

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA
CONCEPCIÓN - CHILE



**“Desarrollo de pantallas móviles mediante
Inteligencia Artificial: Diseño de prompts como
principal herramienta de programación enfocado en
experiencia e interfaz de usuario”**

BASTIÁN ALONSO LÓPEZ PARRA

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN INFORMÁTICA**

Profesor Guía: José Luis Carrasco Sáez

Junio - 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Desarrollo de pantallas móviles mediante Inteligencia Artificial: Diseño de prompts como principal herramienta de programación enfocado en experiencia e interfaz de usuario

Nombre del candidato(a): Bastían Alonso López Parra

Carrera / Grado: Ingeniería en Informática

Campus: Concepción Departamento: Electrónica e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, JOSÉ LUIS CARRASCO SÁEZ, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☐ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 30-07-26

Firma: 

Estudiante o Candidato(a):

Fecha: 07/07/2026

Firma: 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATORIA

A mis amigos quienes me acompañaron en toda la carrera, a pesar de todas las adversidades y a pesar de todos los problemas que les pude causar, nunca me dieron la espalda y me ayudaron a desarrollarme como persona. Muchas gracias por toda la ayuda que me brindaron, porque si no fuera por ustedes no pudiera haber completado este ciclo normalmente.

AGRADECIMIENTOS

En primera instancia, me gustaría agradecer a mi profesor guía, José Luis Carrasco Sáez el cual se ha tomado el tiempo de orientarme en cada capítulo, cada paso y que me ha acompañado desde inicios de la carrera en informática. Siendo un profesor el cual me ha dado apoyo en varias fases de mi carrera.

En segunda instancia, quiero agradecer por mi familia, ya que a pesar de todos los obstáculos que he tenido, ellos me han aconsejado y me han ayudado en todos los aspectos posibles cuando más lo necesitaba. Nunca me faltó nada gracias a ellos

Y no menos importante, me gustaría agradecer por los profesores y mi equipo de trabajo, sin ellos no sería posible este momento. Sin su ayuda nunca habría progresado en esta fase de mi vida.

RESUMEN

La presente memoria aborda el diseño y desarrollo de una solución móvil multiplataforma orientada a optimizar la gestión y trazabilidad de órdenes de trabajo en terreno para empresas de servicios informáticos. La problemática identificada radica en la descentralización de la información y la dispersión de datos operativos en canales informales, lo que incrementa significativamente el *overhead* operativo. Para mitigar esto, se propone una aplicación móvil desarrollada bajo el *framework* React Native y Expo, integrada en un ecosistema híbrido con una plataforma web administrativa en SvelteKit. La innovación principal de este estudio radica en el uso de la Inteligencia Artificial generativa mediante la metodología de *vibe coding*, delegando la escritura técnica de sintaxis a los agentes a través de una rigurosa arquitectura de *prompts*. Durante la fase de validación, se evaluó comparativamente el desempeño de tres modelos de lenguaje (Gemini 3.1 Pro Avanzado, Claude Sonnet 4.6 Max Effort y Microsoft Copilot) en la maquetación de interfaces que cumplieran con estándares de experiencia de usuario (UX). Los resultados demuestran que una ingeniería de instrucciones precisa acorta los ciclos de desarrollo tradicionales de tres meses a solo dos semanas, validando un cambio de paradigma donde el desarrollador humano evoluciona hacia un rol estratégico de arquitecto crítico y supervisor de la calidad de los sistemas automatizados.

Palabras clave: Aplicación móvil, Trazabilidad de servicios, Inteligencia Artificial generativa, *Vibe coding*, Experiencia de usuario (UX), React Native.

ABSTRACT

This thesis addresses the design and development of a cross-platform mobile solution oriented to optimize the management and traceability of field work orders for IT service companies. The identified problem lies in the decentralization of information and the scattering of operational data across informal channels, which significantly increases operational overhead. To mitigate this, a mobile application developed under the React Native and Expo framework is proposed, integrated into a hybrid ecosystem with an administrative web platform in SvelteKit. The main innovation of this study lies in the use of generative Artificial Intelligence through the *vibe coding* methodology, delegating the technical syntax writing to AI agents through a rigorous prompt architecture. During the validation phase, the performance of three large language models (Gemini 3.1 Pro Advanced, Claude Sonnet 4.6 Max Effort, and Microsoft Copilot) was comparatively evaluated in layout generation that complied with user experience (UX) design standards. The results demonstrate that precise prompt engineering shortens traditional development cycles from three months to just two weeks, validating a paradigm shift

where the human developer evolves towards a strategic role as a critical architect and supervisor of the quality of automated systems.

Keywords: Mobile application, Service traceability, Generative Artificial Intelligence, Vibe coding, User experience (UX), React Native.

GLOSARIO

- **API (Interfaz de Programación de Aplicaciones):** Interfaz que permite la comunicación y el intercambio de datos entre la plataforma web (backend) y la aplicación móvil
- **Backend:** Capa del sistema encargada de procesar las reglas del negocio
- **Bottom Tab Bar:** Barra de navegación dinámica ubicada en la parte inferior de la pantalla de una aplicación móvil
- **Endpoint:** Punto final de comunicación de una API que permite a la aplicación móvil consumir los servicios del servidor.
- **Frontend:** Interfaz visual de una plataforma o aplicación mediante la cual los usuarios interactúan y visualizan los datos.
- **Framework:** Marco de trabajo o entorno de desarrollo que proporciona una estructura base
- **JSON (JavaScript Object Notation):** Formato ligero de estructura de datos utilizado para representar e intercambiar información entre el servidor y la interfaz
- **Mock:** Conjunto de datos simulados que sirven como referencia realista
- **Overhead:** Sobrecarga de tiempo, costos o recursos adicionales que requiere una empresa para mantener sistemas complejos
- **Prompt:** Instrucciones detalladas y escritas en lenguaje natural que se utilizan para guiar y estructurar el comportamiento de una Inteligencia Artificial generativa
- **SafeAreaView:** Componente de interfaz móvil que asegura que el contenido de la pantalla no interfiera con los bordes o áreas intrusivas del dispositivo
- **ScrollView:** Componente visual que permite a los usuarios desplazarse vertical u horizontalmente a través de una lista de elementos
- **UI:** Proceso de diseño enfocado en la apariencia estética y la distribución de los elementos visuales e interactivos

Desarrollo de pantallas móviles mediante Inteligencia Artificial: Diseño de prompts como principal herramienta de programación enfocado en experiencia e interfaz de usuario

- **UX:** Enfoque de diseño que abarca todos los aspectos de la interacción del usuario final con el producto

INDICE DE CONTENIDOS

RESUMEN.....	4
ABSTRACT	4
GLOSARIO	5
INDICE DE FIGURAS	9
INDICE DE TABLAS	11
INTRODUCCIÓN.....	12
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	13
1.1 CONTEXTO GENERAL	13
1.2 SITUACIÓN ACTUAL	13
1.3 ACTORES INVOLUCRADOS	14
1.4 DESCRIPCIÓN Y CONTEXTUALIZACION DEL PROBLEMA.....	14
1.5 OBJETIVO GENERAL	15
1.6 OBJETIVOS ESPECIFICOS.....	15
CAPÍTULO 2: MARCO CONCEPTUAL	17
2.1 INTRODUCCIÓN	17
2.2 EXPERIENCIA DE USUARIO	17
2.3 INTERFAZ DE USUARIO.....	17
2.4 LOS PRINCIPIOS DE DISEÑO.....	17
2.4.1 SIMPLICIDAD	18
2.4.2 CONSISTENCIA.....	18
2.4.3 JERARQUIA VISUAL.....	18
2.4.4 ESTETICA Y ATRACCIÓN VISUAL	19
2.5 HERRAMIENTAS DE DESARROLLO	19
2.5.1 VIBE CODING.....	19
2.5.2 REACT NATIVE.....	20
2.5.3 EXPO Y EXPO ROUTER	21
2.6 ECOSISTEMA DE LA APLICACIÓN	21
2.6.1 IMPLEMENTACION DE IA EN EL DESARROLLO UI/UX.....	22
CAPITULO 3: PROPUESTA DE SOLUCIÓN	24
3.1 INTRODUCCIÓN Y DEFINICION DE LA ESTETICA.....	24
3.1.1 ESTRUCTURA BASE DE LA INTERFAZ	24

3.1.2 PALETA DE COLORES Y PSICOLOGIA DEL COLOR.....	24
3.1.3 NAVEGACIÓN DE LAS PANTALLAS Y FUNCIONALIDADES.....	25
3.2 GENERACIÓN DE LAS PANTALLAS MEDIANTE <i>PROMPTS</i>	26
3.2.1 PANTALLA ORDENES	27
3.2.2 PANTALLA DETALLES ORDEN	31
3.2.3 PANTALLA CHAT DE ORDEN.....	37
3.2.4 PANTALLA REPORTES	41
3.2.5 PANTALLA DETALLES DE REPORTE.....	45
3.2.6 PANTALLA RECTIFICACION DE REPORTE.....	49
3.2.7 PANTALLA DE PERFIL DE USUARIO.....	53
CAPITULO 4: VALIDACION DE LA SOLUCIÓN	57
4.1 INTRODUCCIÓN	57
4.2 CRITERIO DE EVALUACIÓN	57
4.3 RESULTADOS DE LA VALIDACIÓN.....	60
CAPITULO 5: CONCLUSIONES.....	60
5.1 APRENDIZAJE.....	61
5.2 LIMITACIONES	61
5.3 VISION A FUTURO DE LAS INTELIGENCIAS ARITIFICIALES.....	61

INDICE DE FIGURAS

FIGURA 1: ECOSISTEMA COMPLETO DE LA APLICACIÓN FUENTE: ELABORACIÓN PROPIA	23
FIGURA 2: PALETA DE COLORES SELECCIONADA PARA LA APLICACIÓN FUENTE: COOLORS (ELABORACIÓN PROPIA).....	25
FIGURA 3: FLUJO DE NAVEGACIÓN DE LA APLICACIÓN FUENTE: ELABORACIÓN PROPIA.....	26
FIGURA 4: PANTALLA DE ORDENES GENERADA MEDIANTE <i>PROMPT</i> FUENTE: GEMINI IA (3.1 PRO AVANZADO)	29
FIGURA 5: PANTALLA DE ORDENES GENERADA MEDIANTE <i>PROMPT</i> FUENTE: CLAUDE IA (SONNET 4.6 <i>MAX EFFORT</i>)	30
FIGURA 6: PANTALLA DE ORDENES GENERADA MEDIANTE <i>PROMPT</i> FUENTE: MICROSOFT COPILOT (RAZONAMIENTO PROFUNDO).....	31
FIGURA 7: PANTALLA DE DETALLES DE ORDEN GENERADA MEDIANTE <i>PROMPT</i> FUENTE: GEMINI IA (PRO 3.1 AVANZADO)	34
FIGURA 8: PANTALLA DE DETALLES DE ORDEN GENERADA MEDIANTE <i>PROMPT</i> FUENTE: CLAUDE IA (SONNET 4.6 <i>MAX EFFORT</i>)	35
FIGURA 9: PANTALLA DE DETALLES ORDEN GENERADA MEDIANTE <i>PROMPT</i> FUENTE: MICROSOFT COPILOT (RAZONAMIENTO PROFUNDO).....	36
FIGURA 10: PANTALLA DE CHAT DE ORDEN GENERADA MEDIANTE <i>PROMPT</i> FUENTE: GEMINI IA (PRO 3.1 AVANZADO)	38
FIGURA 11: PANTALLA DE CHAT DE ORDEN GENERADA MEDIANTE <i>PROMPT</i> FUENTE: CLAUDE IA (SONNET 4.6 <i>MAX EFFORT</i>)	39
FIGURA 12: PANTALLA DE CHAT DE ORDEN GENERADA MEDIANTE <i>PROMPT</i> FUENTE: MICROSOFT COPILOT (RAZONAMIENTO PROFUNDO).....	40
FIGURA 13: PANTALLA DE REPORTES GENERADA MEDIANTE <i>PROMPT</i> FUENTE: GEMINI (3.1 PRO AVANZADO)	42
FIGURA 14: PANTALLA DE REPORTES GENERADA MEDIANTE <i>PROMPT</i> FUENTE: CLAUDE IA (SONNET 4.6 <i>MAX EFFORT</i>)	43
FIGURA 15: PANTALLA DE REPORTES GENERADA MEDIANTE <i>PROMPT</i> FUENTE: MICROSOFT COPILOT (RAZONAMIENTO PROFUNDO).....	44
FIGURA 16: PANTALLA DE DETALLES DE REPORTE GENERADA MEDIANTE <i>PROMPT</i> FUENTE: GEMINI IA (3.1 PRO AVANZANDO).....	46
FIGURA 17: PANTALLA DE DETALLES DE REPORTE GENERADA MEDIANTE <i>PROMPT</i> FUENTE: CLAUDE IA (SONNET 4.6 <i>MAX EFFORT</i>).....	47
FIGURA 18: PANTALLA DE DETALLES DE REPORTE GENERADA MEDIANTE <i>PROMPT</i> FUENTE: MICROSOFT COPILOT (RAZONAMIENTO PROFUNDO).....	48

