PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

FACULTAD DE INGENIERIA

ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN SISTEMAS Y COMPUTACIÓN

IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DE INFORMACIÓN PARA LA GESTIÓN DE CONSULTORIOS MÉDICOS DE DIFERENTES ESPECIALIDADES

AUTOR:

ALEJANDRO MANUEL VIVANCO MOSQUERA

DIRECTOR:

ING. ALFREDO CALDERÓN

QUITO DM, 2020

Resumen

El presente trabajo de disertación busca diseñar, implementar y probar un sistema de información web utilizando la metodología ágil XP para un desarrollo integro y corrección de errores en cada iteración de XP. El sistema permite la gestión de citas, datos e historial médico de pacientes, y la emisión de prescripción médica. Para el desarrollo de esta aplicación se habló con médicos que señalaron la falta de sistemas de información en sus consultorios para administrar los datos e historias clínicas de los pacientes, de igual manera la ONU señala la falta de sistemas de E-Salud en Latinoamérica. Con esto en mente se realizó un levantamiento de requerimientos recopilando las opiniones de diferentes médicos para las funcionalidades del sistema.

Una vez se tenga claro los requerimientos esquematizados en historias de usuario se realizó el desarrollo del sistema utilizando la metodología XP en siete iteraciones, las cuales están compuestas por diseño codificación y pruebas. En cada iteración se aplicaron estándares para la programación, el diseño y las pruebas, de esta forma se obtuvo un código limpio y guiado por pruebas cuyos diseños son simples para el usuario. Finalmente, la aplicación fue desplegada en diferentes plataformas electrónicas para su uso práctico.

Dedicatoria

A mis abuelos,

Por su ayuda incondicional a lo largo de toda mi vida estudiantil y su enorme esfuerzo por para ayudarme a alcanzar mis objetivos.

A mi madre,

Por haberme inculcado valores a lo largo de mi vida, por su esfuerzo de estar conmigo durante mi vida estudiantil y mostrarme la importancia de estudiar y no parar de aprender.

A mi hermano,

Por estar siempre conmigo y ayudarme a recorrer mi camino siempre a mi lado.

Agradecimientos

Agradezco a mi familia por ayudarme a conseguir mis objetivos profesionales y apoyarme a lo largo de toda mi vida estudiantil.

Agradezco al Ing. Alfredo Calderón por sus consejos, ayuda y observaciones a lo largo del desarrollo de este trabajo de disertación.

Agradezco a mis profesores de la facultad de Ingeniería quienes me han compartido su conocimiento y consejos para triunfar como profesional.

Agradezco a mis amigos de carrera quienes me han apoyado y aconsejado a lo largo de la carrera.

Tabla de Contenidos

1. Introducción

1.1. Justificación

Según la OMS eSalud se define como "el uso coste-efectivo y seguro de las tecnologías de la información y comunicación (TIC) en apoyo a salud y a los ámbitos relacionados con la salud" (Castañeda, 2019). Este término incluye la implementación de sistemas de administración o recursos como: agenda de citas, laboratorio clínico, expediente clínico, prescripción electrónica, uso de dispositivos móviles, sistemas de imagenología, sistemas de atención a distancia y la enseñanza a través de medios digitales. Hoy en día estudios han demostrado la eficiencia y efectividad que supone la integración de sistemas de información en el área de la salud, es por esto que la OMS, OPS, OCDE y Cepal han emitido políticas orientadas a la implementación de TIC (Castañeda, 2019).

Con este contexto se puede mencionar que los sistemas de información orientados a la salud son de gran importancia dentro de consultorios médicos ya que son una herramienta efectiva para el manejo eficiente de los procesos médicos.

En la actualidad los sistemas de información se han convertido en la herramienta principal para la gestión y control de procesos dentro de empresas, negocios, organizaciones, etc. Esto se debe a la eficiencia que ofrecen para administrar, procesar y almacenar grandes cantidades de datos.

Gracias a las tecnologías actuales de desarrollo web, los sistemas de información se han transformado en herramientas integrales en línea que ayudan en las operaciones y procesos de las grandes organizaciones (Céspedes, 2019). La tecnología se ha vuelto el medio principal para la gestión de las áreas y actividades en las organizaciones, las cuales desean cambiar la forma tradicional de administrar sus procesos y aumentar su rentabilidad mediante la implementación de un sistema de información.

Por esta razón, en el presente trabajo se implementará un sistema de información web para la gestión de consultorios médicos particulares, con el fin de mejorar la experiencia para el paciente, facilitar la gestión del historial clínico y gestionar de forma eficiente diversos procesos dentro del área de salud.

1.2. Planteamiento del problema

Según el doctor Juan José Suárez Martínez profesor principal de medicina en la Universidad San Francisco de Quito, los hospitales cantonales, parroquiales o provinciales en Ecuador, están mal o insuficientemente equipados con relación a su nivel de complejidad (Martínez, 2019).

Las tecnologías de la información y comunicación se han convertido en herramientas eficientes para el manejo y mejora de los diversos procesos en el área de la salud, puesto que se encargan de gestionar agendas de citas, expedientes electrónicos, imagenología, atención a distancia,

entre otros procesos. Es por esto que diversos estudios han marcado la importancia de implementar este tipo de software en distintas áreas de la salud (Castañeda, 2019).

El no implementar software en el área de la salud dificulta la ejecución adecuada de diversos procesos, por ejemplo, la gestión de citas en donde siempre existen problemas y quejas por parte de los usuarios que necesitan de atención médica. Otro ejemplo es el manejo de la historia clínica de los pacientes, la cual, en gran parte de los casos, se gestiona de forma física y es necesario un proceso largo de análisis y búsqueda para atender a un paciente.

Otro aspecto clave en la carencia de software de eSalud, es que dificulta que los consultorios médicos escalen para atender a más pacientes ya que a medida que aumenta el número de pacientes recurrentes se complica el manejo de los datos e impide que los procesos administrativos y de los médicos se lleven a cabo de forma adecuada.

Para dar solución a esta problemática se plantea desarrollar un software de eSalud orientado a consultorios médicos particulares de diferentes especialidades, el cual será el encargado de gestionar citas, expedientes médicos, datos de pacientes y facturación.

En base al contexto anterior se puede identificar el siguiente problema principal:

- El no contar con un sistema de información de eSalud en los consultorios médicos particulares dificulta la gestión de procesos principales como el agendar citas, manejo de expedientes médicos, facturación, entre otros.

Y los siguientes problemas secundarios:

- La carencia de software complica al consultorio proveer sus servicios a una gran cantidad de pacientes
- Gran parte de consultorios médicos de diferentes especialidades en el país no cuentan con un sistema de información de eSalud
- Para los consultorios médicos es complicado encontrar un software de eSalud que se adapte a sus necesidades

1.3. Objetivos

1.3.1. Objetivo General

- Desarrollar un prototipo de sistema de información web para la gestión de consultorios médicos particulares de diferentes especialidades médicas

1.3.2. Objetivos específicos

- Analizar los requerimientos correspondientes del sistema de información
- Diseñar la arquitectura back-end y el modelo de datos del sistema de información
- Desarrollar el front-end y back-end del sistema de información
- Implementar el prototipo de sistema de información para la gestión de consultorios médicos

2. Fundamentos Teóricos

2.1. Aplicaciones de eSalud

La salud electrónica o eSalud hace referencia al uso de tecnologías de información para el mejoramiento de los sistemas de atención a la salud. La Organización Mundial de la Salud la define como "el uso coste-efectivo y seguro de las tecnologías de la información y comunicación en apoyo a salud y a los ámbitos relacionados con esta, incluyendo los servicios de atención sanitaria, vigilancia sanitaria, información en salud y educación, conocimiento e investigación en salud" (Castañeda, 2019). El avance tecnológico ha hecho que las TIC se desarrollen en diferentes campos, los sistemas web trabajan de forma íntegra con aplicaciones móviles, correo electrónico, redes sociales, etc. Lo que ha llevado a las TIC's a penetrar el campo de la salud mejorando sus procesos de atención, prevención y manejo de datos.

La evolución de los dispositivos inteligentes como: "smartphones" o "wearables" (tecnología vestible) ha hecho posible la prevención y monitoreo de enfermedades mediante el uso de aplicaciones de salud las cuales son capaces de informar las condiciones de un usuario. Por otro lado, los sistemas web de asistencia médica ayudan a mejorar la atención a pacientes en lugares remotos y, por medio de los datos almacenados del paciente, el sistema apoya a la toma de decisiones del médico.

2.1.1. Problemas de eSalud

Es claro que las tecnologías de la información y comunicación suponen una gran mejora para los procesos relacionados con la salud es por esto que organizaciones como la OMS, OPS, Cepal, entre otras promueven políticas para el desarrollo de aplicaciones de eSalud. Sin embargo, existen problemas, principalmente en América Latina, para la implementación de estos sistemas debido a las brechas digitales de los países en desarrollo y la falta de políticas que promuevan el desarrollo de sistemas que resuelvan problemas de salud en áreas específicas.

2.1.2. Componentes de las aplicaciones de eSalud

El avance tecnológico del internet, sistemas de almacenamiento, dispositivos inteligentes, inteligencia artificial ha permitido que los componentes que conforman los sistemas de eSalud aumenten para una mejor atención y administración de los recursos y procesos. Una aplicación de eSalud puede estar conformada por: sistemas administrativos electrónicos, expedientes

electrónicos, prescripción electrónica, sistemas de apoyo a la decisión clínica, telemedicina, imagenología, mSalud (salud por tecnología móvil), e-learinng, IoT, sistemas de big data o aprendizaje artificial.

2.2. Aplicaciones Web

Las aplicaciones web son sistemas a los cuales se puede acceder por medio de una conexión a internet y un dispositivo inteligente. Estos programas se alojan en servidores que están conectados a internet para que los usuarios puedan acceder desde cualquier lugar a cualquier hora. En la actualidad gran parte de las aplicaciones web se conforman de un front end, back end y una base de datos los cuales pueden estar alojados en diferentes servidores o en uno solo.

El funcionamiento de este tipo de aplicaciones se da cuando el usuario, por medio de un cliente, comienza a interactuar con la interfaz de usuario (UI o front-end) la cual, de ser necesario, se comunica mediante mensajes con el servidor de back-end el cual se encarga de realizar todas las funciones pesadas como: procesamiento de datos, búsqueda y recuperación de archivos, etc. A su vez los datos relevantes son extraídos y enviados de vuelta al servidor de back-end, desde la base de datos, para que este los procese y envíe de vuelta al cliente para su visualización.

2.3. Arquitectura de aplicaciones Web

A lo largo del tiempo se han presentado algunas arquitecturas para el desarrollo de aplicaciones web como el modelo punto a punto (P2P – Peer to Peer) o el modelo con servidor de aplicaciones. Sin embargo, el más usado es el modelo cliente-servidor, el cual se compone comúnmente por:

- Front-end: Conocido como interfaz de usuario (UI), es la parte visual de una aplicación web. Contiene los componentes necesarios para que el usuario interactúe con la aplicación. Actualmente se da una enorme importancia a la parte visual de las aplicaciones, de esto nace UX (experiencia de usuario) lo cual define la percepción de la aplicación.
- Back-end: Se encarga de recibir las peticiones que realiza el cliente y ejecutar las tareas pesadas como procesamiento masivo de datos o almacenamiento de archivos, básicamente se encarga de la lógica del negocio.
- Base de datos: Es el motor de base de datos usado para el almacenamiento de información. El back-end se comunica con la base de datos para extraer o almacenar datos.

Todos los componentes mencionados se comunican por medio del protocolo Http y trabajan en sincronía para componer una aplicación web. La arquitectura cliente servidor es la más usada y otorga ventajas como escalabilidad, fácil mantenimiento, seguridad, etc. Sin embargo, una desventaja determinante es el tráfico al servidor, es decir, la cantidad de clientes conectados simultáneamente a un servicio. Este problema se relaciona con el hardware que compone un servidor es por esto a medida que una aplicación crece es recomendable usar servidores robustos y potentes.

2.4. Servicios REST

Existen lenguajes de programación que destacan por su desempeño en ciertas tareas, por ejemplo, PHP es un lenguaje muy bueno para back-end en el lado del servidor, pero no es bueno para el desarrollo de interfaces, por otro lado, JavaScript destaca mucho del lado del cliente para el desarrollo de interfaces. Pero ¿cómo se comunican dos aplicaciones en lenguajes y ambientes diferentes?

API son las siglas de Application Programming Interface o Interfaz de Programación de Aplicaciones, un api es un conjunto de reglas que va a definir como se comunican dos aplicaciones por medio de http (protocolo de transferencia de hipertexto). Uno de los protocolos más usados en el desarrollo de API's es REST (Representational State Transfer o Transferencia de Estado Representacional), una arquitectura de software basada en la transferencia de recursos (datos). La información que se transmite y se recibe de una API REST puede ser en formato JSON o XML, actualmente gran parte de las aplicaciones usan el formato de texto sencillo JSON debido a su fácil entendimiento y velocidad.

Una API REST debe estar diseñada bajo los siguientes conceptos:

- **Recurso:** entidad que forma parte de la lógica de negocio y puede ser accedida por medio de la API
- **URI:** uniform resource identifier o identificador uniforme de recursos, los recursos Rest siempre se manipulan y acceden por medio de una URI.
- **Acción:** todas las peticiones a una API REST deben estar asociadas a uno de los verbos de HTTP:
    - GET para obtener un recurso
    - POST para escribir un recurso
    - PUT para modificar un recurso
    - DELETE para borrar un recurso

2.5. Frameworks de desarrollo web

A lo largo del tiempo han surgido diversos frameworks de desarrollo web en diversos lenguajes de programación: .NetCore con c#, NodeJS con JavaScript, Spring Boot con Java, Django con Python, entre otros. Sin embargo, uno de los frameworks más usados en el mundo por su rendimiento, versatilidad, integración con bibliotecas de terceros y documentación es Laravel con PHP. PHP es reconocido como uno de los lenguajes más robustos en el desarrollo web del lado del servidor por los diversos métodos que provee y su extensa documentación. De igual forma su administrador de paquetes Composer provee los estándares necesarios para administrar las dependencias de PHP de forma ágil y organizada.

Laravel es la solución para desarrollar de forma ágil y rápida una aplicación web basada en el modelo vista controlador ya que permite administrar las vistas con su sistema de “blade templates”, y los modelos y controladores con su ORM Eloquent (Laravel, 2021).

En la última década el desarrollo front-end sufrió una gran evolución gracias al lenguaje de programación JavaScript y sus frameworks de desarrollo como: React.js, Vue.js y Angular.js. JavaScript es reconocido por su versatilidad al momento de manipular el DOM (Document Object Model) y cuenta con su administrador de paquetes NPM el cual tiene una extensa comunidad y permite manejar las dependencias de forma ágil. Existen grupos de desarrolladores que debaten entre si React.js es un framework o una librería, su página web lo define como una biblioteca de JavaScript para construir interfaces de usuario, lo que hace destacar a React.js son sus vistas declarativas basadas en componentes (React.js, 2021).

2.6. Frameworks de Pruebas

A medida que se desarrolla una aplicación web esta se vuelve más extensa por lo que en algún momento el código puede fallar en el flujo de la aplicación, para evitar esto los desarrolladores deben probar constantemente el código para que no exista falla. Sin embargo, probar la aplicación cada vez que se realiza un cambio en el código puede considerarse agotador y una pérdida de tiempo. Para solucionar esto surgen los frameworks de pruebas los cuales cuentan con un conjunto de herramientas para la automatización de pruebas, y probar por separado el adecuado funcionamiento del código.

El testing es una rama del desarrollo que se dedica a la automatización de procesos, mediante la programación, con el objetivo de validar resultados en el código y verificar que no existan errores al modificar la estructura de la aplicación. Existen diversos frameworks de pruebas como PHPUnit el cual permite la creación de pruebas unitarias para evaluar toda la estructura en un proyecto de Laravel, es decir, permite comprobar si un resultado se insertó exitosamente en la base de datos, si una vista retorna un resultado esperado o si un controlador realiza la acción deseada. Este framework viene dentro de la suite de herramientas de Laravel.

Otro framework conocido que forma parte del testing de front-end es Jest el cual, mediante el uso de código en Javascript, permite diseñar de forma rápida pruebas que pueden determinar si un componente de React hizo render correctamente o si una petición a una api se ejecutó de manera adecuada.

2.7. Metodologías ágiles de desarrollo de software

Para entender de una mejor manera las metodologías ágiles de desarrollo es necesario aclarar algunos aspectos de las metodologías tradicionales.

Las metodologías tradicionales buscaban como resultado el desarrollo de un proyecto con grandes dimensiones mediante la implementación de un modelo rígido, poco flexible y que buscaba tener una disciplina estricta durante el proceso de desarrollo de software. Debido a estos factores este tipo de

metodologías se centraban en la elicitación de requerimientos y en una rigurosa planificación para tener un modelo predecible y eficiente. Sin embargo, este no era el caso en los proyectos de pequeña y mediana escala ya que se debía seguir muchas actividades y esto retrasaba el desarrollo de software. Ante este problema surgen las metodologías ágiles de desarrollo en la década de los noventa las cuales buscaban reducir la probabilidad de fracaso por subestimación de costos, tiempos y funcionalidades en los proyectos de desarrollo de software (Navarro.A, 2013).

A diferencia de las metodologías tradicionales las ágiles tienen más flexibilidad ya que pueden ser modificadas para ajustarse a las necesidades de cada equipo. De igual forma este tipo de metodologías permite regresar a las actividades iniciales del proceso de desarrollo lo que supone una ventaja ya que los proyectos se vuelven adaptables a cambios. Otro aspecto importante de las metodologías ágiles es la constante comunicación con el cliente, de esta forma existe una constante retroalimentación de lo que desea el cliente en el proyecto y se pueden desarrollar entregables como prototipos para dar una idea clara de lo que se espera en la aplicación.

Ambos tipos de metodologías tienen sus puntos fuertes lo importante es determinar ciertos factores para escoger de manera correcta la metodología con la que se va a desarrollar un proyecto, por ejemplo: los costos del proyecto, cuan involucrado está el cliente, la probabilidad de que se den cambios constantes a lo largo del proyecto, que tan familiar es el problema a resolver, etc.

2.8. Metodología de desarrollo eXtreme Programming (XP)

Extreme programming es una metodología ágil de desarrollo de software creada por Kent Beck en 1996, XP proporciona un modelo flexible, ligero y de bajo riesgo para manejar de manera adecuada requerimientos que cambian rápidamente (Anwer.F, 2017). Dentro de esta metodología existen 4 variables importantes para todo proyecto de software: costo, tiempo, calidad y alcance, de las cuales 3 serán fijadas por actores externos y la variable restante puede ser fijada por el grupo de desarrolladores. XP propone un ciclo de vida dinámico el cual admite que los clientes no son capaces de especificar sus requerimientos al inicio de un proyecto, es por esto que XP se realiza en ciclos de desarrollo cortos (iteraciones) que están compuestos por un análisis, diseño, desarrollo y pruebas. (Joskowicz, 2008). Esta metodología se pude separar en las siguientes fases:

2.8.1. Fase de Exploración

Dentro de esta fase se define el alcance general del proyecto, en la cual el cliente dialoga con los desarrolladores los diferentes requerimientos del sistema y se generan las “historias de usuario”. Estas historias son importantes ya que sirven para estimar los tiempos de desarrollo. El resultado final de esta fase es una visión general del sistema, y un plazo total estimado (Joskowicz, 2008).

2.8.2. Fase de planificación

Esta fase consiste en una o varias reuniones grupales de planificación en las que el cliente, gerentes y el grupo de desarrollo acuerdan el orden en el cual se deberá implementar las historias de usuario para obtener como resultado un plan de entregas (Joskowicz, 2008).

2.8.3. Fase de Iteraciones

Esta fase es la más importante de la metodología ya que se desarrollan las funcionalidades del sistema usando las historias de usuario. Al final de cada iteración se genera un entregable en base a los requerimientos de las historias de usuario y se mide el avance del proyecto. Cabe recalcar que el cliente debe participar activamente en esta fase del ciclo ya que en algunos casos las historias de usuario no son sus suficientes para aclarar adecuadamente los requerimientos. Dentro de esta fase del ciclo se dan tres aspectos importantes para un correcto desarrollo (Joskowicz, 2008):

2.8.3.1. Diseño

Los diseños deben ser simples y claros usando los siguientes conceptos:

- **Simplicidad**: XP sugiere implementar un diseño simple que se pueda desarrollar rápidamente
- **Soluciones *Spike***: programas pequeños que se usan para probar o evaluar una solución
- **Recodificación**: desarrolla nuevamente una parte del código para hacerlo más simple y entendible
- **Metáforas**: usar el concepto de metáfora para explicar el propósito del proyecto y su estructura

2.8.3.2. Codificación

Existen factores que pueden hacer de la codificación un proceso organizado y ágil, por ejemplo:

- **Uso de estándares**: aplicación de estándares de programación
- **Programación dirigida a pruebas (TDD-Test Driven Development)**: codificación de pruebas que evalúen el código desarrollado
- **Programación en pares**: dos desarrolladores trabajan en conjunto sobre una misma estructura de código
- **Trabajo sobre versiones actualizadas**: es importante que todos los miembros tengan las versiones actualizadas del proyecto
- **Propiedad colectiva de código**: todo el equipo contribuye con ideas del proyecto, corrección de errores o recodificación

2.8.3.3. Pruebas

Es posible desarrollar el siguiente tipo de pruebas:

- **Pruebas unitarias**: mediante programación se desarrollan pruebas que evalúen la estructura del proyecto por partes
- **Detección y corrección de errores**: al encontrar un bug este debe ser corregido inmediatamente y registrado
- **Pruebas de aceptación**: el cliente especifica un escenario y comprueba la implementación correcta de una prueba de usuario

2.8.4. Fase de puesta en producción

Una vez desarrolladas todas las funcionalidades del sistema este pasa a producción y se libera a la web. Dentro de esta fase se pueden dar tareas de ajuste ("fine tuning") del sistema.

3. Herramientas para el desarrollo del sistema

3.1. Editor de código

El editor permite modificar la estructura del código dentro un sistema, debido a esto es fundamental el uso de un editor que facilite el flujo de trabajo y permite agilizar el proceso de desarrollo tanto individual como en conjunto. Dentro del mundo del desarrollo existen diversos editores de código como: SublimeText, Atom, Brackets, Vim, etc. Sin embargo, hay uno que destaca por su gran flexibilidad y fácil uso de extensiones que ayudan al desarrollo de código, este es Visual Studio Code (VSCode). Visual Studio Code fue seleccionado en las encuestas a desarrolladores, echa por Stackoverflow en 2019, como el editor de código más usado por los desarrolladores web, ya que el 50.7% de la comunidad lo utiliza (Stackoverflow, 2019).

VSCode cuenta con un administrador de extensiones bastante amplio el cual permite la instalación de intellisense para cada lenguaje de programación, de igual forma estas extensiones son personalizables y extensibles permitiendo al desarrollador integrar herramientas como debugging, integración con git, integración con servicios de Devops, etc.

3.2. Base de datos relacional

Este tipo de base de datos se encargan de almacenar toda la información relevante de un sistema siguiendo un modelo relacional en donde cada tabla del motor de datos representa una entidad referente a la lógica de negocios del sistema y cada entidad puede, o no, estar relacionada con otra mediante una clave primaria. Existen diversos sistemas gestores de base de datos que mantienen una integridad referencial como: MySQL, MariaDB, Oracle, DB2, SQLServer, PostgreSQL, etc.

PostgresSQL es un potente sistema de base de datos el cual se ha ganado una reputación por su fiabilidad, solidez de funciones y rendimiento, cuenta con más de 30 años de desarrollo y es de código abierto lo que facilita su uso a la comunidad de desarrolladores (Postgresql, 2021).

3.3. Backend

El backend se encarga de recibir las peticiones del cliente y ejecutar una acción como: inserción de un nuevo registro, búsqueda de un grupo de datos, manejo de archivos, etc. Y enviar esta información procesada de vuelta al cliente. Existen diversos lenguajes de programación y frameworks que ayudan al desarrollo de backend: Django con Python, Node.js con JavaScript, .NetCore con C#, Laravel con PHP, entre otros.

3.3.1. Laravel

Laravel es un framework progresivo y escalable para el desarrollo de aplicaciones web completas y modernas. Laravel es considerado progresivo ya que crece con las necesidades del negocio, es decir, funciona tanto para páginas web pequeñas como para páginas web empresariales ya que su documentación y guías son completas y claras. Laravel brinda herramientas sólidas para la inyección de dependencias, pruebas unitarias, colas, eventos en tiempo real, etc. (Laravel, 2021)

3.4. Frontend

Es la parte visual dentro de un navegador web con la cual el usuario interactúa para obtener una respuesta. En la última década el diseño de interfaces y experiencia de usuario (UI y UX) han tomado gran importancia dando como resultado diversos frameworks para el desarrollo ágil de frontend como: React.js, Vue.js, Angular.js, entre otros.

