# Proyecto Final DAW - 2023


Noa González Salgado

CIFP A Carballeira

# Índice

# 1. Introducción

Subsify es un gestor de suscripciones online que permite al usuario realizar un seguimiento de todos sus compromisos con servicios digitales en un solo lugar.

Hoy en día vivimos en una era digital donde la oferta de servicios es amplia y variada, desde plataformas de entretenimiento y formación en línea hasta herramientas de productividad y software especializado. La mayoría de estos servicios adoptan un modelo de suscripción, en general, con precios muy asequibles. Sin embargo, esto puede derivar en que los usuarios acaben suscritos a servicios que apenas utilizan y por tanto realizando gastos innecesarios.

Subsify surge como respuesta a esta realidad, tratando de satisfacer la creciente necesidad de los usuarios de tener un mayor control sobre sus suscripciones digitales. Subsify ofrece una visión clara del gasto mensual del usuario, lo que facilita la toma de decisiones sobre qué servicios mantener y/o cancelar.

# 2. Objetivos

El desarrollo se Subsify tiene una serie de objetivos fundamentales que buscan mejorar la experiencia digital de sus usuarios:

- **Centralización de suscripciones:** la aplicación actúa como un repositorio centralizado de todas las suscripciones digitales de un usuario. De esta forma se elimina la necesidad de consultar múltiples servicios para conocer importes y/o fechas de renovación
- **Recordatorios:** Subsify informa a sus usuarios de las fechas en las que se producirán las renovaciones de los servicios que tiene contratados, de forma que pueda planificar y tomar decisiones sobre la continuidad de sus suscripciones. Además, recuerda al usuario que debe dar de baja aquellas suscripciones que haya decidido cancelar
- **Análisis de gasto:** la app proporciona funcionalidades que permiten al usuario conocer exactamente cuánto gasta cada mes en estos servicios y detectar oportunidades de ahorro
- **Gestión de cancelaciones:** Subsify trata de simplificar el proceso ofreciendo enlaces directos a las diferentes plataformas para proceder a la baja del servicio
- **Facilidad de uso:** la aplicación ofrece una interfaz intuitiva y fácil de usar, garantizando que los usuarios puedan sacar el máximo provecho de las funcionalidades desarrolladas

## 3. Estudio de mercado

La economía de las suscripciones ha tenido un notable crecimiento durante los últimos años. Hoy en día podemos encontrarlo en muchos ámbitos de nuestras vidas: desde servicios digitales hasta la compra de productos frescos, ropa o automóviles. La llegada de esta modalidad de pago ha transformado la manera en que accedemos a productos y servicios.

Este crecimiento destaca sobre todo en la industria del contenido digital, debido a que los usuarios están cambiando sus hábitos de consumo y cada vez están más dispuestos a pagar por estos contenidos para poder acceder desde cualquier dispositivo y lugar o no ver anuncios. En este cambio de hábitos ha tenido especial relevancia la situación de pandemia mundial vivida en los últimos años, que supuso un aumento exponencial del consumo de contenido digital.

A este escenario se suma la situación de elevada inflación actual. Puesto que estos servicios ofrecen generalmente precios muy asequibles, los usuarios encuentran una forma económica de poder acceder a servicios de calidad.

Algunos servicios de suscripción con mayor número de usuarios en España que ejemplifican este modelo son los siguientes[1]:

- Netflix: 9.600.000
- Amazon Prime Video: 6.600.000
- HBO: 3.600.000

Según un informe realizado por Telecoming[2], empresa especializada en monetización y pago móvil, en el año 2021 había en España 29,4 millones de suscripciones activas, y sus previsiones son que estas cifras sigan creciendo año a año, triplicando estos datos para 2026.

El pago de suscripciones supone pequeños gastos de los que el usuario puede no ser consciente. Por ello surge la necesidad de un nuevo producto que complemente a las aplicaciones de finanzas tradicionales, de forma que el usuario sea más consciente de lo que gasta en estos servicios y pueda tomar decisiones sobre su consumo y gasto en contenidos digitales.

[1] **Statista** - Número de suscriptores de las plataformas de vídeo en streaming en España en 2022
[2] **Telecoming** – Contenido digital. Evolución España & Europa 2021 - 2026

## 4. Planificación

Este proyecto ha sido realizado utilizando metodologías ágiles, en concreto Scrum. El desarrollo se ha dividido en 4 sprints de dos semanas de duración cada uno en los que se han ido añadiendo funcionalidades de forma iterativa e incremental

### Sprint 1

- Funcionalidad que permite al usuario registrar sus suscripciones activas
- Listado y detalle de suscripciones activas
- Estilos generales de la aplicación

### Sprint 2

- Gráfica de gasto
- Gestión de permisos de usuarios y administradores
- Funcionalidades de administrador: añadir plataformas y planes de precio

### Sprint 3

- Panel de inicio
- Funcionalidad de compartir cuentas
- Funcionalidad de añadir plataformas personalizadas

### Sprint 4

- Funcionalidad de añadir promociones temporales a las suscripciones
- Funcionalidad de administrador: validar plataformas personalizadas para que estén disponibles para el resto de los usuarios

## 5. Presupuesto

La realización de este proyecto no requiere de una gran inversión inicial, ya que inicialmente solo es necesario afrontar los costes asociados con el desarrollo y el alojamiento de la aplicación.

En la elaboración del presupuesto se ha tenido en cuenta lo siguiente:

- **Coste mensual del hosting:** se estima un gasto mensual de 10€. Se trata de un precio aproximado, ya que puede variar en función del proveedor de los servicios. El presupuesto se ha calculado con el coste anual.
- **Tarifa por hora de desarrollo:** se ha fijado en 20€. Este importe también podría variar en función de la complejidad de las funcionalidades a desarrollar.

| Concepto | Horas | Coste |
|---|---|---|
| Hosting | - | 100€ |
| Diseño y planificación | 10 | 200€ |
| Desarrollo | 200 | 4.000€ |
| Despliegue | 5 | 100€ |
| **TOTAL** | | **4.400€** |

## 6. Tecnologías y herramientas utilizadas

### Backend

- **Java:** lenguaje multiplataforma y orientado a objetos, ampliamente utilizado en el desarrollo de aplicaciones web

- **Spring:** framework de código abierto que facilita el desarrollo de aplicaciones en Java

- **Ontimize Boot:** framework basado en Spring Boot que simplifica el desarrollo de una API REST

- **Maven:** herramienta de compilación y gestión de dependencias

### Frontend

- **CSS:** lenguaje basado en reglas que permite añadir estilos a una aplicación o página web

- **TypeScript:** extensión del lenguaje JavaScript que permite añadir tipado estricto

- **Angular:** framework de JavaScript que permite el desarrollo de SPAs (Single Page Applications) y que facilita la reutilización de código mediante la creación de componentes

- **Ontimize Web:** framework basado en Angular que facilita la autenticación/autorización de usuarios, permite la gestión de permisos a usuarios con diferentes roles y proporciona componentes que simplifican las llamadas a la API.

- **NodeJS/npm:** herramientas para la gestión de paquetes y generar el build de producción

## Base de datos

- **PostgreSQL:** base de datos de código abierto que ofrece un gran rendimiento

- **DBeaver:** herramienta para la administración de la base de datos

## Otras herramientas

- **VSCode:** IDE con el que ha sido desarrollado el frontend

- **IntelliJ:** IDE con el que ha sido desarrollado el backend

- **Postman:** herramienta utilizada para testear el funcionamiento de la API

- **GIT:** sistema de control de versiones de código abierto

- **Github:** repositorio remoto en el que se puede encontrar el código de la aplicación

- **Docker:** plataforma de código abierto utilizara para desplegar el backend en un contenedor virtual

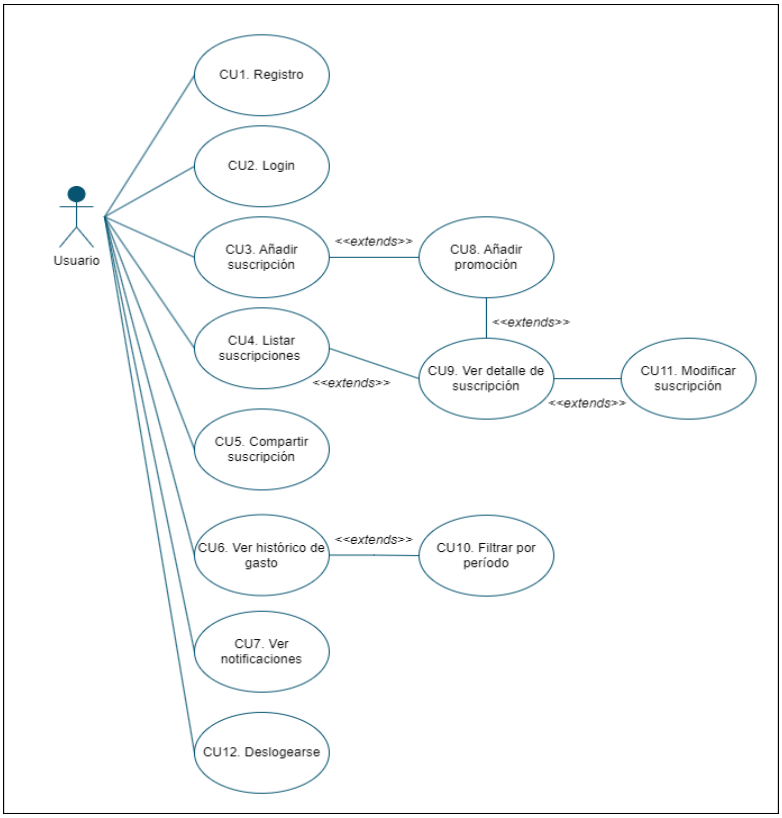
- **Draw.io:** herramienta para la realización de diagramas y diseño del modelo de datos

