Universidad Politécnica Internacional

Proyecto#1 SplitBuddies

Técnicas de programación

Profesor: Luis Felipe Mora Umaña

Integrantes:

Yanicsa Galilea Casco

Melissa Fallas Jiménez

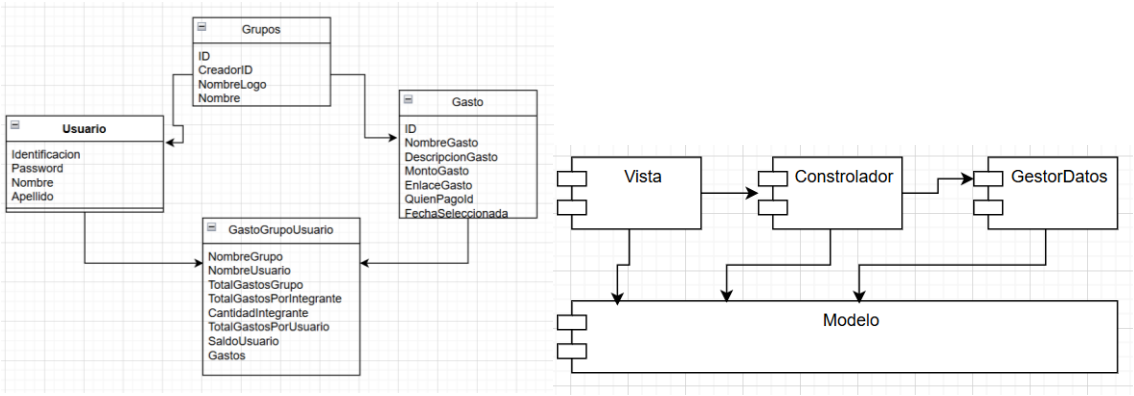
30 de julio del 2025

Índice

Decisiones de diseño

Estructura General del Proyecto

Aquí se muestra un diagrama de la estructura del proyecto que fue elaborado con Draw.io con el fin de representar visualmente la organización de los componentes principales del sistema:



La estructura y el código del proyecto se divide en cuatro partes, una que es el controlador, otro gestor de datos, modelo y la parte de la vista.

Se utilizó el patrón MVC para una mejor organización y se incorpora un Gestor Datos para cumplir con el principio SOLID de responsabilidad única, se separa el manejo de datos del controlador vista y modelo para que pueda gestionar todas las operaciones de escritura y lectura sin modificar los comportamientos del sistema en caso se desee cambiar el origen de los datos.

Controlador

Este fundamental en la arquitectura del sistema ya que encarga de la lógica del proyecto y es el responsable de manejar la interacción entre la vista, el modelo y en este caso también interactúa con el gestor datos, este controlador trabaja con interfaces para que no fueran consumidos directamente por las vistas mediante clases concretas, sino a través de interfaces bien definidas.

Este diseño permite mayor flexibilidad y desacoplamiento entre componentes. Por ejemplo, si en futuras iteraciones se desea reemplazar Gastos Controlador por una nueva clase GastosControladorAvanzado que maneje validaciones adicionales o integración con una base de datos, no será necesario modificar los formularios que consumen el controlador, siempre que la nueva clase implemente IGastosControlador. Basta con inyectar la nueva implementación, lo que también facilita el uso de inyección de dependencias, incluso en futuros entornos más complejos.

Se dividió en cuatro controladores:

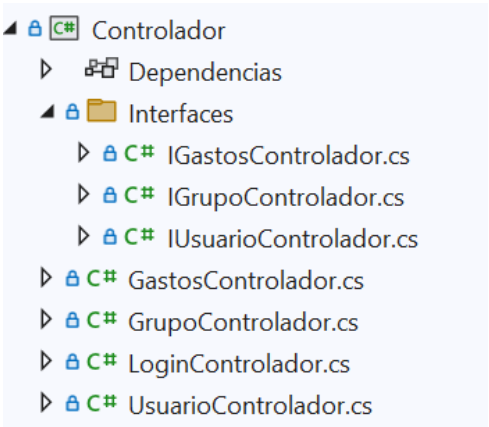
1-Una parte donde se va a gestionar todo lo que es la información de los gastos.

2- Otra parte donde se gestiona toda la información de los grupos.

