UNIVERSITARIO RUMIÑAHUI

Pregrado

Carrera: Sistemas y Gestión y Data

Asignatura (UIC): Ejecución de Proyectos

Trabajo de titulación previo a la obtención del Título en: Tecnólogo Superior Universitario en Sistemas Y Gestión y Data

Tema: Desarrollo De Una Aplicación Web Para La Gestión Integral De Pacientes, Citas E Historiales Clínicos En El Consultorio Odontológico Perfect Smile.

Autor/s: Michael Eduardo Guamán Tapia

Tutor/s:

Ing. Danny Santiago Páez Oscullo Msc.

Ing. Jorge Eduardo Chapaca Garzón Msc.

Fecha: Sangolquí, 05 de septiembre del 2025

Autor: Michael Eduardo Guamán Tapia

Título a obtener: Tecnólogo Universitario en Sistemas y Gestión de Data

Matriz: Sangolquí – Ecuador

Correo electrónico: michael.guaman@ister.edu.ec

Dirigido por: Danny Santiago Páez Oscullo

Título/s: Ingeniero en Computación Gráfica

Máster en Lógica, Computación e Inteligencia Artificial

Matriz: Sangolquí -Ecuador

Correo electrónico: danny.paez@ister.edu.ec

Dirigido por: Jorge Eduardo Chapaca Garzón

Título/s: Ingeniero de Sistemas

Magister en Gerencia de Sistemas y tecnología empresarial

Gestión administrativa

Matriz: Sangolquí -Ecuador

Correo electrónico: jorge.chapaca@ister.edu.ec

Guamán Tapia Michael Eduardo

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN INTEGRAL DE PACIENTES, CITAS E HISTORIALES CLÍNICOS EN EL CONSULTORIO ODONTOLÓGICO PERFECT SMILE.

CARTA DE CESIÓN DE DERECHOS DEL TRABAJO DE TITULACIÓN

CT-ANX-2025-ISTER-2-2.3

Sangolquí, (05) de (septiembre) del 2025

MSc. Elizabeth Ordoñez
DIRECTORA DE DOCENCIA

MSc. Mónica Loachamín
COORDINADORA DE TITULACIÓN

INSTITUTO SUPERIOR TECNOLÓGICO RUMIÑAHUI CON CONDICIÓN DE UNIVERSITARIO

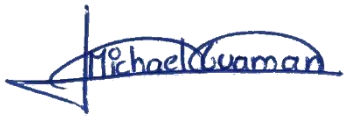
Presente

Por medio de la presente, yo, MICHAEL EDUARDO GUAMÁN TAPIA declaro y acepto en forma expresa lo siguiente: Ser autor del trabajo de titulación denominado DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN INTEGRAL DE PACIENTES, CITAS E HISTORIALES CLÍNICOS EN EL CONSULTORIO ODONTOLÓGICO PERFECT SMILE., de la Tecnología Superior Universitaria Sistemas y Gestión de Data; y a su vez manifiesto mi voluntad de ceder al Instituto Superior Tecnológico Rumiñahui con condición de Universitario los derechos de reproducción, distribución y publicación de dicho trabajo de titulación, en cualquier formato y medio, con fines académicos y de investigación.

Esta cesión se otorga de manera no exclusiva y por un periodo indeterminado. Sin embargo, conservo los derechos morales sobre mi obra.

En fe de lo cual, firmo la presente.

Atentamente,



MICHAEL EDUARDO GUAMÁN TAPIA
C.I.: 1723856280

FORMULARIO PARA ENTREGA DE PROYECTOS EN BIBLIOTECA INSTITUTO SUPERIOR TECNOLÓGICO RUMIÑAHUI CON CONDICIÓN DE UNIVERSITARIO

CT-ANX-2025-ISTER-3

CARRERA: TECNOLOGÍA UNIVERSITARIA EN SISTEMAS Y GESTIÓN DE DATA
AUTOR/ES: MICHAEL EDUARDO GUAMÁN TAPIA
TUTOR (METODOLÓGICO Y ACADÉMICO): DANNY SANTIAGO PAEZ OSCULLO – JORGE EDUARDO CHAPACA GARZÓN
CONTACTO ESTUDIANTE: 0992239775
CORREO ELECTRÓNICO: michael.guaman@ister.edu.ec
TEMA: DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN INTEGRAL DE PACIENTES, CITAS E HISTORIALES CLÍNICOS EN EL CONSULTORIO ODONTOLÓGICO PERFECT SMILE
OPCIÓN DE TITULACIÓN: UNIDAD DE INTEGRACIÓN CURRICULAR
RESUMEN EN ESPAÑOL:

El proyecto que estamos desarrollando busca crear una aplicación web para el consultorio Perfect Smile. La idea principal es mejorar cómo se gestionan los pacientes, las citas y sus historiales clínicos. Hoy en día, casi todo se hace a mano, con papeles y agendas, lo que provoca problemas como pérdida de información, datos duplicados y atención más lenta para los pacientes. Esto no solo complica el trabajo administrativo, sino que también afecta la calidad del servicio.

Para solucionar esto, queremos una herramienta tecnológica que digitalice y automatice los procesos del consultorio. La propuesta es un sistema web usando la arquitectura MVC (Modelo-Vista-Controlador), con Laravel en el backend, PostgreSQL como base de datos y Bootstrap

para que la interfaz sea moderna y adaptable a cualquier dispositivo. Con esta combinación buscamos que el sistema sea seguro, fácil de mantener y escalable.

La aplicación permitirá registrar pacientes de manera ordenada, gestionar y agendar citas odontológicas, y mantener los historiales clínicos actualizados. Todo esto ayudará a organizar mejor el consultorio, reducir tiempos de gestión, minimizar errores humanos y facilitar el acceso a la información. En pocas palabras, el sistema hará que la atención sea más ágil, confiable y profesional, beneficiando tanto al personal como a los pacientes.

PALABRAS CLAVE: gestión odontológica, aplicación web, Laravel, PostgreSQL, citas médicas, historiales clínicos.

ABSTRACT:

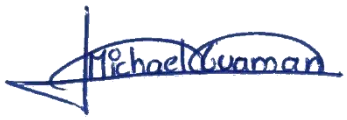
The project we are developing aims to create a web application for the Perfect Smile practice. The main idea is to improve how patients, appointments, and their medical records are managed. Today, almost everything is done manually, using paperwork and calendars, which leads to problems such as lost information, duplicate data, and slower patient care. This not only complicates administrative work but also affects the quality of service.

To solve this, we want a technological tool that digitizes and automates the practice's processes. The proposal is a web system using the MVC (Model-View-Controller) architecture, with Laravel as the backend, PostgreSQL as the database, and Bootstrap to make the interface modern and adaptable to any device. With this combination, we aim for a secure, easy-to-maintain, and scalable system.

The application will allow for orderly patient registration, management and scheduling of dental appointments, and keeping medical records up-to-date. All of this will help better organize the practice, reduce processing times, minimize human errors, and facilitate access to information.

Simply put, the system will make care more efficient, reliable, and professional, benefiting both staff and patients.

PALABRAS CLAVE: dental management, web application, Laravel, PostgreSQL, medical appointments, medical records.



Firma del Estudiante
Michael Eduardo Guamán Tapia
C.I. : 1723856280

SOLICITUD DE PUBLICACIÓN DEL TRABAJO DE TITULACIÓN

CT-ANX-2025-ISTER-4

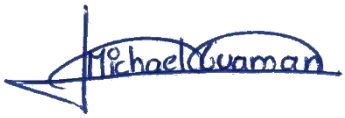
Sangolquí, (05) de (septiembre) del 2025

Sres.-
INSTITUTO SUPERIOR TECNOLÓGICO RUMIÑAHUI CON CONDICIÓN DE UNIVERSITARIO

Presente

Yo MICHAEL EDUARDO GUAMÁN TAPIA, con C.I.: 1723856280 alumno de la Carrera SISTEMAS Y GESTIÓN DE DATA, cedo al INSTITUTO SUPERIOR TECNOLÓGICO RUMIÑAHUI CON CONDICIÓN DE UNIVERSITARIO, los derechos de publicaciones del presente trabajo de Titulación en el Repositorio Institucional para hacer uso de todos los contenidos con fines estrictamente académico o de investigación.

Atentamente,



_____
Firma del Estudiante
Michael Eduardo Guamán Tapia
C.I.: 1723856280

AGRADECIMINETO

Quiero dedicar este logro primero que nada a mi familia.

A mis padres, que con su esfuerzo callado y su amor incondicional me enseñaron que los sueños se construyen paso a paso, con paciencia y confianza. A mis hermanos y abuelos, por ser ese apoyo que nunca duda y siempre anima.

También dedico estas páginas a mis amigos, esos compañeros de vida que se volvieron cómplices en las madrugadas de estudio, en los cafés reparadores y en las risas que aliviaron el cansancio. Ustedes convirtieron este camino en una experiencia compartida y memorable.

Y no puedo olvidar dedicarle este esfuerzo también a mí mismo, a ese yo que a veces dudaba, pero que siempre volvía a intentarlo, a quien aprendió que crecer duele a veces, pero vale la pena.

Esta victoria es tuya tanto como de quienes te acompañaron.

DEDICATORIA

Agradezco profundamente a todas las personas que hicieron posible que este proyecto llegara a su fin. En especial:

A mi familia, por su apoyo infinito. A mis padres, por creer en mí incluso cuando yo no lo hacía. Su ejemplo de resiliencia y trabajo honesto es la base sobre la cual construyo todo lo que hago.

A mis profesores y mentores, por compartir no sólo conocimiento, sino también su tiempo y experiencia. Gracias por las correcciones, las sugerencias y, sobre todo, por mostrarme que el aprendizaje va más allá de lo académico.

A mis amigos y compañeros de clase, por esas largas jornadas de estudio en grupo, los repasos de última hora y los ánimos cuando el estrés apretaba. No hubiera sido lo mismo sin ustedes.

A quienes, de una forma u otra, me regalaron una palabra de aliento en los momentos clave.

A los que escucharon, leyeron, sugirieron y acompañaron.

Este trabajo es también el resultado de su compañía.

Gracias, de verdad.

Sin cada uno de ustedes, este logro no sería igual.

Contenido

INDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

RESUMEN

El fin de este trabajo es crear una aplicación en la web para el consultorio dental “Perfect Smile”. Esta herramienta dejará mejorar la atención completa a los pacientes, ayudando con programar citas y ver los registros clínicos. Ahora, el consultorio usa sus datos por medio de papeles, lo que lleva desorden y pérdida de información.

Ante esta situación, la puesta en funcionamiento de un sistema en la web se plantea como una buena alternativa. Para su creación se usa el patrón MVC con el lenguaje Laravel; además, se tiene una base datos en PostgreSQL y un diseño que sigue Bootstrap.

Con esta idea se desea mejorar mucho la organización interna del consultorio, reducir el tiempo en los trámites y dar un acceso fácil y seguro a los datos. En consecuencia, se intenta asegurar un servicio de dentista más rápido, ordenado y profesional.

Palabras clave: gestión odontológica, aplicación web, Laravel, PostgreSQL, citas médicas, historiales clínicos.

CAPITULO I

1. Introducción

En los últimos años, los avances tecnológicos han tenido un gran impacto en el área de la salud, impulsando la digitalización de numerosos procesos clínicos y administrativos. Gracias a esto, se ha logrado una mayor eficiencia en la gestión médica, una mejor atención al paciente y una protección más segura de los datos clínicos. Hoy en día, los sistemas de información en salud, como las historias clínicas electrónicas, son herramientas muy importantes tanto en hospitales como en consultorios pequeños o medianos, porque permiten acceder rápido a la información y ayudan a tomar mejores decisiones médicas.

Aun así, muchos consultorios odontológicos siguen usando papel y lápiz para organizar citas, guardar historiales y registrar datos de los pacientes. Esto es diferente a lo que ocurre en otros países de la región, donde la tecnología ya ha traído beneficios claros. Por ejemplo, en México, Argentina y Perú se han usado sistemas informáticos que facilitan el intercambio de información médica, mejoran los procesos administrativos y aumentan la calidad de la atención. En Perú, el proyecto "Wawared-Perú" ayudó a mejorar la salud materna usando historias clínicas electrónicas, y el Ministerio de Salud también ha creado normas para asegurar que estos registros se guarden de forma correcta (Adiel Joshua Preciado Rodríguez, 2021).

El consultorio odontológico Perfect Smile enfrenta la misma situación, pues sus procesos siguen siendo manuales. Esto provoca desventajas como pérdida de información, dificultades para acceder a los historiales médicos, errores en la programación de citas, baja eficiencia en la atención y riesgos en la seguridad de los datos sensibles al estar almacenados en papel. Estas limitaciones justifican la necesidad de modernizar la gestión mediante una solución tecnológica que permita digitalizar la administración de pacientes, citas e historiales clínicos.

La idea de este proyecto es crear una aplicación web que ayude a automatizar estos procesos, haciendo todo más seguro, rápido y eficiente. Con este sistema, el personal del consultorio podrá trabajar de manera más ordenada y los pacientes recibirán una atención más ágil y confiable. Además, diferentes estudios han resaltado cómo la integración de tecnologías en odontología contribuye a mejorar el acceso y la gestión de los servicios, tal como ocurrió en el siglo XX con la creación de unidades odontológicas móviles en países como Colombia para atender zonas rurales (Albarracín Valderrama Astrid Eliana, 2021).

El proyecto se plantea con una arquitectura Modelo-Vista-Controlador (MVC), utilizando Laravel como framework principal por su robustez y escalabilidad, PostgreSQL como gestor de base de datos por su seguridad y rendimiento, y Bootstrap para diseñar una interfaz moderna, atractiva y adaptable a diferentes dispositivos. De esta manera, no solo se atiende una necesidad específica del consultorio, sino que también se sigue la tendencia global de modernización tecnológica en salud, aplicando en un caso real conocimientos de desarrollo web, bases de datos, usabilidad y seguridad informática.

2. Antecedentes

En investigaciones anteriores se han creado sistemas informáticos para consultorios odontológicos con el fin de mejorar cómo se registran los pacientes, se programan las citas y se manejan los historiales clínicos. Un ejemplo es el centro Dental Group en Guayaquil, que implementó una plataforma para automatizar procesos en servicios como odontología general, estética y cirugía. Este sistema permitió registrar y controlar los tratamientos, generar reportes clínicos y organizar las citas, incluso enviando notificaciones automáticas por correo. Se desarrolló usando una arquitectura cliente-servidor con Odoo 8, PostgreSQL y herramientas como iReport, lo que facilitó ver los historiales médicos y odontogramas. La experiencia de este centro muestra que la tecnología ayuda mucho a mejorar tanto la gestión clínica como la administrativa, y por eso es importante crear un sistema web

adaptado a las necesidades del consultorio Perfect Smile (Jeannette Elizabeth Sanunga Totoy, 2018).

3. Problema

En el consultorio odontológico Perfect Smile la gestión de pacientes, citas e historiales clínicos todavía se realiza de forma manual, mediante registros en papel y el uso de agendas físicas. Este método ha generado varios inconvenientes, entre ellos la pérdida de información, la dificultad para localizar rápidamente los datos de cada paciente, errores en la programación de citas y complicaciones al revisar los historiales clínicos. Estos problemas afectan de manera directa la calidad del servicio que se ofrece, ya que se invierte demasiado tiempo en tareas administrativas, reduciendo el tiempo disponible para la atención odontológica.

Como consecuencia, algunos pacientes han mostrado inconformidad por los retrasos y errores, lo que afecta tanto su confianza en el consultorio como los ingresos económicos. A esto se suma el riesgo de extraviar documentos o que se deterioren por factores como incendios o humedad, ya que no existen respaldos digitales.

Frente a estas dificultades surge la necesidad de implementar una solución tecnológica que facilite la organización de la información, agilice la atención, reduzca los errores y brinde mayor seguridad en el manejo de los datos. Por ello, se propone el desarrollo de una herramienta informática que permita optimizar la gestión integral del consultorio y mejorar la experiencia tanto del personal como de los pacientes.

4. Justificación

El desarrollo de una aplicación web para gestionar pacientes, citas e historiales clínicos en el consultorio odontológico Perfect Smile se justifica por la necesidad de modernizar y optimizar los procesos que actualmente se hacen de manera manual. Contar con una solución tecnológica permitirá automatizar tareas, reducir errores, disminuir los tiempos de espera y

asegurar que la información médica se almacene de forma segura. Además, el uso de herramientas modernas como Laravel, PostgreSQL y Bootstrap brinda una base sólida y escalable para construir un sistema web eficiente y seguro. La arquitectura MVC ayuda a organizar el código de manera ordenada, facilitando su mantenimiento y posibles mejoras futuras.

4.1. Justificación Social

Desde el punto de vista social, este proyecto busca mejorar la atención odontológica ofrecida a la comunidad. Tener una herramienta digital que agilice la atención y permita un seguimiento adecuado de la salud bucal de los pacientes hace que el servicio sea más humano, eficiente y organizado. También mejora la experiencia del paciente, reduce el tiempo de espera y fortalece la relación entre el profesional de la salud y el usuario, garantizando un acceso más rápido y confiable a su historial clínico. Además, digitalizar los procesos ayuda a la sostenibilidad, ya que disminuye el uso de papel y, con ello, el impacto ambiental.

4.2. Justificación Técnica

En el aspecto técnico, el proyecto permite aplicar conocimientos de desarrollo web y manejo de bases de datos en un entorno real. Laravel facilita crear una aplicación robusta y segura siguiendo el patrón MVC, PostgreSQL asegura la gestión eficiente de los datos con integridad y soporte para consultas complejas, y Bootstrap permite diseñar interfaces adaptables y agradables, mejorando la experiencia del usuario. Este sistema no solo resuelve una necesidad concreta del consultorio Perfect Smile, sino que también puede replicarse o adaptarse a otros consultorios con características similares, lo que le da escalabilidad y proyección a futuro.

5. Objetivos

5.1. Objetivo General

Desarrollar una aplicación web basada en la arquitectura MVC, utilizando Laravel, PostgreSQL y Bootstrap, para la gestión integral de pacientes, citas e historiales clínicos en el consultorio odontológico Perfect Smile, con el fin de optimizar los procesos administrativos y mejorar la calidad del servicio ofrecido a los pacientes.

5.2. Objetivos Específicos

- Analizar los procesos actuales de gestión en el consultorio Perfect Smile para identificar las necesidades funcionales y técnicas del sistema.
- Diseñar la estructura del sistema web aplicando el modelo MVC para garantizar una separación clara entre la lógica de negocio, la presentación y la gestión de datos.
- Implementar el backend del sistema utilizando el framework Laravel y PostgreSQL como base de datos para asegurar la integridad, seguridad y eficiencia en el manejo de la información clínica.
- Desarrollar una interfaz web responsiva y fácil de usar con Bootstrap, que facilite la interacción del personal administrativo y clínico con el sistema.
- Validar la funcionalidad y usabilidad de la aplicación web mediante pruebas técnicas y de usuario para asegurar que cumpla con los requerimientos del consultorio.

6. Alcance

Este proyecto tiene como finalidad desarrollar una aplicación web para optimizar la gestión del consultorio odontológico Perfect Smile. La herramienta centralizará y organizará toda la información relacionada con pacientes, citas e historiales clínicos, haciendo que los procesos administrativos y clínicos sean más ágiles y seguros. Se busca que el personal pueda

trabajar de manera más ordenada y eficiente, reduciendo errores y mejorando la calidad del servicio.

La aplicación permitirá llevar un registro completo de cada paciente, programar y hacer seguimiento a sus citas sin conflictos de horario, almacenar toda la información clínica relevante de manera estructurada (incluyendo observaciones, tratamientos y evolución), y generar reportes detallados que faciliten el análisis y la continuidad en la atención. De este modo, el consultorio contará con un control más preciso sobre cada caso, lo que se traducirá en un mejor servicio para los pacientes.

Cabe destacar que el sistema no incluirá funcionalidades asociadas a facturación, pagos, integración con aseguradoras, agendamiento de citas en línea por parte de los pacientes, ni envío automatizado de recordatorios por correo o SMS. El alcance se limita exclusivamente a la gestión interna de pacientes, citas e historiales clínicos, con el fin de cumplir de manera efectiva y enfocada con los objetivos planteados.

7. Marcos referenciales

La implementación de sistemas de información en salud ha demostrado ser muy útil para mejorar la calidad y eficiencia de los servicios médicos. Diversos estudios muestran que digitalizar los procesos clínicos y administrativos ayuda a ofrecer una atención más rápida y segura a los pacientes. En el ámbito odontológico, el uso de tecnologías de la información permite gestionar mejor los historiales clínicos, programar citas y dar seguimiento a los tratamientos. Además, facilita que las decisiones clínicas y administrativas se tomen basándose en datos precisos y actualizados. Por otro lado, la arquitectura MVC, junto con herramientas como Laravel, PostgreSQL y Bootstrap, proporciona un entorno de desarrollo robusto y escalable, ideal para crear aplicaciones web que respondan a las necesidades específicas de los consultorios odontológicos.

8. Marcos Legales

El desarrollo e implementación de los sistemas de información en salud en Ecuador se rige por normativas y políticas que buscan proteger los derechos de los pacientes y mejorar la eficiencia del sistema de salud. Una de estas políticas es la Política Nacional de Transformación Digital del Sector Salud, establecida mediante el Acuerdo Ministerial Nro. 00068-2024. Esta política tiene como objetivo fortalecer la digitalización del sistema nacional de salud mediante el uso estratégico de tecnologías de la información y comunicación, mejorando la calidad, equidad, accesibilidad y cobertura de los servicios de salud en todo el país (Ministerio de Salud Pública (MPS), 2025).

9. Marcos Políticos Sociales

El contexto político y social en Ecuador ha promovido la modernización del sistema de salud, reconociendo que la digitalización es clave para mejorar la eficiencia y la calidad de los servicios médicos. La implementación de tecnologías de la información en salud se considera una estrategia importante para lograr una cobertura universal y equitativa(Ministerio de Salud Pública (MSP), 2024).

La Política Nacional de Transformación Digital del Sector Salud destaca la necesidad de modernizar tecnológicamente los establecimientos de salud, incluso en zonas rurales o de difícil acceso, mediante herramientas como la historia clínica electrónica, la telemedicina y sistemas de gestión hospitalaria (Hospital General Docente de Calderón, 2023).

Estos marcos políticos y sociales respaldan la relevancia de proyectos como el desarrollo de una aplicación web para la gestión integral de pacientes, citas e historiales clínicos en el consultorio odontológico Perfect Smile, asegurando que esté alineado con las políticas nacionales de salud y transformación digital.

CAPITULO II

10. Marco Teórico

Desarrollar una aplicación web para la gestión de pacientes, citas e historiales clínicos en un consultorio odontológico requiere comprender conceptos básicos sobre salud digital, organización del software y metodologías modernas de desarrollo.

10.1. Sistemas de Información en Salud (SIS)

Los Sistemas de Información en Salud (SIS) son herramientas tecnológicas que permiten recoger, organizar, guardar y distribuir información relacionada con la salud de manera eficiente y segura. Su principal objetivo es ayudar en la gestión clínica y administrativa de los centros de salud, facilitando que las decisiones se tomen con base en datos reales y actualizados. De acuerdo con un artículo revisado (Jaume Canela-Soler, 2010) estos sistemas deben integrar toda la información en un solo lugar, usando normas claras y un lenguaje común para que los datos se interpreten correctamente y sin errores.

10.2. Historia Clínica Electrónica (HCE)

La historia clínica electrónica es la versión digital de la historia clínica tradicional. Permite almacenar de forma segura diagnósticos, tratamientos, antecedentes médicos y observaciones de los pacientes. Entre sus ventajas están el acceso rápido desde cualquier dispositivo autorizado, la reducción del uso de papel, una mejor organización de la información y la prevención de pérdidas de datos por daño o extravío de documentos físicos.

Un estudio realizado en Uruguay (Juan Eduardo Gil Yacobazzo, 2018) señala que en ese país se está implementando un sistema nacional de Historia Clínica Electrónica (HCEN). Este sistema obliga a todos los prestadores de salud a contar con

su propia HCE y a compartir la información médica de los pacientes, con el fin de mejorar la continuidad y calidad de la atención.

No obstante, este tipo de iniciativas enfrenta importantes retos en cuanto a la privacidad y seguridad de los datos personales. Por ello, el artículo también revisa el marco legal existente que protege la información médica, especialmente en lo relacionado con su acceso, manejo y protección.

10.3. Arquitectura de Software Modelo-Vista-Controlador (MVC)

La arquitectura de software Modelo-Vista-Controlador (MVC) es una manera de organizar las aplicaciones dividiéndolas en tres partes: el Modelo, que maneja los datos y la lógica; la Vista, que muestra la información al usuario; y el Controlador, que conecta las otras dos partes y gestiona las acciones del usuario. Este modelo hace que el código sea más ordenado, fácil de entender y que se puedan hacer cambios o mejoras en el sistema sin tantas complicaciones.

Tabla 1 - Arquitectura de Software Modelo-Vista-Controlador (MVC)

Modelo	Vista	Controlador
Gestiona la lógica de datos y la interacción con la base de datos.	Se encarga de la presentación y la interfaz de usuario.	Actúa de intermediario, controlando la comunicación entre el modelo y la vista.

Esta separación ayuda a que el sistema sea más fácil de mantener, que se pueda ampliar y que sea más sencillo implementar mejoras, lo que permite que las aplicaciones sean robustas y eficientes.

10.4. Herramientas Tecnológicas

Para desarrollar el sistema web de gestión clínica y administrativa, se usaron las siguientes herramientas:

- **Laravel:** Framework de backend basado en PHP que facilita el uso de la arquitectura Modelo-Vista-Controlador (MVC), permitiendo un sistema organizado, reutilizable y seguro (Laravel, 2025).
- **PostgreSQL:** Sistema de bases de datos relacional que permite almacenar y manejar grandes volúmenes de información clínica de forma eficiente y segura (PostgreSQL Global Development Group, 2025).
- **Bootstrap:** Biblioteca de diseño web que ayuda a crear interfaces limpias, responsivas y adaptables a distintos dispositivos, mejorando la experiencia del usuario (Bootstrap, 2025).
- **JavaScript:** Lenguaje de programación que agrega dinamismo e interactividad a la interfaz, especialmente en formularios y validaciones del lado del cliente (Mozilla, 2025).
- **HTML y CSS:** Lenguajes base para estructurar y diseñar las páginas del sistema, asegurando una presentación clara y funcional.
- **Git:** Sistema de control de versiones que permite llevar un seguimiento de los cambios en el código y facilita el trabajo en equipo.(Git, 2025).

Gracias a estas tecnologías, se pudo crear una aplicación web completa que ayuda a los consultorios odontológicos a dejar de usar papel y trabajar de manera más rápida, eficiente y segura.

10.5. Definición de Variable

Variable Independiente:

Se refiere a todo lo relacionado con el diseño y funcionamiento del sistema, la facilidad de uso de la interfaz, la seguridad de la información y la integración de herramientas que permitan digitalizar y automatizar los procesos clínicos y administrativos del consultorio. En este proyecto, la variable independiente es:

"Desarrollo de una aplicación web para la gestión de pacientes, citas e historiales clínicos, usando tecnologías como Laravel, PostgreSQL y Bootstrap."

Variable Dependiente:

La variable dependiente es:

"Eficiencia en la gestión clínica y administrativa del consultorio odontológico Perfect Smile."

Esto incluye cómo se organiza y controla la información, la rapidez en manejar los datos clínicos y administrativos, la correcta administración de historiales, la disponibilidad de información de los pacientes, la reducción de errores en registros manuales, la mejora en la programación de citas y la optimización del tiempo en los procesos internos.

CAPITULO III

11. Metodología

La metodología usada en este proyecto es cuantitativa-aplicada, con un enfoque descriptivo y tecnológico, porque busca desarrollar una solución informática para un problema específico en la gestión clínica y administrativa del consultorio “Perfect Smile”. Para construir el sistema, se eligió la metodología ágil SCRUM, gracias a su flexibilidad, eficiencia y trabajo colaborativo.

SCRUM permite dividir el desarrollo en ciclos cortos llamados Sprints, lo que facilita entregar funciones del sistema de manera progresiva. Varios estudios muestran que esta metodología ayuda a organizar mejor el trabajo, permite un enfoque incremental y mejora los resultados en proyectos tecnológicos. Sin embargo, también se destaca que es importante planificar bien los proyectos y formar equipos estructurados para que SCRUM funcione correctamente (Estrada Velasco Marco Vinicio, 2021)

11.1. Fases de Desarrollo bajo Scrum

El proyecto se organizará en varias fases principales, que se desarrollarán mediante Sprints.

11.1.1. Planificación Inicial (Pre-Sprint)

Entrevistas con el personal de "Perfect Smile": Se harán reuniones para entender cómo trabajan actualmente, identificar los problemas principales y conocer sus expectativas sobre el nuevo sistema.

Análisis de documentos existentes: Se revisarán los registros en papel, agendas e historiales clínicos para ver qué información debe digitalizarse y cómo organizarla.

Definición de funcionalidades clave: Junto con el consultorio, se determinarán las funciones más importantes del sistema, como registrar pacientes, gestionar citas y acceder a los historiales clínicos.

11.1.2. Sprints Individuales

Cada Sprint tenía como objetivo entregar un conjunto de funciones que ya fueran utilizables, lo que permitió avanzar poco a poco y de forma organizada en la construcción del sistema.

Tabla 2 - Desarrollo por Sprints y Cierre del Proyecto

Etapa	Fecha	Actividades Principales
Sprint 1	17 - 30 de junio de 2025	Configuración del entorno de desarrollo, instalación de Laravel y PostgreSQL, diseño de la base de datos y sistema de autenticación con roles de usuario.
Sprint 2	1 - 14 de julio de 2025	Desarrollo del módulo de pacientes: registro, edición, validaciones, búsqueda y control de acceso según rol.
Sprint 3	15 - 28 de julio de 2025	Implementación del módulo de citas: calendario interactivo, asignación, reprogramación y cancelación de citas.
Sprint 4	29 jul - 11 ago de 2025	Creación del módulo de historiales clínicos: diagnósticos, tratamientos, observaciones y generación de reportes en PDF.
Cierre del Proyecto	12 - 31 de agosto de 2025	Validación del sistema completo, pruebas con usuarios, ajustes finales, documentación técnica (diagramas, manuales) y entrega oficial al consultorio “Perfect Smile”.