# 7. Documentación técnica

## 7.1 Casos de uso

La aplicación ha sido diseñada para atender a dos tipos de usuarios distintos:

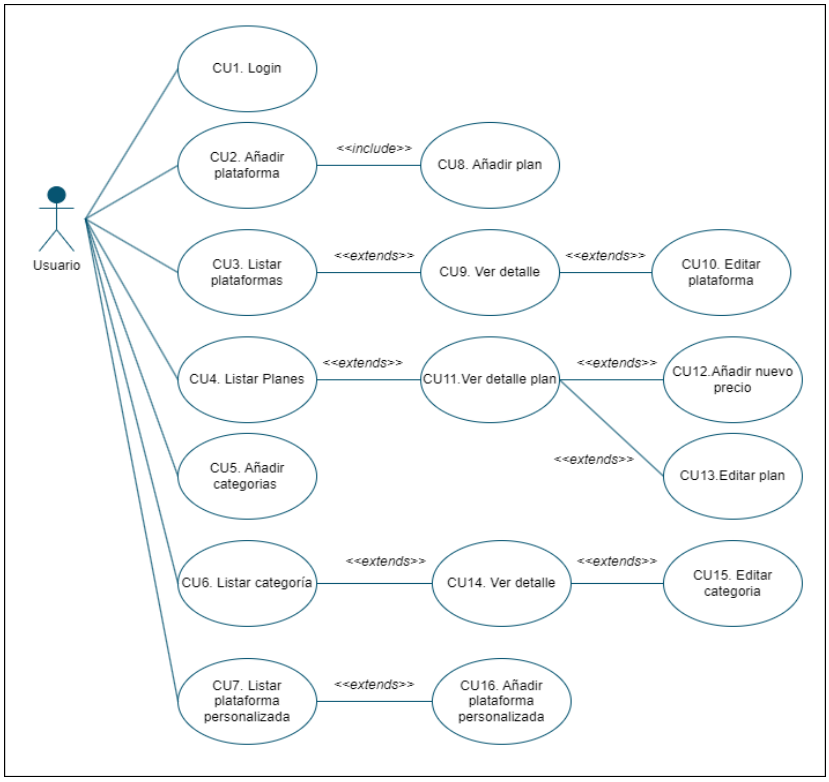
**Usuario registrado (cliente)**




*1 Diagrama casos de uso usuario*

Utilizará la aplicación para hacer seguimiento de sus suscripciones. Podrá agregar nuevas suscripciones, consultar un listado con las que mantiene activas y actualizarlas.

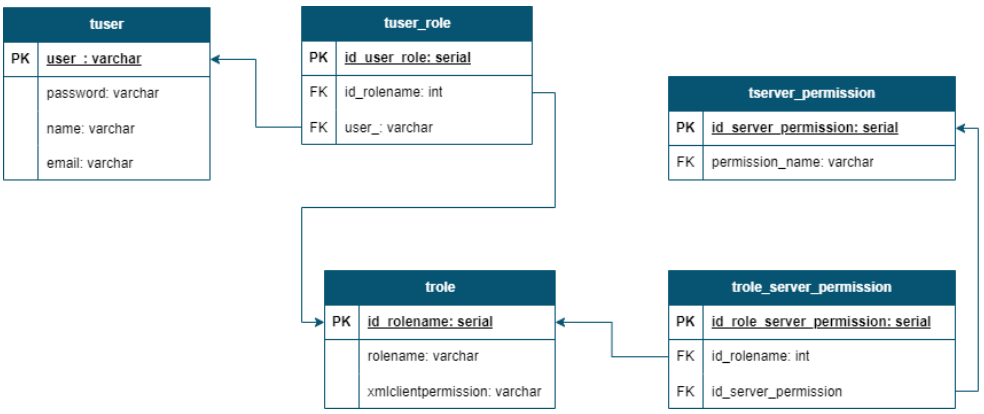
## Administrador




*2 Diagrama casos de uso administrador*

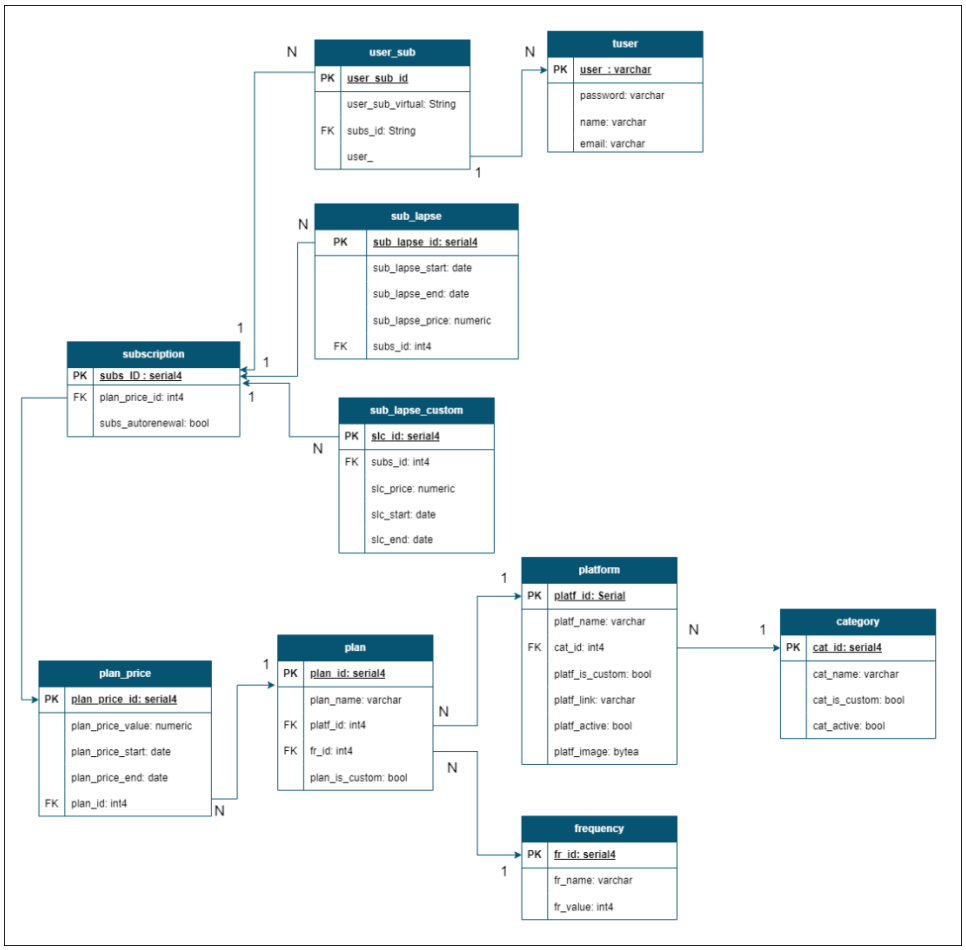
Su función principal es la de mantener y dar de alta las plataformas y los planes a los que los usuarios se suscriben.

## 7.2 Modelo de datos




*3 Modelo de datos permisos de usuarios*

Estas tablas del modelo de datos se utilizan para poder asignar roles a los distintos tipos de usuarios que puede tener la aplicación y otorgarles permisos según corresponda a cada rol.



