Instituto Politécnico Nacional

Escuela Superior de Cómputo

Plan de Pruebas

Proyecto: Sistema Médico

Unidad de Aprendizaje: Ingeniería de Software

Profesor: Gabriel Hurtado

Integrantes del Equipo:

Contla Mota Luis Andrés

Escárcega Hernández Steven Arturo

García Mayorga Rodrigo

Gómez Sánchez Héctor Rodrigo

Índice

Introducción

Este documento presenta el plan de pruebas del sistema “Sistema Médico”, una solución tecnológica compuesta por una aplicación móvil multiplataforma (.NET MAUI) y un backend desarrollado en ASP.NET Core, conectados a través de servicios RESTful y respaldados por una base de datos SQL Server. El sistema está diseñado para optimizar la gestión de expedientes clínicos, consultas médicas y comunicación entre profesionales de la salud y pacientes, ofreciendo una experiencia confiable, segura y eficiente.

El propósito de este plan es garantizar la calidad del software mediante pruebas sistemáticas de funcionalidad, rendimiento, usabilidad y seguridad. Para ello, se utilizarán pruebas unitarias e integradas con herramientas como xUnit, Moq y EF Core In-Memory, cubriendo desde los controladores del backend hasta las vistas móviles. Este proceso forma parte del enfoque ágil SCRUM adoptado en el proyecto, permitiendo una validación continua y una mejora iterativa durante los sprints de desarrollo.

Objetivos

El presente Plan de Pruebas tiene como finalidad principal definir y estructurar las acciones necesarias para asegurar la calidad, funcionalidad y robustez del sistema “Sistema Médico”, compuesto por una aplicación móvil en .NET MAUI y un backend desarrollado en ASP.NET Core. Para ello, se establecen los siguientes objetivos específicos:

- **Garantizar la conformidad funcional** de cada componente del sistema respecto a los requerimientos definidos en los documentos de especificación, verificando que todas las funcionalidades previstas sean implementadas y operen correctamente.
- **Detectar errores de forma temprana** a través de pruebas unitarias automatizadas y pruebas de integración, lo cual permite reducir los costos de corrección y mejora la eficiencia del ciclo de desarrollo.
- **Verificar la robustez y estabilidad del backend** ante diferentes escenarios de uso, incluyendo entradas válidas, entradas malformadas, uso concurrente intensivo, y condiciones de borde, con el fin de evaluar su comportamiento bajo estrés.
- **Evaluar la correcta integración e interoperabilidad** entre la aplicación móvil y la API RESTful del backend, asegurando que las solicitudes,

respuestas y transformaciones de datos sean consistentes, seguras y eficientes.

- **Simular condiciones de red adversas** (como pérdida de conectividad, latencia elevada o reconexiones intermitentes) para validar la resiliencia de la aplicación móvil, comprobando que mantiene su integridad funcional y presenta mensajes adecuados al usuario en situaciones de error.
- **Validar las reglas de negocio críticas** implementadas en el backend, tales como la gestión de permisos por rol, validaciones contextuales de formularios médicos, control de agendas y lógica de aceptación o rechazo de solicitudes médicas.
- **Asegurar el funcionamiento correcto de las funcionalidades esenciales** del sistema, como la autenticación y autorización de usuarios, la creación y gestión de citas médicas, la consulta del historial clínico, y la administración de perfiles de usuario, ya que estas representan flujos fundamentales para la operación del sistema médico.

Estos objetivos servirán como eje rector para la planificación, ejecución y evaluación de las pruebas, permitiendo obtener métricas de cobertura, fiabilidad y rendimiento que respalden la calidad del producto final antes de su liberación.

Alcance

Incluye

- Pruebas unitarias de controladores, modelos y servicios (API y móvil).
- Pruebas de integración de endpoints de la API con base de datos en memoria.
- Mockeo de llamadas a servicios externos (ej. HttpClient).
- Validación de lógica de navegación entre páginas en la app móvil.
- Verificación de flujos de inicio de sesión, registro, y creación de citas.

Excluye

- Pruebas de UI en dispositivo físico/emulador (se recomienda en fases futuras).
- Pruebas de carga y estrés.
- Pruebas de seguridad (penetración, SQL injection, etc.).