11.1.3. Cierre del Proyecto

Después de completar los cuatro Sprints, se pasó al cierre del proyecto, que incluyó la validación general del sistema, ajustes finales, documentación técnica y la entrega oficial del producto. En esta fase se hicieron pruebas para asegurarse de que todos los módulos funcionaran bien juntos y también se realizaron pruebas con el personal del consultorio para evaluar si el sistema era fácil de usar y eficiente.

Finalmente, el sistema se entregó al consultorio "Perfect Smile", junto con la base de datos configurada y una guía de usuario. Con esto, el proyecto se dio por concluido, logrando automatizar los procesos administrativos y clínicos, y contribuyendo a que la gestión del consultorio sea más organizada, segura y moderna.

11.2. Herramientas y Tecnologías Utilizadas

Para desarrollar la aplicación web del consultorio odontológico "Perfect Smile" se utilizaron varias herramientas y tecnologías que ayudaron a crear un sistema eficiente, seguro y fácil de mantener. Cada herramienta fue elegida pensando en su funcionalidad, su compatibilidad con el modelo de desarrollo ágil SCRUM y su facilidad de uso tanto en entornos académicos como profesionales.

En el backend se usó el framework Laravel, basado en PHP, que permite construir aplicaciones web siguiendo el patrón Modelo-Vista-Controlador (MVC). Laravel se eligió por su estructura clara, seguridad, soporte de la comunidad y facilidad para integrar funciones modernas como rutas, autenticación, middleware y migraciones de base de datos.

Para la base de datos se empleó PostgreSQL, un sistema relacional confiable que permitió organizar y manejar la información de manera segura. Su soporte para

integridad de datos, transacciones y consultas complejas ayudó a mantener toda la información de pacientes, citas e historiales clínicos correctamente estructurada.

En la parte visual se utilizó Bootstrap, un framework CSS con muchos componentes y estilos prediseñados que permiten crear interfaces responsivas. Gracias a esto, el sistema se puede usar fácilmente en computadoras, tabletas y móviles, mejorando la experiencia del personal del consultorio.

El desarrollo se hizo en Visual Studio Code, que facilita trabajar con Laravel, integrar Git y navegar por el proyecto. También se usaron herramientas complementarias como Draw.io para crear diagramas UML y de la base de datos.

11.3. Técnicas de Recolección de Información

Para entender cómo funciona el consultorio odontológico “Perfect Smile” y recopilar los requerimientos del sistema, se usaron tres técnicas principales: entrevistas, observación directa y análisis de documentos.

Se hicieron entrevistas semiestructuradas al personal administrativo y a la odontóloga para conocer sus necesidades, los problemas que enfrentan y sus sugerencias para mejorar la gestión de pacientes, citas e historiales clínicos.

Con la observación directa, se analizó cómo se llevan a cabo los procesos actualmente, desde la atención a los pacientes hasta el registro en papel. Por último, el análisis de los documentos permitió revisar los formularios y registros físicos usados, ayudando a definir la estructura de los datos y los módulos importantes del sistema.

Estas técnicas ayudaron a obtener información clara y confiable, necesaria para diseñar una solución tecnológica adecuada a la realidad del consultorio.

Capítulo IV

12. Arquitectura del Software

La solución desarrollada se construyó bajo una arquitectura basada en el patrón Modelo-Vista-Controlador (MVC), implementada utilizando el framework Laravel.

Se eligió este patrón porque ofrece una separación clara entre la lógica de negocio (Controlador), la presentación de la información (Vista) y la gestión de datos (Modelo). Esta división no solo mejora la organización del código, sino que también permite una mayor flexibilidad y escalabilidad a largo plazo.

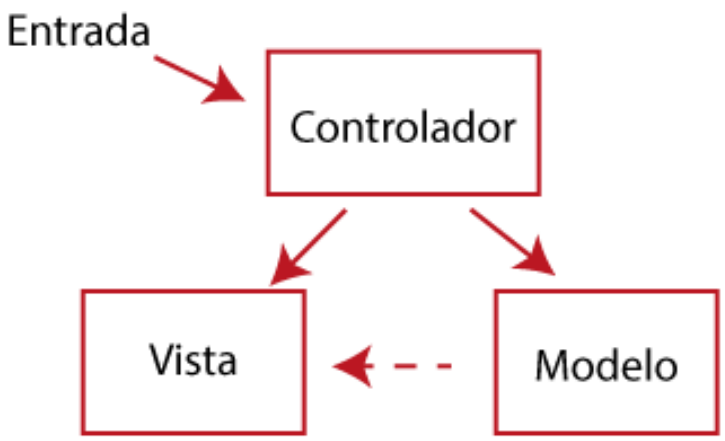


Ilustración 1- Arquitectura MVC

Entre las principales razones para adoptar esta arquitectura se destacan:

Mantenimiento simplificado

Al estar cada componente desacoplado, se pueden realizar cambios en una parte del sistema sin afectar el resto. Por ejemplo, modificar el diseño de las vistas no implica tocar la lógica de negocio.

Escalabilidad y modularidad

Permite añadir nuevos módulos como gestión de reportes, historial clínico o agendamiento de citas sin necesidad de reestructurar todo el sistema.

Reutilización de código

Funciones y controladores pueden reutilizarse en diferentes secciones, reduciendo tiempos de desarrollo.

Compatibilidad con estándares modernos

Laravel proporciona utilidades para control de rutas, seguridad (CSRF, validaciones), migraciones de base de datos y plantillas Blade, alineándose con las buenas prácticas de desarrollo web.

En esta arquitectura, el Controlador se encarga de recibir las solicitudes del usuario y ejecutar la lógica necesaria, el Modelo interactúa con la base de datos para obtener o almacenar información, y la Vista se encarga de mostrar los resultados de manera amigable y ordenada.

La implementación en Laravel permitió aprovechar funcionalidades integradas como el sistema de rutas, Eloquent ORM para manejo de datos, Blade para plantillas dinámicas y middlewares para la seguridad. Gracias a esta estructura, el sistema no solo fue más fácil de desarrollar, sino que quedó preparado para futuras ampliaciones, garantizando su adaptabilidad a las necesidades cambiantes del consultorio.

13. Diseño de la Base de Datos

La base de datos fue creada en PostgreSQL y estructurada para almacenar toda la información relacionada con los pacientes, las citas, los historiales clínicos y los usuarios del sistema. Se crearon tablas específicas para cada entidad, relacionadas mediante claves primarias y foráneas. La estructura fue optimizada para responder de forma eficiente a las consultas realizadas por los distintos módulos del sistema.

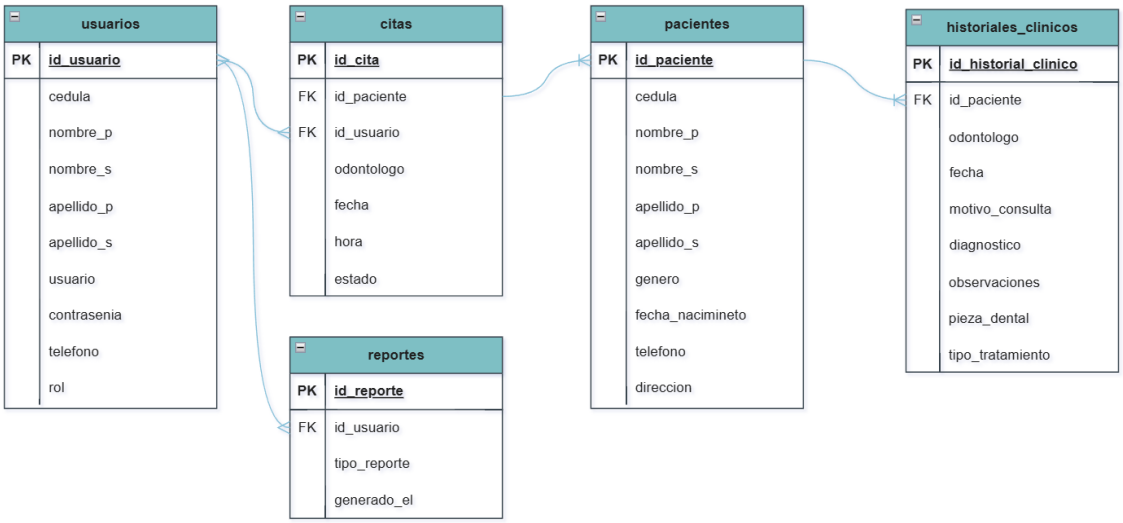


Ilustración 2- Demo Base de Datos

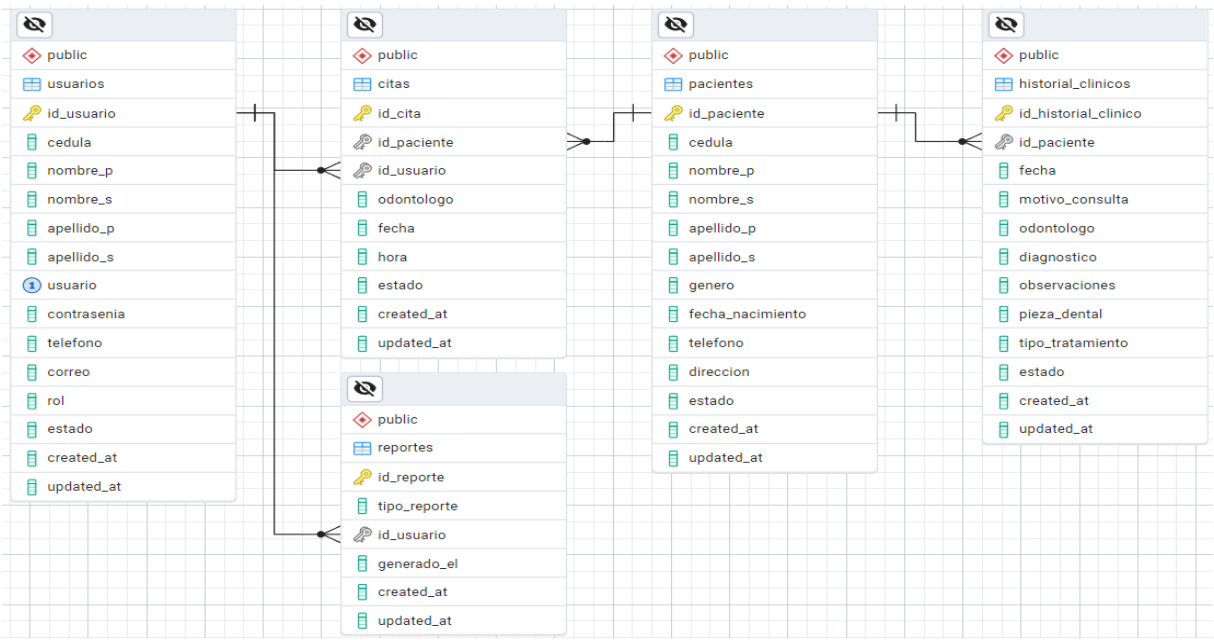


Ilustración 3- Base de Datos PostgreSQL

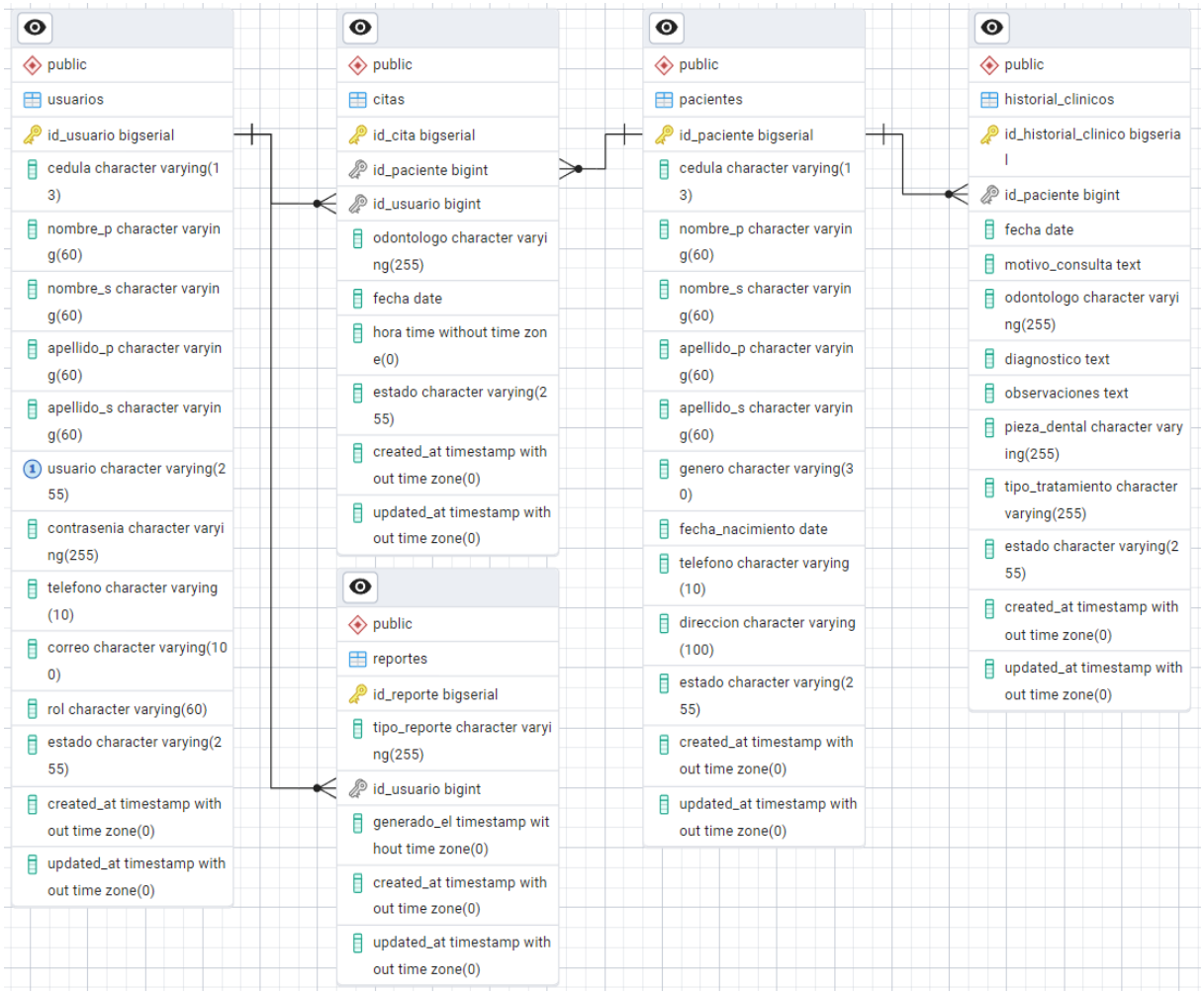


Ilustración 4- Detalle Base de Datos PostgreSQL

Este diseño permitió mantener la integridad de los datos y asegurar su trazabilidad.

Diccionario de Datos

Tabla: usuarios

Esta tabla almacena la información de los usuarios del sistema, que pueden tener diferentes roles como administrador, odontólogo o recepcionista. Los datos registrados permiten la autenticación y la gestión de accesos.

Tabla 3- Tabla Usuarios Diccionario de Datos

Campo	Tipo de Dato	Descripción	Clave	Restricciones
id_usuario	bigserial	Identificador único del usuario	PK	Auto incremental
cedula	varchar(13)	Número de cédula del usuario		ÚNICO
nombre_p	varchar(60)	Primer nombre		NOT NULL
nombre_s	varchar(60)	Segundo nombre		
apellido_p	varchar(60)	Primer apellido		NOT NULL
apellido_s	varchar(60)	Segundo apellido		
usuario	varchar(255)	Nombre de usuario		ÚNICO, NOT NULL
contrasena	varchar(255)	Contraseña encriptada		NOT NULL
telefono	varchar(10)	Teléfono del usuario		
correo	varchar(100)	Correo electrónico		
rol	varchar(60)	Rol asignado (administrador, odontólogo, secretaria)		
estado	varchar(255)	Estado del usuario (activo/inactivo)		
created_at	timestamp	Fecha de creación del registro		
updated_at	timestamp	Fecha de última actualización		

Tabla: pacientes

Contiene los datos personales de los pacientes registrados en el consultorio. Esta información es esencial para relacionarla con las citas, historiales clínicos y seguimientos médicos.

Tabla 4- Tabla Pacientes Diccionario de Datos

Campo	Tipo de Dato	Descripción	Clave	Restricciones
id_paciente	bigserial	Identificador único del paciente	PK	Auto incremental
cedula	varchar(13)	Número de cédula del paciente		ÚNICO
nombre_p	varchar(60)	Primer nombre		NOT NULL
nombre_s	varchar(60)	Segundo nombre		
apellido_p	varchar(60)	Primer apellido		NOT NULL
apellido_s	varchar(60)	Segundo apellido		
genero	varchar(30)	Género del paciente		
fecha_nacimiento	date	Fecha de nacimiento		
telefono	varchar(10)	Teléfono del paciente		
direccion	varchar(100)	Dirección domiciliaria		
estado	varchar(255)	Estado del paciente (activo/inactivo)		
created_at	timestamp	Fecha de creación del registro		
updated_at	timestamp	Fecha de última actualización		

Tabla: historiales_clinicos

Guarda los registros clínicos de cada paciente. Aquí se documentan los diagnósticos, tratamientos, observaciones, pieza dental atendida y el odontólogo responsable. Es una parte clave para el seguimiento médico.

Tabla 5- Tabla Historial Clínico Diccionario de Datos

Campo	Tipo de Dato	Descripción	Clave	Restricciones
id_historial_clinico	bigserial	Identificador único del historial clínico	PK	Auto incremental
id_paciente	bigint	ID del paciente relacionado	FK	Referencia a pacientes
fecha	date	Fecha de atención médica		
motivo_consulta	text	Motivo de la consulta		
odontologo	varchar(255)	Odontólogo que atendió		
diagnostico	text	Diagnóstico realizado		
observaciones	text	Observaciones adicionales		
pieza_dental	varchar(255)	Pieza dental tratada		
tipo_tratamiento	varchar(255)	Tipo de tratamiento aplicado		
estado	varchar(255)	Estado del tratamiento (en proceso, finalizado)		

created_at	timestamp	Fecha de creación del registro		
updated_at	timestamp	Fecha de última actualización		

Tabla: citas

Registra las citas médicas agendadas entre pacientes y odontólogos. Incluye la fecha, hora, estado de la cita y el profesional asignado. Se vincula directamente con las tablas de pacientes y usuarios.

Tabla 6 Tabla Citas Diccionario de Datos

Campo	Tipo de Dato	Descripción	Clave	Restricciones
id_cita	bigserial	Identificador único de la cita	PK	Auto incremental
id_paciente	bigint	ID del paciente relacionado	FK	Referencia a pacientes
id_usuario	bigint	ID del usuario que agendó la cita	FK	Referencia a usuarios
odontologo	varchar(255)	Nombre del odontólogo asignado		
fecha	date	Fecha programada de la cita		NOT NULL
hora	time	Hora asignada de la cita		NOT NULL
estado	varchar(255)	Estado de la cita (pendiente, completada)		
created_at	timestamp	Fecha de creación del registro		
updated_at	timestamp	Fecha de última actualización		

Tabla: reportes

Esta tabla guarda la información de los reportes generados en el sistema, como el tipo de reporte, el usuario que lo creó y la fecha de creación. Sirve para llevar un control y tener un respaldo de todas las acciones realizadas.

Tabla 7- Tabla Reportes Diccionario de Datos

Campo	Tipo de Dato	Descripción	Clave	Restricciones
id_reporte	bigserial	Identificador único del reporte	PK	Auto incremental
tipo_reporte	varchar(255)	Tipo de reporte generado (ej. citas, historial)		
id_usuario	bigint	ID del usuario que generó el reporte	FK	Referencia a usuarios
generado_el	timestamp	Fecha y hora de generación del reporte		NOT NULL
created_at	timestamp	Fecha de creación del registro		
updated_at	timestamp	Fecha de última actualización		

14. Diseño de la Interfaz y Experiencia de Usuario (UI/UX)

Para planificar y ver cómo sería la experiencia del usuario antes de desarrollar el sistema, se hicieron plantillas de diseño (mockups) de las pantallas principales. Estas sirvieron como guía para crear las interfaces finales con Bootstrap, logrando que fueran claras, organizadas y adaptables a diferentes dispositivos.

Mockup 1: Pantalla de Inicio de Sesión

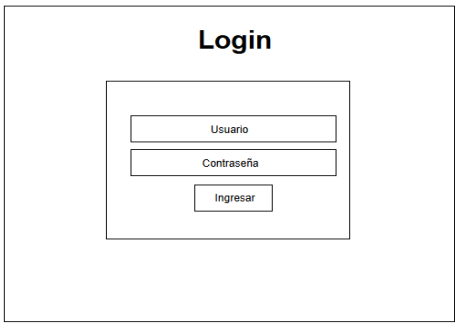


Ilustración 5- Mockup Inicio de Sesión

Se hizo un mockup para el acceso al sistema con usuario y contraseña. La plantilla tenía un diseño simple, enfocado en que fuera fácil de usar. Después, se implementó con Bootstrap, logrando que la interfaz fuera limpia, responsiva y cómoda para los usuarios.

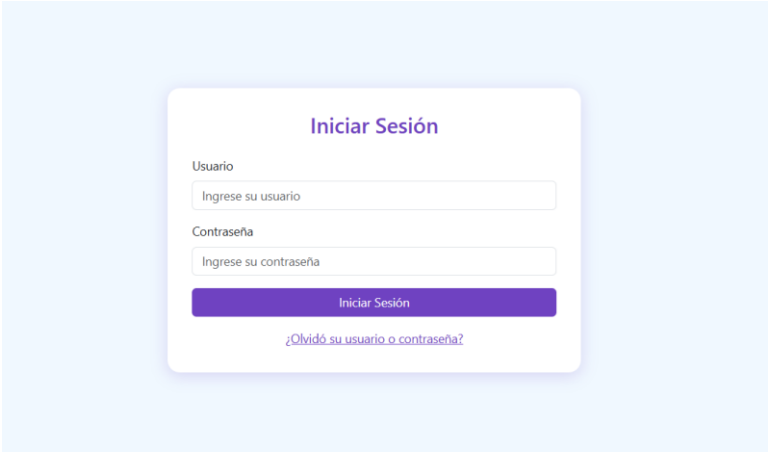


Ilustración 6- Inicio de Sesión

Mockup 2: Dashboard o Panel Principal

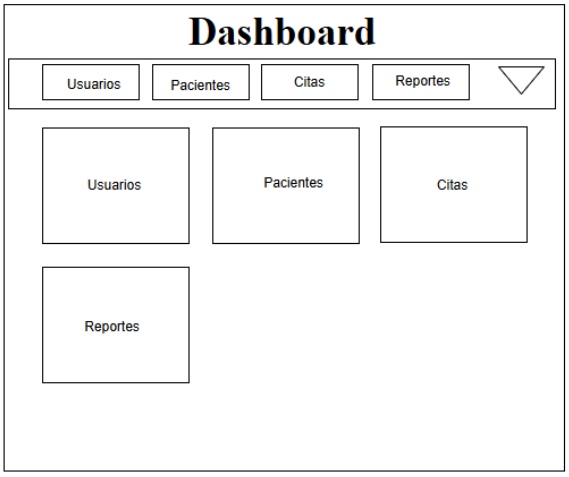


Ilustración 7- Mockup Dashboard

Se hizo una plantilla inicial del panel de control que reúne todos los accesos a los módulos del sistema. Este diseño sirvió como guía para crear una interfaz moderna, con menús superiores y secciones bien organizadas. Usando Bootstrap, se logró una navegación fácil e intuitiva, que funciona bien en diferentes dispositivos.

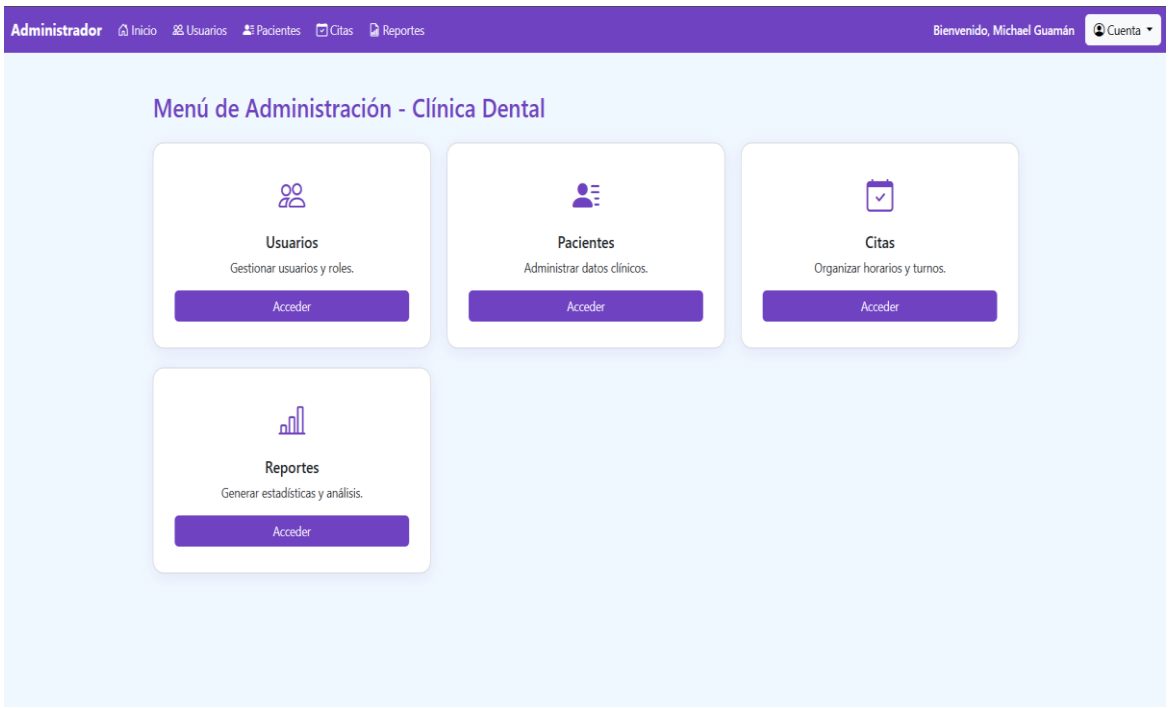


Ilustración 8- Dashboard

Modulo 1: Interfaz del Módulo de Usuarios

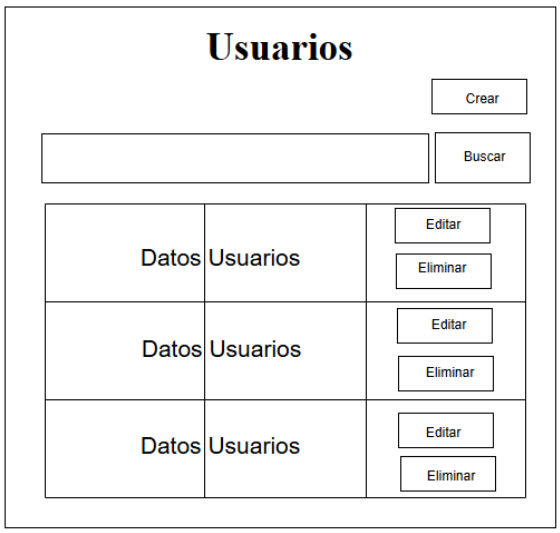


Ilustración 9- Mockup Modulo de Usuario

Se creó un mockup para la sección de usuarios, que mostraba una tabla con los datos y botones para crear, editar o eliminar registros. Luego, se implementó con Bootstrap, logrando una interfaz clara, funcional y adaptada a las necesidades del personal administrativo del consultorio.

