

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



**“PROYECTO STUTRABAJO: DESARROLLO DE
APLICACIÓN MÓVIL DE EMPLEOS ESPORÁDICOS
PARA ESTUDIANTES DE INSTITUCIONES DE
EDUCACIÓN SUPERIOR”**

FRANZ ALONSO MOLINA RIVERA

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

Profesor Guía: Cristian Antonio Lara Valenzuela

Julio - 2025



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: PROYECTO STUTRABAJO: DESARROLLO DE APLICACIÓN MÓVIL DE EMPLEOS ESPORÁDICOS PARA ESTUDIANTES DE INSTITUCIONES DE EDUCACIÓN SUPERIOR

Nombre del candidato(a): FRANZ ALONSO MOLINA RIVERA

Carrera / Grado: INGENIERIA EN INFORMÁTICA

Campus: Concepcion ; Departamento: INFORMÁTICA

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Cristian Antonio Lara Valenzuela, 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

☒ 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:

☐ 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:

CRISTIAN LARA VALENZUELA
Firmado digitalmente por
CRISTIAN LARA VALENZUELA
Fecha: 2025.07.24 18:19:03
-04'00'

Fecha: 24-07-2025 ; Firma: _____

Estudiante o Candidato(a):

Fecha: 24-07-2025 ; 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.

BIB-FO2385 01

DEDICATORIA

A todos aquellos que me han acompañado en este largo camino, por los días y noches de esfuerzo, por las cosas que me fortalecieron, y por los momentos que me impulsaron a seguir adelante.

A mis padres, quienes, con su apoyo incondicional y sacrificios, me han brindado la oportunidad de llegar hasta acá. A mi madre, que, durante toda mi vida, su amor y fortaleza han sido mi faro en los momentos más desafiantes

A mis hermanos y cuñadas, por ser pilares de aliento constante, y a mis sobrinos, por regalarme la alegría que alimenta el corazón.

A mis amigos, por su compañía en los buenos y malos momentos, y a mi polola, cuyo respaldo, amor y confianza han sido esenciales en este proceso.

Todo es por ustedes, quienes de una forma u otra han sido parte de toda esta trayectoria.

AGRADECIMIENTOS

Quiero expresar mi más profundo agradecimiento a todas las personas que hicieron posible la realización de este proyecto.

A los docentes de la Universidad Técnica Federico Santa María, por su ayuda constante, sus valiosas observaciones técnicas y metodológicas, y por su compromiso en guiar este proyecto.

A los estudiantes que participaron en las encuestas y entrevistas, quienes con su disposición y honestidad contribuyeron significativamente al desarrollo de este proyecto, permitiéndome comprender mejor sus necesidades y desafíos.

A mi equipo de trabajo, por su dedicación, compromiso y esfuerzo constante. Su colaboración no solo facilitó el avance del proyecto, sino que enriqueció cada aspecto del desarrollo con su profesionalismo y creatividad.

Finalmente, a todas las personas que directa o indirectamente ofrecieron su apoyo y motivación, su ayuda fue fundamental para superar los desafíos y alcanzar este logro. A todos ustedes, mi sincero agradecimiento.

RESUMEN

Este proyecto aborda la problemática de la deserción estudiantil en Instituciones de Educación Superior en Chile, causada por dificultades económicas, mediante el desarrollo de una aplicación móvil de empleos temporales llamada *StuTrabajo*. La aplicación está diseñada para conectar a estudiantes con empleos temporales, promoviendo su permanencia en los estudios sin presentar dificultades financieras. El desarrollo se realizó utilizando tecnologías modernas como React Native para la interfaz móvil y un backend basado en microservicios desplegados en Django. Los resultados obtenidos demuestran que la plataforma es capaz de reducir significativamente el tiempo de búsqueda de empleos temporales y aumentar las oportunidades laborales para los estudiantes. Este proyecto no solo proporciona una herramienta tecnológica, sino que también contribuye al bienestar de los estudiantes, ayudándoles a equilibrar sus responsabilidades académicas y laborales, y fomentando su desarrollo personal y profesional.

Palabras Clave: empleos temporales; deserción estudiantil; estudiantes; react native; microservicios.

ABSTRACT

This project addresses the problem of student dropout in higher education institutions in Chile, caused by economic difficulties, through the development of a mobile application for temporary jobs called *StuTrabajo*. The application is designed to connect students with temporary jobs, promoting their retention in studies without facing financial difficulties. The development was carried out using modern technologies such as React Native for the mobile interface and a backend based on microservices deployed with Django. The results demonstrate that the platform significantly reduces the time required to search for temporary jobs and increases employment opportunities for students. This project not only provides a technological tool but also contributes to students' well-being by helping them balance their academic and work responsibilities while fostering their personal and professional development.

Keywords: temporary jobs; students; student dropout; react native; microservices.

GLOSARIO

AWS: Amazon Web Services

CV: Currículum Vitae

GUI: Graphical User Interface

MINEDUC: Ministerio de Educación

MVC: Modelo Vista Controlador

OCDE: Organización para la Cooperación y el Desarrollo Económico.

ORM: Object-Relational Mapping

SDLC: Ciclo de Vida del Desarrollo de Software

SIES: Servicio de Información de Educación Superior

SQL: Structured Query Language

SRP: Principio de Responsabilidad única

UI/UX: User Interface/User Experience

UTFSM: Universidad Técnica Federico Santa María.

INDICE DE CONTENIDOS

RESUMEN.....	4
ABSTRACT	4
INDICE DE FIGURAS.....	9
INDICE DE TABLAS.....	10
INTRODUCCIÓN	11
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	12
1.1 Organización y contexto de trabajo	12
1.2 Situación actual	12
1.3 Actores o usuarios involucrados.....	13
1.4 Justificación y motivación.....	14
1.5 Competencia.....	15
1.6 Objetivos.....	17
1.6.1 Objetivo general	17
1.6.2 Objetivos específicos	17
CAPÍTULO 2: MARCO CONCEPTUAL	18
2.1 Metodología de desarrollo de software.....	18
2.2 Patrón de arquitectura para el desarrollo.....	20
2.3 Metodología SCRUM	21
2.3.1 Roles en Scrum	21
2.3.2 Eventos en Scrum	22
2.3.3 Artefactos en Scrum	24
2.4 Fundamentos del Desarrollo de Software.....	25

2.4.1	Ciclo de Vida del Desarrollo de Software (SDLC).....	25
2.5	Frameworks y herramientas para desarrollo móvil	27
2.6	Consideraciones y buenas prácticas en el desarrollo de aplicaciones móviles ...	30
2.6.1	Recolección y análisis de requisitos	31
2.6.2	Diseño centrado en el usuario: Estándares UI/UX	32
2.6.3	Estándares técnicos en arquitectura y desarrollo	34
2.6.4	Seguridad en aplicaciones móviles.....	35
2.6.5	Pruebas, control de versiones y mantenimiento	37
2.6.6	Conclusiones sobre estándares en desarrollo móvil	40
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN		41
3.1	Descripción general de la solución	41
3.2	Funcionalidades de la aplicación	42
3.2.1	Requerimientos funcionales.....	43
3.2.2	Requerimientos no funcionales.....	44
3.3	Usuarios dentro del sistema	44
3.4	Diagrama de Casos de Uso	47
3.5	Casos de uso detallados	48
3.6	Modelo de Datos	57
3.7	Diseño preliminar de interfaz de Software	58
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		68
4.1	Métodos de validación	69
4.1.1	Pruebas funcionales	69
4.1.2	Pruebas de usabilidad y aceptación de usuarios.....	72
4.1.3	Pruebas de comparación con otras plataformas	75

CAPÍTULO 5: CONCLUSIONES	77
5.1 Aprendizaje.....	77
5.2 Recomendaciones.....	78
5.3 Conclusión	79
REFERENCIAS BIBLIOGRÁFICAS.....	80

INDICE DE FIGURAS

<i>Figura 1: Visualización encuesta a estudiantes.....</i>	<i>14</i>
<i>Figura 2: Metodología de desarrollo de una aplicación móvil en MVC.....</i>	<i>20</i>
<i>Figura 3: Visualización Sprints Estudiantes.</i>	<i>22</i>
<i>Figura 4: Visualización Sprints Solicitantes.</i>	<i>23</i>
<i>Figura 5: Conexión movil – servidor.....</i>	<i>42</i>
<i>Figura 6: Diagrama de Casos de Uso.....</i>	<i>47</i>
<i>Figura 7: Modelo de Datos - StuTrabajo.....</i>	<i>57</i>
<i>Figura 8 – Interfaz “Home”.....</i>	<i>59</i>
<i>Figura 9 – Interfaz “Login”.....</i>	<i>59</i>
<i>Figura 10 – Interfaz “Crear cuenta”.....</i>	<i>60</i>
<i>Figura 11 – Interfaz “Perfil”.....</i>	<i>60</i>
<i>Figura 12 – Interfaz “Ofertas”.....</i>	<i>61</i>
<i>Figura 13 – Interfaz “Detalles empleo”.....</i>	<i>61</i>
<i>Figura 14 – Interfaz “Mis Pegas”.....</i>	<i>62</i>
<i>Figura 15 – Interfaz “Match”.....</i>	<i>63</i>
<i>Figura 16 – Interfaz “Chat”.....</i>	<i>63</i>
<i>Figura 17 – Interfaz “Mis publicaciones”.....</i>	<i>64</i>
<i>Figura 18 – Interfaz “Crear empleo”.....</i>	<i>64</i>
<i>Figura 19 – Interfaz “Ver postulantes”.....</i>	<i>65</i>
<i>Figura 20 – Interfaz “Presentación perfil”.....</i>	<i>65</i>
<i>Figura 21 – Interfaz “Valoración Estudiante”.....</i>	<i>66</i>
<i>Figura 22 – Interfaz “Valoración Solicitante”.....</i>	<i>66</i>

INDICE DE TABLAS

<i>Tabla 1: Comparación de metodologías de desarrollo frente a SCRUM.</i>	<i>19</i>
<i>Tabla 2: Caso de Uso - Registrar Cuenta.</i>	<i>48</i>
<i>Tabla 3: Caso de Uso - Iniciar Sesión.</i>	<i>49</i>
<i>Tabla 4: Caso de Uso - Crear/Editar Perfil</i>	<i>50</i>
<i>Tabla 5: Caso de Uso - Buscar empleo.....</i>	<i>51</i>
<i>Tabla 6: Caso de Uso - Postular empleo.</i>	<i>52</i>
<i>Tabla 7: Caso de Uso - Calificar Perfil.</i>	<i>53</i>
<i>Tabla 8: Caso de Uso - Chat.</i>	<i>54</i>
<i>Tabla 9: Caso de Uso - Publicar empleo.</i>	<i>55</i>
<i>Tabla 10: Caso de Uso - Gestionar postulaciones.....</i>	<i>56</i>
<i>Tabla 11: Pruebas funcionales detalladas.</i>	<i>70</i>
<i>Tabla 12: Criterios para realizar comparaciones.....</i>	<i>75</i>
<i>Tabla 13: Comparación con otras plataformas.</i>	<i>76</i>

INTRODUCCIÓN

La deserción estudiantil en las Instituciones de Educación Superior en Chile es un problema poco discutido que afecta tanto a los estudiantes como a las instituciones. Diversos estudios realizados por el Servicio de Información de Educación Superior (SIES)¹, el Ministerio de Educación (MINEDUC) y Psicomatrix² indican que, en el año 2024, 1.385.828 estudiantes se matricularon en programas de Pregrado, Posgrado y Postítulo. De estos, el 21% aproximadamente abandona sus estudios por motivos económicos, vinculados a la falta de recursos para cubrir necesidades básicas como alojamiento, alimentación, transporte, matrícula, arancel, etc. Este problema surge de la incompatibilidad entre las responsabilidades académicas y laborales, lo que obliga a muchos estudiantes a abandonar sus estudios para trabajar. Además, aquellos que intentan equilibrar estudio y trabajo a menudo ven afectado negativamente su desarrollo académico y futuro profesional.

El presente proyecto aborda esta problemática mediante el desarrollo de un portal de empleos de corto periodo de tiempo exclusivamente para estudiantes de Educación Superior, que facilita la búsqueda de trabajos temporales adecuados a sus necesidades y disponibilidad. Esta solución no solo busca proporcionar una fuente de ingresos para los estudiantes, sino también apoyar su éxito académico y profesional a largo plazo.

Se basó en un enfoque centrado en los usuarios del portal, comenzando con un análisis exhaustivo del problema para identificar los principales desafíos y necesidades de los estudiantes. Esto incluirá un estudio de las competencias existentes y la definición clara de objetivos y alcances. A continuación, abordaremos la parte conceptual, describiendo conceptos técnicos, metodologías de trabajo y herramientas que se utilizarán en la construcción de la propuesta. Posteriormente, presentaremos la propuesta de solución, detallando los elementos clave que conformarán la plataforma. Incluiremos también un plan de validación para demostrar la efectividad y pertinencia de la solución para nuestros usuarios. Finalmente, concluiremos con una reflexión sobre los aprendizajes obtenidos a lo largo del trabajo realizado y las implicaciones futuras para la continuidad y mejora de la plataforma.

¹ <https://www.mifuturo.cl/sies/>

² <https://psicomatrix.cl/abandono-de-los-estudios-datos-sobre-la-desercion-universitaria-en-chile/>

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1 Organización y contexto de trabajo

El proyecto se desarrollará en el contexto de la plataforma **StuTrabajo**, una iniciativa destinada a brindar apoyo financiero a estudiantes de Educación Superior en Chile. La organización detrás de este proyecto está comprometida con la misión de reducir la deserción estudiantil por motivos económicos, facilitando a los estudiantes la posibilidad de generar ingresos que les permitan cubrir necesidades básicas como alojamiento, alimentos, transporte, matrícula, arancel, etc.

El proyecto se enmarca en varias líneas de la informática, destacándose el desarrollo de software, ciberseguridad, inteligencia artificial, aprendizaje automático, bases de datos, sistemas de información, experiencia de usuario, entre otras.

1.2 Situación actual

Actualmente, los estudiantes suelen recurrir a diferentes métodos para encontrar empleo, entre ellos:

- Portales de empleo genéricos: Utilizan plataformas que no están específicamente diseñadas para estudiantes y no siempre se ofrecen empleos con la flexibilidad necesaria.
- Redes sociales y preferencias: Se apoyan en redes sociales y recomendaciones de amigos o familiares para encontrar trabajos temporales, aunque esta estrategia no siempre resulta efectiva ni fiable.
- Ferias de empleo: Participación en ferias de empleo organizadas por las mismas instituciones o entidades externas, aunque las oportunidades son demasiado limitadas y no siempre se ajustan a sus necesidades de tiempo.
- Bolsas de trabajo universitarias: Algunas instituciones ofrecen bolsas de trabajo, pero estas suelen ser limitadas de alcance ya que existen pocos cupos para poder postular y no hay mucha diversidad de ofertas.

