



APLICACIÓN ANDROID PARA LA GESTIÓN DE SERVICIOS DE SUSCRIPCIÓN

Por Luis Sánchez Mateos

ÍNDICE

1. - Sobre este proyecto.....	3
1.1 - Control de versiones.....	3
2. - Análisis del problema.....	3
2.1 - Introducción al problema.....	3
2.2 - Antecedentes.....	4
2.3 - Objetivos.....	4
2.4 - Requisitos.....	4
2.4.1 - Funcionales.....	4
2.4.2 - No funcionales.....	5
2.5 - Recursos.....	5
2.5.1 - Software.....	5
2.5.2 - Hardware.....	5
3. - Diseño de la solución software.....	6
3.1 - Prototipado gráfico.....	6
3.1.1 - Tablets / Smartphones.....	6
3.2 - Base de datos.....	7
3.2.1 - Diseño Conceptual (ER).....	7
3.2.2 - Modelo relacional.....	7
4 - Implementación.....	7
4.1 - Usabilidad.....	8
4.2 Backend.....	13
4.3 - Frontend.....	17
4.4 - Pruebas.....	17
5 - Documentación.....	18
5.1 - Empaquetado / Distribución.....	18
5.2 - Instalación.....	18
5.3 - Manual de Usuario / Referencia.....	19
6 - Conclusiones.....	22
7 - Bibliografía.....	22

1. - Sobre este proyecto

La aplicación desarrollada es un gestor de servicios de suscripción para dispositivos Android, diseñado para ayudar a los usuarios a administrar y controlar sus gastos en suscripciones mensuales. Con esta app, los usuarios pueden agregar sus diversos servicios de suscripción, como Netflix, HBO, revistas, entre otros, y monitorear fácilmente su consumo en cada uno. La aplicación permite visualizar el tiempo restante de cada suscripción, percibir cuando una de ellas está a punto de expirar y recibir notificaciones frecuentes. Además de los servicios de suscripción, la aplicación ofrece la flexibilidad de gestionar otros tipos de pagos recurrentes, como compartir gastos de transporte, ya que no tiene ninguna limitación a la hora de agregar pagos personalizados. La tecnología utilizada incluye Android Studio para el desarrollo de la aplicación móvil, una API REST en Python para la comunicación con el backend, y MySQL en la nube para almacenar y gestionar los datos de los usuarios. La aplicación busca ofrecer una solución integral para el control financiero de los usuarios, adaptándose a sus necesidades individuales y brindando una experiencia intuitiva.

1.1 - Control de versiones

Se ha utilizado este repositorio de GitHub para controlar las versiones del software, documentación, etc. Ha sido incluido tanto el backend como el frontend.

2. - Análisis del problema

2.1 - Introducción al problema

En la actualidad, la suscripción a servicios digitales se ha convertido en una práctica común para millones de personas en todo el mundo. Sin embargo, la proliferación de estas suscripciones ha llevado a un problema creciente de gestión financiera para los usuarios. En un momento donde la estabilidad económica no es garantizada y los presupuestos personales son ajustados, es fácil perder de vista la cantidad total de dinero que se destina a diferentes servicios de suscripción. Esta falta de conciencia puede resultar en gastos innecesarios y en una acumulación de pagos que afecta negativamente la salud financiera del usuario. Por lo tanto, existe una necesidad urgente de una solución que permita a los usuarios controlar de manera efectiva sus gastos en suscripciones, asegurando una gestión más consciente y eficiente de su dinero.

2.2 - Antecedentes

Con el auge de los servicios de suscripción y la creciente complejidad de las finanzas personales en la era digital, las herramientas de gestión financiera han adquirido una importancia cada vez mayor. Si bien existen algunas aplicaciones disponibles en el mercado que abordan aspectos de la gestión de suscripciones, ninguna ofrece una solución integral y personalizable que se adapte a las necesidades específicas de los usuarios. La falta de una herramienta consolidada y versátil brinda la oportunidad para el desarrollo de una aplicación como la realizada en este proyecto.

2.3 - Objetivos

El objetivo principal de esta aplicación es proporcionar a los usuarios una herramienta efectiva para gestionar y controlar sus gastos en servicios de suscripción, así como otros pagos recurrentes, promoviendo una mayor conciencia financiera y ayudando a optimizar el uso de sus recursos económicos. Además, la aplicación busca ofrecer una UX intuitiva que garantice la adaptabilidad a las necesidades individuales de cada usuario. A través de funciones como el seguimiento del tiempo restante de las suscripciones y la recepción de notificaciones oportuna, se pretende facilitar la toma de decisiones informadas en cuanto a la renovación o cancelación de servicios, contribuyendo así a una gestión más eficiente y responsable de las finanzas personales.

2.4 - Requisitos

2.4.1 - Funcionales

En cuanto a requisitos funcionales, a nivel general, han sido incluidos:

- Registro de usuarios: la aplicación permite a los usuarios crear una cuenta personalizada para acceder a todas las funcionalidades.
- Agregación de servicios de suscripción: los usuarios pueden agregar y gestionar una variedad de servicios de suscripción, así como otros pagos recurrentes, según sus necesidades.
- Seguimiento del gasto: la aplicación advierte al usuario cuando queda menos de una semana para que una suscripción expire, siempre y cuando no sea un pago de periodicidad semanal.

- Creación de recordatorios: usando la consola de Google Firebase, se generan notificaciones personalizadas a modo de campañas, recordatorios, etc.

2.4.2 - No funcionales

Como requisitos no funcionales:

- Usabilidad: la interfaz de usuario es simple, intuitiva y fácil de navegar, garantizando una experiencia fluida para usuarios de todos los niveles de habilidad.
- Rendimiento: La aplicación es rápida y eficiente en su funcionamiento, con cortos tiempos de carga y capacidad para manejar grandes volúmenes de datos sin problemas.
- Disponibilidad: La aplicación está disponible para su uso en todo momento, con un tiempo de inactividad mínimo y una alta disponibilidad del servicio.

