Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

ESTUDIO Y DESARROLLO DE UN PROTOTIPO DE APLICACIÓN WEB GESTIÓN DE UNA CLÍNICA DENTAL

Alumno: Antonio López García

Tutor: Prof. D. José Ramón Balsas Almagro
Dpto: Departamento de Informática

Septiembre, 2023

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don José Ramón Balsas Almagro , tutor del Proyecto Fin de Carrera titulado: Estudio y desarrollo de un prototipo de aplicación web de gestión de una clínica dental, que presenta Antonio López García, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2023

El alumno: Los tutores:

Antonio López García José Ramón Balsas Almagro

ÍNDICE

Índice de Tablas

Índice de Ilustraciones

1. INTRODUCCIÓN

La odontología es una práctica médica que ha existido desde la antigüedad. Los registros históricos muestran que los antiguos egipcios, griegos, romanos y otros pueblos tenían conocimientos sobre la salud dental y el tratamiento de enfermedades dentales.

Por ejemplo, en Egipto, se han encontrado registros de tratamientos dentales realizados hace más de 5.000 años, incluyendo la extracción de dientes y la colocación de prótesis dentales. En la antigua Grecia, Hipócrates, el padre de la medicina, escribió sobre la importancia de la salud dental y el tratamiento de las enfermedades dentales. (Couto, 2018)

En la Edad Media, los barberos eran a menudo los encargados de realizar procedimientos dentales, incluyendo extracciones de dientes y la colocación de prótesis dentales. Sin embargo, con el tiempo la odontología se fue separando de la cirugía y la medicina general para convertirse en una disciplina médica y dental independiente.
(Couto, 2018)

La odontología es una ciencia altamente especializada y se utiliza para tratar y prevenir una amplia variedad de dolencias dentales y enfermedades, desde caries hasta enfermedad periodontal y cáncer oral. Además, la odontología también incluye tratamientos estéticos, como la ortodoncia y la odontología cosmética, que buscan mejorar la apariencia de los dientes y la sonrisa.

Hoy en día, en pleno siglo XXI donde la tecnología es dominante en todas las áreas de la vida cotidiana, el número de personas que tienen acceso a internet, de acuerdo con el último reporte de (DigitalReport, 2023), "...alcanzó los 5.160 millones de personas, lo que representa al 64,4% de la población mundial."

El mismo informe indica que la cantidad de usuarios que hacen uso de telefonía móvil es de 5,440 millones de personas y que estas en su gran mayoría usan el móvil para hacer búsquedas, transacciones financieras, y otras tantas operaciones diarias sin necesidad de moverse de un lado a otro.

Visto esto, no resulta descabellado que el área de salud, en este caso la odontología, sea una de las áreas que se involucre con el uso de la tecnología y las aplicaciones informáticas para llegar hasta sus clientes, facilitando así la comunicación del paciente con el odontólogo, y en especial la mejoría en la atención al cliente.

Y es que, pensando en esto último, cabe preguntarse: ¿cómo se puede mejorar el servicio de atención odontología al cliente/paciente haciendo uso de las herramientas tecnológicas?

Pensando en un sitio web, el cual puede ser utilizado para promocionar la clínica y atraer a nuevos pacientes, ofrece a su vez otras ventajas como:

- Mayor accesibilidad: Los pacientes pueden acceder a la información de la clínica en cualquier momento y desde cualquier lugar.
- Mayor comodidad: Los pacientes pueden programar citas en línea o ponerse en contacto con la clínica a través del sitio web, lo que les resulta más cómodo y rápido.
- Mayor credibilidad: Un sitio web profesional y bien diseñado puede dar a la clínica una mayor credibilidad y confianza ante los pacientes.
- Diferenciación de la competencia: Un sitio web profesional y completo puede diferenciar la clínica de la competencia, y mostrar a los pacientes potenciales que la clínica se preocupa por ofrecer una experiencia de atención al paciente de calidad.

Es importante destacar que para que el sitio web sea efectivo, debe ser fácil de navegar, tener una buena presentación visual y ser actualizado regularmente con nueva información y contenido. También es fundamental que el sitio web sea compatible con dispositivos móviles, ya que cada vez más personas utilizan sus teléfonos para buscar información sobre servicios médicos y odontológicos.

1.1. Justificación

Si bien es cierto que la gran mayoría de las clínicas odontológicas presentan una página web para promocionar la clínica y dar a conocer su personal y los servicios que prestan, también es cierto que son muy pocas, o casi ninguna, las que consiguen gestionar sus citas a través de dicha herramienta; consiguiendo como el caso más generalizado el de presentar a través de la página web los números de contacto telefónico o e-mail para poder ser contactado y agendar una cita.

La sola idea de poder ofrecer la comodidad al cliente de poder realizar citas en línea, sin necesidad de instalar una aplicación especial para ello, a través de una interfaz amigable, es lo que motiva la realización de este proyecto.

Esta oportunidad de promoción y difusión nos permite llegar a todo el mundo sin necesidad de realizar grandes inversiones en publicidad, ya que con una mínima inversión una clínica dental puede contar con una web, red social o publicidad en ellas. Hecho que nos ayudará a competir de forma más igualada frente los servicios ofrecidos por otras clínicas, al poder crear comunidades entre usuarios para conseguir confianza, compromiso y fidelidad entre los pacientes de nuestra clínica.

Ahora bien, sabiendo que previamente se ha llevado a cabo un proyecto afín en la Escuela Politécnica Superior de Jaén como parte de un trabajo de fin de curso denominado *"Estudio y*

Desarrollo de un Prototipo de Aplicación Web para la Gestión de Citas en una Clínica Dental", (Martinez, 2018), que si bien comparten objetivos similares con el proyecto que se encuentra en desarrollo consideramos que este prototipo presenta un enfoque básico tanto en su alcance como en su implementación.

Por lo que, tomando en consideración esta base, la mejora y optimización es la justificación principal para el presente proyecto, aprovechando los aprendizajes y experiencias derivados del mencionado prototipo, por lo que se desea desarrollar una solución más robusta, eficiente y versátil que permita una gestión aún más fluida y efectiva de las citas en una clínica dental. A través de la incorporación de tecnologías actuales y buenas prácticas en diseño web y desarrollo de aplicaciones, se aspira a elevar la funcionalidad y la experiencia del usuario a nuevos niveles. De ello que el enfoque se centra en enriquecer la interfaz de usuario, mejorar el proceso de reserva de citas, optimizar la administración de registros y proporcionar herramientas de seguimiento más útiles al usuario final, que se distinga por su usabilidad, eficiencia y confiabilidad.

Otro aspecto a considerar dentro del desarrollo del proyecto es el considerar la visión que guiará la creación del prototipo. En este contexto, no contamos con un cliente específico como punto de partida, sino que nos basamos en los requisitos establecidos en el proyecto original, nuestro propio criterio y experiencia, así como en el análisis de herramientas similares.

Esta perspectiva nos permite diseñar y desarrollar el prototipo de manera integral y eficiente, asegurando que cumpla con los estándares de calidad y funcionalidad necesarios, pues existe otro factor de importancia es la disponibilidad de información para el profesional de la salud bucal, dado que la facilidad de comunicación con los consultorios o clínicas dentales son puntos cruciales para la percepción del tiempo, la planificación semanal, pero también, y muy importante, para evitar conflictos de horarios.

Es común que el horario del odontólogo con los consultorios sea controlado por la secretaría de este, lo que en muchos casos ocasionan los citados “conflictos de horario” con la agenda personal del odontólogo o incluso con sus otros compromisos profesionales.

En estudios realizados a través del enfoque sistémico de (Deming, 1992), para mejorar la calidad en una organización, en la que se incluye la calidad del cuidado de la salud y la relación profesional/paciente, se menciona el “cumplimiento del tiempo programado”, es por ello que gestionar la agenda de citas de pacientes con las más variadas condiciones e incluso poder el administrador de la base de datos acceder a la información en cualquier momento, permitirá mejorar la planificación organizacional logrando así ser más efectivo y eficaz a nivel gerencial.

1.2. Objetivos de la Investigación

1.2.1. Objetivo General

Estudio y desarrollo de un prototipo de aplicación web para la gestión de citas en una clínica dental.

1.2.2. Objetivos específicos

1.2.1 Realizar un estudio de necesidades y soluciones para la gestión de una clínica dental.

1.2.2 Diseñar un sistema informático especialmente orientado a la utilización de arquitecturas y metodologías de desarrollo web.

.

1.2.3 Implementar un prototipo de aplicación web usando tecnologías de desarrollo centradas en el servidor.

1.3. Metodología

Para elaborar la propuesta para la gestión de citas en una clínica dental, se necesitó en primera instancia definir la población objeto a la cual está destinada la propuesta, que en este caso que se sigue, está conformada por pacientes, especialistas en salud bucal y el personal administrativo.

Se determinan los datos susceptibles de recopilar de la población objetivo para luego determinar el medio de distribución apropiado para la entrega de esta, tomando en consideración el aspecto tempo - espacial de acuerdo con criterios evaluados a tal fin.

En función de lo antes descrito y para alcanzar los objetivos propuestos en el presente proyecto, y, pensando que el desarrollo del proyecto está centrado en satisfacer las necesidades del usuario, se utilizará la metodología de Diseño Centrado en el usuario, también denominado DCU, por sus siglas.

El DCU, es una metodología que se enfoca en comprender las necesidades, deseos y comportamientos de los usuarios para diseñar productos y servicios que satisfagan sus necesidades de manera efectiva y eficiente. (Canal, 2015)

Es necesario reseñar que esta metodología será usada hasta la fase de Diseño del prototipo, tras lo cual se hará la transición a un enfoque iterativo que incorporará algunas técnicas y elementos derivados de SCRUM, como lo son el uso de historias de usuario, el uso de sprint, y otras prácticas ágiles que permitirán un desarrollo más eficiente y flexible del proyecto.

SCRUM es una metodología ágil de gestión de proyectos, que se utiliza principalmente en el desarrollo de software. Se basa en un enfoque iterativo e incremental para la planificación y el desarrollo de proyectos, con el objetivo de entregar un producto funcional y de alta calidad de manera más rápida y eficiente.

Si bien SCRUM es una metodología que se utiliza comúnmente en proyectos de equipos grandes y complejos, también se puede aplicar en proyectos más pequeños
y con equipos más pequeños. De hecho, Scrum se puede adaptar, para adaptarse a las necesidades de proyectos de cualquier tamaño.

Con ello encontramos en la etapa de desarrollo las llamadas interacciones del proceso (Sprint), que son entregas regulares y parciales del producto final.

En el caso que sigue al desarrollo del presente proyecto, el equipo desarrollador estará conformado por el tutor de TFG y el desarrollador del TFG, asumiendo cada quién un papel dentro del SCRUM, siendo el tutor el Dueño del producto (*Product Owner*), dado que es quién decide sobre el resultado final y sobre el orden en el que se van construyendo los diferentes incrementos.

El facilitador (*Scrum Master*) y Equipo de desarrollo (*Developer Team*), en todo caso el desarrollador de TFG, pues, planificará los Sprint, el seguimiento y control de proyecto.

Ahora con los roles bien definidos se llevarán a cabo las diferentes fases de la Metodología, con el orden predefinido:

Sprint Planning: planificación del Sprint donde se describen las tareas que realizaré, así como el tiempo que necesitaré para concluirlas.

Scrum team meeting: reuniones cortas con el tutor para evaluar el trabajo realizado y los problemas que se han presentado.

Backlog refinement: repaso de las tareas y su evolución por parte del tutor (Product Owner) para evaluar el tiempo y esfuerzo que he dedicado en cada tarea.

Sprint Review: reunión puntual con cualquier usuario para conseguir un feedback real y de calidad.

Retrospective: reunión final tras concluir el proyecto donde se revisará todo lo que ha ocurrido durante el sprint.

1.4. Estructura del Documento

En cuanto a la estructura del proyecto, el mismo se dividirá como se especifica a continuación.

Fase de análisis, en el cual se determina el público objetivo, así como el estado de diferentes clínicas odontológicas y los *frameworks* usados para la aplicación web, estableciendo comparaciones para poder determinar la idónea para desarrollar el prototipo.

