

## ***La Ubereta***

Memoria del Proyecto del Ciclo Formativo de GS IFC303

**Desarrollo de aplicaciones web**  
Curso: 2024/2025

**Marco Formación**

Autor: Valentín López Tapia-Gómez



## INDICE

<u>1. Descripción.</u>	1
<u>2. Objetivos</u>	2
<u>3. Recursos Empleados</u>	3
<u>4. Desarrollo o Explicación</u>	9
<u>5. Conclusión y Posibles Mejoras</u>	17
<u>6. Bibliografía y Referencias</u>	18
<u>7. Anexos</u>	19

# 1. Descripción.

## 1.1 Introducción



La presente memoria corresponde al desarrollo del proyecto "**La Uberneta**", una plataforma web pensada para gestionar y administrar las operaciones internas de una empresa de transporte similar a Uber, de uso exclusivo para empleados y administradores. A través de esta intranet, se facilita la gestión de conductores, facturas, usuarios entre otras cosas, ofreciendo un entorno centralizado y seguro para que los empleados interactúen y gestionen la información.

Este proyecto se realiza en el marco del **Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Web (IFC303)** y tiene como objetivo aplicar los conocimientos adquiridos en tecnologías como PHP, MySQL y otros nuevos como ha sido el caso de DART, así como en la creación de sistemas seguros y eficientes.

## 1.2 Descripción del Proyecto

El proyecto consiste en el desarrollo de una plataforma web interna de gestión para empleados de una empresa de transporte. **Uberneta** permitirá a los administradores gestionar los perfiles de conductores, contabilizar facturas,, comunicarse con el personal a través del tablón de anuncios y gestionar dichas facturas de los clientes.

A través de esta aplicación web, se pueden realizar las siguientes acciones:

- **Gestión de usuarios:** Registro y edición de perfiles de conductores y administradores.
- **Gestión de facturas:** Subir, visualizar y descargar diferentes tipos de facturas.
- **Generador de facturas:** Permite generar la factura para un cliente en base al precio y posterior envío de dicha factura al correo del cliente registrado en nuestra BBDD, en caso de que no estuviera registrado podría registrarla el propio conductor con el dni y correo de la persona interesada para más facturas en un futuro.
- **Visualización de calendario:** Consulta del calendario de festivos de cada empleado.
- **Mensajería interna:** Comunicación entre los distintos departamentos y conductores a través de un tablón general

## 2. Objetivos

### 2.1 Objetivo General



El objetivo general del proyecto es desarrollar una intranet corporativa eficiente, segura y fácil de usar que permita gestionar las facturas, conductores y vehículos de una empresa de transporte.

### 2.2 Objetivo Específico

- Desarrollar una plataforma web que facilite la gestión de conductores y sus facturas.
- Implementar una base de datos segura y eficiente que almacene la información necesaria (usuarios, facturas, calendario).
- Crear una interfaz de usuario intuitiva y adaptativa a diferentes dispositivos.
- Establecer un sistema de autenticación y autorización de usuarios con roles diferenciados.
- Garantizar la seguridad de los datos mediante cifrado y validación adecuada.
- Optimizar la escalabilidad del sistema para futuras ampliaciones.



### 3. Recursos Empleados

#### 3.1 Tecnologías y Herramientas Utilizadas:

##### Frontend:

- **Flutter** (SDK de Google para desarrollo multiplataforma)
- **Dart** (Lenguaje principal para lógica y UI)
- **State Management:** Provider y Riverpod (gestión del estado)
- **Flutter Packages:**

##### 1. GESTIÓN DE ESTADO Y UI

- provider (^6.0.0): Gestión del estado de la aplicación
- flutter\_bloc (^9.0.0): Implementación del patrón BLoC
- font\_awesome\_flutter (^10.1.0): Iconos Font Awesome

##### 2. COMUNICACIÓN Y REDES

- http (^1.2.2): Peticiones HTTP básicas
- dio (^5.2.0): Cliente HTTP avanzado

##### 3. ALMACENAMIENTO Y ARCHIVOS

- shared\_preferences (^2.0.15): Almacenamiento local
- flutter\_secure\_storage (^9.2.2): Almacenamiento seguro
- file\_picker (^6.1.1): Selección de archivos
- path\_provider (^2.0.11): Gestión de rutas

#### 4. PDF Y DOCUMENTOS

- pdf (^3.11.1): Generación de PDFs
- printing (^5.13.4): Impresión de documentos
- syncfusion\_flutter\_pdfviewer (28.2.12): Visualizador de PDFs
- html\_to\_pdf (^0.8.1): Conversión HTML a PDF

#### 5. CALENDARIO

- table\_calendar (^3.0.0): Widget de calendario
- syncfusion\_flutter\_calendar (^28.2.7): Calendario avanzado

#### 6. SEGURIDAD

- bcrypt (^1.1.3): Encriptación de contraseñas
- crypto (^3.0.0): Funciones criptográficas
- permission\_handler (^11.3.1): Gestión de permisos

### Backend:

- **PHP 8.0**, orientado a objetos y organizado bajo el patrón **MVC**
- Controladores por módulos: usuarios, facturas, vehículos, mensajes
- **REST API** para conexión entre frontend y backend (respuesta en formato JSON)

La API cuenta con estos documentos:

#### Gestión de Usuarios

- **login.php**: Maneja el proceso de inicio de sesión de usuarios
  - Usado en: `login.dart`
- **forgot\_password.php**: Gestiona la recuperación de contraseñas
  - Usado en: `login.dart`
- **reset\_password.php**: Procesa el restablecimiento de contraseña después de la solicitud de recuperación
  - Usado en: `login.dart`
- **update\_password.php**: Actualiza las contraseñas de los usuarios
  - Usado en: `configuracion/change\_password\_page.dart`
- **eliminar\_cuenta.php**: Maneja el proceso de eliminación de cuentas de usuario
  - Usado en: `delete\_account\_page.dart`
- **get\_user\_data.php**: Obtiene los datos del usuario
  - Usado en: `profile\_page.dart`
- **update\_user\_data.php**: Actualiza la información del usuario
  - Usado en: `profile\_page.dart`
- **gestion\_usuarios.php**: Administra las operaciones CRUD de usuarios
  - Usado en: `admin\_usuarios\_page.dart`
- **dar\_alta\_cliente.php**: Maneja el registro de nuevos clientes
  - Usado en: `factura\_abono\_page.dart`, `factura\_servicio\_uberneta\_page.dart`

## Gestión de Facturas

- **insertar\_factura\_cliente.php**: Inserta nuevas facturas de clientes
  - Usado en: `factura\_abono\_page.dart`, `factura\_servicio\_uberneta\_page.dart`
- **obtener\_razon\_social.php**: Obtiene la razón social de un cliente por DNI
  - Usado en: `factura\_abono\_page.dart`, `factura\_servicio\_uberneta\_page.dart`
- **obtener\_id\_cliente.php**: Obtiene el ID de un cliente por DNI
  - Usado en: `factura\_abono\_page.dart`, `factura\_servicio\_uberneta\_page.dart`
- **listar\_factura\_cliente.php**: Lista las facturas de un cliente específico
  - Usado en: `listado\_facturas\_page.dart`
- **fetch\_facturas.php**: Obtiene el listado de facturas
  - Usado en: `admin\_facturas\_page.dart`
- **actualizar\_estado\_factura.php**: Actualiza el estado de las facturas
  - Usado en: `admin\_facturas\_page.dart`

## Gestión de Documentos

- **subirFactura.php**: Maneja la subida de facturas
  - Usado en: `upload\_pdf\_page.dart`
- **upload\_pdf\_web.php**: Gestiona la subida de archivos PDF
  - Usado en: `upload\_pdf\_page.dart`
- **fetch\_pdfs.php**: Obtiene los PDFs almacenados
  - Usado en: `tablon.dart`
- **fetch\_pdfs\_dashboard.php**: Obtiene los PDFs para el dashboard
  - Usado en: `dashboard.dart`, `tablon.dart`, `documentos\_anio\_page\_tablon.dart`, `services/documentos\_service.dart`
- **fetch\_pdfs\_estado.php**: Obtiene los PDFs para la página de estado
  - Usado en: `estado\_facturas\_page.dart`

- **delete\_document.php**: Elimina documentos
  - Usado en: `estado\_facturas\_page.dart`
- **upload\_tablon\_document.php**: Sube documentos al tablón
  - Usado en: `admin\_tablon\_page.dart`

### Gestión del Tablón

- **fetch\_tablon\_documents.php**: Obtiene los documentos del tablón
  - Usado en: `tablon.dart`, `admin\_tablon\_page.dart`

### Otros

- **enviar\_correo.php**: Maneja el envío de correos electrónicos
  - Usado en: `factura\_abono\_page.dart`, `factura\_servicio\_uberneta\_page.dart`
- **calendario.php**: Gestiona las operaciones relacionadas con el calendario
  - Usado en: `calendario.dart`
- **eliminar\_token.php**: Elimina tokens de sesión
  - Usado en: `dashboard.dart`
- **config.php**: Archivo de configuración básico para la conexión a la base de datos y configuraciones generales
  - Usado en: Todos los archivos PHP que requieren conexión a la base de datos
- **header.php**: Archivo de encabezado común que incluye configuraciones, sesiones y funciones compartidas
  - Usado en: Todos los archivos PHP que requieren configuración inicial

**Base de Datos:**

- **MySQL 8**
- Herramientas: PhpMyAdmin, MySQL Workbench
- Diseño normalizado (3FN) para evitar redundancia

**Desarrollo y Entorno:**

- IDEs: Visual Studio Code y Android Studio (PHP y Flutter)
- **XAMPP** para entorno local (Apache + MySQL + PHP)
- **Postman** para pruebas de API

**3.2 Recursos Humanos y Académicos:**

- Formación recibida en el ciclo formativo IFC303.
- Asesoramiento y orientación del profesorado.
- Documentación técnica oficial de PHP y MySQL.
- Comunidad técnica de Stack Overflow y foros especializados.



## 4. Desarrollo o Explicación

El proyecto se desarrolló siguiendo una arquitectura **cliente-servidor** que facilita la separación de la lógica de negocio (backend) y la interfaz de usuario (frontend). A continuación se explica el desarrollo en varias fases.

