La Uberneta

Memoria del Proyecto del Ciclo Formativo de GS IFC303

Desarrollo de aplicaciones web

Curso: 2024/2025

Marco Formación

Autor: **Valentín López Tapia-Gómez**

INDICE

1. Descripción. 1
2. Objetivos 2
3. Recursos Empleados 3
4. Desarrollo o Explicación 4
5. Conclusión y Posibles Mejoras 5
6. Bibliografía y Referencias 6
7. Anexos 7

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 casa 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 registrarlo 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 Uberneta* ha sido desarrollado completamente con **Flutter Web**, usando **Dart** como lenguaje de programación. Esto permite construir una **aplicación web responsiva, 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.
- UusuarioProvider: 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:
- id (PK)
- fecha
- descripcion
- activo

Relaciones:
- No tiene relaciones con otras tablas

2. TABLA: clientes

Atributos:
- id (PK)
- nombre
- apellidos
- email
- telefono
- direccion
- fecha_alta
- activo

Relaciones:
- No tiene relaciones con otras tablas

3. TABLA: documentos

Atributos:
- id (PK)
- nombre
- tipo
- fecha_creacion
- estado_id (FK)
- usuario_id (FK)
- activo

Relaciones:
- Con estadosdocumento: 1:N (estado_id)
- Con usuario: N:1 (usuario_id)
- Con documentosfacturas: 1:N

4. TABLA: documentosfacturas

Atributos:
- id (PK)
- documento_id (FK)
- factura_id (FK)
- fecha_asociacion

Relaciones:
- Con documentos: N:1 (documento_id)
- Con facturas: N:1 (factura_id)

5. TABLA: estadosdocumento

Atributos:
- id (PK)
- nombre
- descripcion
- activo

Relaciones:
- Con documentos: 1:N

6. TABLA: facturas

Atributos:
- id (PK)
- numero
- fecha
- importe
- cliente_id (FK)
- estado
- fecha_creacion
- activo

Relaciones:
- Con clientes: N:1 (cliente_id)
- Con documentosfacturas: 1:N

7. TABLA: options

Atributos:
- id (PK)
- nombre
- valor
- descripcion
- activo

Relaciones:
- No tiene relaciones con otras tablas

8. TABLA: tipousuario

Atributos:
- id (PK)
- nombre
- descripcion
- permisos
- activo

Relaciones:
- Con usuario: 1:N

9. TABLA: usuario

Atributos:
- id (PK)
- nombre
- apellidos
- email
- password
- tipo_usuario_id (FK)
- ultimo_acceso
- activo

Relaciones:
- Con tipousuario: N:1 (tipo_usuario_id)
- Con documentos: 1:N

NOTAS:
- Todas las tablas incluyen campos de auditoría (fecha_creacion, fecha_modificacion)
- 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 Entidad-Relación (DER)

7.2 Diagrama de Casos de Uso (UML)

7.3 Capturas de Pantalla

- Login
- Panel del administrador
- Calendario del conductor
- Tablón de Anuncios
- Factura en PDF
- Vista Móvil de Subir Factura

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 defecto:
  - Usuario: cooperativa
  - Contraseña: [consultar con administrador]

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)