El proceso que los estudiantes siguen para encontrar empleos típicamente incluye:

- Búsqueda de empleo: buscan activamente empleos en múltiples plataformas, lo que puede ser un proceso largo y agotador.
- Aplicación y envío de CV: Una vez que encuentran la oportunidad, hacen envío de su currículum y esperar una respuesta, lo cual puede tardar días o incluso semanas.

- Entrevistas y selección: Si son seleccionados, pasan por entrevistas que pueden ser presenciales o virtuales, requiriendo una inversión adicional de tiempo.
- Negociación de horarios: Muchos estudiantes deben negociar los horarios con los empleadores para asegurarse de que no interfieran con sus clases, lo cual no siempre es posible y provoca que el estudiante pierda la oportunidad de trabajar.

1.3 Actores o usuarios involucrados

Aquellas actores o usuarios que están involucrados en esta problemática son principalmente los estudiantes de Instituciones de Educación Superior y empleadores de Chile.

Los estudiantes buscan empleos temporales y flexibles que les permitan generar ingresos sin interferir con sus estudios. Actualmente, enfrentan diversas dificultades, como la falta de flexibilidad en los empleos disponibles, la escasa disponibilidad de plataformas especializadas en empleos temporales, y el largo y agotador proceso de búsqueda de empleo. Además, la inseguridad sobre la confiabilidad de los empleadores y la calidad de los empleos es una preocupación constante. Según datos del MINEDUC³ de Chile, hay aproximadamente 1.3 millones de estudiantes de Educación Superior en el país y según la OCDE (2019)⁴, cerca del 40% de los estudiantes chilenos trabajan mientras estudian, lo que representa alrededor de 520,000 estudiantes.

Por otra parte, tenemos a los empleadores, que incluyen empresas, organizaciones y particulares que ofrecen empleos temporales y flexibles. Estos empleadores enfrentan dificultades para encontrar estudiantes que se ajusten a las necesidades específicas del empleo, y el proceso de reclutamiento y selección puede ser largo y consume mucho tiempo. Además, la falta de información y confianza en la calidad de los candidatos es una barrera significativa. La encuesta de la Cámara de Comercio de Santiago (2021) reveló que un 35% de las empresas en Chile contrata estudiantes para empleos temporales, lo que representa un mercado considerable para una plataforma que facilite este proceso.

³ <https://www.mineduc.cl/educacion-superior-crecio-33-alcanzando-14-millones-de-estudiantes/>

⁴ https://www.chile.gob.cl/ocde/site/docs/20211116/20211116163546/2019_t3_julio_agosto_sept_best_ocde.pdf

1.4 Justificación y motivación

Si en el corto o mediano plazo no se solucionan las dificultades que enfrentan los estudiantes de Educación Superior en Chile para encontrar empleos temporales y flexibles, las consecuencias podrían ser significativas tanto a nivel individual como social.

A nivel individual, uno de los impactos más críticos es la posible deserción. Los estudiantes que no pueden equilibrar sus necesidades económicas con sus responsabilidades académicas pueden verse obligados a abandonar sus estudios. Según datos del Ministerio de Educación de Chile, la deserción universitaria en el país ya es un problema considerable, con tasas que oscilan entre el 20% y el 30% en algunas instituciones. Además, la continua preocupación por la falta de ingresos puede generar altos niveles de estrés y ansiedad entre los estudiantes. Este estrés no solo afecta su bienestar emocional, sino también su rendimiento académico, creando un ciclo negativo que es difícil de romper.

12. ¿Cuáles fueron los motivos que te llevaron a pensar en abandonar tus estudios?



Figura 1: Visualización encuesta a estudiantes.

Fuente: Elaboración propia.

A nivel social, la deserción tiene un impacto directo en las instituciones de Educación Superior. Cuando los estudiantes abandonan sus estudios debido a dificultades económicas, estas instituciones sufren una reducción en sus tasas de retención y graduación, lo que afecta negativamente su reputación y acreditación. Las universidades, centros de formación técnica, institutos profesionales, dependen de las tasas de retención para mantener su financiación y apoyo gubernamental, y la deserción estudiantil puede resultar en menos recursos y oportunidades para todos los estudiantes.

Además, la falta de acceso a empleos temporales y flexibles también afecta a las instituciones de Educación Superior al limitar su capacidad para apoyar a sus estudiantes.

Los estudiantes de bajos recursos, que son los más afectados por la necesidad de trabajar mientras estudian, pueden quedar en desventaja en comparación con sus pares que no enfrentan las mismas presiones económicas. Esto genera una brecha en el rendimiento académico y en las oportunidades de éxito profesional, lo que a su vez impacta negativamente en la misión educativa de estas instituciones.

Cada estudiante que no puede continuar su educación y no puede desarrollar su potencial completo representa una pérdida significativa para las instituciones de Educación Superior en términos de innovación y contribuciones al desarrollo académico. La fuga de talento también puede reducir la capacidad de estas instituciones para atraer financiamiento para investigación y desarrollo, así como para establecer colaboraciones con la industria y otros sectores. Por lo tanto, abordar estas dificultades es esencial no solo para el bienestar de los estudiantes, sino también para la sostenibilidad y crecimiento de las instituciones educativas en el país.

La motivación para llevar a cabo este proyecto se basa principalmente en contribuir a la reducción de deserción estudiantil causada por problemas económicos, equilibrar las responsabilidades académicas con lo laboral, reduciendo el estrés y mejorando el bienestar de los estudiantes y permitirles adquirir experiencia laboral valiosa, que mejorará sus perspectivas de empleo a largo plazo y su desarrollo profesional.

1.5 Competencia

En el contexto actual, existen varias soluciones y plataformas que intentan abordar el problema de la búsqueda de empleo para estudiantes de Educación Superior en Chile. Sin embargo, estas plataformas presentan limitaciones que impiden una solución efectiva y adaptada a las necesidades específicas de los estudiantes. A continuación, se presentan algunas de las principales competencias y se analiza por qué la solución propuesta por StuTrabajo sería mejor.

- Plataformas como LinkedIn, CompuTrabajo y Laborum son ampliamente utilizadas para la búsqueda de empleo en general. Aunque estos portales ofrecen una gran cantidad de ofertas laborales, no están específicamente diseñados para estudiantes y, por lo tanto, no siempre proporcionan la flexibilidad necesaria. Los empleos disponibles en estos sitios suelen requerir un compromiso de tiempo completo o fijo, lo que no se ajusta a los horarios variables y exigencias académicas de los estudiantes. Además, estos portales no suelen ofrecer mecanismos para asegurar que las ofertas sean temporales o de corto plazo, lo cual es una necesidad clave para los estudiantes.

- Los estudiantes a menudo recurren a redes sociales y referencias personales para encontrar trabajos temporales. Aunque esta estrategia puede ser útil, es inherentemente informal y no garantiza la seguridad ni la calidad de los empleos. Las ofertas encontradas a través de estos medios pueden carecer de la estructura y formalidad necesarias, y no siempre proporcionan un entorno confiable y seguro para los estudiantes.
- En otros países, se han desarrollado plataformas especializadas que abordan problemas similares. Por ejemplo, en Estados Unidos y Europa, existen plataformas como Handshake⁵ y StudentJob⁶ que se centran en conectar a estudiantes con empleos temporales y flexibles. Estas plataformas han tenido éxito al ofrecer funcionalidades específicas como algoritmos de emparejamiento, perfiles dinámicos de estudiantes y sistemas de valoración de empleadores. Sin embargo, estas soluciones están adaptadas a los contextos y necesidades de sus respectivos países, y pueden no ser directamente aplicables al contexto chileno sin adaptaciones significativas.

Aunque existen varias alternativas en el mercado, ninguna aborda de manera integral y especializada las necesidades de los estudiantes de Educación Superior en Chile. StuTrabajo se posiciona como una solución superior al ofrecer una plataforma adaptada, funcional y segura, que facilita la conexión entre estudiantes y empleadores de manera eficiente y efectiva.

⁵ <https://joinhandshake.com/>

⁶ <https://www.studentjob.es/>

1.6 Objetivos

1.6.1 Objetivo general

Desarrollar una aplicación móvil que permita a los estudiantes registrarse, crear perfiles, buscar empleos temporales y flexibles, y aplicar a ofertas de trabajo.

1.6.2 Objetivos específicos

- Analizar los procesos relacionados a bolsas de empleo de tiempo parcial para identificar requerimientos funcionales.
- Diseñar interfaces, modelos de proceso y almacenamiento, necesarios para dar soporte a los requerimientos funcionales
- Implementar una aplicación móvil de empleos temporales, según requerimientos funcionales y especificaciones de diseño
- Realizar un plan de pruebas, que de muestra de la correcta implementación de la funcionalidad.

CAPÍTULO 2: MARCO CONCEPTUAL

El marco conceptual es fundamental para definir los aspectos clave que guían el desarrollo de la aplicación móvil *StuTrabajo*. En este capítulo se describen los conceptos técnicos y metodológicos involucrados, así como las consideraciones necesarias para asegurar un diseño, desarrollo y despliegue exitosos. Este análisis incluye referencias a metodologías de desarrollo de software, principios de diseño y herramientas tecnológicas.

2.1 Metodología de desarrollo de software

Una metodología de desarrollo es una colección de procedimientos, técnicas, herramientas y un soporte documental que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está conformada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo.

No obstante, una metodología de desarrollo es mucho más que una serie de elementos, esto es porque se basa en una filosofía, distinguiéndose de los métodos 10 o de simples instrucciones, que marcan unas pautas a seguir y ya está. De esta forma, las metodologías difieren ya sea por la cantidad de fases, las técnicas de cada fase, el contenido de la fase o en su base ideológica. Todo esto se aplica, dependiendo del contexto de desarrollo, tamaño del proyecto o del equipo de trabajo, entre otros aspectos, por lo que, en el caso de los desarrollos móviles, es de vital importancia su selección, para garantizar un producto de calidad.

Metodologías de desarrollo tradicionales y ágiles son las bases de desarrollo que existen en la actualidad. La metodología tradicional tiene un enfoque, por su naturaleza, en donde se requiere hacer documentación constantemente y planificación. En contraste, una metodología ágil se basa en los eventuales cambios que ocurran en los requerimientos y no se arraiga a un control estricto de planificación.

En este proyecto, al tratarse de un proyecto realizado por 5 personas, las metodologías tradicionales para el desarrollo de software están pensadas para grandes equipos de trabajo y se adaptan de forma adecuada a lo que se debe trabajar.

Por los motivos expuestos anteriormente, y por la tendencia que se está llevando hoy en día para la creación de aplicaciones móviles, este proyecto tomará como base elemental de desarrollo una metodología SCRUM y, como plataforma tecnológica para desarrollar el proyecto siguiendo esta metodología, se hará uso del patrón de arquitectura MVC.

A continuación, se exponen algunas importantes metodologías de desarrollo con sus principales características frente a SCRUM.

Tabla 1: Comparación de metodologías de desarrollo frente a SCRUM.

Fuente: Elaboración propia.

Metodología	Descripción	Ventajas	Desventajas	Aplicabilidad
Scrum	Marco de trabajo ágil que organiza el desarrollo en ciclos cortos llamados sprints, con roles y eventos definidos.	- Entregas rápidas y frecuentes. - Alta colaboración y comunicación. - Flexibilidad ante cambios.	- Requiere compromiso total del equipo. - Puede ser desafiante para equipos no experimentados.	Proyectos con requisitos cambiantes y necesidad de entregas rápidas.
Kanban	Método ágil que se centra en visualizar el flujo de trabajo y limitar el trabajo en curso mediante un tablero.	- Mejora continua del proceso. - Flexibilidad en la gestión de tareas. - Reducción de tiempos de entrega.	- Menos estructura definida que Scrum. - Puede carecer de plazos claros.	Procesos de producción continua y equipos que buscan mejorar la eficiencia.
Waterfall (Cascada)	Metodología tradicional con fases secuenciales: análisis, diseño, implementación, pruebas y mantenimiento.	- Estructura clara y definida. - Fácil gestión de proyectos con requisitos bien entendidos.	- Poco flexible ante cambios. - Riesgo de identificar problemas en etapas tardías.	Proyectos con requisitos estables y bien definidos desde el inicio.

2.2 Patrón de arquitectura para el desarrollo

Es preciso señalar que la rama de la ingeniería en informática se preocupa por crear procesos que aseguren calidad en los programas que se realizan y esa calidad atiende a diversos parámetros que son deseables para todo desarrollo, como la estructuración de los programas o reutilización del código, lo que debe influir positivamente en la facilidad de desarrollo, el mantenimiento y la escalabilidad. Los ingenieros se dedican a estudiar de qué manera se pueden mejorar los procesos de creación de software y una de las soluciones a las que han llegado es a la arquitectura basada en capas que separan el código en función de sus responsabilidades o conceptos (Modelo-Vista-Controlador).

De forma general, MVC (Modelo-Vista-Controlador) es un patrón de arquitectura de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Esta metodología surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos, como se mencionó en el párrafo anterior.

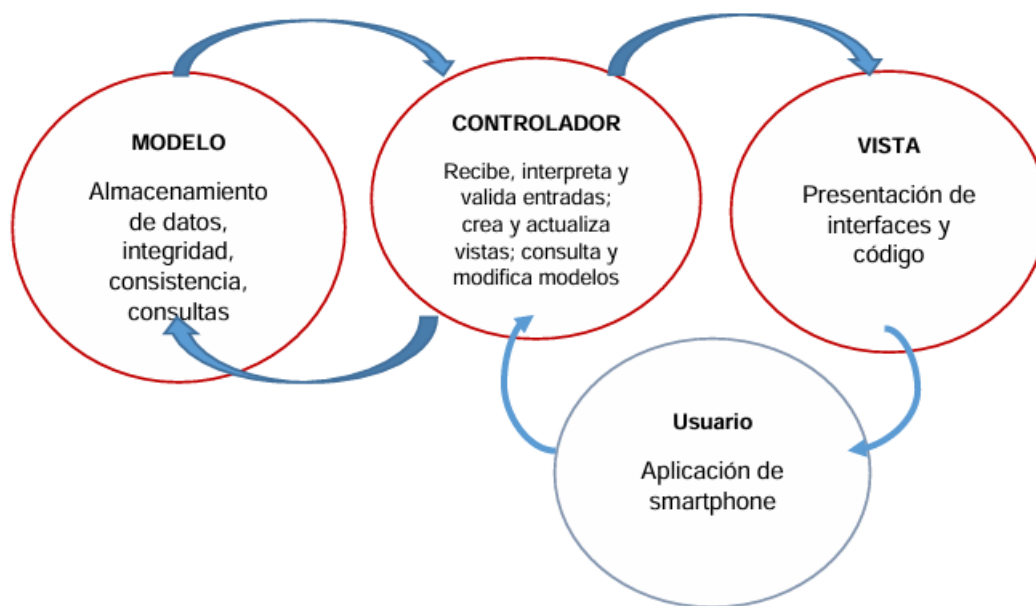


Figura 2: Metodología de desarrollo de una aplicación móvil en MVC.

Fuente: Repositorio UDLA – Jan Carlos Espinoza Paucar.

