Proyecto final "Sistema Gestión Barberia"

Daniela Fernanda Mora Ramírez

Delroy Campbell Thomas

Rolando Vinicio Madrigal Rojas.

Asesor

Prof. Ing. Andrey Mauricio Padias Calvo.

Universidad Americana.

Escuela de Ingeniería en Sistemas

Programa:

Programación III

202503_001_BIS-15_1_00

2025

Contenido

Resumen (Abstract)

El presente documento describe y se elige la opción de diseño e implementación de un **Sistema de Gestión de Barbería** desarrollado como una aplicación web robusta, segura y fácil de usar.

El sistema tiene como objetivo principal digitalizar y optimizar la administración de citas, la gestión de la cartera de clientes y el catálogo de servicios de una barbería. La aplicación se sustenta en tecnologías de Microsoft, utilizando **ASP.NET Core** (MVC o Razor Pages), el ORM **Entity Framework Core** para la persistencia de datos y una base de datos relacional. Se implementan mecanismos rigurosos de **autenticación y autorización** basados en roles (Administrador y Usuario Regular/Cliente) para asegurar el acceso controlado y la seguridad de la información. El sistema proporciona un **CRUD completo** para la gestión de sus módulos centrales (Citas, Clientes y Servicios), asegurando la integridad de los datos mediante validaciones y la aplicación de buenas prácticas de programación, como la **inyección de dependencias** y una arquitectura limpia, garantizando así un sistema escalable y de alto rendimiento.

Introducción

La gestión eficiente de los negocios de servicios es crucial para la rentabilidad y la satisfacción del cliente. Específicamente, en el sector de las barberías, la administración manual de citas, clientes y servicios consume tiempo y es propensa a errores. Para mitigar estas ineficiencias, se propone el desarrollo de un Sistema de Gestión de Barbería que automatice y centralice estas tareas. Este proyecto aborda la necesidad de una herramienta digital que permita a los clientes agendar, consultar y cancelar sus citas de manera autónoma, mientras que al personal administrativo le facilita la gestión del catálogo de servicios, la configuración de precios y el seguimiento del historial de clientes.

El sistema se desarrollará bajo los estándares de ASP.NET Core, empleando Entity Framework Core para la manipulación de la base de datos a través de migraciones, lo que asegura una capa de acceso a datos mantenible. La seguridad será un pilar fundamental, implementando un sistema de Login y Registro con autenticación para el acceso y autorización por roles para restringir las operaciones CRUD, según lo especificado en los requisitos. La interfaz de usuario, construida con Razor Pages/MVC, CSS y Bootstrap, se enfocará en la intuitividad y la experiencia de usuario, ofreciendo una solución moderna y eficiente.

Objetivos

Objetivo General

Desarrollar un sistema web seguro y eficiente para la **Gestión Integral de una Barbería**, que incluya la administración de citas, el control de clientes y la configuración de servicios, utilizando **ASP.NET Core**, Entity Framework Core y aplicando principios de arquitectura limpia.

Objetivos Específicos

1. **Implementar un sistema robusto de autenticación y autorización** por roles (Administrador y Cliente), garantizando la seguridad del acceso al sistema mediante formularios de *Login* y *Registro*, y asegurando que las operaciones CRUD críticas estén protegidas por permisos.

2. **Desarrollar la funcionalidad completa de gestión de citas**, que incluya la creación con validaciones de fecha y anticipación, la consulta del historial y la cancelación por parte de clientes y administradores, conforme a la lógica definida en el diagrama de clases.

3. **Configurar y exponer un módulo administrativo** que permita el **CRUD completo** para la gestión del catálogo de servicios (nombre, precio, tipo) y el control básico de clientes (registro, búsqueda y visualización de contactos), facilitando la actualización y mantenimiento del negocio.

Desarrollo

Arquitectura y Tecnología

El sistema se concibe bajo una arquitectura que sigue las buenas prácticas, probablemente un diseño basado en el patrón **MVC (Modelo-Vista-Controlador)** o **Razor Pages**, aprovechando la inyección de dependencias inherente a **ASP.NET Core**. La lógica de negocio y la gestión de datos se desacoplarán de la capa de presentación.

- **Tecnología Base:** ASP.NET Core (C#).
- **Persistencia:** Entity Framework Core, utilizando **Migraciones** para la creación y evolución de la base de datos. Esto mapeará las clases del diagrama (Persona, Cliente, Empleado, Cita, Servicio, Horario) a tablas relacionales.
- **Interfaz de Usuario (UI):** Se empleará Razor Pages o MVC con **CSS y Bootstrap** para garantizar una interfaz responsiva, atractiva e intuitiva, cumpliendo con los requisitos de diseño.