### 4.1 Estructura General del Sistema:

El proyecto se divide en tres capas claramente separadas:

1. **Frontend (Flutter/Dart)**: interfaz gráfica y lógica de usuario
2. **Backend (PHP)**: lógica de negocio, validaciones, generación de datos
3. **Base de datos (MySQL)**: almacenamiento de datos persistente

La comunicación se realiza mediante **peticiones HTTP** seguras (HTTPS) y datos en formato **JSON**.

### 4.2 Desarrollo del Frontend con Flutter:

El **frontend** de *La Ubernetá* ha sido desarrollado completamente con **Flutter Web**, usando **Dart** como lenguaje de programación. Esto permite construir una **aplicación web responsive, rápida y moderna** desde un único código base que, en un futuro, también puede adaptarse a dispositivos móviles.

#### 4.2.1. ¿Por qué Flutter Web?

- Permite construir interfaces ricas en el navegador sin necesidad de HTML/CSS tradicionales.
- Reutilización de componentes (widgets).
- Alto rendimiento gracias a compilación a JavaScript.
- Compatible con diseño adaptativo y temas oscuros/claros.
- Posibilita una futura transición a app móvil sin reescribir toda la lógica.

#### 4.2.2. Arquitectura del Frontend

Se utilizó una **arquitectura limpia y modular** para mantener el código escalable y mantenible:

- **Presentación (UI)**: Widgets como Scaffold, AppBar, Drawer, ListView etc.
- **Gestión de estado**: Usamos Provider para desacoplar lógica de negocio de la interfaz.
- **Servicios HTTP**: Clases Dart que consumen la API REST desarrollada en PHP.
- **Rutas y navegación**: Implementación con Navigator 2.0 o GoRouter para navegar entre pantallas protegidas por roles.

#### 4.2.3. Pantallas Principales del Frontend

Cada vista o módulo del sistema ha sido implementado como una pantalla (StatefulWidget o StatelessWidget) con widgets propios.

##### a) Pantalla de Login

- Formulario validado con, Validator.
- Botón de login que llama a un servicio HTTP con http.post.
- Al autenticar, se guarda un token o ID en memoria para mantener la sesión.

##### b) Panel de Administración

- Dashboard con paneles y accesos directos a diferentes sitios de la web
- Navegación lateral (Drawer) con rutas a: gestión de usuarios, facturas, calendario.

##### c) Gestión de Conductores

- Tablas con DataTable para visualizar y editar registros.
- Formularios para añadir o modificar información.
- Botones con iconos (IconButton) para editar o eliminar.

#### d) Facturación

- Subida y descarga de facturas con FilePicker y url\_launcher.
- Visualización de facturas PDF generadas en el backend.
- Filtrado y ordenación por fecha y conductor.

#### e) Calendario

- Integración de paquetes como table\_calendar.
- Cada conductor puede ver sus días festivos.
- Los administradores pueden actualizar el calendario desde el backend.

### 4.2.4. Gestión del Estado

Usamos Provider como patrón de gestión de estado principal:

- AuthProvider: para manejar login, logout y sesión.
- FacturaProvider: para cargar, filtrar y actualizar facturas.
- UsuarioProvider: para CRUD de usuarios.

Esto permite que los widgets escuchen cambios en los datos y se actualicen de forma automática sin necesidad de setState() en exceso.

### 4.2.5. Comunicación con el Backend

- Se usa el paquete http para hacer llamadas REST a la API en PHP.
- Las respuestas en JSON se parsean con dart:convert.
- Gestión de errores y desconexiones con try-catch.

### 4.2.6. Seguridad del Frontend

- **Validación de formularios** en el cliente antes de enviar los datos.
- **Autenticación**: sólo acceden a pantallas protegidas los usuarios autenticados.
- **Autorización**: según el rol del usuario (admin, conductor), se limita lo que puede ver/hacer.

- Los tokens de sesión o ID se almacenan con SharedPreferences o Provider (no en la URL).

#### 4.2.7. Diseño y Responsividad

- Todo el diseño se adapta a navegadores de escritorio y tablets.
- Se usan layouts con LayoutBuilder, MediaQuery, y Flexible para adaptarse.
- Interfaz clara, con espacios amplios y paleta de colores institucional.
- Iconos de material.dart, navegación fluida con animaciones entre vistas.

#### 4.2.8. Pruebas y Mantenimiento

- Pruebas manuales del frontend en Chrome y Firefox.
- Separación en widgets reutilizables mejora la mantenibilidad.
- Código documentado con comentarios y tipos estrictos.



### 4.3 Base de Datos:

La base de datos fue diseñada utilizando **MySQL** y contiene varias tablas interrelacionadas, entre otras:

#### 1. TABLA: calendariofestivos

Atributos:

- IdCalendarioFestivo (PK)
- IdFestivo
- FechaFestivo
- TtitleFestivo
- Festivo
- Color

Relaciones:

- No tiene relaciones con otras tablas

## 2. TABLA: clientes

Atributos:

- IdCliente (PK)
- Nombre
- Contrasena
- IdTipoUsuario
- TelefonoUsuario
- DNIIUsuario
- Direccion
- Poblacion
- Provincia
- CodigoPostal
- CorreoUsuario
- IdEmpresa
- CodUsuario
- EstadoUsuario

Relaciones:

- Con tipousuario: N:1 (IdTipoUsuario)

## 3. TABLA: documentos

Atributos:

- IdDocumento (PK)
- IdUsuario
- TituloDocumento
- DescripcionDocumento
- URLDocumento
- FechaPublicacionDocumento
- AnioDocumento
- TipoDocumento
- IdEstadoDocumento

Relaciones:

- Con usuario: N:1 (IdUsuario)
- Con estadosdocumento: N:1 (IdEstadoDocumento)

#### 4. TABLA: documentosfacturas

Atributos:

- IdDocumentoFactura (PK)
- IdUsuario
- IdConductor
- TituloDocumento
- DescripcionDocumento
- URLDocumento
- FechaPublicacionDocumento
- IdEstadoDocumento
- ObservacionesFactura

Relaciones:

- Con usuario: N:1 (IdUsuario)
- Con estadosdocumento: N:1 (IdEstadoDocumento)

#### 5. TABLA: estadosdocumento

Atributos:

- IdEstadoDocumento (PK)
- NombreEstado
- DescripcionEstado

Relaciones:

- Con documentos: 1:N
- Con documentosfacturas: 1:N

#### 6. TABLA: facturas

Atributos:

- IdFactura (PK)
- IdCliente
- FechaFactura
- NumeroFactura
- IdUsuarioConductor
- Concepto
- RazonSocial
- ObservacionesFactura
- Precio
- IVA
- Enlace
- IdUsuarioFacturacion

Relaciones:

- Con clientes: N:1 (IdCliente)
- Con usuario: N:1 (IdUsuarioConductor, IdUsuarioFacturacion)

### 7. TABLA: options

Atributos:

- id (PK)
- group
- item
- value

Relaciones:

- No tiene relaciones con otras tablas

### 8. TABLA: tipousuario

Atributos:

- idTipoUsuario (PK)
- DescripcionTipo
- IdEmpresa

Relaciones:

- Con usuario: 1:N
- Con clientes: 1:N

### 9. TABLA: usuario

Atributos:

- IdUsuario (PK)
- Nombre
- Contrasena
- IdTipoUsuario
- TelefonoUsuario
- DNIUsuario
- CorreoUsuario
- IdEmpresa
- CodUsuario
- Direccion
- Provincia
- CodigoPostal
- EstadoUsuario
- IdFestivo
- Recordatorio
- reset\_expiry
- Token

Relaciones:

- Con tipousuario: N:1 (IdTipoUsuario)
- Con documentos: 1:N
- Con documentosfacturas: 1:N
- Con facturas: 1:N (como IdUsuarioConductor e IdUsuarioFacturacion)

NOTAS:

- Las relaciones 1:N indican que un registro puede tener múltiples registros relacionados
  - Las relaciones N:1 indican que múltiples registros pueden estar relacionados con un solo registro
  - Las claves foráneas (FK) mantienen la integridad referencial de la base de datos

#### 4.4 Seguridad:

El sistema implementa buenas prácticas de seguridad, como el **cifrado de contraseñas** mediante **bcrypt** y **validaciones de entrada** para evitar inyecciones SQL. Además, se utiliza **HTTPS** para proteger la comunicación entre el cliente y el servidor.



## 5. Conclusión y Posibles Mejoras

El desarrollo de **Intranet Uber** ha sido un éxito, cumpliendo con los objetivos iniciales del proyecto. El sistema es funcional, seguro y fácil de usar. Los usuarios pueden gestionar facturas, consultar estadísticas y comunicarse eficientemente a través de la mensajería interna.

### 5.1 Posibles Mejoras:

- **Versión Móvil:** Una futura versión podría permitir la integración con aplicaciones móviles.
- **Apartado Clientes:** Crear un apartado donde pueden entrar los Clientes con las cuentas que le son creadas con su DNI y una contraseña para que puedan ver facturas de otros viajes, incluso poder pedir un coche para ese momento o reservarlo.
- **Calendario:** Que el calendario se pueda gestionar mediante la web por los administradores y no solo por la bbdd
- **Notificaciones en tiempo real:** Implementar **WebSockets** para notificaciones en vivo, como la contabilización de las facturas y actualizaciones de estado.
- **Optimización del rendimiento:** Mejorar la velocidad de carga de las páginas, especialmente en secciones con grandes volúmenes de datos.
- **Pruebas automatizadas:** Implementar **tests unitarios** y **pruebas funcionales automatizadas** para garantizar la estabilidad del sistema.



## 6. Bibliografía y Referencias

### 6.1 Documentación:

[PHP - Documentación Oficial](#)

[MySQL - Documentación Oficial](#)

[Flutter.dev – Documentación oficial](#)