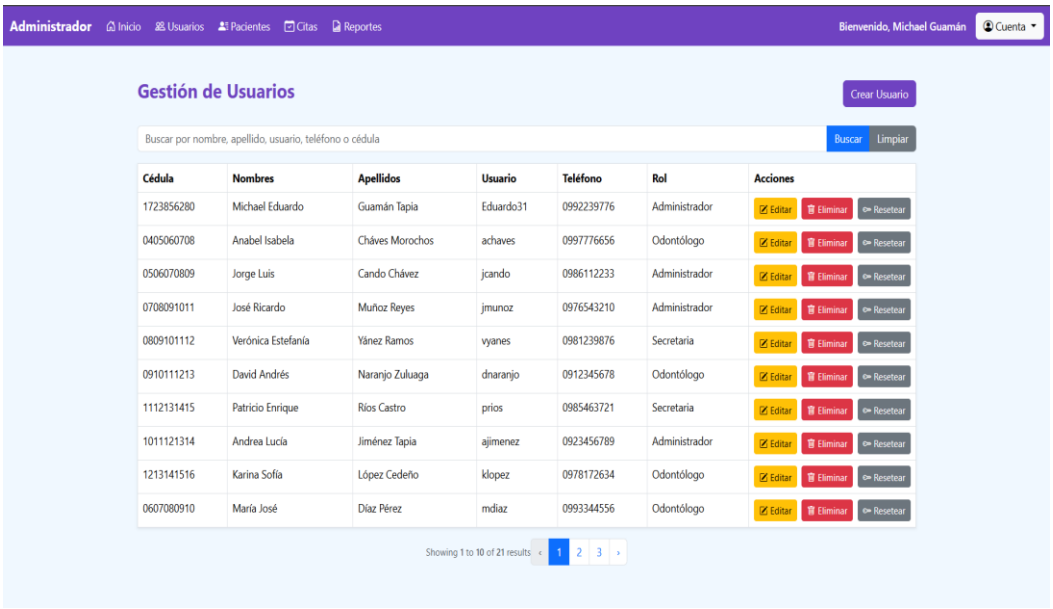


Ilustración 10- Modulo de Usuario

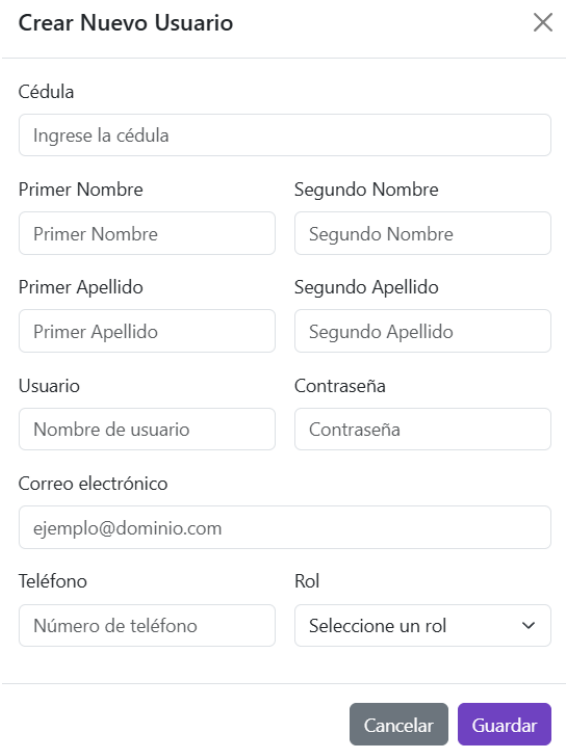


Ilustración 11- Crear Modulo de Usuario

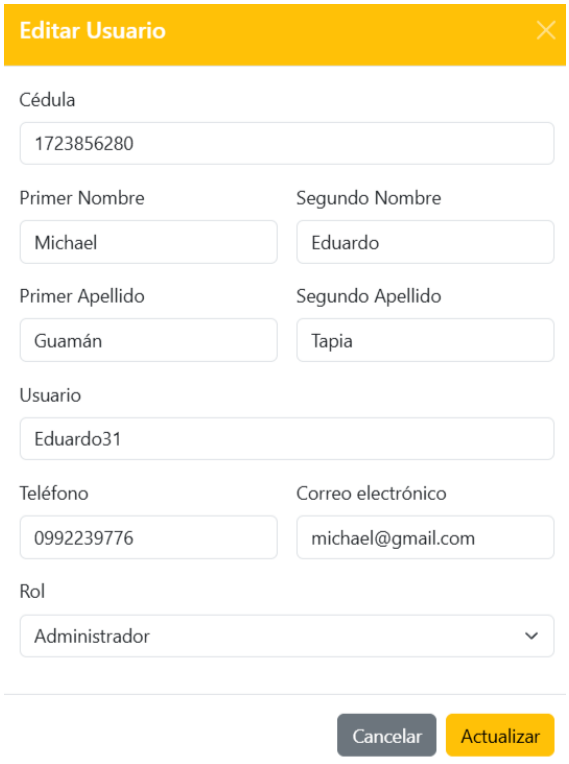


Ilustración 12- Editar Modulo de Usuario

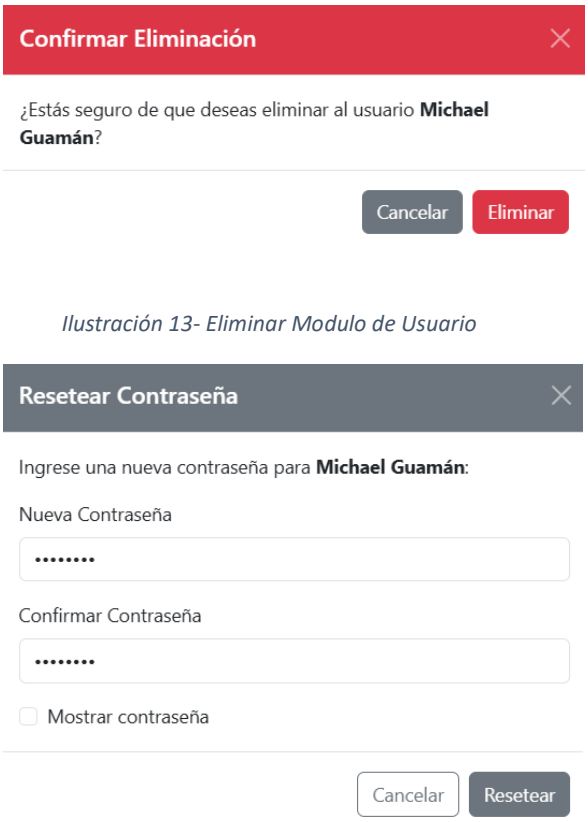


Ilustración 13- Eliminar Modulo de Usuario

Ilustración 14- Resetear Contraseña Modulo de Usuario

Módulo 2: Interfaz del Módulo de Pacientes

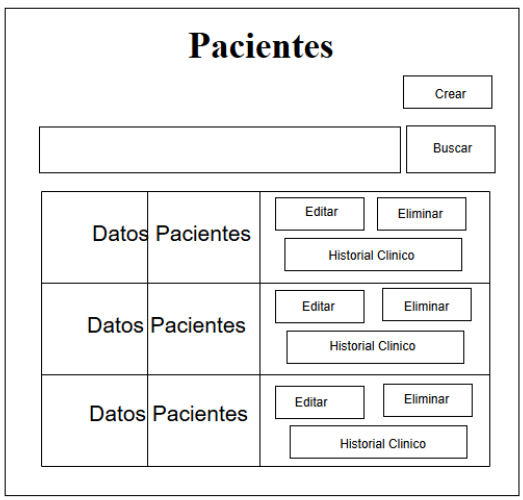


Ilustración 15- Mockup Modulo Pacientes

Se hizo un mockup para la sección de pacientes, que incluía una tabla con la información básica de cada uno y botones para registrar nuevos pacientes, editar datos o ver su historial

clínico. Después, se implementó con Bootstrap, enfocándose en que la información fuera clara, organizada y que las funciones más usadas fueran fáciles de acceder.

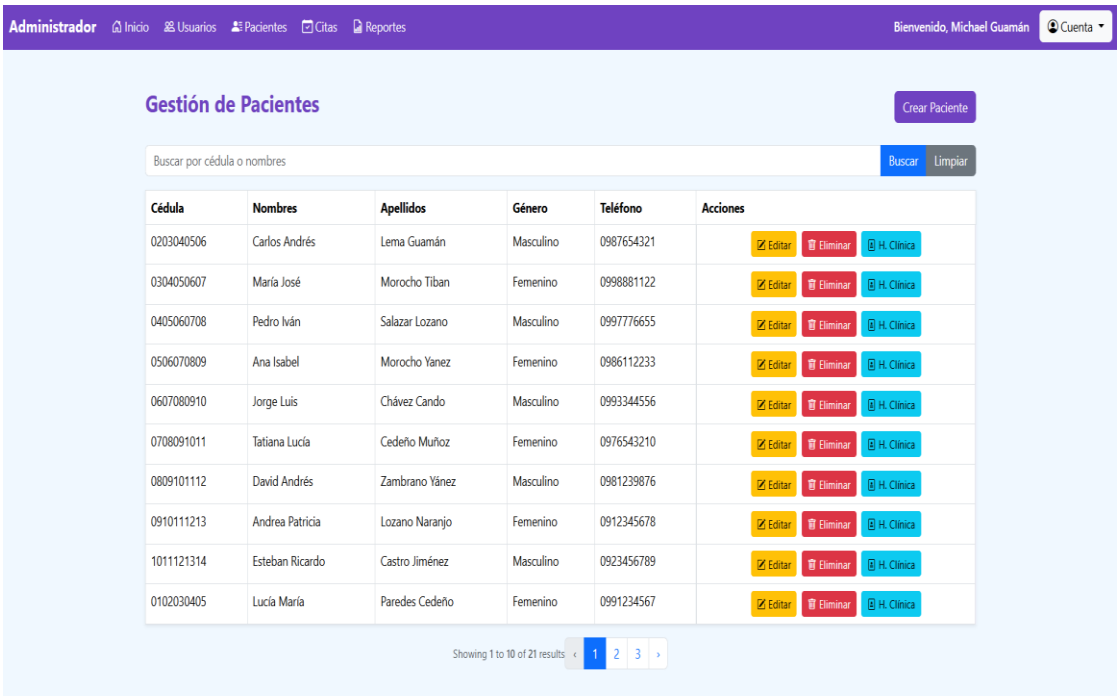


Ilustración 16- Modulo Pacientes

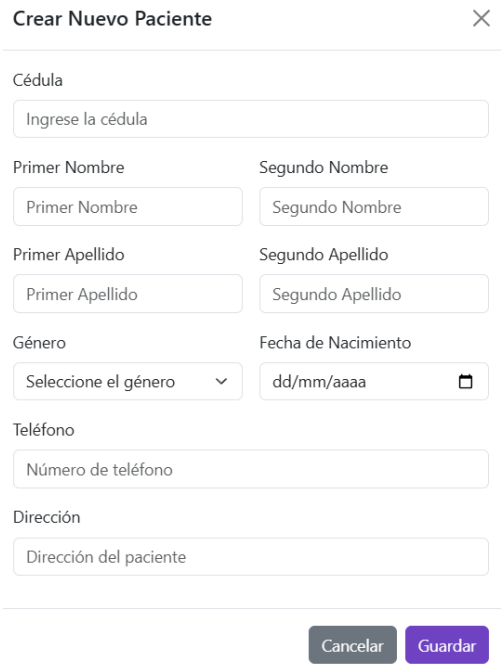


Ilustración 17- Crear Modulo Pacientes

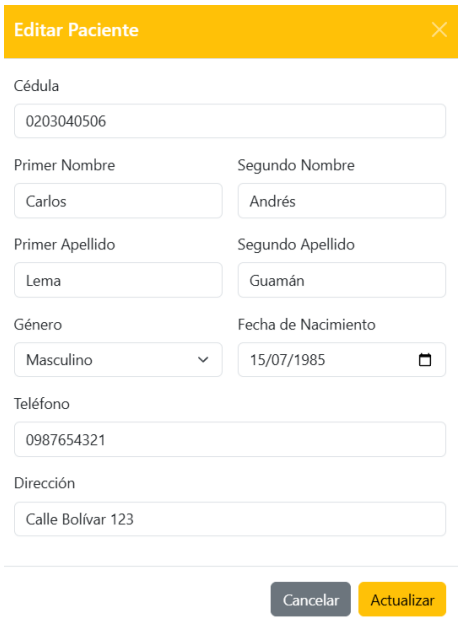


Ilustración 18- Editar Modulo Pacientes

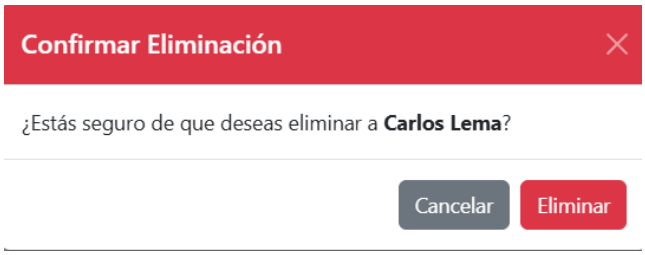


Ilustración 19- Eliminar Modulo Pacientes

Módulo 3: Interfaz del Módulo de Historial Clínico

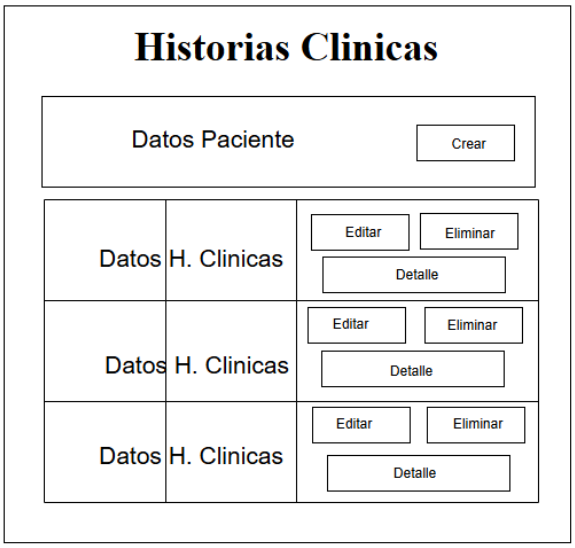


Ilustración 20- Mockup Modulo Historias Clínicas

Se diseñó un mockup para el historial clínico, que mostraba la información médica de cada paciente de manera organizada y con acceso rápido a registros anteriores. Una vez que se validó la plantilla, se implementó con Bootstrap, logrando que los datos se vieran claros, ordenados y fáciles de usar para el personal del consultorio.

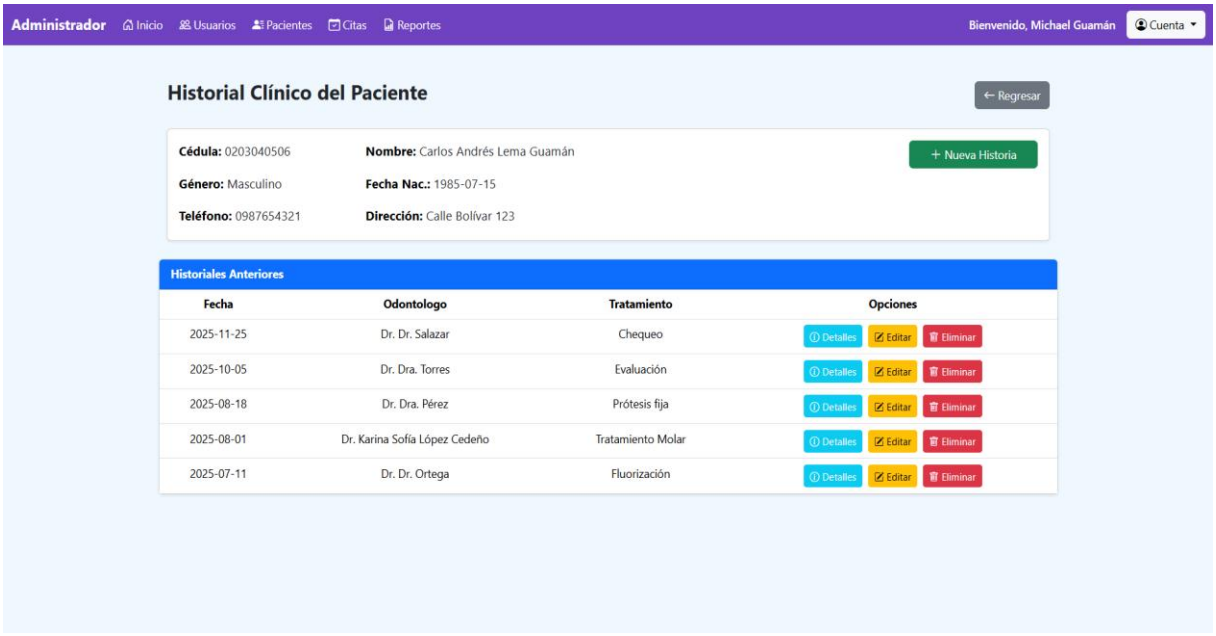


Ilustración 21- Modulo Historias Clínicas

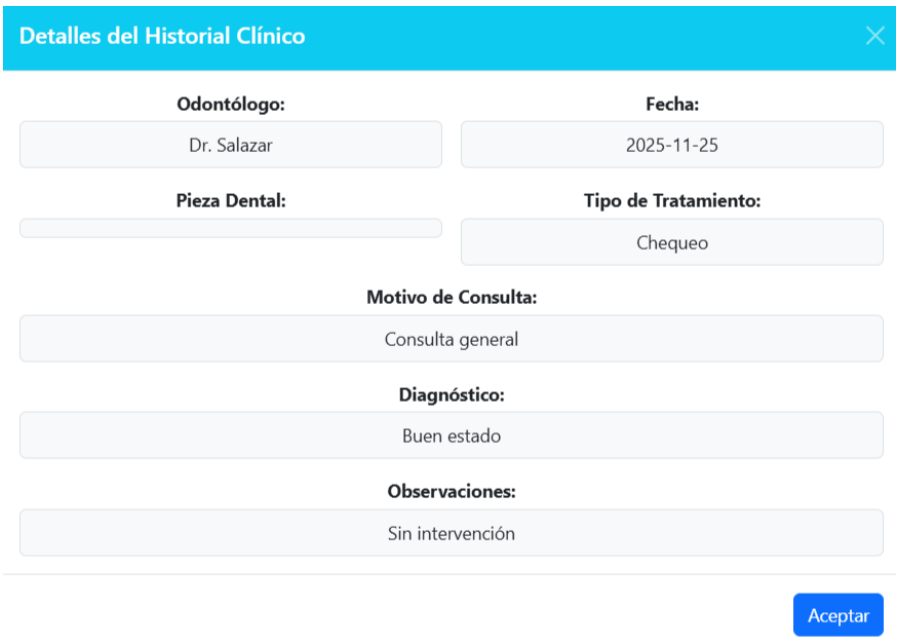


Ilustración 22- Detalle Modulo Historias Clínicas

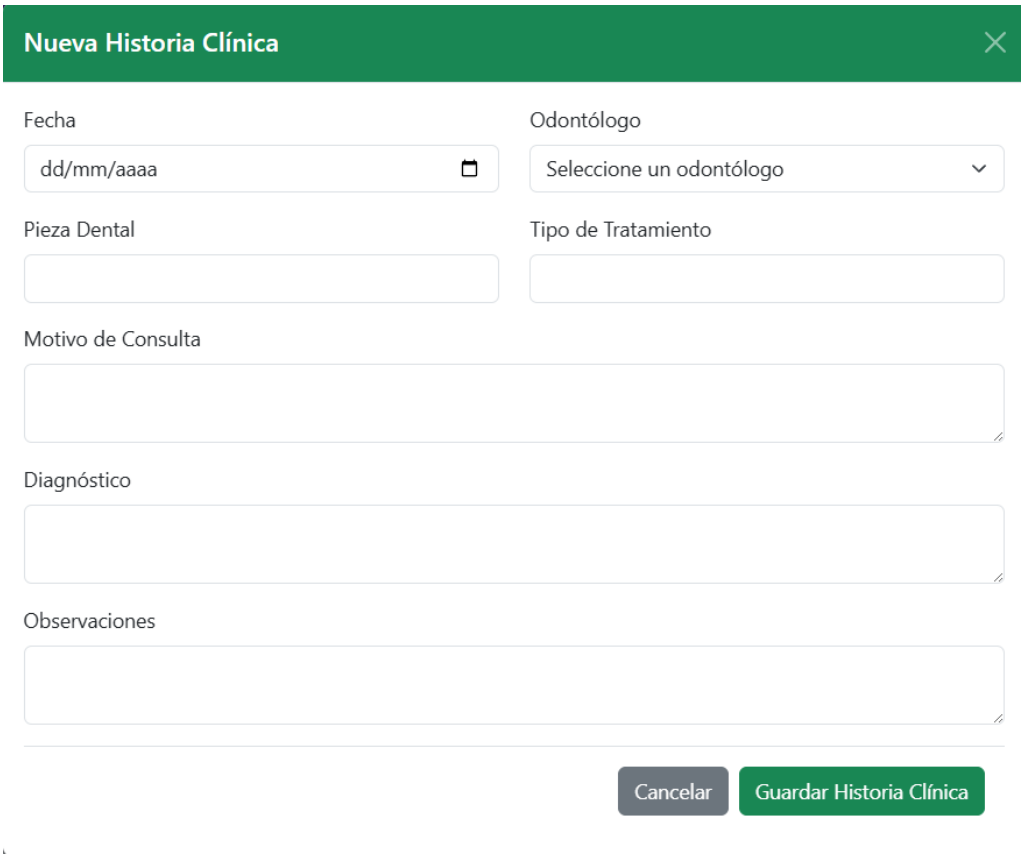


Ilustración 23- Crear Modulo Historias Clínicas

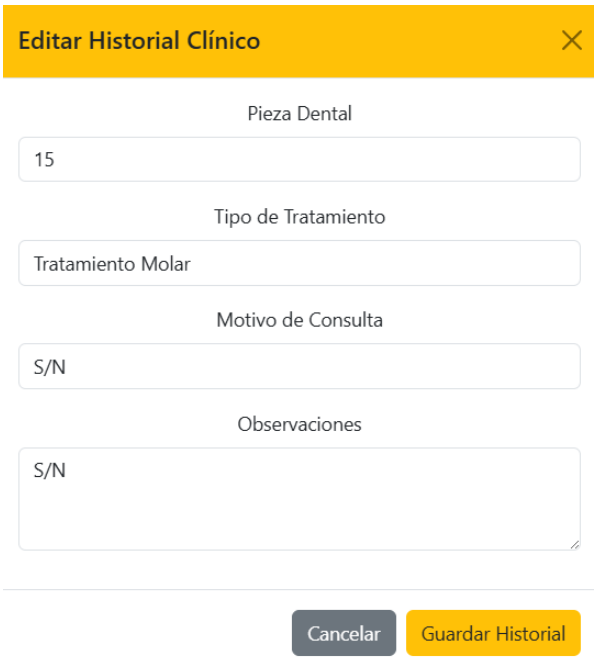


Ilustración 24- Editar Modulo Historias Clínicas

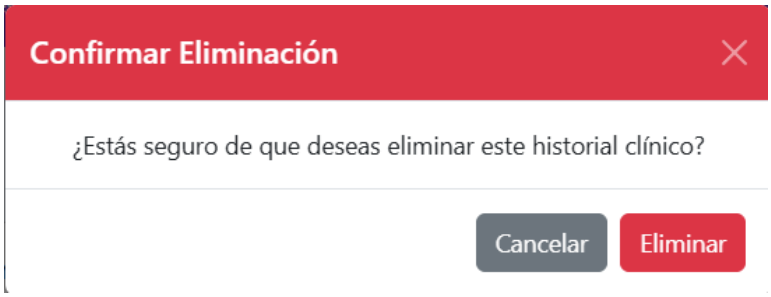


Ilustración 25- Eliminar Modulo Historias Clínicas

Módulo 4: Interfaz del Módulo de Citas

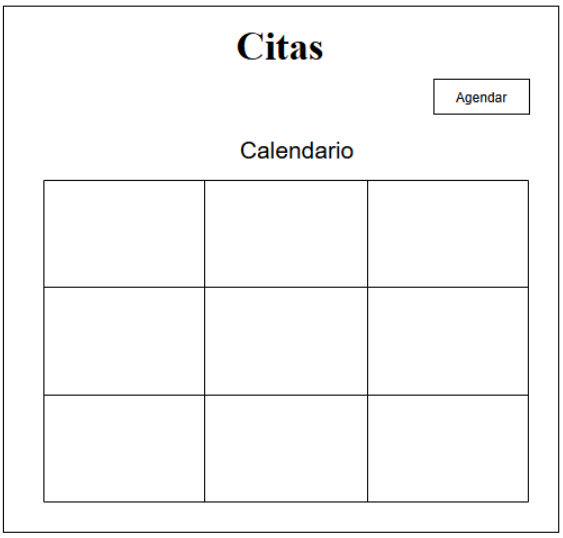


Ilustración 26- Mockup Calendario Modulo Citas

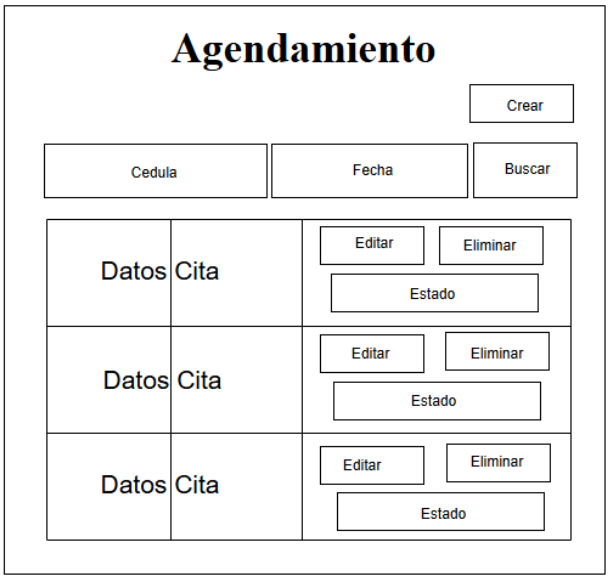


Ilustración 27- Mockup Modulo Citas

Se hizo un mockup para la gestión de citas, usando un calendario para que fuera más fácil programar y ver las atenciones. La plantilla incluía botones para agregar, modificar o cancelar citas, y también mostraba la disponibilidad. Después se desarrolló con Bootstrap, logrando una interfaz clara, adaptable y fácil de usar.

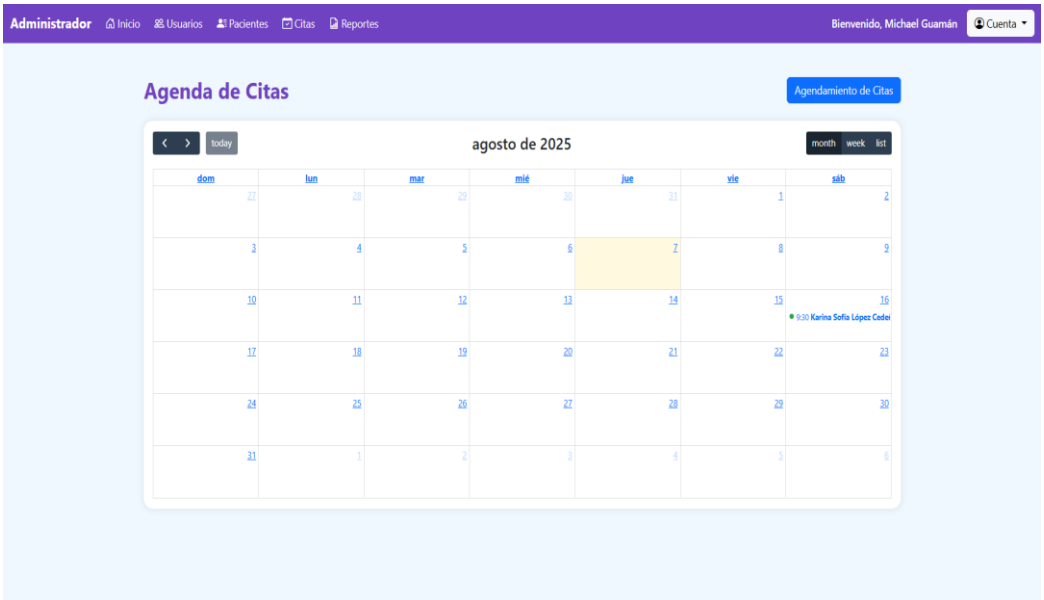


Ilustración 28- Calendario Modulo Citas

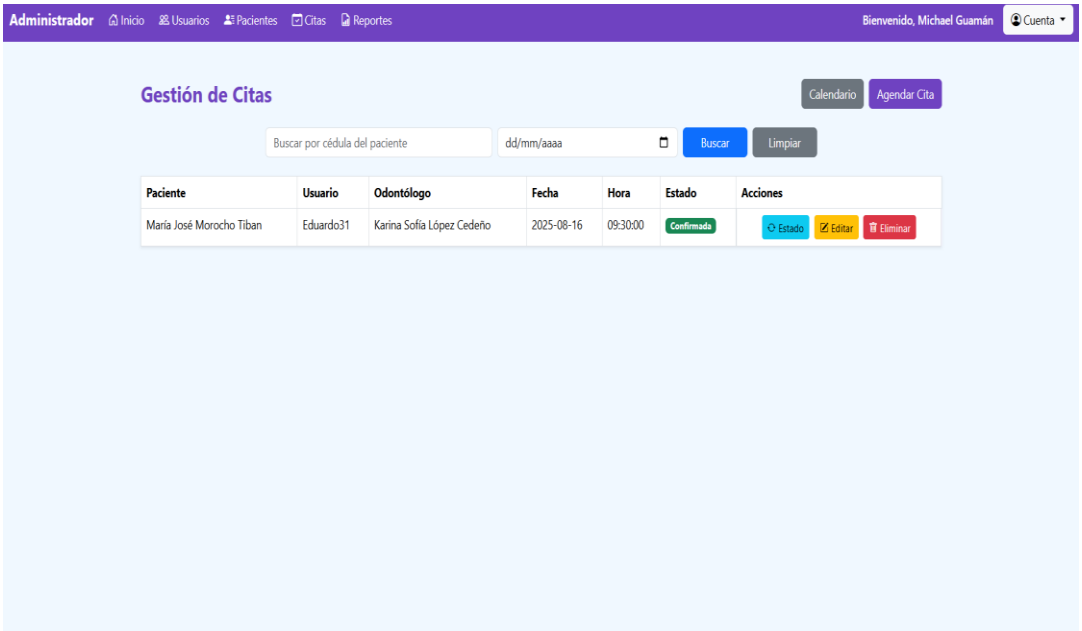


Ilustración 29- Modulo Citas

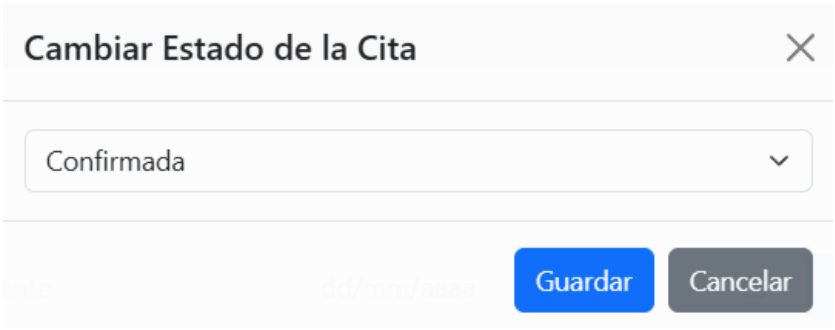


Ilustración 30- Estados Modulo Citas

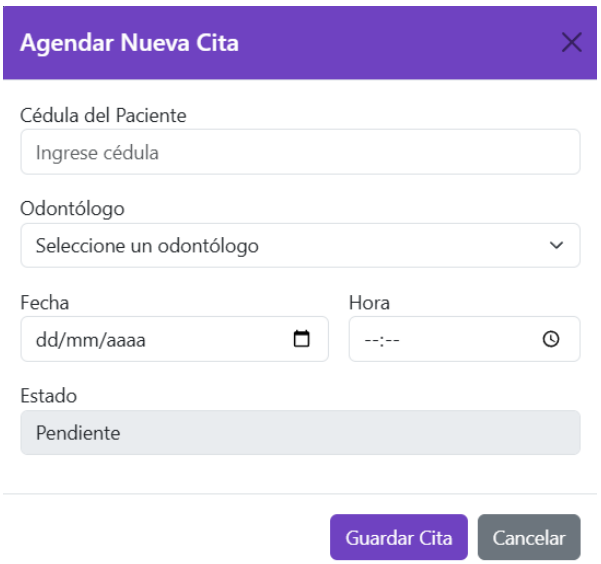


Ilustración 31- Crear Modulo Citas

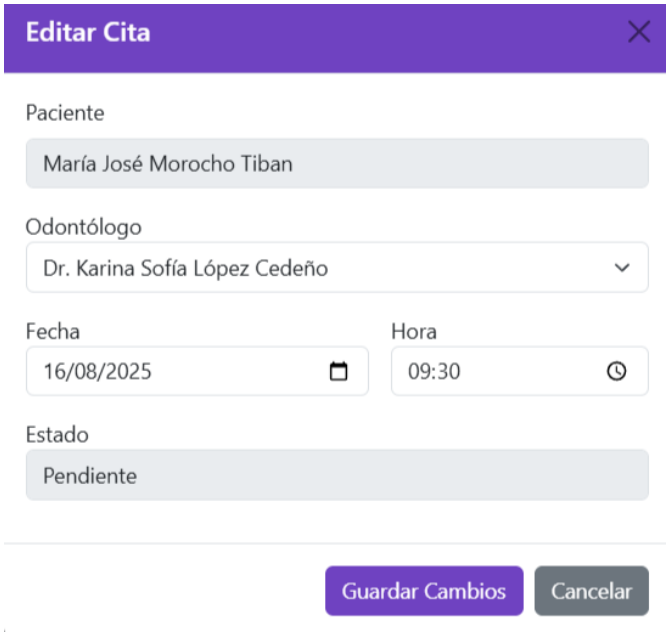


Ilustración 32- Editar Modulo Citas

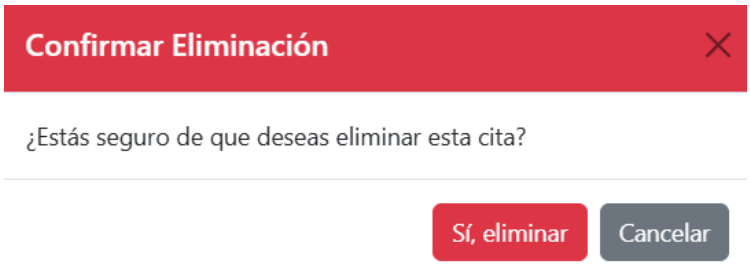


Ilustración 33- Eliminar Modulo Citas

Módulo 5: Interfaz del Módulo de Reportes

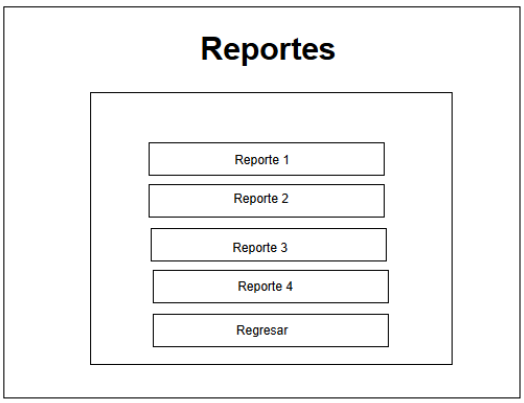


Ilustración 34- Mockup Modulo Reportes

Se diseñó un mockup para crear y ver reportes, especialmente para mostrar de manera detallada las historias clínicas de cada paciente. Después, esta plantilla se implementó con Bootstrap, asegurando que la información se vea clara, que las listas largas tengan paginación y que los botones para exportar o imprimir los reportes sean fáciles de usar.