Posterior a ello, se presenta la propuesta de solución, en el cual se determina en función del apartado anterior la tecnología a usar para el desarrollo del proyecto. Se realiza el diseño de prototipo, las historias de usuario y los sprints, así como el modelo de dominio o diagrama de clases conceptuales.

En el tercer apartado, el diseño, se encuentra enmarcado los diferentes diagramas de clases, de secuencia, tablas entidad-relación, y diseño de interfaz.

La cuarta sección es la Implementación, que comienza con la arquitectura del proyecto y continúa con detalles de la implementación. Esta sección incluye el modelo del *framework* utilizado, así como las funciones de creación de citas y de inicio de sesión.

En el quinto apartado, pruebas, se expondrán las pruebas de las historias de usuario, estableciendo sus criterios de aceptación, que son los requisitos que debe cumplir para considerarse implementada de manera satisfactoria. Estos criterios de aceptación son la base para las pruebas de sistemas, ya que son la guía que se sigue para comprobar que cada historia de usuario funciona como se espera.

En el apartado 6, las conclusiones, así como se presentan las posibles mejoras y trabajos futuros.

2. ANÁLISIS

2.1. Análisis Preliminar

La primera fase de este trabajo fue la investigación a través de internet a fin de recopilar información en relación con las páginas web o aplicaciones tecnológicas en España que brindarán al usuario la posibilidad de agendar, a través de este, la cita para el centro odontológico, siendo el objetivo principal determinar qué se ha hecho hasta ahora y qué queda por hacer.

Se usó a tal fin el buscador de Google, con las siguientes expresiones:

- "Aplicaciones webs en España para agendar cita odontológica": arrojando como resultado una serie de recomendaciones sobre aplicaciones web, dirigidas en su mayoría al odontólogo para gestionar su consultorio, no se encontró una aplicación para clínica odontológica que registrara citas.
- "Agendar cita odontológica en España", esta búsqueda arrojó dos sitios web para agendar citas odontológicas online Caser dental y Adeslas, no observándose alguna otra clínica odontológica con este servicio al usuario, dado que en su mayoría aportan los datos de contacto o formularios para ser llenados por el usuario para luego ser contactados.

Cabe destacar que ambas empresas odontológicas pertenecen a consorcios de gran magnitud, por lo que en consultorios pequeños o en crecimiento no se observa esta modalidad, lo que representa una oportunidad para nuestro proyecto.

En términos de relevancia, esta investigación resalta la importancia de ofrecer soluciones de gestión de citas accesibles y eficientes para clínicas dentales de todos los tamaños, aportando una simplificación de las funcionalidades, lo que podría hacer que nuestra solución sea más atractiva y fácil de adoptar para una amplia variedad de clínicas odontológicas.

En el caso de Caser dental[1], la reserva online de citas ocurre a través de tres pasos: siendo la primera una encuesta de la localidad solicitada, odontólogo de preferencia si lo tiene, fecha estimada para la consulta, tipo de consulta y horario de preferencia. Tal como se muestra en la Ilustración 1.

[1] https://citaclinicadental.es/cita-online

Posteriormente muestra un segundo paso en el cual se muestra la disponibilidad de días y horarios, como se muestra en la Ilustración 2; fin de que el usuario elija el de su preferencia y por último paso, acceder a la reserva si ya es un cliente o realizar el registro si es un nuevo usuario del sistema.

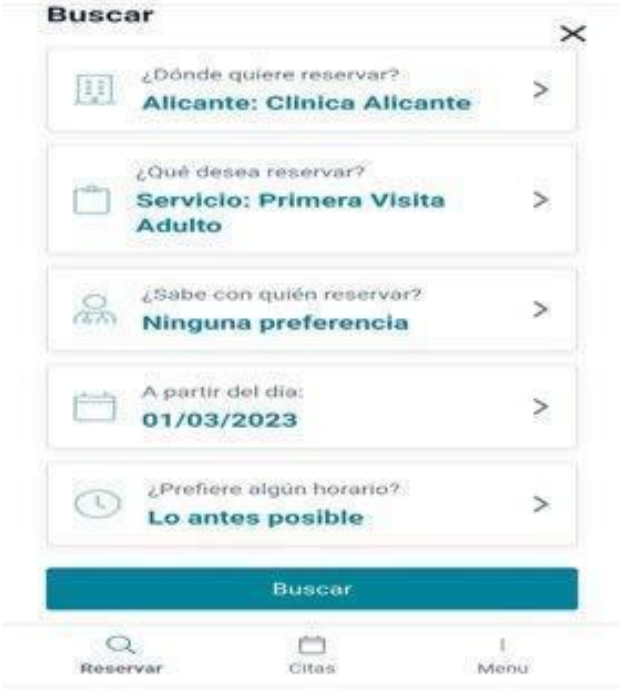


Ilustración 1: *Muestra de reserva online de Caser dental, solicitud de datos*

Ilustración 2: *Muestra de reserva online de Caser dental, selección de días y hora de la cita. Imagen tomada de 2021 Caser Servicios de Salud.https://citaclinicadental.es/cita-online*

Para el caso de Adeslas[2], las páginas de reserva son exactamente igual al de Caser dental, con la salvedad que hace una marcada diferenciación entre los asegurados Ilustración 3, los no asegurados y estos últimos si desean reservar solo como invitado.

Ilustración 3: *Muestra de reserva online de Adeslas para clientes de la empresa.*

[2] https://www.adeslasdental.es/cita-online/

2.1.1. Estudios de Tecnologías

Al desarrollar un aplicativo Web, es necesario conocer los diferentes tipos de lenguaje de programación que existen en el mercado para tal fin, y establecer una comparación entre ellos, en función del objetivo que se persigue y de la viabilidad del uso de estos en el desarrollo del proyecto.

A tal fin se realizó una búsqueda de los lenguajes de programación web más usados, así como de manejo de bases de datos y se estableció los pro y contras de cada uno de ellos.

2.1.1.1 Python[3] : es un lenguaje de programación multipropósito de alto nivel. Se puede utilizar para múltiples tareas, desde análisis y visualización de datos hasta desarrollo web, creación de prototipos y automatización. Tiene como ventaja el ser de código abierto, que no requiere licencia, altamente escalable, amplia selección de marcos para GUI (interfaz gráfica de usuario), compatible con Mac y Windows.

Es compatible con frameworks como Pyramid y Django, micro-frameworks como Bottle y Flask, y CMS avanzados como Django CMS y Plone, siendo estos flexibles, escalables, seguros y vienen con una serie de módulos y bibliotecas útiles para simplificar tareas como interactuar con bases de datos, administrar contenido, etc..; adicionalmente también admite protocolos de Internet como HTML, XML, FTP, IMAP, POP, entre otros, (Python software foundation, s/f).

Tiene en contra el ser más lento debido a que es un lenguaje interpretado, consume relativamente más memoria, no es ideal para el desarrollo de aplicaciones.

2.1.1.2 JavaScript[4] , junto con HTML y CSS, es el mejor lenguaje de programación para aprender desarrollo web *front-end*; se utiliza generalmente para mejorar la interactividad de las páginas web. Permite a los desarrolladores web agregar elementos dinámicos a las páginas de destino, como gráficos animados, botones en los que se puede hacer clic y efectos que aparecen cuando se desplaza sobre una sección específica del sitio.

Ventajas, es de código abierto, del lado del cliente es increíblemente rápido y no requiere compilación, funciona muy bien con otros lenguajes de programación populares, es ideal para mejorar la experiencia del usuario y la participación en el sitio web.

Tiene en contra, problemas de seguridad, rendimiento inestable en diferentes navegadores.

[3] https://www.python.org/
[4] https://www.javascript.com/

2.1.1.3 Structured Query Language (SQL) [5] , los desarrolladores web utilizan SQL para organizar los bancos de datos de los sitios web, así como también varios sistemas de gestión de bases de datos utilizan el lenguaje SQL, incluidos los populares MySQL y MariaDB.

Tiene como ventaja el procesamiento rápido de consultas, excelente portabilidad, seguridad robusta, altamente interactivo. Como contra, se consigue que no es de código abierto, no es el lenguaje de programación ideal para construir aplicaciones, pero sí para manejar datos.

2.1.1.4 Ruby[6] es otro popular lenguaje de programación de código abierto, se usa comúnmente en el desarrollo de aplicaciones web, pero los programadores también pueden usar Ruby para el análisis de datos y la creación de prototipos.

Las ventajas que presenta Ruby es que es un lenguaje de código abierto, rápido de escribir y fácil de depurar, sistema eficiente de *garbage collector*. Contras, es menos flexible y bastante lento, presenta problemas de subprocesos múltiples.

Es necesario reseñar que estas herramientas ofrecen una ventaja cuando de desarrollar aplicaciones alojadas en la nube se trata, pues proporcionan versatilidad, facilidad de desarrollo, escalabilidad, seguridad y capacidad de integración, lo que contribuye en gran medida al éxito y la eficiencia de la aplicación, dado que no solo acelera el desarrollo, sino que también garantiza que la aplicación esté preparada para crecer y enfrentar los desafíos que puedan surgir en la medida que la aplicación crece y atrae a más usuarios, ya que no requieren cambios drásticos en la infraestructura técnica; en su lugar, el sistema se ajusta y expande de manera eficiente.

[5] https://www.mysql.com/
[6] http://www.ruby-lang.org/es/

2.2. Propuesta de Solución

En función del objetivo que se persigue con el desarrollo del presente proyecto, la gestión de citas en una clínica dental online, teniendo presente el desarrollo de una aplicación web para tal fin, se considera usar un enfoque iterativo que incorporará algunas técnicas y elementos derivados de SCRUM, como dicho en apartados anteriores y con un patrón de arquitectura de software tipo MVC (MVC Framework, s.d.), que será el responsable de contribuir a la optimización de la velocidad entre las solicitudes realizadas por los comandos del usuario.

Dicho patrón se divide en tres componentes esenciales: modelo, controlador y vista, los cuales interactúan siguiendo el siguiente patrón:

Todo comienza con la interacción del usuario en la capa Vista. A partir de ahí, el controlador toma esta información y la envía al Modelo, que se encarga de evaluar esos datos y transmitir una respuesta.

El controlador recibe estas respuestas y envía una notificación de validación de esa información a la capa de vista, haciendo que presente el resultado de forma gráfica y visual.

Los lenguajes de programación seleccionados una vez estudiados los existentes en el mercado son:

Python: elegido por la variedad de opciones que ofrece, siendo su principal ventaja el permitir implementar autenticación de usuarios y control de acceso de manera efectiva, adicional a su carácter libre y abierto.

En cuanto al *frameworks* a usar se consideró a Flask de código abierto, que, al ser comparado con Django en cuanto a velocidad, por su diseño minimalista funciona más rápido, logrando cientos de consultas sin ningún impacto negativo; posee varias plantillas disponibles con la biblioteca de idiomas estándar de Jinja2. Además, la aplicación viene con la biblioteca del kit de herramientas WSGI y un servidor web incorporado, por lo que no necesita depender de servidores externos como Apache para probar su aplicación.

[7] https://developer.mozilla.org/es/docs/Glossary/MVC

Flask posee facilidad de integración de la base de datos dado que no tiene una capa adicional de acceso a la base de datos ni depende de ORM (Mapeo relacional de objetos). Como resultado, la integración con kits de herramientas de bases de datos como SQLAlchemy, NoSQL, MongoDB y DynamoDB es muy sencilla.
Jinja2, es un motor de plantillas para Python que facilita el uso de HTML, dentro de las aplicaciones de Python, tiene compatibilidad total con Unicode y, para las aplicaciones que necesitan mayor seguridad, agrega sandbox.

Por su parte WSGI, es la interfaz de puerta de enlace del servidor web, o dicho de otra forma, es una especificación para una interfaz universal simple entre servidores web y aplicaciones web o frameworks de Python.

Por último, se menciona a MySQL, como base de datos para el desarrollo de la aplicación Web, siendo esta la elegida por su elevado desempeño, uso gratuito, ser multiusuario, robusto y con un alto grado de seguridad, así como la compatibilidad que presenta con los diversos sistemas operativos.

