Universidad Politécnica Internacional

Ingeniería Informática

Año 2024

Tercer cuatrimestre 2024

Curso: Técnicas de Programación

Código: 90501

Profesor:

Luis Felipe Mora Umaña

Alumnos:

María Fernanda Alemán Ruiz

Jonathan Morales Barrientos

Proyecto # 1

Sistema de Gestión de Gimnasio

Índice

1. Introducción

Este documento describe el desarrollo del Sistema de Gestión de Gimnasio, una aplicación diseñada para facilitar la administración de las operaciones de un gimnasio. El proyecto se centra en el uso de C# y el patrón de arquitectura MVC, con un fuerte enfoque en los principios de Programación Orientada a Objetos (POO), Clean Code y SOLID. El objetivo es crear un sistema escalable y mantenible que permita gestionar usuarios, membresías, reservas, inventario, reportes y facturación.

2. Objetivo

Objetivo Principal

Desarrollar un sistema de gestión integral para un gimnasio, que permita manejar de manera eficiente las operaciones de los usuarios, las membresías, las clases (Spinning, zumba, Cardio, etc), reservas de clases, inventario de máquinas, manejo de reportes y Sistema de facturación.

Objetivos Secundarios

1. Implementar una estructura modular basada en el patrón MVC dentro de C#, que nos permita la separación clara entre la lógica del negocio, la interfaz del usuario y la relacion de los datos.
2. Aplicar principios de Clean Code y Principios SOLID para asegurar la claridad, la legibilidad y mejorar la eficiencia del código, asegurando la escalabilidad para futuras modificaciones.
3. Automatizar y optimizar los procesos administrativos como la notificación de vencimientos de membresías a los usuarios(clientes), la reserva de clases, la gestión de inventario y la generación de reportes para el control interno.

3. Requerimientos

Requerimientos Funcionales

- Gestión de los usuarios (clientes y entrenadores) con manejo de datos por medio de un archivo CSV o JSON.
- La administración de membresías, incluyendo notificaciones de vencimiento para la comodidad de los clientes.
- Reservas de clases y visualización de reservas para los entrenadores.
- Gestión del inventario de los equipos (maquinas) y notificación de vida útil para los administradores del sistema.
- Generación de los reportes sobre las membresías y de las clases con mayor asistencia.
- Proceso de facturación mensual automática de las membresías (para clientes habituales) y proceso de facturación para manejo de clientes nuevos.

Requerimientos No Funcionales

- El sistema debe estar estructurado de manera modular, que sea fácil de mantener y escalable en el tiempo para incluir mejoras.
- Las vistas deben implementarse por medio de WinForms de acuerdo con el requerimiento inicial.
- La carga de datos deberá realizarse por medio de archivos de tipo CSV o JSON (con al menos 100 registros).

4. Desarrollo

4.1 Diseño del Sistema

4.1.1 Patrones del diseño a aplicar

Se utilizará el patrón Modelo-Vista-Controlador (MVC) en C# para dividir el sistema en tres capas principales: la capa de presentación (vistas), la capa del negocio (controlador) y la capa de los datos (modelo).

4.1.2 Principios SOLID y Clean Code

El diseño seguirá los principios SOLID de acuerdo a lo visto en clase:

- Principio de Responsabilidad única: Cada clase deberá tener una única responsabilidad.
- Principio abierto / cerrado: Las clases estarán abiertas para la extensión, pero cerradas para su modificación.
- Principio de Sustitución de Liskov: Las clases hijas podrán heredar las propiedades de las clases padres.
- Principio de Segregación de Interfaces: Se deberá contar con interfaces más pequeñas en lugar de un interfaz general.
- Principio de Inversión de Dependencias: Las clases deberán depender de abstracciones más no de detalles específicos.

4.1.3 Estructura del proyecto

El proyecto deberá estar organizado en carpetas de acuerdo con el patrón MVC y la lógica del negocio, con carpetas adicionales para el manejo de Datos, Servicios y controladores.

4.2 La Arquitectura del Sistema

4.2.1 Modelo-Vista-Controlador (MVC)

La arquitectura MVC divide el sistema en tres capas:

- **Modelo:** Representará los datos y la lógica del negocio.
- **Vista:** Se implementará en WinForms, permitiendo la interacción con el usuario (pantallas).
- **Controlador:** Gestionará las solicitudes de los usuarios y actualizará las vistas y los modelo.

