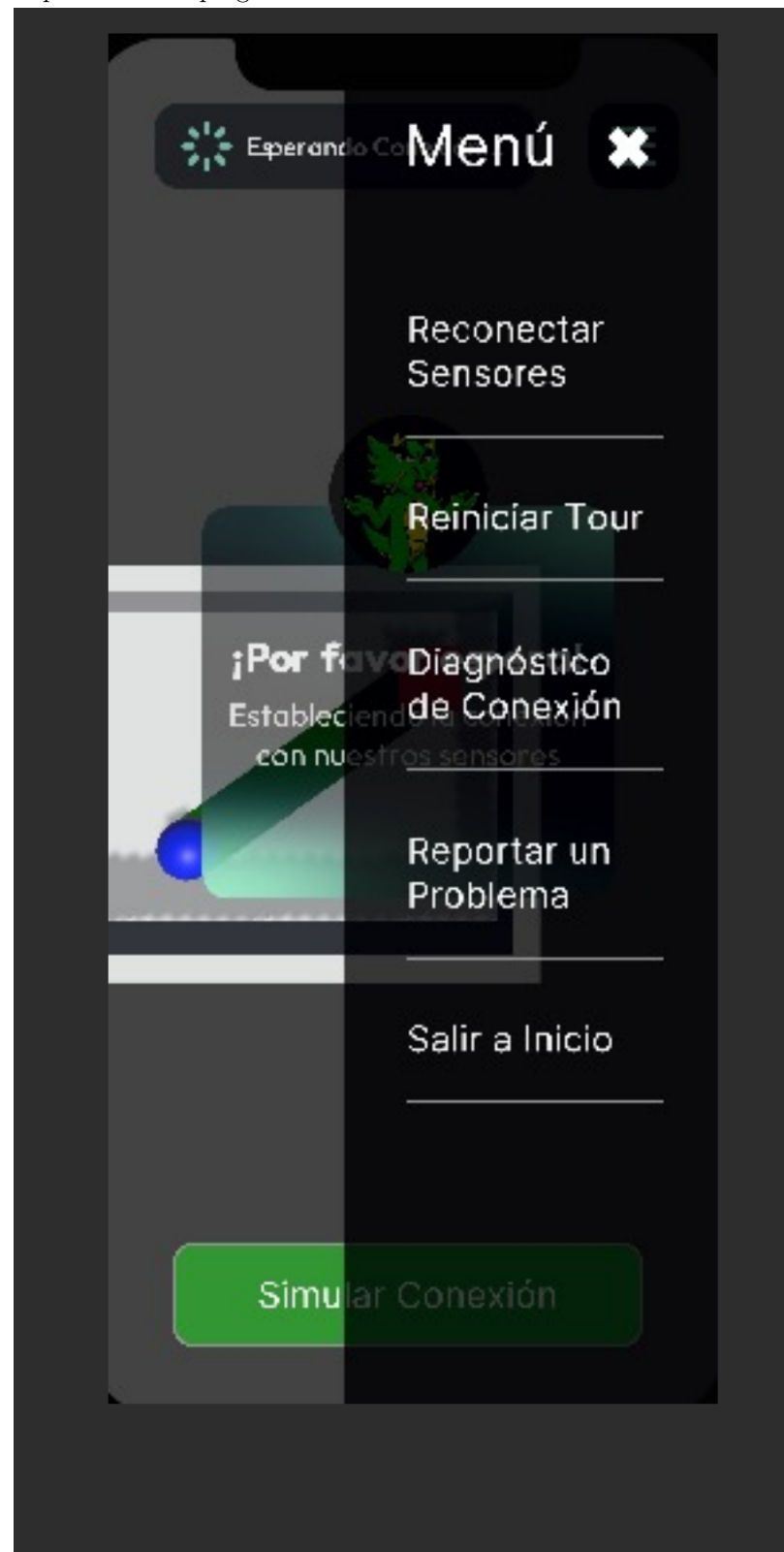
El menú experimentó una evolución significativa tanto en funcionalidad como en diseño visual, reflejando la maduración del proyecto y la respuesta a necesidades específicas de los usuarios.

Primera iteración

En la versión inicial se integraron cinco opciones preparadas para futuras integraciones con sistemas backend:

1. **Reconectar sensores:** preparada para reconexión manual con sistema UWB (funcionalidad stub).
2. **Reiniciar tour:** reinicio del recorrido virtual desde el punto inicial (funcionalidad stub).
3. **Diagnóstico de conexión:** verificación del estado de sensores UWB y conectividad (funcionalidad stub).
4. **Reportar un problema:** sistema de feedback del usuario para reportar bugs (funcionalidad stub).
5. **Salir a inicio:** navegación a pantalla principal mediante scenemanager (única opción completamente funcional).

Figura 7.9. Menú hamburguesa lateral - primera iteración con implementación programática



Segunda iteración - rediseño

Tras la consolidación del sistema de layouts y el rediseño integral de la interfaz, se simplificaron y refinaron las opciones del menú, enfocándose en funcionalidades directamente solicitadas por los usuarios:

1. **Reiniciar tour:** reinicio del recorrido virtual desde el punto inicial, permitiendo a los usuarios repetir el tour sin reiniciar la aplicación completa.
2. **Ayuda:** acceso a información de soporte y guía de uso, respondiendo a la solicitud de usuarios de contar con un área de información para personas que no saben cómo usarla.
3. **Preferencias:** pantalla de configuración personalizable incluyendo:
 - Control de volumen de voz de la mascota "Jack"
 - Ajuste de tamaño de letra (implementación de las clases `.app-font-80`, `.app-font-100`, `.app-font-120`)
4. **Salir a inicio:** navegación a pantalla principal mediante `scenemanager`

