



MÓDULO PROYECTO

CFGS Desarrollo de Aplicaciones
Multiplataforma
Informática y Comunicaciones

RiberPublic Fichajes

Tutor individual: Rodrigo Iglesias Gorrón

Tutor colectivo: Cristina Silvan Pardo

Año: 2025

Fecha de presentación: 23/05/2025

Nombre y Apellidos: Adrián Alonso Pérez

Email: adrianalonso200@gmail.com

**Foto actual
del alumno**

Contenido

1	Identificación proyecto	5
2	Organización de la memoria	5
3	Descripción general del proyecto	6
3.1	Objetivos	6
•	Objetivo principal	6
•	Objetivos secundarios.....	6
3.2	Cuestiones metodológicas	6
3.3	Entorno de trabajo (tecnologías de desarrollo y herramientas).....	6
4	Descripción general del producto.....	7
4.1	Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.....	7
4.2	Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.....	8
4.3	Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha.....	8
5	Planificación y presupuesto	10
5.1	Planificación	10
5.2	Presupuesto.....	11
6	Documentación Técnica: análisis, diseño, implementación y pruebas.....	11
6.1	Especificación de requisitos	11
6.2	Análisis del sistema	11
6.2.1	Gestión de fichajes	11
6.2.2	Gestión de usuarios.....	12
6.2.3	Gestión de grupos	12
6.2.4	Gestión de ausencias	13
6.2.5	Gestión de horarios.....	13
6.2.6	Generación de Excel.....	14

6.3	Diseño del sistema:	15
6.3.1	Diseño de la Base de Datos.....	15
6.3.2	Diseño de la Interfaz de usuario.	18
6.3.3	Diseño de la Aplicación.	18
6.4	Implementación:	20
6.4.1	Entorno de desarrollo.	20
6.4.2	Estructura del código.	21
6.4.3	Cuestiones de diseño e implementación reseñables.....	22
6.5	Pruebas.....	24
6.5.1	Seguridad	24
6.5.2	Pruebas La siguiente tabla sirve para entender los resultados que pueden ocurrir por las peticiones del postman.	24
7	Manuales de usuario	27
7.1	Manual de usuario.....	27
7.1.1	Paso por paso.....	27
7.2	Manual de instalación	43
7.2.1	Requisitos mínimos	43
7.3	Manual de instalación	44
8	Conclusiones y posibles ampliaciones	44
8.1	Conclusiones.....	44
8.2	Posibles mejoras.....	44
9	Bibliografía	45
9.1	Frontend	45
9.2	Backend	45
9.3	Otros.....	45
10	Anexos	46
10.1	Flutter	46

10.1.1	Modelos	46
10.1.2	Providers	52
10.1.3	Servicios	53
10.1.4	Utils	59
10.1.5	Main	61
10.2	Java	62
10.2.1	Modelos	62
10.2.2	Repositorios	67
10.2.3	Documentación	69
10.2.4	Seguridad	69
10.2.5	Application.properties	72
10.2.6	Base de datos	73

1 Identificación proyecto

Mi proyecto, trata de una aplicación **multiplataforma**, para **registrar y administrar la jornada laboral de empleados**. Permitiendo fichar con NFC para confirmar que este en su espacio laboral para trabajar. Ofrece la opción de exportar las ausencias de los empleados con su información.

La parte visual, esta desarrollada en Dart, con el FrameWork de **Flutter**, ya que es muy útil para hacer aplicaciones multiplataforma.

La parte de la lógica esta desarrollada en Java con el FrameWork de **SpringBoot**, ya que es muy potente y escalable.

En cuanto a la base de datos, utilizo **SQL** al funcionar muy bien con SpringBoot.

2 Organización de la memoria

Apartados	Descripción
1- Identificación del Proyecto	Breve resumen del proyecto y su distribución.
2- Organización de la memoria	Breve descripción de los apartados.
3- Descripción general del proyecto	Objetivos, metodologías y tecnologías.
4- Descripción general del producto	Funcionalidades básicas, arquitecturas y despliegue.
5- Planificación y presupuesto	Tiempo trabajado en el proyecto y cuánto costaría el mismo.
6- Documentación técnica	Análisis, diseño, implementación y pruebas.
7- Manual de usuario	Manual de usuario y manual de instalación.
8- Conclusiones	Mis conclusiones y mejoras que quiero hacer.
9- Bibliografía	Fuentes consultadas para realizar el proyecto.
10- Anexo	Demostración de la gran mayoría del código.

3 Descripción general del proyecto

3.1 Objetivos

- **Objetivo principal**

Crear una aplicación multiplataforma que simule un entorno laboral real para que mis compañeros de administración de empresa puedan registrar las entradas y salidas de su horario. Facilitando al profesorado el registro de ausencias reales.

- **Objetivos secundarios**

1. **NFC:** Implementar NFC para empezar y finalizar la jornada.
2. **Exportar a Excel:** Exportar los grupos, junto a sus usuarios a Excel.

3.2 Cuestiones metodológicas

Seguí un modelo de **ciclo de vida en cascada**.

1. **Análisis de requisitos:** Necesidades de alumnos y profesorado.
2. **Diseño:** Diagramas ER, UML y prototipos de interfaz.
3. **Implementación:** Desarrollo incremental en módulos.

3.3 Entorno de trabajo (tecnologías de desarrollo y herramientas)

Front-end (APP - Flutter)

- **Framework:** Flutter 3.1
- **Lenguaje:** Dart 3.6.0
- **IDE:** Visual Studio Code
- **Diseño:** Figma
- **NFC:** NFC Tools
- **Arquitectura de software:** Model, view, view-model.

Back-end (API - SpringBoot)

- **Framework:** Spring Boot 2.6.8
- **Lenguaje:** Java 21
- **IDE:** IntelliJ IDEA Ultimate
- **Comprobaciones:** PostMan

- **Arquitectura de software:** RESTful API (controlador, servicio, repositorio)

Base de datos

- **Gestor:** MySQL 8.0
- **IDE:** MySQL Workbench
- **Tipo:** Relacional
- **Esquema:** Tablas de:
 - Usuarios
 - Grupos
 - Fichajes
 - Ausencias
 - Horarios

Otras herramientas y utilidades

- **Control de versiones:** Git con repositorios en GitHub para la API y la aplicación.
- **Documentación API:** Swagger UI y JavaDOC.
- **Documentación APP:** DartDOC.

4 Descripción general del producto

4.1 Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.

- **Límites del sistema:**

El proyecto, está dividido esencialmente en dos programas, la **API** y la **APP**. La **APP** es dependiente de la **API** ya que necesita los datos para mostrarlos por pantalla. La **API** es dependiente únicamente de que este compilada la base de datos.

- **Funcionalidades básicas:**

- Gestión de fichajes (registros).
- Gestión de usuarios.

- Gestión de grupos.
- Gestión de ausencias.
- Gestión de horarios.
- Generación de Excel.

- **Usuarios:**

Existen dos tipos de Usuarios, “**jefe**” y “**empleado**”. Cuando tu entras en la aplicación accedes a un login, ahí introduces los datos y según la respuesta del API indica si eres un jefe o un empleado y te redirecciona a la administración o a la aplicación.

La aplicación, no tiene opción de registrarse, al ser una aplicación privada, un administrador tendría que crear previamente el usuario.

La aplicación esta preparada principalmente para Windows y Android, pero en teoría se podría utilizar tanto en IOS como en Linux (digo en teoría porque no he tenido la oportunidad de probarlo, pero los endpoints están preparados para ello).

4.2 Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.

- **Arquitectura Cliente - Servidor RESTful:** Elegí Spring Boot para el backend, porque es lo que utilizo día a día en el trabajo, aparte ser fácil de mantener.
- **Patrón Controlador - Servicio - Repositorio:** Es una buena forma de organizar el código dejando claro la función de cada uno.
- **Provider como gestión de estados:** En flutter opte a utilizar los providers por su integración nativa con los widgets y su fácil gestión.
- **Persistencia con JPA y MySQL:** JPA es óptimo para usar con SQL y es sencillo de implementar y SQL es un Sistema Gestor relacional muy potente.
- **Seguridad con Spring Security y JWT:** Proporciona un mecanismo de autenticación bueno, ligero, “sencillo de implementar” y fácil de integrar con flutter.

4.3 Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha

- **Plataforma tecnológica:**
 - Windows 11
 - IntelliJ IDEA Ultimate (o community)
 - Visual Studio Code
 - MySQL Workbench 8.0 (funcionando)
 - Git

- **Despliegue en local:**

Clonamos ambas aplicaciones por la línea de comando:

“git clone <https://github.com/adree3/RiberRepublicFichajeApp.git>”

“git clone <https://github.com/adree3/RiberRepublicFichajeApi.git>”

Desplegamos la base de datos en MySQL **Workbench**, el script está en el api en “**/src/main/resources/bd/RiberPublicFichajesBd.sql**”. En caso de que el usuario y la contraseña del Workbench no sea “root” y “toor”. Ir al **application.properties** y poner el user y password correspondiente.

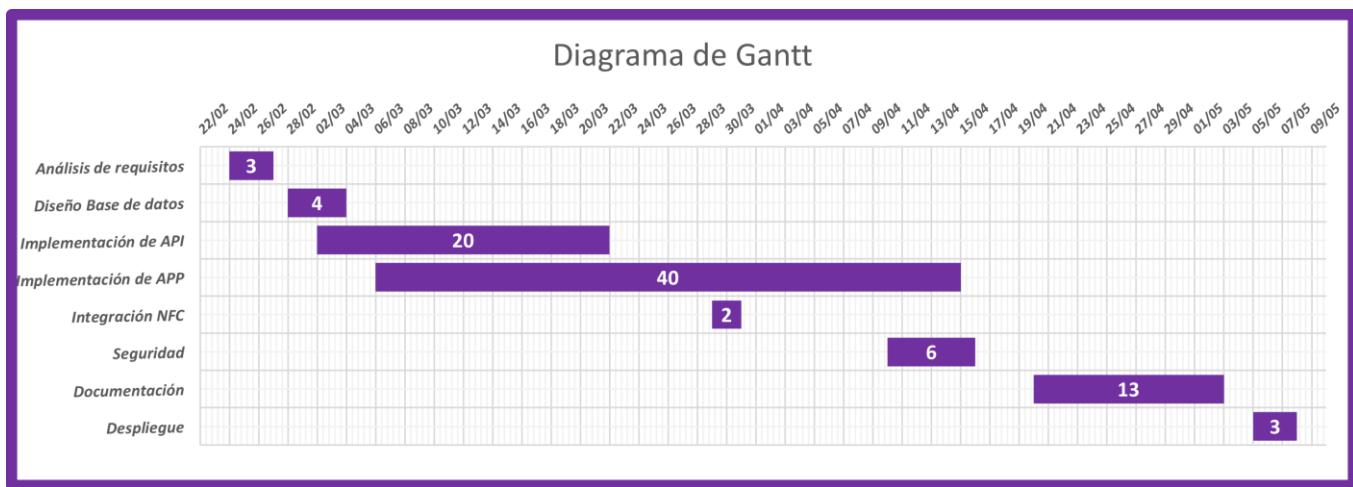
Ahora iniciamos desde el **IntelliJ** el API y desde **Visual Studio** iniciamos el proyecto (La parte de usuario esta echo para móvil, lo cual si no pones un emulador puede que de fallos. La parte de administración está adaptada para ambas vistas)



5 Planificación y presupuesto

5.1 Planificación

Excel con el diagrama de Gantt calculado en el repositorio.



Actividad	Inicio	Días	Final
Análisis de requisitos	24/02/2025	3	27/02/2025
Diseño Base de datos	28/02/2025	4	04/03/2025
Implementación de API	02/03/2025	20	22/03/2025
Implementación de APP	06/03/2025	40	15/04/2025
Integración NFC	29/03/2025	2	31/03/2025
Seguridad	10/04/2025	6	16/04/2025
Documentación	20/04/2025	13	03/05/2025
Despliegue	05/05/2025	3	08/05/2025

5.2 Presupuesto

Concepto	Unidad	Cantidad	Coste unitario (€)	Total (€)
Análisis	h	30	20.00 €	600.00 €
Diseño	h	24	20.00 €	480.00 €
Desarrollo Backend	h	75	20.00 €	1500.00 €
Desarrollo Frontend	h	150	20.00 €	3000.00 €
Licencia MySQL	licencia	1	50.00 €	50.00 €
Servidor Cloud	mes	3	30.00 €	30.00 €
IntelliJ Ultimate	licencia	1	100.00€	100.00 €
Total				5760.00 €

6 Documentación Técnica: análisis, diseño, implementación y pruebas.

6.1 Especificación de requisitos

6.2 Análisis del sistema

Según viene en el punto 4 “Funcionalidades Básicas”, profundizare en cada punto.

6.2.1 Gestión de fichajes

- **Alta de fichaje**
 - Campos
 - UsuarioId: Identificador del empleado
 - FechaHora: Fecha y hora de entrada y salida
 - Nfc: Saber si ha utilizado el nfc o no para fichar
 - Validaciones
 - FechaHora: Posterior al último registro del usuario
 - Nfc: Comprobar si lo ha utilizado
- **Listado de fichajes**
 - Campos

- UsuarioId: Identificador del empleado
- FechaDesde: Fecha cuando empezó a trabajar
- FechaHasta: Fecha cuando dejó de trabajar

6.2.2 Gestión de usuarios

- **Alta de usuario**
 - Campos
 - Nombre
 - Apellidos
 - Email
 - Contraseña
 - Rol
 - Estado
 - Validaciones
 - Email: Formato de email valido y único
 - Contraseña: Contraseña segura
- **Listado de usuarios**
 - Filtrado del usuario por su estado => activo.
- **Modificación de usuario**
 - Campos editables
 - Nombre
 - Apellidos
 - Email
 - Contraseña
 - Rol
 - Estado
 - Validaciones
 - Email: Formato de email valido y único
 - Contraseña: Contraseña segura
- **Baja de usuario**
 - Solicitud de confirmación de la baja del usuario.

6.2.3 Gestión de grupos

- **Alta de grupo**
 - Campos
 - Nombre
 - Lista de usuarios
 - Validaciones
 - Nombre: Nombre único.

- **Listado de grupos**
 - Lista de grupos con los usuarios asociados.
- **Modificación de grupo**
 - Campos editables
 - Nombre
 - Lista de usuarios
- **Baja de grupo**
 - Solicitud de confirmación de la baja del grupo.
 - Reasignación de usuarios a grupo por defecto (Sin Asignar).

6.2.4 Gestión de ausencias

- **Alta de ausencia**
 - Campos
 - Usuario
 - Fecha
 - Motivo
 - Descripción
 - Estado
 - Justificada
 - Validaciones
 - Usuario: Que exista el usuario
 - Una única ausencia al día.
- **Listado de ausencias**
 - Lista de ausencias filtrado por grupo y nombre.
- **Modificación de ausencia**
 - Campos editables
 - Estado

6.2.5 Gestión de horarios

- **Alta de horario**
 - Campos
 - Grupo
 - Día
 - HoraEntrada
 - HoraSalida
 - Validaciones
 - Horas: la hora de entrada no puede ser posterior a la de salida.
- **Listado de horarios**

- Lista de horarios, filtrado por día y grupo.
- **Modificación de horario**
 - Campos editables
 - Grupo
 - Día
 - HoraEntrada
 - HoraSalida
- **Baja de horario**
 - Solicitud de confirmación de la baja del horario.

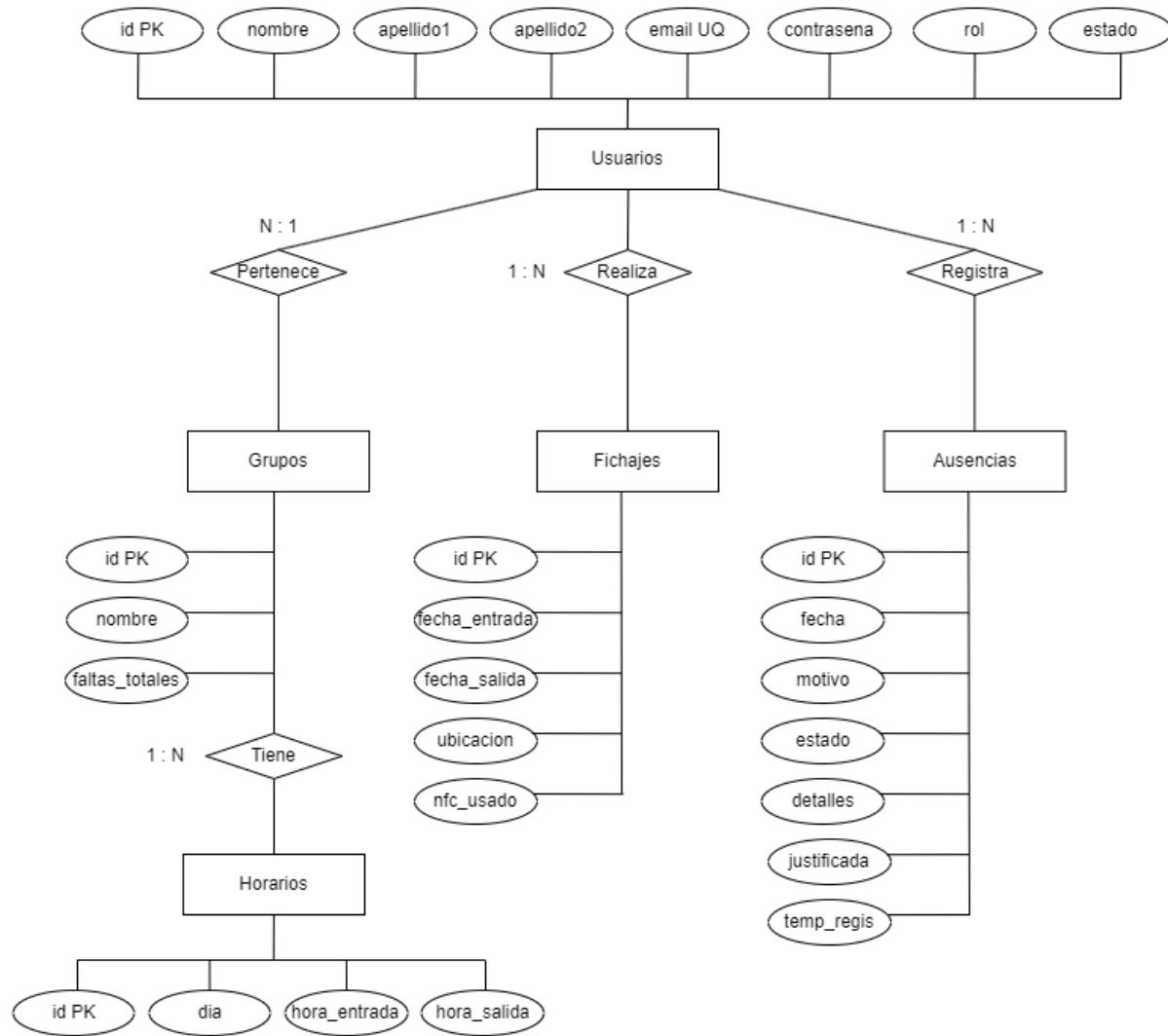
6.2.6 Generación de Excel

- **Exportación de datos**
 - Selección
 - Un grupo
 - Todos los grupos
 - Validaciones
 - El grupo tenga al menos 1 alumno
- **Formato**
 - Fichero .xlsx con una única hoja con cabecera:
 - Nombre
 - Apellidos
 - Email
 - Grupo
 - N.º ausencias

