



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Aplicación de un marco ágil Scrum y diseño arquitectónico para el desarrollo de VisitaConnect: Plataforma de gestión de visitas en un hospital de alta complejidad

Nombre del candidato(a): Nicolas Andree Buguño Lanas

Carrera / Grado: Ingeniería en Informática

Campus: Sede Viña del Mar Departamento: Departamento de Electrotecnia e informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Lais San Martín Navarro, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 08-03-2026

Firma:

Estudiante o Candidato(a):

Fecha: 06-03-2026

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Aplicación de un marco ágil Scrum y diseño arquitectónico para el desarrollo de VisitaConnect: Plataforma de gestión de visitas en un hospital de alta complejidad

Nicolas Andree Bugueño Lanas

nicolas.bugueno@usm.cl

Lais San Martín Navarro
Profesora Guía

Francis Fuentes Chacana
Profesora Correferente



Resumen:

El sistema público de salud en Chile enfrenta desafíos en la gestión de visitas hospitalarias, especialmente por el uso de procesos manuales que provocan esperas, confusión y sobrecarga en el personal administrativo. En respuesta a esta situación, se desarrolló VisitaConnect, una plataforma digital diseñada para optimizar la reserva y control de visitas en el Hospital Dr. Gustavo Fricke de Viña del Mar, entregando mayor autonomía a los visitantes y eficiencia en la gestión interna del hospital.

El proyecto se ejecutó mediante un trabajo colaborativo bajo la metodología ágil Scrum, lo que permitió al equipo de desarrollo organizar sus esfuerzos en Sprints y validar los avances de forma iterativa. Un aspecto central de este trabajo conjunto fue el análisis y diseño de la arquitectura del software, donde se definió una estructura robusta basada en el patrón Modelo-Vista-Controlador (MVC). Sobre estos cimientos de diseño, se implementó la solución utilizando tecnologías como ASP.NET MVC, SQL Server y MailKit para la gestión de notificaciones automáticas.

El resultado fue una plataforma moderna, escalable y segura, que mejora la experiencia tanto de los usuarios como del personal hospitalario. Se espera que su implementación aporte a la modernización de los procesos internos, reduzca los tiempos de espera y contribuya a una atención más humana y eficiente en los hospitales públicos de la región.

Palabras Clave:

- Plataforma digital VisitaConnect, Gestión de visitas hospitalarias, Metodología ágil, Marco de trabajo Scrum, ASP.NET MVC.

1 Introducción

En Chile, el sistema público de salud atiende diariamente a miles de personas, siendo los hospitales los principales centros de referencia para la población. No obstante, la alta afluencia de pacientes y visitantes ha evidenciado diversas dificultades en la gestión interna de estos recintos, particularmente en lo que respecta al acceso y registro de visitas. Si bien en muchos centros de salud se han implementado iniciativas como el programa Hospital Digital [5], orientadas a modernizar la atención, persisten procesos que continúan realizándose de forma manual, lo que provoca extensas esperas, falta de claridad respecto a horarios y normativas, y confusión entre quienes desean acompañar a sus familiares.

Estas ineficiencias no solo afectan a los visitantes, sino que también generan una carga adicional para el personal administrativo, que debe atender consultas repetitivas y realizar gestiones de manera poco eficiente para un proceso anticuado. Las cifras reflejan esta situación: en 2023, el Ministerio de Salud informó que más de 2,5 millones de personas se encontraban en listas de espera, reflejo de la presión estructural que enfrenta el sistema público de salud. Aunque gran parte de los esfuerzos de digitalización se ha enfocado en reducir los tiempos de atención clínica, áreas complementarias como la gestión de visitas familiares se continúan realizando mediante procesos manuales y carentes de modernización y trazabilidad de la información, lo que afecta negativamente en la percepción de calidad del servicio y evidencia la necesidad de avanzar hacia soluciones tecnológicas centradas en el usuario.

Implementar un sistema digital eficaz para la reserva y control de visitas hospitalarias permitiría reducir los tiempos de espera, entregar información clara y accesible, y mejorar significativamente la experiencia tanto de los usuarios como del personal. De este modo, se contribuiría a una atención más moderna, eficiente y centrada en las personas.

1.1 Contexto Y Antecedentes

Actualmente, el Hospital Dr. Gustavo Fricke de Viña del Mar maneja el registro de visitas de forma manual. El proceso es el siguiente: los visitantes deben acudir al mesón de informaciones, donde el personal verifica manualmente en un listado impreso si el paciente está hospitalizado y si tiene visitas permitidas.

En caso de que el paciente tenga las visitas permitidas, el visitante recibe un sticker para ingresar a la sección del hospital donde están las salas. En este procedimiento, el guardia de seguridad cumple un rol clave, ya que se encarga de guiar y validar que la persona llegue correctamente a la sala de su familiar.

Este proceso manual provoca la existencia de largas esperas y presenta varios vacíos de información que generan gran confusión para los visitantes:

- **Horarios y Cupos:** Los horarios de visita dependen del piso del hospital. Para el área médica de adultos (Medicina), las horas de visitas son desde las 11:00 AM a 13:00 PM, pero no hay un control centralizado de cuántas visitas ha recibido el paciente. Se permite el ingreso de 1 familiar por bloque de visita, lo que provoca que familiares viajen al hospital solo para descubrir que el cupo de visita ya fue utilizado.
- **Restricciones de Paciente:** Las reglas para unidades críticas como UCI o UTI son distintas (ej. UCI tiene un horario de 15:00 PM a 17:00 PM), permitiendo solo familiares directos y prohibiendo el ingreso a niños. Esta información no siempre es comunicada de forma clara al visitante en la entrada, generando frustración.
- **Trazabilidad y Carga Administrativa:** El registro se basa en un cuaderno físico, lo que dificulta la trazabilidad y consume tiempo valioso del personal administrativo, que debe atender consultas repetitivas.

El resultado de esta problemática es que la experiencia de los visitantes termina siendo poco agradable, reduciendo la sensación de calidad del servicio y reforzando la sensación de desorden y falta de modernización dentro del hospital.

1.2 Definición del problema

Actualmente, el Hospital Dr. Gustavo Fricke todavía maneja el registro de visitas de forma manual. Esto provoca la existencia de largas esperas, falta de claridad en los horarios y desorientación dentro de los sectores del hospital. El resultado de esta problemática es que la experiencia de los visitantes termina siendo poco agradable, reduciendo la sensación de calidad del servicio, mientras que el personal administrativo se ve saturado por tener que responder las mismas dudas o ayudar a los visitantes que se encuentran perdidos dentro del recinto. Todo esto retrasa la atención y evita que el personal pueda concentrarse en funciones más relevantes de su trabajo.

El acompañamiento hospitalario resulta ser una parte fundamental del bienestar emocional de los pacientes, por lo que se vuelve una función clave el tener un sistema que permita organizar estas visitas de manera ágil y clara. Sin una herramienta digital que facilite el proceso, lo único que se logra es reforzar la sensación de desorden y falta de modernización dentro del hospital.

1.3 Breve descripción de la propuesta de solución

VisitaConnect se plantea como una respuesta a este problema, diseñada para un hospital de alta complejidad en la V Región. La idea es ofrecer una plataforma simple y moderna que permita a las personas reservar su hora de visita sin tener que hacer filas o esperar demasiado tiempo para ello. Además, busca aliviar la carga del personal administrativo, automatizando parte de los procesos, otorgando autonomía al visitante y entregando herramientas para controlar y organizar las visitas de manera eficaz.

El proyecto se llevó a cabo bajo la metodología ágil Scrum, lo que permite trabajar por etapas y adaptarse a los cambios que surgen durante el desarrollo de manera ágil. La plataforma web estará disponible tanto en computadores como en celulares, otorgando comodidad para que los visitantes puedan acceder fácilmente desde cualquier lugar.

Entre sus principales funciones se encuentran:

- **Envío de notificaciones automáticas por correo electrónico** para confirmar y recordar las visitas.
- **Validación mediante código QR**, que facilita el ingreso y mejora la seguridad.
- **Administración de reglas de visita**, Permite configurar los permisos de acceso asociados a cada paciente (ej. visitas bloqueadas, restringidas a grupo familiar o habilitadas), verificar su ubicación en salas y controlar el ingreso mediante validación de RUT.

Es importante destacar que VisitaConnect no busca eliminar por completo la atención presencial, sino complementarla. Para aquellos visitantes con escaso o nulo manejo informático (como adultos mayores), el proceso manual seguirá existiendo. El visitante podrá acudir al mesón de informaciones y será el (o la) recepcionista quien utilizará la plataforma VisitaConnect para gestionar la reserva en el momento. De esta forma, todas las visitas (tanto digitales como presenciales) quedan registradas en un único sistema, manteniendo la trazabilidad y el control de cupos, pero asegurando la inclusión de todos los usuarios.

Con esto, el sistema busca mejorar la organización y seguridad en el proceso de visitas que actualmente es manual, además de ofrecer una experiencia más fluida para las personas que asisten al hospital.

1.4 Objetivos general y Objetivos específicos

Objetivo general: Diseñar e implementar una plataforma digital de gestión de visitas hospitalarias para el Hospital Dr. Gustavo Frické de Viña del Mar, utilizando la metodología ágil y el marco de trabajo Scrum para desarrollar un sistema seguro, escalable y centrado en la experiencia del usuario, que optimice los procesos de registro, control y acceso.

Objetivos específicos:

- Analizar la situación actual de la gestión de visitas en hospitales públicos.
- Levantar requerimientos funcionales del sistema.
- Investigar tecnologías adecuadas para el desarrollo del sistema.
- Diseñar la arquitectura del sistema utilizando el patrón MVC.
- Implementar el módulo de reservas en línea como núcleo central de la plataforma.
- Desarrollar el sistema de administración y control de acceso.

1.5 Justificación del proyecto

El desarrollo de VisitaConnect se justifica por la necesidad de modernizar el proceso de visitas internas en los hospitales públicos y ofrecer una experiencia más eficiente a los usuarios. Hoy en día, la gestión manual de visitas genera demoras, confusiones y sobrecarga de trabajo para los funcionarios. Automatizar este proceso no solo mejora la organización, sino que también ayuda a reducir errores y tiempos de espera, aumentando la sensación de calidad dentro del recinto.

Además, al incorporar funciones como la validación por QR y las notificaciones automáticas, se fortalece la seguridad y la comunicación con los visitantes. En conjunto, esta propuesta busca avanzar hacia un sistema hospitalario más moderno, con procesos digitales que beneficien tanto a los pacientes como al personal, promoviendo una atención autónoma y ordenada.

2 Marco conceptual / Estado del arte

2.1 Metodologías ágiles en el desarrollo de software

Las metodologías ágiles nacen como una respuesta a las limitaciones claras del enfoque tradicional para el desarrollo de proyectos, como el modelo en cascada [3][4], que se caracteriza por su rigidez y baja adaptabilidad a los cambios, lo cual lo hace útil para proyectos predecibles con requisitos estables. En contraparte, el Manifiesto Ágil (2001) dispone de valores y principios orientados a la flexibilidad [1][2], la comunicación y la entrega continua de valor al cliente.

Entre las metodologías más utilizadas se encuentran Extreme Programming (XP), Kanban y Scrum [1][2], siendo esta última la de mayor uso en proyectos modernos de software por su estructura clara y cómodamente adaptable.

Scrum se basa en ciclos cortos llamados Sprints, que tienen un rango estimado de 1 a 4 semanas, los cuales permiten entregar versiones funcionales del producto en períodos definidos, entregando valor y constante retroalimentación a los desarrolladores. Cada Sprint tiene metas claras y priorizadas, lo que facilita la planificación, la colaboración y la mejora continua.

Dentro del marco de trabajo se definen tres roles principales:

- **Product Owner:** responsable de representar al cliente y priorizar los requerimientos del producto.
- **Scrum Master:** encargado de guiar al equipo, eliminar impedimentos y asegurar el cumplimiento de la metodología.
- **Equipo de Desarrollo:** conformado por los integrantes que diseñan, construyen y prueban el sistema.

Además, Scrum define eventos importantes como la planificación del Sprint (Sprint Planning), las reuniones diarias (**Daily Scrum**), la revisión del Sprint (**Sprint Review**) y la retrospectiva (**Sprint Retrospective**), que permiten mantener un ritmo de trabajo ordenado y transparente.

El uso de una metodología ágil, especialmente Scrum, aporta grandes beneficios y mejoras a los equipos de desarrollo, ya que fomenta la comunicación continua, mejora la capacidad de adaptación ante los cambios y reduce el riesgo de errores acumulados. En el contexto de proyectos académicos o institucionales como VisitaConnect, este enfoque resulta ideal porque permite iterar sobre los avances, validar los resultados con el equipo y ajustar el producto a las necesidades reales del usuario final.

2.2 Arquitectura de software

El objetivo principal de la arquitectura de software es la implementación de la estructura general que define cómo se organizan los componentes de un sistema y cómo se comunican entre sí [3][4]. Su principal objetivo es garantizar que el sistema a desarrollar sea escalable, mantenible y eficiente, permitiendo que las distintas partes trabajen de manera estable y sin errores a lo largo del tiempo.

Una **definición** menos técnica para la función de arquitectura es la implementación del "esqueleto" donde se construirán los cimientos de una plataforma o aplicación, seleccionando qué tecnologías se utilizan, cómo se separan las responsabilidades y de qué forma se conectan las diversas capas del sistema. En el desarrollo de VisitaConnect, una buena arquitectura es clave para mantener la claridad del código, facilitar futuras mejoras y mejorar la escalabilidad de la plataforma.

Dentro de las arquitecturas mayormente utilizadas se encuentra el patrón Modelo–Vista–Controlador (MVC) [4][6], el cual divide la lógica de negocio, la interfaz de usuario y el control de las interacciones en tres componentes distintos:

- **Modelo:** es el encargado de manejar los datos, las reglas de negocio y la conexión con la base de datos.
- **Vista:** responsable de mostrar la información al usuario de forma clara e interactiva.
- **Controlador:** actúa como intermediario entre el modelo y la vista, gestionando las solicitudes del usuario y actualizando los datos o la interfaz según corresponda.

La implementación del patrón MVC otorga una ventaja sumamente relevante para el desarrollo de una plataforma web, como una mayor modularidad (ya que cada parte de los componentes puede modificarse sin afectar directamente a los demás), mejor mantenibilidad durante el tiempo al facilitar la organización y lectura de código, y finalmente agregar una mayor escalabilidad, debido a que permite agregar nuevas funciones sin modificar la estructura base del proyecto.

Para el caso específico de VisitaConnect, se optó por implementar el patrón MVC utilizando el framework ASP.NET MVC [6], ya que este otorga un robusto entorno, seguro y largamente documentado. Esta elección fue definida debido a que permite una integración simple entre capas de presentación, la lógica de negocios y el manejo de datos, logrando un gran equilibrio entre el rendimiento y la organización para un proyecto.

De manera complementaria al patrón MVC, se estableció una arquitectura por capas como pilar fundamental del diseño. Esta decisión se tomó para garantizar una separación estricta de responsabilidades, evitando que la lógica de negocio se mezclara con los controladores y facilitando el mantenimiento futuro del código.

