



UNIVERSIDAD POLITÉCNICA INTERNACIONAL

TECNICAS DE PROGRAMACIÓN

PROYECTO 1

SplitBuddies

ESTUDIANTES

MANUEL NAVIDAD VALVERDE

YODAN ESPINOZA CARBALLO

PROFESOR ACADÉMICO

LUIS FELIPE MORA UMAÑA

Índice

Introducción

Decisiones de Diseño y Arquitectura

- 2.1. Estructura del Proyecto
- 2.2. Patrón Modelo-Vista-Controlador (MVC)
- 2.3. Principios SOLID y Clean Code
- 2.4. Patrón de Diseño Aplicado: Singleton

Desarrollo: Guía de Uso y Ejecución

- 3.1. Requisitos del Sistema
- 3.2. Configuración y Ejecución
- 3.3. Interacciones del Sistema: Un Recorrido Visual
 - 3.3.1. Autenticación: Login y Registro
 - 3.3.2. Pantalla Principal: Dashboard de Balances
 - 3.3.3. Gestión de Grupos y Balances Detallados
 - 3.3.4. Gestión de Gastos
 - 3.3.5. Generación de Reportes

Análisis de Resultados, Aprendizajes y Conclusiones

1. Introducción

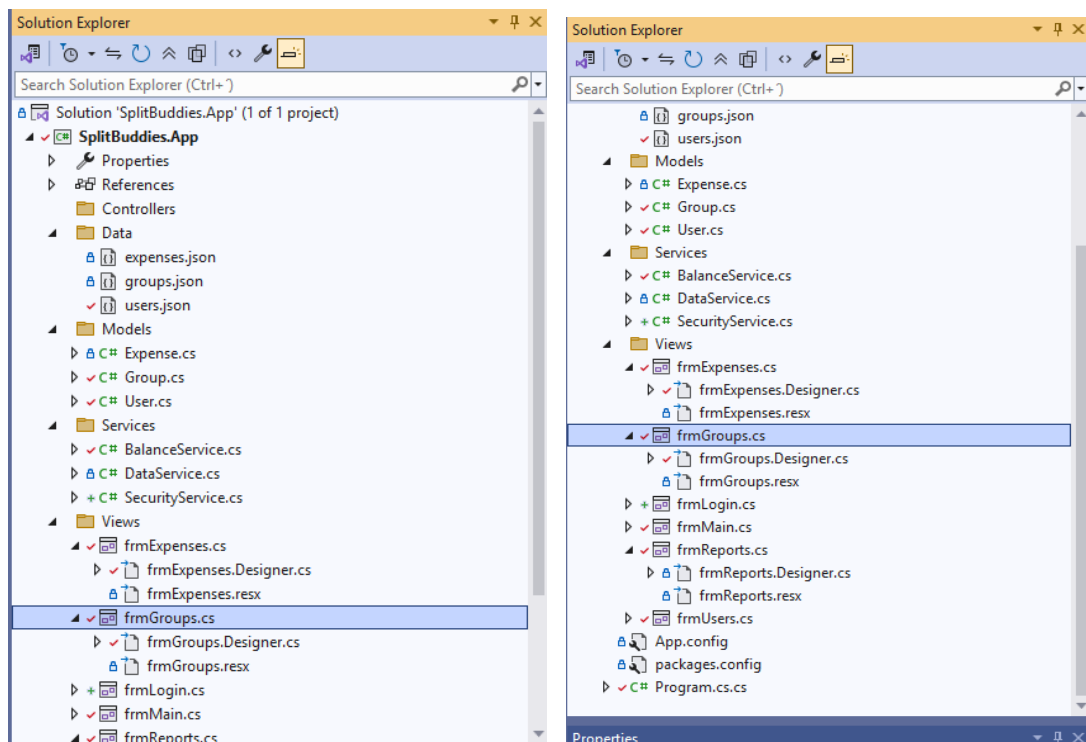
SplitBuddies es una aplicación de escritorio robusta, desarrollada en C# con Windows Forms, que aborda la necesidad de una gestión eficiente y transparente de gastos compartidos. El sistema permite a los usuarios autenticarse de forma segura, gestionar usuarios y grupos, registrar gastos detallados y visualizar balances tanto generales como específicos por grupo.