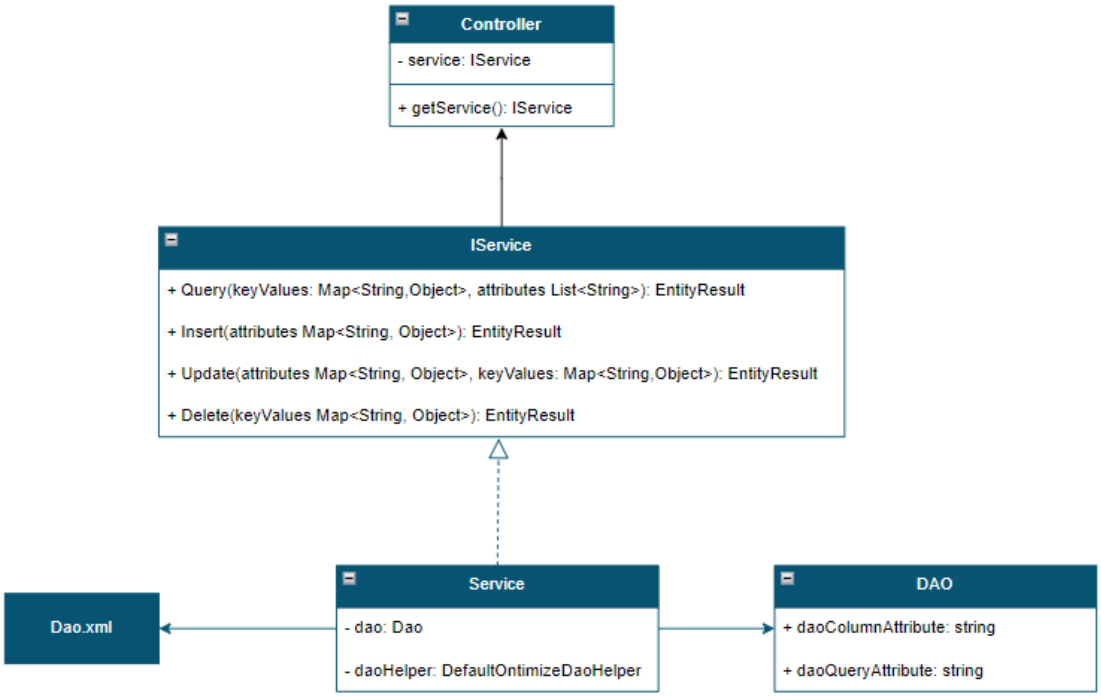

*4 Modelo de datos de la aplicación*

El propósito de cada una de las entidades anteriores es el siguiente:

- **Subscription:** representa una suscripción de usuario. Contiene el campo *subs_autorenewal* que permite identificar las sucripciones que un usuario desea renovar y las que no
- **Sub_lapse:** representa el lapso de tiempo de una suscripción activa hasta la siguiente renovación. De esta forma, a futuro, se podrá consultar los períodos en los que un usuario ha estado suscrito a una plataforma y por tanto conservar un historial de gasto en caso de que el usuario desee cancelar temporalmente una suscripción
- **Sub_lapse_custom:** representa un lapso de tiempo en el que una suscripción ha tenido un precio promocional
- **User_sub:** representa al usuario que posee una suscripción. Es una tabla intermedia, ya que un usuario (modelado en la entidad ***tuser***) puede tener varias suscripciones y una suscripción puede ser compartida por varios usuarios. El campo *user_sub_virtual* se utiliza en caso de que el usuario con el que se desea compartir cuenta no se encuentre registrado en la plataforma

- **Platform:** representa una plataforma o servicio al que un usuario puede suscribirse
- **Category:** se utiliza para asignar una categoría a cada plataforma
- **Plan:** se utiliza para registrar los planes de precios asociados a cada plataforma
- **Frequency:** frecuencia de pago que puede tener un plan de precio: mensual, trimestral o anual
- **Plan_price:** representa el precio asociado a cada plan. Se genera un nuevo registro en esta tabla cada vez que un plan actualiza su precio

## 7.3 Clases




*5 Estructura API*

Cada una de las entidades de la base de datos contiene su correspondiente definición de clases en el backend siguiendo la estructura de la imagen.

Tomando como ejemplo la entidad *Category*, las clases que se generan son las siguientes:

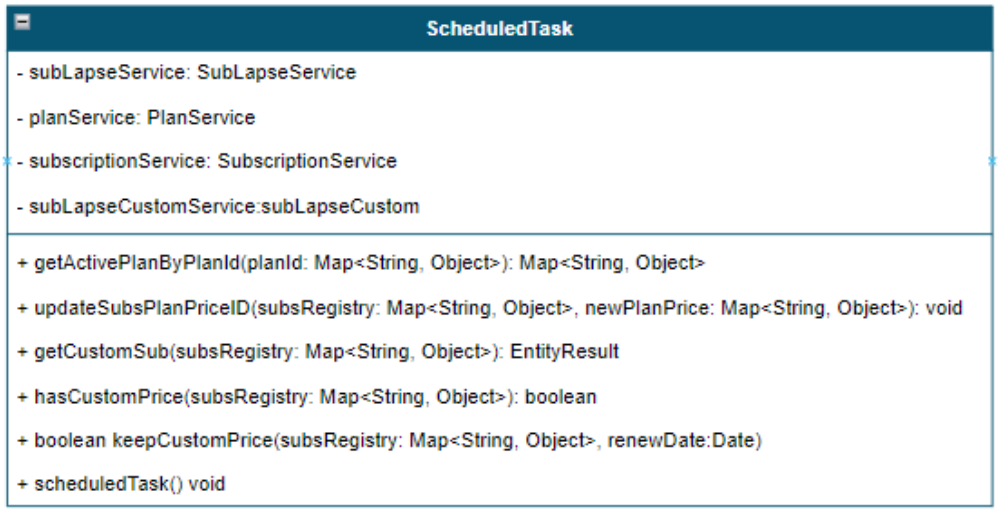
- **CategoryController:** controlador que define el endpoint al que debe llamar el frontend para presentar los datos

- **ICategoryService:** interfaz que establece los métodos que debe implementar el servicio asociado
- **CategoryService:** servicio en el que se definen las operaciones CRUD relacionadas con la entidad
- **CategoryDao:** se definen como constantes los nombres de las columnas de la entidad para facilitar el trabajo con ellas
- **CategoryDao.xml:** archivo que contiene las consultas SQL que se realizarán a la base de datos relacionadas con la entidad

Todas estas clases se organizan siguiendo la siguiente estructura:

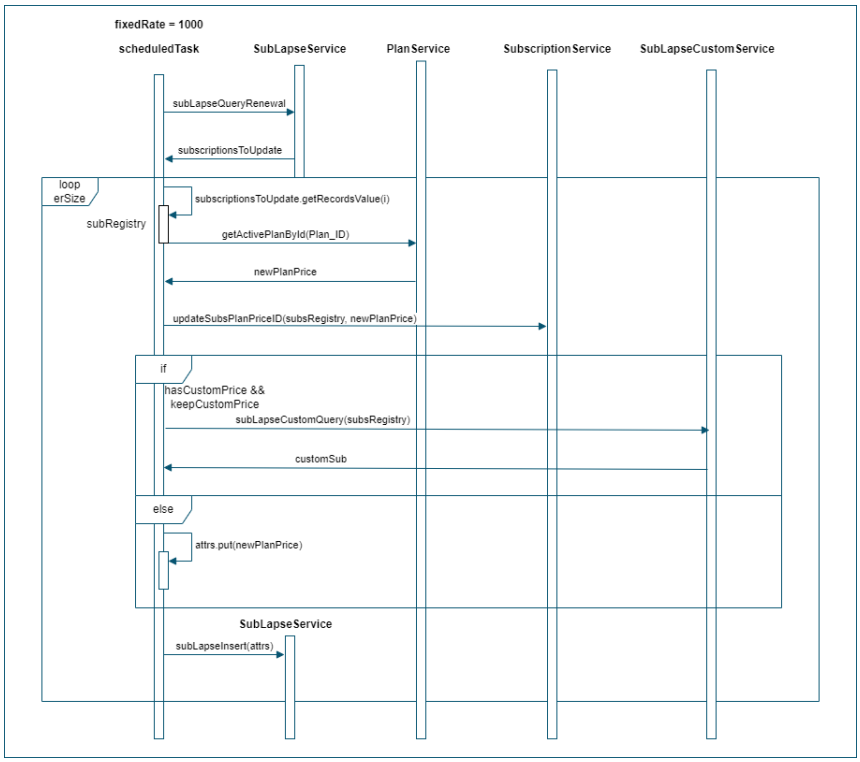
- **/backend-api:** almacena las interfaces que debe implementar cada servicio (IService)
- **/backend-boot:** contiene la clase principal que permite ejecutar la aplicación
- **/backend-model:** incluye las clases Dao y los archivos Dao.xml
- **/backend-ws:** contiene los controladores

El backend también contiene una tarea recurrente que se encarga de renovar las suscripciones que hayan caducado o actualizar sus precios si es necesario. La clase que se ocupa de realizar esto es `ScheduledTask` y contiene los siguientes métodos y atributos:




*6 Tarea Recurrente*

El funcionamiento de esta clase puede verse en el siguiente diagrama de secuencia:



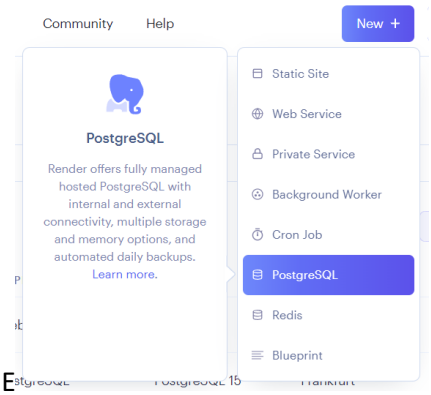

*7 Diagrama de secuencia de tarea recurrente*

## 8. Despliegue

Para el despigue de la aplicación se han utilizado dos servicios en la nube: **Netlify** para el frontend y **Render** para la base de datos y el backend. Ambas plataformas ofrecen planes gratuitos limitados para pequeños proyectos o proyectos personales.

