



## Sistema de logística y resguardo de diversos productos

### **ALUMNO:**

*Canales Bernal Carlos Natividad*

De La Torre López Lilian Denisse

García Arreola Howard Isaí

Paredes Nevárez Alexis Omar

Tarango Méndez Derian Omar

Zamorano Palma Jose Hector

### **ASIGNATURA:**

Desarrollo Móvil Integral

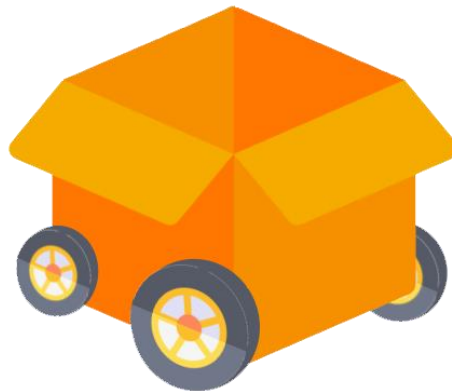
### **GRUPO:**

10-B

### **FECHA DE CREACIÓN:**

Enero del 2025

**ESTANDAR IEEE-830**



WareBox  
Logistics

---

**ESPECIFICACIÓN DE  
REQUERIMIENTOS DE SOFTWARE**

**Proyecto: Sistema 3PL: Traslado y  
Almacenamiento**

# **1. INTRODUCCIÓN**

## **1.1 PROPÓSITO**

### **1.1.1 Propósito del documento**

Este documento describe la arquitectura, tecnologías y patrones de diseño utilizados en el desarrollo del sistema, proporcionando una guía clara sobre su estructura y funcionamiento. Se establecen las bases para su implementación, mantenimiento y escalabilidad, asegurando una mejor organización y comprensión del sistema.

### **1.1.2 Audiencia a la que va dirigido**

Este documento está dirigido a desarrolladores, arquitectos de software, gestores del proyecto y cualquier otra persona involucrada en el diseño, desarrollo e implementación del sistema. También puede servir como referencia para futuras mejoras y mantenimientos.

## **1.2 ALCANCE**

El sistema desarrollado abarcará tanto una aplicación web como una aplicación móvil, utilizando arquitecturas adecuadas para cada una de estas plataformas. La aplicación web se desarrollará con ReactJS y se basará en la arquitectura MVC, mientras que la aplicación móvil se desarrollará con Jetpack Compose en Kotlin, siguiendo la arquitectura MVVM. Además, se contará con una API en Laravel para la gestión de datos y la comunicación entre ambas plataformas.

### 1.3 PERSONAL INVOLUCRADO

<b>Nombre</b>	Carlos Natividad Canales Bernal
<b>Rol</b>	Developer
<b>Categoría profesional</b>	Desarrollador
<b>Responsabilidades</b>	Front
<b>Información de contacto</b>	0321101710@ut-tijuana.edu.mx
<b>Aprobación</b>	

<b>Nombre</b>	Lilian Denisse de la Torre López
<b>Rol</b>	Programador
<b>Categoría profesional</b>	Analista
<b>Responsabilidades</b>	Programación de módulos
<b>Información de contacto</b>	0321101300@ut-tijuana.edu.mx
<b>Aprobación</b>	

<b>Nombre</b>	García Arreola Howard Isaí
<b>Rol</b>	Developer
<b>Categoría profesional</b>	Desarrollador
<b>Responsabilidades</b>	Back y Front
<b>Información de contacto</b>	0321101266@ut-tijuana.edu.mx
<b>Aprobación</b>	

<b>Nombre</b>	Paredes Nevárez Alexis Omar
<b>Rol</b>	Developer
<b>Categoría profesional</b>	Analista
<b>Responsabilidades</b>	Back y Front
<b>Información de contacto</b>	0321101363@ut-tijuana.edu.mx
<b>Aprobación</b>	

Nombre	Tarango Méndez Derian Omar
Rol	Programador
Categoría profesional	Analista
Responsabilidades	Programación de Módulos
Información de contacto	0320127922@ut-tijuana.edu.mx
Aprobación	

Nombre	Zamorano Palma Jose Hector
Rol	Programador
Categoría profesional	Analista
Responsabilidades	Diseñador Grafico
Información de contacto	0321101251@ut-tijuana.edu.mx
Aprobación	

## 1.4 Definición, acrónimos y abreviaturas

### DEFINICIONES

**Actualización;** Proceso de mejora o modificación de un sistema, software o base de datos para corregir errores, agregar nuevas funciones o mejorar el rendimiento.

**BackUp;** Copia de seguridad de datos o archivos con el propósito de prevenir la pérdida de información en caso de fallos, errores o ataques.

**Base de Datos;** Conjunto organizado de datos almacenados digitalmente que permite la gestión, recuperación y manipulación eficiente de la información.

**Botón;** Elemento interactivo en una interfaz gráfica que ejecuta una acción cuando el usuario lo selecciona o presiona.

**Conexión;** Establecimiento de comunicación entre dos o más sistemas, dispositivos o redes para el intercambio de información.

**Interfaz;** Medio de comunicación entre un usuario y un sistema, ya sea una aplicación, un dispositivo o una página web. Puede ser gráfica (GUI) o basada en texto (CLI).

**Internet;** Red global de computadoras interconectadas que permite la transmisión de datos y la comunicación a nivel mundial.

**Login;** Proceso de autenticación mediante el cual un usuario ingresa a un sistema proporcionando credenciales como nombre de usuario y contraseña.

**Password;** Clave secreta utilizada para autenticar el acceso de un usuario a un sistema o cuenta.

**Protocolo; Conjunto** de reglas y estándares que rigen la comunicación entre dispositivos o sistemas en una red.

Ejemplo: HTTP, TCP/IP.

**Sistema Operativo;** Software fundamental que administra los recursos de hardware y software de un dispositivo, permitiendo la ejecución de programas.

Ejemplos: Windows, Linux, macOS.

**Sitio Web;** Conjunto de páginas web relacionadas y accesibles a través de Internet mediante un dominio o dirección web.

**PostgreSQL;** Sistema de gestión de bases de datos relacional de código abierto, conocido por su estabilidad, escalabilidad y cumplimiento con estándares SQL.

**Tabla;** Estructura dentro de una base de datos que organiza los datos en filas y columnas, permitiendo su almacenamiento y consulta eficiente.