6.3 Diseño del sistema:

6.3.1 Diseño de la Base de Datos

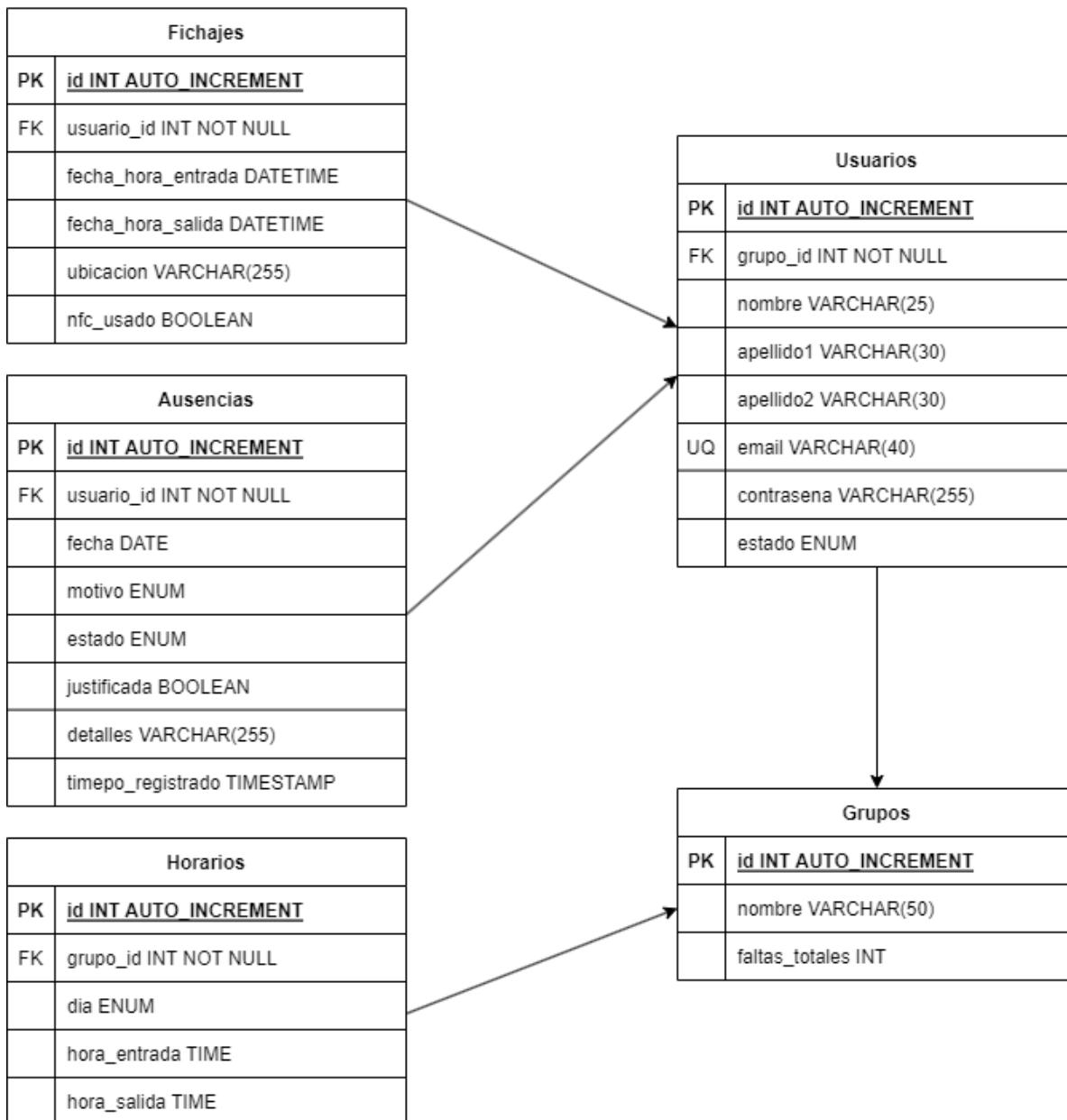
Modelo Entidad Relacion



(archivo .drawio disponible en el repositorio)



Diagrama de Tablas



(archivo .drawio disponible en el repositorio)

Esquema Base de Datos

```

5 • ◑ CREATE TABLE grupos (
6     id int auto_increment primary key,
7     nombre varchar(50) not null,
8     faltas_totales int not null
9 );
10 • ◑ CREATE TABLE usuarios (
11     id int auto_increment primary key,
12     nombre varchar(25) not null,
13     apellido1 varchar(30) not null,
14     apellido2 varchar(30),
15     email varchar(40) unique not null,
16     contrasena varchar(255) not null,
17     rol ENUM('empleado', 'jefe') default 'empleado',
18     grupo_id int,
19     estado ENUM('activo', 'inactivo') default 'activo',
20     FOREIGN KEY (grupo_id) REFERENCES grupos(id) ON DELETE CASCADE
21 );
22 • ◑ CREATE TABLE horarios (
23     id int auto_increment primary key,
24     grupo_id int not null,
25     dia ENUM('lunes', 'martes', 'miercoles', 'jueves', 'viernes') not null,
26     hora_entrada time not null,
27     hora_salida time not null,
28     FOREIGN KEY (grupo_id) REFERENCES grupos(id) ON DELETE CASCADE
29 );
30 • ◑ CREATE TABLE fichajes (
31     id int auto_increment primary key,
32     usuario_id int not null,
33     fecha_hora_entrada DATETIME DEFAULT NULL,
34     fecha_hora_salida DATETIME DEFAULT NULL,
35     ubicacion varchar(255),
36     nfc_usado BOOLEAN DEFAULT FALSE,
37     FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE
38 );
39 • ◑ CREATE TABLE ausencias (
40     id int auto_increment primary key,
41     usuario_id int not null,
42     fecha date not null,
43     motivo ENUM('falta_injustificada', 'enfermedad', 'vacaciones', 'permiso', 'retraso', 'otro') DEFAULT 'falta_injustificada',
44     estado ENUM('vacio', 'pendiente', 'aceptada', 'rechazada') default 'vacio',
45     justificada boolean default false,
46     detalles varchar(255),
47     timestamp registrado TIMESTAMP DEFAULT CURRENT_TIMESTAMP.

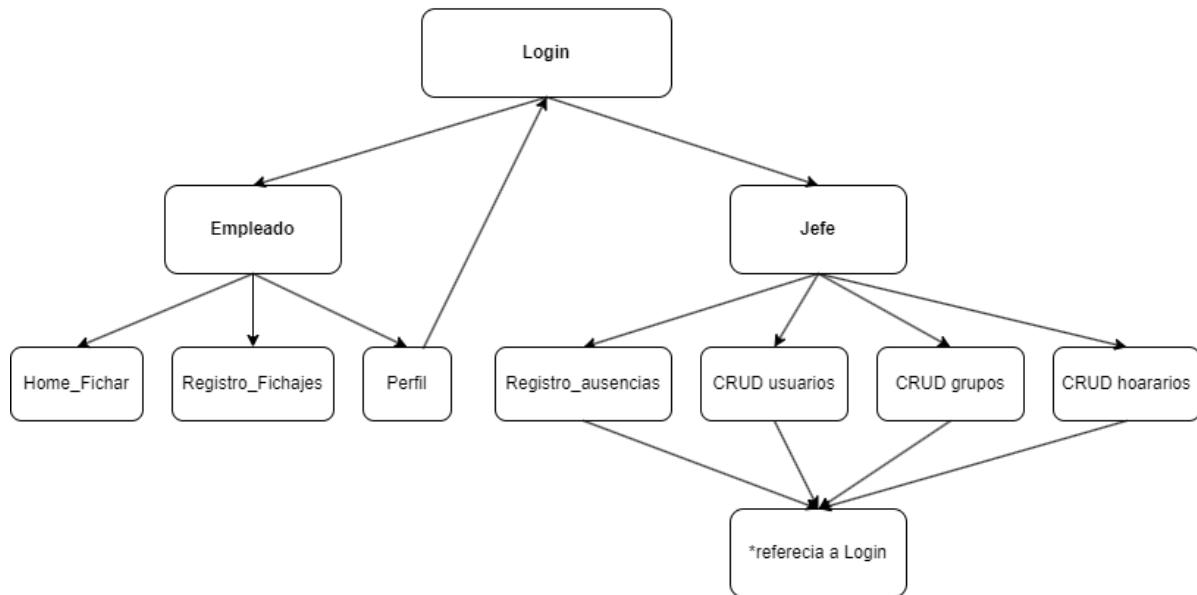
```

(archivo .sql disponible en el repositorio)

6.3.2 Diseño de la Interfaz de usuario.

En cuanto el diseño de la interfaz, al ser el proyecto para un cliente, me pidieron un diseño del interfaz del usuario, y lo hice en **Figma**, en cuanto la interfaz del jefe no me la pidieron:
[LINK](#)

Flujo de Pantallas



(archivo .drawio disponible en el repositorio)

6.3.3 Diseño de la Aplicación.

6.3.3.1 Login / Autenticación - Empleado

- El empleado introduce sus credenciales
- La APP los valida con un POST a /usuarios/login
- La API valida las credenciales con la base de datos y genera un TOKEN y devuelve el usuario completo sin la contraseña
- La APP recibe el token, lo guarda en el provider junto al usuario y si tiene el rol de empleado avanza a Fichar

6.3.3.2 Registro de fichaje

- El empleado pulsa fichar, elige el método
- Si elige NFC, se invoca al lector NFC y a su correspondiente pantalla
- Si es correcto, se genera una petición POST a /fichajes/abrirFichaje, JWT valida el Token y la API crea en la base de datos el fichaje

- Si le vuelve a dar, después de abrir el fichaje, hace un PUT a /fichajes/cerrarFichaje, JWT valida el Token y la API edita el fichaje añadiendo la hora de salida

6.3.3.3 Listado de fichajes

- La APP pide los fichajes, horarios a la API, esta comprueba el token y devuelve de la base de datos los fichajes y horarios
- Si registra una ausencia, la APP envía un POST a /ausencias/nuevaAusencia con los datos necesarios, el API valida el token, y crea la ausencia en la base de datos

6.3.3.4 Perfil

- Si el usuario cambia el tema, la APP guarda en el provider de tema y se cambia
- Si da a cambiar contraseña, la APP valida que la contraseña actual sea correcta, y la contraseña nueva, sea una contraseña segura (6 caracteres o más, 1 mayúscula y 1 número)
- Si cierra sesión, la APP elimina del provider de usuarios tanto el token como el usuario y le navega al login.

6.3.3.5 Login / Autenticación – Jefe

- El empleado introduce sus credenciales
- La APP los valida con un POST a /usuarios/login
- La API valida las credenciales con la base de datos y genera un TOKEN y devuelve el usuario completo sin la contraseña
- La APP recibe el token, lo guarda en el provider junto al usuario y si tiene el rol de jefe avanza a ausencias

6.3.3.6 Listado de ausencias

- La APP pide un GET a /ausencias/ las ausencias
- La API valida el token y envia las ausencias
- Si se modifica el estado de la ausencia, se le envia un PUT a /ausencias/editarAusencia, comprueba el token y edita el estado
- Si se da a generar ausencias, la APP envia un POST a /ausencias/generarAusencias y la API por unos requisitos, genera las ausencias que no estaban creadas.

6.3.3.7 CRUD usuarios

- La APP pide los usuarios y grupos a la API, esta valida el token y envía el Json
- Si se edita un usuario, la APP valida los campos y se envía un PUT a /usuarios/editarUsuario el API valida el token y edita el usuario
- Si elimina un usuario, la APP confirma que lo quieres eliminar y envía un DELETE a /usuarios/eliminarUsuario, la API valida el token y elimina el usuario

- Si se crea un usuario, la APP pide los parámetros y los valida, envía un POST a /usuarios/crearUsuario, el API valida el token y crea el usuario.

6.3.3.8 CRUD grupos

- La APP pide los grupos y usuarios a la API, esta valida el token y envía los datos
- Si se edita un grupo, la APP valida los campos y se envía un PUT a /grupos/editarGrupo el API valida el token y edita el grupo
- Si elimina un grupo, la APP confirma que lo quieres eliminar y envía un DELETE a /grupos/eliminarGrupo, la API valida el token y elimina el grupo y redirige los usuarios de ese grupo a él por defecto “Sin Asignar”, el cual no se puede ni editar ni eliminar
- Si se crea un grupo, la APP pide los parámetros y los valida, envía un POST a /grupos/crearGrupo, el API valida el token y crea el grupo

6.3.3.9 CRUD horarios

- La APP pide los grupos, usuarios y horarios a la API, esta valida el token y envía los datos
- Si se edita un horario, la APP valida los campos y se envía un PUT a /horarios/editarHorario el API valida el token y edita el horario
- Si elimina un horario, la APP confirma que lo quieres eliminar y envía un DELETE a /horarios/eliminarHorario, la API valida el token y elimina el
- Si se crea un horario, la APP pide los parámetros y los valida, envía un POST a /horarios/crearHorario, el API valida el token y crea el horario

6.4 Implementación:

6.4.1 Entorno de desarrollo.

En cuanto el entorno de desarrollo, es decir las aplicaciones utilizadas para la construcción del proyecto, son las siguientes:

- **Visual Studio Code:** IDE utilizado para el desarrollo del FRONT, junto al lenguaje Dart y el frameWork Flutter. Utilizo flutter por su gran eficaz a crear aplicaciones multiplataforma.
- **IntelliJ IDEA Ultimate:** IDE utilizado para el desarrollo del BACK, junto al lenguaje Java 21 y el framWork SpringBoot. Utilizo la combinación de java + SpringBoot por si gran eficiencia, escalabilidad y desarrollo.
- **MySQL WorkBench:** Utilizado como cliente para el diseño, creación y gestión de la base de datos MySQL 8.0.
- **NFC Tools:** Utilizado para la escritura de las tarjetas NFC, utilizadas para el fichaje.
- **PostMan:** Aplicación muy útil para las pruebas a los endpoints del BACK.

- **Figma:** Aplicación utilizada para el diseño de la parte de usuarios, para poderlo presentar al cliente.
- **Git – GitHub:** Utilizado para los flujos de trabajo.

6.4.2 Estructura del código.

6.4.2.1 Librerías descargadas

- **FRONT**

- **Http:** Para hacer peticiones REST.
- **Table_calendar:** Para seleccionar fechas de forma más gráfica.
- **FlutterToast:** Para notificar al usuario mensajes.
- **Shared_preferences:** Para guardar en local del usuario, datos como el usuario o el token.
- **Provider:** Para gestionar el estado, como el tema o el usuario, viene de la mano con shared_preferences.
- **Path:** Sirve para manipular rutas de archivos, en mi caso para el file_selector del Excel.
- **Collection:** Para agrupar, buscar y ordenar datos en memoria.
- **Fluter_typeAhead:** Sirve para sugerencias automáticas, en mi caso lo empleo a la hora de editar o crear grupos, para que me sugiera usuarios.
- **File_selector:** Para que el usuario pueda seleccionar donde quiere guardar en su equipo un documento. En mi caso para saber dónde quiere guardar el Excel.
- **Intl:** Sirve para el formateo de fechas principalmente.
- **Excel:** Para generar archivos .xlsx, lo utilizo para exportar los grupos a Excel.
- **Path_provider:** Sirve para encontrar rutas en el dispositivo, lo utilizo con Excel.
- **Nfc_manager:** Sirve para acceder al NFC nativo de flutter, lo utilizo para leer la etiqueta del NFC.
- **Flutter_Keyboard_visibility:** Para detectar el teclado de la pantalla en dispositivos móviles.

- **BACK**

- **spring-boot-starter-web:** Expone controllers REST; Spring MVC, Jackson para JSON, etc...
- **spring-boot-starter-data-jpa:** Proporciona JPA e Hibernate para la creación de datos en la base de datos.
- **spring-boot-starter-security:** Proporciona Spring Security para la autenticación en el API.
- **spring-security-config:** Proporciona configuración de Spring Security más avanzada.
- **io.jsonwebtoken:jwt:** Para crear y validar JWT en la autenticación.
- **org.json:json:** Para parsear y generar objetos JSON.

- **org.springdoc:springdoc-openapi-ui:** Para la generación de la documentación automática de Swagger.
- **mysql:mysql-connector-java:** Driver JBCD para conectar con MySQL
- **org.projectlombok:lombok:** Para ahorrar código con los getter, setter, constructores, etc.
- **org.projectlombok:lombok-mapstruct-binding:** Para generar mapeos entre DTOs y entidades.

6.4.2.2 Librerías desarrolladas por el usuario

No desarrolle ninguna librería.

6.4.2.3 Arquitectura

- Mi arquitectura en general es **Orientada a Objetos** tanto en el FRONT como en el BACK.
- En la interfaz del usuario es **Reactivo**, ya que cada interacción del usuario es un evento.
- En las peticiones HTTP es **Reactivo** también, porque cada petición se trata como un evento.