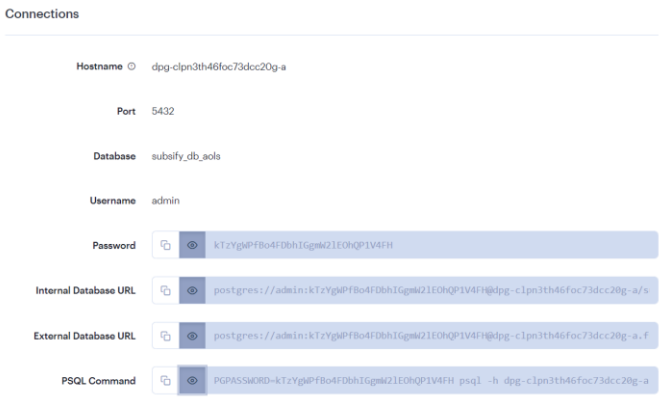
### Base de datos

En el panel principal de **Render**, se crea una nueva base de datos haciendo clic en el botón *"New"* y seleccionando *PostgreSQL* en el menú desplegable



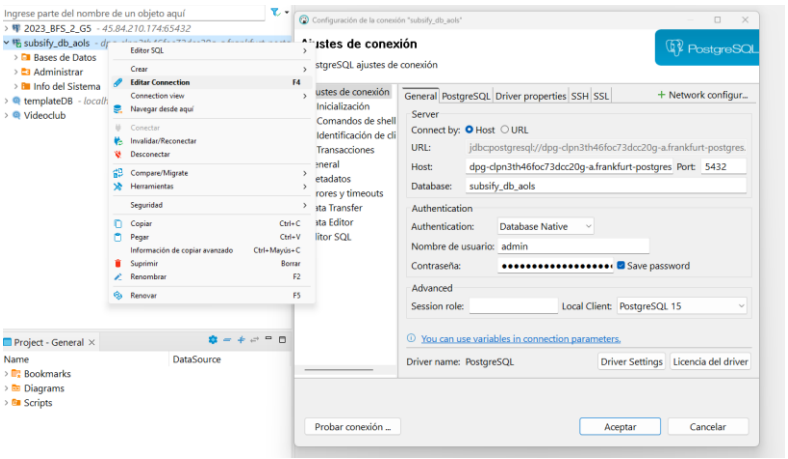

*8 Menú para crear nueva base de datos en Render*

Completar el formulario con el nombre de la base de datos y un usuario y clic en "*Create Database*". Tras unos segundos, se obtienen los datos de conexión para la base de datos




*9 Datos de conexión para la base de datos*

Por último, hay que actualizar los datos de conexión de la base de datos local en **DBveaber** para poder tener acceso a las tablas

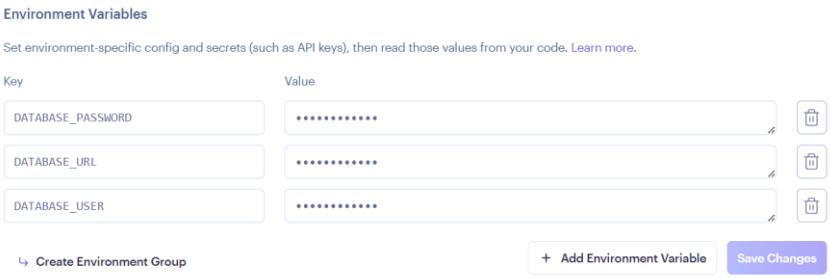



*10 Actualización de los datos de conexión en DBeaver*

## Backend

En el panel principal se crea un nuevo servicio web seleccionando la opción *"Web Service"* desde el menú desplegable. A continuación, se conecta **Render** con la cuenta de **Github** que contiene el repositorio que se va a desplegar y se selecciona.

Una vez que se ha establecido la conexión, se deben crear las variables de entorno que contendrán los datos de conexión a la base de datos.




*11 Variables de entorno que permiten no exponer los datos de conexión*

Dentro del proyecto, hay que editar los datos de conexión a la base de datos en el archivo *`application.yml`* (ubicado en la ruta `backend/backend-boot/src/main/resources`) con las variables de entorno creadas previamente

```yaml
spring:
  datasource:
    driver-class-name: org.postgresql.Driver
    jdbc-url: ${DATABASE_URL}
    username: ${DATABASE_USER}
    password: ${DATABASE_PASSWORD}
```

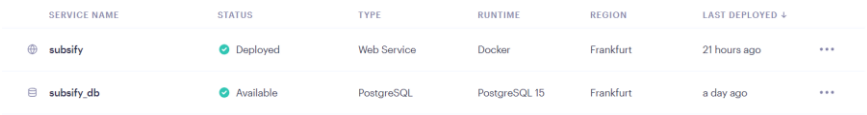
*12 Datos de conexión actualizados en el archivo application.yml*

En **Render**, el código se ejecuta en un contenedor de **Docker**, por lo que hay que crear un *Dockerfile* en la raíz del proyecto con las instrucciones necesarias para instalar **Java** y **Maven** dentro del contenedor y ejecutar el código

```dockerfile
FROM ubuntu:latest as build

RUN apt-get update
RUN apt-get install openjdk-17-jdk -y
COPY . .

RUN apt-get install maven -y
RUN mvn clean install

FROM openjdk:17-jdk-slim

EXPOSE 8080

COPY --from=build /backend-boot/target/backend-boot.jar app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]
```

*13 Contenido del Dockerfile*

Una vez subidos estos cambios a **Github** y hecho clic en el botón *"Create Web Service",* el despliegue comienza de forma automática.




*14 Panel que muestra que el despliegue se ha realizado correctamente*

### Frontend

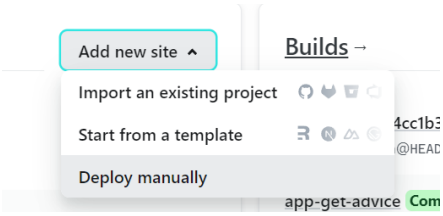
Actualización el archivo *app.config.ts* (situado en la ruta *frontend\src\app*), con la dirección de la API proporcionada por **Render**

```
7   //apiEndpoint: "http://localhost:33333",
8   apiEndpoint: "https://subsify.onrender.com",
```

***15** Línea del archivo app.config.ts a actualizar*

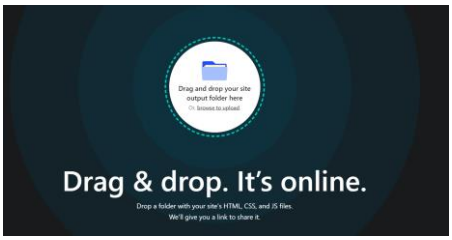
A continuación, desde la terminal de **VSCode**, se ejecuta el comando *npm run build* para generar el build de producción (carpeta */dist*)

Desde el panel principal de **Netlify**, se agrega un nuevo sitio haciendo clic en *"Add new site"* y seleccionando la opción *"Deploy Manually"* del menú desplegable




***16** Menú para crear un nuevo sitio en Netlify*

Aparece una nueva ventana para arrastrar y soltar la carpeta */dist* generada con el comando *build*. Tras unos minutos el despliegue se realiza correctamente y se obtiene el link para acceder a la aplicación a través de internet




***17** Drag & Drop en Netlify*

## 9. Instalación

### Requisitos previos

Para ejecutar la aplicación en local hay que realizar las siguientes instalaciones:

- Java[3]

[3] Descarga de Java - https://adoptium.net/es/temurin/releases/?variant=openjdk11

- Maven[4]
- NVM[5]

## Configuración del backend

Desde la terminal, clonar el repositorio con el comando *`git clone https://github.com/NoaSalgado/Subsify`*. Si Git no está instalado, el código puede descargarse en formato *.zip* desde el mismo enlace.