FIGURA 19: PANTALLA DE RECTIFICACIONES DE REPORTE GENERADA MEDIANTE <i>PROMPT</i> FUENTE: GEMINI IA (3.1 PRO AVANZADO)	50
FIGURA 20: PANTALLA DE RECTIFICACIONES DE REPORTE GENERADA MEDIANTE <i>PROMPT</i> FUENTE: CLAUDE IA (SONNET 4.6 MAX EFFORT).....	51
FIGURA 21: PANTALLA DE RECTIFICACIONES DE REPORTE GENERADA MEDIANTE <i>PROMPT</i> FUENTE: MICROSOFT COPILOT IA (RAZONAMIENTO PROFUNDO)	52
FIGURA 22: PANTALLA DEL PERFIL DE USUARIO GENERADA MEDIANTE <i>PROMPT</i> FUENTE: GEMINI IA (3.1 PRO AVANZADO)	54
FIGURA 23: PANTALLA DEL PERFIL DE USUARIO GENERADA MEDIANTE <i>PROMPT</i> FUENTE: CLAUDE IA (SONNET 4.6 MAX EFFORT)	55
FIGURA 24: PANTALLA DEL PERFIL DE USUARIO GENERADA MEDIANTE <i>PROMPT</i> FUENTE: MICROSOFT COPILOT (RAZONAMIENTO PROFUNDO).....	56

INDICE DE TABLAS

TABLA 1: EVALUACIÓN DE LA INTELIGENCIAS ARTIFICIALES EN LA GENERACIÓN DE PANTALLAS ACERCA DEL MODULO DE “ORDENES” FUENTE: ELABORACIÓN PROPIA	58
TABLA 2: EVALUACIÓN DE LA INTELIGENCIAS ARTIFICIALES EN LA GENERACIÓN DE PANTALLAS ACERCA DEL MODULO DE “REPORTES” FUENTE: ELABORACIÓN PROPIA.....	59
TABLA 3: EVALUACIÓN DE LA INTELIGENCIAS ARTIFICIALES EN LA GENERACIÓN DE PANTALLAS ACERCA DEL MODULO DE “PERFIL”	60

INTRODUCCIÓN

En el escenario tecnológico actual, la Inteligencia Artificial (IA) ha evolucionado de ser una herramienta de asistencia básica a consolidarse como la infraestructura de la productividad y ejecución autónoma global. Este avance ha redefinido la ingeniería de software mediante la aparición de modelos de alto razonamiento y entornos de desarrollo agenticos. Dentro de este nuevo paradigma surge el *vibe coding*, una metodología donde el programador transfiere la carga de escritura manual de sintaxis a la IA, interactuando con ella a través de descripciones conceptuales y *prompts* en lenguaje natural. Bajo este enfoque, el profesional humano se desplaza hacia una posición de pensamiento crítico y supervisión arquitectónica.

El presente trabajo aplica este enfoque agentic para resolver una problemática crítica identificada en una empresa de servicios informáticos: la falta de trazabilidad y centralización en la gestión de órdenes de trabajo en terreno. Al no contar con una plataforma unificada, la información operativa suele dispersarse en correos y mensajería instantánea, provocando el extravío de datos esenciales y un incremento innecesario en el *overhead* operativo. Para solucionar estas deficiencias, se plantea el diseño de la identidad visual y la maquetación de una aplicación móvil multiplataforma utilizando React Native y Expo, la cual se comunica con un backend centralizado y una interfaz administrativa web estructurada en SvelteKit. El sistema móvil no solo organiza las tareas, sino que aísla el entorno visual dinámicamente según el estado operativo del técnico y recolecta evidencias digitales rigurosas, tales como firmas y capturas fotográficas en terreno.

La metodología empleada consiste en la formulación de una arquitectura estructurada de *prompts* aplicados de forma transversal a tres de los principales agentes de IA de la industria: Gemini (3.1 Pro Avanzado), Claude (Sonnet 4.6 Max Effort) y Microsoft Copilot (Razonamiento Profundo). El objetivo de este procedimiento es evaluar cuantitativa y cualitativamente la capacidad de estos modelos para materializar prototipos de interfaces móviles que cumplan estrictamente con principios heurísticos de diseño como la simplicidad, la consistencia, la jerarquía visual y el atractivo estético.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1 CONTEXTO GENERAL

En el panorama tecnológico contemporáneo, la inteligencia artificial (IA) ha trascendido su fase de asistencia básica para consolidarse como la infraestructura de productividad global. Desde el año 2024, se ha consolidado un cambio de paradigma impulsado por la adopción masiva de estas tecnologías; de acuerdo con McKinsey & Company (2024), el 72% de las organizaciones a nivel mundial ya han integrado la IA en sus flujos operativos para generar valor real. Esta evolución ha progresado rápidamente hacia la denominada "IA agéntica", donde los sistemas no solo procesan información, sino que ejecutan tareas de manera autónoma. Proyecciones de Gartner (2025) estiman que, para finales de 2026, el 40% de las aplicaciones empresariales incorporarán agentes de IA específicos para tareas, lo que representa un salto cualitativo hacia sistemas de ejecución proactiva. Este avance es respaldado por las investigaciones de la Wharton School (2025), las cuales posicionan a la autonomía tecnológica como el motor principal del crecimiento de la productividad futura, permitiendo una reconfiguración total en la asignación de recursos y la reducción de la carga operativa técnica (TechClass, 2024).

1.2 SITUACIÓN ACTUAL

Esta transición hacia la autonomía ha redefinido drásticamente la ingeniería de software, especialmente ante el surgimiento de modelos con capacidades de razonamiento técnico avanzado como Claude. En la situación actual, este progreso tecnológico obligó a una reorientación estratégica en las herramientas de desarrollo; el caso de Cursor es emblemático, pues es una plataforma que evolucionó de ser un editor de código con funciones de asistencia hacia un entorno donde la IA asume el rol de programador primario. Esta mutación responde a un escenario donde la capacidad técnica de la IA para escribir, refactorizar y desplegar código ha comenzado a superar la velocidad de ejecución manual humana, forzando a las empresas a pivotar hacia modelos de desarrollo agénticos.

Bajo este nuevo esquema, el problema del costo tecnológico y el *overhead* operativo se aborda delegando la ejecución técnica a la inteligencia artificial, mientras que el desarrollador humano se desplaza hacia una posición de pensamiento crítico y supervisión arquitectónica (Forbes Argentina, 2026). En consecuencia, el desarrollo de aplicaciones móviles contemporáneo ya no se centra en la escritura de sintaxis, sino en la capacidad del profesional para actuar como un revisor estratégico que valida la lógica, la seguridad y la experiencia de usuario, permitiendo que la IA gestione la complejidad técnica subyacente y garantice el progreso ágil del proyecto.

1.3 ACTORES INVOLUCRADOS

La transición hacia modelos de autonomía agentica y la imperativa reducción de los costos operativos en el desarrollo de software convencional exigen una redefinición de las entidades que participan en el ciclo de vida de las aplicaciones. Bajo este nuevo paradigma, la eficiencia no subyace en la ejecución manual de código, sino en la sinergia estratégica entre la capacidad técnica de la inteligencia artificial y el juicio crítico humano. En este sentido, se identifican los siguientes actores clave dentro del ecosistema de desarrollo contemporáneo:

- **El desarrollador:** Como arquitecto crítico, representa el eje de la validación y supervisión del sistema. Su función ha evolucionado desde la escritura de la sintaxis hacia el pensamiento crítico, siendo el responsable de la lógica, garantizar la seguridad y asegurar que las soluciones generadas por la IA se alineen con los objetivos arquitectónicos y funcionales establecidos
- **El agente de IA:** Basado en los modelos de lenguaje de alto razonamiento como Claude, Gemini, entre otros. Este actor asume la responsabilidad de la programación primaria, la refactorización y la gestión de la complejidad técnica, funcionando como el motor de la ejecución que absorbe la carga operativa para mitigar el *overhead* financiero tradicional
- **La Organización y el Usuario final:** Constituyendo el estrato receptor del valor que se genera por la optimización del flujo de trabajo, en donde la mitigación del *overhead* tecnológico permite una transición hacia soluciones móviles de alta fidelidad donde la viabilidad económica y la calidad técnica convergen de manera integral. Este entorno se beneficia de ciclos de entrega acelerados y de una estabilidad operativa, garantizada por el despliegue que han refinados mediante una supervisión humana rigurosa sobre la base técnica automatizada

1.4 DESCRIPCIÓN Y CONTEXTUALIZACIÓN DEL PROBLEMA

A partir del contacto directo con una empresa de servicios informáticos, se logró identificar una problemática interna crítica relacionada con la centralización de la información. El foco del conflicto se encuentra en la gestión de las órdenes de trabajo en terreno, donde actualmente no es posible mantener un seguimiento claro y fidedigno del progreso de las actividades.

Esta situación se origina principalmente por fallas en los canales de comunicación. Al no contar con una plataforma única, las órdenes de trabajo suelen dispersarse entre correos electrónicos y aplicaciones de mensajería instantánea. Este manejo informal provoca el extravío de datos relevantes y la interrupción constante del flujo de trabajo, ya que la información no se encuentra disponible de manera inmediata para quienes la necesitan.

En consecuencia, la falta de una estructura digital centralizada genera un aumento innecesario en el *overhead* operativo. La empresa debe destinar tiempo y recursos adicionales a la recuperación de información perdida y a la resolución de problemas de comunicación, lo que impide que el personal se enfoque en el pensamiento crítico y en la ejecución eficiente de las tareas. Esta falta de trazabilidad evidencia la necesidad de una solución que organice los datos y permita una gestión operativa mucho más fluida y profesional.

Con el objetivo de solventar las deficiencias operativas detectadas, se plantea el desarrollo de una solución móvil centralizada que permita la gestión del registro y la trazabilidad integral de las actividades en terreno. Esta herramienta tecnológica debe facilitar el monitoreo constante del desempeño operativo, permitiendo la captura de retroalimentación técnica y la habilitación de un canal de comunicación directo para el intercambio de detalles específicos en tiempo real, evitando así la dispersión de datos en plataformas externas.

Asimismo, el sistema debe integrar mecanismos de validación de cumplimiento mediante la captura de evidencia fotográfica y la recolección de firmas digitales. Estas funciones actúan como protocolos de verificación obligatorios para confirmar la finalización formal de cada actividad, asegurando la integridad de la información y eliminando la ambigüedad en el cierre de las órdenes de trabajo dentro de la empresa.

1.5 OBJETIVO GENERAL

Diseñar la identidad visual de una aplicación móvil multiplataforma para empresas con trabajo en terreno mediante el framework React Native, optimizando el proceso de programación y diseño de interfaces a través de herramientas de inteligencia artificial. El objetivo es centralizar la información y mejorar la experiencia de usuario en terreno, reduciendo los tiempos de implementación técnica.

1.6 OBJETIVOS ESPECIFICOS

- Definir la estética visual de la aplicación, asegurando un diseño minimalista que facilite la legibilidad de los datos técnicos en dispositivos móviles además de estructurar las pantallas de la aplicación mediante el uso de componentes de React Native, estableciendo una jerarquía de información clara.

- Evaluar el desempeño de los agentes de inteligencia artificial en la generación de software, validando el cumplimiento de estándares de diseño y principios de experiencia de usuario (UX) para garantizar la calidad funcional dentro de entornos de programación agenticos.
- Validar la experiencia de la aplicación junto al atractivo visual para la percepción de las personas logrando una experiencia placentera para el uso regular de la aplicación

CAPÍTULO 2: MARCO CONCEPTUAL

2.1 INTRODUCCIÓN

En este capítulo se define los conceptos fundamentales que sustentan el desarrollo visual y técnico de la aplicación. Se abordan temas clave como la Interfaz de Usuario (*UI*), la experiencia de usuario (*UX*). El propósito es establecer un marco de referencia que justifique las decisiones de diseño adoptadas para optimizar la labor del personal en terreno.

2.2 EXPERIENCIA DE USUARIO

"La experiencia de usuario abarca todos los aspectos de la interacción del usuario final con la empresa, sus servicios y sus productos" (Norman & Nielsen, 1998). Dentro de la aplicación móvil, la experiencia debe ser intuitiva, visualmente coherente y sencilla de comprender, con el fin de reducir la curva de aprendizaje del personal técnico.

Al priorizar una estructura minimalista, se busca eliminar cualquier elemento distractor que pueda entorpecer la captura de datos en terreno. De este modo, la UX no solo se enfoca en la satisfacción visual, sino en asegurar que el flujo de trabajo —desde la recepción de una orden hasta su cierre— sea fluido, lógico y libre de fricciones operativas, garantizando así la integridad de la información recolectada.

2.3 INTERFAZ DE USUARIO

A diferencia de la Experiencia de Usuario (*UX*), la Interfaz de Usuario (*UI*) se define como el proceso de diseñar la apariencia y la sensación de un producto digital. Este ámbito se encarga de gestionar los elementos visuales e interactivos —como imágenes, íconos y tipografías— que permiten al usuario navegar por el sistema de forma clara (UX Design Institute, n.d.).

En el desarrollo de esta solución, la UI se enfoca en proyectar la identidad visual del proyecto a través de una maquetación limpia y organizada. Al centrarse en la estética visual y la jerarquía de los elementos, se garantiza que el técnico en terreno pueda interactuar con las órdenes de trabajo de manera fluida, asegurando que el diseño no solo sea agradable a la vista, sino también funcional y coherente con las necesidades operativas.