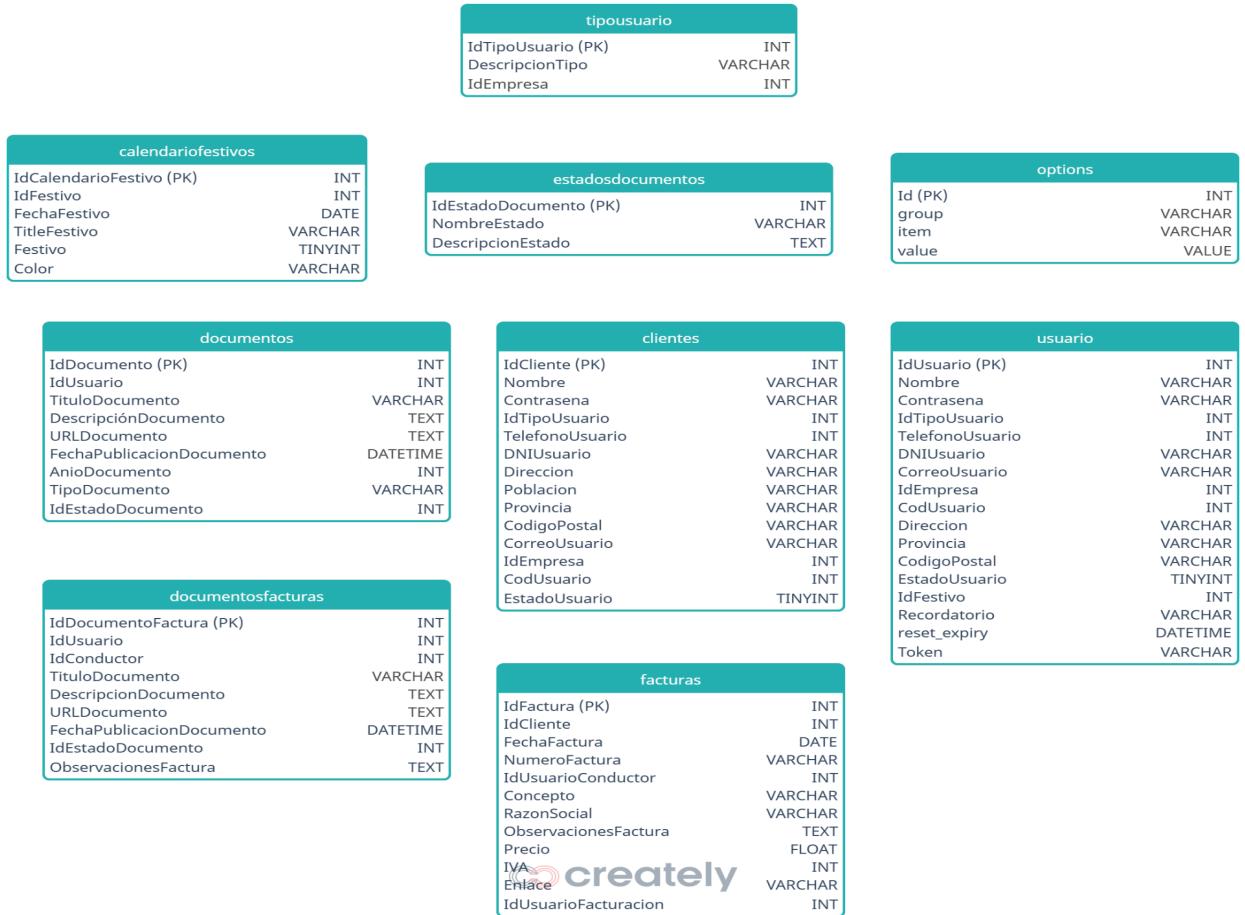
[Dart.dev – Lenguaje Dart](#)

**Stack Overflow:** Foro de desarrolladores donde se resolvieron dudas técnicas durante el proyecto.



## 7. Anexos

### 7.1 Diagrama de Clases (DC)



created by

## 7.2 Diagrama de Casos de Uso (UML)

### ACTORES PRINCIPALES

1. Superadmin 2. Admin 3. Conductor 4. Sistema

### CASOS DE USO POR ACTOR

#### 1. SUPERADMIN

##### 1.1 Gestión de Administradores

- Crear administradores
- Modificar administradores
- Desactivar administradores
- Asignar permisos de administrador

##### 1.2 Gestión del Sistema

- Configuración global del sistema
- Gestión de tipos de usuario
- Configuración de permisos
- Gestión de opciones del sistema

#### 2. ADMIN

##### 2.1 Gestión de Conductores

- Registrar nuevos conductores
- Modificar datos de conductores
- Desactivar conductores
- Asignar permisos de conductor

## 2.2 Gestión de Documentos

- Subir documentos
- Asignar estado a documentos
- Gestionar tipos de documentos
- Asociar documentos a facturas

## 2.3 Gestión de Facturas

- Crear facturas
- Modificar facturas
- Asignar documentos a facturas
- Gestionar estados de facturas

## 3. CONDUCTOR

### 3.1 Gestión de Perfil

- Ver perfil
- Modificar datos personales
- Cambiar contraseña

### 3.2 Gestión de Documentos

- Subir documentos personales
- Ver documentos asociados
- Descargar documentos
- Gestionar estados de documentos

### 3.3 Gestión de Facturas

- Ver facturas asociadas
- Descargar facturas
- Ver documentos de facturas
- Gestionar observaciones
- Generar Facturas