4.2.2 Estructura de clases y herencia (POO)

Las clases para Clientes y Entrenadores heredarán de la clase Usuario, aplicando el encapsulamiento y el polimorfismo para gestionar sus comportamientos específicos.

4.3 Gestión y manejo de Usuarios

4.3.1 Clases de usuario (Cliente y Entrenador)

Se crearán las clases Cliente y Entrenador que heredarán de la clase base Usuario, para diferenciar sus datos y funcionalidades de acuerdo con cada rol.

4.3.2 Autenticación de los usuarios

Se implementará un formulario de inicio de sesión para validar las credenciales de los usuarios (Log In).

4.3.3 Notificación de vencimiento de membresía

Se implementará un sistema de notificación para avisar al cliente cuando su membresía esté próxima a vencer. Para control en ambas vías, tanto administrador como cliente.

4.4 Gestión de las Membresías

4.4.1 Clase Membresía

Dentro de esta clase se definirán las propiedades como FechaInicio, FechaVencimiento, y Costo, que permitirán una buena gestión y control de las membresías de cada cliente.

4.4.2 Renovación de membresías

Se implementará un sistema que permita la renovación de las membresías directamente desde el formulario de los clientes.

4.4.3 Notificación de vencimiento

Dentro del sistema se implementará notificaciones cuando la membresía está próxima a vencer para cada cliente.

4.5 Gestión de Clases y Reservas

4.5.1 Clases de las actividades (Clases y Reservas)

Dentro del sistema se implementarán las clases Clase y Reserva que permitirán la administración de las actividades y reservas de los clientes.

4.5.2 Reservas de clases por clientes

Los clientes podrán reservar un espacio en una clase específica por medio de un calendario.

4.5.3 Visualización de reservas por entrenadores

Los entrenadores podrán ver quiénes han reservado en sus clases para mayor control.

4.6 Gestión de Inventario

4.6.1 Clase Inventario

Registra las maquinas del gimnasio y su estado de vida útil.

4.6.2 Notificación de vida útil de los equipos

Se notificará a los entrenadores del sistema cuando un equipo está próximo a cumplir su vida útil con cada inicio de sesión.

4.7 Generación de Reportes

4.7.1 Reporte de crecimiento de las matrículas

Mostrará la cantidad de nuevos clientes en un período de tiempo.

4.7.2 Reporte de popularidad de las clases(actividades)

Indicará las clases más reservadas.

4.7.3 Reporte contable de ingresos vs. egresos

Se compara los ingresos del gimnasio con los gastos en la compra de equipos.

4.8 Facturación

4.8.1 Clase Factura

Se registrarán las facturas emitidas a los clientes por sus membresías pago de mensualidad.

4.8.2 Generación y almacenamiento de facturas

Se contará con un repositorio (base de datos) para el almacenamiento de las facturas emitidas (membresía y pago de mensualidad por cliente).

4.9 Carga de Datos desde Archivos

4.9.1 Estructura de los archivos CSV o JSON

Se cargará la información básica de los usuarios, las clases (actividades) y los horarios desde archivos en formato CSV o JSON.

4.9.2 Carga de usuarios, clases y horarios

Los datos iniciales son precargados en el sistema a partir de archivos de configuración.

5. Herramientas de Gestión del Proyecto

5.1 Jira

5.1.1 Especificaciones

La herramienta Jira se utilizará para gestionar las tareas y el flujo de trabajo del proyecto.

5.1.2 Uso de historias de usuario y tareas

El proyecto estará organizado en historias de usuario y tareas que representan cada funcionalidad principal (Epica y Features).

5.2 Git

5.2.1 Especificaciones

Git permite el control de versiones, facilitando la colaboración y el seguimiento de cambios durante el proceso del proyecto.

5.2.2 Estrategia de control de versiones

Se utiliza una estructura de ramas (main, develop, feature) para organizar el desarrollo del proyecto.

6. Análisis de Aprendizaje

Durante el desarrollo y de este proyecto se adquirirán conocimientos sobre la implementación de un sistema en C# utilizando una estructura y el patrón MVC, los principios de POO y Clean Code, y así como el uso de las herramientas de gestión como Jira y GitHub para organizar y controlar el flujo del proyecto.