2.4 LOS PRINCIPIOS DE DISEÑO

En el ámbito del diseño y la experiencia de usuario (*UX*), existen principios fundamentales que actúan como estándares universales. Estos lineamientos permiten estructurar interfaces eficientes, asegurando que la interacción del usuario con la aplicación sea fluida y orientada al cumplimiento de objetivos.

Dada la naturaleza operativa de esta solución y la necesidad de uso en entornos de movilidad, se han adoptado lineamientos que privilegian la simplicidad y la claridad frente a la complejidad visual. El objetivo principal es disminuir la carga cognitiva del personal en terreno, minimizando las distracciones para gestionar la información y la trazabilidad de las tareas de manera más ágil y eficiente. Bajo este enfoque, los criterios seleccionados son los siguientes:

2.4.1 SIMPLICIDAD

Como su nombre lo indica, la simplicidad consiste en reducir la complejidad visual y técnica para los usuarios. El objetivo es mantener una interfaz limpia que permita al personal técnico enfocarse únicamente en las tareas esenciales de su jornada, eliminando cualquier elemento o función que no sea estrictamente necesario para el trabajo diario.

Al priorizar un diseño minimalista centrado en lo importante —como las órdenes de trabajo y su estado actual—, se logra disminuir la carga cognitiva del técnico. Esto significa que el usuario no tiene que procesar información innecesaria, favoreciendo la agilidad y eficiencia en terreno.

2.4.2 CONSISTENCIA

El principio de la consistencia se basa en mantener una uniformidad visual y operativa en la interfaz entre pantallas. El objetivo es que los elementos se vean y funcionen de la misma manera en cada sección, asegurando que el comportamiento de la aplicación sea predecible para el usuario.

La consistencia es una característica clave en el desarrollo de las aplicaciones, ya que genera familiaridad repitiendo patrones visuales y flujos de navegación. De esta forma el técnico no necesita aprender a usar cada pantalla, disminuyendo la confusión y permitiendo una interacción mucho más rápida y fluida durante la jornada laboral.

2.4.3 JERARQUIA VISUAL

La jerarquía visual se refiere a la distribución de los elementos en la pantalla, siguiendo un orden de prioridad para captar la atención del usuario. Este principio ayuda a comprender la estructura de la aplicación y cómo se organiza la información, facilitando la navegación entre los diferentes niveles de importancia de los datos.

En el desarrollo de esta herramienta, se aplicó dicho criterio para destacar la información crítica sobre los datos secundarios. Mediante el uso de colores llamativos en los botones de acción y espacios estratégicos, se logra que el técnico pueda identificar lo más importante a simple vista. Esto optimiza el tiempo de respuesta y permite que el flujo de trabajo sea lógico y fácil de seguir en terreno.

2.4.4 ESTETICA Y ATRACCIÓN VISUAL

Para que los principios de usabilidad mencionados anteriormente logren su propósito, es fundamental considerar la estética y la atracción visual como el primer punto de contacto. La apariencia de la interfaz no es solo un aspecto decorativo, sino el factor inicial que capta la atención del usuario y genera la confianza necesaria para interactuar con la herramienta desde el primer vistazo.

Un diseño estéticamente cuidado genera una percepción positiva de la funcionalidad. Una interfaz atractiva reduce la resistencia al cambio de los técnicos que están acostumbrados a procesos manuales, logrando que confíen más en las herramientas digitales desde su primer uso.

2.5 HERRAMIENTAS DE DESARROLLO

Debido a que el enfoque del proyecto reside en la funcionalidad y la inmediatez de la información, se optó por una combinación de tecnologías que prioriza la rapidez en el despliegue. Bajo estos criterios, la elección de las herramientas se fundamenta en su capacidad para reducir la complejidad técnica y facilitar una adopción ágil durante el proceso de desarrollo:

2.5.1 VIBE CODING

En el contexto del desarrollo de software contemporáneo, la integración de inteligencias artificiales generativas ha impulsado un nuevo paradigma conocido en la industria como *vibe coding*. A diferencia de los agentes autónomos que ejecutan proyectos completos sin supervisión, estas tecnologías operan como asistentes técnicos que requieren la dirección constante del usuario para estructurar y generar soluciones.

Este concepto es una metodología de desarrollo en donde el programar transfiere la carga de escritura de sintaxis a la inteligencia artificial. En lugar de codificar manualmente las líneas de código, en donde el desarrollador interactúa con los asistentes mediante descripciones conceptuales de alto nivel transmitiendo la intención arquitectónica, la

lógica funcional o el aspecto visual deseado. Bajo este enfoque, el desarrollador se convierte en un arquitecto crítico o director del proyecto.

Al delegar la ejecución técnica a las inteligencias artificiales a través del *vibe coding*, se puede lograr una reducción significativa de los tiempos operativos y del *overhead* técnico. Permitiendo que el desarrollador centre sus actividades exclusivamente en el diseño estratégico, la orquestación de los componentes y la validación de la calidad, acelerando la entrega de soluciones tecnológicas funcionales.

2.5.1.1 GEMINI AI

Su elección se justifica por su capacidad para procesar ventanas de contexto masivas de código. Esta característica técnica le permite ingerir y analizar múltiples archivos, librerías y documentaciones del repositorio simultáneamente. En la generación de código mediante *vibe coding*, Gemini resulta fundamental para comprender la arquitectura global de la aplicación, mantener la coherencia de los estados y variables entre distintas pantallas, y refactorizar componentes interdependientes sin perder el contexto general del proyecto

2.5.1.2 CLAUDE AI

Esta herramienta se integra de manera general debido a su alto rendimiento comprobado en evaluaciones de ingeniería de software. Su arquitectura está fuertemente optimizada para la programación lógica, destacando por la generación de código limpio y su capacidad excepcional para seguir instrucciones de diseño de manera estricta. En el proyecto, actúa como el ejecutor principal para la estructuración, ordenamiento y refactorización de componentes complejos en React Native, minimizando la introducción de errores lógicos.

2.5.1.3 MICROSOFT COPILOT

Al ser un asistente conversacional avanzado impulsado por modelos de lenguaje de última generación, su rol se enfoca en la generación ágil de código funcional y la resolución de problemas en tiempo real. Es utilizado para la creación rápida de estructuras base, la implementación de algoritmos específicos y la asistencia directa con la sintaxis de las librerías de Expo. Su uso acelera la resolución de bloqueos técnicos, permitiendo mantener un flujo de desarrollo continuo.

2.5.2 REACT NATIVE

Desarrollado y lanzado por Facebook en 2015, React Native surgió como una respuesta técnica a la complejidad del desarrollo nativo tradicional. Anteriormente, la creación de aplicaciones para Android y iOS requería programar en lenguajes distintos y mantener dos códigos independientes, lo que generaba problemas de consistencia y duplicaba los tiempos de trabajo.

En contraste con otras alternativas como Flutter, que utiliza el lenguaje Dart con una estructura más rígida y cercana a lenguajes como C o Java, React Native ofrece una curva de aprendizaje mucho más accesible y simplificada para el diseño de las interfaces. Para los objetivos de este proyecto, se priorizo esta herramienta por la flexibilidad que brinda al momento de maquetear, lo que permite centrar los esfuerzos en alcanzar un estilo minimalista y una jerarquía visual clara. Facilitando que la trazabilidad no solo sea funcional, si no que sea visualmente ligera.

2.5.3 EXPO Y EXPO ROUTER

Expo es un *framework* de código abierto que funciona sobre React Native, ofreciendo un flujo de trabajo gestionado. Esta plataforma actúa como una capa de gestión que automatiza las configuraciones complejas de los sistemas operativos *iOS* y *Android*, permitiendo que el desarrollo se centre exclusivamente en la lógica y el diseño de la interfaz.

Dentro de este ecosistema, se integra la librería Expo Router, cuya función principal es el enrutamiento basado en archivos. Bajo este esquema, la estructura de la aplicación se organiza de forma intuitiva, donde cada archivo generado en el directorio del proyecto se convierte automáticamente en una pantalla. Para el desarrollo de esta herramienta, dicha arquitectura facilita una navegación fluida y jerárquica, asegurando que la simplicidad visual se mantenga a través de una organización de software limpia, eficiente y fácil de escalar.

La elección de este framework se fundamenta en su capacidad para mitigar el overhead de configuración inicial, permitiendo que el desarrollo se ejecute directamente sobre una base técnica organizada y preconfigurada. Al reducir la complejidad de los entornos nativos, esta tecnología optimiza los tiempos de implementación y asegura que el esfuerzo se concentre en generar valor funcional para la organización, garantizando un flujo de trabajo ágil, eficiente y escalable.

2.6 ECOSISTEMA DE LA APLICACIÓN

El sistema se articula a través de un ecosistema híbrido compuesto por dos entornos principales, diseñados para garantizar la integridad de los datos y la fluidez operativa entre la administración y el personal en terreno:

- **Plataforma Web de Gestión (SvelteKit):** Este entorno funciona siendo el núcleo central del sistema. Desarrollado con SvelteKit, este integra tanto la interfaz administrativa mostrando los datos (*Frontend*), como la lógica del servidor y acceso a los datos (*Backend*). La plataforma es responsable de procesar las reglas del negocio, gestionar la persistencia de la información y exponer una interfaz de programación de aplicaciones (API) para la comunicación con clientes externos y gestionar la persistencia de la información en una base de datos (PostgreSQL y Prisma) respaldada por un almacenamiento optimizado de archivos (CloudFlare R2)
- **Aplicación Móvil Operativa (Expo y React Native):** Diseñada para los trabajadores en terreno, esta herramienta se construye sobre el framework de Expo utilizando React Native. Su función principal es el consumo de los servicios proporcionados por la plataforma web, permitiendo a los usuarios finales interactuar con las órdenes de trabajo, registrar evidencias y sincronizar el progreso en tiempo real mediante el consumo de los *endpoints* definidos en el *backend* centralizado.

2.6.1 IMPLEMENTACION DE IA EN EL DESARROLLO UI/UX

Dentro de la arquitectura de la aplicación móvil reside la integración de herramientas de inteligencia artificial aplicadas directamente a la capa de construcción técnica. Como se evidencia en el diagrama del ecosistema, el desarrollo de la aplicación es asistido por asistentes de IA generativa (como Claude, Gemini o Copilot), los cuales asumen la ejecución técnica de la interfaz, permitiendo que el desarrollador ejerza un rol de arquitecto crítico

Dicha implementación está enfocada exclusivamente en optimizar la experiencia del usuario (UX) y la interfaz (UI), garantizando que el diseño visual responda con precisión a las necesidades de los trabajadores en terreno y reduciendo de manera significativa el overhead operativo asociado a la maquetación tradicional de componentes.

Desarrollo de pantallas móviles mediante Inteligencia Artificial: Diseño de prompts como principal herramienta de programación enfocado en experiencia e interfaz de usuario

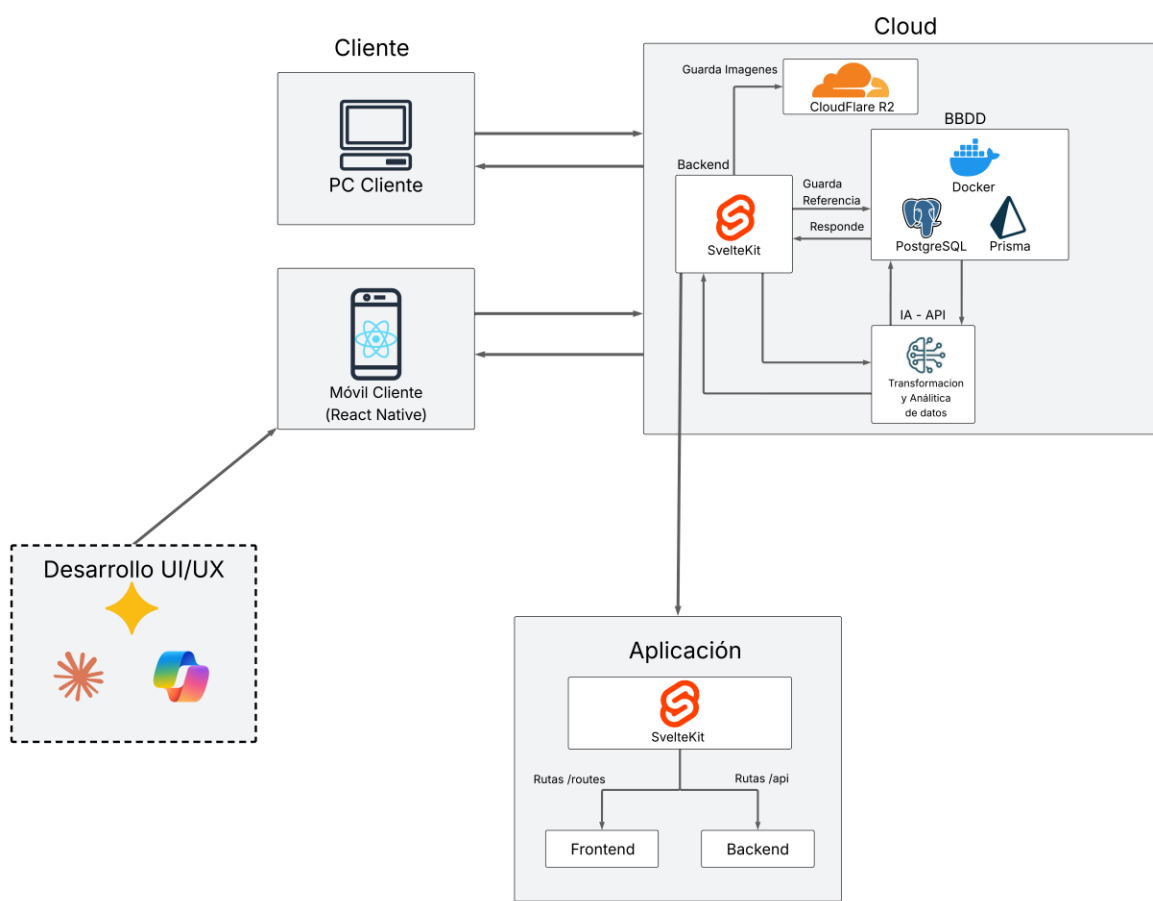


Figura 1: Ecosistema completo de la aplicación
Fuente: Elaboración Propia

CAPITULO 3: PROPUESTA DE SOLUCIÓN

3.1 INTRODUCCIÓN Y DEFINICION DE LA ESTETICA

En el presente capítulo se documenta el proceso de construcción de la interfaz móvil bajo un modelo de desarrollo asistido, donde las inteligencias artificiales actúan como ejecutores técnicos. En este flujo, los agentes asumen la responsabilidad de la generación de código y la maquetación visual, asegurando que la implementación técnica responda de manera estricta a las heurísticas y principios de diseño detallados en el capítulo anterior para optimizar la experiencia del usuario en terreno.