Ilustración 35- Modulo Reportes

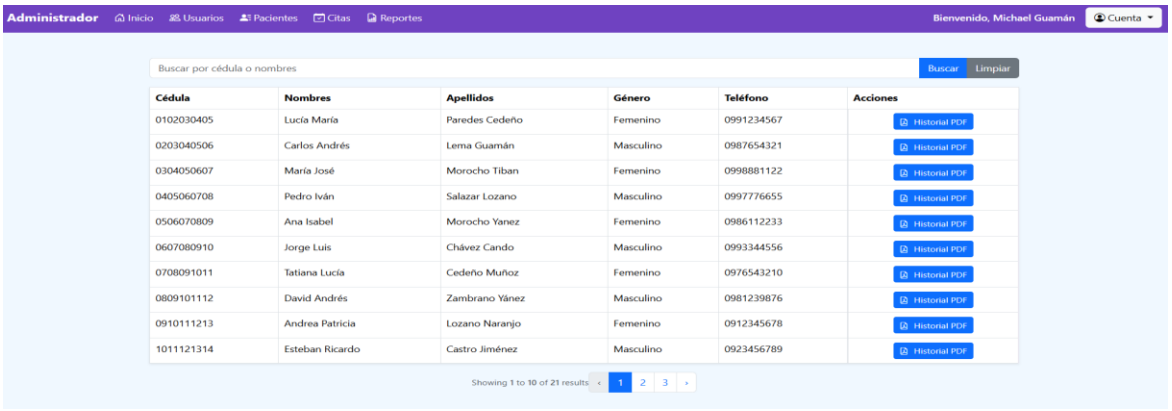


Ilustración 36- Reporte Historia Clínica Por Paciente Modulo Reportes

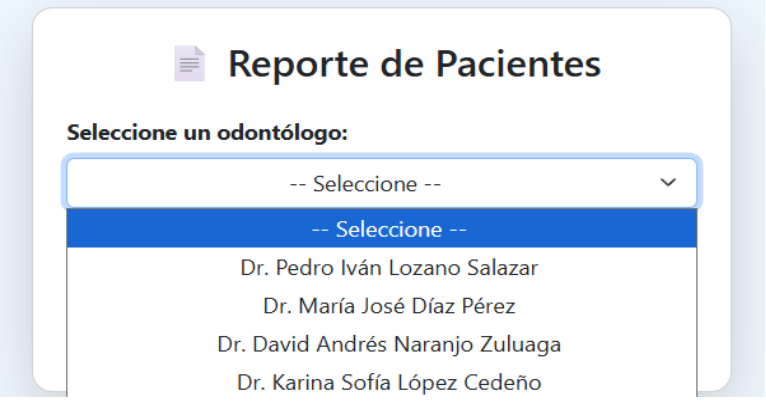


Ilustración 37- Reporte Por Odontólogo Modulo Reportes

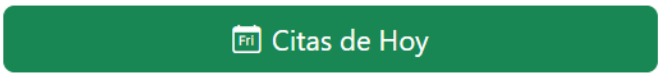


Ilustración 38- Reporte Cita de Hoy Modulo Reportes

15. Desarrollo de Módulos Principales

La mayoría de los módulos que se desarrollaron tienen funciones CRUD (crear, leer, actualizar y eliminar), lo que facilita manejar la información de manera eficiente. Pero uno de los más importantes fue el historial clínico por paciente. Desde el módulo de pacientes, cada registro tiene un acceso directo a una vista donde se puede ver todo el historial de atenciones, tratamientos y observaciones.

15.1. Gestión de Citas

Se creó una vista de calendario interactivo usando la librería FullCalendar, lo que permitió ver todas las citas de manera visual y dinámica. Cada cita tenía un color según su estado (pendiente, confirmada o cancelada), y al hacer clic sobre ella se abría un recuadro con los detalles, como el nombre del paciente, el odontólogo, la fecha y la hora.

```php
//Calendario
public function verCalendario()
{
    $citas = Citas::with('paciente')->where('estado', '!=', 'Eliminada')->get();

    $eventos = $citas->map(function ($cita) {
        // Definir el color según el estado
        $color = match ($cita->estado) {
            'Pendiente' => '#f39c12',   // Naranja
            'Confirmada' => '#28a745',  // Verde
            'Cancelada' => '#dc3545',   // Rojo
            default => '#6c757d',       // Gris (para cualquier otro estado)
        };

        return [
            'title' => $cita->odontologo . ' - ' . $cita->paciente->nombre_p . ' ' . $cita->paciente->apellido_p,
            'start' => $cita->fecha . 'T' . $cita->hora,
            'color' => $color,

            // Propiedades para el modal
            'paciente' => $cita->paciente->nombre_p . ' ' . $cita->paciente->apellido_p,
            'odontologo' => $cita->odontologo,
            'fecha' => $cita->fecha,
            'hora' => $cita->hora,
            'estado' => $cita->estado,
        ];
    });

    return view('citas/calendarioCitas', ['eventos' => $eventos]);
}
```

Ilustración 39- Fragmento de Código Modulo Citas P1

En este fragmento de código, cada cita se convierte en un evento de FullCalendar usando un map, lo que permite que todas las citas se muestren correctamente en el calendario de manera dinámica.

- title → nombre del odontólogo + nombre completo del paciente.
- start → fecha y hora en formato YYYY-MM-DDT / HH:MM.
- color → se asigna dinámicamente dependiendo del estado (Pendiente, Confirmada, Cancelada, otro).

También se agregan propiedades personalizadas (paciente, odontologo, fecha, hora, estado) para mostrarlas luego en el modal.

```
document.addEventListener('DOMContentLoaded', function () {
    const calendarEl = document.getElementById('calendar');
    const container = document.getElementById('data-container');
    const eventosJson = container.dataset.eventos;
    const eventos = JSON.parse(eventosJson);

    const calendar = new FullCalendar.Calendar(calendarEl, {
        locale: 'es',
        initialView: 'dayGridMonth',
        headerToolbar: {
            left: 'prev,next today',
            center: 'title',
            right: 'dayGridMonth,timeGridWeek,listWeek'
        },
        events: eventos,
        eventClick: function(info) {
            const event = info.event;
            const props = event.extendedProps;

            document.getElementById('modalTitulo').textContent = 'Detalles de la Cita';
            document.getElementById('modalPaciente').textContent = props.paciente;
            document.getElementById('modalOdontologo').textContent = props.odontologo;
            document.getElementById('modalFecha').textContent = props.fecha;
            document.getElementById('modalHora').textContent = props.hora;

            // Muestra el modal
            const citaModal = new bootstrap.Modal(document.getElementById('detalleCitaModal'));
            citaModal.show();
        }
    });

    calendar.render();
});
```

Ilustración 40- Fragmento de Código Modulo Citas P2

En el script, se toma el JSON de los eventos desde un atributo data de un contenedor HTML, para poder mostrarlos en el calendario.

Inicializa FullCalendar con:

- locale: 'es' (en español).
- initialView: 'dayGridMonth' (vista mensual por defecto).
- Botones de navegación (prev, next, today) y cambio de vista (dayGridMonth, timeGridWeek, listWeek).
- La lista de eventos cargada desde Laravel.

Evento eventClick:

- Cuando el usuario hace clic en un evento:
- Se obtienen las propiedades personalizadas (extendedProps).
- Se rellenan los campos del modal con los datos de la cita (paciente, odontologo, fecha, hora).
- Se abre el modal de Bootstrap para mostrar la información.

```
<!-- Modal Detalle Cita -->
<div class="modal fade" id="detalleCitaModal" tabindex="-1" aria-labelledby="detalleCitaLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title fw-bold" id="modalTitulo">Detalles de la Cita</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Cerrar"></button>
            </div>
            <div class="modal-body">
                <p><strong>Paciente:</strong> <span id="modalPaciente"></span></p>
                <p><strong>Odontólogo:</strong> <span id="modalOdontologo"></span></p>
                <p><strong>Fecha:</strong> <span id="modalFecha"></span></p>
                <p><strong>Hora:</strong> <span id="modalHora"></span></p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cerrar</button>
            </div>
        </div>
    </div>
</div>
```

Ilustración 41- Fragmento de Código Modulo Citas P3

En la segunda parte, se desarrolló una vista administrativa donde las citas podían ser gestionadas mediante un sistema CRUD, permitiendo crear, editar, filtrar y eliminar registros. El filtrado se podía realizar por cédula del paciente o por fecha específica, facilitando la localización de información. Esta estructura dual mejoró tanto la visualización como la administración de las citas, optimizando el flujo de trabajo del personal.

Capitulo V

16. Resultados

La aplicación web desarrollada para el consultorio odontológico Perfect Smile permitió digitalizar y automatizar los procesos de gestión de pacientes, citas e historiales clínicos. El sistema fue implementado bajo la arquitectura MVC, con Laravel como framework, PostgreSQL como gestor de base de datos y Bootstrap para el diseño de la interfaz, cumpliendo con los objetivos planteados.

Entre los principales resultados se destacan:

- **Gestión de pacientes:**

  El registro y actualización de información de los pacientes ahora se realiza en una base de datos centralizada, lo que evita la pérdida de información y permite consultar de manera inmediata el historial de cada paciente.

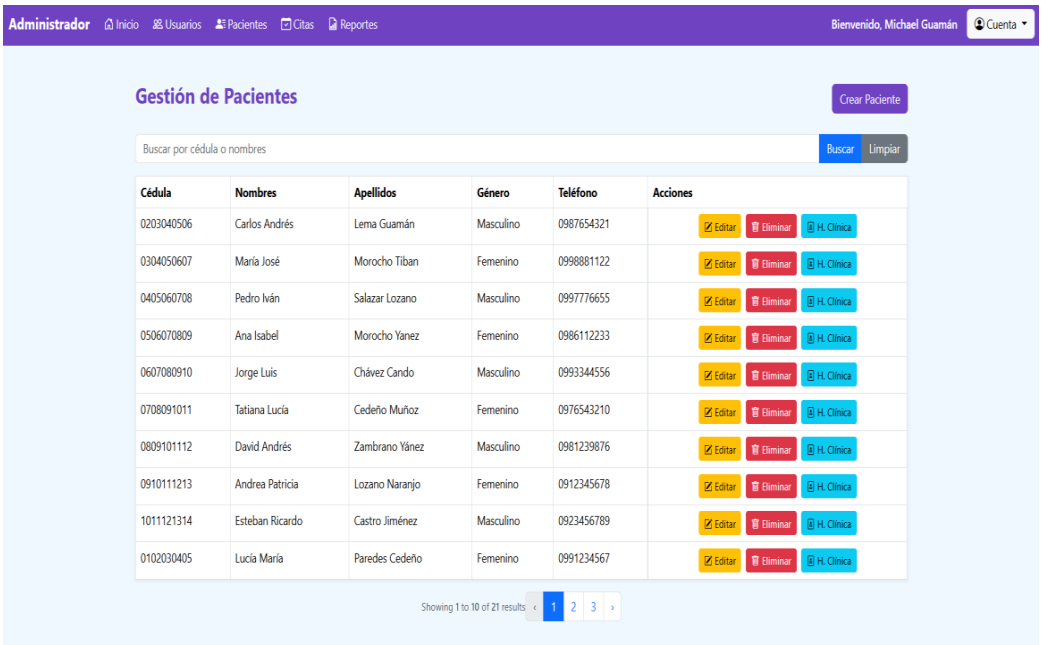


Ilustración 42- Resultado Modulo Pacientes

- **Programación de citas:**

  Se creó un calendario interactivo que hace más fácil organizar y ver las citas médicas. El personal puede registrar, editar y eliminar citas, lo que ayuda a evitar errores como asignar dos citas al mismo tiempo o confundir horarios.

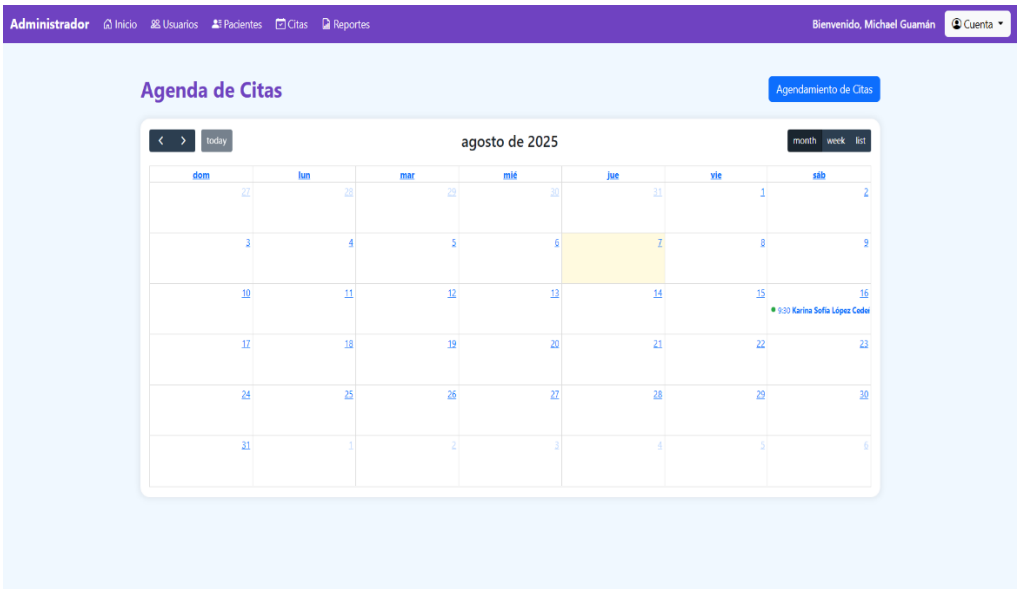


Ilustración 43- Resultado Calendario Modulo Citas

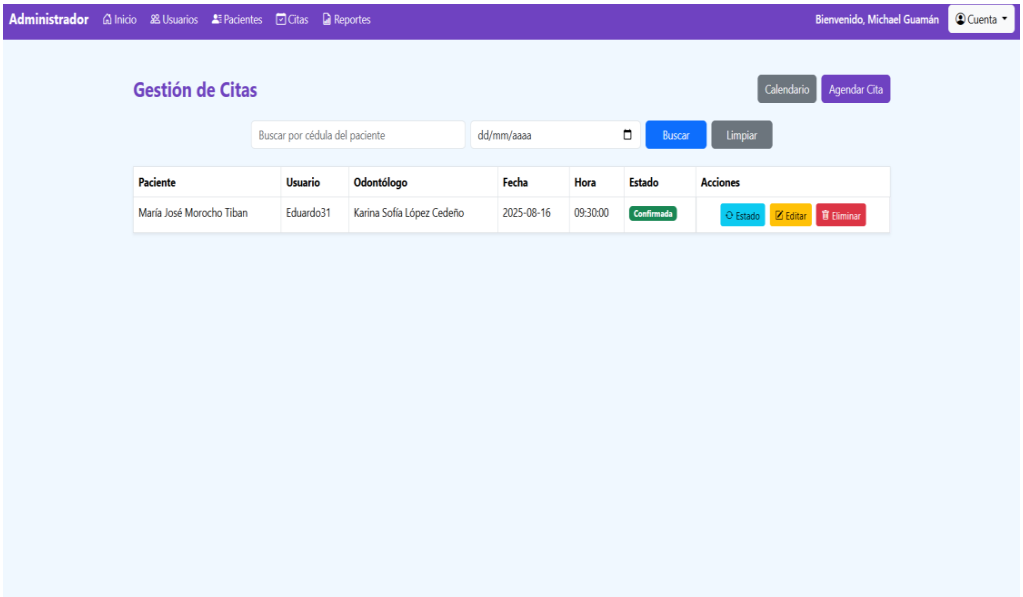


Ilustración 44- Resultado Modulo Citas

- **Historial clínico digital:**

  Cada paciente cuenta con un historial clínico que permite registrar diagnósticos, tratamientos realizados y observaciones, esto garantiza la trazabilidad de los procedimientos y una mejor continuidad en la atención.

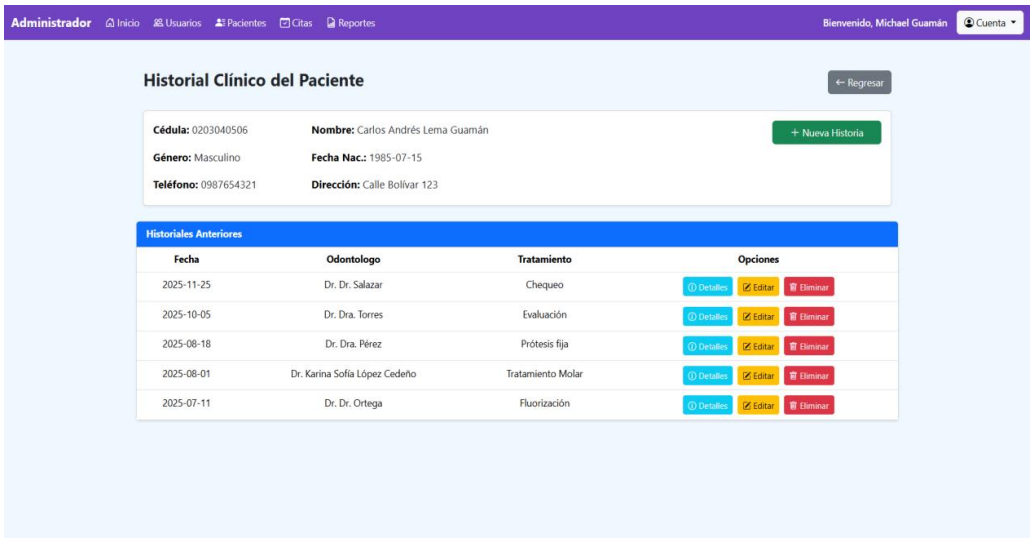


Ilustración 45- Resultado Historial Clínico Por Paciente Modulo Historias Clínicas

- **Interfaz amigable y responsiva:**

  Gracias a Bootstrap, el sistema se puede usar tanto en computadoras como en dispositivos móviles, lo que facilita consultar información en cualquier momento.

- **Seguridad y respaldo de la información:**

  Se implementó un mecanismo de eliminación lógica y física de los registros, lo que permite un mayor control y protección de los datos.

  - **Eliminación lógica:** el registro queda marcado como inactivo, no se ve en la interfaz, pero se mantiene un historial para auditorías.

  Con esta estrategia, se asegura un manejo más seguro y confiable de la información clínica, reduciendo riesgos de pérdida accidental y manteniendo la integridad del sistema.

Tabla 8- Casos de prueba de la aplicación web

Casos de prueba de la aplicación web					
Caso de Prueba	**Descripción**	**Entrada**	**Resultado Esperado**	**Resultado Obtenido**	**Estado**
Registro de paciente	Registrar un nuevo paciente en el sistema con datos completos	Nombre, apellido, cédula, correo, teléfono	El sistema guarda el paciente en la base de datos y muestra mensaje de confirmación	El paciente se guarda correctamente y aparece en la lista de pacientes	Aprobado
Edición de paciente	Modificar datos de un paciente existente	Cambio de teléfono y correo	El sistema actualiza la información en la base de datos	La información se actualiza correctamente	Aprobado
Programación de cita	Crear una cita para un paciente con fecha y hora definidas	Fecha: 21/08/2025, Hora: 10:00, Odontólogo asignado	La cita se registra y aparece en el calendario	La cita aparece en el calendario sin errores	Aprobado
Doble agendamiento	Intentar agendar dos citas en la misma fecha y hora para un mismo odontólogo	Fecha y hora repetidas	El sistema debe mostrar mensaje de error y no registrar la cita duplicada	El sistema bloquea la duplicación y muestra advertencia	Aprobado
Registro de historial clínico	Agregar un nuevo registro clínico a un paciente	Diagnóstico, tratamiento, observaciones	El historial clínico se asocia al paciente y queda guardado	El historial se registra correctamente y es consultable	Aprobado
Eliminación de cita	Eliminar una cita previamente registrada	ID de cita seleccionada	El sistema debe eliminar la cita y actualizar el calendario	La cita se elimina y ya no aparece en el calendario	Aprobado

17. Discusión

17.1. Interpretación

Los resultados muestran que digitalizar la gestión de pacientes, citas e historiales clínicos en el consultorio Perfect Smile ayudó a resolver de manera importante problemas que antes eran frecuentes, como la desorganización, la pérdida de información y los retrasos en la atención a los pacientes. Ahora, con el sistema, es más fácil mantener todo registrado de forma ordenada y consultar los datos cuando se necesite. Además, al implementar métodos de eliminación lógica y física de los registros, se fortaleció la seguridad de la información. Esto significa que los datos se pueden controlar mejor y que se reducen los riesgos de pérdida accidental, lo que brinda mayor confianza al personal y a los pacientes.

17.2. Comparación

Los resultados que obtuvimos son muy parecidos a los de investigaciones anteriores. Por ejemplo, el sistema desarrollado para Dental Group en Guayaquil (Jeannette Elizabeth Sanunga Totoy, 2018) también mostró ventajas al automatizar procesos clínicos y administrativos. Esto confirma que la digitalización realmente ayuda a organizar mejor el trabajo en los consultorios. Además los resultados se alinean con lo establecido en el marco teórico sobre los Sistemas de Información en Salud (SIS) y la Historia Clínica Electrónica (HCE), los cuales son reconocidos como herramientas fundamentales para optimizar la gestión clínica y asegurar la continuidad en la atención (Jaume Canela-Soler, 2010).

En la parte técnica, la implementación del patrón Modelo-Vista-Controlador (MVC), facilitó la escalabilidad y el mantenimiento del sistema, confirmando su capacidad para proyectos de este tipo.

17.3. Implicaciones

El impacto práctico de los resultados se traduce en:

- Mayor eficiencia administrativa, al reducir tiempos de búsqueda de información y agendamiento.
- Incremento en la satisfacción del paciente, gracias a una atención más ágil y ordenada.
- Mayor seguridad de la información, por el control en los procesos de eliminación y la posibilidad de auditoría.
- Esto permite que el consultorio mejore su competitividad frente a otros servicios de odontología y refuerce la confianza de sus pacientes.

17.4. Limitaciones

Es importante reconocer ciertas limitaciones del proyecto:

- Las pruebas del sistema se llevaron a cabo únicamente en el consultorio Perfect Smile, lo que significa que no se evaluó su funcionamiento en un entorno de mayor escala.
- El número de usuarios participantes en las pruebas fue reducido (personal administrativo y clínico del consultorio), lo que limita la posibilidad de generalizar los resultados.
- No se incluyeron funcionalidades de notificación automática de citas ni reportes estadísticos avanzados, las cuales podrían mejorar la gestión en futuras versiones.

A pesar de estas limitaciones, los resultados obtenidos demuestran que la propuesta es factible, cumple con los objetivos establecidos y proporciona una base sólida para futuras mejoras y escalabilidad del sistema.

CAPITULO VI

18. Conclusiones

- Primero, la aplicación web que desarrollé ayudó mucho a organizar toda la información del consultorio Perfect Smile. Antes, todo se hacía con papeles y agendas, lo que hacía que se perdieran documentos o que fuera difícil encontrar los datos de los pacientes. Con el sistema, todo está más ordenado y se puede acceder a la información de manera rápida y segura, lo que mejora el trabajo del personal y la atención a los pacientes.
- Además, usar el patrón MVC permitió que el sistema sea más fácil de manejar y de actualizar. Esto quiere decir que la parte que muestra la información, la que procesa los datos y la que almacena todo en la base de datos están separadas, así que si en el futuro se quiere agregar algo nuevo, no se rompe el resto del sistema.
- También implementamos un método para eliminar registros de forma lógica, es decir, que los datos no se borran completamente y se pueden recuperar si es necesario. Esto le da más seguridad y control a la información que se maneja en el consultorio.
- En general, el sistema cumplió los objetivos que nos propusimos al inicio. Mostró que es posible usar tecnología moderna en consultorios dentales, haciendo más eficiente la administración y mejorando la organización de los historiales clínicos. Los resultados demuestran que no solo las clínicas grandes se benefician de la digitalización; incluso consultorios pequeños pueden organizarse mejor y ahorrar tiempo en sus procesos diarios.
- Por último, podemos decir que esta aplicación web no solo facilita el trabajo administrativo, sino que también aporta al servicio al paciente. Permite tener todo más controlado, más rápido y más seguro, cumpliendo con las normas actuales de salud, y abre la posibilidad de seguir mejorando el sistema en el futuro.

19. Recomendaciones

- Se recomienda que en el futuro se agregue un sistema de recordatorios automáticos para las citas, usando correo electrónico o mensajes SMS. Esto ayudaría a que los pacientes no falten a sus citas y permitiría que el personal del consultorio pueda organizar mejor su trabajo y planificar las actividades diarias.
- Además, se recomienda incluir un módulo de reportes estadísticos dinámicos, que permita obtener información detallada sobre la frecuencia de pacientes, los tratamientos más solicitados, así como el número de citas atendidas y canceladas. Estos reportes serán herramientas clave para la toma de decisiones estratégicas y para evaluar el desempeño general del consultorio.
- Considerar el desarrollo de una versión móvil o aplicación nativa (Android/iOS) que permita a pacientes y personal acceder al sistema desde cualquier lugar, aumentando la accesibilidad y usabilidad de la herramienta.
- Se recomienda realizar capacitaciones periódicas para el personal administrativo y odontológico en el uso del sistema, con el fin de garantizar un aprovechamiento adecuado de todas las funcionalidades y evitar errores de operación.
- Evaluar en el futuro la integración con servicios de facturación electrónica, sistemas de inventario o plataformas de salud pública, lo cual permitiría que el sistema se convierta en una herramienta más completa e integral.
- Aplicar el sistema en otros consultorios odontológicos con características similares, con el fin de evaluar su adaptabilidad y realizar mejoras que permitan escalar el software a nivel institucional o regional.

