

Índice

[1. Introducción 2](#_heading=h.iultsaozl4ff)

[2. Metodología de Desarrollo de Software y Arquitectura del Software. 3](#_heading=h.hk87b2yvgivh)

[2.1 Arquitectura del Software: 3](#_heading=h.hnmk7frvvzja)

[2.2 Patrón de diseño arquitectónico: 3](#_heading=h.okt49ktv1d0j)

[2.3 Frameworks utilizados (Front y Backend): 3](#_heading=h.l66bu6sxnt9o)

[2.4 Metodología de gestión del proyecto: 3](#_heading=h.8os88mg9sibf)

[3. Atributos de Calidad (ISO 25010) 4](#_heading=h.atkfx268l463)

[4. Plan de Riesgos 6](#_heading=h.h4m3neg7w17)

[5. Plan de Pruebas 7](#_heading=h.ajaaty8uwh5z)

[5.1 Introducción: 7](#_heading=h.9tqv135lznki)

[5.2 Recursos: 7](#_heading=h.i1vh2ohzqywi)

[5.3 Alcance: 7](#_heading=h.34ve1hgdhs4x)

[5.4 Fuera del Alcance: 7](#_heading=h.pre2k6d1uhfn)

[5.5 Pruebas de Rendimiento: 7](#_heading=h.hcjuox3thdyt)

[5.6 Pruebas de Usabilidad: 7](#_heading=h.hsh89wnn8ufq)

[5.7 Pruebas de Aceptación (UAT): 7](#_heading=h.dm8hy0hg0qvn)

[5.8 Infraestructura: 8](#_heading=h.32bd1ylz2mpc)

[5.9 Suposiciones: 8](#_heading=h.qb4put4s1r27)

[6. Casos de Prueba 8](#_heading=h.kqz71jpjs1mz)

# Introducción

El proyecto “CARWASH VIP” fue creado con el objetivo de mejorar la gestión y organización de una empresa dedicada al lavado de autos. A través de un sistema informático, se busca facilitar tareas como la creación de usuarios, la programación de servicios, la asignación de personal y recursos, la gestión de capacitaciones, bonificaciones y reportes administrativos.

Para su desarrollo, se utilizó una arquitectura basada en microservicios, lo que permite dividir el sistema en partes independientes que se pueden trabajar y mantener por separado. Esto ayuda a que el sistema sea más flexible, escalable y fácil de actualizar.

Durante el proceso se aplicó la metodología ágil Scrum, que permite organizar el trabajo en etapas cortas llamadas sprints. Esto ayudó a avanzar de forma ordenada, haciendo entregas parciales del sistema y adaptándonos a posibles cambios.

También se consideraron atributos de calidad basados en la norma ISO 25010, para asegurar que el sistema sea confiable, seguro, rápido y fácil de usar. Este informe incluye la arquitectura del software, el enfoque de desarrollo, el plan de riesgos, pruebas, casos de uso, control de versiones y los diseños de pantalla (mockups) que muestran cómo funcionará el sistema.

# Metodología de Desarrollo de Software y Arquitectura del Software.

## Arquitectura del Software:

Se utilizó una arquitectura basada en microservicios, que permite modularizar el sistema en componentes independientes, favoreciendo la escalabilidad, el mantenimiento y la reutilización del código. Esta arquitectura facilita además el trabajo colaborativo, ya que diferentes equipos pueden desarrollar y desplegar servicios por separado.

## Patrón de diseño arquitectónico:

El patrón seleccionado es el patrón multicapa, que permite separar la lógica de presentación, negocio y acceso a datos, mejorando la organización del código y facilitando las pruebas y el mantenimiento.

## Frameworks utilizados (Front y Backend):

Backend: Se utilizó Spring Boot como framework principal para el desarrollo del backend, lo que permite crear servicios REST de forma rápida, segura y con integración a bases de datos.