Módulos Clave

1. Módulo de Usuarios, Autenticación y Autorización

Este módulo es la puerta de entrada y control de acceso.

- **Login y Registro:** Se implementarán formularios de usuario para nuevos registros y un sistema de *login* que valide las credenciales.
- **Autenticación:** Solo usuarios registrados podrán acceder a las funcionalidades internas, protegiendo las páginas sensibles.

- **Autorización por Roles:**
    - **Administrador:** Acceso completo a todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en Clientes, Servicios y Citas.
    - **Usuario/Cliente:** Podrá realizar la creación, consulta y cancelación de **sus propias citas**, y la visualización del catálogo de servicios, pero sin permisos para eliminar o acceder a la información de otros clientes.

2. Gestión de Citas (CRUD)

Basado en las clases Cita, Horario, GestorCitas y GestorHorarios del diagrama.

- **Creación de Citas:** El cliente seleccionará el servicio (List<Servicio>), el empleado (Empleado) y la hora (Horario). Se aplicarán **validaciones** para asegurar que la fecha y hora estén disponibles (IsDisponible) y que la cita se agende dentro de un límite de anticipación lógico (ej., no el mismo minuto).
- **Consulta:** Clientes y empleados podrán visualizar sus citas pendientes y el historial.
- **Cancelación:** Se implementará una función que actualice el EstadoCita (Pendiente, Confirmada, Cancelada) en la base de datos, con confirmación para el usuario.

3. Control de Clientes y Empleados (CRUD)

Las clases Cliente y Empleado heredan de Persona, modelando una estructura de usuarios diferenciada.

- **Gestión de Clientes (GestorClientes):** El administrador tendrá acceso al CRUD completo para los datos de contacto (nombre, apellido, telefono, correo). La funcionalidad de **búsqueda de clientes** (BuscarCliente) será esencial, incluyendo la posibilidad de ver el historial de citas asociadas.

- **Gestión de Empleados (GestorEmpleados):** Similar a clientes, para mantener la lista de barberos disponibles, su puesto (enum PuestoEmpleado) y salario (double Salario).

4. Gestión de Servicios y Precios (CRUD)

El módulo GestorServicios administra la clase Servicio.

- **Visualización:** El catálogo de servicios (List<Servicio>) debe ser visible para todos los usuarios.
- **Configuración Administrativa:** El rol de Administrador tendrá permisos para realizar el **CRUD** sobre los servicios, permitiendo agregar nuevos, modificar el nombre, la descripción, el precio (double Precio) y el tipo de servicio, manteniendo actualizado el catálogo.

Buenas Prácticas y Documentación

- **Inyección de Dependencias:** Se utilizará para desacoplar la lógica de negocio de la infraestructura (ej., inyectando DbContext y *managers* como GestorCitas en los controladores o *Page Models*).
- **Comentarios:** Se comentará el código para explicar la lógica, especialmente en las validaciones de citas y la lógica de autenticación/autorización.
- **Documentación:** Se incluirá un archivo **README** con una guía de instalación que detalle la configuración de la base de datos y los comandos de migración de Entity Framework Core.

Conclusiones

En resumen, el sistema de gestión de barbería desarrollado busca proporcionar una solución tecnológica para las barberías y facilitar el proceso de agendamiento para los clientes. Por esto haciendo una retrospectiva en el sistema, hemos identificado el provecho que le sacaran las barberías a esta herramienta, ya que por lo implementado tendrán un orden y un mayor manejo sobre los turnos que soliciten los clientes y así evitar el incumplimiento con algunas de ellas. A su vez se detectó que el punto más débil y con futuras mejoras es la base de datos, ya que deberá ser automatizada reduciendo sus campos con información no tan redundante para así agilizar todos los procesos que vamos a llevar a cabo en nuestro sistema.