3.4.1. React.JS

React.js es una librería de código abierto para la construcción de interfaces la cual cuenta con un sistema declarativo que permite un desarrollo sencillo. React se encarga de actualizar y renderizar los componentes si los datos en una página cambian, de igual forma cuenta con un sistema basado en componentes lo que permite diseñar piezas de código encapsuladas para usarse después en interfaces de usuario complejas. React cuenta con un ciclo de vida dentro de cada componente lo que permite manejar de forma ágil las peticiones a las APIs de forma asíncrona (React.js, 2021).

3.4.2. NPM

NPM es el administrador de paquetes de Node.js el cual permite manejar las dependencias de cualquier proyecto que utilice JavaScript. Este gestor posee el registro de software más grande del mundo y posee una extensa comunidad lo que permite el uso libre de paquetes de software que tienen un gran soporte por detrás, como es React.js (NPM, 2021).

3.4.3. Jest

Jest es un framework orientado a pruebas de JavaScript el cual permite crear piezas de código para evaluar las funcionalidades del frontend. Jest ya tiene gran parte de sus funciones configuradas lo que permite agilizar las pruebas y correrlas en paralelo. De igual forma permite la creación de snapshots y un informe completo y detallado en HTML de la ejecución de las pruebas (Jest, 2021).

Tabla 1 Descripción de herramientas tecnológicas

HERRAMIENTA TECNOLÓGICA	VERSIÓN	DESCRIPCIÓN
PostgreSQL	Pgsql 12	Base de datos relacional de código abierto cuyo lanzamiento inicial fue en 1996. Se maneja bajo una licencia PostgreSQL y cuenta con 30 años de desarrollo activo apoyado por la comunidad (Postgresql, 2021).
Laravel	Laravel 8.x	Framework para el desarrollo de aplicaciones y servicios web mediante el uso de PHP. Su lanzamiento fue en 2011 y provee diversas herramientas para agilizar el desarrollo web (Laravel, 2021).
React JS	React 17.0.1	Librería de JavaScript de código abierto la cual fue lanzada en 2013 por Facebook. React ayuda a la creación de interfaces de usuario de forma sencilla y basada en componentes (React.js, 2021).
Npm	Npm 6.14.9	Npm es el gestor de paquetes de Node JS para el desarrollo de software con JavaScript. Cuenta con diversos paquetes y un gran apoyo de la comunidad (NPM, 2021).
Jest	Jest 26.6.3	Jest es un framework de pruebas que se enfoca en proyectos con TypeScript, Node, React, Vue, etc.

		Provee métodos simples para el desarrollo de pruebas unitarias (Jest, 2021).
Next JS	Next 10.0.9	Framework para el desarrollo de aplicaciones web utilizando React JS. Next fue lanzado por la empresa Vercel en 2016 y cuenta con funciones ya integradas como refresco rápido y renderizado por servidor (Vercel, 2021).

Tabla 2 Descripción de metodología de desarrollo

METODOLOGÍA USADA PARA EL DESARROLLO DE LA APLICACIÓN	DESCRIPCIÓN
Extreme Programming	Es un marco de desarrollo de software ágil que tiene como objetivo producir software de mayor calidad para el equipo de desarrollo. XP es el más específico de los marcos ágiles con respecto a las prácticas de ingeniería adecuadas para el desarrollo de software. Consta de cuatro fases: fase de exploración, fase de planificación, fase de iteraciones, fase de puesta en producción.

4. Desarrollo de la aplicación mediante la metodología Extreme Programming (XP)

En esta sección se desarrolla el sistema web utilizando la metodología ágil extreme programming la cual fue seleccionada ya que el sistema cuenta con una sola persona para su desarrollo, en un tiempo corto y es necesario pedir constantemente la opinión de médicos para comprobar las funcionalidades del sistema. Esta metodología ágil evita costos innecesarios por documentación y se enfoca en aplicar un ciclo de vida dinámico en el desarrollo de software definiendo los requerimientos del cliente desde el inicio del proyecto. También se entregan prototipos al cliente para conocer el nivel de satisfacción y la necesidad de cambiar las funcionalidades. Debido a estos aspectos XP es una metodología que favorece el desarrollo óptimo de este sistema.

4.1. Fase de exploración

Para realizar la elicitación de requerimientos se desarrollaron diversos diálogos con diferentes profesionales en el campo de la medicina, de esta forma se extrajo las funcionalidades del sistema en forma de historias de usuario las cuales brindan una idea general del alcance del sistema. Las historias de usuario, para la metodología XP, se elaboraron en forma de tarjeta para complementar el proceso de desarrollo de software con un tablero Kanban. Estas tarjetas describen los elementos más importantes del dialogo con los “skateholders” y contienen las funcionalidades del sistema, seguido de una descripción más detallada de cómo desarrollar dichas funcionalidades.

La elicitación de requerimientos se realizó con la Doctora Lizbeth Aguilar Pérez (dentista), el doctor Christian Rojas (cirujano) y la Doctora Karina Pérez Vega (dentista). A continuación, se lista las diferentes historias de usuario extraídas de las reuniones:

Tabla 3 Historia de usuario #01

#01	Como **<<médico cliente>>** quiero **<<los datos de mis pacientes>>** para poder **<<acceder a sus historias clínicas e información>>**	
CRITERIOS DE ACEPTACIÓN		
1.	**<<Observar Información>>**	En caso de que **<<busque un paciente>>** y adicionalmente **<<desee ver su información>>**, cuando **<<ingreso su DNI>>**, el sistema **<<obtendrá la información correspondiente al paciente ingresado>>**
2.	**<<Observar Historias Clínicas>>**	En caso de que **<<acceda a un paciente>>**, cuando **<<busque su historial>>**, el sistema **<<debe mostrar**

		su historia clínica con toda su información correspondiente>>
3.	**<<Agregar información a las historias clínicas>>**	En caso de que **<<llegue un paciente>>**, cuando **<<se le realice una consulta, tratamiento o diagnóstico>>**, el sistema **<<debe permitir agregar un nuevo diagnóstico a la historia clínica del paciente>**
4.	**<<Modificar la información de historias clínicas y evoluciones>>**	En caso de que **<<el médico requiera modificar la información de historias o evoluciones>>**, el sistema **<<debe facilitar la edición o eliminación de datos dentro de historias clínicas o evoluciones>>**

Tabla 4 Historia de usuario #2

#02	Como **<<médico cliente>>** quiero **<<que exista un calendario organizado de citas>>** para poder **<<observar en que días y a qué horas hay una cita marcada en el calendario>>**
CRITERIOS DE ACEPTACIÓN	
1.	**<<Administración de citas>>**	En caso de que **<<quiera ver las citas de un día específico>>**, cuando **<<ingrese al módulo de citas>>**, el sistema **<<debe mostrar de forma organizada las citas de un día con sus respectivas horas>>**
2	**<<Creación y eliminación de citas>>**	En caso de que **<<un paciente desee agendar una cita en administración>>**, cuando **<< el personal administrativo ingrese al módulo se citas>>**, el sistema **<<debe permitir agendar una nueva cita o removerla del calendario>>**

Tabla 5 Historia de usuario #3

#03	Como **<<médico cliente>>** quiero **<<un recordatorio en las citas>>** para poder **<<informar tanto a pacientes como médicos una cita agendada>>**
CRITERIOS DE ACEPTACIÓN	
1.	**<<Notificaciones de citas>>**	En caso de que **<<tenga una cita>>**, cuando **<<esta cita esté próxima>>**, el sistema **<<debe enviar un correo o notificación al paciente y al médico>>**

Tabla 6 Historia de usuario #4

#04	Como **<<médico cliente>>** quiero **<<un listado de los tratamientos y materiales en el consultorio con su respectivo precio>>** para poder **<<informar a los pacientes los precios e información de los tratamientos, y conocer como médico los materiales disponibles en el consultorio>>**	
CRITERIOS DE ACEPTACIÓN		
1.	**<<Información de Tratamientos>>**	En caso de que **<<un paciente quiera conocer el precio de un tratamiento>>**, cuando **<<pregunte su precio en administración>>**, el sistema **<<deberá mostrar un listado de los tratamientos con sus precios respectivos>>**
2.	**<<Información de materiales>>**	En caso de que **<<el personal desee conocer los materiales disponibles para cualquier tratamiento>>**, el sistema **<<deberá mostrar un listado de los materiales con la cantidad y precio respectivos>>**

Tabla 7 Historia de usuario #5

#05	Como **<<médico cliente>>** quiero **<<una página para llenar un consentimiento informado>>** para poder **<<tener este documento y hacerlo firmar al paciente en caso de un tratamiento de riesgo>>**	
CRITERIOS DE ACEPTACIÓN		
1.	**<<Formulario de consentimiento>>**	En caso de que **<<se deba realizar un tratamiento riesgoso>>**, cuando **<<se acceda al formulario de consentimiento>>**, el sistema **<<debe presentar dicho formulario para ser llenado con los datos del paciente y el médico, y debe permitir la impresión del documento>>**

Tabla 8 Historia de usuario #6

#06	Como **<<médico cliente>>** quiero **<<una página publicitaria>>** para poder **<<aumentar el número de pacientes que llegan al consultorio>>**
CRITERIOS DE ACEPTACIÓN	

1.	**<<Páginas de publicidad>>**	El sistema **<<debe contar con una página inicial que describa al consultorio y otra para presentar los servicios que proporciona el consultorio>>**

Tabla 9 Historia de usuario #7

#07	Como **<<medico cliente>>** quiero **<<que cada especialista dentro del consultorio tenga acceso al sistema>>** para poder **<<visualizar la información del paciente>>**	
CRITERIOS DE ACEPTACIÓN		
1.	**<<Diseño de un Login>>**	El sistema **<<debe organizar los usuarios que ingresan para tener un acceso ético a la información de los pacientes>>**
2.	**<<Alta seguridad en el acceso a los datos>>**	El sistema **<<debe contar con la seguridad apropiada para proteger los historiales clínicos y datos de los pacientes>>**

En base a las historias de usuario extraídas se identifica 7 funcionalidades las cuales serán accedidas mediante la autenticación y autorización ética de usuarios. En el dialogo se identificó que todos los médicos recalcaban la importancia de la seguridad en el sistema y un acceso a la información ética ya que el sistema contiene datos sensibles de los pacientes. En base a los requerimientos obtenidos se estima que el sistema será desarrollado en el plazo de 80 días.

4.2 Fase de planificación

Para la planificación y proceso de desarrollo de software se utiliza la herramienta Jira de Attlassian (Atlassian, 2021) la cual supone una gran ayuda para el desarrollo ágil de un sistema ya que cuenta con herramientas como: tablero Kanban, calendario para la planificación, notas en cuanto a bugs o problemas, integración con GitHub, etc.

La implementación de las historias de usuario se organizó según su nivel de complejidad:

1. **Historia de Usuario #1**: Implica la implementación de la gestión de pacientes junto la gestión de historias clínicas y evoluciones
2. **Historia de Usuario #7:** Implementar la autenticación y autorización de usuarios con un sistema seguro de manejo de sesiones.
3. **Historia de Usuario #2**: Elaboración de un calendario personalizado que permite organizar las citas de forma adecuada
4. **Historia de Usuario #3**: Notificar por correo o SMS el horario de una cita agendada al paciente y médico

5. **Historia de Usuario #6:** Elaborar un home page publicitario y una página que detalle los servicios del consultorio
6. **Historia de Usuario #4:** Desarrollar un listado de los materiales y tratamientos que tiene el consultorio para brindar información al personal
7. **Historia de usuario #5:** Elaborar un formulario de consentimiento formado en caso de darse un tratamiento riesgoso.

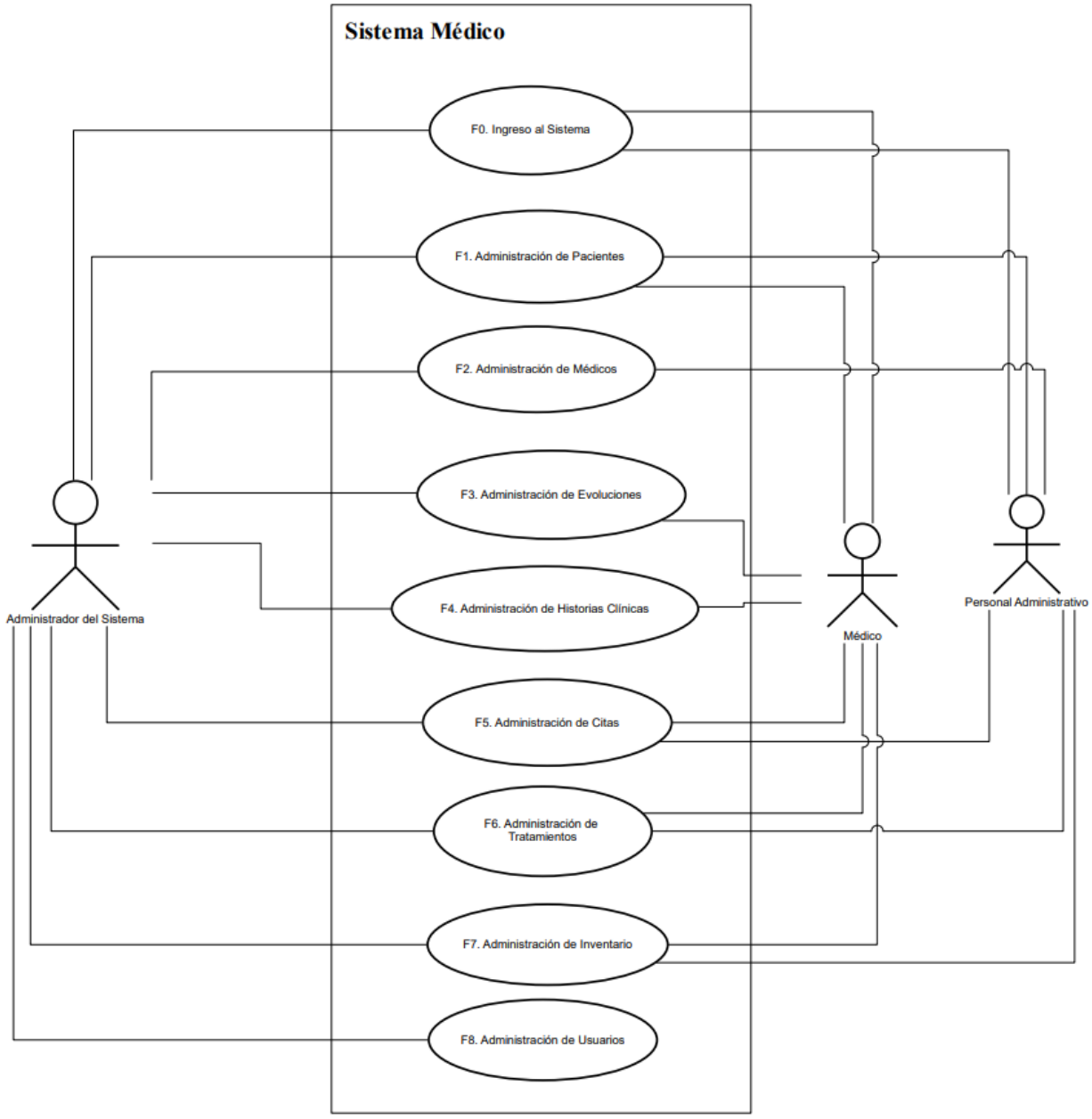
Ilustración 1 Planificación para el desarrollo del sistema

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Historia de Usuario #1	14 days	Wed 17/3/21	Sun 4/4/21		
2		Historia de Usuario #7	6 days	Mon 5/4/21	Sun 11/4/21	1	
3		Historia de Usuario #2	6 days	Mon 12/4/21	Sun 18/4/21	2	
4		Historia de Usuario #3	6 days	Mon 19/4/21	Sun 25/4/21	3	
5		Historia de Usuario #6	6 days	Mon 26/4/21	Sun 2/5/21	4	
6		Historia de Usuario #4	6 days	Mon 3/5/21	Sun 9/5/21	5	
7		Historia de Usuario #5	6 days	Mon 10/5/21	Sun 16/5/21	6	

4.3 Fase de Iteraciones

Diagrama general de casos de uso

Ilustración 2 Diagrama general de casos de uso

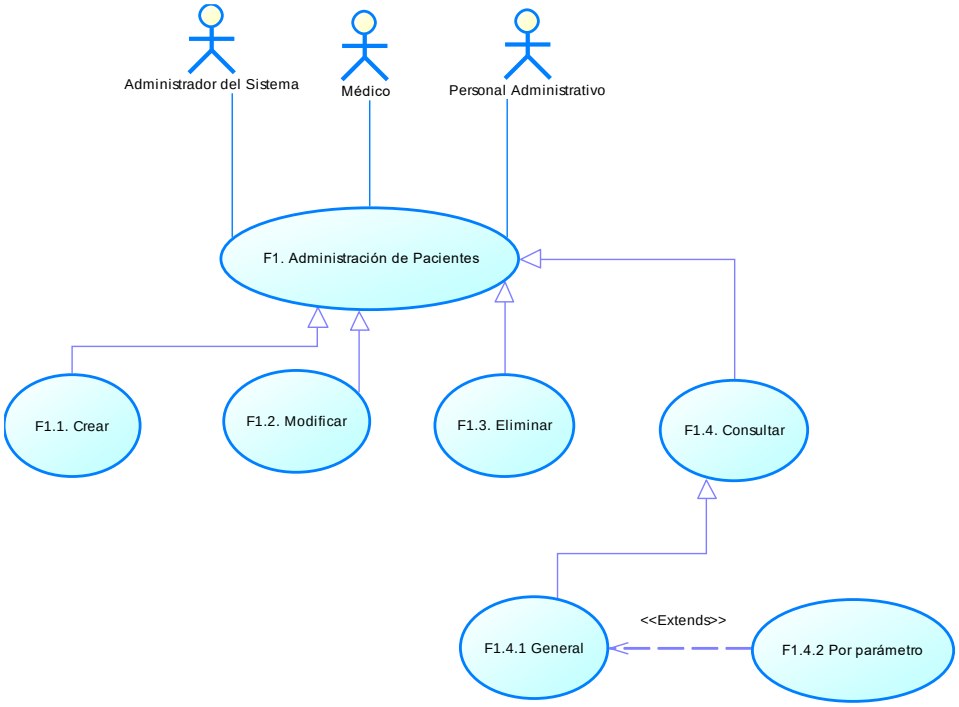


4.3.1 Iteración #1 – Historia de Usuario #1

4.3.1.1 Diseño

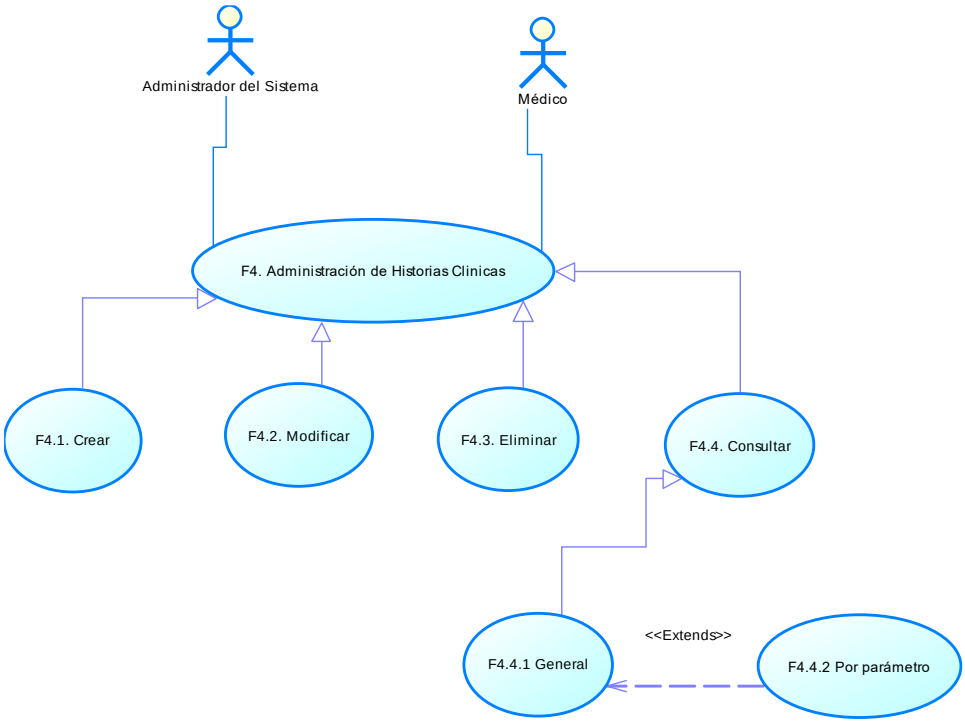
Casos de uso Pacientes

Ilustración 3 Casos de uso pacientes



Casos de uso Historias Clínicas

Ilustración 4 Casos de uso historias clínicas



Casos de uso Evoluciones

Ilustración 5 Casos de uso evoluciones

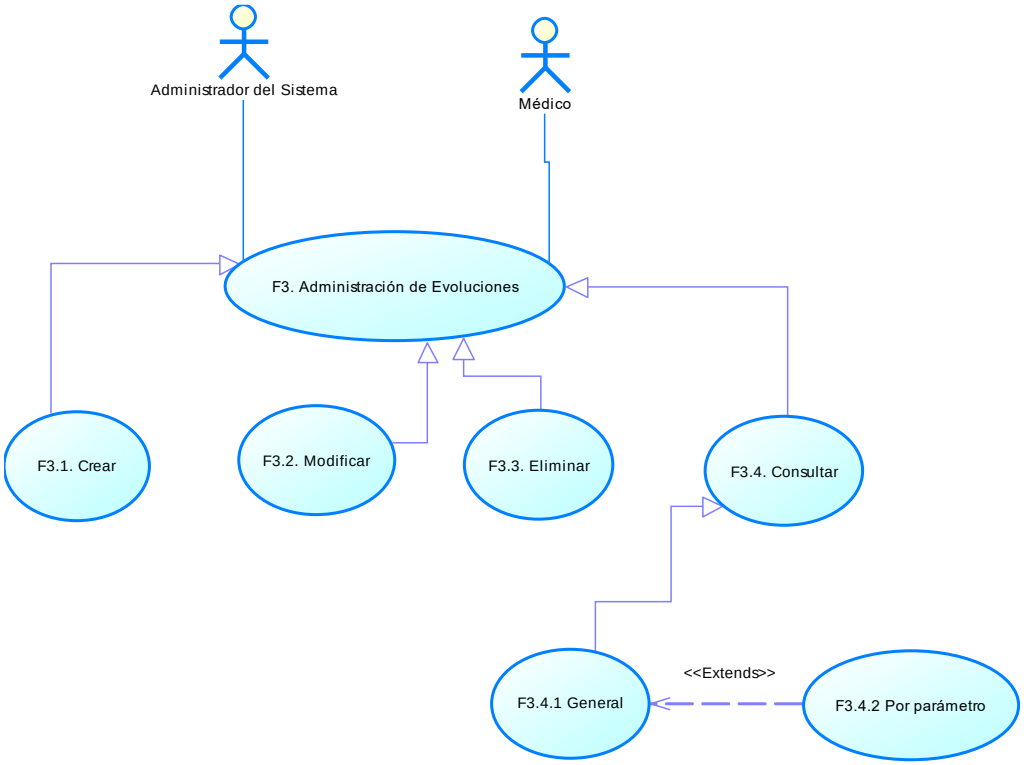
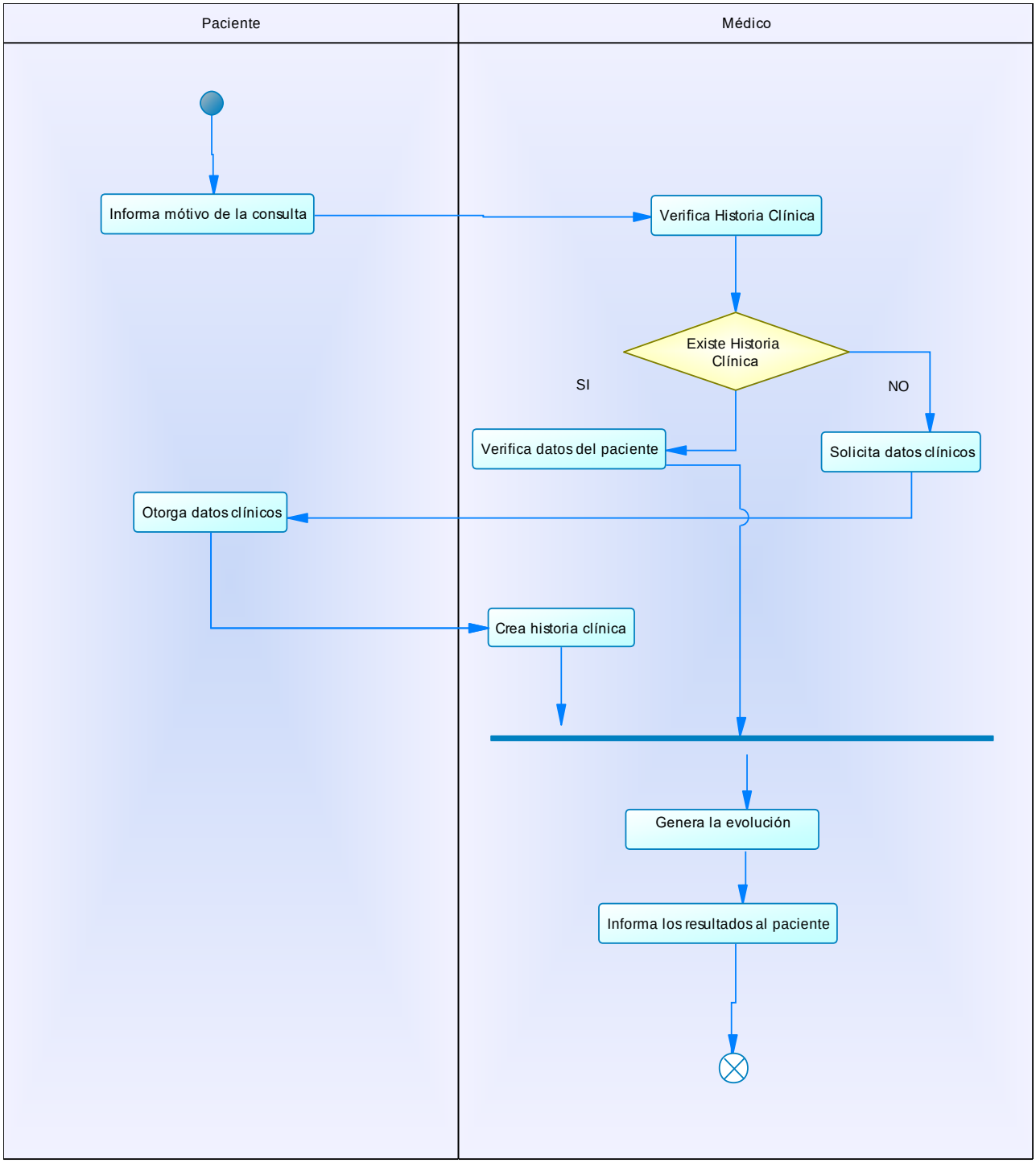


