Proyecto

“Gestor financiero”

Integrantes:

Antonia Paredes (21194240121)

Leandro Matamoros (20978573021)

Docente: Samuel Sepulveda

Asignatura: Programación orientada a objetos

Fecha: 10 de diciembre de 2023

Índice

Introducción

En la era digital, nos encontramos inmersos en un mundo de comodidades y facilidades, aún así, persisten desafíos en nuestra vida cotidiana, siendo uno de ellos la dificultad que enfrentan muchas personas para llevar un registro detallado y preciso de sus gastos diarios. Esta falta de conocimiento sobre en qué se gasta el dinero, puede llevar a una serie de problemas financieros, desde la incapacidad de establecer presupuestos realistas, hasta dificultades para tomar decisiones financieras informadas. A menudo, esto resulta en gastos incontrolados, ahorros insuficientes y, en el peor de los casos, problemas de endeudamiento.

Es en este contexto que surge la necesidad de una solución eficiente y accesible, que aborde esta problemática, una solución que permita a las personas llevar un registro detallado de sus finanzas, guiándolos a gestionar sus recursos de manera inteligente y proactiva.

En las siguientes secciones, se presentará la solución construida, cuyo objetivo es proporcionar a los usuarios las herramientas necesarias para llevar un registro de sus gastos, transformando la manera en que manejan sus gastos diarios y permitiéndoles tomar decisiones financieras más conscientes, estableciendo una base sólida para su bienestar económico a largo plazo.

Explicación del caso

El caso abordado se centra en el desafío común que enfrentan muchas personas al intentar llevar un registro detallado y preciso de sus gastos diarios. Esta falta de conocimiento sobre el destino de sus ingresos puede resultar en dificultades para establecer presupuestos realistas y tomar decisiones financieras responsables. Consecuentemente, se pueden experimentar problemas como gastos incontrolados, ahorros insuficientes y endeudamiento.

Para abordar esta problemática, se planea desarrollar una aplicación de gestión financiera. Esta aplicación permite a los usuarios registrar sus gastos diarios y asignarlos a diferentes categorías, proporcionando así un seguimiento detallado de cómo se gasta el dinero. Además, ofrece un resumen analítico para comprender la distribución de los gastos. También incluye la capacidad de establecer recordatorios para las cuentas por pagar.

La aplicación está diseñada para ser accesible para usuarios de cualquier rango etario y profesión, con el objetivo de capacitar a las personas para que tomen el control de sus finanzas personales y realicen decisiones más informadas, basándose en sus ingresos, gastos y metas. La solución busca transformar la forma en que las personas gestionan sus finanzas diarias, promoviendo la estabilidad financiera y el bienestar económico de los usuarios.

Clases del proyecto y diagrama UML

Para la construcción de nuestro gestor financiero creamos diferentes clases empaquetadas en su package correspondiente, a continuación se nombrara detalladamente las clases:

1. **Package “modelo”:**

Clase Usuario: Esta clase modela la entidad de un usuario; cuenta con atributos privados para almacenar el nombre, la dirección de correo electrónico y la contraseña del usuario. El constructor permite la creación de objetos Usuario con valores específicos, y los métodos de acceso (get) facilitan la obtención de información de estos objetos. Además, se incluye un método de modificación (set) para actualizar la contraseña del usuario.

Clase Gasto: Esta clase se encarga de modelar un gasto; tiene atributos para el nombre del gasto, la cantidad gastada, la categoría a la que pertenece y el correo electrónico del usuario asociado. El constructor permite crear objetos de gasto con valores específicos. Además, se proporcionan métodos de acceso para obtener información sobre el gasto.

Clase CuentaUsuario: Esta clase proporciona funcionalidades relacionadas con la gestión de cuentas de usuarios en un sistema. Incluye métodos para iniciar sesión, registrar nuevos usuarios, verificar la existencia de un correo electrónico, cambiar contraseñas, entre otras. Además, la clase utiliza funciones de la clase DatosUsuario para interactuar con los archivos de datos.

Clase Categorías: Esta clase tiene como objetivo gestionar categorías de productos, inicializa un conjunto predefinido de cinco categorías ("Alimentación", "Transporte", "Entretenimiento", "Educación", "Otras") y crea listas de productos asociadas a cada categoría. Además, proporciona métodos para obtener la lista de categorías y las listas de productos por categoría.

