



PLANTILLA DE ESPECIFICACIÓN Y VALIDACIÓN DE REQUERIMIENTOS

INTEGRANTES:

Breton Rendon Gael

Perez Bello Vanory Esperanza

Pliego Cortes Giovanni

Chavez Madrigal Daniel

Lopez Gonzalez Eduardo

ASIGNATURA:

Desarrollo Móvil Integral

GRUPO:

10 B

PARCIAL:

Primero

FECHA DE CREACIÓN:

16/10/2024

ESTÁNDAR IEEE-830

ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

Proyecto: DELI-BIRD



FICHA DEL DOCUMENTO

Fecha	Revisión	Autor	Verificó

Insertar una fila por cada revisión, en la tabla anterior. La columna Revisión, se refiere al número de revisión.

VALIDACIONES DEL DOCUMENTO

Documento validado por las partes en fecha:

Por el cliente	Por la empresa suministradora

ÍNDICE DE CONTENIDO

1 INTRODUCCIÓN	5
PROPÓSITO	6
1.1.1 Propósito del documento	6
1.1.2 Audiencia a la que va dirigido	7
1.1 ALCANCE	7
1.2 PERSONAL INVOLUCRADO	7
1.3 DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS	10
1.5 REFERENCIAS	12
1.6 RESUMEN	12
2 DESCRIPCIÓN GENERAL	13
2.1 PERSPECTIVA DEL PRODUCTO	13
2.2 FUNCIONALIDAD DEL PRODUCTO	14
2.3 CARACTERÍSTICAS DE LOS USUARIOS	14
2.4 RESTRICCIONES	16
2.5 SUPOSICIONES Y DEPENDENCIAS	17
3 REQUISITOS ESPECÍFICOS	19
3.1 REQUISITOS COMUNES DE LAS INTERFACES	34
3.1.1 INTERFACES DE USUARIO	35
3.1.2 INTERFACES DE HARDWARE	37
3.1.3 INTERFACES DE SOFTWARE	38
3.1.4 INTERFACES DE COMUNICACIÓN	40
3.2. REQUISITOS NO FUNCIONALES	41
3.2.1 REQUISITOS DE RENDIMIENTO	43
3.2.2 REQUISITOS DE SEGURIDAD	44
3.3.3 REQUISITOS DE FIABILIDAD	44
3.3.4 REQUISITOS DE DISPONIBILIDAD	45
3.3.5 REQUISITOS DE MANTENIBILIDAD	46
3.3.6 PORTABILIDAD	48
3.4 OTROS REQUISITOS	49

1 INTRODUCCIÓN

En el entorno aeroportuario, la gestión eficiente de los cargamentos es un factor clave para garantizar la rapidez y seguridad en el manejo de mercancías. La necesidad de herramientas tecnológicas que optimicen estos procesos es evidente, especialmente en un contexto donde la digitalización y la automatización juegan un papel fundamental en la logística moderna.

*Nuestra aplicación, **DELI-BIRD**, surge como una solución integral diseñada para facilitar el trabajo de administradores y operadores de montacargas en aeropuertos. Su objetivo principal es mejorar la gestión de los cargamentos a través de herramientas tecnológicas avanzadas, permitiendo un control más preciso sobre el registro, la ubicación, el monitoreo y la planificación logística de los envíos y recepciones de mercancías.*

La aplicación incorpora funcionalidades clave como el escaneo de códigos QR para el registro automático de cargamentos, la integración con Google Maps para el seguimiento en tiempo real, el uso de datos meteorológicos para la planificación de rutas seguras y un sistema de seguridad basado en reconocimiento facial para el acceso a la plataforma.

*Al centralizar estas operaciones en una sola herramienta, **DELI-BIRD** optimiza los tiempos de carga y descarga, minimiza errores humanos y proporciona información en tiempo real para la toma de decisiones.*

PROPÓSITO

1.1.1 Propósito del documento

Este documento tiene como propósito describir los objetivos, funcionalidades y alcances del sistema DELI-BIRD, una aplicación diseñada para optimizar la gestión de cargamentos en aeropuertos. Se detallan los requisitos, características técnicas y operativas que permitirán mejorar la eficiencia en el

registro, ubicación, monitoreo y planificación logística de los cargamentos, facilitando la labor de los administradores y empleados de montacargas.

1.1.2 Audiencia a la que va dirigido

Este documento está dirigido a los siguientes grupos de interés:

- Administradores de aeropuertos y terminales de carga, quienes supervisan la logística de los cargamentos y necesitan herramientas para su eficiente gestión.
- Desarrolladores y equipo de TI, responsables del diseño, implementación y mantenimiento del sistema.
- Inversores y tomadores de decisiones, interesados en evaluar la viabilidad y beneficios del sistema para su adopción en aeropuertos.


1.1 ALCANCE


El sistema **DELI-BIRD** permitirá la gestión eficiente de cargamentos en aeropuertos mediante:

- **Funciones principales:** Registro, asignación de ubicación, rastreo en tiempo real y control de acceso con reconocimiento facial.
- **Limitaciones:** No se integrará con sistemas externos de aerolíneas ni incluirá pagos o facturación
- **Objetivo:** Optimizar la logística aeroportuaria, reduciendo tiempos de manejo y mejorando la trazabilidad de cargamentos.

1.2 PERSONAL INVOLUCRADO

Nombre	Pliego Cortez Giovanni
Rol	Analista de Requerimientos, Desarrollador, Implementación del sistema y pruebas funcionales

Categoría profesional	Líder de Proyecto, Ingeniero de Software
Responsabilidades	Supervisión del desarrollo y cumplimiento de objetivos
Información de contacto	0321101279@ut-tijuana.edu.mx
Aprobación	

Nombre	Chavez Madrigal Daniel
Rol	Analista de Requerimientos, Desarrollador, Implementación del sistema y pruebas funcionales
Categoría profesional	Ingeniero de Software
Responsabilidades	Recopilación y documentación de requisitos, Implementación del sistema y pruebas funcionales
Información de contacto	0321101285@ut-tijuana.edu.mx
Aprobación	

Nombre	Perez Bello Vanory Esperanza
Rol	Analista de Requerimientos, Desarrollador, Implementación del sistema y pruebas funcionales
Categoría profesional	Ingeniero de Software
Responsabilidades	Recopilación y documentación de requisitos, Implementación del sistema y pruebas funcionales

Información de contacto	0321101704@ut-tijuana.edu.mx
Aprobación	✓

Nombre	Breton Rendon Gael
Rol	Desarrollador, diseñador
Categoría profesional	Ingeniero de Software
Responsabilidades	Implementación del sistema y pruebas funcionales
Información de contacto	0321101247@ut-tijuana.edu.mx
Aprobación	✓

Nombre	Lopez Gonzales Eduardo
Rol	Analista de Requerimientos, Desarrollador
Categoría profesional	Ingeniero de Software
Responsabilidades	Implementación del sistema y pruebas funcionales
Información de contacto	0319125005@ut-tijuana.edu.mx
Aprobación	✓

1.3 DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

Definiciones

<i>Nombre</i>	<i>Descripción</i>
<i>Código QR</i>	<i>Imagen de matriz de puntos que almacena información codificada y permite identificar cargamentos de manera única en el sistema.</i>
<i>Cargamento</i>	<i>Paquete o conjunto de paquetes registrados en el sistema para ser transportados a su destino.</i>
<i>Orden de Envío</i>	<i>Solicitud formal de un cliente para el traslado de un cargamento de un origen a un destino específico.</i>
<i>Cliente</i>	<i>Usuario del sistema que tiene la capacidad de registrar órdenes de envío y escanear códigos QR para vincular cargamentos.</i>
<i>Personal de Carga</i>	<i>Usuario encargado de la manipulación y registro físico de los cargamentos en almacenes y aviones.</i>
<i>Administrador</i>	<i>Usuario con acceso completo al sistema, encargado de la gestión de usuarios, almacenes, aviones y rutas de vuelo.</i>
<i>Almacén</i>	<i>Instalación física donde se resguardan los cargamentos antes de ser enviados a su destino.</i>

<i>Avión</i>	<i>Medio de transporte aéreo utilizado para trasladar los cargamentos de un almacén de origen a otro de destino.</i>
<i>Ruta de Vuelo</i>	<i>Trayectoria que sigue un avión para el transporte de cargamentos, incluyendo información climática relevante.</i>
<i>Estado de Cargamento</i>	<i>Fase en la que se encuentra un cargamento dentro del proceso logístico (Ej. Registrado, Enviado, Entregado, Defectuoso).</i>
<i>Notificación</i>	<i>Mensaje generado por el sistema para alertar a los usuarios sobre cambios en sus órdenes o situaciones relevantes.</i>

Acrónimos y Abreviaturas

<i>Nombre</i>	<i>Descripción</i>
<i>API</i>	<i>Application Programming Interface (Interfaz de Programación de Aplicaciones).</i>
<i>QR</i>	<i>Quick Response (Código de Respuesta Rápida).</i>
<i>PWA</i>	<i>Progressive Web App (Aplicación Web Progresiva).</i>
<i>DB</i>	<i>Database (Base de Datos).</i>
<i>ID</i>	<i>Identifier (Identificador Único).</i>
<i>UI</i>	<i>User Interface (Interfaz de Usuario).</i>

UX	<i>User Experience (Experiencia de Usuario).</i>
AES	<i>Advanced Encryption Standard (Estándar de Cifrado Avanzado).</i>
JWT	<i>JSON Web Token (Token Seguro de Autenticación).</i>

1.5 REFERENCIAS

Referencia	Título	Fecha	Autor u organización

Relación completa de todos los documentos relacionados en la especificación de requisitos de software, identificando de cada documento el título, referencia (si procede), fecha y organización que lo proporciona.