**ReactJS;** Biblioteca de JavaScript para construir interfaces de usuario

**Jetpack Compose;** Kit de herramientas moderno de UI para Android basado en Kotlin

**Kotlin;** Lenguaje de programación para desarrollo en Android

**Laravel;** Framework de PHP para desarrollo de aplicaciones web

## ACRONIMOS

- **3PL** – *Third-Party Logistics* (Logística de Terceros)
- **API** – *Application Programming Interface* (Interfaz de Programación de Aplicaciones)
- **MAPS API** – API utilizada para la integración de mapas interactivos y geolocalización
- **GPS** – *Global Positioning System* (Sistema de Posicionamiento Global)
- **MVC** – *Model-View-Controller* (Modelo-Vista-Controlador)
- **MVVM** – *Model-View-ViewModel* (Modelo-Vista-ViewModel)

**1.5 Referencias.** - Este proyecto no cuenta con referencias anteriores, este es un proyecto original.

## 1.6 Resumen

El SRS está compuesto de la siguiente manera:

- **Introducción:** En esta sección se detalla los objetivos que tiene el SRS y de nuestro sistema en forma general.
- **Descripción General:** Describe una perspectiva general del producto a desarrollarse, como también las características del usuario y las limitaciones que podría tener.
- **Requerimientos Específicos:** Muestra paso a paso todos los requerimientos que el usuario desea en el producto final.

## 2 DESCRIPCIÓN GENERAL

### 2.1 PERSPECTIVA DEL PRODUCTO

WareBox es un sistema 3PL (third party logistics) el brinda dos servicios principales:

- El primero es el servicio de transporte, en donde los servicios de monitoreo WareBox, se visualiza el transporte de los bienes de cualquier empresa en necesidad de un origen y un destino dentro del estado de Baja California Tijuana, se ofrece la facilidad de supervisar el viaje y el control de los diferentes cargamentos.
- El segundo servicio incluye el primer servicio adicionando el uso de los almacenes de WareBox para guardar los bienes de alguna empresa, para ello se le dan un par de credenciales a la empresa que solicite el servicio para poder monitorear su inventario dentro de nuestros almacenes. El flujo general de este servicio comienza por recoger y transportar los bienes de un origen a un almacén de nuestras instalaciones para poder ser guardado, posteriormente, cuando la empresa necesite dispersar sus bienes WareBox se encarga de transportar los bienes a los destinos solicitados.

## 2.2 FUNCIONALIDAD DEL PRODUCTO

El sistema 3PL WearBox debe ofrecer las siguientes funcionalidades principales:

### 1. Gestión de Rutas

1. Crear rutas principales y secundarias en un mapa interactivo.
2. Establecer puntos de origen y destino (almacenes, sucursales, ...).
3. Registrar rutas alternas en caso de contingencias.
4. Visualizar rutas con detalles como distancia y tiempo estimado.

### 2. Gestión de sedes de distribución (almacén)

1. Registrar sedes de distribución (los almacenes de WareBox que guardan los bienes de nuestros clientes con el segundo servicio).

### 3. Gestión de Puertos



1. Establecer la cantidad de puertos de carga / descarga que contiene cada sede de distribución (almacén).
2. Gestionar cuando está en uso y cuando este desocupado dichos puertos de carga / descarga.

#### **4. Gestión de Cajones de los Trailers**

1. Establecer la cantidad de cajones para estacionar los trailers dentro de cada sede de distribución (cuando no se estén utilizando).
2. Gestionar cuando está en uso y cuando este desocupado cada uno de los cajones.

#### **5. Gestión de Viajes**

5. Asignar viajes a choferes para fechas específicas.
6. Registrar los productos transportados en cada viaje.
7. Registrar camiones y remolques asignados para cada viaje.
8. Establecer notificaciones para choferes sobre viajes asignados.

#### **6. Gestión de Productos**

9. Registrar productos y sus detalles (tipo, peso, cantidad).
10. Asociar productos a sus dueños (el cliente que requirió el servicio).
11. Asociar productos a viajes específicos (traslado).
12. Controlar el inventario de productos después del traslado.

#### **7. Aplicación Móvil para Choferes y almacenistas**

15. Mostrar los viajes asignados diariamente.
16. Visualizar un mapa con la ruta que debe de tomar (MAPS API).
17. Utilizar el GPS del móvil para mostrar en que parte se encuentra el chofer y se pueda apegar a la ruta.

- 18. Mostrar información del camión y remolque asignados.
- 19. Permitir a los choferes ver su perfil y detalles de contacto.
- 20. El sistema móvil permite registrar cajas con sus productos dentro su interior al igual que su volumen y peso que ocupan. El sistema les otorga un identificador único para su rastreo dentro y fuera del almacén de WareBox.

## 8. Gestión Climática

- 20. Integrar una API de clima para mostrar condiciones meteorológicas en tiempo real.
- 21. Sugerir cambios en rutas según el clima.
- 22. Mostrar alertas climáticas relevantes para los viajes programados.

## 9. Administración General

- 23. Gestión de usuarios con roles (administradores, operadores, choferes, ...).
- 24. Control de accesos basado en roles.

## 10. Seguimiento y Monitoreo

- 28. Rastreo en tiempo real de los camiones en el mapa.

## 11. Dispatch

- 29. Dentro de la aplicación móvil, el chofer puede contactar a dispatch para comunicar una problemática o algo que pueda llegar a suceder en el transcurso del traslado de mercancía.

## 12. Capacidad y Volúmenes

- 30. El sistema guarda el volumen de capacidad y peso total de carga de los trailers con el fin de saber la carga máxima que pueden ocupar.

32. Dentro del almacén se utilizan racks con las mismas dimensiones y especificaciones, con el fin de simplificar el proceso de saber la capacidad de los almacenes. Las cantidades de los racks es la incógnita que el sistema guarda para saber la capacidad total del almacén.

## REQUERIMIENTOS FUNCIONALES

### Gestión de Rutas

Identificador del Requerimiento RF-01	
Nombre del Requerimiento	Crear rutas principales y secundarias en un mapa interactivo
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe permitir la creación de rutas principales y secundarias directamente en un mapa interactivo.
Características del Requerimiento	El mapa debe ser dinámico y permitir definir puntos intermedios entre el origen y destino. Los usuarios deben poder guardar las rutas creadas para su uso posterior.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El usuario debe estar autenticado y tener permisos de planificación.