Diagrama de actividades Administración de Evoluciones

Ilustración 6 Diagrama de actividades - Administración de evoluciones

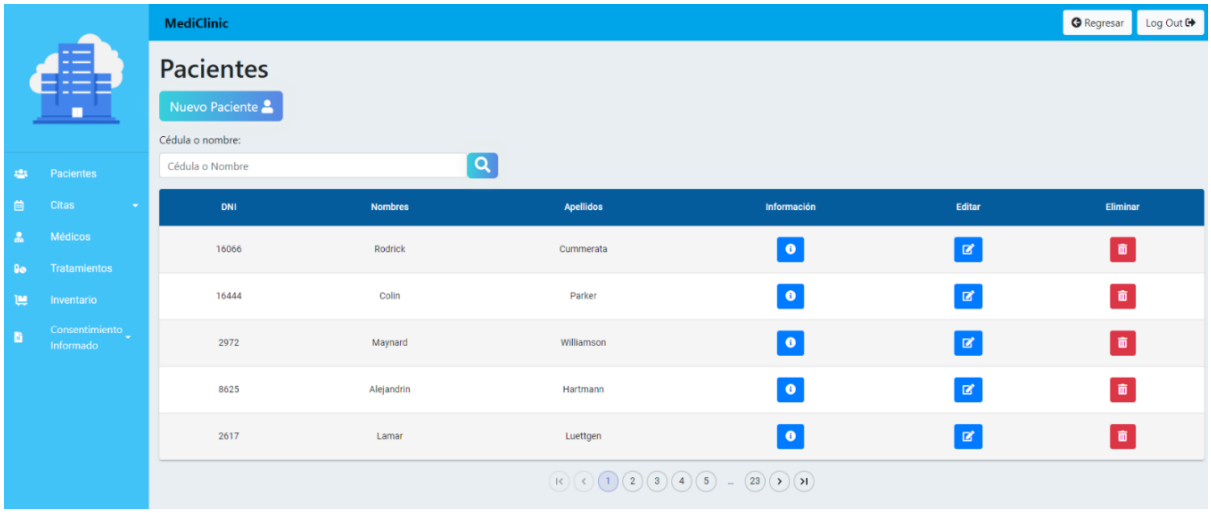


Vistas de la Historia de Usuario #1

Al ingresar al sistema, ya sea con un usuario administrador, médico o administrativo, se presenta una pantalla con una tabla que contiene los datos más relevantes de los pacientes y permite realizar diversas acciones como: la edición de un paciente, la eliminación de un paciente, ver información más detallada de un paciente o buscar un paciente en específico mediante la cédula o nombre.

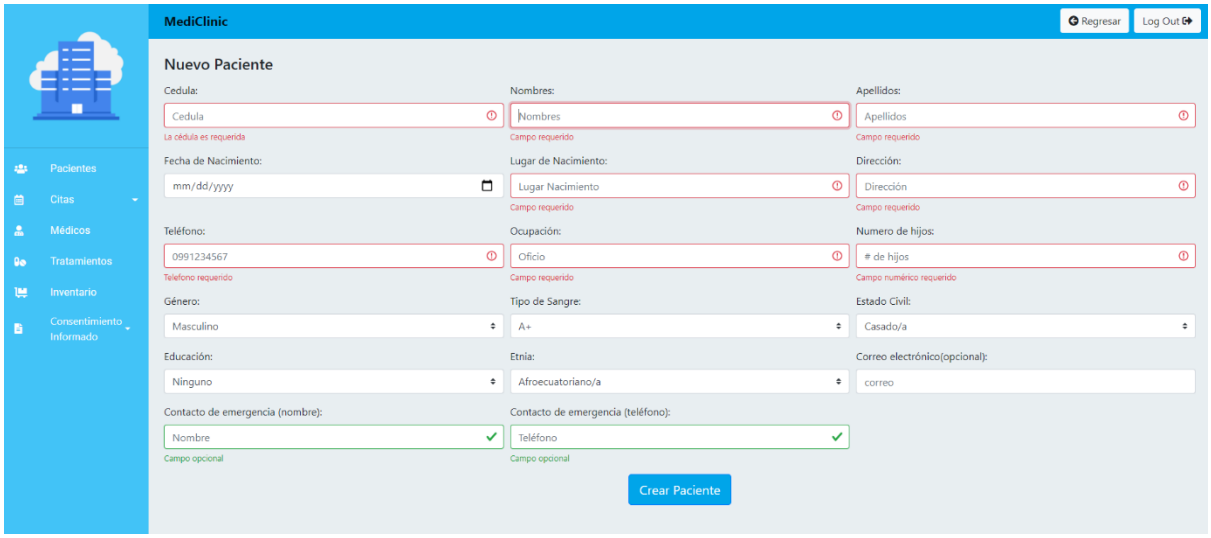
Cabe recalcar que tanto usuarios administradores del sistema, médicos y administrativos (personal administrativo) tienen permisos para crear o mutar la información de un paciente.

Ilustración 7 Página con tabla de Pacientes



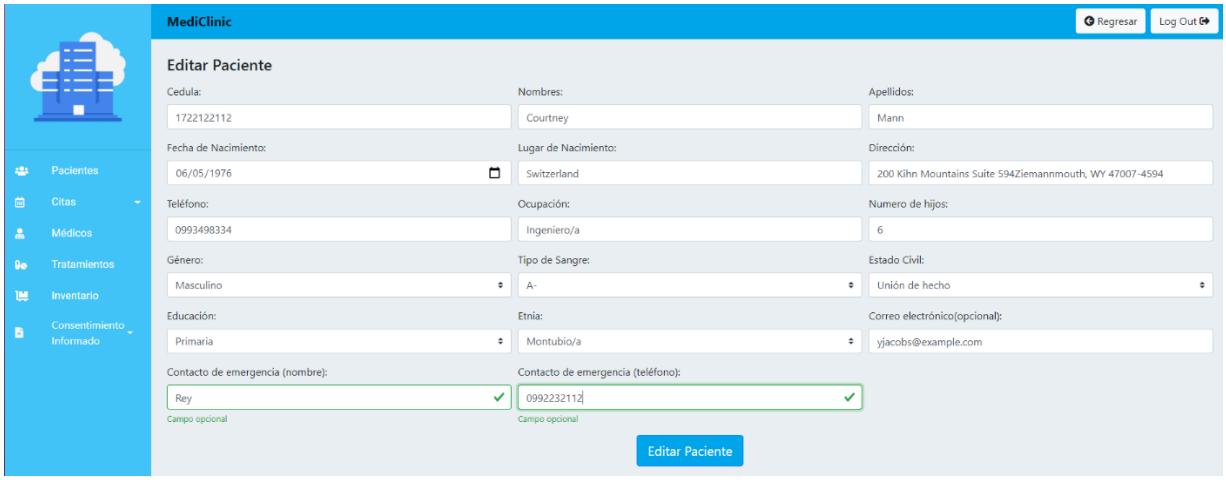
Al presionar en el botón de "Nuevo Paciente" el sistema redirige al usuario hacia un formulario con validaciones para la creación de un nuevo paciente.

Ilustración 8 Página para la creación de un paciente



En caso de presionar el botón de editar en la tabla, el sistema redirige al usuario a una nueva página la cual contiene los datos del paciente listos para editar con las respectivas validaciones.

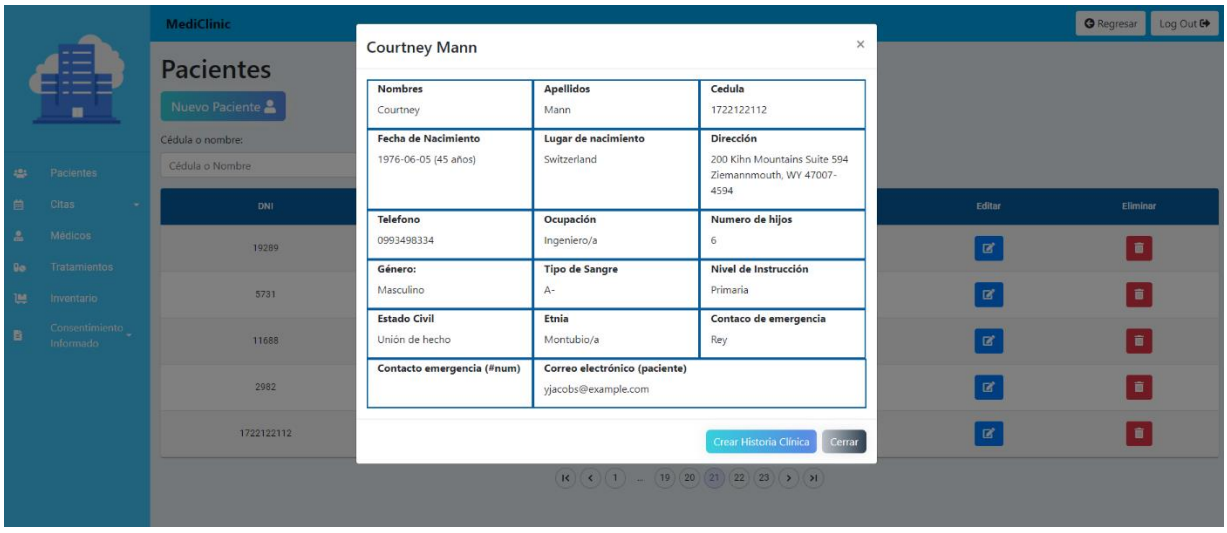
Ilustración 9 Página para editar datos de un paciente



Al presionar el botón de información dentro de la tabla el sistema despliega un modal que presenta a detalle todos los datos del paciente. En caso de no existir una historia clínica asociada al paciente el sistema presenta un botón "Crear Historia Clínica" el cual redirige al usuario hacia la página de creación de historia clínica, caso contrario se presenta un botón de "Historial Clínico" que redirige al usuario a una página con todos los datos detallados de la historia del paciente asociado.

Cabe recalcar que solamente los usuarios de tipo administrador del sistema y médicos pueden visualizar el botón. Los usuarios del personal administrativo no pueden visualizarlo ya que no tienen los permisos para observar ni mutar información sensible de las historias clínicas de los pacientes.

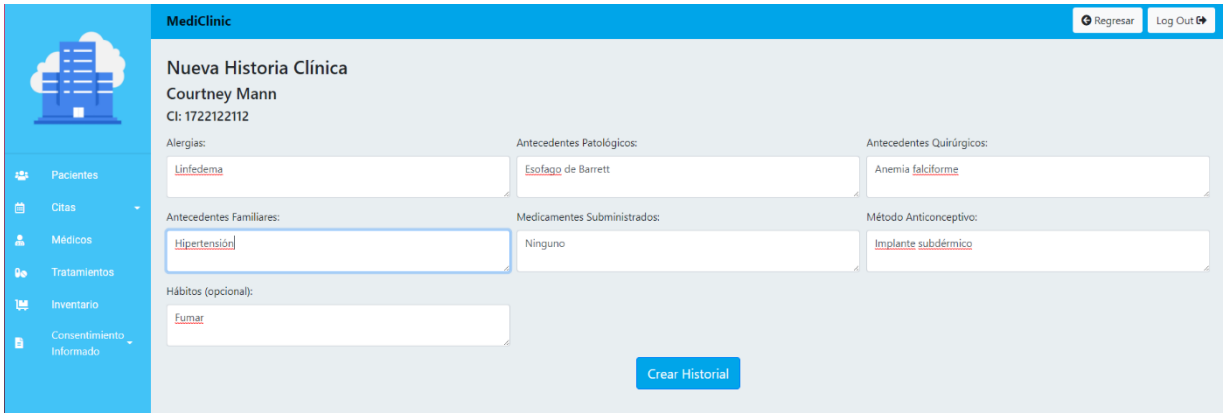
Ilustración 10 Modal que muestra los datos de un paciente



En caso de no existir una historia clínica y presionar el botón de "Crear Historia Clínica", el sistema redirige al usuario a un formulario para llenar las alergias y antecedentes de un paciente.

Solamente médicos y administradores del sistema pueden acceder a la página de creación de historia clínica, en caso de no contar con los permisos necesarios se redirecciona de inmediato al usuario a la página de pacientes.

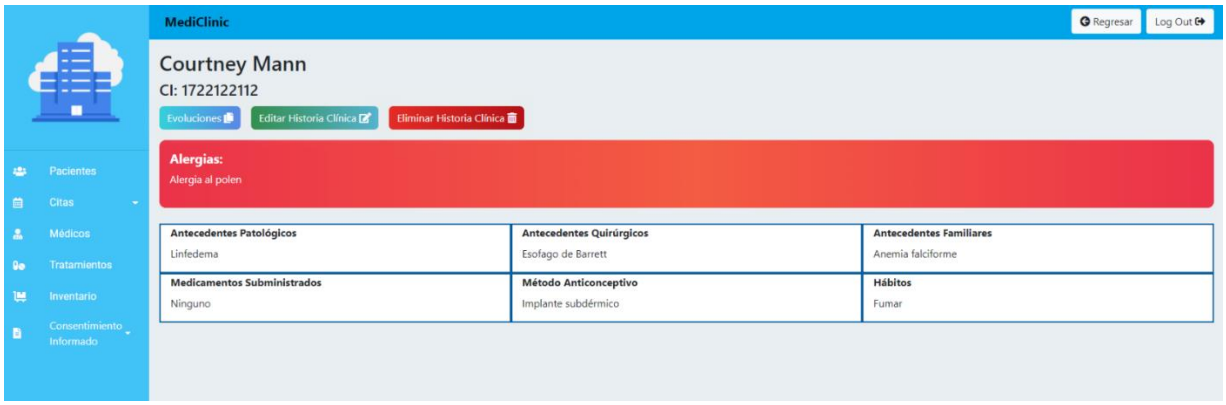
Ilustración 11 Página de creación para el Historial Clínico de un paciente



En caso de existir una historia clínica asociada y presionar el botón de "Historial Clínico", el sistema redirige al usuario a una página con todos los detalles de la historia clínica.

Si el usuario no es médico ni administrador del sistema, se redirige de inmediato a la página de pacientes.

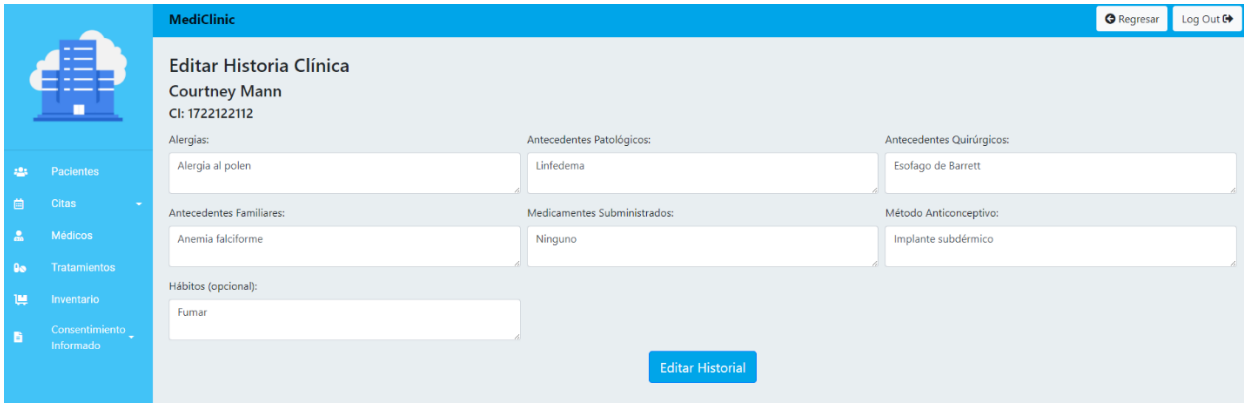
Ilustración 12 Página que muestra los datos del Historial Clínico de un paciente



Dentro de la página que detalla la información de una historia clínica, si se presiona el botón de editar historia el sistema redirige al usuario a una página con los datos de la historia clínica asociada al paciente, listos para ser editados.

Si el usuario no es médico ni administrador del sistema, se redirige de inmediato a la página de pacientes.

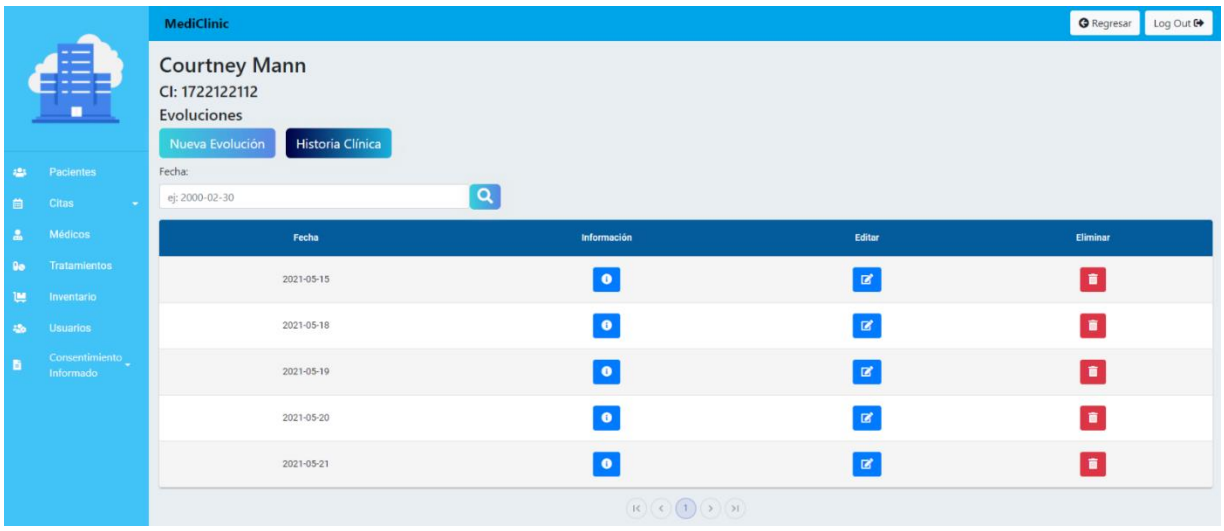
Ilustración 13 Página para la edición de una historia clínica



Mediante la página de historia clínica es posible acceder a las evoluciones del paciente presionando el botón “Evoluciones”, lo cual redirige al usuario a una tabla la cual permite realizar diversas acciones como: crear una nueva evolución, observar la información detallada de una evolución, editar o eliminar una evolución en específico y buscar evoluciones según su fecha.

Al igual que la historia clínica, la información de las evoluciones no pueden ser accedidas por usuarios que no sean médicos o administradores del sistema, si no se tienen permisos se redirige de inmediato a la página de pacientes.

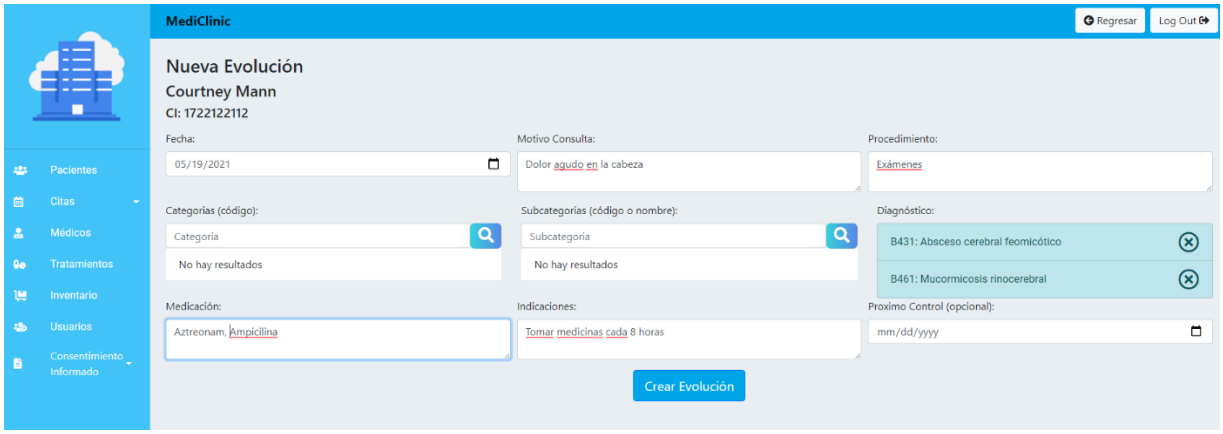
Ilustración 14 Página con tabla de Evoluciones



Para la creación de una nueva evolución se maneja **la clasificación internacional de enfermedades CIE-10**, el médico que genera la evolución puede buscar enfermedades ingresando el código o nombre dentro de una subcategoría o categoría de esta forma se presentan las enfermedades con sus códigos las cuales pueden ser agregadas a la evolución generada. También se debe indicar la fecha de la evolución, indicaciones, medicamentos, etc.

Si un usuario no es administrador del sistema ni médico se lo redirecciona a la página de pacientes.