Es de reseñar que la selección de las diversas herramientas de desarrollo del aplicativo web, se han seleccionado para brindar robustez en el programa, mayor simplicidad y flexibilidad, así como presentar una web más segura y escalable en función del crecimiento de la clínica dental.

Esto se deriva de las posibles mejoras que podría ofrecer tales herramientas, una vez analizado el proyecto previo mencionado en el apartado del Análisis preliminar. Dichas mejoras pueden resumirse a través de la Tabla 1.

Aspecto / Herramienta	Proyecto Anterior	Proyecto Actual	Mejora en el Proyecto Actual
Lenguaje de Programación	Java	Python	Python ofrece sintaxis más concisa y legible, agilizando el desarrollo y mantenimiento del código. Reduce probabilidades de errores.
Interfaz Web y Estilo	Bootstrap, HTML5, CSS	Bootstrap, JQuery, CSS	Incorporación de JQuery para una interfaz más interactiva y la dinámica.
Base de Datos	(No especificado)	MySQL	MySQL ofrece mayor rendimiento y escalabilidad.
Backend Framework	SpringMVC (Simplicidad)	Flask Framework (Eficiencia)	Utilización de Flask Framework para un backend más eficiente.

Tabla 1: *Cuadro comparativo herramientas tecnológicas*

2.3. Épicas del proyecto

Como se ha mencionado en apartados anteriores, el desarrollo del prototipo se realizará bajo el estilo de un enfoque iterativo que incorporará algunas técnicas y elementos derivados de SCRUM, motivo por el cual se usan ciertos aspectos de este.

En Agile, una épica (o *epic*) es un método para clasificar los elementos de la cartera de productos como parte de la mentalidad ágil. Sirve para nombrar tareas que implican una gran cantidad de trabajo y, por tanto, necesitan dividirse en partes más pequeñas (historias de usuario o *User Stories*).

Como descrito en apartados anteriores, la mejora que se propone del proyecto realizado por (Martínez, 2018), implica el uso de épicas, dado que estas permiten evaluar y priorizar funcionalidades de manera más informada, evitando así que la falta de agrupación conlleve a priorizar características individualmente sin considerar su relación o impacto en conjunto. Con las *epics*, es más fácil comparar y tomar decisiones basadas en el valor y la complejidad de las funcionalidades en un contexto más amplio.

Para el proyecto que se sigue y en función de realizar una mayor organización y visión general a través de una estructura efectiva para organizar funcionalidades en forma coherente bajo un tema en común, se plantea las épicas como se muestran en la Tabla 2.

En dicha tabla, se puede observar no solo la distribución de los epics, si no las historias de usuario planteadas como necesidades dentro de la organización, desglosando así las funcionalidades de la aplicación en unidades más manejables, facilitando de esta forma la futura planificación y ejecución de las tareas.

Épicas	Historias de usuario
Gestión de usuarios y personal	Historia de Usuario 1: Dar de alta
	Historia de Usuario 2: Modificar datos personales
	Historia de Usuario 3: Dar de baja
	Historia de Usuario 5: Listado de Clientes
	Historia de Usuario 6: Listado de Personal
Gestión de citas y agenda	Historia de usuario 7: Dar de alta como paciente de la clinica
	Historia de Usuario 4: Eliminar citas
	Historia de Usuario 8: Concretar cita
	Historia de Usuario 9: Visualización de cita
	Historia de Usuario 10: Anular cita
	Historia de Usuario 11: Historial de citas
	Historia de Usuario 29: Modificar Dr. Asignado a la cita
	Historia de Usuario 14: Visualizar listado de citas
Gestión de observaciones	Historia de Usuario 12: Añadir observaciones
	Historia de Usuario 13: Visualizar observaciones
Reportes e informes	Historia de Usuario 15: Obtener informe
	Historia de Usuario 30: Revisar avisos por email
Comunicación y notificaciones	Historia de Usuario 16: Aviso por email
	Historia de Usuario 31: Comunicaciones para el empleado/doctor
Odontograma y visualización de datos	Historia de Usuario 17: Visualizar odontograma del paciente
	Historia de Usuario 18: Editar el odontograma de un cliente
Facturación y stock	Historia de Usuario 19: Crear una nueva factura
	Historia de Usuario 20: Editar factura
	Historia de Usuario 21: Eliminar factura
	Historia de Usuario 22: Mostrar listado de morosos
	Historia de Usuario 23: Tener en cuenta el stock de materiales
	Historia de Usuario 24: Consultar stock
	Historia de Usuario 25: Modificar stock
	Historia de Usuario 26: Ordenar productos por Trabajo/Operación
	Historia de Usuario 27: Ordenar productos por Familia
	Historia de Usuario 28: Mostrar listado de productos sin stock

Tabla 2: *Épicas del prototipo de aplicación web para la gestión de citas en una clínica dental*

Es de hacer notar que estas las épicas están marcadas con un distintivo color azul con el fin de diferenciar las épicas realizadas en el proyecto base con las propuestas como mejoras en el proyecto actual y serán representadas en la sección 2.4.1, con el fin de diferenciarlas en las historias de usuario.

Es relevante señalar que, aunque el proyecto previo incluía entre sus historias de usuario la creación, edición y visualización de odontogramas, estas funcionalidades no fueron desarrolladas y, en consecuencia, no se lograron implementar y tomando en consideración que la incorporación y adecuado manejo de los odontogramas son elementos cruciales que contribuyen a mejorar la calidad de la atención bucal tanto para los profesionales de la salud dental como para los pacientes, esta carencia representa una oportunidad destacada para la optimización del proyecto actual.

2.4. Historias de Usuario

De acuerdo con el estilo de metodología seleccionado para el proyecto, el próximo paso consiste en dividir las épicas en historias de usuario. En metodologías ágiles, una historia de usuario es una descripción breve, informal y en lenguaje sencillo de lo que es un usuario quiere hacer dentro de un producto de software para obtener algo que considera valioso.

Las historias de usuario suelen seguir el patrón (o modelo) rol-característica-beneficio:

- Como (tipo de usuario)
- Quiero (una acción)
- Para que (un beneficio/valor)

Adicionalmente a un formato estandarizado y elementos completos, una buena historia de usuario también debe seguir los principios de INVEST (Scrum Manager Bok, s.d.):

- Independiente
- Negociable
- Valioso
- Estimable
- Pequeño (small)
- Comprobable (testable)

De acuerdo con el prototipo a desarrollar se plantea un primer diagrama para dar una mejor visualización de las necesidades más básicas de los involucrados en el sistema, según se observa en la Ilustración 4.

Esta propuesta del prototipo en desarrollo, se plantea abordar inicialmente, y siguiendo el enfoque de épicas, las historias de usuario delineadas en el proyecto de (Martinez, 2018).

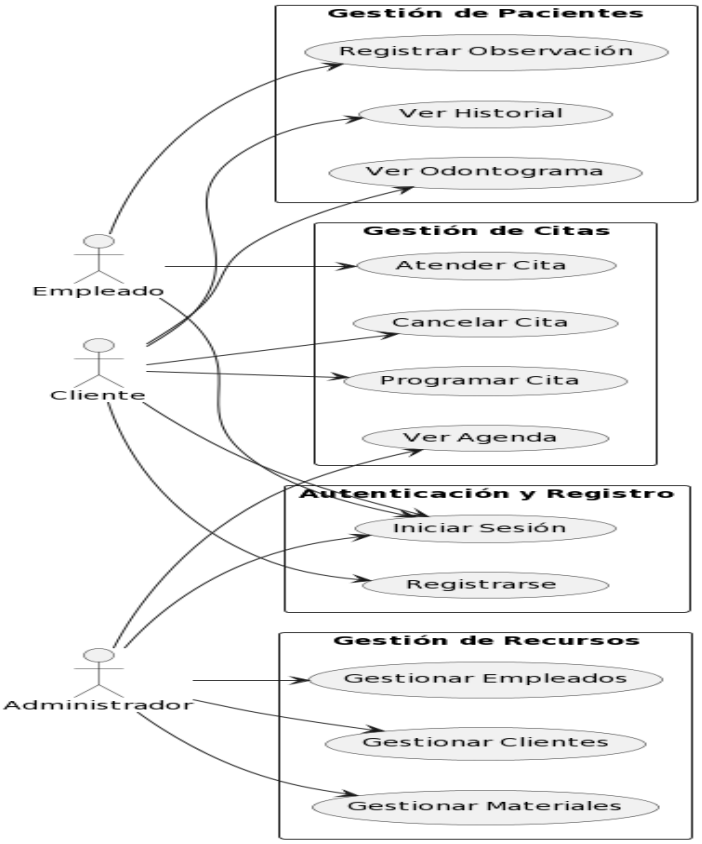


Ilustración 4: *Diagrama de casos de uso*

Adicionalmente, es fundamental reforzar la justificación expuesta en la sección de referencia en la que se aborda el enfoque sistémico de (Derming, 1992) y su aplicación para la mejora de la calidad en una organización, y es que, la creación de nuevas historias de usuario en el presente proyecto no solo tiene el propósito de otorgar mayor funcionalidad al sistema, sino también de aprovechar un enfoque sistémico para optimizar la experiencia global del usuario, elevar los estándares de atención y mejorar la eficiencia en la gestión de la clínica dental.

En función de lo anteriormente expuesto se realiza una plantilla donde se anotan las historias de los usuarios, reflejando en ella aspectos de importancia y relevancia en la prioridad, así como la descripción de esta.

Esto con la finalidad de poder a posteriori determinar los sprint siguiendo el enfoque iterativo dado en este proyecto

2.4.1. Detalles de las historias de usuario

Se muestran a continuación las tablas de historias, identificada a través de un número (representada en la tabla), tomada y adaptada de (Scrum Manager Bok, s.d.)

Historia de Usuario			
Número	1	**Usuario**	Administrador
Nombre de la historia	Dar de alta		
Prioridad	Alta	**Riesgo**	
Puntos estimados	8	**Responsable**	
Descripción			
Como administrador quiero entrar en el sistema para dar de alta al usuario y al personal			
Validación			
El administrador podrá acceder a la aplicación para activar un nuevo usuario			

Historia de Usuario			
Número	2	**Usuario**	Administrador
Nombre de la historia	Modificar datos personales		
Prioridad	Media	**Riesgo**	
Puntos estimados	3	**Responsable**	
Descripción			
Como administrador poder editar y modificar los datos errados en pacientes y personal de la clínica			
Validación			
El administrador podrá acceder a la aplicación para modificar datos personales			

Historia de Usuario			
Número	3	**Usuario**	Administrador
Nombre de la historia	Dar de baja		
Prioridad	Alta	**Riesgo**	
Puntos estimados	5	**Responsable**	
Descripción			
Como administrador quiero entrar en el sistema para dar de baja al usuario y personal			
Validación			
El usuario final podrá acceder a la aplicación para desactivar un nuevo usuario			

Historia de Usuario			
Número	4	**Usuario**	Administrador
Nombre de la historia	Eliminar citas		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como administrador quiero entrar en el sistema para poder eliminar una cita a petición del cliente			
Validación			
El usuario final podrá acceder a la aplicación para borrar una cita			

Historia de Usuario			
Número	5	**Usuario**	Administrador
Nombre de la historia	Listado de Clientes		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como administrador quiero obtener un listado de mis clientes			
Validación			
El administrador tiene acceso a la lista de clientes			

Historia de Usuario			
Número	6	**Usuario**	Administrador
Nombre de la historia	Listado de Personal		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como administrador quiero obtener un listado del personal que trabaja dentro de la clínica.			
Validación			
El administrador tiene acceso a la lista de personal			

Historia de Usuario			
Número	7	**Usuario**	Paciente
Nombre de la historia	Dar de alta como paciente en la clínica		
Prioridad	Alta	**Riesgo**	
Puntos estimados	5	**Responsable**	
Descripción			
Como usuario web quiero darme de alta para ser paciente de la clínica.			
Validación			
El usuario final podrá acceder a la sección nuevo usuario de la aplicación para rellenar el formulario de alta con los datos solicitados			

Historia de Usuario			
Número	8	**Usuario**	Paciente
Nombre de la historia	Concretar cita		
Prioridad	Alta	**Riesgo**	
Puntos estimados	5	**Responsable**	
Descripción			
Como paciente quiero concretar mi cita en la clínica			
Validación			
El usuario final podrá acceder a la sección calendario de la aplicación para elegir la fecha y hora de su cita.			