6.4.3 Cuestiones de diseño e implementación reseñables.

Algunas partes del código que son razonables para destacar serían las siguientes.

```
/// Inicia la lectura de NFC, comprueba que sea NFC y que el texto sea FICHAJE
void _empezar() async {
  if (_escaneando) return;
  _escaneando = true;
  // Si el nfc no esta activado salta un mensaje
  if (!await NfcManager.instance.isAvailable()) {
    setState(() => _status = 'NFC no disponible');
    return;
  }

  // Inicia la antena NFC
  NfcManager.instance.startSession(onDiscovered: (tag) async {
    String resultado = 'Error inesperado';
    bool ok = false;

    try {
      final ndef = Ndef.from(tag);
      if (ndef == null) throw Exception('No es NDEF');
      await ndef.read();

      // Recorre los registros y extrae el texto del NFC
      for (var record in ndef.cachedMessage!.records) {
        if (record.typeNameFormat == NdefTypeFormat.nfcWellknown &&
            record.type.length == 1 &&
            record.type[0] == 0x54) {
          final payload = record.payload;
          final statusByte = payload[0];
          final langlen = statusByte & 0x3F;
          final text = utf8.decode(payload.sublist(1 + langLen));

          if (text == 'FICHAJE') {
            ok = true;
            resultado = ';Tarjeta válida!';
          } else {
            resultado = 'Tarjeta no válida: $text';
          }
          break;
        }
      }
    } catch (e) {
      resultado = 'Error lectura: $e';
    } finally {
      // Cierra el nfc
      await NfcManager.instance.stopSession();
      _escaneando = false;
    }
  });
}
```



```
// Pinta las primeras celdas para la cabecera con el estilo y auto ajustandose
final cabecera = ['Nº', 'Nombre', 'Apellidos', 'Email', 'Nº Ausencias', 'Grupo'];
for (var col = 0; col < cabecera.length; col++) {
    final celda = hoja.cell(CellIndex.indexByColumnRow(columnIndex: col, rowIndex: 0));
    celda.value = cabecera[col];
    celda.cellStyle = estilоСabecera;
    hoja.setColAutoFit(col);
}

// Recorre los usuarios y va rellenando las celdas se pinta la siguiente fila
// por el indice del for
for (var i = 0; i < usuariosFiltrados.length; i++) {
    final usuario = usuariosFiltrados[i];
    final grupoNombre = grupoNombres[usuario.grupoId]!;
    final ausenciasNoAceptadas = ausencias
        .where((a) => a.usuario.id == usuario.id && a.estado != EstadoAusencia.aceptada)
        .length;
    final fila = [
        i + 1,
        usuario.nombre,
        (usuario.apellido2 != null && usuario.apellido2!.isNotEmpty)
            ? '${usuario.apellido1} ${usuario.apellido2}'
            : usuario.apellido1,
        usuario.email,
        ausenciasNoAceptadas,
        grupoNombre,
    ];
    // esto es lo que pinta los valores en si(el nombre, apellido...)
    for (var col = 0; col < fila.length; col++) {
        final celda = hoja.cell(
            CellIndex.indexByColumnRow(columnIndex: col, rowIndex: i + 1),
        );
        celda.value = fila[col];
        celda.cellStyle = estilоСuerpo;
    }
}
// Convierte todo lo que hemos pintado en un array de bytes
final bytes = excel.encode();
if (bytes == null) throw Exception('Error codificando Excel');
// Crea el fichero y escribe los bytes
File(path)
    ..createSync(recursive: true)
    ..writeAsBytesSync(bytes);

scaffold.showSnackBar(
    SnackBar(content: Text('✓ Excel guardado en: $path')),
);
```

- Exportar uno o todos los grupos a Excel (Codigo de exportar todos los grupos). En lib/widgets/admin/exportar_excel.dart

- Al terminar la duración del token, te hace logearte. Aquí hay varios procesos para tener en cuenta. Para un usuario que se logea normal, la duración del token es de **9 horas** (para que, durante la jornada laboral, no se tenga que loggear) y para el usuario que se logea con la opción de guardar sesión, token se prolonga **por 30 días**. Cuando ese token expira, directamente te lleva al login.

Comprueba si da error **401** o **403**. Este método lo llevan todos los servicios que llaman a la API. En /lib/utils/api_client.

```
/// Comprueba si da el error 401 o 403 para la expiración del token, en ese caso, se elimina
/// el token y el usuario del sharedPreferences, cierra sesión y se redirige al login
static Future<void> _error401(http.Response resp) async {
    if (resp.statusCode == 401 || resp.statusCode == 403) {
        final prefs = await SharedPreferences.getInstance();
        await prefs.remove('token');
        await prefs.remove('usuario');

        final ctx = navigatorKey.currentContext;
        if (ctx != null) {
            Provider.of<AuthProvider>(ctx, listen: false).cerrarSesion();
        }

        navigatorKey.currentState
            ?.pushNamedAndRemoveUntil('/', (_)> false);
    }
}
```

Dos tipos de expiración, **default** y **recuérdame**. En caso de que en el login den a la opción de guardar sesión. En src/resources/application.properties.

```
# Expiración por defecto 9 horas 32400000 || prueba 2 min 120000
jwt.expiration-ms= 32400000
# Recuérdame, 30 días
jwt.expiration-recuerdame-ms=2592000000
```

Crea el **token** según el **booleano**. En src/java/security/JwtTokenProvider.

```
public String createToken(String username, List<String> roles, boolean recuerdame) { 1 usage ± AdrianAlonso35
    long now = System.currentTimeMillis();
    long exp = now + (recuerdame ? recuerdaExpiracion : defaultExpiracion);
    Claims claims = Jwts.claims().setSubject(username);
    claims.put("roles", roles);

    return Jwts.builder()
        .setSubject(username)
        .claim("authorities", roles)
        .setIssuedAt(new Date(now))
        .setExpiration(new Date(exp))
        .signWith(SignatureAlgorithm.HS256, secretKey.getBytes())
        .compact();
}
```

6.5 Pruebas.

6.5.1 Seguridad

- En cuanto seguridad utilice **Spring Security + JWT** para gestionar la autenticación.
- Tengo rol **Empleado**, y rol **jefe**. Pero en los endpoints no lo contemplo. Ya que no tiene posibilidad de que un empleado llegue a hacer una funcionalidad de un jefe.

6.5.2 Pruebas

La siguiente tabla sirve para entender los resultados que pueden ocurrir por las peticiones del postman.

Objetivo	Acción	Resultado	Estado
Login valido	{email: adrian@educa.jcyl.es , contraseña: correcta} hacia /usuarios/login	200 OK + Token	OK
Login invalido	{email: adrian@educa.jcyl.es , contraseña: error} hacia /usuarios/login	401 Unauthorized	OK
Crear grupo con Token	{nombre: 1 B, usuariosIds: [1,2,3,4]} hacia /grupos/crearGrupo con Token	201 + grupo	OK
Crear grupo sin token o invalido	{nombre: 1 B, usuariosIds: [1,2,3,4]} hacia /grupos/crearGrupo sin Token	403 Forbidden	OK
SQL Injection en login	{email: '' OR '1'='1"; pass: x}	401 Unauthorized	OK



Capturas de postman, de las peticiones de arriba: Login valido

The screenshot shows a POST request to `http://localhost:9999/usuarios/login`. The Body tab is selected, showing a JSON payload:

```
1
2   "email": "adrian@educa.jcyl.es",
3   "contrasena": "adrian",
4   "recuerdame": false
5
```

The response status is 200 OK, with a token returned in the response body:

```
1
2   "token": "eyJhbGciOiJIUzI1NiJ9.
eyJzdWIiOiJhZnJpYm5AZWR1Y2EuamN5bC51cyIsImFidGhvcmI0aWVzIjpbImplZmUiXSwiaWF0IjoxNzQ3NTExMTcwLC1leHAiOjE3NDc1NDM1
Nz89.jD15tkyzeuoUMphzP_vFpxR2Mxf1b07GA0ZM00TMGdI",
3   "id": 16,
4   "nombre": "Adrian",
5   "apellido1": "Alonso",
6   "apellido2": "Perez",
7   "email": "adrian@educa.jcyl.es",
8   "rol": "jefe",
9   "estado": "activo"
10
```

Login invalido

The screenshot shows a POST request to `http://localhost:9999/usuarios/login`. The Body tab is selected, showing a JSON payload:

```
1
2   "email": "adrian@educa.jcyl.es",
3   "contrasena": "adrian1",
4   "recuerdame": false
5
```

The response status is 401 Unauthorized:

```
1
```



Crear grupo con token

POST <http://localhost:9999/grupos/crearGrupo> Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary JSON

```
1 {  
2   "nombre": "1 B",  
3   "usuariosIds": [1,2,3,4]  
4 }
```

Body Cookies Headers (11) Test Results Status: 201 Created Time: 283 ms Size: 391 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 9,  
3   "nombre": "1 B",  
4   "usuariosIds": []  
5 }
```

Crear grupo sin token o invalido

POST <http://localhost:9999/grupos/crearGrupo> Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary JSON

```
1 {  
2   "nombre": "1 B",  
3   "usuariosIds": [1,2,3,4]  
4 }
```

Body Cookies Headers (10) Test Results Status: 403 Forbidden Time: 8 ms Size: 312 B Save Response

Pretty Raw Preview Visualize Text

```
1
```

SQL Injection en login

POST <http://localhost:9999/usuarios/login> Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary JSON