- **Capa de presentación:** interfaz visual para el usuario, desarrollada con Razor Pages, HTML y CSS personalizados.
- **Capa de lógica de negocio:** encargada de procesar las solicitudes, aplicar reglas y validar los datos antes de enviarlos o recibirlos desde la base.
- **Capa de datos:** gestiona la comunicación con el sistema de almacenamiento, en este caso SQL Server, y mantiene la integridad de la información.

Esta estructura favorece la independencia entre capas, logrando una gran reducción de errores y haciendo mucho más simple la localización de problemas. Finalmente, facilita la implementación de pruebas y futuras actualizaciones sin alterar el sistema completo.

La combinación del patrón MVC con la arquitectura por capas otorga a VisitaConnect no solo una base sólida, sino una estructura preparada para crecer en el tiempo, asegurando un desarrollo enfocado en un rendimiento estable y un mantenimiento ágil por parte del equipo de desarrollo.

2.3 Principios de Diseño de Software

Para el desarrollo de los sistemas modernos, los valores del diseño de software son claves para lograr aplicaciones y plataformas amplias, mantenibles y escalables en el tiempo. Estos principios tienen el rol de guiar y orientar a los desarrolladores en la toma de decisiones técnicas, lo cual permite crear soluciones eficientes, robustas y fáciles de mejorar durante todo el ciclo de desarrollo.

Su buena implementación no influye solo en la calidad con la cual se programa el código, sino también en cómo se organiza el equipo y en la facilidad con la que se pueden agregar nuevas funcionalidades. En VisitaConnect, estos principios fueron aplicados desde primera instancia para garantizar una estructura clara y ordenada del sistema.

Entre los principios mayormente conocidos están SOLID, KISS y YAGNI, los cuales han sido adoptados ampliamente en el ámbito del desarrollo de software profesional.

Principios SOLID

Los principios SOLID, propuestos inicialmente por Robert C. Martin [2], tienen como propósito facilitar la creación de sistemas de software flexibles y de bajo acoplamiento. Dado que VisitaConnect fue desarrollado bajo el paradigma de Orientación a Objetos utilizando el lenguaje C#, la aplicación de estos principios resultó fundamental para garantizar la calidad del diseño arquitectónico.

Cada una de sus letras representa un fundamento aplicado en el proyecto:

- **S - Principio de Responsabilidad Única:** Cada clase tiene una única razón para cambiar. En el caso de VisitaConnect, esto se realizó mediante una clara separación entre controladores, modelos y servicios.
- **O - Principio Abierto/Cerrado:** El sistema debe permitir extender funcionalidades sin modificar código existente. Se aplicó en servicios como validación de RUT y envío de correos, que pueden ampliarse sin afectar el núcleo.
- **L - Principio de Sustitución de Liskov:** Las clases derivadas pueden reemplazar a las bases sin alterar el comportamiento. Esto garantizó coherencia en el uso de herencia y polimorfismo.
- **I - Principio de Segregación de Interfaces:** Se privilegiaron interfaces específicas sobre generalistas. Se implementó mediante servicios especializados para correo, autenticación y gestión de usuarios.
- **D - Principio de Inversión de Dependencias:** Los módulos de alto nivel dependen de abstracciones, no de implementaciones concretas. Esto facilitó el reemplazo de componentes como repositorios y servicios externos.

Principio KISS

El principio KISS (Keep It Simple, Stupid) promulga la simplicidad para el desarrollo, evitando por completo la complejidad innecesaria en el código o la estructura de este. La idea principal es que las soluciones más simples suelen ser también las soluciones más indicadas y efectivas, especialmente cuando hablamos del trabajo en equipo. En el proyecto VisitaConnect, se tradujo en aplicar una estructura simple para cualquier desarrollador, priorizando la funcionalidad por sobre la complejidad. Con este enfoque se permitió un desarrollo más rápido para los cortos tiempos de entrega establecidos y una integración fluida y simple entre los módulos más complejos del sistema.

Principio YAGNI

El principio YAGNI (You Aren't Gonna Need It) contempla como pilar fundamental el que no se deben implementar características que aún no se necesitan. En muchas oportunidades, al anticipar funciones futuras o no previstas, se agregan elementos que provocan complicaciones en el sistema sin aportar un valor inmediato al proyecto. Para el desarrollo de VisitaConnect, este principio ayudó a enfocar todos los esfuerzos en el Producto Mínimo Viable (MVP), evitando módulos o implementaciones innecesarias para llevar el desarrollo por buen camino, evitando distracciones y priorizando funcionalidades claves, como el módulo de reserva y administración [1][2].

2.3.1. Síntesis de la Aplicación de Principios

En conclusión, la aplicación conjunta de **SOLID, KISS y YAGNI** permitió construir una plataforma sólida, ordenada y adaptable, donde cada componente cumple su función específica y puede mejorarse o reemplazarse sin afectar al resto del sistema.

2.4 Tecnologías Clave Utilizadas

Durante el desarrollo de VisitaConnect se utilizaron diversas tecnologías que permitieron construir un sistema funcional, escalable y alineado con las necesidades reales de un entorno hospitalario. La elección de cada herramienta se basó en su estabilidad, documentación disponible y facilidad de integración con el patrón de arquitectura MVC.

ASP.NET MVC

Este sistema fue desarrollado utilizando el framework ASP.NET MVC [6]. Esta es una tecnología creada por Microsoft, la cual combina el modelo de desarrollo orientado a objetos con la estructura Modelo-Vista-Controlador. ASP.NET facilita esta separación de responsabilidad, lo cual permite que la capa de presentación, lógica y de manejo de datos trabajen de forma independiente.

Cabe agregar que su integración con C# y .NET Core ofrece un entorno robusto, seguro y escalable para el desarrollo de una plataforma web de visitas hospitalarias. Para VisitaConnect, ASP.NET MVC fue la base para la construcción del desarrollo de las vistas dinámicas, controladores y las interacciones con la base de datos, asegurando con su estructura un flujo claro y ordenado de la información.

SQL Server

Para la gestión de los datos, se utilizó la plataforma llamada Microsoft SQL Server, la cual es un motor de base de datos relacional ampliamente usado en entornos empresariales. Su capacidad de manejo ante grandes cantidades de información y su alta compatibilidad con .NET la hace una plataforma clave para cualquier proyecto que requiera de seguridad y consistencia en los datos.

Para el contexto del desarrollo de VisitaConnect, SQL Server es la clave para almacenar los datos de pacientes, usuarios visitantes, horarios de reserva y registro de las visitas, asegurando la integridad de los datos y permitiendo consultas de alta velocidad para los módulos del sistema [3][4].

Integración con Sistema de Información Hospitalario (HIS)

Un pilar fundamental de VisitaConnect es la obtención de los datos de los pacientes. El sistema no funcionará como una base de datos aislada, sino que debe integrarse mediante una API al Sistema de Información Hospitalario (HIS) existente en el Hospital Dr. Gustavo Fricke. Esta integración es crítica para consultar en tiempo real

- La lista de pacientes admitidos.
- La ubicación exacta del paciente (piso, sala y cama).

Esto asegura que VisitaConnect opere con información actualizada, sin duplicar datos. Toda la comunicación con el HIS deberá realizarse bajo los estrictos protocolos de seguridad y protección de datos del hospital, garantizando la confidencialidad de la ficha clínica.

MailKit y Notificaciones Automáticas

En el desarrollo del módulo de notificaciones, se utilizó la librería MailKit, la cual es una herramienta de código abierto que permite enviar correos electrónicos desde una aplicación .NET. Esta tecnología fue sumamente clave para el correcto desarrollo del módulo de notificaciones automáticas. El uso de esta herramienta permitió automatizar un proceso que era de suma importancia para el desarrollo del producto MVP.

Validación de RUT mediante API Externa

Otra funcionalidad clave fue la integración de una API externa para la consulta de datos mediante el RUT chileno. A diferencia de una validación algorítmica local (que solo verifica si el dígito verificador es correcto), esta API permite corroborar en tiempo real la existencia del usuario y obtener su nombre completo desde fuentes públicas. Esta integración aportó un nivel superior de seguridad y confiabilidad, ya que previene el ingreso de datos falsos o el uso de RUTs válidos pertenecientes a terceros, asegurando que cada reserva esté correctamente asociada a la identidad real del visitante.

Generación y Lectura de Códigos QR

El sistema también incorporó la generación de códigos QR, una tecnología esencial para controlar el acceso de los visitantes al recinto. Al confirmar una reserva, el sistema genera un código único que puede ser leído por el personal de seguridad al ingreso del hospital, validando la información en segundos. Esto no solo agiliza el flujo de entrada, sino que también mejora la trazabilidad y seguridad del proceso.

Diseño de Interfaz y Estilos

Para la parte visual, el equipo de diseño y front-end utilizó HTML5, CSS3 y Bootstrap, combinando con estilos personalizados para mantener una apariencia limpia, moderna y que se adapta correctamente a las necesidades del usuario. El diseño se enfocó en la simplicidad y accesibilidad, asegurando que tanto usuarios externos como personal hospitalario pudieran utilizar la plataforma sin dificultad. Además, se optimizó la compatibilidad con dispositivos móviles, permitiendo el acceso desde distintos tipos de pantallas.

2.4.1. Síntesis de la Infraestructura Tecnológica

En síntesis, la integración de estas herramientas tecnológicas permitió construir una plataforma robusta, moderna y confiable. Cada componente cumple un rol específico y crítico dentro de la arquitectura del sistema, aportando a la eficiencia operativa, la seguridad de los datos y la escalabilidad futura de VisitaConnect.

2.5 Documentación y Comunicación del Equipo de Desarrollo

Durante el completo desarrollo de VisitaConnect, la comunicación y la documentación fueron aspectos realmente fundamentales para mantener la coordinación correcta del equipo. A pesar de la existencia de herramientas especializadas para control de versiones, como GitHub, en este proyecto se optó por un enfoque más manual y accesible para todos los integrantes, utilizando Google Drive como repositorio principal de archivos y versiones del sistema.

Esta herramienta permitió almacenar, compartir y actualizar documentos, manteniendo un historial claro de los avances en cada etapa del proyecto, otorgando al equipo comodidad y adaptabilidad en el control de versiones.

Para la gestión de tareas y priorización de objetivos, se utilizó Notion, donde se creó un tablero con las funcionalidades claves del producto y se clasificaron según su prioridad en el backlog del proyecto. Para cada fase, se priorizaron distintos tipos de desarrollos y se enfocó en base al módulo clave que se debía desarrollar para la entrega del primer Sprint. Este entorno cumplió una función clave para el correcto desarrollo de los Sprints, optimizando los registros y los avances, lo que permitió mantener la transparencia en el progreso general.

Las reuniones de coordinación y revisión se realizaron a través de Discord, utilizando sus características de canales de voz para realizar reuniones ágiles tipo Daily Scrum y mantener una comunicación fluida entre los miembros del equipo. Este método de trabajo resultó eficiente, ya que permitió resolver dudas en tiempo real y mantener un ambiente colaborativo sin depender de la presencialidad, otorgando flexibilidad al equipo de trabajo.

En conjunto, el uso de estas herramientas permitió mantener un flujo de trabajo organizado, una comunicación constante y una trazabilidad de decisiones acorde con la metodología Scrum, adaptada a las condiciones reales del equipo de desarrollo.

2.6 Evaluación Continua y Mejora del Sistema

Durante el desarrollo de VisitaConnect, la mejora constante fue una parte importante del proceso. Gracias a la metodología Scrum (cuya implementación práctica y planificación se detallan en profundidad en el Capítulo 3), el sistema se fue construyendo y evaluando de forma progresiva, permitiendo revisar los avances y aplicar correcciones a tiempo. Cada Sprint finalizaba con una revisión donde el equipo analizaba los resultados, discutía los problemas encontrados y definía los ajustes necesarios para el siguiente ciclo.

A lo largo del proyecto se realizaron distintos tipos de **pruebas y validaciones** que ayudaron a comprobar el funcionamiento correcto del sistema:

- **Pruebas unitarias:** Se usaron para comprobar que cada módulo cumpliera correctamente su función.
- **Pruebas de integración:** Verificaron que los distintos componentes del sistema (como el módulo de reservas, la base de datos y el envío de correos) trabajaran juntos sin errores.
- **Pruebas de usuario:** Se realizaron para observar la experiencia de uso y detectar posibles mejoras en la interfaz o en el flujo de las reservas.

Estas pruebas se repitieron en cada iteración, aplicando mejoras según los resultados y la retroalimentación del equipo y del patrocinador. El enfoque iterativo permitió reducir errores y optimizar el rendimiento antes de llegar a la versión final del sistema.

Además, el equipo mantuvo una retroalimentación constante mediante reuniones breves, donde se discutían las tareas diarias y se resolvían posibles bloqueos. Esta dinámica favoreció la comunicación, la organización del trabajo y el cumplimiento de los objetivos dentro de los plazos establecidos.

La evaluación continua y las mejoras aplicadas durante todo el desarrollo ayudaron a que VisitaConnect se convirtiera en una plataforma más estable, segura y fácil de mantener, cumpliendo con los requisitos iniciales y abriendo la posibilidad de seguir evolucionando en el futuro.

3 Metodología y Plan de Trabajo

La metodología utilizada para el desarrollo del Sistema de Visitas Hospitalarias VisitaConnect fue Scrum. Este marco de trabajo ágil promueve la colaboración, la mejora continua del desarrollo y la gestión eficiente de la organización.

En este proyecto se definieron los roles esenciales del marco: Product Owner, Scrum Master y Equipo de Desarrollo. El rol asumido fue el de Scrum Master, encargado de facilitar el proceso, gestionar el cumplimiento de la metodología y eliminar impedimentos que dificultaran el avance del equipo.

Para cumplir el rol asumido y abordar un sistema de la magnitud de un gestor de visitas hospitalarias en un hospital de alta complejidad de la V Región, se diseñó un plan estructurado en fases, manteniendo los pilares de Scrum: transparencia, inspección y adaptación.

3.1 Inicio y levantamiento de requerimientos

El objetivo principal de esta fase fue comprender el problema y definir una solución viable. Para ello, en reuniones junto al patrocinador, se elaboró un Backlog inicial con las funcionalidades del Producto Mínimo Viable (MVP). Los temas con mayor relevancia fueron: gestión de pacientes, reserva de visitas online y validación mediante código QR.

Se realizaron reuniones quincenales, en las cuales se fueron refinando y priorizando los avances necesarios para llevar el proyecto adelante. Desde el rol de Scrum Master, se facilitó la comunicación entre el equipo y el patrocinador, poniendo énfasis en las solicitudes del equipo y las limitaciones existentes en base al conocimiento técnico del área.

Teniendo en cuenta tanto las limitaciones como las habilidades disponibles, se llegó a la conclusión de que los tres pilares fundamentales para el MVP eran los ya mencionados. Esta definición clara permitió al equipo comenzar el desarrollo iterativo con objetivos correctamente delimitados.