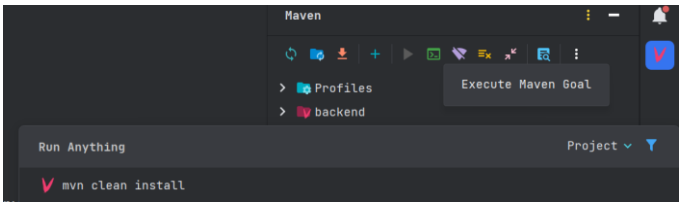
Abrir el directorio */backend* con el IDE para actualizar el archivo *appliaction.yml* situado en la ruta *backend/backend-boot/src/main/resources* con los datos de conexión de la base de datos de desarrollo.

```yaml
spring:
  datasource:
    driver-class-name: org.postgresql.Driver
    jdbc-url: jdbc:postgresql://45.84.210.174:65432/2023_BFS_2_G5
    username: 2023_BFS_2_G5
    password: b2f34VKzTvt2
```

*18 Datos de conexión a la base de datos de desarrollo*

Esta base de datos ya se encuentra en un servidor remoto, por lo que no es necesario ejecutar el script para crearla de nuevo (si fuese necesario, en el anexo de la memoria se encuentra un enlace para acceder al script).

Desde la terminal, se ejecuta el comando `mvn clean install` para cargar las dependencias. Si se utiliza **IntelliJ**, puede ejecutarse directamente desde la terminal de **Maven** que tiene integrada.




*19 Terminal de Maven integrada en IntelliJ*

Por último, hay que ejecutar la clase principal, *ServerApplication* (ubicada en la ruta *backend-boot\src\main\java\com\campusdual*) para poner en funcionamiento la aplicación.

[4] Descarga e instalación de Maven - https://maven.apache.org
[5] Descarga e instalación de NVM - https://github.com/coreybutler/nvm-windows

### Configuración del frontend

Actualizar el archivo *app.config.ts* (situado en la ruta *frontend\src\app\app.config.ts*) con la dirección de la API de desarrollo.

```
7   apiEndpoint: "http://localhost:33333",
8   //apiEndpoint: "https://subsify.onrender.com",
```

*20 Línea a actualizar del archivo app.config.ts*

Desde la terminal se ejecuta el comando `nvm use 12.22.10` para instalar la versión de **NodeJS** bajo la que funciona **Ontimize Web**. A continuación, ejecutar `npm install` para instalar las dependencias y *`npm start`* para iniciar la aplicación. Una vez finalizada la compilación del proyecto, se puede acceder a la aplicación desde el navegador desde la dirección *http://localhost:4200/.*

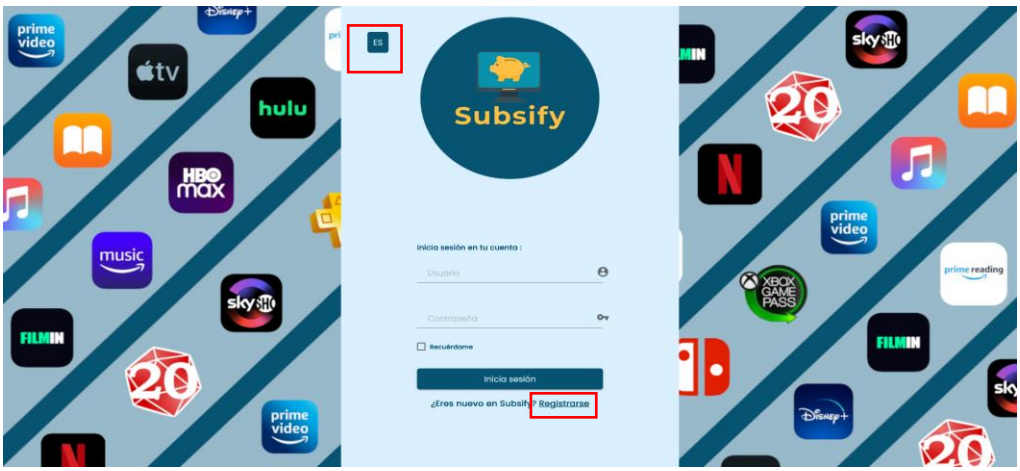
# 10. Manual de usuario

## 10.1 Usuario

### Login/Registro

Para acceder a la aplicación, es necesario identificarse como usuario. En el caso de no contar con una cuenta, haciendo clic en el botón *"Registrarse"* redirigirá al formulario de registro para crearla.

La primera vez que se accede a la aplicación, se muestra por defecto en inglés. Se puede cambiar el idioma fácilmente seleccionando la opción *"ES"* en el menú desplegable de idiomas.




*21 Login de usuario*

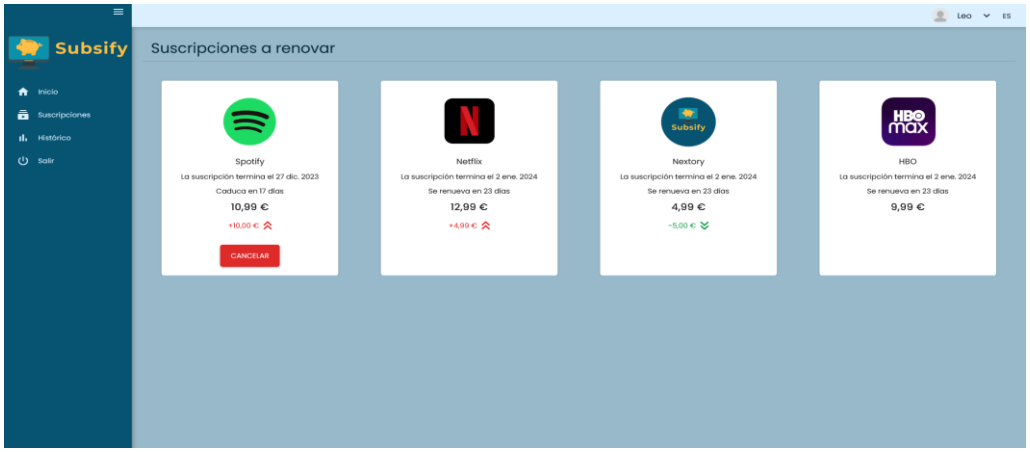
## Pantalla de inicio

En esta pantalla aparecen las suscripciones que se van a renovar en los próximos días, indicando la fecha en la que se producirá el pago, el precio de renovación y si sube o baja de precio.

En la imagen se puede observar como la suscripción a Spotify, que se contrató por 0.99€, sube de precio. Aparece también el botón *"CANCELAR"*, que recuerda al usuario que debe dar de baja esta suscripción que ha decidido no renovar y es, además, un enlace directo a la plataforma para completar el proceso de cancelación.

En el caso de Nextory, plataforma no registrada en Subsify e introducida por el usuario, se indica que el precio de renovación es de 4.99€. El precio de renovación disminuye en 5€ debido a un descuento registrado por el usuario.

En el caso de Netflix, se avisa de que la plataforma ha decidido subir los precios. Estos cambios serán registrados por el administrador de la aplicación.



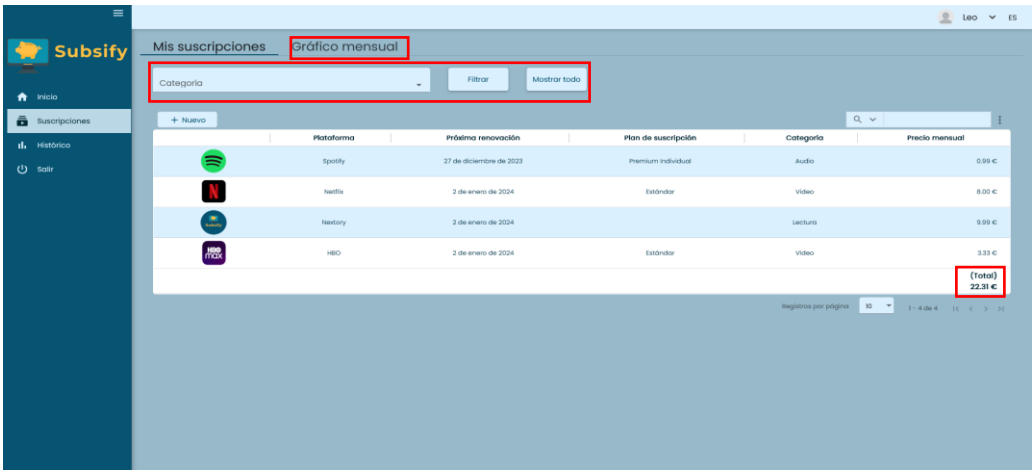

*22 Pantalla de inicio*

## Ver suscripciones activas

En el menú lateral aparece la opción *"Suscripciones"*. Al seleccionarla, se puede ver un listado con las suscripciones activas en la actualidad.

El listado ofrece información sobre el coste mensual de cada suscripción, la fecha de la próxima renovación, el plan contratado y la categoría a la que pertenece. Además, se puede ver el importe total gastado en suscripciones en el mes en curso.