```
1 {  
2   "email": "'" OR '1'='1",  
3   "contrasena": "adrian",  
4   "recuerdame": false  
5 }
```

Body Cookies Headers (10) Test Results Status: 401 Unauthorized Time: 23 ms Size: 315 B Save Response

Pretty Raw Preview Visualize Text

7 Manuales de usuario

7.1 Manual de usuario

Mi aplicación es multiplataforma, lo que quiere decir que la aplicación se puede utilizar, tanto para móvil como para ordenador. Pero la parte del empleado está pensada para móvil(android), ya que, sino no podrás fichar con NFC y la parte de jefe, está pensada para ordenador(windows), aunque sea completamente compatible los widgets y todo, menos el poder exportar los grupos. Aclarado esto, las pantallas de la parte del empleado serán en vista móvil y las pantallas del jefe serán en vista ordenador.

7.1.1 Paso por paso

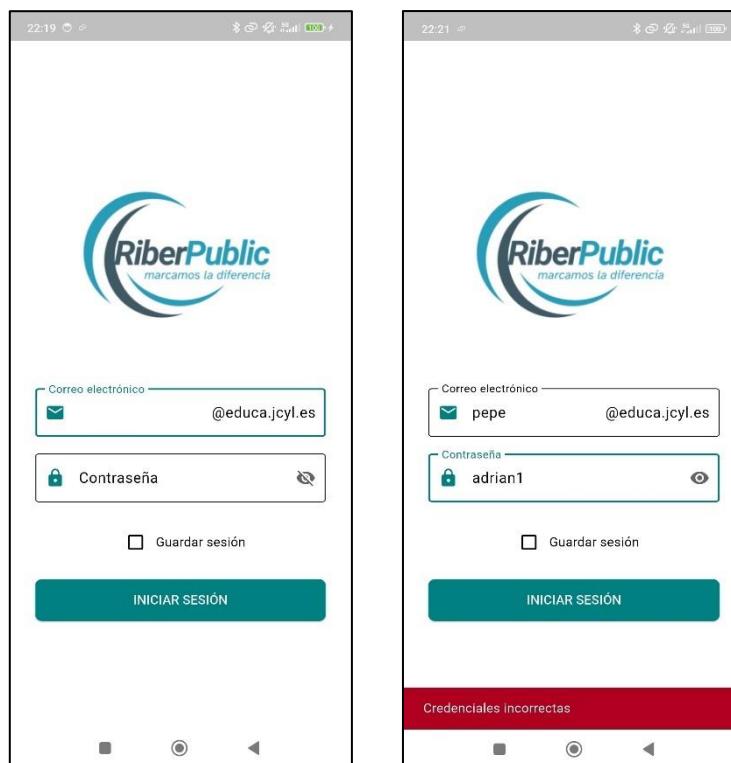
El manual de usuario, lo divido en dos partes, **jefe** y **empleado**. Empecemos por empleado:

- **Login – empleado**

Para el **login**, tienes que introducir el **correo y contraseña**. El correo tiene que ser una cuenta de **educa.jcyl** y previamente la tiene que crear tu jefe.

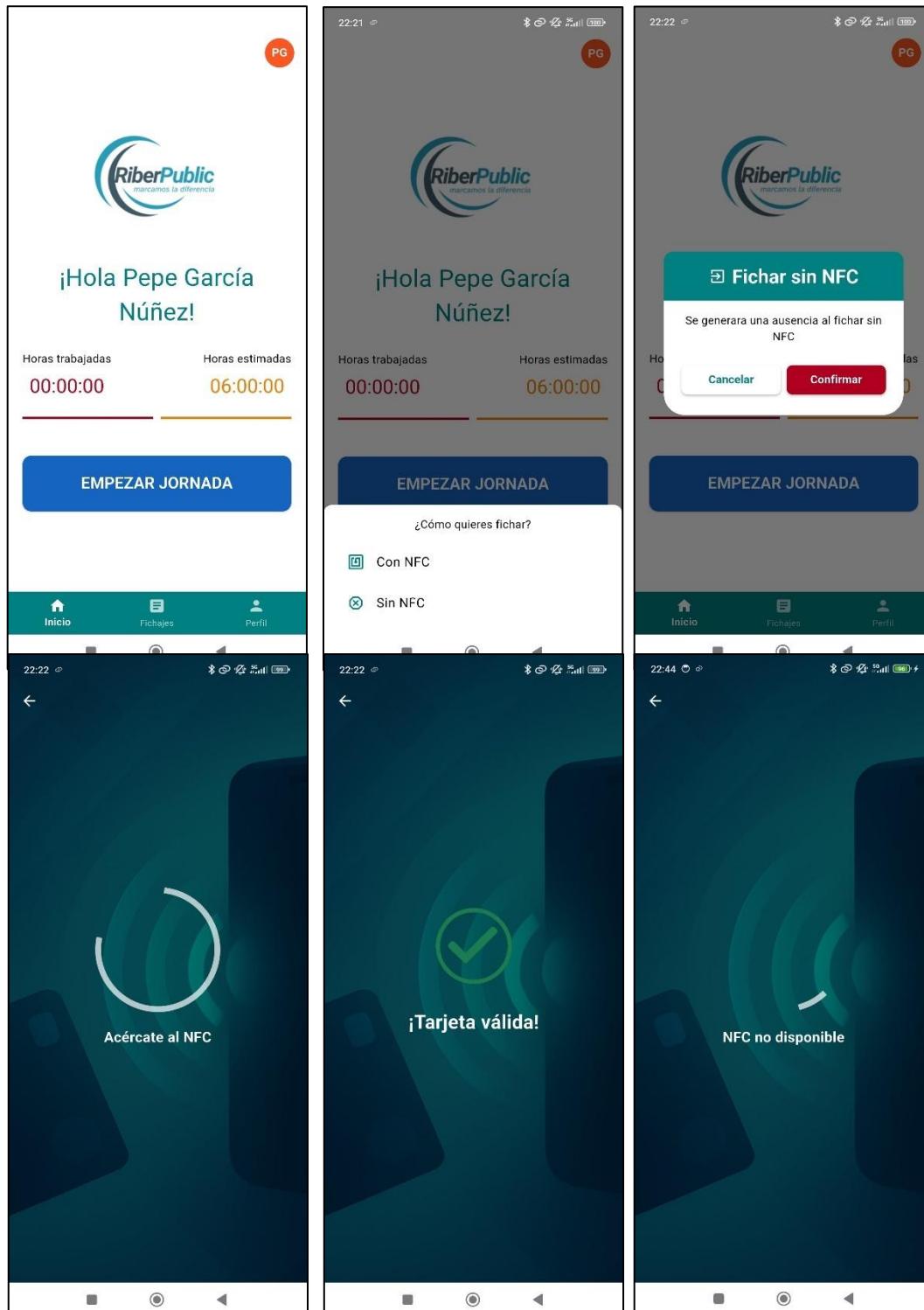
Credenciales empleado: {email: empleado@educa.jcyl.es, contraseña: Empleado1}

- En caso de que **no hayas introducido las credenciales correctamente**, te saldrá una notificación en forma de snackbar para informarte.
- Si has introducido las credenciales correctamente, detectara si **eres jefe o empleado** y te llevara a su pantalla correspondiente.



- **Registrar Ausencia**

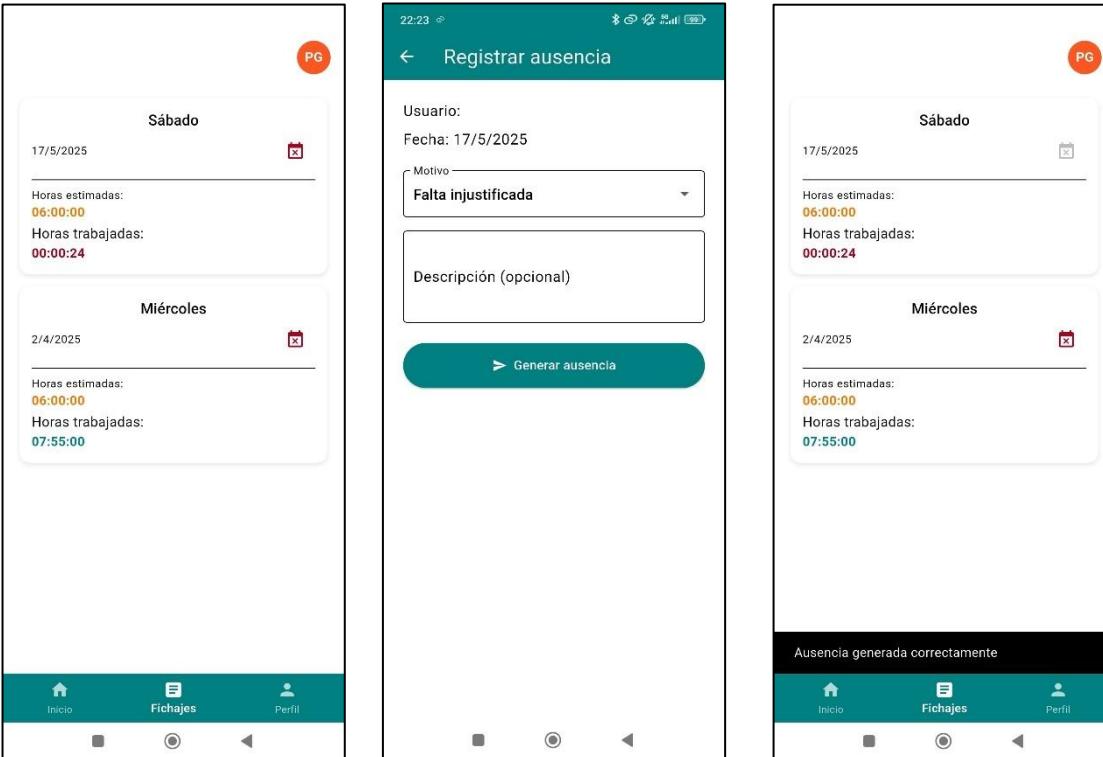
Para registrar una ausencia, tienes dos opciones: **Con NFC** y **sin NFC**. La diferencia entre una y otra es que, si fichas sin NFC tanto la entrada como la salida, se **generara una ausencia**. Al dar a entender que has fichado fuera de tu oficina. Sin NFC esta implementada por dos motivos; por si la tarjeta **no llegara a funcionar** por cualquier motivo, y por si el **dispositivo no admite NFC**.



- **Listado de fichajes**

En el listado de fichajes, aparecen los **fichajes** que tienes hechos con el usuario. El color de las horas trabajadas vendrá dado según si has cumplido las horas estimadas, o no y tienes la posibilidad de **registrar una ausencia**.

- Para **registrar una ausencia**, tienes que dar al icono rojo del calendario que hay en cada fichaje, según cual ausencia quieras registrar.
- Esta funcionalidad esta implementada por si el empleado, **no pudiera completar las horas** por algún motivo y lo quiera comunicar.
- Al generar la ausencia, el icono **se desactiva**, ya que solo se puede generar una ausencia por día.



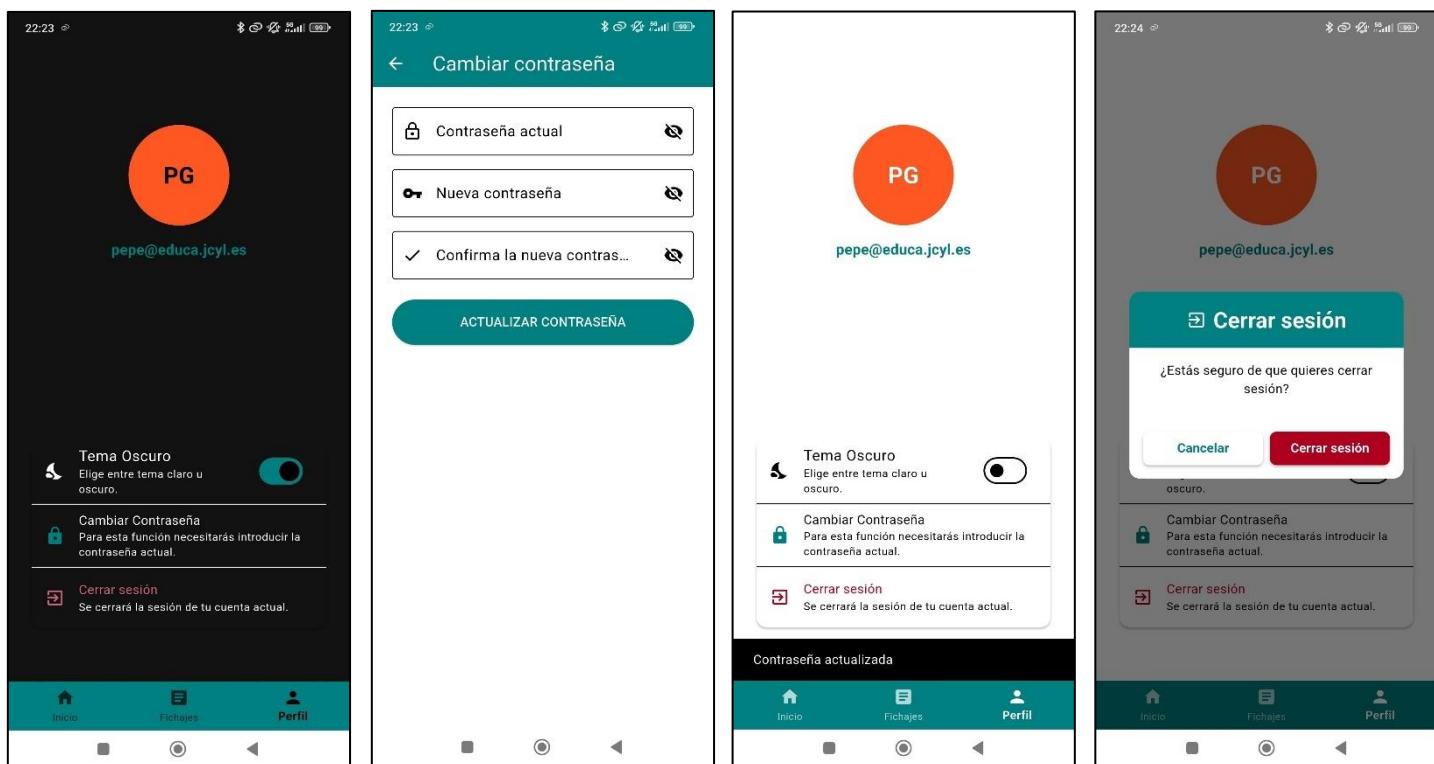
The figure consists of three vertically stacked screenshots of a mobile application interface:

- Screenshot 1 (Left):** Shows a list of attendance records. It displays two entries: "Sábado" (Saturday) on 17/5/2025 and "Miércoles" (Wednesday) on 2/4/2025. Each entry shows estimated hours (06:00:00) and worked hours. The worked hours for Saturday are 00:00:24, and for Wednesday are 07:55:00. Each entry has a red calendar icon in the top right corner.
- Screenshot 2 (Middle):** A modal window titled "Registrar ausencia" (Register absence). It shows the user is "User" and the date is "17/5/2025". A dropdown menu under "Motivo" (Reason) is set to "Falta injustificada" (Unjustified absence). Below it is a text input field labeled "Descripción (opcional)" (Optional description). At the bottom is a teal button with the text "► Generar ausencia" (Generate absence).
- Screenshot 3 (Right):** Shows the result of generating the absence. The "Sábado" entry now shows worked hours of 00:00:24. The "Miércoles" entry now shows worked hours of 07:55:00. The red calendar icon for Saturday is now grayed out, indicating the absence has been registered. At the bottom of the screen, a black bar displays the message "Ausencia generada correctamente" (Absence generated correctly).

- **Perfil**

En el perfil, hay diferentes funcionalidades; **tema oscuro**, **cambiar contraseña** y **cerrar sesión**.

- **Tema oscuro:** Cambia en el provider hecho para el tema a oscuro y en todas las pantallas se cambia el tema.
- **Cambiar contraseña:** Introduces la contraseña actual e introduces la contraseña nueva y la confirmas. Esta contraseña tiene que ser segura, quiere decir. Debe tener 6 o más caracteres, 1 mayúscula y 1 número.
- **Cerrar sesión:** Al confirmar, elimina de sharedPreferences, tanto el usuario como el token almacenado. Y nos lleva al login nuevamente.

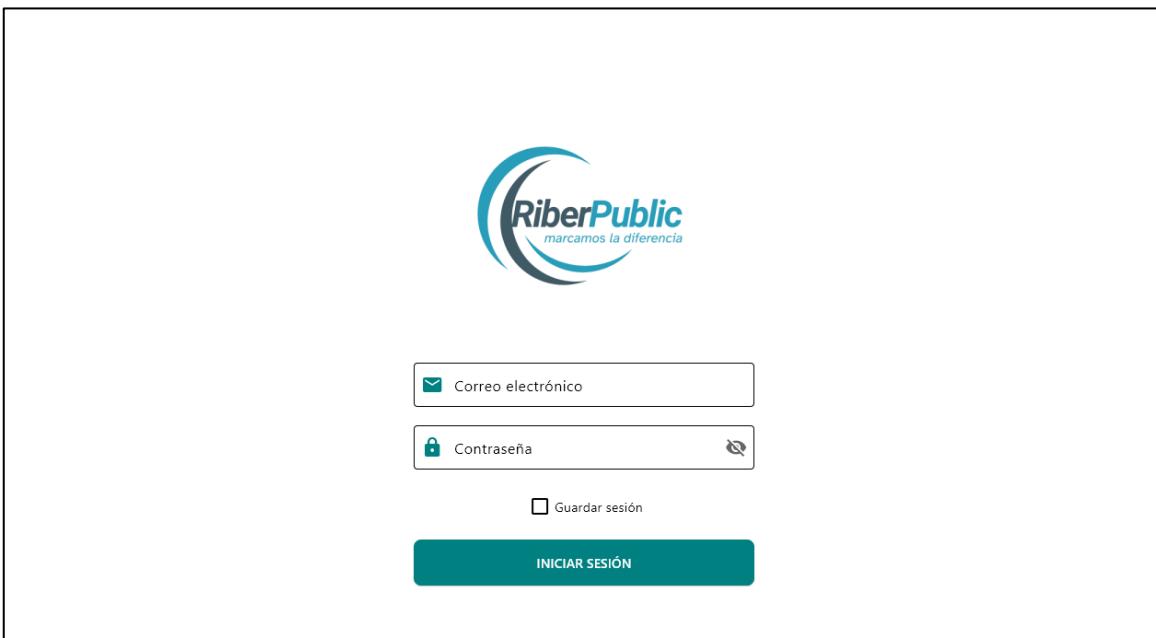


- **Login – Administrador**

El login sería **el mismo** tanto para empleado como para jefe. **Introduces tus credenciales** y la API lo valida, si son correctas detecta si eres jefe o empleado y te lleva a la correspondiente pantalla.

En ese caso **somos administradores**.

Credenciales administrador {email: root@educa.jcyl.es, contraseña: Admin1}



- **Ausencias**

Lo que permite esta pantalla sería: **Generar, editar y ver** las ausencias de los usuarios:

- **Generar las ausencias:** Al dar al botón generar, lo que hace es hacer una llamada a la API (un post) y **crea las ausencias por los fichajes** y tiene unos requisitos.
 - Si un fichaje **no tiene fecha de entrada o de salida**, genera la ausencia con el detalle de “Sin asignar la fecha de entrada/salida”.
 - Si un usuario **ha trabajado menos horas de las que tenía estimadas**, se genera la ausencia con el detalle de “Horas trabajadas menores que las estimadas”.
 - Si el usuario **ficho sin NFC** tanto la salida como la entrada, se genera una ausencia con el detalle de “No se utilizó el NFC para fichar”
 - Si el usuario debía fichar por que su horario lo indicaba, pero **no registro un fichaje**, se genera una ausencia con el detalle de “No registró ningún fichaje”.
- **Editar las ausencias:** Lo que se puede editar de las ausencias, es **el estado**, para identificar si **esa ausencia es justificada o no**.



- **Ver las ausencias:** Se lista todas las ausencias con el nombre del usuario, la fecha y el motivo, a la vez se pueden **filtrar** por el nombre y apellidos o por el estado (aceptada, rechazada, pendiente).

Ausencias

The screenshot shows a list of student absences. Each entry includes the student's initials in a colored circle, their name, date, and reason. A green checkmark indicates accepted absences, while a red X indicates rejected ones. A dropdown menu allows filtering by state: 'Todos' (All), 'Aceptada' (Accepted), or 'Rechazada' (Rejected). A search bar at the top allows searching by name and surname.

Estado	Nombre	Fecha	Motivo
Aceptada	Luis Fernández Ortiz	2025-04-01	enfermedad
Rechazada	Elena García Suárez	2025-04-02	falta_injustificada
Rechazada	Miguel Santos	2025-04-03	vacaciones
Aceptada	Sonia Ruiz Pérez	2025-04-04	permiso
Rechazada	Pepe García Núñez	2025-05-17	falta_injustificada

Buscar por nombre y apellidos

Filtrar por estado

Todos

AA adrian@educa.jcyl.es

Ausencias

Usuarios Grupos Horarios

Cerrar sesión Generar

Ausencia editada:

Ausencias

The screenshot shows the 'Ausencias' (Absences) page with a single entry highlighted. The student's name is Luis Fernández Ortiz, the date is 2025-04-01, and the reason is 'enfermedad'. The 'Estado' (State) dropdown is set to 'aceptada' (Accepted). Below the list, there is a dropdown for 'Justificado' (Justified) with options: 'vacio', 'pendiente', 'aceptada', and 'rechazada'. The 'rechazada' option is selected, and its status is shown as 'No' in a red box. A blue 'Generar' button is visible at the bottom right.

Estado	Nombre	Fecha	Motivo
Aceptada	Luis Fernández Ortiz	2025-04-01	enfermedad

Buscar por nombre y apellidos

Filtrar por estado

Todos

AA adrian@educa.jcyl.es

Ausencias

Usuarios Grupos Horarios

Cambiar estado

aceptada

Justificado: Sí

Elena García Suárez

vacio

pendiente

aceptada

rechazada

Justificado: No

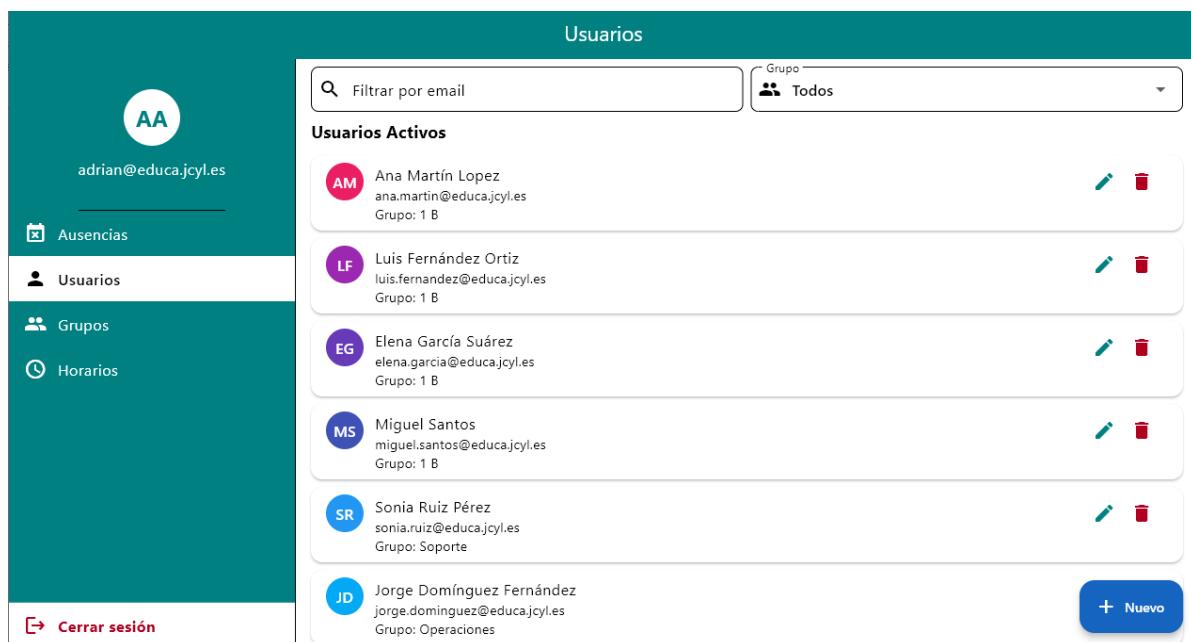
Generar

Estado actualizado

- **Usuarios**

En esta pantalla tendremos la opción de hacer el CRUD de usuarios, es decir: **Listar**, **crear**, **editar** y **eliminar** usuarios.

- **Listar usuarios:** Se **listan todos los usuarios** que hay, tanto activos como inactivos. En ellos se pueden ver el nombre y apellidos, el email y al grupo que pertenecen. Además de la posibilidad de **filtrar los usuarios**, tanto por su email, como por su grupo.
- **Crear usuario:** El crear usuario, esta representado por un Floating Action Button, al pulsarlo se despliega **un dialogo para introducir los datos** del usuario; nombre, apellido1, apellido2, email, contraseña, rol y grupo, se realizan las **validaciones necesarias**, como que el email no se repita o que la contraseña sea una contraseña segura y se crea el usuario.
- **Editar usuario:** Representado por un lápiz en cada usuario, se desplegará **un dialog** como en crear usuario, pero esta vez **con los campos rellenos**, menos la contraseña. Se añade un campo, que sería el **estado del usuario**.
- **Eliminar usuario:** Representado por una basura, al darle sale un dialog para **asegurarte que se eliminara al usuario**. Si confirmas se elimina y si tiene alguna **ausencia o fichaje se eliminará también**.



Usuarios			
	<input type="text" value="Filtrar por email"/>	<input type="button" value="Grupo"/>	<input type="button" value="Todos"/>
Usuarios Activos			
	Ana Martín Lopez ana.martin@educa.jcyl.es Grupo: 1 B		
	Luis Fernández Ortiz luis.fernandez@educa.jcyl.es Grupo: 1 B		
	Elena García Suárez elena.garcia@educa.jcyl.es Grupo: 1 B		
	Miguel Santos miguel.santos@educa.jcyl.es Grupo: 1 B		
	Sonia Ruiz Pérez sonia.ruiz@educa.jcyl.es Grupo: Soporte		
	Jorge Domínguez Fernández jorge.dominguez@educa.jcyl.es Grupo: Operaciones		
+ Nuevo			

**Crear usuario:**

The screenshot shows the 'Crear Usuario' (Create User) dialog box over a list of existing users. The dialog fields are as follows:

- Nombre: [Input field]
- Apellido 1: [Input field]
- Apellido 2 (opcional): [Input field]
- Email: [Input field]
- Contraseña: [Input field with eye icon]
- Rol: [Dropdown menu]
- Grupo: [Dropdown menu]

At the bottom are 'Cancelar' (Cancel) and 'Crear' (Create) buttons.

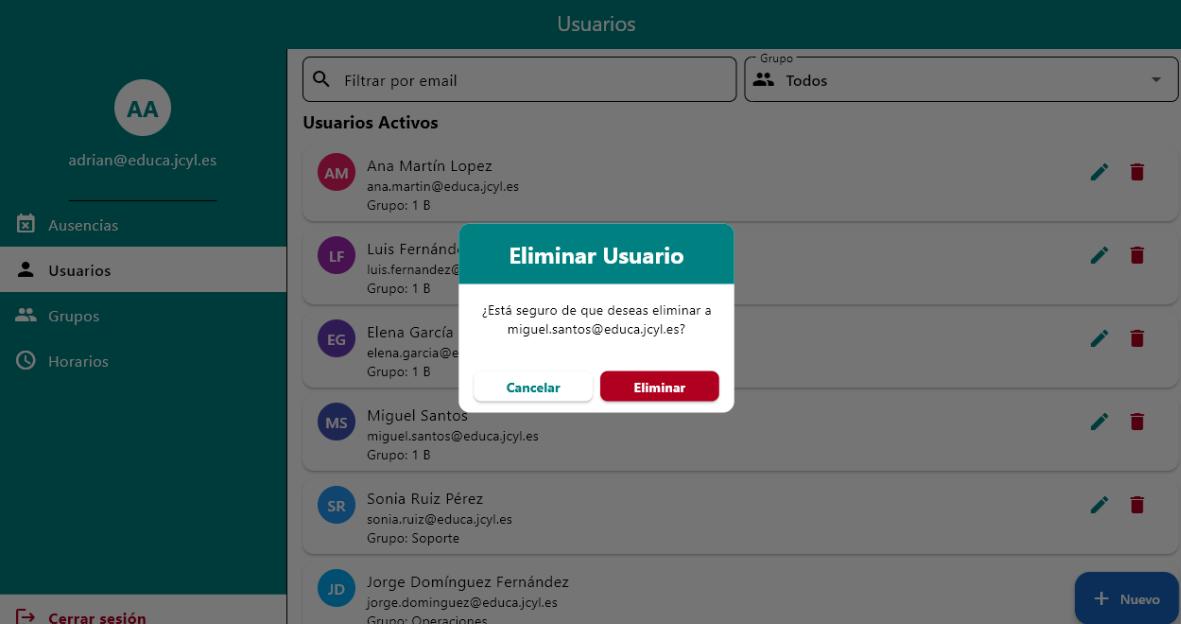
Editar usuario:

The screenshot shows the 'Editar Usuario' (Edit User) dialog box over a list of existing users. The dialog fields are as follows:

- Nombre: Ana
- Apellido 1: Martín
- Apellido 2 (opcional): Lopez
- Email: ana.martin@educa.jcyl.es
- Nueva contraseña (opcional): [Input field with eye icon]
- Rol: empleado
- Grupo: 1 B
- Estado: activo

At the bottom are 'Cancelar' (Cancel) and 'Actualizar' (Update) buttons.

Eliminar usuario:



The screenshot shows the 'Users' section of a web application. On the left, there's a sidebar with icons for 'Ausencias', 'Usuarios', 'Grupos', and 'Horarios'. The 'Usuarios' icon is selected. At the bottom of the sidebar is a 'Cerrar sesión' button. The main area is titled 'Usuarios' and contains a search bar 'Filtrar por email' and a dropdown 'Grupo' set to 'Todos'. Below this is a section titled 'Usuarios Activos' listing several users with their initials, names, emails, and groups. One user, 'Miguel Santos', has a red delete icon next to it. A modal dialog box is overlaid on the screen, titled 'Eliminar Usuario'. It contains the message '¿Está seguro de que deseas eliminar a miguel.santos@educa.jcyl.es?' with 'Cancelar' and 'Eliminar' buttons.

