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 # 2

Sistema de Gestión de Gimnasio V2

Contenido

1. Introducción

El presente documento describe la segunda iteración del Sistema de Gestión de Gimnasio, realizada en el curso de Técnicas de Programación. En esta etapa, se incorporarán mejoras sustanciales respecto al Proyecto 1, tales como el uso de bases de datos relacionales (SQL), la migración a una interfaz web con Blazor y la implementación de nuevas funcionalidades orientadas a la gestión de usuarios y métricas.

2. Objetivo

Objetivo principal:

Diseñar e implementar una versión avanzada del Sistema de Gestión de Gimnasio, en su versión 2, que integre tecnologías modernas y funcionalidades optimizadas, con el propósito de mejorar la eficiencia, escalabilidad y experiencia del usuario, tanto en la administración interna como en el acceso de los clientes y entrenadores.

Objetivos secundarios:

1. Sustituir los archivos planos en formato CSV por una base de datos relacional que centralice y optimice el manejo de la información, garantizando la integridad y disponibilidad de los datos.
2. Migrar la interfaz gráfica desde una aplicación de escritorio a un entorno web interactivo, utilizando Blazor como marco de desarrollo, con el fin de ofrecer una experiencia de usuario más accesible, responsiva y moderna.
3. Incorporar funcionalidades avanzadas como:
    - Gestión de métricas corporales y de progreso por parte de los usuarios.
    - Generación y descarga de reportes en múltiples formatos (PDF, CSV y TXT) para facilitar el análisis de datos administrativos y operativos.
    - Configuración y administración de roles diferenciados (Administrador, Entrenador, Cliente) con permisos personalizados que aseguren el correcto acceso a las funcionalidades del sistema.
4. Implementar un conjunto de pruebas automatizadas, enfocadas en la lógica del negocio y los principales flujos operativos, asegurando una cobertura mínima del 60% y priorizando la detección temprana de errores para mejorar la calidad del producto.
5. Garantizar la escalabilidad del sistema mediante la adopción de buenas prácticas de desarrollo, como el uso de principios SOLID y Clean Code, así como una arquitectura modular y mantenible que facilite futuras expansiones.

3. Mejoras en la Versión 2

1. **Uso de Base de Datos:** Implementación de una base de datos relacional para la gestión de datos.
2. **Migración de la Capa de Vista:** Cambio de WinForms a Blazor para un entorno web.
3. **Nuevas Funcionalidades:**
    - Gestión avanzada para administradores.
    - Descarga de reportes en formatos CSV, TXT y PDF.
    - Registro y seguimiento de métricas corporales y progreso.
4. **Testing:** Asegurar cobertura de pruebas del 60% en la lógica del negocio.
5. **Resolución de Pendientes:** Finalizar al menos el 80% de los elementos no completados en la versión anterior.

4. Patrón de Diseño

Patrón de Diseño Seleccionado: Repository

El patrón Repository es ideal para manejar la capa de acceso a datos, especialmente cuando se trabaja con una base de datos relacional. Este patrón actúa como un intermediario entre la capa de lógica de negocios (Controlador) y la base de datos, lo que aporta las siguientes ventajas:

1. Separación de responsabilidades (SRP del principio SOLID):

    - La lógica de negocios no estará atada directamente a consultas SQL u operaciones específicas de la base de datos.
    - Facilita el cambio de proveedor de base de datos o estructura de datos sin modificar la lógica principal.

2. Testabilidad:

    - Puedes sustituir el repositorio real por uno simulado (mock) durante las pruebas unitarias, mejorando la cobertura del 60-80% que se solicita.

3. Reutilización y centralización:

- Todas las operaciones relacionadas con datos se concentran en un único lugar (el repositorio), lo que reduce redundancias y errores.

4. Escalabilidad:

    - Es más sencillo extender funcionalidades relacionadas con la base de datos, como agregar filtros dinámicos o manejo de transacciones.