Historia de Usuario			
Número	9	**Usuario**	Paciente
Nombre de la historia	Visualización de cita		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como paciente quiero visualizar el día de mi cita.			
Validación			
El usuario final podrá acceder a la sección calendario de la aplicación para consultar y recordar su cita			

Historia de Usuario			
Número	10	**Usuario**	Paciente
Nombre de la historia	Anular cita		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como paciente quiero cancelar el día de mi cita			
Validación			
El usuario final podrá acceder a la sección calendario de la aplicación para desactivar su cita y anularla			

Historia de Usuario			
Número	11	**Usuario**	Paciente
Nombre de la historia	Historial de citas		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como paciente quiero ver mi historial de citas con sus respectivas observaciones			
Validación			
El usuario final podrá acceder al historial de citas, después de haber ingresado en ella con su contraseña, se despliega las citas y las observaciones.			

Historia de Usuario			
Número	12	**Usuario**	Personal Dental
Nombre de la historia	Añadir observaciones		
Prioridad	Alta	**Riesgo**	
Puntos estimados	5	**Responsable**	
Descripción			
Como personal quiero añadir observaciones a la historia del paciente			
Validación			
El usuario final podrá acceder al historial de citas y añadir observaciones.			

Historia de Usuario			
Número	13	**Usuario**	Personal Dental
Nombre de la historia	Visualizar observaciones		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como persona quiero visualizar las observaciones en la historia del paciente			
Validación			
El usuario final podrá acceder al historial de citas y visualizar las observaciones.			

Historia de Usuario			
Número	14	**Usuario**	Personal/Administrador
Nombre de la historia	Visualizar listado de citas		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como personal y administrador quiero visualizar las citas en un período determinado de tiempo			
Validación			
El usuario final podrá acceder al listado de citas en el periodo solicitado			

Historia de Usuario			
Número	15	**Usuario**	Personal Dental
Nombre de la historia	Obtener informes		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como personal médico quiero obtener informe del estado de salud bucal del cliente			
Validación			
El personal médico obtiene el informe específico del cliente solicitado			

Historia de Usuario			
Número	16	**Usuario**	Paciente
Nombre de la historia	Aviso por email		
Prioridad	Media	**Riesgo**	
Puntos estimados	3	**Responsable**	
Descripción			
Como paciente quiero recibir una confirmación de la cita por email			
Validación			
El paciente recibe la confirmación de la cita por email			

Historia de Usuario			
Número	17	**Usuario**	Personal dental
Nombre de la historia	Visualizar odontograma del paciente		
Prioridad	Alta	**Riesgo**	
Puntos estimados	5	**Responsable**	
Descripción			
Como personal dental quiero visualizar la odontograma del paciente			
Validación			
El personal dental visualiza sin problema el odontograma del paciente			

Historia de Usuario			
Número	18	**Usuario**	Personal dental
Nombre de la historia	Editar el odontograma de un cliente		
Prioridad	Media	**Riesgo**	
Puntos estimados	3	**Responsable**	
Descripción			
Como personal dental quiero editar la odontograma del paciente en cada cita de ser necesario			
Validación			
El personal dental quiero editar la odontograma del paciente en cada cita de ser necesario			

Historia de Usuario			
Número	19	**Usuario**	Personal/Administrador
Nombre de la historia	Crear una nueva factura		
Prioridad	Alta	**Riesgo**	
Puntos estimados	5	**Responsable**	
Descripción			
Como personal deseo elaborar una factura por los servicios prestados			
Validación			
El personal elabora una factura por los servicios prestados			

Historia de Usuario			
Número	20	**Usuario**	Personal/Administrador
Nombre de la historia	Editar factura		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como personal deseo poder editar una factura por los servicios prestados en caso de error, antes de la impresión.			
Validación			
Como personal puedo editar una factura por los servicios prestados en caso de error antes de la impresión			

Historia de Usuario			
Número	21	**Usuario**	Administrador
Nombre de la historia	Eliminar factura		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como administrador deseo poder eliminar una factura por los servicios prestados en caso de error.			
Validación			
Como administrador puedo editar una factura por los servicios prestados en caso de error.			

Historia de Usuario			
Número	22	**Usuario**	Personal/Administrador
Nombre de la historia	Mostrar listado de morosos		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como personal deseo poder obtener un listado de morosos			
Validación			
Como personal obtengo un listado de morosos			

Historia de Usuario			
Número	23	**Usuario**	Personal Dental
Nombre de la historia	Tener en cuenta el stock de materiales		
Prioridad	Alta	**Riesgo**	
Puntos estimados	5	**Responsable**	
Descripción			
Como personal deseo poder ser informado de la necesidad de adquirir materiales para el servicio dental.			
Validación			
Como personal deseo poder ser informado de la necesidad de adquirir materiales para el servicio dental.			

Historia de Usuario			
Número	24	**Usuario**	Personal/Administrador
Nombre de la historia	Consultar stock		
Prioridad	Media	**Riesgo**	
Puntos estimados	3	**Responsable**	
Descripción			
Como personal deseo poder consultar individualmente el stock de un material			
Validación			
Como personal puedo consultar individualmente el stock de un material			

Historia de Usuario			
Número	25	**Usuario**	Administrador
Nombre de la historia	Modificar stock		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como Administrador deseo poder modificar individualmente el stock de un material			
Validación			
Como Administrador puede modificar individualmente el stock de un material			

Historia de Usuario			
Número	26	**Usuario**	Administrador
Nombre de la historia	Ordenar productos por Trabajo/Operación		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como administrador deseo poder saber que productos son usados por cada servicio prestado			
Validación			
Como administrador puedo saber que productos son usados por cada servicio prestado			

Historia de Usuario			
Número	27	**Usuario**	Administrador
Nombre de la historia	Ordenar productos por Familia		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como administrador deseo poder saber que productos pertenecen a cada categoría			
Validación			
Como administrador puedo saber que productos pertenecen a cada categoría			

Historia de Usuario			
Número	28	**Usuario**	Personal/Administrador
Nombre de la historia	Mostrar listado de productos sin stock		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como personal/administrador deseo poder consultar el listado de productos sin stock			
Validación			
Como personal/administrador puede consultar el listado de productos sin stock			

Historia de Usuario			
Número	29	**Usuario**	Administrador
Nombre de la historia	Modificar Dr. Asignado a la cita		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como administrador deseo poder reasignar o modificar el Dr. asignado a una cita			
Validación			
Como administrador puede reasignar o modificar el Dr. asignado a una cita			

Historia de Usuario			
Número	30	**Usuario**	Administrador
Nombre de la historia	Revisar avisos por email		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como personal deseo poder verificar si se han enviado los emails de confirmación			
Validación			
Como personal puedo verificar si se han enviado los emails de confirmación			

Historia de Usuario			
Número	31	**Usuario**	Administrador
Nombre de la historia	Comunicaciones para el empleado/doctor		
Prioridad	Baja	**Riesgo**	
Puntos estimados	2	**Responsable**	
Descripción			
Como administrador deseo poder enviar comunicaciones a los empleados/doctores			
Validación			
Como administrador puedo enviar comunicaciones a los empleados/doctores			

2.5. Planificación inicial de tareas

Para la planificación de las tareas se toman en consideración los *story points* y la prioridad establecida a través de ciertas consideraciones previas, tales como la maximización del valor, que no es más que abordar en primera instancia las historias que aportan mayor valor a la empresa, seguido de la identificación de las dependencias para abordar aquellas que son requisitos previos de otras historias de usuarios y de aquellas historias que posean un mayor impacto dentro del proyecto.

Esto implica que en ocasiones las historias de usuario no se desarrollen en el orden presentado, si no que se prioricen en los *sprints* de acuerdo con lo expresado en el párrafo anterior y en el grado de dificultad que represente para ser desarrollado, es así como se obtiene la Tabla 3, que será la que determinará la planificación de los *sprints*.

Nº	Épicas	Historias de usuario	Story/Point	Prioridad
1	Gestión de usuarios y personal	Historia de Usuario 1: Dar de alta	8	alta
		Historia de Usuario 2: Modificar datos personales	3	media
		Historia de Usuario 3: Dar de baja	5	alta
		Historia de Usuario 5: Listado de Clientes	2	baja
		Historia de Usuario 6: Listado de Personal	2	baja
2	Gestión de citas y agenda	Historia de usuario 7: Dar de alta como paciente de la clinica	5	alta
		Historia de Usuario 4: Eliminar citas	2	baja
		Historia de Usuario 8: Concretar cita	5	alta
		Historia de Usuario 9: Visualización de cita	2	media
		Historia de Usuario 10: Anular cita	2	baja
		Historia de Usuario 11: Historial de citas	2	baja
		Historia de Usuario 29: Modificar Dr. Asignado a la cita	2	baja
		Historia de Usuario 14: Visualizar listado de citas	2	media
3	Gestión de observaciones	Historia de Usuario 12: Añadir observaciones	5	alta
		Historia de Usuario 13: Visualizar observaciones	2	media
4	Reportes e informes	Historia de Usuario 15: Obtener informe	2	baja
		Historia de Usuario 30: Revisar avisos por email	2	baja
5	Comunicación y notificaciones	Historia de Usuario 16: Aviso por email	3	media
		Historia de Usuario 31: Comunicaciones para el empleado/doctor	2	baja
6	Odontograma y visualización de datos	Historia de Usuario 17: Visualizar odontograma del paciente	5	alta
		Historia de Usuario 18: Editar el odontograma de un cliente	3	media
7	Facturación y stock	Historia de Usuario 19: Crear una nueva factura	5	alta
		Historia de Usuario 20: Editar factura	2	baja
		Historia de Usuario 21: Eliminar factura	2	baja
		Historia de Usuario 22: Mostrar listado de morosos	2	baja
		Historia de Usuario 23: Tener en cuenta el stock de materiales	5	alta
		Historia de Usuario 24: Consultar stock	3	media
		Historia de Usuario 25: Modificar stock	2	baja
		Historia de Usuario 26: Ordenar productos por Trabajo/Operación	2	baja
		Historia de Usuario 27: Ordenar productos por Familia	2	baja
		Historia de Usuario 28: Mostrar listado de productos sin stock	2	baja

Tabla 3: *Determinación de story points y prioridades por historias de usuario*

Para establecer las tareas que conformarán cada Sprint, se toma como base los puntos estimados de cada historia de usuario.

Un *Story Point* es una unidad de complejidad de esfuerzo y riesgo que presenta una determinada tarea o que se necesita en la creación de software, por ejemplo, una historia de usuario o cualquier tipo de solicitud que necesite crear. A menor punto de historia, menor la complejidad, el riesgo y el esfuerzo.

Esto determina la cantidad de *story point* que se involucran en un sprint, estimando que cada sprint puede durar entre 2 y 4 semanas.
En el apartado anterior de historias de usuario se estableció la complejidad y se determinó la estimación de los puntos de historias, con un total de 93, motivo por el cual se procede a establecer los sprint para el desarrollo del prototipo web.

Se hace preciso acotar que si bien algunas historias de usuarios son similares al proyecto realizado por (Martinez, 2018), como ya ha sido plasmado en apartados anteriores, se hace necesario su reimplementación, siendo en algunos casos un esfuerzo menor en el aspecto de la codificación, pues al usar Python, este permite una reimplementación más fluida por su sintaxis más legible y el uso de los *frameworks* que permiten enfoques más eficientes para generar y presentar listas, validar y procesar las observaciones, entre otros.

Siendo así las posibles historias de usuario con menor esfuerzo de codificación las siguientes: dar de alta, modificar datos personales, listado de clientes, visualizar listado de citas, añadir observaciones y visualizar observaciones.

Sin embargo, este hecho no implica por sí mismo una reducción de los *story points* pues, aunque la reimplementación puede ser técnicamente más sencilla debido a similitudes en la lógica o en la estructura de datos, existen factores que influyen en la estimación de estos, como es la interdependencia de los procesos.