3.2 Desarrollo iterativo

Se definió separar en 3 sprints el proyecto

- Fase 1: sistema de reservas,
- Fase 2: Sistema de administración,
- Fase 3: implementación de QR y refinamiento de detalles

Para la primera etapa, se planificaron Sprints de cuatro semanas enfocados en construir el núcleo del sistema de reservas. Para comenzar con el desarrollo, se llevó a cabo una reunión previa donde se definieron metas claras, entre ellas: diseñar la arquitectura del sistema, implementar la base de datos en SQL Server y establecer el patrón de desarrollo MVC.

Durante esta primera fase, el equipo de desarrollo propuso trabajar en ASP.NET MVC para el backend, decisión que fue tomada en conjunto. Desde el rol de Scrum Master, se facilitó el entendimiento dentro del equipo para asegurar la toma de la mejor decisión posible, garantizando que todas las alternativas técnicas fueran evaluadas antes de priorizar la opción más adecuada para el futuro del proyecto.

En la segunda instancia, se realizaron Sprints de tres semanas. En un inicio se priorizaron las tareas fundamentales para el correcto avance del proyecto, enfocadas en el sistema administrativo de la plataforma. Esto incluyó la creación de un sistema de autenticación (login) tanto para el personal de seguridad como hospitalario, un módulo de "Grupo Familiar" y "Bloqueo de RUTs", y una función para configurar los permisos de acceso del paciente (sin visita, grupo familiar, habilitado). Para finalizar, se comenzó con la investigación de la tecnología necesaria para el lector QR.

Como último ciclo, se llevó a cabo un Sprint de dos semanas donde la prioridad fue el refinamiento de las interfaces de usuario y la implementación de reglas de negocio, tales como: la validación de RUTs mediante API, el envío de códigos QR vía correo electrónico y el correcto escaneo de estos en la vista de guardia.

Durante los Sprints, con los criterios y herramientas ya definidas, se realizaron reuniones diarias (Daily Scrum) para revisar avances, identificar complejidades en el desarrollo y priorizar tareas. Entre los hitos realizados destacaron: la correcta implementación de la base de datos, la generación de un diseño web con criterios de usabilidad, la construcción del módulo de reservas y la implementación de las funcionalidades de administrador.

Entre las iteraciones diarias se efectuaron reuniones con el patrocinador del proyecto, con quien se discutieron cambios y aprobaciones de implementaciones. El Scrum Master asumió la responsabilidad de la comunicación con el patrocinador, transmitiendo sus requerimientos al equipo de trabajo. Con esta metodología se logró mantener un orden y un progreso adecuado en el desarrollo del sistema en todos sus ciclos.

3.3 Pruebas y validación

A medida que el desarrollo de las funcionalidades principales del proyecto fue completado, se aplicaron pruebas en tres niveles en todas las fases del proyecto:

- **Pruebas unitarias:** para validar la lógica de cada módulo.
- **Pruebas de integración:** para verificar la comunicación entre componentes.
- **Pruebas de usuario:** para evaluar la usabilidad y obtener retroalimentación.

Desde el rol de Scrum Master, se supervisó que estas validaciones se realizaran dentro de cada Sprint, gestionando las correcciones necesarias para el correcto funcionamiento.

3.4 Cierre y entrega

Finalmente, se llevaron a cabo correcciones basadas en las pruebas y el feedback entregado por el patrocinador. En esta etapa se elaboró la documentación técnica, preparando la entrega final del proyecto.

3.5 Plan de trabajo

El desarrollo de VisitaConnect se llevó a cabo aplicando la metodología ágil Scrum, como fue mencionado anteriormente, dividiendo la implementación de módulos específicos en 3 fases principales y una etapa final de cierre. Cada fase se organizó en Sprints semanales, con reuniones y revisiones para la mejora continua del producto. El siguiente cronograma se extendió entre agosto y noviembre de 2025, siguiendo una planificación

progresiva que permitió entregar versiones funcionales a medida que se iban realizando las entregas requeridas.

3.6 Tabla de Plan de Trabajo

Fase / Sprint	Duración	Periodo estimado año 2025	Actividades principales	Entregables
Fase 1: Investigación y desarrollo del módulo de reservas	4 semanas	2 al 29 de agosto	Definición de requerimientos funcionales, diseño de base de datos, prototipos iniciales, desarrollo del módulo de reservas y correo automático.	Módulo funcional de reservas con sistema de notificaciones.
Fase 2: Desarrollo del sistema de administración y guardia	3 semanas	8 al 26 de septiembre	Implementación del módulo administrativo y panel de guardia, mejoras de interfaz, conexión con base de datos y ajustes visuales.	Módulo administrativo operativo con gestión de accesos y roles.
Fase 3: Refinamiento e implementación de mejoras (API y QR)	2 semanas	1 al 17 de octubre	Integración del sistema de validación por QR, pruebas funcionales y refinamiento de requerimientos principales.	Sistema integrado con lectura de QR y validaciones mejoradas.
Pruebas finales y documentación	3 semanas	18 de octubre al 10 de noviembre	Pruebas de usabilidad, validación general del sistema, corrección de errores y documentación final para el evento "Feria de Software".	Versión final del sistema y documentación técnica.

Tabla 3.1 Fuente: Elaboración propia

CARTA GANTT				
ACTIVIDADES	SEMANA 1	SEMANA 2	SEMANA 3	SEMANA 4
Definición de requerimientos funcionales	02-08-2025			
Diseño de base de datos	04-08-2025			
Prototipos iniciales	06-08-2025			
Desarrollo módulo de reservas	09-08-2025			
Sistema de correo automático	29-08-2025			
Implementación módulo administrativo		08-09-2025		
Desarrollo panel de guardia		08-09-2025		
Mejoras de interfaz		15-09-2025		
Conexión con base de datos		15-09-2025		
Ajustes visuales		22-09-2025		
Integración sistema de validación QR			01-10-2025	
Pruebas funcionales del sistema			08-10-2025	
Refinamiento de requerimientos			17-10-2025	
Pruebas de usabilidad				18-10-2025
Validación general del sistema				25-10-2025
Corrección de errores				03-11-2025
Documentación final y preparación Feria de Software				10-11-2025

Tabla 3.2 Fuente: Elaboración propia

3.7 Síntesis general

Con el plan de trabajo visto en la figura 3.1 y una constante preocupación por la entrega continua de valor, se permitió mantener un flujo constante de desarrollo y revisión. Cada Sprint cerró con una entrega funcional y una reunión de retroalimentación donde se analizaron los avances, los errores detectados y las posibles mejoras para la plataforma.

Gracias a esta organización, VisitaConnect evolucionó progresivamente hasta lograr una versión final estable y lista para su presentación en la Feria de Software 2025.

4 Análisis de Requerimientos

En este capítulo se detallan los requerimientos funcionales y no funcionales del sistema VisitaConnect. Estos requerimientos fueron levantados a partir de la definición del problema y las necesidades detectadas en el Hospital Dr. Gustavo Fricke de Viña del Mar.

4.1 Requerimientos Funcionales (RF)

Los requerimientos funcionales definen los comportamientos, funciones y servicios específicos que el sistema debe proveer, abarcando tanto las interacciones de los usuarios como los procesos internos automáticos necesarios para la operación de VisitaConnect.

- **RF-01 Gestión de Pacientes (Mantenedor):** El sistema debe estar preparado para procesar la información de los pacientes (RUT, ubicación, estado) proveniente de fuentes externas. *Nota: Para efectos de este proyecto y debido a restricciones de seguridad sobre datos reales, esta funcionalidad se implementa mediante una carga de datos simulada en la base de datos local.*
- **RF-02 Configuración de Permisos de Visita:** El sistema debe permitir configurar las reglas de acceso asociadas a cada paciente (ej. "Visitas Habilitadas", "Restringido a Grupo Familiar" o "Visitas Bloqueadas"), determinando quién está autorizado para realizar una reserva.
- **RF-03 Gestión de Reglas y Restricciones (Clínico):** El sistema debe permitir al personal autorizado configurar reglas de acceso a dos niveles:
 1. **Nivel Global:** Basado en el estado del paciente (ej. "Con Visitas", "Aislamiento" o "grupo familiar").
 2. **Nivel Específico:** Permitiendo el bloqueo o autorización de un RUT específico (ej. prohibir el acceso a un visitante particular por orden judicial o administrativa), registrando el motivo de dicha restricción.
- **RF-04 Reserva de Visitas Web:** El sistema debe disponer de un portal público donde los visitantes puedan agendar un bloque horario. Para ello, deben ingresar el RUT del paciente y el RUT del visitante. El sistema validará automáticamente:
 1. La existencia del paciente.
 2. Que el paciente tenga permisos de visita habilitados.
 3. La disponibilidad de cupos (aforo) para el bloque seleccionado.

- **RF-05 Validación de Visitantes:** El sistema debe aplicar restricciones lógicas durante la reserva, impidiendo el agendamiento si el visitante tiene restricciones previas o si intenta reservar más de una visita simultánea para el mismo paciente en el mismo día.
- **RF-06 Generación y Envío de Comprobante:** Tras finalizar una reserva exitosa, el sistema debe generar automáticamente un código QR único y enviarlo al correo electrónico del visitante, sirviendo como llave de acceso validada.
- **RF-07 Control de Acceso (Vista Guardia):** El sistema debe proveer una interfaz de escaneo para el personal de seguridad, que permita leer el código QR, verificar su vigencia en tiempo real y registrar la hora de entrada y salida de la visita (Check-in / Check-out).
- **RF-08 Gestión de Usuarios del Sistema:** El sistema debe permitir a un usuario administrador ("Super Admin") crear y gestionar las cuentas de acceso para el personal de guardia y administrativo, asignando los roles y credenciales correspondientes.
- **RF-09 Gestión de Salas y Aforos:** El sistema debe permitir la configuración de los recintos hospitalarios (Salas), definiendo parámetros críticos como el nombre, ubicación (piso) y, fundamentalmente, la **Capacidad Máxima** (aforo). Este parámetro será utilizado por el algoritmo de reserva para calcular la disponibilidad de cupos en tiempo real.
- **RF-010 Configuración de Horarios y Turnos:** El sistema debe permitir definir ventanas de tiempo específicas (Turnos) asociadas a cada Sala (ej. Bloques AM/PM o régimen 24x7). Esto permite que el sistema habilite o deshabilite dinámicamente las opciones de reserva según la normativa horaria de cada sector del hospital.

4.1.1 Requerimientos No Funcionales (RNF)

Los requerimientos no funcionales definen los atributos de calidad y restricciones técnicas del sistema:

- **RNF-01 Accesibilidad Web (Responsive):** La plataforma debe ser una aplicación web adaptable (*responsive*), que se enfoque en funcionar correctamente tanto en computadores de escritorio (para administrativos) como en dispositivos móviles (para visitantes y guardias).
- **RNF-02 Usabilidad:** La interfaz de reserva para los visitantes debe ser intuitiva, y accesible requiriendo como máximo 4 pasos para completar una solicitud.

- **RNF-03 Rendimiento de Validación:** El proceso de escaneo y validación del código QR por el lado del guardia no debe tardar más de 10 segundos para evitar filas en el acceso.
- **RNF-04 Acceso al sistema de administración:** Tanto los guardias como personal clínico deben contar con una cuenta de acceso para el sistema que corresponda para cada área

5 Diseño de la Arquitectura

En el presente capítulo se detallan los componentes estructurales y el funcionamiento técnico de VisitaConnect, utilizando para ello diagramas y tablas explicativas que facilitan la comprensión de la solución propuesta.

5.1 Diagrama de Arquitectura de Alto Nivel (HLD)

En la **Figura 5-1** se presenta una visión general de la estructura del sistema.

El diseño se fundamenta en un **patrón de arquitectura por capas**, decisión estratégica que, como se mencionó anteriormente, permite organizar la lógica del sistema de manera eficiente. El propósito de este diagrama es ilustrar las interacciones entre los distintos componentes para dar cumplimiento a los requerimientos funcionales definidos previamente.

Como se aprecia en la ilustración, la estructura se organiza en tres capas principales:

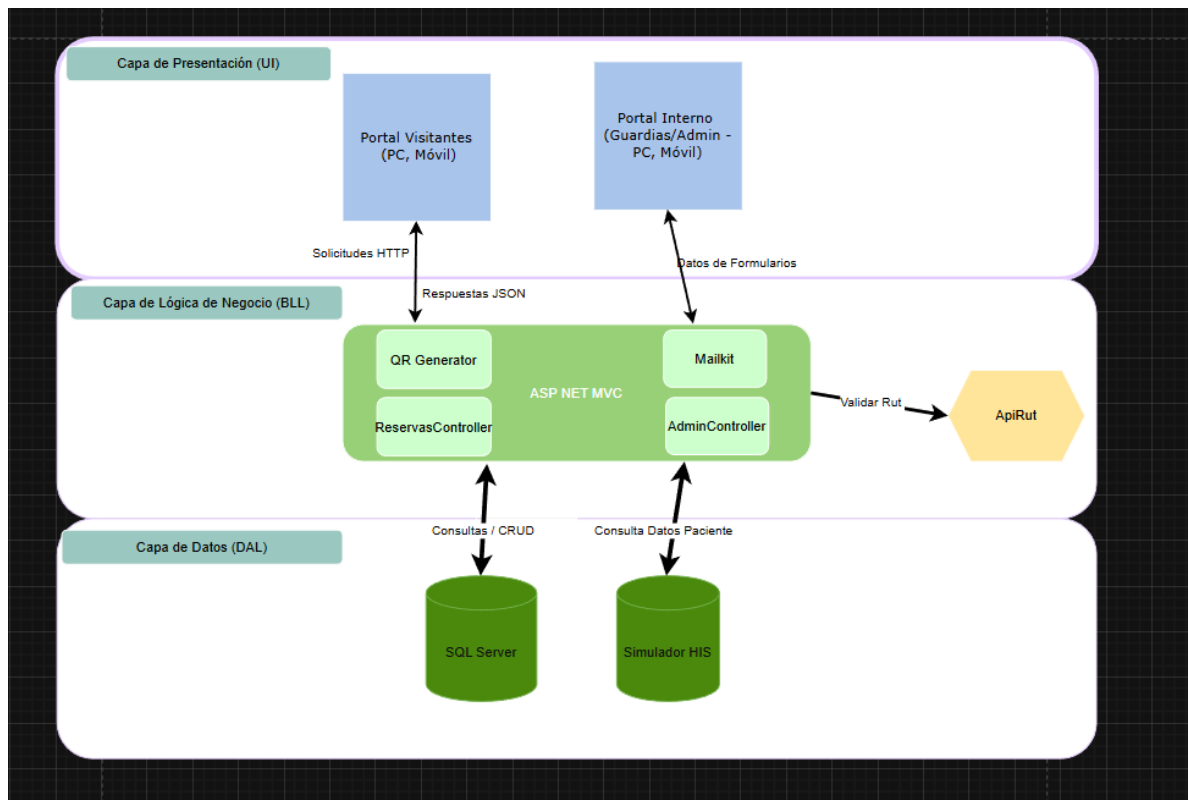


Figura 5-1. Diagrama de Arquitectura de Alto Nivel (HLD) de VisitaConnect.
Fuente: Elaboración propia.