2.3 Metodología SCRUM

Scrum es un marco de trabajo (framework) dentro de las metodologías ágiles que se utiliza para desarrollar, entregar y mantener productos complejos, especialmente en el ámbito del software. Nacido de los principios ágiles, Scrum se basa en iteraciones cortas y ciclos de trabajo denominados *Sprints*, donde se busca entregar valor de manera incremental y continua al cliente.

2.3.1 Roles en Scrum

Scrum organiza los equipos y roles para asegurar una colaboración eficaz entre las partes interesadas y el equipo de desarrollo. Los roles principales son:

1. Product Owner (Dueño del Producto):

- Actúa como el puente entre el equipo de desarrollo y los interesados del proyecto.
- Se encarga de gestionar el *Product Backlog* priorizando las tareas según el valor que aporten al cliente.
- Su objetivo principal es maximizar el retorno de inversión (ROI) mediante una visión clara y bien comunicada del producto.

2. Scrum Master:

- Es el facilitador del equipo Scrum, asegurándose de que se sigan las reglas y principios de Scrum.
- Elimina obstáculos que puedan afectar el progreso del equipo y fomenta la autogestión.
- Trabaja como coach para el equipo, ayudando a adoptar prácticas ágiles.

3. Development Team (Equipo de Desarrollo):

- Un grupo multidisciplinario y autoorganizado responsable de entregar un incremento funcional del producto.
- Incluye desarrolladores, diseñadores, testers y cualquier otro rol necesario para cumplir con las metas del Sprint.

2.3.2 Eventos en Scrum

Scrum define un conjunto de reuniones estructuradas que proporcionan puntos de inspección y adaptación:

1. Sprint Planning (Planificación del Sprint):

- Ocurre al inicio de cada Sprint.
- El equipo selecciona tareas del *Product Backlog* para incluirlas en el *Sprint Backlog*, definiendo qué se puede lograr en el ciclo.

Estudiantes	Sprint 1	Sprint 2	Sprint 3
Objetivo	Postulación a empleo	Selección de estudiantes y match	Recuperación de contraseña y chat
Tareas Clave	Que los estudiantes puedan postular a un empleo	Selección de estudiantes, match entre empleadores y estudiantes	Implementación de la recuperación de contraseñas, implementación del sistema de chat
Impacto	Los estudiantes postulan a empleos	Los estudiantes pueden ser seleccionados y evaluados	Los estudiantes pueden comunicarse a través del chat y recuperar contraseñas

Figura 3: Visualización Sprints Estudiantes.

Fuente: Elaboración propia.



Figura 4: Visualización Sprints Solicitantes.

Fuente: Elaboración propia.

2. Daily Scrum (Scrum Diario):

- Una reunión diaria de 15 minutos donde los miembros del equipo responden tres preguntas:
 - ¿Qué hice ayer?
 - ¿Qué planeo hacer hoy?
 - ¿Hay algún impedimento?

3. Sprint Review (Revisión del Sprint):

- Al final del Sprint, el equipo presenta el incremento desarrollado a los interesados.
- Permite recibir retroalimentación y ajustar las prioridades del producto.

4. Sprint Retrospective (Retrospectiva del Sprint):

- Reunión interna del equipo para reflexionar sobre el proceso.
- Se identifican logros, problemas y oportunidades de mejora.

2.3.3 Artefactos en Scrum

Scrum utiliza artefactos específicos para garantizar la transparencia del proyecto:

1. Product Backlog:

- Lista priorizada de todas las características, requisitos y correcciones del producto.
- Es un documento vivo gestionado por el Product Owner.

2. Sprint Backlog:

- Subconjunto del *Product Backlog* que el equipo se compromete a completar durante un Sprint.
- Incluye tareas desglosadas para facilitar su ejecución.

3. Incremento:

- El resultado tangible y funcional del Sprint.
- Representa el progreso acumulado del proyecto, que debe estar listo para usarse.

2.4 Fundamentos del Desarrollo de Software

2.4.1 Ciclo de Vida del Desarrollo de Software (SDLC)

El ciclo de vida del desarrollo de software (SDLC) es un marco estructurado que organiza y optimiza el desarrollo de software. Este modelo proporciona una guía para todas las etapas del proceso, asegurando que el producto final cumpla con los requisitos técnicos y de los usuarios. En el contexto de StuTrabajo, se adoptó el SDLC con un enfoque iterativo, incluyendo las siguientes etapas fundamentales:

1. Análisis de Requisitos:

- **Definición de objetivos:** En esta etapa, se identifican claramente los propósitos del sistema a desarrollar. Esto incluye el análisis de necesidades específicas de los usuarios finales, como estudiantes y empleadores.
- **Recopilación de datos:** Se utilizaron herramientas como encuestas, entrevistas y observación directa para obtener una comprensión completa de las necesidades del usuario.
- **Priorización:** Se categorizaron los requisitos en funcionales (qué debe hacer el sistema) y no funcionales (rendimiento, seguridad, usabilidad). Esto asegura que las funcionalidades críticas sean desarrolladas primero.

2. Diseño del Sistema:

- **Diseño de la arquitectura:** La arquitectura se basó en un enfoque de microservicios, permitiendo que cada componente funcione de manera independiente pero colaborativa. Esto aumenta la escalabilidad y facilita el mantenimiento.
- **Modelado de datos:** Se crearon diagramas entidad-relación para estructurar las bases de datos. Esto garantiza una organización eficiente de los datos relacionados con usuarios, empleos y transacciones.
- **Diseño de la UI/UX:** Se elaboraron wireframes y prototipos de alta fidelidad para la interfaz de usuario. Se priorizó una experiencia intuitiva y accesible para todos los usuarios.
- **Herramientas utilizadas:** En esta etapa, se emplearon herramientas como Figma para el diseño visual.

3. Desarrollo (Implementación):

- **Codificación modular:** Cada funcionalidad se implementó como un módulo independiente, siguiendo el principio de responsabilidad única (SRP). Esto facilita la reutilización y el mantenimiento del código.
- **Uso de tecnologías modernas:** Se utilizó React Native para el desarrollo del frontend y Django Rest Framework para la creación de APIs backend. Esto permitió una integración fluida entre cliente y servidor.
- **Control de versiones:** GitHub se utilizó para gestionar el código fuente y coordinar el trabajo entre los desarrolladores.

4. Pruebas:

- **Pruebas unitarias:** Cada módulo fue evaluado individualmente para garantizar que funcionara según lo previsto.
- **Pruebas de integración:** Se verificó que los diferentes módulos interactuaran correctamente.
- **Pruebas de regresión:** Estas pruebas garantizaron que las nuevas funcionalidades no afectaran negativamente las existentes.
- **Pruebas de usuario:** Se llevaron a cabo sesiones de prueba con estudiantes y empleadores para evaluar la usabilidad y efectividad de la aplicación. Los resultados permitieron ajustes específicos en la interfaz y funcionalidad.

5. Despliegue y Mantenimiento:

- **Despliegue en producción:** La aplicación se implementó utilizando instancias creadas en Amazon Web Services (AWS). Se configuró una instancia EC2 basada en Linux para alojar el backend y los servicios de la aplicación.
- **Base de datos:** Se utilizó un sistema MySQL alojado en una instancia separada para gestionar los datos de los usuarios, empleos y transacciones. Esto asegura un manejo eficiente y seguro de los datos.
- **Actualizaciones:** Basándose en la retroalimentación de los usuarios y el análisis de datos, se desarrollaron iteraciones periódicas para mejorar la funcionalidad y experiencia del usuario.

2.5 Frameworks y herramientas para desarrollo móvil

Durante el desarrollo del proyecto StuTrabajo, se optó por herramientas modernas que favorecen la rapidez de desarrollo, la mantenibilidad del código y la compatibilidad multiplataforma. Entre ellas destacan:

➤ **React Native**

- React Native es un framework de desarrollo móvil basado en JavaScript que permite crear aplicaciones nativas para sistemas operativos iOS y Android utilizando un único código base. A través de su enfoque modular, React Native facilita la implementación de componentes reutilizables que mejoran la eficiencia del desarrollo. Además, la integración con librerías externas y su ecosistema robusto permitieron construir una interfaz de usuario intuitiva, dinámica y adaptable a diferentes dispositivos y tamaños de pantalla.
- En este proyecto, React Native se utilizó para desarrollar la capa de presentación de la solución, asegurando que la experiencia de usuario fuera coherente y accesible en múltiples plataformas móviles. Los componentes diseñados incluyeron formularios, sistemas de navegación y pantallas interactivas para que los usuarios pudieran buscar, aplicar y gestionar ofertas laborales de forma eficiente.

➤ **Node.js**

- Node.js es un entorno de ejecución de JavaScript basado en el motor V8 de Google Chrome. Este entorno permite ejecutar código JavaScript en el lado del servidor, lo que facilita el desarrollo de aplicaciones backend rápidas y escalables. Node.js utiliza un modelo de entrada/salida no bloqueante y dirigido por eventos, lo que lo hace ideal para manejar múltiples solicitudes concurrentes.
- En el proyecto, Node.js sirvió como el núcleo del backend, gestionando la lógica de negocio y las interacciones con la base de datos. También fue utilizado para implementar servicios RESTful que conectan el frontend desarrollado con React Native y los datos almacenados en el servidor. Este enfoque permitió un flujo de comunicación eficiente y seguro entre los diferentes componentes de la aplicación.

➤ **Expo**

- Expo es una plataforma de desarrollo para aplicaciones React Native que proporciona herramientas y servicios diseñados para acelerar el desarrollo y despliegue de aplicaciones móviles. Entre sus características más destacadas se encuentran su sistema de hot reloading, que permite probar cambios en tiempo real, y la disponibilidad de un conjunto de APIs nativas preconfiguradas para funcionalidades comunes como la cámara, la localización y las notificaciones.
- En el proyecto, Expo se utilizó para simplificar el desarrollo y las pruebas de la aplicación móvil, eliminando la necesidad de configurar entornos nativos complejos para Android y iOS. Su compatibilidad con el ecosistema React Native permitió integrar características avanzadas en la aplicación, mientras que su herramienta Expo Go facilitó las pruebas en dispositivos reales durante todo el ciclo de desarrollo.

➤ **MySQL**

- MySQL es un sistema de gestión de bases de datos relacional que utiliza lenguaje SQL (Structured Query Language) para manejar datos. Es ampliamente utilizado debido a su robustez, rendimiento y escalabilidad. MySQL permite organizar, almacenar y recuperar grandes volúmenes de información de manera eficiente, con soporte para operaciones complejas como consultas, transacciones y relaciones entre tablas.
- En el proyecto, MySQL fue utilizado para gestionar los datos de usuarios, empleos y valoraciones. La estructura relacional permitió implementar relaciones entre entidades clave, garantizando la consistencia y la integridad de los datos almacenados. Además, se configuraron índices para optimizar las consultas y mejorar el rendimiento del sistema.

➤ **GitHub**

- Git es un sistema de control de versiones distribuido que se utilizó para rastrear los cambios en el código fuente del proyecto. GitHub, como plataforma de alojamiento de repositorios, permitió gestionar de manera centralizada el código y facilitó la colaboración entre los desarrolladores. Las ramas y solicitudes de extracción se utilizaron para mantener un flujo de trabajo estructurado y garantizar la calidad del código mediante revisiones antes de integrar nuevas funcionalidades.

➤ **Django**

- Django es un framework de desarrollo web de alto nivel basado en Python que sigue el patrón de arquitectura MVC (Modelo-Vista-Controlador). Es conocido por su enfoque en la rapidez de desarrollo, la seguridad y la simplicidad, proporcionando herramientas integradas como sistemas de autenticación, administración automática y ORM (Object-Relational Mapping) para la interacción con bases de datos.
- En el proyecto, Django fue utilizado para gestionar la lógica del backend, proporcionando APIs RESTful que conectan el frontend con la base de datos. Además, Django facilitó la implementación de funcionalidades clave como la autenticación de usuarios y la validación de datos, reduciendo el tiempo de desarrollo y mejorando la seguridad del sistema.

➤ **Visual Studio Code**

- Visual Studio Code es un editor de código fuente desarrollado por Microsoft, diseñado para ofrecer una experiencia de desarrollo ágil y personalizable. Entre sus características destacan la compatibilidad con múltiples lenguajes de programación, integración con Git y una amplia colección de extensiones que mejoran su funcionalidad.
- En este proyecto, Visual Studio Code fue utilizado como el entorno principal de desarrollo para escribir y depurar código tanto del frontend como del backend. Extensiones específicas, como las de React Native, Node.js y Docker, facilitaron la integración y el manejo de las herramientas empleadas en el proyecto.

➤ **Figma**

- Figma es una herramienta de diseño colaborativa basada en la web que permite a los equipos trabajar juntos en tiempo real. Es ideal para crear interfaces de usuario, prototipos interactivos y sistemas de diseño. Figma también permite compartir diseños con desarrolladores, quienes pueden extraer especificaciones directamente desde la herramienta.
- En el proyecto, Figma fue empleado para diseñar la interfaz gráfica de la solución. Los wireframes y prototipos de alta fidelidad creados en Figma sirvieron como guías visuales para los desarrolladores, asegurando una implementación coherente con el diseño planificado.

2.6 Consideraciones y buenas prácticas en el desarrollo de aplicaciones móviles

El desarrollo de aplicaciones móviles constituye un área dinámica dentro del desarrollo de software, caracterizada por la necesidad de ofrecer soluciones rápidas, seguras, accesibles y eficientes en un entorno altamente competitivo. A diferencia del desarrollo web o de escritorio, el entorno móvil impone condiciones particulares, como limitaciones de hardware, diversidad de dispositivos, variabilidad de conexiones de red y expectativas de experiencia inmediata por parte del usuario.

Para garantizar la calidad del producto final, es fundamental abordar el proceso de desarrollo con base en estándares técnicos, buenas prácticas de diseño UX/UI y metodologías centradas en el usuario. La correcta aplicación de estos elementos no solo optimiza el rendimiento de la aplicación, sino que también asegura su sostenibilidad, seguridad y adaptabilidad futura.

2.6.1 Recolección y análisis de requisitos

La etapa de recolección y análisis de requisitos representa el pilar fundamental sobre el cual se construye cualquier aplicación móvil. No se trata simplemente de definir una lista de funciones deseadas, sino de comprender en profundidad las necesidades reales de los usuarios, el contexto tecnológico en el que se desarrollará la solución, y las limitaciones o desafíos que podrían surgir en el camino. Esta fase tiene como objetivo establecer tanto los requisitos funcionales (acciones y comportamientos esperados del sistema) como los requisitos no funcionales (rendimiento, usabilidad, accesibilidad, seguridad, entre otros).

Para lograr un levantamiento eficaz, es necesario utilizar una combinación de métodos cualitativos y cuantitativos. Entre las técnicas más comunes se encuentran las entrevistas con usuarios potenciales, la observación directa del contexto de uso, las encuestas estructuradas, los talleres colaborativos con partes interesadas (stakeholders) y el análisis de plataformas existentes que ofrecen servicios similares. Estas herramientas permiten identificar problemas recurrentes, expectativas de valor, comportamientos habituales, y puntos de frustración (*pain points*) que el sistema debe considerar desde su diseño.