## 4. SISTEMA

### 4.1 Gestión de Sesiones

- Validar credenciales
- Gestionar tokens
- Controlar expiración de sesiones

### 4.2 Gestión de Archivos

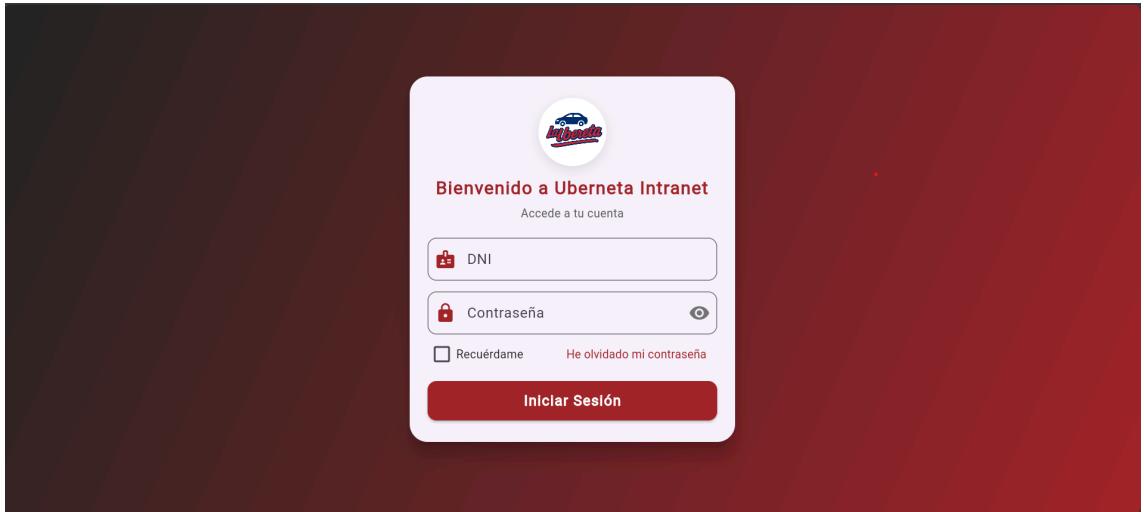
- Almacenar documentos
- Gestionar URLs de documentos
- Controlar acceso a archivos

## NOTAS

- Todos los actores pueden iniciar sesión y cerrar sesión
- El sistema mantiene un registro de todas las acciones realizadas
- Los permisos se gestionan de forma jerárquica (Superadmin > Admin > Conductor > Cliente)
- Los documentos y facturas poseen tipos de estados
- Cada nivel de administración tiene acceso a las funcionalidades de los niveles inferiores

## 7.3 Capturas de Pantalla

- Login



- Panel del administrador

Este es un dashboard administrativo. En la parte superior izquierda, se muestra la información del usuario logeado: 'Uberneta', nombre 'Valentín López Ta...', correo electrónico 'valeenn811@gmail.com' y un icono de perfil. A la derecha, se presentan tres cuadros azules: 'Documentación', 'Tablón de Anuncios' y 'Subir Facturas'. Abajo de estos, se encuentran más cuadros: 'Generar Factura' (destacado en azul), 'Calendario', 'Administración' (título central) y tres cuadros oscuros: 'Administrar Tablón', 'Administrar Facturas' y 'Administrar Usuarios'. En la parte inferior, hay un botón rojo que dice 'Cerrar sesión'.

- Panel del Conductor

The screenshot shows the Uber conductor panel. At the top left is the profile icon and name "Uberneta". Below it is a sidebar with navigation links: Panel de control, Documentación, Tablón de anuncios, Subir facturas, Generar factura, Calendario, and Configuración. To the right are five cards: "Documentación" (red), "Tablón de Anuncios" (dark blue), "Subir Facturas" (red), "Generar Factura" (dark blue), and "Calendario" (red). At the bottom is a red "Cerrar sesión" button.

- Calendario del conductor

The screenshot shows the Uber conductor calendar for June 2025. The sidebar includes the same navigation links as the previous panel. The main area displays a grid of dates from June 26 to July 6. Red boxes highlight specific dates: June 26, 27, 3, 9, 10, 16, 23, 24, 30, and July 1, 5, 6. A red circle labeled "Festivo" is placed over the date July 1. The days of the week are labeled LUN, MAR, MIÉ, JUE, VIE, SÁB, and DOM.

- Tablón de Anuncios

The screenshot shows the Uberneta application interface. At the top, there's a header with the logo 'Uberneta' and a user profile for 'Valentín López Ta...' with the email 'valeemn811@gmail.com'. Below the header is a sidebar with various menu items: 'Panel de control', 'Documentación', 'Tablón de anuncios' (which is highlighted in red), 'Subir facturas', 'Generar factura', 'Calendario', and 'Configuración'. To the right of the sidebar, there's a main content area titled 'Documentos del Año 2025'. It contains a single document entry: '2025-333333333-0001.pdf' (PDF icon), with the description 'Primer anuncio' and the date 'Fecha: 2025-06-08 12:55:00'. A red circular button with a white question mark is located in the top right corner of the content area. At the bottom of the screen is a red button labeled 'Cerrar sesión'.