3.1.1 ESTRUCTURA BASE DE LA INTERFAZ

Previo a la ejecución técnica asistida por la inteligencia artificial, es fundamental establecer los elementos base de la identidad visual: la tipografía y la paleta de colores. Se ha definido una estética cuya prioridad es el orden estructural y la legibilidad.

En este sentido, se ha optado por un diseño minimalista funcional orientado a un entorno técnico profesional, asegurando que la apariencia de la interfaz facilite la concentración y reduzca la fatiga visual del operario durante la jornada en terreno. Estos parámetros actúan como las directrices fundamentales para que el ejecutor técnico pueda generar componentes coherentes con las necesidades operativas de la empresa.

3.1.2 PALETA DE COLORES Y PSICOLOGIA DEL COLOR

La paleta de colores es un aspecto fundamental para la imagen que se proyecta. Diversos estudios demuestran que los colores poseen la capacidad de evocar pensamientos o emociones específicas dependiendo de los contextos, un fenómeno estudiado por la psicología del color que resulta crítico al diseñar interfaces de *software*.

Bajo este enfoque práctico, la elección de tonalidades azules en combinación con el blanco responde a la necesidad de otorgar un entorno de trabajo serio y corporativo. El azul se integra para proyectar profesionalismo en la gestión de datos, mientras que el blanco se utiliza como base para estructurar la visualización de las tareas asignadas. De esta manera, el operador técnico dispone de un espacio de trabajo digital despejado, donde los elementos de la *interfaz* se distinguen con claridad sin importar la carga de información de la orden de trabajo.

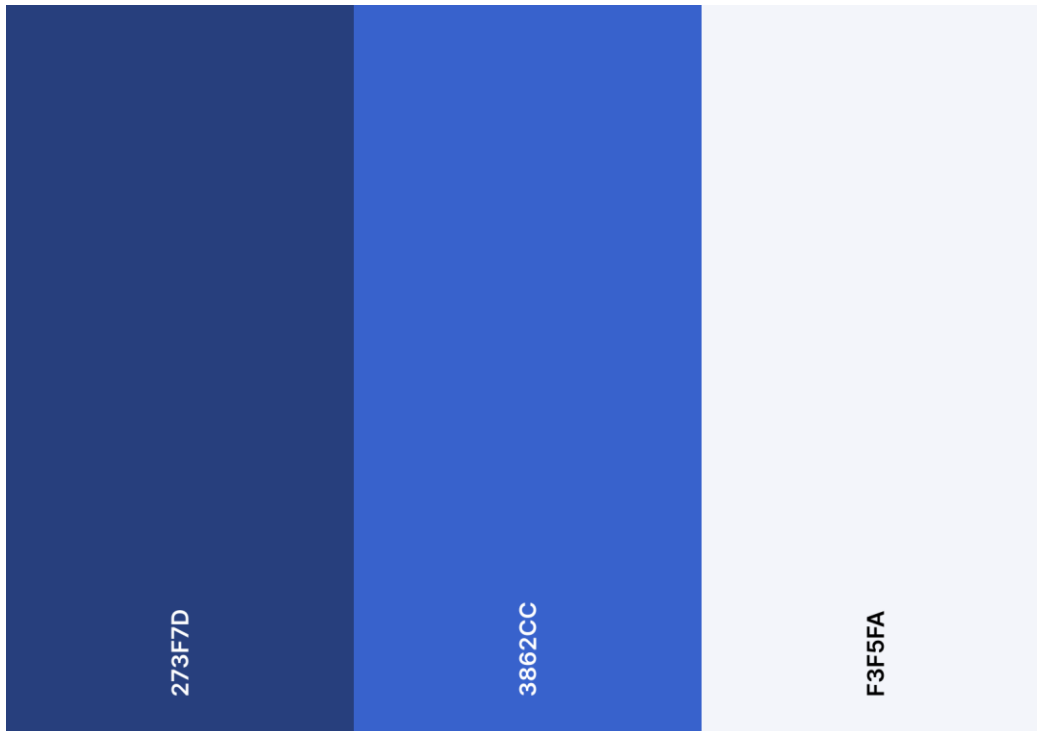


Figura 2: Paleta de colores seleccionada para la aplicación
Fuente: Coolors (Elaboración propia)

3.1.3 NAVEGACIÓN DE LAS PANTALLAS Y FUNCIONALIDADES

La arquitectura de la solución móvil se proyecta en un modelo de navegación contextual, diseñado para optimizar la gestión de tareas y mitigar la carga cognitiva en terreno. Para lograrlo, la estructura base debe organizar los componentes mediante una barra de navegación inferior (*Bottom Tab Bar*) dinámica, capaz de adaptarse al estado operativo del usuario para mostrar únicamente las herramientas pertinentes.

En cuanto al flujo de la navegación, se establece que, tras el inicio de sesión, la pantalla debe mostrar el listado de trabajos/ordenes activos junto a una barra principal dividida en tres secciones: Ordenes (siendo la pantalla mencionada anteriormente), Reportes (para consultar trabajos finalizados) y Perfil (gestión del usuario).

Para garantizar la simplicidad y evitar las distracciones operativas, al adentrarse en una tarea específica el sistema deberá aislar el entorno visual, modificando la barra de navegación de la siguiente manera:

- **Ordenes en curso:** La barra principal será reemplazada por un entorno de ejecución con solo dos pestañas: Orden (detalles y requerimientos técnicos) y Chat (comunicación contextual en tiempo real).

- **En trabajos finalizados:** Al acceder desde la sección de reportes, el flujo transicionara a una barra de auditoría compuesta por la pestaña de Reporte (evidencia del cierre) y Rectificación (para el anexo de notas correctivas).

Esta separación jerárquica y dinámica establece el plano arquitectónico base. Con estas directrices estructurales claras, el diseño garantiza responder con precisión a las fases del trabajo en terreno, sirviendo como mapa de ruta para que el ejecutor técnico (IA) inicie la etapa de codificación de la interfaz.

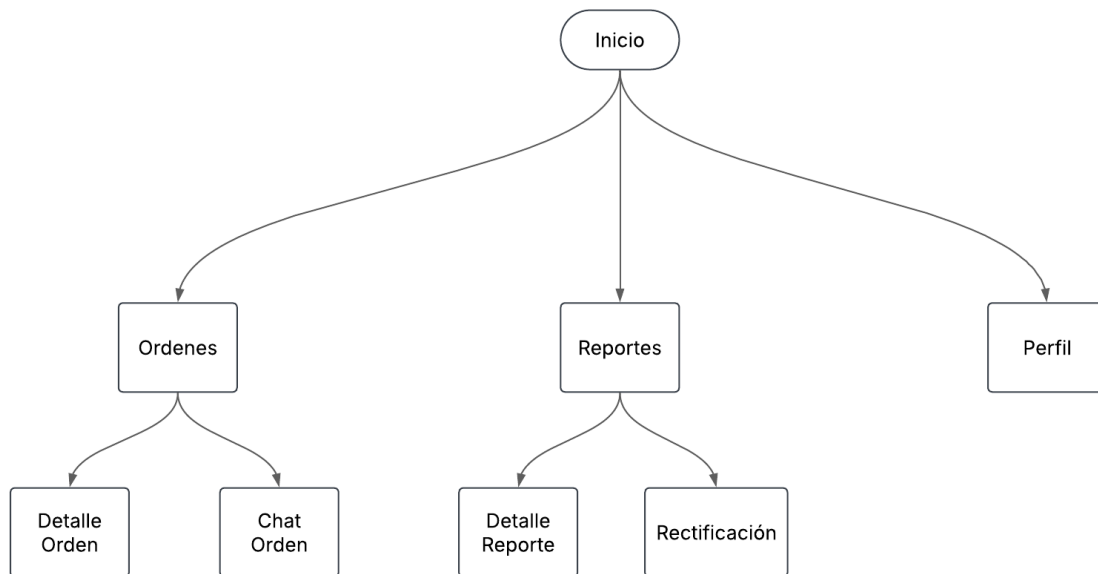


Figura 3: Flujo de navegación de la aplicación
Fuente: Elaboración Propia

3.2 GENERACIÓN DE LAS PANTALLAS MEDIANTE *PROMPTS*

Una vez definida la estructura base de la aplicación móvil, se procede a la construcción de las pantallas mediante el uso de herramientas de inteligencia artificial. Esta generación se realiza a través de *prompts* (instrucciones escritas de forma natural que guían el comportamiento de la IA), los cuales entregan indicaciones detalladas para crear cada parte de la interfaz. En estas instrucciones se explican los elementos visuales, los flujos y los requisitos que debe tener cada pantalla, asegurando que se cumplan los estándares estéticos y de usabilidad establecidos en el capítulo anterior. Se aplicarán los mismos *prompts* en las tres inteligencias artificiales para verificar los resultados y evaluar la efectividad de la especificidad de las instrucciones en cada una de ellas.

3.2.1 PANTALLA ORDENES

Dentro del primer *prompt* que se les brindará a las inteligencias artificiales, **destinado a** generar la pantalla principal donde el técnico visualizará y gestionará sus órdenes de trabajo, se mencionará el *framework* a trabajar, la paleta de colores y el funcionamiento de las pantallas. Asimismo, se establecerán los requerimientos técnicos exactos de los componentes visuales y se integrará una estructura de datos *JSON* (JavaScript Object Notation) para que los modelos utilicen información de referencia realista al momento de estructurar la interfaz. A continuación, se detalla la instrucción exacta ingresada

“Tomando en cuenta el desarrollo UX/UI para una aplicación móvil, genera una pantalla utilizando Expo Router, React Native y Typescript enfocándose únicamente en el diseño con los siguientes atributos:

- Debe contener un Bottom Tab Bar con 3 redirecciones con los siguientes iconos correspondientes: Ordenes (con el icono de un portapapeles) / Reportes (con el icono de un papel) / perfil (con el icono de una persona), asegúrate que la tab bar sea de color #3862CC y el texto sea de color blanco, al igual que los iconos puedan verse claramente (siendo estos también de color blanco). La tab bar debe tener una pista visual sobre la pantalla que se esté utilizando, también mantener un orden entre el espacio superior y la pista visual de la tab bar
- La pantalla debe contener un header donde debe darle la bienvenida al usuario para mostrar cual usuario está usando la aplicación, aquí un ejemplo: "Hola (usuario)!" y abajo se agregue un texto que se diga "Ordenes"
- La pantalla debe listar las ordenes dependiendo del estado, dichos estados pueden ser: Programadas y En progreso. Las ordenes deben estar en una lista desplegable tipo acordeón para la elección del estado de la orden
- La orden debe mostrar: Título de la orden, fecha de creación (debe estar en la esquina superior derecha), descripción (content) y el estado en el que esta y un botón con el texto "Ver Detalles" ejecutar la redirección a la pantalla de dicha orden, los detalles deben estar con contornos redondeados y sean de un color blanco que se diferencie del fondo con un sombreado ligero en abajo para mostrar un poco de profundidad en las cartas
- Se debe importar SafeAreaView para que no exista información en los extremos de las pantallas y esta pueda ser accesible

Elementos necesarios para la construcción:

- Los colores a utilizar son los siguientes: #273F7D para los textos a excepción del texto dentro del botón "Ver Detalles" (el cual se debe usar el color blanco); #3862CC para el

color del fondo del botón "Ver Detalles"; finalmente usaras el color #F3F5FA para el fondo de la pantalla general

-La redirección para el botón "Ver detalles" debe ser con el siguiente directorio: `router.push(`/orders/${id}`)`