1.6 RESUMEN

Este documento está estructurado en varias secciones para proporcionar una visión clara del sistema a desarrollar.

- **Sección 1: Introducción**

Presenta el propósito, alcance y audiencia del documento, así como el personal involucrado.

- **Sección 2: Descripción General**

Proporciona un panorama del sistema, sus funciones principales y restricciones.

- **Sección 3: Requerimientos Específicos**

Detalla los requerimientos funcionales y no funcionales del sistema.

- **Sección 4: Modelado del Sistema**

Incluye diagramas y explicaciones sobre la arquitectura y componentes del sistema.

- **Sección 5: Referencias**

Lista fuentes de información utilizadas para la elaboración del documento.

- **Sección 6: Arquitecturas**

2 DESCRIPCIÓN GENERAL

Nuestra aplicación es un portal diseñado para facilitar a los administradores y empleados de montacargas de aeropuertos la gestión eficiente de los cargamentos que recibirán y enviarán las paqueterías. Proporciona herramientas integradas para el registro, ubicación, monitoreo y planificación logística de los cargamentos, utilizando tecnologías modernas como reconocimiento facial, códigos QR, y mapas.

2.1 PERSPECTIVA DEL PRODUCTO

El sistema de control de cargamentos del aeropuerto centraliza las operaciones logísticas dentro de un ecosistema móvil. Nuestra aplicación aborda las siguientes áreas:

- Usuarios principales: Administradores y operadores de montacargas del aeropuerto.
- Entorno de operación: La aplicación será utilizada en dispositivos móviles dentro de las instalaciones aeroportuarias, con funcionalidades en línea para garantizar la conectividad y actualización de la información.

Integraciones clave:

- Google Maps para ubicación y seguimiento de contenedores.
- Datos meteorológicos para planificación de rutas de vuelo.
- Reconocimiento facial para inicio de sesión seguro.

2.2 FUNCIONALIDAD DEL PRODUCTO

A) Gestión de la mercancía:

Registro de cargamentos mediante la lectura de códigos QR.

Generación automática de reportes de entrega y recibidos basados en los registros QR.

Recepción y registro de mercancía proveniente de almacenes internos del aeropuerto.

Lista de cargamento:

Mantenimiento de una lista actualizada de mercancías programadas para llegar.

B) Monitoreo y ubicación:

Seguimiento de la ubicación de contenedores dentro del aeropuerto mediante Google Maps.

C) Planificación de rutas:

Acceso a datos climáticos en tiempo real para facilitar la planificación de rutas de vuelo seguras y eficientes.

D) Gestión de almacenes:

Organización y almacenamiento de mercancías antes de su envío.

E) Seguridad y accesibilidad:

Inicio de sesión mediante reconocimiento facial para garantizar el acceso seguro al sistema.

2.3 CARACTERISTICAS DE LOS USUARIOS

Rol	Formación	Habilidades	Actividades
Administrador	Formación en logística, administración o sistemas.	<ul style="list-style-type: none"> - Gestión de usuarios - Manejo de reportes - Administración de rutas y aviones - Supervisión de operaciones 	<ul style="list-style-type: none"> - Administrar usuarios y permisos en la plataforma. - Generar reportes de operación. - Supervisar el estado de almacenes y cargamentos. - Configurar y gestionar rutas de vuelo.
Personal de Carga	Formación en logística o transporte.	<ul style="list-style-type: none"> - Manejo de sistemas de registro y seguimiento - Escaneo de códigos QR - Control de inventario - Organización de cargamentos 	<ul style="list-style-type: none"> - Registrar cargamentos en el sistema. - Asignar cargamentos a almacenes y aviones. - Verificar la llegada y salida de mercancías. - Reportar incidencias y estado de los cargamentos.
Cliente	No requiere formación específica.	<ul style="list-style-type: none"> - Uso básico de plataformas digitales - Registro y seguimiento de envíos 	<ul style="list-style-type: none"> - Registrar órdenes de envío en la plataforma. - Consultar el estado de sus paquetes. - Recibir notificaciones sobre su pedido. - Gestionar

		- Comunicación con el servicio de atención al cliente para reclamaciones o cancelaciones si es necesario.
--	--	---

2.4 RESTRICCIONES

Restricción	Descripción
Lenguaje de Programación	El backend se desarrollará en Laravel (PHP), mientras que el frontend para dispositivos móviles utilizará React Native y para la web React.js.
Base de Datos	Se empleará PostgreSQL como gestor de base de datos debido a su capacidad de manejar transacciones complejas y grandes volúmenes de datos.
Autenticación y Seguridad	Se implementará autenticación mediante JWT (JSON Web Tokens) y se cifrarán las contraseñas utilizando AES o bcrypt para garantizar la seguridad de los datos.
Hardware Requerido	Los dispositivos de los usuarios deberán contar con una cámara funcional para el escaneo de códigos QR y con suficiente capacidad de almacenamiento para ejecutar la PWA o la aplicación móvil sin problemas.
Conectividad	El sistema requiere conexión a Internet para registrar y actualizar el estado de los cargamentos en tiempo real.

	Sin embargo, algunas funciones limitadas estarán disponibles en modo offline para sincronización posterior.
Integración con APIs Externas	Se integrará con Google Maps para la localización de almacenes y rutas, y con AccuWeather para obtener datos meteorológicos en tiempo real para la planificación de vuelos.
Escalabilidad	El sistema debe ser escalable para soportar el crecimiento de usuarios y la cantidad de transacciones sin afectar el rendimiento.

2.5 SUPOSICIONES Y DEPENDENCIAS

Suposición/Dependencia	Descripción
Disponibilidad de Internet	Se asume que los usuarios tendrán acceso a una conexión a Internet estable para la actualización de datos en tiempo real. En caso de fallos de conexión, algunas funcionalidades podrían verse limitadas.
Compatibilidad con Dispositivos	Se espera que los dispositivos móviles utilizados por el personal cuenten con cámaras funcionales para el escaneo de códigos QR y con hardware suficiente para ejecutar la aplicación sin problemas.

Infraestructura de Servidores	El sistema depende de servidores en la nube para el almacenamiento y procesamiento de datos. Un fallo en la infraestructura del proveedor de hosting podría afectar la disponibilidad del sistema.
APIs de Terceros	Se asume que servicios externos como Google Maps (para localización) y AccuWeather (para datos meteorológicos) estarán disponibles. Si alguna API cambia sus condiciones o deja de estar disponible, se requerirán modificaciones en el sistema.
Soporte de PostgreSQL	Se da por hecho que PostgreSQL seguirá siendo el gestor de base de datos principal. Si se requiere migración a otro sistema de base de datos, se necesitarán modificaciones en la arquitectura del sistema.
Disponibilidad de Personal	Se considera que el equipo de desarrollo, soporte y operaciones estará disponible para realizar mantenimiento y mejoras continuas en el sistema.
Uso de Laravel y React Native	El desarrollo del sistema se basa en Laravel para el backend y React Native para la aplicación móvil. Si se requiere cambiar de framework o tecnología, el desarrollo podría verse afectado.

3 REQUISITOS ESPECÍFICOS

A) Registro e inicio de sesión de usuarios:

El sistema contará con una funcionalidad que permita a los usuarios registrarse en la plataforma utilizando sus datos personales o mediante reconocimiento facial. Esto incluye la creación de perfiles únicos para los administradores y el personal de carga. Además, se podrán iniciar sesión utilizando correo y contraseña o el reconocimiento facial previamente configurado.

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-A01
Nombre del Requerimiento	Registro e inicio de sesión de usuarios
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema permitirá a los usuarios registrarse e iniciar sesión mediante correo y contraseña, así como a través de reconocimiento facial para mayor comodidad y seguridad.
Características del Requerimiento	Registro de usuario: Campos obligatorios: Nombre completo. Correo electrónico único y válido. Contraseña que cumpla con las siguientes validaciones: Al menos 10 caracteres. Incluir una letra mayúscula, un número y un símbolo especial. Configuración de reconocimiento facial: Escaneo inicial del rostro del usuario durante el registro para vincular su perfil.