2.5.1. Evolución de la planificación de tareas

Respecto a la realización de las historias de usuario que se han ido desarrollando según dicha planificación, hay que tener en cuenta que a pesar de sufrir algunas desviaciones que se fueron ajustando según las prioridades del sistema, se han podido realizar todas las épicas que se concretaron.

Este hecho no quiere decir que parte de ellas se completarán completamente, ya que la épica de Facturación y Stock no ha podido desarrollarse en toda su plenitud a causa de haber agotado el tiempo de trabajo.

Debido a la reimplementación de las historias existentes y todas las historias añadidas posteriormente, se ha producido un desbordamiento del trabajo planificado, en el que no han podido cubrirse todas las historias de usuario planificadas en un principio.

Fecha de inicio	08-mar
Fecha final	28-ago

Nombre de la tarea	H.usuario	Responsable	Fecha de inicio	Fecha final	Puntos Historias
Sprint 1		Analista/Programador	08-03-2023	15-04-2023	16
Dar de alta administrador: personal/cliente	1	Analista/Programador	08-03-2023	15-03-2023	8
Modificar datos personales	2	Analista/Programador	15-03-2023	04-04-2023	3
Dar de alta como paciente de la clinica	7	Analista/Programador	04-04-2023	15-04-2023	5
Sprint 2		Analista/Programador	04-04-2023	26-04-2023	16
Concretar cita	8	Analista/Programador	04-04-2023	10-04-2023	5
Visualizar cita	9	Analista/Programador	10-04-2023	13-04-2023	2
Listado de clientes	5	Analista/Programador	10-04-2023	17-04-2023	2
Listado de personal	6	Analista/Programador	10-04-2023	17-04-2023	2
Dar de baja	3	Analista/Programador	17-04-2023	26-04-2023	5
Sprint 3		Analista/Programador	26-04-2023	15-05-2023	10
Visualizar odontograma	17	Analista/Programador	26-04-2023	10-05-2023	5
Editar el odontograma de un cliente	18	Analista/Programador	10-05-2023	13-05-2023	3
Historial de citas	11	Analista/Programador	13-05-2023	15-05-2023	2
Sprint 4		Analista/Programador	26-04-2023	25-05-2023	13
Modificar Dr. Asignado a la cita	29	Analista/Programador	26-04-2023	03-05-2023	2
Eliminar cita	4	Analista/Programador	26-04-2023	03-05-2023	2
Añadir observaciones	12	Analista/Programador	03-05-2023	24-05-2023	5
Anular cita	10	Analista/Programador	03-05-2023	25-05-2023	2
Visualizar lista de citas	14	Analista/Programador	03-05-2023	25-05-2023	2
Sprint 5		Analista/Programador	25-05-2023	13-06-2023	12
Añadir observación	12	Analista/Programador	25-05-2023	28-05-2023	5
Aviso por email	16	Analista/Programador	28-05-2023	05-06-2023	3
Visualizar observaciones	13	Analista/Programador	05-06-2023	10-06-2023	2
Eliminar cita	4	Analista/Programador	10-06-2023	13-06-2023	2
Sprint 6		Analista/Programador	13-06-2023	06-07-2023	11
Crear una nueva factura	19	Analista/Programador	16-06-2023	26-06-2023	5
Editar factura	20	Analista/Programador	26-06-2023	29-06-2023	2
Eliminar factura	21	Analista/Programador	29-06-2023	02-07-2023	2
Mostrar listado de morosos	22	Analista/Programador	02-07-2023	06-07-2023	2
Sprint 7		Analista/Programador	06-07-2023	12-08-2023	14
Tener en cuenta el stock de materiales	23	Analista/Programador	06-07-2023	26-07-2023	5
Consultar stock	24	Analista/Programador	26-07-2023	30-07-2023	3
Modificar stock	25	Analista/Programador	30-07-2023	04-08-2023	2
Ordenar productos por Trabajo/Operación	26	Analista/Programador	04-08-2023	08-08-2023	2
Ordenar productos por Familia	27	Analista/Programador	08-08-2023	12-08-2023	2
Sprint 8		Analista/Programador	12-08-2023	28-08-2023	8
Mostrar listado de productos sin stock	28	Analista/Programador	12-08-2023	16-08-2023	2
Comunicaciones para el empleado/doctor	31	Analista/Programador	16-08-2023	20-08-2023	2

2.6. Estudio de Viabilidad

En este apartado, se analiza la viabilidad del prototipo basándose en tres (3) aspectos, como se menciona a continuación:

- ***Factibilidad técnica**:* entendida esta como la disponibilidad de tecnología que satisfaga las necesidades inherentes al desarrollo del proyecto en curso, que en el caso que se sigue por ser un prototipo basado en las tecnologías ya mencionadas en el apartado anterior y siendo de código abierto y de gran uso en la actualidad, facilita su instalación, uso y por ende el desarrollo del proyecto. Se considera en este apartado los equipos de hardware con el cual se cuenta en la actualidad, siendo las características de este las siguientes:

Nombre del dispositivo	LAPTOP-VQ0UC3VN
Procesador	Intel® Core™ i7-5500U CPU@ 2.40GHz
RAM instalada	6.00GB
Identificador de dispositivo	9C617E53-16ª4-4007-A834-D41E74FCE0FF
Id. Del producto	00326-10000-00000-AA023
Tipo de Sistema	64 bits procesador basado en x64

- ***Factibilidad Operativa:*** dependiente de los recursos humanos necesarios para desarrollar el prototipo, refiriéndose a tal fin de un técnico - programador con las habilidades para desarrollar el aplicativo en los lenguajes seleccionados, siendo que el autor del presente proyecto cuenta con el perfil deseado, puede considerarse factible a nivel operativo.

- ***Factibilidad Financiera:*** en este apartado se mencionan o tratan los costes de realizar el prototipo web para solicitar cita en una clínica dental, para ello se divide en costes de software, personal involucrado en el desarrollo y otros gastos inherentes al proyecto.

En cuanto a los gastos de personal, se requiere como se mencionó en el párrafo anterior de un técnico programador, que en este caso es el que desarrolló el proyecto presentado.

Tomando en consideración que el TFG se corresponde con 12 créditos de dedicación del estudiante lo que supone 300 horas de trabajo que, distribuidas a lo largo de un cuatrimestre, 8

de marzo al 28 de agosto se calcula en aproximadamente 24 semanas de 5 días laborables, lo que implica en que promedio se trabaja 2,5 hrs diarias.

En cuanto a los costos de licencia de software a utilizar, se tiene lo siguiente:

Costos de licencias de software desarrollo aplicativo Web clínica dental

Licencia	Costo
Python	Gratuito
Flask	Gratuito
Jinja2	Gratuito
MySQL	Gratuito

Es importante tener en cuenta que en el contexto de la viabilidad del proyecto de TFG, los costos relacionados con el desarrollo del programa son asumidos por el propio autor del proyecto. En este sentido, se ha asignado un valor simbólico al tiempo invertido, estableciendo una tarifa de 15 euros por hora como referencia para calcular los costos asociados a la mano de obra.

Realizando un resumen de los costos totales a invertir en el proyecto se tiene lo siguiente:

Item	Costo
Software	0 €
Hardware	0 €
Internet (33 € * 6)	99 €
Gasto de Personal (300*15)	4.500 €
Total Gastos proyecto	4.599 €

2.7. Modelo de dominio

Un modelo de dominio es una representación conceptual de los objetos, entidades, relaciones y reglas de negocio que existen dentro de un área específica de conocimiento. Es una abstracción de la realidad que permite a los desarrolladores y expertos de software comprender mejor el contexto en el que se está trabajando y cómo los diferentes elementos interactúan entre sí.

En el desarrollo de software, un modelo de dominio se utiliza para definir las clases, atributos y métodos que se utilizarán en el diseño y la implementación de una aplicación.
Al crear un modelo de dominio, los desarrolladores pueden identificar las entidades clave y las relaciones entre ellas, lo que les permite crear una representación más precisa y coherente del sistema que están construyendo. (IBM, 2021).

En el proyecto que se sigue el Diagrama modelo de dominio se encuentra representado como se muestra en la Ilustración 5; en el cual se muestran claramente las clases y relaciones que contiene el sistema web a desarrollar.

Como se observa en dicha Ilustración, el diagrama contiene:

- Nombre de la clase: El nombre debe ser descriptivo y representar de manera clara el concepto que se está modelando.
- Atributos: La descripción de la clase debe incluir una lista de todos los atributos que la clase posee. Cada atributo debe ser descrito detalladamente, indicando su tipo, rango de valores posibles, y si es opcional o requerido.
- Relaciones con otras clases: La descripción de la clase debe explicar las relaciones que ésta tiene con otras clases en el modelo de dominio, indicando si la relación es de uno a uno, uno a muchos o muchos a muchos.

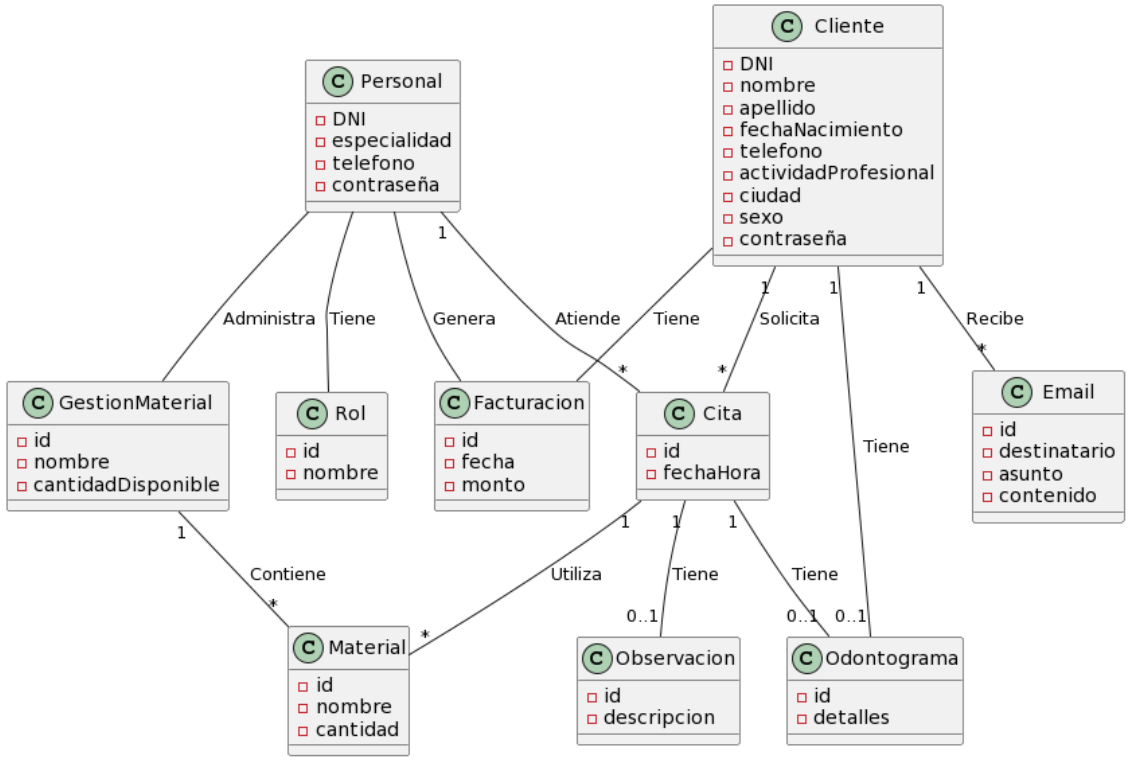


Ilustración 5: *Diagrama modelo de dominio de un prototipo de aplicación web para la gestión de citas en una clínica dental*

Entonces se cuenta con la clase *Personal* el cual representa a los miembros del personal de la clínica dental, en el que se incluye a los empleados y el administrador del sistema y contiene información sobre su identificación, especialidad, número telefónico, contraseña, los cuales tendrán acceso al sistema de acuerdo con el rol que desempeñen y que se comenta en la *clase roles*.

