# G

# Manual Técnico

El presente apéndice adjunta el Manual Técnico de la aplicación Time2Sport y describe sus aspectos técnicos. Su principal objetivo es servir como referencia técnica para futuros desarrolladores encargados del mantenimiento del sistema o la implementación de nuevas funcionalidades. Por ello, el lenguaje empleado será de carácter técnico, distinto al empleado en los capítulos iniciales del trabajo. En concreto, este documento incluirá un listado de requerimientos necesarios para la instalación y configuración del sistema, una guía paso a paso para la ejecución del programa y una descripción detallada del sistema de ficheros. Finalmente, se adjuntará un diagrama entidad-relación representando la base de datos utilizada.

La aplicación Time2Sport ha sido desarrollada y verificada en el sistema operativo Ubuntu 22.04 LTS, garantizando su correcto funcionamiento en dicho entorno y en versiones posteriores. El sistema se ha implementado en Python 3.10.16, siendo compatible con versiones iguales o superiores a 3.10. Para la gestión de paquetes se emplea pip. Se recomienda encarecidamente el uso de un entorno virtual para facilitar la instalación y el aislamiento de dependencias. Los requisitos que deben instalarse a partir del archivo *requirements.txt* son los siguientes:

```
asgiref==3.8.1
certifi==2025.1.31
celery==5.5.1
cffi==1.17.1
chardet==5.2.0
charset-normalizer==3.4.1
cryptography==44.0.1
Django==5.1.6
django-allauth==65.4.1
django-extensions==3.2.3
django-paypal==2.1
freezegun==1.5.1
idna==3.10
```

```
msal==1.31.1
pillow==11.1.0
psycopg2-binary==2.9.10
pycparser==2.22
PyJWT==2.10.1
python-dotenv==0.21.1
redis==5.2.1
reportlab==4.3.1
requests==2.32.3
sqlparse==0.5.3
typing_extensions==4.12.2
urllib3==2.3.0
```

El servidor está desarrollado con Django 5.1.6, utilizando por defecto la base de datos SQLite proporcionada por el propio framework. En el frontend se ha integrado Bootstrap 5.0.2 para el diseño visual (CSS) y Bootstrap 5.3.3 para las funcionalidades interactivas (JavaScript), ambos mediante enlaces CDN.

La aplicación ha sido verificada en los navegadores Firefox (versión 130.0, 64 bits) y Google Chrome (versión 130.0.6723.58, 64 bits).

## G.1. Instalación

A continuación se detallan los pasos necesarios para instalar y configurar el entorno de desarrollo de la aplicación Time2Sport en un sistema basado en Ubuntu.

### G.1.1. Instalar Git y clonar repositorio

Se requiere Git para obtener el código fuente desde el repositorio remoto.

```
sudo apt update
sudo apt install git
git clone https://github.com/esthermtnz/Time2Sport.git
cd Time2Sport
```

### G.1.2. Verificar e instalar Python ($\geq$ 3.10)

Comprobar la versión instalada e instalar Python y pip si fuera necesario.

```
python3 --version
sudo apt install python3
sudo apt install python3-pip
```

### G.1.3. Crear y activar entorno virtual

Se recomienda el uso de un entorno virtual para garantizar un entorno aislado. Antes de crear el entorno virtual, es necesario instalar el paquete correspondiente ejecutando el siguiente comando en la terminal:

```
sudo apt install python3.10-venv
```

Una vez instalado, se crea el entorno virtual con el siguiente comando:

```
python3 -m venv venv
```

Para activar el entorno virtual previamente creado:

```
source venv/bin/activate
```

#### G.1.3.1. Desactivar el entorno virtual

En caso de requerir salir del entorno virtual, desactivarlo.

```
deactivate
```

### G.1.4. Instalar dependencias

Instalar todas las dependencias listadas en el fichero *requirements.txt*.

```
cd time2sport
pip install --upgrade pip
pip install -r requirements.txt
```

### G.1.5. Instalar y configurar Redis

Redis es necesario para la gestión de tareas en segundo plano con Celery.

```
sudo apt install redis-server
sudo systemctl start redis-server
sudo systemctl enable redis-server
redis-cli ping
```

Si *Redis* está operativo, el comando "*redis-cli ping*" devolverá "*PONG*".