- Los datos del json son el siguiente para tomar referencia de que datos utilizar en la pantalla, también se debe procurar con rellenar datos de prueba para comprobar la visualización:

```
{  
  "id": null, "title": "", "content": "", "response": "", "status": "",  
  "longitude": null, "latitude": null, "deviceId": "",  
  "activities": [], "materials": [], "signature": null, "evidence": [],  
  "scheduledAt": null, "startedAt": null, "closedAt": null, "completedAt": null,  
  "memberId": "", "organizationId": "", "createdAt": "", "updatedAt": "",  
  "assignee": {  
    "id": "", "organizationId": "", "userId": "", "role": "", "createdAt": "",  
    "user": { "id": "", "name": "", "email": "" }  
  }  
}
```



Figura 4: Pantalla de ordenes generada mediante *prompt*
Fuente: Gemini IA (3.1 Pro Avanzado)

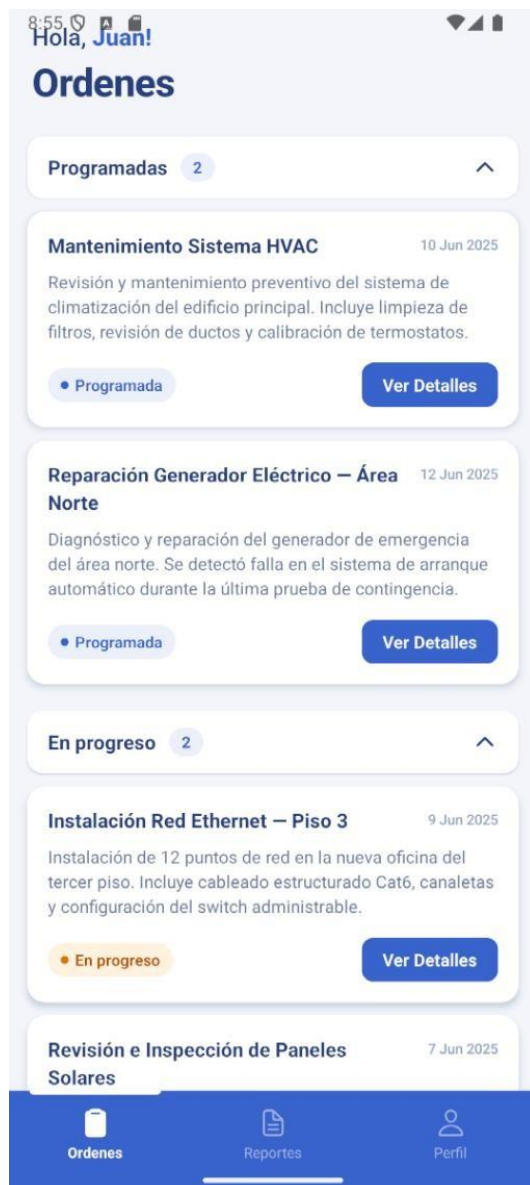


Figura 5: Pantalla de ordenes generada mediante *prompt*
Fuente: Claude IA (Sonnet 4.6 Max Effort)



Figura 6: Pantalla de ordenes generada mediante *prompt*
Fuente: Microsoft Copilot (Razonamiento Profundo)

3.2.2 PANTALLA DETALLES ORDEN

Aprovechando la memoria contextual de las herramientas de inteligencia artificial respecto a las instrucciones entregadas previamente, en los siguientes *prompts* se definirán únicamente las características y funcionalidades de las nuevas pantallas. La siguiente vista a generar corresponde a los detalles de la orden, cuyo objetivo es

desplegar la información específica de la tarea, permitir la actualización de estados con comentarios y mantener un registro de trazabilidad sobre el trabajo en curso. A continuación, se detalla la instrucción ingresada.

“ Con las instrucciones mencionadas anteriormente con respecto al framework, paleta de colores y formato de datos, genera una nueva pantalla con los siguientes atributos:

- El directorio de la pantalla está disponible en `apps/(tabs)/orders/[orderId]`
- Debe contener un bottom tab bar con el mismo diseño que el de la página anterior, a excepción que esta debe 2 iconos: Order (con un icono de un documento); y el otro icono debe ser chat(con un icono de burbuja de chat con 3 puntos), este bottom bar se debe generar en un archivo `_layout.tsx` dentro del directorio `apps/(tabs)/orders/[orderId]`, ya que en el directorio mencionado se generan 2 pantallas diferentes además del `_layout.tsx`
- La pantalla debe mostrar un único formulario, siendo un recuadro con el mismo estilo de la pantalla anterior sobre las ordenes listada, repitiendo el sombreado
- El recuadro debe contener la siguiente información:

1) Titulo, estando en la esquina superior izquierda del recuadro

2) ID

3) Contenido, siendo este la descripción del trabajo

4) Estado actual, para mostrar el estado actual del trabajo sin necesidad de volver a la ventana anterior para saber el estado de este

5) Campo de "Respuesta" para las actualizaciones de estado, dejando un mensaje estilo GitHub

6) Debe tener una sección llamada "actividades" en donde se puedan agregar y eliminar actividades, la función de eliminar debe estar al lado de la actividad y con icono de basurero para que sea minimalista

7) Debe tener una sección llamada "materiales" con las mismas características las cuales debe tener la sección "actividades"

8) Debe desplegar una lista tipo acordeón para cambiar el estado de la actividad

9) Debe tener un log de las veces que se cambió el estado de la actividad, además de mostrar la fecha en la que se cambió, al estado el cual se cambió y el responsable del que

realizo el cambio. Este cambio se debe aplicar exclusivamente cuando la orden se haya actualizado

10) En caso de que la actividad este en completado, se debe desplegar una sección en la cual se deba agregar una foto y una firma. La foto debe tener una preview de la imagen además de la posibilidad de eliminarla en caso de que el usuario saque mal la foto. Asegúrate de que este punto sea completamente funcional, pidiendo permisos de cámara o de galería, el área de la firma debe ser un área grande y que la pantalla no se mueva al momento de ir utilizando el dedo para firmar

Al actualizar la orden, debe haber un botón de confirmación o cancelación para evitar toques accidentales”

Desarrollo de pantallas móviles mediante Inteligencia Artificial: Diseño de prompts como principal herramienta de programación enfocado en experiencia e interfaz de usuario

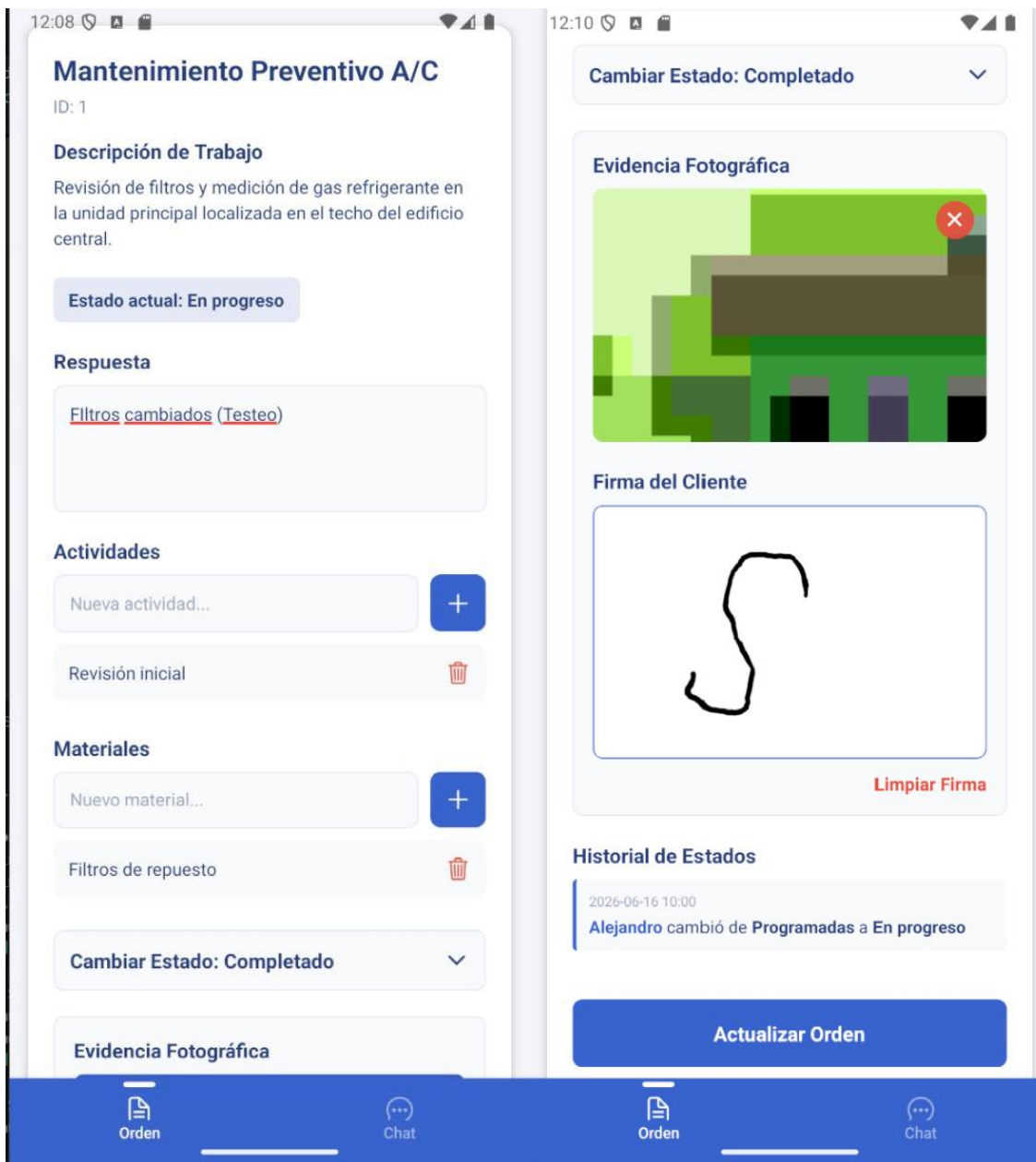


Figura 7: Pantalla de detalles de orden generada mediante *prompt*
Fuente: Gemini IA (Pro 3.1 Avanzado)

Desarrollo de pantallas móviles mediante Inteligencia Artificial: Diseño de prompts como principal herramienta de programación enfocado en experiencia e interfaz de usuario

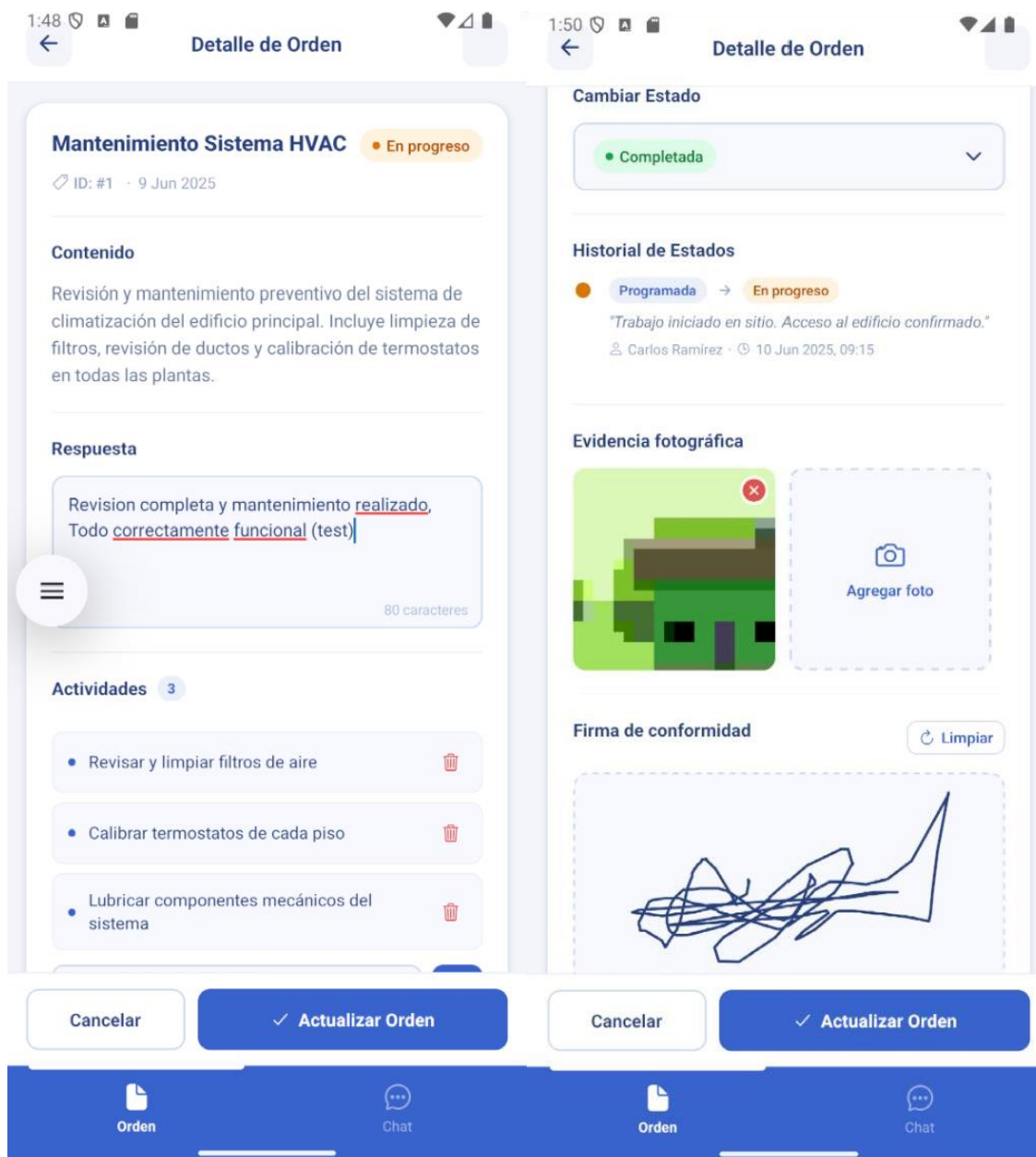


Figura 8: Pantalla de detalles de orden generada mediante *prompt*
Fuente: Claude IA (Sonnet 4.6 Max Effort)

The screenshot shows a mobile application interface for an order management system. At the top, the status bar displays the time 1:56 and various icons. The app's header is a blue bar with a back arrow and the title 'Orden'. The main content area is a white card with the following sections:

- Revisión de equipo A** (ID: 1)
- Contenido**: Inspección general y reemplazo de filtro.
- Estado actual**: A button labeled 'Programadas'.
- Respuesta**: A text input field containing 'Filtros Realizado (test)'.
- Actividades**: A section with a blue '+' button and a dropdown arrow.
- Materiales**: A section with a blue '+' button.
- Material List**: Two items, 'Filtro tipo X' and 'Guantes', each with a red trash icon.
- Evidencia**: Two buttons, 'Tomar foto' (with a camera icon) and 'Galería' (with a photo icon).
- Firma**: A large white box containing a black ink signature.