Clase Finanzas: Esta clase gestiona aspectos financieros de un usuario en el sistema. Contiene funciones para manejar el saldo, registrar gastos, y obtener información sobre gastos y saldos. Entre las funciones destacadas se encuentran anadirDinero que permite añadir dinero al saldo actual, getSaldoActual para obtener el saldo actual del usuario, y registrarGasto que registra un gasto actualizando saldos y almacenando la información correspondiente.

2. **Package “utils”:**

Clase CalculadoraPorcentajeGastos: Esta clase proporciona métodos para calcular el total gastado y el total gastado por categoría a partir de una lista de gastos. La función calcularTotalGastado suma las cantidades de todos los gastos en la lista, mientras que calcularTotalGastadoPorCategoria suma las cantidades de los gastos que pertenecen a una categoría específica.

Clase ValidarEntradaUsuario: Esta clase contiene métodos para validar diferentes tipos de entrada de usuario en el sistema. Proporciona funciones como validarDouble y validarInt para verificar si un JTextField contiene un valor numérico (double o entero) válido. Además, incluye métodos para validar el formato de un correo electrónico, la longitud y formato de una contraseña, entre otras.

3. **Package “datos”:**

Clase DatosGatos: Se encarga del manejo de los datos relacionados con los gastos de los usuarios en un archivo CSV. Proporciona métodos para guardar gastos en dicho archivo, cargar gastos desde él y procesar la lectura y escritura de datos, entre otros métodos relacionados.

Clase DatosUsuario: La clase se encarga de la persistencia de los datos relacionados con los usuarios en un archivo CSV. Proporciona métodos para registrar nuevos usuarios, cargar la lista de usuarios desde el archivo y actualizar la información de un usuario existente.

Clase Saldo: La clase se encarga de gestionar los datos de saldos asociados a usuarios en un archivo CSV. Proporciona métodos para guardar y cargar saldos, así como para actualizar esta información en el archivo.

4. **Package “launcher”:**

Clase Launcher: La clase "Launcher" es la clase principal que contiene el método main y sirve como punto de entrada del programa. Al ejecutar el programa, se llama al método main, que a su vez invoca la función abrirVentanaInicioSesion() de la clase ManejoGuis.

5. **Package “guis”:**

Clase ManejoGuis: Esta clase contiene métodos relacionados con la gestión de interfaces gráficas de usuario (GUIs) en una aplicación. Estos métodos están diseñados para abrir diversas ventanas de la interfaz, mostrar información específica en los componentes gráficos, y realizar acciones como cerrar el programa o confirmar la salida gráfica.

Clase GuiInicioSesion: La clase constituye la interfaz gráfica de usuario (GUI) destinada al proceso de inicio de sesión en una aplicación.

Clase GuiCrearCuenta: La clase modela interfaz gráfica destinada al proceso de creación de cuenta del usuario, donde permite ingresar su nombre, correo electrónico y contraseña.

Clase GuiPrincipal: La clase configura una la interfaz gráfica principal de la aplicación, destinada a la gestión financiera del usuario. Aquí el usuario puede escoger entre las diferentes opciones presentadas.