Anexos

Diagrama

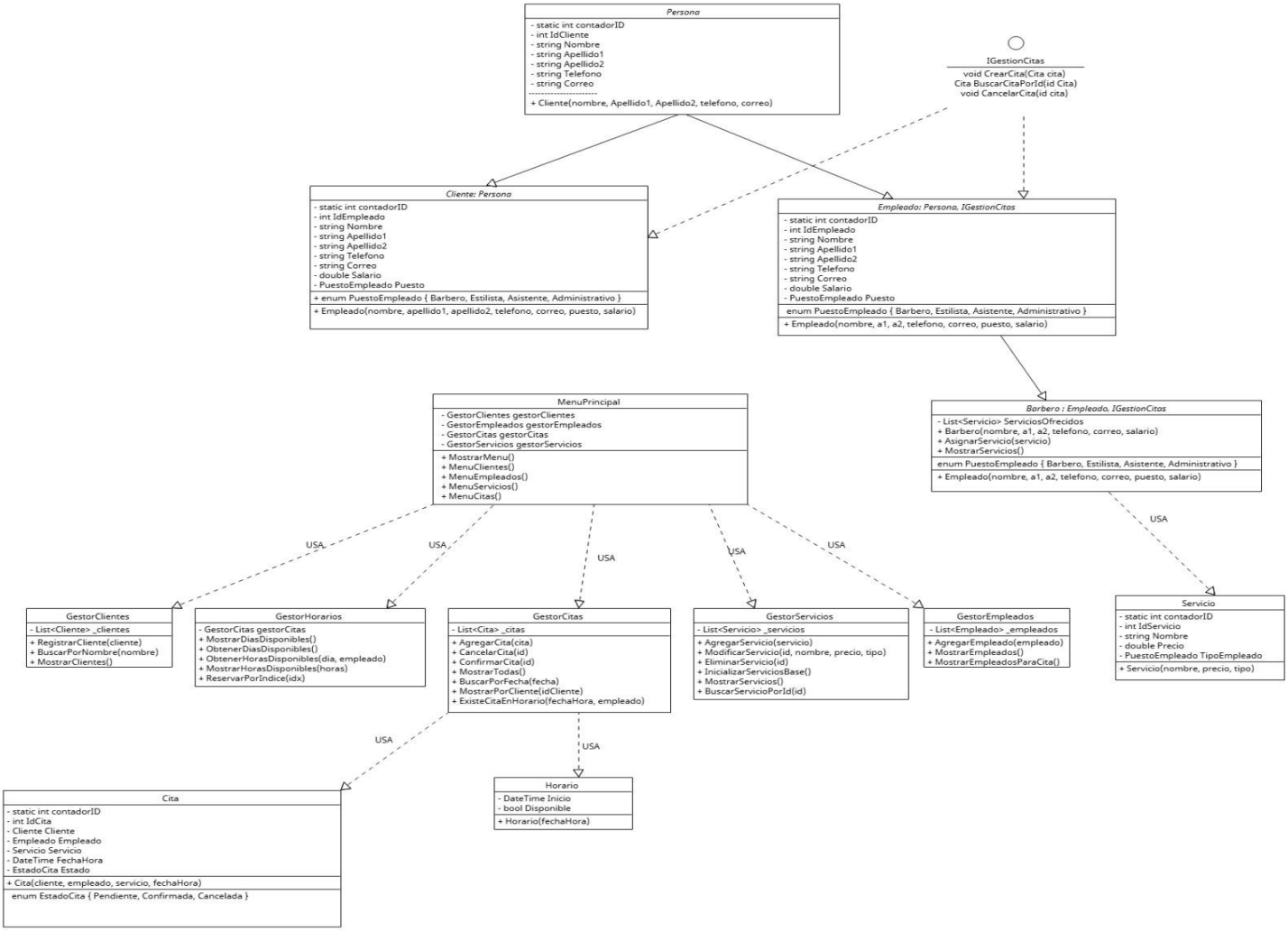
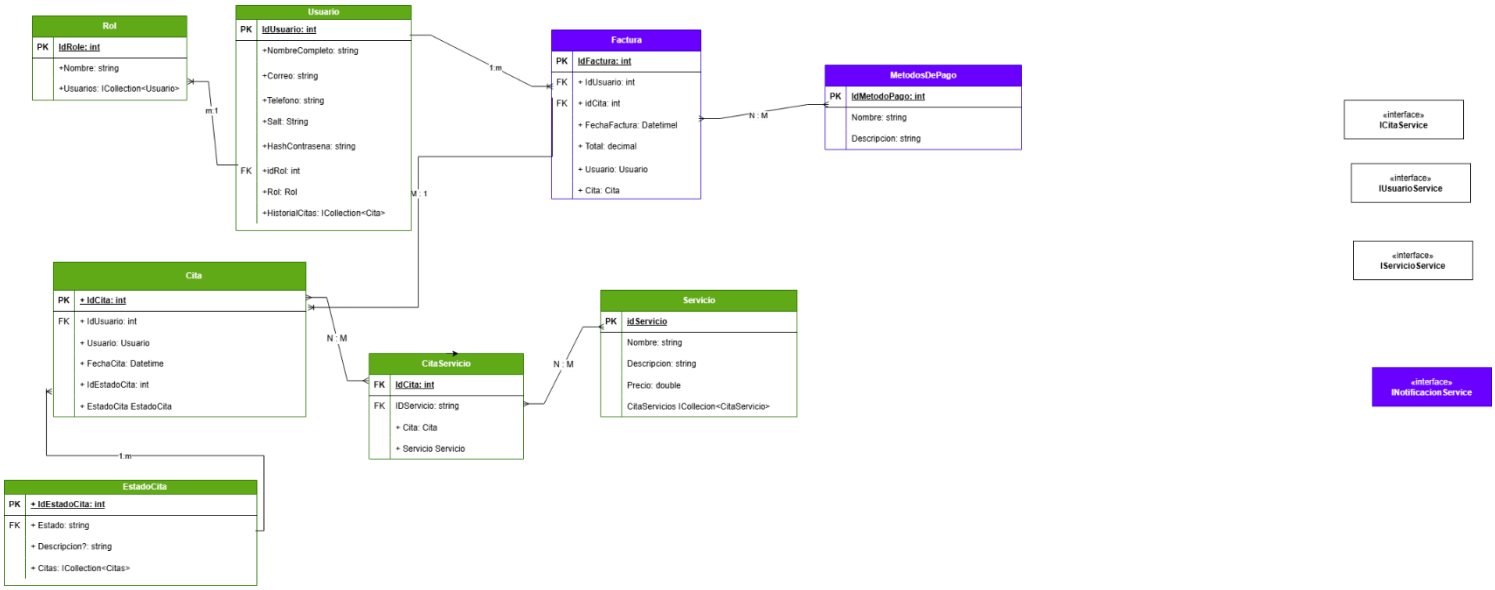