### G.1.6. Instalación y configuración de PostgreSQL

Para instalar y habilitar PostgreSQL, se debe ejecutar el siguiente comando en la terminal:

```
sudo apt install postgresql
sudo systemctl enable postgresql
```

#### G.1.6.1. Creación de un usuario con privilegios de superusuario

Una vez instalado PostgreSQL, es necesario crear un usuario con privilegios de superusuario en la base de datos. Para ello, se debe acceder al usuario *postgres* y ejecutar:

```
sudo -i -u postgres
psql
CREATE USER alumnodb WITH SUPERUSER PASSWORD 'alumnodb';
\q
exit
```

De esta manera, se crea un usuario *alumnodb* con privilegios.

#### G.1.6.2. Creación de la base de datos

Para crear una base de datos, *time2sport*, asociada al superusuario *alumnodb*, ejecutar desde la terminal:

```
createdb time2sport -U alumnodb -h localhost
```

Durante la ejecución de este comando, será necesario introducir la contraseña *alumnodb* del usuario *alumnodb*.

#### G.1.6.3. Acciones sobre la base de datos

Una vez creada la base de datos, se pueden realizar diversas acciones como la conexión, visualización de tablas y eliminación de la base de datos.

- Conectarse a la base de datos:

```
psql time2sport -U alumnodb -h localhost
```

  - Ver las tablas existentes:

```
\dt
```

- Eliminar la base de datos:

```
dropdb -U alumnodb -h localhost time2sport
```

Tanto al conectarse como al eliminar la base de datos, se solicitará la contraseña *alumnodb* del usuario *alumnodb*.

#### G.1.6.4. Configuración de .pgpass para evitar introducir la contraseña

Si no se quiere tener que introducir la contraseña *alumnodb* constantemente al ejecutar comandos como *dropdb*, *psql* o cuando se ejecuta *./db.sh*, se puede configurar un archivo *.pgpass* que almacenará la contraseña de manera segura.

Los pasos para crear este fichero son los siguientes:

1. Crea el archivo *.pgpass* escribiendo en la terminal:

```
nano ~/.pgpass
```

2. Añade la información:

```
localhost:5432:postgres:alumnodb:alumnodb
```

3. Guarda y cierra el archivo:

- Para guardar los cambios en *nano*: *Ctrl + O* y luego *Enter*.
- Para salir de *nano*: *Ctrl + X*.

4. Establece permisos seguros para el archivo:

```
chmod 600 ~/.pgpass
```

### G.1.7. Aplicar migraciones y cargar datos iniciales

Generar las migraciones de las aplicaciones y cargar datos con el *populate*.

```
python3 manage.py makemigrations
python3 manage.py migrate
python3 populate.py
```

También es posible ejecutar las migraciones especificando los modelos:

```
python3 manage.py makemigrations sgu sbai src slegpn
```

Alternativamente se puede ejecutar el siguiente *script* que automatiza todo el proceso relacionado con la base de datos. Este *script* realiza las siguientes acciones:

- Eliminación de migraciones anteriores y de la base de datos.
- Eliminación y creación de tablas.
- Generación de nuevas migraciones.
- Población de la base de datos.

Para ejecutarlo:

```
./db.sh
```

Si no se ha configurado *.pgpass* se pedirá dos veces la contraseña *alumnodb*: una para la eliminación de la tabla y otra para la creación.

## G.2. Ejecución

### G.2.1. Ejecutar el servidor

Iniciar el servidor de desarrollo de Django:

```
python3 manage.py runserver
```

### G.2.2. Ejecutar los workers de Celery

Para el manejo de tareas asincrónicas, ejecutar en otra terminal:

```
source ../venv/bin/activate
celery -A time2sport worker --loglevel=info
```

O bien, ambos procesos en segundo plano en una sola terminal:

```
celery -A time2sport worker --loglevel=info &
python3 manage.py runserver
```

### G.2.3. Acceder a la aplicación desde el navegador

Una vez iniciado el servidor, la aplicación estará disponible en:

```
http://localhost:8000
```

## G.3. Tests

### G.3.1. Todos los tests

Ejecutar todos los tests de la aplicación:

```
python3 manage.py test
```

### G.3.2. Por incrementos

Tests correspondientes al primer incremento (subsistemas SGU y SBAI):

```
python3 manage.py test sgu sbai
```

Tests correspondientes al segundo incremento (subsistemas SRC y SLEGPN):

```
python3 manage.py test src slegpn
```

### G.3.3. Tests por aplicación específica

Ejecutar los tests de una única aplicación:

```
python3 manage.py test <app>
```

Donde <*app*> puede ser uno de los siguientes valores: *sgu*, *sbai*, *src*, o *slegpn*.