	Inicio de sesión: Opciones: Mediante correo y contraseña. Mediante reconocimiento facial si el usuario lo activó durante el registro. Validaciones: Si el reconocimiento facial falla, el sistema solicitará autenticación mediante correo y contraseña. Seguridad: Los datos de reconocimiento facial serán cifrados y almacenados localmente o en un servidor seguro. Las contraseñas serán cifradas con un algoritmo de hash seguro.		
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/Opcional

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-A02
Nombre del Requerimiento	Recuperación de contraseña
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema permitirá a los usuarios recuperar su contraseña en caso de olvido mediante un enlace enviado a su correo electrónico registrado.
Características del Requerimiento	<ul style="list-style-type: none"> - El usuario ingresará su correo electrónico. - El sistema enviará un enlace seguro para restablecer la contraseña. - El enlace expirará después de 24 horas.

Prioridad del requisito	<input type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

B) Plataforma de escritorio (Administrador):

La aplicación de escritorio estará destinada exclusivamente a los administradores del sistema. Este módulo permitirá gestionar órdenes, rutas de vuelo, cargamentos, y tomar decisiones informadas basadas en datos climáticos.

Identificador del Requerimiento	RF-B01
Nombre del Requerimiento	Gestión de órdenes de envío
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema permitirá al administrador registrar nuevas órdenes de envío y gestionar las existentes.
Características del Requerimiento	<p>Opciones: Crear, editar, eliminar y consultar órdenes de envío.</p> <p>Campos para registrar una orden:</p> <p>ID de la orden (generado automáticamente).</p> <p>Detalles del cargamento (descripción, peso, dimensiones).</p> <p>Ruta asignada.</p>

	Fecha de carga. Fecha estimada de entrega. Destinatario final (nombre, contacto).
Prioridad del requisito	<input type="checkbox"/> Alta/Eencial <input type="checkbox"/> <input type="checkbox"/> Baja/ Media/De Opcional seado

Telefono

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-B02
Nombre del Requerimiento	Registro de orden de envío
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema permitirá al administrador registrar una nueva orden de envío. La orden incluirá todos los cargamentos relacionados y sus detalles, como ruta, almacén, y fechas clave.
Características del Requerimiento	Campos obligatorios: Código QR del cargamento: Escaneado para verificar que el cargamento está completo antes de registrarlo. Ruta: Ruta del vuelo asignada. Fecha de salida: Fecha en la que el cargamento será subido al avión. Destinatario: Persona o empresa que recibirá el cargamento. Fecha estimada de entrega: Proyección de cuándo llegará al destino.

	<p>Almacén de origen: Identificación del almacén donde está almacenado el cargamento.</p> <p>Estado del cargamento: Indica si el cargamento está completo o incompleto.</p> <p>Validaciones:</p> <p>Todos los campos obligatorios deben completarse antes de registrar la orden.</p> <p>El sistema verificará que todos los códigos QR del cargamento se encuentren registrados en la base de datos.</p>
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

WEB

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-B03
Nombre del Requerimiento	Gestión de rutas de vuelo
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema permitirá al administrador gestionar las rutas de vuelo, incluyendo la asignación de cargamentos y la consulta de datos climáticos.
Características del Requerimiento	<ul style="list-style-type: none"> - Crear, editar, eliminar y consultar rutas de vuelo. - Asignar cargamentos a rutas específicas.

	- Consultar datos climáticos en tiempo real para la planificación de rutas.
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

YA NO EXISTE

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-B04
Nombre del Requerimiento	Generación de reportes
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema permitirá al administrador generar reportes sobre órdenes de envío, cargamentos y rutas de vuelo.
Características del Requerimiento	<ul style="list-style-type: none"> - Reportes de órdenes completadas, pendientes y en proceso. - Reportes de cargamentos por almacén. - Exportación de reportes en formatos PDF y Excel.
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

C) Plataforma móvil (Personal de carga):

La aplicación móvil estará destinada al personal de carga, permitiendo realizar actividades relacionadas con el manejo físico de los cargamentos, incluyendo escaneo de QR y acceso rápido a información del cargamento.

ESO NO ESTA EN LA API

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-C01
Nombre del Requerimiento	Iniciar sesión con reconocimiento facial
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	La aplicación móvil permitirá a los usuarios iniciar sesión mediante reconocimiento facial para garantizar mayor seguridad y eficiencia.

Características del Requerimiento	<ul style="list-style-type: none"> ● Validaciones: <ul style="list-style-type: none"> ○ Verificar que el rostro coincida con el usuario registrado. ○ Opción alternativa de inicio de sesión con usuario y contraseña. ● Privacidad: <ul style="list-style-type: none"> ○ Los datos faciales serán almacenados de manera segura.
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-C02
Nombre del Requerimiento	Escaneo y verificación de cargamentos
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El personal de carga (Transportista) deberá escanear los códigos QR de cada cargamento para verificar su estado e integrarlo en la orden correspondiente.
Características del Requerimiento	<p>Escaneo QR:</p> <ul style="list-style-type: none"> • Cada código QR escaneado mostrará detalles del cargamento, como el avión asignado, fecha de subida, y ubicación en el almacén. • Si un código QR no coincide con la orden registrada, el sistema generará una alerta. <p>Generación de reporte:</p> <ul style="list-style-type: none"> • El personal de carga podrá generar un reporte de envío tras confirmar que todos los códigos QR están completos. • El reporte incluirá información sobre los cargamentos escaneados y cualquier discrepancia encontrada.
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-C03
Nombre del Requerimiento	Consulta de información de cargamentos
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El personal de carga podrá consultar información detallada de los cargamentos escaneados, incluyendo su ubicación en el almacén y estado actual.
Características del Requerimiento	<ul style="list-style-type: none"> - Mostrar detalles como peso, dimensiones, destinatario y ruta asignada. - Integración con Google Maps para mostrar la ubicación exacta del cargamento en el almacén.
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-001
Nombre del Requerimiento	Registro de Usuario
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del Requerimiento	El usuario podrá registrarse en la aplicación proporcionando su nombre, correo electrónico, contraseña y teléfono.
Características del Requerimiento	- Validación de datos ingresados antes del registro.- Confirmación de cuenta a través de correo electrónico.- Opción para aceptar términos y condiciones antes de registrarse.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

```

const handleSubmit = async () => {
  if (validateForm()) {
    setLoading(true);

    try {
      const userData = {
        name: formData.name,
        email: formData.email,
        password: formData.password,
        rol: 'cliente',
        paqueteriald: 1
      };

      const response = await axios.post(
        'http://127.0.0.1:8000/api/users/',
        userData
      );

      console.log('Registration successful:', response.data);

      setLoading(false);
      setSuccessVisible(true);

      setTimeout(() => {
        setSuccessVisible(false);
        navigation.navigate('Login');
      }, 2000);

    } catch (error) {
      console.error('Registration error:', error);
      setLoading(false);

      if (error.response && error.response.data) {
        if (error.response.data.email) {
          setErrors({
            ...errors,
            email: 'Este correo electrónico ya está registrado'
          });
        } else {
          Alert.alert(
            'Error',
            'No se pudo completar el registro. Por favor intenta nuevamente.',
            [{ text: 'OK' }]
          );
        }
      } else {
        Alert.alert(
          'Error',
          'Ocurrió un error en la conexión. Por favor verifica tu internet e intenta nuevamente.',
          [{ text: 'OK' }]
        );
      }
    }
  }
};

```

Tengo una función llamada `handleSubmit`, qué es **asíncrona** porque dentro hago una petición a una API.

1. **Primero, valido el formulario** usando `validateForm()`. Si los datos no son válidos, la función se detiene.
2. **Activo el estado de carga** con `setLoading(true)`, para que el usuario vea que el proceso está en marcha.
3. **Creo un objeto `userData`**, donde almaceno los datos del usuario que quiero registrar:
 - Nombre (`name`)
 - Correo electrónico (`email`)
 - Contraseña (`password`)
 - Rol (`rol`), que siempre será `'cliente'`
 - `paqueteriaId`, que en este caso es `1`
4. **Hago una petición `POST` a la API** con `axios.post()`, enviando `userData` al endpoint `'http://127.0.0.1:8000/api/users/'`.
 - Si el registro es exitoso, imprimo un mensaje en la consola:
`'Registration successful'` con los datos de la respuesta.
 - Quito el estado de carga (`setLoading(false)`).
 - Muestro un mensaje de éxito (`setSuccessVisible(true)`).
 - Después de 2 segundos (`setTimeout()`), oculto el mensaje y navego a la pantalla de **Login** (`navigation.navigate('Login')`).
5. **Si ocurre un error en la petición**, manejo los posibles problemas:
 - **Si el correo ya está registrado**, actualizo el estado de `errors.email` para mostrar un mensaje en la UI.
 - **Si es otro error**, muestro una alerta informando que el registro no se pudo completar.
 - **Si la API no responde (problema de conexión)**, muestro una alerta indicando que puede ser un problema de internet.
 - En cualquier caso, quito el estado de carga con `setLoading(false)`.