En el ámbito del desarrollo móvil, este análisis debe incorporar aspectos específicos del entorno digital portátil. Por ejemplo, es esencial definir si la aplicación será nativa, híbrida o multiplataforma, ya que esta decisión impacta profundamente en la arquitectura técnica, el rendimiento y la experiencia del usuario. Asimismo, se deben tener en cuenta las versiones mínimas de los sistemas operativos objetivo (Android, iOS), la compatibilidad con distintos tamaños y resoluciones de pantalla, y el uso de funcionalidades específicas del hardware, como sensores, cámara, GPS, o conectividad Bluetooth.

Otra dimensión importante del análisis de requisitos es el entorno de uso. Las aplicaciones móviles no siempre operan en condiciones ideales. Deben adaptarse a situaciones de conectividad limitada, movilidad constante, cambios de orientación del dispositivo y variabilidad en la calidad de la red. Esto implica que, durante el levantamiento de requisitos, deben considerarse escenarios como el uso offline, la sincronización de datos posterior, y la minimización del consumo de batería y datos móviles.

Además, para alinear el diseño de la aplicación con las expectativas reales del usuario final, es recomendable el uso de herramientas como la definición de personas (usuarios), los mapas de empatía, y los diagramas de flujo de tareas. Estas representaciones permiten visualizar distintos perfiles de usuario, con sus objetivos, frustraciones y capacidades, lo que facilita la creación de interfaces accesibles y orientadas a la tarea.

2.6.2 Diseño centrado en el usuario: Estándares UI/UX

El diseño centrado en el usuario (DCU) es una metodología fundamental en el desarrollo de aplicaciones móviles que busca asegurar que la solución final responda de manera efectiva a las necesidades, expectativas y comportamientos del usuario. Este enfoque no solo considera la estética de la interfaz, sino también la funcionalidad, la usabilidad, la accesibilidad y la eficiencia de cada interacción que ocurre dentro de la aplicación. En el entorno móvil, donde el espacio es limitado y el tiempo de atención del usuario es breve, la aplicación de principios de User Interface (UI) y User Experience (UX) es determinante para el éxito del producto.

Uno de los principios clave del diseño centrado en el usuario es la comprensión profunda del público objetivo. Esto implica investigar sus características demográficas, hábitos tecnológicos, motivaciones y posibles limitaciones cognitivas o físicas. A partir de esta comprensión se pueden definir “personas” que representan distintos tipos de usuarios y permiten orientar las decisiones de diseño hacia soluciones que resulten naturales, accesibles y satisfactorias para ellos.

Otro aspecto fundamental es la jerarquía visual clara. Una buena interfaz debe guiar la mirada del usuario hacia los elementos importantes, utilizando contrastes adecuados, tamaños proporcionales, colores coherentes con la marca y una disposición lógica de los componentes en pantalla. Esta jerarquía facilita la exploración intuitiva y reduce la necesidad de explicaciones adicionales, promoviendo una experiencia fluida.

La simplicidad y reducción de la carga cognitiva son también pilares esenciales. Las aplicaciones deben presentar la información en porciones pequeñas, fáciles de procesar, evitando saturar la pantalla con múltiples opciones o pasos innecesarios. Por ello, es recomendable utilizar formularios segmentados en pasos, menús con pocas categorías, textos breves y lenguaje claro. Cuando el usuario puede completar una acción sin esfuerzo o confusión, se genera una sensación de control y satisfacción.

La consistencia visual y funcional es otro principio UI/UX que no debe pasarse por alto. Esto implica mantener patrones coherentes en colores, tipografías, botones y comportamientos interactivos a lo largo de toda la aplicación. De esta forma, el usuario no tiene que reaprender cómo funciona cada parte del sistema, lo que agiliza su navegación y mejora la retención.

Asimismo, es fundamental ofrecer retroalimentación inmediata y significativa. Toda acción que el usuario realice debe generar una respuesta clara por parte del sistema: un cambio visual, un mensaje de confirmación, una animación o un indicador de carga. Esta respuesta ayuda al usuario a entender que su acción fue recibida y está siendo procesada, lo que reduce la ansiedad y mejora la percepción de calidad.

Otro elemento clave es el diseño adaptado al entorno táctil y móvil. Las interfaces deben considerar el uso con una sola mano, el tamaño mínimo recomendado de los botones (48x48dp según las pautas de Google), el uso de gestos naturales como deslizar o tocar, y evitar la necesidad de entradas largas mediante teclado. Además, es relevante prever condiciones como el uso bajo luz solar, pantallas pequeñas o conexiones lentas, optimizando la visibilidad, los recursos gráficos y los tiempos de carga.

No menos importante es la accesibilidad. Las aplicaciones deben ser utilizables por personas con discapacidades visuales, auditivas o motoras. Esto incluye el uso de texto alternativo en imágenes, soporte para lectores de pantalla, navegación por teclado o mediante voz, contrastes adecuados y la posibilidad de personalizar tamaños de fuente. Cumplir con estándares de accesibilidad como los establecidos por la W3C (WCAG) no solo mejora la inclusión, sino que también amplía el alcance potencial de la aplicación.

Finalmente, el diseño centrado en el usuario no es un proceso lineal ni estático. Se trata de un enfoque iterativo, donde los prototipos deben ser validados con usuarios reales a través de pruebas de usabilidad, sesiones de observación o encuestas de satisfacción. La retroalimentación obtenida debe integrarse en ciclos continuos de mejora, refinando las interfaces y adaptando la experiencia en función del comportamiento observado en el uso real.

2.6.3 Estándares técnicos en arquitectura y desarrollo

En el desarrollo de aplicaciones móviles, los estándares técnicos y las buenas prácticas arquitectónicas son elementos clave para asegurar que la solución sea eficiente, escalable, mantenible y compatible con el ecosistema tecnológico actual. Adoptar una arquitectura sólida no solo permite organizar el código de forma más limpia, sino que también facilita el trabajo colaborativo, reduce el riesgo de errores en producción y permite una evolución progresiva del producto.

Uno de los aspectos fundamentales es la organización estructural del proyecto, basada en el principio de separación de responsabilidades. Este principio sugiere que la lógica de presentación (interfaz de usuario), la lógica de negocio (procesos y validaciones) y la lógica de datos (conexión con servicios o almacenamiento) deben mantenerse desacopladas entre sí. Para lograrlo, se utilizan patrones arquitectónicos como Model-View-Controller (MVC), Model-View-ViewModel (MVVM) o Clean Architecture, los cuales permiten distribuir el código en capas con funciones claramente delimitadas, lo que mejora la mantenibilidad y reduce la complejidad del sistema.

El uso de APIs RESTful o GraphQL es otra práctica ampliamente adoptada. Las aplicaciones móviles suelen consumir datos desde servicios remotos, por lo que es imprescindible definir contratos de comunicación claros, consistentes y seguros. En este contexto, los endpoints deben seguir convenciones estándar (uso adecuado de métodos HTTP como GET, POST, PUT, DELETE), manejar correctamente los códigos de estado, y entregar las respuestas en formatos interoperables como JSON. Una buena documentación de la API, mediante herramientas como Swagger o Postman, también es esencial para facilitar la integración y el testing.

La gestión del ciclo de vida de la aplicación es otro desafío técnico. En entornos móviles, las aplicaciones pueden ser interrumpidas, suspendidas o cerradas por el sistema operativo en cualquier momento, por lo que es vital que el sistema sea capaz de preservar el estado del usuario, restaurar sesiones activas y evitar la pérdida de información. Para esto, se implementan mecanismos como almacenamiento local con SQLite, Room, SharedPreferences o soluciones como AsyncStorage en tecnologías multiplataforma.

Asimismo, debe prestarse atención a la eficiencia en el uso de recursos del dispositivo, especialmente en lo que respecta a consumo de batería, uso de CPU, y tráfico de datos móviles. Por ejemplo, se recomienda evitar llamadas innecesarias al servidor, utilizar estrategias de *caching* inteligentes, procesar datos de forma asíncrona y aplicar técnicas de *lazy loading* (carga bajo demanda) para optimizar el rendimiento general.

Otro punto crítico es la compatibilidad multiplataforma. En función del enfoque adoptado (nativo, híbrido o multiplataforma), se deben seguir las pautas recomendadas para cada sistema operativo. En el desarrollo nativo, se utilizan herramientas como Android Studio (Java/Kotlin) o Xcode (Swift), mientras que para soluciones multiplataforma se recurre a frameworks como React Native, Flutter o Xamarin. Cada opción tiene ventajas y desventajas en cuanto a rendimiento, acceso a funciones del sistema y tiempo de desarrollo, por lo que su elección debe basarse en los objetivos del proyecto, los recursos del equipo y el perfil del público objetivo.

Además, es importante implementar una estrategia de gestión de errores robusta, que permita capturar excepciones, registrar logs, mostrar mensajes claros al usuario y evitar bloqueos inesperados. Para ello, se utilizan herramientas como Crashlytics, Sentry o servicios de monitoreo integrados, que permiten identificar fallos en tiempo real y aplicar correcciones en versiones futuras.

En lo referente a pruebas, se promueve la adopción de pruebas automatizadas para asegurar la calidad del código en cada etapa del desarrollo. Esto incluye pruebas unitarias (que validan funciones individuales), pruebas de integración (que comprueban la interacción entre módulos), y pruebas de UI (que simulan acciones del usuario en la interfaz). Estas pruebas pueden ejecutarse automáticamente en *pipelines* de integración continua (CI) y facilitar un proceso de desarrollo más seguro y eficiente.

Finalmente, el uso de control de versiones mediante sistemas como Git es indispensable para gestionar los cambios del proyecto, trabajar en equipo, y mantener un historial confiable del desarrollo. Es recomendable adoptar una estrategia clara de ramas (por ejemplo, Git Flow) que diferencie el trabajo en desarrollo, pruebas y producción, permitiendo un ciclo de vida más controlado y profesional del producto.

2.6.4 Seguridad en aplicaciones móviles

La seguridad es un aspecto fundamental en el desarrollo de aplicaciones móviles. A medida que estas aplicaciones se convierten en canales principales de comunicación, gestión de datos y transacciones digitales, también se transforman en objetivos frecuentes de ataques cibernéticos. Por ello, es indispensable integrar buenas prácticas de seguridad desde las primeras etapas del desarrollo, adoptando un enfoque conocido como “seguridad desde el diseño” (*security by design*), y considerando amenazas tanto técnicas como humanas.

Una de las principales medidas de seguridad es garantizar la protección de los datos durante su transmisión, especialmente cuando se interactúa con servicios externos.

Toda comunicación entre la aplicación y sus servidores debe realizarse a través de protocolos cifrados como HTTPS, lo que asegura que los datos enviados y recibidos no puedan ser interceptados o modificados por terceros. El uso de certificados SSL/TLS válidos y actualizados es una condición mínima para establecer una conexión segura.

Además de proteger los datos en tránsito, también se deben proteger los datos en reposo, es decir, aquellos almacenados localmente en el dispositivo. Muchas aplicaciones móviles requieren guardar cierta información para funcionar offline o conservar preferencias del usuario, pero almacenar datos sensibles como contraseñas, tokens de autenticación, o información personal en texto plano representa un riesgo grave. En su lugar, se deben utilizar mecanismos seguros de almacenamiento, como el Keystore en Android o el Keychain en iOS, que están diseñados específicamente para proteger credenciales y datos críticos del usuario.

Otro aspecto esencial es la gestión adecuada de la autenticación y autorización. Las aplicaciones deben implementar mecanismos que verifiquen la identidad del usuario de forma segura, como contraseñas robustas, autenticación multifactor (2FA), o inicio de sesión biométrico (huella digital, reconocimiento facial). Asimismo, el acceso a recursos o funcionalidades específicas debe estar controlado mediante tokens de sesión temporales (por ejemplo, JWT) que no puedan ser reutilizados ni manipulados.

Un error común en el desarrollo móvil es dejar expuestos recursos internos o configuraciones sensibles en el código fuente de la aplicación. Es frecuente, por ejemplo, que desarrolladores dejen claves de API, secretos de cifrado o rutas de prueba embebidas en archivos accesibles dentro del paquete de la aplicación (APK o IPA). Esto facilita la ingeniería inversa, técnica mediante la cual un atacante descompila el código y extrae información crítica. Para evitarlo, se recomienda ofuscar el código, utilizar variables de entorno seguras en el servidor y aplicar capas de verificación del entorno de ejecución, como la detección de dispositivos rooteados o emuladores.

También se deben considerar las amenazas asociadas al comportamiento del usuario. La aplicación debe ser capaz de manejar errores de entrada de forma segura, evitando ataques comunes como inyecciones de código, *cross-site scripting* (XSS) o ataques por fuerza bruta. Para ello, es importante validar los datos tanto del lado del cliente como del servidor, limitar el número de intentos de autenticación y aplicar medidas como el CAPTCHA o bloqueos temporales ante comportamientos sospechosos.

Una referencia importante para estructurar la seguridad de las aplicaciones móviles es el estándar OWASP Mobile Top 10, que identifica los riesgos más frecuentes en este tipo de

plataformas. Entre ellos se destacan la comunicación insegura, la falta de almacenamiento seguro, una autenticación débil, exposición de datos sensibles y mala implementación criptográfica. Este marco permite a los desarrolladores evaluar su aplicación en función de buenas prácticas reconocidas a nivel internacional y adoptar medidas correctivas de manera anticipada.

Finalmente, la seguridad debe contemplarse como un proceso continuo y no como una característica estática del software. Esto implica realizar auditorías de seguridad periódicas, utilizar herramientas de análisis de vulnerabilidades, mantener actualizadas las dependencias utilizadas en el proyecto y fomentar una cultura de desarrollo seguro dentro del equipo. Incluir procedimientos de respuesta ante incidentes, backups automáticos y monitoreo en tiempo real son prácticas adicionales que fortalecen la resiliencia de la aplicación ante amenazas externas.

2.6.5 Pruebas, control de versiones y mantenimiento

La calidad de una aplicación móvil no depende únicamente de su diseño visual o su funcionalidad superficial, sino también de su estabilidad, rendimiento y capacidad de evolución en el tiempo. Para garantizar estas cualidades, resulta indispensable implementar estrategias sólidas de pruebas (testing), un sistema eficaz de control de versiones y una planificación estructurada para su mantenimiento y actualización continua.

Pruebas: niveles y técnicas

Las pruebas permiten verificar que la aplicación cumpla con los requisitos especificados, que su comportamiento sea el esperado ante diversos escenarios y que los errores se detecten de manera oportuna antes de llegar al usuario final. En el desarrollo móvil, estas pruebas deben contemplar múltiples niveles:

- **Pruebas unitarias:** Se enfocan en evaluar unidades pequeñas de código de forma aislada, como funciones o clases. Estas pruebas permiten detectar errores lógicos en etapas tempranas, facilitando una corrección rápida y económica.
- **Pruebas de integración:** Verifican cómo interactúan entre sí distintos módulos del sistema, asegurando que los flujos de datos entre capas funcionen correctamente. Son útiles para validar la comunicación con APIs, bases de datos o servicios externos.