Este listado es posible filtrarlo por categoría, para conocer el importe gastado en cada tipo de servicio. También se puede ver una representación visual seleccionando la pestaña "Gráfico Mensual", para reconocer a simple vista qué suscripción cuesta más dinero.



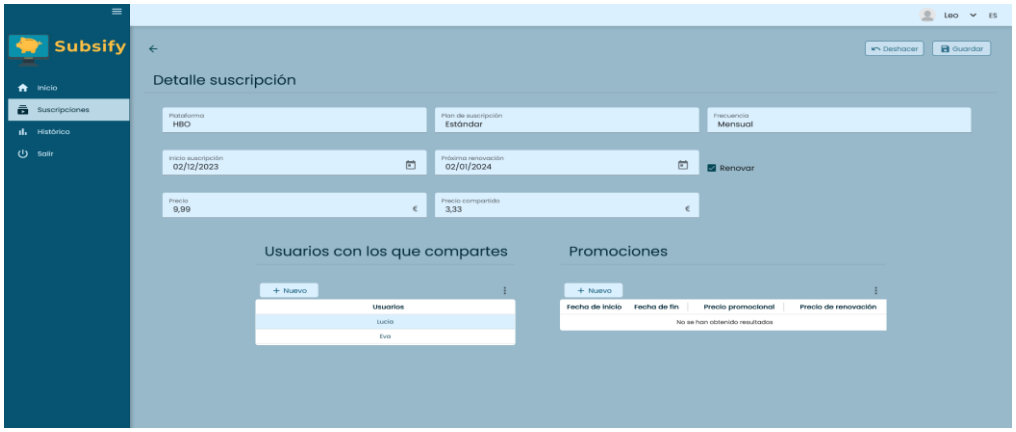

*23 Listado de suscripciones activas*

## Ver detalle

Para conocer los detalles de una suscripción, basta con hacer clic en la plataforma deseada dentro del listado que se puede ver en la imagen anterior.

En este detalle se puede ver toda la información relacionada con una suscripción, si se comparte cuenta y con quién o las promociones temporales que se han disfrutado. Desde aquí, también es posible editar la suscripción en caso de que se haya introducido algún dato erróneo.

Además, hay un checkbox con el texto *"Renovar"* que aparece marcado por defecto. Al desmarcarlo, la aplicación sabrá que el usuario no quiere volver a renovar esa suscripción y le avisará para que no se olvide de cancelarla.




*24 Detalle de una suscripción*

## Compartir cuenta

En la propia vista de detalle, se hace clic en el botón *"Nuevo"* de la tabla de usuarios con los que se comparte. Aparece un modal en que se pueden introducir usuarios de dos formas: seleccionando un usuario que ya está registrado en Subsify o introducirlo de forma manual.


*25 Compartir cuenta*

## Añadir promoción

En ocasiones, es posible que los servicios que se tengan contratados ofrezcan algún tipo de promoción o descuento. Es posible registrar estas ofertas en Subsify directamente desde la vista de detalle.

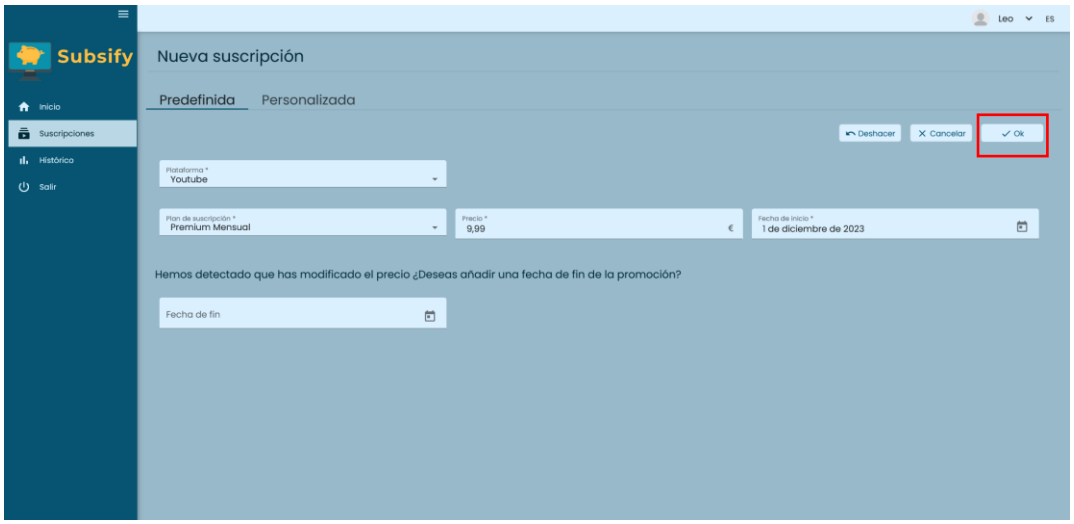
Haciendo clic en el botón "Nuevo" de la tabla de promociones, se abrirá un modal donde se podrá ingresar la fecha de inicio de la promoción, el precio a pagar y una fecha de fin. Esta última es opcional, lo que permite registrar promociones que duran un tiempo indefinido.


*26 Añadir promociones temporales*

## Añadir suscripción

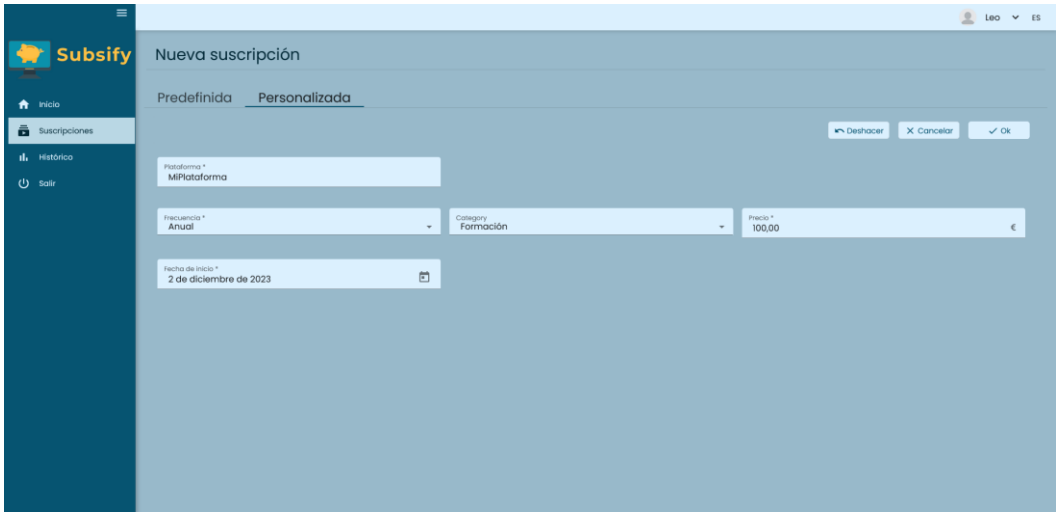
Desde el listado de suscripciones que se ve en la imagen 23, se puede añadir una nueva suscripción haciendo clic en el botón *"Nuevo"*. Aparecerá un formulario en donde se deberán rellenar los datos de la suscripción que se quiere dar de alta. Por defecto, Subsify sugiere el precio del servicio que se quiere registrar. Este precio puede editarse en caso de que el usuario se haya

beneficiado de alguna promoción o descuento. En este caso, aparecerá la opción de introducir una fecha de fin de la promoción.




*27 Formulario para añadir nueva suscripción*

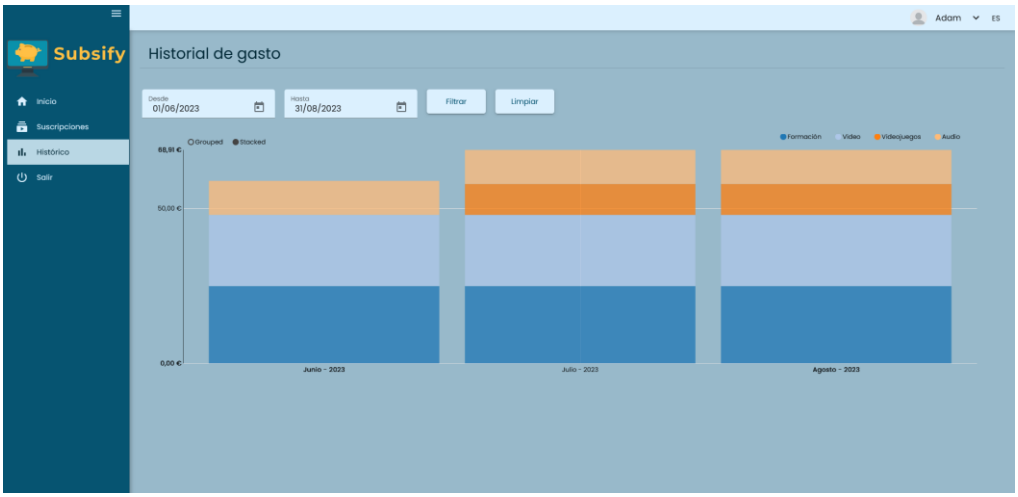
Si el usuario quiere añadir un servicio que no se encuentra registrado en la aplicación, podrá ir a la pestaña *"Personalizada"* y crear el servicio deseado rellenando los datos solicitados.