## G.4. Opcional

### G.4.1. Acceso al panel de administración

Para gestionar usuarios, reservas y demás elementos del sistema, es posible crear un superusuario mediante el siguiente comando:

```
python3 manage.py createsuperuser
```

Una vez creado, se puede acceder al panel de administración desde:

```
http://localhost:8000/admin
```

### G.4.2. Prueba de pago en entorno de pruebas

Para validar el funcionamiento del sistema de pagos sin afectar cuentas reales, se proporciona un entorno de pruebas con las siguientes credenciales:

Cuenta de pago (Payer Account)

```
Email: sb-o4c6a38227363@personal.example.com
Contraseña: m4t1(!Yo
```

Cuenta receptora (Receiver Account)

```
Email: sb-ruupa38090694@business.example.com
Contraseña: ym7hX@]/
```

Este entorno es completamente simulado: las transacciones no implican movimientos reales ni afectan cuentas bancarias.

### G.4.3. Visualización del código con Visual Studio Code (VS Code)

Para editar o visualizar el código fuente de manera sencilla, se recomienda utilizar el editor VS Code. Su instalación en Ubuntu puede realizarse de la siguiente manera:

1. Verficar si Snap está instalado:

```
snap --version
```

2. En caso de no tenerlo, instalar Snap y VSCode

```
sudo apt install snapd
sudo snap install code --classic
```

También puede descargarse desde su sitio web oficial: https://code.visualstudio.com

Una vez instalado, el proyecto puede abrirse con:

```
code .
```

## G.5. Estructura del repositorio

El repositorio del proyecto *Time2Sport* presenta la siguiente estructura general:

- *README.md*: Incluye una versión resumida del manual técnico.
- *ManualUsuario.pdf*: Describe cómo el usuario final interactúa con la aplicación.
- *ManualTecnico.pdf*: Contiene el Apéndice G referente al manual técnico.
- *.gitignore*: Especifica los archivos y directorios que deben excluirse del control de versiones (*db.sqlite3*, *venv/*, *env/*, *__pycache__/*, *migrations/*, etc.).
- *time2sport/*: Directorio principal que contiene la aplicación, el código fuente y los archivos de configuración.

Dentro del directorio *time2sport/* se encuentran:

- *requirements.txt*: Lista de dependencias necesarias para ejecutar el proyecto.
- *manage.py*: Script de administración de Django.
- *populate.py*: Script para la carga inicial de datos.
- *db.sh*: Script para aplicar migraciones y poblar la base de datos.
- *media/*: Archivos multimedia cargados por los usuarios.
- *static/*: Recursos estáticos.
- *templates/*: Plantillas HTML, entre ellas *base.html* y *home.html*.
- *sgu/*, *sbai/*, *src/*, *slegpn/*: Módulos que implementan los diferentes subsistemas del proyecto. Cada uno contiene:
    - *models.py*: Define las estructuras de datos y relaciones.
    - *views.py*: Lógica de control de la aplicación.
    - *urls.py*: Enlaces de vistas del módulo.
    - *templates/*: Plantillas específicas del módulo.
    - *tests/*: Conjunto de pruebas unitarias.

  *slegpn/* incluye además:
    - *tasks.py*: Define tareas asíncronas con *Celery* para la gestión de la lista de espera.
    - *utils.py*: Contiene funciones auxiliares, como el cálculo de descuentos para usuarios de la UAM.
- *time2sport/* (subdirectorio): Contiene la configuración principal del proyecto, destacando:
    - *settings.py*: Configuraciones generales del proyecto Django (bases de datos, rutas estáticas, middleware, aplicaciones instaladas, etc.).
    - *urls.py*: Define las rutas globales del proyecto.
    - *celery.py*: Configuración de *Celery*.
    - *context_processors.py*: Permite mostrar dinámicamente el número de notificaciones no leídas en las vistas.