1. **Capa de presentación (UI):** Conformada por las plataformas web accesibles desde computador de escritorio, notebook y dispositivos móviles. Se distinguen dos portales: el portal público, orientado al visitante, y el portal administrativo, destinado a guardias y personal clínico.
2. **Capa de Lógica de Negocio (BLL):** Constituye la base del sistema y fue desarrollada en **ASP.NET MVC**. En esta capa se encuentran los controladores principales *ReservasController* y *AdminController*, que procesan las solicitudes HTTP, aplican las reglas de negocio e interactúan con servicios externos, garantizando un flujo de datos correcto.
3. **Capa de Datos (DAL):** responsable del almacenamiento y la recuperación de la información. Mantiene la comunicación directa con la base de datos en **SQL Server** y gestiona la persistencia de las entidades, incluyendo la carga de la nómina de pacientes (la cual, como se definió en los requerimientos, se simula localmente para efectos del prototipo).

Adicionalmente, la arquitectura integra servicios externos esenciales para la operación: **MailKit** para el envío de notificaciones, un **Generador de QR** para los accesos y la **API RUT** para la validación de identidad.

5.2 Diccionario de Datos

En esta sección se describe la estructura de la base de datos utilizada por el sistema, detallando las entidades principales, sus atributos y tipos de datos.

Tabla: Pacientes

Almacena la información de los pacientes hospitalizados. En el contexto del MVP, esta tabla se utiliza para simular la información que provendría del HIS.

Campo	Tipo de Dato	Descripción
IdPaciente	Integer (PK)	Identificador único del paciente en el sistema local.
Rut	Varchar(11)	RUT del paciente (identificador civil).
Nombres	Nvarchar(100)	Nombres del paciente.
Apellidos	Nvarchar(100)	Apellido paterno y materno.
FechaDeNacimiento	Date	Fecha de nacimiento del paciente.
IdEstado	TinyInt (FK)	Código numérico que representa el estado (1: Habilitado, 2: Restringido, 3: Bloqueado).
IdSala	Int (FK)	Referencia numérica a la tabla Salas.

Tabla: Salas

Define las ubicaciones físicas dentro del hospital para permitir la trazabilidad de las visitas.

Campo	Tipo de Dato	Descripción
IdSala	Int(PK, Identity)	Identificador único autogenerado.
Piso	TinyInt	Número de piso (0-255). Se usa TinyInt por eficiencia.
CodigoSala	Varchar(10)	Código visual de la sala (ej. "A-104").
NombreSala	Nvarchar(50)	Nombre descriptivo de la sala (ej. "Medicina Hombres").
Capacidad	TinyInt	Cantidad máxima de camas (0-255).

Tabla: HorarioSala

Configura los bloques de disponibilidad y turnos específicos para cada sala, permitiendo establecer regímenes de visita diferenciados.

Campo	Tipo de Dato	Descripción
IdHorario	Int (PK, Identity)	Identificador único de Horario.
IdSala	Int (FK)	Referencia a la sala
Turno	Varchar(2)	Nombre del turno (ej. "AM", "PM").

HoraInicio	Time	Hora de apertura de visitas.
HoraFin	Time	Hora de cierre de visitas.
Es24x7	Bit	Indica si el horario es continuado (1) o restringido (0)
Observacion	Nvarchar(50)	Notas adicionales sobre el turno

Tabla: Usuario

Registra la información de las personas externas (familiares o visitas) que interactúan con la plataforma.

Campo	Tipo de Dato	Descripción
IdUsuario	Int (PK, Identity)	Identificador único del visitante.
Rut	Varchar(12)	RUT del visitante.
Nombres	Nvarchar(100)	Nombre del visitante.
Apellidos	Nvarchar(100)	Apellido paterno y materno.
NumeroDocumento	Nvarchar(100)	Número de serie del carnet (para validación extra).
teléfono	Varchar(15)	Número de contacto para notificaciones SMS (futuro).
Correo	Nvarchar(150)	Email para el envío de comprobantes y códigos QR.

Tabla: Reserva

Entidad transaccional central que vincula a un visitante con un paciente en un bloque horario y fecha específicos.

Campo	Tipo de Dato	Descripción
IdReserva	BigInt (PK, Identity)	Identificador único de la reserva.
IdUsuario	Int (FK)	Referencia al visitante que realizó la reserva.
IdPaciente	Int (FK)	Referencia al paciente que será visitado.
IdEstadoReserva	TinyInt	Estado actual de la reserva (ej. Activa, Completada, Cancelada).
FechaReserva	Date	Fecha para la cual se agendó la visita.
HoraInicio	Time	Hora de inicio del bloque de visita.
HoraFinal	Time	Hora de término del bloque de visita.

Tabla: PacienteVisitanteRegla

Permite la gestión de reglas de negocio específicas, como restricciones legales o permisos especiales de acceso por paciente.

Campo	Tipo de Dato	Descripción
IdPermisoRut	Int(PK, Identity)	Identificador de la regla.
IdPaciente	Int (FK)	Paciente al que se le aplica la regla.
RutCanonical	Varchar(12)	RUT específico autorizado (si aplica, ej. solo un familiar).
TipoRegla	TinyInt	Tipo de restricción (ej. "Bloqueo total", "Solo RUT específico").
Motivo	Nvarchar(255)	Justificación administrativa de la regla.
CreadoPor	Varchar(50)	Usuario administrativo que creó la regla.
CreadoEn	DateTime	Fecha y hora de creación de la regla (auditoría).
Activo	Bin	Otorga trazabilidad al visitante registrado

Tabla: EstadoReserva

Catálogo que normaliza los posibles estados del ciclo de vida de una reserva.

Campo	Tipo de Dato	Descripción
IdEstadoReserva	TinyInt(PK, Identity)	Identificador del estado de la reserva
NombreEstado	Varchar(50)	Nombre del estado en el que se encuentra la reserva (Activa, Cancelada, Completada)

Tabla: EstadoPaciente

Tabla que se encarga determinar si una reserva se encuentra habilitada para ingreso

Campo	Tipo de Dato	Descripción
IdEstado	TinyInt(PK, Identity)	Identificador del estado del paciente
NombreEstado	Varchar(50)	Descripción (ej. Con Visitas, Aislamiento)

Tabla: AspNetUsers (Identity)

Tabla estándar generadas por el framework ASP.NET Core Identity para el manejo de autenticación y autorización del personal (Guardias, Administrativos).

Campo	Descripción
Id	Identificador único del usuario del sistema (GUID).
UserName	Nombre de usuario para inicio de sesión (RUT o correo interno).
PasswordHash	Contraseña almacenada de forma segura (hasheada).
Email	Correo electrónico institucional del funcionario.

5.3 Diagramas de Casos de Uso

El diagrama de casos de uso presentado en la **Figura 5-2** modela la funcionalidad del sistema desde la perspectiva de los usuarios. Su objetivo es ilustrar cómo interactúan los distintos actores con VisitaConnect para alcanzar objetivos de negocio específicos. Se identifican cuatro actores principales para el sistema:

1. **Visitante:** Representa al usuario externo (familiar o cercano del paciente) que requiere agendar una visita. Es quien inicia el flujo principal de reserva a través del portal público.
2. **Recepción:** Personal encargado de la atención al público. Su rol es asistir presencialmente a aquellos visitantes que presenten dificultades tecnológicas (brecha digital), realizando el proceso de reserva en su nombre
3. **Personal Clínico:** Profesionales de la salud o administrativos de piso encargados de gestionar la disponibilidad de los pacientes. Son responsables de configurar las reglas de bloqueo o restricción de visitas según el estado de salud.
4. **Guardia:** Personal de seguridad encargado de controlar el acceso físico al recinto. Su interacción con el sistema se limita a la validación de los códigos QR presentados por los visitantes.

Nota sobre la cobertura de Requerimientos: Cabe destacar que los Requerimientos Funcionales **RF-09 (Gestión de Salas)** y **RF-10 (Configuración de Horarios)** no se representan como Casos de Uso independientes en este diagrama. Esto se debe a que, en la versión actual (MVP), estos parámetros se consideran **configuraciones estructurales del sistema** gestionadas a nivel de Base de Datos (Seed Data) y no mediante interacciones directas de los actores en la interfaz web. Sin embargo, estos requerimientos son consumidos y validados internamente durante la ejecución del caso de uso **(CU-01) Realizar Reserva Web**.

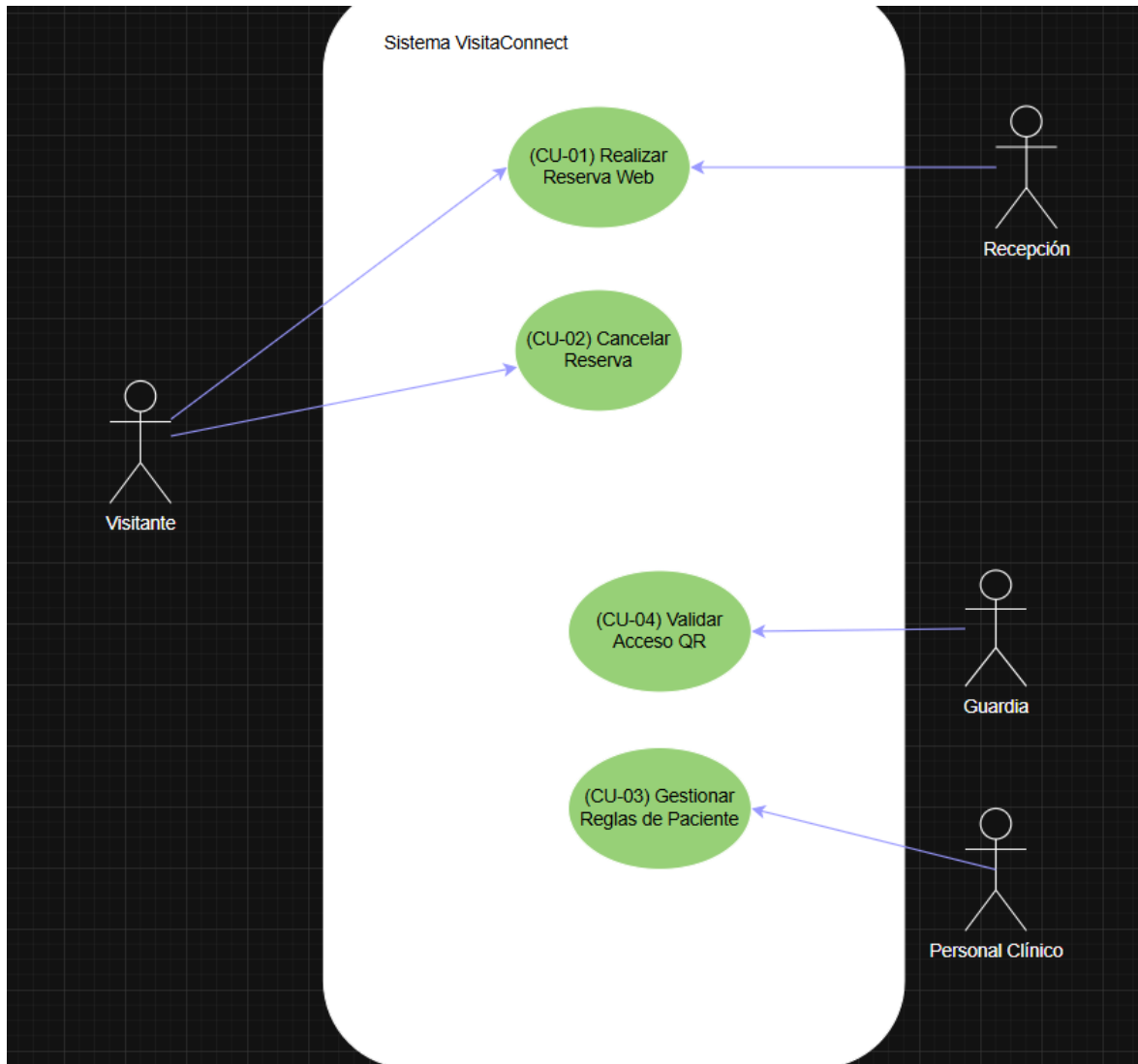


Figura 5-2. Diagrama de Casos de Uso del sistema. Fuente: Elaboración propia.

5.4 Diagrama de Secuencia

A continuación, se presentan los diagramas de secuencia que detallan las interacciones entre los objetos del sistema. En esta sección se describen los flujos principales de la plataforma VisitaConnect, ilustrando cómo los actores interactúan con la interfaz, los controladores y la capa de datos para ejecutar las funciones críticas del sistema.

5.4.1. Verificación de Paciente

En la **Figura 5-3** se describe el inicio del proceso de reserva. El visitante ingresa los datos requeridos y el sistema, a través de sus controladores, valida la existencia del paciente interactuando con el repositorio de datos y verifica si existen restricciones vigentes mediante las reglas de negocio definidas.

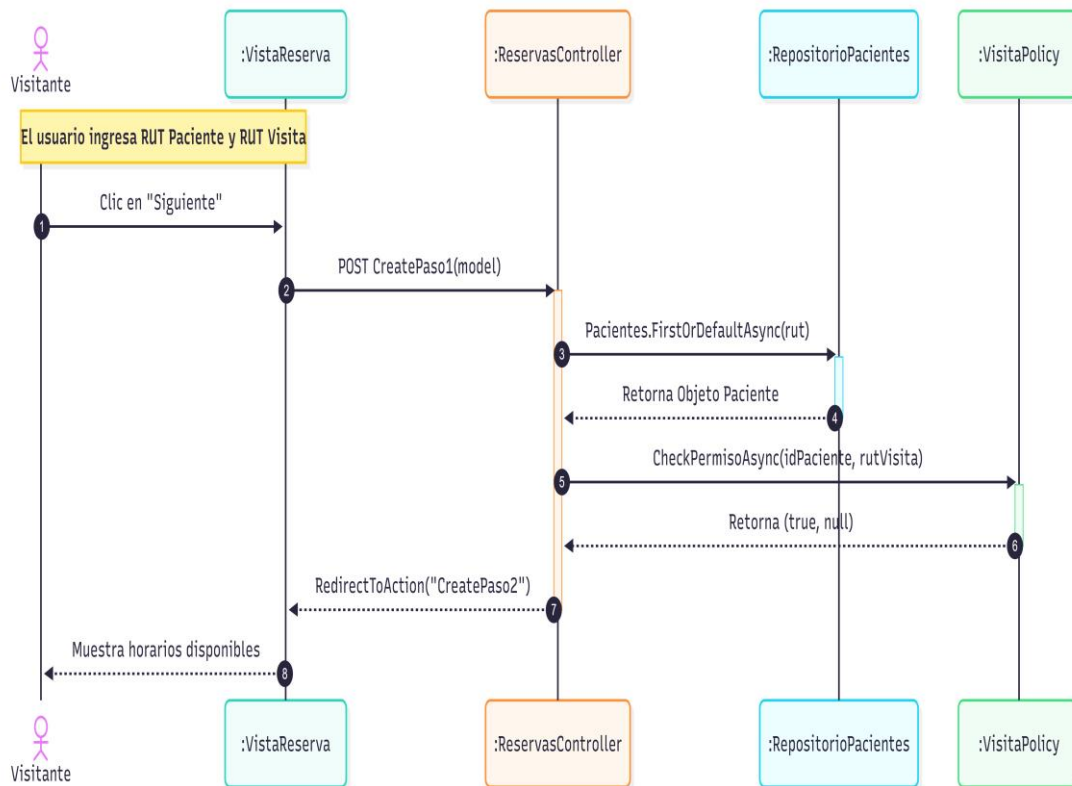


Figura 5-3. Diagrama de Secuencia: Verificación de Paciente. Fuente: Elaboración propia.