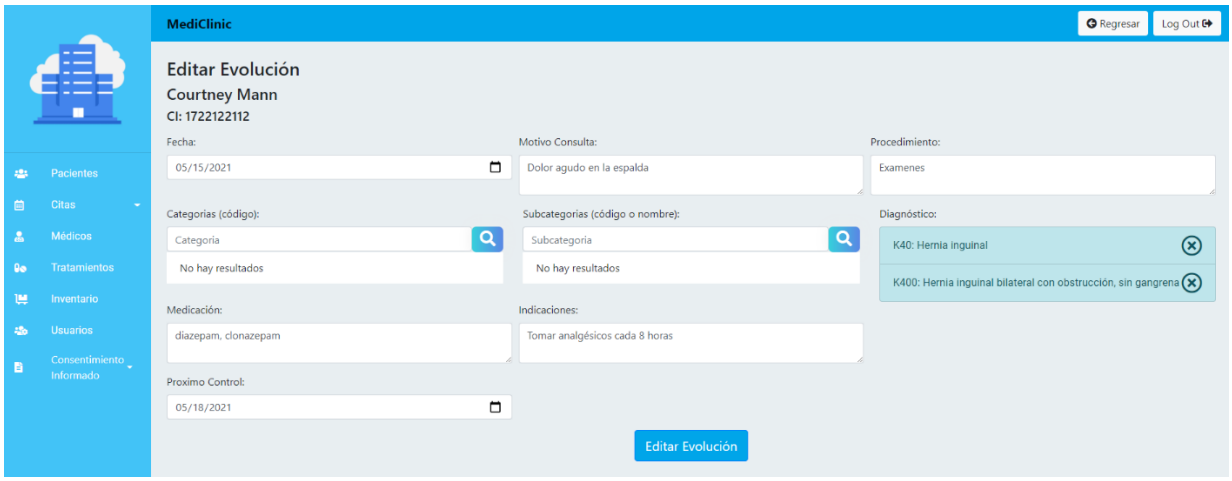
Ilustración 15 Página para la creación de una evolución usando CIE-10



De igual manera para la edición de una evolución se maneja **la clasificación internacional de enfermedades CIE-10**, en este caso la página de edición ya tiene los datos previamente ingresados listos para ser modificados.

Si un usuario no es administrador del sistema ni médico se lo redirecciona a la página de pacientes.

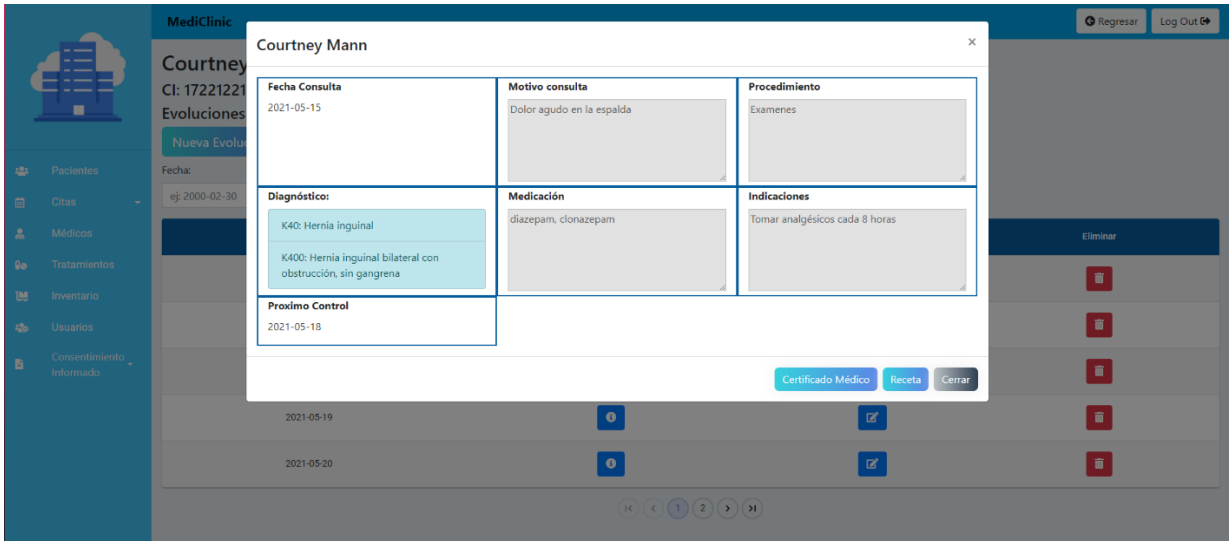
Ilustración 16 Página para la edición de una evolución usando CIE-10



Una vez generada una evolución el médico puede observar los detalles de la evolución accediendo al botón de información en la tabla de evoluciones, de esta forma puede observar a detalle la información y también generar un certificado médico o receta para el paciente asociado.

Si un usuario no es administrador del sistema ni médico se lo redirecciona a la página de pacientes.

Ilustración 17 Modal de información sobre una evolución



En caso de generar un certificado médico para el paciente asociado el sistema redirecciona al usuario hacia una página que contiene la información del consultorio, médico y paciente lista para ser editada (de ser necesario) e impresa para emitir el respectivo certificado médico.

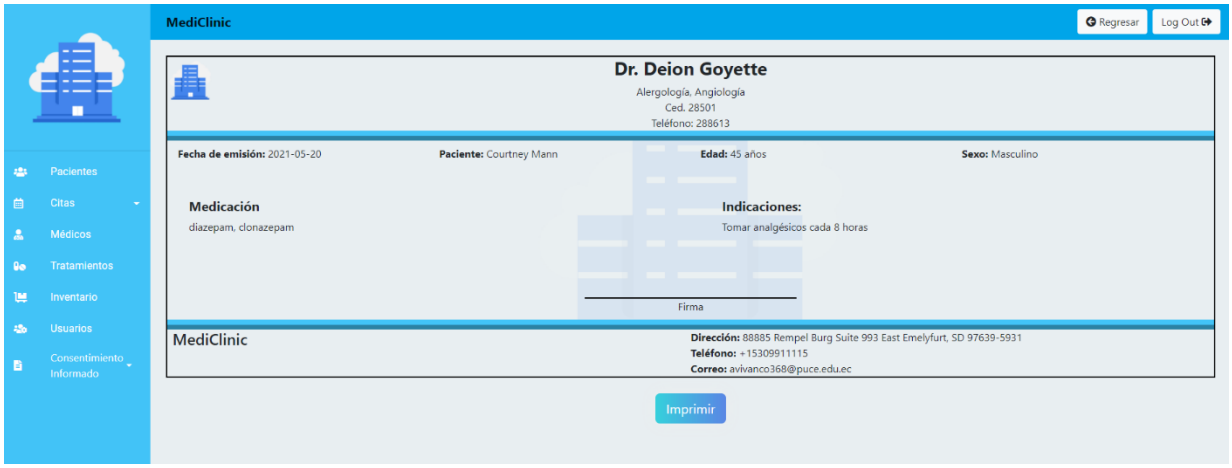
Si un usuario no es administrador del sistema ni médico se lo redirecciona a la página de pacientes.

Ilustración 18 Página para la emisión de un certificado médico

En caso de generar una receta, el sistema toma todo los datos de la evolución, del paciente, y del doctor para generar una receta en base a las indicaciones y medicación guardadas en la evolución. La receta al igual que el certificado pueden ser impresos en PDF.

Si un usuario no es administrador del sistema ni médico se lo redirecciona a la página de pacientes.

Ilustración 19 Página para la emisión de un certificado médico



4.3.1.2 Codificación

Para la codificación de la historia de usuario #1 se utiliza la metodología TDD (test driven development) en la cual a medida que se desarrolla el código también se crean pruebas unitarias para validar la codificación. De igual forma se utiliza herramientas como Jira, Git y Github para mantener una organización correcta entre las versiones del proyecto y dar uso de un tablero Kanban para organizar las tareas.

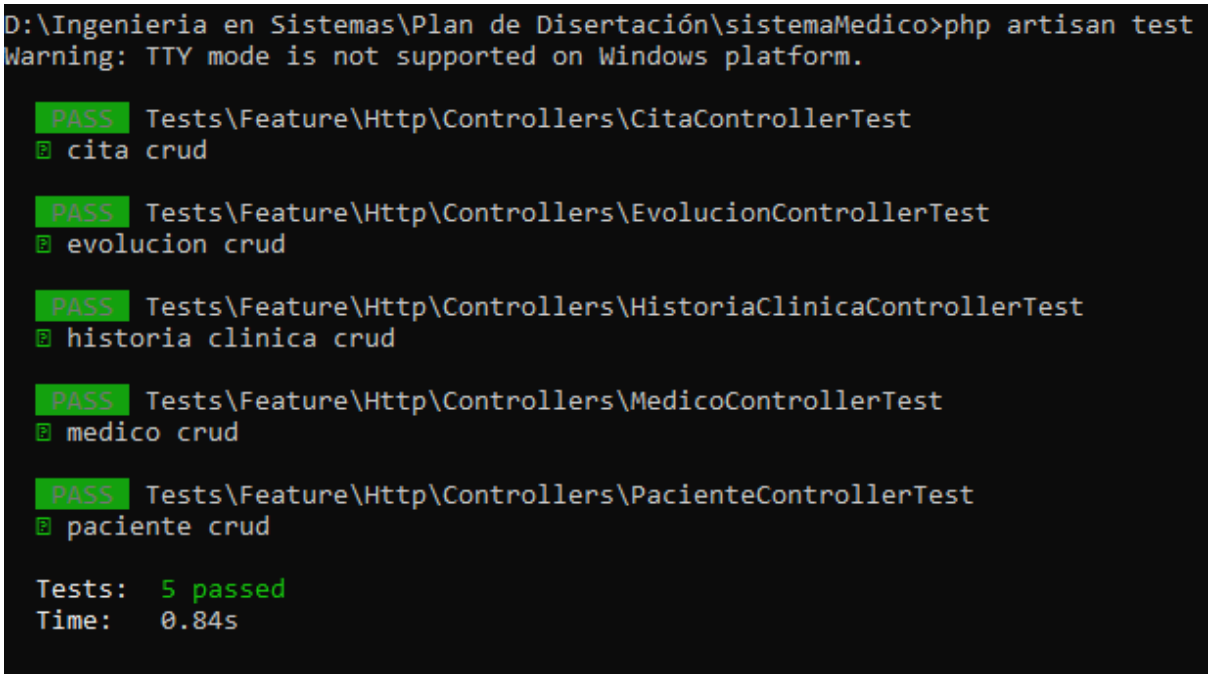
4.3.1.3 Pruebas

A medida que se finalizaba una funcionalidad tanto del back-end como del front-end se creaba una prueba unitaria que validaba la funcionalidad de cada módulo, en el back-end se utilizó PHP-Unit como framework de pruebas y Jest para el front-end. A continuación, se detallan los resultados:

Back-end Laravel php-unit testing:

Cada prueba ejecutada realiza las operaciones crear, leer, editar y eliminar sobre los diferentes endpoints de las entidades descritas en la siguiente ilustración.

Ilustración 20 Pruebas ejecutadas por php-unit las cuales fueron existosas en base a operaciones CRUD para cada entidad



Front-end Jest Javascript unit testing:

El testing en Jest se realizó para comprobar que los componentes desarrollados en REACTJS se monten de manera correcta sobre el navegador, es decir JEST retorna las pruebas exitosas si los componentes se montan exitosamente sobre el DOM (document object model).

Ilustración 21 Pruebas Unitarias para los componentes echos en REACTJS

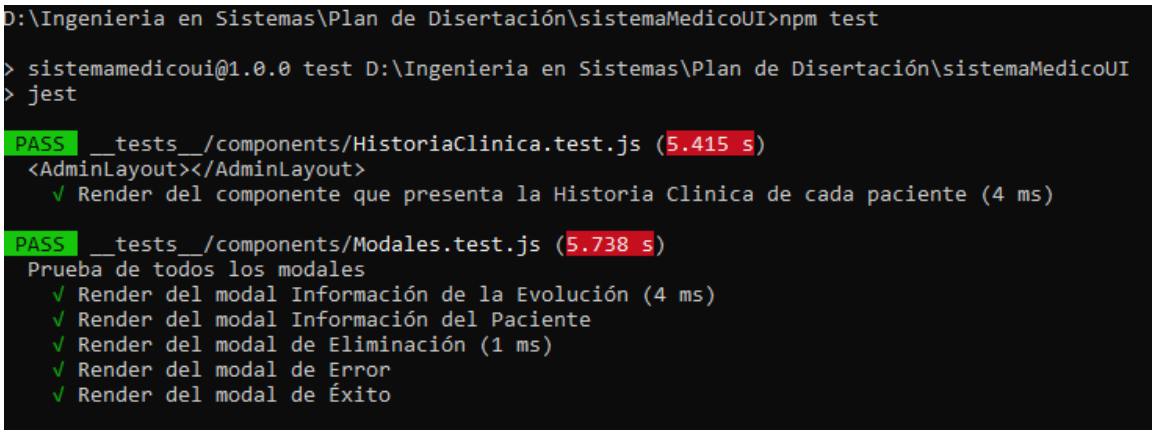


Ilustración 22 Continuación de pruebas unitarias para los componentes echos en REACTJS

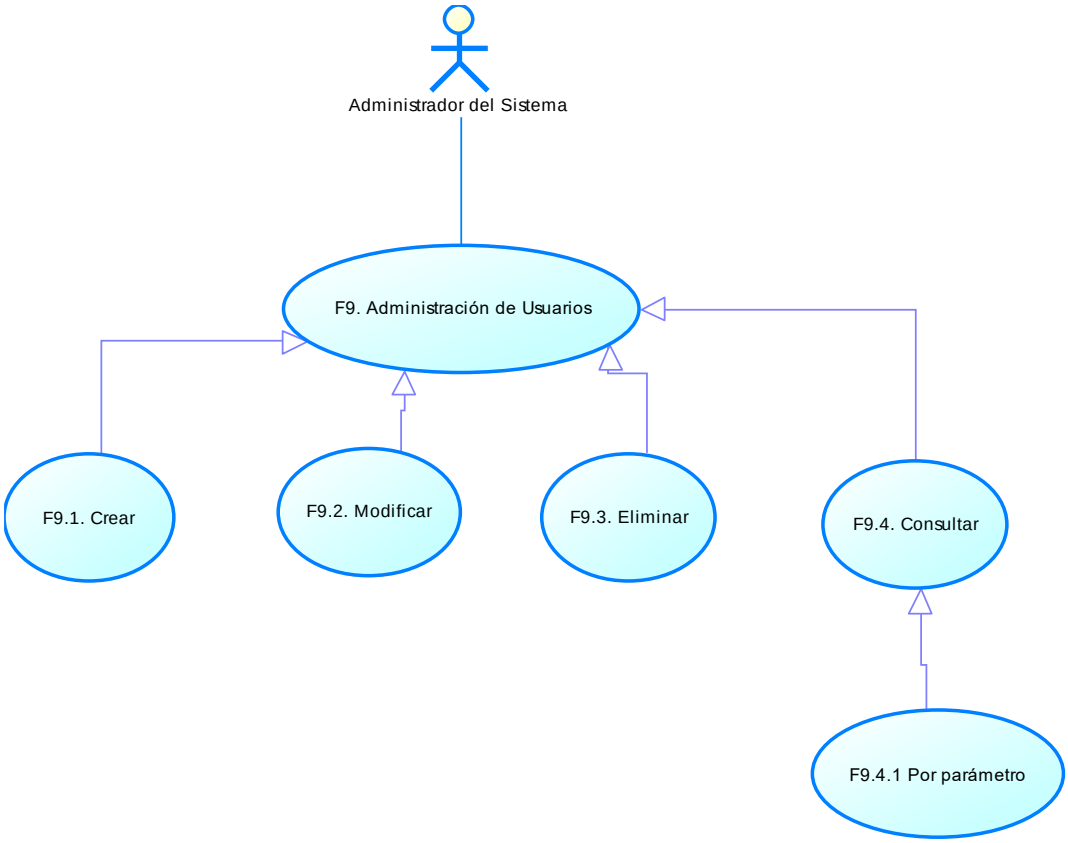
```
PASS  __tests__/components/EvolucionesTable.test.js (6.078 s)
  <EvolucionesTable></EvolucionesTable>
    √ Render del componente Evoluciones Table con información (2 ms)

PASS  __tests__/components/PacientesTable.test.js (6.021 s)
  <AdminLayout></AdminLayout>
    √ Render del componente Pacientes Table con información (3 ms)

PASS  __tests__/components/AdminLayout.test.js (7.364 s)
  <AdminLayout></AdminLayout>
    √ Render del componente AdminLayout como wrapper (4 ms)

Test Suites: 5 passed, 5 total
Tests:       9 passed, 9 total
Snapshots:   0 total
Time:        11.073 s, estimated 24 s
Ran all test suites.
```

4.3.2 Iteración #2 – Historia de Usuario #7

4.3.2.1 Diseño

Casos de uso Usuarios

Ilustración 23 Caso de uso usuarios

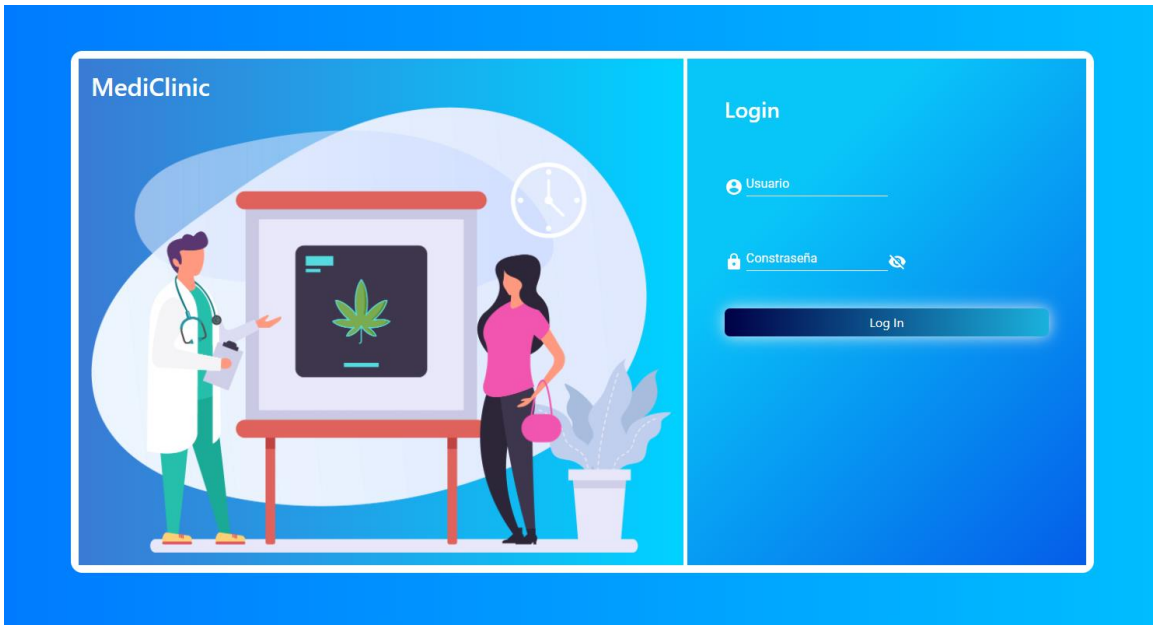


Vistas de la Historia de Usuario #7

Existen 3 tipos de usuarios dentro del sistema el **usuario administrador del sistema** el cual es el doctor a cargo de la Clínica y tiene todos los permisos sobre el sistema. También está el **usuario médico** quien tiene acceso a las historias y evoluciones de los pacientes y al resto del sistema. Finalmente, el sistema también cuenta con el **usuario administrativo** quien carece de permisos sobre las historias clínicas de los pacientes y las evoluciones asociadas a esas historias ya que solo los médicos manejan esa información sensible.

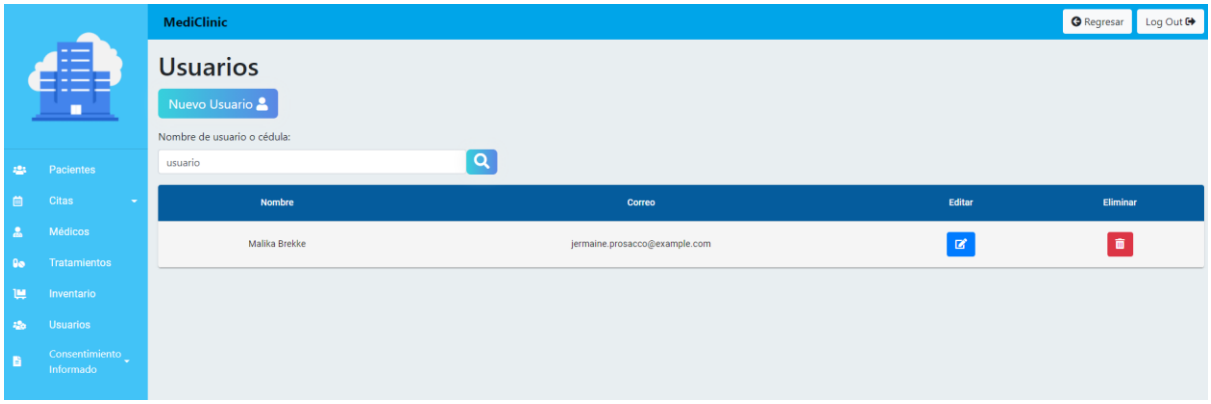
En caso de que un usuario que no tenga permisos intente forzar su entrada a una página por medio de una url el sistema inmediatamente lo redirecciona y no permite el ingreso.

Ilustración 24 Página de Log In para usuarios



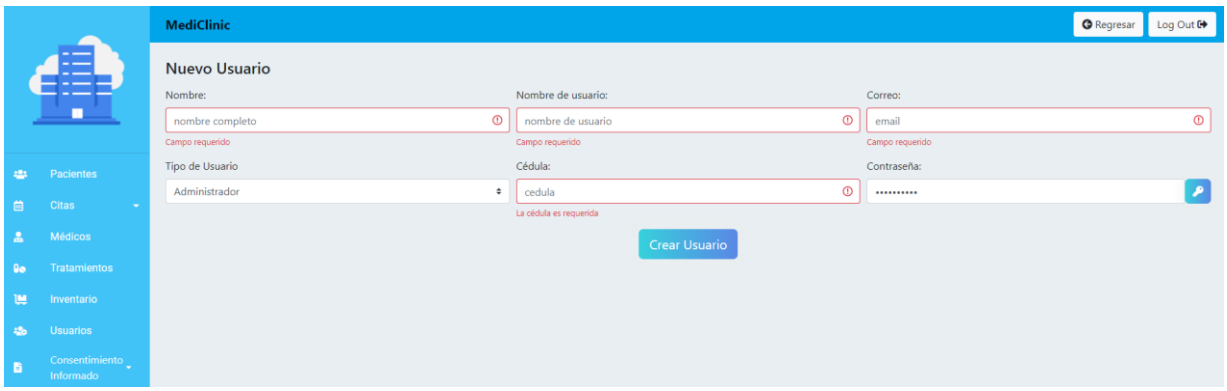
Solamente el usuario administrador del sistema puede acceder a la administración de usuarios. Si desea buscar un usuario en específico requiere del nombre de usuario o cédula para encontrarlo. Por temas de seguridad no se presentan en tabla todos los usuarios se debe buscar uno ingresando los datos previamente mencionados.

Ilustración 25 Página para la búsqueda de un usuario



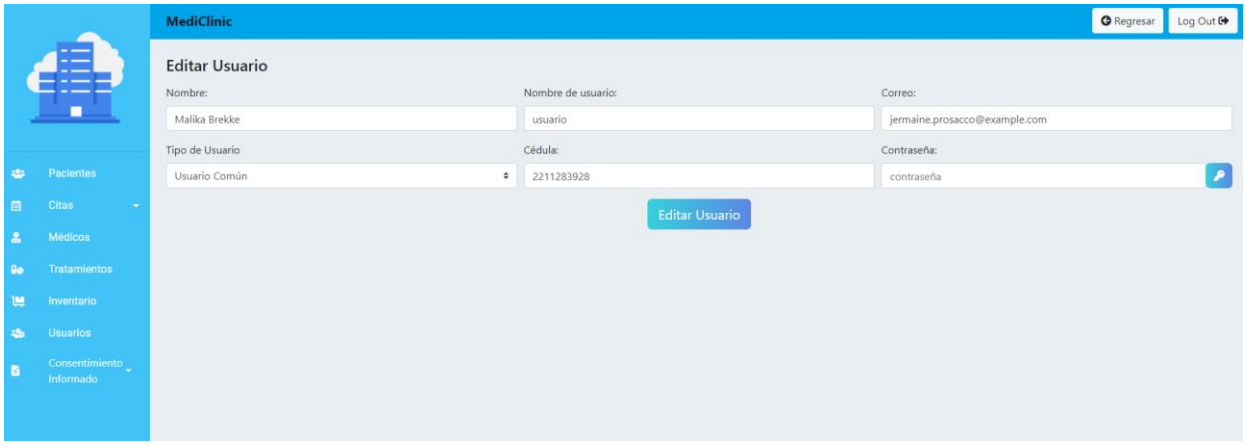
Al crear un nuevo usuario el sistema provee claves seguras aleatorias, mediante un botón, para la creación de usuario y estas credenciales no son reveladas, más bien se envían al correo asociado al usuario creado para que este pueda cambiar su contraseña al ingresar por primera vez al sistema por medio de su perfil. También existe la opción de que el usuario administrador del sistema ingrese una clave propia. El sistema pide datos como cedula, nombre de usuario, correo y tipo de usuario para la creación.