- **Grupos**

Como en la pantalla de usuarios, en grupos se gestiona el CRUD del mismo: **Listar, crear, editar y eliminar**. Tiene dos **detalles a destacar**, que más adelante mostrare.

- **Listar grupos:** Se muestran los grupos que existen y al abrir el desplegable, se muestran **los usuarios que tiene ese grupo**. También se puede **filtrar** por grupos.
- **Crear grupo:** Al crear un grupo te sale un dialogo, para introducir el nombre del grupo nuevo y los usuarios que quieras que se asignen a este grupo. En este campo de usuarios, utilizo **un TypeAheadField**, que es un **text con sugerencias** de lo que puedes buscar. Es decir, en este caso lo que quiero sugerir son usuarios para introducir al grupo, pues al darle clic sale la lista de usuarios que existen, además puedes **filtrar** escribiendo el email del usuario.
- **Editar grupo:** El editar grupo, saltaría un dialogo para **editar** el nombre del grupo y eliminar o añadir usuarios a ese grupo igual que en crear grupo. Hay una excepción que sería que el **grupo por defecto “Sin Asignar” no se le puede editar el nombre**.
- **Eliminar grupo:** Aparece un dialogo para confirmar que quieras eliminar el grupo y al **eliminarlo**, los usuarios que pertenecen a ese grupo **se asignan al grupo por defecto “Sin Asignar”** para que no se eliminan. Además, **este grupo** por defecto **no se puede eliminar**.
- **Exportar a Excel:** Al lado del filtrado, hay un botón, exportar. Este botón lo que hace es exportar o un grupo que seleccionas o todos los grupos, excepto los que no tengan usuarios y el grupo Sin Asignar.



Listado de grupos:

The screenshot shows a user interface for managing groups. On the left, a sidebar menu includes 'Ausencias', 'Usuarios', 'Grupos' (selected), and 'Horarios'. The main area is titled 'Grupos' and contains a table with the following data:

Nombre del grupo	Opciones
R Recursos Humanos	
I IT	
O Operaciones	
C Calidad	
L Logística	
S Sin Asignar	
1 1 B	^
1. AM Ana Martín Lopez ana.martin@educa.jcyl.es Ausencias: 0	
2. MS Miguel Santos miguel.santos@educa.jcyl.es	

A 'Nuevo' (New) button is located at the bottom right of the list.

Crear grupo (con el typeAheadFile):

The screenshot shows a 'Crear Grupo' (Create Group) dialog box overlaid on the 'Grupos' page. The dialog has fields for 'Nombre del grupo' (Group name) and 'Añadir usuario' (Add user). A type-ahead search dropdown lists users:

- ana.martin@educa.jcyl.es
1 B
- elena.garcia@educa.jcyl.es
Ventas
- miguel.santos@educa.jcyl.es
1 B
- sonia.ruiz@educa.jcyl.es
Soporte
- jorge.dominguez@educa.jcyl.es
Operaciones
- raul.ibanez@educa.jcyl.es
Ventas

**Editar Grupo:**

The screenshot shows a modal window titled "Editar Grupo" (Edit Group) overlaid on a list of groups. The modal contains fields for "Nombre del grupo" (Group name) set to "1 B" and "Añadir usuario" (Add user) with a search bar. Below these are sections for "Usuario actuales" (Current users) showing "ana.martin@educa.jcyl.es" and "miguel.santos@educa.jcyl.es", each with a delete button. At the bottom are "Cancelar" (Cancel) and "Actualizar" (Update) buttons. The background shows a list of groups: Recursos Humanos, IT, Operaciones, Calidad, Logística, and Sin Asignar. Under "Recursos Humanos", there is a group named "1 B" with two members: Ana Martín López and Miguel Santos.

Eliminar Grupo:

The screenshot shows a modal window titled "Eliminar Grupo" (Delete Group) with the message "¿Está seguro de que deseas eliminar \"1 B\"?" (Are you sure you want to delete "1 B"?). At the bottom are "Cancelar" (Cancel) and "Eliminar" (Delete) buttons. The background shows the same group list as the previous screenshot, with the "1 B" group selected for deletion.

**Exportar a Excel:**

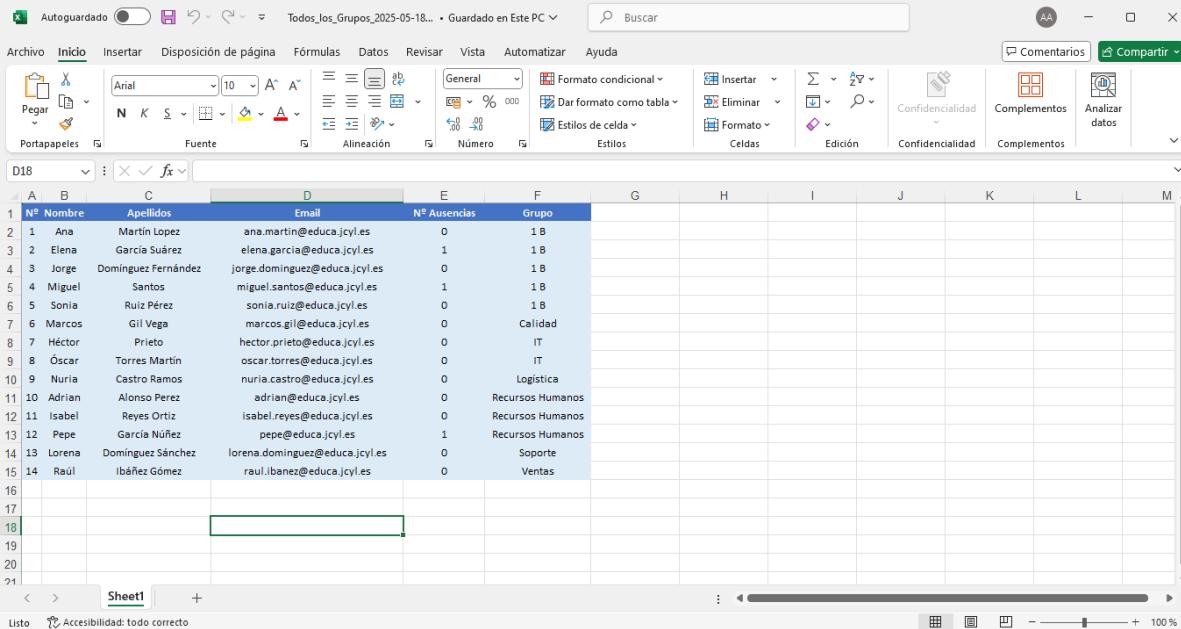
The screenshot shows the 'Grupos' application interface. On the left, there's a sidebar with icons for Ausencias, Usuarios, Grupos, Horarios, and Cerrar sesión. The main area displays a list of groups: Recursos Humanos, IT, Operaciones, Calidad, Logística, and Sin Asignar. A modal dialog titled 'Exportar grupos' is open in the center, prompting the user to 'Elige un grupo' (Choose a group) and providing 'Exportar todos' (Export all) and 'Cancelar' (Cancel) buttons. At the top right of the main area is an 'Exportar' (Export) button.

Resultado de exportar el grupo 1 B:

The screenshot shows an Excel spreadsheet titled '1 B_2025-05-18...'. The data is organized into columns: N°, Nombre, Apellidos, Email, and N° Ausencias. The rows contain the following data:

Nº	Nombre	Apellidos	Email	Nº Ausencias
1	Ana	Martín Lopez	ana.martin@educa.jcyl.es	0
2	Elena	García Suárez	elena.garcia@educa.jcyl.es	1
3	Miguel	Santos	miguel.santos@educa.jcyl.es	1
4	Sonia	Ruiz Pérez	sonia.ruiz@educa.jcyl.es	0
5	Jorge	Domínguez Fernández	jorge.dominguez@educa.jcyl.es	0

Resultado de exportar todos los grupos (Ordenado por grupo, nombre y apellido1):



Nº	Nombre	Apellidos	Email	Nº Ausencias	Grupo
1	Ana	Martín Lopez	ana.martin@educa.jcyl.es	0	1 B
2	Elena	García Suárez	elena.garcia@educa.jcyl.es	1	1 B
3	Jorge	Domínguez Fernández	jorge.dominguez@educa.jcyl.es	0	1 B
4	Miguel	Santos	miguel.santos@educa.jcyl.es	1	1 B
5	Sonia	Ruiz Pérez	sonia.ruiz@educa.jcyl.es	0	1 B
6	Marcos	Gil Vega	marcos.gil@educa.jcyl.es	0	Calidad
7	Héctor	Prieto	hector.prieto@educa.jcyl.es	0	IT
8	Óscar	Torres Martín	oscar.torres@educa.jcyl.es	0	IT
9	Nuria	Castro Ramos	nuria.castro@educa.jcyl.es	0	Logística
10	Adrián	Alonso Pérez	adrian@educa.jcyl.es	0	Recursos Humanos
11	Isabel	Reyes Ortiz	isabel.reyes@educa.jcyl.es	0	Recursos Humanos
12	Pepe	García Núñez	pepe@educa.jcyl.es	1	Recursos Humanos
13	Lorena	Domínguez Sánchez	lorena.dominguez@educa.jcyl.es	0	Soporte
14	Raúl	Ibáñez Gómez	raul.ibanez@educa.jcyl.es	0	Ventas

• Horarios

En horarios se puede hacer el CRUD: **Listar, crear, editar y eliminar** horarios.

- **Listar horarios:** Aquí se listan los grupos y al abrir el desplegable, **aparecen los días que están asociados al grupo**. Se puede **filtrar** por grupo y por día.
- **Crear horario:** Aparece un dialogo, para **crear el horario**, hay que seleccionar el grupo, el día, la hora de entrada y la hora de salida. Hay que tener en cuenta dos cosas aquí, **no se puede crear dos días iguales en un mismo grupo**, es decir no puede haber dos lunes en 1 B y **la hora de entrada no puede ser posterior a la hora de salida**.
- **Editar horario:** Aparece un dialogo similar al de crear horario, con los campos rellenos, para seleccionar las horas de entrada y salida, como en crear horario utilizo un **datePicker**.
- **Eliminar horario:** Para eliminar un horario sale un dialogo para **confirmar** que lo quieras **eliminar**.

**Listado de horarios:**

Horarios

Filtrar por Grupo: Todos Filtrar por Día: Todos

R Recursos Humanos

O Operaciones

- 1. L Lunes: 08:30 – 16:30
- 2. M Martes: 08:30 – 16:30
- 3. M Miércoles: 08:30 – 16:30
- 4. J Jueves: 08:30 – 16:30
- 5. V Viernes: 08:30 – 16:30

C Calidad

L Logística

+ Nuevo

Ausencias Usuarios Grupos Horarios

adrian@educa.jcyl.es

Cerrar sesión

Crear horario:

Horarios

Filtrar por Grupo: Todos Filtrar por Día: Todos

R Recursos Humanos

O Operaciones

- 1. L Lunes: 08:30 – 16:30
- 2. M Martes: 08:30 – 16:30
- 3. M Miércoles: 08:30 – 16:30
- 4. J Jueves: 08:30 – 16:30
- 5. V Viernes: 08:30 – 16:30

C Calidad

L Logística

Crear Horario

Grupo: [dropdown]

Día: [dropdown]

Entrada: 9:00 Cambiar

Salida: 15:00 Cambiar

Cancelar Crear

Ausencias Usuarios Grupos Horarios

adrian@educa.jcyl.es

Cerrar sesión

**Editar horario:**

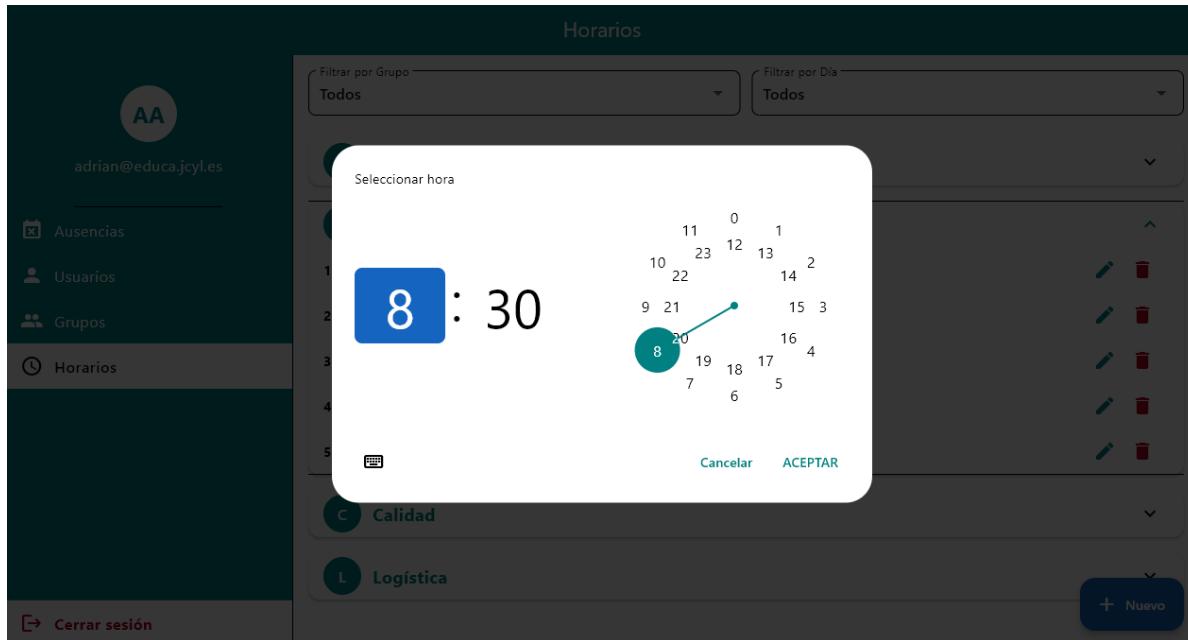
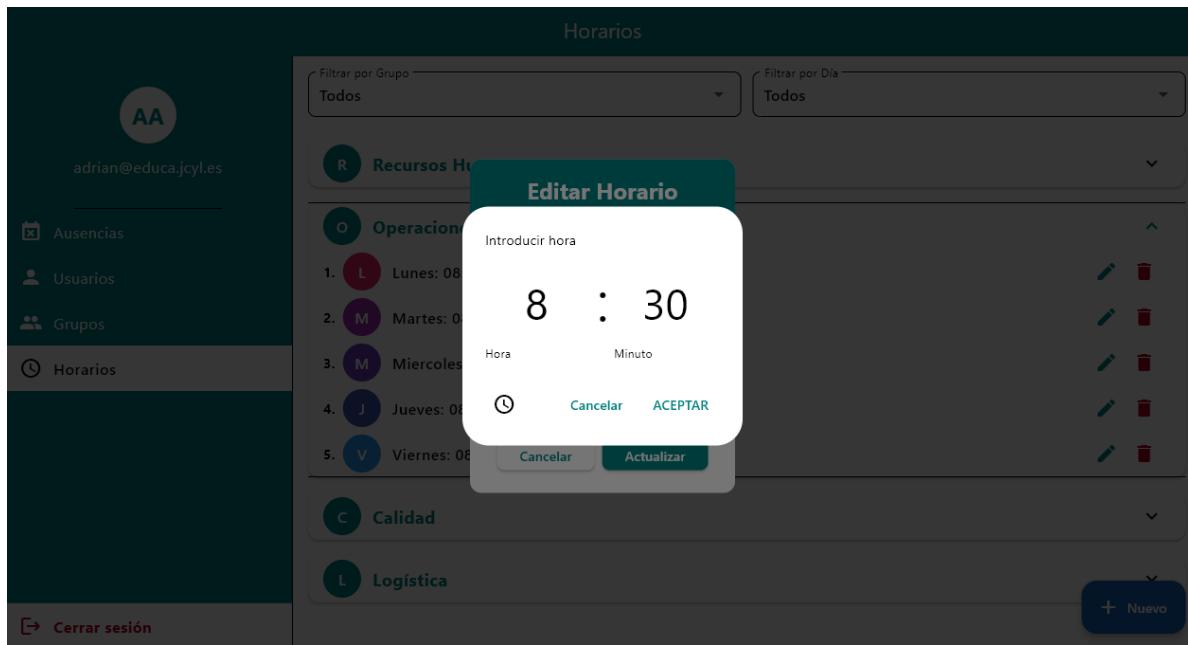
The screenshot shows a modal dialog titled "Editar Horario" (Edit Schedule) overlaid on a list of schedules. The dialog has fields for "Grupo" (Operaciones) and "Día" (Lunes). Below the dialog, the main interface shows a list of five scheduled entries, each with edit and delete icons.

Posición	Día	Entrada	Salida
1.	Lunes	08:30	16:30
2.	Martes	08:30	16:30
3.	Miercoles	08:30	16:30
4.	Jueves	08:30	16:30
5.	Viernes	08:30	16:30

Eliminar horario:

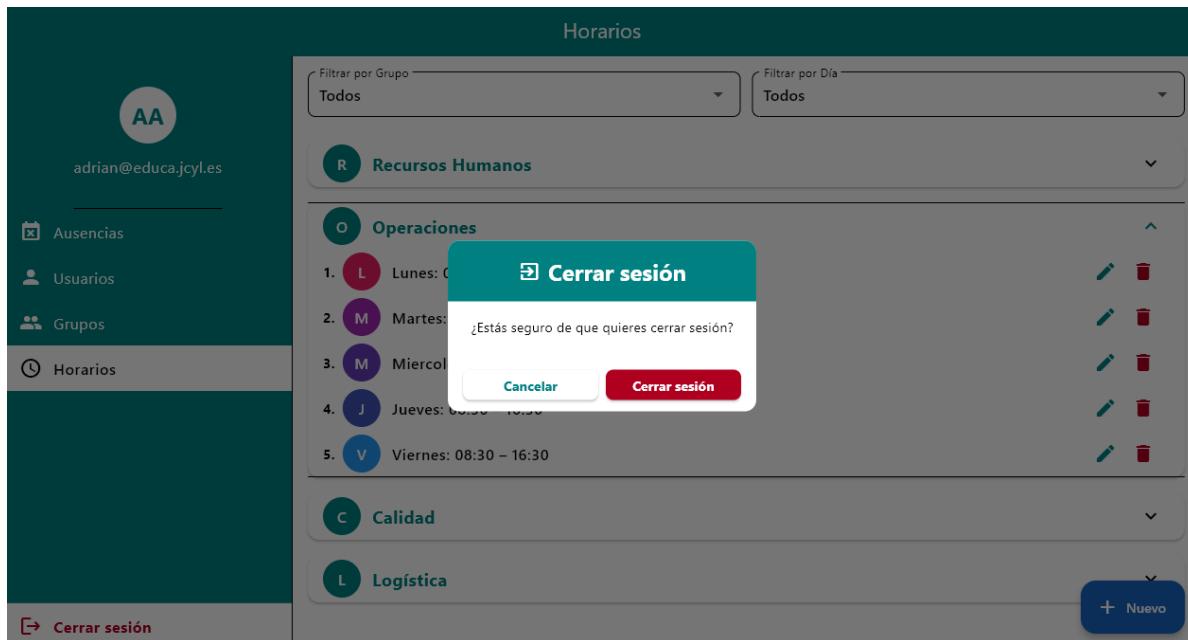
The screenshot shows a modal dialog titled "Eliminar Horario" (Delete Schedule) with a confirmation message: "¿Está seguro de que deseas eliminar el \"Día.lunes\"?" (Are you sure you want to delete "Day.Monday"?). It has "Cancelar" (Cancel) and "Eliminar" (Delete) buttons.

Posición	Día	Entrada	Salida
1.	Lunes	08:30	16:30
2.	Martes	08:30	16:30
3.	Miercoles	08:30	16:30
4.	Jueves	08:30	16:30
5.	Viernes	08:30 – 16:30	

**DatePicker opción 1:****DatePicker opción 2:**

• Cerrar Sesión

Al cerrar sesión, se elimina del sharedPreferences tanto el usuario, como el token.



7.2 Manual de instalación

Para el manual de instalación, os enseñare los requisitos mínimos para poder ejecutar la aplicación y el proceso para instalarlo.

7.2.1 Requisitos mínimos

- Windows 10 (administración)
- Android 9.0 (empleado con NFC)
- Git 2.4
- IntelliJ IDEA 2023.1
- Java 21
- Visual Studio Code
- Dart/Flutter
- MySQL 8.0

7.3 Manual de instalación

1- Clonar los repositorios de la APP y del API:

Ir a la terminal y pegar los siguientes comandos respectivamente en VSC y IntelliJ:
Visual Studio Code: “git clone <https://github.com/adree3/RiberRepublicFichajeApp.git>”.
IntelliJ IDEA: “git clone <https://github.com/adree3/RiberRepublicFichajeApi.git>”.

2- Desplegar la Base de datos en mi caso en MySQL Workbench:

En MySQL Workbench abrir el script de la base de datos, alojado en el API y desplegarla.

3- Desplegar el API y la APP:

*En caso de que se esté desplegando para la aplicación móvil, en la APP, en /lib/utils/api_config.dart seguir los pasos indicados.

Cuando estén cargadas las aplicaciones, ya nos saldrá el login.

4- Credenciales para el login:

Al ser una aplicación privada debes tener una cuenta previamente, usuarios predefinidos:

- Administrador {email: root@educa.jcyl.es, contraseña: Admin1}.
- Empleado {email: empleado@educa.jcyl.es, contraseña: Empleado1}

8 Conclusiones y posibles ampliaciones

8.1 Conclusiones

Tanto mi cliente como yo, estamos muy contentos y satisfechos de que haya podido cumplir los objetivos propuestos. Crear una aplicación multiplataforma, con sistema de fichaje NFC, con la exportación de la información de los usuarios centrada en ausencias no justificadas.

Además de haber podido desarrollar personalmente, tanto el trato con el cliente como mi conocimiento de programación.

8.2 Posibles mejoras

Tengo en mente algunas mejoras o tareas a realizar en el futuro, como:

- Implementar **servicio de correo** para el cambio de contraseña.
- Implementar **huella dactilar** para asegurar que no hay suplantación a la hora de fichar.
- Poder **personalizar** mas los perfiles, incluyendo imagen de perfil, etc.
- **Internalización** de la aplicación, es decir, poder cambiar el idioma.

9 Bibliografía

9.1 Frontend

- [Documentación Flutter.](#)
- [Buscador con sugerencias.](#)
- [Desplegable hamburguesa responsive.](#)
- [NFC con flutter.](#)
- [Exportar datos a Excel.](#)
- [Providers.](#)

9.2 Backend

- [Springboot.](#)
- [Spring security + JWT.](#)

9.3 Otros

- [Inteligencia artificial.](#)



10 Anexos

Demostración de gran parte del código de mi proyecto.

10.1 Flutter

No añado ni los widgets, ni las screens, ya que son muy extensos.

10.1.1 Modelos