5.4.2. Confirmación de Reserva y Envío de Notificación

La **Figura 5-4** representa el flujo transaccional más importante del sistema. Una vez seleccionada la hora, el sistema persiste la reserva en SQL Server, genera dinámicamente el código QR mediante una librería interna y coordina el envío del correo electrónico de confirmación a través del servicio MailKit.

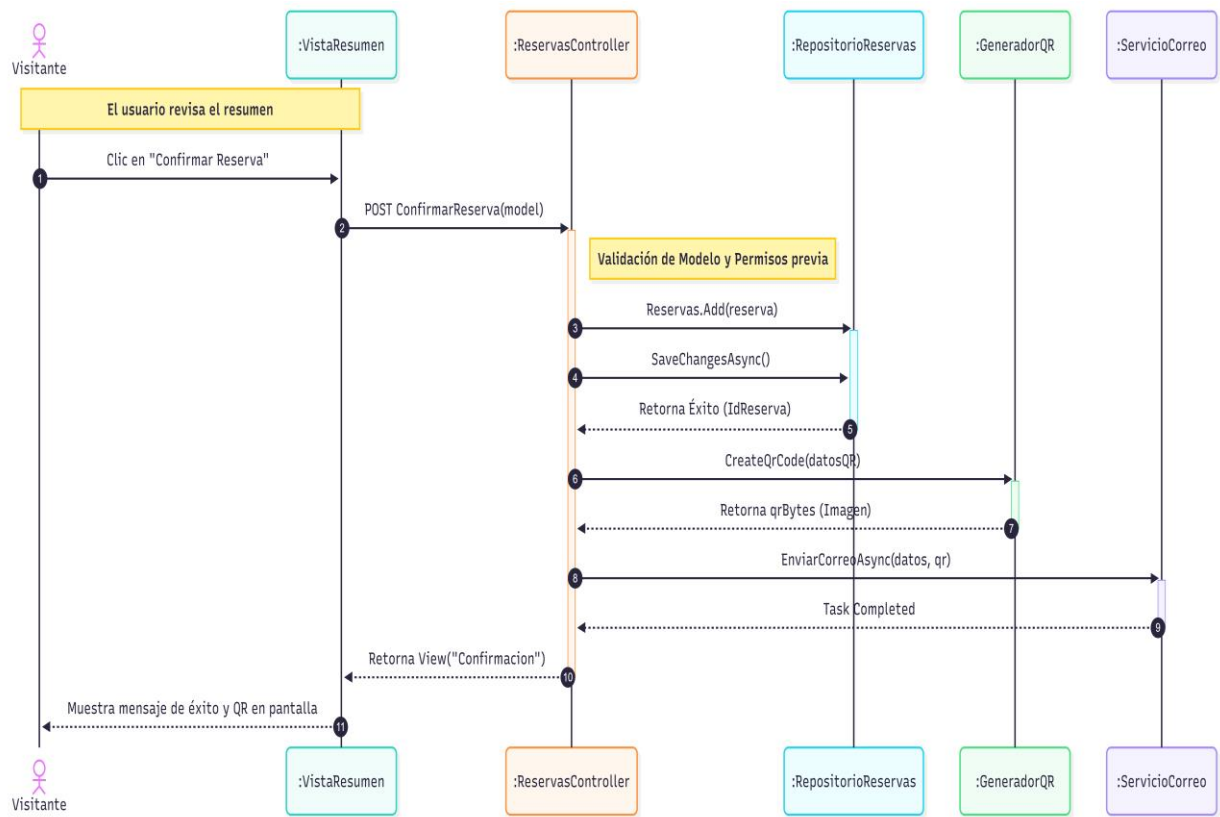


Figura 5-4. Diagrama de Secuencia: Confirmación de Reserva. Fuente: Elaboración propia.

5.4.3. Cancelación de Reserva

En el siguiente **figura 5-5** el flujo ilustra la cancelación de una visita ya reservada. Permite al usuario anular su hora agendada, liberando el cupo en la base de datos para que pueda ser utilizado por otros visitantes.

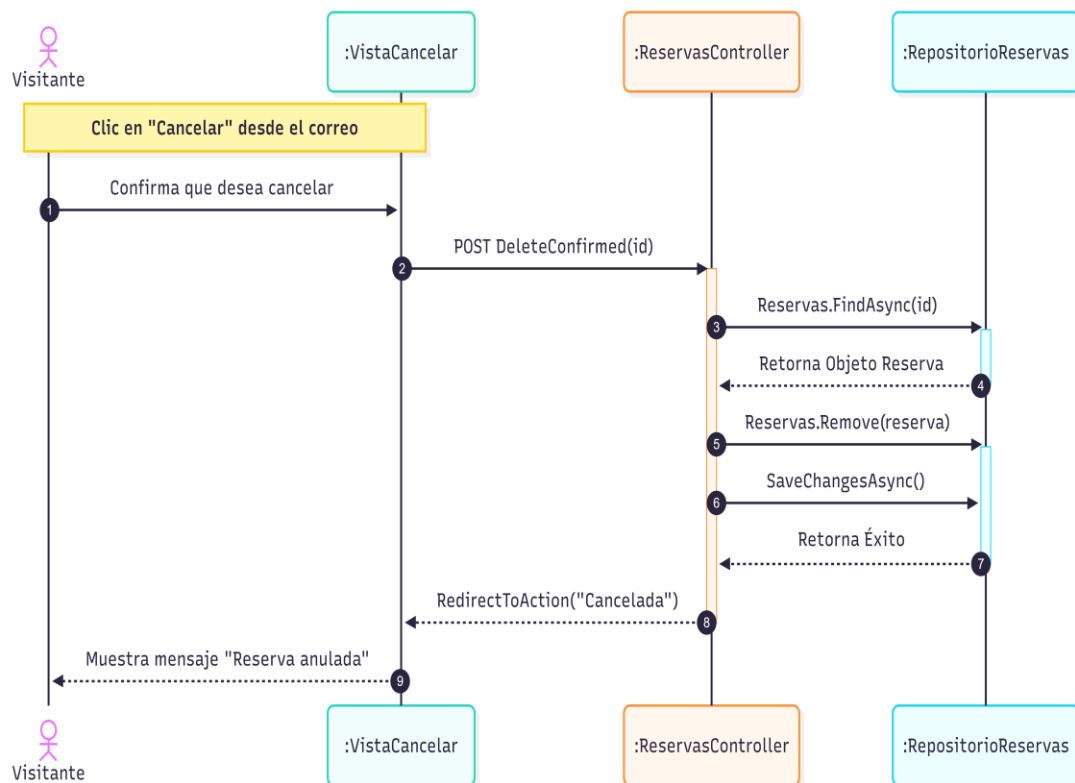


Figura 5-5. Diagrama de Secuencia: Cancelación de Reserva. Fuente: Elaboración propia.

5.4.4. Autenticación de Personal (Login)

La **figura 5-6** describe el proceso de seguridad mediante el cual el personal del hospital y los guardias acceden al sistema. El controlador AccountController valida las credenciales utilizando *ASP.NET Identity* y redirige al usuario al panel correspondiente (GuardiaDashboard o ClinicoDashboard) según su rol asignado.

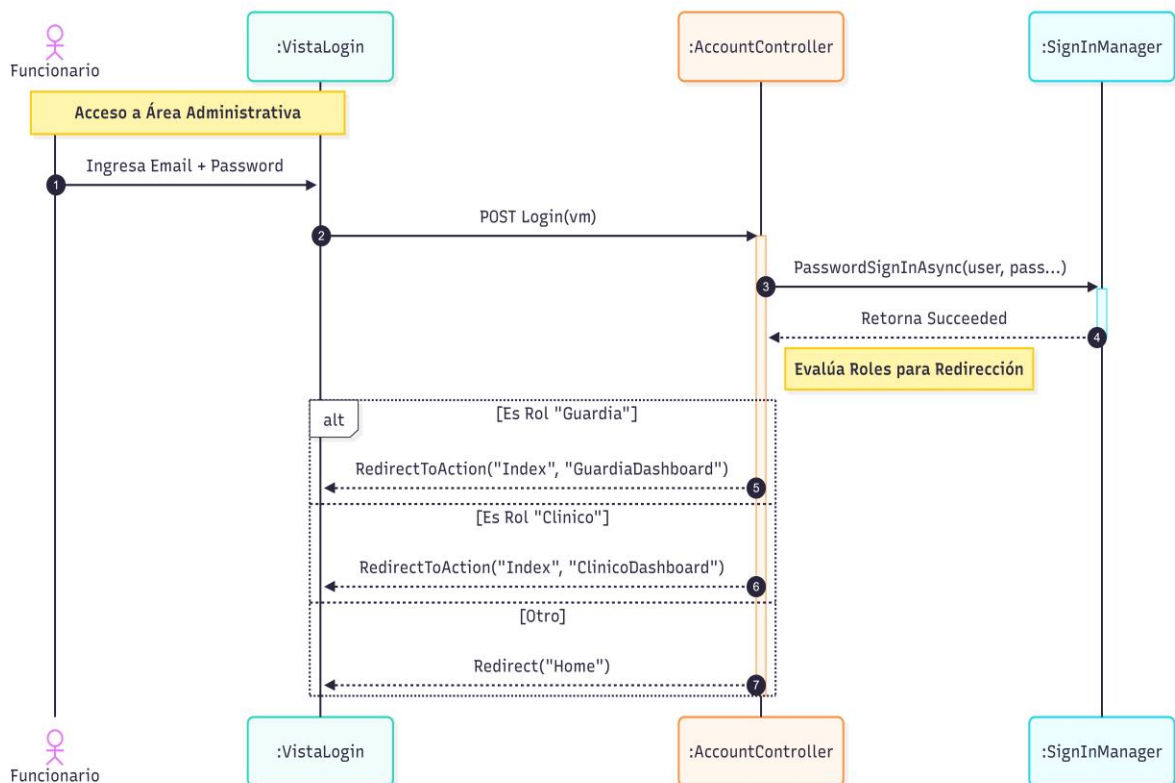


Figura 5-6. Diagrama de Secuencia: Autenticación de Personal. Fuente: Elaboración propia.

5.4.5. Gestión de Permisos y Bloqueos (Perfil Clínico)

Este diagrama (**Figura 5-7**) detalla las capacidades administrativas del personal clínico. Muestra la secuencia para buscar a un paciente y registrar una nueva restricción en la entidad *PacienteVisitanteRegla*, impidiendo futuras reservas según el criterio médico o administrativo.

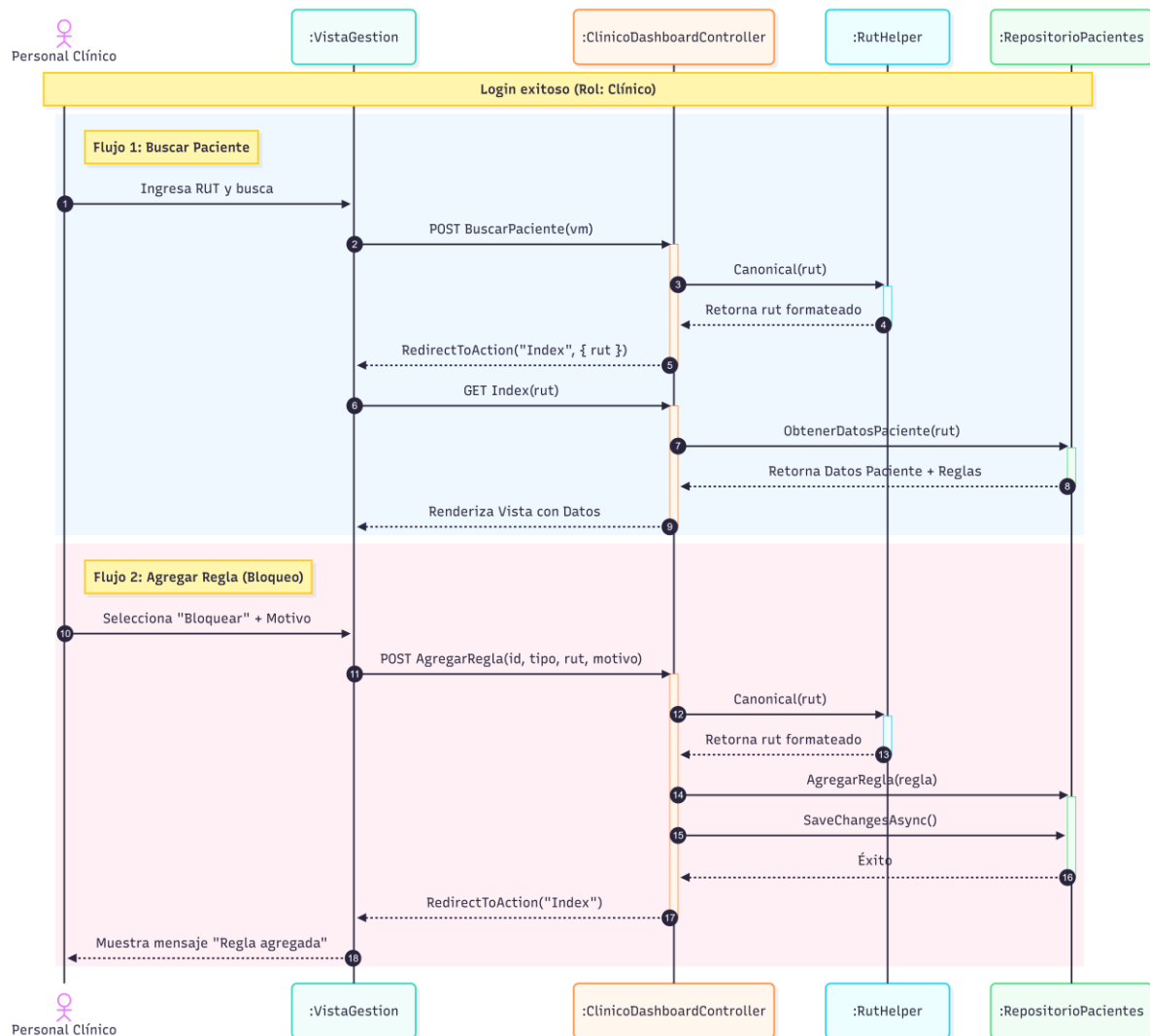


Figura 5-7. Diagrama de Secuencia: Gestión de Permisos (Clínico). Fuente: Elaboración propia

5.4.6. Validación de Acceso (Perfil Guardia)

Finalmente, la **Figura 5-8** exhibe el protocolo de control físico en el acceso al recinto. El guardia utiliza el lector de la aplicación para enviar la data del código QR al servidor; este último valida la integridad de la cadena y verifica la existencia del archivo en el sistema de archivos para autorizar o denegar el ingreso.