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-002
Nombre del Requerimiento	Inicio de Sesión
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del Requerimiento	El usuario podrá iniciar sesión utilizando su correo electrónico y contraseña.
Características del Requerimiento	- Validación de credenciales antes de permitir el acceso.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional


```

const handleLogin = async () => {

  console.log('Iniciando login...');
  if (!email || !password) {
    setError('Por favor ingresa el correo y la contraseña.');
```

return;

}

setLoading(true);

setError("");

try {

const endpoint = `\${config.API_URL}/login`;

const response = await fetch(endpoint, {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ email, password }),

});

console.log('Respuesta recibida:', response.status);

const data = await response.json();

console.log('Datos de respuesta:', data);

if (response.ok) {

await AsyncStorage.setItem('userToken', data.access_token);

await AsyncStorage.setItem('userRol', data.user.rol);

await AsyncStorage.setItem('userId', String(data.user.id));

await AsyncStorage.setItem('userName', data.user.name);

setIsAuthenticated(true);

setUserRol(data.user.rol);

setUserId(data.user.id);

Setusername(data.user.name)

} else {

setError(data.message || 'Credenciales inválidas');

}

} catch (error) {

setError('Hubo un problema al intentar iniciar sesión. ' + error);

console.error('Error en login:', error);

} finally {

setLoading(false);

console.log('Finalizando login...');

}

};

Tengo una función llamada `handleLogin`, que es asíncrona porque hace una petición a la API para iniciar sesión.

1. Imprimo en consola que el proceso de login ha comenzado con `console.log('Iniciando login...');`
2. Valido que el usuario haya ingresado correo y contraseña:
 - Si alguno está vacío, muestro un error con `setError('Por favor ingresa el correo y la contraseña.')` y detengo la ejecución con `return`.
3. Inicio el estado de carga y limpio errores previos:
 - `setLoading(true)`: Indico que la solicitud está en proceso.
 - `setError('')`: Borro cualquier mensaje de error anterior.
4. Hago la petición a la API enviando los datos del usuario:
 - Construyo el endpoint usando `config.API_URL`.
 - Hago una petición **POST** con `fetch()`, enviando el correo y la contraseña en formato JSON.
 - Especifico en los headers que el contenido es de tipo `application/json`.
5. Recibo la respuesta y la proceso:
 - Imprimo en consola el estado de la respuesta (`response.status`).
 - Convierto la respuesta en JSON y la muestro en consola (`console.log('Datos de respuesta:', data);`).
6. Si el login es exitoso (`response.ok`):
 - Guardo los datos en `AsyncStorage` para que el usuario siga autenticado en la app:

- `userToken`: El token de acceso.
- `userRol`: El rol del usuario.
- `userId`: El ID del usuario.
- `userName`: El nombre del usuario.
- Actualizo los estados globales para reflejar la sesión activa:
 - `setIsAuthenticated(true)`: Indica que el usuario ha iniciado sesión.
 - `setUserRol(data.user.rol)`: Guarda el rol del usuario.
 - `setUserId(data.user.id)`: Guarda el ID del usuario.
 - `Setusername(data.user.name)`: Guarda el nombre del usuario.

7. Si el login falla (`response.ok === false`):

- Muestro un mensaje de error al usuario.
- Si la API envió un mensaje de error (`data.message`), lo uso.
- Si no, muestro un mensaje por defecto: "`Credenciales inválidas`".

8. Si ocurre un error inesperado (por ejemplo, problemas con la red o la API):

- Muestro un mensaje de error que incluya el problema (`setError('Hubo un problema al intentar iniciar sesión. ' + error);`).
- Registro el error en la consola (`console.error('Error en login:', error);`).

9. Finalizo la ejecución:

- `setLoading(false)`: Quito el estado de carga, sin importar si el login fue exitoso o no.
- Imprimo "`Finalizando login...`" en la consola.

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-003
Nombre del Requerimiento	Registro de Pedido
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del Requerimiento	El usuario podrá registrar un nuevo pedido proporcionando los detalles necesarios.
Características del Requerimiento	- Ingreso de datos como destino, descripción del paquete y tipo de envío
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

```

const handleSubmit = async () => {
  if (validateStep2()) {
    setIsLoading(true);

    try {
      const dataToSubmit = {
        ...formData,
        peso: parseFloat(formData.peso),
        largo: parseFloat(formData.largo),
        ancho: parseFloat(formData.ancho),
        alto: parseFloat(formData.alto)
      };

      const response = await axios.post(
        config.API_URL+'/ordenes/cliente',
        dataToSubmit
      );

      console.log('Order created successfully:', response.data);

      setIsLoading(false);
      setShowSuccess(true);

      setTimeout(() => {
        setShowSuccess(false);
        navigation.navigate('Inicio');
      }, 2000);

    } catch (error) {
      console.error('Error creating order:', error);
      setIsLoading(false);

      Alert.alert(
        'Error',
        'No se pudo crear el pedido. Por favor intenta nuevamente.',
        [{ text: 'OK' }]
      );
    }
  }
};

```

Tengo una función llamada `handleSubmit`, que es asíncrona porque envía una orden a la API y espera la respuesta.

1. Primero, valido que los datos sean correctos usando `validateStep2()`.
 - Si la validación falla, la función se detiene.
2. Activo el estado de carga con `setIsLoading(true)`, para indicar que el proceso está en marcha.
3. Preparo los datos antes de enviarlos:
 - Copio todos los valores de `formData`.
 - Aseguro que los campos `peso`, `largo`, `ancho` y `alto` sean números convirtiéndolos con `parseFloat()`.
4. Hago la petición `POST` a la API usando `axios.post()`, enviando `dataToSubmit` al endpoint `config.API_URL+'/ordenes/cliente'`.
 - Si la orden se crea con éxito, imprimo en consola: `'Order created successfully:'` junto con los datos recibidos.
5. Quito el estado de carga y muestro un mensaje de éxito:
 - `setIsLoading(false)`: Indica que el proceso ha terminado.
 - `setShowSuccess(true)`: Muestra un mensaje de confirmación.
 - Después de 2 segundos (`setTimeout()`), oculto el mensaje y navego a la pantalla "Inicio" (`navigation.navigate('Inicio')`).
6. Si ocurre un error en la petición, lo manejo adecuadamente:
 - Imprimo el error en consola (`console.error('Error creating order:', error)`).
 - Quito el estado de carga (`setIsLoading(false)`).
 - Muestro una alerta indicando que la orden no pudo ser creada y sugiero intentarlo nuevamente.

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-004
Nombre del Requerimiento	Consulta de Pedidos
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del Requerimiento	El usuario podrá ver una lista con los pedidos registrados.
Características del Requerimiento	- Listado de pedidos con detalles relevantes como fecha, estado y destino.- Posibilidad de filtrar pedidos por estado.- Acceso a detalles individuales de cada pedido.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

```

const fetchOrders = async () => {
  try {
    setLoading(true);

    const id = await AsyncStorage.getItem('userId');
    const token = await AsyncStorage.getItem('userToken');

    if (!token) {
      Alert.alert('Error', 'No se encontró token de autenticación');
      setLoading(false);
      return;
    }

    const response = await axios.get(`${config.API_URL}/ordenes/usuario/${id}`, {
      headers: {
        'Authorization': `Bearer ${token}`,
        'Content-Type': 'application/json',
      }
    });

    setOrders(response.data);
    filterOrdersByStatus(response.data, selectedTab);
    setLoading(false);
  } catch (error) {
    console.error('Error fetching orders:', error);
    Alert.alert(
      'Error',
      'No se pudieron cargar los pedidos. Por favor intenta nuevamente.'
    );
    setLoading(false);
  }
};

```


Tengo una función llamada `fetchOrders`, que es asíncrona porque obtiene las órdenes del usuario desde la API.

1. Activo el estado de carga con `setLoading(true)`, para indicar que los datos se están obteniendo.

2. Recupero el `userId` y el `userToken` desde `AsyncStorage`.

- El `userId` me ayuda a obtener las órdenes específicas del usuario.
- El `userToken` es necesario para autenticar la petición.

3. Verifico si hay un token disponible.

- Si no hay un token, muestro una alerta (`Alert.alert()`) diciendo que no se encontró autenticación.
- Quito el estado de carga (`setLoading(false)`) y detengo la ejecución con `return`.

4. Hago la petición `GET` a la API usando `axios.get()`.

- Construyo la URL con el `userId`:
`${config.API_URL}/ordenes/usuario/${id}`.
- Envío los headers de autenticación con el `token` en formato `Bearer`.

5. Si la petición es exitosa:

- Guardo las órdenes en el estado con `setOrders(response.data)`.
- Aplico un filtro por estado usando `filterOrdersByStatus()`, basado en la pestaña activa (`selectedTab`).
- Quito el estado de carga con `setLoading(false)`.

6. Si ocurre un error en la petición:

- Lo imprimo en consola (`console.error('Error fetching orders:', error)`).
- Muestro una alerta al usuario indicando que los pedidos no pudieron cargarse.
- Quito el estado de carga (`setLoading(false)`).