*28 Formulario para añadir suscripción personalizada*

## Ver histórico de gasto de meses anteriores

En el menú lateral, seleccionando la opción *"Histórico"*, se puede ver un gráfico con el gasto total por mes, agrupado además por categoría. Hay además un filtro que permite seleccionar las fechas que se desean consultar.



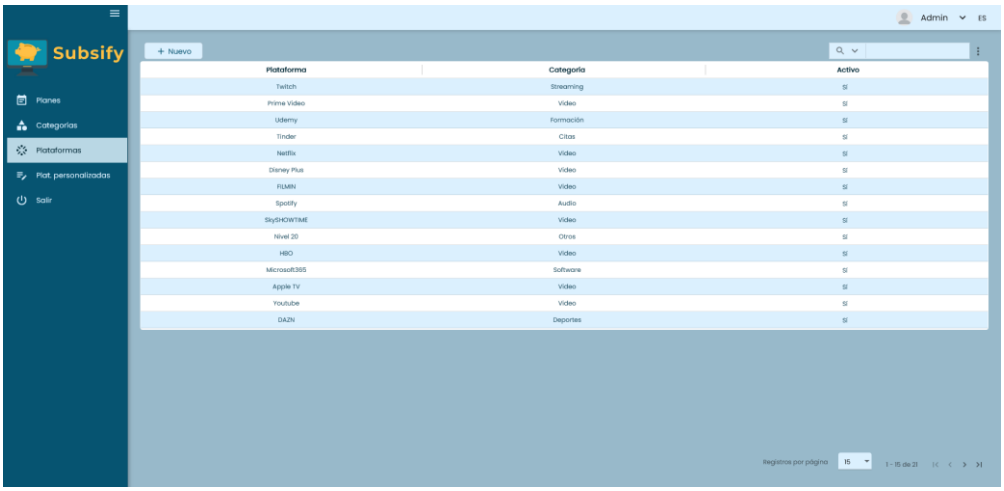

*29 Gráfica de gasto filtrada por fecha*

## 10.2 Administrador

**Ver/añadir plataformas, categorías o planes de precios registrados**

El administrador debe seleccionar la opción deseada en el menú lateral y le aparecerá un listado con todos los registros existentes. Haciendo clic en uno de ellos, podrá ver el detalle del mismo.

En todos los listados aparece un botón *"Nuevo".* Al hacer clic, aparece un formulario con los datos a rellenar para registrar el concepto deseado.

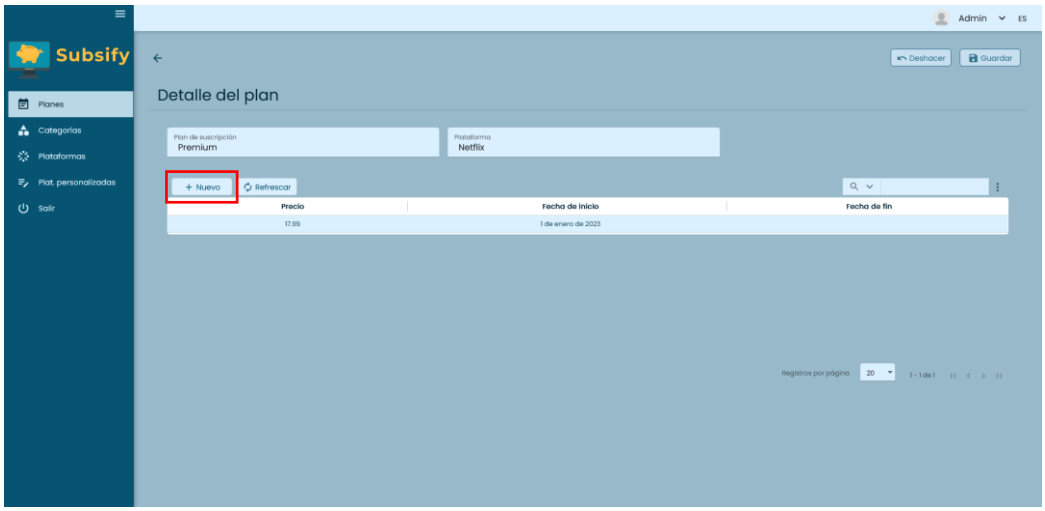



***30** Listado de plataformas registradas en Subsify*

**Actualizar planes de precio**

Al entrar en el detalle de un plan de precio, se puede ver una tabla con todos los cambios de precio que ha tenido ese plan a lo largo del tiempo. Para registrar uno nuevo, hay que hacer clic en el botón *"Nuevo"* y aparecerá un formulario en el que se deberá rellenar el nuevo precio y la

fecha en la que entrará en vigor. De esta forma, cuando el usuario tenga que renovar un servicio y vaya a entrar en vigor el nuevo precio, se le notificará el cambio.

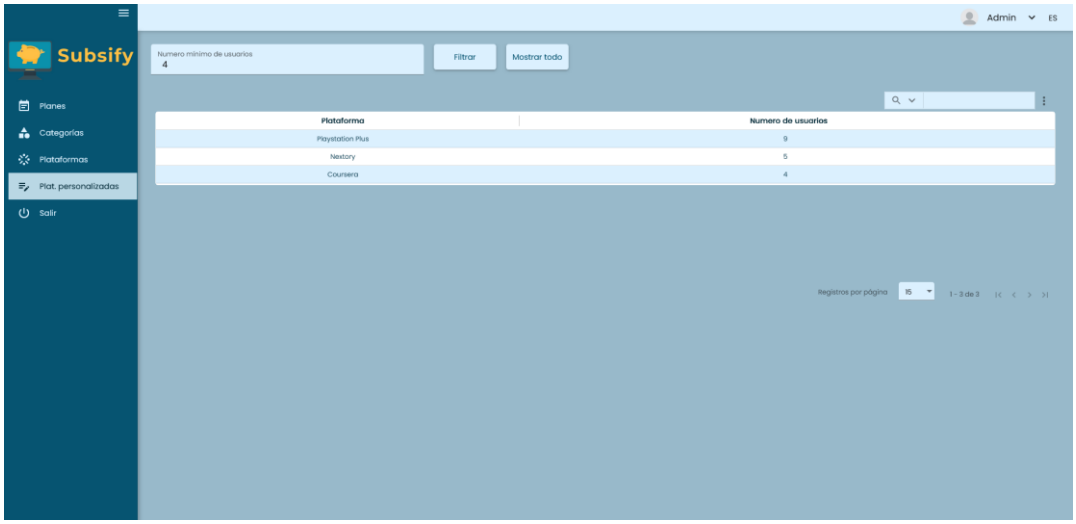



*31 Detalle de un plan de precio*

## Validar plataformas personalizadas

Seleccionando la opción del menú lateral *"Platf. Personalizadas"*, el administrador podrá ver un listado con todas las plataformas que han creado los usuarios. Podrá filtrarlo por número mínimo de usuarios para seleccionar aquellas que tienen mayor demanda.

Para validarla, deberá hacer clic sobre la plataforma deseada, que le redirigirá hacia el formulario de creación de nuevas plataformas y así poder registrarla en el sistema para que esté disponible para todos los usuarios.



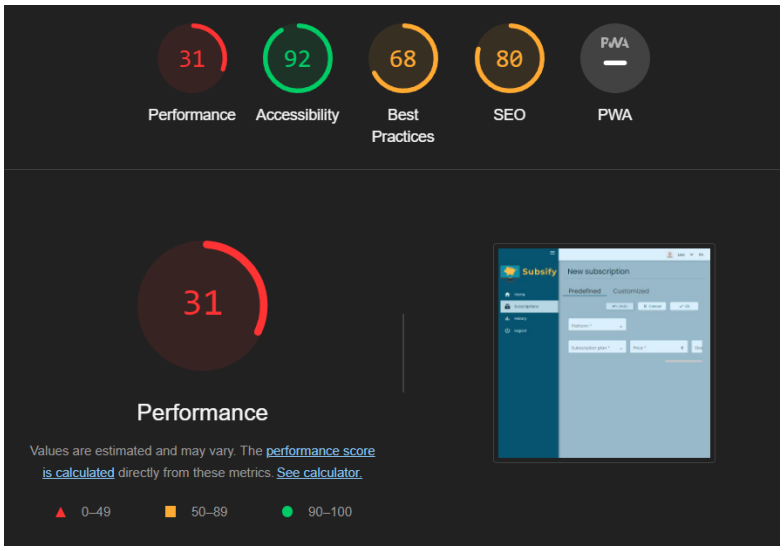

*32 Listado de plataformas  personalizadas con al menos 4 usuarios*

# 11. Pruebas realizadas y resultados

Todos los formularios están validados y se ha comprobado que efectivamente el usuario recibe mensajes de error en un cuadro de diálogo en caso de que no haya introducido los datos adecuados.