Frontend: (Si corresponde) Se planea la integración futura con un frontend moderno como React o Angular, aunque actualmente el sistema solo expone APIs REST.

## Metodología de gestión del proyecto:

Se aplicó una metodología ágil basada en Scrum, con reuniones semanales para la planificación de sprints, definición de historias de usuario, backlog priorizado y revisión iterativa del avance.

# Atributos de Calidad (ISO 25010)

| **Característica o Atributo de Calidad** | **Describa Alcance** | **Identifique Indicadores** | **Justifique si se cumple con el Indicador** |
| --- | --- | --- | --- |
| Funcionalidad | Asegurar que todas las funciones requeridas por los usuarios estén implementadas y operen según lo especificado. | Cobertura de requisitos funcionales | Se implementaron todos los módulos definidos (usuarios, servicios, reportes, etc.) y se verificó su correcto funcionamiento. |
| Precisión en la salida de datos | Las funciones entregan datos esperados y consistentes, comprobados mediante casos de prueba. |
| Cumplimiento normativo | El sistema cumple con las normativas de seguridad y protección de datos |
| Confiabilidad | Garantizar la precisión en consultas, tiempo de actividad continuo y manejo correcto de los datos. | Tasa de tiempo activo ≥ 99.9% | Se emplean servidores con alta disponibilidad y respaldo automático para garantizar la continuidad del servicio. |
| Errores de disponibilidad | Se monitorean continuamente los servicios para detectar y corregir fallos rápidamente. |
| Precisión en reportes | Se han implementado validaciones cruzadas en los reportes para asegurar consistencia y fidelidad de los datos. |
| Usabilidad | Proporcionar una interfaz intuitiva para usuarios y colaboradores con fácil navegación. | Tiempo promedio de aprendizaje | Se diseñó una interfaz con íconos claros y guías interactivas, reduciendo la curva de aprendizaje. |
| Número de clics por tarea | Las tareas más frecuentes requieren como máximo tres clics, optimizando la experiencia del usuario. |
| Encuestas de satisfacción | Se aplicaron encuestas después de sesiones de prueba y se ajustaron elementos según el feedback recibido. |
| Eficiencia | Garantizar tiempos de respuesta bajos y operaciones rápidas, incluso bajo alta carga. | Tiempo de carga < 3s | El sistema utiliza consultas optimizadas y almacenamiento en caché para minimizar los tiempos de carga. |
| Rendimiento bajo carga | Pruebas de estrés demostraron que mantiene su rendimiento con múltiples usuarios concurrentes. |
| Tiempos de respuesta por módulo | Se midió el tiempo de respuesta individual por módulo asegurando eficiencia en todas las funciones. |
| Mantenibilidad | Facilitar actualizaciones, correcciones de errores y mejoras evolutivas. | Tiempo promedio de resolución | Los errores son corregidos en menos de 24 horas gracias a una estructura modular y clara. |
| Frecuencia de actualizaciones | Se programan revisiones y actualizaciones quincenales para mantener el sistema vigente. |
| Calidad del código | El código sigue estándares como PEP8 y está documentado, facilitando su comprensión y modificación. |
|  |  |
| Portabilidad | Permitir la ejecución del sistema en distintos entornos y dispositivos compatibles. | Compatibilidad con sistemas operativos | La aplicación fue probada en Windows, Linux y macOS garantizando su funcionalidad. |
| Navegadores | Es compatible con Chrome, Firefox y Edge mediante estándares web actuales. |
| Dispositivos móviles | El diseño es responsivo y se adapta correctamente a pantallas móviles y tablets. |

