EDIX | MP_0492 Proyecto final
Desarrollo de Aplicaciones Multiplataforma

Dashboard de finanzas personales integrado con Google Sheets

Alumnos:

Pedro Vivo Filardi, Ricard Penin Honrubia, Roberto Bueno García.

Tutora:

Raquel Cerdá.

Especialidad del ciclo formativo:

Desarrollo de Aplicaciones Multiplataforma.

Instituto:

Instituto Tecnológico EDIX.

Introducción:

El trabajo actual llamado Dashboard de finanzas personales integrado con Google Sheets; tiene como objetivo desarrollar un sistema de gestión de finanzas personales que combine la potencia de las hojas de cálculo con un dashboard de reporting, superando las limitaciones gráficas, estéticas y funcionales de las hojas de cálculo tradicionales.

Es fundamental realizar una evaluación adecuada de los gastos e ingresos cuando se trata de administrar las finanzas personales. Históricamente, esta tarea se realizaba con libretas de papel. Con la digitalización, se comenzaron a utilizar hojas de cálculo, lo que permitió agilizar las operaciones y generar gráficos para un análisis más visual de las finanzas personales. Sin embargo, junto con el surgimiento de las hojas de cálculo, han surgido varias aplicaciones destinadas a facilitar la gestión financiera mediante interfaces gráficas más fáciles de usar. Sin embargo, muchas de estas aplicaciones son de pago o tienen limitaciones en cuanto a lo que pueden hacer.

Se plantea la necesidad de desarrollar un entorno híbrido que combine las ventajas de las hojas de cálculo con sus limitaciones en términos de diseño, apariencia y funcionalidad. El objetivo es crear un tablero de reportes que brinde una experiencia de gestión de finanzas personales más completa y satisfactoria. Debido a su naturaleza basada en la nube y su disponibilidad en múltiples plataformas, Google Sheets se utilizará como plataforma principal para lograrlo. Esto permitirá a los usuarios acceder y administrar sus finanzas personales desde cualquier dispositivo con acceso a internet. En resumen, el trabajo actual se centrará en crear un dashboard de finanzas personales que se integre con Google Sheets. El objetivo es brindar a los usuarios una herramienta más poderosa y flexible para controlar y analizar sus finanzas personales.

ÍNDICE

4. Módulos formativos

A continuación, se detallan los módulos y las competencias que se aplicarán en el proyecto:

Programación	Se desarrollará un script de parseo utilizando habilidades de programación avanzadas que permitirán procesar los datos de entrada de manera eficiente y precisa. Programación de Procesos y Servicios: - Los conceptos y métodos de programación de procesos y servicios se utilizarán para crear comunicarnos con una API, la de google auth.
Acceso a Datos	Se utilizarán métodos de acceso a datos programáticos para interactuar con la base de datos del sistema. Para establecer conexiones, hacer consultas y actualizar la información almacenada, será necesario utilizar los conectores y librerías adecuados.
Bases de Datos	Se llevará a cabo el modelado de datos necesario para mostrar la estructura de los datos en la base de datos. Además, se implementarán las consultas adecuadas para obtener y procesar datos de manera efectiva.
Desarrollo de Interfaces	Se utilizarán técnicas de desarrollo de interfaz de usuario para personalizar y mejorar la experiencia de usuario del sistema. Esto incluirá el diseño e implementación de elementos visuales, interacciones y flujos de trabajo fáciles de entender.
DevOps	Además de los módulos mencionados anteriormente, también se incluirán competencias relacionadas con DevOps, lo que permitirá que el proyecto se despliegue de manera eficiente y escalable. El procesamiento y la carga de datos se llevarán a cabo mediante tecnología de contenedores Docker y ETLs (Extract, Transform, Load).
Programación de servicios y procesos	Para utilizar Google Sheets API ha sido fundamental los conocimientos adquiridos en este módulo para comprender la utilización de sockets seguros mediante protocolos de comunicación en la red para acceder a servicios web REST

5. Herramientas/Lenguajes utilizados:

Streamlit

Streamlit es una biblioteca de código abierto para crear aplicaciones web interactivas en Python. Está diseñada para simplificar el proceso de desarrollo y despliegue de aplicaciones basadas en datos y visualizaciones, permitiendo a los científicos de datos y desarrolladores crear interfaces de usuario interactivas de manera eficiente

Seleccionamos Streamlit por su facilidad de uso, rápido prototipado, integración con Python y ciencia de datos, personalización y control, y despliegue sencillo.

Cuando se utiliza, se comienza escribiendo código Python para cargar y manipular los datos que se mostrarán en la aplicación. Luego, se agregan elementos de interfaz de usuario, como botones, controles deslizantes, campos de entrada, gráficos y tablas, utilizando funciones específicas de Streamlit.

A medida que se ejecuta el código, Streamlit actualiza automáticamente la interfaz de usuario para reflejar los cambios realizados en los datos o en la lógica de la aplicación. Esto permite una experiencia interactiva fluida, donde los usuarios pueden interactuar con la aplicación y ver los resultados actualizados en tiempo real.