Epic 1. ProyectoGimnasio

Feature 1.1: Configuración Inicial y Modelos

Product Backlog Item 1.1.1: Configuración del Proyecto y Herramientas

Descripcion: Como desarrolladores se debe realizar una planificación del proyecto y elegir una herramienta (Jira) que vaya de la mano con la elaboración de este.

Criterio de aceptación:

Tareas:

- Reunión del equipo
- Elección de herramienta de agilidad.
- Lluvia de ideas.

Product Backlog Item 1.1.2: Configurar Jira y crear el tablero de seguimiento.

Descripción: Como desarrolladores se necesita una herramienta para seguimiento y asignación de las tareas, utilizando Jira.

Criterio de aceptación: En la herramienta se puede ver el lapso de tiempo de cada subtarea(feature).

Tareas:

- Crear cuenta en Jira.
- Crear tablero.
- Asignar tareas a los integrantes del equipo.
- Añadir las épicas y subtareas.

Product Backlog Item 1.1.3: Configurar Git y crear el repositorio master.

Descripción: Como desarrolladores se debe utilizar Git para los trabajos colaborativos.

Criterio de aceptación: Se deben de ver los commits o actualizaciones del código por parte del equipo de desarrollo.

Tareas:

- Crear cuenta en GitHub.

- Crear repositorio.
- Enlazar Visual Studio 2022 con GitHub.

Product Backlog Item 1.1.4: Definir el flujo de trabajo (ramas y estrategias de merge).

Descripcion: Como desarrolladores se debe definir una estrategia para trabajo en conjunto en las ramas clonadas.

Criterio de aceptacion: Cada integrante deberá trabajar en las ramas asignadas (features) y subir los cambios al repositorio

Tareas:

- Creación de rama principal
- Creación de ramas clonadas

Feature 1.2: Desarrollo de la Capa de Datos (Models)

Product Backlog Item 1.1.1: Implementar Modelos de Usuario: Usuario, Cliente, Entrenador.

Descripcion: Como desarrolladores se deberá crear dentro del modelo del proyecto las capas principales del sistema.

Criterio de aceptación: Creación dentro Visual Studio de los modelos de Usuario.

Tareas:

- Crear clase Usuario
- Crear clase Cliente
- Crear clase Entrenador

Product Backlog Item 1.1.2: Implementar Modelos de Clases y Reservas: Clase, Reserva.

descripción: Como desarrolladores se deberá crear dentro del modelo del proyecto las clases(actividades) y reservas.

Criterio de aceptación: Creación dentro de Visual Studio de los modelos de Clases(actividades) y Reserva.

Tareas:

- Crear clase Reserva
- Crear clase Clases(actividades)

Product Backlog Item 1.1.3: Implementar Modelos de Inventario: Maquina.

Descripción: Como desarrolladores se deberá crear dentro del modelo del proyecto las clases Inventario.

Criterio de aceptación: Creación dentro de Visual Studio del modelo Inventario.

Tareas:

- Crear clase Inventario

Product Backlog Item 1.1.4: Implementar Modelos de Facturación: Factura.

Descripción: Como desarrolladores se deberá crear dentro del modelo del proyecto la clase Factura.

Criterio de aceptación: Creación dentro de Visual Studio de los modelos de Facturación.

Tareas:

- Crear clase Factura.

Feature 1.3: Controladores y Lógica del Negocio

Product Backlog Item 1.1.1: Controlador de Usuarios: Inicio de sesión, carga de datos desde archivos, notificaciones.

Descripción: Como desarrolladores se deberá crear dentro del modelo del proyecto el Controlador de Usuario.

Criterio de aceptación: Creación dentro de Visual Studio del Controlador de Usuario.

Tareas:

- Crear clase Controlador Usuario.

Product Backlog Item 1.1.2: Controlador de Clases y Reservas: Listado de clases, reservas, asignación de entrenadores.

Descripción: Como desarrolladores se deberá crear dentro del modelo del proyecto el Controlador de Clases y Reservas.

Criterio de aceptación: Creación dentro de Visual Studio del Controlador de Clases y Reservas.

Tareas:

- Crear clase Controlador de Clases y Reservas.