Diagrama 2


Rol
PK IdRole: int
+Nombre: string
+Usuarios: ICollection<Usuario>
Usuario
PK IdUsuario: int
+NombreCompleto: string
+Correo: string
+Telefono: string
+Salt: String
+HashContrasena: string
FK +idRol: int
+Rol: Rol
+HistorialCitas: ICollection<Cita>
Factura
PK IdFactura: int
FK + IdUsuario: int
FK + idCita: int
+ FechaFactura: Datetimel
+ Total: decimal
+ Usuario: Usuario
+ Cita: Cita
MetodosDePago
PK IdMetodoPago: int
Nombre: string
Descripcion: string
Cita
PK + IdCita: int
FK + IdUsuario: int
+ Usuario: Usuario
+ FechaCita: Datetime
+ IdEstadoCita: int
+ EstadoCita EstadoCita
CitaServicio
FK IdCita: int
FK IDServicio: string
+ Cita: Cita
+ Servicio Servicio
Servicio
PK idServicio
Nombre: string
Descripcion: string
Precio: double
CitaServicios ICollecion<CitaServicio>
EstadoCita
PK + IdEstadoCita: int
FK + Estado: string
+ Descripcion?: string
+ Citas: ICollection<Citas>
1:m
m:1
M : 1
N : M
N : M
1:m
«interface»
ICitaService
«interface»
IUsuarioService
«interface»
IServicioService
«interface»
INotificacionService

Referencias Bibliográficas

Sauron. (s. f.). ***How to Seed Users and Roles with Code First Migration using Identity ASP.NET Core*****. Stack Overflow. https://stackoverflow.com/questions/34343599/how-to-seed-users-and-roles-with-code-first-migration-using-identity-asp-net-cor**

Google Gemini. (s. f.). Gemini. https://gemini.google.com/app/cd7e93c288d1ec7e?utm_source=app_launcher&utm_medium=owned&utm_campaign=base_all

SamMonoRT. (s. f.). Migrations with Multiple Providers - EF Core. Microsoft Learn. https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/providers?tabs=vs

Wadepickett. (s. f.). Scaffold Identity in ASP.NET Core projects. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/core/security/authentication/scaffold-identity?view=aspnetcore-10.0&tabs=visual-studio

Wadepickett. (s. f.-a). Add, download, and delete user data to Identity in an ASP.NET Core project. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/core/security/authentication/add-user-data?view=aspnetcore-10.0&tabs=visual-studio

Wadepickett. (s. f.-b). Role-based authorization in ASP.NET Core. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/core/security/authorization/roles?view=aspnetcore-10.0&utm_source=chatgpt.com

Mjrousos. (s. f.-a). Custom Authorization Policy Providers in ASP.NET Core. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/core/security/authorization/iauthorizationpolicyprovider?view=aspnetcore-10.0&utm_source=chatgpt.com

Tdykstra. (s. f.-a). Filtros en ASP.NET Core. Microsoft Learn. https://learn.microsoft.com/es-es/aspnet/core/mvc/controllers/filters?view=aspnetcore-10.0&utm_source=chatgpt.com