Identificador del Requerimiento RF-02	
Nombre del Requerimiento	Establecer puntos de origen y destino (almacenes, sucursales)
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Los usuarios deben poder definir puntos de origen y destino para las rutas, indicando almacenes o sucursales registrados previamente.
Características del Requerimiento	Cada punto debe estar asociado a una ubicación geográfica exacta en el mapa. Se deben mostrar detalles del almacén o sucursal seleccionados.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Los almacenes y sucursales deben estar registrados previamente.

Identificador del Requerimiento RF-03	
Nombre del Requerimiento	Registrar rutas alternas en caso de contingencias

Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe permitir la creación y registro de rutas alternas asociadas a una ruta principal en caso de contingencias.
Características del Requerimiento	Las rutas alternas deben estar vinculadas a una ruta principal y deben incluir información como tiempo y distancia estimada.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Debe existir una ruta principal registrada.

Identificador del Requerimiento	RF-04
Nombre del Requerimiento	Visualizar rutas con detalles como distancia y tiempo estimado
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe permitir visualizar las rutas registradas con información detallada, incluyendo distancia total y tiempo estimado.
Características del Requerimiento	Las rutas deben mostrarse en el mapa interactivo con etiquetas de distancia y tiempo estimado. Debe ser posible filtrar por rutas principales o alternas.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Debe existir al menos una ruta registrada.
Postcondición	Las rutas deben ser visibles en el sistema con la información solicitada.

## Gestión de Viajes

Identificador del Requerimiento RF-05	
Nombre del Requerimiento	Asignar viajes a choferes para fechas específicas
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe permitir asignar viajes a los choferes, especificando fechas y horas para cada traslado.
Características del Requerimiento	Cada viaje debe incluir la fecha de salida, hora estimada, destino y chofer asignado. Los datos deben ser modificables en caso de cambios antes del inicio del viaje.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El chofer debe estar registrado en el sistema y disponible para la fecha asignada.
Postcondición	El viaje debe quedar correctamente asociado al chofer y registrado en el sistema.

Identificador del Requerimiento RF-06	
Nombre del Requerimiento	Registrar los productos transportados en cada viaje
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe permitir registrar los productos transportados en cada viaje, especificando información relevante.
Características del Requerimiento	Cada producto debe estar relacionado con un viaje específico y registrarse en un listado detallado. Se debe incluir información de inventario para facilitar el control antes y después del traslado.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Los productos deben estar registrados previamente en el sistema.
Postcondición	Los productos deben quedar asociados al viaje correspondiente, con su información almacenada.

Identificador del Requerimiento RF-07	
Nombre del Requerimiento	Registrar camiones y remolques asignados para cada viaje
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe permitir asignar camiones y remolques a los viajes, registrando detalles como placas, modelo y capacidad.
Características del Requerimiento	Cada viaje debe incluir información sobre el camión y remolque asignados, y se debe validar que estén disponibles para la fecha del traslado.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Los camiones y remolques deben estar registrados previamente en el sistema.
Postcondición	La asignación del camión y remolque debe quedar registrada para cada viaje.

Identificador del Requerimiento RF-08	
Nombre del Requerimiento	Establecer notificaciones para choferes sobre viajes asignados
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe enviar notificaciones a los choferes con detalles de los viajes asignados, incluyendo fecha, hora, e información de en que puerto cargara y de donde sacara el remolque.
Características del Requerimiento	Las notificaciones deben enviarse en tiempo real.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El chofer debe estar registrado en el sistema y contar con un dispositivo para recibir notificaciones.
Postcondición	El chofer debe recibir las notificaciones en su aplicación móvil o correo electrónico.

Identificador del Requerimiento RF-09	
Nombre del Requerimiento	Visualización de rutas asignadas según el chofer logeado
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe mostrar las diferentes ordenes de trabajo pendientes que se le han asignado por parte de los de logística.
Características del Requerimiento	
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	

## Gestión de Productos

Identificador del Requerimiento RF-10	
Nombre del Requerimiento	Registrar productos y sus detalles
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe permitir registrar productos con sus atributos principales.
Características del Requerimiento	Los productos deben incluir campos como tipo, descripción, peso, cantidad y unidad de medida. Debe ser posible editar o eliminar los registros en caso de errores.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El sistema debe contar con permisos para que el usuario pueda registrar nuevos productos.
Postcondición	Los productos quedan registrados en la base de datos, listos para ser asociados a viajes o almacenamientos.



Identificador del Requerimiento	RF-11
Nombre del Requerimiento	Asociar productos a viajes específicos
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe permitir asociar productos registrados previamente a un viaje específico, indicando las cantidades transportadas.
Características del Requerimiento	Cada viaje debe tener un listado de productos asociados, especificando cantidad y peso total. Los datos deben actualizarse en tiempo real para reflejar la asignación.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Los productos deben estar registrados en el sistema, y el viaje debe estar planificado.
Postcondición	Los productos quedan vinculados al viaje, y el inventario se actualiza en consecuencia.

Identificador del Requerimiento	RF-12
Nombre del Requerimiento	Controlar el inventario de productos después del traslado
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe gestionar el inventario de productos, reflejando las cantidades disponibles después de cada traslado.
Características del Requerimiento	El sistema debe registrar el movimiento de inventario, actualizando las existencias en tiempo real según los productos asignados a viajes o entregados en destinos finales.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Los productos deben estar registrados en el inventario, y el traslado debe estar asociado a un viaje.
Postcondición	El inventario refleja correctamente las cantidades de productos antes y después del traslado.

**Gestión de sedes de Distribución**

Identificador del Requerimiento RF-13	
Nombre del Requerimiento	Registrar sedes de distribución
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Permitir registrar las sedes de distribución, incluyendo información como ubicación, capacidad y otros datos relevantes.
Características del Requerimiento	<input type="checkbox"/> Debe almacenar información detallada. <input type="checkbox"/> Accesible solo por usuarios con rol de administrador.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El usuario debe autenticarse con un rol de administrador.
Postcondición	

Identificador del Requerimiento RF-14	
Nombre del Requerimiento	Establecer la cantidad de cajones para estacionar los tráilers
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Permitir a los administradores definir el número de cajones disponibles para estacionar tráilers en cada sede de distribución.
Características del Requerimiento	Cantidad de cajones
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Debe existir una sede de distribución previamente registrada.
Postcondición	

Identificador del Requerimiento RF-15	
Nombre del Requerimiento	Gestionar el estado de los cajones de estacionamiento
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Permitir administrar el estado de los cajones, definiendo si están ocupados o disponibles.
Características del Requerimiento	Actualización en tiempo real del estado.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Deben existir cajones previamente configurados.
Postcondición	