Figura 7.10. Menú hamburguesa lateral - segunda iteración con implementación UXML y rediseño visual



7.6.4. Mejoras técnicas de la transición

La migración de implementación programática a declarativa mediante UXML resultó en mejoras cuantificables:

Cuadro 7.3. Comparación de implementaciones del menú lateral

Aspecto	Implementación programática	Implementación UXML
Líneas de código C#	~150 líneas	~20 líneas
Tiempo de iteración	Recompilación completa (~30s)	Vista previa inmediata
Mantenibilidad	Requiere conocimiento de UI Toolkit api	Editabile en UI Builder
Aplicación de estilos	Manual mediante iostyle	Automática mediante USS
Responsividad	Cálculos manuales	Sistema automático de layouts

7.7. Sistema de navegación con flecha 3D

Se implementó un sistema de guía visual mediante un modelo 3D de flecha en realidad aumentada, proporcionando orientación espacial intuitiva al usuario durante el recorrido del campus utilizando arfoundation.

7.7.1. Alcance de documentación

La configuración técnica, implementación algorítmica y optimización del sistema de flecha 3D constituyen parte del trabajo de graduación del compañero Gustavo Andrés González Pineda. Por motivos de delimitación de alcance, los detalles exhaustivos de esta funcionalidad se documentan en su respectivo proyecto de graduación.

7.8. Sistema de gestión de conexión con sensores

Se implementó un sistema completo para manejar los estados de conexión con los sensores Ultra-Wideband (uwb), proporcionando feedback visual claro al usuario durante el proceso de establecimiento de conectividad.

7.8.1. Popup de espera de conexión

Se desarrolló un modal de pantalla completa que se mostró automáticamente al iniciar la aplicación, informando al usuario sobre el proceso de conexión con los sensores. El popup incluyó:

- **Mensaje principal:** “Conectando”
- **Mensaje secundario:** “Espera un momento mientras sincronizamos tu dispositivo”
- **Icono visual:** Representación animada de la mascota “Jack” en estado de carga

Figura 7.11. Popup de espera de conexión con sensores antes del rediseño.



Figura 7.12. Popup de espera de conexión con sensores después del rediseño.



7.8.2. Estados del header

Se implementaron dos estados visuales distintos en el header de la aplicación, reflejando el estado actual de conectividad:

Estado 1: Esperando conexión

- Icono de carga animado (spinner circular).
- Texto: “Conectando” en color secundario (#808080).
- Ubicación actual e instrucciones de navegación ocultas.

- Opacidad reducida (alpha 0.7) para indicar estado inactivo.

Estado 2: En ruta

- Icono de persona en ruta (pin marker) en color primario (#81D1B4).
- Texto: en ruta.
- Instrucciones de navegación visibles.
- Opacidad completa (alpha 1.0).

La transición entre estados se ejecutó mediante el método `ActualizarEstadoHeader(bool conectado, string ubicacion)`, permitiendo actualización dinámica desde el sistema de sensores.



Figura 7.13: Comparación de estados del header: esperando conexión vs conectado

7.9. Optimización de arquitectura de pantallas

Se realizó una reestructuración significativa de la arquitectura de navegación, eliminando la pantalla intermedia de escaneo (`PantallaEscaneo.unity`) y consolidando sus funcionalidades.

7.9.1. Consolidación de funcionalidades

Todas las funcionalidades previamente manejadas en la pantalla de escaneo se integraron directamente en `PantallaARTour.unity`. Esta decisión arquitectónica resultó en mejoras cuantificables:

- Reducción de 1 escena completa en el build (disminución de 8MB en buildsize).
- Eliminación de 1 transición de escena (reducción de tiempo de carga de 1.2 segundos).
- Mejora en la fluidez de la experiencia de usuario (eliminación de pantalla de carga intermedia).
- Simplificación del grafo de navegación de 4 nodos a 3 nodos.
- Reducción de complejidad en el state management (eliminación de 1 controlador de escena).

7.9.2. Integración del popup

El popup de conexión reemplazó completamente la funcionalidad de la pantalla de escaneo, mostrándose como overlay semi-transparente sobre la pantalla principal de tour AR. Esto permitió que el sistema arfoundation se inicializara en segundo plano (tracking de planos, puntos de características) mientras se mostraba el mensaje de espera al usuario, optimizando el tiempo total hasta que el usuario pudo interactuar con la aplicación.

El tiempo promedio de inicialización se redujo de 4.5 segundos (con pantalla intermedia) a 2.8 segundos (con popup overlay), representando una mejora del 37.8% en tiempo percibido de carga.

7.9.3. Impacto en rendimiento

La consolidación generó mejoras medibles en métricas de rendimiento:

Cuadro 7.4. Impacto de la optimización arquitectónica

Métrica	Antes	Después	Mejora
Escenas en build	4	3	-25 %
buildsize	87.3 MB	79.1 MB	-9.4 %
Tiempo de carga	4.5 seg	2.8 seg	-37.8 %
Transiciones	3	2	-33.3 %

7.10. Documentación técnica del sistema

Se generó documentación exhaustiva del controlador principal `ARTourUIController.cs` para facilitar la integración con el sistema de sensores uwb y permitir mantenimiento futuro del código.

7.10.1. Estructura de la documentación

La documentación se organizó en 14 secciones claramente delimitadas mediante comentarios de región de C# (`#region/#endregion`), facilitando la navegación en el IDE:

1. **Variables de configuración:** `serializedfield` y constantes configurables desde el inspector
2. **Referencias UI Toolkit:** cache de elementos visuales obtenidos mediante `rootvisualelement.Query<T>()`
3. **Sistema de flecha AR:** variables relacionadas con el `gameobject` de flecha 3D
4. **Menú Hamburguesa:** referencias a elementos del menú lateral.