Este documento técnico detalla la arquitectura de software, las decisiones de diseño y el flujo de interacción del usuario con la aplicación. El proyecto se construyó sobre una base sólida de principios de Programación Orientada a Objetos (POO), el patrón MVC, los principios SOLID y prácticas de Clean Code para garantizar un producto final que no solo es funcional, sino también mantenible, escalable y robusto.

2. Decisiones de Diseño y Arquitectura

2.1. Estructura del Proyecto

El proyecto se organizó en una estructura de carpetas lógica y escalable, separando claramente las responsabilidades, como se puede observar en el Explorador de Soluciones:



Models: Contiene las clases que definen las entidades de datos (User, Group, Expense).

Views: Contiene todos los formularios de la interfaz de usuario (frmLogin, frmMain, etc.).

Services: Contiene la lógica de negocio desacoplada de la interfaz (DataService, BalanceService, SecurityService).

Data: Almacena los archivos de persistencia de datos (users.json, etc.).

2.2. Patrón Modelo-Vista-Controlador (MVC)

El patrón MVC fue la piedra angular de la arquitectura para garantizar la separación de conceptos:

Modelo: Compuesto por las clases en Models/ y Services/. Se encarga de la lógica de negocio, el acceso a datos (lectura/escritura de archivos JSON) y los cálculos complejos (balances, hashing de contraseñas).

Vista: Los formularios de Windows Forms en Views/. Su única función es presentar los datos al usuario y capturar sus interacciones.

Controlador: El código "code-behind" de cada formulario (.cs). Actúa como el cerebro que responde a los eventos de la Vista, solicita al Modelo que procese la información y actualiza la Vista con los resultados.

2.3. Principios SOLID y Clean Code

Principio de Responsabilidad Única (SRP): Cada clase tiene un propósito definido. SecurityService solo cifra contraseñas, BalanceService solo calcula balances, y DataService solo gestiona la persistencia.

Nombres Significativos: Se utilizaron nombres descriptivos para variables, métodos y controles (btnManageUsers, RefreshBalances) para que el código sea autoexplicativo.

Manejo de Excepciones: Se implementaron bloques try-catch en operaciones críticas (lectura/escritura de archivos) para evitar que la aplicación falle y para informar al usuario de posibles problemas, garantizando la robustez del sistema.

2.4. Patrón de Diseño Aplicado: Singleton

Se implementó el patrón Singleton en la clase DataService. Esto asegura que solo exista una única instancia de DataService en toda la aplicación, proporcionando un punto de acceso global y consistente a los datos en memoria y evitando conflictos de sincronización.

3. Desarrollo: Guía de Uso y Ejecución

3.1. Requisitos del Sistema

Windows con .NET Framework 4.7.2 o superior.

Paquete NuGet Newtonsoft.Json.

3.2. Configuración y Ejecución

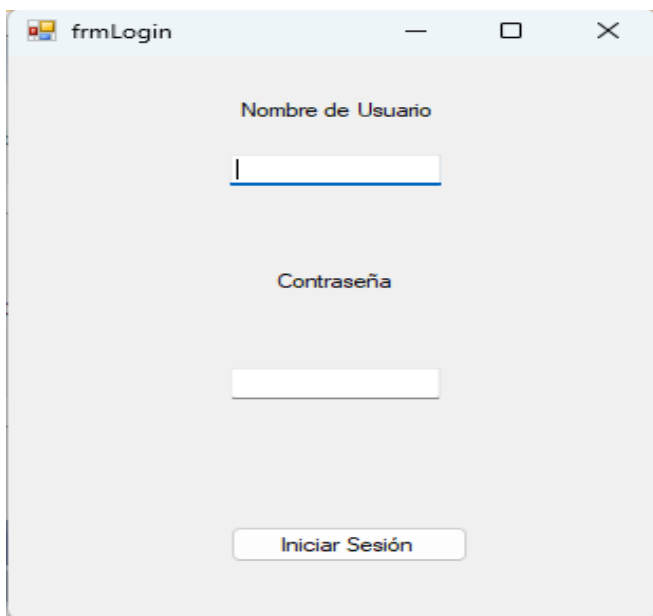
Abrir la solución .sln en Visual Studio.