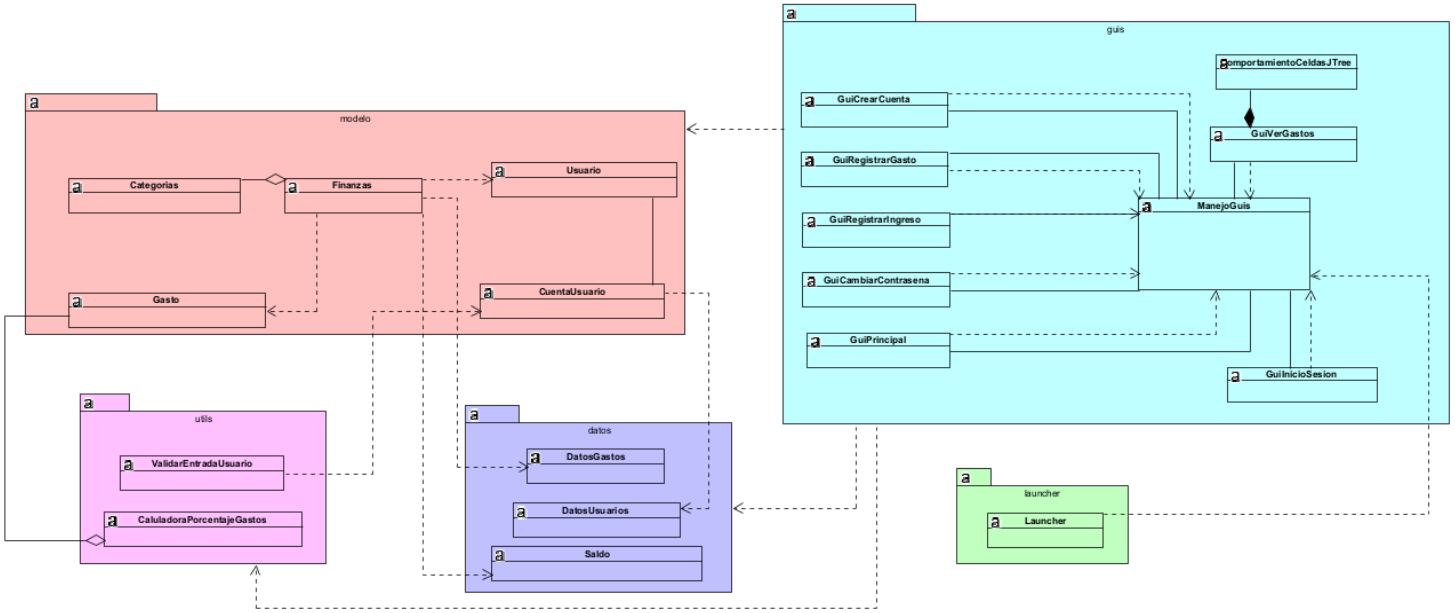
Clase GuiRegistrarGasto: La clase constituye una interfaz gráfica que permite al usuario registrar un gasto, ingresando el monto gastado, la categoría a la que pertenece y el nombre.

Clase RegistrarIngreso: La clase modela una interfaz gráfica que te permite ingresar un monto a la cuenta del usuario.

Clase VerGastos: La clase modela la interfaz gráfica encargada de mostrar todos los gastos realizados por categorías, además permite ver el total gastado en la categoría específica.

Clase GuiCambiarContrasena: La clase modela la interfaz gráfica que permite cambiar la contraseña actual del usuario por una nueva.

A continuación puede observar el diagrama de clases reducido del proyecto:



Para visualizar el diagrama de manera detallada puede ir al siguiente link:

Diagrama De Clases.

O puede descargar el archivo VPP por medio de este enlace:

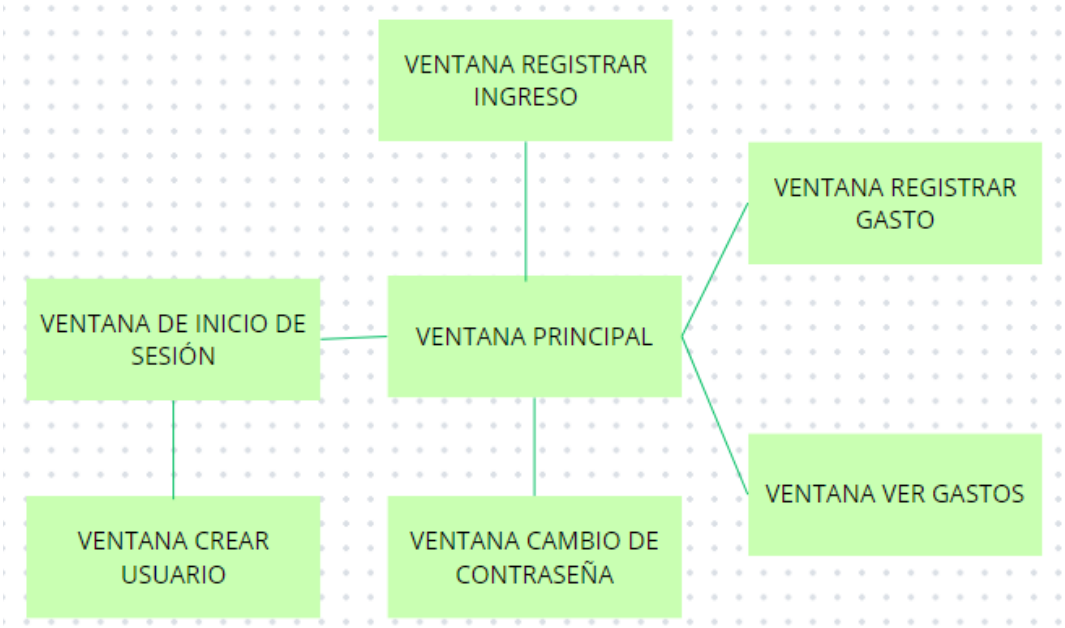
Archivo VPP.