# Plan de Riesgos

| **N°** | **Riesgo identificado** | **Categoría PESTEL** | **Impacto (1-5)** | **Probabilidad (1-5)** | **Nivel de riesgo  (I x P)** |
| --- | --- | --- | --- | --- | --- |
| R1 | |  | | --- |  | Cambios en los requerimientos del cliente | | --- | | Tecnológica | 4 | 4 | 16 |
| R2 | |  | | --- |  | Falta de experiencia del equipo en Spring Boot/React | | --- | | Social/Tecnológica | 3 | 3 | 9 |
| R3 | Retrasos por falta de coordinación entre integrantes | Social | 3 | 4 | 12 |
| R4 | |  | | --- |  | Pérdida de datos por errores en pruebas o desarrollo | | --- | | Tecnológica | 5 | 2 | 10 |
| R5 | Baja disponibilidad de recursos (notebook, internet) | Económica/Tecnológica | 3 | 3 | 9 |

| PROBABILIDAD | Constante (5) |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Moderado (4) |  |  |  | **R1** |  |
| Ocasional (3) |  |  | **R2/R5** | **R3** |  |
| Posible (2) |  |  |  |  | R4 |
| Improbable (1) |  |  |  |  |  |
|  |  | Insignificante (1) | Menor (2) | Crítico (3) | Mayor (4) | Catastrófico (5) |
|  |  | IMPACTO | | | | |

# Plan de Pruebas

## Introducción:

Este plan de pruebas establece la base para identificar qué aspectos del sistema serán evaluados, qué funcionalidades entran dentro y fuera del alcance de las pruebas, y cuáles son los posibles riesgos y suposiciones. El enfoque será ágil, utilizando la metodología Scrumban, lo que nos permite adaptar y priorizar pruebas según los cambios en los requerimientos durante el desarrollo del sistema WashVIP, el cual automatiza la gestión de servicios, colaboradores, capacitaciones, bonificaciones y reportes de una empresa de aseo industrial.

## Recursos:

| **Tester** | **% Participación** |
| --- | --- |
| Jaime Álvarez A. | 50 % |
| Lucas Fuentes C. | 100% |

## Alcance:

El presente documento tiene como objetivo definir y planificar las pruebas necesarias para garantizar la calidad del sistema WashVIP. Se cubrirán funcionalidades críticas como autenticación, programación de servicios, asignación de recursos, generación de reportes y gestión de usuarios. Se dará prioridad a las funcionalidades con mayor impacto en la experiencia del cliente y operatividad del sistema.

## Fuera del Alcance:

- Pruebas de seguridad ofensiva (pentesting).

- Pruebas de funcionalidades externas no incluidas en el desarrollo.

- Traducción o localización del sistema a otros idiomas.

## Pruebas de Rendimiento:

- Tiempo de respuesta en login (< 3 segundos).

- Carga máxima simultánea de usuarios accediendo a servicios.

- Tiempo promedio en asignación de usuarios y recursos a un servicio.

## Pruebas de Usabilidad:

- Facilidad de uso de la interfaz de usuario (UI).

- Navegación intuitiva entre secciones y módulos del sistema.

- Claridad en los formularios de programación, asignación y reportes.

## Pruebas de Aceptación (UAT):

- Facilidad de uso de la interfaz de usuario (UI).

- Navegación intuitiva entre secciones y módulos del sistema.

-Claridad en los formularios de programación, asignación y reportes.

## Infraestructura:

Las pruebas se ejecutarán en entornos locales de desarrollo, con instancias del backend desplegadas en servidores locales o nube (dependiendo de la etapa del proyecto). Se utilizarán herramientas como Postman para pruebas manuales de API, Jest (si se implementa en frontend), y base de datos MySQL.

## Suposiciones:

- Todas las funcionalidades están integradas y disponibles para probar al momento de ejecutar las pruebas.

- Los datos de prueba están preparados y cargados correctamente en el sistema.

- Los módulos han pasado por pruebas unitarias antes de las pruebas funcionales e integradas.