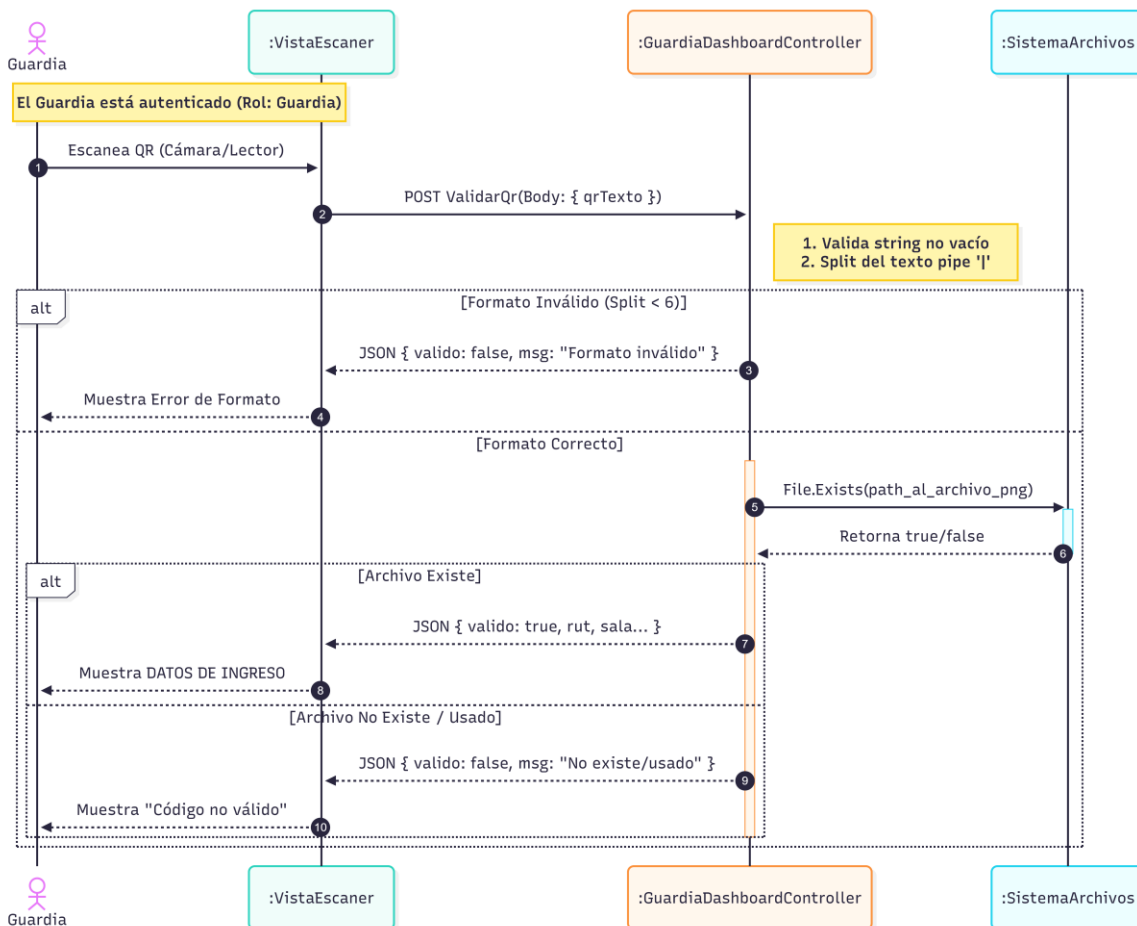


Figura 5-8. Diagrama de Secuencia: Validación de Acceso (Guardia). Fuente: Elaboración propia.

5.5 Diagrama del Modelo Relacional

El diagrama presentado en la Figura 5-9 corresponde al **Modelo Relacional** de la base de datos, generado directamente desde la implementación física en SQL Server. Este esquema representa la estructura estática de las tablas, sus atributos y las relaciones de integridad referencial que sustentan la lógica del sistema.

A continuación, se describen las entidades principales y cómo estas resuelven los requerimientos de información:

- **Gestión de Reservas:** La tabla Reservas actúa como la entidad transaccional central. Vincula mediante Claves Foráneas (FK) a un Usuario (visitante) con un Paciente en un bloque horario determinado. Además, se conecta con la tabla EstadoReserva para trazar el ciclo de vida de la cita (ej. Activa, Cancelada, Completada).
- **Ubicación y Capacidad:** La tabla Pacientes mantiene una relación referencial con Salas. Esto permite al sistema validar la ubicación física del paciente y controlar el aforo, verificando cuántos pacientes activos hay asignados a una sala específica en comparación con su capacidad máxima.
- **Estados y Normalización:** El esquema implementa la normalización de estados mediante tablas dimensionales como EstadoPaciente y EstadoReserva. Esto garantiza la integridad referencial de los datos, evitando valores inconsistentes en los registros clínicos y transaccionales.
- **Gestión de Turnos:** Se incorpora la entidad HorarioSala, la cual establece los bloques horarios permitidos y turnos de visita específicos para cada sala, permitiendo configurar regímenes diferenciados (ej. 24x7 o ventanas restringidas).
- **Reglas de Acceso:** La entidad PacienteVisitanteRegla permite aplicar lógica restrictiva granular. A través de esta tabla, el sistema registra bloqueos o permisos especiales, asociándolos opcionalmente a un RUT específico para cumplir con las normativas de visita del recinto.
- **Seguridad:** La tabla AspNetUsers gestiona las credenciales y roles del personal administrativo y de guardia, separando la infraestructura de autenticación de la lógica de negocio de las visitas.

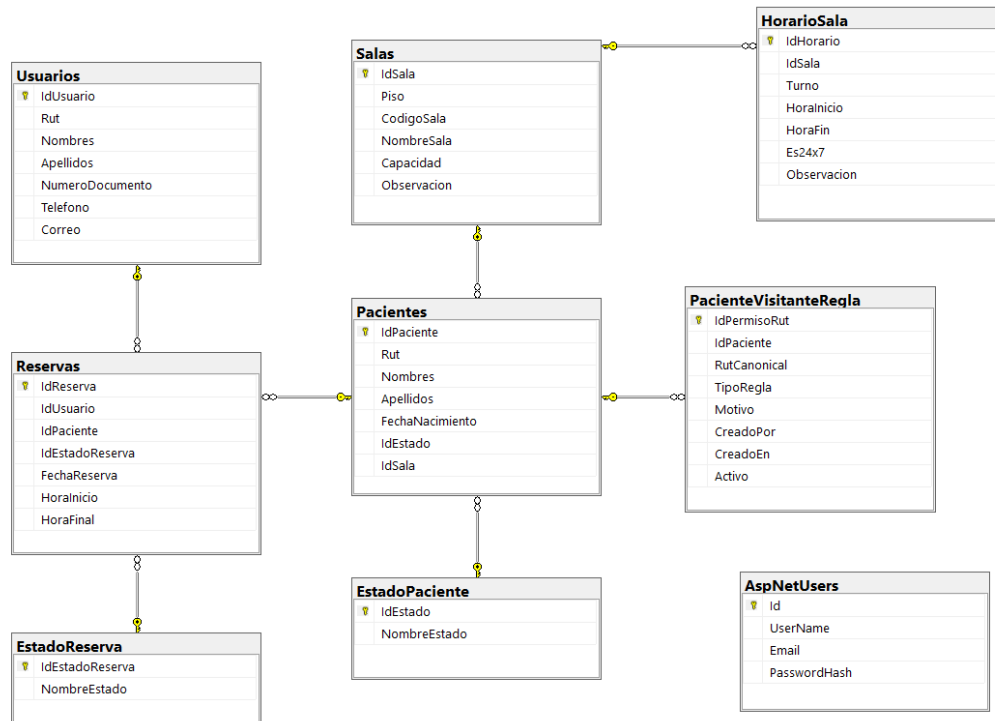


Figura 5-9. Diagrama de Clases (Generado desde SQL Server). Fuente: Elaboración propia.

5.6 justificación del Diseño

La selección de los artefactos de diseño presentados en este capítulo. Cumple con la necesidad de documentar el sistema de manera clara, estandarizada y escalable.

El uso del estándar UML y diagramas de base de datos relacionales permite que VisitaConnect no sea una "caja negra", sino un sistema transparente y entendible. Esto facilita la comprensión del flujo de datos entre la capa web y el servidor, asegura la integridad referencial de la información crítica (pacientes y reservas) y, fundamentalmente, deja una base documental sólida para que futuros equipos de desarrollo puedan mantener o extender la plataforma sin depender del conocimiento del equipo original.

6 Definición de Arquitectura y Patrones de Diseño

Para determinar la estructura fundamental de **VisitaConnect**, se analizaron los enfoques de despliegue y los patrones de diseño de software más adecuados para el

alcance y los recursos del proyecto. A continuación, se definen los conceptos claves utilizados.

6.1 Arquitectura de Despliegue: Enfoque Monolítico

La arquitectura monolítica es un modelo unificado donde todas las funcionalidades del sistema (interfaz de usuario, lógica de negocio y acceso a datos) se desarrollan, empaquetan y despliegan como una única unidad ejecutable.

- **Ventajas:** Es fácil de desarrollar inicialmente y simple de desplegar, ya que todo el sistema es un solo archivo o paquete.
- **Desventajas:** A medida que el sistema crece, el código se vuelve difícil de mantener. Un cambio pequeño en un módulo puede requerir recompilar y desplegar toda la aplicación. Además, si un componente falla, puede derribar el sistema completo.

6.2 Patrón de Diseño: Modelo-Vista-Controlador (MVC)

MVC es un patrón de arquitectura de software que estructura el código interno de la aplicación, dividiéndolo en tres componentes lógicos con responsabilidades separadas:

- **Modelo (Model):** Gestiona los datos y la lógica de negocio (ej. las reglas para reservar una visita).
- **Vista (View):** Es la interfaz gráfica que ve el usuario (HTML/CSS).
- **Controlador (Controller):** Actúa como intermediario, recibiendo las peticiones del usuario, procesando los datos con el Modelo y seleccionando la Vista adecuada para responder.

Este enfoque promueve la separación de responsabilidades, permitiendo que la lógica de negocio sea independiente de la interfaz visual.

6.3 Selección final para VisitaConnect

Para el desarrollo de **VisitaConnect**, se ha tomado la decisión estratégica de implementar una **Arquitectura Monolítica Modular basada en el patrón MVC**.

Esta elección no descarta las ventajas del monolito, sino que las potencia organizando su código interno mediante MVC. Las razones técnicas para esta configuración híbrida son:

1. **Eficiencia Operativa:** Al ser un MVP (Producto Mínimo Viable) para un entorno hospitalario, el despliegue monolítico garantiza una implementación rápida y un consumo de recursos optimizado, evitando la sobre ingeniería que implicaría una arquitectura de microservicios en esta etapa.
2. **Organización y Mantenimiento (El Patrón MVC):** Al separar las vistas de la lógica mediante controladores, se mitiga la principal desventaja del monolito (el

"código espagueti"). Si se necesita modernizar el diseño del "Portal de Visitantes", esto se puede realizar sin poner en riesgo la lógica crítica de validación de reservas.

3. **Desacoplamiento y Futuro:** El uso estricto de MVC prepara al sistema para la escalabilidad. Si el Hospital decide crear una aplicación móvil nativa a futuro, la lógica de negocio ya se encuentra aislada en los controladores y modelos, facilitando su exposición como API REST sin necesidad de reescribir el núcleo del sistema.

7 Implementación

7.1 Tecnologías utilizadas

Para la correcta construcción del sistema se seleccionó un conjunto de tecnologías robustas y modernas dentro del ecosistema Microsoft, complementadas con librerías de código abierto para funcionalidades específicas del sistema.

- **Lenguaje y Framework:** Se utilizó **C#** sobre el framework **.NET 8.0 (ASP.NET Core MVC)**. Esta elección garantiza un alto rendimiento, seguridad de tipos (validación de datos en tiempo de compilación) y soporte a largo plazo (LTS).
- **Base de Datos: Microsoft SQL Server** como motor de base de datos relacional, gestionado a través del ORM **Entity Framework Core** (Code-First), lo que facilitó la creación y migración de las tablas desde el código.
- **Frontend:** Se empleó **Razor Pages** con **HTML5** y **CSS3**, utilizando el framework **Bootstrap 5** para asegurar que el diseño sea *responsive* (adaptable a móviles) tal como exigen los requerimientos no funcionales.
- **Librerías Clave:**
 - **QRCode:** Para la generación dinámica de códigos QR en formato imagen (PNG) sin depender de APIs externas.
 - **MailKit:** Para la gestión segura del envío de correos electrónicos mediante el protocolo SMTP.
 - **ASP.NET Core Identity:** Para el manejo robusto de autenticación, autorización y roles (Guardia, Clínico).

7.2 Estructura del proyecto

El código fuente de VisitaConnect se organizó siguiendo las convenciones estándar de ASP.NET Core MVC, pero implementando una separación lógica mediante el uso de **Áreas** para desacoplar el portal público de la gestión administrativa.

Como se observa en la **Figura 7-1**, la estructura principal de la solución se divide en tres grandes bloques lógicos:

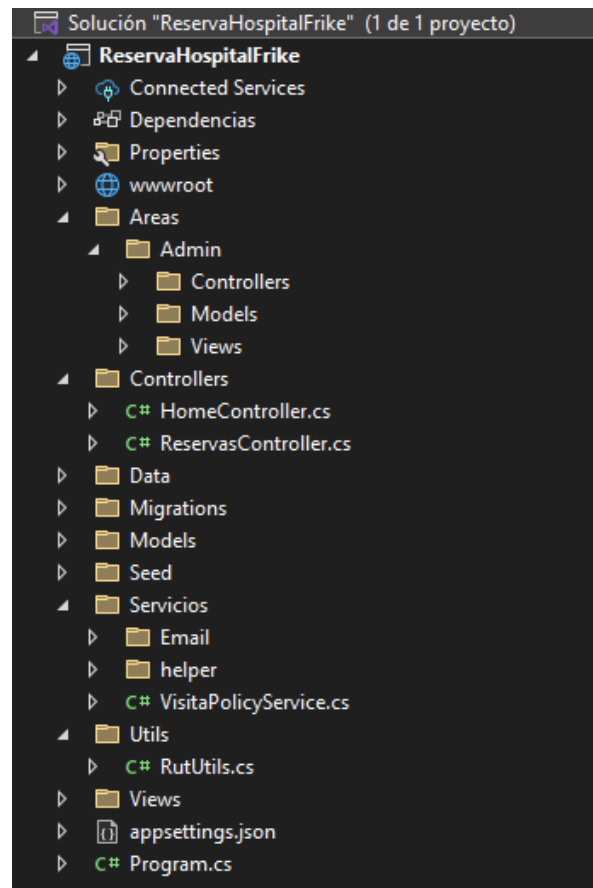


Figura 7-1. Estructura de carpetas de la solución en Visual Studio. Fuente: Elaboración propia.