Identificador del Requerimiento RF-16	
Nombre del Requerimiento	Gestionar y supervisar el estado de muelles de carga de las sedes
eTipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Se adminsitra y asigna la disponibilidad de los muelles de carga para los choferes
Características del Requerimiento	Se asigna el estado de disponibilidad dentro de los puertos de carga (Disponible, Ocupado) Tiempos de carga y descarga Problemas o avisos
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Deben existir almacenes previamente registrados
Postcondición	

**Aplicación Móvil para Choferes**

Identificador del Requerimiento RF-17	
Nombre del Requerimiento	Mostrar los viajes asignados diariamente
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Permitir a los choferes visualizar una lista de los viajes asignados para el día actual, incluyendo origen, destino y horario.
Características del Requerimiento	Accesible desde la aplicación móvil. Debe mostrar detalles claros y organizados de cada viaje.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El chofer debe iniciar sesión en la aplicación móvil.
Postcondición	Se despliega una lista con los viajes asignados al chofer.

Identificador del Requerimiento RF-18	
Nombre del Requerimiento	Visualizar un mapa con la ruta que debe de tomar (MAPS API)
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Mostrar en la aplicación móvil un mapa interactivo que incluye la ruta desde el origen hasta el destino del viaje asignado, utilizando la API de mapas.
Características del Requerimiento	Actualización en tiempo real del progreso del viaje.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Debe haber un viaje asignado al chofer
Postcondición	

Identificador del Requerimiento RF-19	
Nombre del Requerimiento	Utilizar el GPS del móvil para mostrar la ubicación del chofer
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Usar el GPS del dispositivo móvil para rastrear y mostrar la ubicación actual del chofer en tiempo real dentro de la ruta asignada.
Características del Requerimiento	Compatible con servicios de geolocalización del móvil
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El chofer debe habilitar el GPS en su dispositivo y tener conexión internet.
Postcondición	

Identificador del Requerimiento RF-20	
Nombre del Requerimiento	Mostrar información del camión y remolque asignados
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Proveer detalles del camión y remolque asignados al chofer, como modelo, placa, directamente en la aplicación móvil.
Características del Requerimiento	Información descargada desde el sistema central
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El viaje debe estar asignado y registrado en el sistema
Postcondición	

Identificador del Requerimiento RF-21	
Nombre del Requerimiento	Permitir a los choferes ver su perfil y detalles de contacto
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Los choferes deben poder visualizar su perfil en la aplicación móvil, incluyendo información personal y detalles de contacto con la empresa.
Características del Requerimiento	

Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El chofer debe iniciar sesión en la aplicación móvil
Postcondición	

Identificador del Requerimiento	RF-22
Nombre del Requerimiento	Permitir a los choferes revisar el area de carga y descarga
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Los choferes deben poder visualizar dentro de su aplicacion en que puerto deben descargar y cargar
Características del Requerimiento	Dentro de su aplicacion movil de los choferes se visualiza en que lugar debe cargar la mercancia y en que puerto del destino debe descargar
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El chofer debe iniciar sesión en la aplicación móvil
Postcondición	

Identificador del Requerimiento	RF-23
Nombre del Requerimiento	Permitir a los choferes ver su perfil y detalles de contacto
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Los choferes deben poder visualizar su perfil en la aplicación móvil, incluyendo información personal y detalles de contacto con la empresa.
Características del Requerimiento	
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El chofer debe iniciar sesión en la aplicación móvil
Postcondición	

**Gestión Climática**

Identificador del Requerimiento RF-24	
Nombre del Requerimiento	Integrar una API de clima para mostrar condiciones meteorológicas en tiempo real
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Integrar un servicio de API de clima para proporcionar información meteorológica en tiempo real sobre las rutas y ubicaciones relevantes para los viajes programados y a programar.
Características del Requerimiento	Mostrar temperatura, precipitación, condiciones de viento, ...
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El sistema debe tener acceso a internet para conectarse a la API de clima
Postcondición	

Identificador del Requerimiento RF-25	
Nombre del Requerimiento	Mostrar alertas climáticas relevantes para los viajes programados
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Enviar notificaciones sobre alertas meteorológicas críticas, como tormentas, nevadas o lluvias intensas, que puedan afectar los viajes programados.
Características del Requerimiento	Alertas generadas automáticamente según la API de clima
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El sistema debe estar sincronizado con la API de clima
Postcondición	

**Administración General**

Identificador del Requerimiento RF-26	
Nombre del Requerimiento	Gestión de usuarios con roles
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Permitir la creación, edición, y eliminación de usuarios con roles específicos como administradores, operadores y choferes
Características del Requerimiento	Definir roles con permisos específicos.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	El usuario administrador debe estar autenticado
Postcondición	

Identificador del Requerimiento RF-27	
Nombre del Requerimiento	Control de accesos basado en roles
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Implementar un sistema de control de acceso para garantizar que los usuarios puedan realizar acciones solo según los permisos asociados a su rol
Características del Requerimiento	
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Los roles y permisos deben estar previamente configurados en el sistema
Postcondición	



## Seguimiento y Monitoreo

Identificador del Requerimiento RF-28	
Nombre del Requerimiento	Rastreo en tiempo real de los camiones en el mapa
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Permitir el monitoreo en tiempo real de la ubicación de los camiones mediante un mapa integrado.
Características del Requerimiento	
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional
Precondición	Los choferes deben tener la aplicación móvil activa y conexión a internet
Postcondición	

Identificador del Requerimiento RF-29	
Nombre del Requerimiento	Chat disponible para incidencias
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Integrar un chat de texto disponible para conductores como para los operadores del dispatch, para reportar incidencias o inconvenientes de los conductores en el camino.
Características del Requerimiento	<ul style="list-style-type: none"> <li>Después de resolver las incidencias los chats serán cerrados</li> </ul>
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional

Identificador del Requerimiento RF-30	
Nombre del Requerimiento	Crear reportes del dispatch
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	Integrar un interfaz con un formulario para crear los reportes de incidencia o inconveniente, estos reportes son creados por los operadores de dispatch.
Características del Requerimiento	<ul style="list-style-type: none"><li>• La interfaz para crear un reporte será un formulario</li><li>• Se tendrá una interfaz de reportes creados por el dispatch</li><li>• Los reportes tendrán un estado de la situación (Abierto, Pendiente, Cerrado)</li></ul>
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional

## REQUERIMIENTOS NO FUNCIONALES

Identificador del Requerimiento RNF-01	
Nombre del Requerimiento	Respuesta rápida en el sistema
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe responder de manera rápida a las acciones de los usuarios para que la experiencia sea fluida.
Características del Requerimiento	<ul style="list-style-type: none"><li>Las acciones principales, como consultar rutas o visualizar mapas, deben completarse en pocos segundos.</li><li>Se debe evitar que el sistema se congele o demore notablemente al interactuar con él.</li></ul>
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional

Identificador del Requerimiento RNF-02	
Nombre del Requerimiento	Seguridad en el acceso y manejo de datos
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe proteger la información de los usuarios y los datos sensibles, evitando accesos no autorizados.
Características del Requerimiento	<ul style="list-style-type: none"><li>Solo usuarios autorizados pueden acceder a áreas específicas del sistema.</li><li>La información importante se debe proteger durante el uso y el almacenamiento.</li></ul>
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional

Identificador del Requerimiento RNF-03	
Nombre del Requerimiento	Interfaz amigable y fácil de usar
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción

Descripción del requerimiento	El sistema debe contar con pantallas y menús claros para que todos los usuarios puedan utilizarlo sin dificultades.
Características del Requerimiento	<ul style="list-style-type: none"><li>• Diseños simples que faciliten la navegación y la comprensión de las funciones.</li><li>• Los usuarios podrán realizar sus tareas sin necesidad de una capacitación extensa.</li></ul>
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional

Identificador del Requerimiento	RNF-04
Nombre del Requerimiento	Fácil mantenimiento y actualizaciones
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento	El sistema debe ser fácil de actualizar y mantener, permitiendo corregir errores o agregar nuevas funciones sin complicaciones.
Características del Requerimiento	<ul style="list-style-type: none"><li>• El código y la organización del sistema deben facilitar su revisión y mejora.</li><li>• Las actualizaciones no deben afectar el funcionamiento normal del sistema.</li></ul>
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Baja/Media/Deseado Opcional

## PATRON DE DISEÑO

### Clean Architecture:

Es un patrón que separa la lógica de negocio de los detalles de infraestructura (bases de datos, controladores, frameworks, APIs externas). Esto facilita el mantenimiento y escalabilidad del sistema, permitiendo cambios en la infraestructura sin afectar el núcleo del negocio.

En un sistema 3PL como WareBox, donde hay múltiples interacciones con almacenes, rutas y transportistas, es fundamental que la lógica de negocio no dependa directamente de Laravel u otros frameworks. Así, si en el futuro se quiere cambiar la base de datos, añadir una capa de microservicios o usar otra tecnología, el impacto será mínimo.

### Container-Presenter:

Este patrón divide los componentes de React en dos categorías:

Componentes Contenedores (Container Components): Se encargan de manejar el estado, llamadas a la API y lógica de negocio.

Componentes de Presentación (Presentational Components): Son responsables únicamente de la interfaz gráfica y reciben datos mediante props.

Dado que WareBox tendrá una UI con múltiples módulos (gestión de rutas, almacenes, transportes), este patrón permite que los componentes sean reutilizables y que la lógica no esté acoplada a la UI, lo que mejora la mantenibilidad.

## ARQUITECTURAS

### MVC (Web)

La arquitectura MVC (Modelo-View-Controller) fue seleccionada para el desarrollo de la aplicación web la cual será desarrollada con Reactjs. El enfoque de la arquitectura MVC es separar la lógica de negocios de la visualización. Esta “separación de responsabilidades” proporciona una mejor división del trabajo y una mejora de mantenimiento.

### MVVM (Móvil)

Se utilizará la arquitectura MVVM (Model-View-ViewModel) para el desarrollo de la aplicación móvil con Jetpack Compose porque ofrece una separación clara entre la interfaz de usuario, la lógica de presentación y la lógica de negocio, lo que mejora la mantenibilidad, escalabilidad y modularidad del proyecto. Jetpack Compose, al ser declarativo y basado en estados, funciona perfectamente con MVVM, ya que el ViewModel puede gestionar el estado de la aplicación y exponerlo a través de flujos observables, como StateFlow o LiveData, que las composables pueden observar y renderizar automáticamente cuando hay cambios. Esto simplifica la construcción de una interfaz dinámica y reactiva, alineándose con el enfoque del proyecto que requiere características como seguimiento en tiempo real, notificaciones y personalización basada en datos.

## TECNOLOGIAS

- React Js Vite
- Jetpack Compose (Kotlin/Android)
- Laravel API Rest

## SERVICIOS EXTERNOS

- Maps API
- Weatherapi
- FirebaseMessaging

## Flujo de Trabajo y Gestión de Versiones para WareBox

### 1. Control de Versiones y Repositorios

El desarrollo del sistema WareBox se gestionará mediante **GitHub**, utilizando una organización denominada **WareBox**. Dentro de esta organización, se han creado los siguientes repositorios:

- **backend-api**: Contendrá el código del backend de la API.
- **frontend**: Contendrá el código del sistema web.
- **mobile-app**: Contendrá el código de la aplicación móvil para choferes y almacenistas.

#### 1.1 Estrategia de Ramas

Se utilizará una estrategia de ramas basada en **Git Flow**, con las siguientes ramas principales:

- **main**: Contiene la versión estable y desplegada en producción.
- **develop**: Rama donde se integran todas las nuevas características antes de ser desplegadas.
- **feature/**: Ramas para el desarrollo de nuevas funcionalidades.
- **hotfix/**: Para correcciones urgentes en producción.

Cada desarrollador trabajará en su propia rama **feature/** y, una vez completado el desarrollo, realizará un **pull request** para revisión y fusión en la rama **develop**.

## 2. Integración y Despliegue Continuo (CI/CD)

Se utilizará **GitHub Actions** para la automatización de pruebas y despliegues.

### 2.1 Backend (backend-api)

- Cuando se haga un **push** a la rama **main**, la API será desplegada automáticamente en la VPS.
- Se ejecutarán pruebas unitarias antes de permitir la fusión en la rama **main**.

### 2.2 Frontend (frontend)

- Cada vez que se haga un **push** a **main**, el código se desplegará automáticamente en la VPS.
- Se ejecutarán pruebas de integración antes de desplegar.

### 2.3 Aplicación Móvil (mobile-app)

- La versión se liberará en Google Play y/o TestFlight cuando se apruebe el código en la rama **main**.

## 3. Gestión de Tareas con Jira y Kanban

Se utilizará **Jira** para la gestión de tareas con la metodología **Kanban**.