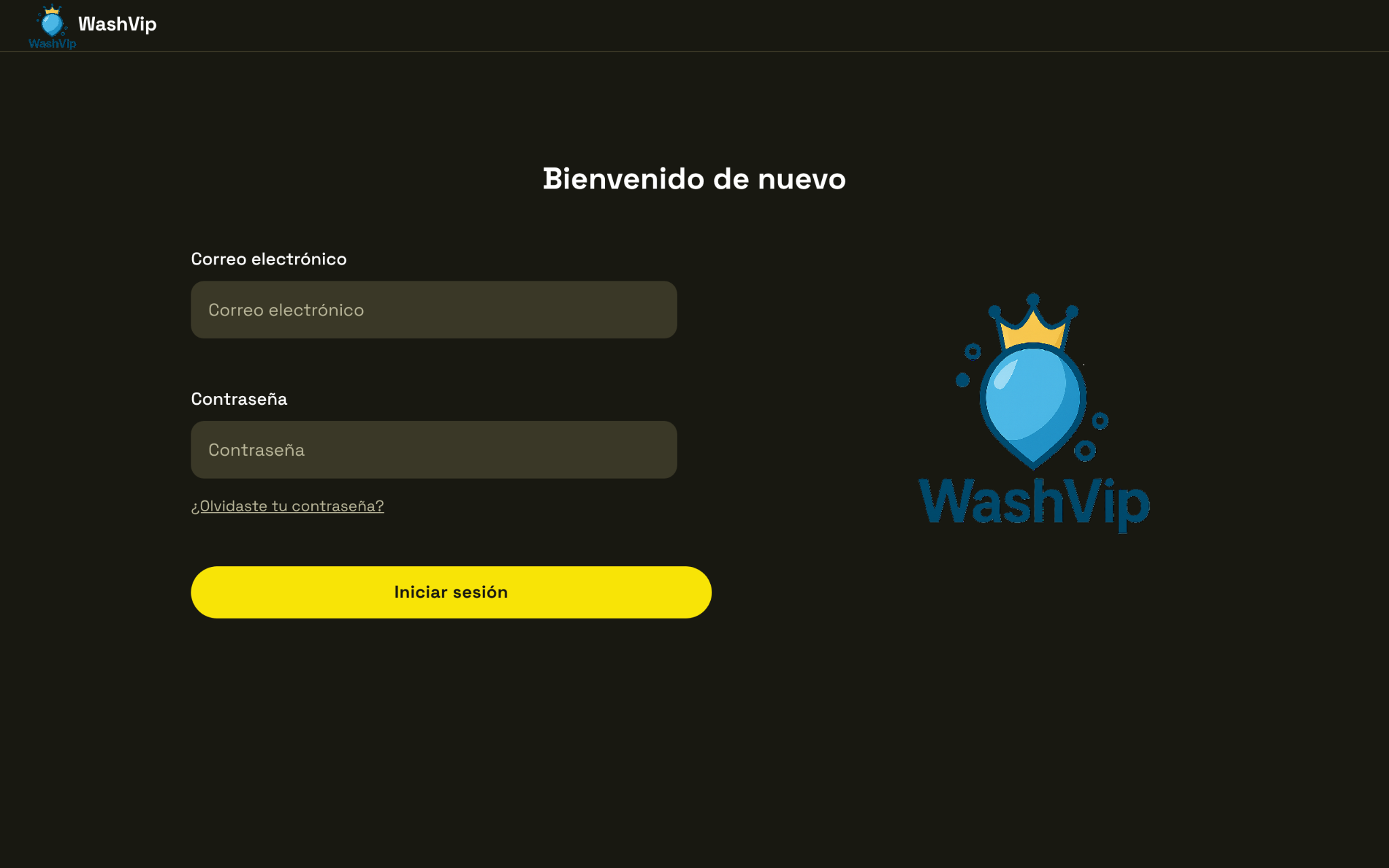
- Los entornos de prueba reproducen condiciones reales de operación.