Interfaces gráficas

En el marco de la planificación del proyecto, se ha tomado una decisión estratégica con respecto a la interfaz gráfica que se emplea. Se ha optado por desarrollar una interfaz gráfica específicamente diseñada para dispositivos móviles, como teléfonos celulares, basándose en la consideración de su portabilidad y su uso frecuente en la vida cotidiana. Este enfoque se alinea con el propósito de brindar a los usuarios un acceso conveniente a la aplicación en cualquier momento y lugar.

En relación con el diseño de la interfaz, se ha adoptado una filosofía de simplicidad y usabilidad. La interfaz se configurará de manera sencilla y se orientará hacia una experiencia de usuario intuitiva, siguiendo los estándares de aplicaciones ampliamente reconocidas y exitosas en el mercado. Este enfoque implica evitar cualquier sobrecarga de información o características complejas que puedan generar confusión. En su lugar, se priorizará la creación de una experiencia de usuario fluida y eficaz, en la que los usuarios puedan llevar a cabo sus actividades financieras de manera eficiente y sin dificultades.

Para mejorar la visualización de la interfaz lograda, se ha creado un diagrama conciso que ilustra la relación entre las ventanas. Además, se ha elaborado otro diagrama más detallado que identifica específicamente los botones destinados a la navegación entre estas ventanas. Para acceder a estas representaciones, por favor, haga clic en este enlace.

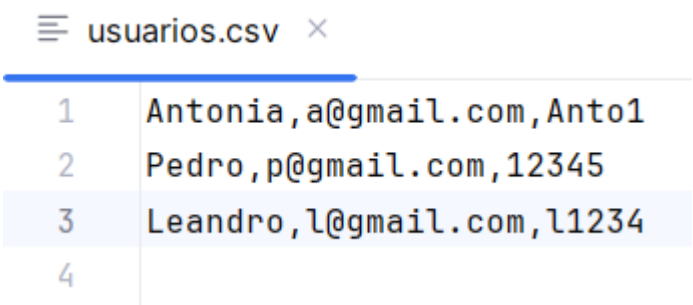


Manejo de los datos

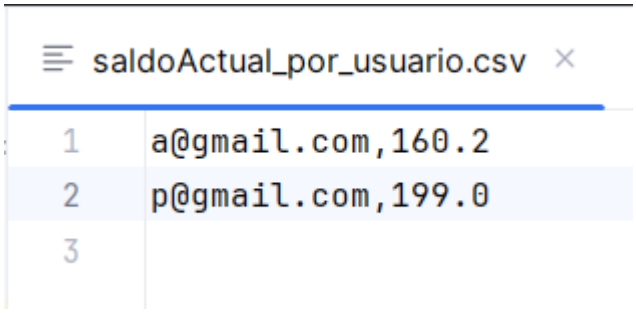
Después de realizar un proceso de investigación y evaluación de las opciones disponibles para el manejo de datos en el contexto de nuestro proyecto, se decidió utilizar archivos CSV (Comma-Separated Values) como el formato para almacenar y gestionar la información. Esta elección se basa en las ventajas significativas que los archivos CSV ofrecen en términos de organización de datos, manipulación de números y búsqueda eficiente de información. Además, “los archivos CSV son una opción popular para el intercambio de datos, ya que son compatibles con una amplia gama de aplicaciones de negocio, de consumo y científicas.” (Zoiner Tejada, 2023) lo que facilita la integración y la transferencia de datos en el entorno del proyecto.