Arquitectura del Sistema

La siguiente tabla describe los principales componentes tecnológicos que conforman la arquitectura del sistema “Sistema Médico”. Se detallan las tecnologías utilizadas y el propósito específico de cada componente dentro del entorno de desarrollo y pruebas, proporcionando una visión clara de la estructura técnica que respalda el funcionamiento del sistema.

Componente	Tecnología	Descripción
Backend API	ASP.NET Core 9.0	Provee endpoints RESTful para usuarios, citas, autenticación
Base de Datos	Entity Framework Core	Persistencia de entidades (Usuario, Cita, Perfil)
Aplicación Móvil	.NET MAUI	Interfaz de usuario y consumo de la API
Comunicación	HTTP/HTTPS	Intercambio de datos JSON vía API REST
Test Framework	xUnit	Ejecución y organización de pruebas
Mocking	Moq	Simulación de servicios y dependencias
Pruebas EF	EF Core InMemory	Base de datos simulada para pruebas aisladas

Estrategia de Pruebas

Enfoque

Las pruebas del sistema “Sistema Médico” se desarrollarán siguiendo un enfoque modular y progresivo, lo que permite identificar defectos de forma temprana, reducir riesgos técnicos y validar cada componente de manera precisa. Inicialmente, se ejecutarán **pruebas unitarias** sobre funciones aisladas, como controladores, modelos y servicios, sin involucrar dependencias externas, para asegurar que cada unidad de código cumpla con su propósito específico.

Posteriormente, se llevarán a cabo **pruebas de integración**, centradas en verificar la correcta interacción entre los distintos módulos del backend, incluyendo la conexión con la base de datos, flujos de datos entre capas y cumplimiento de reglas de negocio. Finalmente, se evaluará la lógica **transversal entre la API y la aplicación móvil**, mediante pruebas funcionales y simulaciones de consumo de servicios HTTP desde MAUI. Este enfoque garantiza que tanto las funcionalidades internas como la experiencia de usuario final se encuentren alineadas con los requisitos del sistema y con estándares de calidad del sector salud.

Tipos de pruebas a aplicar

Tipo	Descripción
Pruebas Unitarias	Validación de funciones individuales sin dependencias externas
Pruebas de Integración	Validación del flujo de datos entre controladores y base de datos
Pruebas de Servicios HTTP	Simulación de respuestas y validación de consumo desde MAUI
Pruebas de Navegación	Verificación de la lógica de cambio entre pantallas en la app móvil
Pruebas de Validación	Evaluación de restricciones de entrada, campos requeridos, etc.

Configuración de Entorno de Pruebas

Creación de proyectos de prueba

```
# Para backend

dotnet new xunit -n ProyectoISNuevo.Tests

dotnet add ProyectoISNuevo.Tests/ProyectoISNuevo.Tests.csproj reference ProyectoISNuevo/

# Para app móvil

dotnet new xunit -n SistemaMedicoApp.Tests

dotnet add SistemaMedicoApp.Tests/SistemaMedicoApp.Tests.csproj reference SistemaMedicoApp/
```

Paquetes NuGet requeridos

Ambos proyectos de pruebas:

- xunit
- xunit.runner.visualstudio
- Microsoft.NET.Test.Sdk
- Moq

Solo para el backend:

- Microsoft.EntityFrameworkCore.InMemory

Componentes a Probar

Backend

Controladores

Ejemplo: AuthApiController

- Verificar métodos Login, Register, Logout
- Validar respuestas ante credenciales válidas e inválidas
- Confirmar códigos HTTP correctos (200, 401, 400)

Modelos y DbContext

Ejemplo: Usuario, Cita, ApplicationDbContext

- Validar atributos requeridos, formatos de campos
- Probar relaciones uno-a-muchos
- Verificar inserciones, actualizaciones y eliminaciones en la BD simulada

Aplicación Móvil

ViewModels y Páginas

Ejemplo: LoginPageViewModel, CitasPage

- Simular entrada de usuario y verificar navegación
- Evaluar control de errores ante respuestas negativas
- Comprobar lógica de visualización y carga de datos

Servicios HTTP

Ejemplo: ApiService.cs

- Mockear respuestas de API (éxito, error, red caída)
- Confirmar que el ViewModel maneja correctamente cada escenario

Diseño de los Casos de Prueba

Backend: AuthApiControllerTests.cs

La siguiente tabla muestra los casos de prueba del controlador de autenticación del backend. Se validan escenarios de login y registro, contemplando respuestas esperadas ante entradas válidas, errores de autenticación y usuarios duplicados.