2.5 - Recursos

2.5.1 - Software

Las tecnologías usadas son las siguientes:

- Android Studio: se utiliza como entorno de desarrollo integrado (IDE) para la creación de la aplicación móvil compatible con dispositivos Android.
- Python: se emplea para el desarrollo del backend de la aplicación, aprovechando su versatilidad.
- Flask: se utiliza este framework de Python para la creación de una API RESTful que permita la comunicación entre el frontend y el backend de la aplicación.
- MySQL: se empleará como sistema de gestión de base de datos relacional para almacenar y gestionar los datos de los usuarios y sus suscripciones.
- Git: se utiliza para el control de versiones del código fuente, facilitando el seguimiento de los cambios realizados en el proyecto.
- Firebase Cloud Messaging: esta plataforma de mensajería en la nube se usa para enviar notificaciones push a los usuarios en tiempo real, programadas, frecuentes...

2.5.2 - Hardware

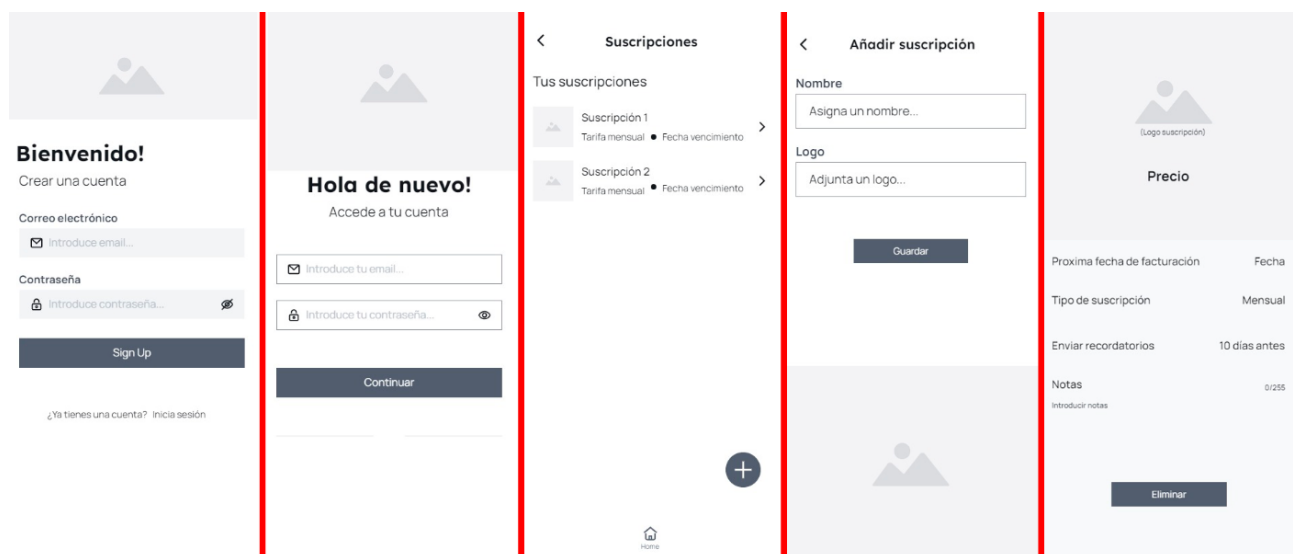
Muy simple. Un terminal Android con la capacidad de almacenamiento suficiente para instalar la aplicación. No se requiere de un dispositivo potente.

3. - Diseño de la solución software

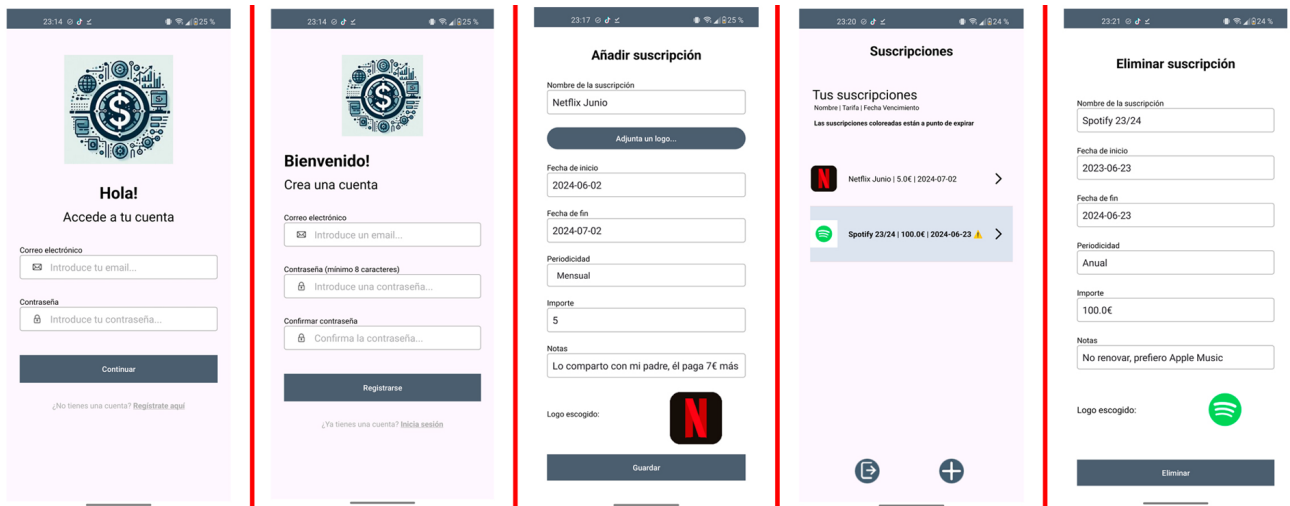
3.1 - Prototipado gráfico

3.1.1 - Tablets / Smartphones

Este fue el diseño inicial pensado.