Google Sheets API

La API de Google Sheets nos permite acceder y manipular hojas de cálculo de Google de forma programática con solicitudes HTTP. Esta API que requiere OAuth 2.0, te permite realizar diversas operaciones en una hoja de cálculo de Google. Por ejemplo, si queremos agregar una fila a una hoja de cálculo de Google, podemos enviar una solicitud HTTP POST a la API con los datos de la fila a agregar en formato JSON. La API procesará la solicitud y agregará la fila a la hoja de cálculo que deseamos.

¿Por qué Google Sheets API?

- Seleccionamos Google Sheets API debido a su fácil integración con Python y su capacidad para acceder y actualizar datos en tiempo real.
- Aprovechamos la escalabilidad y el almacenamiento en la nube del usuario.
- Google Sheets API ofrece seguridad y control de acceso para proteger los datos financieros sensibles y personales del usuario.
- Google nos proporciona un amplio conjunto de funcionalidades para leer, escribir y manipular datos en hojas de cálculo, ofreciendo gran escalabilidad en el futuro.
- La API nos permite centrarnos en el desarrollo de la aplicación sin preocuparnos por la infraestructura subyacente gracias a los scopes REST API.

PostgreSQL

Seleccionamos PostgreSQL como backend para administrar finanzas debido a su confiabilidad, estabilidad y capacidad de gestión de transacciones. Ofrece funcionalidades avanzadas, seguridad y cumplimiento normativo, respaldado por una comunidad activa.

PostgreSQL es un poderoso sistema de gestión de bases de datos relacionales (RDBMS) de código abierto. Fue creado por el Departamento de Ciencias de la Computación de la Universidad de California en Berkeley en 1986 y se ha convertido en una de las bases de datos más populares y avanzadas disponibles en la actualidad.

PostgreSQL es conocido por su capacidad para manejar grandes cantidades de datos y por su capacidad para soportar una amplia variedad de aplicaciones y casos de uso, incluyendo aplicaciones web, análisis de datos y almacenamiento de datos geoespaciales. Además, PostgreSQL es altamente personalizable y se puede utilizar en una amplia variedad de sistemas operativos, incluyendo Linux, Windows y macOS.

PostgreSQL es compatible con SQL estándar y también ofrece una amplia variedad de características avanzadas, como transacciones, integridad referencial, recuperación de desastres y replicación, lo que lo hace una excelente opción para aplicaciones críticas que requieren alta disponibilidad y confiabilidad.

Docker

Docker es una plataforma de contenedores que permite a los desarrolladores empaquetar y distribuir aplicaciones junto con todas sus dependencias y configuraciones necesarias en un paquete portátil y liviano llamado contenedor. Porque en lugar de virtualizar todo un sistema operativo como lo hace una máquina virtual, Docker utiliza la virtualización a nivel de sistema operativo para encapsular una aplicación y todas sus dependencias en un contenedor. Esto significa que los contenedores pueden compartir el mismo kernel del sistema operativo, lo que los hace más eficientes y rápidos que las máquinas virtuales.

Docker Compose

Docker Compose es una herramienta que permite definir y ejecutar aplicaciones Docker que consisten en varios contenedores Docker. Con Compose, se puede definir y configurar múltiples contenedores, cada uno con su propia configuración y dependencias, y luego ejecutarlos como una aplicación única y cohesiva.

Con Docker Compose, se pueden definir todos los servicios que una aplicación necesita en un archivo YAML, lo que hace que sea fácil de entender y administrar las dependencias y configuraciones de cada contenedor. Por ejemplo, se pueden definir contenedores para una base de datos, un servidor web y una aplicación en el mismo archivo YAML y luego ejecutarlos todos juntos. Además, Docker Compose permite escalar los servicios de la aplicación.

6. Componentes del equipo y aportación realizada

Ricard Penin Honrubia.:

- Diseño del modelo de datos: definición de las entidades que contiene el gestor.
- Creación de la plantilla de Microsoft Excel: También se encargó de crear una plantilla de Excel que se ajustará al modelo de datos que se diseñó. Esto permitirá a los usuarios capturar y organizar correctamente sus datos financieros.

Pedro Vivo Filardi.:

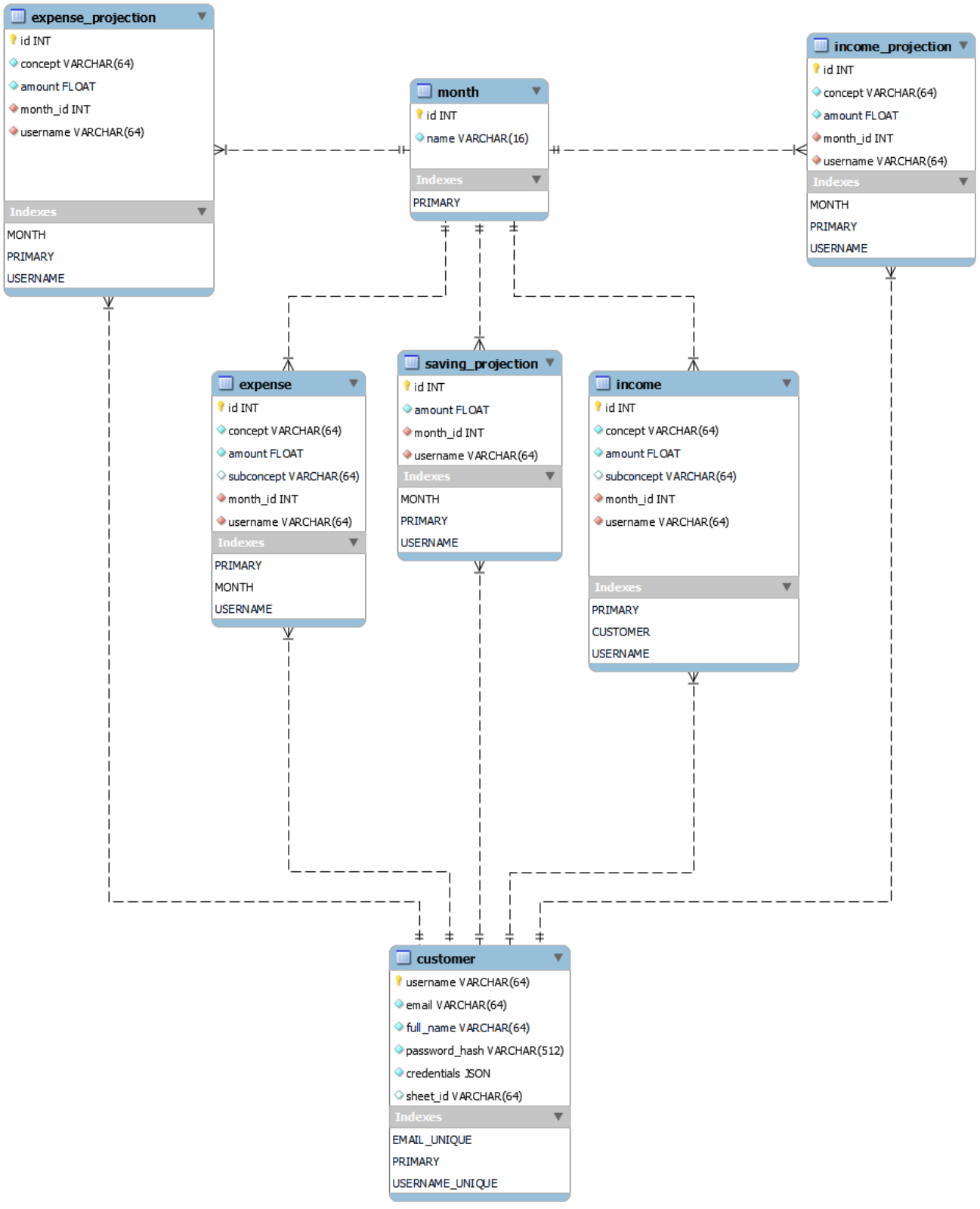
- Creación del sistema de autenticación de Google: Implementó el sistema utilizando las herramientas y API de Google. Esto garantizará la seguridad y la privacidad de los datos de los usuarios, permitiéndoles usar el sistema de manera segura.
- Creación de gráficos con Python: Se encargó de crear gráficas con Python. Creó visualizaciones interactivas que permiten a los usuarios comprender y analizar de manera efectiva sus datos financieros utilizando bibliotecas y herramientas adecuadas.
- Dockerización del proyecto: Creó los archivos y configuraciones necesarios para que la aplicación se desplegará en contenedores Docker.

Roberto Bueno García.:

- Persistencia de datos en bases de datos: Supervisó la implementación de la capa de persistencia de datos en bases de datos. creó las tablas y relaciones necesarias para administrar de manera efectiva y segura los datos financieros de los usuarios.
- El sistema de acceso a Streamlit: Creó el sistema de acceso a través de la plataforma Streamlit. Esto permitirá a los usuarios autenticarse en el sistema.
- Creación de vistas en SQL: También se encargó de hacer vistas en SQL. Estas vistas ofrecen una variedad de puntos de vista sobre los datos financieros y permiten consultas efectivas para obtener información relevante.

7.1 Modelo de datos utilizado

Durante esta fase, queremos diseñar y definir la estructura de datos que permita almacenar y organizar la información de manera eficiente y coherente.



La calidad del modelo de datos desempeña un papel clave en el éxito de cualquier sistema, ya que determina la forma en que los datos se relacionan y se acceden. Porque el modelo de datos desarrollado será la piedra angular de nuestro proyecto, ya que determinará la forma en que los datos se almacenan, se relacionan y se acceden dentro del sistema.

Entidad "customer": Esta tabla almacena información sobre los clientes. Tiene las columnas "username" (nombre de usuario), "email" (correo electrónico), "full_name" (nombre completo), "password_hash" (hash de contraseña), "credentials" (credenciales en formato JSON) y "sheet_id" (ID de hoja). La columna "email" tiene un índice único y la columna "username" tiene otro índice único. La clave primaria es la columna "username".

Entidad"month": Esta tabla almacena información sobre los meses. Tiene las columnas "id" (identificador) y "name" (nombre). La clave primaria es la columna "id".