Ilustración 26 Página para la creación de un nuevo usuario



El usuario administrador del sistema también puede modificar los datos de un usuario, esto puede resultar útil en caso de problemas con contraseñas.

Ilustración 27 Página para la edición de un usuario



4.3.2.2 Codificación

Las SPA (single page apps) son sistemas web que ofrecen una reducción en el TTFB (time to first byte), es decir dan una mayor velocidad de navegación. Esto se debe a que no refrescan el navegador cuando se navega entre páginas, el DOM (document object model) ya tiene listo gran parte de los componentes visuales de la aplicación por lo que no hay necesidad de tomar mucho tiempo al cargar los recursos. Sin embargo, las SPA no manejan sesiones ya que no se refresca la página. Para solucionar este problema, y saber si un usuario esta autenticado y tiene autorización, existen los API tokens, que son cadenas de caracteres aleatorias que genera el servidor para acceder a los recursos e información del back-end. El proceso es descrito en la siguiente ilustración:

Ilustración 28 López Magaña, L. M. (2020, 17 enero). Qué es Json Web Token y cómo funciona [Ilustración]. OpenWebinars. https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/

El sistema utiliza JWT (jason web tokens), los cuales son un método RFC 7519 estándar de la industria abierto para representar reclamaciones de forma segura entre dos partes (JWT.io, 2021).

El sistema funciona de la siguiente manera:

1. El usuario se autentica con sus credenciales de forma correcta, caso contrario el servidor lanza una excepción
2. El servidor envía un JWT valido por 2 horas hacia el front-end o cliente
3. El front-end almacena el JWT con una cookie que posee una encriptación de un solo camino y con llave secreta durante 2 horas.
4. En cada petición se accede a la cookie y se envía al back-end el JWT para la obtención de información
5. Si el JWT es correcto el back-end devuelve la información del recurso accedido caso contrario envía errores http 401 (no autorizado) o 403 (prohibido) respectivamente.

De esta forma se tiene bloqueado el back-end con el método de autenticación API tokens y el front-end tiene una encriptación de un solo camino para que el token no sea visible en el navegador. Esto hace al sistema seguro y evita accesos no autorizados.

Cabe recalcar que este tipo de autenticación es más seguro en la actualidad, siendo más seguro que la autenticación vía HTTP y autenticación HMAC.

4.3.2.3 Pruebas

Resultados de las pruebas unitarias del front-end y back-end:

Back-end con PHP-unit testing:

Ilustración 29 Pruebas en php unit, se agregó la prueba usuario autenticación

```
D:\Ingenieria en Sistemas\Plan de Disertación\sistemaMedico>php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\Http\Controllers\CitaControllerTest
  ⨯ cita crud

  PASS  Tests\Feature\Http\Controllers\EvolucionControllerTest
  ⨯ evolucion crud

  PASS  Tests\Feature\Http\Controllers\HistoriaClinicaControllerTest
  ⨯ historia clinica crud

  PASS  Tests\Feature\Http\Controllers\MedicoControllerTest
  ⨯ medico crud

  PASS  Tests\Feature\Http\Controllers\PacienteControllerTest
  ⨯ paciente crud

  PASS  Tests\Feature\Http\Controllers\UsuarioTest
  ⨯ usuario autenticacion

  Tests:  6 passed
  Time:   0.97s
```

La prueba agregada fue usuario autenticación y lo que hace es crear un usuario de forma momentánea y realizar una petición de Log In al recurso de autenticación, si se obtiene un status 200 el recurso funciona caso contario da error.

Front-end con JEST:

Se agrego una prueba para verificar que el componente de Log In que tiene la página log in page se monte exitosamente.

Ilustración 30 Pruebas unitarias con JEST

```
PASS  __tests__/components/EvolucionesTable.test.js (6.587 s)
  <EvolucionesTable></EvolucionesTable>
    √ Render del componente Evoluciones Table con información (4 ms)

PASS  __tests__/components/PacientesTable.test.js (6.592 s)
  <AdminLayout></AdminLayout>
    √ Render del componente Pacientes Table con información (6 ms)

PASS  __tests__/components/AdminLayout.test.js (7.969 s)
  <AdminLayout></AdminLayout>
    √ Render del componente AdminLayout como wrapper (4 ms)

PASS  __tests__/components/LoginPage.test.js (14.833 s)
  <LoginPage></LoginPage>
    √ Render del componente Login en la pagina login_page (4 ms)

Test Suites: 6 passed, 6 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        18.755 s
Ran all test suites.
```

4.3.3 Iteración #3 – Historia de usuario #2

4.3.3.1 Diseño

Casos de uso Citas

Ilustración 31 Casos de uso citas

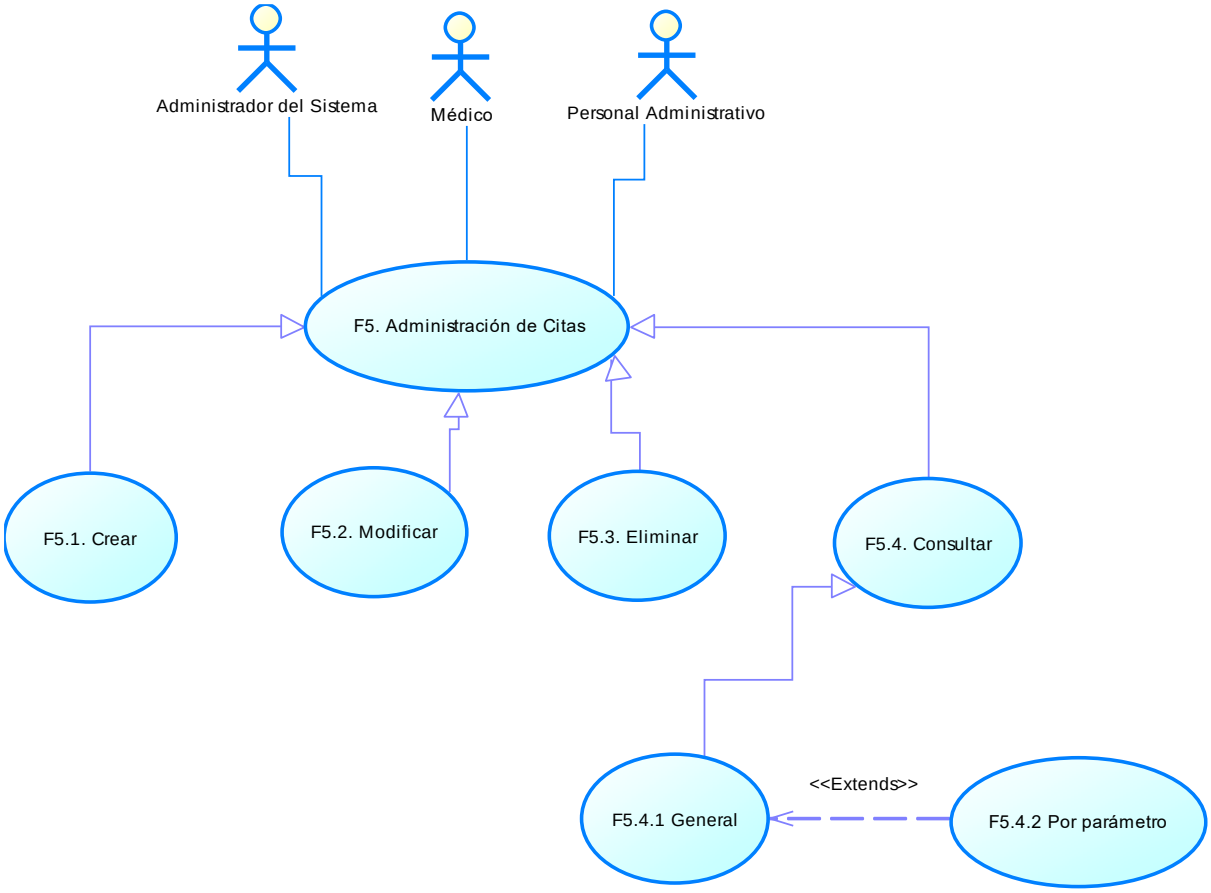
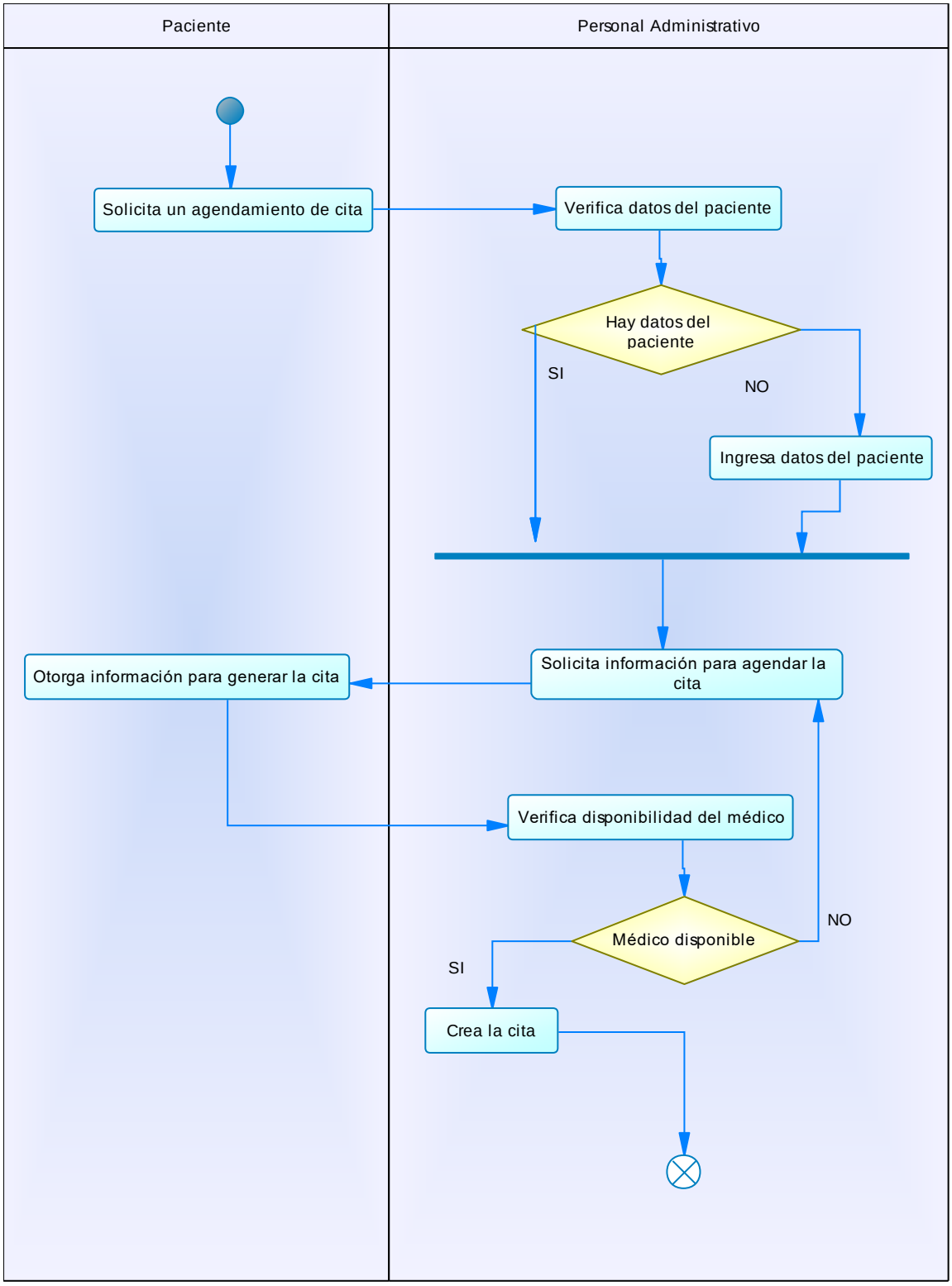


Diagrama de actividades para Citas

Ilustración 32 Diagrama de actividades para citas

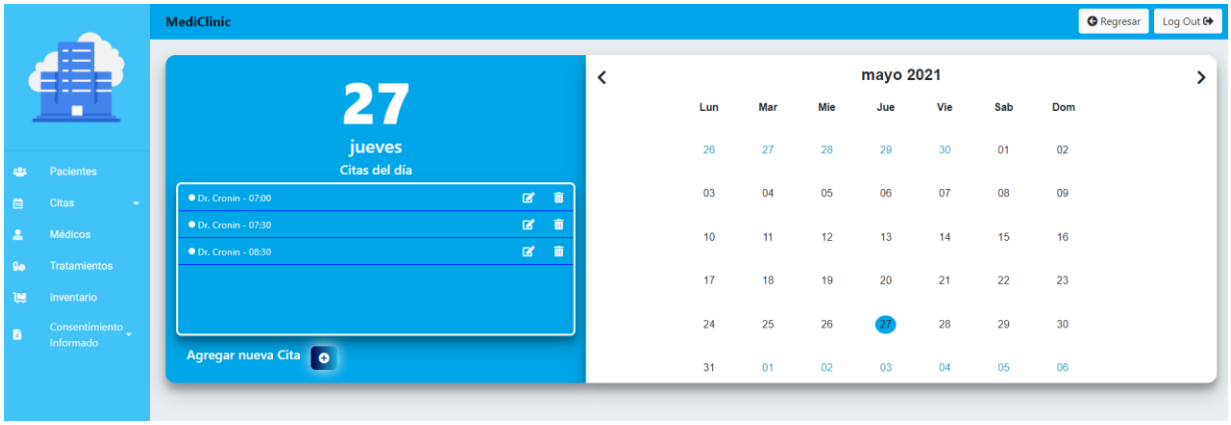


Vistas de la Historia de Usuario #2

Calendario Variante 1

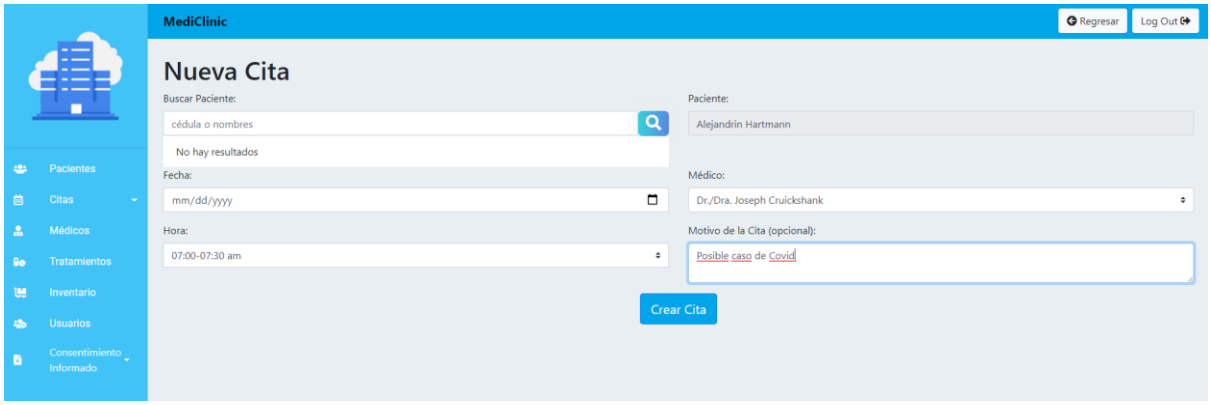
En el sistema se desarrollaron dos variantes de calendarios. En ambas variantes si el usuario que ingresa es doctor se presentan solo las citas de ese médico, si el usuario es del personal administrativo entonces puede seleccionar al médico del cual desea observar las citas. La primera variante del calendario se presenta a continuación:

Ilustración 33 Página de citas con la variante 1 de calendario de eventos, usuario médico Dr. Cronin



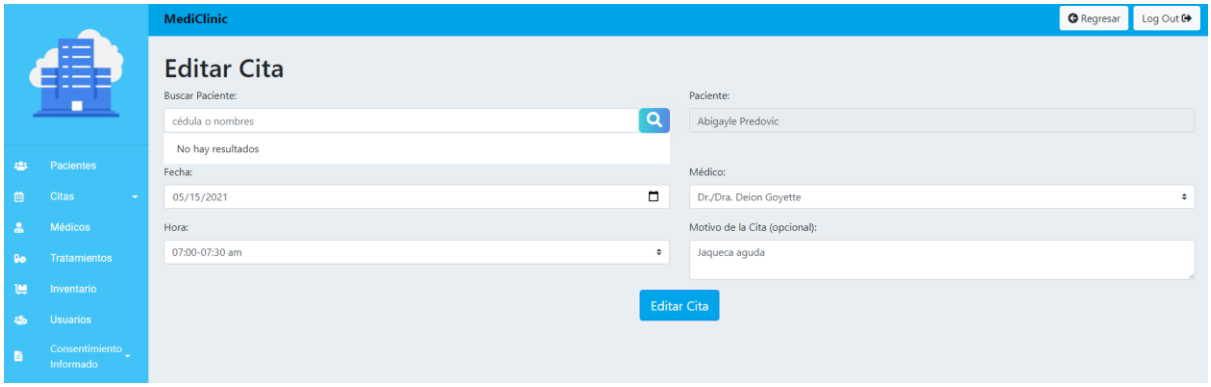
En caso de dar click sobre el botón de Agregar nueva Cita el sistema presenta la siguiente página:

Ilustración 34 Página para la creación de citas, calendario variante 1



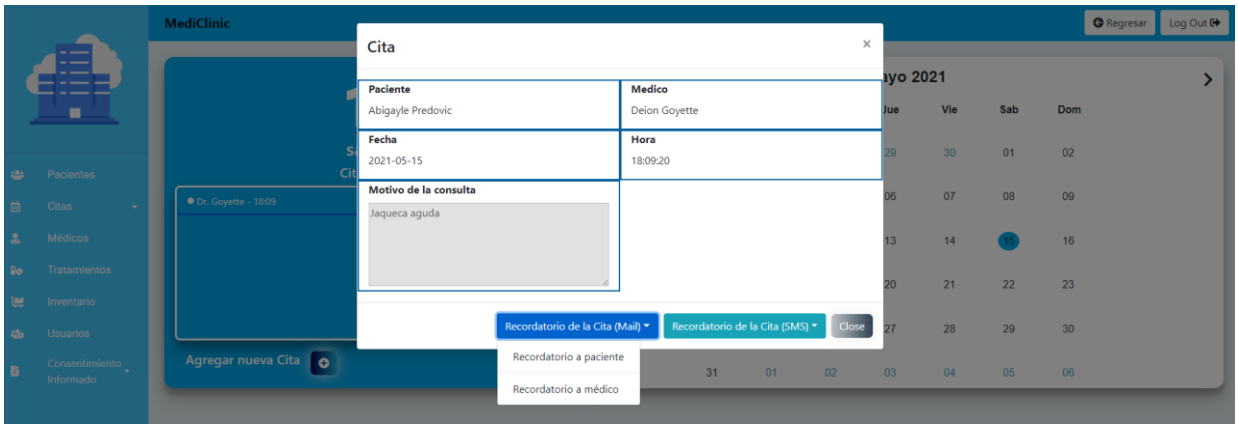
En caso de presionar el botón de edición en el calendario el sistema presenta la siguiente página:

Ilustración 35 Página para la edición de citas, calendario variante 1



En caso de presionar sobre la cita el sistema muestra un modal que tiene los datos de la cita, este modal también le permite al usuario enviar recordatorios sms y mail, tanto al paciente como al médico.

Ilustración 36 Modal para ver información de una cita, calendario variante 1



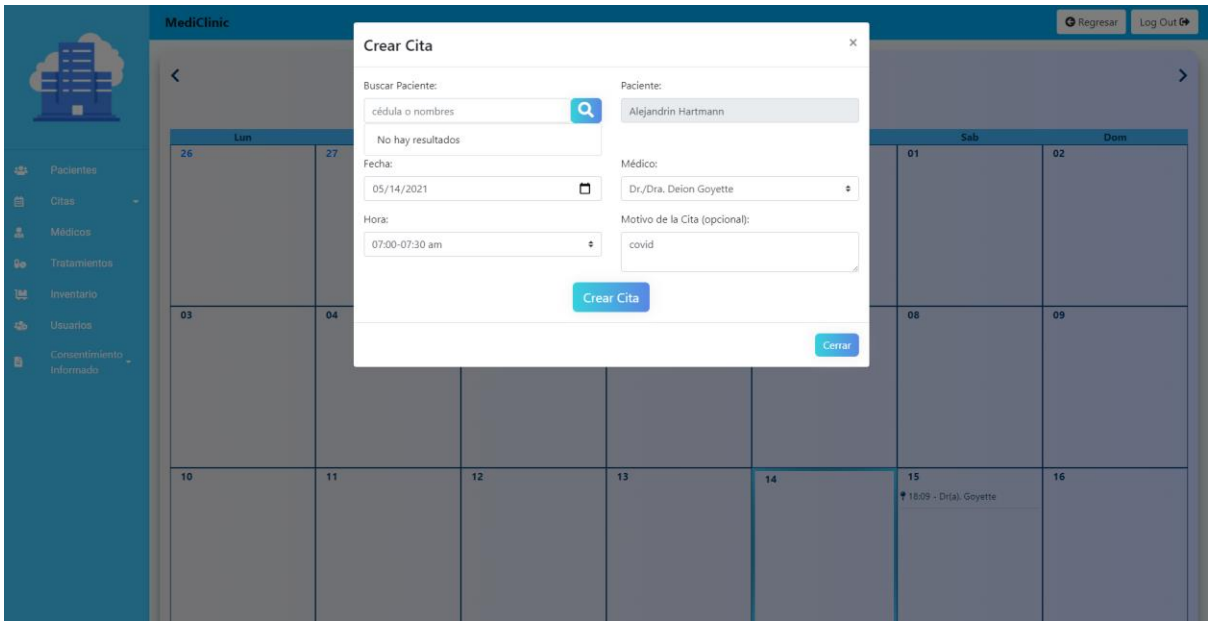
Calendario variante 2

La segunda variante del calendario marca las citas de un doctor dentro de cada día del mes, tiene todas las funcionalidades que el anterior calendario mencionado solo cambia su presentación:

Ilustración 37 Página de citas con la variante 2 de calendario de eventos, usuario médico Dr. Cronin

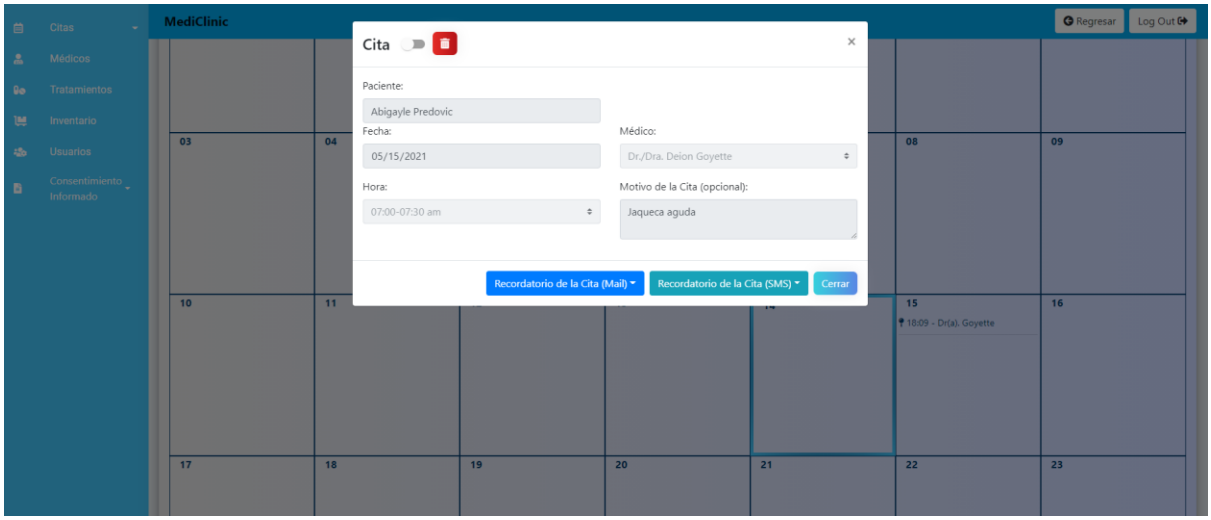
Si se desea crear una cita se presiona en el día que se requiere la cita, esto despliega un modal para crear la cita respectiva:

Ilustración 38 Modal para la creación de citas, calendario variante 2



En caso de ver la información de una cita, editarla o eliminarla se debe presionar la cita, esto desplegará un modal que permite ver la información de la cita y contiene botones que permiten al usuario editar la cita, eliminarla y enviar recordatorios por sms o mail.