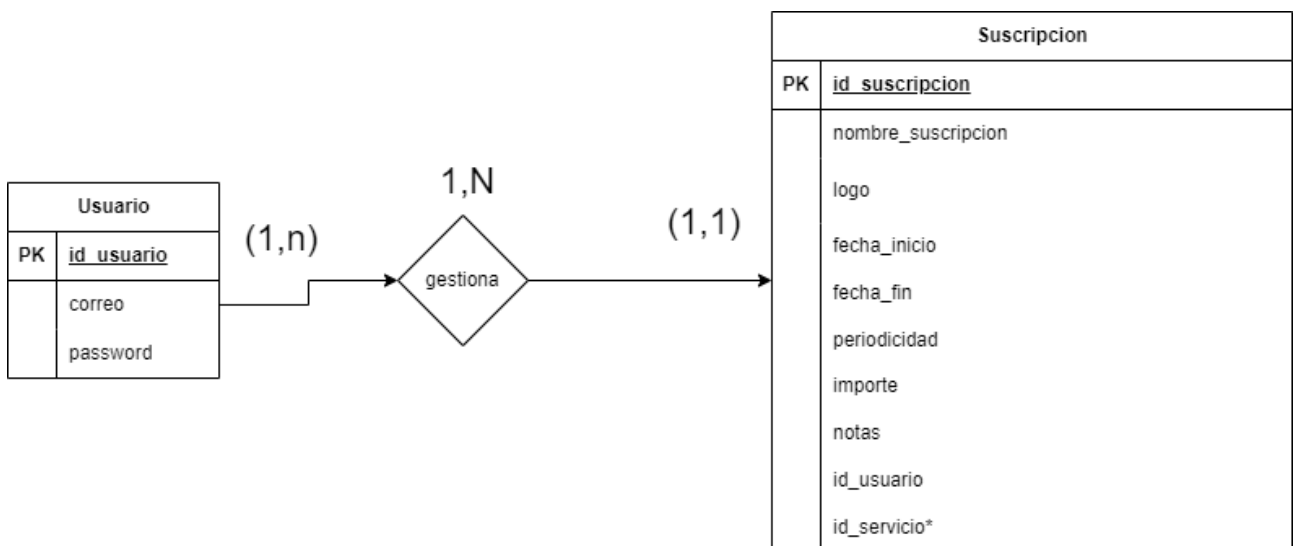


Esta es la versión final.



3.2 - Base de datos

3.2.1 - Diseño Conceptual (ER)



3.2.2 - Modelo relacional

En la versión actual sólo se están utilizando dos tablas:

Usuario: id_usuario, correo, password.

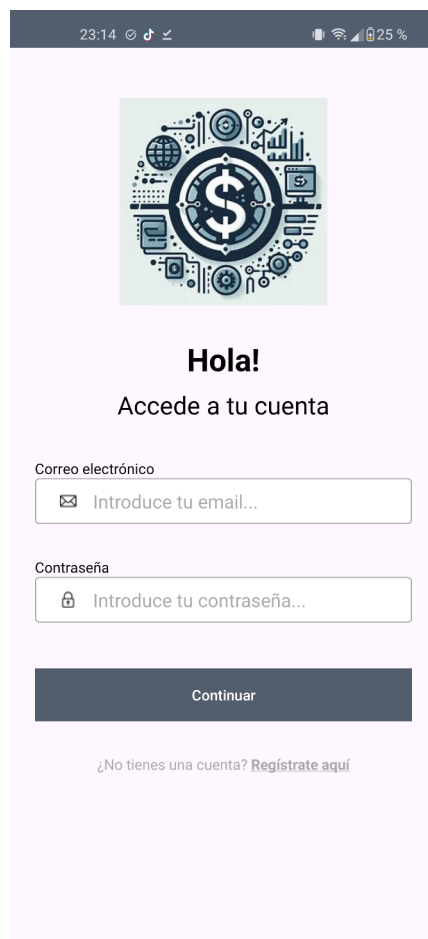
Suscripcion: id_suscripcion, nombre_suscripcion, logo, fecha_inicio, fecha_fin, periodicidad, importe, notas, id_usuario, id_servicio*.

*Están implementadas diversas tablas en la base de datos y elementos del backend relativos a una futura actualización de la aplicación más compleja.

4 - Implementación

4.1 - Usabilidad

Al iniciar la aplicación, el usuario se encuentra con la pantalla de login. En esta se muestra el logo de la aplicación junto a dos campos de texto en los que puede introducir el correo y la contraseña de su cuenta, en caso de tenerla. De no ser el caso, tiene la opción de dirigirse a la pantalla de registro pulsando en el texto inferior.



Si no tiene cuenta y se quiere registrar, debe introducir un correo válido y una contraseña con un mínimo de 8 caracteres. Se le obliga a confirmar la contraseña, para que se asegure de configurar aquella con la que esté conforme.



23:14 25 %



Bienvenido!

Crea una cuenta

Correo electrónico

Introduce un email...

Contraseña (mínimo 8 caracteres)

Introduce una contraseña...

Confirmar contraseña

Confirma la contraseña...