Asegurarse de que los archivos users.json, groups.json, y expenses.json estén en la carpeta Data y configurados para (Copiar si es posterior).

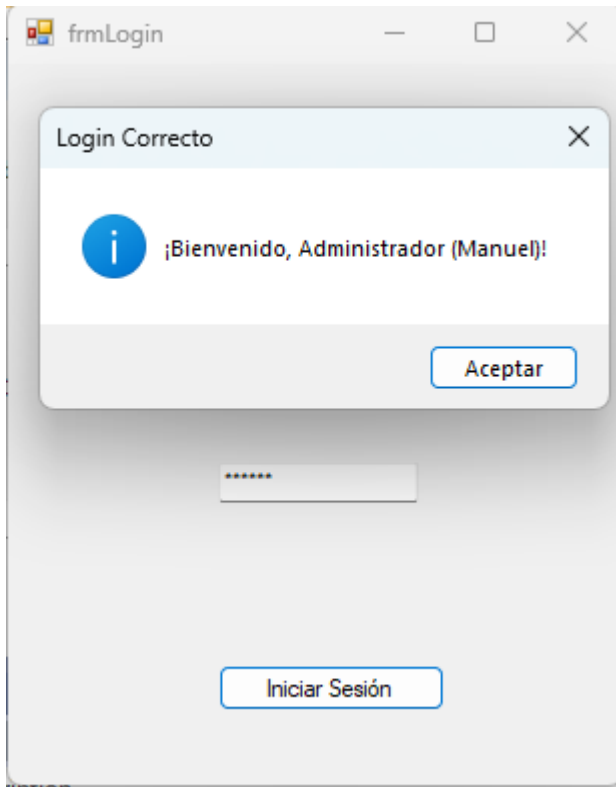
Presionar F5 o el botón (Iniciar) para compilar y ejecutar la aplicación.

3.3. Interacciones del Sistema: Un Recorrido Visual

El flujo de la aplicación comienza con una pantalla de autenticación segura.



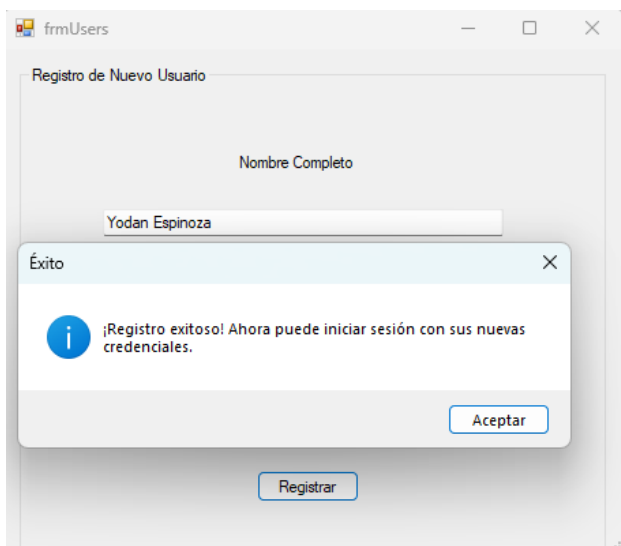
El usuario debe ingresar sus credenciales. Si el login es exitoso, el sistema le da la bienvenida.



Si un usuario no tiene cuenta, puede registrarse. El formulario de registro (frmUsers) solicita la información necesaria.

The image shows a Windows application window titled 'frmUsers'. The form is titled 'Registro de Nuevo Usuario'. It contains three input fields: 'Nombre Completo' with the text 'Yodan Espinoza', 'Nombre de Usuario (para login)' with the text 'Yodan.Espinoza', and 'Contraseña' with six dots. At the bottom center is a button labeled 'Registrar'.

Tras un registro exitoso, el sistema lo confirma y el nuevo usuario puede proceder a iniciar sesión.



Una vez autenticado, el usuario accede al dashboard principal (frmMain), que muestra una tabla con el balance general de todos los usuarios del sistema.