- **Raíz del Proyecto (Portal Público):**

Contiene los controladores de acceso público

- **HomeController:** Gestiona la página de inicio (landing page) y la navegación estática, sirviendo como punto de entrada liviano al sistema.
-
- **ReservasController:** Centraliza toda la lógica de negocio crítica, incluyendo la búsqueda de pacientes, el flujo secuencial para realizar una reserva exitosamente y los servicios externos para el envío de correos y generación de QR.

- **Áreas/Admin:**

Para implementar un área de administración se creó un apartado dedicado para el personal del hospital (Guardias y Personal clínico) Esta sección funciona como un subsistema independiente al de reservas, que cuenta con controladores y vistas propias.

- **ClinicoDashboardController** para la gestión de pacientes.

- **GuardiaDashboardController** para la validación de accesos.
- **AccountController** para el manejo de sesiones mediante ASP.NET Core Identity.
- **Services & Utils:** Se extrajo la lógica reutilizable a carpetas específicas para mantener los controladores ligeros (*Thin Controllers*).
 - **Utils:** Contiene clases estáticas como RutUtils, encargada de la validación algorítmica y formateo del RUT.
 -
 - **Servicios:** Aloja la lógica de negocio desacoplada, como el servicio de envío de correos (IEmailSender) y las políticas de visita (VisitaPolicy).
 -
 - **Seed:** Incluye la lógica de inicialización de datos (IdentitySeeder) para crear los usuarios y roles base del sistema.

7.3 Desafíos de implementación

Durante la fase de implementación de la plataforma, se abordaron desafíos de alta complejidad técnica que requirieron el diseño de algoritmos específicos para garantizar tanto la integridad de los datos transaccionales como la seguridad en el control de acceso. A continuación, se detallan los dos desafíos más relevantes:

1. Lógica de Generación Dinámica de Bloques Horarios

Uno de los requisitos críticos fue la transformación de los turnos configurados por sala (AM/PM) en bloques temporales discretos y seleccionables por el usuario final.

El Problema: No era suficiente con generar bloques estáticos para la disponibilidad de horarios ya que debían ser modificables con el tiempo, por ello se debían generar bloques de horario dinámicamente con intervalos de 60 minutos, filtrar aquellos que ya tenían una reserva confirmada en la base de datos y validar que no existieran dos reservas en el mismo tiempo

La Solución: Se implementó un algoritmo en el ReservasController que genera una lista de objetos TimeOnly para el día seleccionado. Mediante consultas LINQ optimizadas con AsNoTracking, el sistema cruza esta lista con la tabla de Reservas existentes, excluyendo matemáticamente los bloques ocupados antes de enviarlos a la vista. Esto garantiza que el usuario solo visualice cupos reales y evita las reservas duplicadas con el mismo horario.

2. Generación y Validación de Credenciales de Acceso (QR)

El control de acceso requería un mecanismo seguro y verificable que no dependiera exclusivamente de la conexión a internet del visitante en el momento del ingreso, en un inicio se optó por la lectura del código QR del carnet de identidad, pero rápidamente fue descartado por problemas de complejidad al momento de retornar los datos. Por ello se requirió implementar una forma de validación QR generada dinámicamente en la plataforma.

El Problema: Enviar un correo de confirmación no era suficiente; se necesitaba un código visual único que el guardia pudiera validar físicamente en el hospital.

La Solución: Se integró la librería QRCode para generar una imagen PNG en el servidor al momento de confirmar la reserva. El contenido del QR es una cadena de texto estructurada (RUT|Nombre|Documento|ID_Archivo...). Para la validación, el sistema del guardia no solo decodifica el texto, sino que verifica la existencia física del archivo generado en el servidor (System.IO.File.Exists), impidiendo así el uso de códigos QR falsificados o generados externamente en otras plataformas.

8 Resultados y Evaluación

Una vez finalizada la etapa de implementación, la plataforma *VisitaConnect* fue sometida a un proceso de pruebas funcionales para validar el cumplimiento de los requerimientos definidos en el Capítulo 4.

El objetivo de esta evaluación fue verificar que los flujos críticos del sistema (reserva, notificación, validación y gestión) operaran correctamente bajo condiciones de uso normales y ante posibles errores de entrada.

8.1 Pruebas del Portal de Visitantes

Caso de Prueba 1: Verificación de Paciente

El objetivo de esta prueba es validar que el sistema identifique correctamente a un paciente habilitado en la base de datos simulada (HIS).

- **Procedimiento:** Para ello se seguirá el siguiente Procedimiento: Se debe realizar una reserva hasta llegar campo "Rut Paciente", una vez escrito el Rut se debe rellenar el bloque de "nombre paciente" y debe permitir ingresar a selección de horarios
- **Resultado Obtenido:** Como se aprecia en la imagen 8.1 al paciente satisfactoriamente redirigió exitosamente a la selección de horarios.

Evidencia: Ver [Figura 8-1](#): Pantalla de selección de horario tras validación exitosa.

Caso de Prueba 2: Validación de Paciente No Existente

En este Objetivo se verificará que el sistema maneje correctamente los RUTs no registrados de pacientes.

- **Procedimiento:** Se debe ingresar un RUT de paciente que no existe en la base de datos, y verificar que no permita redirigir a selección de horario.
- **Resultado Obtenido:** El controlador retornó el modelo con un error y la vista desplegó una alerta: "El paciente no se encuentra en la base de datos".
- **Evidencia:** Ver [Figura 8-2](#): Alerta de error en el formulario de búsqueda.

Caso de Prueba 3: Confirmación de Reserva y Envío de Correo

Para este caso de prueba se busca verificar la persistencia de datos y la integración con los servicios de correo y QR para ello verificaremos que llegue correctamente un correo electrónico con la información requerida (Código QR).

- **Procedimiento:** Se debe realizar un reserva en el sistema relleno el formulario de visitante, una vez realizada la reserva exitosamente se debe revisar correo para verificar el envío automático.
- **Resultado Obtenido:** La reserva se guardó en la base de datos. El sistema generó el código QR en el servidor y el correo electrónico llegó a la bandeja de entrada del usuario con el adjunto correcto.
- **Evidencia:** Ver [Figura 8-3](#), [Figura 8-4](#): Correo de confirmación con código QR
Ver [Figura 8-5](#): Creación de reserva en base de datos

Caso de Prueba 4: Cancelación de Reserva

Como ultimo Objetivo se debe comprobar que el usuario puede anular su visita y liberar el cupo para otro visitante.

- **Procedimiento:** Se hace clic en el botón "Cancelar Reserva" proveniente del correo electrónico generado al momento de realizar una reserva, este debe redirigir a una vista para cancelar la reserva.
- **Resultado:** El sistema eliminó el registro de la base de datos y mostró la confirmación de anulación dejando disponible el bloque para otro visitante.
- **Evidencia:** Ver [Figura 8-6](#): Proceso de cancelación de hora
Ver [Figura 8-7](#): Resultados en base de datos

8.2 Pruebas del Portal Administrativo

Caso de Prueba 5: Autenticación y Seguridad

- **Objetivo:** Asegurar que el acceso al área administrativa esté restringido por roles.
- **Procedimiento:** Se intentó ingresar con credenciales de Guardia.
- **Resultado Obtenido:** El sistema autenticó las credenciales mediante *Identity* y redirigió específicamente al GuardiaDashboard, restringiendo el acceso a las vistas clínicas.
- **Evidencia:** Ver [Figura 8-8](#): Panel de control del Guardia tras el inicio de sesión
Ver [Figura 8-9](#): resultados de la validación.

Caso de Prueba 6: creación de reglas para personal clínico

- **Objetivo:** Asegurar que el sistema de bloqueo de ruts y estados de paciente funciona correctamente
- **Procedimiento:** Se ingresará a la vista administrativa clínica y se bloqueará por ejemplo el rut '21.508.578-3' luego se intentará hacer una reserva a el paciente que tiene bloqueado el rut ya dicho. Para finalizar se revisará la Base de Datos para visualizar la regla creada

- **Resultado Obtenido:** El sistema muestra en pantalla satisfactoriamente que no se puede realizar una reserva con el rut bloqueado y queda registrado satisfactoriamente en la base de datos
-
- **Evidencia:** Ver [Figura 8-10](#), [Figura 8-11](#), [Figura 8-12](#) y [Figura 8-13](#): : Proceso de creación de regla
Ver [Figura 8-14](#): regla creada en base de datos.

9 Conclusiones y Trabajo Futuro

9.1 Conclusiones Generales

El desarrollo de *VisitaConnect* ha permitido abordar una problemática crítica en la gestión hospitalaria del Hospital Dr. Gustavo Fricke: la ineficiencia de los procesos manuales de registro de visitas. Mediante la implementación de una plataforma web basada en **ASP.NET Core MVC** y la aplicación de la metodología ágil **Scrum**, se logró construir un sistema robusto, escalable y centrado en el usuario.

El proyecto cumplió satisfactoriamente con su objetivo general de diseñar e implementar una solución digital que optimiza el registro y control de acceso. La validación mediante pruebas funcionales demostró que la digitalización reduce la incertidumbre de los visitantes y otorga herramientas de control efectivas al personal de seguridad y clínico, modernizando la experiencia de servicio en el recinto.

Desde la perspectiva de la ingeniería de software, la adopción de una arquitectura por capas y el uso de patrones de diseño como Inyección de Dependencias permitieron crear un código limpio y mantenible. Asimismo, el rol de Scrum Master facilitó la adaptación ante desafíos técnicos complejos, como la lógica de bloques horarios y la generación de credenciales QR, asegurando la entrega continua de valor en cada Sprint.

9.2 Cumplimiento de Objetivos Específicos

A continuación, se detalla el cumplimiento de los objetivos planteados al inicio del proyecto:

1. **Analizar la situación actual:** Se logró identificar las falencias del proceso manual (cuaderno de registro, falta de información de horarios) y las necesidades de los distintos actores (Visitante, Guardia, Clínico), lo que fundamentó los requerimientos del sistema.
2. **Levantar requerimientos funcionales:** Se definieron y validaron los flujos críticos, estableciendo la necesidad de roles diferenciados y un mecanismo de validación (QR), los cuales fueron documentados en el Capítulo 4.
3. **Investigar tecnologías adecuadas:** La selección del stack tecnológico (.NET 8, SQL Server, QRCode, MailKit) demostró ser acertada, proporcionando el rendimiento y la seguridad necesarios para un entorno hospitalario.

4. **Diseñar la arquitectura del sistema:** Se implementó exitosamente el patrón MVC con una separación clara de responsabilidades (UI, Lógica, Datos) y un subsistema administrativo desacoplado mediante el uso de *Areas*, tal como se detalló en el Capítulo 5.
5. **Implementar el módulo de reservas:** Se desarrolló un flujo de reserva intuitivo que gestiona la disponibilidad de horarios en tiempo real, evitando solapamientos y mejorando la experiencia del visitante.
6. **Desarrollar el sistema de administración:** Se entregó un panel funcional para el personal interno, permitiendo la gestión de estados de pacientes y la validación segura de accesos mediante escaneo de QR.

9.3 Limitaciones del Proyecto

Si bien el sistema es funcional y cumple con los requerimientos del MVP (*Producto Mínimo Viable*), existen limitaciones propias del alcance académico:

- **Integración HIS Simulada:** Debido a restricciones de seguridad y acceso a la red hospitalaria real, la integración con el Sistema de Información Hospitalario (HIS) se realizó mediante una simulación de base de datos local. En un entorno productivo, esto debe reemplazarse por el consumo de APIs reales del hospital.
- **Cobertura de Notificaciones:** Actualmente, las notificaciones se envían exclusivamente por correo electrónico. No se implementó el envío de SMS o WhatsApp, lo cual podría limitar el alcance en usuarios con menor acceso a datos móviles.
- **Validación de RUT con Servicio Externo:** La integración con el servicio de validación de identidad (APIRUT) no pudo ser completada debido a inestabilidad y falta de soporte técnico por parte del proveedor del servicio gratuito durante el ciclo de desarrollo, optándose por una validación algorítmica local.

9.4 Trabajo Futuro

Para evolucionar VisitaConnect desde un MVP hacia una solución productiva de gran escala, se proponen las siguientes líneas de trabajo:

- **Implementación de API Gateway:** Desarrollar una capa de integración real para conectar el sistema con el HIS del hospital mediante protocolos estándar de salud HL7 o FHIR.
- **Desarrollo de Aplicación Móvil Nativa:** Crear una app dedicada (Android/iOS) que permita a los visitantes guardar sus tickets QR en la "billetera digital" del teléfono y recibir notificaciones *push* sobre el estado de salud de su familiar.
- **Módulo de Analítica de Datos:** Incorporar un dashboard de inteligencia de negocios (BI) para que la administración del hospital pueda visualizar estadísticas

de flujo de visitas, horarios punta y saturación de salas, apoyando la toma de decisiones.

10 Agradecimientos

Quiero agradecer desde lo más profundo de mi corazón a las 2 personas que estuvieron siempre conmigo dándome su apoyo y amor incondicional, mis padres Andrés y Jeymmy fueron no solo un pilar fundamental en mi instancia en la universidad si no mi motor principal para seguir adelante, gracias por sus motivación y comprensión incluso en los momentos más difíciles siempre estuvieron creyendo en que podía lograrlo, gracias por darme un abrazo o palmadita en la espalda cuando lo necesite por su comprensión y cariño, son y siempre serán los mejores padres del mundo.

Agradecer también a mi hermano Santino que era mi momento de desconexión jugando un rato o riéndonos de alguna cosa.

A mi novia Claudia que siempre me dio su amor incondicional y comprensión en todo momento.

Agregar a esto a la Universidad Técnica Federico Santa María y a mis profesores, en especial a la profesora **Lais San Martín**, por su guía, exigencia y conocimientos transmitidos, nunca tuvo un problema para darme un minuto de su tiempo y poder resolver mis dudas. Siempre estaré agradecido por su compromiso y cariño los cuales fueron claves para mi formación profesional.

También A mis amigos, Christian, Alexander, Melissa, Camilo, Matias y Cristóbal por el compañerismo, el apoyo mutuo y el trabajo en equipo durante esta etapa académica.

Cada una de las personas con las que compartí, reí o hablé se quedan con un espacio en mi corazón

Finalmente, a Dios y a la vida, por darme la fortaleza para concluir exitosamente este importante desafío y logro personal.

11 Referencias

- [1] Beck, K. (2004). *Extreme Programming Explained: Embrace Change.* Addison-Wesley.
- [2] Martin, R. C. (2002). *Agile Software Development: Principles, Patterns, and Practices.* Prentice Hall.
- [3] Pressman, R. S. (2010). *Ingeniería del Software: Un enfoque práctico.* McGraw-Hill.
- [4] Sommerville, I. (2011). *Ingeniería del Software (9ª ed.).* Pearson Educación.
- [5] Ministerio de Salud de Chile. (2018). *Hospital Digital: Estrategia nacional de transformación digital en salud.* Recuperado de <https://www.hospitaldigital.gob.cl>
- [6] Microsoft Docs. (2024). *ASP.NET MVC Overview.* Recuperado de <https://learn.microsoft.com/aspnet/mvc>
- [7] Fowler, M. (2003). *Patterns of Enterprise Application Architecture.* Addison-Wesley.