### 3.1 Tablero de Tareas

El tablero de Jira estará dividido en las siguientes columnas:

- **To Do:** Tareas listas para ser tomadas.
- **In Progress:** Tareas en desarrollo.



- **Review:** Tareas en revisión de código.
- **Testing:** Tareas en fase de pruebas.
- **Done:** Tareas completadas y desplegadas.

Cada desarrollador tomará una tarea del **To Do**, la moverá a **In Progress**, y seguirá el flujo hasta **Done**.

## 4. Despliegue y Versionado

### 4.1 Ambientes de Desarrollo y Producción

- **Desarrollo:** Se utilizará la rama **develop** para mantener una versión funcional del sistema sin afectar producción.
- **Producción:** Se desplegará desde la rama **main** para garantizar estabilidad.

### 4.2 Versionado

El versionado seguirá el esquema **SemVer** (X.Y.Z):

- **X (Mayor):** Cambios disruptivos o incompatibles.
- **Y (Menor):** Nuevas funcionalidades sin romper compatibilidad.
- **Z (Parche):** Correcciones de errores o mejoras menores.

Cada nueva versión será documentada en un **CHANGELOG.md** en cada repositorio.

## 5. Conclusión

El flujo de trabajo está diseñado para asegurar estabilidad, colaboración eficiente y automatización en los despliegues. Con esta estructura, el proyecto WareBox puede garantizar un desarrollo ágil y confiable del sistema 3PL.

## Capturas de Desarrollo Móvil:

**Api Services:** Contiene los GET y POST que utiliza la app movil

```
interface ApiService {

    //AUTH
    @POST("loginEmployee")
    suspend fun loginEmployee(
        @Body request: LoginRequest
    ): LoginResponse

    @POST("logout")
    suspend fun logout(
        @Header("Authorization") token: String
    )

    //Routes testing
    @GET("route/{id}")
    suspend fun getRoute(
        @Header("Authorization") token: String,
        @Path("id") id: Int
    ): RouteResponse

    //Create a Box
    @POST("box-inventory")
    suspend fun createBox(
        @Header("Authorization") token: String,
        @Body request: CreateBoxRequest
    )
}
```

```
//Create a Pallet
@POST("pallet")
suspend fun createPallet(
    @Header("Authorization") token: String,
    @Body request: CreatePalletRequest
): PalletResponse

//get warehouse
@GET("warehouse/{id}")
suspend fun getWarehouse(
    @Header("Authorization") token: String,
    @Path("id") id: Int
): WarehouseResponse

//get company
@GET("company/{id}")
suspend fun getCompany(
    @Header("Authorization") token: String,
    @Path("id") id: Int
): CompanyResponse

//get product
@GET("product/{id}")
suspend fun getProduct(
    @Header("Authorization") token: String,
    @Path("id") id: Int
): ProductResponse
```

```
//get box
@GET("box-inventory/{id}")
suspend fun getBox(
    @Header("Authorization") token: String,
    @Path("id") id: Int
): BoxResponse

//get all companies
@GET("company")
suspend fun getAllCompanies(
    @Header("Authorization") token: String
): Companies

//get all warehouses
@GET("warehouse")
suspend fun getAllWarehouses(
    @Header("Authorization") token: String
): Warehouses

@GET("product/company/{company}")
suspend fun getProductsByCompany(
    @Header("Authorization") token: String,
    @Path("company") company: Int
): List<SimpleProductResponse>
}
```

## Repositories:

```
class WarehouseRepository @Inject constructor(private val apiService: ApiService) {  
    suspend fun createBox(token: String, qty: Int, weight: Float, volume: Float, pallet: Int, product: Int): Unit {  
        Dispatchers.IO {  
            try {  
                apiService.createBox(token: "Bearer $token", CreateBoxRequest(qty, weight, volume, pallet, product))  
            } catch (e: HttpException) {  
                println("HTTP Error: ${e.code()} - ${e.message()}")  
            }  
        }  
    }  
  
    suspend fun createPallet(  
        token: String,  
        company: Int,  
        warehouse: Int,  
        weight: Float,  
        volume: Float,  
        status: String,  
        verified: Boolean  
    ): PalletResponse = withContext(Dispatchers.IO) {  
        try {  
            apiService.createPallet(  
                token: "Bearer $token",  
                CreatePalletRequest(company, warehouse, weight, volume, status, verified)  
            )  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
}
```

```
class WarehouseRepository @Inject constructor(private val apiService: ApiService) {  
  
    suspend fun getWarehouse(token: String, id: Int): WarehouseResponse{  
        return try {  
            withContext(Dispatchers.IO) {  
                apiService.getWarehouse(token: "Bearer $token", id)  
            }  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
  
    suspend fun getCompany(token: String, id: Int): CompanyResponse {  
        return try{  
            withContext(Dispatchers.IO) {  
                apiService.getCompany(token: "Bearer $token", id)  
            }  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
  
    suspend fun getProduct(token: String, id: Int): ProductResponse{  
        return try {  
            withContext(Dispatchers.IO) {  
                apiService.getProduct(token: "Bearer $token", id)  
            }  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
        }  
    }  
}
```

```
class WarehouseRepository @Inject constructor(private val apiService: ApiService) {  
  
    suspend fun getPallet(token: String, id: Int): PalletResponse{  
        return try {  
            withContext(Dispatchers.IO) {  
                println("Getting pallet with id here: $id")  
                apiService.getPallet( token: "Bearer $token", id)  
            }  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
  
    suspend fun getBox(token: String, id: Int): BoxResponse {  
        return try{  
            withContext(Dispatchers.IO) {  
                apiService.getBox( token: "Bearer $token", id)  
            }  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
  
    suspend fun getAllCompanies(token: String): Companies {  
        return try{  
            withContext(Dispatchers.IO) {  
                apiService.getAllCompanies( token: "Bearer $token")  
            }  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
}
```

```
class WarehouseRepository @Inject constructor(private val apiService: ApiService) {  
    suspend fun getAllCompanies(token: String): Companies {  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
  
    suspend fun getAllWarehouses(token: String): Warehouses {  
        return try{  
            withContext(Dispatchers.IO) {  
                apiService.getAllWarehouses( token: "Bearer $token")  
            }  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
  
    suspend fun getProductsByCompany(token: String, company: Int): List<SimpleProductResponse> {  
        return try{  
            withContext(Dispatchers.IO) {  
                apiService.getProductsByCompany( token: "Bearer $token", company)  
            }  
        } catch (e: HttpException) {  
            println("HTTP Error: ${e.code()} - ${e.message()}")  
            throw e  
        }  
    }  
}
```

```
class AuthRepository @Inject constructor(private val apiService: ApiService)  
{  
    suspend fun loginEmployee(email: String, password: String): LoginResponse {  
        return withContext(Dispatchers.IO) {  
            apiService.loginEmployee(LoginRequest(email, password))  
        }  
    }  
  
    suspend fun logout(token: String) {  
        withContext(Dispatchers.IO) {  
            apiService.logout( token: "Bearer $token")  
        }  
    }  
}
```