## G.6. Base de datos

La base de datos del sistema está construida a partir de los modelos definidos en Django, generando automáticamente el esquema relacional correspondiente. El esquema se ha realizado utilizando la herramienta *graph_models*. En la Figura G.1 se presenta una visión completa de la base de datos, que incluye tanto los modelos propios del sistema como las tablas generadas por aplicaciones externos utilizadas en el desarrollo del proyecto. Por otro lado, la Figura G.2 muestra únicamente los modelos definidos por el equipo de desarrollo, permitiendo una visión más clara y simple de la aplicación.

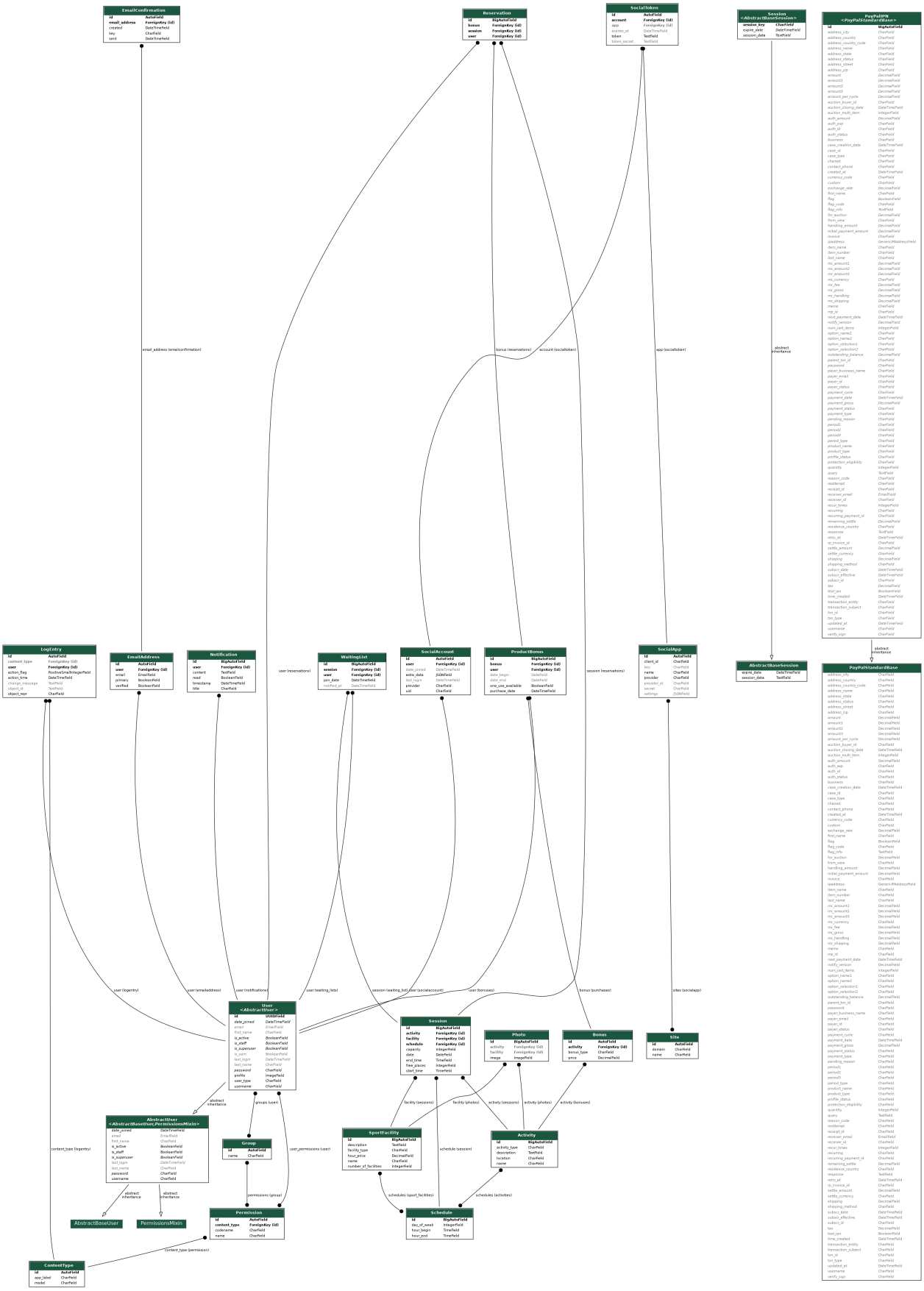

**Figura G.1:** Esquema completo base de datos

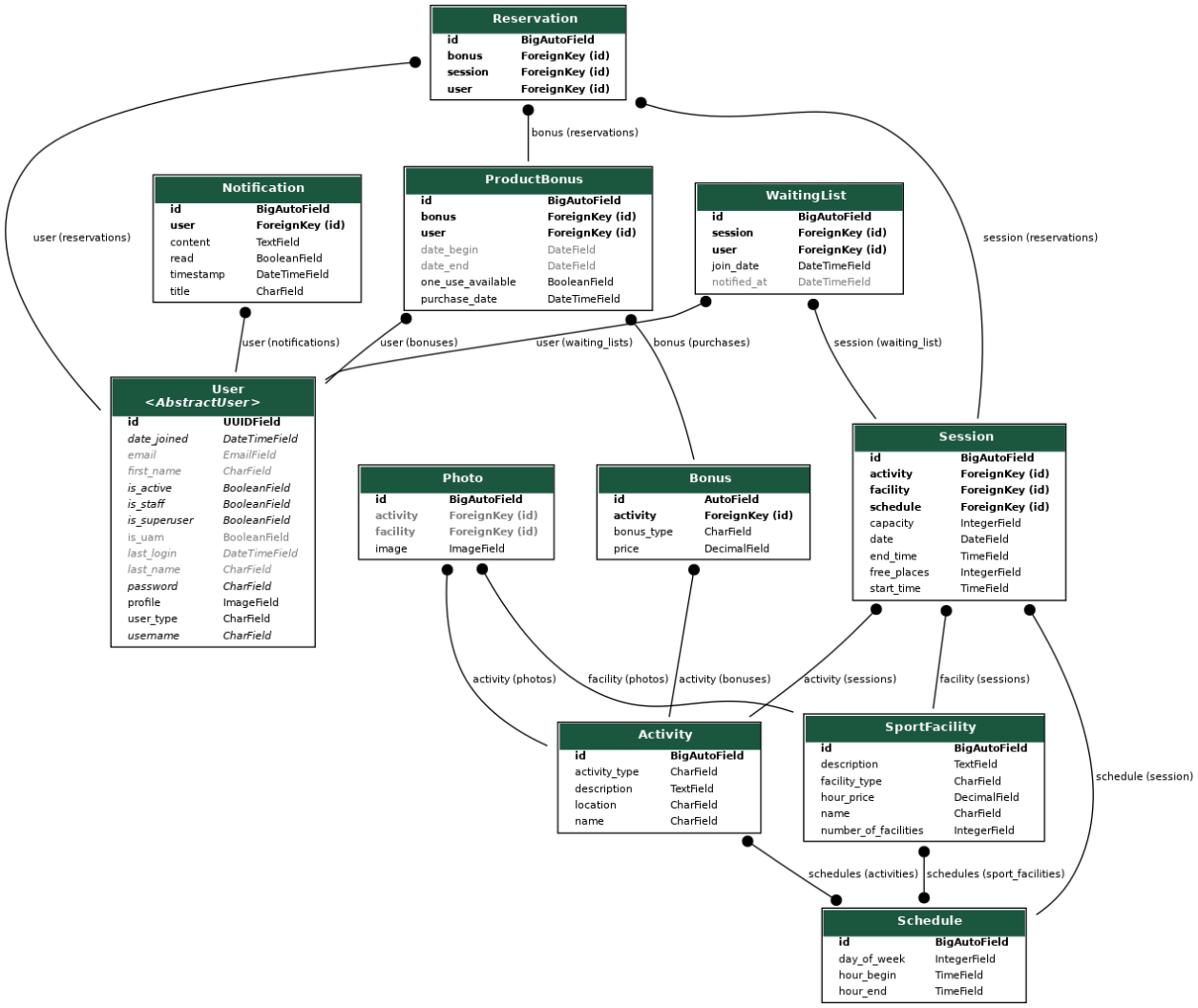


**Figura G.2:** Esquema modelos base de datos