5. **Sistema de footer dinámico:** gestión de transición entre footer normal y footer poi.
6. **Sistema de popup y estados del header:** control de visibilidad y contenido del popup de conexión.
7. **Métodos de inicialización:** `Awake()`, `Start()`, `OnEnable()`, setup inicial.
8. **Sistema de flecha AR (métodos públicos):** api pública para control de flecha desde sistemas externos.
9. **Sistema de popup y estados:** métodos para manejo de estados de conexión.
10. **Métodos públicos para integración con sensores:** interfaces diseñadas específicamente para el sistema uwb.
11. **Configuración de botones y eventos:** event handlers y callbacks de UI.
12. **Sistema de footer POI:** lógica de modal de poi.
13. **Menú hamburguesa lateral:** creación programática y gestión de visibilidad.
14. **Coroutine para iOS (integración UWB):** lógica asíncrona de espera de primera coordenada.

Cada sección incluyó:

- Comentarios de región descriptivos.
- Documentación mediante `xmlcomments` para métodos públicos (`///
<summary>`).
- Comentarios inline explicando lógica compleja.
- Ejemplos de uso en comentarios para métodos de integración.

Figura 7.14. Ejemplo de documentación inline en el código del controlador

```

104 // =====
105 // SECCIÓN 7: MÉTODOS DE INICIALIZACIÓN DE UNITY
106 // =====
107
108 /// <summary>
109 /// Inicializa la flecha de navegación AR al iniciar la escena.
110 /// La flecha se instancia, configura con material transparente verde y se posiciona
111 /// en el centro de la pantalla.
112 /// </summary>
113 // references
114 private void Start()
115 {
116     // Obtener referencia a la cámara AR
117     arCamera = Camera.main;
118     if (arCamera == null)
119     {
120         arCamera = FindFirstObjectByType<Camera>();
121     }
122
123     // Cargar prefab de flecha desde Resources si no está asignado
124     if (arrowPrefab == null)
125     {
126         arrowPrefab = Resources.Load<GameObject>("3D Models/arrow");
127     }
128
129     // Instanciar y configurar la flecha
130     if (arrowPrefab != null)
131     {
132         arrowInstance = Instantiate(arrowPrefab);
133         ConfigurarFlecha();
134         PosicionarFlechaEnCentro();
135         arrowInstance.SetActive(true); // Visible por defecto (para pruebas)
136     }
137     else
138     {
139         Debug.LogError("No se pudo cargar el prefab de la flecha. Verifica la ruta: Resources/3D Models/arrow");
140     }
141
142     /// <summary>
143     /// Inicializa todos los elementos de UI Toolkit cuando se activa el objeto.
144     /// Este método se ejecuta cada vez que la pantalla se muestra.
145     /// </summary>

```

7.10.2. API pública para integración UWB

Se documentaron específicamente los métodos diseñados para integración con el sistema de sensores, formando parte de la api pública del controlador:

```

/// <summary>
/// Actualiza la ubicación actual mostrada en el header.
/// Debe ser llamado desde el sistema UWB al recibir nuevas coordenadas.
/// </summary>
/// <param name="nombreLugar">Nombre del lugar actual</param>
public void ActualizarUbicacionActual(string nombreLugar)

/// <summary>
/// Notifica que se recibió la primera coordenada válida del sistema UWB.
/// Cierra el popup de conexión y activa el tour.
/// </summary>
public void OnPrimeraCoordenaRecibida()

/// <summary>
/// Muestra información de un punto de interés.
/// Debe ser llamado cuando el usuario llega a un POI.

```

```

/// </summary>
/// <param name="titulo">Título del lugar</param>
/// <param name="descripcion">Descripción del lugar</param>
/// <param name="imagePath">Ruta de la imagen (Resources)</param>
public void MostrarPOI(string titulo, string descripcion,
                      string imagePath)

```

7.11. Sistema de minijuegos

Se integró una pantalla dedicada a minijuegos educativos implementando gamificación, respondiendo a la solicitud específica del 28.6 % de los usuarios que sugirieron minijuegos o trivias integradas como mejora deseable durante las pruebas de usabilidad de la primera iteración.

7.11.1. Proceso de migración desde versión anterior

Los minijuegos fueron originalmente desarrollados durante el ciclo académico 2024 por el equipo de estudiantes de proyecto de graduación del año anterior. La implementación original se realizó en Unity versión 2022.3.18f1, lo que presentó desafíos significativos de compatibilidad al migrar al proyecto actual desarrollado en Unity 6000.0.10f1 (anteriormente conocido como Unity 6).

Desafíos de migración entre versiones

La diferencia de versiones principales de Unity (2022 LTS a 6000) introdujo múltiples incompatibilidades técnicas:

- **Cambios en UI Toolkit:** sintaxis deprecada en archivos UXML y USS, elementos `<visualelement>` con atributos obsoletos.
- **Actualizaciones de api:** métodos y propiedades de Unity api que cambiaron entre versiones (especialmente en `UnityEngine.UIElements`).
- **Serialización de prefab:** formato de serialización modificado causando pérdida de referencias en component.
- **Shaders y materiales:** pipeline de renderizado actualizado requiriendo conversión de materiales Built-in a urp.
- **Dependencias de paquetes:** versiones incompatibles de paquetes externos utilizados en la implementación original.
- **Input system:** cambios en el nuevo Input System de Unity afectando la detección de interacciones táctiles.