```
/// Modelo Grupo
class Grupo {
    /// Atributos
    final int? id;
    final String nombre;
    final int faltasTotales;
    final List<Usuario> usuarios;
    final List<Horario> horarios;

    /// Constructor
    Grupo({
        required this.id,
        required this.nombre,
        required this.faltasTotales,
        required this.usuarios,
        required this.horarios,
    });

    /// Convierte el json al modelo Grupo
    factory Grupo.fromJson(Map<String, dynamic> json) {
        final gid = json['id'] as int;

        // si viene el objeto completo lo mapeamos y sino una lista vacia
        final usuarios = (json['usuarios'] as List<dynamic>?)?.map((u) => Usuario.fromJson(u as Map<String, dynamic>))
            .toList() ??
        <Usuario>[];

        // lo mismo para horarios
        final horarios = (json['horarios'] as List<dynamic>?)?.map((h) => Horario.fromJsonWithGroup(h as Map<String, dynamic>, gid))
            .toList() ??
        <Horario>[];

        return Grupo(
            id: gid,
            nombre: json['nombre'] as String,
            faltasTotales: json['faltasTotales'] as int? ?? 0,
            usuarios: usuarios,
            horarios: horarios,
        );
    }

    /// Convierte del modelo Grupo a Json
    Map<String, dynamic> toJson() {
        return {
            'id': id,
            'nombre': nombre,
            'faltasTotales': faltasTotales,
            'usuarios': usuarios.map((u) => u.toJson()).toList(),
            'horarios': horarios.map((h) => h.toJson()).toList(),
        };
    }
}
```



```
/// Modelo Ausencia
class Ausencia {
  /// Atributos
  int? id;
  DateTime fecha;
  Motivo motivo;
  EstadoAusencia estado;
  bool justificada;
  String? detalles;
  DateTime tiempoRegistrado;
  Usuario usuario;
  /// Constructor
  Ausencia({
    this.id,
    required this.fecha,
    this.motivo = Motivo.falta_injustificada,
    this.estado = EstadoAusencia.vacio,
    required this.justificada,
    this.detalles,
    required this.tiempoRegistrado,
    required this.usuario,
  });
  /// Convierte el json al modelo Ausencia
  factory Ausencia.fromJson(Map<String, dynamic> json) {
    final usuarioField = json['usuario'];
    Usuario usuario;
    if (usuarioField is int) {
      usuario = Usuario(
        id: usuarioField,
        nombre: '',
        apellido1: '',
        apellido2: null,
        email: '',
        contrasena: null,
        rol: Rol.empleado,
        estado: Estado.activo,
        grupoId: null,
      );
    } else if (usuarioField is Map<String, dynamic>) {
      usuario = Usuario.fromJson(usuarioField);
    } else {
      throw Exception('Campo "usuario" en ausencia inválido: $usuarioField');
    }
    return Ausencia(
      id: json['id'] as int,
      fecha: DateTime.parse(json['fecha']),
      motivo: Motivo.values.firstWhere(
        (e) => e.toString().split('.').last == json['motivo'],
        orElse: () => Motivo.falta_injustificada,
      ),
      estado: EstadoAusencia.values.firstWhere(
        (e) => e.toString().split('.').last == json['estado'],
        orElse: () => EstadoAusencia.vacio,
      ),
      justificada: json['justificada'] as bool,
      detalles: json['detalles'] as String?,
      tiempoRegistrado: DateTime.parse(json['tiempoRegistrado']),
      usuario: usuario,
    );
  }
  /// Convierte del modelo Ausencia a Json
  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'fecha': fecha.toIso8601String(),
      'motivo': motivo.toString().split('.').last,
      'estado': estado.toString().split('.').last,
      'justificada': justificada,
      'detalles': detalles,
      'tiempoRegistrado': tiempoRegistrado.toIso8601String(),
      'usuario': usuario.toJson(),
    };
  }
}
enum Motivo {
  retraso,
  permiso,
  vacaciones,
  enfermedad,
  falta_injustificada,
  otro
}
enum EstadoAusencia{
  vacio,
  pendiente,
  aceptada,
  rechazada
}
```



```
/// Modelo Fichaje
class Fichaje {
    /// Atributos
    final int? id;
    final DateTime? fechaHoraEntrada;
    final DateTime? fechaHoraSalida;
    final String? ubicacion;
    final bool nfcUsado;
    final Usuario usuario;

    /// Constructor
    Fichaje({
        this.id,
        this.fechaHoraEntrada,
        this.fechaHoraSalida,
        this.ubicacion,
        required this.nfcUsado,
        required this.usuario,
    });

    /// Convierte el json al modelo Fichaje
    factory Fichaje.fromJson(Map<String, dynamic> json) {
        final usuarioField = json['usuario'];
        late final Usuario usuario;
        if (usuarioField is int) {
            usuario = Usuario(
                id: usuarioField,
                nombre: '',
                apellido1: '',
                apellido2: null,
                email: '',
                contrasena: null,
                rol: Rol.empleado,
                estado: Estado.activo,
                grupoId: null,
            );
        } else if (usuarioField is Map<String, dynamic>) {
            usuario = Usuario.fromJson(usuarioField);
        } else {
            throw Exception('Campo "usuario" inválido en Fichaje: $usuarioField');
        }
        return Fichaje(
            id: json['id'] as int?,
            fechaHoraEntrada: json['fechaHoraEntrada'] != null
                ? DateTime.parse(json['fechaHoraEntrada'] as String)
                : null,
            fechaHoraSalida: json['fechaHoraSalida'] != null
                ? DateTime.parse(json['fechaHoraSalida'] as String)
                : null,
            ubicacion: json['ubicacion'] as String?,
            nfcUsado: json['nfcUsado'] as bool? ?? false,
            usuario: usuario,
        );
    }

    /// Convierte del modelo Fichaje a Json
    Map<String, dynamic> toJson() {
        return {
            'id': id,
            'fechaHoraEntrada': fechaHoraEntrada?.toISOString(),
            'fechaHoraSalida': fechaHoraSalida?.toISOString(),
            'ubicacion': ubicacion,
            'nfcUsado': nfcUsado,
            'usuario': usuario.toJson(),
        };
    }
}
```



```
/// Modelo Horario
class Horario {
    /// Atributos
    final int id;
    final Dia dia;
    final String horaEntrada;
    final String horaSalida;
    final int grupoId;

    /// Constructor
    Horario({
        required this.id,
        required this.dia,
        required this.horaEntrada,
        required this.horaSalida,
        required this.grupoId,
    });

    /// Convierte el json al modelo Horario
    factory Horario.fromJsonWithGroup(Map<String, dynamic> json, int grupoId) {
        return Horario(
            id: json['id'] as int,
            dia: Dia.values.firstWhere((e) => e.name == json['dia']),
            horaEntrada: json['horaEntrada'] as String,
            horaSalida: json['horaSalida'] as String,
            grupoId: grupoId,
        );
    }

    /// Convierte el json al modelo Horario
    factory Horario.fromJson(Map<String, dynamic> json) {
        // El grupo puede venir o como int o como {int, }
        final dynamic rawGrupo = json['grupo'] ?? json['grupoId'];
        late final int gid;
        if (rawGrupo is int) {
            gid = rawGrupo;
        } else if (rawGrupo is Map<String, dynamic>) {
            gid = rawGrupo['id'] as int;
        } else {
            throw FormatException('No aparece el grupoId en el JSON de Horario');
        }

        final diaEnum = Dia.values.firstWhere(
            (e) => e.name == (json['dia'] as String),
            orElse: () => throw FormatException('Día inválido: ${json['dia']}'),
        );

        return Horario(
            id: json['id'] as int,
            dia: diaEnum,
            horaEntrada: json['horaEntrada'] as String,
            horaSalida: json['horaSalida'] as String,
            grupoId: gid,
        );
    }

    /// Convierte del modelo Horario a Json
    Map<String, dynamic> toJson() {
        return {
            'id': id,
            'dia': dia.name,
            'horaEntrada': horaEntrada,
            'horaSalida': horaSalida,
            'grupo': {
                'id': grupoId,
            },
        };
    }
}

enum Dia {
    lunes,
    martes,
    miercoles,
    jueves,
    viernes,
    sabado,
    domingo
}
```



```
import 'package:ribet_republic_fichaje_app/model/usuario.dart';

/// Dto de usuario para login con Token
class LoginResponse {
    /// Atributos
    final String token;
    final int id;
    final String nombre;
    final String apellido1;
    final String? apellido2;
    final String email;
    final Rol rol;
    final Estado estado;
    final int? grupoId;

    ///Constructor
    LoginResponse({
        required this.token,
        required this.id,
        required this.nombre,
        required this.apellido1,
        this.apellido2,
        required this.email,
        required this.rol,
        required this.estado,
        this.grupoId,
    });

    /// Convierte el json al modelo LoginResponse
    factory LoginResponse.fromJson(Map<String,dynamic> json) {
        return LoginResponse(
            token : json['token'] as String,
            id : json['id'] as int,
            nombre : json['nombre'] as String,
            apellido1 : json['apellido1'] as String,
            apellido2 : json['apellido2'] as String?,
            email : json['email'] as String,
            rol : Rol.values.firstWhere((e)=>e.name==json['rol']),
            estado : Estado.values.firstWhere((e)=>e.name==json['estado']),
            grupoId : json['grupoId'] as int?,
        );
    }
    /// Convierte de LoginResponse a Json
    Map<String, dynamic> toJson() {
        return {
            'token' : token,
            'id' : id,
            'nombre' : nombre,
            'apellido1' : apellido1,
            if (apellido2 != null) 'apellido2': apellido2,
            'email' : email,
            'rol' : rol.name,
            'estado' : estado.name,
            if (grupoId != null) 'grupoId': grupoId,
        };
    }
}
```



```
/// Modelo Usuario
class Usuario {
    /// Atributos
    final int id;
    final String nombre;
    final String apellido1;
    final String? apellido2;
    final String email;
    final String? contrasena;
    final Rol rol;
    final Estado estado;
    final int? grupoId;

    /// Constructor
    Usuario({
        required this.id,
        required this.nombre,
        required this.apellido1,
        this.apellido2,
        required this.email,
        this.contrasena,
        required this.rol,
        required this.estado,
        this.grupoId,
    });

    /// Convierte el json al modelo Usuario
    factory Usuario.fromJson(Map<String, dynamic> json) {
        final grupoField = json['grupo'];
        int? grupoId;
        if (grupoField is int) {
            grupoId = grupoField;
        } else if (grupoField is Map<String, dynamic>) {
            grupoId = grupoField['id'] as int?;
        }

        return Usuario(
            id: json['id'] as int,
            nombre: json['nombre'] as String,
            apellido1: json['apellido1'] as String,
            apellido2: json['apellido2'] as String?,
            email: json['email'] as String,
            contrasena: json['contrasena'] as String?,
            rol: Rol.values.firstWhere((e) => e.name == json['rol']),
            estado: Estado.values.firstWhere((e) => e.name == json['estado']),
            grupoId: grupoId,
        );
    }

    /// Convierte del modelo Usuario a Json
    Map<String, dynamic> toJson() {
        return {
            'id': id,
            'nombre': nombre,
            'apellido1': apellido1,
            'apellido2': apellido2,
            'email': email,
            'contrasena': contrasena,
            'rol': rol.name,
            'estado': estado.name,
            'grupo': grupoId != null ? {'id': grupoId} : null,
        };
    }
}

enum Rol {
    empleado,
    jefe,
}
enum Estado {
    activo,
    inactivo,
}
```



10.1.2 Providers

```
/// Provider utilizado para el tema oscuro o claro
class ThemeProvider extends ChangeNotifier {
    ThemeMode _mode = ThemeMode.light;
    ThemeProvider() {
        _cargarPreferencias();
    }

    ThemeMode get mode => _mode;

    bool get esOscuro => _mode == ThemeMode.dark;

    /// Al llamarlo cambia de oscuro a claro y al revés
    Future<void> toggle() async {
        _mode = esOscuro ? ThemeMode.light : ThemeMode.dark;
        notifyListeners();
        final prefs = await SharedPreferences.getInstance();
        await prefs.setBool('darkMode', esOscuro);
    }

    /// Carga las preferencias al sharedPreferences y notifica
    Future<void> _cargarPreferencias() async {
        final preferencia = await SharedPreferences.getInstance();
        final oscuro = preferencia.getBool('darkMode') ?? false;
        _mode = oscuro ? ThemeMode.dark : ThemeMode.light;
        notifyListeners();
    }
}
```

```
/// Provider utilizado para guardar el usuario loggeado
class AuthProvider with ChangeNotifier {
    LoginResponse? _usuario;

    LoginResponse? get usuario => _usuario;

    bool get estaLogueado => _usuario != null;

    /// Añades el usuario logeado
    void setUsuario(LoginResponse? usuario) {
        _usuario = usuario;
        notifyListeners();
    }

    /// Cierras sesión
    void cerrarSesion() {
        _usuario = null;
        notifyListeners();
    }
}
```



10.1.3 Servicios

```
/// Conecta el API de ausencias con la aplicacion de flutter
class AusenciaService {
    static String get baseUrl => ApiConfig.baseUrl + '/ausencias';