# 6. Casos de Prueba

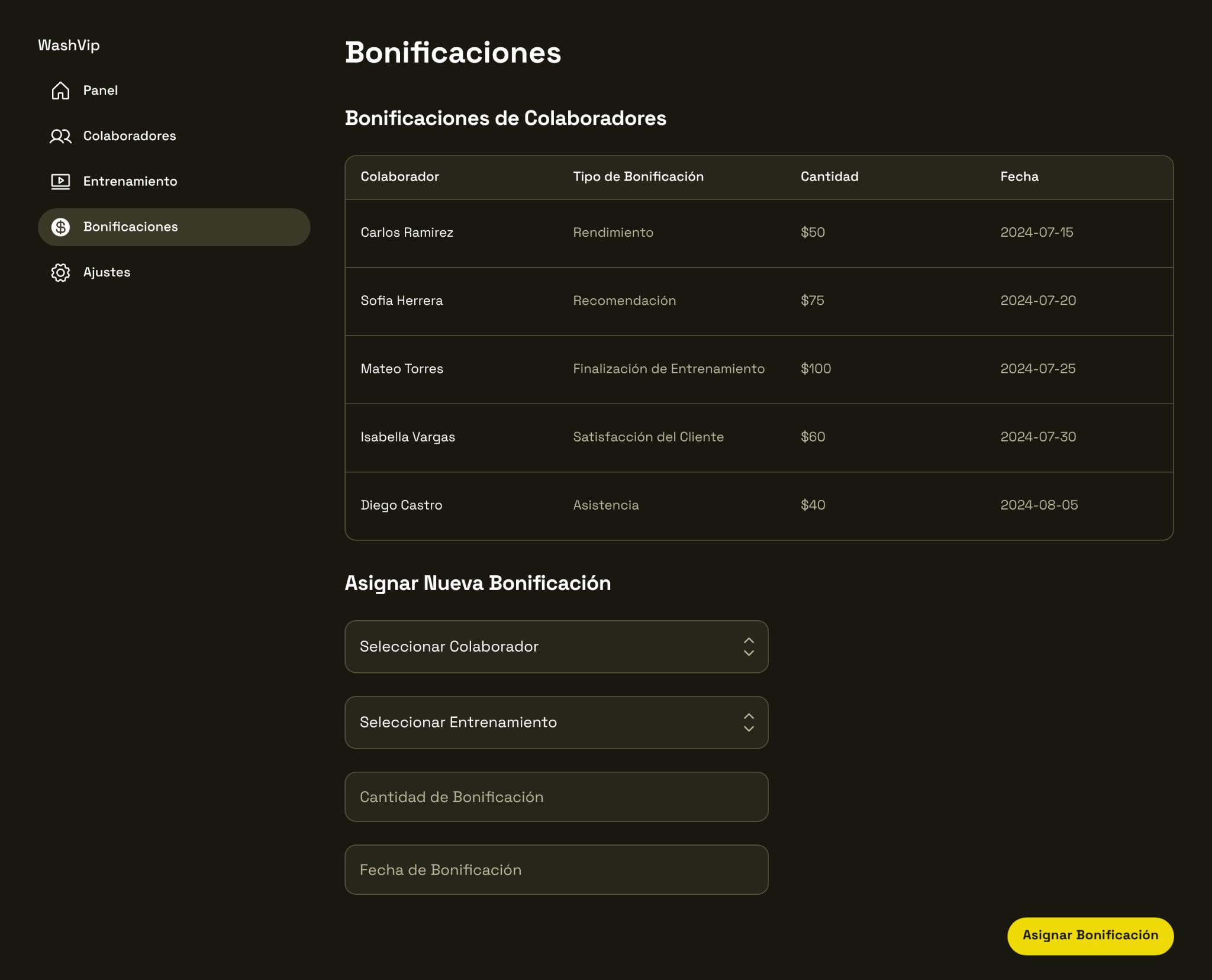
En el siguiente archivo se encuentran distintos casos de prueba.



# 7. Mockups







(los demás mockups se encuentran en la carpeta encontrada en el repositorio)

# 8. Github

https://github.com/DanteEII1/Ing\_Software.git