12 Anexos



Figura 8-1. Pantalla de selección de horario tras validación exitosa. Fuente: Elaboración propia.

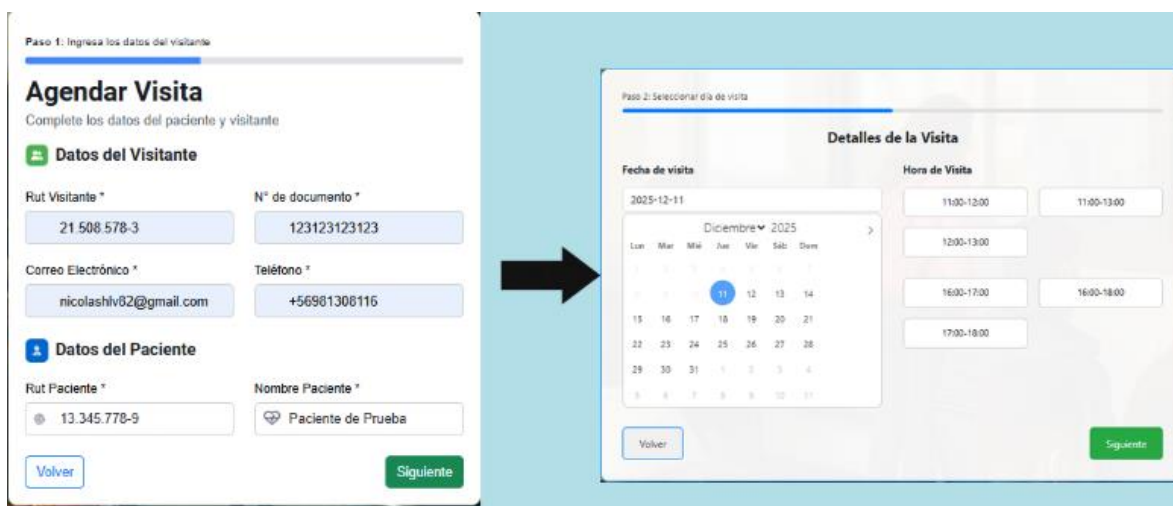


Figura 8-2. Alerta de error: Paciente no encontrado. Fuente: Elaboración propia.



Figura 8-3. Correo de confirmación con código QR adjunto. Fuente: Elaboración propia.

Por favor, revise los detalles a continuación:

Nombre del Paciente: Paciente de Prueba
Fecha: 01-12-2025
Hora: 11:00
Sala paciente: Médica Adulto - 4 Oriente
Contacto visita: Tel: 981308116 · Correo: nicolashlv82@gmail.com

Cancelar reserva

Por favor, llegue 15 minutos antes de su visita. Si necesita reprogramar o cancelar, le pedimos que nos contacte con al menos 24 horas de anticipación.

Agradecemos su colaboración y estamos a su disposición para cualquier pregunta o asistencia adicional.

Atentamente,

Equipo de Atención al Paciente de Hospital Gustavo Frick

Utiliza este QR para poder ingresar a tu visita:

Un archivo adjunto · Analizados por Gmail ·  Añadir a Drive



Figura 8-4. Correo de confirmación con código QR adjunto. Fuente: Elaboración propia.

	IdReserva	IdUsuario	IdPaciente	IdEstadoReserva	FechaReserva	Horainicio	HoraFinal
82	1128	1010	1009	1	2025-11-22	16:00:00.0000000	17:00:00.0000000
83	1129	1010	1009	1	2025-11-24	16:00:00.0000000	17:00:00.0000000
84	1130	1011	1009	1	2025-11-22	17:00:00.0000000	18:00:00.0000000
85	1131	1011	1008	1	2025-11-27	12:00:00.0000000	13:00:00.0000000
86	1132	1011	1009	1	2025-11-25	16:00:00.0000000	17:00:00.0000000
87	1133	1011	1007	1	2025-11-26	12:00:00.0000000	13:00:00.0000000
88	1134	1010	1009	1	2025-11-27	16:00:00.0000000	17:00:00.0000000
89	1135	1011	1007	1	2025-11-25	16:00:00.0000000	17:00:00.0000000
90	1136	1010	1009	1	2025-11-26	12:00:00.0000000	13:00:00.0000000
91	1137	1010	1009	1	2025-11-30	12:00:00.0000000	13:00:00.0000000
92	1138	1010	1009	1	2025-11-30	16:00:00.0000000	17:00:00.0000000
93	1139	1011	1009	1	2025-11-29	11:00:00.0000000	12:00:00.0000000
94	1142	1011	1005	1	2025-12-01	11:00:00.0000000	12:00:00.0000000

Figura 8-5. Base de datos con la reserva agregada. Fuente: Elaboración propia.

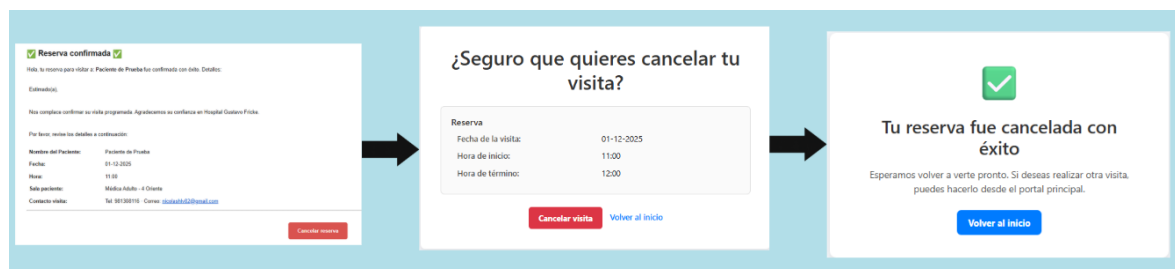


Figura 8-6. Mensaje de confirmación de reserva cancelada. Fuente: Elaboración propia.

	IdReserva	IdUsuario	IdPaciente	IdEstadoReserva	FechaReserva	Horainicio	HoraFinal
81	1127	1010	1009	1	2025-11-25	11:00:00.0000000	12:00:00.0000000
82	1128	1010	1009	1	2025-11-22	16:00:00.0000000	17:00:00.0000000
83	1129	1010	1009	1	2025-11-24	16:00:00.0000000	17:00:00.0000000
84	1130	1011	1009	1	2025-11-22	17:00:00.0000000	18:00:00.0000000
85	1131	1011	1008	1	2025-11-27	12:00:00.0000000	13:00:00.0000000
86	1132	1011	1009	1	2025-11-25	16:00:00.0000000	17:00:00.0000000
87	1133	1011	1007	1	2025-11-26	12:00:00.0000000	13:00:00.0000000
88	1134	1010	1009	1	2025-11-27	16:00:00.0000000	17:00:00.0000000
89	1135	1011	1007	1	2025-11-25	16:00:00.0000000	17:00:00.0000000
90	1136	1010	1009	1	2025-11-26	12:00:00.0000000	13:00:00.0000000
91	1137	1010	1009	1	2025-11-30	12:00:00.0000000	13:00:00.0000000
92	1138	1010	1009	1	2025-11-30	16:00:00.0000000	17:00:00.0000000
93	1139	1011	1009	1	2025-11-29	11:00:00.0000000	12:00:00.0000000

Figura 8-7. Imagen de la base de datos con la reserva eliminada. Fuente: Elaboración propia.

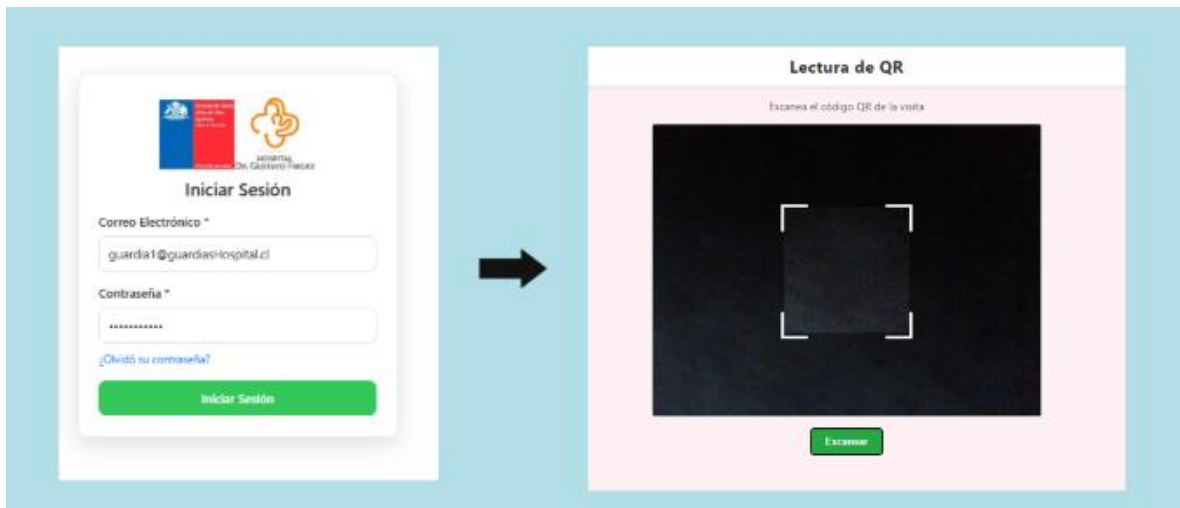


Figura 8-8. Panel de control del Guardia tras inicio de sesión. Fuente: Elaboración propia

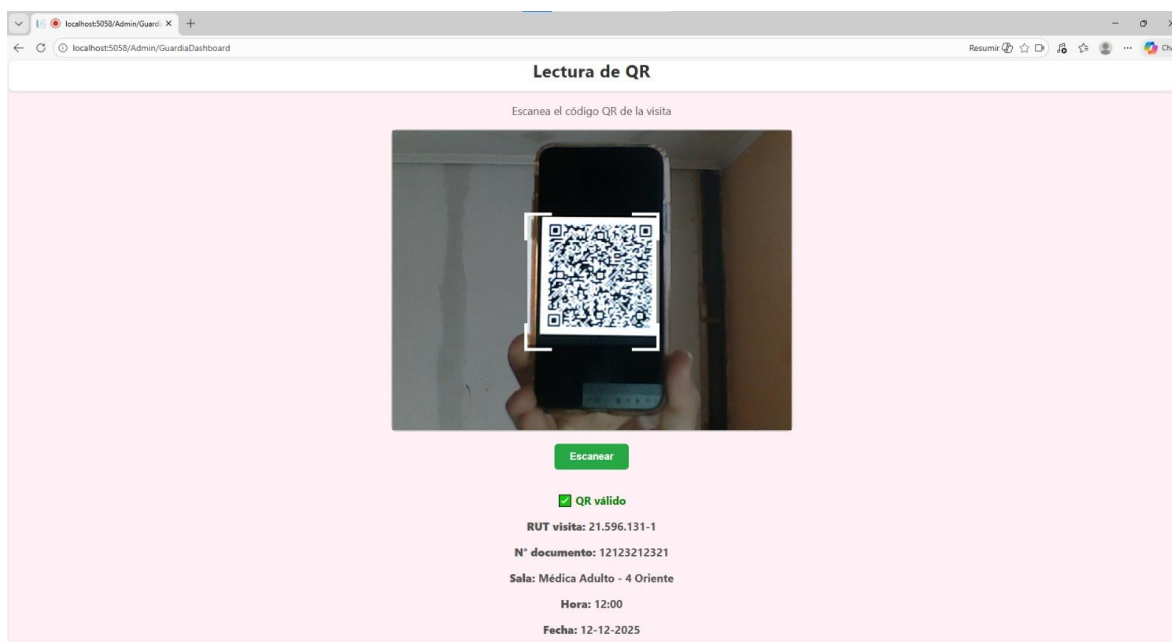


Figura 8-9. Resultado de validación de acceso exitosa (QR Válido). Fuente: Elaboración propia.

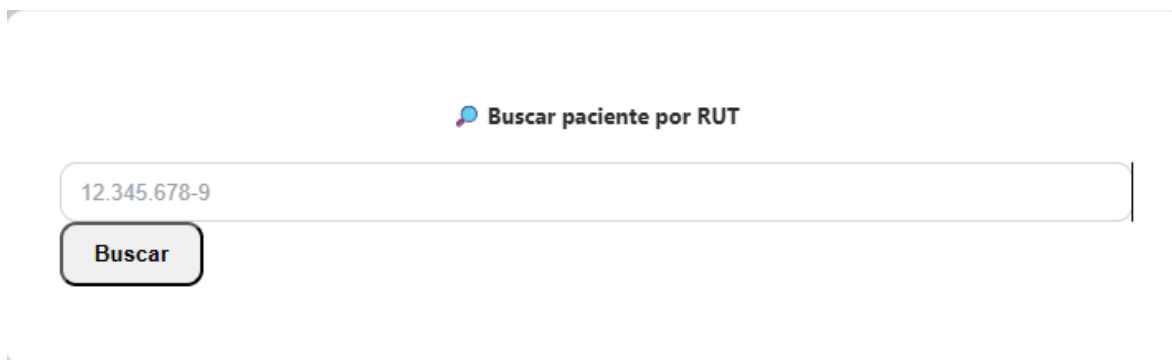


Figura 8-10. Vista de gestión de reglas de bloqueo para personal clínico buscar paciente. Fuente: Elaboración propia.



 Paciente

Nombre: Paciente de Prueba

RUT: 13.345.778-9

Piso: —

Estado: 1

Estado de visitas

1 – Con visitas

Actualizar estado

 Agregar regla por RUT

RUT

21.508.578-3

Tipo

Bloquear

Motivo (opcional)

Orden de Alejamiento

Agregar

Figura 8-11. Resultado al buscar paciente y se rellenan los datos para agregar una regla.
Fuente: Elaboración propia

 RUTs bloqueados

RUT	Motivo	Acción
215085783	Orden de Alejamiento	<button>Quitar</button>

 RUTs habilitados

RUT	Motivo	Acción
No hay habilitados		

Figura 8-12. Resultado al agregar una regla. Fuente: Elaboración propia

Paso 1: Ingrese los datos del visitante

Agendar Visita

Complete los datos del paciente y visitante

Datos del Visitante

- Tu RUT está bloqueado para este paciente.

Rut Visitante *	N° de documento *
21.508.578-3	123123123123
Correo Electrónico *	Teléfono *
nicolashlv82@gmail.com	+56981308116

Datos del Paciente

Rut Paciente *	Nombre Paciente *
13.345.778-9	Paciente de Prueba

Figura 8-13. Resultado al intentar realizar una reserva con un Rut bloqueado. Fuente: Elaboración propia



IdPermisoRut	IdPaciente	RutCanonical	TipoRegla	Motivo	CreadoPor	CreadoEn
10	1005	215085783	BLOQUEAR	Orden de Alejamiento	90DFBF59-5E51-422D-A68D-B9D1D7D30683	2025-12-01 21:12:59.8246778

Figura 8-14. Imagen de base de datos con regla creada Fuente: Elaboración propia