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-005
Nombre del Requerimiento	Estado de los Pedidos
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del Requerimiento	El usuario podrá visualizar el estado actualizado de cada pedido.
Características del Requerimiento	- Estados del pedido: En proceso, En camino, Entregado.- Notificaciones automáticas sobre cambios de estado.- Opción de rastreo del pedido si está disponible.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

```

const fetchOrders = async () => {
  try {
    setLoading(true);

    const id = await AsyncStorage.getItem('userId');
    const token = await AsyncStorage.getItem('userToken');

    if (!token) {
      Alert.alert('Error', 'No se encontró token de autenticación');
      setLoading(false);
      return;
    }

    const response = await axios.get(`${config.API_URL}/ordenes/usuario/${id}`, {
      headers: {
        'Authorization': `Bearer ${token}`,
        'Content-Type': 'application/json',
      }
    });

    setOrders(response.data);
    filterOrdersByStatus(response.data, selectedTab);
    setLoading(false);
  } catch (error) {
    console.error('Error fetching orders:', error);
    Alert.alert(
      'Error',
      'No se pudieron cargar los pedidos. Por favor intenta nuevamente.'
    );
    setLoading(false);
  }
};

```

Tengo una función llamada `fetchOrders`, que es asíncrona porque obtiene las órdenes del usuario desde la API y espera la respuesta.

1. Activo el estado de carga con `setLoading(true)`, para indicar que la solicitud está en proceso.

2. Obtengo los datos del usuario desde `AsyncStorage`:

- `userId`: Identifica al usuario para traer sus órdenes.
- `userToken`: Se usa para autenticar la petición.

3. Verifico si hay un token disponible:

- Si el token no existe, muestro una alerta (`Alert.alert()`) con un mensaje de error.
- Quito el estado de carga (`setLoading(false)`) y detengo la ejecución con `return`.

4. Hago la petición `GET` a la API usando `axios.get()`, enviando:

- La URL con el `userId`:
`${config.API_URL}/ordenes/usuario/${id}`.
- Los headers de autenticación, incluyendo el `token`.

5. Si la petición es exitosa:

- Guardo las órdenes en el estado con `setOrders(response.data)`.
- Aplico un filtro por estado con `filterOrdersByStatus()`, basado en la pestaña activa (`selectedTab`).
- Quito el estado de carga con `setLoading(false)`.

6. Si ocurre un error en la petición:

- Imprimo el error en consola con `console.error()`.

- Muestro una alerta indicando que las órdenes no pudieron cargarse.
- Quito el estado de carga con `setLoading(false)`.

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-006
Nombre del Requerimiento	Detalles de Envío
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del Requerimiento	El usuario podrá visualizar la información detallada de los envíos asignados.
Características del Requerimiento	- Visualización de la ruta estimada y tiempo de entrega previsto
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

```

const filterOrdersByStatus = (ordersList, tab) => {
  const filtered = ordersList.filter(order => {
    const currentStatus = getCurrentStatus(order.historial_ordenes);

    if (tab === 'pending') {
      if (!['completada', 'cancelada'].includes(currentStatus)) {
        if (searchQuery) {
          return (
            order.identificador.toLowerCase().includes(searchQuery.toLowerCase()) ||

            order.destinatario.nombre_completo.toLowerCase().includes(searchQuery.toLowerCase())
          );
        }
        return true;
      }
      return false;
    } else if (tab === 'completed') {
      if (['completada', 'cancelada'].includes(currentStatus)) {
        if (searchQuery) {
          return (
            order.identificador.toLowerCase().includes(searchQuery.toLowerCase()) ||

            order.destinatario.nombre_completo.toLowerCase().includes(searchQuery.toLowerCase())
          );
        }
        return true;
      }
      return false;
    }
    return false;
  });

  const sortedFiltered = filtered.sort((a, b) =>
    new Date(b.fecha_creacion) - new Date(a.fecha_creacion)
  );

  setFilteredOrders(sortedFiltered);
};

```

Tengo una función llamada `filterOrdersByStatus`, que filtra y ordena las órdenes según su estado y el criterio de búsqueda.

Paso a paso:

1. Recibo dos parámetros:

- `ordersList`: Lista de órdenes a filtrar.
- `tab`: Indica si se mostrarán órdenes pendientes o completadas.

2. Uso `filter()` para filtrar las órdenes según su estado:

- Primero, obtengo el estado actual de cada orden con `getCurrentStatus(order.historial_ordenes)`.
- Si la pestaña activa es `'pending'`:
 - Solo incluyo órdenes que no estén completadas ni canceladas.
- Si la pestaña activa es `'completed'`:
 - Solo incluyo órdenes que sí estén completadas o canceladas.

3. Aplico la búsqueda si hay un `searchQuery` activo:

- Verifico si el `identificador` de la orden o el nombre del destinatario incluye el texto ingresado en la búsqueda, ignorando mayúsculas y minúsculas.
- Si hay coincidencias, la orden se mantiene en la lista filtrada.

4. Ordeno las órdenes filtradas por fecha de creación en orden descendente (`más recientes primero`), usando `sort()`.

5. Actualizo el estado de `setFilteredOrders` con la lista filtrada y ordenada.

WEB

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-D01
Nombre del Requerimiento	Visualización de gráficos dinámicos
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema debe permitir la visualización de gráficos dinámicos en el dashboard para representar distintos conjuntos de datos de manera clara e interactiva.
Características del Requerimiento	<p>Tipos de gráficos soportados:</p> <ul style="list-style-type: none"> - Gráficos de barras. - Gráficos de líneas. <p>Otros gráficos según necesidades del usuario.</p> <p>Interacción con los gráficos:</p> <ul style="list-style-type: none"> - Los gráficos deben permitir acercamiento y desplazamiento en los datos. <p>Visualización adaptable:</p> <p>Los gráficos deben ajustarse automáticamente según el tamaño de la pantalla.</p>
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

WEB

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-D02
Nombre del Requerimiento	Aplicación de filtros interactivos en gráficos
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema permitirá a los usuarios aplicar filtros interactivos para modificar los datos mostrados en los gráficos y obtener análisis personalizados.
Características del Requerimiento	<p>Opciones de filtrado:</p> <ul style="list-style-type: none"> - Filtrar por rango de fechas. - Filtrar por categoría de datos. - Filtrar por ubicación o región. <p>Interfaz de usuario:</p> <ul style="list-style-type: none"> - Filtros accesibles mediante botones o menús desplegables. <p>Persistencia de filtros:</p> <p>Los filtros aplicados se mantendrán hasta que el usuario los restablezca.</p>
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-D03
Nombre del Requerimiento	Control de acceso por roles
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema debe gestionar los permisos de acceso al dashboard en función del rol del usuario, asegurando que cada usuario solo pueda visualizar y operar sobre los datos y secciones permitidas según su nivel de autorización.
Características del Requerimiento	<p>Roles del sistema:</p> <ul style="list-style-type: none"> ● Administrador: Acceso total a todos los datos y configuraciones. ● Personal de carga: Acceso a información operativa, sin privilegios de administración. ● Cliente: Acceso limitado a su información y estado de envíos. <p>Control de acceso:</p> <ul style="list-style-type: none"> ● Restricción de vistas y datos según el rol del usuario. ● Deshabilitación de opciones no permitidas para ciertos usuarios.
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

TODO

REQUERIMIENTOS FUNCIONALES

Identificador del Requerimiento	RF-D04
Nombre del Requerimiento	Visualización de datos históricos
Tipo	<input type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Descripción del requerimiento:	El sistema debe permitir a los usuarios visualizar y analizar datos históricos para evaluar tendencias y comportamientos a lo largo del tiempo.
Características del Requerimiento	<p>Opciones de visualización:</p> <ul style="list-style-type: none"> - Selección de períodos específicos (diario, semanal, mensual, anual). <p>Interactividad:</p> <ul style="list-style-type: none"> - Posibilidad de aplicar filtros a los datos históricos.
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/Opcional

3. Requisitos específicos

Comprobación de validez de las entradas

- Validación de datos ingresados por los empleados (número de carga, tipo de mercancía, peso, dimensiones).
- Restricción de acceso a funcionalidades según el rol del usuario (administrador, operador de montacargas).

- *Uso de escaneo de códigos QR para minimizar errores en el registro de datos.*

Secuencia exacta de operaciones

1. *Registro de nuevo cargamento con validación de datos.*
2. *Asignación de ubicación en almacén mediante mapeo en tiempo real.*
3. *Generación de código QR para rastreo.*
4. *Monitoreo del estado del cargamento (pendiente, en tránsito, entregado).*
5. *Notificación automática en caso de retrasos o cambios de ubicación.*

Respuesta a situaciones anormales

- *Si se detecta un error en el código QR, se notificará al operador para corrección manual.*
- *En caso de pérdida de conexión, el sistema almacenará datos localmente y los sincronizará una vez restablecida la red.*

Parámetros

- *Peso y dimensiones máximas por unidad de carga.*
- *Accesos restringidos por permisos de usuario.*

Generación de salidas

- *Reportes de movimientos de carga en tiempo real.*
- *Alertas y notificaciones para seguimiento de carga.*

Relaciones entre entradas y salidas

- ***Entrada:*** Registro de nuevo cargamento → ***Salida:*** Código QR generado.
- ***Entrada:*** Escaneo de código QR → ***Salida:*** Ubicación y estado actual del cargamento.
- ***Entrada:*** Escaneo de código QR de recibido → ***Salida:*** Notificación de cargamento recibido al punto destino.