Entidad "expense": Esta tabla almacena información sobre los gastos. Tiene las columnas "id" (identificador), "concept" (concepto), "amount" (cantidad), "subconcept" (subconcepto), "month_id" (ID de mes) y "username" (nombre de usuario). La columna "month_id" y la columna "username" tienen un índice. La tabla también tiene restricciones de clave externa que hacen referencia a las tablas "month" y "customer". La clave primaria es la columna "id".

Entidad "income": Esta tabla almacena información sobre los ingresos. Tiene una estructura similar a la tabla "expense", con las mismas columnas y restricciones de clave externa.

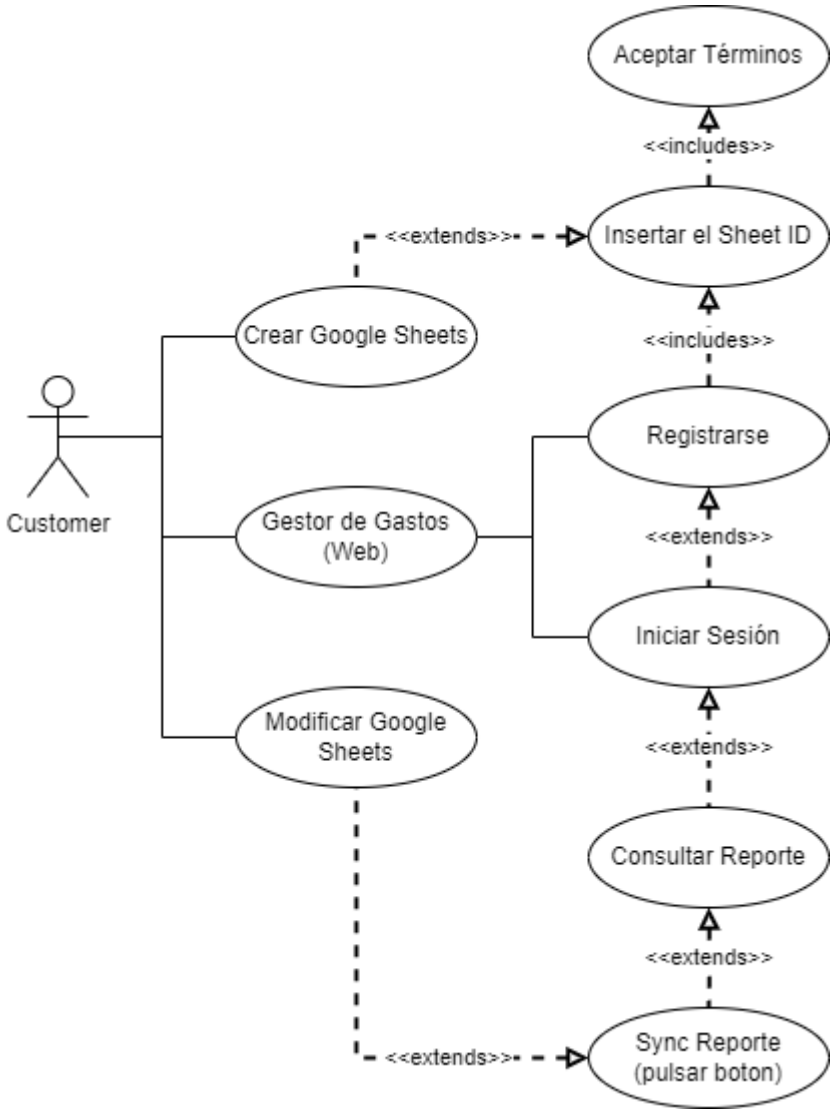
Entidad "income_projection": Esta tabla almacena proyecciones de ingresos. Tiene las columnas "id" (identificador), "concept" (concepto), "amount" (cantidad), "month_id" (ID de mes) y "username" (nombre de usuario). La columna "month_id" tiene un índice. La tabla tiene restricciones de clave externa que hacen referencia a la tabla "month" y "customer". La clave primaria es la columna "id".

Entidad "expense_projection": Esta tabla almacena proyecciones de gastos. Tiene una estructura similar a la tabla "income_projection", con las mismas columnas y restricciones de clave externa.

Entidad "saving_projection": Esta tabla almacena proyecciones de ahorro. Tiene las columnas "id" (identificador), "amount" (cantidad), "month_id" (ID de mes) y "username" (nombre de usuario). La columna "month_id" tiene un índice. La tabla tiene restricciones de clave externa que hacen referencia a la tabla "month" y "customer". La clave primaria es la columna "id".