    /// Obtiene todas las ausencias
    /// GET /ausencias/
    Future<List<Ausencia>> getAusencias() async {
        final response = await ApiClient.get(Uri.parse('$baseUrl/'));

        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            final List<dynamic> jsonList = jsonDecode(utf8Body);
            return jsonList.map((e) => Ausencia.fromJson(e)).toList();
        } else {
            throw Exception('Error al obtener las ausencias');
        }
    }

    /// Comprueba si la ausencia existe
    /// GET /ausencias/{idUsuario}/existe
    static Future<bool> existeAusencia(int idUsuario, DateTime fecha) async {
        final iso = fecha.toIso8601String().split('T').first;
        final url = '$baseUrl/$idUsuario/existe?fecha=$iso';
        final resp = await ApiClient.get(Uri.parse(url));
        if (resp.statusCode == 200) {
            return jsonDecode(resp.body) as bool;
        }
        throw Exception('Error comprobando ausencia (${resp.statusCode})');
    }

    /// Crea una nueva ausencia segun el id del usuario
    /// POST /ausencias/nuevaAusencia/{idUsuario}
    static Future<Ausencia> crearAusencia({required int idUsuario, required DateTime fecha, required Motivo motivo, String? detalles}) async {
        final response = await ApiClient.post(
            Uri.parse('$baseUrl/nuevaAusencia?idUsuario=$idUsuario'),
            body: jsonEncode({
                'fecha': fecha.toIso8601String(),
                'motivo': motivo.toString().split('.').last,
                'detalles': detalles,
            }),
        );
        if (response.statusCode == 200 || response.statusCode == 201) {
            return Ausencia.fromJson(jsonDecode(response.body));
        }
        throw Exception('Error al justificar ausencia (${response.statusCode}): ${response.body}');
    }

    /// Edita la ausencia y dice si es justificada o no segun el estado de la ausencia
    /// POST /ausencias/editarAusencia/{idAusencia}
    static Future<Ausencia> actualizarAusencia({required int idAusencia, required EstadoAusencia estado, String? detalles}) async {
        final response = await ApiClient.put(
            Uri.parse('$baseUrl/editarAusencia/$idAusencia'),
            body: jsonEncode({
                'estado': estado.toString().split('.').last,
                if (detalles != null) 'detalles': detalles,
            }),
        );
        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            return Ausencia.fromJson(jsonDecode(utf8Body));
        } else {
            throw Exception('Error al actualizar ausencia (${response.statusCode})');
        }
    }

    /// Genera todas las ausencias posibles a partir de los fichajes
    /// POST /ausencias/generarAusencias
    static Future<void> generarAusencias() async {
        final resp = await ApiClient.post(
            Uri.parse('$baseUrl/generarAusencias'),
        );
        if (resp.statusCode != 204) {
            throw Exception(
                'Error al generar ausencias (${resp.statusCode}): ${resp.body}',
            );
        }
    }
}
```



```
/// Conecta el API de fichajes con la aplicacion de flutter
class FichajeService {
    static String get _baseUrl => ApiConfig.baseUrl + '/fichajes';

    /// GET /fichajes/usuario/{idUsuario}
    static Future<List<Fichaje>> getFichajesPorUsuario(int idUsuario) async {
        final url = "${_baseUrl}/usuario/$idUsuario";
        final response = await ApiClient.get(Uri.parse(url));
        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            final List<dynamic> jsonList = jsonDecode(utf8Body);
            return jsonList.map((json) => Fichaje.fromJson(json)).toList();
        } else {
            throw Exception("Error al cargar fichajes del usuario");
        }
    }

    /// GET /fichajes/totalHorasHoy/{idUsuario}
    static Future<TotalHorasHoy> getTotalHorasHoy(int idUsuario) async {
        final uri = Uri.parse("${_baseUrl}/totalHorasHoy/$idUsuario");
        final response = await ApiClient.get(uri);
        if (response.statusCode == 200) {
            return TotalHorasHoy.fromJson(jsonDecode(response.body));
        }
        throw Exception('Error al obtener total horas hoy (${response.statusCode})');
    }

    /// POST /fichajes/abrirFichaje/{idUsuario}
    static Future<Fichaje> abrirFichaje(int idUsuario, {required bool nfcUsado, required String ubicacion}) async {
        final response = await ApiClient.post(
            Uri.parse('${_baseUrl}/abrirFichaje/$idUsuario'),
            body: jsonEncode({
                'nfcUsado': nfcUsado,
                'ubicacion': ubicacion,
            })
        );
        if (response.statusCode == 201 || response.statusCode == 200) {
            return Fichaje.fromJson(jsonDecode(response.body));
        }
        throw Exception('Error al abrir fichaje (${response.statusCode})');
    }

    /// PUT /fichajes/cerrarFichaje/{idUsuario}
    static Future<Fichaje> cerrarFichaje({required int idUsuario, required bool nfcUsado}) async {
        final response = await ApiClient.put(
            Uri.parse('${_baseUrl}/cerrarFichaje/$idUsuario?nfcUsado=$nfcUsado'));
        if (response.statusCode == 200) {
            return Fichaje.fromJson(jsonDecode(response.body));
        } else if (response.statusCode == 404) {
            throw Exception('Usuario no encontrado');
        } else if (response.statusCode == 409) {
            throw Exception('No hay jornada abierta hoy');
        }
        throw Exception('Error al cerrar fichaje (${response.statusCode})');
    }
}
```



```
/// Conecta el API de grupos con la aplicacion de flutter
class GrupoService {
    static String get baseUrl => ApiConfig.baseUrl + '/grupos';

    /// Obtiene todos los grupos
    /// GET /grupos/
    Future<List<Grupo>> getGrupos() async {
        final response = await ApiClient.get(Uri.parse('$baseUrl/'));

        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            final List<dynamic> lista = jsonDecode(utf8Body);
            return lista.map((e) => Grupo.fromJson(e)).toList();
        } else {
            throw Exception('Error al obtener los grupos');
        }
    }

    /// POST /grupos/crearGrupo
    static Future<Grupo> crearGrupo({required String nombre, required List<int> usuariosIds}) async {
        final response = await ApiClient.post(
            Uri.parse('$baseUrl/crearGrupo'),
            body: jsonEncode({
                'nombre': nombre,
                'usuariosIds': usuariosIds,
            }),
        );
        if (response.statusCode == 201) {
            return Grupo.fromJson(jsonDecode(response.body));
        }
        throw Exception('Error creando grupo (${response.statusCode})');
    }

    /// Actualiza un grupo y si se han asignado o desasignado usuarios al grupo también se actualiza
    /// PUT /grupos/editarGrupo/{id}
    static Future<Grupo> actualizarGrupo({required int id, required String nombre, required List<int> usuariosIds}) async {
        final response = await ApiClient.put(
            Uri.parse('$baseUrl/editarGrupo/$id'),
            body: jsonEncode({
                'nombre': nombre,
                'usuariosIds': usuariosIds,
            }),
        );
        if (response.statusCode == 200) {
            return Grupo.fromJson(jsonDecode(response.body));
        }
        throw Exception('Error actualizando grupo (${response.statusCode})');
    }

    /// Eliminar el grupo recibido y re asigna los usuarios a "Sin Asingar"
    /// DELETE /grupos/eliminarGrupo/{id}
    static Future<bool> eliminarGrupo(int id) async {
        final response = await ApiClient.delete(
            Uri.parse('$baseUrl/eliminarGrupo/$id'));
        if (response.statusCode == 204) {
            return true;
        }
        throw Exception('Error borrando grupo (${response.statusCode})');
    }
}
```



```
/// Conecta el API de horarios con la aplicacion de flutter
class HorarioService {
    static String get baseUrl => ApiConfig.baseUrl + '/horarios';

    /// Obtiene todos los horarios
    /// GET /horarios/
    static Future<List<Horario>> getHorarios() async {
        final response = await ApiClient.get(Uri.parse('$baseUrl'));
        if (response.statusCode == 200) {
            final List<dynamic> data = jsonDecode(response.body) as List<dynamic>;
            return data
                .map((json) => Horario.fromJson(json as Map<String, dynamic>))
                .toList();
        } else {
            throw Exception(
                'Error obteniendo horarios (${response.statusCode}): ${response.body}');
        }
    }

    /// Obtiene una lista de horarios por un grupo
    /// GET /horarios/{idgrupo}/horarios
    Future<List<Horario>> getHorariosPorGrupo(int idGrupo) async {
        final url = Uri.parse('$baseUrl/$idGrupo/horarios');
        final response = await ApiClient.get(url);

        if (response.statusCode != 200) {
            throw Exception('Error al obtener los horarios del grupo $idGrupo');
        }
        final List<dynamic> decoded = jsonDecode(response.body);
        return decoded
            .map((e) => Horario.fromJsonWithGroup(e as Map<String, dynamic>, idGrupo))
            .toList();
    }

    /// Crea un horario, con los datos recibidos y se le asigna el grupo indicado
    /// POST /horarios/nuevoHorario
    static Future<Horario> crearHorario({required int groupId, required String dia, required String horaEntrada, required String horaSalida}) async {
        final response = await ApiClient.post(
            Uri.parse('$baseUrl/nuevoHorario?idGrupo=$groupId'),
            body: jsonEncode({
                'dia': dia,
                'horaEntrada': horaEntrada,
                'horaSalida': horaSalida,
            }));
        if (response.statusCode == 201) {
            final Map<String, dynamic> data = jsonDecode(response.body);
            return Horario.fromJson(data);
        } else {
            throw Exception(
                'Error creando horario: ${response.statusCode} ${response.body}');
        }
    }

    /// Edita un horario, por el id, y los datos a editar
    /// PUT /horarios/editarHorario/{id}
    static Future<Horario> editarHorario({ required int id, required String dia, required String horaEntrada, required String horaSalida, required int groupId}) async {
        final response = await ApiClient.put(
            Uri.parse('$baseUrl/editarHorario/$id'),
            body: jsonEncode({
                'dia': dia,
                'horaEntrada': horaEntrada,
                'horaSalida': horaSalida,
                'groupId': groupId,
            }));
        if (response.statusCode == 200) {
            final Map<String, dynamic> json = jsonDecode(response.body);
            return Horario.fromJson(json);
        } else {
            throw Exception(
                'Error al editar horario (${response.statusCode}): ${response.body}');
        }
    }

    /// Elimina el horario por el id recibido
    /// DELETE /horarios/eliminarHorario/{id}
    static Future<bool> eliminarHorario(int id) async {
        final response = await ApiClient.delete(Uri.parse('$baseUrl/eliminarHorario/$id'));
        if (response.statusCode == 204) {
            return true;
        } else if (response.statusCode == 404) {
            throw Exception('Horario no encontrado (404)');
        } else {
            throw Exception(
                'Error al eliminar horario (${response.statusCode}): ${response.body}');
        }
    }
}
```



```
/// Conecta el API de usuarios con la aplicacion de flutter
class UsuarioService {
    static String get baseUrl => ApiConfig.baseUrl + '/usuarios';

    /// Obtiene todos los usuarios
    /// GET /usuarios/
    Future<List<Usuario>> getUsuarios() async {
        final response = await ApiClient.get(Uri.parse('$baseUrl/'));
        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            final List<dynamic> lista = jsonDecode(utf8Body);
            return lista.map((json) => Usuario.fromJson(json).toList());
        } else {
            throw Exception('Error al obtener usuarios');
        }
    }

    /// Obtiene los usuarios activos
    /// GET /usuarios/activos
    Future<List<Usuario>> getUsuariosActivos() async {
        final response = await ApiClient.get(Uri.parse('$baseUrl/activos'));
        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            final List<dynamic> lista = jsonDecode(utf8Body);
            return lista.map((json) => Usuario.fromJson(json).toList());
        } else {
            throw Exception('Error al obtener usuarios');
        }
    }

    /// Obtiene los horarios de hoy de un usuario
    /// GET /usuarios/{idUsuario}/horarioHoy
    static Future<HorarioHoy> getHorarioDeHoy(int idUsuario) async {
        final response = await ApiClient.get(Uri.parse('$baseUrl/${idUsuario}/horarioHoy'));

        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            final Map<String, dynamic> data = json.decode(utf8Body);
            return HorarioHoy.fromJson(data);
        } else {
            throw Exception('Error al obtener el horario de hoy');
        }
    }

    /// Comprueba si un email ya está registrado
    /// GET /usuarios/existe
    Future<bool> emailEnUso(String email) async {
        final response = await ApiClient.get(Uri.parse('$baseUrl/existe?email=$email'));
        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            return jsonDecode(utf8Body) as bool;
        } else {
            throw Exception('Error comprobando email (${response.statusCode})');
        }
    }

    /// Crea un usuario
    /// POST /usuarios/nuevoUsuario/{idGrupo}
    Future<void> crearUsuario(Usuario usuario, int idGrupo) async {
        final response = await ApiClient.post(
            Uri.parse('$baseUrl/nuevoUsuario?idGrupo=${idGrupo}'),
            body: jsonEncode(usuario.toJson()),
        );

        if (response.statusCode != 201) {
            throw Exception('Error al crear usuario: ${response.body}');
        }
    }

    /// Edita la contraseña a un usuario
    /// PUT /usuarios/{idUsuario}/cambiarContrasena
    static Future<void> cambiarContrasena(int idUsuario, String contrasenaActual, String nuevaContrasena)
async {
        final response = await ApiClient.put(
            Uri.parse('$baseUrl/${idUsuario}/cambiarContrasena'),
            body: jsonEncode({
                'contrasenaActual': contrasenaActual,
                'nuevaContrasena' : nuevaContrasena,
            }),
        );
        if (response.statusCode != 200) {
            throw Exception('Error al cambiar contraseña (${response.statusCode}): ${response.body}');
        }
    }

    /// Edita un usuario
    /// PUT /usuarios/editarUsuario/{id}/update?{idGrupo}={idGrupo}
    static Future<Usuario> editarUsuario(int idUsuario, Map<String, dynamic> usuarioEditado, int idGrupo)
async {
        final response = await ApiClient.put(
            Uri.parse('$baseUrl/editarUsuario/${idUsuario}?idGrupo=${idGrupo}'),
            body: jsonEncode(usuarioEditado),
        );
        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            return Usuario.fromJson(jsonDecode(utf8Body));
        } else {
            throw Exception(
                'Error al actualizar usuario (${response.statusCode}): ${response.body}',
            );
        }
    }

    /// Elimina un usuario
    /// DELETE /usuarios/eliminarUsuario/{id}
    Future<void> eliminarUsuario(int idUsuario) async {
        final response = await ApiClient.delete(Uri.parse('$baseUrl/eliminarUsuario/${idUsuario}'));

        if (response.statusCode != 204) {
            if (response.statusCode == 404) {
                throw Exception('Usuario no encontrado');
            }
            throw Exception(
                'Error al eliminar usuario (${response.statusCode}): ${response.body}',
            );
        }
    }
}
```



```
/// Conecta el API de ausencias con la aplicacion de flutter
class AuthService {
    static String get baseUrl => ApiConfig.baseUrl;

    /// Comprueba si el usuario y contraseña son correctos, y guarda el token
    static Future<LoginResponse?> login(String email, String contrasena, bool recuerdame) async {
        final response = await http.post(
            Uri.parse('$baseUrl/usuarios/login'),
            headers: {'Content-Type': 'application/json'},
            body: jsonEncode({
                'email': email,
                'contrasena': contrasena,
                'recuerdame' : recuerdame
            }),
        );
        if (response.statusCode == 200) {
            final utf8Body = utf8.decode(response.bodyBytes);
            final data = jsonDecode(utf8Body);
            return LoginResponse.fromJson(data);
        } else {
            return null;
        }
    }
}
```

10.1.4 Utils

```

/// Clase para reutilizar código y no tener que añadir el token en cada servicio,
/// se llama al servicio correspondiente y va con el service
class ApiClient {
    /// Guarda los headers y el token por defecto.
    static Future<Map<String, String>> get _defaultHeaders async {
        final preferences = await SharedPreferences.getInstance();
        final token = preferences.getString('token');
        return {
            'Content-Type': 'application/json',
            if (token != null) 'Authorization': 'Bearer $token',
        };
    }

    /// Get para el servicio
    static Future<http.Response> get(Uri uri) async {
        final headers = await _defaultHeaders;
        final response = await http.get(uri, headers: headers);
        await _error401(response);
        return response;
    }

    /// Post para el servicio
    static Future<http.Response> post(Uri uri, {dynamic body}) async {
        final headers = await _defaultHeaders;
        final response = await http.post(uri, headers: headers, body: body);
        await _error401(response);
        return response;
    }

    /// Put para el servicio
    static Future<http.Response> put(Uri uri, {dynamic body}) async {
        final headers = await _defaultHeaders;
        final response = await http.put(uri, headers: headers, body: body);
        await _error401(response);
        return response;
    }

    /// Delete para el servicio
    static Future<http.Response> delete(Uri uri) async {
        final headers = await _defaultHeaders;
        final response = await http.delete(uri, headers: headers);
        await _error401(response);
        return response;
    }

    /// Comprueba si da el error 401 o 403 para la expiración del token, en ese caso, se elimina
    /// el token y el usuario del sharedPreferences, cierra sesión y se redirige al login
    static Future<void> _error401(http.Response resp) async {
        if (resp.statusCode == 401 || resp.statusCode == 403) {
            final prefs = await SharedPreferences.getInstance();
            await prefs.remove('token');
            await prefs.remove('usuario');

            final ctx = navigatorKey.currentContext;
            if (ctx != null) {
                Provider.of<AuthProvider>(ctx, listen: false).cerrarSesion();
            }

            navigatorKey.currentState
                ?.pushNamedAndRemoveUntil('/', (_)> false);
        }
    }
}

```



```
● ● ●

class FichajeUtils {
    /// Filtra los fichajes de la lista que correspondan al día actual.
    static List<Fichaje> filtradosDeHoy(List<Fichaje> fichajes) {
        final ahora = DateTime.now();
        return fichajes.where((f) {
            final entrada = f.fechaHoraEntrada;
            if (entrada == null) return false;
            return entrada.year == ahora.year && entrada.month == ahora.month && entrada.day == ahora.day;
        }).toList();
    }
    /// Calcula el total de los fichajes de hoy cuya horaSalida sea distinto a null
    static Duration calcularFichajesHoy (List<Fichaje> fichajesFiltrados){
        return fichajesFiltrados.fold(Duration.zero, (sum, f) {
            if (f.fechaHoraSalida != null) {
                return sum + f.fechaHoraSalida!.difference(f.fechaHoraEntrada!);
            }
            return sum;
        });
    }
    /// Suma las horas totales por dia
    static Map<DateTime, Duration> sumarHorasPorDia(List<Fichaje> fichajes) {
        final Map<DateTime, Duration> totales = {};
        for (final f in fichajes) {
            if (f.fechaHoraEntrada != null && f.fechaHoraSalida != null) {
                final entrada = f.fechaHoraEntrada!;
                final salida = f.fechaHoraSalida!;
                final dia = DateTime(entrada.year, entrada.month, entrada.day);
                final dur = salida.difference(entrada);
                totales.update(dia, (ant) => ant + dur, ifAbsent: () => dur);
            }
        }
        return totales;
    }
}
```