Registrarse

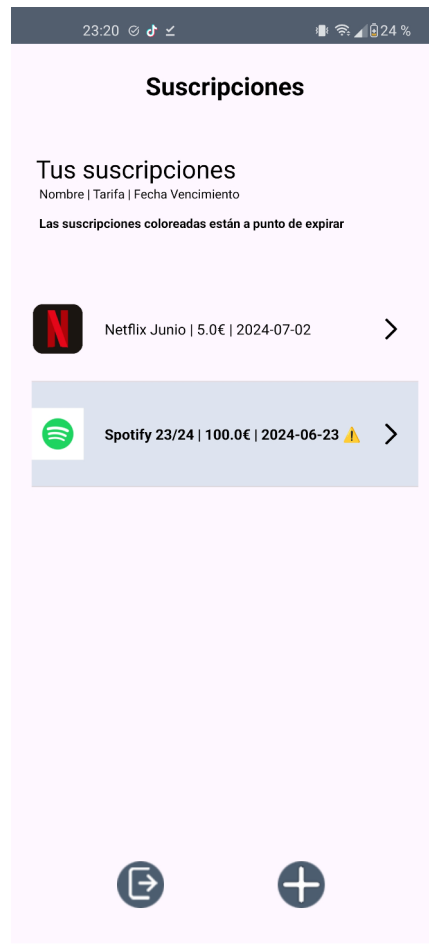
¿Ya tienes una cuenta? [Inicia sesión](#)

Si ha introducido los datos correctamente, automáticamente se redirige al usuario a su listado de suscripciones. En ella se indica que las suscripciones coloreadas están a punto de expirar.

Se consideran "a punto de expirar" aquellas suscripciones que vencen en menos de una semana y no tienen configuradas una periodicidad de tipo semanal.

En este listado, son visibles el logo, el nombre, el importe y la fecha de vencimiento de cada suscripción.

Se presentan en la parte inferior de la pantalla dos botones. El de la parte izquierda cierra la sesión del usuario mientras que el de la derecha le redirige a la pantalla de añadido de suscripción.



En la pantalla "Añadir suscripción", el usuario debe rellenar todos los campos debidamente, siendo la fecha de vencimiento posterior a la fecha de inicio, escogiendo una periodicidad (aunque puede elegir la opción "No"). En cuanto adjunta un logo se le muestra una previsualización para que compruebe por él mismo si le parece conveniente.

23:17 25 %

Añadir suscripción

Nombre de la suscripción

Netflix Junio

Adjunta un logo...

Fecha de inicio

2024-06-02

Fecha de fin

2024-07-02

Periodicidad

Mensual


Importe

5

Notas

Lo comparto con mi padre, él paga 7€ más

Logo escogido:



Guardar

Si pulsa dentro de una suscripción en el listado anterior, se le redirige al detalle de esa suscripción, donde se le muestra todos los atributos o elementos de la suscripción y se le da la opción de eliminarla. En caso de pulsar el botón de eliminar, aparece una ventana de confirmación para que no existan eliminaciones accidentales.

23:21 24 %

Eliminar suscripción

Nombre de la suscripción
Spotify 23/24

Fecha de inicio
2023-06-23

Fecha de fin
2024-06-23

Periodicidad
Anual

Importe
100.0€

Notas
No renovar, prefiero Apple Music

Logo escogido: 

Eliminar

0:22 20 %

Eliminar suscripción

Nombre de la suscripción
Spotify 23/24

Fecha de inicio
2023-06-23

Fecha de fin
2024-06-23


Eliminar suscripción

¿Estás seguro de que quieres eliminar esta suscripción?

CANCELAR CONFIRMAR

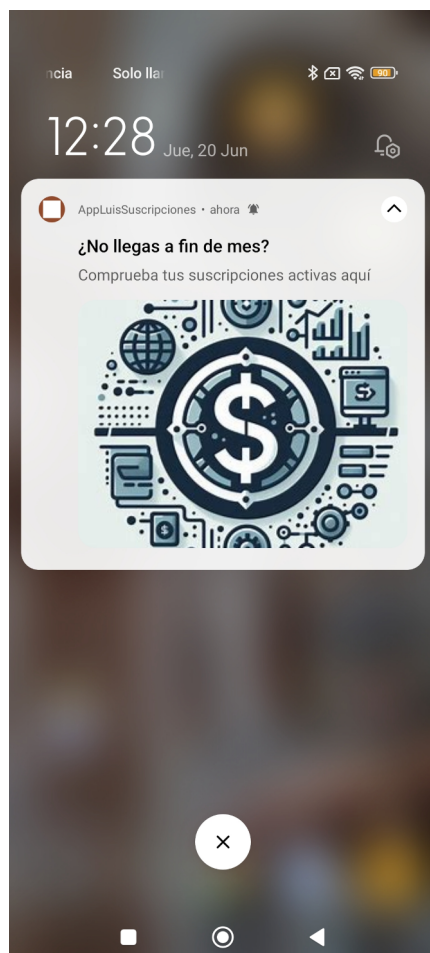
100.0€

Notas
No renovar, prefiero Apple Music

Logo escogido: 

Eliminar

Ocasionalmente, el usuario recibirá notificaciones push que, tras pulsar sobre ellas, le redirigirá a la aplicación.



Cabe destacar que cada vez que el usuario introduce datos no válidos en una pantalla, enseguida le aparece un error indicándoselo.


Si el usuario ha iniciado sesión con su cuenta y ha cerrado la aplicación, no deberá volver a introducir sus credenciales ya que la aplicación hace uso de SharedPreferences, en pos de su comodidad. Al volver a abrir la app se encuentra con el listado de sus suscripciones.

4.2 - Backend

Se ha usado Flask y SQLAlchemy para la API Rest en PythonAnywhere. De esta forma se proveen una serie de endpoints a los que hacer peticiones y con los que poder interactuar con usuarios y suscripciones en la base de datos de MySQL utilizada.

La estructura seguida ha sido la siguiente:

-Archivos de configuración donde se especifican la base de datos a utilizar, credenciales, versiones, endpoints...