Los archivos necesarios y utilizados en el programa se almacenan en una carpeta que se crea automáticamente la primera vez llamada Usuarios, los archivos almacenados son los siguientes:

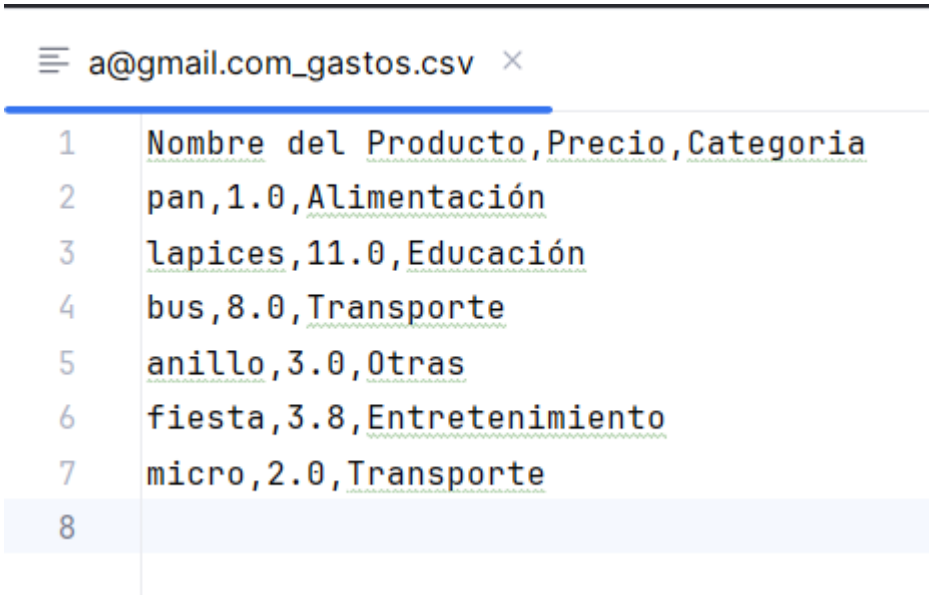
usuarios.csv: En este archivo csv se almacenan los datos de las cuentas de usuario del programa: nombre de usuario , correo electrónico y contraseña.



saldoActual_por_usuario.csv: En este archivo se almacena el saldo total de la cuenta de todos los usuarios.



(correodelusuario).csv: En este archivo se registran todos los gastos del usuario con un nombre asociado al gasto, el monto del gasto y la categoría a la correspondiente. Se crea uno por usuario y se almacenan en la carpeta "gastos_por_usuarios".



Las clases que modifican y guardan los datos están dentro del package “datos” y son las siguientes:

- Clase DatosGastos: Se encarga de leer y gestionar los CSV relacionados a los gastos por usuario, de formato “(correodelusuario).csv”.
- Clase DatosUsuarios: Se encarga de leer y gestionar los datos de usuario, archivo CSV de nombre “usuarios.csv”.
- Clase Saldo: Esta clase lee y edita el archivo CSV de saldo total de los usuarios, este archivo es llamado “saldoActual_por_usuario.csv”.

Manejo de situaciones excepcionales

Durante la fase de desarrollo del programa, se evaluaron diferentes situaciones problemáticas que podrían surgir durante la ejecución del software. Para abordar estos escenarios, se integró un mecanismo de gestión de errores utilizando la estructura "Try-Catch" y se incorporaron métodos de validación para mejorar la robustez y confiabilidad del programa.

5 potenciales errores en el programa:

1. Ingreso de valores de formato incorrecto (NumberFormatException):

Este error refiere a la posibilidad de que los usuarios, por ejemplo, ingresen caracteres no numéricos donde se esperan valores numéricos; se implementaron métodos para validar las entradas de usuario y múltiples “Try-Catch” en el código.

2. Ingresar un correo o una contraseña no válida:

Es necesario que el programa solo acepte correos válidos, no sólo porque también se busca seguir las reglas para la creación de los correos, también para evitar posibles errores con la lectura de los datos, para esto se creó un método validador, además, para evitar que correos esencialmente iguales, sean considerados diferentes, se aplicó “.trim().toLowerCase()” al leer correos.

3. Ingresar una categoría no disponible:

El manejo de categorías en la aplicación es importante por lo tanto se debe asegurar que el usuario siempre escoja entre las categorías disponibles para no generar errores, esto fue controlado mediante un método validador.

4. Ingreso de valor nulo (Campo de texto vacío):

Al intentar ingresar un campo vacío se genera el error “NullPointerException”,se controló este problema mostrando un error en pantalla al intentar realizar esta acción.