7.2 Diagrama de clases de uso

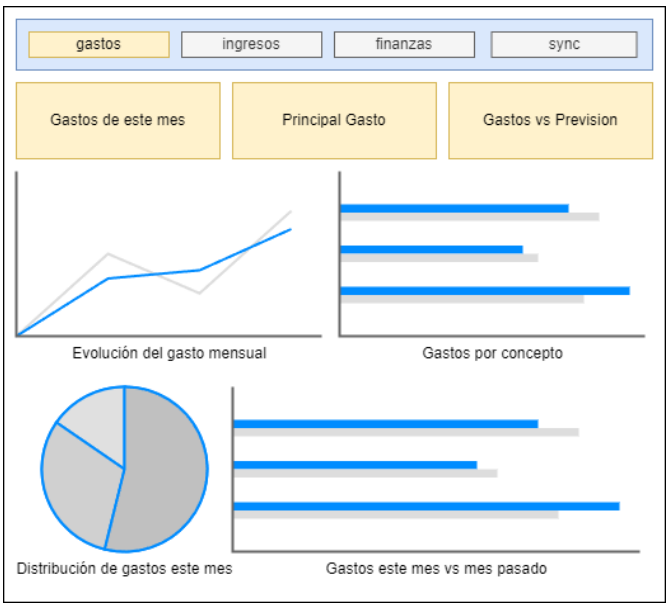
Durante esta fase, se busca identificar y comprender los diferentes actores y sus interacciones con el sistema en estudio. Nuestro diagrama de casos de uso nos proporciona una vista de alto nivel de los diferentes casos de uso que el sistema debe manejar y cómo se relacionan entre sí:

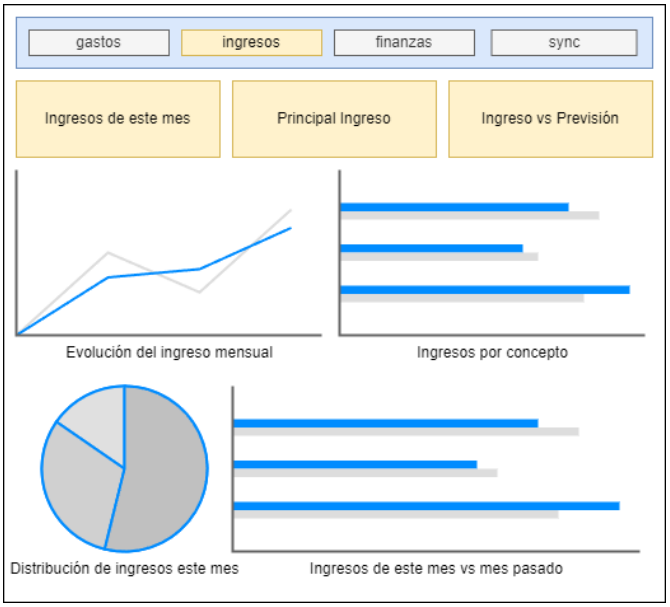


Es importante tener en cuenta que un diagrama de casos de uso no nos proporciona detalles de implementación como veremos a continuación, sino que se centra en los comportamientos externos del sistema.

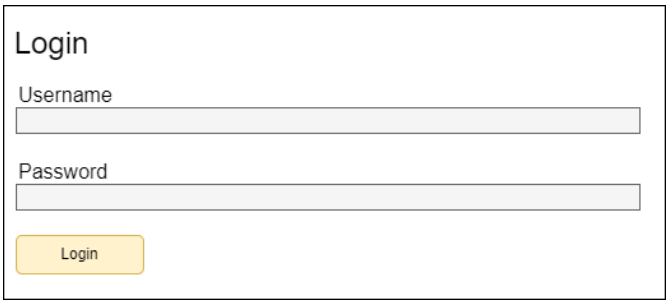
7.3 Diseño de las interfaces, wireframes y mockups

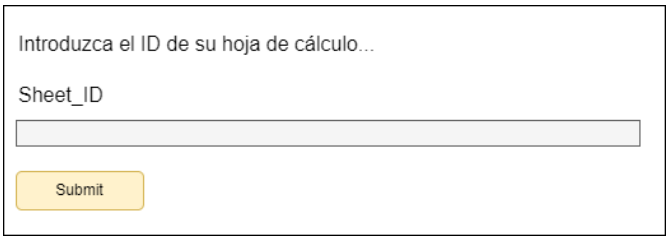
Durante esta fase, se buscará diseñar y crear representaciones visuales y funcionales de la interfaz de usuario de una aplicación o sistema que gestione los gastos e ingresos del usuario final en la aplicación, es decir, con estos mockups podemos visualizar las interfaces y simular la experiencia del usuario al interactuar con la aplicación, permitiendo evaluar y refinar el diseño antes de la implementación.

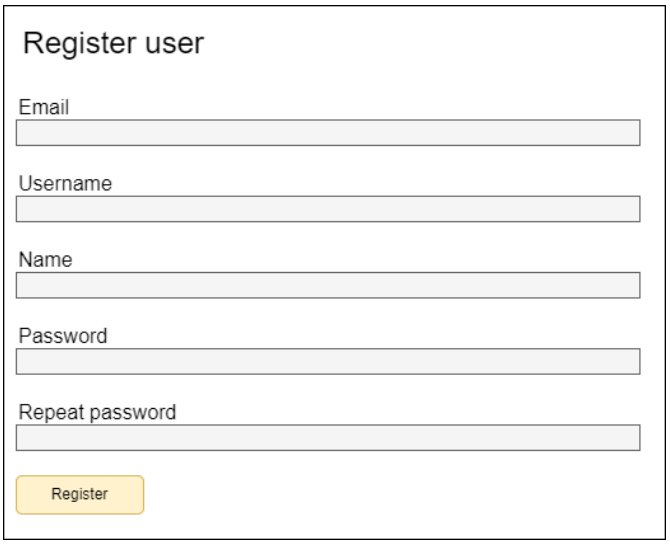




Las siguientes interfaces son formularios básicos para registrarse (Register user), iniciar sesión (Login) e introducir el id del formulario que el usuario tiene que haber creado previamente para acceder a las interfaces que mencionamos previamente.