3-Otra parte que es específicamente nada más para la validación del login donde se van a manejar el usuario y la contraseña.

4-La otra que es un controlador de usuarios donde se va a manejar la información de los usuarios para buscarlos, obtenerlos, crearlos.

Y se crearon 3 interfaces que van a ser representadas para la parte del usuario, grupo y gastos.

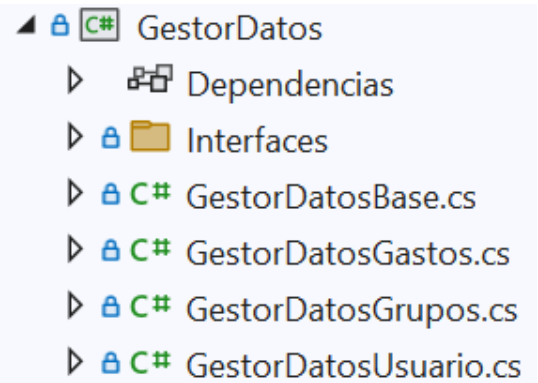


Gestor de datos

Este fue diseñado con el objetivo de encapsular todas las operaciones relacionadas con la persistencia y recuperación de datos, siguiendo el principio de separación de responsabilidades. Esta capa actúa como intermediaria entre los controladores del sistema y las fuentes de datos, que en esta iteración inicial son archivos JSON.

Su principal responsabilidad es proporcionar métodos y funciones reutilizables, eficientes y desacoplados que permitan cargar, guardar y actualizar la información necesaria para el funcionamiento del sistema según sea el modelo. En lugar de permitir que los controladores accedan directamente a los archivos del sistema, se introdujo una arquitectura basada en interfaces como IGestorDatosUsuario, IGestorDatosGrupo e IGestorDatosGasto.

Un ejemplo de este desacoplamiento se da en el controlador de gastos: en lugar de invocar directamente métodos que manipulen archivos json, el controlador utiliza una interfaz de tipo IGestorDatosGasto. De esta forma, si en futuras iteraciones se requiere reemplazar los archivos Json por una base de datos bastará con implementar una nueva clase, por ejemplo, GestorDatosGastoSQL, sin necesidad de modificar el controlador, la vista o el resto del sistema.



En la parte de gestor de datos también se utilizaron cuatro gestores de datos:

1- Uno general que lo que hace es encargarse de leer el json y escribir en el json.
2- El del gestor de gastos se va a encargar de obtener información de los gastos, cargarlos y guardarlos.
3- La parte del gestor de datos grupos, este se encarga de cargar grupos, crearlos para guardarlos en los json.
4- Gestor de datos usuario es el encargado de crear los usuarios y guardarlos en los jsons.

Modelo

Este se encarga de las entidades como usuario, grupo y gasto, etc. Cada clase del modelo se diseñó utilizando principios claves de POO aplicando conceptos como herencia, Por ejemplo, la clase Gasto contiene atributos como Nombre, Descripción, Monto, Fecha. Estos métodos son internos al modelo, lo que asegura que la lógica esté centralizada en el lugar correcto, sin depender de la vista o del controlador para validaciones básicas.

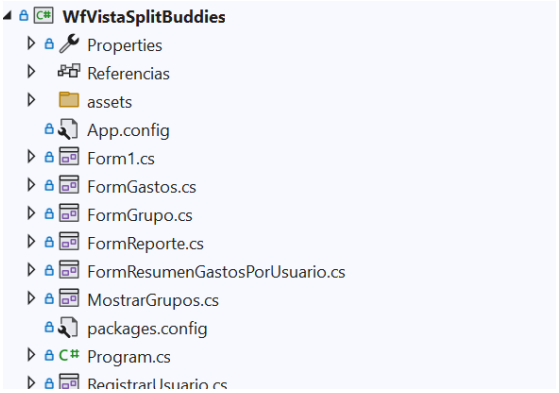
Contiene:

- Una clase para gastos
- Una clase para la relación de gasto grupo usuario. La relación de grupo usuario es para relacionar un grupo tiene ciertos usuarios, y el usuario tiene ciertos grupos. Esta lo que tiene es el nombre del Grupo, nombre del usuario, total gastos grupos, total gastos por integrante, cantidad de integrantes, total gastos por usuario, y el saldo usuario.
- Una clase de grupo, que va a representar los grupos creados.
- La relación grupo gasto, que es para relacionar todos los gastos que tiene un grupo.
- La relación usuario gasto es para relacionar todos los gastos que tiene un usuario.
- Y relación usuario grupo es para relacionar todos los grupos en los cual puede estar un usuario.
- La clase reporte es la que va a manejar los gastos totales que ha tenido el usuario, qué ha pagado y la deuda, que es lo que debe, así como, la disponibilidad de lo de lo que tiene a favor y en contra.
- La clase usuario, que contiene toda la información personal del usuario, que también es la que se usa para crear el usuario y validar el login.

Vista

Es la responsable de toda la interacción directa con el usuario. En esta primera iteración del sistema, se ha implementado utilizando el framework Windows Forms. Cada formulario FormUsuarios, FormGrupos, FormGastos, FormEstadisticas, representa una unidad visual independiente, diseñada para mostrar datos provenientes del modelo y recoger entradas del usuario que luego son delegadas al controlador correspondiente. Esta capa nunca contiene lógica de negocio ni lógica de persistencia, respetando estrictamente el principio de separación de responsabilidades.

Un aspecto relevante del diseño de las vistas es que cada formulario interactúa exclusivamente con interfaces de los controladores como IGastosControlador, IGrupoControlador, en lugar de depender de instancias concretas. También, se agrega una carpeta assets con los archivos json para facilitar el acceso a los datos desde los formularios de Windows Forms durante la ejecución del sistema sin necesidad de configurar rutas absolutas o entornos de despliegue más complejos; y una carpeta de imágenes donde están todos los logos de los grupos



Desarrollo

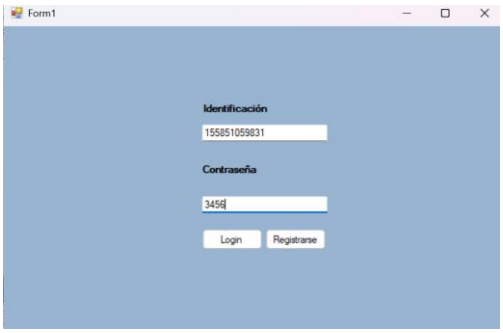
Splitbuddies es un sistema que aborda el problema del control de gastos compartidos entre múltiples usuarios y se desarrolló bajo el enfoque de la programación orientada a objetos, también se implementaron principios como SOLID, Clean Code y el patrón MVC, utilizando el lenguaje C# con Windows Forms. En esta primera iteración se muestra una interfaz gráfica funcional e intuitiva que permite a los usuarios gestionar grupos y gastos compartidos. A continuación, se detalla el proceso de ejecución del sistema y las interacciones fundamentales que ofrece.

Ejecución y Uso del Sistema

Primeramente, el sistema no requiere de instalación. Para ejecutarlo basta con compilar el proyecto en Visual Studio y ejecutar el formulario principal.

1.Inicio del Programa

Al iniciar el programa aparece una pantalla de gestión de usuarios el cual permite registrarse con id, nombre, apellido y contraseña o iniciar sesión con el Id y contraseña el cual contiene validaciones de error.