5. Lectura de archivo inexistente:

Cuando el programa se ejecuta por primera vez, aún no se han creado los archivos de datos de usuario, por lo que la lectura de estos sería imposible; lo mismo ocurre cuando un usuario nuevo intenta visualizar sus gastos o el porcentaje de dinero que ha gastado por categoría. Este posible error se manejó imprimiendo un texto simple por monitor, y en las ventanas de usuario, mostrando textos que dan a entender que no se registran los datos a visualizar.

Pruebas unitarias

En el proceso de desarrollo del programa se analizaron cinco posibles errores que podrían ocurrir durante la ejecución del programa, para ello se implementaron las pruebas unitarias correspondientes.

1. Validar contraseña test:

Se realizó una prueba unitaria para comprobar que la validación de la contraseña era la correcta, obligando a tener un mínimo de 5 caracteres.

2. Validar formato correo test:

Se realizó una prueba unitaria al método de validación del correo electrónico para asegurar que el correo que se guarda cumple con los requisitos como no tener puntos antes del " @", o que cumpla con las estructura (String@String.String)

3. Registrar gasto test:

Se llevó a cabo una prueba unitaria específica para evaluar si la funcionalidad de registro de gastos en el sistema funcionaba correctamente.

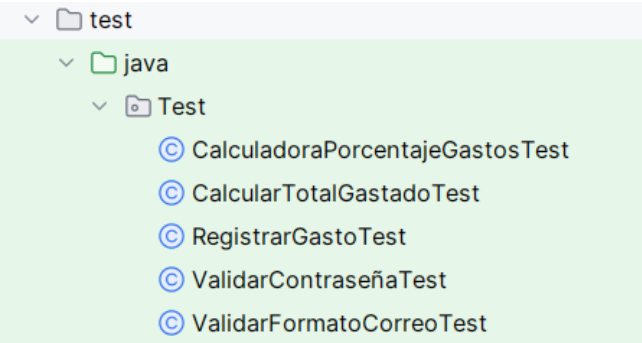
4. Calcular total gastado test:

La prueba unitaria se diseñó con el propósito de confirmar la correcta ejecución del cálculo del total gastado para una categoría específica.

5. Calcular porcentaje gastos test:

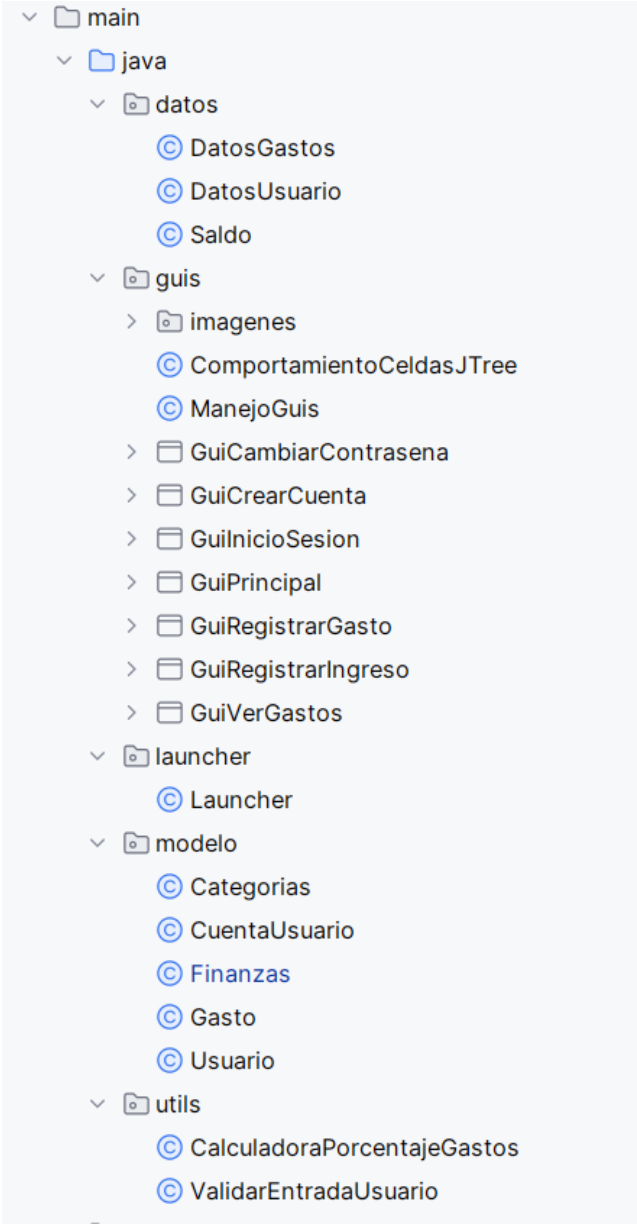
Se implementó una prueba unitaria focalizada en garantizar la precisión del cálculo del porcentaje de gastos en relación con el total general.