/home/LuisSanchezMat/  proyectoIntegrado [Open Bash console here](#)

Directories


[New directory](#)


[__pycache__/](#)


[migrations/](#)


[models/](#)

[resources/](#)


















Files




 [app.py](#)

 [config.py](#)

 [extensions.py](#)

2024-06-17 23:13

1.9 KB

2024-06-16 09:21

276 bytes

2024-06-16 09:26

155 bytes

[Upload a file](#)

100MiB maximum size



/home/LuisSanchezMat/proyectoIntegrado/app.py

Keyt

```
23 app.config.from_object(config)
24 register_extensions(app)
25 register_resources(app)
26 return app
27
28 def register_extensions(app):
29     db.init_app(app)
30     migrate = Migrate(app, db)
31
32 def register_resources(app):
33     api = Api(app)
34
35     api.add_resource(SuscripcionListResource, '/suscripciones')
36     api.add_resource(SuscripcionResource, '/suscripciones/<int:id_suscripcion>')
37
38
39
40     api.add_resource(UsuarioListResource, '/usuarios')
41     api.add_resource(UsuarioResource, '/usuarios/<int:id_usuario>')
42     api.add_resource(PagoListResource, '/pagos')
43     api.add_resource(PagoResource, '/pagos/<int:id_pago>')
44     api.add_resource(ServicioListResource, '/servicios')
45     api.add_resource(ServicioResource, '/servicios/<int:id_servicio>')
46     api.add_resource(LoginResource, '/login')
47     api.add_resource(SuscripcionesUsuarioResource, '/usuarios/<int:id_usuario>/suscripciones')
48     api.add_resource(SuscripcionEditResource, '/usuarios/<int:id_usuario>/suscripciones/<int:id_suscripcion>')
49
50
51
52 app = create_app()
53
54 @app.shell_context_processor
55 def make_shell_context():
56     return dict(db=db, Usuario=Usuario, Pago=Pago, Suscripcion=Suscripcion, Servicio=Servicio)
57
58 if __name__ == '__main__':
59     app.run()
```

-Modelos, que representan tablas de la base de datos con sus atributos y métodos:

/home/LuisSanchezMat/proyectoIntegrado/ models

[Open Bash console here](#)

Directories

Enter new directory name

[New directory](#)

[__pycache__](#)



Files

Enter new file name, eg hello.py

[pago.py](#)

2024-06-16 11:36 1.3 KB

[servicio.py](#)

2024-06-16 10:19 1.2 KB

[suscripcion.py](#)

2024-06-16 22:37 2.2 KB

[usuario.py](#)

2024-06-16 18:24 1.8 KB

[Upload a file](#)

100MiB maximum size



/home/LuisSanchezMat/proyectoIntegrado/models/usuario.py

```
1 from extensions import db
2 from models.suscripcion import Suscripcion
3
4 class Usuario(db.Model):
5     __tablename__ = 'Usuario'
6
7     id_usuario = db.Column(db.Integer, primary_key=True)
8     correo = db.Column(db.String(100), nullable=False)
9     password = db.Column(db.String(255))
10
11     @classmethod
12     def get_by_correo(cls, correo):
13         """
14         Busca un usuario con el correo pasado como parámetro.
15         Retorna un objeto Usuario si lo encuentra, None si no.
16         """
17         return cls.query.filter_by(correo=correo).first()
18
19     @classmethod
20     def get_by_id_usuario(cls, id_usuario):
21         """
22         Busca un usuario con el id pasado como parámetro.
23         Retorna un objeto Usuario si lo encuentra, None si no.
24         """
25         return cls.query.filter_by(id_usuario=id_usuario).first()
26
27     def get_suscripciones(self):
28         """
29         Retorna las suscripciones asociadas a este usuario.
30         """
31         return Suscripcion.query.filter_by(id_usuario=self.id_usuario).all()
32
33     def guardar(self):
34         """
35         Envía el usuario a la base de datos.
36         """
37         db.session.add(self)
38         db.session.commit()
39
40     def borrar(self):
41         """
42         Borra el usuario de la base de datos.
43         """
44         db.session.delete(self)
45         db.session.commit()
46
47     def verificar_password(self, password):
48         """
49         Verifica si la contraseña proporcionada coincide con la almacenada.
```

-Recursos, donde se gestionan las acciones una vez se reciben peticiones de X tipo (POST, GET, PUT, etc):

/home/LuisSanchezMat/proyectoIntegrado/  resources

 [Open Bash console here](#)

















Directories

[New directory](#)

[__pycache__/](#)



Files

 recursosPago.py	  	2024-06-16 11:04	2.5 KB
 recursosServicio.py	  	2024-06-16 11:04	2.1 KB
 recursosSuscripcion.py	  	2024-06-17 18:56	3.3 KB
 recursosUsuario.py	  	2024-06-17 23:16	8.2 KB

[Upload a file](#)