- **Pruebas de interfaz (UI testing):** Evalúan la interacción entre el usuario y la aplicación. Simulan acciones reales como pulsaciones, deslizamientos o entradas de texto, y validan que la interfaz responda correctamente en distintas condiciones y dispositivos.
- **Pruebas funcionales:** Aseguran que cada una de las funcionalidades principales de la aplicación se ejecute de acuerdo con los requisitos definidos. Suelen realizarse con herramientas automatizadas y manuales.
- **Pruebas de usabilidad y experiencia de usuario:** Involucran usuarios reales o perfiles representativos que utilizan la aplicación en condiciones similares a las reales. Estas pruebas permiten identificar problemas de comprensión, navegación o satisfacción que no se evidencian en pruebas técnicas.
- **Pruebas de rendimiento y estrés:** Evalúan la capacidad de respuesta de la aplicación bajo condiciones extremas, como múltiples peticiones simultáneas, poca memoria disponible o cambios abruptos en la red. Ayudan a detectar cuellos de botella y a mejorar la escalabilidad.

La automatización de pruebas es una práctica cada vez más común, ya que permite repetir escenarios complejos de forma rápida y consistente. Herramientas como JUnit, Espresso, XCTest, Detox o Appium permiten automatizar distintas pruebas tanto en aplicaciones nativas como multiplataforma.

Control de versiones

El uso de un sistema de control de versiones es esencial para gestionar el desarrollo de una aplicación móvil de manera organizada, colaborativa y trazable. Git es la herramienta más utilizada en la industria debido a su flexibilidad, escalabilidad y soporte extendido.

Implementar una estrategia de ramificación clara (como Git Flow, GitHub Flow o trunk-based development) permite separar el trabajo en curso del código estable, gestionar entornos de desarrollo, pruebas y producción, y reducir los riesgos de errores durante despliegues. Además, facilita la colaboración entre múltiples desarrolladores, integradores, diseñadores y testers, permitiendo fusionar cambios, revertir errores y auditar el historial completo del proyecto.

El uso de plataformas como GitHub, GitLab o Bitbucket ofrece funcionalidades adicionales como revisión de código, gestión de incidencias, integración continua (CI) y automatización de despliegues, lo que incrementa la productividad del equipo y mejora la calidad del software entregado.

Mantenimiento y evolución

El mantenimiento de una aplicación no finaliza con su primera versión publicada. Por el contrario, comienza una nueva etapa donde se deben gestionar actualizaciones, corrección de errores, ajustes por cambios en plataformas externas (como nuevas versiones de Android/iOS), y la incorporación de nuevas funcionalidades según la retroalimentación de los usuarios.

Existen distintos tipos de mantenimiento:

- **Correctivo:** Soluciona errores que se detectan después del despliegue.
- **Adaptativo:** Ajusta la aplicación ante cambios del entorno tecnológico (sistemas operativos, librerías, políticas de tiendas de aplicaciones).
- **Perfectivo:** Mejora funcionalidades existentes o la experiencia general del usuario.
- **Preventivo:** Refactoriza el código o actualiza dependencias para reducir el riesgo de futuros errores o vulnerabilidades.

Para facilitar este proceso, se recomienda adoptar prácticas como la integración y entrega continua (CI/CD), que permiten compilar, probar y desplegar nuevas versiones de forma automatizada, segura y rápida. Herramientas como Jenkins, CircleCI, Bitrise o GitHub Actions son ampliamente utilizadas con este fin.

También es fundamental establecer un sistema de monitoreo en producción, mediante herramientas que detecten caídas, errores de ejecución o patrones anómalos de uso. Servicios como Firebase Crashlytics, Sentry, New Relic o Datadog permiten recibir alertas en tiempo real, visualizar estadísticas y responder de forma proactiva ante incidentes.

2.6.6 Conclusiones sobre estándares en desarrollo móvil

El desarrollo de aplicaciones móviles es un proceso que va mucho más allá de la simple escritura de código o el diseño de interfaces atractivas. Requiere una planificación cuidadosa, una ejecución disciplinada y una evaluación constante, considerando los múltiples factores técnicos, humanos y contextuales que influyen en la calidad y el éxito de un producto digital⁷.

En este escenario, los estándares en el desarrollo móvil actúan como guías fundamentales que orientan las decisiones de diseño, arquitectura, implementación, seguridad y mantenimiento. Estos estándares no son reglas arbitrarias, sino el resultado de años de experiencia acumulada por comunidades de desarrollo, grandes empresas tecnológicas y organismos internacionales que buscan establecer buenas prácticas reproducibles y confiables⁸.

Aplicar estos estándares permite enfrentar con mayor eficacia los desafíos propios del ecosistema móvil: diversidad de dispositivos, fragmentación de sistemas operativos, limitaciones de hardware, expectativas de experiencia fluida, amenazas de seguridad y ciclos de actualización constantes. Cuando los equipos de desarrollo respetan estas buenas prácticas, se reduce significativamente la probabilidad de errores críticos, se optimiza el uso de recursos y se mejora la escalabilidad y sostenibilidad de la aplicación en el tiempo⁹.

⁷ <https://www.arkasoftwares.com/blog/guide-for-mobile-app-development-standards/>

⁸ <https://www.pulsion.co.uk/blog/software-development-standards/>

⁹ <https://owasp.org/www-project-mobile-app-security/>

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

3.1 Descripción general de la solución

Considerando las circunstancias del proyecto mencionadas en el Capítulo II, ahora se procede a describir en términos generales el diseño del sistema que será implementado en el proyecto de investigación.

El primer paso para determinar la solución es revisar en mayor detalle los requisitos que los stakeholders esperan que el sistema deba cumplir. Dentro de todos los requisitos solicitados, el de mayor relevancia y trascendencia corresponde a poder utilizar el sistema en cualquier momento del día y en cualquier lugar. Por consecuencia, la solución que será implementada corresponderá a una aplicación móvil, en virtud de las características que ofrece esta tecnología, ya sea por otorgar portabilidad, bajos costos asociados, su amplio uso en la vida diaria y su masificación.

Por otro lado, la implementación del núcleo del sistema estará compuesto de dos componentes principales. Por un lado, se encuentra la aplicación móvil y por otra parte se encuentra el servidor de datos, el funcionamiento de ambos componentes en conjunto permite una solución robusta.

Se detallan los componentes a implementar:

1. **Aplicativo móvil:** Componente utilizado por los usuarios finales (Investigadores).
2. **Servidor de datos:** Componente encargado del almacenamiento, procesamiento y distribución de datos, desde y hacia la aplicación móvil.

Concretamente la aplicación móvil, apoyará en todo momento los procesos de investigación durante las fases preliminares de estudios de usuarios, permitiendo así, agilizar la obtención de datos, análisis y generación de información. Dicha aplicación móvil estará conectada directamente con un servidor de datos, quién será el encargado de procesar la información enviada por el aplicativo y enviarla de vuelta para su correcta visualización.

Con estos componentes funcionando en conjunto, el sistema será capaz de procesar los datos de la investigación en tiempo real entre todos los investigadores registrados, adicionando a su vez, la portabilidad y rapidez de uso que ofrecen los dispositivos móviles en los tiempos actuales.

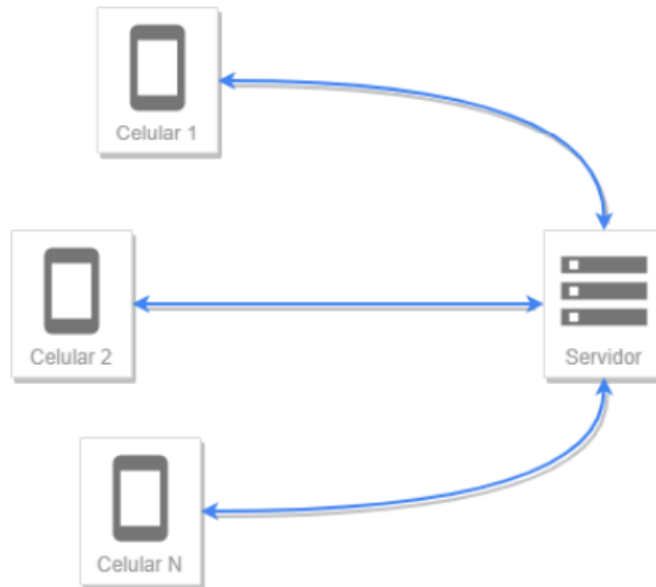


Figura 5: Conexión móvil – servidor.

Fuente: Elaboración propia.

3.2 Funcionalidades de la aplicación

Dentro del desarrollo de un proyecto de software, el punto más relevante y crucial por sobre todos los demás es la “Ingeniería de requerimientos”, su importancia radica principalmente en que los requerimientos afectan el nivel de productividad del proceso de desarrollo y pueden desencadenar malos resultados finales. Los requerimientos mal identificados provocarán retrasos en el desarrollo del software, lo que finalmente se verá reflejado en costos asociados.

Por otro lado, el objetivo de la ingeniería de requerimientos es poder identificar aquellos requerimientos que puedan resolver las problemáticas o necesidades planteadas por los stakeholders, por consecuencia, si los requerimientos del proyecto se cumplen a cabalidad, el software cumple su función para lo que fue diseñado y entonces, se clasifica como un software de calidad, entendiéndose este último concepto como la satisfacción del usuario ante el sistema entregado.

A continuación, se detallan los dos tipos de requerimientos de software presentes en el proyecto

3.2.1 Requerimientos funcionales

Como introducción al lector, los requerimientos funcionales son aquellos que corresponden a los servicios ofrecidos por el software desarrollado y que realizan transformaciones de entrada de los datos mediante comportamientos o acciones, para finalmente generar una salida procesada de los mismos.

Se describen de manera general los requisitos funcionales del sistema:

- La aplicación debe permitir el registro y autenticación de usuarios (estudiantes y solicitantes)
- Los estudiantes deben poder crear y editar su perfil, incluyendo información como nombre, carrera, disponibilidad horaria y habilidades.
- Los solicitantes deben poder registrar su empresa o cuenta personal, proporcionando información sobre su identidad y tipo de empleos que ofrecen.
- Posibilidad de recuperar contraseña en caso de olvido.
- Los empleadores deben poder crear, editar y eliminar ofertas de empleo, indicando detalles como descripción, duración, requisitos y remuneración.
- Los estudiantes deben poder buscar empleos mediante filtros como ubicación, tipo de trabajo, horario y remuneración.
- Se debe implementar un sistema de notificaciones en tiempo real para alertar a los estudiantes sobre nuevas ofertas relevantes.
- Los empleadores pueden ver postulaciones y seleccionar candidatos de forma rápida y sencilla.
- Los estudiantes pueden postularse a empleos con un solo clic, enviando su perfil al empleador.
- Los empleadores pueden revisar perfiles de postulantes y aceptar/rechazar candidatos según sus necesidades.
- Implementación de un chat interno para facilitar la comunicación entre empleadores y estudiantes.

- Implementación de un sistema de calificación y comentarios, donde estudiantes y empleadores puedan evaluarse mutuamente.
- Se debe incluir un mecanismo de reporte de fraudes o problemas con ofertas de trabajo o candidatos.

3.2.2 Requerimientos no funcionales

Por otra parte, los requerimientos no funcionales son aquellos que hacen referencia a propiedades del sistema y no a su comportamiento específico, en simples palabras, los requerimientos no funcionales, describen atributos de calidad del software y están restringidos según las necesidades de los stakeholders, por ejemplo: restricciones de tiempo o restricciones de recursos económicos disponibles.

Con la explicación anterior mencionada. En referencia al proyecto, los requisitos no funcionales son:

- La aplicación debe contar con una interfaz intuitiva y fácil de usar, permitiendo que los usuarios naveguen sin dificultades desde su primera interacción.
- El sistema debe ser escalable horizontal y verticalmente, permitiendo el crecimiento del número de usuarios sin afectar el rendimiento.
- Los datos de los usuarios deben ser cifrados para proteger información sensible como contraseñas y datos personales.
- El sistema debe ser capaz de responder a las solicitudes del usuario en las operaciones más críticas (búsqueda de empleo, carga de perfil, postulaciones).

3.3 Usuarios dentro del sistema

Para definir a los usuarios dentro del sistema, es necesario considerar un importante detalle, como se mencionó anteriormente, la aplicación móvil debe ser capaz de ser autoadministrable, para lograr esto, es requerida una división de roles dentro del sistema, en donde cada rol tenga limitados permisos para el acceso y modificación de los recursos asociados.

Los roles dentro del sistema quedaran definidos de la siguiente manera:

1. Estudiantes

Los estudiantes son el público objetivo principal de **StuTrabajo**, ya que la plataforma está diseñada para ayudarlos a encontrar empleos esporádicos compatibles con sus estudios. Sus funciones y permisos serán:

- Registro y autenticación en la aplicación.
- Creación y edición de su perfil, incluyendo información como carrera, disponibilidad y habilidades.
- Búsqueda de ofertas laborales mediante filtros (ubicación, tipo de empleo, horario, remuneración).
- Postulación a empleos con un solo clic.
- Comunicación con solicitantes a través del chat interno.
- Calificación y reseñas a solicitantes después de finalizar un trabajo.
- Recepción de notificaciones sobre nuevas ofertas de empleo o cambios en postulaciones.
- Reporte de irregularidades en las ofertas laborales o solicitantes.

2. Solicitantes

Los solicitantes representan a las empresas, organizaciones o particulares que publican ofertas de empleo en la plataforma. Su función principal es ofrecer oportunidades de trabajo flexible y seleccionar candidatos adecuados. Sus funciones y permisos serán:

- Registro y autenticación en la aplicación.
- Creación de un perfil de empresa o solicitante individual con información verificable.
- Publicación, edición y eliminación de ofertas de empleo.
- Revisión de postulaciones y selección de candidatos.
- Comunicación con estudiantes a través del chat interno.
- Evaluación de los estudiantes después de la finalización del trabajo.
- Reporte de usuarios por comportamiento inapropiado o incumplimiento de acuerdos.

3. Administrador del sistema

El administrador es el encargado de la gestión y mantenimiento de la plataforma. Se asegura de que el sistema funcione correctamente, previene abusos y resuelve disputas entre usuarios. Sus funciones y permisos serán:

- Gestión de usuarios (bloqueo, suspensión o eliminación de cuentas en caso de incumplimientos).
- Moderación de contenido (ofertas de empleo fraudulentas o inadecuadas).
- Resolución de disputas entre estudiantes y solicitantes.
- Supervisión del correcto funcionamiento del sistema y soporte técnico.
- Administración de bases de datos y seguridad de la información.
- Generación de reportes sobre el uso de la plataforma y su crecimiento.

3.4 Diagrama de Casos de Uso



Figura 6: Diagrama de Casos de Uso.

Fuente: Elaboración propia.

3.5 Casos de uso detallados

Tabla 2: Caso de Uso - Registrar Cuenta.