20. Bibliografías

Adiel Joshua Preciado Rodríguez, M. A. (1 de Abril de 2021). *Importancia del uso de sistemas de información en la automatización de historiales clínicos, una revisión sistemática*. Obtenido de http://scielo.sld.cu/scielo.php?pid=S1684-18592021000100012&script=sci_arttext&tlng=pt

Albarracín Valderrama Astrid Eliana, R. C. (Junio de 2021). *Servicios odontológicos móviles.* Obtenido de https://repositorio.unbosque.edu.co/items/58500170-dc27-4d0d-b691-580a283f1e6e

Bootstrap. (2025). *Build fast, responsive sites with Bootstrap*. Obtenido de https://getbootstrap.com/

Estrada Velasco Marco Vinicio, N. V. (2021). *Revisión Sistemática de la Metodología Scrum para el Desarrollo de Software*. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=8384028

Git. (2025). *Documentación*. Obtenido de https://git-scm.com/doc

Hospital General Docente de Calderón. (15 de Marzo de 2023). *Política Nacional de Transformación Digital del Sector Salud*. Obtenido de https://www.hgdc.gob.ec/images/Hospital/Base%20legal/AC-00068-2024%20DIC%2018-Tercer%20Suplemento%20Nro.%20715%20Politica%20de%20Transformacion%20Digital%20de%20la%20Salud.pdf

Jaume Canela-Soler, D. E.-M.-B.-E. (Obtubre de 2010). *Sistemas de Información en Salud e indicadores de salud: una perspectiva integradoraInformation systems in health and health indicators: an integrating perspective*. Obtenido de https://www.sciencedirect.com/science/article/abs/pii/S0025775310700026

Jeannette Elizabeth Sanunga Totoy, K. N. (Diciembre de 2018). *Implementacion del sistema para el control de Historia clinica de pacientes en centro odontologico Dental Group*. Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/16767/1/UPS-GT002446.pdf

Juan Eduardo Gil Yacobazzo, M. J. (Diciembre de 2018). *Historia clínica electrónica: confidencialidad y privacidad de los datos clínicos*. Obtenido de http://www.scielo.edu.uy/scielo.php?pid=S1688-03902018000400102&script=sci_arttext

Laravel. (2025). *A PHP framework with a robust ecosystem*. Obtenido de https://laravel.com/

Ministerio de Salud Pública (MPS). (6 de Enero de 2025). *Acuerdos. 00068-2024 Se aprueba y se autoriza la publicación de la "Política Nacional de Transformación Digital del Sector Salud"*. Obtenido de https://vlex.ec/vid/acuerdos-00068-2024-aprueba-1064082502

Ministerio de Salud Pública (MSP). (16 de Diciembre de 2024). *Ecuador avanza hacia una salud digital al alcance de todos*. Obtenido de https://www.salud.gob.ec/ecuador-avanza-hacia-una-salud-digital-al-alcance-de-todos/

Mozilla. (2025). *JavaScript (JS)*. Obtenido de https://developer.mozilla.org/en-US/docs/Web/JavaScript

PostgreSQL Global Development Group. (2025). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Obtenido de https://www.postgresql.org/

21. Anexos

El consultorio odontológico Perfect Smile se encuentra ubicado en la parroquia Alóag, cantón Mejía, provincia de Pichincha. Está situado en una zona de fácil acceso para los pacientes, cercana a vías principales y con disponibilidad de transporte público. Su localización estratégica permite brindar atención a la comunidad del sector y sus alrededores.

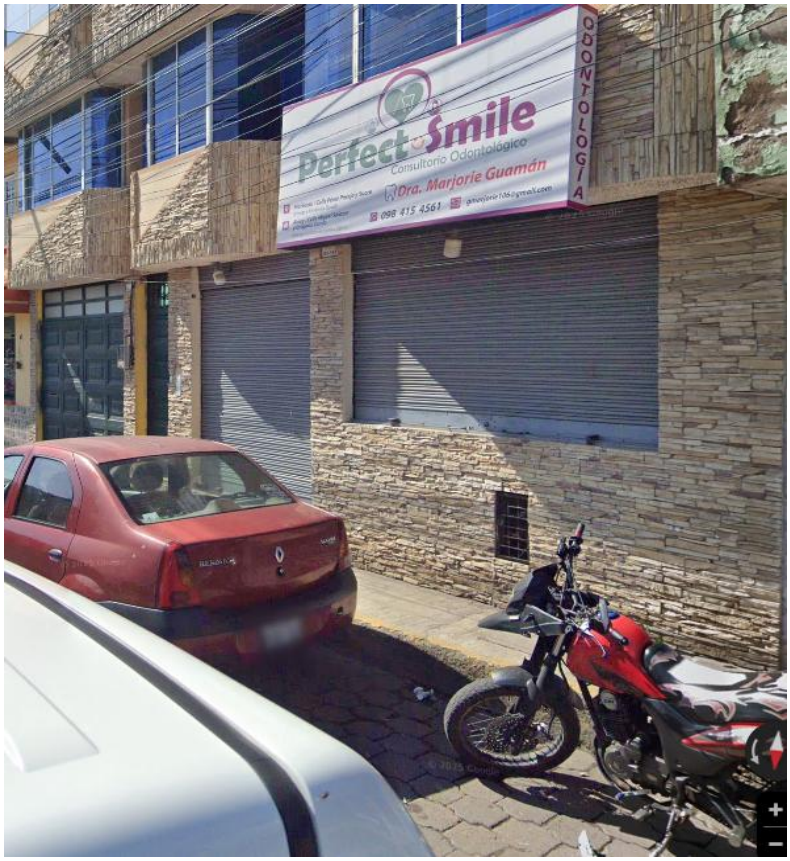


Ilustración 46- Anexo 1 Consultorio Perfect Smile

Manual De Programador

Sistema de Gestión Integral de Pacientes, Citas e Historiales Clínicos

Consultorio Odontológico Perfect Smile

Versión: 1.0

Autor: Michael Eduardo Guamán Tapia

Índice

Tabla de Ilustraciones

1. Introducción

El presente Manual del Programador describe de manera detallada el funcionamiento interno, la arquitectura y las pautas técnicas necesarias para la instalación, configuración, mantenimiento y ampliación del sistema web "Perfect Smile", desarrollado para la gestión integral de pacientes, citas e historiales clínicos en el consultorio odontológico.

Este manual tiene como finalidad servir de guía a futuros programadores que requieran comprender el código fuente, realizar modificaciones o implementar mejoras en el sistema. La documentación está orientada a facilitar la comprensión de la estructura del software, las tecnologías empleadas y las buenas prácticas aplicadas en su desarrollo.

El sistema fue implementado bajo el framework Laravel, utilizando PostgreSQL como gestor de base de datos y Bootstrap para el diseño de la interfaz gráfica. La arquitectura adoptada es el modelo MVC (Modelo-Vista-Controlador), lo que garantiza una separación clara entre la lógica de negocio, la gestión de datos y la capa de presentación.

Con este manual, cualquier programador podrá desplegar el sistema en un nuevo entorno, comprender su funcionamiento y asegurar la continuidad en su mantenimiento y evolución.

2. Herramientas Utilizadas

Para el desarrollo del sistema web "Perfect Smile", se emplearon diversas herramientas y tecnologías que facilitaron el proceso de construcción, pruebas y despliegue. A continuación, se detallan:

2.1. Lenguaje de Programación

PHP 8.2: Lenguaje principal utilizado en el desarrollo del backend, compatible con el framework Laravel.

Ilustración 1 - Versión de PHP

2.2. Framework

Laravel 12: Framework de PHP que permitió implementar la arquitectura MVC, facilitando la organización del código, la seguridad y el mantenimiento del sistema.

Ilustración 2 - Versión de Laravel

2.3. Base de Datos

PostgreSQL 17.2 (64 bits, Windows, compilado con MSVC 19.42.34435): Gestor de base de datos relacional, elegido por su robustez, seguridad y soporte para consultas avanzadas.

	version text
1	PostgreSQL 17.2 on x86_64-windows, compiled by msvc-19.42.34435, 64-bit

Ilustración 3 - Versión de PostgreSQL

2.4. Frontend

Bootstrap 5: Framework de CSS que permitió diseñar una interfaz amigable y responsive.

JavaScript: Utilizado para interactividad en formularios, validaciones y componentes dinámicos.

Blade: Motor de plantillas propio de Laravel que facilitó la integración de la lógica en las vistas.

2.5. Servidor Web

Se utilizó el servidor embebido de PHP (Symfony Server) a través del comando ***php artisan serve*** para el desarrollo y pruebas locales del sistema.

2.6. Entorno de Desarrollo

Visual Studio Code: Editor de código principal, con extensiones para Laravel, PostgreSQL y control de versiones.

2.7. Control de Versiones

Git: Sistema de control de versiones para gestionar cambios en el código.

GitHub: Plataforma de alojamiento remoto utilizada para respaldar y colaborar en el desarrollo.

2.8. Generación de Reportes

Laravel DomPDF: Librería utilizada para la generación de reportes en formato PDF, como historiales clínicos y citas médicas.

3. Requisitos del Programa

3.1. Requisitos de Software

Sistema Operativo: Windows 10 o superior, Ubuntu 20.04 LTS o equivalente.

Backend:

- PHP 8.x
- Laravel 12
- Composer para gestión de dependencias

Base de Datos: PostgreSQL 14 o superior

Frontend:

- Bootstrap 5
- JavaScript (Vanilla JS)
- Blade (motor de plantillas de Laravel)

Generación de Reportes: Laravel DomPDF

Control de Versiones: Git y GitHub

3.2. Requisitos de Hardware

Procesador:

- Intel i5 o equivalente

Memoria RAM:

- 8 GB mínimo

Espacio en disco:

- 20 GB libre

Conexión a Internet:

- Para instalación de dependencias y actualizaciones

4. Desarrollo del Programa

4.1. Arquitectura General

El sistema “Perfect Smile” está desarrollado bajo la arquitectura MVC (Modelo-Vista-Controlador), lo que permite una separación clara entre:

- **Modelo:** Gestiona la comunicación con la base de datos (Eloquent ORM de Laravel).
- **Vista:** Interfaz de usuario en Blade + Bootstrap, mostrando formularios, tablas y reportes.
- **Controlador:** Contiene la lógica de negocio, valida datos, gestiona peticiones y prepara la información para las vistas o generación de PDF.

4.2. Estructura de Carpetas Principal

4.2.1. Models

/app/Models: Contiene los modelos del sistema: Usuario, Paciente, Cita, Historial Clínico, Reporte.

Usuario

app > Models > usuario.php > PHP Intelephense > usuario

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class usuario extends Model
{
    use HasFactory;

    protected $table = 'usuarios';

    protected $primaryKey = 'id_usuario';
    public $incrementing = true;
    protected $keyType = 'int';

    protected $fillable = [
        'cedula',
        'nombre_p',
        'nombre_s',
        'apellido_p',
        'apellido_s',
        'usuario',
        'contrasenia',
        'telefono',
        'correo',
        'rol',
        'estado',
    ];

}
```

Ilustración 4 - Modelo Usuario

Paciente

app > Models > paciente.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class paciente extends Model
{
    use HasFactory;

    protected $table = 'pacientes';

    protected $primaryKey = 'id_paciente';
    public $incrementing = true;
    protected $keyType = 'int';

    protected $fillable = [
        'cedula',
        'nombre_p',
        'nombre_s',
        'apellido_p',
        'apellido_s',
        'genero',
        'fecha_nacimiento',
        'telefono',
        'direccion',
        'estado',
    ];
}
```

Ilustración 5- Modelo Paciente

Historial Clínico

app > Models > historial_clinico.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class historial_clinico extends Model
{
    use HasFactory;

    protected $table = 'historial_clinicos';

    protected $primaryKey = 'id_historial_clinico';
    public $incrementing = true;
    protected $keyType = 'int';

    protected $fillable = [
        'id_paciente',
        'odontologo',
        'fecha',
        'motivo_consulta',
        'diagnostico',
        'observaciones',
        'pieza_dental',
        'tipo_tratamiento',
        'estado',
    ];

    // Relación: un historial pertenece a un paciente
    public function paciente()
    {
        return $this->belongsTo(Paciente::class, 'id_paciente', 'id_paciente');
    }

}
```

Ilustración 6- Modelo Historial Clínico

Cita

app > Models > citas.php > PHP Intelephense > citas

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class citas extends Model
{
    protected $table = 'citas';
    protected $primaryKey = 'id_cita';
    public $timestamps = true;

    protected $fillable = [
        'id_paciente',
        'id_usuario',
        'odontologo',
        'fecha',
        'hora',
        'estado',
    ];

    public function paciente()
    {
        // tercer parámetro: clave primaria de la tabla pacientes
        return $this->belongsTo(Paciente::class, 'id_paciente', 'id_paciente');
    }

    public function usuario()
    {
        return $this->belongsTo(Usuario::class, 'id_usuario', 'id_usuario');
    }
}
```

Ilustración 7- Modelo de Cita

Reporte

app > Models > reporte.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Reporte extends Model
{
    protected $table = 'reportes';
    protected $primaryKey = 'id_reporte';
    public $timestamps = true;

    protected $fillable = [
        'tipo_reporte',
        'id_usuario',
        'generado_el',
    ];

    // Relación con el modelo Usuario
    public function usuario()
    {
        return $this->belongsTo(Usuario::class, 'id_usuario');
    }
}
```

Ilustración 8- Modelo de Reporte

4.2.2. Controllers

/app/Http/Controllers: En esta carpeta encontraremos los diferentes controladores que gestionan cada módulo:

Usuario_Controlador

Función Index

```php
public function index(Request $request)
{
    $buscar = strtolower($request->input('buscar'));

    $usuarios = Usuario::where('estado', 'Activo') // Filtrar solo los activos
        ->when($buscar, function ($query, $buscar) {
            $query->whereRaw('LOWER(nombre_p) LIKE ?', ["%$buscar%"])
                ->orWhereRaw('LOWER(apellido_p) LIKE ?', ["%$buscar%"])
                ->orWhereRaw('LOWER(usuario) LIKE ?', ["%$buscar%"])
                ->orWhereRaw('LOWER(cedula) LIKE ?', ["%$buscar%"]);
        })
        ->orderBy('updated_at', 'desc')
        ->paginate(10);

    return view('usuario/indexUsuario', compact('usuarios'));
}
```

Ilustración 9 - Función Index Controlador Usuario

La función index del controlador de usuarios permite obtener y mostrar todos los usuarios activos del sistema en la interfaz. Esta función incluye búsqueda por nombre, apellido, usuario o cédula, de forma insensible a mayúsculas y minúsculas, lo que facilita encontrar registros específicos. Los resultados se muestran de manera paginada, con 10 usuarios por página, y se ordenan según la fecha de última actualización, mostrando primero los más recientes. Finalmente, envía los datos a la vista indexUsuario para su visualización.

Puntos importantes:

- Filtra solo usuarios con estado “Activo”.
- La búsqueda es opcional; si no se ingresa un término, se muestran todos los usuarios activos.
- La búsqueda permite coincidencias parciales en nombre, apellido, usuario o cédula.
- Los resultados se ordenan por fecha de actualización (updated_at) de manera descendente.
- Se utiliza paginación para mejorar la navegación y visualización de los usuarios.

Función Store

```php
public function store(Request $request)
{
    $validatedData = $request->validate([
        'cedula'      => 'required|string|max:13|unique:usuarios,cedula',
        'nombre_p'    => 'required|string|max:60',
        'nombre_s'    => 'nullable|string|max:60',
        'apellido_p'  => 'required|string|max:60',
        'apellido_s'  => 'nullable|string|max:60',
        'usuario'     => 'required|string|max:60|unique:usuarios,usuario',
        'contrasenia' => 'required|string|min:3',
        'telefono'    => 'required|string|max:20',
        'correo'      => 'required|string|max:100|unique:usuarios,correo',
        'rol'         => 'required|string|in:Secretaria,Odontólogo,Administrador',
    ]);

    // Encriptar contraseña antes de guardar
    $validatedData['contrasenia'] = bcrypt($validatedData['contrasenia']);
    $validatedData['estado'] = 'Activo';

    Usuario::create($validatedData);

    return redirect()->route('indexUsuarios')->with('success', 'Usuario creado correctamen
}
```

Ilustración 10 - Función Store Controlador Usuario

La función store del controlador de usuarios se encarga de recibir los datos de un nuevo usuario desde un formulario, validarlos, procesarlos y guardarlos en la base de datos. Primero verifica que todos los campos requeridos cumplan con las reglas de validación, como longitud máxima, unicidad y formato correcto. Antes de guardar, encripta la contraseña para proteger la seguridad del usuario y establece el estado del usuario como "Activo". Finalmente, crea el registro en la base de datos y redirige al listado de usuarios mostrando un mensaje de éxito.

Puntos importantes:

- Valida que los campos obligatorios estén presentes y cumplan con las restricciones (longitud, formato y unicidad).
- Campos opcionales como segundo nombre y segundo apellido permiten valores nulos.
- La contraseña se encripta con bcrypt antes de guardarla.
- El rol del usuario debe ser uno de los permitidos: Secretaria, Odontólogo o Administrador.
- Establece automáticamente el estado del usuario como "Activo".
- Crea el registro en la tabla usuarios de la base de datos.

Función Update

```php
public function update(Request $request, usuario $usuario)
{
    $usuario = Usuario::findOrFail($request->id_usuario);

    // Valida y actualiza los datos
    $usuario->cedula = $request->cedula;
    $usuario->nombre_p = $request->nombre_p;
    $usuario->nombre_s = $request->nombre_s;
    $usuario->apellido_p = $request->apellido_p;
    $usuario->apellido_s = $request->apellido_s;
    $usuario->usuario = $request->usuario;
    $usuario->telefono = $request->telefono;
    $usuario->correo = $request->correo;
    $usuario->rol = $request->rol;

    $usuario->save();

    return redirect()->back()->with('success', 'Usuario actualizado correctamente.');
}
```

Ilustración 11- Función Update Controlador Usuario

La función update permite modificar los datos de un usuario existente. Primero busca al usuario mediante su ID; si no lo encuentra, genera un error. Luego actualiza los campos con la información recibida desde el formulario, incluyendo cédula, nombres, apellidos, usuario, correo, teléfono y rol. Finalmente, guarda los cambios en la base de datos y redirige a la misma página mostrando un mensaje de éxito.

Puntos importantes:

- Actualiza campos clave del usuario: cédula, nombre(s), apellido(s), usuario, correo, teléfono y rol.
- Utiliza findOrFail para asegurar que el usuario exista antes de actualizar.
- Guarda los cambios en la base de datos con $usuario->save().
- Redirige al usuario mostrando un mensaje de confirmación de la actualización.

Función Destroy

```php
public function destroy(Request $request, Usuario $usuario)
{
    $usuario = Usuario::findOrFail($request->id_usuario);

    // Eliminar lógicamente cambiando el estado
    $usuario->estado = 'Eliminado';
    $usuario->save();

    return redirect()->route('indexUsuarios')->with('success', 'Usuario eliminado correcta
}
```

Ilustración 12- Función Destroy Controlador Usuario

La función destroy permite eliminar un usuario de manera lógica, sin borrar físicamente el registro de la base de datos. Busca al usuario por su ID, cambia su estado a "Eliminado" y guarda los cambios. Después, redirige al listado de usuarios mostrando un mensaje de éxito.

Función ResetPassword

```php
public function resetPassword(Request $request, usuario $usuario)
{
    $request->validate([
        'contrasenia' => 'required|string|min:8',
    ]);

    $usuario = Usuario::findOrFail($request->id_usuario);
    $usuario->contrasenia = bcrypt($request->contrasenia);
    $usuario->save();

    return redirect()->route('indexUsuarios');
}
```

Ilustración 13- Función ResetPassword Controlador Usuario

La función resetPassword permite restablecer la contraseña de un usuario. Primero valida que la nueva contraseña cumpla con la longitud mínima requerida. Luego busca al usuario por su ID, encripta la contraseña usando bcrypt y la guarda en la base de datos. Finalmente, redirige al listado de usuarios.

Paciente_Controlador

Función Index

```php
public function index(Request $request)
{
    $buscar = strtolower($request->input('buscar'));

    $pacientes = Paciente::where('estado', 'Activo')
        ->when($buscar, function ($query, $buscar) {
            $query->where(function ($q) use ($buscar) {
                $q->whereRaw('LOWER(nombre_p) LIKE ?', ["%$buscar%"])
                    ->orWhereRaw('LOWER(apellido_p) LIKE ?', ["%$buscar%"])
                    ->orWhereRaw('LOWER(telefono) LIKE ?', ["%$buscar%"])
                    ->orWhereRaw('LOWER(cedula) LIKE ?', ["%$buscar%"]);
            });
        })
        ->orderBy('updated_at', 'desc')
        ->paginate(10);

    return view('paciente.indexPaciente', compact('pacientes'));
}
```

Ilustración 14- Función Index Controlador Paciente

La función index obtiene y muestra la lista de pacientes activos en la interfaz. Permite realizar búsquedas por nombre, apellido, cédula o teléfono de manera insensible a mayúsculas y minúsculas. Los resultados se muestran paginados, con 10 registros por página, y se ordenan por fecha de última actualización, mostrando primero los pacientes más recientes. Finalmente, envía los datos a la vista indexPaciente.

Puntos importantes:

- Filtra solo pacientes con estado “Activo”.
- La búsqueda es opcional y permite coincidencias parciales.
- Ordena los resultados por fecha de actualización (updated_at) descendente.
- Paginación de 10 pacientes por página.
- Retorna la vista indexPaciente con los datos listos para mostrar.

Función Store

```php
public function store(Request $request)
{
    // Validación
    $validatedData = $request->validate([
        'cedula' => 'required|max:13|unique:pacientes,cedula',
        'nombre_p' => 'required|string|max:60',
        'nombre_s' => 'nullable|string|max:60',
        'apellido_p' => 'required|string|max:60',
        'apellido_s' => 'nullable|string|max:60',
        'genero' => 'required|in:Masculino,Femenino',
        'fecha_nacimiento' => 'required|date',
        'telefono' => 'required|string|max:10',
        'direccion' => 'nullable|string|max:255',
    ]);

    $validatedData['estado'] = 'Activo';

    // Crear nuevo paciente
    Paciente::create($validatedData);

    // Redirigir con mensaje
    return redirect()->back()->with('success', 'Paciente registrado correctamente.');
}
```

Ilustración 15- Función Store Controlador Paciente

La función store permite registrar un nuevo paciente en el sistema. Valida que los datos ingresados cumplan con las reglas definidas, como longitud máxima, unicidad y formatos correctos. Establece automáticamente el estado del paciente como "Activo" y guarda el registro en la base de datos. Finalmente, redirige a la misma página mostrando un mensaje de éxito.

Puntos importantes:

- Valida campos obligatorios y opcionales (como segundo nombre o dirección).
- Asegura que la cédula sea única en la tabla pacientes.
- El campo genero solo permite "Masculino" o "Femenino".
- Establece automáticamente el estado del paciente como "Activo".
- Crea el registro en la tabla pacientes.
- Redirige mostrando un mensaje de confirmación.

Función Update

```php
public function update(Request $request, paciente $paciente)
{
    // Validacion
    $request->validate([
        'id_paciente' => 'required|exists:pacientes,id_paciente',
        'cedula' => 'nullable|string|max:13',
        'nombre_p' => 'required|string|max:60',
        'nombre_s' => 'nullable|string|max:60',
        'apellido_p' => 'required|string|max:60',
        'apellido_s' => 'nullable|string|max:60',
        'genero' => 'required|string|in:Masculino,Femenino',
        'fecha_nacimiento' => 'required|date',
        'telefono' => 'required|string|max:10',
        'direccion' => 'nullable|string|max:255',
    ]);

    $paciente = Paciente::findOrFail($request->id_paciente);

    // Actualizar campos
    $paciente->cedula = $request->cedula;
    $paciente->nombre_p = $request->nombre_p;
    $paciente->nombre_s = $request->nombre_s;
    $paciente->apellido_p = $request->apellido_p;
    $paciente->apellido_s = $request->apellido_s;
    $paciente->genero = $request->genero;
    $paciente->fecha_nacimiento = $request->fecha_nacimiento;
    $paciente->telefono = $request->telefono;
    $paciente->direccion = $request->direccion;

    // Guardar
    $paciente->save();

    // Redireccionar con mensaje
    return redirect()->back()->with('success', 'Paciente actualizado correctamente.');
}
```

Ilustración 16- Función Update Controlador Paciente

La función update permite modificar los datos de un paciente existente. Primero valida los datos recibidos y verifica que el paciente exista mediante su ID. Luego actualiza los campos del paciente, incluyendo cédula, nombres, apellidos, género, fecha de nacimiento, teléfono y dirección. Finalmente, guarda los cambios y redirige mostrando un mensaje de éxito.

Puntos importantes:

- Valida que el paciente exista antes de actualizar (id_paciente).
- Actualiza campos clave como cédula, nombres, apellidos, género, fecha de nacimiento, teléfono y dirección.
- Permite que algunos campos sean opcionales (segundo nombre, segundo apellido, dirección).
- Guarda los cambios en la base de datos con $paciente->save().
- Redirige mostrando un mensaje de confirmación.

Función Destroy

```php
public function destroy(Request $request)
{
    $paciente = Paciente::findOrFail($request->id_paciente);

    // Cambiar el estado a "Eliminado"
    $paciente->estado = 'Eliminado';
    $paciente->save();

    return redirect()->route('indexPaciente')->with('success', 'Paciente eliminado
}
```

Ilustración 17- Función Destroy Controlador Paciente

La función destroy permite eliminar un paciente de manera lógica. Busca al paciente por su ID, cambia su estado a “Eliminado” y guarda los cambios en la base de datos. Finalmente, redirige al listado de pacientes mostrando un mensaje de éxito.

Puntos importantes:

- No elimina físicamente el registro, realiza una eliminación lógica cambiando el estado.
- Utiliza findOrFail para asegurar que el paciente exista antes de eliminar.
- Guarda el cambio de estado con $paciente->save().
- Redirige al listado de pacientes (indexPaciente) mostrando un mensaje de confirmación.

HistorialClinco_Controlador

Función Index

```php
public function index($id_paciente)
{
    $paciente = Paciente::findOrFail($id_paciente);

    $historiales = Historial_Clinico::where('id_paciente', $id_paciente)
        ->where('estado', 'Activo')
        ->orderBy('fecha', 'desc')
        ->paginate(10);

    // Odontólogos
    $odontologos = Usuario::where('rol', 'Odontólogo')->where('estado', 'Activo')->get();

    return view('historialClinico.indexHistorialClinico', compact('paciente', 'historiales
}
```

Ilustración 18- Función Index Controlador Historia Clínica

La función index muestra el historial clínico de un paciente específico. Primero obtiene la información del paciente mediante su ID, luego busca todos los historiales clínicos relacionados que estén activos, los ordena por fecha descendente y los pagina de 10 en 10. Además, obtiene la lista de odontólogos activos para ser usada en la vista. Finalmente, envía estos datos a la vista indexHistorialClinico.

Puntos importantes:

- Obtiene al paciente por su ID (findOrFail).
- Filtra historiales clínicos activos del paciente.
- Ordena por fecha descendente (fecha).
- Paginación de 10 historiales por página.
- También recupera la lista de odontólogos activos.
- Retorna la vista indexHistorialClinico con los datos del paciente, historiales y odontólogos.

Función Store

```php
public function store(Request $request)
{
    $request->validate([
        'id_paciente' => 'required|exists:pacientes,id_paciente',
        'fecha' => 'required|date',
        'motivo_consulta' => 'nullable|string',
        'diagnostico' => 'nullable|string',
        'observaciones' => 'nullable|string',
        'pieza_dental' => 'nullable|string|max:20',
        'tipo_tratamiento' => 'nullable|string|max:100',
        'odontologo' => 'required|string|max:100',
    ]);

    $historial = new Historial_Clinico();
    $historial->id_paciente = $request->id_paciente;
    $historial->odontologo = $request->odontologo;
    $historial->fecha = $request->fecha;
    $historial->motivo_consulta = $request->motivo_consulta;
    $historial->diagnostico = $request->diagnostico;
    $historial->observaciones = $request->observaciones;
    $historial->pieza_dental = $request->pieza_dental;
    $historial->tipo_tratamiento = $request->tipo_tratamiento;
    $historial->estado = 'Activo';

    $historial->save();

    return redirect()->back()->with('success', 'Historia clínica registrada correctamente.
}
```

Ilustración 19- Función Store Controlador Historia Clínica

La función store permite registrar un nuevo historial clínico para un paciente. Valida que los datos cumplan con las reglas establecidas (fecha válida, odontólogo requerido, textos opcionales, etc.). Luego crea un nuevo objeto de tipo Historial_Clinico, asigna los datos recibidos y establece su estado como “Activo”. Guarda el registro en la base de datos y redirige mostrando un mensaje de éxito.

Puntos importantes:

- Valida que el paciente exista (id_paciente) y que los campos sean correctos.
- Registra información como fecha, motivo de consulta, diagnóstico, observaciones, pieza dental, tipo de tratamiento y odontólogo.
- Establece automáticamente el estado como “Activo”.
- Guarda el nuevo historial en la tabla historial_clinicos.
- Redirige mostrando un mensaje de confirmación.

Función Update

```php
public function update(Request $request)
{
    $historial = historial_clinico::findOrFail($request->id_historial_clinico);

    $historial->pieza_dental = $request->pieza_dental;
    $historial->tipo_tratamiento = $request->tipo_tratamiento;
    $historial->motivo_consulta = $request->motivo_consulta;
    $historial->observaciones = $request->observaciones;

    $historial->save();

    return redirect()->back()->with('success', 'Historial actualizado correctamente.');
}
```

Ilustración 20- Función Update Controlador Historia Clínica

La función update actualiza los datos de un historial clínico ya existente. Busca el historial mediante su ID, modifica los campos que pueden actualizarse (pieza dental, tipo de tratamiento, motivo de consulta y observaciones), guarda los cambios y redirige mostrando un mensaje de éxito.

Puntos importantes:

- Utiliza findOrFail para garantizar que el historial exista.
- Permite modificar solo algunos campos: pieza dental, tipo de tratamiento, motivo de consulta y observaciones.
- Guarda los cambios en la base de datos con $historial->save().
- Redirige a la misma vista mostrando un mensaje de confirmación.

Función Destroy

```php
public function destroy(Request $request)
{
    $historial = historial_clinico::findOrFail($request->id_historial_clinico);

    // Eliminación lógica
    $historial->estado = 'Eliminado';
    $historial->save();

    return redirect()->back()->with('success', 'Historial clínico eliminado correctamente.
}
```

Ilustración 21- Función Destroy Controlador Historia Clínica

La función destroy elimina un historial clínico de manera lógica. Busca el historial mediante su ID, cambia su estado a “Eliminado” y guarda los cambios. No elimina físicamente el registro, lo que permite mantener trazabilidad en la base de datos. Finalmente, redirige mostrando un mensaje de éxito.

Puntos importantes:

- Busca el historial clínico por ID con findOrFail.
- Realiza eliminación lógica cambiando el estado a “Eliminado”.
- No borra el registro físicamente de la base de datos.
- Guarda el cambio de estado con $historial->save().
- Redirige a la vista anterior mostrando un mensaje de confirmación.

Cita_Controlador

Función Index

```php
public function index(Request $request)
{
    $cedulaPaciente = $request->input('cedula');
    $fecha = $request->input('fecha');