5. Desarrollo y Cambios a Implementar

Para cumplir con los requisitos, se rediseñará el sistema integrando una base de datos relacional (SQL) y migrando la capa de vista a Blazor. Además, se añadirán funcionalidades específicas para cada rol y se optimizará la estructura del proyecto con principios SOLID, Clean Code, Herencia, Polimorfismo y Encapsulamiento.

6. Herramientas de Gestión del Proyecto

Jira: Para el seguimiento de tareas y organización del flujo de trabajo.

Git: Para control de versiones y colaboración en equipo.

7. Epic 1: Sistema de Gestión de Gimnasio v2

Feature 1.1: Autenticación y Gestión de Usuarios

Descripción: Esta feature cubre la implementación de la autenticación de usuarios, el registro de nuevos usuarios y la gestión de roles (administrador, entrenador, cliente).

Product Backlog Item 1.1.1: Generación del Formulario de Login

Descripción: Como desarrollador, necesito crear un formulario de inicio de sesión donde los usuarios puedan introducir sus credenciales para acceder al sistema.

Criterio de Aceptación: El sistema debe permitir a los usuarios introducir su correo y contraseña, validando las credenciales antes de permitir el acceso.

Tareas:

- Crear formulario de inicio de sesión.
- Implementar validación de correo y contraseña.
- Enlazar formulario de inicio de sesión con el sistema de autenticación.

Product Backlog Item 1.1.2: Registro de Nuevos Usuarios

Descripción: Como administrador, quiero permitir que nuevos usuarios se registren en el sistema, eligiendo su rol (administrador, entrenador, cliente).

Criterio de Aceptación: Los usuarios deben poder registrarse proporcionando sus datos y rol. El sistema asignará permisos de acuerdo con el rol seleccionado.

Tareas:

- Crear formulario de registro de nuevos usuarios.
- Validar la entrada de datos (nombre, correo, contraseña, rol).
- Implementar lógica para asignar roles a los usuarios durante el registro.

Feature 1.2: Gestión de Roles y Permisos

Descripción: Esta feature permite definir y gestionar los roles en el sistema, controlando el acceso a las funcionalidades según el rol del usuario.

Product Backlog Item 1.2.1: Creación y Gestión de Roles

Descripción: Como administrador, quiero poder crear roles de usuario (Administrador, Entrenador, Cliente) y asignarlos a los usuarios para controlar el acceso al sistema.

Criterio de Aceptación: Los roles deben poder ser creados y asignados a los usuarios. Cada rol tendrá permisos específicos definidos.

Tareas:

- Crear clase para la gestión de roles en la base de datos.
- Crear interfaz para asignar roles a los usuarios.
- Implementar control de acceso según el rol del usuario.

Product Backlog Item 1.2.2: Verificación de Permisos de Acceso

Descripción: Como administrador, quiero verificar que cada usuario sólo tenga acceso a las funcionalidades correspondientes a su rol.

Criterio de Aceptación: Los usuarios deben ser redirigidos a la página correspondiente según sus permisos.

Tareas:

- Implementar lógica para verificar los permisos de acceso.
- Crear redirecciones a páginas según el rol del usuario.
- Probar diferentes accesos con usuarios de distintos roles.

Feature 1.3: Gestión de Clases y Reservas

Descripción: Esta feature cubre las funcionalidades relacionadas con la gestión de clases ofrecidas por el gimnasio y la posibilidad de que los clientes realicen reservas.

Product Backlog Item 1.3.1: Crear Clases de Actividad

Descripción: Como administrador, quiero agregar nuevas clases (spinning, zumba, etc.) al sistema para que los usuarios puedan reservarlas.

Criterio de Aceptación: El sistema debe permitir agregar clases, incluyendo el nombre, horario, duración y entrenador responsable.

Tareas:

- Crear formulario para agregar nuevas clases.

- Validar la entrada de datos (nombre, horario, duración).
- Añadir clases al sistema con los detalles correspondientes.

Product Backlog Item 1.3.2: Realizar Reservas de Clases

Descripción: Como cliente, quiero poder reservar clases según mi disponibilidad y el horario de las clases.

