Documentación Técnica, Sistema De Gestión Para Gimnasio

Proyecto - III Cuatrimestre 2024

Castro Mora Fabiola María,

Santamaría Allen Akane,

Ingeniería Informática

Universidad Politécnica Internacional

Técnicas de Programación

Mora Umaña Luis Felipe

Noviembre, 2024

Contenido

Introducción

En el presente proyecto se llevó a cabo el desarrollo de un sistema para la gestión de un gimnasio, con el objetivo de facilitar la administración de clientes, entrenadores, membresías y reservas de clases. Este proyecto busca ofrecer una solución práctica y eficiente que permita registrar a los usuarios, planificar actividades y organizar los recursos del gimnasio de manera sencilla.

El trabajo se centró en construir un programa funcional y bien estructurado, utilizando herramientas y métodos que aseguren un manejo claro y organizado de la información. A través de formularios diseñados para registrar e iniciar sesión, los usuarios pueden interactuar fácilmente con el sistema. Además, se incorporaron mecanismos para validar los datos ingresados, garantizando que la información sea correcta y confiable.

A lo largo del desarrollo, se enfrentaron diversos desafíos, como la creación de archivos para guardar los datos y la adaptación del sistema a diferentes tipos de usuarios, como clientes y entrenadores. Estos retos permitieron mejorar habilidades en la organización y solución de problemas, resultando en un sistema que responde a las necesidades del gimnasio y de sus usuarios. Este proyecto representa un esfuerzo significativo por aprender y aplicar los conocimientos adquiridos, buscando siempre un enfoque práctico y orientado a resultados.

Decisiones de Diseño

Durante el desarrollo del sistema, se tomaron varias decisiones de diseño que permitieron estructurar y organizar el proyecto de forma eficiente. Estas decisiones facilitaron la implementación de las funcionalidades requeridas, así como la escalabilidad y el mantenimiento del sistema.

Organización del Código

El sistema se organizó utilizando una estructura de carpetas que separa los diferentes componentes del proyecto:

Controller: Contiene las clases encargadas de manejar la lógica de la aplicación y el acceso a los datos, como FileDataHandler y PersonController.

DataHandler: Incluye las interfaces y clases necesarias para la manipulación de los datos almacenados en los archivos CSV, diferenciando entre clientes y entrenadores.

Model: Agrupa las clases relacionadas con la lógica de negocio, como Cliente, Entrenador, Factura, y Clase, lo que permite mantener una separación clara entre los datos y las operaciones que los manipulan.

Personas: Contiene las clases que representan las entidades de usuarios, como Cliente y Entrenador, implementando interfaces para garantizar la flexibilidad y adherirse a principios de programación orientada a objetos.

Forms (Windows Forms): Los formularios del sistema, como RegistroDeUsuario, InicioDeSesion, y FRMFacturas, se organizan en carpetas independientes, lo que permite una navegación más clara y facilita futuras modificaciones.

Diseño de la Interfaz de Usuario

Se diseñaron varios formularios para interactuar con los diferentes módulos del sistema:

***Formulario de Inicio de Sesión*:** Incluye validaciones para asegurar que el usuario y la contraseña coincidan con los registros existentes en los archivos CSV.

***Formulario de Registro de Usuarios*:** Permite registrar clientes y entrenadores en archivos separados, asegurando que los datos ingresados sean válidos y no duplicados.

***Formularios Funcionales*:** Incluyen interfaces para gestionar horarios, facturas, ingresos, inventarios, entre otros. Cada formulario está diseñado para ser intuitivo y funcional.

Gestión de Datos

***Uso de Archivos CSV*:** Los datos de clientes y entrenadores se almacenan en archivos CSV separados. Esta elección simplifica el manejo de los datos y garantiza la portabilidad del sistema.

***Rutas Relativas*:** Se utilizaron rutas relativas para garantizar que el programa pueda ejecutarse en cualquier máquina sin necesidad de configuraciones adicionales.

Validación de Índices: Debido a las diferencias en las estructuras de los CSV de clientes y entrenadores, se implementaron validaciones específicas para cada tipo de usuario, asegurando que la lectura y escritura de datos sean correctas.

Modularidad

El diseño modular del sistema permite que cada componente cumpla con una única responsabilidad:

***Controladores*:** Manejan la interacción entre los formularios y los datos.

***Modelos*:** Representan las entidades y encapsulan su lógica de negocio.

***Manejo de Datos*:** Centraliza las operaciones de lectura y escritura en los archivos, permitiendo un fácil mantenimiento.

Escalabilidad y Mantenimiento

Se optó por un diseño que permite agregar nuevas funcionalidades, como otros tipos de usuarios o integraciones con bases de datos, sin afectar la estructura existente. Además, la separación de responsabilidades asegura que los cambios en un módulo no impacten negativamente en otros.

Manejo de Errores

Se implementaron mensajes de error y advertencias en los formularios para guiar al usuario en caso de fallos, como campos vacíos, datos inválidos o intentos de registro duplicados. Esto mejora la experiencia del usuario y reduce posibles confusiones al interactuar con el sistema.