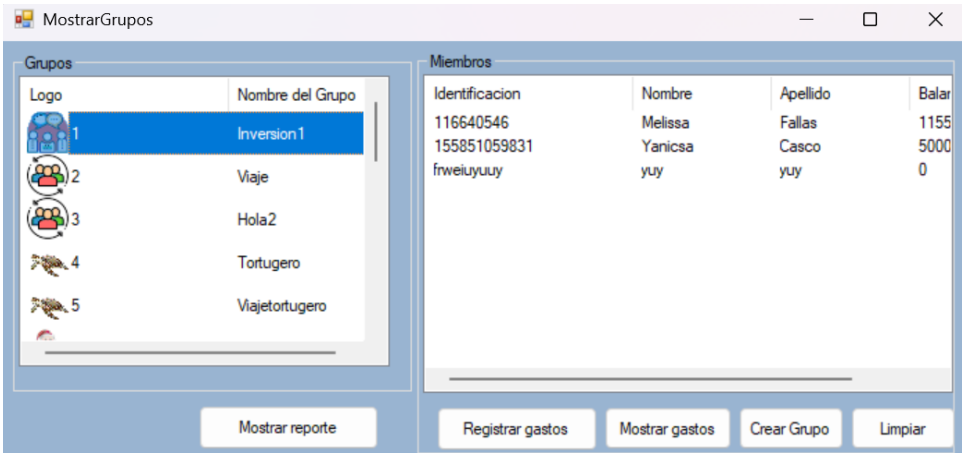


El sistema carga automáticamente archivos JSON con datos simulados: 50 usuarios, 30 grupos, y más de 100 gastos, distribuidos en fechas diversas. Estos archivos se encuentran dentro de la carpeta Assets de la Vista para facilitar su localización

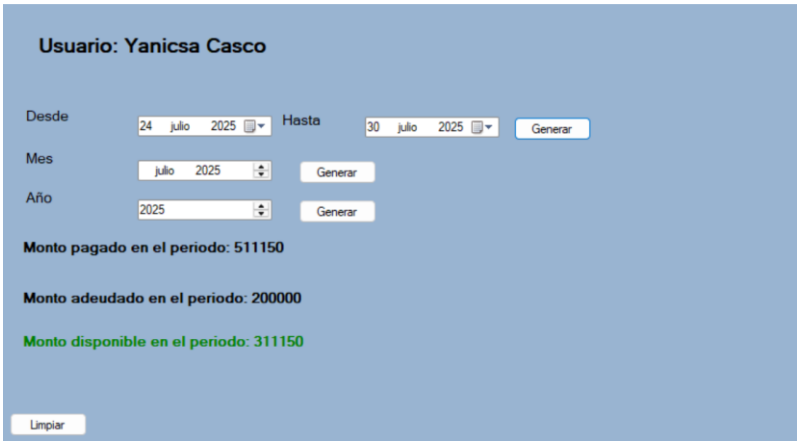
2. Funcionalidades

Una vez se de en registrar o login y esté dentro del programa se mostrará la primera pantalla donde se muestra los grupos y sus miembros, los usuarios pueden visualizar todos los grupos registrados, cada uno representado con su imagen y nombre.

Al seleccionar un grupo se muestran los miembros del grupo con su información y balance. Esta pantalla incluye botones a otras pantallas que serian mostrar reporte, Registrar gastos, mostrar gastos, crear grupo y un botón de limpiar que este lo que hace es que se quita el grupo seleccionado y se deja de mostrar la información de los miembros en pantalla.



Cuando se selecciona el botón de mostrar reporte se despliega la segunda pantalla con el reporte individual del usuario con el que accedió al sistema. Esta sección permite filtrar y visualizar datos financieros en función de un período determinado, seleccionado mediante los campos de fecha. El botón Generar activa el análisis de datos para el rango seleccionado, mientras que Limpiar permite reiniciar la información.



Ahora volviendo a la pantalla de Mostrar Grupos, al seleccionar el siguiente botón que seria Registrar gastos se despliega la tercera pantalla la cual contiene un formulario con campos para ingresar el nombre del gasto, su descripción, un enlace relacionado, el monto involucrado, así como el responsable del pago, los integrantes participantes, la fecha en que se registró y un botón de guardar y otro de cancelar. La inclusión de enlaces y participantes ayuda a mejorar la trazabilidad y claridad de cada gasto, evitando cualquier confusión dentro del grupo.

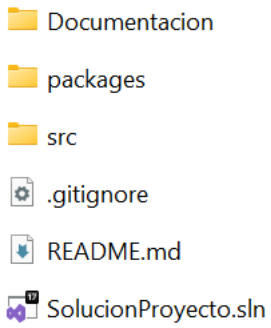
De la misma forma, volviendo a la pantalla principal se selecciona el botón de Mostrar Gastos y se mostrará una pantalla el cual permite seleccionar el usuario y el grupo y un botón de mostrar el balance que mostrará una tabla con información de los gastos del usuario dentro de ese grupo. Debajo de esta tabla, se encuentran datos como el total de gastos del grupo, total por integrante, tu saldo personal, la cantidad de miembros del grupo y los gastos que tú has pagado, si sale en verde esta positivo es decir pagaste lo que debías, si sale rojo es negativo, es decir, que debes dinero.