**View Models Where API functions are being called**



```
fun getPallet(context: Context, id: Int) {  
    viewModelScope.launch {  
        try {  
            println("Getting pallet with id: $id")  
            _isSearching.value = true  
            val token = UserManager.getToken(context).toString()  
            val pallet = withContext(Dispatchers.IO) {  
                warehouseRepository.getPallet(token, id)  
            }  
            if (pallet == null) {  
                println("Pallet is null!")  
                return@launch  
            }  
            println("Pallet retrieved: $pallet")  
            _pallet.value = pallet  
        } catch (e: Exception) {  
            println("Error getting pallet: ${e.message}")  
        } finally {  
            _isSearching.value = false  
        }  
    }  
}
```

```
private fun createBox(
    context: Context,
    qty: Int,
    weight: Float,
    volume: Float,
    pallet: Int,
    product: Int
){
    viewModelScope.launch {
        try{
            -> val token = UserManager.getToken(context).toString()
            -> warehouseRepository.createBox(token, qty, weight, volume, pallet, product)
        } catch (e: Exception){
            println("Error creating box: ${e.message}")
        }
    }
}

fun getProduct(context: Context, id: Int){
    viewModelScope.launch {
        -> val token = UserManager.getToken(context).toString()
        try{
            -> warehouseRepository.getProduct(token, id)
        } catch (e: Exception){
            println("Error getting product: ${e.message}")
        }
    }
}
```

```
class CreatePalletViewModel @Inject constructor(private val warehouseRepository: WarehouseRepository) : View {  
    fun getBox(context: Context, id: Int){  
        viewModelScope.launch {  
            val token = UserManager.getToken(context).toString()  
            try{  
                warehouseRepository.getBox(token, id)  
            } catch (e: Exception){  
                println("Error getting box: ${e.message}")  
            }  
        }  
    }  
  
    fun getAllCompanies(context: Context) {  
        viewModelScope.launch {  
            val token = UserManager.getToken(context).toString()  
            try {  
                val result = warehouseRepository.getAllCompanies(token)  
                _companies.value = result.companies  
            } catch (e: Exception) {  
                println("Error getting all companies: ${e.message}")  
            }  
        }  
    }  
  
    fun getAllWarehouses(context: Context){  
        viewModelScope.launch {  
            val token = UserManager.getToken(context).toString()  
            try{  
                val result = warehouseRepository.getAllWarehouses(token)  
                _warehouses.value = result.warehouses  
                println("Warehouses: ${_warehouses.value}")  
            }  
        }  
    }  
}
```

```
class CreatePalletViewModel @Inject constructor(private val warehouseRepository: WarehouseRepository) : View {
    fun getProductsByCompany(context: Context, company: Int) {
        viewModelScope.launch {
            val token = UserManager.getToken(context).toString()
            try {
                _isSearching.value = true
                val result = warehouseRepository.getProductsByCompany(token, company)
                _products.value = result
                if (_products.value.isEmpty()){
                    showToast(context, message: "No products found for this company")
                }
                println("Products: ${_products.value}")
                _isSearching.value = false
            } catch (e: Exception) {
                println("Error getting products by company: ${e.message}")
            }
        }
    }

    fun createPalletAndBoxes(
        context: Context,
        company: Int,
        warehouse: Int,
        weight: Float,
        volume: Float,
        verified: Boolean,
        boxes: List<CreateBoxRequest>,
    ) {
        viewModelScope.launch {
            try {
                createPalletAndBoxes(context, company, warehouse, weight, volume, verified, boxes)
            } catch (e: Exception) {
                println("Error creating pallet and boxes: ${e.message}")
            }
        }
    }
}
```

```
class CreatePalletViewModel @Inject constructor(private val warehouseRepository: WarehouseRepository) : View {  
    fun createPalletAndBoxes(  
        context: Context,  
        company: Int,  
        warehouse: Int,  
        weight: Float,  
        volume: Float,  
        verified: Boolean,  
        boxes: List<CreateBoxRequest>,  
    ) {  
        viewModelScope.launch {  
            try {  
                setCreating(true)  
                val token = UserManager.getToken(context).toString()  
                val status = "Created"  
  
                // Create the pallet and retrieve its ID  
                val palletResponse = warehouseRepository.createPallet(  
                    token,  
                    company,  
                    warehouse,  
                    weight,  
                    volume,  
                    status,  
                    verified  
                )  
                val palletId = palletResponse.id  
                _palletId.intValue = palletId  
  
                for (box in boxes) {  
                    createBox(  

```

```
class CreatePalletViewModel @Inject constructor(private val warehouseRepository: WarehouseRepository) : View {
    fun createPalletAndBoxes() {
        viewModelScope.launch {
            _palletId.invariant = palletId

            for (box in boxes) {
                createBox(
                    context,
                    box.qty,
                    box.weight,
                    box.volume,
                    palletId, // the pallet ID
                    box.product
                )
            }
            _success.value = true
        } catch (e: Exception) {
            println("Error creating pallet and boxes: ${e.message}")
            _success.value = false
            showToast(context, message: "Error creating pallet and boxes: ${e.message}")
        } finally {
            setCreating(false)
        }
    }

    fun resetSuccess() {
        _success.value = false
    }
}
```

```
@HiltViewModel
class ProfileViewModel @Inject constructor(private val authRepository: AuthRepository) : ViewModel() {
    private val _roleName = mutableStateOf<String>("")
    val roleName: State<String> = _roleName

    fun logout(context: Context, navController: NavController) {
        viewModelScope.launch {
            try {
                val token = UserManager.getToken(context).toString()
                authRepository.logout(token)
                UserManager.clearUser(context)
                navController.navigate(route: "login") {
                    popUpTo(route: "home") { inclusive = true } // Remove all previous screens from stack
                }
            } catch (e: Exception) {
                println("Logout failed: ${e.message}")
            }
        }
    }
}
```

```
@HiltViewModel
class LoginViewModel @Inject constructor( private val authRepository: AuthRepository) : ViewModel() {
    private val _isLoggedIn = mutableStateOf( value: false)
    val isLoggedIn: State<Boolean> = _isLoggedIn

    fun loginEmployee(context: Context, email: String, password: String){
        viewModelScope.launch {
            try{
                val response = authRepository.loginEmployee(email, password)
                //save our token securely
                UserManager.saveUser(context, response.user, response.token)
                _isLoggedIn.value = true
                println("Login successful: ${response.user.first_name} , ${response.token}")
            }catch (e: Exception){
                println("Login failed: ${ e.message }")
                _isLoggedIn.value = false
            }
        }
    }
}
```