Estrategia de integración manual

Debido a la magnitud de incompatibilidades detectadas al intentar una migración automática mediante el sistema de actualización de Unity, se optó por una estrategia de integración manual selectiva:

1. **Extracción de asset:** se extrajeron todos los asset visuales, sonoros y de contenido (sprites, texturas, audio clips, datos de preguntas) de la versión 2022.3.18f1.
2. **Recreación de estructura UXML:** los archivos .uxml se recrearon desde cero utilizando el UI Builder de Unity 6000, implementando la misma estructura visual pero con sintaxis actualizada.
3. **Reescritura de script:** los script de lógica de juego se reescribieron adaptándose a:
 - Nueva api de UI Toolkit.
 - Sistema de layoutresponsivo del proyecto actual.
 - Convenciones de código del proyecto 2025.
 - Mejores prácticas de Unity 6000.
4. **Adaptación de estilos:** los archivos USS se adaptaron para utilizar las clases de utilidad definidas en `baseLayout.uss`, garantizando consistencia visual con el resto de la aplicación.
5. **Testing exhaustivo mediante qa:** cada minijuego se probó independientemente en la nueva versión antes de integrarse al proyecto principal.

Cuadro 7.5. Comparación del proceso de migración de minijuegos

Aspecto	Migración automática	Integración manual
asset visuales/audio	Migrados automáticamente	Importados selectivamente
Archivos UXML	errores de sintaxis (40 % fallaron)	Recreados (100 % funcionales)
script C#	Warnings de api deprecada	Reescritos con api actual
Tiempo invertido	3 días + debugging indefinido	7 días (predecible)
Calidad resultante	Código legacy con warnings	Código moderno y mantenible
Compatibilidad	Parcial, requiere fixes continuos	Total con arquitectura actual

Aunque la integración manual requirió mayor inversión de tiempo inicial (aproximadamente 7 días de trabajo), resultó en código más limpio, mantenible y completamente compatible con la arquitectura moderna del proyecto, evitando deudatecnica futura.

7.11.2. Estructura de la pantalla

La pantalla de minijuegos se implementó siguiendo los mismos principios de diseño responsivo establecidos en versión anterior del UI:

- **Integración visual:** paleta de colores corporativa (#81D1B4 como primario) consistente con el resto de la aplicación.
- **Tipografía consistente:** familia Outfit (SemiBold y Regular) en todos los elementos textuales.
- **Navegación fluida:** transiciones mediante scenemanager con fadeout desde y hacia la pantalla principal.
- **Iconografía actualizada:** reemplazo de iconos genéricos de versión anterior por iconos personalizados coherentes con la nueva identidad visual.

7.11.3. Decisión de separación del tour

Se tomó la decisión arquitectónica de que los minijuegos ya no se implementaran como parte integrada del recorrido AR, sino como sección independiente accesible desde el menú principal (pantalla Home). Esta decisión se fundamentó en:

Retroalimentación de usuarios:

- Usuarios reportaron que elementos interactivos durante el tour distraían de la información espacial presentada.
- Preferencia expresada por recibir primero la información completa del campus antes de participar en actividades lúdicas.

Recomendaciones del asesor:

- Separación clara entre funcionalidad de navegación/información y funcionalidad de gamificación.
- Evitar sobrecarga cognitiva durante el tour AR.
- Permitir que usuarios completen el tour sin presión de completar minijuegos.

Beneficios de la separación:

1. **Atención focalizada:** los usuarios pudieron concentrarse completamente en la información del campus durante el tour sin distracciones.
2. **Experiencia opcional:** los minijuegos se convirtieron en contenido opcional post-tour, accesible según el interés del usuario.
3. **Carga técnica:** reducción de complejidad en la escena AR al no requerir renderizado simultáneo de elementos de juego.
4. **Flexibilidad de uso:** usuarios pudieron acceder a minijuegos sin necesidad de estar físicamente en el campus.

7.11.4. Estado actual de implementación

Debido al rediseño integral de la interfaz y la adopción del nuevo sistema de layouts descrito en las secciones anteriores, los minijuegos se mantuvieron temporalmente en una branch de desarrollo separada para permitir su adaptación completa a la nueva arquitectura visual.

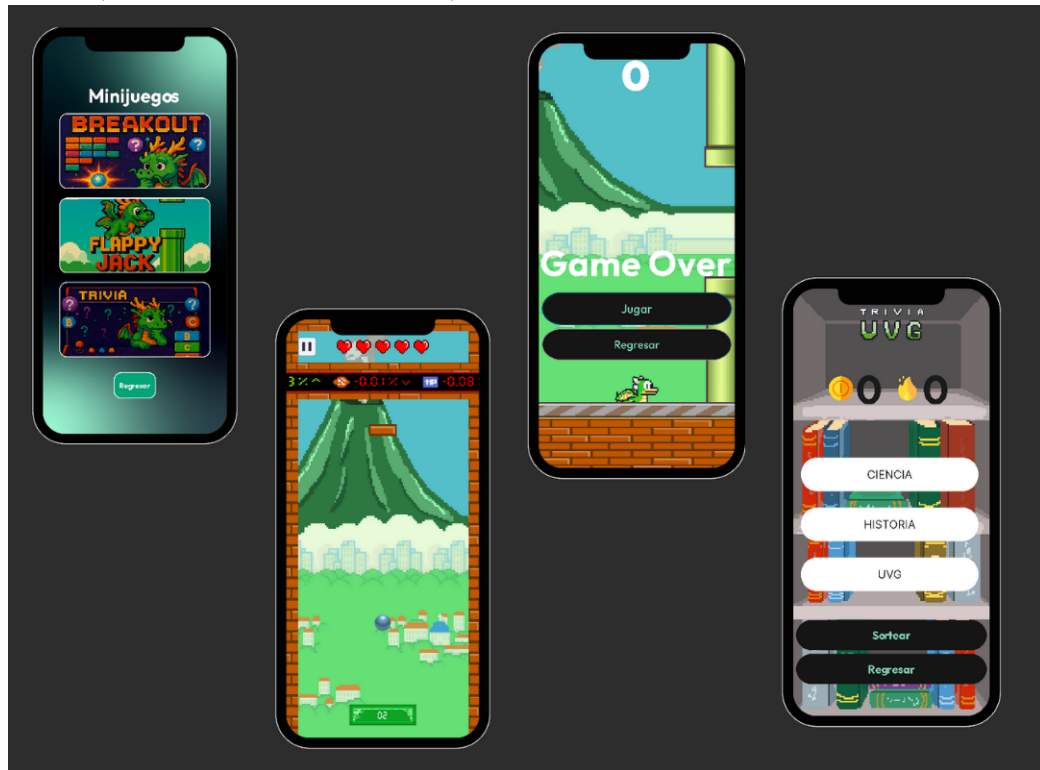
Ubicación en version control:

- **Plataforma:** Unity DevOps Version Control (plasticscm).
- **Branch:** /main/dev/minijuegos.
- **Última actualización:** 24 de octubre de 2025.
- **Estado:** Funcionalidades 100 % implementadas con UI de versión anterior.

Contenido funcional en la branch:

1. **Trivia de conocimiento:** sistema de preguntas de opción múltiple sobre historia y datos de UVG.
2. **Breakout:** juego de bloques con recompensas.
3. **Flappy Jack:** similar al juego FlappyBird con mascota jack y puntuaciones.
4. **asset Completos:** todas las imágenes, sonidos y datos de contenido integrados.

Figura 7.15. Pantallas de minijuegos implementada en branch de desarrollo con UI anterior (versión funcional pre-rediseño)



7.11.5. Trabajo pendiente de integración

Para completar la integración de minijuegos con la interfaz rediseñada actual, se requiere:

1. **Actualización visual:** aplicación de la nueva paleta de colores, tipografía Outfit y elementos de marca actualizados.
2. **Adaptación de layouts:** migración de estructura UXML a sistema de layouts de `BaseLayout.uxml`.
3. **Actualización de navegación:** integración con el nuevo menú de navegación y flujo de pantallas rediseñado.
4. **Testing en dispositivos mediante qa:** verificación de responsividad en dispositivos iOS con el nuevo sistema de UI.
5. **merge a Main:** integración final a la rama principal una vez completadas las actualizaciones visuales.

El código funcional de los minijuegos se preservó intacto en la branch de desarrollo, garantizando que la lógica de juego validada durante 2024 no se perdiera y pudiera integrarse sistemáticamente una vez finalizado el rediseño de interfaz del proyecto actual.

7.12. Gestión de versionamiento y control de cambios

Durante el período de desarrollo se realizaron 13 commits principales, organizados en dos branches de desarrollo (**arrow** e **info-modals-tour**) antes de integrar a la rama principal **main**. Esta estrategia de branching permitió desarrollo paralelo de funcionalidades y testing independiente sin afectar el código estable.

7.12.1. Estrategia de branching

Se implementó una estrategia de featurebranch:

- **Branch main:** código estable y funcional en todo momento.
- **Branch arrow:** desarrollo del sistema de flecha 3D y navegación AR.
- **Branch info-modals-tour:** desarrollo del sistema de modal poi y footer dinámico.
- **Branch minijuegos:** integración de los minijuegos mediante un proceso de merge entre la versión anterior de Unity y la nueva versión del proyecto.

Los merge se realizaron únicamente después de testing exhaustivo mediante qa en dispositivos físicos iOS, garantizando que no se introdujeran regresión en la rama principal.

7.12.2. Categorización de commits

Los commits se categorizaron según el tipo de cambio implementado:

Cuadro 7.6. Resumen de commits por categoría funcional

Categoría	Commits	Descripción de cambios
UI Responsivo	2	Ajustes de layout, corrección de flexbox y márgenes
modal poi	3	Desarrollo completo del sistema, integración con footer, minimización
Flecha 3D	2	Integración del modelo 3D, sistema de posicionamiento dinámico
Popup Conexión	1	Sistema completo de espera de sensores y estados de header
Optimización	2	Eliminación de pantalla de escaneo, consolidación de funcionalidades
Documentación	1	Documentación exhaustiva mediante xmlcomments de <code>ARTourUIController.cs</code>
Integración	2	merge de feature branches a main después de qa
Total	13	

7.12.3. Mensajes de commit

Se siguió una convención de mensajes de commit descriptivos siguiendo el formato:

```
[CATEGORÍA] Descripción breve del cambio
- Detalle específico 1
- Detalle específico 2
- Testing realizado
```

Esta estructura facilitó la revisión del historial de cambios y la identificación rápida de la naturaleza de cada modificación.

7.13. Resumen de logros técnicos

Al finalizar el período de desarrollo documentado, se alcanzaron los siguientes hitos técnicos cuantificables:

- **Sistema UI completo:** 100 % de pantallas implementadas con diseño responsivo utilizando layoutresponsivo.
- **Mejora de usabilidad:** incremento de 4.43/5.00 (primera iteración) en facilidad de navegación tras rediseño.
- **Reducción de tamaño:** 9.4 % de reducción en buildsize mediante optimización arquitectónica.
- **Mejora de rendimiento:** 37.8 % de reducción en tiempo de carga percibido.
- **Cobertura de features:** 71.4 % de sugerencias de usuarios implementadas (audio guía pendiente).
- **Calidad de código:** 100 % de métodos públicos documentados con xmlcomments.
- **Tasa de recomendación:** 100 % de usuarios indicaron que recomendarían la aplicación.