7.4 Planificación del desarrollo

Fase 1 (entrega parcial)

- Definir los campos del Google Sheet
- Modelado de datos
- Acceder a los datos del Sheet mediante el scope de la API
- Procesar y limpiar los datos a través del script de python
- Persistir los datos en Postgres

Fase 2 (entrega final)

- Crear el sistema de autentificación en streamlit
- Persistir los tokens de google auth en postgres
- Generar las vistas de SQL
- Generar los gráficos con Python (librería Plotly)
- Crear la interfaz de visualización con los gráficos

7.5 Explicaciones de la funcionalidad del proyecto

OAuth 2.0 (Google Sheets API)

Permite a la aplicación obtener acceso al google sheet sin necesidad de saber la contraseña.

1. El usuario inicia sesión y autoriza a la aplicación a acceder a sus recursos de Google Sheets, aceptando los términos estándar.
2. La aplicación solicita y obtiene un token de acceso a la API de Google Sheets en nombre del usuario.
3. La aplicación utiliza el token de acceso para realizar solicitudes a la API de Google Sheets en nombre del usuario.

Módulo “data handler.py”

Se encarga de hacer las llamadas a la API para descargar los datos de la hoja de cálculo. Para cada campo de interés en el google sheet tenemos establecidos los rangos de las celdas, que son los que se emplean para descargar los datos con la API. Se emplea la librería Pandas para limpiar los datos y prepararlos para ser persistidos en Postgres. La conexión a postgres se realiza con la librería pyscopg2. Esta funcionalidad se invoca para sincronizar los datos de Google Sheet con nuestra base de datos, y es llamada en dos momentos: cuando se hace login, para que el usuario vea los datos más recientes, y cuando se apreta al botón sync, por si se hubiesen producido cambios durante la sesión.

Google Sheets API + OAuth 2.0 = SCOPES

Los "scopes" de la API de Google Sheets se refieren a los permisos que un usuario otorga a una aplicación para acceder a su cuenta de Google Sheets y realizar acciones en su nombre. Algunos ejemplos de scopes en español podrían ser:

- `https://www.googleapis.com/auth/spreadsheets` - acceso a las hojas de cálculo de Google Sheets
- `https://www.googleapis.com/auth/drive.file` - acceso a los archivos de Google Drive relacionados con las hojas de cálculo de Google Sheets
- `https://www.googleapis.com/auth/drive` - acceso a todos los archivos de Google Drive
- `https://www.googleapis.com/auth/spreadsheets.readonly` - acceso de solo lectura a las hojas de cálculo de Google Sheets
- `https://www.googleapis.com/auth/drive.readonly` - acceso de solo lectura a los archivos de Google Drive

Es importante tener en cuenta que cada scope otorga diferentes niveles de acceso y privacidad, por lo que es importante que los usuarios revisen cuidadosamente los permisos que están otorgando a las aplicaciones antes de autorizarlas.

8. Conclusiones y mejoras del proyecto

En este proyecto, hemos logrado desarrollar una aplicación utilizando Streamlit, que integra Google Sheets y una base de datos PostgreSQL. En este sentido hemos acometido los principales objetivos establecidos en el anteproyecto, ajustándonos en el tiempo. Sin embargo, a lo largo del proceso, hemos identificado algunos desafíos y oportunidades de mejora que aún debemos abordar.

Un desafío pendiente es llevar nuestra aplicación a un entorno de producción para que esté disponible y accesible para los usuarios. Actualmente, estamos ejecutando la aplicación en un entorno local o de desarrollo, y es importante realizar el proceso de hosting en un servidor o en la nube para que sea accesible en línea, tanto para el server de streamlit como para la base de datos Postgres.

También hemos observado que hay margen para mejorar la eficiencia de las llamadas a la API de Google Sheets. Actualmente, realizamos múltiples llamadas acotadas a la API, lo cual puede afectar el rendimiento. Sería más rápido hacer menos llamadas, con muchos más datos, y limpiarlos con python.

Otra área en la que podemos mejorar es en el sistema de caché de Streamlit. En la configuración actual, se realizan consultas repetitivas a la base de datos cada vez que los usuarios navegan por la aplicación. Podemos trabajar en implementar una estrategia de caché más eficiente, lo cual ayudaría a acelerar la carga de datos y a mejorar la experiencia del usuario.

Además, hemos notado que hay espacio para mejorar la eficiencia de las consultas SQL a la base de datos PostgreSQL. Optimizar las consultas, revisar los índices, estructurar las consultas de manera eficiente y aprovechar las características avanzadas de PostgreSQL pueden contribuir a mejorar el rendimiento y reducir los tiempos de respuesta de la aplicación.