Criterio de Aceptación: Los clientes deben poder ver las clases disponibles y reservar su espacio.

Tareas:

- Crear sistema de visualización de clases disponibles.
- Implementar funcionalidad para realizar reservas.
- Notificar a los entrenadores sobre las reservas realizadas.

Feature 1.4: Generación de Reportes

Descripción: Esta feature permite generar reportes para monitorear el rendimiento del gimnasio, incluyendo ventas, membresías y clases populares.

Product Backlog Item 1.4.1: Reporte de Ventas Diarias

Descripción: Como administrador, quiero generar un reporte de ventas diarias para evaluar el rendimiento financiero del gimnasio.

Criterio de Aceptación: El sistema debe mostrar un reporte con las ventas realizadas en un rango de fechas específico.

Tareas:

- Crear clase para cálculo de ventas diarias.
- Implementar ventana para mostrar las ventas por día.
- Agregar la opción de seleccionar el rango de fechas para el reporte.

Product Backlog Item 1.4.2: Reporte de Clases Populares

Descripción: Como administrador, quiero generar un reporte de las clases más populares para ajustar la programación de actividades.

Criterio de Aceptación: El sistema debe mostrar un listado de clases con el número de participantes en un período de tiempo seleccionado.

Tareas:

- Crear lógica para calcular las clases más reservadas.
- Desarrollar la interfaz para visualizar el reporte.
- Agregar opción para exportar el reporte a PDF.

Feature 1.5: Gestión de la Base de Datos

Descripción: Esta feature se encargará de la creación y gestión de la base de datos, incluyendo la inicialización de datos y la configuración de la base de datos relacional.

Product Backlog Item 1.5.1: Creación de Esquema de Base de Datos

Descripción: Como desarrollador, necesito definir el esquema de la base de datos para gestionar usuarios, roles, clases, reservas, facturas, inventario y métricas.

Criterio de Aceptación: La base de datos debe estar configurada correctamente con las tablas y relaciones adecuadas para almacenar la información del sistema.

Tareas:

- Crear las tablas para Usuarios, Roles, Clases, Facturas, Inventario, Métricas y Reportes.
- Definir relaciones entre las tablas.
- Establecer claves primarias y foráneas.

Product Backlog Item 1.5.2: Inicialización de Datos en la Base de Datos

Descripción: Como desarrollador, debo poblar la base de datos con datos de ejemplo como usuarios, roles, clases y otros elementos necesarios.

Criterio de Aceptación: Los usuarios y roles predeterminados deben estar creados en la base de datos.

Tareas:

- Crear datos iniciales para Usuarios y Roles.
- Crear datos iniciales para Clases e Inventario.
- Implementar el archivo SeedData.cs para insertar los datos en la base de datos.

Feature 1.6: Testing y Validaciones

Descripción: Esta feature cubre las pruebas automatizadas y de integración para asegurar que todas las funcionalidades del sistema trabajen correctamente.

Product Backlog Item 1.6.1: Pruebas Unitarias para el Sistema de Autenticación

Descripción: Como desarrollador, quiero crear pruebas unitarias para asegurar que el sistema de autenticación funcione correctamente.

Criterio de Aceptación: El sistema debe pasar las pruebas unitarias de autenticación sin errores.

Tareas:

- Crear pruebas unitarias para el formulario de inicio de sesión.
- Validar la correcta autenticación de usuarios con diferentes roles.

Product Backlog Item 1.6.2: Pruebas de Integración del Sistema de Reportes

Descripción: Como desarrollador, quiero realizar pruebas de integración para asegurar que los reportes se generen correctamente con datos en tiempo real.

Criterio de Aceptación: El sistema debe generar reportes correctos con datos de la base de datos sin errores.

Tareas:

- Implementar pruebas de integración para el reporte de ventas.
- Validar la integridad de los datos al generar reportes.

8. Bibliografía

1. anshurawate48e9c921e. (2024, May 1). Understanding the Repository Pattern. NashTech Insights. https://blog.nashtechglobal.com/understanding-the-repository-pattern/