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Registrar cuenta
N.º del caso de uso	CU01
Actores que participan	Estudiante, Solicitante
Quien lo comienza	Estudiante, Solicitante
Quien lo termina	Sistema
Precondiciones	El usuario debe proporcionar un correo válido, contraseña y datos básicos.
Postcondiciones	La cuenta es creada y el usuario puede iniciar sesión.
Excepciones	1) El correo ya está registrado. 2) Error en la conexión a la base de datos.
Descripción del caso de uso	1. El usuario accede a la pantalla de registro. 2. Ingresa sus datos personales (nombre, correo, contraseña, tipo de usuario). 3. Hace clic en "Registrar". 4. El sistema valida los datos. 5. Si los datos son correctos, el sistema almacena la información y muestra un mensaje de éxito. 6. El usuario recibe un correo de confirmación.

Tabla 3: Caso de Uso - Iniciar Sesión.

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Iniciar Sesión
N.º del caso de uso	CU02
Actores que participan	Estudiante, Solicitante.
Quien lo comienza	Estudiante, Solicitante.
Quien lo termina	Sistema
Precondiciones	El usuario debe estar registrado y proporcionar credenciales correctas.
Postcondiciones	El usuario accede a su perfil con sesión iniciada.
Excepciones	1) Credenciales incorrectas. 2) Cuenta bloqueada o inactiva.
Descripción del caso de uso	1. El usuario accede a la pantalla de inicio de sesión. 2. Ingresa su correo y contraseña. 3. Hace clic en "Iniciar sesión". 4. El sistema verifica las credenciales. 5. Si son correctas, el sistema permite el acceso y redirige al usuario a su perfil.

Tabla 4: Caso de Uso - Crear/Editar Perfil

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Crear/Editar Perfil
N.º del caso de uso	CU03
Actores que participan	Estudiante, Solicitante
Quien lo comienza	Estudiante, Solicitante
Quien lo termina	Sistema
Precondiciones	Usuario debe haber iniciado sesión.
Postcondiciones	El perfil es actualizado con éxito.
Excepciones	1) Faltan datos obligatorios. 2) Error en la conexión con la base de datos.
Descripción del caso de uso	1. El usuario accede a su perfil. 2. Modifica su información (foto, nombre, habilidades, etc.). 3. Guarda los cambios. 4. El sistema valida y almacena la información. 5. Se muestra un mensaje de confirmación.

Tabla 5: Caso de Uso - Buscar empleo.

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Buscar empleo
N.º del caso de uso	CU04
Actores que participan	Estudiante
Quien lo comienza	Estudiante
Quien lo termina	Sistema
Precondiciones	Usuario debe haber iniciado sesión.
Postcondiciones	Se muestran empleos disponibles y los filtros según criterios de búsqueda.
Excepciones	1) No hay empleos disponibles según los filtros seleccionados.
Descripción del caso de uso	1. El estudiante accede a la pantalla de búsqueda. 2. Ingresa filtros (categoría, ubicación, horario, remuneración). 3. Hace clic en "Buscar". 4. El sistema muestra los resultados disponibles.

Tabla 6: Caso de Uso - Postular empleo.

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Postular empleo
N.º del caso de uso	CU05
Actores que participan	Estudiante, Solicitante.
Quien lo comienza	Estudiante
Quien lo termina	Sistema
Precondiciones	El estudiante debe tener un perfil completo y haber iniciado sesión.
Postcondiciones	La postulación se registra y se notifica al solicitante.
Excepciones	1) El empleo ya no está disponible. 2) El estudiante no cumple con los requisitos.
Descripción del caso de uso	1. El estudiante accede a una oferta de empleo. 2. Hace clic en "Postular". 3. El sistema verifica si el usuario cumple los requisitos. 4. Si es válido, la postulación se envía al solicitante. 5. El solicitante recibe una notificación.

Tabla 7: Caso de Uso - Calificar Perfil.

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Calificar perfil
N.º del caso de uso	CU06
Actores que participan	Estudiante, Solicitante
Quien lo comienza	Estudiante, Solicitante.
Quien lo termina	Sistema
Precondiciones	El trabajo debe haber finalizado.
Postcondiciones	La calificación y comentarios quedan registrados en el perfil del usuario.
Excepciones	1) No se permite calificar antes de finalizar el empleo. 2) Error en la base de datos.
Descripción del caso de uso	1. El usuario accede a la opción de calificación. 2. Selecciona la persona a calificar. 3. Ingresa una puntuación y comentario. 4. Hace clic en "Enviar". 5. El sistema guarda la calificación.

Tabla 8: Caso de Uso - Chat.

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Chat
N.º del caso de uso	CU07
Actores que participan	Estudiante, Solicitante
Quien lo comienza	Estudiante, Solicitante
Quien lo termina	Sistema
Precondiciones	El usuario debe estar autenticado y haber iniciado una interacción con otro usuario.
Postcondiciones	Mensajes enviados correctamente entre usuarios.
Excepciones	1) Error en la conexión a Internet. 2) Usuario bloqueado por el otro usuario.
Descripción del caso de uso	1. El usuario accede al chat. 2. Selecciona una conversación activa. 3. Escribe un mensaje. 4. Hace clic en "Enviar". 5. El sistema transmite el mensaje en tiempo real.

Tabla 9: Caso de Uso - Publicar empleo.

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Publicar empleo
N.º del caso de uso	CU08
Actores que participan	Solicitante
Quien lo comienza	Solicitante
Quien lo termina	Sistema
Precondiciones	El solicitante debe tener un perfil completo y estar autenticado.
Postcondiciones	La oferta de empleo se publica correctamente en la plataforma.
Excepciones	1) Falta información obligatoria. 2) Error en la base de datos.
Descripción del caso de uso	1. El solicitante accede a la sección de publicación. 2. Completa los detalles de la oferta (título, requisitos, pago, ubicación). 3. Hace clic en "Publicar". 4. El sistema verifica los datos y publica la oferta.

Tabla 10: Caso de Uso - Gestionar postulaciones.

Fuente: Elaboración propia.

Item	Detalle
Nombre del caso de uso	Gestionar postulaciones
N.º del caso de uso	CU09
Actores que participan	Solicitante
Quien lo comienza	Solicitante
Quien lo termina	Sistema
Precondiciones	Debe haber postulaciones activas en la oferta de empleo.
Postcondiciones	Se acepta o rechaza a un estudiante y se notifica su estado.
Excepciones	1) No hay postulaciones disponibles. 2) Error al actualizar el estado del postulante.
Descripción del caso de uso	1. El solicitante accede a la lista de postulantes. 2. Revisa los perfiles de los estudiantes. 3. Selecciona a un candidato y hace clic en "Aceptar" o "Rechazar". 4. El sistema actualiza el estado de la postulación y envía una notificación.

3.6 Modelo de Datos

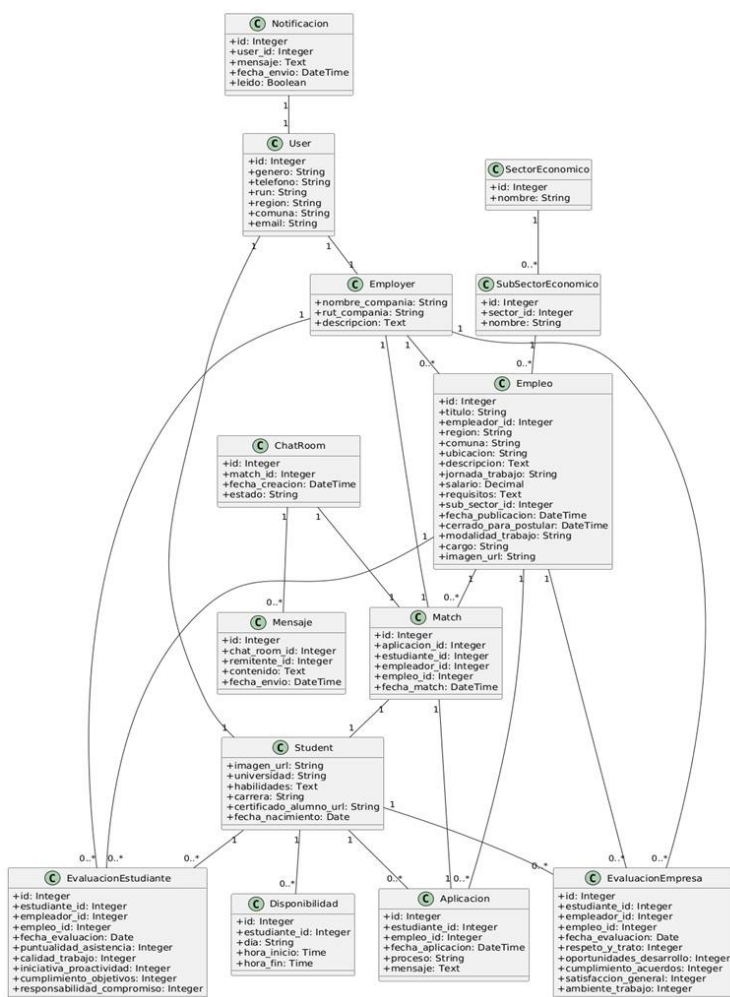


Figura 7: Modelo de Datos - StuTrabajo.

Fuente: Tesis Sebastián Poblete Godoy.

3.7 Diseño preliminar de interfaz de Software

Una interfaz de usuario es el componente de software que permite la interacción entre el usuario y el sistema, es decir, actúa como un medio o canal para la comunicación entre ambas partes.

Los objetivos que persiguen las interfaces de usuario corresponden a facilitar el uso y la interacción de los usuarios con el sistema. La interfaz diseñada para este proyecto tiene por objetivos:

- Que la interfaz sea capaz de comunicar información relevante.
- Ser auto explicativa, es decir en simples palabras, que la interfaz no requiere explicaciones adicionales y/o documentación externa, para ser utilizada por los investigadores.
- Que posea una estructura y flujo de interacciones entre pantallas coherente con los conceptos que la metodología de diseño participativo aplica.

En particular, la interfaz diseñada se clasifica como interfaz de tipo GUI, a consecuencia de que la información es representada gráficamente mediante estructuras de control como menús desplegables, barras de navegación, íconos y ventanas; los que son utilizados mediante un dispositivo señalador, el cual en este caso corresponden a los dedos sobre una pantalla táctil.

Para el desarrollo de la interfaz gráfica de la aplicación móvil, se trabajó en Figma junto con la ayuda de mi compañero de proyecto Daniel Soto Benavides, encargado de diseño UI/UX de la plataforma. Luego de sucesivas iteraciones y retroalimentaciones por parte de los stakeholders, las maquetas finales previas a la implementación física de la interfaz son las siguientes:



Figura 8 – Interfaz “Home”



Figura 9 – Interfaz “Login”

En la *Figura 8* *Figura 8 – Interfaz “Home”* se puede observar la interfaz “Home”, que esta pantalla será la inicial cuando se abra la aplicación. Por otra parte, la *Figura 9* representa la interfaz “Login” que será en donde el usuario ingresa con una cuenta ya creada. Cuenta con opciones de recordar usuario para no estar ingresando los datos reiteradas veces y también un botón de olvidé contraseña en caso de que ocurra.

En la *Figura 10* podemos observar la interfaz “Crear cuenta”, en donde el usuario deberá elegir un tipo de usuario y luego completar sus datos para poder crear su cuenta. Una vez creada su cuenta, podrá iniciar sesión y comenzar a navegar por al APP. En la *Figura 11* se encuentra el apartado de “Mi perfil” en donde podrá visualizar todos los datos del usuario y otro tipo de configuraciones.



Figura 11 – Interfaz “Crear cuenta”



Figura 11 – Interfaz “Perfil”

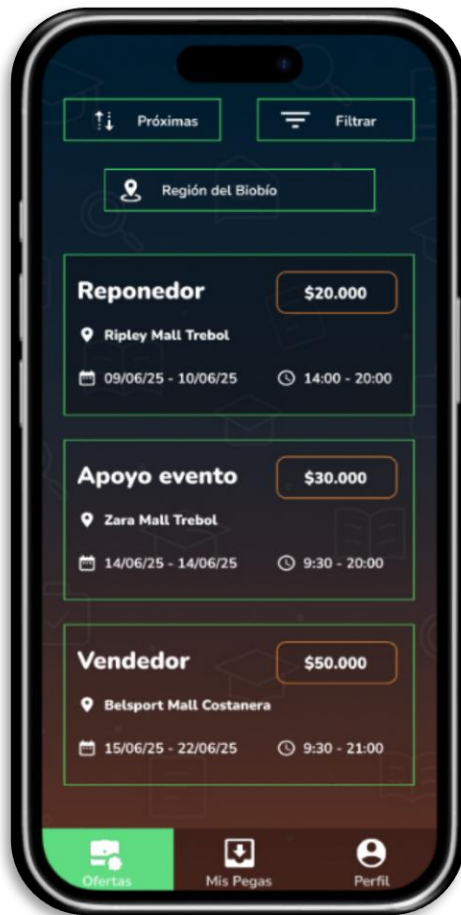


Figura 13 – Interfaz “Ofertas”



Figura 13 – Interfaz “Detalles empleo”

En la *Figura 12* podemos observar las ofertas creadas por los solicitantes, en donde sale con una pequeña tarjeta de presentación los datos principales. Si se hace click sobre una tarjeta, podremos visualizar como se ve en la *Figura 13* todos los datos de la oferta y su respectivo botón de “Postular”.

En la *Figura 14* podemos observar el apartado de “Mis Pegas” en donde se podrá visualizar todas las ofertas a las que se ha postulado y también su respectivo estado. Cabe destacar que al momento de hacer click sobre una tarjeta, saldrá la opción de “Anular postulación” en caso de requerirlo.

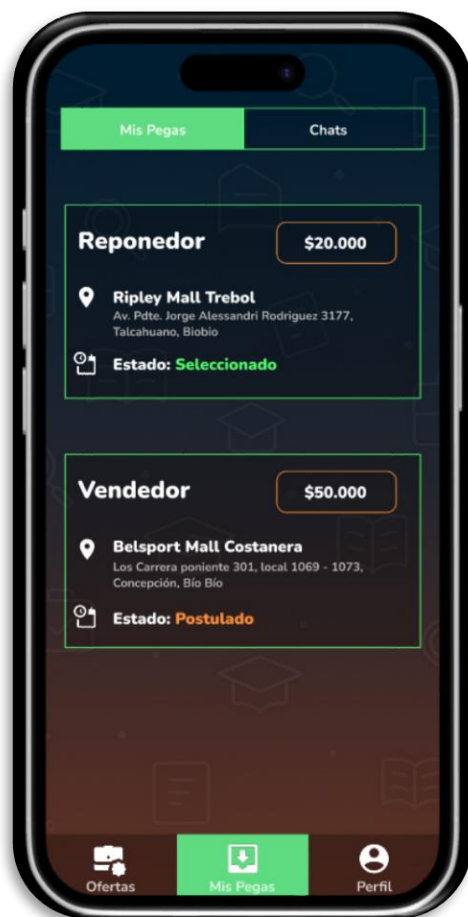


Figura 14 – Interfaz “Mis Pegas”

En la *Figura 15* se observa la interfaz de “Match” que se concreta cuando un estudiante postula y el solicitante lo selecciona. Al momento de ser seleccionado saldrá este pop up en donde se podrá crear un chat en tiempo real, tal como se puede visualizar en la *Figura 16*.