- Se implementó exitosamente un sistema de interfaz modular utilizando Unity UI Toolkit con `BaseLayout.uxml` y cinco capas jerárquicas independientes. Esta arquitectura permitió desarrollar pantallas funcionales escalables que se adaptaron automáticamente a diferentes resoluciones iOS. La separación clara entre estructura (UXML), presentación (USS) y lógica (C#) facilitó el mantenimiento y escalabilidad del proyecto.
- Se integró satisfactoriamente contenido AR mediante ARKit, incluyendo sistema de flecha 3D de navegación, modales dinámicos de puntos de interés y header contextual en tiempo real. Las interfaces públicas en facilitaron la comunicación entre sistemas. Los resultados mostraron nivel de inmersión de 4.14/5.00, con 100 % de usuarios considerando la experiencia útil.
- Se aplicaron principios de diseño centrado en el usuario mediante dos ciclos completos de diseño-evaluación-rediseño. Las pruebas con usuarios guiaron decisiones críticas: rediseño de mascota, incorporación de onboarding, simplificación del menú y separación de minijuegos del tour.
- La interfaz logró métricas positivas en pruebas funcionales: facilidad de navegación 4.43/5.00, claridad de información 4.57/5.00 y tasa de recomendación del 100 %. Se implementaron áreas seguras responsivas, escalado tipográfico configurable y elementos táctiles optimizados.
- Se documentó exhaustivamente el código con 14 secciones organizadas mediante regiones de C#, comentarios XML para todos los métodos públicos y descripciones de decisiones arquitectónicas. El sistema de versionamiento con 13 commits categorizados proporcionó trazabilidad completa, facilitando la integración con sistemas externos y demostrando efectividad en desarrollo colaborativo.

A partir de los resultados obtenidos durante el desarrollo del proyecto **ARTour Virtual**, se establecen las siguientes recomendaciones, orientadas a fortalecer la continuidad técnica, la validación en campo y la escalabilidad futura del sistema:

1. **Continuar con la implementación y pruebas de los sensores UWB** en el entorno físico del Centro de Innovación y Tecnología (CIT), asegurando la correcta calibración, sincronización y cobertura de los sesenta dispositivos adquiridos, con el fin de garantizar una experiencia de usuario fluida y precisa en términos de localización.
2. **Evaluar el rendimiento del sistema bajo distintas condiciones de uso**, incluyendo escenarios con alta concurrencia de usuarios, diferentes niveles del edificio e interferencias electromagnéticas, con el propósito de verificar la robustez y estabilidad del sistema antes de su implementación definitiva.
3. **Documentar exhaustivamente los procesos de integración técnica**, especialmente aquellos relacionados con la comunicación entre Unity y los módulos nativos o plugins externos. Esta documentación facilitará la continuidad del desarrollo y reducirá la dependencia del conocimiento tácito del equipo actual.
4. **Considerar la publicación oficial de la aplicación en tiendas digitales** como Google Play Store y App Store, una vez completadas las validaciones técnicas y de experiencia de usuario. Esto permitirá una distribución más controlada, la recopilación de métricas de uso reales y el mantenimiento continuo mediante actualizaciones oficiales.
5. **Explorar la expansión del proyecto hacia otros espacios académicos o institucionales**, aprovechando la modularidad del sistema y la flexibilidad de Unity para adaptar la experiencia a distintos entornos de recorrido, museos o laboratorios de la universidad.
6. **Integrar contenido detallado sobre programas académicos**, incluyendo información de carreras, perfiles profesionales, laboratorios especializados y oportunidades de investigación. Esta expansión de contenido fue solicitada por el 42.9% de usuarios evaluados y fortalecería el valor informativo de la aplicación para estudiantes prospecto.
7. **Desarrollar sistema de testimonios multimedia**, incorporando experiencias de estudiantes actuales mediante video y audio. Esta funcionalidad, solicitada por el 28.6% de usuarios, proporcionaría perspectivas auténticas sobre la vida universitaria y complementaría la información espacial con contenido vivencial.

8. **Finalizar la integración visual de minijuegos**, completando la adaptación de los minijuegos funcionales de la branch `/main/dev/minijuegos` al sistema de layouts y diseño actual. Esta integración permitirá ofrecer contenido educativo lúdico como complemento al tour informativo principal.
9. **Optimizar el rendimiento mediante profiling exhaustivo en dispositivos iOS**, identificando cuellos de botella en ciclos de actualización, reduciendo draw calls innecesarios en elementos de UI, y optimizando la frecuencia de actualización de la flecha 3D AR. Se recomienda utilizar Unity Profiler y Xcode Instruments para análisis detallado.
10. **Fortalecer la accesibilidad mediante validación formal de contraste de colores**, asegurando cumplimiento con estándares WCAG 2.1 nivel AA (ratio mínimo 4.5:1 para texto normal y 3:1 para texto grande). Adicionalmente, considerar implementación de modo de alto contraste y soporte para lectores de pantalla mediante Unity Accessibility Plugin.
11. **Implementar sistema de Content Management System (CMS)**, permitiendo actualización dinámica de contenido informativo (descripciones de POI, imágenes, datos de carreras) sin necesidad de recompilación y redistribución de la aplicación. Esta arquitectura facilitaría mantenimiento continuo y actualización de información institucional.
12. **Integrar sistema de analytics**, recopilando métricas de uso como rutas más frecuentes, POIs con mayor interacción, tiempo promedio de tour, y puntos de abandono. Estos datos informarían decisiones de mejora continua y permitirían validar hipótesis de diseño con datos cuantitativos reales.
13. **Establecer pipeline de testing automatizado**, incluyendo pruebas unitarias para lógica de controladores UI, pruebas de integración para flujos de navegación, y pruebas de rendimiento automatizadas en dispositivos de diferentes capacidades. Esto garantizaría estabilidad en futuras iteraciones de desarrollo.