The image shows a software window titled 'frmMain'. On the left side, there is a sidebar with four buttons: 'Gestionar Usuarios', 'Gestionar Grupos', 'Gestionar Gastos', and 'Ver Reportes'. At the bottom of the sidebar is an 'Actualizar' button. The main area of the window displays a table with two columns: 'Usuario' and 'Balance'.

Usuario	Balance
Ana García	₡4 444,41
Luis Martínez	₡727,79
Sofía Rodríguez	₡97,00
Carlos Sánchez	₡350,79
Lucía Pérez	-₡1 317,72
Javier Gómez	₡1 056,00
Elena Fernández	₡636,50
David Moreno	-₡193,58
Laura Jiménez	₡199,15
Daniel Díaz	-₡895,39
Paula Ruiz	-₡79,45
Adrián Vázquez	-₡42,75
Marta Castro	-₡38,75
Pablo Serrano	₡2,75
Carmen Ramos	-₡1 509,24

Al hacer clic en "Gestionar Grupos", se abre la ventana frmGroups. Aquí el usuario puede crear nuevos grupos y, lo más importante, ver un desglose detallado de quién debe a quién dentro de un grupo seleccionado.

Crear Nuevo Grupo

Nombre del Grupo

Ruta de Imagen (opcional)

Seleccionar Miembros

- ☐ Rocio Medina
- ☐ Mario Santana
- ☐ Alicia Domínguez
- ☐ José Vicente
- ☐ Administrador (Manuel)
- ☐ Yodan Espinoza

Crear Grupo

Grupos Existentes

Nombre	Miembros
Piso Compartido Centro	Ana García, Lucía Pérez, Daniel Díaz, Car...
Viaje a Roma	Luis Martínez, Javier Gómez, Adrián Vázqu...
Equipo de Fútbol Sala	Sofía Rodríguez, Elena Fernández, Marta ...
Cena de Navidad 2023	Carlos Sánchez, David Moreno, Pablo Serr...
Club de Lectura	Laura Jiménez, Paula Ruiz, Sergio Romero,...
Vacaciones en la Montaña	Ana García, Luis Martínez, Sofía Rodrígue...
Proyecto Universidad	Isabel Suárez, Alejandro Vega, Óscar Peña...
Compañeros de Oficina	Guillermo Santos, Esther Lorenzo, Roberto ...
Fin de Semana Rural	Eva Herrero, Francisco Núñez, Silvia Aguil...
Regalo Cumpleaños Laura	David Moreno, Laura Jiménez, Daniel Díaz,...
Amigos del Gimnasio	Átor Ferrer, Lorena Castillo, César Ortega, ...
Concierto Rock Fest	Luis Martínez, Carlos Sánchez, Javier Góm...

Balance Detallado del Grupo

Lucía Pérez debe a Ana García: €1 447,49
Daniel Díaz debe a Ana García: €1 445,54
Carmen Ramos debe a Ana García: €1 525,99

Como se observa, al seleccionar "Piso Compartido Centro", el panel "Balance Detallado del Grupo" se actualiza en tiempo real, mostrando las deudas simplificadas entre los miembros.

La ventana frmExpenses es el núcleo funcional. Permite registrar gastos de forma detallada. La interfaz es dinámica: al seleccionar un grupo, los ComboBox de "Pagado por" y "Participantes" se actualizan para mostrar solo los miembros de ese grupo.

Registrar Nuevo Gasto

Grupo

Monto

Fecha

Nombre del Gasto

Descripción

Pagado por

Enlace (opcional)