Requisitos lógicos para la base de datos

- *integración de un módulo para el registro de la información de cada cargamento (ID, tipo, peso, dimensiones, ubicación, estado).*
- *seguimiento al transporte de carga (ubicación, tiempo estimado de llegada)*
- *Integración con un módulo de usuarios para el status de llegada de la carga*

3.1 REQUISITOS COMUNES DE LAS INTERFACES

El sistema DELI-BIRD debe cumplir con los siguientes requisitos comunes de las interfaces para garantizar una experiencia de usuario consistente, accesible y eficiente:

1. Consistencia Visual y Funcional

- **Diseño uniforme:** Todas las pantallas del sistema deben seguir un diseño consistente en cuanto a colores, tipografías e íconos, basado en la paleta de colores y estilo visual de la marca DELI-BIRD.
- **Componentes reutilizables:** Los elementos de la interfaz (botones, menús, formularios, etc.) deben ser reutilizables y alineados con el diseño general para evitar inconsistencias.

2. Usabilidad

- **Navegación sencilla:** La estructura de la interfaz debe ser clara y permitir al usuario realizar tareas en un máximo de 3 clics desde cualquier punto del sistema.
- **Textos intuitivos:** Los botones y etiquetas deben usar términos claros, evitando jerga técnica innecesaria.

3. Accesibilidad

- **Cumplimiento de WCAG:** Las interfaces deben cumplir con los estándares de accesibilidad Web Content Accessibility Guidelines (WCAG) en nivel AA, garantizando acceso para personas con discapacidades.

- **Modo oscuro y alto contraste:** Para facilitar el uso en ambientes de poca iluminación y mejorar la accesibilidad visual.
- **Teclado y lector de pantalla:** Compatibilidad con navegación mediante teclado y lectores de pantalla como NVDA o VoiceOver.

4. Rendimiento

- **Carga rápida:** Las interfaces deben cargar completamente en un tiempo inferior a 3 segundos bajo una conexión estándar (4G o equivalente).
- **Bajo consumo de recursos:** La aplicación debe estar optimizada para funcionar sin problemas en dispositivos con recursos limitados.

5. Compatibilidad

- **Multidispositivo:** La interfaz debe ser completamente responsiva y adaptarse a diferentes tamaños de pantalla (smartphones, tablets y computadoras de escritorio).
- **Soporte multilenguaje:** Todos los textos deben ser traducibles para soportar español e inglés.

6. Seguridad

- **Protección de datos:** Los formularios de ingreso de información deben validar los datos de entrada para evitar inyecciones SQL o ataques XSS.
- **Sesiones seguras:** Las interfaces deben manejar sesiones con expiración automática y mecanismos para evitar el acceso no autorizado.

3.1.1 INTERFACES DE USUARIO

El sistema DELI-BIRD contará con una interfaz intuitiva y optimizada para dispositivos móviles y web, diseñada para facilitar la gestión de cargamentos en aeropuertos.

Diseño General

- **Estilo Visual:** La interfaz tendrá un diseño moderno y minimalista, siguiendo los principios de Material Design para garantizar accesibilidad y facilidad de uso.
- **Colores:** Se utilizará una combinación de tonos verdes y grises, alineados con la paleta de colores de la marca DELI-BIRD. Estos colores incluyen:
 - Verde oscuro para botones principales y elementos destacados.
 - Tonos más claros de verde para fondos y secciones secundarias.
 - Blanco y gris claro como colores neutros para un diseño limpio y profesional.
- **Tipografía:** Se empleará una fuente clara y legible, como Roboto o Open Sans, asegurando buena visibilidad en distintos dispositivos.
- **Íconos:** Se integrarán íconos de fácil reconocimiento para mejorar la navegación y reducir la carga cognitiva del usuario.

Pantallas Clave

1. **Pantalla de Inicio de Sesión:**
 - Opción de autenticación con correo y contraseña o reconocimiento facial.
 - Botón para recuperación de contraseña.
2. **Panel Principal (Dashboard):**
 - Vista general de cargamentos en proceso, recientes y pendientes.
 - Acceso rápido a reportes y notificaciones en tiempo real.
3. **Gestión de Cargamentos:**
 - Escaneo de códigos QR para registrar o verificar cargamentos.
 - Listado de cargamentos con filtros por fecha, estado y ubicación.
 - Información detallada de cada cargamento: peso, dimensiones, destino, estado y responsable.
4. **Gestión de Rutas de Vuelo:**
 - Mapa interactivo con integración de Google Maps para visualizar rutas de vuelo.
 - Datos meteorológicos en tiempo real para planificación logística.
5. **Reportes y Estadísticas:**

- Generación y exportación de reportes en PDF y Excel.
- Gráficos dinámicos sobre tiempos de entrega, eficiencia en manejo de cargamentos y rutas más utilizadas.

6. Notificaciones:

- Alertas en tiempo real sobre cambios en rutas, problemas en cargamentos o condiciones climáticas adversas.

Accesibilidad y Usabilidad

- Compatibilidad con dispositivos móviles (Android) y web (PWA).
- Interfaz optimizada para uso en entornos de trabajo con poca iluminación y alta actividad.

3.1.2 INTERFACES DE HARDWARE

El sistema DELI-BIRD debe integrarse con diversos componentes de hardware esenciales para garantizar el correcto funcionamiento en las operaciones de gestión de cargamentos. A continuación, se detallan las características lógicas de cada interfaz y los requerimientos de configuración:

1. Escáner de Códigos QR y Barras

- **Compatibilidad:** Soporte para escáneres QR y de códigos de barras mediante conexión USB o Bluetooth.
- **Configuración:** El sistema debe reconocer automáticamente el dispositivo conectado sin necesidad de instalación manual de controladores adicionales.
- **Velocidad de lectura:** Reconocimiento de códigos en menos de 1 segundo, con capacidad para leer formatos comunes (QR, EAN-13, Code 128, etc.).
- **Integración:** El sistema debe procesar los datos escaneados directamente en los formularios correspondientes, validando el formato y mostrando errores en tiempo real si el código es inválido.

2. Impresoras de Etiquetas

- **Compatibilidad:** Soporte para impresoras de etiquetas térmicas mediante conexión USB, Bluetooth o red local (Wi-Fi/Ethernet).
- **Configuración:** Posibilidad de configurar tamaños de etiqueta, fuentes y alineación desde el sistema.
- **Salida directa:** Generación automática de etiquetas a partir de datos de cargamentos registrados, incluyendo códigos QR y texto personalizado.

3. Dispositivos Móviles y Tablets

- **Compatibilidad:** Soporte para dispositivos móviles Android, con conexión al sistema a través de aplicaciones nativas o PWA (Progressive Web Apps).
- **Configuración:** Capacidad para sincronizar datos en tiempo real con el servidor mediante redes Wi-Fi o 4G/5G.
- **Sensores:** Uso de cámaras para escaneo de códigos QR y captura de imágenes de cargamentos.

5. Sistemas de Monitoreo y Localización

- **GPS Integrado:** Integración con dispositivos GPS para rastreo de cargamentos y visualización de rutas en tiempo real.
- **Configuración:** Permitir la sincronización de dispositivos GPS externos mediante Bluetooth o Wi-Fi Direct.
- **Precisión:** Garantizar una actualización de ubicación en intervalos de no más de 10 segundos.

3.1.3 INTERFACES DE SOFTWARE

El sistema **DELI-BIRD** requiere la integración de distintos componentes de software para garantizar la comunicación entre la aplicación web, la aplicación móvil y servicios externos.

1. API del sistema

- **Descripción:** API desarrollada en **Django** que gestiona la comunicación entre la aplicación web y la aplicación móvil.
- **Propósito:** Facilitar el intercambio de información entre administradores y carguistas, incluyendo checklists, tareas, paquetes y viajes.
- **Definición del interfaz:**
 - **Formato de datos:** JSON.
 - **Autenticación:** Basada en tokens (JWT o similar).

2. Base de datos central

- **Descripción:** Base de datos utilizada para almacenar información relevante del sistema, como usuarios, paquetes, checklists, tareas y viajes.
- **Propósito:** Garantizar la persistencia y sincronización de datos en tiempo real entre la aplicación móvil y la aplicación web.
- **Definición del interfaz:**
 - **Estructura:** Tablas organizadas para manejar la información de usuarios, paquetes, checklists y viajes.
 - **Tecnología:** Base de datos relacional compatible con **Django ORM**.

3. Servicios de autenticación

- **Descripción:** Sistema de autenticación basado en **Django** para gestionar el acceso seguro a la plataforma.
- **Propósito:** Controlar los permisos de administradores y carguistas dentro de sus respectivas aplicaciones.
- **Definición del interfaz:**
 - **Formato de autenticación:** Uso de tokens para sesiones seguras.