100MiB maximum size

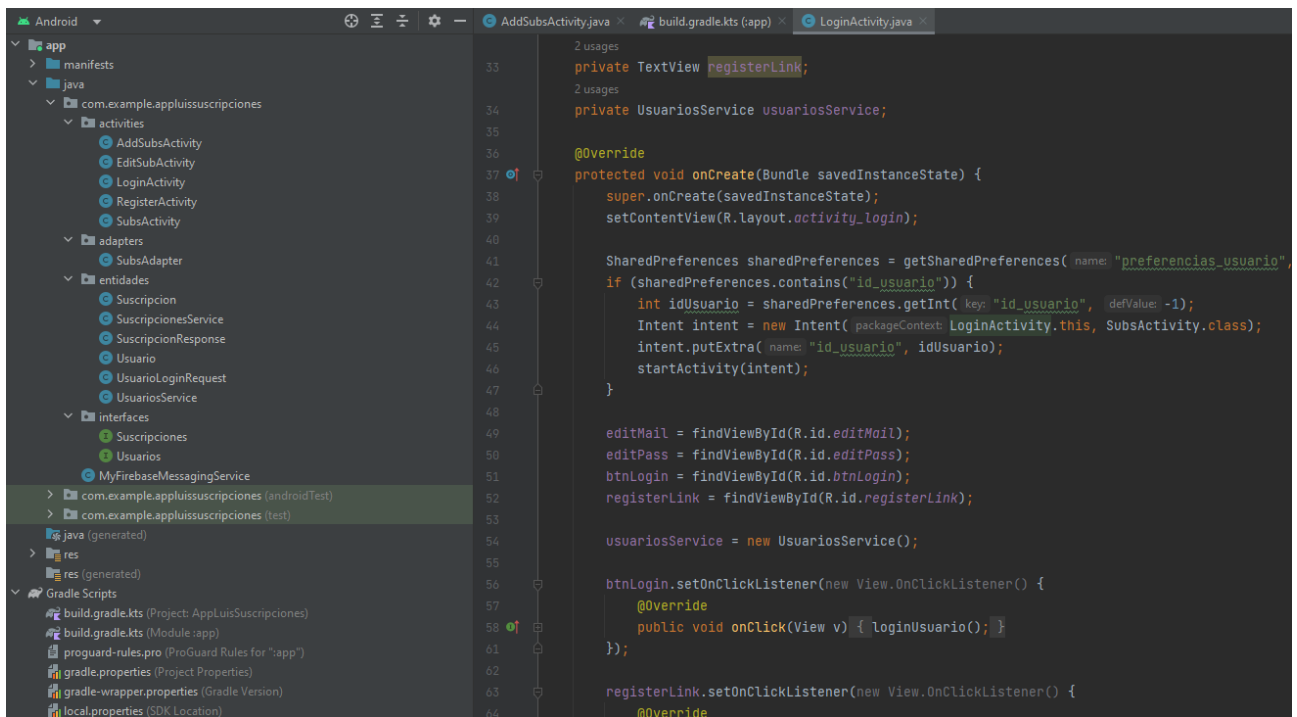


```
1 from flask import request
2 from flask_restful import Resource
3 from http import HTTPStatus
4 from models.suscripcion import Suscripcion
5
6 class SuscripcionListResource(Resource):
7     """ Responde a las peticiones /suscripciones """
8
9     def get(self):
10         """ Respuesta al método HTTP GET. Retorna todas las suscripciones """
11         suscripciones = Suscripcion.query.all()
12         datos = [suscripcion.data for suscripcion in suscripciones]
13         return {'data': datos}, HTTPStatus.OK
14
15     def post(self):
16         """
17         Respuesta al método HTTP POST.
18         Espera recibir los datos de una suscripción para guardarla.
19         Retorna la suscripción creada.
20         """
21         datos = request.get_json()
22
23         nombre_suscripcion = datos.get('nombre_suscripcion')
24         id_usuario = datos.get('id_usuario')
25         if Suscripcion.query.filter_by(nombre_suscripcion=nombre_suscripcion, id_usuario=id_usuario).first():
26             return {'message': 'Ya existe una suscripción con ese nombre.'}, HTTPStatus.BAD_REQUEST
27
28         suscripcion = Suscripcion(nombre_suscripcion=nombre_suscripcion,
29                                   logo=datos.get('logo'),
30                                   fecha_inicio=datos.get('fecha_inicio'),
31                                   fecha_fin=datos.get('fecha_fin'),
32                                   periodicidad=datos.get('periodicidad'),
33                                   importe=datos.get('importe'),
34                                   notas=datos.get('notas'),
35                                   id_usuario=datos.get('id_usuario'),
36                                   id_servicio=datos.get('id_servicio'))
37         suscripcion.guardar()
38
39         return suscripcion.data, HTTPStatus.CREATED
40
41 class SuscripcionResource(Resource):
42     """ Responde a las peticiones /suscripciones/<int:id_suscripcion> """
```

4.3 - Frontend

Android Studio provee de infinidad de opciones a la hora de realizar diseños y funciones complejas, en varios lenguajes. Se ha utilizado el lenguaje Java, para los activities y clases relacionadas, y XML para los layouts, configuraciones de textos, etc.

La estructura llevada a cabo es la siguiente:



Se ven diferenciados los activities, las entidades y clases relativas a la comunicación con la API, sus interfaces, el adapter personalizado para el listado de suscripciones y la clase para Firebase.

Se ha utilizado Retrofit2 para las peticiones a la API.

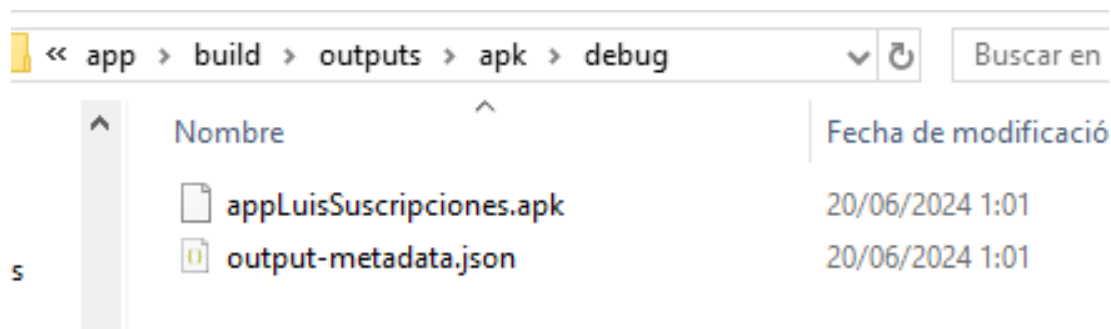
4.4 - Pruebas

Las pruebas han sido a nivel de usuario, probando a introducir datos no válidos, imágenes con resoluciones muy grandes para ver si saturaba la red, etc.