Figura 16 – Interfaz “Match”



Figura 15 – Interfaz “Chat”

En la *Figura 17* se puede observar el apartado de “Mis publicaciones” en donde el solicitante podrá visualizar sus ofertas creadas. En este caso, el podrá ver los detalles de su oferta y también podrá editarla en caso de ser necesario. También cuenta con la opción de ver postulantes en donde salen todos los estudiantes que han postulado a esa oferta. En la *Figura 18* se puede observar el apartado para “Crear oferta de empleo”

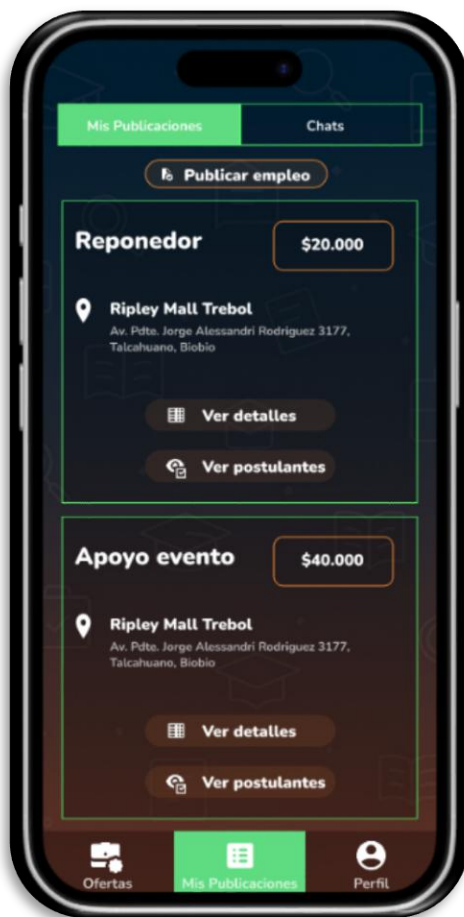


Figura 18 – Interfaz “Mis publicaciones”



Figura 17 – Interfaz “Crear empleo”

El solicitante al presionar sobre el botón “Ver postulantes” podrá visualizar todos los estudiantes que han postulado con su respectiva tarjeta, con foto de perfil, nombre y su valoración, como se puede ver en la *Figura 19*. También al presionar sobre una tarjeta de perfil, se podrá visualizar datos del estudiante con su respectivo video de presentación y valoración en cada ámbito, como se puede ver en la *Figura 20*.

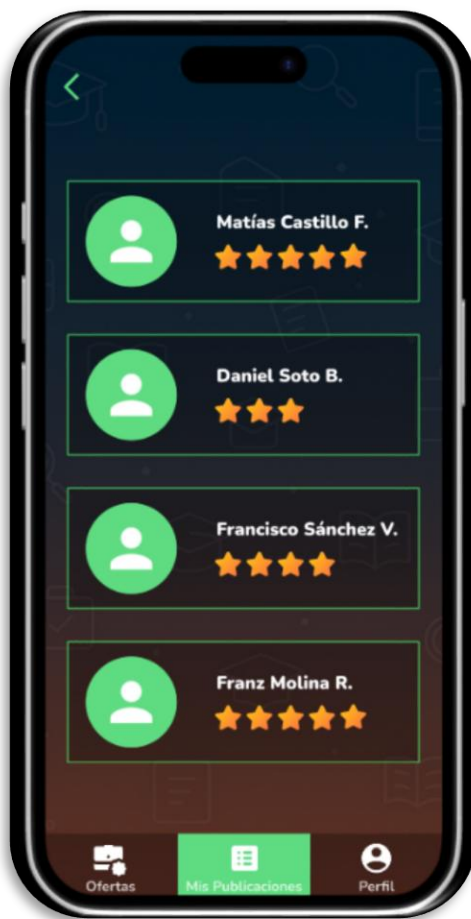


Figura 20 – Interfaz “Ver postulantes”

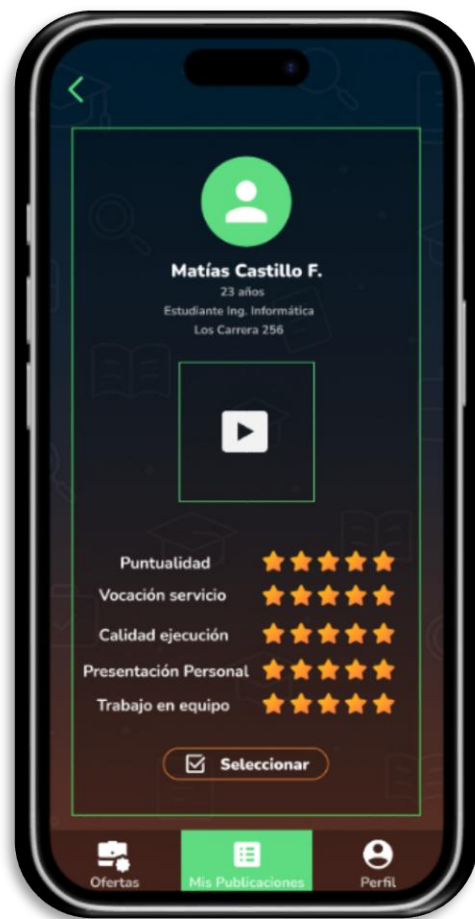


Figura 19 – Interfaz “Presentación perfil”

Al finalizar el trabajo, saldrá un pop up como se ve en la *Figura 21*, en donde el solicitante deberá evaluar al estudiante cada ámbito en el entorno de trabajo. También podrá agregar alguna observación respecto al estudiante. En la *Figura 22* podemos observar lo mismo, pero en este caso, el estudiante evalúa cada ámbito del solicitante.

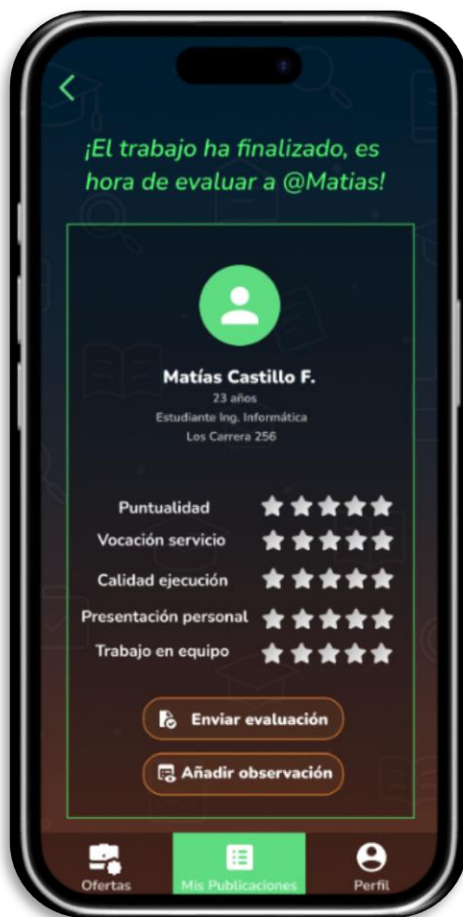


Figura 22 – Interfaz “Valoración Estudiante”

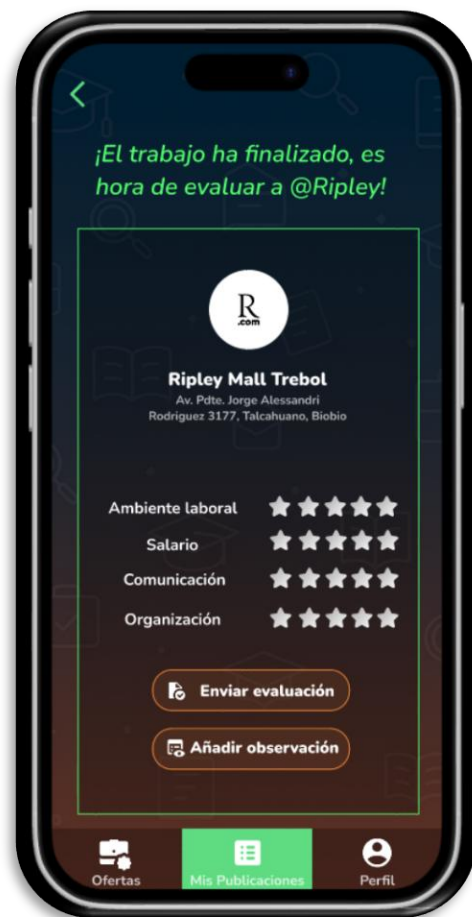


Figura 21 – Interfaz “Valoración Solicitante”

El desarrollo del prototipo de StuTrabajo representa un paso fundamental en la materialización de la solución propuesta, ya que permite visualizar la estructura y funcionalidad clave de la aplicación antes de su implementación final. A través de este diseño preliminar, se han definido aspectos esenciales como la navegabilidad, la disposición de los elementos en la interfaz y la experiencia de usuario, asegurando que cada componente responda a las necesidades de los estudiantes y solicitantes de empleo.

Este prototipo no solo proporciona una representación gráfica del sistema, sino que también actúa como un puente entre la fase de diseño y la implementación del software, facilitando la comunicación entre los diseñadores, desarrolladores y partes interesadas. Al establecer un modelo claro de la interfaz y sus interacciones, se minimizan posibles ambigüedades en el desarrollo y se optimiza el tiempo de programación, evitando cambios estructurales tardíos.

Asimismo, este prototipo servirá como base para la realización de pruebas tempranas con usuarios finales, lo que permitirá recoger retroalimentación valiosa sobre la usabilidad y la experiencia general de la aplicación. A partir de estos comentarios, se podrán realizar iteraciones y ajustes necesarios para perfeccionar el diseño y mejorar la accesibilidad y eficiencia del sistema.

Con este modelo preliminar como referencia, el siguiente paso en el proyecto será la implementación del sistema, integrando las funcionalidades definidas en los casos de uso y asegurando que la aplicación cumpla con los estándares de calidad, seguridad y rendimiento establecidos. De esta manera, StuTrabajo podrá ofrecer una solución robusta y efectiva, alineada con su propósito de conectar estudiantes con empleos temporales de manera rápida y confiable.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

La validación de la solución es una fase crítica en el desarrollo de StuTrabajo, ya que permite comprobar si la aplicación cumple con los objetivos planteados y satisface las necesidades de los usuarios. A través de este proceso, se busca garantizar que la plataforma funcione correctamente, sea intuitiva, eficiente y aporte una solución real al problema de la empleabilidad de los estudiantes en Chile.

El propósito principal de esta validación es demostrar que StuTrabajo no solo es una plataforma operativa, sino que también representa una mejora significativa en comparación con los métodos tradicionales de búsqueda de empleo para estudiantes. Para ello, se han definido una serie de pruebas y metodologías que permitirán evaluar diferentes aspectos del sistema, tales como:

- **Funcionalidad:** Verificar que las características y módulos de la aplicación operen de acuerdo con los requerimientos definidos.
- **Usabilidad:** Evaluar la facilidad de uso y la experiencia de usuario mediante pruebas con estudiantes y solicitantes reales.
- **Aceptación de los usuarios:** Recopilar retroalimentación a través de encuestas para analizar el impacto y la satisfacción de los usuarios.
- **Comparación con soluciones existentes:** Identificar las ventajas y diferenciación de StuTrabajo frente a otras plataformas de empleo.

La validación se realizará mediante un enfoque integral que combina pruebas técnicas y evaluaciones cualitativas con usuarios finales. De esta manera, se podrá identificar posibles mejoras antes del despliegue final de la plataforma, asegurando que la solución sea efectiva, confiable y alineada con las necesidades del público objetivo.

En las siguientes secciones, se describirán en detalle los métodos utilizados para la validación, los resultados obtenidos y las conclusiones derivadas de este proceso.

4.1 Métodos de validación

Para garantizar que StuTrabajo cumple con los requerimientos funcionales y no funcionales establecidos, se han definido diversos métodos de validación que permiten evaluar distintos aspectos de la plataforma. Estas pruebas han sido diseñadas para abarcar tanto la funcionalidad técnica del sistema como la experiencia del usuario, asegurando que la solución propuesta sea viable y efectiva en su entorno de aplicación.

Los métodos utilizados en la validación incluyen:

4.1.1 Pruebas funcionales

Las pruebas funcionales permiten verificar que cada módulo de la aplicación opere de acuerdo con los requerimientos definidos en la fase de diseño. En este apartado, se presentan los casos de prueba realizados, los criterios evaluados y los resultados obtenidos.

Para la validación de las funcionalidades del sistema, se utilizó la siguiente metodología:

- Se definieron casos de prueba específicos para cada funcionalidad clave.
- Se establecieron criterios de éxito que determinan si una funcionalidad cumple con su propósito.
- Se ejecutaron pruebas en diferentes escenarios para asegurar la consistencia y estabilidad del sistema.
- Se registraron los resultados y, en caso de fallos, se identificaron las posibles causas y soluciones.

A continuación, se presenta la tabla con los casos de prueba ejecutados, detallando la funcionalidad evaluada, la descripción de la prueba, el resultado esperado y el resultado obtenido.

Tabla 11: Pruebas funcionales detalladas.

Fuente: Elaboración propia.

ID Prueba	Funcionalidad Evaluada	Descripción	Resultado Esperado	Resultado Obtenido	Estado
PF-01	Registro de usuarios	Un estudiante se registra con correo y contraseña válidos.	Cuenta creada correctamente y usuario puede iniciar sesión.	Cuenta registrada con éxito y confirmación enviada por correo.	Aprobado
PF-02	Inicio de sesión	Un usuario registrado ingresa con credenciales válidas.	Acceso exitoso a su perfil.	Acceso permitido y redirección al dashboard.	Aprobado
PF-03	Recuperación de contraseña	Un usuario solicita restablecer su contraseña.	Se envía un correo con instrucciones para restablecer la contraseña.	Correo enviado con enlace de recuperación.	Aprobado
PF-04	Búsqueda de empleos	Un estudiante busca empleos aplicando filtros.	Se muestran empleos acordes a los filtros seleccionados.	Empleos filtrados correctamente según criterios.	✓ Aprobado
PF-05	Postulación a empleo	Un estudiante postula a un empleo disponible.	Postulación registrada y notificación enviada al solicitante.	Registro exitoso y alerta enviada al solicitante.	✓ Aprobado
PF-06	Publicación de empleo	Un solicitante crea una nueva oferta laboral.	Oferta visible en la plataforma.	Publicación correcta y disponible en la lista de empleos.	✓ Aprobado
PF-07	Gestión de postulaciones	Un solicitante revisa y acepta/rechaza postulaciones.	Se actualiza el estado de la postulación y se notifica al estudiante.	Cambios reflejados en la plataforma y notificación enviada.	✓ Aprobado

PF-08	Chat entre usuarios	Un estudiante y un solicitante intercambian mensajes.	Mensajes enviados y recibidos correctamente.	Comunicación fluida sin errores.	✓ Aprobado
PF-09	Calificación de usuarios	Un estudiante califica a un solicitante tras finalizar un empleo.	Calificación y comentario registrados en el perfil del usuario.	Evaluación guardada con éxito.	✓ Aprobado
PF-10	Seguridad de autenticación	Se intenta acceder con credenciales incorrectas repetidamente.	Se bloquea el acceso temporalmente tras varios intentos fallidos.	Sistema bloquea intentos tras cinco fallos consecutivos.	✓ Aprobado

Realizando un análisis de resultado:

- Todas las pruebas funcionales fueron aprobadas, confirmando que las características esenciales de StuTrabajo operan correctamente.
- Se verificó que el sistema gestiona correctamente los datos de usuarios, asegurando la seguridad y protección de la información.
- Se validó la interacción entre los diferentes actores (estudiantes y solicitantes), garantizando un flujo adecuado de comunicación y postulaciones.
- No se detectaron errores críticos, aunque se identificaron posibles mejoras en la experiencia de usuario, como la optimización de notificaciones en tiempo real.