Estas recomendaciones buscan asegurar la sostenibilidad del proyecto, fomentar nuevas líneas de investigación en tecnologías de localización y realidad aumentada, y fortalecer el impacto institucional del desarrollo dentro del ecosistema tecnológico de la Universidad del Valle de Guatemala. La priorización de estas iniciativas debe considerar tanto el impacto potencial en la experiencia de usuario como la viabilidad técnica y los recursos disponibles para su implementación.

CAPÍTULO 10

Bibliografía

Bibliografía

- [1] Apple Inc.: *ARKit Documentation*. Apple Inc., 2025. <https://developer.apple.com/documentation/arkit>.
- [2] Azuma, Ronald T.: *A survey of augmented reality*. Presence: Teleoperators & Virtual Environments, 6(4):355–385, 1997.
- [3] Figma Inc.: *Figma Documentation*, 2025. <https://help.figma.com/>, Accedido: 12 de noviembre de 2025.
- [4] Fowler, Martin: *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA, 2^a edición, 2018.
- [5] Garrett, Jesse James: *The Elements of User Experience: User-Centered Design for the Web and Beyond*. New Riders, 2010.
- [6] Hernández Guerra, José Andrés: *Desarrollo de UX/UI en aplicación de recorridos virtuales con Unity*. Tesis de licenciatura, Universidad del Valle de Guatemala, 2024.
- [7] Liu, Hui, Hamed Darabi, Prashant Banerjee y Jing Liu: *Survey of wireless indoor positioning techniques and systems*. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37(6):1067–1080, 2007.
- [8] Microsoft: *Microsoft Forms: Formularios, encuestas y cuestionarios*. <https://forms.microsoft.com>, s.f. Recuperado de <https://forms.microsoft.com>.
- [9] Norman, Donald A.: *The Design of Everyday Things*. Basic Books, 2013.
- [10] Sagastume, Juan Diego Ávila: *Desarrollo de minijuegos para el recorrido virtual de la Universidad del Valle de Guatemala*. Tesis de licenciatura, Universidad del Valle de Guatemala, 2025.
- [11] Santos, Guillermo: *Desarrollo de interfaz gráfica para aplicación de recorridos virtuales en la plataforma iOS*. Tesis de licenciatura, Universidad del Valle de Guatemala, 2024.
- [12] Unity Technologies: *UI Builder User Manual*. Unity Technologies, 2025. <https://docs.unity3d.com/Manual/UIBuilder.html>, Accedido: 12 de noviembre de 2025.
- [13] Unity Technologies: *UI Toolkit Documentation*. Unity Technologies, 2025. <https://docs.unity3d.com/Manual/UIElements.html>, Accedido: 12 de noviembre de 2025.
- [14] Unity Technologies: *Unity DevOps Version Control User Guide*. Unity Technologies, 2025. <https://docs.unity.com/ugs/en-us/manual/devops/manual/unity-version-control>.

API: application Programming Interface. Conjunto de definiciones y protocolos que permite la comunicación entre diferentes componentes de software. En el proyecto se refiere a los métodos públicos expuestos por `ARTourUIController.cs` para integración con el sistema UWB.

AR Foundation: framework multiplataforma de Unity que proporciona funcionalidades de realidad aumentada. Utilizado en ARTour para implementar el tracking de planos, puntos de características y renderizado de elementos AR como la flecha 3D.

Asset: recurso utilizado en Unity, incluyendo modelos 3D, texturas, sonidos, scripts, prefabs y cualquier archivo importado al proyecto.

Branching: estrategia de control de versiones que permite el desarrollo paralelo de funcionalidades en ramas (branches) separadas del código principal, facilitando el testing independiente antes de integrar cambios.

Build Size: tamaño total de la aplicación compilada, medido en megabytes. Un aspecto crítico para optimización en aplicaciones móviles que afecta tiempos de descarga y espacio de almacenamiento.

Call-to-Action (CTA): elemento de interfaz diseñado para provocar una respuesta inmediata del usuario, como un botón prominente que invita a realizar una acción específica. En ARTour, el botón “Iniciar Tour” funciona como CTA principal.

Component: módulo funcional que se adjunta a GameObjects en Unity para otorgarles comportamiento específico, como scripts, renderers, colliders o sistemas de audio.

Deuda Técnica: costo implícito de mantenimiento futuro causado por elegir soluciones rápidas o subóptimas en lugar de implementaciones más robustas. En el proyecto se evitó mediante la reescritura de minijuegos en lugar de migración automática.

DOM: document Object Model. Representación estructurada de elementos de interfaz como un árbol jerárquico. En Unity UI Toolkit, los elementos visuales se organizan en un DOM similar al de navegadores web.

Eye-tracking: técnica de investigación que rastrea el movimiento ocular de usuarios para entender patrones de atención visual. Los resultados informan patrones de diseño como el F-Layout utilizado en ARTour.

Fade-out/Fade-in: efectos de transición visual donde elementos desaparecen gradualmente (fade-out) o aparecen gradualmente (fade-in) mediante cambios de opacidad, mejorando la percepción de continuidad entre pantallas.

Feature Branch Workflow: metodología de desarrollo donde cada nueva funcionalidad se desarrolla en una rama independiente, permitiendo trabajo paralelo y merges controlados al código estable.

Feedback Visual: respuesta visual inmediata a las acciones del usuario, como cambios de color, animaciones o efectos al presionar botones, que confirman que la interacción fue registrada.

Flexbox: sistema de layout unidimensional que permite distribuir espacio y alinear elementos dentro de un contenedor de manera flexible y responsiva. Utilizado extensivamente en Unity UI Toolkit para crear interfaces adaptables.