La clase *Roles*, representa los diferentes roles que pueden tener los miembros del personal en el sistema de la página web, y contiene información sobre el identificador y el nombre del rol.

La clase *Cliente* representa a los pacientes que solicitan citas en la clínica dental, y contiene información sobre su identificación, nombre, apellido, fecha de nacimiento, número telefónico, ciudad, sexo y contraseña de acceso. Cada cliente tiene asociado su odontograma donde se registran detalles específicos de cada uno de sus dientes para control en las citas odontológicas.

La clase *Cita* representa a las citas agendadas entre un cliente y un miembro del personal de la clínica dental. Esta clase contiene información sobre la identificación de la cita, la fecha y hora de la cita, el identificador del cliente y el identificador del personal que atenderá la cita, se encuentra asociada a observaciones y a odontograma para control de lo realizado en cada cita, así como el material usado en el paciente.

La clase *Observación* representa las observaciones realizadas por el personal de la clínica dental durante la cita, y contiene información sobre la identificación de la observación, la descripción de la observación y el identificador de la cita correspondiente.

En cuanto a *Facturación*, en ella se registra la información sobre las factura generada para la cita realizada. Contiene detalles como la fecha de facturación, el monto total y el cliente asociado. Las facturas son generadas por el personal y están relacionadas con los clientes y sus tratamientos.

Siguiendo con *Material*, se representan los suministros utilizados en los procedimientos dentales, se registra el nombre del material y la cantidad utilizada en la cita.

Mientras que el email representa los correos electrónicos que pueden ser enviados a los clientes con su cita correspondiente y al personal dentro del sistema de gestión de la clínica dental. Contiene atributos como id para identificación única, destinatario para indicar el receptor del correo, asunto para el tema del correo y contenido para el cuerpo del mensaje.

Finalmente, en *Gestión de Material*, se representa el control administrativo de los recursos y suministros disponibles o fuera de stock de la clínica dental.

En la relación entre Cliente y Personal, el campo DNI es la clave que los relaciona. La relación entre Personal y Cita es una relación uno a muchos, es decir, un personal puede tener varias citas asociadas, pero una cita solo puede pertenecer a un personal.

3. DISEÑO

El diseño de una aplicación o sistema de software puede involucrar varios tipos de diagramas que describen diferentes aspectos de la estructura, la funcionalidad y la interacción entre los elementos del sistema. A continuación, se describe brevemente cada uno de los diagramas mencionados:

Diagrama de entidad-relación (ER): Es un modelo conceptual utilizado para representar las entidades y las relaciones entre ellas en una base de datos. Este tipo de diagrama muestra las tablas de la base de datos, las columnas de cada tabla y las relaciones entre las tablas.

Diagrama de clases: Es un modelo de objetos utilizado para representar las clases, los atributos y los métodos de un sistema. Este tipo de diagrama muestra las clases del sistema, las propiedades de cada clase y las relaciones entre las clases.

Diagrama de secuencia: Son diagramas de comportamiento que muestran la interacción entre objetos en un sistema en secuencia cronológica. Estos diagramas se utilizan para modelar cómo los objetos colaboran para lograr una tarea específica.

Diagrama de interfaz: Es un diagrama que muestra la interfaz gráfica de usuario de aplicación o sistema. Este tipo de diagrama se utiliza para mostrar la estructura de la pantalla, los componentes de la interfaz de usuario y las interacciones entre ellos.

Cada uno de estos tipos de diagramas pueden ser utilizados en diferentes etapas del proceso de diseño del software, desde la conceptualización y planificación hasta la implementación y pruebas.

3.1. Modelo Entidad - Relación

Para mostrar este tipo de modelo se hará uso de las tablas más relevantes del sistema, como a continuación se muestran:

Nombre atributo	**Tipo**	**Clave**	**Descripción**
Id_usuario	integer	Primary Key	Identificador de usuario
rol	Varchar		Rol usuario, privilegios

Tabla 4: *Tabla entidad -relación roles*

Nombre atributo	Tipo	Clave	Descripción
id	integer		Identificador de usuario
dni	Varchar	Primary Key	Identificador personal de acceso
nombre	Varchar		Nombre del cliente
apellidos1	Varchar		Primer apellido del cliente
apellido2	Varchar		Segundo apellido del cliente del cliente
fechaNacimiento	Date		Fecha de nacimiento del cliente
Email:	varchar		Correo electrónico del cliente

Tabla 5*: Tabla entidad -relación Cliente*

Nombre atributo	Tipo		Clave	Descripción
id	integer			Identificador de usuario (personal)
dni	Varchar		Primary Key	Identificador personal de acceso
Nombre	Varchar			Nombre del empleado
apellidos1	Varchar			Primer apellido del empleado
apellido2	varchar			Segundo apellido del empleado
Email:	varchar			Correo electrónico del empleado

Tabla 6: *Tabla entidad -relación Empleado*

Nombre Atributo	Tipo		Clave	Descripción
id	integer		Primary Key	Identificador de la cita
dni	Varchar			Identificador personal de cliente
fechainicio	timestamp			Fecha de inicio del tratamiento
fechafin	timestamp			Fecha final del tratamiento
url	varchar			Dirección url para acceder

Tabla 7: *Tabla entidad -relación Citas*

Nombre Atributo	Tipo		Clave	Descripción
id	integer		Primary Key	Identificador de la observación
dni	Varchar			Identificador personal de cliente
fecha	date			Fecha de la observación
nota	Varchar			Información que el especialista considera relevante

Tabla 8: *Tabla entidad -relación Observación*

Después de examinar cada una de las tablas que conforman la base de datos del prototipo de sistema a desarrollar, es importante comprender cómo estas se interrelacionan y se conectan entre sí. Al entender la manera en que estas tablas se relacionan, se puede obtener una comprensión más profunda y completa de la estructura de la base de datos, lo que permitirá realizar consultas más precisas y efectivas.

Como se observa, en la mayoría de las tablas, se ha elegido utilizar el id como clave primaria debido a que, en algunos casos, como en la tabla de citas, el DNI no puede ser utilizado como la clave primaria ya que un mismo cliente puede tener varias citas con distintos profesionales. Al utilizar el id como clave primaria, se facilita la gestión de estos casos.

Sin embargo, en otras tablas como la tabla de clientes y empleados, se ha optado por utilizar el DNI como clave primaria, debido a que, en este caso, cada cliente y/o empleado tiene un DNI único que lo identifica y no hay riesgo de que se repita.

La inclusión de una tabla de roles es un componente crítico en cualquier base de datos que requiera diferentes niveles de acceso y permisos para sus usuarios. En este caso, la tabla de roles registrará el rol de cada usuario junto con su id_usuario correspondiente, lo que facilitará la búsqueda del rol de cada usuario y, por ende, su acceso a la plataforma.

La tabla de roles permitirá asignar diferentes niveles de permisos a los usuarios, determinando el tipo de acceso que tendrán a la información de la base de datos. Por ejemplo, los administradores tendrán acceso completo a toda la información, mientras que los profesionales de la salud podrán acceder solo a los datos de sus pacientes y las citas agendadas. Por otro lado, los pacientes podrán acceder únicamente a su propia información personal y de citas.

Además, al tener la información de los roles en una tabla separada, se simplificará la tarea de asignar permisos y controlar el acceso a la información. La tabla de roles también permitirá añadir nuevos roles y modificar los permisos existentes según sea necesario, lo que proporcionará una mayor flexibilidad y adaptabilidad para asegurar que el acceso a la información sea siempre adecuado y seguro.

Es de hacer notar que las tablas presentadas en el modelo entidad-relación cumplen rigurosamente con las condiciones de la Primera Forma Normal (1NF), la Segunda Forma Normal (2NF) y la Tercera Forma Normal (3NF). Este cumplimiento se traduce en una estructura de base de datos sólida y eficiente, libre de dependencias transitivas y otras anomalías que podrían afectar la integridad y la eficacia del sistema.

3.2. **Diagrama de Clases**

En esta sección se describe la distribución de las clases en la aplicación, utilizando el patrón Modelo-Vista-Controlador (MVC), condicionado en cierta medida por la propia estructura que determina el framework seleccionado, en este caso, Flask, pues este último por su arquitectura flexible encaja naturalmente con el patrón MVC, aunque con una flexibilidad que permite una organización más personalizada de la lógica de la aplicación.

Este patrón de diseño divide la aplicación en tres componentes principales: el modelo, la vista y el controlador, lo que permite separar la lógica de la aplicación en capas y mejorar la organización y mantenimiento del código.

El patrón MVC establece que la vista es la capa que se encarga de mostrar la información al usuario, mientras que el controlador maneja las acciones del usuario y el modelo se encarga de manejar los datos y la lógica de la aplicación. En este caso, se ha implementado una separación total entre las vistas, los controladores y los modelos, que están organizados en paquetes específicos para facilitar su visualización y gestión.

Por lo que se muestran los diferentes paquetes creados para las vistas, los controladores y los modelos, y cómo se distribuyen en la estructura de la aplicación. Esto permite visualizar de manera clara y ordenada cómo se ha implementado el patrón MVC en la aplicación.

La utilización del patrón MVC en esta aplicación permite separar la lógica de la aplicación en capas bien definidas y organizadas, lo que mejora la eficiencia y el mantenimiento del código. La distribución de las clases en paquetes específicos permite visualizar de manera clara y organizada la estructura de la aplicación.

En la Ilustración 6, se pueden observar los componentes del modelo, específicamente los *Data Access Object* (DAO). Anteriormente, se pudo observar el modelo de dominio, que mostraba las entidades que existían en el sistema. En este momento, se puede ver la relación entre estas entidades y cómo se relacionan con la base de datos a través de los DAO.

Los diferentes DAO que se han creado para cada entidad en la base de datos:
EmpleadoDAO, ClienteDAO, CitaDAO y ObservaciónDAO. Estos DAO manejan la iteración entre la aplicación y la base de datos a través de su respectivo DAOJDBC.

Los DAO son una parte fundamental del modelo en la arquitectura MVC, ya que permiten separar la lógica de la aplicación de la lógica de acceso a la base de datos. Cada DAO es responsable de manejar una entidad específica en la base de datos y proporciona métodos para insertar, actualizar, eliminar y consultar los datos de la entidad.

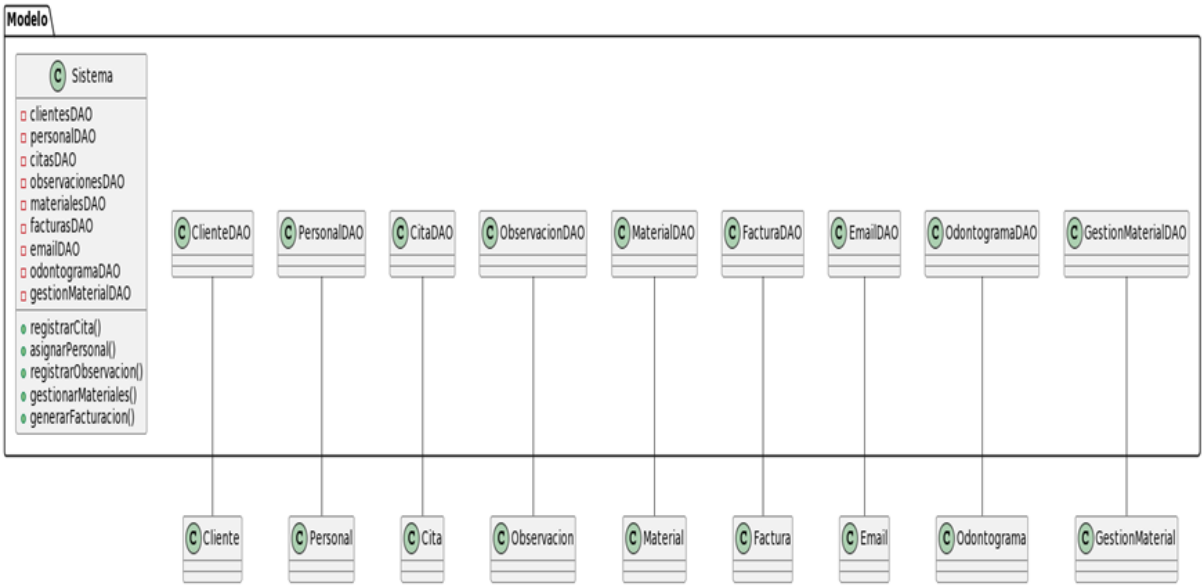


Ilustración 6: *Diagrama MVC Modelo de clases*

Ahora bien, siguiendo con los modelos, se ha utilizado un paquete llamado "vistas" para mostrar los resultados al usuario. En la Figura 7, se muestra algunos ejemplos de las vistas correspondientes al controlador "RegistroController", que se encarga de la creación de citas y usuarios. Sin embargo, no se han incluido todas las vistas en la ilustración.