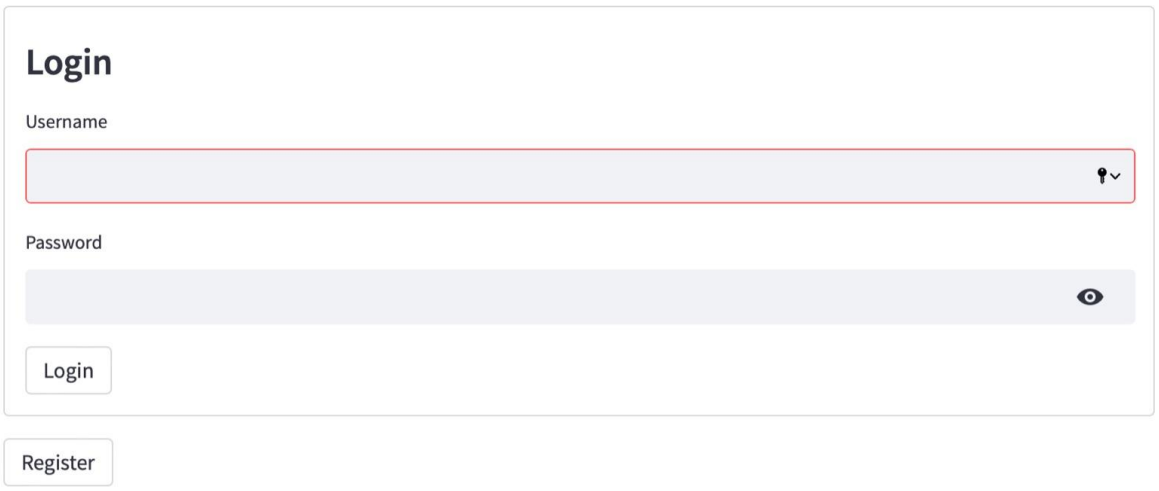
Por último, hemos identificado la oportunidad de desarrollar un esquema más flexible en nuestra aplicación. Actualmente, estamos limitados por una plantilla rígida, lo cual restringe las opciones de personalización para los usuarios. Sería valioso permitir a los usuarios tener un mayor control sobre cómo desean integrar y adaptar la aplicación a sus necesidades individuales.

En conclusión, este proyecto ha sido una experiencia enriquecedora que nos ha permitido aplicar nuestros conocimientos y habilidades adquiridas en distintos módulos de formación profesional. Ha sido una oportunidad para aprender nuevas herramientas, enfrentar desafíos reales y explorar soluciones creativas. A través de este proyecto, hemos fortalecido nuestra capacidad para desarrollar aplicaciones prácticas y hemos ampliado nuestra comprensión en el ámbito de la programación y la gestión de datos.

9. Bibliografía

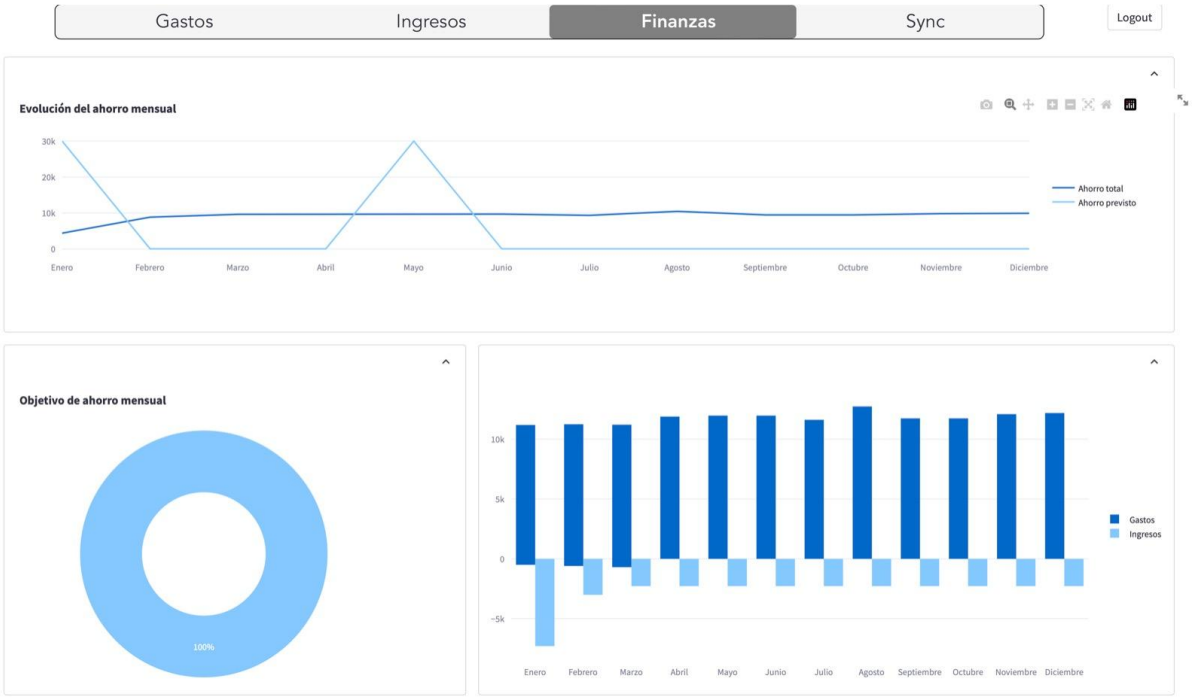
1. McKinney, W. (2018). Python for Data Analysis. O'Reilly Media.
2. Streamlit Documentation. https://docs.streamlit.io/
3. Google Sheet API Reference. https://developers.google.com/sheets/api/reference/rest/
4. PostgreSQL Global Development Group. (2021). PostgreSQL: The world's most advanced open source relational database. https://www.postgresql.org/
5. Google Developers. (2021). Google Sheets API Python Quickstart. Recuperado de https://developers.google.com/sheets/api/quickstart/python