- Factura en PDF

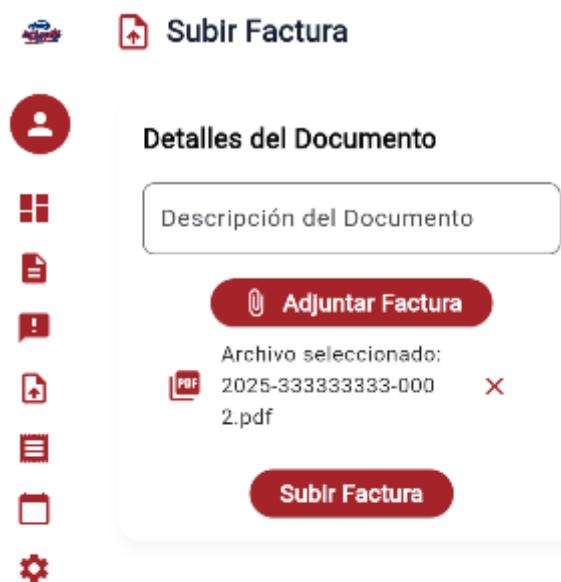
The screenshot shows the Uberneta application interface. At the top, there's a header with the logo 'Uberneta' and a user profile for 'Valentín López Ta...' with the email 'valeemn811@gmail.com'. Below the header is a sidebar with various menu items: 'Panel de control', 'Documentación', 'Tablón de anuncios', 'Subir facturas', 'Generar factura' (which is highlighted in red), 'Calendario', and 'Configuración'. To the right of the sidebar, there's a main content area titled 'Facturas - Año 2025'. It contains a single document entry: '2025333333330001.pdf' (PDF icon), with the description 'Primera factura, Fecha: 2025-06-08 12:53:53'. A red circular button with a white question mark is located in the top right corner of the content area. At the bottom of the screen is a red button labeled 'Cerrar sesión'.

- Vista Móvil Generar Factura

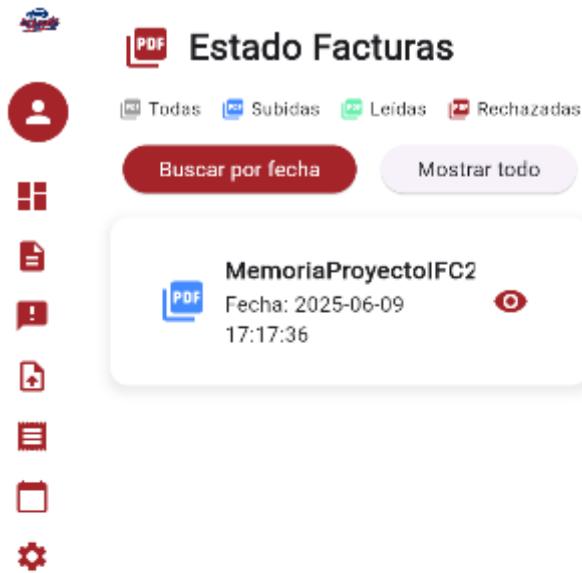
The image shows a mobile application interface for generating a facture. On the left, there is a vertical sidebar with red icons: a car at the top, followed by a person, a grid, a document, an exclamation mark, a plus sign, a list, a calendar, and a gear at the bottom. To the right of the sidebar, the main content area has a title 'Factura Servicio Ubernet' with an icon of a car. Below it is a field labeled 'Concepto\*: Servicio Ubernet'. There are three input fields: 'DNI/NIE/CIF Cliente\*' (with a document icon), 'Precio (€)\*' (with an exclamation mark icon), and 'Observaciones' (with a list icon). At the bottom is a red button labeled 'Guardar y Generar Factura'.



- Vista Móvil de Subir Factura



- Vista Móvil de Estado Facturas





## 7.4 Manual de Usuario

### 7.4.1 Requisitos del Sistema

#### Requisitos Mínimos

- Sistema Operativo: Windows 10/11, macOS 10.14+, o Linux
- Procesador: Dual Core 2.0 GHz o superior
- Memoria RAM: 8GB mínimo (16GB recomendado)
- Espacio en Disco: 10GB libres
- Conexión a Internet: Requerida para la instalación inicial

#### Requisitos de Software

##### 1. XAMPP

- Versión: 8.2.x o superior
- Componentes necesarios:
  - Apache 2.4.x
  - MySQL 8.0.x
  - PHP 8.2.x
  - phpMyAdmin

##### 2. Flutter SDK

- Versión: ^3.7.2
- Dart SDK: Incluido con Flutter
- Android Studio (recomendado para desarrollo)
- Visual Studio Code con extensiones:
  - Flutter
  - Dart
  - PHP Intelephense
  - Git

##### 3. Git (opcional pero recomendado)

- Versión: 2.x o superior

### 7.4.2 Instalación de Software Necesario

#### 1. Instalación de XAMPP

1. Descargar XAMPP desde <https://www.apachefriends.org/es/index.html>
2. Ejecutar el instalador como administrador
3. Seleccionar los componentes:
  - Apache
  - MySQL
  - PHP
  - phpMyAdmin
4. Elegir la carpeta de instalación (por defecto: C:\xampp)
5. Completar la instalación
6. Iniciar XAMPP Control Panel
7. Verificar que Apache y MySQL estén funcionando

### 2. Instalación de Flutter

1. Descargar Flutter SDK desde <https://flutter.dev/docs/get-started/install>
2. Extraer el archivo ZIP en una ubicación permanente (ej: C:\src\flutter)
3. Agregar Flutter a las variables de entorno:
  - Abrir Variables de Entorno del Sistema
  - Editar la variable PATH
  - Agregar la ruta al directorio flutter\bin
4. Verificar la instalación:  
`flutter doctor`
5. Instalar Android Studio y configurar el SDK de Android
6. Instalar las extensiones de VS Code:
  - Flutter
  - Dart

### 3. Instalación de Git

1. Descargar Git desde <https://git-scm.com/downloads>
2. Ejecutar el instalador
3. Usar la configuración por defecto
4. Verificar la instalación:  
`git --version`