La aplicación se ha probado también en diferentes navegadores y se ha podido comprobar que funciona correctamente en los navegadores más utilizados (Chrome, Mozilla, Opera y Edge).

Además, se ha hecho un reporte utilizando la herramienta de **Google LightHouse** y se han obtenido los siguientes resultados:




*33 Reporte de Lighthouse*

Los resultados son similares en todas las páginas de la web. Aunque se puede observar que la puntuación en accesibilidad es adecuada, los problemas de rendimiento son evidentes. Esto se debe principalmente a la falta de optimización de **Ontimize Web**, ya que carga numerosas librerías y código innecesario.

Como se puede observar en la siguiente imagen, el tiempo de respuesta del servidor es superior a los 200ms, por lo que la carga de la página es lenta y afecta al rendimiento general de la web.

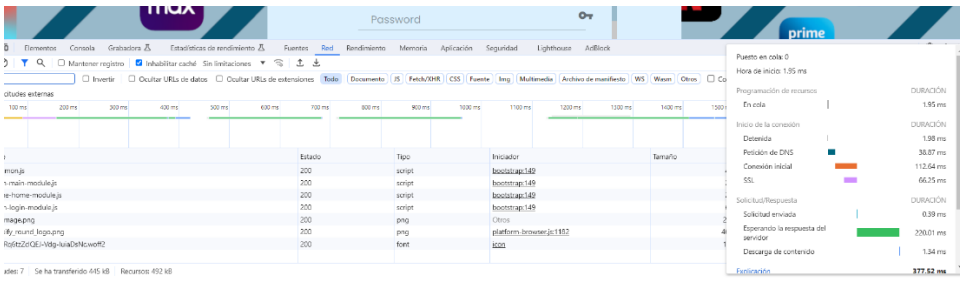


*Ilustración 34 Tiempos de respuesta del servidor*

## 12. Mejoras y líneas futuras

Como mejoras principales a implementar en la aplicación, destacan las siguientes:

- **Desarrollar versión móvil:** El uso de **Ontimize Web**, orientado principalmente al desarrollo de aplicaciones empresariales, no permite trabajar con un desarrollo *"mobile first"* y no ha sido posible desarrollarla por falta de tiempo.
- **Mejorar el rendimiento:** identificar todas las posibles causas que pueden afectar al rendimiento de la web para poder corregirlas y conseguir una navegación fluida.
- **Migración 100% a Angular:** permitiría incrementar notablemente la performance de la app y mejorar ciertos aspectos de accesibilidad.

Subsify se encuentra todavía en una etapa inicial de desarrollo y se pretende ir añadiendo las siguientes funcionalidades:

- **Tips de ahorro:** la aplicación propondrá oportunidades de ahorro al usuario, proponiendo, por ejemplo, servicios similares a precio más bajo. Informará también de aquellas suscripciones de las que se podría prescindir, como la posibilidad de cancelar Spotify si ya dispone de Amazon Prime, que incluye acceso a Amazon Music.
- **Tracking de uso:** el usuario podrá registrar el uso que hace de cada servicio que tiene contratado. Por ejemplo, en un servicio de video, el número de películas o series vistas en un mes. Esto permitirá a los usuarios ser conscientes de cuánto utilizan realmente un servicio y decidir de forma informada si mantener la suscripción activa.
- **Comunicación entre usuarios:** en la aplicación habrá la opción de seguir a otros usuarios y comunicarse entre ellos para compartir casos de uso de un servicio, alternativas, etc.
- **Integración de un sistema de reseñas:** permitir que los usuarios califiquen y dejen reseñas sobre los servicios que utilizan.
- **Estadísticas de uso de los usuarios:** el administrador tendrá la posibilidad de obtener datos agregados sobre el uso que hacen los usuarios de las plataformas registradas, por ejemplo, conocer el número de altas y bajas en una plataforma por mes, servicios con mayor número de suscriptores, etc.

## 13. Problemas encontrados

En el desarrollo de un proyecto de software es inevitable encontrarse con varios problemas que deben solventarse de la mejor manera posible. Durante el transcurso del desarrollo de Subsify, los principales han sido los siguientes:

- **Trabajar con tecnologías desconocidas:** el proyecto ha sido desarrollado con varias tecnologías de las que no se tenía ningún conocimiento previo. La forma de afrontarlo ha sido desarrollar pequeñas funcionalidades sin relación con el proyecto para entender cómo funcionan. Leer de forma detallada la documentación de estas tecnologías también fue clave en este proceso.
- **Diseño del modelo de datos:** el modelo de datos requerido para esta aplicación resultaba demasiado complejo como para diseñarlo al completo antes de iniciar el desarrollo. Por ese motivo se optó por diseñarlo de forma iterativa a medida que se iban desarrollando las funcionalidades. Esto provocaba un cierto trabajo extra, ya que cada vez que se modificaba el modelo de datos se rompía el funcionamiento ya implementado de la aplicación, aunque sin duda mereció la pena debido al aprendizaje y soltura adquiridas en el trabajo con base de datos.

  Otro problema relacionado con este era el manejo de las consultas a realizar, ya que al intervenir varias tablas resultaban demasiado complejas. La solución para simplificarlas fue el uso de la sentencia *`WITH`*, disponible en varios SGBD como PostgreSQL y que permite crear tablas temporales, evitando así el uso de subconsultas que son más complicadas de manejar y leer.

# 14. Conclusiones

Destacar que se han conseguido los objetivos indicados al inicio de la memoria y el resultado es una app, que aún en sus inicios, puede ofrecer un servicio útil a sus usuarios. Se han detectado varios puntos de mejora que se han de subsanar, pero en general el resultado es muy satisfactorio. Se partía de una idea sencilla, pero la sensación final es que se trata de una app que puede tener bastante potencial de desarrollo.

Debido al desconocimiento previo de muchas de las tecnologías y herramientas utilizadas, ha sido necesario dedicar bastante tiempo a investigar y entender cómo funcionan. Esto ha permitido un notable aumento en los conocimientos sobre desarrollo web, así como una mayor soltura en el manejo de Git y el trabajo con bases de datos.

La realización de este proyecto ha supuesto todo un resto por todo lo comentado anteriormente, pero ha sido una experiencia muy positiva e instructiva. Proporcionó la oportunidad de conocer cómo se trabaja en un entorno real, qué metodologías se utilizan y cómo es el proceso de desarrollar una aplicación desde cero.

Quiero expresar mi satisfacción por los conocimientos adquiridos, que serán, sin duda, de gran utilidad en el futuro y que me hacen sentir mucho más preparada para afrontar mi primer empleo en el sector.

# Bibliografía


- Documentación Ontimize Boot https://www.ontimize.com/xwiki/bin/view/Ontimize+Boot+Training
- Documentación Ontimize Web - https://ontimizeweb.github.io/docs/v4
- API Java 11 - https://docs.oracle.com/en/java/javase/11/docs/api/index.html
- Documentación de Spring - https://docs.spring.io/spring-framework/reference/index.html
- Baeldung (web con numerosa información sobre Java y Spring) - https://www.baeldung.com/
- MDN - https://developer.mozilla.org/es/
- Documentación TypeScript - https://www.typescriptlang.org/docs/
- Documentación Angular - https://angular.io/docs

# ANEXO - Enlaces de acceso y usuarios de prueba

**URL de la aplicación**

Si la aplicación tarda en cargar, intentarlo de nuevo en una pestaña de incógnito

https://subsify.netlify.app

**Repositorio de Github**

https://github.com/NoaSalgado/Subsify

**Script para la creación de la base de datos**

https://gist.github.com/NoaSalgado/b1147b511eaf967a9b762ecc65085743

**Diagramas**

Desde este enlace se puede acceder al documento de creación de los diagramas mostrados en el apartado 7 de la memoria. Es necesario tener sesión iniciada en Google.

https://drive.google.com/file/d/1VaEqFtEFxoCQGcFr3YpqFwi0jOsUDv7M/view

**Presentación**

https://www.canva.com/design/DAF2tZzrD7A/9KRy-ceTr5WlSqzN6-6llg/edit?utm_content=DAF2tZzrD7A&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

**Usuarios de prueba**

- **Usuarios:** Leo, Adam
- **Administrador:** Admin
- **Contraseña:** demouser (es la misma para los 3 usuarios)