Participantes

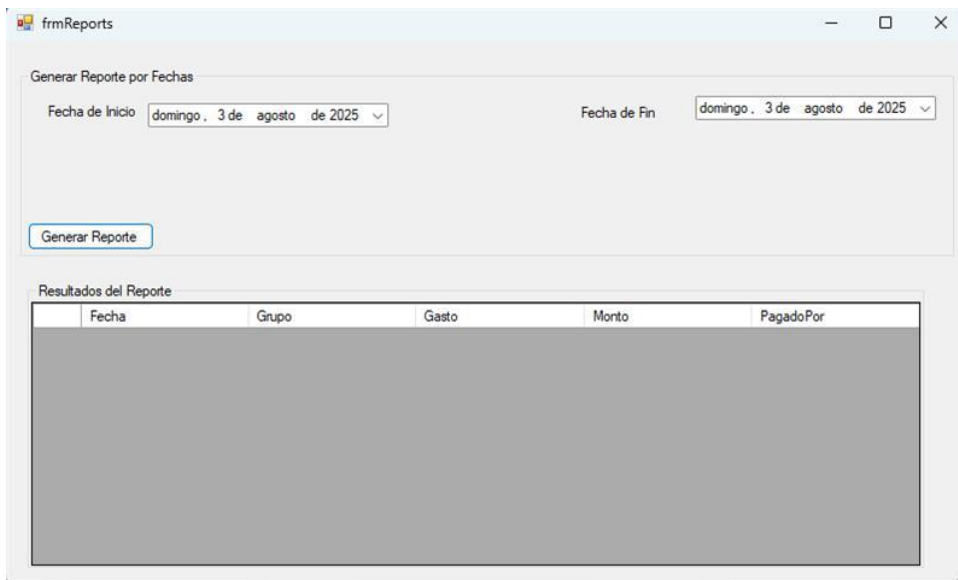
- ☒ Ana García
- ☐ Lucía Pérez
- ☐ Daniel Díaz
- ☐ Carmen Ramos

Agregar Gasto

Gastos Registrados

Grupo	Gasto	Monto	Fecha	PagadoPor
Equipo de Baloncesto	Alquiler de pista	€50,00	8/5/2024	Luis Martínez
Vacaciones en la Montaña	Actividades de aventura	€200,00	6/4/2024	Sofía Rodríguez
Vacaciones en la Montaña	Cena en el pueblo	€160,00	6/4/2024	Carlos Sánchez
Proyecto de Software	Licencias de software	€300,00	5/7/2024	Carlos Sánchez
Gastos del Hogar (Familia...)	Factura de Internet	€45,00	5/6/2024	Lucía Pérez
Piso Compartido Centro	Alquiler Mayo	€1 200,00	5/5/2024	Ana García
Vacaciones en la Montaña	Gasolina viaje	€70,00	5/4/2024	Ana García
Vacaciones en la Montaña	Casa Rural	€400,00	5/4/2024	Luis Martínez
Piso Compartido Centro	Alquiler Abril	€1 200,00	5/4/2024	Ana García

La sección de reportes (frmReports) permite al usuario filtrar y visualizar todos los gastos ocurridos dentro de un rango de fechas específico, proporcionando una herramienta útil para el análisis financiero.



The screenshot shows a Windows application window titled "frmReports". It contains a section for generating reports by date range. The "Fecha de Inicio" (Start Date) and "Fecha de Fin" (End Date) are both set to "domingo, 3 de agosto de 2025". A "Generar Reporte" button is located below the date fields. Below the button is a section titled "Resultados del Reporte" which contains a table with the following columns: Fecha, Grupo, Gasto, Monto, and PagadoPor. The table body is currently empty.

Fecha	Grupo	Gasto	Monto	PagadoPor
-------	-------	-------	-------	-----------

4. Análisis de Resultados, Aprendizajes y Conclusiones

El desarrollo de SplitBuddies ha sido un éxito, cumpliendo con todos los requisitos funcionales y técnicos. El resultado es una aplicación de escritorio completamente funcional, robusta y con una arquitectura bien definida.

Aprendizajes Clave:

La importancia de una **arquitectura planificada (MVC)** fue fundamental para evitar código acoplado y facilitar la depuración.

La implementación del **Principio de Responsabilidad Única** a través de clases de servicio (DataService, BalanceService) simplificó enormemente la lógica en los controladores y aumentó la reutilización del código.

Enfrentar y resolver los **errores del diseñador de Windows Forms** fue un aprendizaje práctico crucial, subrayando la necesidad de entender cómo el IDE gestiona los archivos de código y de diseño.

Conclusiones

SplitBuddies demuestra la aplicación exitosa de los principios de la Programación Orientada a Objetos y patrones de diseño en un proyecto práctico. La base de código actual es sólida, mantenible y está preparada para futuras expansiones, como la integración con una base de datos real o la implementación de una API para una versión móvil. El proyecto no solo cumple con las especificaciones, sino que también establece un estándar de calidad para el desarrollo de software futuro.