Las vistas son una parte importante de cualquier aplicación web, ya que son la interfaz que los usuarios ven y con la que interactúan. En este proyecto en particular, el paquete "vistas" parece estar organizado en función de los controladores correspondientes.

Esto significa que cada controlador tiene su propio conjunto de vistas asociadas, que se utilizan para mostrar los resultados al usuario.

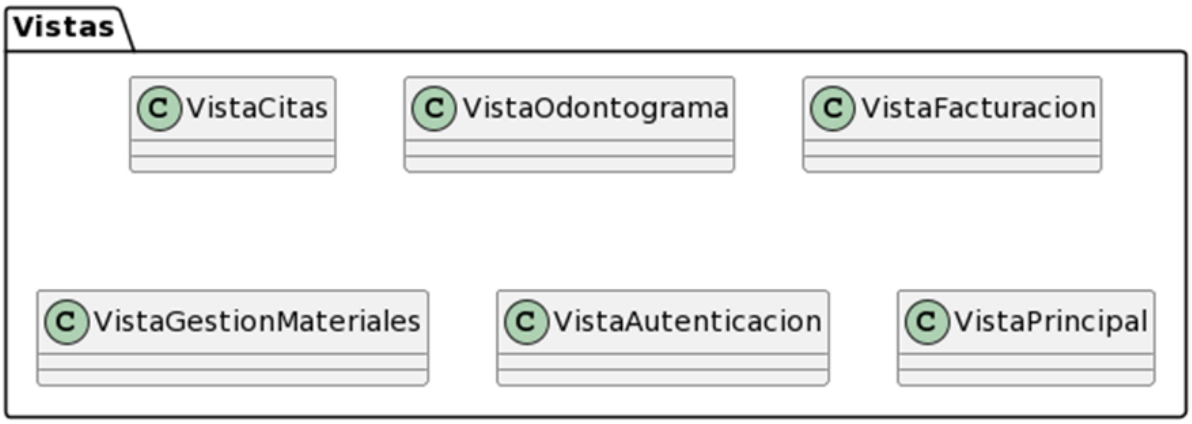


Ilustración 7: *Vistas, diagrama de clases*

El controlador "RegistroController" es responsable de la creación de citas y usuarios, por lo que sus vistas correspondientes estarán diseñadas para mostrar formularios de registro y

confirmación de registro. Estas vistas pueden incluir campos para ingresar información como nombres, correos electrónicos, fechas y otros datos relevantes para la creación de citas y usuarios.

Es importante destacar que la Ilustración 7, solo muestra algunos ejemplos de las vistas correspondientes al controlador "RegisroController". Es posible que existan otros controladores en el paquete "vistas" que tengan sus propias vistas y funcionalidades específicas.

En cuanto al paquete controlador es uno de los tres componentes principales del patrón Modelo - Vista - Controlador (MVC), y se encarga de manejar las acciones del usuario en la aplicación. En este caso, el paquete controlador está compuesto por los cinco controladores que forman parte del sistema, Ilustración 8.

Cada uno de estos controladores es responsable de manejar una funcionalidad específica de la aplicación. Los cinco controladores que conforman el paquete controlador son: EmpleadoController, ClienteController, AdminController, LogController y RegistroController.

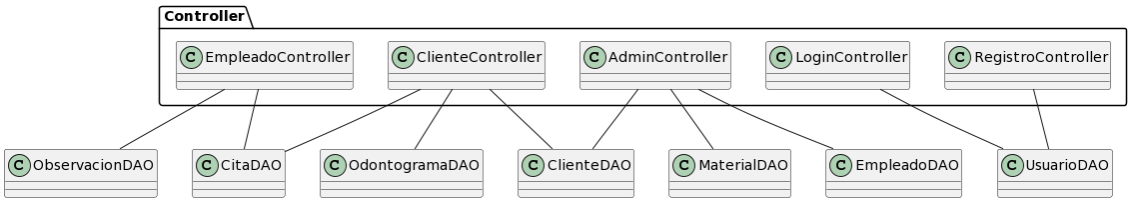


Ilustración 8: *Paquete controlador, diagrama de clases*

El EmpleadoController se encarga de manejar las acciones relacionadas con el personal de la clínica, como la creación, actualización y eliminación de datos del personal.

El ClienteController, por su parte, maneja las acciones relacionadas con los clientes de la clínica, como la creación, actualización y eliminación de datos de los clientes, y la asignación de citas.

El AdminController es responsable de recibir las solicitudes del usuario y comunicarse con el modelo para realizar las operaciones necesarias, como la creación, modificación o eliminación de entidades en la base de datos. Además, también se encarga de validar las solicitudes del usuario y de generar respuestas adecuadas a las mismas.

El RegistroController es una clase del paquete controlador en la arquitectura Modelo-Vista-Controlador (MVC) que se encarga de manejar las solicitudes de registro de nuevos usuarios en el sistema. Es responsable de procesar los datos enviados por el usuario a través de la vista

de registro y enviarlos al modelo correspondiente para su validación y almacenamiento en la base de datos.

Entre sus responsabilidades, se encuentran verificar que los datos ingresados por el usuario sean válidos y no estén duplicados, crear una nueva instancia de usuario en el modelo y actualizar la vista correspondiente para mostrar al usuario el resultado del proceso de registro.

El RegistroController también puede ser responsable de manejar errores que puedan ocurrir durante el proceso de registro, como la falta de información requerida o la entrada de información incorrecta. Además, puede ser necesario implementar medidas de seguridad para proteger la información sensible del usuario, como la encriptación de contraseñas y la verificación de la identidad del usuario.

Finalmente, el LogoController se encarga de manejar el inicio de sesión de los usuarios en la aplicación. Este controlador se encarga de recibir las credenciales de inicio de sesión, verificarlas y permitir el acceso a la aplicación si las credenciales son correctas. En caso contrario, se mostrará un mensaje de error al usuario indicando que las credenciales son inválidas.

Además de manejar el inicio de sesión, también se encarga de redirigir a los usuarios a la página de inicio correspondiente según su rol de usuario. Por ejemplo, si el usuario es un administrador, se redirigirá a la página de inicio del administrador, mientras que, si es un cliente, se redirigirá a la página de inicio del cliente.

El diagrama de clases con todas sus relaciones se observa en la Figura 9.

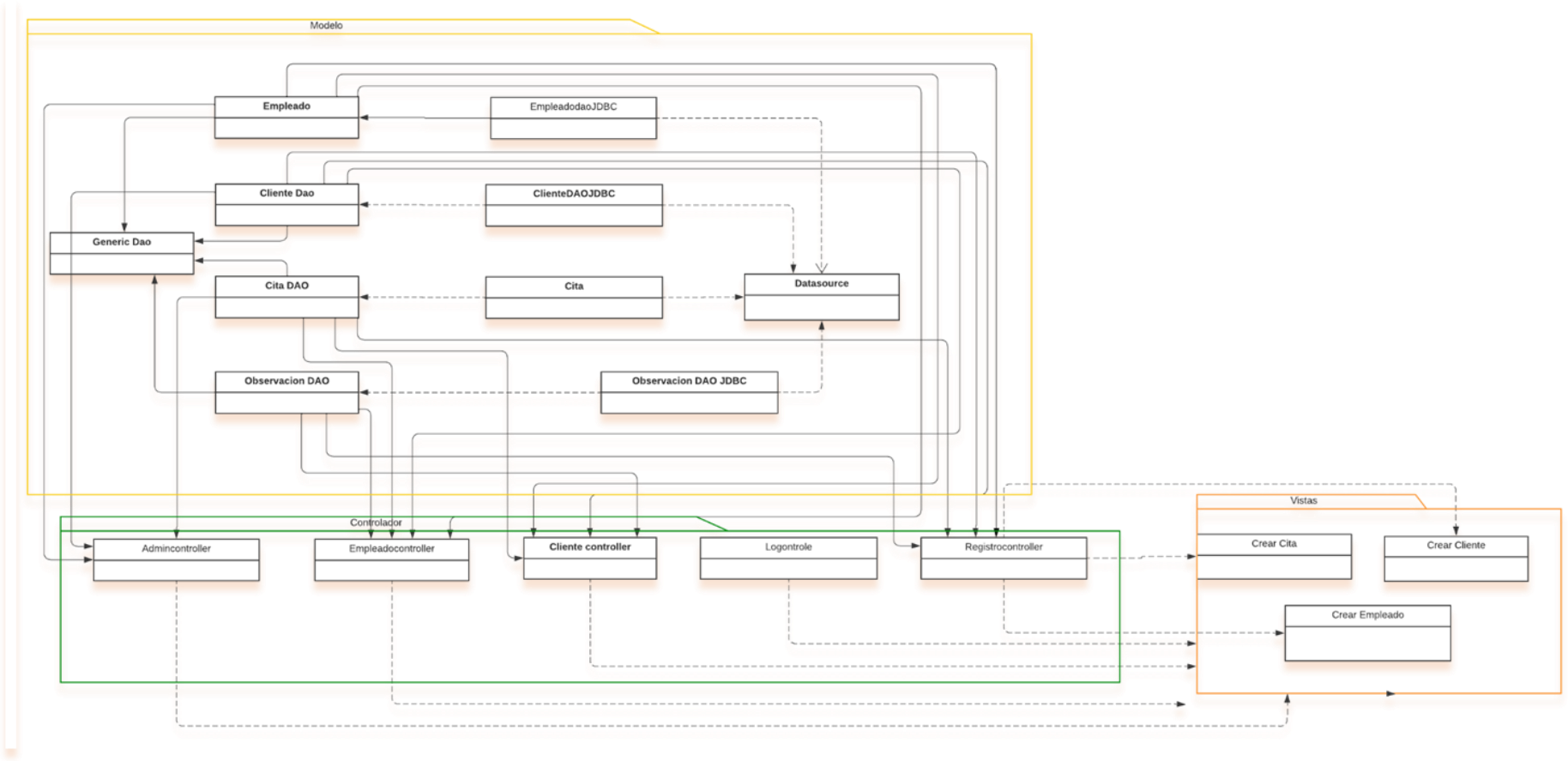


Ilustración 9: *Diagrama de clases*

3.3. Diagrama de secuencia

Los diagramas de secuencia son una herramienta importante en el modelado y documentación de procesos complejos en aplicaciones que utilizan el patrón MVC; este tipo de diagrama se enfoca en mostrar cómo los objetos o componentes interactúan entre sí y cómo se comunican para realizar una serie de acciones o funciones en un orden secuencial.

3.3.1. Diagrama de secuencia: inicio de sesión

En el siguiente diagrama se representan las vistas *Login* e Inicio, así como el controlador de inicio y autenticación de credenciales. El proceso de autenticación comienza cuando el usuario, ya sea cliente, personal o administrador, ingresa sus credenciales, es decir, su número de identificación personal y su contraseña.