10. Anexos

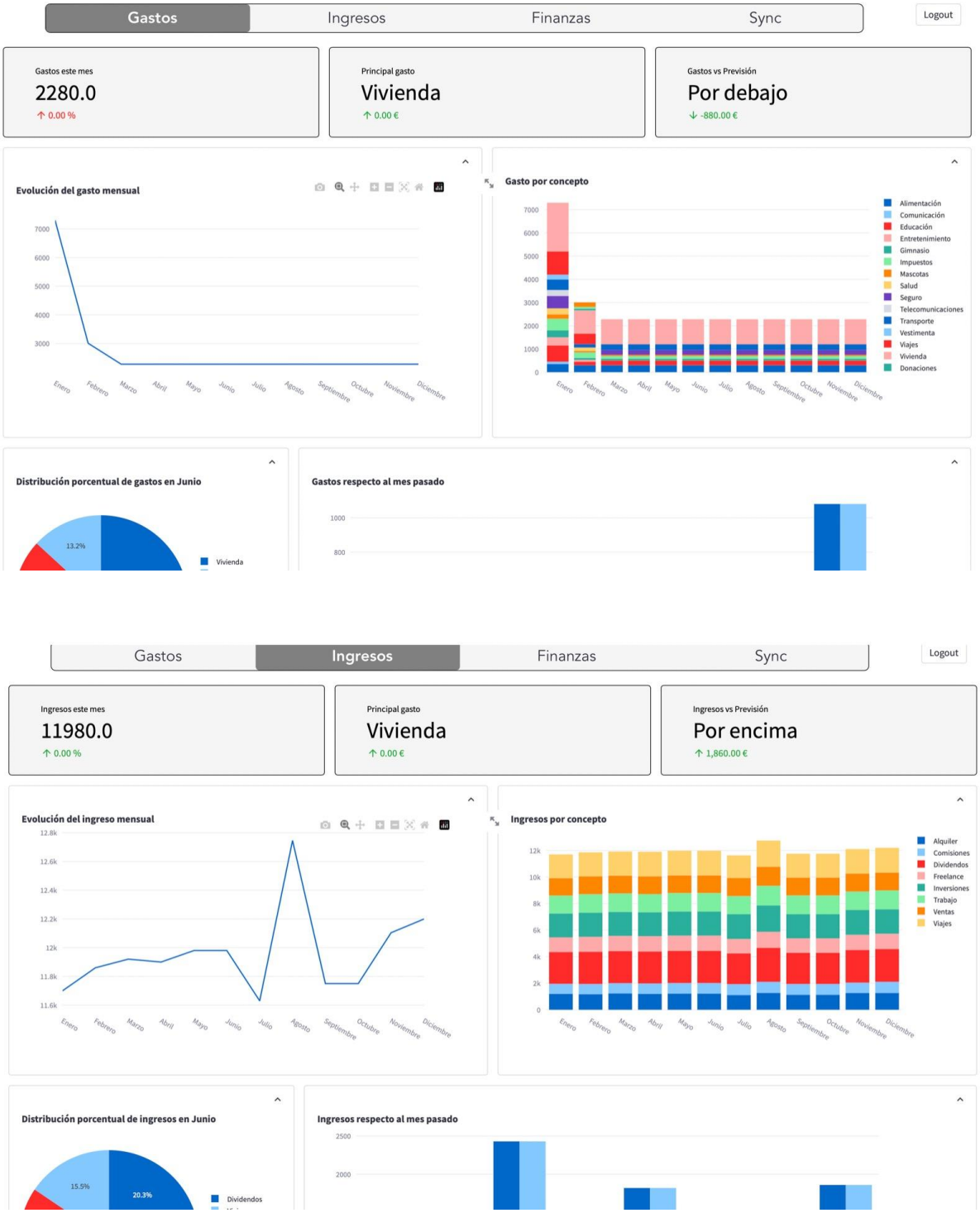


El primer flujo de interacción con el usuario es el registro o inicio sesión como vemos en las capturas de pantalla. El usuario después de registrarse deberá especificar el identificador de un google sheet que será utilizado para grabar sus gastos personales.

La aplicación utiliza un scope/recurso que es el identificador que podemos encontrar en la URL del Google Sheet para visualizar un análisis de gastos a través de internet.



En la interfaz de Finanzas podemos ver la evolución del ahorro mensual, la diferencia entre gastos e ingresos y el objetivo de ahorro mensual.



Tanto la interfaz de Gastos e Ingresos tienen el mismo diseño, donde podemos ver el ingreso de este mes, el principal gasto, gastos contra previsión, evolución del gasto mensual, gastos por concepto, la diferencia con el mes pasado y la distribución porcentual de gastos.