```
● ● ●

/// Configuracion de las rutas de la API segun la plataforma que sean
class ApiConfig {
    static String get host {
        if (Platform.isAndroid) {
            // En caso de estar ejecutando la aplicacion en un dispositivo Android,
            // se tiene que usar el mismo WIFI tanto para el dispositivo que ejecute
            // el API como el del dispositivo Android y hay que configurar la ip
            // del host segun la del dispositivo que corra el API aqui:
            return '192.168.48.182';
        } else {
            return 'localhost';
        }
    }

    static const String port = '9999';

    static String get baseUrl => 'http://$host:$port';
}
```



10.1.5 Main

```
final GlobalKey<NavigatorState> navigatorKey = GlobalKey<NavigatorState>();  
void main() async{  
    WidgetsFlutterBinding.ensureInitialized();  
  
    final preferences = await SharedPreferences.getInstance();  
    final loginString = preferences.getString('usuario');  
  
    LoginResponse? loginGuardado;  
    if (loginString != null) {  
        loginGuardado = LoginResponse.fromJson(jsonDecode(loginString));  
    }  
    runApp(  
        MultiProvider(  
            providers: [  
                ChangeNotifierProvider(create: (_) => AuthProvider)..setUsuario(loginGuardado),  
                ChangeNotifierProvider(create: (_) => ThemeProvider()),  
            ],  
            child: MyApp(),  
        ),  
    );  
}  
  
class MyApp extends StatelessWidget {  
    const MyApp({super.key});  
    @override  
    Widget build(BuildContext context) {  
        final temaProv = Provider.of<ThemeProvider>(context);  
        final usuarioProv = context.watch<AuthProvider>().usuario;  
  
        final initialRoute = usuarioProv == null  
            ? '/'  
            : (usuarioProv.rol == Rol.jefe ? '/admin_home' : '/home');  
        return MaterialApp(  
            navigatorKey: navigatorKey,  
            debugShowCheckedModeBanner: false,  
            theme: ThemeData(  
                colorScheme: const ColorScheme.light(  
                    primary: Color(0xFF008800),  
                    secondary: Color(0xFFFF5700),  
                    primaryContainer: Color(0xFF1565C0),  
                    secondaryContainer: Colors.grey,  
                    tertiary: Color(0xFFe4d9ce)  
                ),  
                appBarTheme: const AppBarTheme(  
                    backgroundColor: Color(0xFF008800),  
                    foregroundColor: Colors.white,  
                ),  
                bottomNavigationBarTheme: const BottomNavigationBarThemeData(  
                    backgroundColor: Color(0xFF008800),  
                    selectedItemColor: Colors.white,  
                    unselectedItemColor: Colors.white70,  
                ),  
                darkTheme: ThemeData(  
                    colorScheme: const ColorScheme.dark(  
                        primary: Color(0xFF008800),  
                        secondary: Color(0xFFFF5700),  
                        primaryContainer: Color(0xFF1565C0),  
                        secondaryContainer: Colors.grey,  
                        tertiary: Color(0xFF76bcad)  
                    ),  
                    appBarTheme: const AppBarTheme(  
                        backgroundColor: Color(0xFF008800),  
                        foregroundColor: Colors.white,  
                    ),  
                    bottomNavigationBarTheme: const BottomNavigationBarThemeData(  
                        backgroundColor: Color(0xFF008800),  
                        selectedItemColor: Colors.white,  
                        unselectedItemColor: Colors.white70,  
                    ),  
                    themeMode: temaProv.mode,  
                    initialRoute: initialRoute,  
                    routes: {  
                        '/': (_) => const LoginScreen(),  
                        '/home': (_) => const HomeScreen(),  
                        '/admin_home': (_) => const AdminHomeScreen(),  
                    },  
                    locale: const Locale('es', 'ES'),  
                    supportedLocales: const [ Locale('es', 'ES') ],  
                    localizationsDelegates: const [  
                        GlobalMaterialLocalizations.delegate,  
                        GlobalWidgetsLocalizations.delegate,  
                        GlobalCupertinoLocalizations.delegate,  
                    ],  
                );  
    }  
}
```



10.2 Java

No muestro ni los controllers, ni los servicios ya que son muy extensos.

10.2.1 Modelos

```
@Setter  
@Getter  
@Entity  
@Table(name = "grupos")  
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")  
public class Grupo {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer id;  
  
    @Column(nullable = false, length = 50)  
    private String nombre;  
  
    @Column(nullable = false)  
    private int faltasTotales;  
  
    @OneToMany(mappedBy = "grupo", cascade = CascadeType.ALL, orphanRemoval = true)  
    private List<Usuario> usuarios;  
  
    @OneToMany(mappedBy = "grupo", cascade = CascadeType.ALL, orphanRemoval = true)  
    private List<Horario> horarios;  
  
    public Grupo() {  
    }  
  
    public Grupo(Integer id, String nombre, int faltasTotales, List<Usuario> usuarios, List<Horario>  
    horarios) {  
        this.id = id;  
        this.nombre = nombre;  
        this.faltasTotales = faltasTotales;  
        this.usuarios = usuarios;  
        this.horarios = horarios;  
    }  
}
```



```
@Getter  
@Setter  
@Entity  
@Table(name = "ausencias")  
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")  
public class Ausencia {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer id;  
  
    @Column(nullable = false)  
    private LocalDate fecha;  
  
    @Enumerated(EnumType.STRING)  
    @Column(nullable = false)  
    private Motivo motivo = Motivo.falta_injustificada;  
  
    @Enumerated(EnumType.STRING)  
    private Estado estado = Estado.vacio;  
  
    private boolean justificada;  
  
    private String detalles;  
  
    @Column(nullable = false, updatable = false)  
    private LocalDateTime tiempoRegistrado;  
  
    @ManyToOne  
    @JoinColumn(name = "usuario_id", nullable = false)  
    @JsonIdentityReference(alwaysAsId = true)  
    private Usuario usuario;  
  
    public enum Motivo {  
        retraso, permiso, vacaciones, enfermedad, falta_injustificada, otro  
    }  
    public enum Estado {  
        vacio, pendiente, aceptada, rechazada  
    }  
  
    public Ausencia() {  
    }  
  
    public Ausencia(Integer id, LocalDate fecha, Motivo motivo, boolean justificada, String detalles,  
    LocalDateTime tiempoRegistrado, Usuario usuario) {  
        this.id = id;  
        this.fecha = fecha;  
        this.motivo = motivo;  
        this.justificada = justificada;  
        this.detalles = detalles;  
        this.tiempoRegistrado = tiempoRegistrado;  
        this.usuario = usuario;  
    }  
}
```



```
@Setter
@Getter
@Entity
@Table(name = "fichajes")
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Fichaje {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private LocalDateTime fechaHoraEntrada;

    private LocalDateTime fechaHoraSalida;

    private String ubicacion;

    private boolean nfcUsado;

    @ManyToOne
    @JoinColumn(name = "usuario_id", nullable = false)
    @JsonIdentityReference(alwaysAsId = true)
    private Usuario usuario;

    public Fichaje() {
    }

    public Fichaje(Integer id, LocalDateTime fechaHoraEntrada, LocalDateTime fechaHoraSalida, String ubicacion, boolean nfcUsado, Usuario usuario) {
        this.id = id;
        this.fechaHoraEntrada = fechaHoraEntrada;
        this.fechaHoraSalida = fechaHoraSalida;
        this.ubicacion = ubicacion;
        this.nfcUsado = nfcUsado;
        this.usuario = usuario;
    }
}
```



```
@Setter  
@Getter  
@Entity  
@Table(name = "horarios")  
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")  
public class Horario {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer id;  
  
    @Enumerated(EnumType.STRING)  
    @Column(nullable = false)  
    private Dia dia;  
  
    @Column(nullable = false)  
    private LocalTime horaEntrada;  
  
    @Column(nullable = false)  
    private LocalTime horaSalida;  
  
    @ManyToOne  
    @JoinColumn(name = "grupo_id", nullable = false)  
    @JsonIdentityReference(alwaysAsId = true)  
    private Grupo grupo;  
  
    public enum Dia {  
        lunes, martes, miercoles, jueves, viernes, sabado, domingo  
    }  
  
    public Horario() {}  
  
    public Horario(Integer id, Dia dia, LocalTime horaEntrada, LocalTime horaSalida, Grupo grupo) {  
        this.id = id;  
        this.dia = dia;  
        this.horaEntrada = horaEntrada;  
        this.horaSalida = horaSalida;  
        this.grupo = grupo;  
    }  
}
```



```
● ● ●

@Setter
@Getter
@Entity
@Table(name = "usuarios")
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")
public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false, length = 25)
    private String nombre;

    @Column(nullable = false, length = 30)
    private String apellido1;

    @Column(length = 30)
    private String apellido2;

    @Column(nullable = false, unique = true, length = 40)
    private String email;

    @Column(length = 255)
    private String contrasena;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private Rol rol = Rol.empleado;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private Estado estado = Estado.activo;

    @ManyToOne
    @JoinColumn(name = "grupo_id")
    @JsonIdentityReference(alwaysAsId = true)
    private Grupo grupo;

    public enum Rol {
        empleado, jefe
    }

    public enum Estado {
        activo, inactivo
    }

    public Usuario() {
    }

    public Usuario(Integer id, String nombre, String apellido1, String apellido2, String email, String
contrasena, Rol rol, Estado estado, Grupo grupo) {
        this.id = id;
        this.nombre = nombre;
        this.apellido1 = apellido1;
        this.apellido2 = apellido2;
        this.email = email;
        this.contrasena = contrasena;
        this.rol = rol;
        this.estado = estado;
        this.grupo = grupo;
    }
}
```



10.2.2 Repositorios

```
@Repository
public interface HorarioRepository extends JpaRepository<Horario, Integer> {

    List<Horario> findByGrupoIdAndDia(int grupoId, Horario.Dia dia);

    List<Horario> findByGrupoId(int grupoId);

    Optional<Horario> findFirstByGrupoIdAndDia(Integer grupoId, Horario.Dia dia);

    List<Horario> findByDia(Horario.Dia dia);
}
```

```
@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Integer> {
    Usuario findByEmail(String email);

    boolean existsByEmail(String email);

    List<Usuario> findByEstado(Usuario.Estado estado);

    List<Usuario> findByGrupo(Grupo grupo);

    List<Usuario> findByGrupoId(int grupoId);
}
```

```
@Repository
public interface AusenciaRepository extends JpaRepository<Ausencia, Integer> {
    boolean existsByUsuarioIdAndFecha(int usuarioId, LocalDate fecha);
}
```



```
@Repository
public interface FichajeRepository extends JpaRepository<Fichaje, Integer> {
    List<Fichaje> findFichajesByUsuario(Usuario usuario);

    Optional<Fichaje> findFirstByUsuarioAndFechaHoraEntradaBetween(Usuario usuario, LocalDateTime
        inicioHoy, LocalDateTime inicioManana);

    Optional<Fichaje> findFirstByUsuarioAndFechaHoraEntradaBetweenAndFechaHoraSalidaIsNull(Usuario
        usuario, LocalDateTime desde, LocalDateTime hasta);

    List<Fichaje> findAllByUsuarioAndFechaHoraEntradaBetween(Usuario usuario, LocalDateTime inicioHoy,
        LocalDateTime inicioManana);

    boolean existsByUsuarioIdAndFechaHoraEntradaBetween(int usuarioId, LocalDateTime inicioDia,
        LocalDateTime finDia);
}
```

```
@Repository
public interface GrupoRepository extends JpaRepository<Grupo, Integer> {
    Optional<Grupo> findByNombre(String nombre);
}
```



10.2.3 Documentación

```
  @Configuration
  public class SwaggerConfig {

    /**
     * Configuración de Swagger sobre los metadatos del API
     */
    @Bean
    public OpenAPI customOpenAPI() {
        return new OpenAPI()
            .components(new Components())
            .info(new Info().title("API RiberPublicFichaje")
                .description("API de la aplicación de RiberPublicFichaje")
                .contact(new Contact()
                    .name("Adrian Alonso Perez")
                    .email("adrianaalonso200@gmail.com"))
                .version("1.0")));
    }
}
```

10.2.4 Seguridad

```
  public class JwtTokenFilter extends OncePerRequestFilter {
    private final JwtTokenProvider jwtProvider;

    public JwtTokenFilter(JwtTokenProvider provider) {
        this.jwtProvider = provider;
    }

    /**
     * Se encarga de leer y validar el token y construir la autenticación del usuario.
     */
    @Override
    protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res, FilterChain chain)
        throws ServletException, IOException {
        String header = req.getHeader("Authorization");
        if (header != null && header.startsWith("Bearer ")) {
            String token = header.substring(7);
            if (jwtProvider.validateToken(token)) {
                Authentication auth = jwtProvider.getAuthentication(token);
                SecurityContextHolder.getContext().setAuthentication(auth);
            }
        }
        chain.doFilter(req, res);
    }
}
```

```
@Component
public class JwtTokenProvider {

    @Value("${jwt.secret}")
    private String secretKey;

    @Value("${jwt.expiration-ms}")
    private long defaultExpiracion;

    @Value("${jwt.expiration-recuerdame-ms}")
    private long recuerdaExpiracion;

    private final UserDetailsService userDetailsService;

    public JwtTokenProvider(UserDetailsService uds) {
        this.userDetailsService = uds;
    }

    /**
     * Crea el token con las especificaciones recibidas.
     *
     * @param username nombre del usuario
     * @param recuerdame boolean para extender el Token
     * @return devuelve el token
     */
    public String createToken(String username, List<String> roles, boolean recuerdame) {
        long now = System.currentTimeMillis();
        long exp = now + (recuerdame ? recuerdaExpiracion : defaultExpiracion);
        Claims claims = Jwts.claims().setSubject(username);
        claims.put("roles", roles);

        return Jwts.builder()
            .setSubject(username)
            .claim("authorities", roles)
            .setIssuedAt(new Date(now))
            .setExpiration(new Date(exp))
            .signWith(SignatureAlgorithm.HS256, secretKey.getBytes())
            .compact();
    }

    public Authentication getAuthentication(String token) {
        UserDetails userDetails = userDetailsService.loadUserByUsername(getUsername(token));
        return new UsernamePasswordAuthenticationToken(userDetails, "", userDetails.getAuthorities());
    }

    public String getUsername(String token) {
        return Jwts.parser().setSigningKey(secretKey.getBytes())
            .parseClaimsJws(token).getBody().getSubject();
    }

    public boolean validateToken(String token) {
        try {
            Jwts.parser().setSigningKey(secretKey.getBytes())
                .parseClaimsJws(token);
            return true;
        } catch (JwtException | IllegalArgumentException e) {
            return false;
        }
    }
}
```



```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    private final JwtTokenProvider jwtProvider;
    private final UserDetailsService userDetailsService;

    public SecurityConfig(JwtTokenProvider jwtProvider, UserDetailsService uds) {
        this.jwtProvider = jwtProvider;
        this.userDetailsService = uds;
    }

    @Bean
    public AuthenticationManager authManager(HttpSecurity http) throws Exception {
        return http.getSharedObject(AuthenticationManagerBuilder.class)
            .userDetailsService(userDetailsService)
            .passwordEncoder(passwordEncoder())
            .and()
            .build();
    }

    /**
     * Encargado de la seguridad, quien puede y quien no acceder.
     *
     * @return devuelve el HTTP con el filtro
     */
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .csrf(csrf -> csrf.disable())
            .sessionManagement(sm -> sm.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .authorizeHttpRequests(auth -> auth
                .antMatchers(HttpMethod.POST, "/usuarios/login").permitAll()
                .antMatchers("/v3/api-docs/**", "/swagger-ui.html",
                    "/swagger-ui/**",
                    "/webjars/**").permitAll()
                .anyRequest().authenticated()
            )
            .addFilterBefore(new JwtTokenFilter(jwtProvider),
        UsernamePasswordAuthenticationFilter.class);
        return http.build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```



10.2.5 Application.properties

```
spring.application.name=RiberPublicFichajeApi
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/riberPublicFichaje
spring.datasource.username=root
spring.datasource.password= toor
server.port=9999
springdoc.api-docs.enabled=true
springdoc.swagger-ui.enabled=true
jwt.secret=MiClaveSuperSecretaQueDebeSerLargaYDifícil
# Expiracion por defecto 9 horas 32400000 || prueba 2 min 120000
jwt.expiration-ms= 32400000
# Recuerdame, 30 días
jwt.expiration-recuerdame-ms=2592000000
```



10.2.6 Base de datos

```
DROP DATABASE IF EXISTS riberPublicFichaje;
CREATE DATABASE riberPublicFichaje;
USE riberPublicFichaje;

CREATE TABLE grupos (
    id int auto_increment primary key,
    nombre varchar(50) not null,
    faltas_totales int not null
);

CREATE TABLE usuarios (
    id int auto_increment primary key,
    nombre varchar(25) not null,
    apellido1 varchar(30) not null,
    apellido2 varchar(30),
    email varchar(40) unique not null,
    contraseña varchar(255) not null,
    rol ENUM('empleado', 'jefe') default 'empleado',
    grupo_id int,
    estado ENUM('activo', 'inactivo') default 'activo',
    FOREIGN KEY (grupo_id) REFERENCES grupos(id) ON DELETE CASCADE
);

CREATE TABLE horarios (
    id int auto_increment primary key,
    grupo_id int not null,
    dia ENUM('lunes', 'martes', 'miercoles', 'jueves', 'viernes') not null,
    hora_entrada time not null,
    hora_salida time not null,
    FOREIGN KEY (grupo_id) REFERENCES grupos(id) ON DELETE CASCADE
);

CREATE TABLE fichajes (
    id int auto_increment primary key,
    usuario_id int not null,
    fecha_hora_entrada DATETIME DEFAULT NULL,
    fecha_hora_salida DATETIME DEFAULT NULL,
    ubicacion varchar(255),
    nfc_usado BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE
);

CREATE TABLE ausencias (
    id int auto_increment primary key,
    usuario_id int not null,
    fecha date not null,
    motivo ENUM('falta_injustificada', 'enfermedad', 'vacaciones', 'permiso', 'retraso', 'otro') DEFAULT 'falta_injustificada',
    estado ENUM('vacio','pendiente', 'aceptada', 'rechazada') default 'vacio',
    justificada boolean default false,
    detalles varchar(255),
    tiempo_registrod TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE
);
```