Product Backlog Item 1.1.3: Verificar vida útil de las máquinas, listar inventario.

Descripción: Como desarrolladores se deberá crear dentro del modelo del proyecto la lógica para la verificación de la vida de las máquinas y listado del inventario.

Criterio de aceptación: Creación dentro de Visual Studio de la lógica para calcular la vida útil de las máquinas y generar un reporte del inventario.

Tareas:

- Desarrollar la lógica para el cálculo de la vida útil de las maquinas.
- Desarrollar el reporte del inventario

Product Backlog Item 1.1.4: Generación de reportes de matrícula, informes contables, clases más atractivas.

Descripción: Como desarrolladores se deberá crear dentro del modelo del proyecto la lógica la generación de reportes de matrícula, informes contables, clases más atractivas.

Criterio de aceptación: Creación dentro de Visual Studio de la lógica para la generación de reportes de matrícula, informes contables, clases más atractivas.

Tareas:

- Desarrollar la lógica para la generación de reportes de matrícula.
- Desarrollar la lógica para la generación de reportes de informes contables.
- Desarrollar la lógica para la generación de reportes de clases más atractivas.

Feature 1.4: Gestión de Archivos de Configuración

Product Backlog Item 1.1.1: Configurar la carga de datos desde archivos CSV/JSON.

Descripción: Como desarrolladores se deberá crear un documento por cada clase, con los registros solicitados para la carga de la información.

Criterio de aceptación: Creación dentro de Visual Studio de la lógica para acceder a los registros por medio de los archivos CSV o JSON.

Tareas:

- Crear documentos CSV o JSON.

Product Backlog Item 1.1.2: Validar y documentar la carga de datos.

Descripción: Como desarrolladores se deberá implementar la lógica para acceder a los registros desde los archivos CSV o JSON.

Criterio de aceptación: Creación dentro de Visual Studio de la lógica para el acceso a los registros desde los archivos CSV o JSON por medio del repositorio local.

Tareas:

- Desarrollar la lógica para acceder al repositorio donde se encuentran los archivos CSV o JSON.

Feature 1.5: Interfaz de Usuario

Product Backlog Item 1.1.1: Desarrollo de Formularios de WinForms

Descripción: Se requiere utilizar WinForms para que el programa tenga una interfaz gráfica para seleccionar opciones según lo que desee el Usuario.

Criterios de aceptación: El sistema abre ventanas con las diferentes opciones que tenga el programa, donde el usuario pueda matricular, reservar clases y facturas.

Tareas:

- LoginForm: Formulario de inicio de sesión.
- MainDashboard: Tablero principal para roles de Cliente y Entrenador.
- ReservaForm: Formulario para gestionar reservas de clases.
- InventarioForm: Formulario para visualizar y gestionar el inventario.

Feature 1.6: Conectar Vistas con Controladores

Product Backlog Item 1.1.1: Integrar la lógica de negocio con la interfaz.

Descripción: Se requiere integrar la lógica de negocio con la interfaz gráfica para generar un entorno amigable para Usuario.

Criterios de aceptación: El sistema debe mostrar las diferentes opciones dentro de la pantalla grafica donde el usuario pueda elegir las diferentes opciones.

Tareas:

- Desarrollar la lógica para la conexión entre las vistas y los controladores.

Product Backlog Item 1.1.2: Realizar pruebas básicas de flujo para asegurarse de que las vistas se conecten correctamente.

Descripción: Se requiere la validación y ejecución del sistema para comprobar su funcionalidad total.

Criterios de aceptación: El sistema deberá estar 100% funcional.

Tareas:

- Realizar prueba de funciones y deBugeo

7. Conclusiones

El sistema de gestión de gimnasio cumplirá con los requisitos solicitados, aplicando los principios SOLID y Clean Code en la creación de un sistema modular y escalable demostrando mayor efectividad además de una mejor estructura. Esta base ofrece una plataforma sólida para futuras expansiones y mejoras.

8. Bibliografía

1. Martin, R. C. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.
2. Freeman, E., & Robson, E. (2004). *Head First Design Patterns*. O'Reilly Media.
3. Microsoft Docs. (n.d.). *ASP.NET Core MVC Overview*. Retrieved from https://docs.microsoft.com/en-us/aspnet/core/mvc/overview