4. Servicios externos

- **AWS (Reconocimiento Facial)**

- **Descripción:** *Uso de los servicios de reconocimiento facial de AWS para la autenticación de usuarios.*
- **Propósito:** *Permitir el inicio de sesión mediante reconocimiento facial para mejorar la seguridad y agilizar el acceso.*
- **Definición del interfaz:**
 - **Formato de datos:** *JSON.*
 - **Integración:** *Envío de imágenes capturadas por la aplicación móvil a AWS Rekognition para verificación de identidad.*
- **Servicio de mapas**
 - **Descripción:** *Integración con un servicio de mapas para la visualización de rutas y ubicaciones de carga.*
 - **Propósito:** *Permitir a los carguistas ubicar los puntos de carga y destino dentro del aeropuerto.*
 - **Definición del interfaz:**
 - **Formato de datos:** *JSON / GeoJSON.*
 - **Integración:** *Solicitud de rutas y visualización en la aplicación móvil.*

3.1.4 INTERFACES DE COMUNICACIÓN

El sistema **DELI-BIRD** requiere comunicación entre sus diferentes componentes (aplicación web, aplicación móvil y backend) y con servicios externos. A continuación, se detallan los protocolos y mecanismos de comunicación utilizados:

1. Comunicación entre la aplicación web, móvil y la API (Backend en Django)

- Protocolo de comunicación: HTTP/HTTPS.
- Formato de datos: JSON.
- Mecanismo de comunicación:

- La aplicación móvil y la aplicación web interactúan con la API mediante solicitudes RESTful.
- Todas las transacciones se realizan a través de HTTPS para garantizar la seguridad.
- Uso de JWT (JSON Web Tokens) para autenticación y autorización.

2. Comunicación con la base de datos

- Protocolo de comunicación: PostgreSQL o MySQL a través de conexiones seguras.
- Mecanismo de comunicación:
 - El backend en Django usa Django ORM para gestionar la base de datos.
 - Conexión cifrada para garantizar la protección de los datos.

3. Comunicación con AWS (Reconocimiento Facial y Mapas)

- Protocolo de comunicación: HTTPS con AWS SDK o REST API.
- Mecanismo de comunicación:
 - Reconocimiento facial: La aplicación móvil captura imágenes y las envía a AWS Rekognition para validación de identidad.
 - Servicio de mapas: La aplicación móvil obtiene rutas y ubicaciones en tiempo real desde AWS o el servicio de mapas seleccionado.

4. Seguridad y cifrado

- Todas las comunicaciones se realizan mediante HTTPS para proteger los datos en tránsito.
- Autenticación basada en tokens JWT para evitar accesos no autorizados.
- Cifrado de datos sensibles antes de ser almacenados en la base de datos.

3.2. REQUISITOS NO FUNCIONALES

4.1 Rendimiento

- La aplicación móvil debe cargar y registrar paquetes en **menos de 3 segundos** por operación en condiciones de red óptimas.
- La aplicación web debe procesar y mostrar listas de checklists y tareas en **menos de 2 segundos**.
- El backend debe manejar **hasta 500 solicitudes concurrentes** sin degradación significativa del rendimiento.

4.2 Seguridad

- La autenticación de usuarios debe realizarse mediante **tokens JWT** y contar con **dobles factor de autenticación (2FA)** opcional.
- La información almacenada en la base de datos debe estar **cifrada con AES-256** para datos sensibles.
- Toda comunicación entre cliente y servidor debe realizarse a través de **HTTPS con TLS 1.2 o superior**.
- Implementación de **políticas de acceso basadas en roles (RBAC)** para restringir funciones según el usuario.

4.3 Usabilidad

- La aplicación móvil debe ser intuitiva y permitir a los carguistas completar un registro en **menos de 5 toques**.
- La aplicación web debe ser compatible con los principales navegadores (Chrome, Firefox, Edge, Safari).
- La interfaz debe ser accesible para personas con discapacidad, cumpliendo con **WCAG 2.1 AA**.

4.4 Disponibilidad y Confiabilidad

- El sistema debe garantizar una **disponibilidad del 99.5%** mensual.
- Se deben realizar copias de seguridad automáticas **cada 24 horas** en AWS.
- El sistema debe ser capaz de operar en **modo offline** en la aplicación móvil y sincronizar datos cuando se recupere la conexión.

3.2.1 REQUISITOS DE RENDIMIENTO

Tiempo de respuesta:

- *La aplicación debe procesar solicitudes de registro, actualización y consulta de cargamentos en un tiempo máximo de **3 segundos**.*
- *La validación de códigos QR para el escaneo de cargamentos debe realizarse en **menos de 2 segundos**.*

Escalabilidad:

- *El sistema debe soportar al menos **500 empleados de montacargas activos simultáneamente**, sin degradar el rendimiento.*
- *La aplicación debe poder manejar un incremento del **30% en la cantidad de registros diarios de cargamentos** sin afectar la velocidad de respuesta.*

Utilización de recursos:

- *El consumo de batería de dispositivos móviles no debe superar el **5% por cada hora de uso continuo**.*
- *La aplicación debe funcionar con **una conexión de datos mínima de 3G** y ajustarse a condiciones de baja conectividad.*

Disponibilidad:

- *El sistema debe estar disponible al **99.9% del tiempo**, permitiendo solo un máximo de 5 horas de inactividad anual debido a mantenimiento.*

3.2.2 REQUISITOS DE SEGURIDAD

Autenticación y autorización:

- El acceso a la aplicación debe requerir **credenciales de usuario únicas**, autenticación con reconocimiento facial y doble factor de autenticación (MFA) para administradores.
- Los empleados de montacargas solo pueden ver y registrar los cargamentos asignados a su turno.

Protección de datos:

- Los datos de los cargamentos, incluyendo origen, destino y detalles del transportista, deben **cifrarse con AES-256 antes** de ser almacenados.
- La transmisión de datos debe realizarse mediante **protocolos seguros (HTTPS y TLS 1.3)** para evitar interceptaciones.

Prevención de ataques:

- El sistema debe contar con **medidas contra ataques de inyección SQL**, evitando accesos no autorizados a la base de datos.
- Se deben implementar **filtros de seguridad** para evitar ataques como Cross-Site Scripting (XSS) en la interfaz web.

Respaldo y recuperación:

- Los datos de los cargamentos deben **respaldarse cada 12 horas**, asegurando la recuperación de información en caso de fallos.
- En caso de una caída del sistema, la restauración del servicio debe realizarse en **menos de 30 minutos**

3.3.3 REQUISITOS DE FIABILIDAD

Tasa de fallos:

- La aplicación debe mantener una tasa de fallos menor al **0.1% en un año de operación**.

Mantenimiento y actualizaciones:

- Las actualizaciones deben realizarse de manera **automatizada y fuera del horario pico** para minimizar interrupciones.
- Se debe permitir a los administradores programar ventanas de mantenimiento sin afectar la operatividad diaria.

Tolerancia a fallos:

- Si un servidor falla, el sistema debe **redireccionar automáticamente** la conexión a un servidor de respaldo sin interrupciones.
- En caso de fallo en la conexión de internet, la aplicación debe permitir la **operación en modo offline**, sincronizando los datos cuando la conexión se restablezca.

Pruebas de fiabilidad:

- La aplicación debe ser sometida a **pruebas de carga** simulando **al menos 1000 transacciones concurrentes** para garantizar su estabilidad.
- Se deben realizar pruebas de **recuperación ante fallos** cada 6 meses para validar los mecanismos de respaldo y restauración.

3.3.4 REQUISITOS DE DISPONIBILIDAD

El sistema debe garantizar un nivel adecuado de disponibilidad para asegurar su operatividad continua y minimizar interrupciones en el servicio. A continuación, se detallan los requisitos de disponibilidad exigidos:

1. Disponibilidad en Horarios Críticos:

- Durante los horarios de operación crítica (08:00 - 22:00 UTC), la disponibilidad del sistema debe ser de al menos **99.9%**, lo que reduce el

tiempo máximo de inactividad en estos períodos a aproximadamente **43 minutos al mes**.

2. Mantenimiento Programado:

- Las ventanas de mantenimiento programado deben realizarse preferentemente en horarios de menor impacto (ej. de 00:00 a 04:00 UTC).
- El mantenimiento no debe exceder **4 horas por mes** y deberá ser notificado con al menos **48 horas de anticipación** a los usuarios.

3. Recuperación ante Fallos:

- En caso de fallos críticos, el sistema deberá restaurar su operatividad en un tiempo máximo de **1 hora** en condiciones normales.
- Para fallos mayores que requieran intervención manual, el tiempo máximo de recuperación no deberá exceder **4 horas**.

3.3.5 REQUISITOS DE MANTENIBILIDAD

El sistema DELI-BIRD debe ser diseñado para facilitar su mantenimiento, asegurando que las tareas puedan ser realizadas de manera eficiente y con un impacto mínimo en las operaciones. A continuación, se especifican los requisitos de mantenibilidad:

1. Tipos de Mantenimiento

- **Mantenimiento Correctivo:** Identificación y solución de errores o fallos en el sistema detectados durante su uso.
- **Mantenimiento Preventivo:** Actualización de software, bibliotecas y dependencias utilizadas para prevenir fallos futuros.
- **Mantenimiento Evolutivo:** Incorporación de nuevas funcionalidades o modificaciones solicitadas para adaptarse a los cambios en los procesos operativos.
- **Mantenimiento Adaptativo:** Ajustes necesarios para garantizar la compatibilidad con cambios en los sistemas operativos, hardware o tecnologías utilizadas.