    $citas = Citas::with('paciente', 'usuario')
        ->where('estado', '!=', 'Eliminada')
        ->when($cedulaPaciente, function ($query) use ($cedulaPaciente) {
            $query->whereHas('paciente', function ($q) use ($cedulaPaciente) {
                $q->where('cedula', 'like', "%{$cedulaPaciente}%");
            });
        })
        ->when($fecha, function ($query) use ($fecha) {
            $query->whereDate('fecha', $fecha);
        })
        ->orderBy('updated_at', 'desc')
        ->paginate(10);

    $pacientes = Paciente::all();
    $odontologos = Usuario::where('rol', 'Odontólogo')->get();

    return view('citas/indexCitas', compact('citas', 'pacientes', 'odontologos'));
}
```

Ilustración 22- Función Index Controlador Citas

Se encarga de listar todas las citas registradas en el sistema, permitiendo filtros por cédula del paciente y por fecha de la cita. Solo muestra las citas que no han sido eliminadas. Además, obtiene la lista completa de pacientes y odontólogos para facilitar la asignación en la vista.

Puntos importantes:

- Filtra citas por cédula del paciente y/o fecha.
- Excluye citas con estado Eliminada.
- Ordena las citas por fecha de actualización descendente.
- Usa with() para traer datos relacionados de paciente y usuario.
- Paginación de 10 citas por página.
- Retorna la vista indexCitas con citas, pacientes y odontólogos.

Función BuscarPacientePorCedula

```php
public function buscarPacientePorCedula(Request $request)
{
    $cedula = $request->input('cedula');
    $paciente = Paciente::where('cedula', $cedula)->first();

    if ($paciente) {
        return response()->json([
            'existe' => true,
            'paciente' => $paciente
        ]);
    } else {
        return response()->json([
            'existe' => false
        ]);
    }
}
```

Ilustración 23- Función Buscar Paciente Por Cedula Controlador Citas

Permite buscar si un paciente existe en la base de datos a partir de su cédula. Devuelve una respuesta en formato JSON, con los datos del paciente en caso de existir.

Puntos importantes:

- Busca paciente por cédula con where.
- Retorna JSON con existe = true y los datos del paciente si lo encuentra.
- Retorna JSON con existe = false si no lo encuentra.
- Útil para búsquedas rápidas en formularios mediante AJAX.

Función VerCalendario

```php
//Calendario
public function verCalendario()
{
    $citas = Citas::with('paciente')->where('estado', '!=', 'Eliminada')->get();

    $eventos = $citas->map(function ($cita) {
        // Definir el color según el estado
        $color = match ($cita->estado) {
            'Pendiente' => '#f39c12',   // Naranja
            'Confirmada' => '#28a745',  // Verde
            'Cancelada' => '#dc3545',   // Rojo
            default => '#6c757d',       // Gris (para cualquier otro estado)
        };

        return [
            'title' => $cita->odontologo . ' - ' . $cita->paciente->nombre_p . ' ' . $cita->paciente->apellido_p,
            'start' => $cita->fecha . 'T' . $cita->hora,
            'color' => $color,

            // Propiedades para el modal
            'paciente' => $cita->paciente->nombre_p . ' ' . $cita->paciente->apellido_p,
            'odontologo' => $cita->odontologo,
            'fecha' => $cita->fecha,
            'hora' => $cita->hora,
            'estado' => $cita->estado,
        ];
    });

    return view('citas/calendarioCitas', ['eventos' => $eventos]);
}
```

Ilustración 24- Función Ver Calendario Controlador Citas

Genera los datos necesarios para mostrar las citas en un calendario interactivo. Cada cita se muestra como un evento, con colores que dependen de su estado.

Puntos importantes:

- Obtiene todas las citas que no estén eliminadas.
- Trae información relacionada con el paciente.
- Mapea cada cita a un evento con: título, fecha/hora, color y datos adicionales.
- Colores definidos:
    - Pendiente → naranja.
    - Confirmada → verde.
    - Cancelada → rojo.
- Otros estados → gris.
- Retorna la vista calendarioCitas con los eventos.

Función Store

```php
public function store(Request $request)
{

    $request->validate([
        'cedulaPaciente' => 'required|string|exists:pacientes,cedula',
        'id_paciente' => 'required|integer|exists:pacientes,id_paciente',
        'id_usuario' => 'required|integer|exists:usuarios,id_usuario',
        'odontologo' => 'required|string',
        'fecha' => 'required|date',
        'hora' => 'required',
    ]);

    Citas::create([
        'id_paciente' => $request->id_paciente,
        'id_usuario' => $request->id_usuario,
        'odontologo' => $request->odontologo,
        'fecha' => $request->fecha,
        'hora' => $request->hora,
        'estado' => 'Pendiente' // <-- Fijamos siempre este valor
    ]);

    return redirect()->back()->with('success', 'Cita agendada correctamente.');
}
```

Ilustración 25- Función Store Controlador Citas

Registra una nueva cita en el sistema después de validar los datos. Siempre se guarda con estado inicial Pendiente.

Puntos importantes:

- Valida que paciente y usuario existan en la base de datos.
- Campos requeridos: cédula del paciente, odontólogo, fecha y hora.
- Crea un registro en la tabla citas con estado fijo Pendiente.
- Redirige con mensaje de éxito.

Función CambiarEstado

```php
public function cambiarEstado(Request $request)
{
    $cita = Citas::findOrFail($request->input('id_cita'));
    $cita->estado = $request->input('estado');
    $cita->save();

    return redirect()->route('indexCitas')->with('success', 'Estado actualizado correctamente.');
}
```

Ilustración 26- Función Cambiar Estado Controlador Citas

Permite cambiar el estado de una cita específica.

Puntos importantes:

- Busca la cita por ID.
- Actualiza el campo estado con el valor recibido en la petición.
- Guarda cambios en la base de datos.
- Redirige a indexCitas con un mensaje de confirmación.

Función Destroy

```php
public function destroy(Request $request, Citas $cita)
{

    $cita = Citas::findOrFail($request->id_cita);
    $cita->estado = 'Eliminada';
    $cita->save();

    return redirect()->route('indexCitas');
}
```

Ilustración 27- Función Destroy Controlador Citas

Realiza la eliminación lógica de una cita, sin borrarla físicamente de la base de datos.

Puntos importantes:

- Busca la cita por ID con findOrFail.
- Cambia su estado a Eliminada.
- Guarda los cambios en la base de datos.
- Redirige a indexCitas.

Función Update

```php
public function update(Request $request, citas $citas)
{

    // Validar los datos del formulario
    $request->validate([
        'id_paciente' => 'required|integer|exists:pacientes,id_paciente',
        'id_usuario' => 'required|integer|exists:usuarios,id_usuario',
        'odontologo' => 'required|string',
        'fecha' => 'required|date',
        'hora' => 'required',
        'estado' => 'required|string|in:Pendiente,Confirmada,Cancelada,Atendida'
    ]);

    // Buscar la cita por ID
    $citas = Citas::findOrFail($request->id_cita);

    // Actualizar los datos
    $citas->update([
        'id_paciente' => $request->id_paciente,
        'id_usuario' => $request->id_usuario,
        'odontologo' => $request->odontologo,
        'fecha' => $request->fecha,
        'hora' => $request->hora,
        'estado' => 'Pendiente',
    ]);

    // Redirigir con mensaje de éxito
    return redirect()->back()->with('success', 'Cita actualizada correctamente.');
}
```

Ilustración 28- Función Update Controlador Citas

Actualiza la información de una cita existente.

Puntos importantes:

- Valida que paciente y usuario existan.
- Requiere odontólogo, fecha, hora y estado válido (Pendiente, Confirmada, Cancelada, Atendida).
- Busca la cita por ID.
- Actualiza los datos de la cita, aunque en el código actual el estado siempre se fuerza a Pendiente (posible ajuste futuro).
- Redirige con mensaje de éxito.

Reporte_Controlador

Función Index

```php
public function index()
{
    return view('reportes.indexReportes');
}
```

Ilustración 29- Función Index Controlador Reporte

La función index() se encarga de mostrar la página principal del módulo de reportes. Su propósito es servir como punto de inicio para que el usuario pueda acceder a los diferentes tipos de reportes que ofrece el sistema. No realiza consultas ni genera información adicional, simplemente carga la vista donde se encuentra el menú con las opciones de reportes disponibles.

Función Vista Historial Por Paciente

```php
public function vistaHistorialPorPaciente()
{
    $pacientes = Paciente::where('estado', 'Activo')->paginate(10);
    return view('reportes.historialPorPaciente', compact('pacientes'));
}
```

Ilustración 30- Función Vista Historial Por Paciente Controlador Reporte

La función vistaHistorialPorPaciente() tiene como finalidad obtener una lista de pacientes que se encuentran activos en el sistema, mostrando esta información de forma paginada. Con esta vista, el usuario puede seleccionar un paciente específico y a partir de ahí generar un reporte detallado de su historial clínico. Se convierte, por lo tanto, en una pantalla intermedia que facilita la selección antes de crear el documento final.

Función Generar Historial PDF

```php
public function generarHistorialPDF($id)
{
    $paciente = Paciente::findOrFail($id);
    $historiales = Historial_Clinico::where('id_paciente', $id)->get();

    $nombreCompleto = "{$paciente->nombre_p} {$paciente->nombre_s} {$paciente->apellido_p} {$paciente->

    Reporte::create([
        'tipo_reporte' => "Historial clínico de $nombreCompleto",
        'id_usuario' => session('id'),
        'generado_el' => now(),
    ]);

    $pdf = Pdf::loadView('reportes.pdf.pdfHistorialPaciente', compact('paciente', 'historiales'));
    return $pdf->stream("historial_{$paciente->apellido_p}_{$paciente->nombre_p}.pdf");
}
```

Ilustración 31- Función Generar Historial PDF Controlador Reporte

La función generarHistorialPDF($id) está diseñada para elaborar un reporte completo del historial clínico de un paciente en particular. Una vez que se selecciona el paciente, se recupera toda la información almacenada en la base de datos referente a sus consultas, diagnósticos y tratamientos. Además, al momento de generar el archivo PDF con esos datos, se guarda también un registro en la tabla de reportes para llevar un control de quién lo generó y cuándo.

Función Historial Por Paciente PDF

```php
public function historialPorPacientePDF($id_paciente)
{
    $paciente = Paciente::findOrFail($id_paciente);
    $historiales = Historial_Clinico::where('id_paciente', $id_paciente)
        ->where('estado', 'Activo')
        ->orderBy('fecha', 'desc')
        ->get();

    $pdf = Pdf::loadView('reportes.pdf.pdfHistorialPaciente', compact('paciente', 'historiales'));
    return $pdf->stream('historial_' . $paciente->cedula . '.pdf');
}

public function vistaPorOdontologo()
{
    $odontologos = Usuario::where('rol', 'Odontólogo')->get();

    return view('reportes.pacientePorOdontologo', compact('odontologos'));
}
```

Ilustración 32- Función Historial Por Paciente Controlador Reporte

La función historialPorPacientePDF($id_paciente) cumple una labor similar, pero con un enfoque más filtrado. En este caso, únicamente se obtienen los historiales clínicos que se encuentren activos y se presentan ordenados desde el más reciente al más antiguo. El resultado también se plasma en un archivo PDF, lo cual lo convierte en una versión más precisa y actualizada del historial del paciente.

Función Vista Por Odontologo

```php
public function vistaPorOdontologo()
{
    $odontologos = Usuario::where('rol', 'Odontólogo')->get();

    return view('reportes.pacientePorOdontologo', compact('odontologos'));
}
```

Ilustración 33- Función Vista Por Odontólogo Controlador Reporte

La función vistaPorOdontologo() permite visualizar la lista de usuarios que tienen el rol de odontólogos. De esta manera, se abre la posibilidad de elegir

a un profesional específico para posteriormente generar un reporte de los pacientes que han sido atendidos por él. Es otra pantalla de apoyo que organiza la información antes de llegar al documento final.

Función Generar Por Odontologo

```php
public function generarPorOdontologo(Request $request)
{
    $idOdontologo = $request->input('id_odontologo');

    $odontologo = Usuario::find($idOdontologo);

    if (!$odontologo) {
        return back()->with('error', 'Odontólogo no encontrado');
    }

    $nombreCompleto = $odontologo->nombre_p . ' ' .
        $odontologo->nombre_s . ' ' .
        $odontologo->apellido_p . ' ' .
        $odontologo->apellido_s;

    $pacientes = DB::table('historial_clinicos AS h')
        ->join('pacientes AS p', 'h.id_paciente', '=', 'p.id_paciente')
        ->where('h.odontologo', $nombreCompleto)
        ->select(
            DB::raw("p.nombre_p || ' ' || p.nombre_s || ' ' || p.apellido_p || ' ' || p.apellido_s AS nombre_completo"),
            'h.tipo_tratamiento',
            'h.fecha'
        )
        ->get();

    $pdf = Pdf::loadView('reportes.pdf.pdfPacientesPorOdontologo', compact('pacientes', 'odontologo'));

    Reporte::create([
        'tipo_reporte' => 'Historial por odontólogo',
        'id_usuario' => $idOdontologo,
    ]);

    return $pdf->stream('reporte-odontologo.pdf');
}
```

Ilustración 34- Función Generar Por Odontólogo Controlador Reporte

La función generarPorOdontologo(Request $request) es la que se encarga de crear el reporte de pacientes atendidos por un odontólogo en particular. Al seleccionar al profesional, el sistema recupera todos los registros de las atenciones que este realizó, incluyendo los nombres de los pacientes, los tratamientos aplicados y las fechas correspondientes. Con esta información se genera un archivo PDF y, al igual que en otros casos, se deja constancia en la base de datos de que se elaboró un informe de este tipo.

Función Citas Hoy

```php
// Generar PDF de citas de hoy
public function citasHoy(Request $request)
{
    $idOdontologo = $request->input('id_odontologo');

    $odontologo = Usuario::find($idOdontologo);

    // Fecha actual
    $hoy = Carbon::now('America/Guayaquil')->toDateString();

    $citas = DB::table('citas as c')
        ->join('pacientes as p', 'p.id_paciente', '=', 'c.id_paciente')
        ->select(
            DB::raw("p.nombre_p || ' ' || p.nombre_s || ' ' || p.apellido_p || ' ' || p.apellido_s AS nombre_completo"),
            'c.fecha',
            'c.hora',
            'c.odontologo'
        )
        ->whereDate('c.fecha', $hoy)
        ->get();

    // Generar PDF
    $pdf = Pdf::loadView('reportes.pdf.pdfCitasHoy', compact('citas', 'hoy'));

    // Guardar registro en la tabla reportes
    Reporte::create([
        'tipo_reporte' => 'Citas del día',
        'id_usuario' => session('id'),
        'generado_el' => now(),// Guarda el usuario logueado que generó el reporte
    ]);

    // Abrir en nueva pestaña
    return $pdf->stream('Citas_Hoy.pdf');
}
```

Ilustración 35- Función Citas Hoy Controlador Reporte

La función citasHoy(Request $request) tiene como propósito principal obtener todas las citas agendadas en la fecha actual. Se recopilan los datos del paciente, la hora de la cita y el odontólogo encargado. Con esa información se construye un PDF que refleja de manera ordenada el cronograma de atenciones del día. También se registra en la tabla de reportes, lo que permite mantener un historial de las veces que se generan estos documentos.

4.2.3. Views

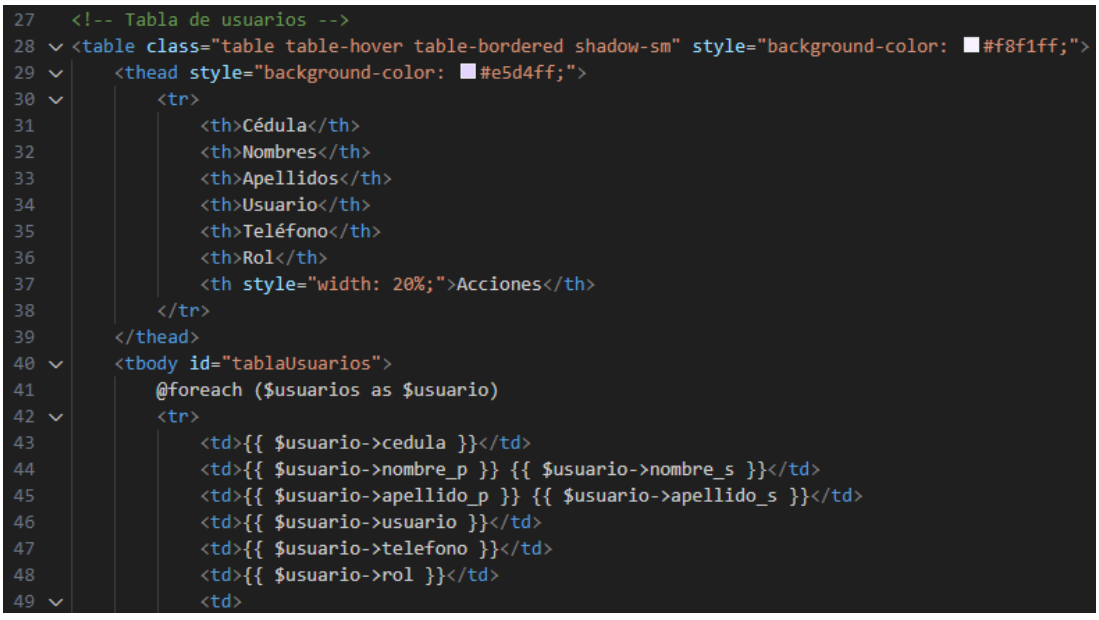
/resources/views: En este apartado se encontrarán todas las vistas en Blade organizadas por módulos (usuarios, pacientes, citas, reportes).

Para este manual se documentará únicamente la vista correspondiente al módulo de Usuarios, dado que las demás vistas (Pacientes, Citas, Historial Clínico y Reportes) siguen la misma lógica de desarrollo, diseño y estructura.

De esta forma, se evita la repetición innecesaria de capturas y explicaciones, manteniendo el documento más claro y conciso. En caso de ser necesario, el programador podrá replicar la misma metodología aplicada en el módulo de Usuarios para los demás módulos del sistema.

```
@extends('layouts.admin')

@section('titulo', 'Gestión-Usuarios')

@section('contenido')
<!-- Título y botón de crear -->
<div class="d-flex justify-content-between align-items-center mb-4">
    <h3 class="text-purple fw-bold" style="color: #6f42c1;">Gestión de Usuarios</h3>
    <button type="button" class="btn btn-primary" style="background-color: #6f42c1; border-color: #6f42c1;" data-bs-toggle="modal" data-bs-target="#crearUsuarioModal
        Crear Usuario
    </button>
</div>

<!-- Incluir el modal crear usuario -->
@include('usuario.crearUsuario')

<!-- Buscador -->
<form method="GET" action="{{ route('indexUsuarios') }}">
    <div class="input-group mb-3">
        <input type="text" name="buscar" id="campoBuscar" class="form-control" placeholder="Buscar por nombre, apellido, usuario, teléfono o cédula" value="{{ request('b
        <button class="btn btn-primary" type="submit">Buscar</button>
        <button class="btn btn-secondary" type="button" onclick="limpiarBuscador()">Limpiar</button>
    </div>
</form>
```

Ilustración 36- Vista Usuario P1



```
<!-- Tabla de usuarios -->
<table class="table table-hover table-bordered shadow-sm" style="background-color: #f8f1ff;">
    <thead style="background-color: #e5d4ff;">
        <tr>
            <th>Cédula</th>
            <th>Nombres</th>
            <th>Apellidos</th>
            <th>Usuario</th>
            <th>Teléfono</th>
            <th>Rol</th>
            <th style="width: 20%;">Acciones</th>
        </tr>
    </thead>
    <tbody id="tablaUsuarios">
        @foreach ($usuarios as $usuario)
        <tr>
            <td>{{ $usuario->cedula }}</td>
            <td>{{ $usuario->nombre_p }} {{ $usuario->nombre_s }}</td>
            <td>{{ $usuario->apellido_p }} {{ $usuario->apellido_s }}</td>
            <td>{{ $usuario->usuario }}</td>
            <td>{{ $usuario->telefono }}</td>
            <td>{{ $usuario->rol }}</td>
            <td>
```

Ilustración 37- Vista Usuario P2

```
                <div class="d-flex justify-content-center align-items-center gap-2">

                    <a href="#" class="btn btn-sm btn-warning d-flex align-items-center"
                        data-bs-toggle="modal" data-bs-target="#editarUsuarioModal"
                        data-usuario='@json($usuario)'
                        onclick="mostrarModalEditar(this)">
                        <i class="bi bi-pencil-square me-1"></i> Editar
                    </a>

                    <a href="#" class="btn btn-sm btn-danger d-flex align-items-center"
                        data-bs-toggle="modal" data-bs-target="#eliminarUsuarioModal"
                        data-usuario='@json($usuario)'
                        onclick="mostrarModalEliminar(this)">
                        <i class="bi bi-trash me-1"></i> Eliminar
                    </a>

                    <a href="#" class="btn btn-sm btn-secondary d-flex align-items-center"
                        data-bs-toggle="modal" data-bs-target="#resetearUsuarioModal"
                        data-usuario='@json($usuario)'
                        onclick="mostrarModalReset(this)">
                        <i class="bi bi-key me-1"></i> Resetear
                    </a>


                </div>
            </td>

        </tr>
        @endforeach

    </tbody>

</table>
```

Ilustración 38- Vista Usuario P2

```
<!-- Controles de paginación -->
<div class="d-flex justify-content-center mt-3">
    {{ $usuarios->links() }}
</div>

@include('usuario.editarUsuario')
@include('usuario.eliminarUsuario')
@include('usuario.contraseniaUsuario')

<!-- Script para limpiar -->
<script>
    function limpiarBuscador() {
        document.getElementById('campoBuscar').value = '';
        window.location.href = "{{ route('indexUsuarios') }}"; // Redirecciona sin filtros
    }
</script>

@endsection
```

Ilustración 39- Vista Usuario P3

En la presente vista de gestión de usuarios se muestra el panel principal donde se pueden realizar las operaciones básicas como crear, buscar, editar, eliminar y resetear contraseñas. La interfaz está organizada con un buscador, una tabla que lista a los usuarios registrados y botones de acción para cada uno.

A continuación, se incluyen los respectivos modales para la creación, edición, eliminación y reseteo de usuarios. Cabe señalar que solo se presentarán estos modales como ejemplo, ya que en los demás módulos la estructura es muy similar y resultaría repetitivo mostrar todos de manera detallada.

Modal Crear Usuario

```blade
<!-- Modal Crear Usuario -->
<div class="modal fade" id="crearUsuarioModal" tabindex="-1" aria-labelledby="crearUsuarioModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <form action="{{ route('crearUsuario') }}" method="POST" class="modal-content">
            @csrf
            <div class="modal-header">
                <h5 class="modal-title" id="crearUsuarioModalLabel">Crear Nuevo Usuario</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Cerrar"></button>
            </div>
            <div class="modal-body">
                <!-- Cédula -->
                <div class="mb-3">
                    <label for="cedula" class="form-label">Cédula</label>
                    <input type="text" class="form-control" id="cedula" name="cedula" placeholder="Ingrese la cédula" required>
                </div>

                <!-- Nombres lado a lado -->
                <div class="row g-3 mb-3">
                    <div class="col">
                        <label for="nombre_p" class="form-label">Primer Nombre</label>
                        <input type="text" class="form-control" id="nombre_p" name="nombre_p" placeholder="Primer Nombre" required>
                    </div>
                    <div class="col">
                        <label for="nombre_s" class="form-label">Segundo Nombre</label>
                        <input type="text" class="form-control" id="nombre_s" name="nombre_s" placeholder="Segundo Nombre">
                    </div>
                </div>

                <!-- Apellidos lado a lado -->
                <div class="row g-3 mb-3">
                    <div class="col">
                        <label for="apellido_p" class="form-label">Primer Apellido</label>
                        <input type="text" class="form-control" id="apellido_p" name="apellido_p" placeholder="Primer Apellido" required>
                    </div>
                    <div class="col">
                        <label for="apellido_s" class="form-label">Segundo Apellido</label>
                        <input type="text" class="form-control" id="apellido_s" name="apellido_s" placeholder="Segundo Apellido">
                    </div>
                </div>
```

Ilustración 40- Modal Crear Usuario P1

```blade
                <!-- Usuario y Contraseña lado a lado -->
                <div class="row g-3 mb-3">
                    <div class="col">
                        <label for="usuario" class="form-label">Usuario</label>
                        <input type="text" class="form-control" id="usuario" name="usuario" placeholder="Nombre de usuario" required>
                    </div>
                    <div class="col">
                        <label for="contrasenia" class="form-label">Contraseña</label>
                        <input type="password" class="form-control" id="contrasenia" name="contrasenia" placeholder="Contraseña" required>
                    </div>
                </div>
```

Ilustración 41- Modal Crear Usuario P2

```
                <!-- Correo -->
                <div class="mb-3">
                    <label for="correo" class="form-label">Correo electrónico</label>
                    <input type="email" class="form-control" id="correo" name="correo" placeholder="ejemplo@dominio.com" required>
                </div>

                <div class="row g-3 mb-3">
                    <!-- Teléfono -->
                    <div class="col">
                        <label for="telefono" class="form-label">Teléfono</label>
                        <input type="text" class="form-control" id="telefono" name="telefono" placeholder="Número de teléfono" required>
                    </div>

                    <!-- Rol -->
                    <div class="col">
                        <label for="rol" class="form-label">Rol</label>
                        <select class="form-select" id="rol" name="rol" required>
                            <option value="" selected disabled>Seleccione un rol</option>
                            <option value="Secretaria">Secretaria</option>
                            <option value="Odontólogo">Odontólogo</option>
                            <option value="Administrador">Administrador</option>
                        </select>
                    </div>
                </div>


            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cancelar</button>
                <button type="submit" class="btn btn-primary" style="background-color: #6f42c1; border-color: #6f42c1;">Guardar</button>
            </div>
        </form>
    </div>
</div>
```

Ilustración 42- Modal Crear Usuario P3

Modal Editar Usuario

```
<!-- Modal Editar Usuario -->
<div class="modal fade" id="editarUsuarioModal" tabindex="-1" aria-labelledby="editarUsuarioModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <form action="{{ route('editarUsuario') }}" method="POST" class="modal-content">
            @csrf
            @method('PUT')
            <input type="hidden" id="editar_id" name="id_usuario">

            <!-- Encabezado amarillo -->
            <div class="modal-header bg-warning text-white">
                <h5 class="modal-title" id="editarUsuarioModalLabel">Editar Usuario</h5>
                <button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal" aria-label="Cerrar"></button>
            </div>

            <div class="modal-body">
                <!-- Cédula -->
                <div class="mb-3">
                    <label for="editar_cedula" class="form-label">Cédula</label>
                    <input type="text" class="form-control" id="editar_cedula" name="cedula" required>
                </div>

                <!-- Nombres -->
                <div class="row g-3 mb-3">
                    <div class="col">
                        <label for="editar_nombre_p" class="form-label">Primer Nombre</label>
                        <input type="text" class="form-control" id="editar_nombre_p" name="nombre_p" required>
                    </div>
                    <div class="col">
                        <label for="editar_nombre_s" class="form-label">Segundo Nombre</label>
                        <input type="text" class="form-control" id="editar_nombre_s" name="nombre_s">
                    </div>
                </div>

                <!-- Apellidos -->
                <div class="row g-3 mb-3">
                    <div class="col">
                        <label for="editar_apellido_p" class="form-label">Primer Apellido</label>
                        <input type="text" class="form-control" id="editar_apellido_p" name="apellido_p" required>
                    </div>
                    <div class="col">
                        <label for="editar_apellido_s" class="form-label">Segundo Apellido</label>
```

Ilustración 43- Modal Editar Usuario P1

```
                    <input type="text" class="form-control" id="editar_apellido_s" name="apellido_s">
                </div>
            </div>

            <!-- Usuario -->
            <div class="mb-3">
                <label for="editar_usuario" class="form-label">Usuario</label>
                <input type="text" class="form-control" id="editar_usuario" name="usuario" required>
            </div>

            <!-- Teléfono y Correo lado a lado -->
            <div class="row g-3 mb-3">
                <div class="col">
                    <label for="editar_telefono" class="form-label">Teléfono</label>
                    <input type="text" class="form-control" id="editar_telefono" name="telefono" required>
                </div>
                <div class="col">
                    <label for="editar_correo" class="form-label">Correo electrónico</label>
                    <input type="email" class="form-control" id="editar_correo" name="correo" required>
                </div>
            </div>

            <!-- Rol -->
            <div class="mb-3">
                <label for="editar_rol" class="form-label">Rol</label>
                <select class="form-select" id="editar_rol" name="rol" required>
                    <option value="" disabled>Seleccione un rol</option>
                    <option value="Secretaria">Secretaria</option>
                    <option value="Odontólogo">Odontólogo</option>
                    <option value="Administrador">Administrador</option>
                </select>
            </div>
        </div>
```

Ilustración 44- Modal Editar Usuario P2

```
            <!-- Pie con botón amarillo -->
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cancelar</button>
                <button type="submit" class="btn btn-warning">Actualizar</button>
            </div>
        </form>
    </div>
</div>

<!--Llenar campos de editar-->
<script>
    function mostrarModalEditar(elemento) {

        const usuario = JSON.parse(elemento.getAttribute('data-usuario'));

        document.getElementById('editar_id').value = usuario.id_usuario;
        document.getElementById('editar_cedula').value = usuario.cedula;
        document.getElementById('editar_nombre_p').value = usuario.nombre_p;
        document.getElementById('editar_nombre_s').value = usuario.nombre_s;
        document.getElementById('editar_apellido_p').value = usuario.apellido_p;
        document.getElementById('editar_apellido_s').value = usuario.apellido_s;
        document.getElementById('editar_usuario').value = usuario.usuario;
        document.getElementById('editar_telefono').value = usuario.telefono;
        document.getElementById('editar_correo').value = usuario.correo;
        document.getElementById('editar_rol').value = usuario.rol;
    }
</script>
```

Ilustración 45- Modal Editar Usuario P3

Modal Eliminar Usuario

```blade
<!-- Modal Eliminar Usuario -->
<div class="modal fade" id="eliminarUsuarioModal" tabindex="-1" aria-labelledby="eliminarUsuarioModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <form action="{{ route('eliminarUsuario') }}" method="POST" >
            @csrf
            @method('DELETE')
            <input type="hidden" id="eliminar_id" name="id_usuario">