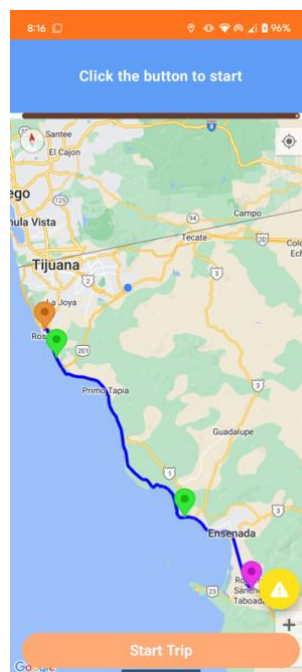
Log in SS where the Login Endpoint (POST) is used:



## Profile GET

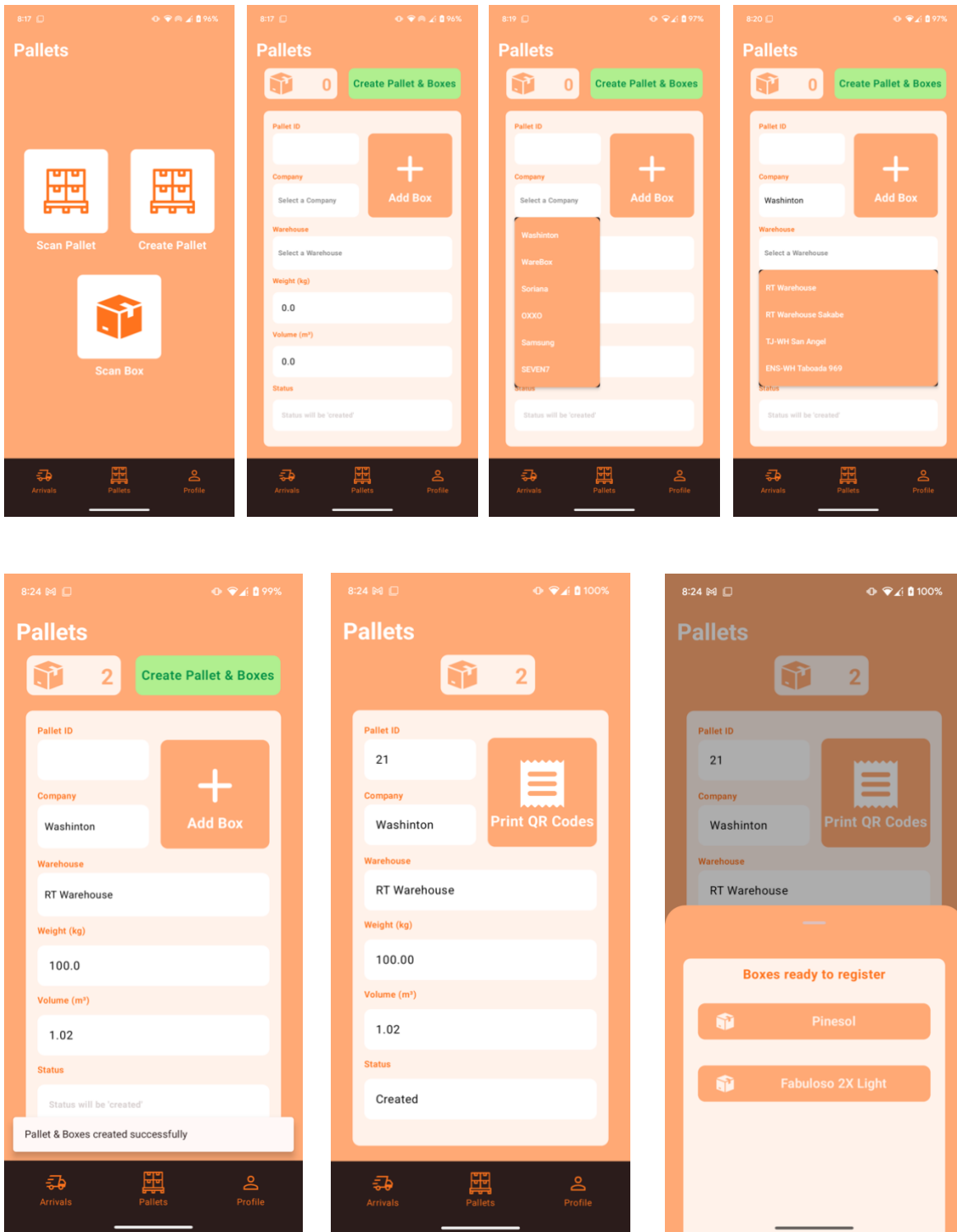


## Route GET





## Pallet &amp; Boxes Post, Company GET, Warehouse GET



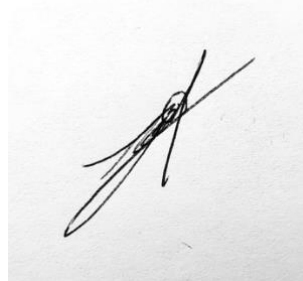


## Get of products based on the company

The screenshots illustrate the process of selecting and registering products. The first screen shows a search for products by company. The second screen shows the details for a selected product, 'Pinesol', including its weight, height, width, length, and depth. The third screen shows the 'Boxes ready to register' section, where the user can add or remove boxes and hide information.

## FIRMAS

Integrantes	
Nombre	Firma
Howard Isaí García Arreola	
Canales Bernal Carlos Natividad	
Derian Omar Tarango Mendez	

Lilian Denisse de la Torre Lopez	
Paredes Nevarez Alexis Omar	
Jose Hector Zamorano Palma	

Maestros	
Nombre	Firma
Ray Brunett Parra Galaviz	
Daniel Enrique Torres Aldana	

--	--