Por último, si se selecciona el botón de crear grupos desde la pantalla principal se desplegará una pantalla con un formulario con espacios para ingresar datos como nombre del grupo que se desea crear, seleccionar una imagen representativa y elegir a los miembros que se desea integrar. Los botones de Crear y Cancelar permiten confirmar o descartar la creación del grupo

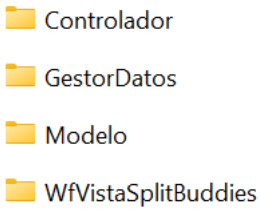
Organización del Proyecto

Para mantener una buena claridad se implementó una estructura de carpetas organizadas según lo visto en clases.

La carpeta principal del repositorio contiene los siguientes directorios:



Dentro de la carpeta SRC se encuentra el código fuente, su estructura MVC y la capa adicional de persistencia denominada Gestor Datos.



Tecnologías Utilizadas:

- ✓ Se utilizo el lenguaje C#
- ✓ Windows Forms para las pantallas
- ✓ Archivos JSON que son la simulación de la base de datos
- ✓ Git y GitHub
- ✓ .NET 9
- ✓ Trello
- ✓ Sonar Analyzer para las buenas prácticas de codificación.

Análisis de Resultados

- La implementación del patrón MVC facilitó la separación clara de responsabilidades.
- La inclusión de una capa intermedia (Gestor de Datos) ayudó a cumplir con el principio de responsabilidad única.
- Las interfaces permitieron un diseño desacoplado, preparado para futuras ampliaciones como integración con bases de datos SQL u otros servicios externos.
- La carga de datos (usuarios, grupos y gastos) mediante archivos JSON simulados funcionó correctamente.

Aprendizajes

- Aplicación de los principios SOLID, en especial el de responsabilidad única y la inversión de dependencias, lo que permitió una arquitectura más limpia y mantenible.
- Comprensión y uso correcto del patrón de diseño MVC, al separar correctamente la lógica del proyecto.
- Uso adecuado de herramientas como Git, GitHub, SonarAnalyzer, diagrama de Gantt y Trello, lo cual ayudó a mantener un flujo de trabajo organizado y más profesional.
- Práctica de Clean Code, adoptando principios como: Nombres de clases, métodos y variables claros y significativos. Métodos cortos y funciones con una sola responsabilidad. Código legible y fácil de mantener para cualquier desarrollador que lo revise posteriormente.
- Diseño de interfaces gráficas con Windows Forms, incluyendo uso de formularios independientes, manejo de eventos y validaciones básicas de usuario.
- Control de versiones con Git y GitHub: Se utilizó para guardar el avance, retroceder cambios y mantener un historial del desarrollo.
- Análisis de calidad del código con Sonar Analyzer: Para detectar errores, mejorar seguridad y mantener buenas prácticas de codificación.

Conclusiones

- Splitbuddies representa una base sólida para una futura plataforma de gestión de gastos colaborativos. Su diseño modular, basado en buenas prácticas de programación orientada a objetos y principios SOLID, permite escalar fácilmente a futuras versiones que podrían incluir almacenamiento en bases de datos.
- El proceso de planificación fue esencial. Usar herramientas como Trello y el diagrama de Gantt para organizar tareas permitió mantener el enfoque, priorizar funcionalidades y evitar retrasos. Aprendimos la importancia de planificar antes de programar.
- El proceso de documentación del proyecto ayudó a consolidar lo aprendido. Al explicar por escrito cada módulo y decisión de diseño, fue más fácil detectar mejoras, repensar soluciones y aclarar la lógica de lo que se estaba construyendo.
- Este proyecto permitió aplicar de forma práctica los conocimientos adquiridos en clase, enfrentarse a problemas reales de diseño y arquitectura, y reforzar habilidades clave para el desarrollo profesional como la colaboración, la documentación y la estructuración lógica del código.

Cosas por mejorar o que no se hicieron

- Aunque es funcional, la interfaz con Windows Forms podría mejorar visualmente, poner mejores colores y acomodar más asimétricamente los labels y buttons.
- Incluir una funcionalidad de búsqueda por nombre, grupo para mayor facilidad y agilidad.
- Una vez que el usuario inicia sesión, no existe una opción para cerrar sesión y volver al formulario de login.