            <div class="modal-content">
                <div class="modal-header bg-danger text-white">
                    <h5 class="modal-title" id="eliminarUsuarioModalLabel">Confirmar Eliminación</h5>
                    <button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal" aria-label="Cerrar"></button>
                </div>
                <div class="modal-body">
                    ¿Estás seguro de que deseas eliminar al usuario <strong id="nombreEliminar"></strong>?
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cancelar</button>
                    <button type="submit" class="btn btn-danger">Eliminar</button>
                </div>
            </div>
        </form>
    </div>
</div>

<script>
    function mostrarModalEliminar(elemento) {
        const usuario = JSON.parse(elemento.getAttribute('data-usuario'));

        document.getElementById('eliminar_id').value = usuario.id_usuario;
        document.getElementById('nombreEliminar').textContent = usuario.nombre_p + ' ' + usuario.apellido_p;
    }
</script>
```

Ilustración 46- Modal Eliminar Usuario P1

Modal Resetear Contraseña Usuario

```blade
C:\Users\USUARIO\Desktop\SistemaGestionDental\resources
<div class="modal fade" id="resetearUsuarioModal" tabindex="-1" aria-labelledby="resetearUsuarioModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <form action="{{ route('resetearUsuario') }}" method="POST" id="formResetearUsuario">
            @csrf
            @method('PUT')
            <input type="hidden" id="reset_id" name="id_usuario"> <!-- ID oculto del usuario -->

            <div class="modal-content">
                <div class="modal-header bg-secondary text-white">
                    <h5 class="modal-title" id="resetearUsuarioModalLabel">Resetear Contraseña</h5>
                    <button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal" aria-label="Cerrar"></button>
                </div>
                <div class="modal-body">
                    <p>Ingrese una nueva contraseña para <strong id="nombreReset"></strong>:</p>

                    <!-- Campo Nueva Contraseña -->
                    <div class="mb-3">
                        <label for="nueva_contrasenia" class="form-label">Nueva Contraseña</label>
                        <input type="password" id="nueva_contrasenia" name="contrasenia" class="form-control" value="12345678" required>
                    </div>

                    <!-- Campo Confirmar Contraseña -->
                    <div class="mb-3">
                        <label for="confirmar_contrasenia" class="form-label">Confirmar Contraseña</label>
                        <input type="password" id="confirmar_contrasenia" class="form-control" value="12345678" required>
                    </div>

                    <!-- Checkbox para ver contraseña -->
                    <div class="form-check">
                        <input class="form-check-input" type="checkbox" id="verClave">
                        <label class="form-check-label" for="verClave">Mostrar contraseña</label>
                    </div>

                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-outline-secondary" data-bs-dismiss="modal">Cancelar</button>
                    <button type="submit" class="btn btn-secondary">Resetear</button>
                </div>
            </div>
        </form>
    </div>
</div>
```

Ilustración 47- Modal Resetear Contraseña P1

```
<script>
    function mostrarModalReset(elemento) {
        const usuario = JSON.parse(elemento.getAttribute('data-usuario'));

        document.getElementById('reset_id').value = usuario.id_usuario;
        document.getElementById('nombreReset').textContent = usuario.nombre_p + ' ' + usuario.apellido_p;

        // Resetear campos por defecto
        document.getElementById('nueva_contrasenia').value = '12345678';
        document.getElementById('confirmar_contrasenia').value = '12345678';

        // Ocultar clave por defecto
        document.getElementById('verClave').checked = false;
        document.getElementById('nueva_contrasenia').type = 'password';
        document.getElementById('confirmar_contrasenia').type = 'password';

    }

    //Ver clave
    document.getElementById('verClave').addEventListener('change', function () {
        const pass1 = document.getElementById('nueva_contrasenia');
        const pass2 = document.getElementById('confirmar_contrasenia');
        const tipo = this.checked ? 'text' : 'password';
        pass1.type = tipo;
        pass2.type = tipo;
    });

    // Validar que ambas contraseñas coincidan
    document.getElementById('formResetearUsuario').addEventListener('submit', function(e) {
        const pass1 = document.getElementById('nueva_contrasenia').value;
        const pass2 = document.getElementById('confirmar_contrasenia').value;

        if (pass1 !== pass2) {
            e.preventDefault();
            alert('Las contraseñas no coinciden.');
        }
    });
</script>
```

Ilustración 48- Modal Resetear Contraseña P2

4.3. Migraciones

4.3.1. Usuario

```php
public function up(): void
{
    Schema::create('usuarios', function (Blueprint $table) {
        $table->id('id_usuario');

        $table->string('cedula', 13)->nullable(false);
        $table->string('nombre_p', 60)->nullable(false);
        $table->string('nombre_s', 60)->nullable(false);
        $table->string('apellido_p', 60)->nullable(false);
        $table->string('apellido_s', 60)->nullable(false);
        $table->string('usuario')->unique()->nullable(false);
        $table->string('contrasenia')->nullable(false);
        $table->string('telefono', 10)->nullable(false);
        $table->string('correo', 100)->nullable(false);
        $table->string('rol', 60)->nullable(false);
        $table->string('estado');

        $table->timestamps();
    });
}
```

Ilustración 49- Migración de Usuario

La migración tiene como finalidad la creación de la tabla usuarios, la cual almacena la información básica de las personas que interactúan con el sistema.

Dentro de la estructura definida se incluyen los siguientes campos principales:

- **id_usuario:** clave primaria autoincremental que identifica de manera única a cada usuario.
- **cedula:** número de cédula del usuario, con una longitud máxima de 13 caracteres, obligatorio.
- **nombre_p y nombre_s:** nombres del usuario, cada uno con un máximo de 60 caracteres.
- **apellido_p y apellido_s:** apellidos del usuario, también limitados a 60 caracteres.
- **usuario:** nombre único de usuario que servirá para el inicio de sesión en el sistema.
- **contrasenia:** contraseña asociada a la cuenta, almacenada de forma obligatoria.
- **telefono:** número de contacto del usuario, con un máximo de 10 dígitos.

- **correo:** dirección de correo electrónico, con una longitud máxima de 100 caracteres.
- **rol:** rol asignado al usuario dentro del sistema (ejemplo: Administrador, Odontólogo, Secretaria).
- **estado:** campo que determina si el usuario está activo o inactivo en el sistema.
- **timestamps:** campos created_at y updated_at, generados automáticamente por Laravel para registrar la fecha de creación y última actualización del registro.

4.3.2. Paciente

```php
public function up(): void
{
    Schema::create('pacientes', function (Blueprint $table) {;
        $table->id('id_paciente');

        $table->string('cedula', 13)->nullable(true);
        $table->string('nombre_p', 60)->nullable(true);
        $table->string('nombre_s', 60)->nullable(true);
        $table->string('apellido_p', 60)->nullable(true);
        $table->string('apellido_s', 60)->nullable(true);
        $table->string('genero', 30)->nullable(true);
        $table->date('fecha_nacimiento')->nullable(true);
        $table->string('telefono', 10);
        $table->string('direccion', 100)->nullable(true);
        $table->string('estado');

        $table->timestamps();
    });
}
```

Ilustración 50- Migración de Paciente

La migración define la estructura de la tabla pacientes, encargada de almacenar la información personal y de contacto de cada paciente registrado en el consultorio odontológico.

Los principales campos incluidos son:

- **id_paciente:** clave primaria autoincremental que identifica de manera única a cada paciente.
- **cedula:** número de identificación personal del paciente, con un máximo de 13 caracteres (puede ser nulo en caso de no disponerlo).
- **nombre_p y nombre_s:** nombres del paciente, con una longitud máxima de 60 caracteres.

- **apellido_p y apellido_s:** apellidos del paciente, también con un límite de 60 caracteres.
- **genero:** define el género del paciente (por ejemplo: masculino, femenino, otro).
- **fecha_nacimiento:** fecha de nacimiento del paciente, permite un mejor control de la edad.
- **telefono:** número de contacto del paciente, limitado a 10 dígitos.
- **direccion:** dirección de residencia del paciente, hasta 100 caracteres.
- **estado:** campo que indica si el paciente se encuentra activo en el sistema o ha sido deshabilitado.
- **timestamps:** campos automáticos created_at y updated_at, para registrar la creación y última actualización de cada registro.

4.3.3. Historial Clínico

```php
public function up(): void
{
    Schema::create('historial_clinicos', function (Blueprint $table) {

        $table->id('id_historial_clinico');

        // Relaciones
        $table->unsignedBigInteger('id_paciente');

        // Información médica
        $table->date('fecha')->nullable();
        $table->text('motivo_consulta')->nullable();
        $table->string('odontologo')->nullable();
        $table->text('diagnostico')->nullable();
        $table->text('observaciones')->nullable();
        $table->string('pieza_dental')->nullable();
        $table->string('tipo_tratamiento')->nullable();
        $table->string('estado');

        $table->timestamps();

        // Claves foráneas
        $table->foreign('id_paciente')->references('id_paciente')->on('pacientes')->onDelete('cascade');

    });
}
```

Ilustración 51- Migración de Historia Clínica

La migración define la estructura de la tabla historial_clinicos, la cual registra toda la información relacionada con las consultas y tratamientos de los pacientes en el consultorio odontológico.

Los principales campos incluidos son:

- **id_historial_clinico:** clave primaria autoincremental que identifica cada historial clínico.

- **id_paciente:** clave foránea que enlaza el historial clínico con un paciente de la tabla pacientes. Esto garantiza la relación directa entre los datos médicos y el paciente correspondiente.
- **fecha:** fecha en la que se realizó la consulta o tratamiento.
- **motivo_consulta:** descripción del motivo por el cual el paciente asistió a la consulta.
- **odontologo:** nombre del odontólogo que atendió al paciente en esa cita.
- **diagnostico:** detalle del diagnóstico realizado al paciente.
- observaciones: comentarios adicionales registrados por el odontólogo, como notas de seguimiento.
- **pieza_dental:** campo que identifica la pieza dental tratada en caso de procedimientos específicos.
- **tipo_tratamiento:** especifica el tratamiento aplicado, como limpieza, extracción, restauración, entre otros.
- **estado:** permite indicar si el historial clínico está activo, finalizado o en seguimiento.
- **timestamps:** campos automáticos created_at y updated_at que registran la creación y última actualización del historial.

Además, se establece una clave foránea entre id_paciente de esta tabla y id_paciente de la tabla pacientes, con la opción onDelete('cascade'), lo que significa que, si un paciente es eliminado, automáticamente se eliminarán todos sus historiales clínicos asociados.

4.3.4. Citas

```php
public function up(): void
{
    Schema::create('citas', function (Blueprint $table) {
        $table->id('id_cita');

        $table->unsignedBigInteger('id_paciente');
        $table->unsignedBigInteger('id_usuario');
        $table->string('odontologo');
        $table->date('fecha');
        $table->time('hora');
        $table->string('estado');
        $table->timestamps();

        // Claves foráneas
        $table->foreign('id_paciente')->references('id_paciente')->on('pacientes')->onDelete('cascade');
        $table->foreign('id_usuario')->references('id_usuario')->on('usuarios')->onDelete('cascade');
    });
}
```

Ilustración 52- Migración de Citas

La migración create_citas_table define la estructura de la tabla citas, que gestiona el agendamiento de las consultas odontológicas de los pacientes.

Los campos principales son:

- id_cita: clave primaria autoincremental que identifica de manera única cada cita.
- id_paciente: clave foránea que enlaza la cita con el paciente correspondiente registrado en la tabla pacientes.
- id_usuario: clave foránea que vincula la cita con el usuario del sistema (ejemplo: secretaria u odontólogo) que la gestionó o registró, referenciado en la tabla usuarios.
- odontologo: almacena el nombre del odontólogo responsable de atender la cita.
- fecha: especifica el día en el que se llevará a cabo la consulta.
- hora: registra la hora exacta en la que está programada la cita.
- estado: define la condición de la cita, como pendiente, confirmada, cancelada o realizada.
- timestamps: incluyen automáticamente los campos created_at y updated_at, que permiten llevar control del momento en que fue creada o actualizada la cita.

Además, se establecen relaciones foráneas:

- id_paciente → referencia a la tabla pacientes, con eliminación en cascada: si se borra un paciente, se eliminan también sus citas.
- id_usuario → referencia a la tabla usuarios, también con eliminación en cascada: si un usuario es eliminado, se eliminan las citas asociadas a su gestión.

4.3.5. Reportes

```php
public function up(): void
{
    Schema::create('reportes', function (Blueprint $table) {
        $table->id('id_reporte');
        $table->string('tipo_reporte');
        $table->unsignedBigInteger('id_usuario')->nullable();
        $table->foreign('id_usuario')->references('id_usuario')->on('usuarios')->onDelete('set null');
        // Fecha de generación automática
        $table->timestamp('generado_el')->useCurrent();