ID	Descripción	Entrada	Resultado esperado
B01	Login válido	user: "admin", pwd: "123"	HTTP 200 + JWT
B02	Login inválido	user: "admin", pwd: "wrong"	HTTP 401
B03	Registro duplicado	user ya registrado	HTTP 400
B04	Registro válido	datos completos	HTTP 201 y creación en BD

App Móvil: LoginPageTests.cs

La siguiente tabla muestra los casos de prueba del controlador de autenticación del backend. Se validan escenarios de login y registro, contemplando respuestas esperadas ante entradas válidas, errores de autenticación y usuarios duplicados.

ID	Descripción	Entrada	Resultado esperado
M01	Ingreso vacío	campos vacíos	Alerta de validación
M02	Login exitoso	credenciales válidas	Navega a pantalla principal
M03	Error de API	Mock de 500	Alerta de “Servidor no disponible”

Estructura de Carpetas

```
ProyectoISNuevo.Tests/
├── Controllers/
│   └── AuthApiControllerTests.cs
├── Models/
│   ├── UsuarioTests.cs
│   └── CitaTests.cs
└── Integration/
    └── ApiIntegrationTests.cs

SistemaMedicoApp.Tests/

├── Pages/
│   ├── LoginPageTests.cs
│   └── CitasPageTests.cs
└── Services/
    └── ApiServiceTests.cs
```

Cronograma Estimado

La siguiente tabla presenta el cronograma estimado para la ejecución del plan de pruebas, dividido por semanas y asignado según las responsabilidades de los equipos de desarrollo. Cada etapa contempla actividades específicas que abarcan desde la configuración inicial hasta la documentación final de resultados.

Semana	Actividad	Responsable
Semana 1	Configuración de proyectos y entorno	Equipo Backend
Semana 2	Pruebas de controladores y modelos	Equipo Backend
Semana 3	Pruebas de integración y base InMemory	Equipo Backend
Semana 4	Pruebas de ViewModels y navegación	Equipo Móvil
Semana 5	Mocking de HttpClient y manejo de errores	Equipo Móvil
Semana 6	Documentación de resultados y cobertura	Todo el equipo

Criterios de Aceptación

- Cobertura de código mínima del 80% en pruebas unitarias.
- Todos los casos de prueba críticos deben pasar sin errores.
- No deben existir excepciones no controladas en producción.
- Validación correcta de flujos de autenticación y citas.

Riesgos y Mitigaciones

A continuación, se identifican los principales riesgos que podrían afectar la efectividad del plan de pruebas, junto con su probabilidad de ocurrencia y las estrategias propuestas para mitigarlos. Estas acciones buscan reducir el impacto negativo en la calidad y en los tiempos del proyecto.

Riesgo	Probabilidad	Mitigación
Baja cobertura de pruebas	Media	Establecer métricas de cobertura mínima
Errores de integración entre app y backend	Alta	Probar con mocks y pruebas de extremo a extremo
Cambios en requisitos	Media	Modularidad y pruebas aisladas ayudan a adaptarse

Resultados Esperados

- Validación completa del backend con EF InMemory y simulación de flujos reales.
- Verificación de lógica en app móvil sin necesidad de red.
- Confianza en despliegue a producción gracias a pruebas automatizadas.
- Base para incluir pruebas de interfaz gráfica (MAUI) en el futuro.

Conclusión

El presente plan de pruebas establece una base sólida y estructurada para garantizar la calidad del sistema "Sistema Médico", abordando de forma integral las pruebas unitarias, de integración y funcionales en ambos componentes principales: el backend en ASP.NET Core y la aplicación móvil en .NET MAUI. Gracias a la adopción de herramientas modernas como xUnit, Moq y EF Core In-Memory, se logra simular escenarios de uso real, validar reglas de negocio críticas y anticipar posibles errores antes de su liberación a producción.

Además, la integración de este plan dentro del marco ágil SCRUM permite una validación continua durante cada sprint, fomentando la mejora iterativa del sistema. La identificación de riesgos y la planificación del cronograma aseguran que el equipo mantenga una visión clara y preventiva, contribuyendo así al desarrollo de una solución confiable, segura y alineada con los estándares de calidad del sector salud.