Ilustración 39 Modal para ver información de la cita y ejecutar múltiples acciones, calendario variante 2



4.3.3.2 Codificación

Los calendarios utilizan la librería **date-fns** (date-fns, 2021) la cual provee un conjunto de herramientas simples y consistentes para el manejo adecuado de fechas en el lenguaje de programación JavaScript. Gracias a este paquete se puede personalizar los calendarios y ejecutar cualquier cambio deseado, de esta forma no se depende de calendarios en paquetes externos los cuales son complicados de manipular.

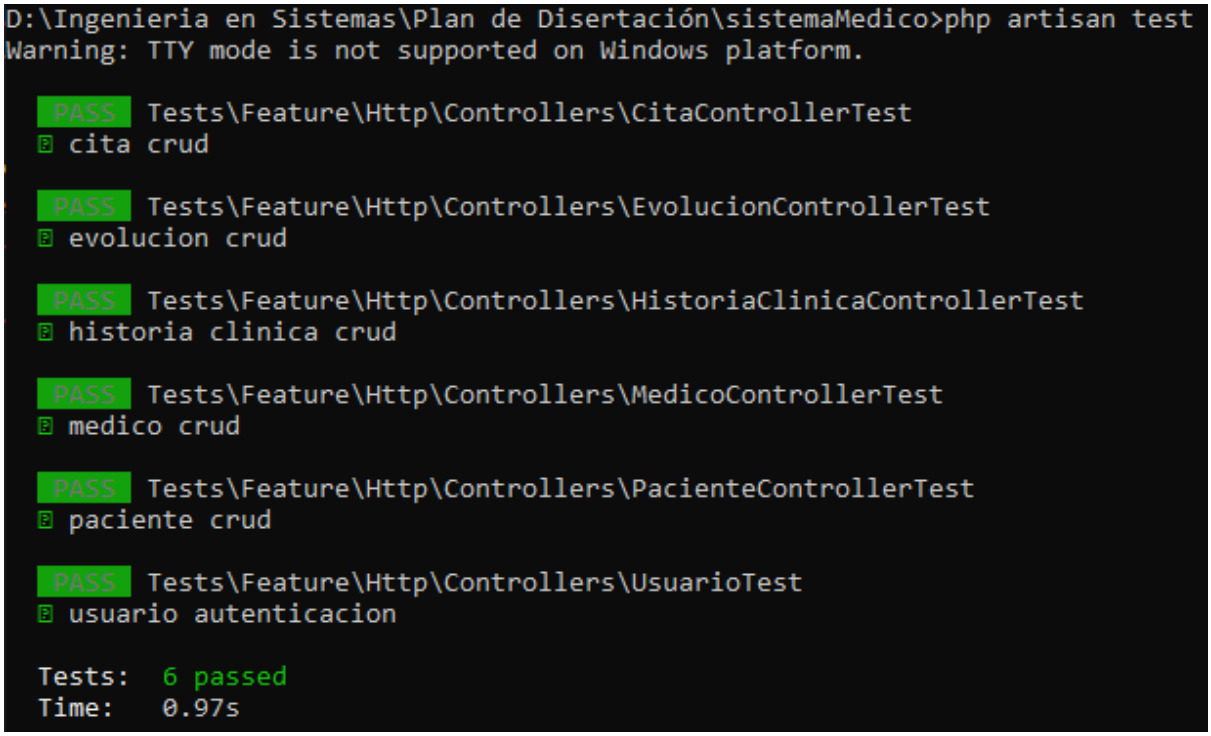
Date-fns provee diversos métodos para manejar la fecha de acuerdo a: el formato ISO de fechas, la región que se requiere, o al idioma deseado. De esta forma se puede extraer los días, meses u horas locales para trabajarlas en el sistema de forma sencilla.

4.3.3.3 Pruebas

Resultados de las pruebas unitarias del front-end y back-end:

Back-end con PHP-unit testing:

Ilustración 40 Pruebas unitarias en php unit



La prueba de esta historia de usuario es cita crud la cual determina si las funcionalidades de post, put, get, delete funcionan correctamente sobre el modelo de cita.

Front-end con JEST:

Las pruebas se realizaron para verificar que los componentes de los calendarios se monten exitosamente sobre el DOM. Las pruebas correspondientes son <Calendario1> y <Calendario2>.

Ilustración 41 Pruebas de front-ent en JEST

```
PASS  __tests__/components/PacientesTable.test.js (8.663 s)
  <AdminLayout></AdminLayout>
    √ Render del componente Pacientes Table con información (7 ms)

PASS  __tests__/components/AdminLayout.test.js (10.834 s)
  <AdminLayout></AdminLayout>
    √ Render del componente AdminLayout como wrapper (5 ms)

PASS  __tests__/components/Calendario1.test.js (14.504 s)
  <Calendario 1></Calendario 1>
    √ Render del componente Calendario en la pagina Citas (6 ms)

PASS  __tests__/components/Calendario2.test.js (14.647 s)
  <Calendario 2></Calendario 2>
    √ Render del componente Calendario Variante en la pagina Citas (6 ms)

PASS  __tests__/components/LoginPage.test.js (19.686 s)
  <LoginPage></LoginPage>
    √ Render del componente Login en la pagina login_page (3 ms)

Test Suites: 8 passed, 8 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        25.923 s
Ran all test suites.
```

4.3.4 Iteración #4 – Historia de Usuario #3

4.3.4.1 Diseño

Vistas de la Historia de Usuario #3

En ambas variantes de los calendarios existe la opción de enviar correos electrónicos o notificaciones SMS con los datos de una cita creada, es decir la hora, la fecha, el doctor, etc. Esto es posible en los modales que dan la información de la cita. El diseño es simple, si se desea enviar un correo o SMS al paciente solo se accede al botón desplegable que se requiere, el cual da la opción de enviar el recordatorio al paciente o al médico.

Ilustración 42 Modal para ver información de una cita, calendario variante 1

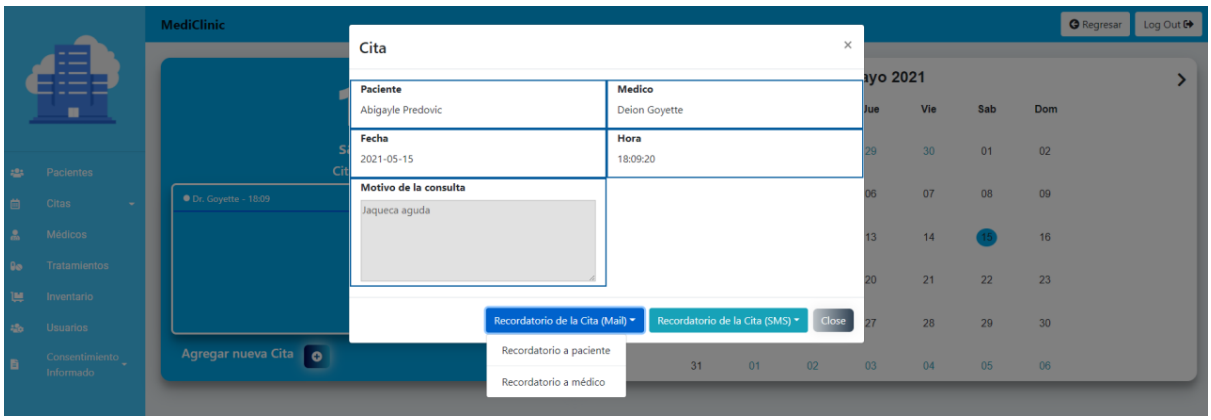
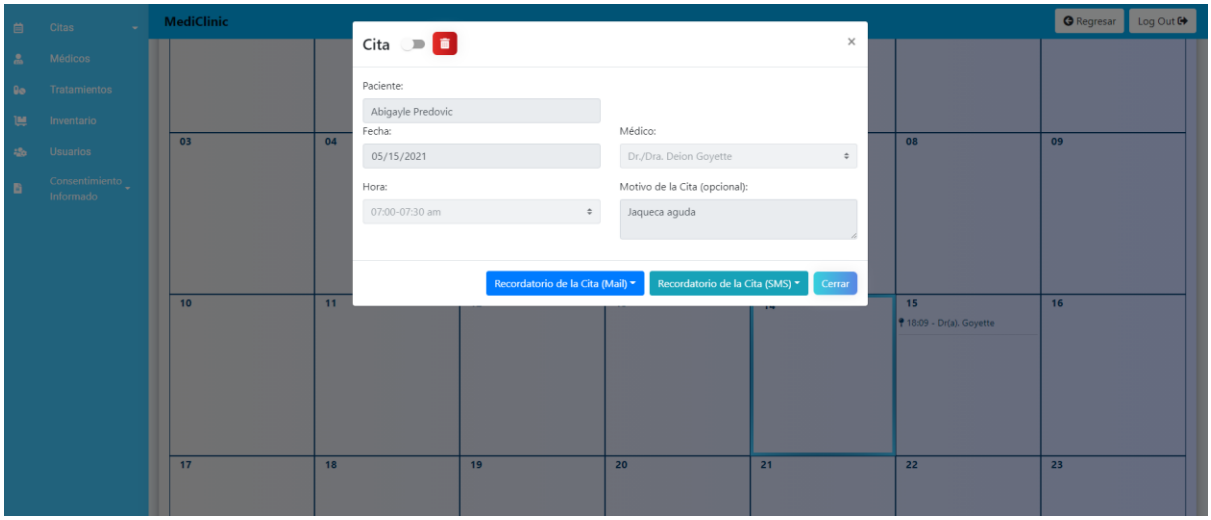


Ilustración 43 Modal para ver información de una cita, calendario variante 2



4.3.4.2 Codificación

El envío de correos electrónicos o notificaciones SMS deben realizarse a través de un proveedor. Existen diversos proveedores que ofrecen este tipo de servicios como Sendinblue, Sendgrid, Mailtrap, Nexmo, etc. Los proveedores dan a sus clientes un punto de conexión o API para integrar estos servicios a un sistema. En el caso de Laravel ya están hechas las configuraciones necesarias para integrar este tipo de servicios lo único que se requiere es un proveedor y colocar las credenciales requeridas dentro de las variables de entorno en el archivo "**.env**" de la aplicación.

En el caso de los correos electrónicos se usó MailTrap para desarrollo ya que los correos enviados por este proveedor SMTP son solo de prueba y así se evita llenar una bandeja de correos innecesarios. Para producción se utilizó Sendinblue que provee un paquete gratis para el envió de 500 correos gratuitos. Laravel ofrece una **Clase** llamada **Mail** la cual, una vez configurado el proveedor, permite enviar correos usando los métodos **to**() y **send**().

En el caso de las notificaciones SMS no hay proveedores que ofrezcan envíos gratuitos, es necesario pagar para acceder a un paquete básico de envíos. En este proyecto se usó una cuenta de prueba creada en Nexmo la cual cuenta con 2 euros para el envió de SMS de prueba solamente a números registrados. Al igual que con los correos, se deben agregar las credenciales requeridas en las variables de entorno de la aplicación y utilizar la **Clase Notification** con los métodos **route**() y **notify**() para enviar el SMS satisfactoriamente.

Cabe recalcar que es necesario Crear vistas en Laravel las cuales sirvan para dar formato tanto al correo electrónico como al SMS.

4.3.4.3 Pruebas

Resultados de las pruebas unitarias del front-end y back-end:

Back-end con PHP-unit testing:

Ilustración 44 Pruebas de back-end en PHP-Unit

```
D:\Ingenieria en Sistemas\Plan de Disertación\sistemaMedico>php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\Http\Controllers\CitaControllerTest
  ⨯ cita crud

  PASS  Tests\Feature\Http\Controllers\EvolucionControllerTest
  ⨯ evolucion crud

  PASS  Tests\Feature\Http\Controllers\HistoriaClinicaControllerTest
  ⨯ historia clinica crud

  PASS  Tests\Feature\Http\Controllers\MedicoControllerTest
  ⨯ medico crud

  PASS  Tests\Feature\Http\Controllers\NotificacionesTest
  ⨯ enviar notificacion

  PASS  Tests\Feature\Http\Controllers\PacienteControllerTest
  ⨯ paciente crud

  PASS  Tests\Feature\Http\Controllers\UsuarioTest
  ⨯ usuario autenticacion

  Tests:  7 passed
  Time:   0.94s
```

La prueba agregada fue 'enviar notificación' la cual ejecuta un postJson hacia el endpoint o recurso REST encargado de enviar el correo con información, si se obtiene un código HTTP 200 la prueba fue satisfactoria.

Front-end con JEST:

Ilustración 45 Pruebas de front-ent en JEST

```
PASS  __tests__/components/PacientesTable.test.js (8.663 s)
  <AdminLayout></AdminLayout>
    √ Render del componente Pacientes Table con información (7 ms)

PASS  __tests__/components/AdminLayout.test.js (10.834 s)
  <AdminLayout></AdminLayout>
    √ Render del componente AdminLayout como wrapper (5 ms)

PASS  __tests__/components/Calendario1.test.js (14.504 s)
  <Calendario 1></Calendario 1>
    √ Render del componente Calendario en la pagina Citas (6 ms)

PASS  __tests__/components/Calendario2.test.js (14.647 s)
  <Calendario 2></Calendario 2>
    √ Render del componente Calendario Variante en la pagina Citas (6 ms)

PASS  __tests__/components/LoginPage.test.js (19.686 s)
  <LoginPage></LoginPage>
    √ Render del componente Login en la pagina login_page (3 ms)

Test Suites: 8 passed, 8 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        25.923 s
Ran all test suites.
```

Esta prueba no difiere de la elaborada en la historia de usuario anterior ya que el envío de notificaciones está dentro del calendario. Debido a esto si la prueba <Calendario1> y <Calendario2> son exitosas significa que los botones para envío de notificaciones ubicados en los modales se montaron con éxito.

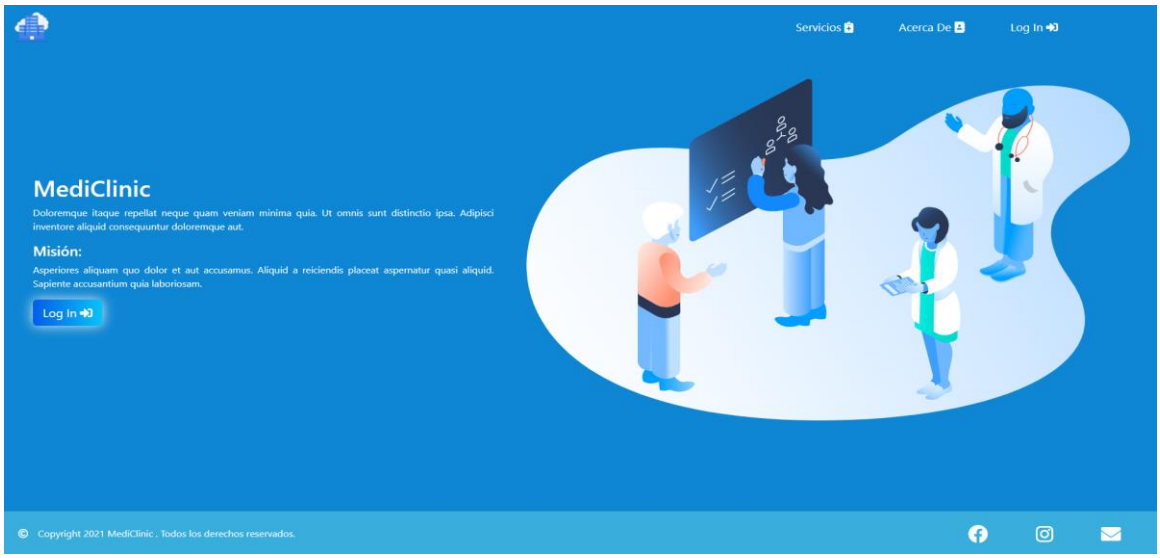
4.3.5 Iteración #5 – Historia de Usuario # 6

4.3.5.1 Diseño

Vistas de la Historia de Usuario #6

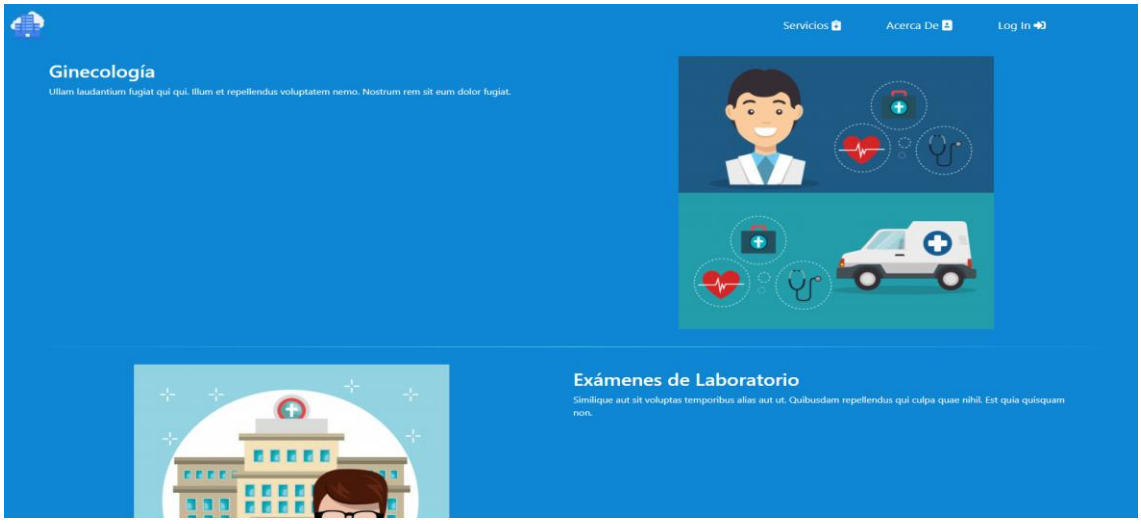
La página principal que publicita al consultorio presenta la misión, visión y nombre del consultorio. También contiene una barra de navegación para acceder a otras páginas y un pie de página que contiene las redes sociales y contactos del consultorio.

Ilustración 46 Página principal del sistema



De igual manera existe la página de servicios la cual muestra todos los servicios que el consultorio provee.

Ilustración 47 Página de servicios del consultorio



4.3.5.2 Codificación

El desarrollo de aplicaciones web SPA en JavaScript provee al cliente gran velocidad para navegar y permite al desarrollador crear sitios novedosos con gran experiencia de usuario (UX) debido a la gran

cantidad de paquetes JavaScript que existen. Sin embargo, existen ciertas desventajas, el data fetching (petición a una api) puede resultar lento y hacer que un sitio web no cargue los datos de forma rápida, lo que da al cliente la impresión de una página web lenta. Otra desventaja, es que los buscadores no pueden indexar el contenido de la página web en su algoritmo, debido a que es código escrito en JS y da a los buscadores un error 404, por lo que no se indexa la página web.

Para solucionar estas desventajas existe el **SSR** (server side render) o renderizado por servidor el cual toma todo el código en JS, lo construye y devuelve un "HTML STRING", es decir una vista web ya construida desde el servidor. Usualmente para esto se combina ReactJS con Express y Webpack. Sin embargo, la configuración es complicada y se requieren muchos paquetes para lograrlo. Ante esto surgen frameworks de desarrollo que ya tienen configurado el SSR como NextJS o Gatsby, por lo que los desarrolladores solo tienen que preocuparse de crear componentes con un código limpio.

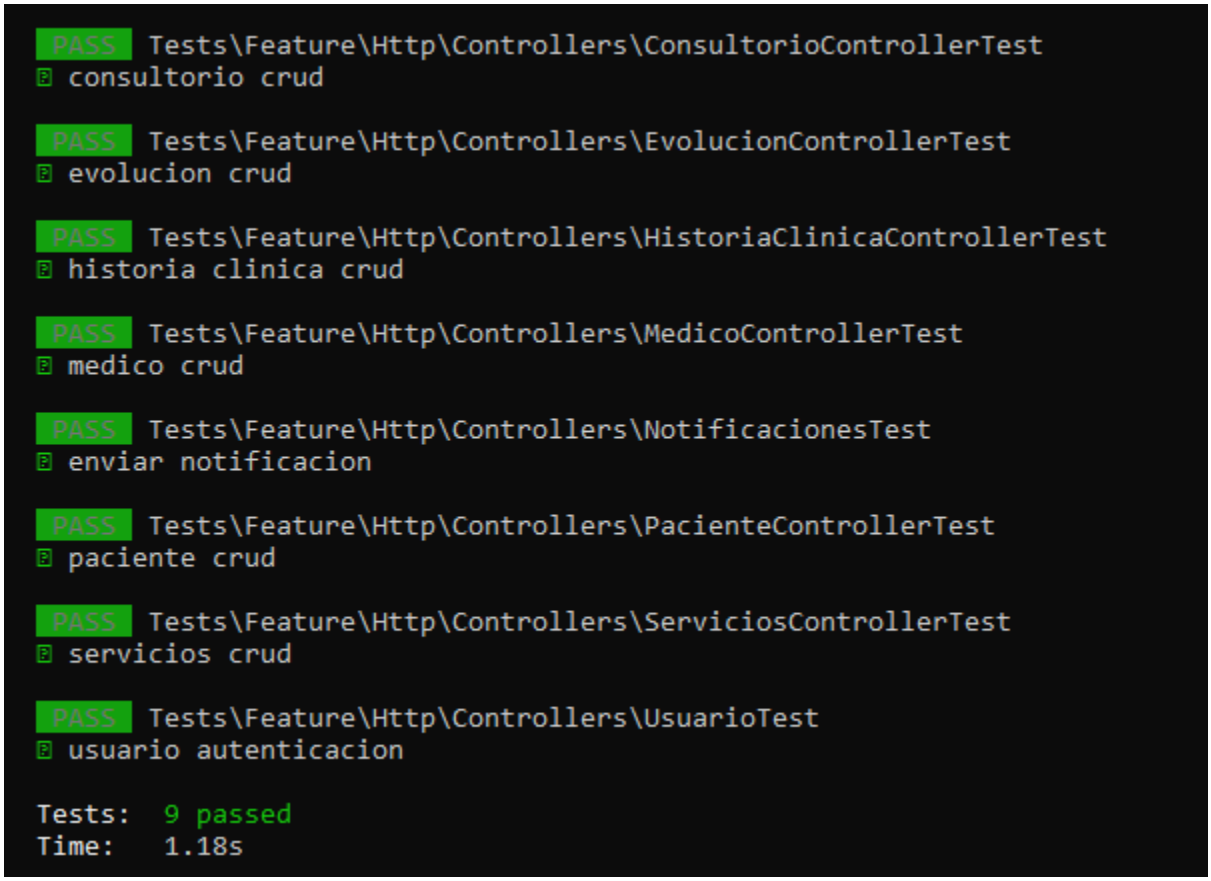
Este proyecto implementa NextJS, un framework que ya integra SSR para ReactJS y que además provee ciertos estándares para obtener datos durante la compilación del código, esto hace que la página web mejore su velocidad y sea capaz de ser indexada por los buscadores. En las piezas de código se utilizó palabras clave para el data fetching como **getServerSideProps** y **getStaticProps** lo que hace que la data obtenida de la api se presente mucho más rápido y en algunos casos esté ya disponible desde la compilación del sistema.