Para ver los test realizados haga click aquí.



Código fuente

A continuación se observa la estructura final de packages y clases del proyecto:



Link GitHub del proyecto

Link documento UML

Link descargable documento VPP

Principales dificultades y comentarios finales

En el transcurso del desarrollo del proyecto, el equipo se enfrentó a diversos desafíos que proporcionaron valiosas lecciones y condujeron a ajustes en la propuesta inicial. A pesar de superar con éxito la implementación de interfaces gráficas (GUIs) y el manejo de archivos, se identificaron dificultades al intentar incorporar los recordatorios personalizados del usuario en la aplicación. La gestión de recordatorios extensos, así como la edición, eliminación y adición de nuevos elementos, se revelaron como tareas complejas, llevando a la decisión de descartar esta característica para priorizar la usabilidad y funcionalidad esenciales.

Otra área que presentó complicaciones fue la implementación del gráfico propuesto inicialmente. La complejidad técnica asociada con nuestros conocimientos actuales, condujo a la conclusión de que su integración no era viable en el contexto actual del proyecto.

En términos de lecciones aprendidas, el equipo reconoce la importancia de la flexibilidad y adaptabilidad en el desarrollo de software. La revisión y ajuste de la propuesta original resultaron fundamentales para la consecución de un producto más efectivo y centrado en las necesidades del usuario. En cuanto a los comentarios finales, a pesar de los obstáculos encontrados, se valora positivamente el resultado obtenido y se identifican áreas pendientes, proporcionando orientación para futuras mejoras y expansiones del proyecto.

Conclusiones y trabajos futuros

Se ha explorado la necesidad de abordar el desafío común de manejar de forma consciente las finanzas personales, para afrontar esta problemática, se ha desarrollado y presentado una aplicación de gestión financiera, que junto con el manejo de archivos CSV como formato de datos, constituye una solución efectiva para abordar los desafíos relacionados con el seguimiento de gastos diarios.

Si bien se lograron los principales objetivos propuestos, el programa aún puede mejorar, para esto se han ideado ciertas propuestas de trabajos futuros:

1. Se propone la implementación de opciones personalizadas de colores y tamaños de letras, específicamente diseñadas para usuarios con visión reducida o daltonismo. Esta medida tiene como fin ofrecer una experiencia visual adaptada a diversas necesidades.

2. En el ámbito funcional, se plantea la integración de consejos financieros en la interfaz, junto con el desarrollo de un acceso directo a una plataforma educativa en temas financieros. Esto permitirá a los usuarios recibir orientación y recursos valiosos para mejorar su comprensión y toma de decisiones financieras.

3. Desde el punto de vista del código, se propone llevar a cabo una reestructuración significativa convirtiendo la clase “ManejoGuis” en una clase abstracta. Esta propuesta tiene como objetivo mejorar la organización interna del programa y facilitar el mantenimiento a medida que se implementan nuevas características. Además, se plantea la incorporación de funciones de recordatorios y gráficos para enriquecer la experiencia del usuario.

Finalmente, puede revisar el código de nuestro programa y los documentos relacionados en el siguiente link:

https://github.com/Antix199/GestorFinanciero.

Bibliografía

Zoiner Tejada. (2023, July 11). Procesamiento de archivos CSV y JSON - Azure Architecture Center. Microsoft Learn. https://learn.microsoft.com/es-es/azure/architecture/data-guide/scenarios/csv-and-json#about-csv-format

Repositorio GitHub del proyecto.

https://github.com/Antix199/GestorFinanciero.

Esquema de navegación GUIs realizado en Canva.

https://www.canva.com/design/DAF2nIu9W9w/-dIbwrqSNBN0qDTulL8AJw/edit?utm_content=DAF2nIu9W9w&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Archivo Vpp descargable del diagrama de clases.

https://github.com/Antix199/GestorFinanciero/blob/403b10adfe4d21a90b9c34e2a60c06f961f41c44/DiagramaClasesGestorFinanciero.vpp.