2. Responsables de Mantenimiento

- **Usuarios Administradores:**

- Realizar tareas básicas, como limpieza de datos antiguos o generación de reportes.
- Configuración de parámetros operativos, como horarios de operación o límites de alertas.

- **Equipo de Desarrollo:**

- Corrección de errores críticos detectados en el sistema.
- Realización de actualizaciones mayores, como nuevas versiones o módulos funcionales.
- Ajustes de compatibilidad con hardware y software.

3. Frecuencia y Tiempos de Mantenimiento

- **Mantenimiento Diario:**

- Revisión automatizada de logs del sistema para identificar anomalías.
- Respaldo automático de datos operativos en servidores seguros.

- **Mantenimiento Semanal:**

- Generación de estadísticas de acceso y operación del sistema.
- Revisión de actualizaciones menores disponibles para su implementación.

- **Mantenimiento Mensual:**

- Limpieza de datos obsoletos en las bases de datos para optimizar el rendimiento.
- Verificación del estado de los dispositivos de hardware conectados al sistema.

- **Mantenimiento Trimestral:**

- Evaluación de desempeño del sistema y ajuste de configuraciones según las necesidades.
- Implementación de actualizaciones críticas de software o dependencias.

- **Mantenimiento Anual:**

- Auditoría completa del sistema para garantizar su seguridad, rendimiento y cumplimiento con las regulaciones.
- Planificación e implementación de mejoras evolutivas requeridas.

4. Herramientas y Documentación

- El sistema debe incluir herramientas internas para:
 - Monitorización en tiempo real de procesos críticos.
 - Generación automática de reportes de mantenimiento.
- La documentación técnica debe estar actualizada y ser accesible para el equipo de desarrollo, incluyendo manuales de usuario y guías para el mantenimiento.

3.3.6 PORTABILIDAD

La portabilidad del sistema **DELI-BIRD** garantiza que las aplicaciones puedan ejecutarse en distintos entornos sin necesidad de modificaciones significativas. Se desarrollarán dos versiones independientes:

1. **Aplicación móvil:** Diseñada para dispositivos **Android e iOS**, optimizada para operarios y administradores en aeropuertos.
2. **Aplicación web:** Accesible desde navegadores modernos, dirigida principalmente a administradores y supervisores de logística.

Dependencia del servidor

- **Porcentaje de componentes dependientes del servidor:** Mínimo, ya que gran parte de la lógica de negocio reside en el cliente para mejorar la eficiencia y la experiencia del usuario.
- **Porcentaje de código dependiente del servidor:** Solo se requerirá comunicación con el backend para el acceso a datos y validaciones críticas.

Compatibilidad con múltiples plataformas

- **Aplicación móvil:**

- Desarrollada en **React Native**, garantizando compatibilidad con **Android e iOS** sin necesidad de código duplicado.
- Optimizada para ejecutarse en dispositivos móviles dentro de aeropuertos, con acceso en línea y offline.
- **Aplicación web:**
 - Desarrollada en **React con Vite**, compatible con navegadores modernos (Chrome, Edge, Firefox, Safari).
 - Accesible desde cualquier sistema operativo (Windows, macOS, Linux) sin requerir instalación.

Estrategias de migración y mantenimiento

- Uso de **APIs RESTful** para la comunicación entre frontend y backend.
- Arquitectura modular que facilita la adaptación a nuevas plataformas en el futuro.
- **Sincronización en la nube** para garantizar la disponibilidad de datos en ambas versiones.

3.4 OTROS REQUISITOS

Requisitos Legales y Regulatorios:

- *La aplicación **DELI-BIRD** debe cumplir con la normativa de seguridad aeroportuaria establecida por la **IATA (International Air Transport Association)** y la **OACI (Organización de Aviación Civil Internacional)**.*
- *La gestión de datos personales de los empleados y administradores debe cumplir con la **Ley Federal de Protección de Datos Personales en Posesión de Particulares (México)** y el **GDPR (Reglamento General de Protección de Datos de la Unión Europea)** en caso de operaciones internacionales.*

- *Los registros de cargamentos deben mantenerse por un periodo mínimo de **5 años**, según las regulaciones de logística y transporte aéreo.*

Requisitos Culturales y Políticos:

- *La aplicación debe ser **multilinguaje**, soportando al menos **español e inglés**, para adaptarse a operadores de distintos países.*
- *Se deben respetar **zonas horarias locales** en el registro y monitoreo de cargamentos para evitar discrepancias en la planificación logística.*

Requisitos de Usabilidad y Accesibilidad:

- *La interfaz debe seguir las pautas de **WCAG 2.1** (Web Content Accessibility Guidelines) para garantizar la accesibilidad a usuarios con discapacidad visual o motriz.*
- *Se debe incluir un **modo de alto contraste** y soporte para lectores de pantalla.*
- *El sistema debe ser compatible con dispositivos móviles **Android**, optimizado para tabletas utilizadas en el entorno aeroportuario.*

Requisitos Éticos y Ambientales:

- *La aplicación debe **minimizar el consumo de energía** en dispositivos móviles, promoviendo el uso eficiente de recursos.*
- *Se priorizará el uso de servidores con **infraestructura ecológica**, con proveedores que utilicen fuentes de energía renovable.*

4 Metodologías

La metodología extreme programming(XP) es una metodología de desarrollo de software que forma parte de lo que se conoce colectivamente como metodologías ágiles. XP se basa en valores, principios y prácticas, y su objetivo es permitir que equipos pequeños y medianos produzcan software de alta calidad y se adapten a los requisitos cambiantes y en evolución.

Se va a utilizar esta metodología ya que permite adaptarse rápidamente a cambios, entregar software funcional de forma frecuente, mejorar la calidad

con prácticas como TDD e integración continua, y mantener una comunicación cercana con el cliente. Además, reduce riesgos, motiva al equipo y asegura que el producto final cumpla con las necesidades del usuario. Es ideal para proyectos que requieren flexibilidad y entrega constante de valor.

5. Selección de tecnologías

Para el desarrollo del sistema DELI-BIRD, se seleccionarán las siguientes tecnologías para garantizar eficiencia, seguridad y escalabilidad:

Backend

- **Lenguaje y Framework:** Laravel (PHP) para la gestión eficiente de la lógica del servidor y la API.
- **Base de Datos:** PostgreSQL para el almacenamiento estructurado y seguro de la información de los cargamentos y usuarios.
- **Autenticación:** Middleware de autenticación de Laravel con tokens JWT para sesiones seguras.

Frontend

- **Framework Web:** React para la creación de una interfaz web dinámica y optimizada.
- **Aplicación Móvil:** React Native para una experiencia fluida en dispositivos iOS y Android.

Infraestructura y Desarrollo

- **Repositorio de Código:** GitHub para el control de versiones y colaboración en el desarrollo.
- **Pruebas y API Testing:** Postman para la validación de endpoints y pruebas de integración.

Seguridad y Monitoreo

- Encriptación de datos: Uso de Hashing Bcrypt para credenciales de usuario y cifrado de datos sensibles.
- Manejo de Errores: Implementación de logs y monitoreo con Laravel Debugbar.

Integraciones

- Mapas y geolocalización: Uso de Google Maps API para el rastreo en tiempo real de los cargamentos.
- Escaneo de códigos QR: Implementación de bibliotecas compatibles con React Native para la identificación de paquetes.

6. Arquitecturas

Para este proyecto se utilizará una arquitectura de microservicios la cual trabajaremos para dividir en diversos servicios que se comunicarán por medio de API, se realizará el servicio del backend de manera encapsulada, la aplicación móvil y la aplicación web que tendrá funcionalidad de PWA.

A su vez se utilizarán microservicios externos para lograr un software robusto que cuente con diversas tecnologías que aseguren funcionalidades específicas para el objetivo principal.

Algunos de los servicios externos que se utilizarán serán:

Neon: Esta herramienta permitirá hostear nuestra base de datos del RBDMS PostgreSQL.

AccuWeather: Este servicio nos permitirá obtener cualidades climatológicas sobre las localizaciones por donde las rutas serán recorridas.

SMTP de Gmail: Se utilizará este servicio para el envío de correos con especificaciones acerca de la orden del cliente.

AWS Rekognition: Esto permitirá tener un método de autenticación adicional a la aplicación para tener un acceso controlado.

7. Flujo de Trabajo

Se utilizará esta herramienta de versionamiento ya que es una la cual provee diversas funcionalidades para un correcto versionamiento y trabajo por medio de ramas a su vez facilita el trabajo en equipo, así también, cuenta con herramientas integradas para el CI/CD como flujos automatizados.

La manera en que se maneja el sistema de versionamiento será el siguiente:

User_branch -> Esta será la rama en que cada desarrollador realizará sus cambios.

Development -> Aquí se unirán los cambios de los desarrolladores para hacer test del correcto funcionamiento de la aplicación.

main -> Una vez probadas las funcionalidades y que nada falla en la rama de desarrollo aquí se unirán los cambios, una vez incluidos los cambios, automáticamente se desplegará en el servidor lo cual permitirá la integración continua y su despliegue automático.