4.3.5.3 Pruebas

Resultados de las pruebas unitarias del front-end y back-end:

Back-end con PHP-unit testing:

Ilustración 48 Pruebas de back-end en PHP-Unit



Se realizaron pruebas sobre los controladores de Consultorio y Servicios para probar que los recursos efectúen un CRUD de manera adecuada. Las pruebas agregadas fueron **consultorio crud** y **servicios crud.**

Front-end con JEST:

Ilustración 49 Pruebas de front-end con JEST

```
PASS  __tests__/components/Modales.test.js (11.132 s)
 Prueba de todos los modales
   √ Render del modal Información de la Evolución (6 ms)
   √ Render del modal Información del Paciente (1 ms)
   √ Render del modal de Eliminación (1 ms)
   √ Render del modal de Error (2 ms)
   √ Render del modal de Éxito (2 ms)

PASS  __tests__/components/HistoriaClinica.test.js (11.466 s)
 <AdminLayout></AdminLayout>
   √ Render del componente que presenta la Historia Clinica de cada paciente (7 ms)

PASS  __tests__/components/PacientesTable.test.js (13.021 s)
 <PacientesTable></PacientesTable>
   √ Render del componente Pacientes Table con información (6 ms)

PASS  __tests__/components/EvolucionesTable.test.js
 <EvolucionesTable></EvolucionesTable>
   √ Render del componente Evoluciones Table con información (2 ms)

PASS  __tests__/components/Servicios.test.js (15.459 s)
 <Servicios></Servicios>
   √ Render de componentes en la pagina servicios (6 ms)

PASS  __tests__/components/InventarioPage.test.js (16.125 s)
 <Inventario></Inventario>
   √ Render del componente InventarioTable dentro de la página Inventario (6 ms)

PASS  __tests__/components/TratamientosPage.test.js (15.774 s)
 <Tratamientos></Tratamientos>
   √ Render del componente TratamientosTable dentro de la página tratamientos (9 ms)

PASS  __tests__/components/AdminLayout.test.js (16.409 s)
 <AdminLayout></AdminLayout>
   √ Render del componente AdminLayout como wrapper (7 ms)

PASS  __tests__/components/Layout.test.js (18.104 s)
 <Layout></Layout>
   √ Render del componente Layout como wrapper en la página principal (5 ms)

PASS  __tests__/components/Calendario1.test.js (18.215 s)
 <Calendario 1></Calendario 1>
   √ Render del componente Calendario en la pagina Citas (5 ms)

PASS  __tests__/components/Calendario2.test.js (18.732 s)
 <Calendario 2></Calendario 2>
   √ Render del componente Calendario Variante en la pagina Citas (4 ms)

PASS  __tests__/components/LoginPage.test.js (24.444 s)
 <LoginPage></LoginPage>
   √ Render del componente Login en la pagina login_page (3 ms)

Test Suites: 12 passed, 12 total
Tests:       16 passed, 16 total
Snapshots:   0 total
Time:        30.767 s, estimated 45 s
Ran all test suites.
```

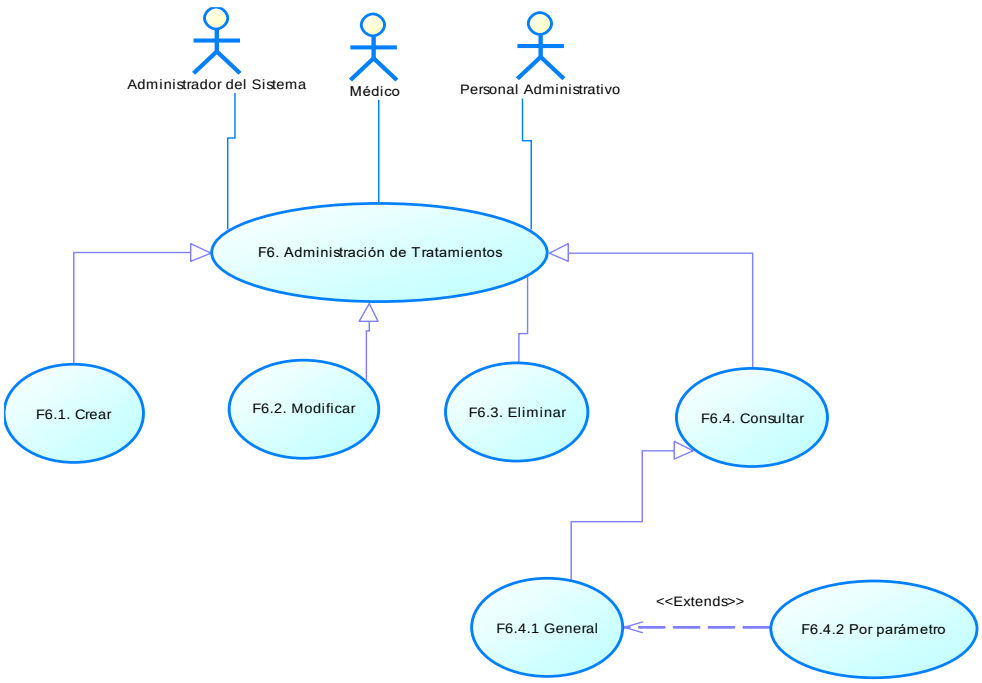
Se probaron los componentes utilizados para la página que publicita los servicios, se comprueba que se hayan montado correctamente en el DOM y también que el Layout utilizado para contener a los componentes se monte adecuadamente. Las pruebas agregadas fueron **<Servicios>** y <**Layout**>.

4.3.6 Iteración #6 – Historia de Usuario # 4

4.3.6.1 Diseño

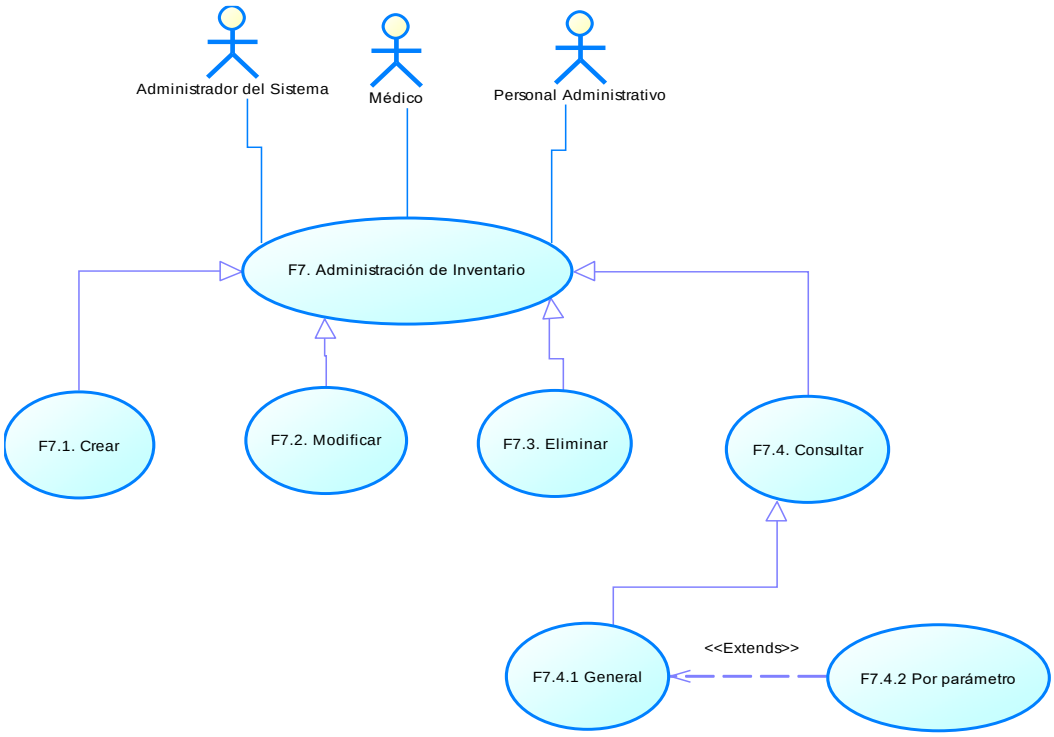
Casos de uso Tratamientos

Ilustración 50 Casos de uso tratamientos



Casos de uso Inventario

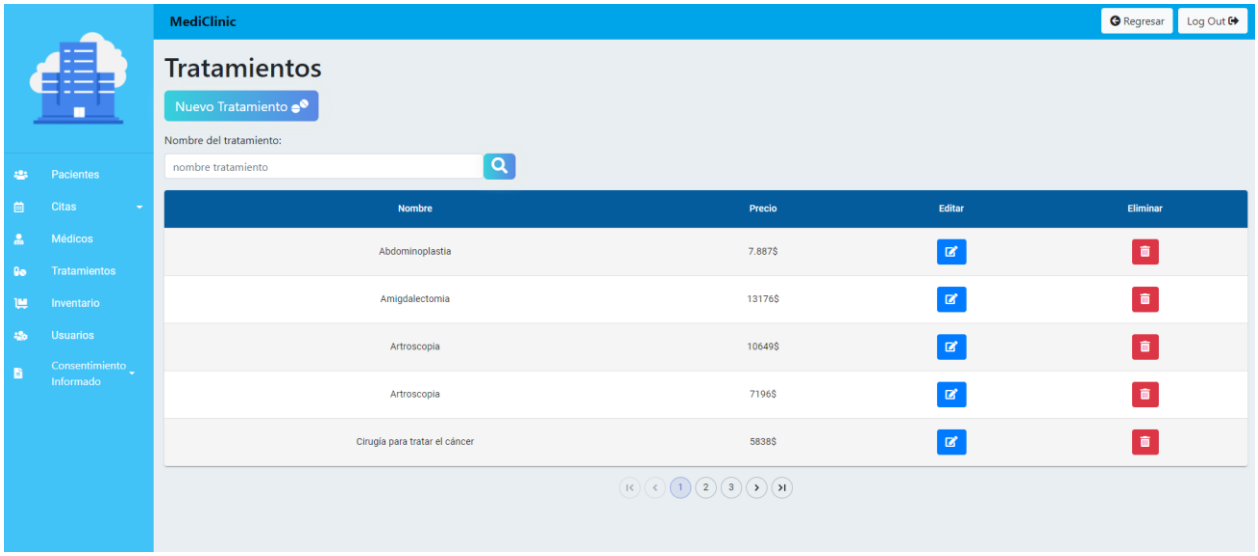
Ilustración 51 Casos de uso inventario



Vistas de la Historia de Usuario # 4

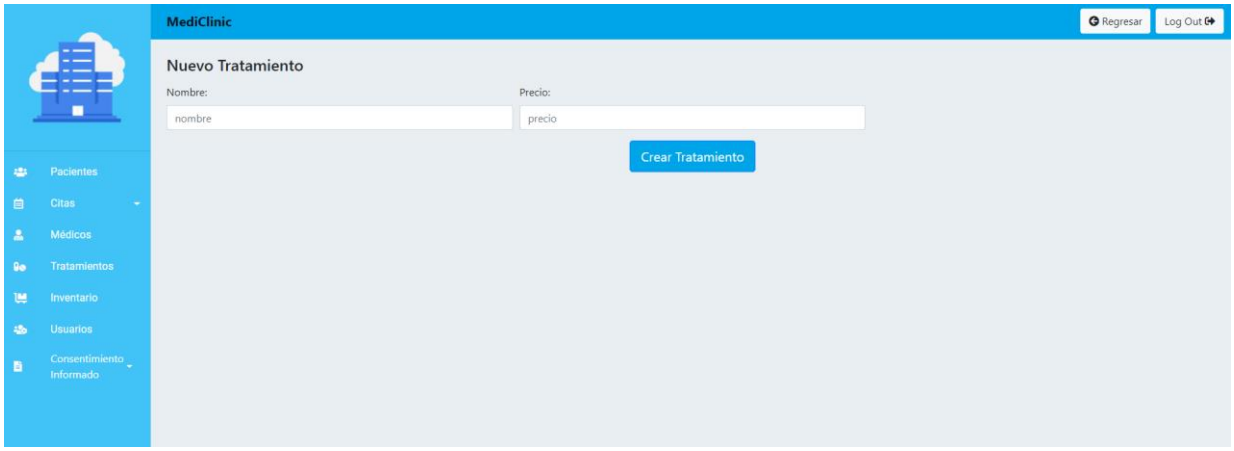
Al acceder a la página de tratamientos se muestra una tabla con todos los tratamientos que ofrece el consultorio y sus respectivos precios. Cabe recalcar que tanto usuario administrador del sistema, usuario médico y usuario administrativo, tienen acceso a los tratamientos.

Ilustración 52 Página con tabla de tratamientos



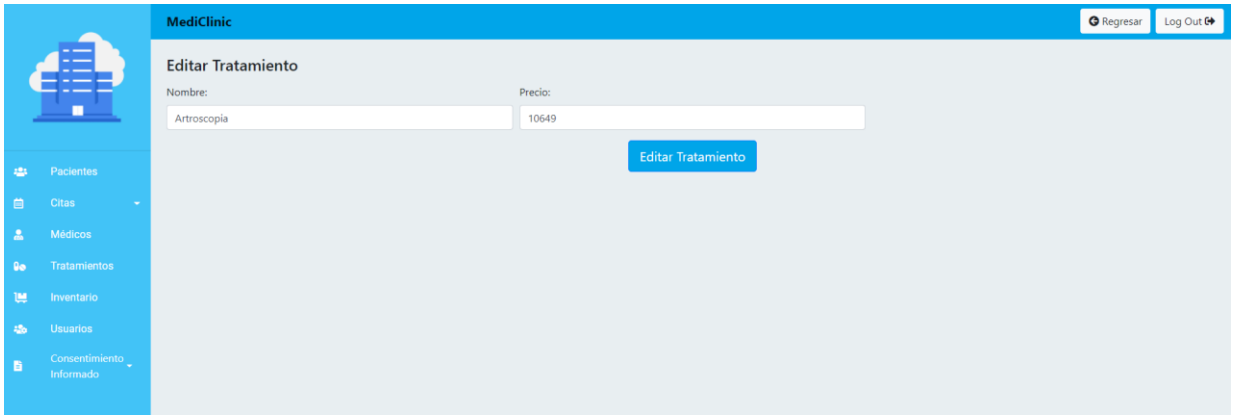
En caso de presionar el botón “nuevo tratamiento” el sistema redirecciona al usuario a una página para la creación de un nuevo tratamiento.

Ilustración 53 Página para la creación de un nuevo tratamiento



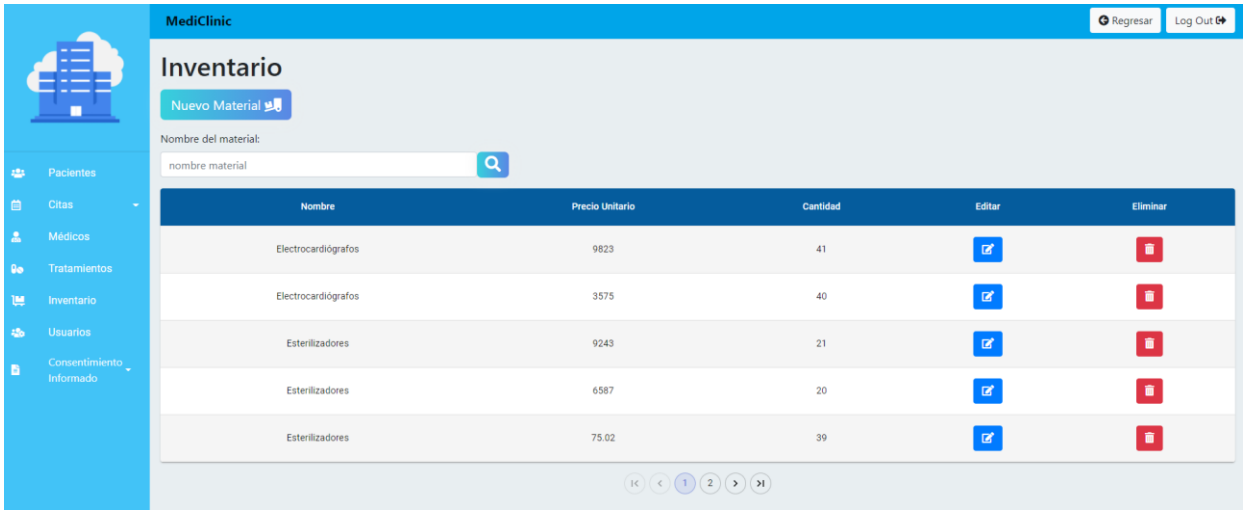
En caso de presionar el botón de edición el sistema redirecciona al usuario a una página para la edición del registro seleccionado.

Ilustración 54 Página para la edición de un tratamiento



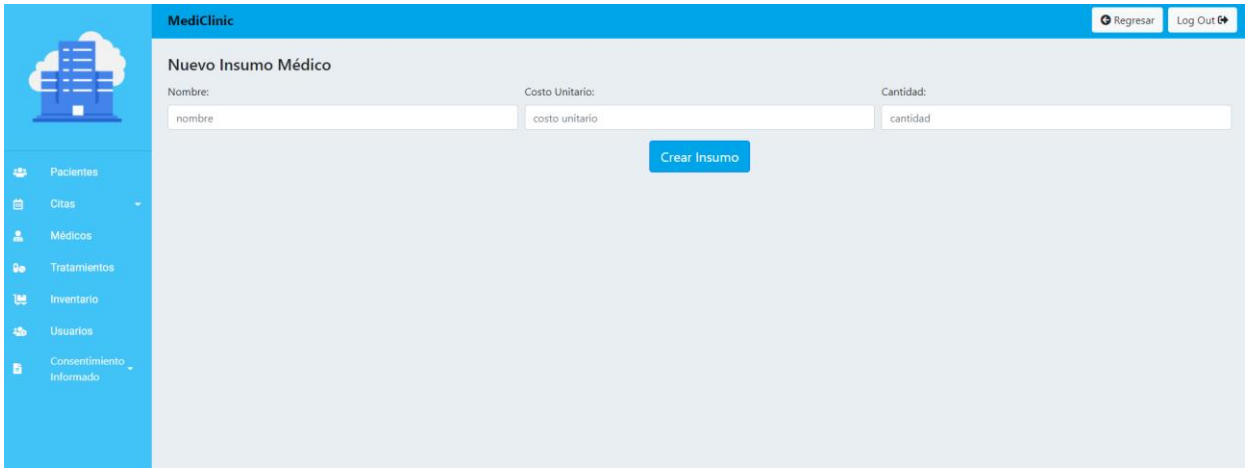
Al acceder a la página de inventario se presenta en una tabla los diferentes materiales disponibles dentro del consultorio con su precio y la cantidad.

Ilustración 55 Página con tabla de Materiales



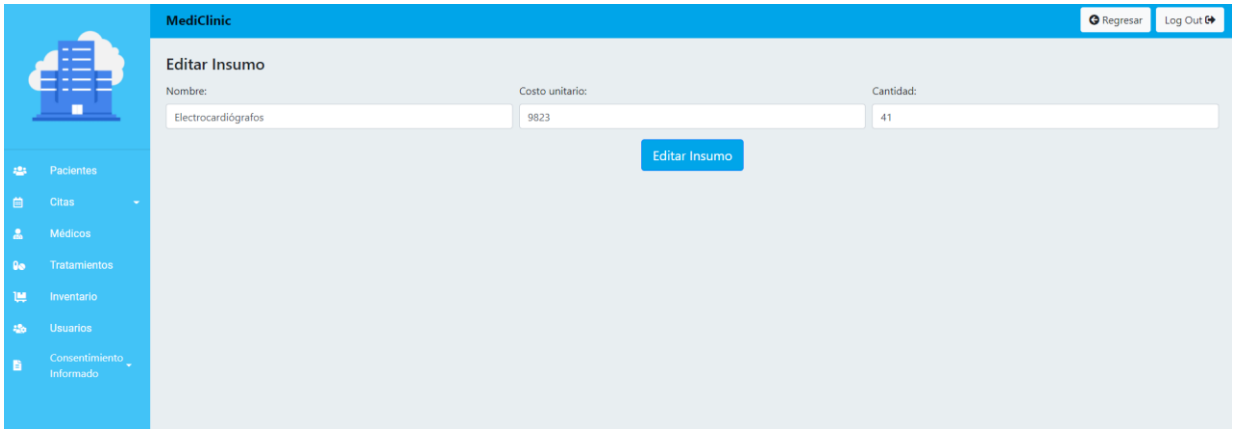
En caso de presionar el botón de “nuevo material” el sistema redirecciona al usuario a una nueva página para el ingreso de un nuevo material.

Ilustración 56 Página para la creación de un nuevo material



En caso de presionar el botón de edición el sistema redirecciona al usuario a una página para editar el registro seleccionado.

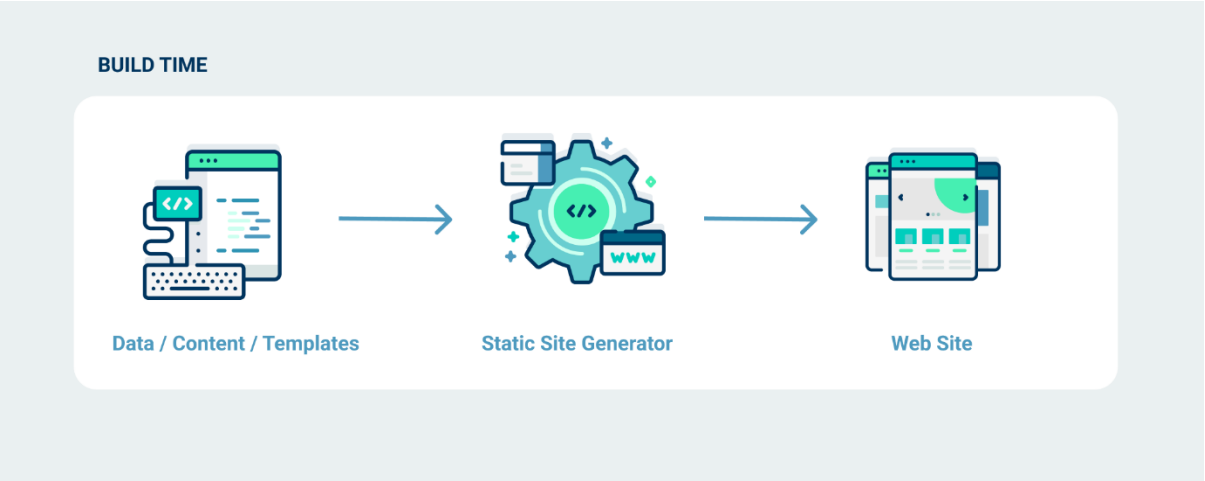
Ilustración 57 Página para la edición de un material



4.3.6.2 Codificación

Tradicionalmente las páginas web desarrolladas en JavaScript cargan su contenido y renderizan, dentro del navegador, en tiempo de ejecución, debido a esto dentro del código se definen "loaders" que muestren al usuario que una página está cargando su contenido en tiempo de ejecución. Next JS provee SSG (static site generation) lo cual es el proceso de compilación y renderizado en el momento de la compilación, es decir los componentes y contenido no cargan en tiempo de ejecución, sino que ya vienen listos desde la compilación del programa. Esto hace que la navegación de una página a otra sea demasiado veloz. Next JS aplica este concepto a todas las páginas estáticas automáticamente, y si se requiere hacer una petición a una api se puede utilizar **getStaticProps** para poblar una página con datos durante la compilación, esto da la impresión de que la página carga de inmediato.

Ilustración 58 Hawksworth, P. (2020, 14 abril). What is a Static Site Generator? And 3 ways to find the best one [Ilustración]. Netlify. https://www.netlify.com/blog/2020/04/14/what-is-a-static-site-generator-and-3-ways-to-find-the-best-one/



El sistema implementa este concepto para las páginas de creación y edición de tratamientos e inventario por lo que son más rápidas en comparación a otras páginas dentro del sistema.

4.3.6.3 Pruebas

Resultados de las pruebas unitarias del front-end y back-end:

Back-end con PHP-unit testing:

Ilustración 59 Pruebas de back-end en PHP-Unit

```
D:\Ingenieria en Sistemas\Plan de Disertación\sistemaMedico>php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\Http\Controllers\CitaControllerTest
  ⨯ cita crud

  PASS  Tests\Feature\Http\Controllers\ConsultorioControllerTest
  ⨯ consultorio crud

  PASS  Tests\Feature\Http\Controllers\EvolucionControllerTest
  ⨯ evolucion crud

  PASS  Tests\Feature\Http\Controllers\HistoriaClinicaControllerTest
  ⨯ historia clinica crud

  PASS  Tests\Feature\Http\Controllers\InventarioControllerTest
  ⨯ inventario crud

  PASS  Tests\Feature\Http\Controllers\MedicoControllerTest
  ⨯ medico crud

  PASS  Tests\Feature\Http\Controllers\NotificacionesTest
  ⨯ enviar notificacion

  PASS  Tests\Feature\Http\Controllers\PacienteControllerTest
  ⨯ paciente crud

  PASS  Tests\Feature\Http\Controllers\ServiciosControllerTest
  ⨯ servicios crud

  PASS  Tests\Feature\Http\Controllers\TratamietosControllerTest
  ⨯ tratamientos crud

  PASS  Tests\Feature\Http\Controllers\UsuarioTest
  ⨯ usuario autenticacion

  Tests:  11 passed
  Time:   1.30s
```

Se agregaron las pruebas de tratamientos crud e inventario crud las cuales prueban que las operaciones de crear, leer, editar y eliminar funcionen de manera correcta para estos recursos.