## **7.4.3 Configuración del Entorno**

### 1. Configuración de XAMPP

1. Iniciar XAMPP Control Panel
2. Configurar puertos (si es necesario):
  - Apache: 80 (por defecto)
  - MySQL: 3306 (por defecto)
3. Copiar la carpeta `servidorUber` a `C:\xampp\htdocs\`

### 2. Configuración de la Base de Datos

1. Abrir phpMyAdmin (<http://localhost/phpmyadmin>)
2. Crear nueva base de datos:
  - Nombre: uberneta
  - Collation: utf8mb4\_unicode\_ci
3. Importar el esquema de la base de datos:
  - Seleccionar la base de datos uberneta
  - Ir a la pestaña "Importar"
  - Seleccionar el archivo SQL del proyecto
  - Ejecutar la importación

## **7.4.4 Configuración del Proyecto Flutter**

### 1. Preparación del Proyecto

1. Clonar o descomprimir el proyecto en: `C:\Users\[usuario]\OneDrive\Documentos\`
2. Abrir una terminal en la carpeta del proyecto
3. Ejecutar:  
`flutter pub get`

## 2. Configuración de Archivos

### 1. Verificar `lib/config.dart`:

```
class Config {  
    static const String baseUrl = 'http://localhost/servidorUber/';  
    static const String baseUrlDocumentos = 'http://localhost/servidorUber/uploads/';  
    static const String baseUrlDocumentosTodos =  
        'http://localhost/servidorUber/uploads/todos/';  
    static const String baseUrlSubirFacturas =  
        'http://localhost/servidorUber/uploads/facturas/';  
    static const String baseUrlDocumentosEstadoFacturas =  
        'http://localhost/servidorUber/uploads/facturas/';  
}
```

### 2. Verificar `pubspec.yaml`:

- Asegurarse de que todas las dependencias estén correctamente especificadas
- Verificar las versiones de las dependencias

## **7.4.5 Estructura del Proyecto**

### 1. Estructura de Carpetas

C:\Users\[usuario]\OneDrive\Documentos\uberneta\

```
lib/  
  └── configuracion/  
  └── models/  
  └── preferences/  
  └── services/  
  └── util/  
    └── main.dart  
    └── [archivos principales]  
  └── img/  
  └── fonts/  
  └── [archivos de configuración]
```

C:\xampp\htdocs\servidorUber\

```
uploads/  
  └── todos/  
    └── facturas/  
    └── [archivos PHP]
```

### 2. Componentes Principales

- Frontend (Flutter)
- Páginas de autenticación
- Gestión de documentos
- Facturación
- Administración
- Interfaz de usuario

- Backend (PHP)
- API REST
- Gestión de archivos
- Autenticación
- Base de datos

#### 7.4.6 Ejecución de la Web

##### 1. Iniciar el Servidor

1. Abrir XAMPP Control Panel
2. Iniciar Apache y MySQL
3. Verificar que ambos servicios estén en verde

##### 2. Iniciar la Aplicación Flutter

1. Abrir Android Studio
2. Abrir el proyecto desde: `C:\Users\[usuario]\OneDrive\Documentos\uberneta`
3. Seleccionar un emulador web; edge, chrome...
4. Presionar el botón de "Run" o usar `Shift + F10`

##### 3. Acceso a la Aplicación

- URL de la aplicación: <http://localhost:XXXX> (puerto dinámico)
- Credenciales por superadmin:
  - Usuario: 12345678A
  - Contraseña: admin1
- Credenciales por admin:
  - Usuario: 12345678B
  - Contraseña: admin2
- Credenciales Conductor
  - Usuario: 12345678C
  - Contraseña conductor1

#### 7.4.7 Solución Problemas

##### 1. Problemas Comunes

Error de Conexión con el Servidor

- Verificar que XAMPP esté corriendo
- Comprobar las URLs en config.dart
- Verificar la conexión a la base de datos

Error de Base de Datos

- Verificar que MySQL esté corriendo
- Comprobar las credenciales
- Verificar la existencia de la base de datos

Error de Dependencias Flutter

flutter clean

flutter pub get

flutter run

## 2. Logs y Depuración

- Logs de Apache: C:\xampp\apache\logs
- Logs de MySQL: C:\xampp\mysql\data
- Logs de Flutter: Ver en la consola de desarrollo

## **7.4.8 Mantenimiento**

1. Actualizaciones
  - Mantener XAMPP actualizado
  - Actualizar Flutter SDK
  - Actualizar dependencias:  
flutter pub upgrade

## **7.4.9 Recursos Adicionales**

### 1. Documentación

- [Documentación de Flutter](<https://flutter.dev/docs>)
- [Documentación de PHP](<https://www.php.net/docs.php>)
- [Documentación de MySQL](<https://dev.mysql.com/doc/>)

### 2. Herramientas de Desarrollo

- [Android Studio](<https://developer.android.com/studio>)
- [Visual Studio Code](<https://code.visualstudio.com/>)
- [Postman](<https://www.postman.com/>) (para probar APIs)
- [Git](<https://git-scm.com/>)

### 3. Comunidades y Soporte

- [Stack Overflow](<https://stackoverflow.com/>)
- [Flutter Community](<https://flutter.dev/community>)
- [PHP Community](<https://www.php.net/support.php>)