5 - Documentación

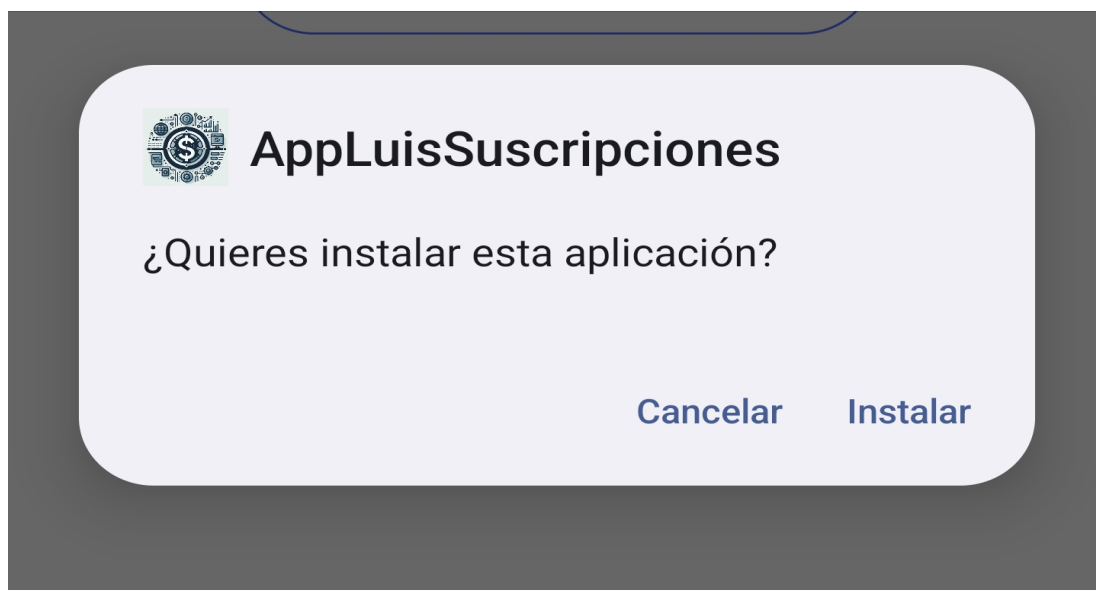
5.1 - Empaquetado / Distribución

Al tratarse de una aplicación de Android, la distribución se lleva a cabo mediante un archivo .apk que se difunde, ya sea en el Google Play Store o, de forma externa. Para generar ese archivo .apk basta con pulsar en Build > Build Bundle(s) / APK(s) > Build APK(s). En la ruta app\build\outputs\apk\debug aparece el archivo .apk, al que se le puede cambiar el nombre.



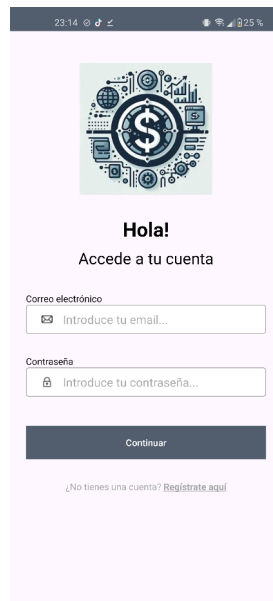
5.2 - Instalación

Una vez el usuario tiene el archivo.apk a su disposición, simplemente tiene que descargarlo y usar el instalador de paquetes de su teléfono móvil. Previamente deberá autorizar la instalación de aplicaciones provenientes de fuentes externas (este proceso aparece al intentar instalarlo).



5.3 - Manual de Usuario / Referencia

Al iniciar la aplicación, usted se encuentra con la pantalla de login. En esta se muestra el logo de la aplicación junto a dos campos de texto en los que puede introducir el correo y la contraseña de su cuenta, en caso de tenerla. De no ser el caso, tiene la opción de dirigirse a la pantalla de registro pulsando en el texto inferior.



Si no tiene cuenta y se quiere registrar, debe introducir un correo válido y una contraseña con un mínimo de 8 caracteres. Se le obliga a confirmar la contraseña, para que se asegure de configurar aquella con la que esté conforme.

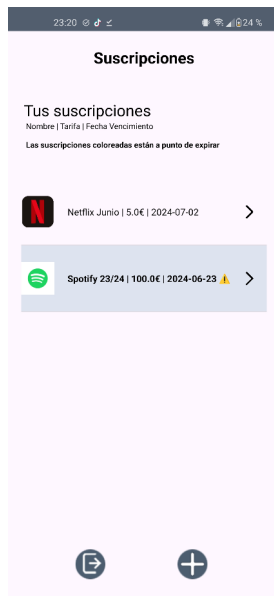


Si ha introducido los datos correctamente, automáticamente usted es redirigido a su listado de suscripciones. En ella se indica que las suscripciones coloreadas están a punto de expirar.

Se consideran "a punto de expirar" aquellas suscripciones que vencen en menos de una semana y no tienen configuradas una periodicidad de tipo semanal.

En este listado, son visibles el logo, el nombre, el importe y la fecha de vencimiento de cada una de sus suscripciones.

Se presentan en la parte inferior de la pantalla dos botones. El de la parte izquierda cierra la sesión de su cuenta mientras que el de la derecha le redirige a la pantalla de añadido de suscripción.