The bottom of the screen features a blue navigation bar with two icons: a document icon labeled 'Order' and a speech bubble icon labeled 'Chat'.

Figura 9: Pantalla de detalles orden generada mediante *prompt*
Fuente: Microsoft Copilot (Razonamiento Profundo)

3.2.3 PANTALLA CHAT DE ORDEN

Como etapa final en la construcción de este módulo operativo, se procede a la generación del chat asociado a la orden. Manteniendo la metodología de aprovechamiento de la ventana de contexto, esta iteración omite las instrucciones sobre el *framework* y la identidad visual, permitiendo que la herramienta de inteligencia artificial se enfoque de lleno en la arquitectura de una interfaz de mensajería clara y responsiva. Con el objetivo de garantizar una correcta interacción del usuario final con este componente, se utilizó la siguiente instrucción:

“Continuando con la creación de pantallas, genera una pantalla de chat interno con las siguientes características:

- Al abrir la ventana de chat, esta debe mostrar directamente el chat de la orden asociada, debido a que el chat es utilizado para mostrar avances o consultar dudas sobre la orden correspondiente.
- Se debe mantener la paleta de colores, la forma del recuadro al estilo mencionado en las pantallas anteriores
- Las burbujas de chat deben ser de color azul oscuro con letras blancas, como se ha hecho en ocasiones anteriores
- En la parte superior del chat, debes mencionar el id de la orden además del título correspondiente a la orden
- El área para escribir y mandar los mensajes debe ubicarse en la parte inferior de la pantalla, asegúrate que tenga un tamaño considerable
- El botón de enviar debe estar en gris cuando la caja del mensaje este vacío y azul cuando si haya texto dentro de la caja de mensajes, agregando también que en la caja de mensajes debe tener el texto "enviar un mensaje" para indicar que es el área de escritura.
- Utiliza un conjunto de mensajes *mock* que simule un historial persistente de conversación, y este debe tener un *ScrollView* vertical, siendo los mensajes más recientes los últimos.”



Figura 10: Pantalla de chat de orden generada mediante *prompt*
Fuente: Gemini IA (Pro 3.1 Avanzado)



Figura 11: Pantalla de chat de orden generada mediante *prompt*
Fuente: Claude IA (Sonnet 4.6 *max effort*)

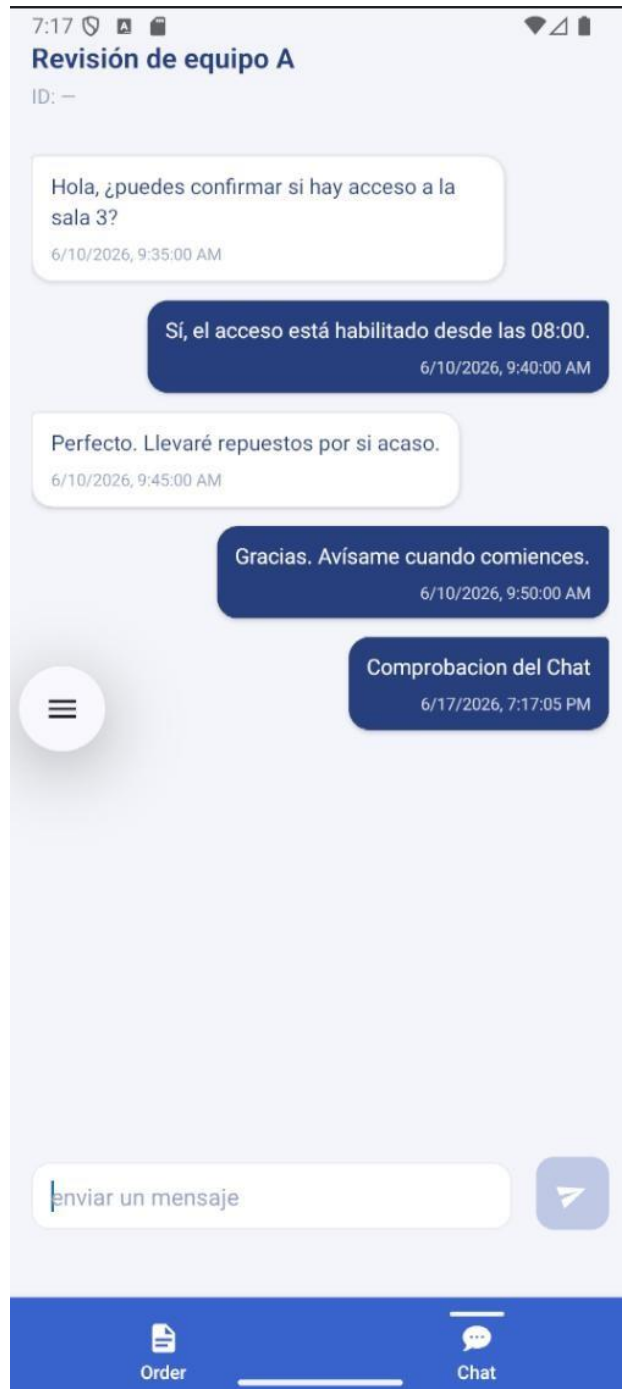


Figura 12: Pantalla de chat de orden generada mediante *prompt*
Fuente: Microsoft Copilot (Razonamiento Profundo)

3.2.4 PANTALLA REPORTES

Una vez concluida la generación de las pantallas correspondientes al módulo de órdenes de trabajo, se da paso a la construcción de las interfaces destinadas a los reportes de las tareas ejecutadas. Utilizando nuevamente la memoria contextual de las herramientas de inteligencia artificial, se formularán *prompts* más concisos y eficientes, garantizando que el diseño mantenga la coherencia visual y técnica con la estructura base de la aplicación.

Para la primera interfaz de este nuevo módulo, se le indicará al modelo que estructure una vista donde se listen las órdenes de trabajo ya finalizadas, presentando esta sección bajo el título de "Reportes". A continuación, se detalla la instrucción exacta ingresada:

“Tomando de referencia las instrucciones acerca de la pantalla cuya descripción general fue el listado de Ordenes en progreso y programadas. Genera una pantalla con las siguientes diferencias:

- Se debe listar exclusivamente todas las ordenes con estado COMPLETADAS
- Se debe mostrar la ID de la orden realizada correspondiente, además de la fecha de cuando se completo
- La redirección debe ser con el siguiente directorio: `router.push(`/reports/${reportId}`);`”



Figura 13: Pantalla de reportes generada mediante *prompt*
Fuente: Gemini (3.1 Pro Avanzado)

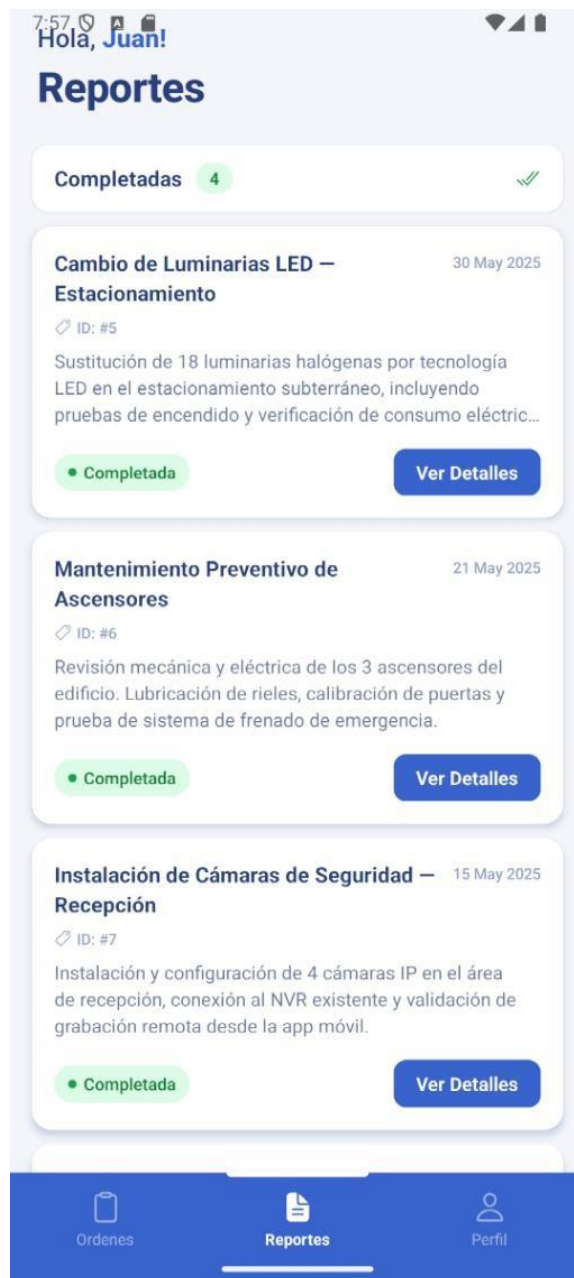


Figura 14: Pantalla de reportes generada mediante *prompt*
Fuente: Claude IA (Sonnet 4.6 *max effort*)



Figura 15: Pantalla de reportes generada mediante *prompt*
Fuente: Microsoft Copilot (Razonamiento Profundo)

3.2.5 PANTALLA DETALLES DE REPORTE

Esta pantalla tiene como objetivo principal desplegar los detalles de las órdenes ya finalizadas, permitiendo verificar la ejecución del trabajo y revisar sus respectivas evidencias. Dado que la interfaz se alimenta con datos simulados exclusivos para la vista previa, no se visualizarán fotografías ni firmas reales. Sin embargo, el diseño deberá estructurar y delimitar claramente las áreas o contenedores visuales donde se alojará dicha información en el entorno de producción:

“Tomando las referencias con respecto a la generación de pantalla acerca de los detalles de la orden. Genera la siguiente pantalla con las siguientes diferencias:

- El directorio correspondiente está ubicado en `app/reports/[reportId]`
- Se debe generar un bottom tab bar con 2 iconos: El de Reporte (con un icono de hoja de papel) y el de Rectificaciones (con un icono de portapapeles)
- Se deben mostrar los siguientes datos: El título del reporte finalizado (la cual es el título de la orden), la Id del reporte, el contenido (o la información del reporte), el responsable de la tarea, el correo del responsable y el Estado, junto a la firma y la evidencia fotográfica, este contenido NO puede ser editado debido a que la tarea esta completada, la pantalla exclusivamente muestra la información de la tarea finalizada
- Aquí tienes el *json* para que tengas referencia de los datos

```
{  
  "createdAt": "", "id": "", "organizationId": "", "role": "",  
  "user": { "email": "", "id": "", "image": null, "name": "", "role": "" },  
  "userId": ""  
}
```



Figura 16: Pantalla de detalles de reporte generada mediante *prompt*
Fuente: Gemini IA (3.1 Pro Avanzando)