Con esta organización y diseño, el sistema "ProyectoGym" no solo cumple con los requerimientos iniciales, sino que también se establece como una solución flexible y adaptable a futuras necesidades.

Desarrollo del Sistema

Uso del Programa

Como utilizar y ejecutar el programa

Requisitos previos

- Tener C# y las bibliotecas necesarias instaladas.
- Acceso a la base de datos configurada.

Pasos para ejecución

- Clonar el repositorio desde GitHub.
- Configurar las credenciales de la base de datos en el archivo config.json.
- Ejecutar el archivo principal del proyecto (Main.cs).

Instrucciones para el usuario

Gestión de usuarios: Permitirá crear clientes y entrenadores con roles específicos.

Gestión de membresías: Activará alertas para membresías por vencer.

Reservas de clases: Reservará clases según disponibilidad en el calendario interactivo del cliente.

Ejecución

Utilización del programa.

Registro de Usuarios: Los administradores pueden crear y gestionar los perfiles de los entrenadores y clientes. Los entrenadores pueden configurar sus horarios y especializaciones, mientras que los clientes pueden visualizar y reservar clases.

Gestión de Membresías: El sistema requiere los datos de almacenamiento necesarios que el usuario debe llenar para crear la membresía a excepción que falta la parte lógica para almacenar datos y enviar notificaciones a los entrenadores.

Gestión de Clases y Reservas: Los entrenadores y clientes pueden crear nuevas clases, asignando detalles como horarios y niveles de dificultad. Los clientes pueden visualizar un calendario de clases y reservar espacios en las que estén interesados.

Gestión de Inventario: El sistema permite agregar, editar y eliminar equipos de gimnasio. Además, registra el uso de equipamiento en cada clase para mantener un control de los recursos.

Generación de Reportes: Los entrenadores pueden generar reportes detallados sobre las clases, inventario, la asistencia y el crecimiento de las membresías.

Facturación: Los usuarios pueden consultar sus facturas en cualquier momento.

Análisis de Resultados

Retos Encontrados

Problemas con la Integración de los archivos .csv : Inicialmente, se presentaron errores al conectar el sistema con los .csv, debido a configuraciones inconsistentes en los entornos de desarrollo.

Solución: Se estandarizó el archivo, vimos la clase del profesor, incluso evacuamos dudas.

Gestión de Colisiones en GitHub: Durante la colaboración en equipo, surgieron conflictos al fusionar ramas con cambios simultáneos en archivos principales.

Solución: Se implementaron mejores prácticas, como crear ramas específicas para cada tarea y realizar revisiones por medio de pull requests antes de fusionar los cambios en la rama principal.

Validaciones Complejas en la Creación de Clases: Hubo dificultades para asegurar que los datos ingresados por los entrenadores y clientes (como los datos) fueran coherentes y no conflictivos.

Solución: Se añadieron validaciones detalladas en los formularios, tanto del lado del cliente como del usuario junto con la clase cliente, y se implementaron pruebas automatizadas para verificar todos los escenarios posibles.

Gestión del Tiempo: Algunas tareas tomaron más tiempo del previsto, especialmente las relacionadas con la generación de reportes visuales, en general los Forms para entrenadores para las clases y las membresías.

Solución: Se ajustó el cronograma en Jira y se redistribuyeron tareas para garantizar que se completaran las funcionalidades prioritarias dentro del plazo establecido.

Para concluir, a pesar de los retos enfrentados, los logros alcanzados demuestran la efectividad de la planificación y las buenas prácticas utilizadas en el desarrollo del

Análisis de Aprendizaje y Conclusiones

Manejo de Herramientas de Desarrollo Colaborativo

El uso de GitHub para el control de versiones permitió un flujo de trabajo más organizado, asegurando que cada miembro del equipo pudiera trabajar en ramas independientes y fusionar cambios mediante pull requests. Esto fortaleció la capacidad para resolver conflictos de código y mantener una versión estable del proyecto.

Jira resultó fundamental para la gestión del proyecto, ya que facilitó el seguimiento de tareas, la asignación de responsabilidades y la visualización del progreso mediante el diagrama de Gantt.

Aplicación de Buenas Prácticas de Programación

La implementación de los principios SOLID y Clean Code mejoró la calidad del código, haciéndolo más legible, mantenible y fácil de escalar.

La adopción del patrón Modelo-Vista-Controlador (MVC) ayudó a organizar el proyecto, separando la lógica de negocio, la interfaz de usuario y la gestión de datos, lo que facilitó el desarrollo colaborativo y la resolución de problemas.

Trabajo en Equipo y Resolución de Problemas

El trabajo colaborativo permitió aprender a coordinar tareas entre distintos integrantes, resolviendo retos técnicos y de comunicación.

La experiencia adquirida al realizar pruebas y ajustes en entornos reales consolidó habilidades para depurar errores y optimizar funcionalidades.