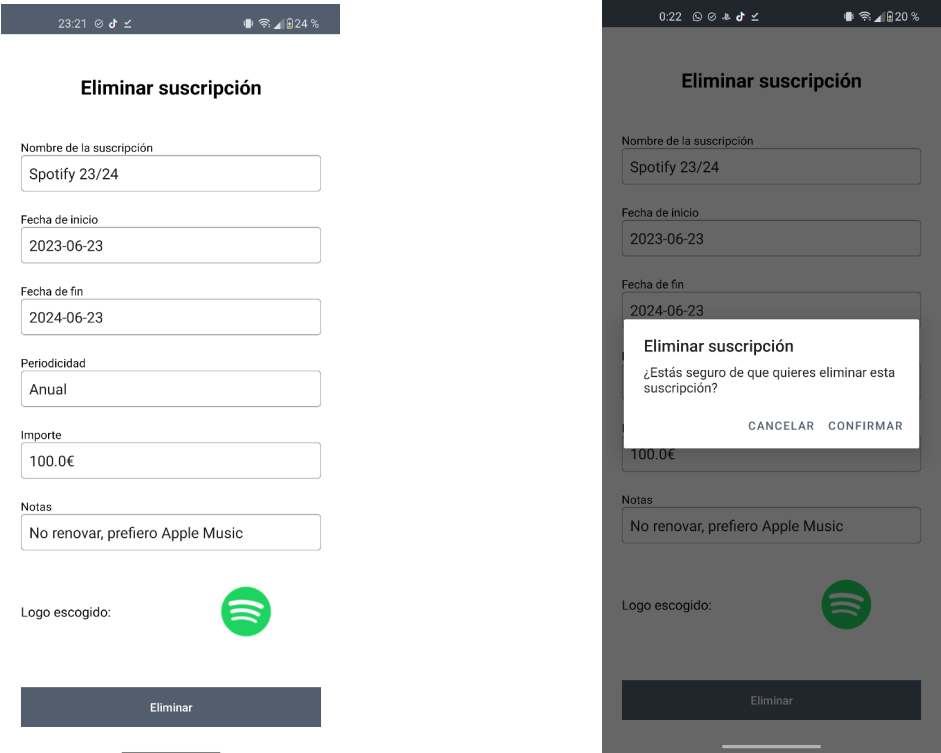


En la pantalla "Añadir suscripción", usted debe rellenar todos los campos debidamente, siendo la fecha de vencimiento posterior a la fecha de inicio, escogiendo una periodicidad (aunque puede elegir la opción "No"). En cuanto adjunta un logo se le muestra una previsualización para que compruebe por sí mismo si le parece conveniente.

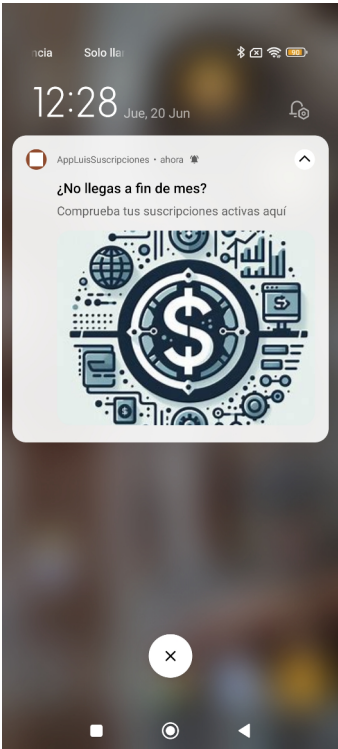
A screenshot of a mobile application screen titled "Añadir suscripción". The screen has a light pink background. At the top, there's a status bar showing the time 23:17 and battery level 25%. Below the title, there's a form with several fields: "Nombre de la suscripción" with the value "Netflix Junio", "Adjunta un logo..." with a button, "Fecha de inicio" with the value "2024-06-02", "Fecha de fin" with the value "2024-07-02", "Periodicidad" with the value "Mensual", "Importe" with the value "5", and "Notas" with the value "Lo comparto con mi padre, él paga 7€ más". Below the form, there's a section "Logo escogido:" with a preview of the Netflix logo. At the bottom of the screen, there's a "Guardar" button.

Si pulsa dentro de una suscripción en el listado anterior, se le redirige al detalle de esa suscripción, donde se le muestra todos los atributos o elementos de la suscripción y se le da la opción de

eliminarla. En caso de pulsar el botón de eliminar, aparece una ventana de confirmación para que confime si efectivamente quiere borrarla.



Ocasionalmente, usted recibirá notificaciones push que, tras pulsar sobre ellas, le redirigirá a la aplicación.



Si usted ha iniciado sesión con su cuenta y ha cerrado la aplicación, no deberá volver a introducir sus credenciales ya que la aplicación hace uso de SharedPreferences, en pos de su comodidad. Al volver a abrir la app se encuentra con el listado de sus suscripciones.

6 - Conclusiones

Pienso que los objetivos que me marqué antes de empezar han sido cumplidos, al menos a grandes rasgos. Se me ocurren muchas funcionalidades que podrían mejorar la aplicación, pese a que ya sea funcional y me parezca útil para llevar el control de los gastos. Creo que satisface una necesidad que muchos tenemos o, al menos, en algún momento habríamos agradecido disponer de una app así para evitar ciertos pagos inesperados.

Me gustaría en un futuro, en mis tiempos libres, añadirle a la aplicación la posibilidad de compartir suscripciones con otros usuarios que ya se hayan instalado la app, un uso más avanzado de las notificaciones... de hecho, parte de la base de datos y backend ya están perfilados a estas nuevas funcionalidades, pero no se han llevado a cabo para esta versión.

En cualquier caso, acabo este proyecto con un buen sabor de boca, satisfecho por implementar una aplicación que puede ser más o menos compleja pero, indudablemente, completamente funcional.

7 - Bibliografía

-Manuales de ayuda para el uso de Flask, por Pablo Blanco en la plataforma Moodle.

-GeeksforGeeks, para el uso de ciertos elementos de Android.

-Stack Overflow y ChatGPT para salir de ciertos apuros que se han ido presentando a lo largo del desarrollo.