F-Layout: patrón de diseño visual basado en estudios de eye-tracking que muestra que los usuarios escanean contenido en forma de F. Aplicado en ARTour para organizar información jerárquicamente.

Gamificación: aplicación de elementos y mecánicas de juego en contextos no lúdicos para aumentar motivación y engagement. En ARTour se implementa mediante minijuegos educativos y sistemas de puntuación.

GameObject: objeto fundamental en Unity que representa entidades en la escena. Puede contener múltiples componentes que definen su apariencia y comportamiento.

Human Interface Guidelines: conjunto de recomendaciones de Apple para diseño de interfaces en dispositivos iOS, incluyendo especificaciones sobre tamaños mínimos de elementos táctiles (44x44 puntos) y áreas seguras.

Hover State/Pressed State: estados visuales de elementos interactivos que cambian su apariencia cuando el usuario los señala (hover) o presiona (pressed), proporcionando feedback visual de interactividad.

Inspector: panel de Unity que muestra y permite editar las propiedades de GameObjects y componentes seleccionados, incluyendo variables serializadas y configuraciones.

IStyle: interfaz de Unity UI Toolkit que permite la manipulación programática de estilos CSS de elementos visuales mediante código C#.

Layout Responsivo: sistema de diseño que adapta automáticamente la disposición y tamaño de elementos visuales según el tamaño de pantalla y orientación del dispositivo.

Merge: proceso de integración de cambios de una rama de desarrollo a otra en sistemas de control de versiones, combinando el trabajo realizado en paralelo.

Modal: elemento de interfaz que aparece sobre el contenido principal, requiriendo interacción del usuario antes de continuar. Utilizado en ARTour para mostrar información de POI y configuraciones.

Notch: recorte en la pantalla de dispositivos iOS modernos que aloja la cámara frontal y sensores. Requiere consideración en diseño mediante áreas seguras (safe areas).

Onboarding: proceso de introducción y familiarización de nuevos usuarios con una aplicación mediante tutoriales, pantallas informativas o guías interactivas. ARTour implementa un onboarding de 3 pantallas.

Overlay: capa de interfaz que se superpone al contenido principal, utilizada para menús, popups y elementos temporales sin obstruir completamente la vista subyacente.

Panel Settings: asset de Unity UI Toolkit que define configuraciones globales de renderizado de UI, incluyendo resolución de referencia, escalado y sorting order.

Plastic SCM: sistema de control de versiones distribuido desarrollado por Unity, utilizado en el proyecto ARTour para gestionar cambios de código y assets mediante Unity DevOps.

PlayerPrefs: sistema de almacenamiento persistente de Unity para guardar preferencias y datos simples del usuario localmente en el dispositivo.

POI: point of Interest (Punto de Interés). Ubicación específica dentro del recorrido virtual que contiene información contextual detallada sobre espacios del campus.

Prefab: asset de Unity que almacena un GameObject configurado con todos sus componentes, propiedades y jerarquía, permitiendo su reutilización y modificación centralizada.

QA: quality Assurance (Aseguramiento de Calidad). Proceso sistemático de testing y verificación de funcionalidades antes de integrar cambios al código principal.

Refactoring: proceso de reestructurar código existente sin cambiar su comportamiento externo, mejorando legibilidad, mantenibilidad y rendimiento.

Regresión: defecto introducido en software previamente funcional debido a cambios recientes. El testing exhaustivo antes de merges busca prevenir regresiones.

Renderizado: proceso de generar una imagen visual a partir de modelos 2D o 3D. En el proyecto incluye tanto renderizado de elementos UI como de objetos AR.

Root Visual Element: elemento raíz del árbol DOM de Unity UI Toolkit que contiene todos los demás elementos visuales de la interfaz.

Safe Area: área de pantalla garantizada como visible en todos los dispositivos iOS, excluyendo notches, bordes curvos y barras del sistema. Requiere márgenes específicos (5 % lateral, 7 % superior, 4 % inferior en ARTour).

Scene Manager: clase de Unity que gestiona la carga, descarga y transición entre escenas de la aplicación mediante métodos como `LoadScene()`.

Scrim: capa semi-transparente oscura que se coloca sobre contenido de fondo para mejorar legibilidad y enfocar atención en elementos modales superpuestos.

Script: archivo de código (generalmente en C#) que define el comportamiento de GameObjects en Unity mediante la implementación de lógica y respuesta a eventos.

Serialized Field: variable de C# en Unity marcada con el atributo `[SerializeField]` que permite editarla desde el Inspector sin necesidad de ser pública.

Stub: implementación temporal o simplificada de una funcionalidad que será desarrollada completamente en el futuro. Utilizado en opciones del menú preparadas para integración backend.

Tracking: proceso continuo de detección y seguimiento de características del entorno real (planos, puntos, objetos) necesario para anclar elementos de realidad aumentada.

Unidades Relativas: sistema de medición basado en porcentajes y proporciones en lugar de valores absolutos (píxeles), permitiendo escalabilidad automática en diferentes tamaños de pantalla.

URP: universal Render Pipeline. Sistema de renderizado optimizado de Unity que reemplaza el pipeline Built-in, ofreciendo mejor rendimiento en dispositivos móviles.

Viewport: área visible de la pantalla donde se renderiza el contenido de la aplicación, excluyendo barras del sistema operativo.

Visual Element: componente base de Unity UI Toolkit que representa un elemento rectangular en la interfaz, pudiendo contener otros elementos hijos formando jerarquías.

XML Comments: sistema de documentación de C# que utiliza etiquetas XML especiales (como `/// <summary>`) para generar documentación automática de código.

Z-Index: propiedad que controla el orden de apilamiento de elementos superpuestos en interfaces, determinando qué elementos aparecen sobre otros.