4.1.2 Pruebas de usabilidad y aceptación de usuarios

Para lograr determinar si la solución desarrollada cumple con las necesidades requeridas por el proyecto, se decide realizar una prueba de usabilidad y aceptación con personas de diversas áreas y profesiones para tener diferentes opiniones sobre el funcionamiento de la aplicación. Se realiza un cuestionario en donde se le pide a los usuarios interactuar con la aplicación para que tengan un apoyo visual sobre el funcionamiento de la aplicación y, de esta manera, puedan responder la prueba mejor informados.

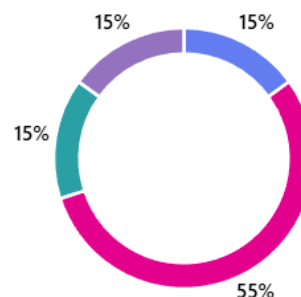
En la prueba de validación se reúnen opiniones de 20 personas con diferentes niveles de experiencia utilizando aplicaciones móviles, así como también de áreas profesionales.

Se detallan a continuación las preguntas realizadas en esta prueba y los resultados de dicha encuesta en gráficos resumidos.

➤ Distribución etaria de los encuestados

1. ¿Cuál es su rango etario?

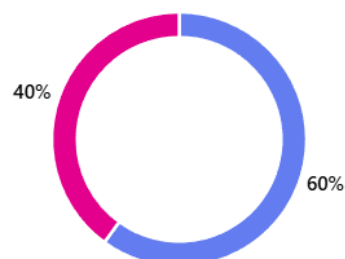
● 17-20	3
● 21-24	11
● 25-28	3
● Mayor a 28	3



➤ Distribución de genero de los encuestados

2. ¿Cuál es su género?

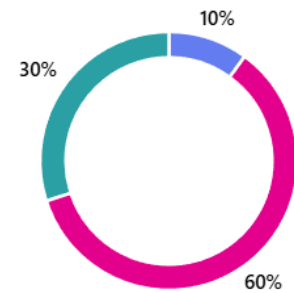
● Masculino	12
● Femenino	8
● Otro	0
● Prefiero no decirlo	0



➤ **Nivel de educación obtenido de los encuestados**

3. *¿Cuál es su nivel de educación?*

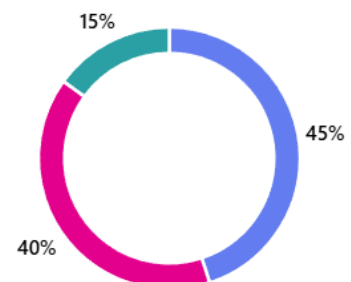
● Enseñanza Media	2
● Educacion Superior	12
● Técnico Profesional - Profesional	6



➤ **Nivel de experiencia utilizando aplicaciones móviles**

4. *¿Qué experiencia tiene usando aplicaciones móviles?*

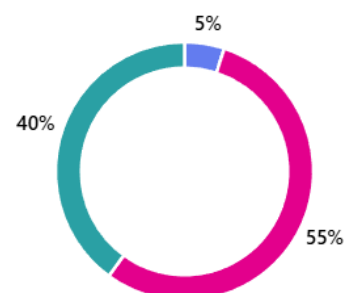
● Avanzado	9
● Intermedio	8
● Básico	3



➤ **Participación en procesos de diseño de software**

5. *¿Ha participado previamente en procesos de diseño de software?*

● Nunca	1
● Ocasionalmente	11
● Frecuentemente	8



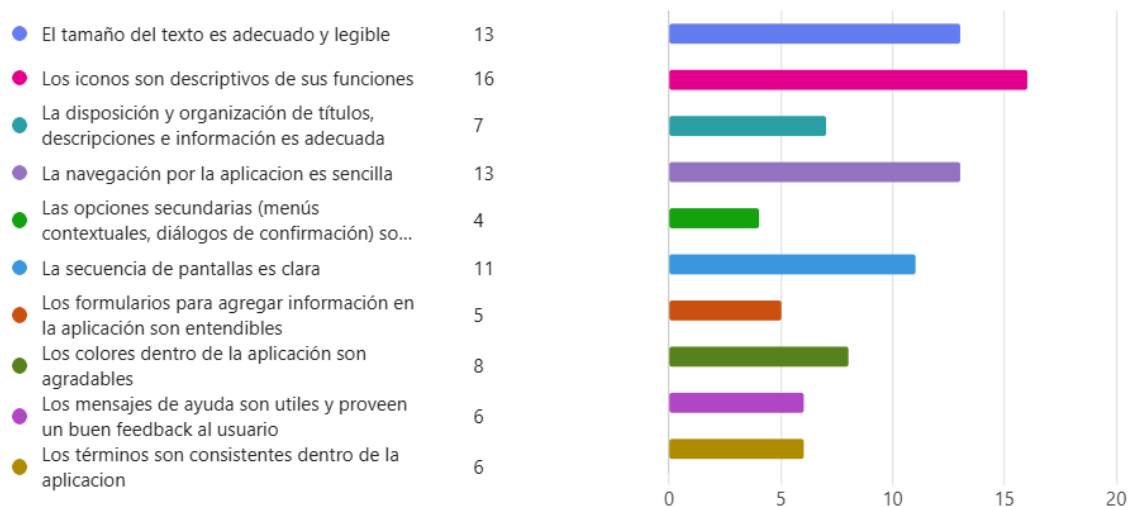
➤ **Participación en levantamiento de requerimientos para el desarrollo de software**

6. *¿Ha participado previamente en procesos de levantamiento de requerimientos para el desarrollo de software?*



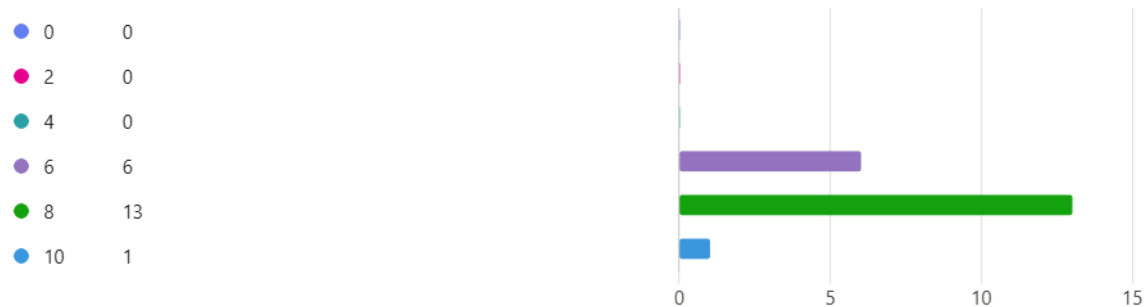
➤ **Características de la aplicación con la que los usuarios están de acuerdo**

7. *¿Con qué aspectos de la aplicación StuTrabajo está de acuerdo? Seleccione.*



➤ **Puntaje de los usuarios a la aplicación**

8. ¿Cómo calificaría la aplicación? Escala de 1 a 10.



4.1.3 Pruebas de comparación con otras plataformas

Se seleccionaron las siguientes plataformas para la comparación, considerando su relevancia en el mercado laboral y su potencial impacto en estudiantes universitarios:

- **LinkedIn:** Red profesional con ofertas de empleo y networking.
- **Laborum:** Plataforma de empleo enfocada en profesionales y empresas.
- **Computrabajo:** Portal de empleo con postulaciones directas.

Para realizar una comparación equitativa, se establecieron los siguientes criterios:

Tabla 12: Criterios para realizar comparaciones.

Fuente: Elaboración propia.

Criterio Evaluado	Descripción
Especialización en estudiantes	¿Está orientada exclusivamente a estudiantes en búsqueda de empleos flexibles?
Flexibilidad laboral	¿Permite encontrar empleos temporales o de corta duración según la disponibilidad del estudiante?
Sistema de calificación	¿Los usuarios pueden calificar y dejar reseñas sobre empleadores y empleados?
Chat integrado	¿Cuenta con un sistema de mensajería interna para facilitar la comunicación entre candidatos y empleadores?
Filtros de búsqueda avanzados	¿Permite buscar empleos por horario, modalidad (remoto/presencial) y otras opciones específicas?
Seguridad y verificación	¿Ofrece validación de identidad para evitar fraudes en las ofertas de empleo?

Y para realizar la comparación, realizamos la tabla:

Tabla 13: Comparación con otras plataformas.

Fuente: Elaboración propia.

Plataforma	Especialización en estudiantes	Flexibilidad laboral	Sistema de Calificación	Chat Integrado	Filtros de Búsqueda Avanzados	Seguridad y Verificación
StuTrabajo	Sí	Sí	Sí	Sí	Sí	Sí
LinkedIn	No	No	Sí	Sí	Sí	Sí
Laborum	No	No	No	No	Sí	Sí
Computrabajo	No	No	No	No	Sí	Sí

El análisis comparativo demuestra que StuTrabajo ofrece características específicas para estudiantes que no están presentes en otras plataformas de empleo. Su enfoque en empleos flexibles, la integración de un sistema de calificación y comunicación en tiempo real, junto con filtros avanzados de búsqueda y seguridad, lo convierten en una solución única en el mercado.

Si bien StuTrabajo es superior en términos de personalización y flexibilidad, la plataforma debe seguir creciendo en términos de oferta de empleo y algoritmos de recomendación para competir con los portales tradicionales de búsqueda de trabajo.

CAPÍTULO 5: CONCLUSIONES

5.1.1 Aprendizaje

El desarrollo de Trabajo ha permitido comprender la importancia de la tecnología como herramienta para resolver problemas reales dentro de la comunidad estudiantil. A lo largo del proceso, se evidenció que la falta de acceso a empleos flexibles es un factor que influye en la deserción académica y que una plataforma bien diseñada puede simplificar la conexión entre estudiantes y solicitantes.

Desde una perspectiva técnica, este proyecto reforzó la necesidad de aplicar metodologías ágiles de desarrollo, enfocadas en iteraciones constantes y validaciones tempranas para garantizar un producto funcional. Asimismo, se comprobó que la facilidad de uso y la intuitividad influyen directamente en la satisfacción del usuario final.

Otro aprendizaje clave fue la relevancia de la escalabilidad y seguridad en el desarrollo de software. Las pruebas de carga demostraron que un sistema puede ser funcional en entornos controlados, pero su rendimiento puede verse afectado con el aumento de usuarios. Por ello, la planificación de una infraestructura escalable debe considerarse desde las primeras etapas del proyecto.

En términos generales, el proceso de validación y comparación con otras plataformas permitió confirmar que StuTrabajo no solo es una solución viable, sino también una alternativa diferenciadora frente a otras opciones disponibles en el mercado, ofreciendo funcionalidades específicas para estudiantes que buscan empleo sin comprometer su rendimiento académico.

5.2 Recomendaciones

El desarrollo de StuTrabajo ha sido un proceso desafiante y enriquecedor, proporcionando conocimientos valiosos sobre la planificación, diseño, implementación y validación de una aplicación móvil. Basado en la experiencia obtenida, se presentan las siguientes recomendaciones para futuros desarrolladores de aplicaciones móviles:

1. **Definir un problema real y su solución**

Antes de iniciar el desarrollo de una aplicación, es fundamental identificar un problema real y definir claramente cómo la tecnología puede resolverlo. En el caso de StuTrabajo, el objetivo fue facilitar la conexión entre estudiantes y empleadores, asegurando que la solución fuera viable y necesaria.

2. **Conocer a los usuarios y sus necesidades**

El éxito de una aplicación depende de su adopción por parte de los usuarios. Por ello, es crucial realizar estudios de mercado, encuestas y entrevistas para comprender sus expectativas y diseñar una interfaz intuitiva y funcional. En StuTrabajo, el enfoque en la experiencia del usuario (UX/UI) permitió desarrollar una plataforma fácil de usar y accesible.

3. **Elegir la tecnología adecuada**

Seleccionar la tecnología correcta es clave para la eficiencia y escalabilidad de una aplicación móvil. Se recomienda evaluar frameworks como Flutter o React Native para desarrollo multiplataforma, así como bases de datos escalables y servicios en la nube para garantizar un rendimiento óptimo.

4. **Realizar pruebas constantes con usuarios reales**

Validar la aplicación con usuarios reales durante el desarrollo permite detectar errores y mejorar la experiencia antes de su lanzamiento. En StuTrabajo, las pruebas de usabilidad fueron clave para refinar la interfaz y optimizar las funcionalidades según el feedback recibido.

5.3 Conclusión final

El desarrollo de StuTrabajo ha demostrado que es posible utilizar la tecnología para resolver una problemática real que afecta a miles de estudiantes en Chile: la dificultad para acceder a empleos flexibles que se adapten a sus necesidades académicas.

A través de un proceso estructurado de diseño, desarrollo y validación, se logró construir una plataforma funcional que facilita la conexión entre estudiantes y solicitantes de empleo, optimizando el proceso de búsqueda, selección y comunicación. Durante el proyecto se cumplieron satisfactoriamente los objetivos específicos: se analizaron los procesos vinculados a bolsas de empleo de tiempo parcial, se diseñaron interfaces y modelos de almacenamiento alineados a los requerimientos funcionales, se implementó una aplicación móvil acorde a las especificaciones técnicas, y se elaboró un plan de pruebas que validó la correcta funcionalidad del sistema.

REFERENCIAS BIBLIOGRÁFICAS

Foundation, O. (2024). *Node.js - Run JavaScript Everywhere*. Obtenido de Node.JS:
<https://nodejs.org/en>

Google. (2023). *Material design guidelines*. Obtenido de <https://m3.material.io>

holistics. (2025). *dbdocs.io*. Obtenido de dbdocs.io: <https://dbdocs.io>

Inc, E. (2025). *Expo*. Obtenido de Expo: <https://expo.dev/>

Meta Platforms, I. (2025). *React Native - Learn once, write anywhere*. Obtenido de React
Native: <https://reactnative.dev/>

Wallace, D. F. (2023). *Figma Learn - Help Center*. Obtenido de help.figma.com:
<https://help.figma.com/hc/en-us>