Front-end con JEST:

Ilustración 60 Pruebas unitarias con JEST

```
D:\Ingenieria en Sistemas\Plan de Disertación\sistemaMedicoUI>npm test

> sistemamedicoui@1.0.0 test D:\Ingenieria en Sistemas\Plan de Disertación\sistemaMedicoUI
> jest

 PASS  __tests__/components/Modales.test.js (12.521 s)
  Prueba de todos los modales
    √ Render del modal Información de la Evolución (8 ms)
    √ Render del modal Información del Paciente (1 ms)
    √ Render del modal de Eliminación (1 ms)
    √ Render del modal de Error (4 ms)
    √ Render del modal de Éxito (3 ms)

 PASS  __tests__/components/HistoriaClinica.test.js (15.434 s)
  <AdminLayout></AdminLayout>
    √ Render del componente que presenta la Historia Clinica de cada paciente (11 ms)

 PASS  __tests__/components/EvolucionesTable.test.js (16.019 s)
  <EvolucionesTable></EvolucionesTable>
    √ Render del componente Evoluciones Table con información (6 ms)

 PASS  __tests__/components/Servicios.test.js (16.544 s)
  <Servicios></Servicios>
    √ Render de componentes en la pagina servicios (7 ms)

 PASS  __tests__/components/TratamientosPage.test.js (16.861 s)
  <Tratamientos></Tratamientos>
    √ Render del componente TratamientosTable dentro de la página tratamientos (7 ms)

 PASS  __tests__/components/InventarioPage.test.js (16.775 s)
  <Inventario></Inventario>
    √ Render del componente InventarioTable dentro de la página Inventario (8 ms)

 PASS  __tests__/components/PacientesTable.test.js (16.333 s)
  <PacientesTable></PacientesTable>
    √ Render del componente Pacientes Table con información (7 ms)

 PASS  __tests__/components/AdminLayout.test.js (17.198 s)
  <AdminLayout></AdminLayout>
    √ Render del componente AdminLayout como wrapper (6 ms)

 PASS  __tests__/components/Calendario1.test.js (19.002 s)
  <Calendario 1></Calendario 1>
    √ Render del componente Calendario en la pagina Citas (7 ms)

 PASS  __tests__/components/Calendario2.test.js (20.04 s)
  <Calendario 2></Calendario 2>
    √ Render del componente Calendario Variante en la pagina Citas (4 ms)

 PASS  __tests__/components/LoginPage.test.js (25.075 s)
  <LoginPage></LoginPage>
    √ Render del componente Login en la pagina login_page (3 ms)

Test Suites: 11 passed, 11 total
Tests:       15 passed, 15 total
Snapshots:   0 total
Time:        31.469 s
Ran all test suites.
```

Las pruebas agregadas fueron <**Tratamientos**> e <**Inventario**> las cuales prueban, con datos, que las tablas renderizadas en las páginas se estén montando correctamente en el DOM.

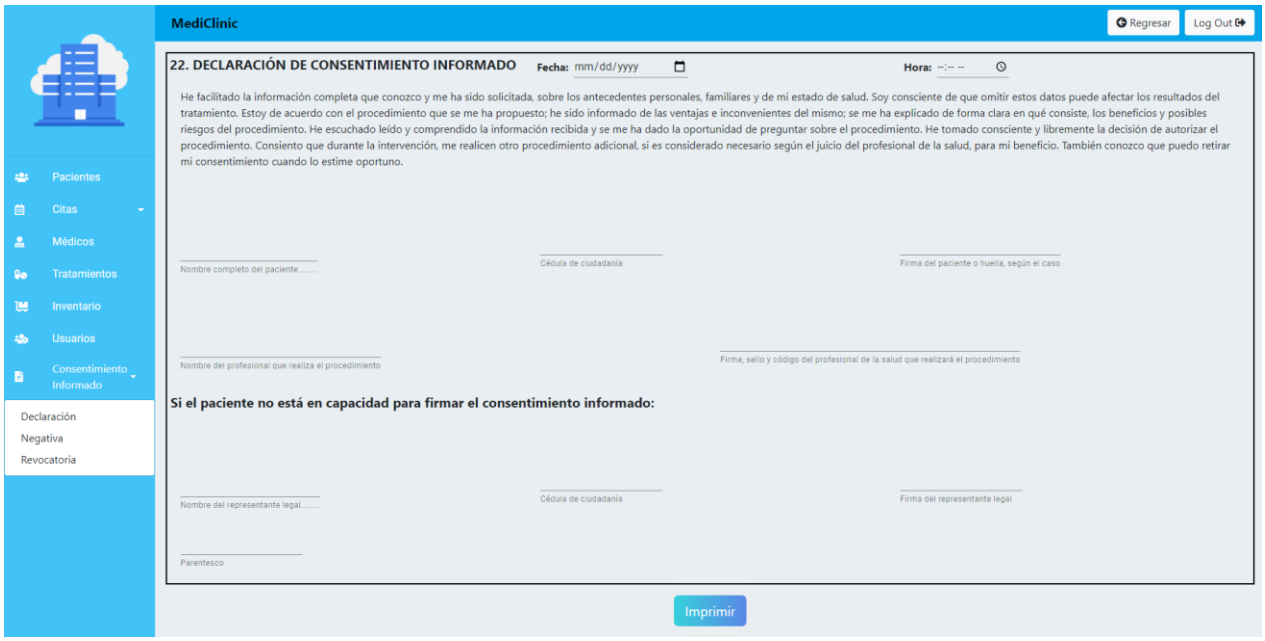
4.3.6 Iteración #7 – Historia de Usuario #5

4.3.6.1 Diseño

Vistas de la historia de Usuario #5

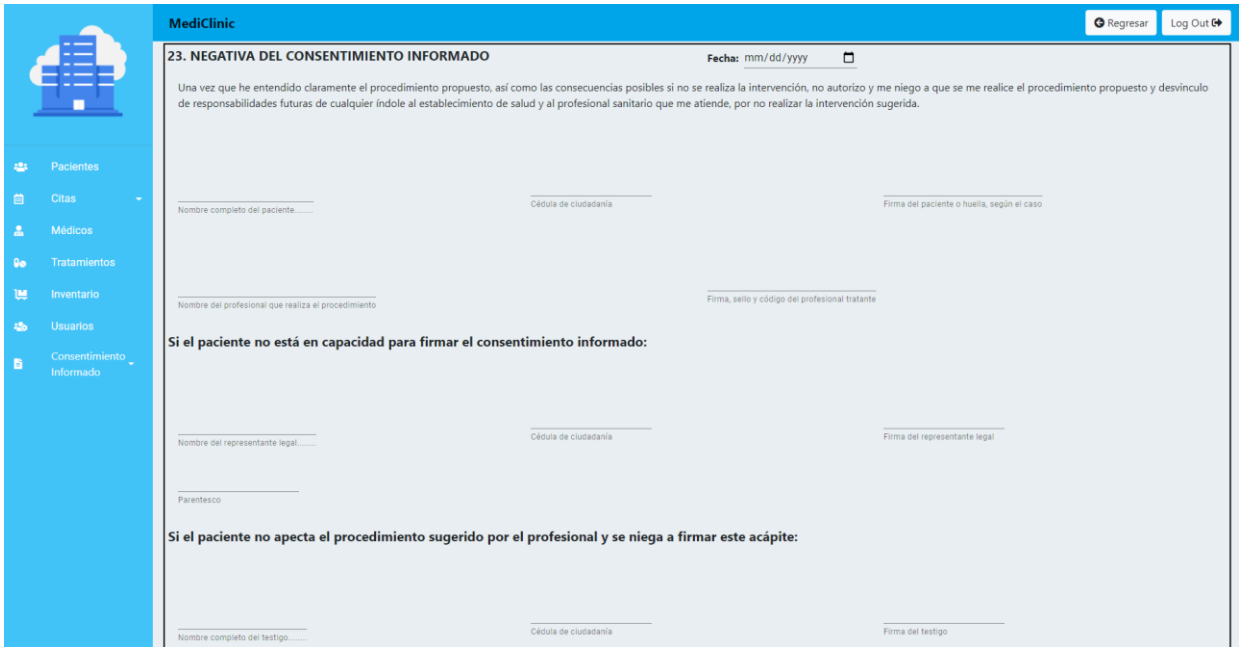
En la barra lateral izquierda para navegación existe un botón desplegable con el nombre "Consentimiento Informado", este botón despliega los 3 modelos de consentimiento según el ministerio de salud. Al presionar en declaratoria el sistema muestra la página con el formulario listo para ingresar los datos e imprimir.

Ilustración 61 Página para la declaración de consentimiento informado



En caso de presionar la Negativa el sistema presenta la página que contiene el documento listo para ingresar los datos respectivos.

Ilustración 62 Página para la negativa del consentimiento informado



En caso de presionar la Revocatoria el sistema presenta la página que contiene el documento listo para ingresar los datos respectivos.

Ilustración 63 Página para la revocatoria del consentimiento informado

4.3.6.2 Codificación

Para la creación de formularios y consentimientos se utilizaron dos librerías, Material UI (MaterialUI, 2021) y React-Bootstrap (ReactBootstrap, 2021). Estas librerías facilitan el desarrollo del front-end ya que incorporan elementos comúnmente usados en las interfaces de usuario como: entradas de texto, etiquetas de texto, áreas de texto, botones, barras de navegación, etc. Estas librerías también permiten la personalización de estilos, gracias a eso el sistema incorpora colores personalizados y estilos pensados según el diseño propuesto inicialmente.

Otra librería aplicada en el sistema fue react-to-print (NPM, 2021) la cual permite, en base al sistema de referencias que maneja ReactJS, referenciar un componente e imprimirlo directamente en pdf. Esto permite que el diseño elaborado dentro del formulario, y lo que el usuario ingrese en pantalla, sea impreso de manera exacta.

4.3.6.3 Pruebas

Resultados de las pruebas unitarias del front-end y back-end:

Back-end con PHP-unit testing:

La implementación de los formularios para los consentimientos se maneja de forma estática, es decir no se hacen llamados a la API y se crea el contenido estáticamente en el HTML. Debido a esto no fue necesario el desarrollo de pruebas en el back-end.

Front-end con JEST:

Ilustración 64 Pruebas unitarias con JEST

```
PASS  __tests__/components/PacientesTable.test.js (12.035 s)
  <PacientesTable></PacientesTable>
    √ Render del componente Pacientes Table con información (6 ms)

PASS  __tests__/components/Modales.test.js (10.571 s)
  Prueba de todos los modales
    √ Render del modal Información de la Evolución (8 ms)
    √ Render del modal Información del Paciente
    √ Render del modal de Eliminación
    √ Render del modal de Error (1 ms)
    √ Render del modal de Éxito (2 ms)

PASS  __tests__/components/EvolucionesTable.test.js
  <EvolucionesTable></EvolucionesTable>
    √ Render del componente Evoluciones Table con información (1 ms)

PASS  __tests__/components/HistoriaClinica.test.js
  <AdminLayout></AdminLayout>
    √ Render del componente que presenta la Historia Clinica de cada paciente (1 ms)

PASS  __tests__/components/Servicios.test.js (14.584 s)
  <Servicios></Servicios>
    √ Render de componentes en la pagina servicios (12 ms)

PASS  __tests__/components/InventarioPage.test.js (15.167 s)
  <Inventario></Inventario>
    √ Render del componente InventarioTable dentro de la página Inventario (6 ms)

PASS  __tests__/components/TratamientosPage.test.js (15.385 s)
  <Tratamientos></Tratamientos>
    √ Render del componente TratamientosTable dentro de la página tratamientos (6 ms)

PASS  __tests__/components/Layout.test.js (15.484 s)
  <Layout></Layout>
    √ Render del componente Layout como wrapper en la página principal (7 ms)

PASS  __tests__/components/AdminLayout.test.js (15.43 s)
  <AdminLayout></AdminLayout>
    √ Render del componente AdminLayout como wrapper (10 ms)

PASS  __tests__/components/Calendario1.test.js (17.194 s)
  <Calendario 1></Calendario 1>
    √ Render del componente Calendario en la pagina Citas (3 ms)

PASS  __tests__/components/Consentimientos.test.js (17.205 s)
  <DeclaracionesTest></DeclaracionesTest>
    √ Render de los formularios de consentimiento (6 ms)

PASS  __tests__/components/Calendario2.test.js (16.724 s)
  <Calendario 2></Calendario 2>
    √ Render del componente Calendario Variante en la pagina Citas (4 ms)

PASS  __tests__/components/LoginPage.test.js (23.367 s)
  <LoginPage></LoginPage>
    √ Render del componente Login en la pagina login_page (3 ms)

Test Suites: 13 passed, 13 total
Tests:       17 passed, 17 total
Snapshots:   0 total
Time:        29.299 s
Ran all test suites.
```

Se comprueba que los diferentes componentes estáticos para los formularios de consentimiento se hayan montado adecuadamente. Se agrego la prueba <**DeclaracionesTest**> que verifica que en las 3 páginas de consentimientos se monten los formularios.

Consideraciones Finales

Ilustración 65 Modelo final de base de datos

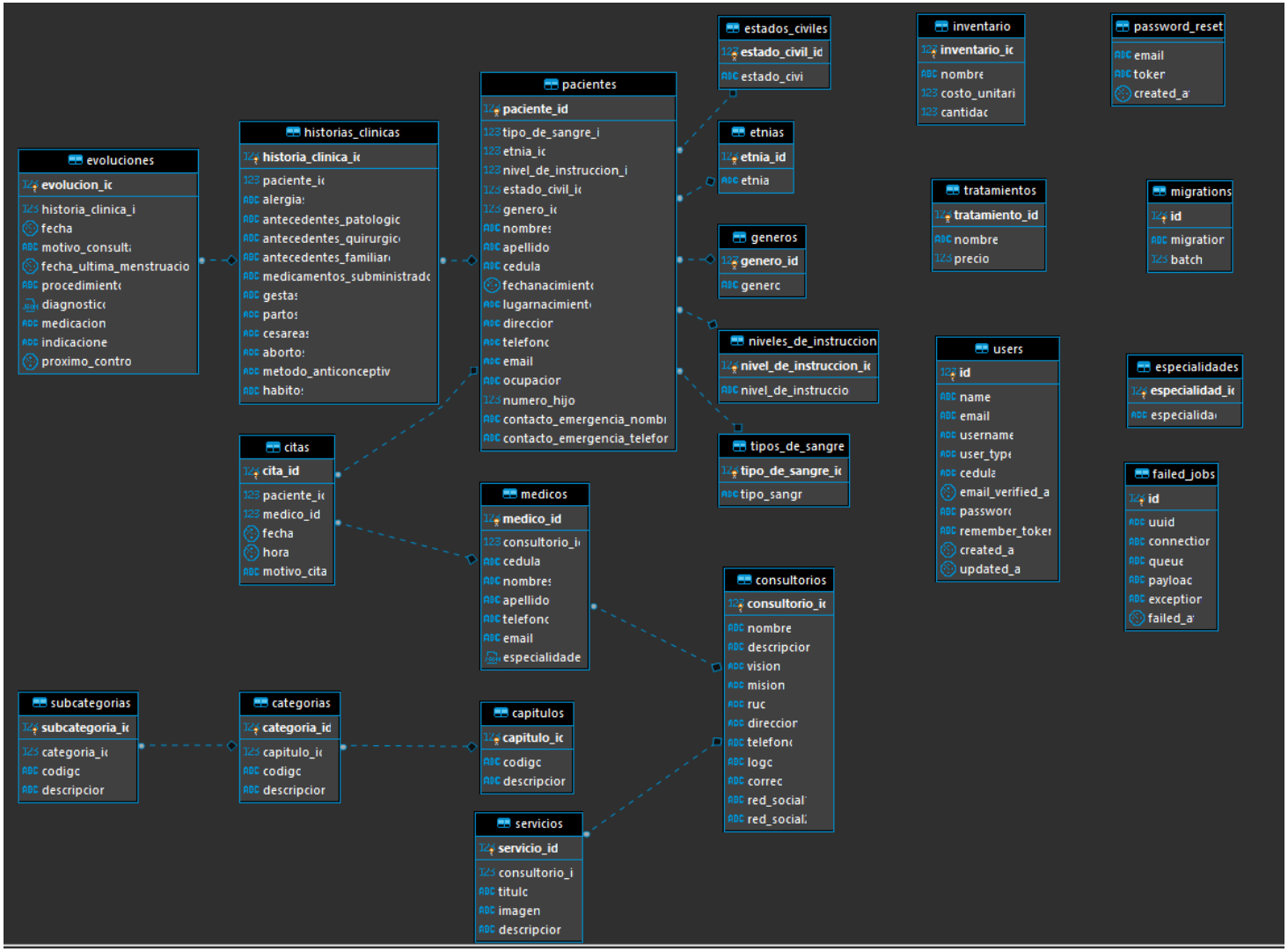


Ilustración 66 Reporte final de los componentes del Front-end en JEST

All files

74.16% Statements 198/267 **33.57%** Branches 48/143 **61.54%** Functions 64/104 **75%** Lines 180/240

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
Admin/Calendario	77.14%	27/35	75%	9/12	64.29%	9/14	76.47%	26/34
Admin/CalendarioVariante	83.87%	26/31	75%	6/8	64.29%	9/14	82.76%	24/29
Admin/Consentimiento	100%	3/3	100%	0/0	100%	3/3	100%	3/3
Admin/Drawer	15.38%	2/13	0%	0/12	0%	0/3	18.18%	2/11
Admin/HistoriaClinica	90%	9/10	50%	2/4	75%	3/4	90%	9/10
Admin/Modales	84.62%	11/13	50%	6/12	71.43%	5/7	84.62%	11/13
Admin/NavBar	12.5%	1/8	0%	0/6	0%	0/5	14.29%	1/7
Admin/Tables	84.21%	32/38	100%	0/0	72.73%	16/22	80%	24/30
CommonStyles	93.75%	15/16	85.71%	6/7	100%	3/3	93.75%	15/16
Error	33.33%	1/3	0%	0/3	0%	0/1	33.33%	1/3
Footer	33.33%	2/6	0%	0/8	0%	0/1	33.33%	2/6
Layouts	70.83%	17/24	50%	4/8	57.14%	4/7	81.25%	13/16
Loader	100%	4/4	50%	1/2	100%	2/2	100%	4/4
Log In	92.86%	13/14	45%	9/20	50%	1/2	92.86%	13/14
NavBar	30%	3/10	0%	0/14	0%	0/5	33.33%	3/9
Servicios	100%	5/5	50%	5/10	100%	2/2	100%	5/5
utils	79.41%	27/34	0%	0/17	77.78%	7/9	80%	24/30

5. Conclusiones y Recomendaciones

Conclusiones:

- El "prototipo de sistema de información para la gestión de consultorios médicos de diferentes especialidades" se desarrolla con el fin de automatizar gran parte de los procesos cotidianos que involucran las actividades médicas, proveer una herramienta eficiente en el manejo de la información sensible de los pacientes y facilitar el manejo de las citas dentro de un consultorio.
- La metodología ágil Extreme Programming XP es ideal para el desarrollo del sistema ya que sus pasos para el desarrollo de software, basados en iteraciones e historias de usuario, son claros, sencillos de comprender y se adecuan a las necesidades del proyecto que comprende factores como: grupo de 1 desarrollador, corto tiempo de desarrollo para el sistema y opiniones de diversos profesionales para los requerimientos.
- El sistema médico desarrollado es escalable y permite que sus funcionalidades sean fácilmente extendidas, esto se debe a que su implementación toma en cuenta una estructura basada en microservicios, ya que está compuesto por diferentes lenguajes y herramientas, lo que permite que sus funcionalidades o diseños hacia otras plataformas (móvil) sean extendidas.
- El time to first byte (**TTFB**), o tiempo para el primer byte, del sistema es corto lo que hace que la navegación a través de la aplicación web desarrollada sea veloz, esto es gracias a la implementación de renderizado por servidor (**SSR**) y generación de sitios estáticos implementado para el front-end.
- Cualquier modificación o cambio elaborado en el código puede ser verificado y comprobado por medio de las pruebas unitarias elaboradas tanto para el back-end como para el front-end. Esto es gracias a la práctica TDD (test driven development) o desarrollo por pruebas que valida las operaciones CRUD para el back-end y el montaje de componentes para el front-end.
- Ya que el sistema utiliza una gran cantidad de datos generados por "**faker**" de Laravel, se puede concluir que el software funcionará de manera adecuada en un ambiente de producción.
- Gracias a los estándares de nombramiento de carpetas, comentarios, nombramiento de variables e indentación del código es posible concluir que cualquier desarrollador, que conozca los lenguajes y frameworks usados en el sistema, puede modificar o extender las funcionalidades del código.
- Se puede concluir que el despliegue a producción del sistema se puede elaborar de forma fácil, rápida y sencilla hacia plataformas como servicio (PAAS) utilizando GitHub, esto es gracias a los frameworks utilizados para el desarrollo del sistema.

Recomendaciones:

- Si se requiere efectuar cualquier cambio sobre el código se recomienda una bifurcación o fork al repositorio principal para trabajar colaborativamente de forma adecuada y evitar errores que afecten al sistema.
- En caso de que el sistema comience a tener muchas visitas y manejar gran cantidad de datos de pacientes se recomienda actualizar el plan de hosting en las diferentes plataformas como servicio utilizadas para evitar cuellos de botella.
- Se recomienda seguir los estándares y la indentación dentro del código para mantenerlo entendible y claro.
- En caso de utilizar constantemente los recordatorios por email y mensajes móviles se recomienda actualizar los planes de pago con los proveedores de correos y mensajes SMS.
- En caso de extender alguna funcionalidad dentro del código se recomienda agregar una nueva historia de usuario y una nueva iteración para tener documentado el cambio de forma adecuada.
- En caso de implementar una nueva versión del sistema se recomienda agregar que los usuarios pacientes existentes en el sistema puedan obtener sus propias citas en base a los doctores disponibles.
- Se recomienda que la facultad de Sistemas y Computación desarrolle más materias que permitan el aprendizaje de tecnologías en vanguardia para el desarrollo, testeo y despliegue de software.

Bibliografía

- Díaz de León-Castañeda, C. (2019). Salud electrónica (e-Salud): un marco conceptual de implementación en servicios de salud. Gaceta medica de Mexico, 155(2), 176-183.
- Laravel. (2021). Laravel Docs. 2021, de Laravel Sitio web: https://laravel.com
- React.js. (2021). React.js. 2021, de React.js Sitio web: https://es.reactjs.org
- Navarro Cadavid, Andrés; Fernández Martínez Daniel; Morales Jonathan. (2013). Revisión de metodologías ágiles para el desarrollo de software. 2021, de Universidad Autónoma del Caribe Colombia Sitio web: https://www.redalyc.org/pdf/4962/496250736004.pdf
- Faiza Anwer , Shabib Aftab , Syed Shah Muhammad Shah and Usman Waheed. (2017). Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum. International Journal of Computer Science and Telecommunications, Vol 8, 1-2.
- Ing. José Joskowicz . (2008). Reglas y Prácticas en eXtreme Programming. 2021, de Universidad de Vigo, España Sitio web: https://d1wqtxts1xzle7.cloudfront.net/31398587/xp_-_jose_joskowicz.pdf?1371313213=&response-content-disposition=inline%3B+filename%3Dxp_jose_joskowicz.pdf&Expires=1615513380&Signature=b8z3SrI30Y1CTPiAFngc6cOcuQ3rrHcnlnSFz2lO8HKJRknE2dX9r6~ht9IXR9imfwps81bMp74vJR4BL7wMWrqBHzs5dsWtZVx-~YGXnZM3QsluQvwl~UgknO-Xkm-C4RT3GK4nlMLDrD16n3dxKqj5DbDEM-PhrVW5TgEdKXK3UbZyhjTZAH0fIiFMkaqFwH999WMoRHX6JSFyKY-dk0sG5fXLKOf8AsBZVk3K9up1jpLaGMYdn5ZTsXMqVmaF6MFM-jrRCi12soo7nPDwPNp-FxYhPg-lb09GISTkIX9g5XDcUlaR7hbcf8CSJ-bXBoIokZVBpVJyy1fEnkHENA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
- Stackoverflow. (2019). Developer Survey Results. 2021, de Stackoverflow Sitio web: https://insights.stackoverflow.com/survey/2019
- Postgresql. (2021). Postgresql. 2021, de Postgresql Sitio web: https://www.postgresql.org
- NPM. (2021). NPM. 2021, de NPM Sitio web: https://www.npmjs.com
- Jest. (2021). Jest. 2021, de Jest Sitio web: https://jestjs.io
- NextJS. (2021). NextJS. 2021, de Vercel Sitio web: https://nextjs.org

Implementación de un prototipo de sistema de información para la gestión de consultorios médicos de diferentes especialidades