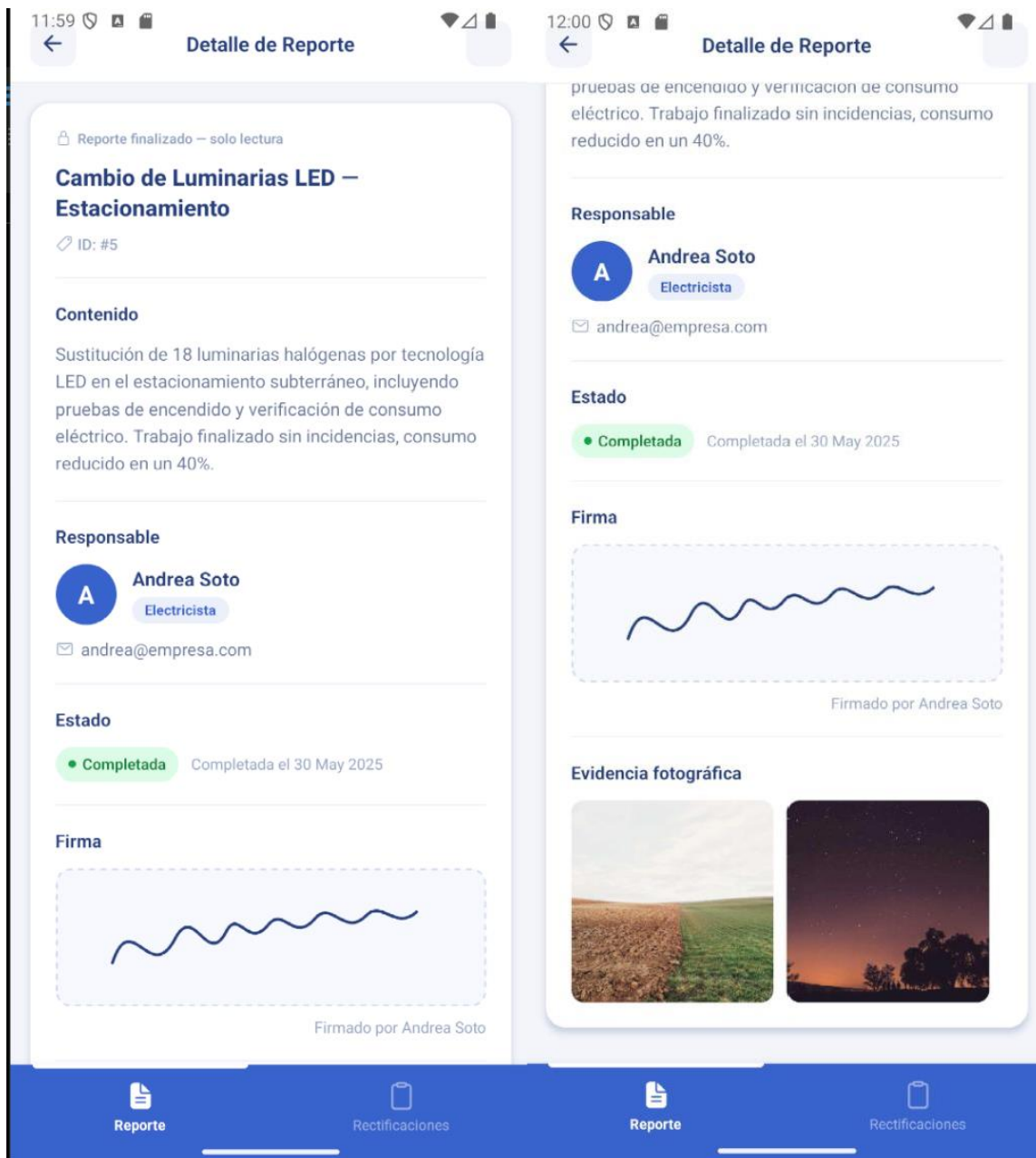


Figura 17: Pantalla de detalles de reporte generada mediante *prompt*
Fuente: Claude IA (Sonnet 4.6 max effort)



Figura 18: Pantalla de detalles de reporte generada mediante *prompt*
Fuente: Microsoft Copilot (Razonamiento Profundo)

3.2.6 PANTALLA RECTIFICACION DE REPORTE

Para concluir con el módulo de reportes, se procede a la generación de la pantalla de rectificaciones. La finalidad de esta interfaz es habilitar un espacio de edición estructurado para subsanar posibles discrepancias en los datos de un trabajo ya finalizado. La instrucción se enfocará en generar una interfaz clara con un enfoque visual similar al de una ventana de chat; no obstante, su comportamiento operativo estará destinado a ser un *log* informativo, mostrando de forma sucesiva cada rectificación realizada como un registro histórico:

“Tomando como referencia las instrucciones utilizadas para la creación de la pantalla de chat. Genera una pantalla de rectificaciones con las siguientes diferencias:

- La pantalla está ubicada en la dirección `apps/reports/[reportId]/rectification`
- Aquí únicamente se mantiene un registro tipo bitácora sobre que arreglos se hacen
- Cada rectificación debe mostrar:
 - 1) Nombre del responsable
 - 2) Rol del responsable
 - 3) Contenido de la Rectificación
 - 4) Fecha y hora del registro
- El contenido de cada rectificación se debe mostrar dentro de una tarjeta blanca con bordes redondeados y una sombra para mantener consistencia con el resto de la aplicación
- La lista debe ser vertical, mostrando las rectificaciones más recientes abajo
- Debe haber una ventana para seguir escribiendo rectificaciones abajo, cumpliendo la misma función que la pantalla de chat”



Figura 19: Pantalla de rectificaciones de reporte generada mediante *prompt*
Fuente: Gemini IA (3.1 Pro Avanzado)



Figura 20: Pantalla de rectificaciones de reporte generada mediante *prompt*
Fuente: Claude IA (Sonnet 4.6 *max effort*)



Figura 21: Pantalla de rectificaciones de reporte generada mediante *prompt*
Fuente: Microsoft Copilot IA (Razonamiento Profundo)

3.2.7 PANTALLA DE PERFIL DE USUARIO

Finalmente, para concluir con las pantallas de la aplicación móvil, se procederá a generar la última interfaz, correspondiente al perfil de usuario. Se trata de una pantalla básica y simple que cuenta con una cantidad reducida de botones, diseñada primordialmente para representar cómo se visualizaría esta sección dentro del sistema. Manteniendo la coherencia metodológica de las fases previas:

“Manteniendo el formato de las pantallas en cuanto a colores y estructura de las tarjetas, genera una pantalla de perfil de usuario con las siguientes características, solo enfocado en UX/UI:

- Este directorio está ubicando en app/(tabs), correspondiendo a la pantalla de profile.tsx

- Debe contener un *header* que diga Perfil de Usuario, procura que no esté tan arriba de la pantalla para que se pueda visualizar de buena manera

- Se debe mostrar una tarjeta que contenga la información de usuario (debe estar al centro de la pantalla), dicha información debe contener (mantener el orden que se menciona a continuación):

- 1) Foto de Perfil Circular, en Caso de no existir foto, se debe mostrar la inicial del nombre de usuario. Para cambiar la foto de perfil se debe tocar la imagen de perfil circular

- 2) Nombre completo

- 3) Correo Electrónico

- 4) Rol correspondiente

- La tarjeta debe contener botones como:

- 1) Cuenta

- 2) Ajustes

- 3) Accesibilidad

- 4) Cerrar Sesión “

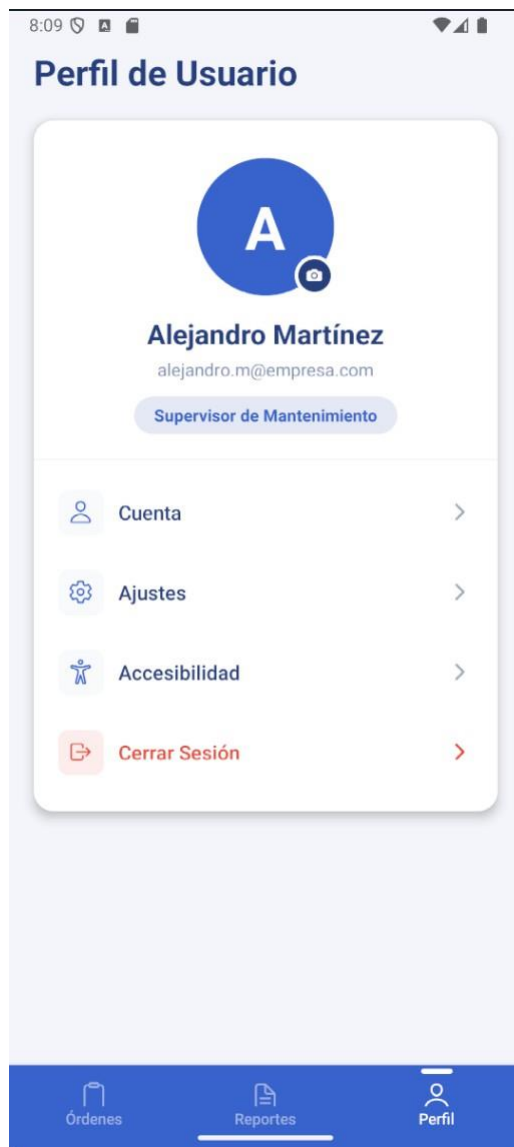


Figura 22: Pantalla del perfil de usuario generada mediante *prompt*
Fuente: Gemini IA (3.1 Pro Avanzado)

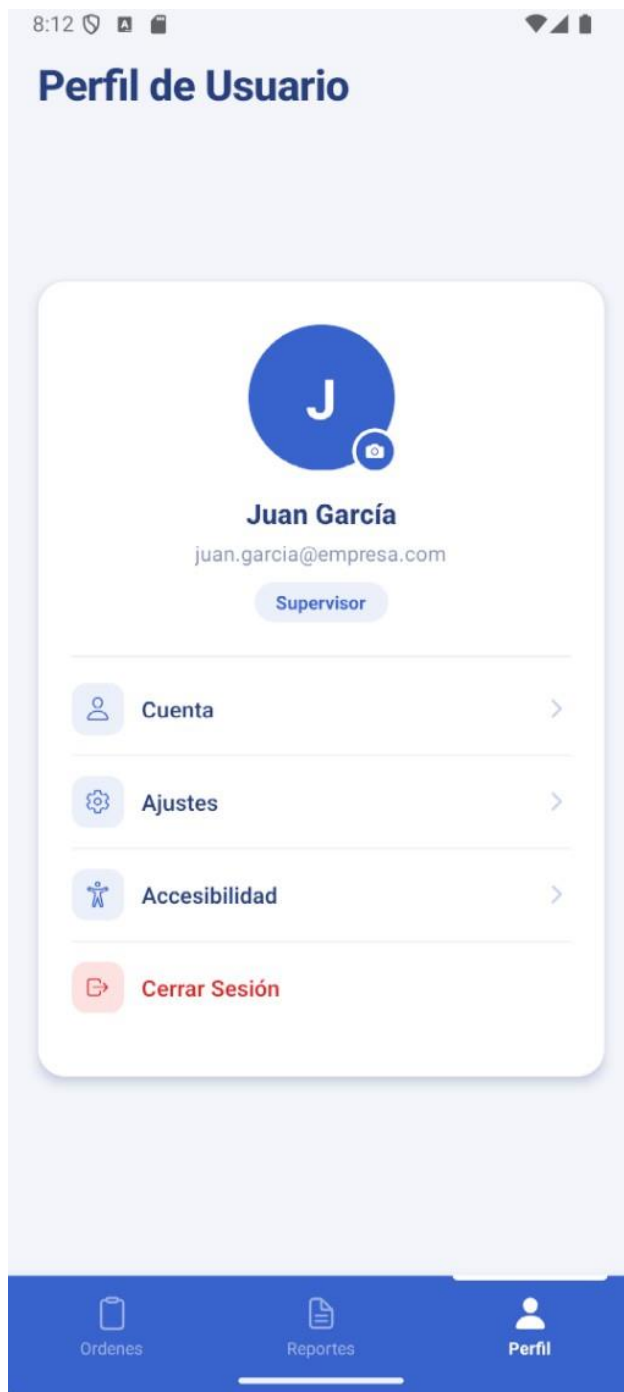


Figura 23: Pantalla del perfil de usuario generada mediante *prompt*
Fuente: Claude IA (Sonnet 4.6 *max effort*)

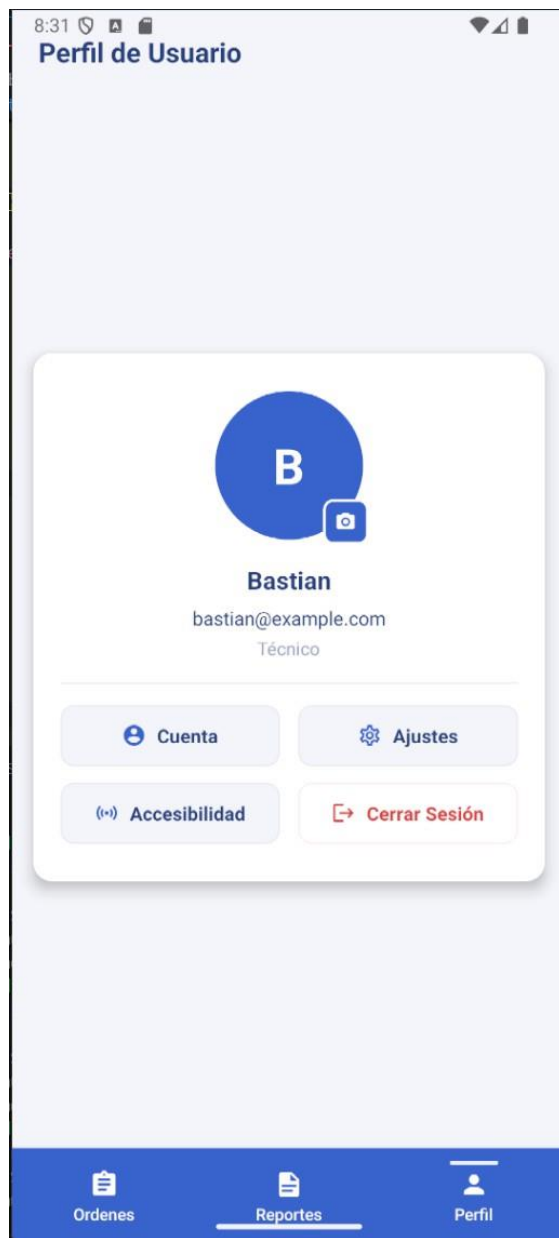


Figura 24: Pantalla del perfil de usuario generada mediante *prompt*
Fuente: Microsoft Copilot (Razonamiento Profundo)

CAPITULO 4: VALIDACION DE LA SOLUCIÓN

4.1 INTRODUCCIÓN

En este capítulo se evaluará el rendimiento y la calidad de cada inteligencia artificial basándose en las pantallas generadas por los *prompts*, verificando qué tan bien se realizaron las ejecuciones con respecto a los principios de diseño mencionados en el Capítulo 2. El enfoque de este análisis no busca establecer una competencia entre las tecnologías, sino validar que estos modelos funcionan como excelentes orquestadores para la creación de prototipos. Se evidenciará que, al enfocarse de manera exclusiva en desarrollar una correcta arquitectura de las instrucciones, la inteligencia artificial puede materializar interfaces gráficas que cumplan con todos los estándares requeridos por la industria.