        $table->timestamps();
    });
}
```

Ilustración 53- Migración Reportes

La migración create_reportes_table establece la estructura de la tabla reportes, la cual almacena la información relacionada con los reportes generados dentro del sistema.

Los campos principales son:

- **id_reporte:** clave primaria autoincremental que identifica de manera única cada reporte.
- **tipo_reporte:** campo de tipo cadena que define la categoría o clase de reporte generado (ejemplo: reporte de pacientes, citas, historial clínico, etc.).
- **id_usuario:** clave foránea opcional (nullable) que vincula el reporte con el usuario que lo generó. En caso de que el usuario sea eliminado, el campo se establece en NULL gracias a la opción onDelete('set null'), lo que permite conservar el historial de reportes.
- **generado_el**: almacena la fecha y hora exacta en que se creó el reporte, configurado con useCurrent() para que se registre automáticamente en el momento de la generación.
- **timestamps:** incluye los campos created_at y updated_at, que permiten llevar control de la creación y modificación de cada registro.

En cuanto a las relaciones, esta tabla mantiene una conexión directa con usuarios, de manera que cada reporte queda asociado al usuario que lo elaboró, lo cual garantiza trazabilidad en la gestión de la información.

5. Conclusiones

El desarrollo del sistema web para la gestión de pacientes, citas, historiales clínicos y reportes en el consultorio odontológico Perfect Smile ha permitido contar con una herramienta eficiente y organizada que facilita el trabajo administrativo y clínico. Gracias a su estructura modular, el sistema brinda un acceso rápido a la información, mejora el control de los procesos y optimiza la toma de decisiones dentro del consultorio.

6. Recomendaciones

- Capacitar a los usuarios antes de la implementación para asegurar un uso adecuado de todas las funciones del sistema.
- Realizar respaldos periódicos de la base de datos para evitar pérdida de información.
- Mantener el sistema actualizado con mejoras en seguridad y nuevas funcionalidades según las necesidades del consultorio.
- Considerar futuras ampliaciones como la integración con sistemas de facturación electrónica o recordatorios automáticos a los pacientes.

Manual De Usuario

Sistema de Gestión Integral de Pacientes, Citas e Historiales Clínicos

Consultorio Odontológico Perfect Smile

Versión: 1.0

Autor: Michael Eduardo Guamán Tapia

Índice

Índice de Imágenes

1. Introducción

Este manual de usuario está hecho para ayudar a las personas que usan el sistema Perfect Smile, un programa que sirve para organizar todo lo relacionado con los pacientes, las citas y los historiales clínicos en un consultorio dental. Está pensado especialmente para administradores, odontólogos y el personal de secretaría, y su objetivo es explicar cómo funciona cada parte del sistema de manera clara. En el manual se pueden seguir instrucciones paso a paso, ver imágenes de las pantallas principales y encontrar consejos prácticos para usar el sistema de forma correcta y sin problemas.

2. Objetivos del Programa

- Podrás registrar, actualizar y ver todos los datos personales y médicos de cada paciente de manera ordenada y accesible.
- La herramienta permite agendar, cambiar o cancelar citas rápidamente, y consultar fácilmente la agenda diaria de cada odontólogo.
- Vas a poder registrar y revisar toda la información relevante de los tratamientos, diagnósticos y observaciones de cada paciente, todo en un mismo lugar.
- El sistema ayuda a crear informes en PDF que resumen las citas programadas y los pacientes atendidos por cada odontólogo, para que tomar decisiones sea más informado y ágil.
- Cada usuario tendrá un perfil con permisos específicos, lo que garantiza que solo las personas autorizadas puedan acceder a información sensible, protegiendo así la privacidad de todos.

3. Requerimientos del Programa

Hardware recomendado:

- Computadora o laptop con mínimo 4 GB de RAM y procesador de 1 GHz o superior.
- Resolución de pantalla mínima: 1024x768 píxeles.
- Espacio en disco: al menos 500 MB para instalación y almacenamiento de datos.

Software requerido:

- Sistema operativo: Windows 10 o superior, Linux o macOS compatible.
- Navegador web moderno (Google Chrome, Microsoft Edge o Mozilla Firefox).
- Servidor local o remoto con soporte para PHP (versión 8 o superior) y base de datos PostgreSQL.
- Conexión a internet para acceso al servidor o actualizaciones del sistema.

4. Funcionamiento del Programa

A continuación, se describe cómo funciona cada módulo y la interacción general del usuario con el programa.

Cuando el usuario inicia sesión en el sistema, es recibido por el dashboard o cuadro de mando, que actúa como el centro de control de todas las operaciones. Esta pantalla muestra información resumida y relevante, como la cantidad de pacientes activos, citas programadas para el día y accesos directos a los módulos principales. La interfaz está diseñada para que los botones y menús sean fáciles de identificar y acceder, evitando confusiones, especialmente para el personal que no tiene mucha experiencia con sistemas informáticos.

4.1. Inicio de Sesión

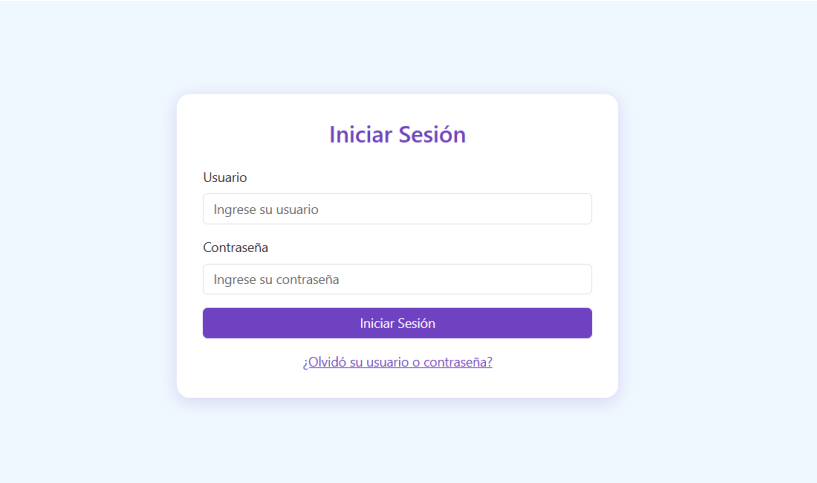


Ilustración 1- Inicio de Sesión

Para ingresar al sistema, cada usuario debe contar con un usuario y contraseña válidos. Cuando un usuario ingresa sus datos de acceso, el sistema verifica que sean correctos. Si todo está bien, lo dirige a un panel principal diseñado especialmente según su tipo de usuario (odontólogo, administrador, etc.). En caso de que haya un error en los datos, se muestra un mensaje claro que sugiere revisar la información ingresada.

Además, cada vez que alguien inicia sesión, el sistema registra quién es el usuario responsable de cada acción. Esto permite mantener un control sobre los cambios realizados y saber en todo momento quién hizo qué dentro del sistema, lo que ayuda a garantizar la transparencia y seguridad de los procesos.

4.2. Recuperación De Contraseña

Ilustración 2- Recuperación De Contraseña P1

Ilustración 3- Recuperación De Contraseña P2

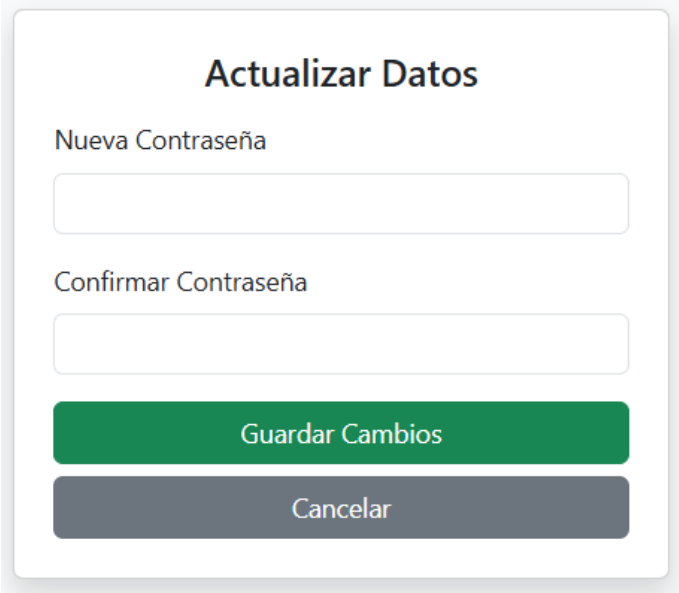


Ilustración 4- Recuperación De Contraseña P3

Si algún usuario olvida su contraseña, el sistema ofrece una forma sencilla y segura de recuperar el acceso, solo necesita ingresar el correo electrónico con el que se registró originalmente y el sistema comprobará en la base de datos si existe una cuenta asociada a ese correo.

4.3. Dashboard / Cuadro de Mando

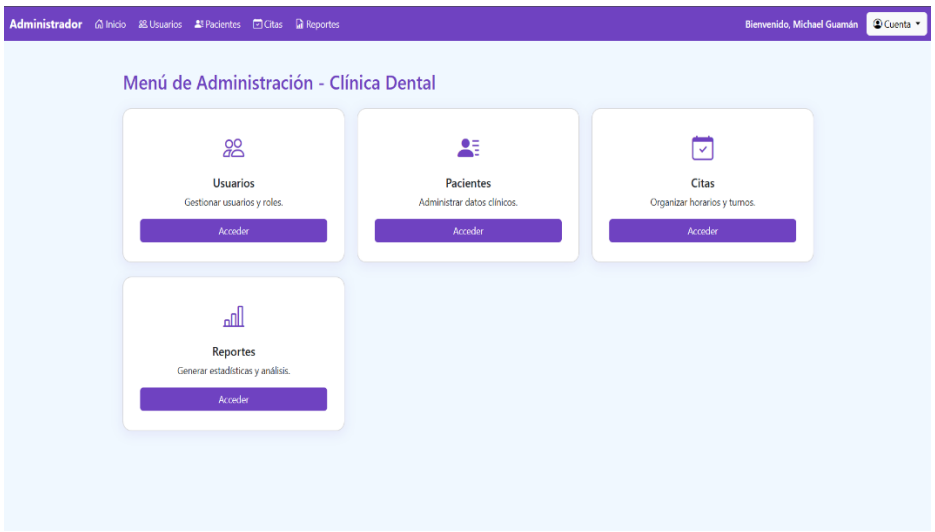


Ilustración 5- Dashboard

El dashboard es la pantalla principal del sistema, desde donde el usuario puede ver y acceder fácilmente a todas las secciones importantes, como Usuarios, Pacientes, Historia Clínica, Citas y Reportes. No muestra los detalles de cada módulo, sino que funciona como un menú central que permite navegar rápido a cualquier parte del sistema, así los usuarios encuentran lo que necesitan de manera fácil y pueden trabajar de forma más ordenada y eficiente.

4.4. Modulo Usuario

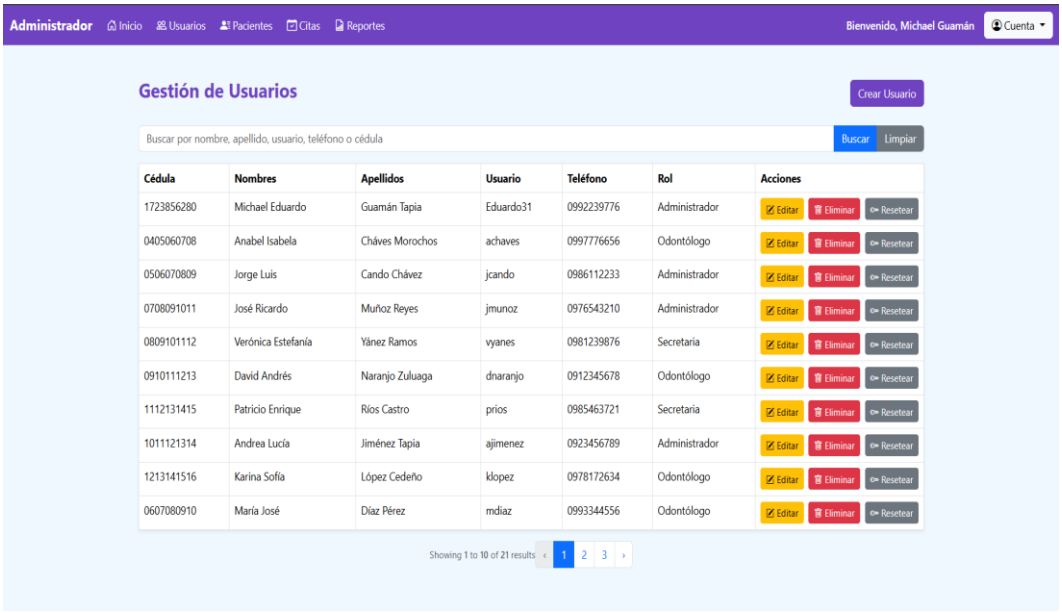


Ilustración 6-Modulo Usuario

Aquí, un usuario con permisos de administrador puede ver, organizar y actualizar la información de todos los perfiles registrados.

La pantalla principal muestra una lista clara y ordenada con todos los usuarios. En esta tabla aparecen datos importantes como su cédula, nombres, apellidos, nombre de usuario, número de teléfono y el rol que tienen dentro del sistema (como administrador, odontólogo, etc.).

Para hacer más fácil encontrar a alguna persona en específico, la tabla permite buscar y filtrar información. Así, en lugar de revisar manualmente toda la lista, se puede ubicar rápidamente a un usuario usando algún dato que ya se conozca de él, como su nombre o cédula.

Funciones del Módulo de Usuario

Crear Usuario

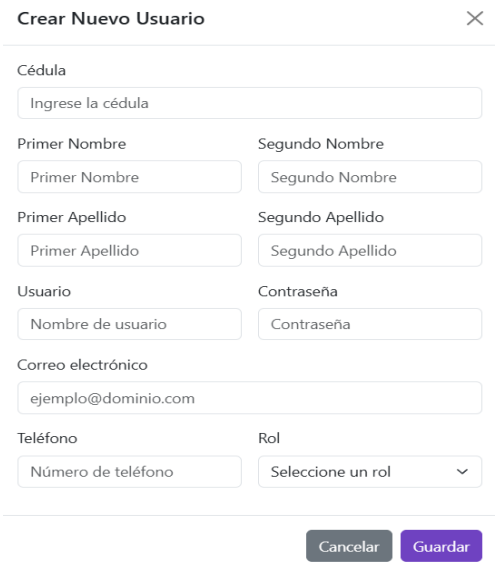


Ilustración 7- Crear Modulo de Usuario

Esta función permite a los usuarios autorizados agregar nuevas cuentas al sistema de manera segura y organizada. Al crear un usuario, se completan datos esenciales como nombres, apellidos, cédula, número de teléfono, correo electrónico, el tipo de perfil (rol) y una contraseña inicial. Esto asegura que todas las personas registradas tengan permisos adecuados y puedan usar el sistema según sus necesidades.

Editar Usuario

Ilustración 8- Editar Modulo de Usuario

Esta función permite modificar la información de un usuario que ya está registrado en el sistema. Se pueden actualizar datos como nombres, apellidos, número de teléfono, correo electrónico y el tipo de perfil asignado.

Para hacerlo más fácil, el sistema muestra un formulario que ya viene con la información actual del usuario. Así, no es necesario volver a llenar todos los campos desde cero, solo hacer los cambios necesarios.

Eliminar Usuario

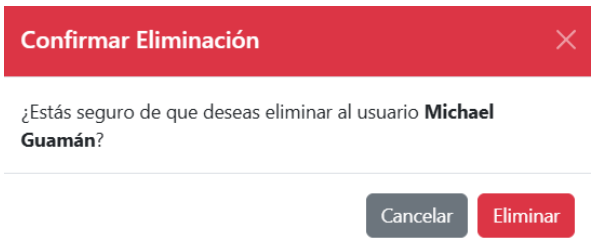


Ilustración 9- Eliminar Modulo de Usuario

Esta función permite eliminar del sistema a usuarios que ya no necesitan acceso. Para evitar que se borren cuentas por error, el sistema siempre pide confirmación antes de proceder con la eliminación. De esta manera, se ayuda a mantener la seguridad del sistema, ya que se pueden eliminar cuentas que ya no son necesarias y así reducir el riesgo de accesos no autorizados o inadecuados.

Resetear Contraseña

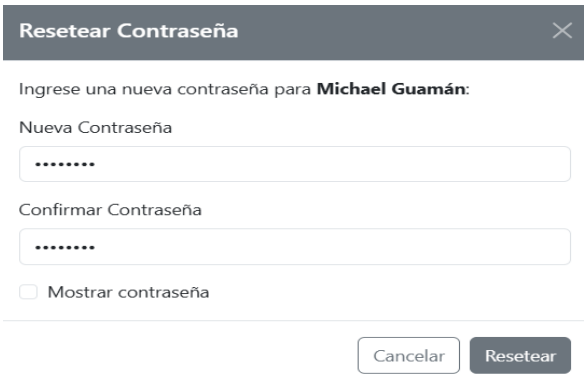


Ilustración 10- Resetear Contraseña Modulo de Usuario

Esta opción permite restablecer la contraseña cuando un usuario ha olvidado sus credenciales de acceso. El proceso se realiza de forma segura a través de un formulario donde se puede establecer una nueva contraseña para la cuenta seleccionada. De esta manera, los usuarios pueden recuperar el acceso a sus cuentas de manera ágil, sin poner en riesgo la seguridad de la información personal y clínica almacenada en el sistema.

4.5. Modulo Paciente

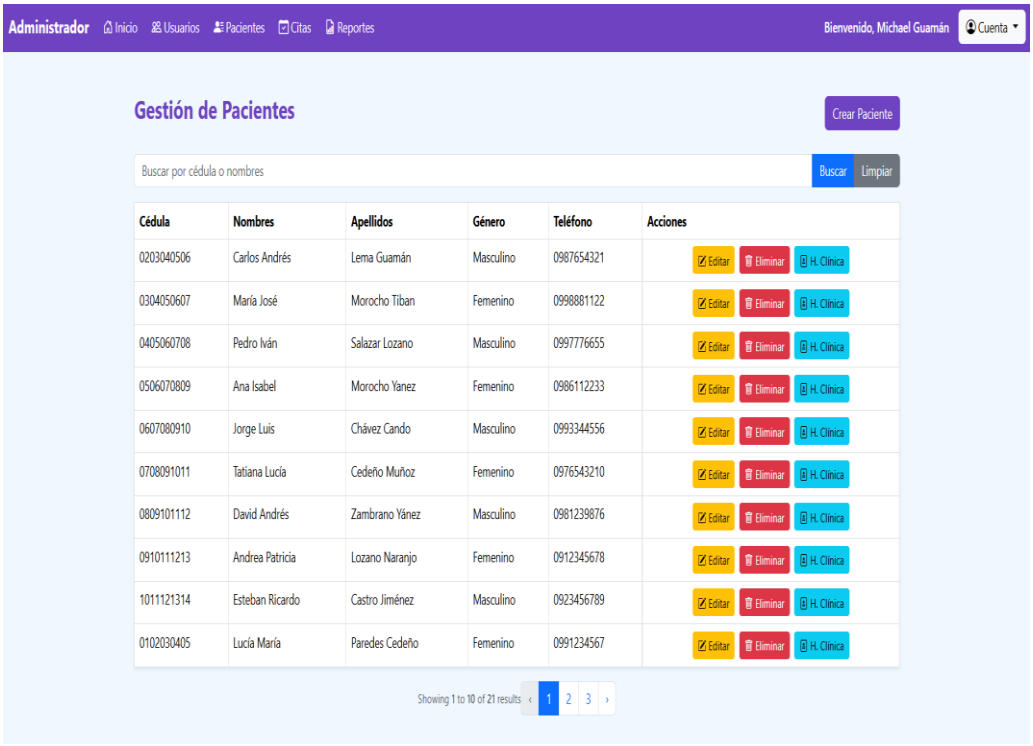


Ilustración 11- Modulo Paciente

El módulo de Pacientes está diseñado para organizar toda la información de las personas que son atendidas en el consultorio, en esta sección se puede llevar un registro completo y ordenado de cada paciente.

En la pantalla principal aparece una tabla con todos los pacientes registrados, mostrando información básica como cédula, nombres, apellidos, género, número de teléfono y estado. Además, la tabla permite buscar y filtrar según lo que se necesite, lo que hace que encontrar cualquier historial o información específica sea mucho más rápido y sencillo.

Funciones del Módulo de Paciente

Crear Paciente

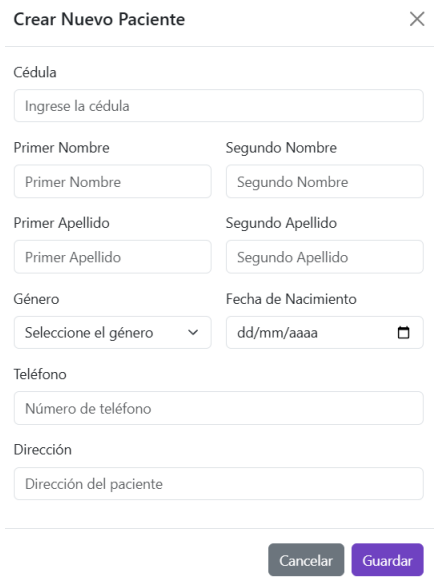


Ilustración 12- Crear Modulo Paciente

Esta opción sirve para agregar nuevos pacientes al sistema. Solo hay que llenar un formulario con datos básicos como nombres, apellidos, cédula, género, fecha de nacimiento, teléfono y dirección, así todos los pacientes quedan registrados y listos para usar en citas e historiales.

Editar Paciente

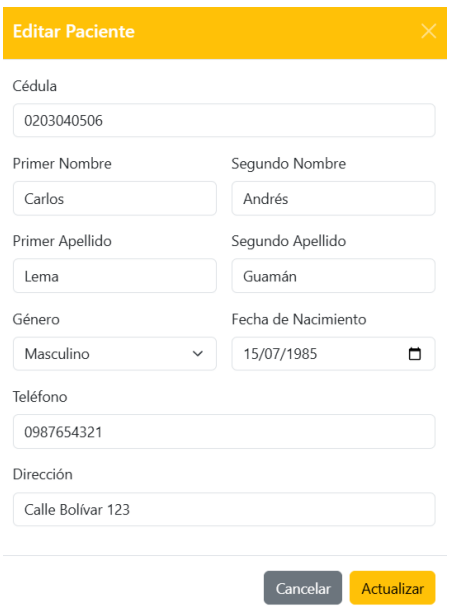


Ilustración 13- Editar Modulo Paciente

La función de edición sirve para actualizar los datos de un paciente. Aparece un formulario con la información actual y se pueden cambiar nombres, apellidos, teléfono, género, dirección u otros datos, así la información siempre está correcta y al día.

Eliminar Paciente

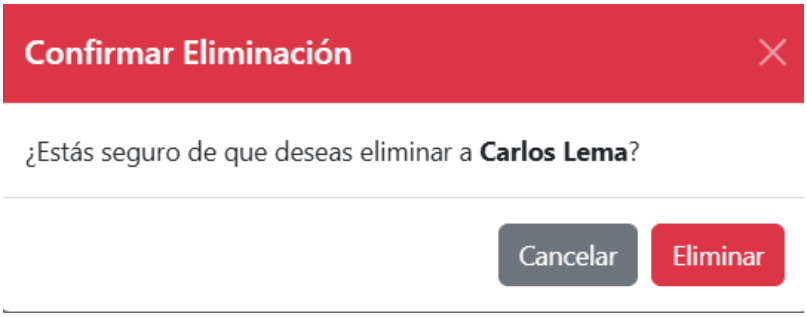


Ilustración 14- Eliminar Modulo Paciente

Esta función permite borrar pacientes que ya no se necesitan en el sistema. Antes de eliminar, pide confirmación para no borrar nada por error, ayudando a mantener la base de datos ordenada y segura.

Historial Clínico

Ilustración 15- Historia Clínica Modulo Paciente

Este botón abre el Historial Clínico del paciente seleccionado. Allí se puede ver todo sobre sus consultas, tratamientos, diagnósticos y anotaciones, lo que ayuda a llevar un control de su evolución y planear próximas citas y tratamientos de forma más fácil.

4.6. Modulo Historia Clínica

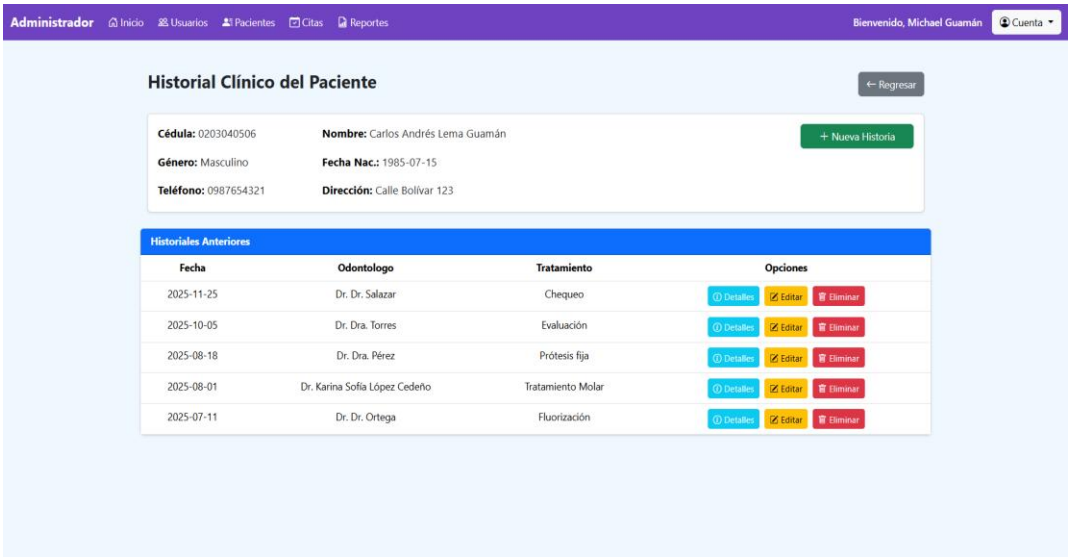


Ilustración 16- Modulo Historias Clínicas

El módulo de Historia Clínica sirve para registrar y manejar toda la información médica y dental de cada paciente. Permite a los profesionales llevar un control de los antecedentes, tratamientos y evolución de los pacientes, asegurando que la atención sea correcta. La pantalla principal muestra una tabla con todos los registros, incluyendo fecha de consulta, odontólogo, motivo de la consulta, diagnóstico y tipo de tratamiento.

Funciones del Módulo de Historia Clínica

Crear Historia Clínica

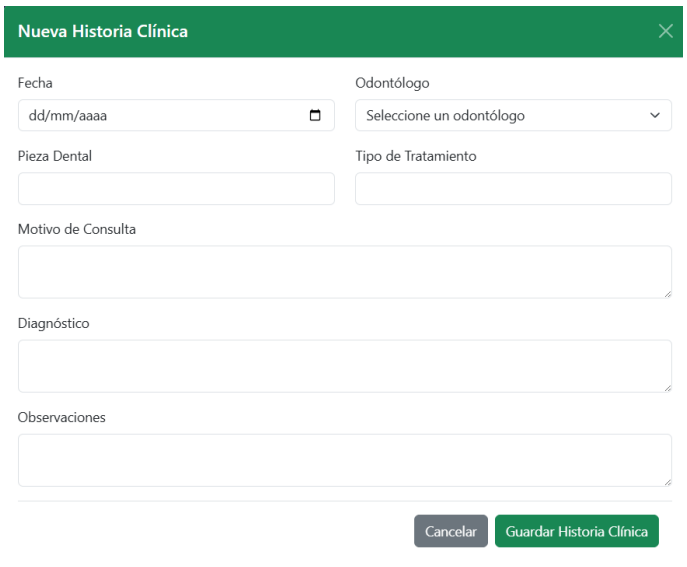


Ilustración 17- Crear Modulo Historias Clínicas

Esta función sirve para agregar un nuevo registro al historial clínico de un paciente. El formulario pide datos como fecha de consulta, motivo, odontólogo, diagnóstico, observaciones, piezas dentales afectadas y tipo de tratamiento, así cada consulta queda registrada y se puede hacer un seguimiento completo de la salud del paciente.

Editar Historia Clínica

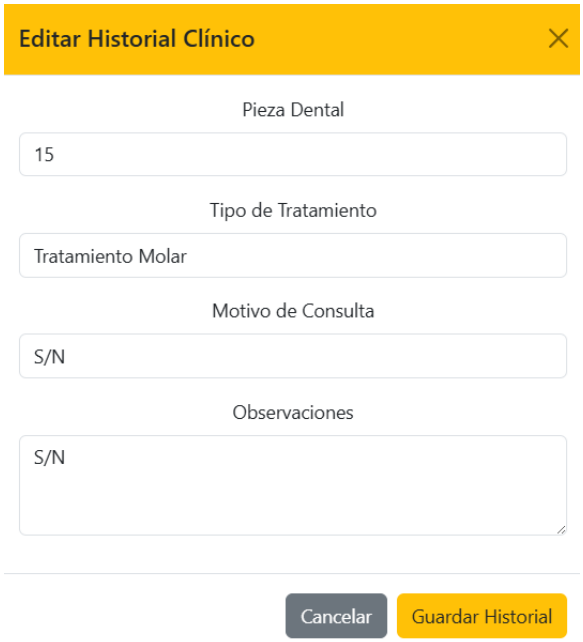


Ilustración 18- Editar Modulo Historias Clínicas

La opción de edición sirve para actualizar un registro del historial clínico. Esto ayuda a corregir datos o agregar información que faltaba, asegurando que la información sea correcta y confiable para el personal odontológico.

Eliminar Historia Clínica

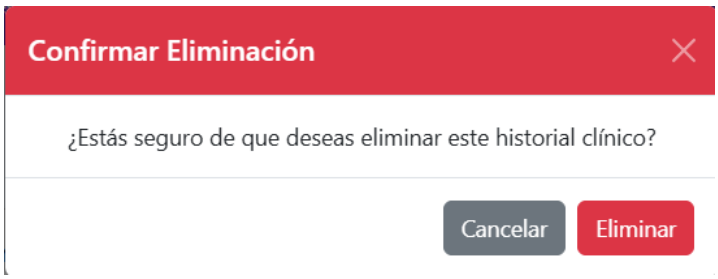


Ilustración 19- Eliminar Modulo Historias Clínicas

Esta función sirve para borrar registros que estén incorrectos o que ya no se necesiten. Antes de eliminar, el sistema pide confirmación para no perder información por error, ayudando a mantener la base de datos limpia y ordenada.

Detalles de Historia Clínica

Ilustración 20- Detalle Modulo Historias Clínicas

El botón de detalles abre una ventana con toda la información del registro seleccionado, mostrando fecha, odontólogo, motivo, diagnóstico, observaciones, piezas dentales y tipo de tratamiento. Esto permite revisar rápidamente la historia clínica sin salir de la tabla y facilita planear tratamientos y hacer seguimiento del paciente.

4.7. Modulo Citas

El módulo de Citas sirve para organizar todas las atenciones del consultorio. Tiene dos vistas principales: un calendario interactivo y una tabla de citas, lo que ayuda a ver y manejar los turnos de manera clara y fácil.

Pantalla Calendario

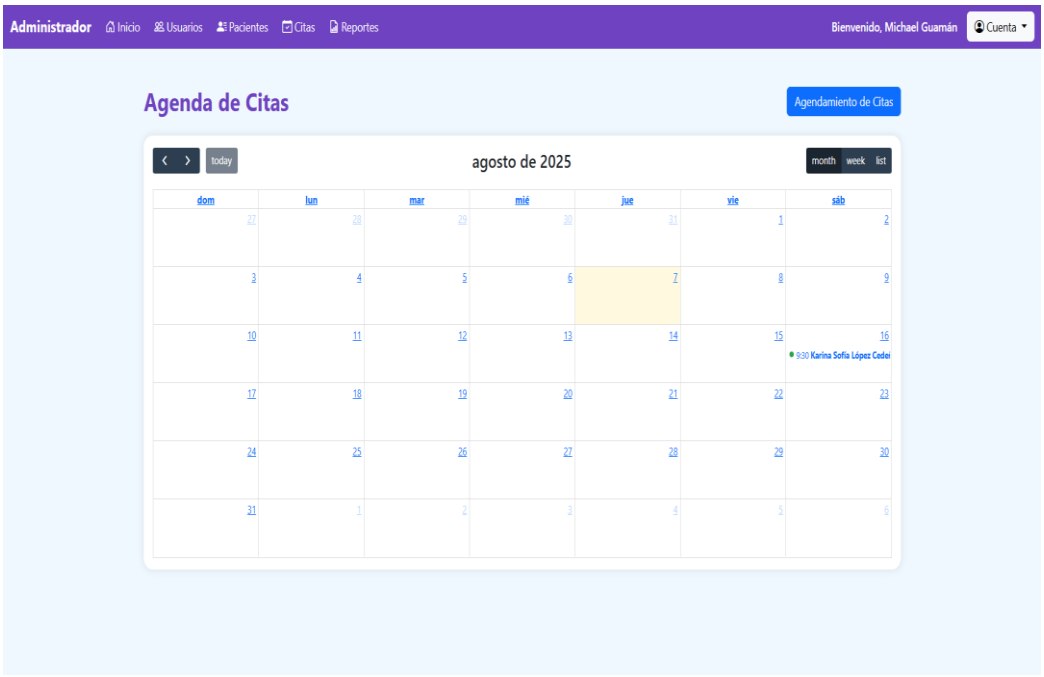


Ilustración 21- Calendario Modulo Citas

En esta pantalla se muestra un calendario, donde aparecen todas las citas registradas en el sistema.

- El usuario puede cambiar la vista del calendario según sus necesidades mensual, semanal o en formato de lista.
- Cada cita aparece identificada con el nombre del paciente.
- Desde esta vista se facilita el control general de la agenda del consultorio.
- En esta misma pantalla existe un botón llamado “Agendamiento de Citas”.

Pantalla Gestión de Citas

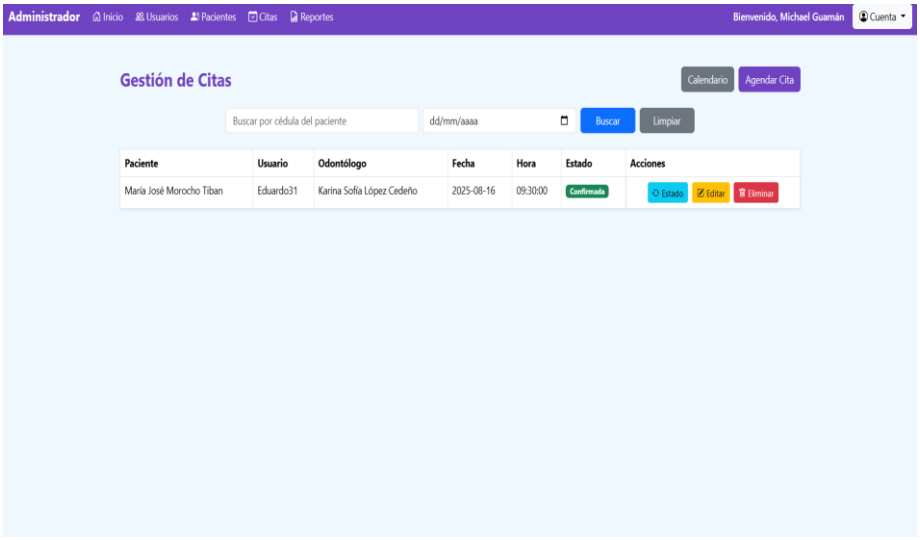


Ilustración 22- Vista Tabla Modulo Citas

En esta sección aparecen todas las citas registradas en el sistema, lo que permite llevar un control más detallado. La tabla muestra información como paciente, odontólogo, fecha, hora y estado de la cita, y desde aquí se pueden realizar varias acciones:

Agendar Citas

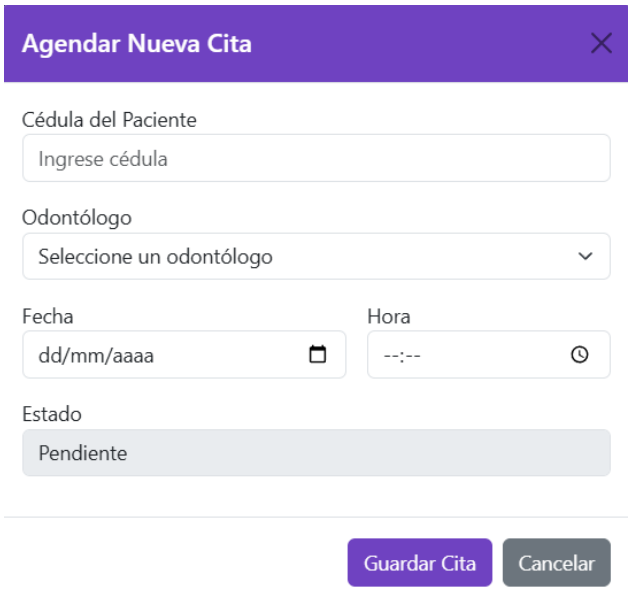


Ilustración 23- Agendar Modulo Citas

Esta opción sirve para asignar un nuevo turno a un paciente, solo hay que llenar fecha, hora y odontólogo, para que la cita quede registrada correctamente.

Editar Citas

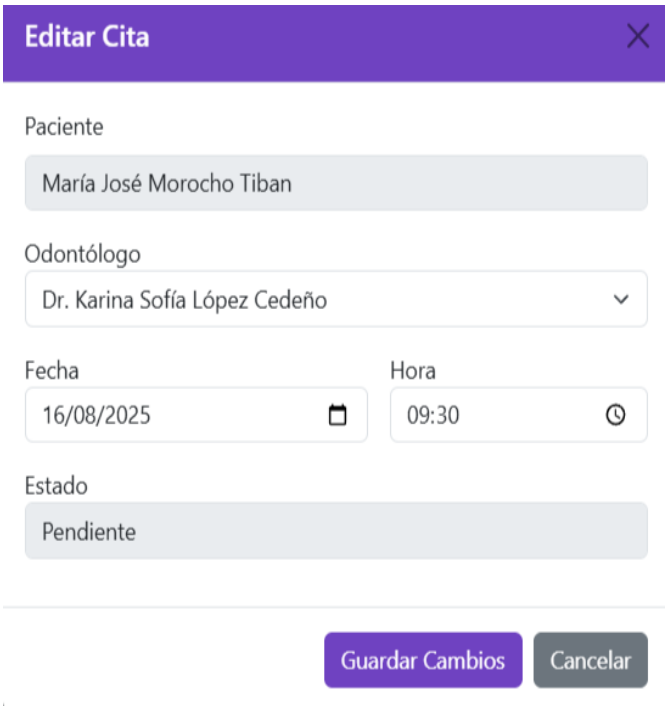


Ilustración 24- Editar Modulo Citas

Esta opción sirve para cambiar los datos de una cita, como la fecha, hora, odontólogo o motivo. Es útil para hacer ajustes o reprogramar turnos.

Eliminar Citas

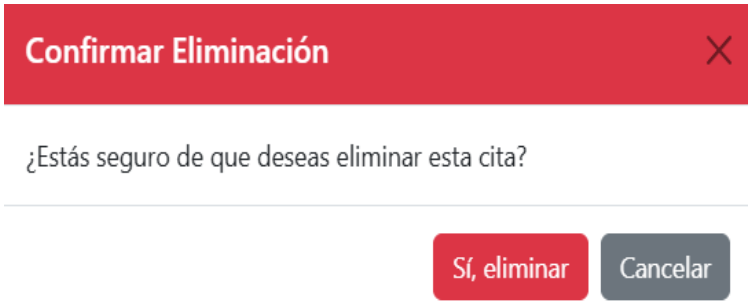


Ilustración 25- Eliminar Modulo Citas

Esta opción sirve para borrar citas que no se van a realizar, antes de eliminar, el sistema pide confirmación para no perder información por error.

Estado de Citas

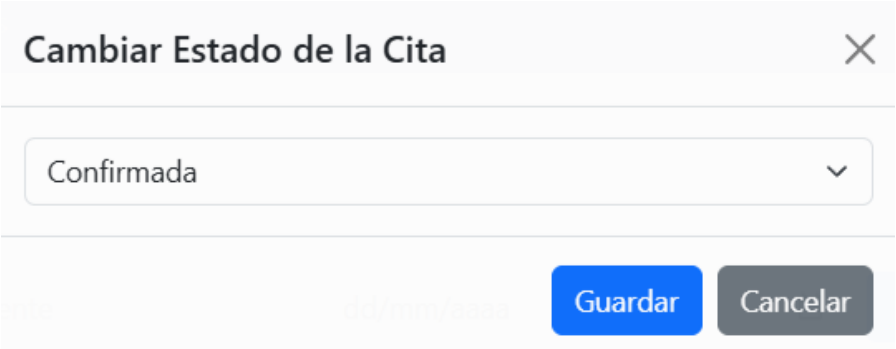


Ilustración 26- Cambiar Estado Modulo Citas

Cada cita puede modificar su estado según la situación actual:

- **Pendiente:** la cita está registrada pero aún no atendida.
- **Confirmada:** el paciente o el consultorio han validado la asistencia
- **Cancelada:** la cita ha sido anulada por el paciente o el consultorio.

Esta funcionalidad facilita llevar un control más preciso de la gestión de turnos.

4.8. Modulo Reportes

Ilustración 27- Modulo Reportes

El módulo de Reportes es una herramienta importante del sistema porque permite crear documentos en PDF con la información más útil del consultorio, además ayuda a organizar y consultar los datos de forma rápida y sencilla, desde el menú principal de Reportes, el usuario puede acceder a tres tipos diferentes de reportes

Historial Clínico por Pacientes

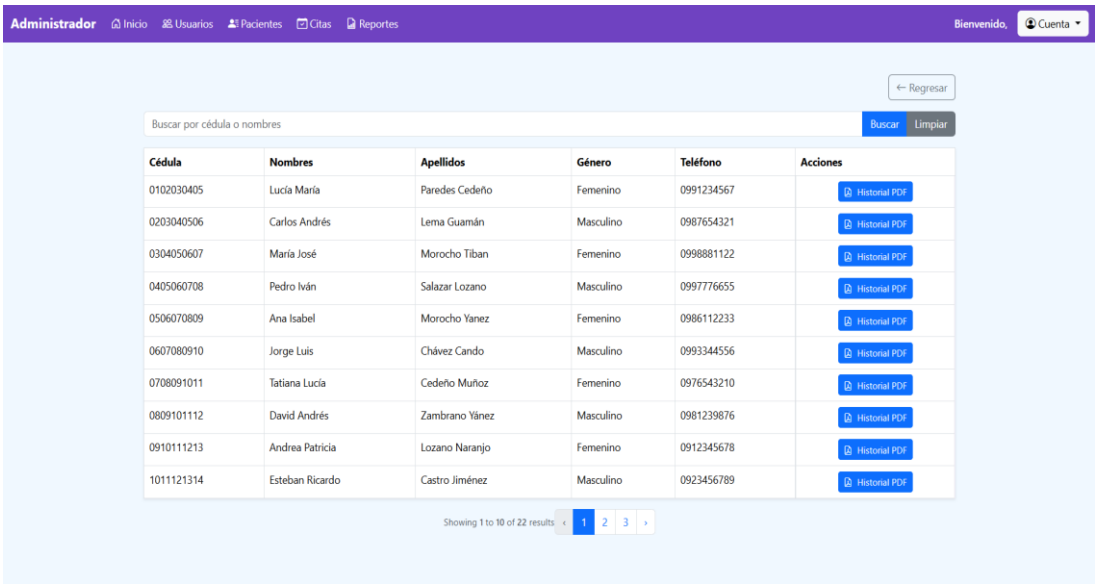


Ilustración 28- Historial Por Paciente Modulo Reportes

En esta pantalla se muestra una vista con la lista de todos los pacientes:

- La vista incluye un buscador, que facilita encontrar al paciente deseado ingresando su nombre o algún dato relacionado.
- Cada paciente cuenta con un botón “Historial PDF”, que genera y descarga un reporte en formato PDF con todo su historial clínico.
- Este historial muestra toda la información de cada consulta, como el motivo de atención, diagnóstico, tratamientos realizados, observaciones y otros datos que ya están registrados en el sistema
- De esta forma, se obtiene una ficha completa del paciente, lista para ser archivada o compartida según sea necesario.

Pacientes Atendidos por Odontólogo

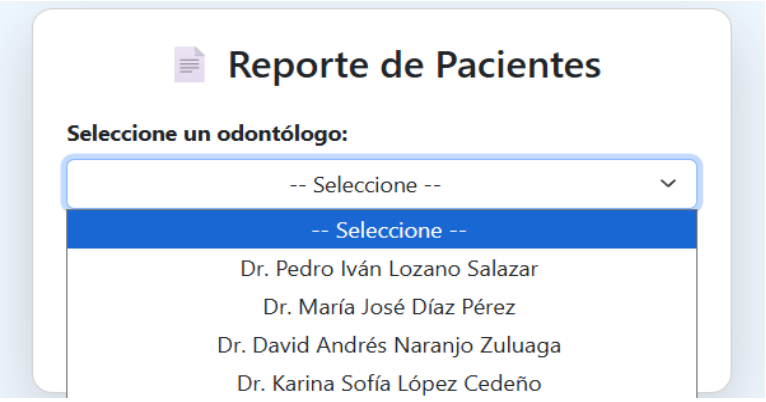


Ilustración 29- Pacientes Atendidos Por Odontólogo Modulo Reportes

Esta opción sirve para generar un reporte según el profesional que atendió a los pacientes, al entrar en esta vista, el sistema muestra un menú donde el usuario puede seleccionar al odontólogo deseado, luego se genera un reporte que muestra cuántos pacientes ha atendido ese odontólogo en un período determinado, el resultado se puede ver en pantalla y también descargar en PDF, lo que facilita manejar la información y controlar mejor el consultorio

Reporte de Citas del Día

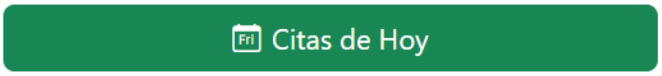


Ilustración 30- Citas de Hoy Modulo Reportes

La última opción del menú general de reportes corresponde al reporte de las citas programadas para el día actual.

- Con un solo clic, el sistema genera un archivo PDF con todas las citas agendadas para la fecha en curso.
- Este reporte incluye información como: paciente, odontólogo asignado, hora de la cita y el estado actual de la misma (pendiente, confirmada o cancelada).
- Es una herramienta muy útil para que el consultorio tenga una agenda diaria impresa o digital, optimizando el control de las atenciones del día.

5. Conclusiones

El presente manual de usuario ha sido elaborado con el propósito de guiar a los usuarios en la correcta utilización del sistema de gestión del consultorio odontológico Perfect Smile. A lo largo del documento, se detallan los diferentes módulos y sus funciones principales, con el fin de que el personal administrativo, odontólogos y secretarias puedan realizar sus tareas de manera eficiente.

El sistema permite gestionar de forma centralizada la información de los pacientes, citas, historiales clínicos y reportes, garantizando orden, control y rapidez en los procesos internos. Gracias a la interfaz amigable y a las funciones implementadas, se optimiza la atención al paciente y se facilita el trabajo del equipo profesional.

6. Recomendaciones

- Leer detenidamente este manual antes de comenzar a usar el sistema para familiarizarse con las funciones disponibles.
- Mantener actualizada la información registrada en los diferentes módulos, ya que esto asegura la confiabilidad de los reportes y del historial clínico.
- Revisar siempre el estado de las citas (pendiente, confirmada, cancelada) para llevar un control más organizado de la agenda diaria.
- Hacer uso frecuente de los reportes en PDF, ya que son útiles tanto para respaldo administrativo como para fines de auditoría.
- En caso de dudas o inconvenientes técnicos, contactar con el área de soporte o con el administrador del sistema.