4.2 CRITERIO DE EVALUACIÓN

Con el fin de medir la efectividad de las pantallas resultantes, el proceso de validación se centrará en la capacidad de las inteligencias artificiales para interpretar y ejecutar con precisión la arquitectura de los *prompts*. El análisis se realizará de forma modular (cada tabla corresponderá a un módulo), evaluando qué tan apegados fueron los diseños finales a los requerimientos técnicos y a los principios de interfaz de usuario detallados previamente. De esta manera, se medirá cualitativamente el grado de cumplimiento de cada modelo frente a las restricciones visuales y operativas del proyecto.

Para llevar a cabo esta medición cualitativa, se ha establecido una escala de cuatro niveles. Esta permite separar el cumplimiento funcional básico de la excelencia visual y arquitectónica:

- **Cumplimiento total:** El modelo asimiló el *prompt* de manera exacta. El código generado es funcional a nivel interfaz (botones, navegación, campos de entrada, firmas), respeta todas las restricciones técnicas y cumple con los principios de diseño (simplicidad, consistencia, jerarquía y estética)
- **Cumplimiento alto:** El modelo generó una interfaz con una ejecución técnica y visual excelente, respetando la mayoría de los principios de diseño y requerimientos del *framework*. Sin embargo, presentó ligeras desviaciones (ligeras alteraciones en el color, error en márgenes o pérdida leve del contexto).
- **Cumplimiento medio:** El modelo logró estructurar los requisitos funcionales y técnicos básicos (generación de botones, contenedores, etc.) pero su ejecución visual es deficiente. La interfaz presenta falencias notorias en los principios de jerarquía visual, simplicidad o estética, generando un diseño saturado o poco profesional.

- **Cumplimiento bajo:** El modelo fracasó tanto a nivel técnico como visual. La inteligencia artificial ignoró instrucciones clave del *prompt*, generó componentes inexistentes o rompió la estructura del código de forma crítica, requiriendo una reescritura de la instrucción.

Tabla 1: Evaluación de la Inteligencias Artificiales en la generación de pantallas acerca del módulo de “Órdenes”

Fuente: Elaboración Propia

Modulo Órdenes	Gemini (3.1 Pro Avanzado)	Claude (Sonnet 4.6 Max Effort)	Microsoft Copilot (Razonamiento Profundo)	Observaciones
Listado de Órdenes	Cumplimiento Total	Cumplimiento Alto	Cumplimiento Alto	Todas las IAs cumplieron correctamente la instrucción solicitada, sin embargo, Claude y Copilot tuvieron problemas con <i>SafeAreaView</i> , generando un <i>header</i> muy cerca en la parte superior de la pantalla
Detalle de Orden	Cumplimiento Total	Cumplimiento Total	Cumplimiento Bajo	Microsoft Copilot tuvo problemas identificando las actividades con el estado de la actividad, además de no cumplir con la consistencia entre título y la descripción del trabajo. Además de que, dentro de la sección de firma, la pantalla se movía junto al dedo, haciendo imposible crear firmas.
Chat de la Orden	Cumplimiento Total	Cumplimiento Total	Cumplimiento Alto	Microsoft Copilot presenta continuos problemas de consistencia o de la posición de los <i>headers</i>

Tabla 2: Evaluación de la Inteligencias Artificiales en la generación de pantallas acerca del módulo de “Reportes”

Fuente: Elaboración Propia

Modulo Reportes	Gemini (3.1 Pro Avanzado)	Claude (Sonnet 4.6 Max Effort)	Microsoft Copilot (Razonamiento Profundo)	Observaciones
Listado de Reportes	Cumplimiento Total	Cumplimiento Alto	Cumplimiento Medio	A pesar de que Claude y Microsoft Copilot tengan el mismo error sobre el header con la pantalla y exista ligero problema de jerarquía visual con respecto al botón de “Ver detalles” siendo un botón pequeño, Claude lo compensa agregando más detalles, haciéndolo más atractivo a la vista sin llegar a saturar, en cambio Copilot no llega aprovechar el espacio que se le brinda, dejando una sensación de vacío
Detalle de Reporte	Cumplimiento Total	Cumplimiento Total	Cumplimiento Medio	Microsoft Copilot llega a requerir ayuda con la jerarquía visual, requiriendo <i>prompts</i> adicionales para corregir dichos errores.
Rectificación	Cumplimiento Alto	Cumplimiento Total	Cumplimiento Medio	Microsoft Copilot a comparación de las otras Inteligencias Artificiales, no logra generar un atractivo visual al ojo humano, además de dejar un espacio considerable en la caja de mensaje.

Tabla 3: Evaluación de la Inteligencias Artificiales en la generación de pantallas acerca del módulo de “Perfil”

Modulo Perfil	Gemini (3.1 Pro Avanzado)	Claude (Sonnet 4.6 Max Effort)	Microsoft Copilot (Razonamiento Profundo)	Observaciones
Perfil de Usuario	Cumplimiento Total	Cumplimiento Total	Cumplimiento Medio	Claude y Gemini a pesar de tener en diferente posición el recuadro de perfil, ambos mantienen una estética visual atractiva. En cambio, la posición de los botones que genero Microsoft Copilot, se siente muy compacto el espacio, además de no remarcar el rol correspondiente, se llega a sentir vacío

4.3 RESULTADOS DE LA VALIDACIÓN

La validación de los módulos evidenció una disparidad en la capacidad de razonamiento visual de los modelos evaluados. Mientras Gemini y Claude asimilaron de forma más eficiente los principios teóricos de diseño, destacando en jerarquía y estética, Microsoft Copilot presento dificultades en la distribución espacial y el aprovechamiento de la interfaz. Sin embargo, la arquitectura estructurada de los *prompts* demuestra ser fundamental para mitigar alucinaciones y estandarizar la funcionalidad del código. A pesar del alto nivel de cumplimiento general en la generación de las pantallas, se detectan limitaciones técnicas transversales en todas las inteligencias artificiales: el uso ineficiente del componente SafeAreaView, que generó sistemáticamente *headers* mal posicionados. Estos hallazgos demuestran que, aunque la metodología es eficiente para acelerar la maquetación de las pantallas de la aplicación móvil, la supervisión técnica humana sigue siendo importante para refinar los márgenes nativos y la usabilidad final de la aplicación.

CAPITULO 5: CONCLUSIONES

5.1 APRENDIZAJE

A lo largo del desarrollo de este proyecto, se evidenció que las herramientas de inteligencia artificial pueden ser de gran utilidad si se utilizan de una manera adecuada. Su verdadero potencial no reside únicamente en su capacidad de generar código, sino en la correcta estructuración de las instrucciones por parte del usuario. El impacto de aplicar una buena arquitectura de *prompts* quedó demostrado al acortar los tiempos de desarrollo considerablemente: la creación de las pantallas, que de manera manual tuvo una duración aproximada de tres meses, se redujo a tan solo dos semanas, incluyendo el propio proceso de creación y refinamiento del *prompt*. Este aprendizaje confirma que, al guiar a estas herramientas con precisión, no solo se optimiza drásticamente el tiempo de trabajo, sino que se logran pantallas con una mayor calidad visual y un cumplimiento estricto de los principios de diseño.

5.2 LIMITACIONES

Una de las principales limitantes durante el proceso de creación de las pantallas fue la restricción impuesta por las versiones de prueba de las herramientas de inteligencia artificial. Al optar por los modelos más avanzados disponibles en sus modalidades gratuitas, el proyecto estuvo sujeto a tiempos de espera obligatorios entre la ejecución de cada *prompt*. Este factor fue especialmente crítico con Claude; a pesar de la gran calidad que ofrecía en sus resultados, su límite de uso restringía la generación a una pantalla cada cinco horas. En consecuencia, un *prompt* mal estructurado causaba retrasos significativos en el tiempo de desarrollo, limitando el flujo de trabajo a un máximo de tres pantallas por día y exigiendo un alto nivel de precisión en las instrucciones desde el primer intento.

Por otro lado, una limitante técnica considerable fue la necesidad de desacoplar y omitir temporalmente funcionalidades importantes de la aplicación, como la mensajería asíncrona y la persistencia de los datos en el servidor. Esta situación derivó de la pérdida de los accesos de administrador al entorno original, debido a la desarticulación del equipo de desarrollo inicial. Al no contar con las credenciales del *backend*, la extracción de datos reales se volvió inviable, lo que obligó a simular la información mediante datos de prueba estáticos. Esto limitó el alcance del prototipo a un entorno estrictamente visual, enfocando la validación exclusivamente en la maquetación de las interfaces y no en su integración con la base de datos.

5.3 VISION A FUTURO DE LAS INTELIGENCIAS ARTIFICIALES

Existe un debate recurrente en la industria tecnológica sobre la posibilidad de que la inteligencia artificial reemplace a los desarrolladores, justificando esta premisa en su velocidad para generar código y en su aparente capacidad de razonamiento lógico. Sin embargo, la realidad operativa ha demostrado ser muy diferente. Más que un sustituto, la inteligencia artificial se ha consolidado como una herramienta de asistencia fundamental. Un ejemplo claro de este cambio de paradigma, como se planteó al inicio de esta memoria, es el surgimiento de entornos de desarrollo nativos de IA como Cursor. Estas plataformas evidencian que, cuando la tecnología se integra directamente en el flujo de trabajo humano, posee un potencial innegable para automatizar tareas repetitivas, maquetar bases estructurales y acelerar drásticamente los ciclos de programación.

A pesar de estas ventajas, dejar el control absoluto a estas herramientas conlleva riesgos importantes. Incluso en entornos como Cursor, las inteligencias artificiales no tienen una visión completa de todo el proyecto ni entienden el contexto del negocio a fondo. Por esto, una instrucción mal hecha o la falta de supervisión pueden causar problemas graves de arquitectura, errores de funcionalidad, o incluso llegar a romper el código original que el programador ya había estructurado.

Esto demuestra que el rol del desarrollador no va a desaparecer, sino que simplemente está cambiando. Apoyado en estas nuevas tecnologías, el programador deja de ser solo alguien que escribe líneas de código para convertirse en un supervisor o director del proyecto. Su criterio lógico y su experiencia técnica siguen siendo indispensables para guiar a la inteligencia artificial, corregir sus errores y asegurar que el sistema final funcione correctamente.

REFERENCIAS BIBLIOGRAFICAS

1. **Forbes Argentina.** (2026). *Los agentes IA cambian el desarrollo de software y amenazan a los gigantes del sector.* Forbes Argentina.
<https://www.forbesargentina.com/innovacion/los-agentes-ia-cambian-desarrollo-software-amenazan-gigantes-sector-n87365>
2. **Gartner.** (2025, 26 de Agosto). *Gartner predicts 40 percent of enterprise apps will feature task-specific AI agents by 2026, up from less than 5 percent in 2025.* Gartner. <https://www.gartner.com/en/newsroom/press-releases/2025-08-26-gartner-predicts-40-percent-of-enterprise-apps-will-feature-task-specific-ai-agents-by-2026-up-from-less-than-5-percent-in-2025>
3. **McKinsey & Company.** (2024). *The state of AI 2024.* McKinsey & Company.
<https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-2024>
4. **TechClass.** (2024). *Role of AI in reducing operational overhead costs.* TechClass.
<https://www.techclass.com/resources/learning-and-development-articles/role-of-ai-in-reducing-operational-overhead-costs>
5. **Wharton School.** (2025, 8 de September). *The projected impact of generative AI on future productivity growth.* Penn Wharton Budget Model.
<https://budgetmodel.wharton.upenn.edu/p/2025-09-08-the-projected-impact-of-generative-ai-on-future-productivity-growth/>