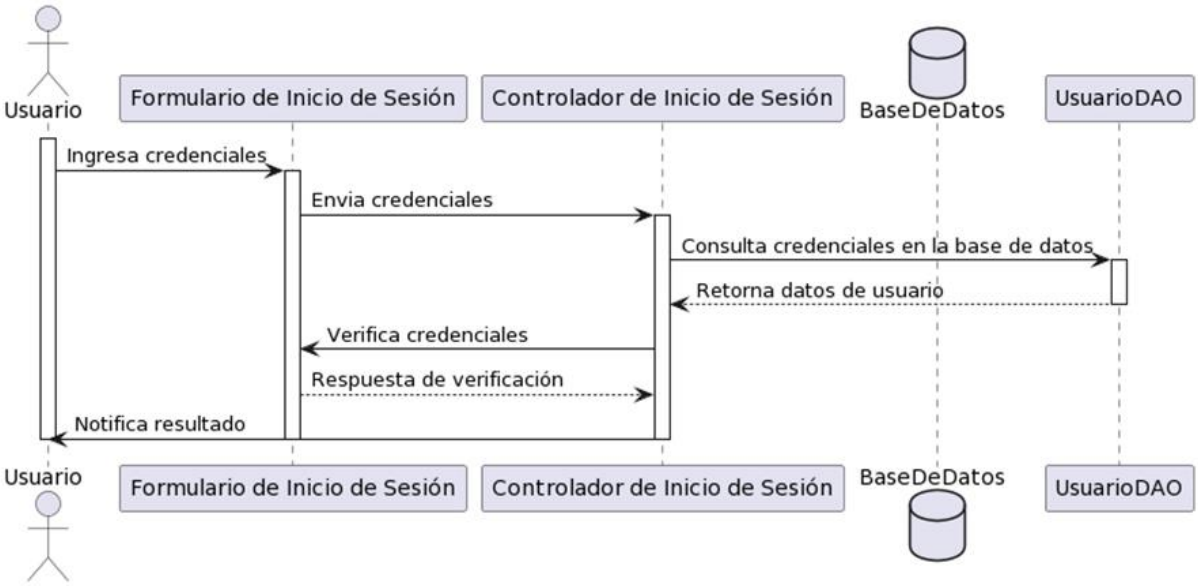


Ilustración 10: *Diagrama de secuencia iniciar sesión*

Antes de enviar el formulario, se valida que los datos ingresados sean correctos. Si las credenciales no son válidas, se mostrará un mensaje de error indicando que el usuario no está registrado en el sistema o que la contraseña es incorrecta.

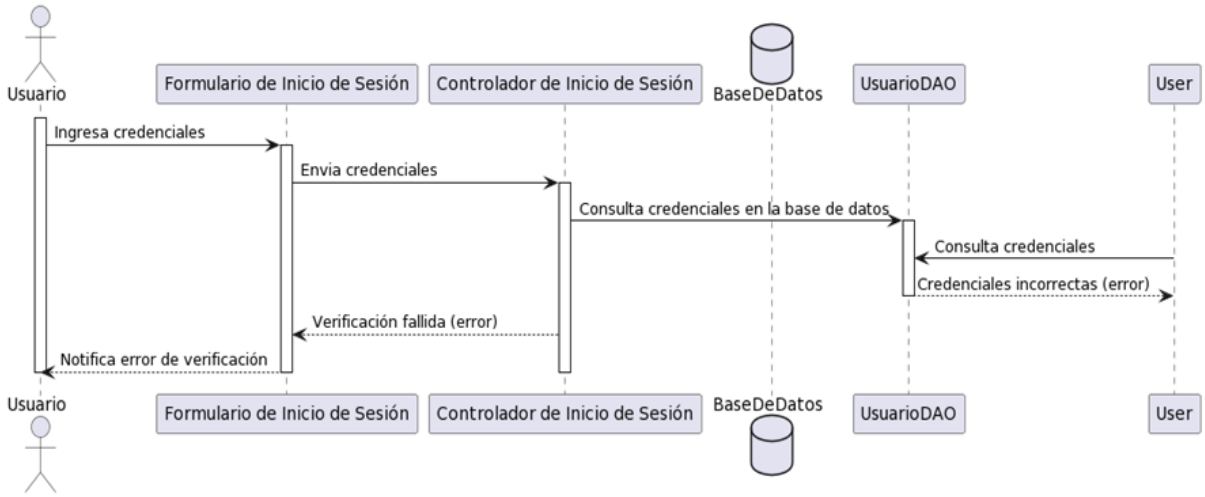


Ilustración 11: *Diagrama de secuencia iniciar sesión, error en credenciales*

En caso contrario, el usuario tendrá acceso a su página de inicio personal sin ningún problema. Cabe mencionar que, para garantizar la seguridad de los datos, Spring Security utiliza una base de datos para almacenar la información de autenticación y autorización de los usuarios.

3.3.2. Diagrama de secuencia: añadir cita

Se representa la funcionalidad de añadir citas, la cual está disponible tanto para el cliente como para el administrador. Por esta razón, se ha incluido al actor “Usuario” para representar esta función común. Ilustración 12.

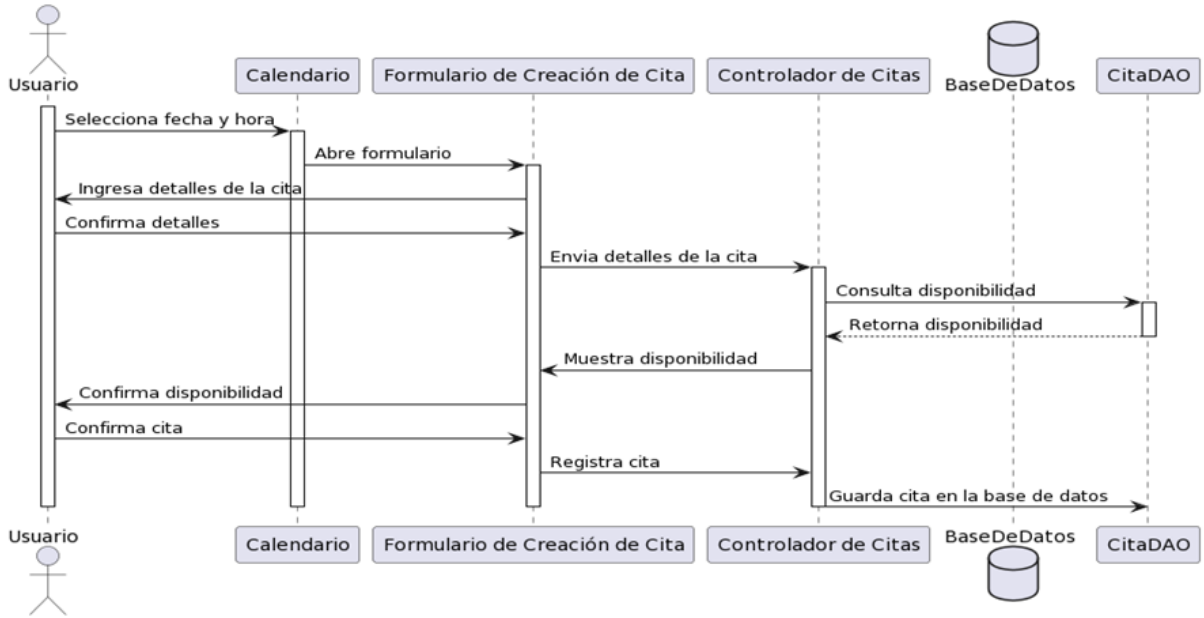


Ilustración 12*:* *Diagrama de secuencia añadir cita*

Para crear una cita, el usuario debe completar un formulario en el que se solicitan ciertos datos. En primer lugar, debe proporcionar su número de identificación personal.
Posteriormente, debe seleccionar la fecha y hora deseada para la cita.

Es importante destacar que cuando el usuario elige una fecha, se lleva a cabo una validación para verificar si esta cita se encuentra disponible en la agenda. Si la fecha elegida ya ha sido reservada, se resalta en color rojo en el calendario para indicar que no está disponible.

El usuario debe entonces seleccionar una fecha diferente que se encuentre libre para programar su cita. En caso contrario, si la fecha elegida está disponible, la cita se programa correctamente y el usuario recibe una confirmación de su cita por email.

3.4. Diseño de la interfaz

En esta sección del diseño de la interfaz de la aplicación, se presentan una serie de bocetos del prototipo. El objetivo de esta sección es mostrar cómo se realiza la transición entre las distintas vistas de la aplicación. Para ello, se utilizará un diagrama de estados que se centrará en las vistas correspondientes al administrador. Ilustración 13

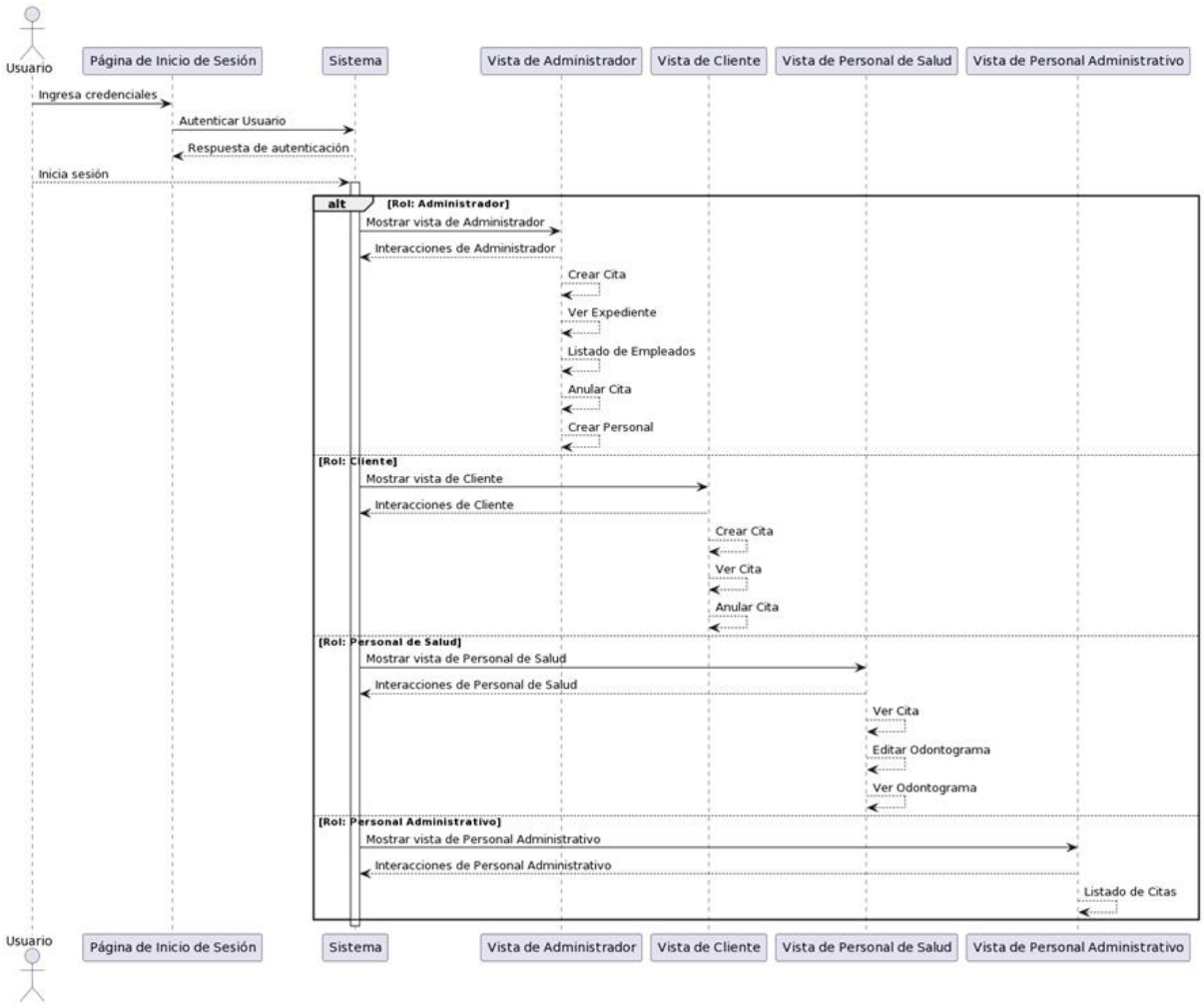


Ilustración 13: *Diagrama de estados de prototipo de página web clínica dental*

El diagrama de estados mostrará de manera clara y concisa cómo el usuario pasa de una vista a otra en función de la acción que realiza en la aplicación. Se presentarán algunas de las vistas disponibles, siendo que el personal puede acceder al sistema a través del rol que desempeñe: sea el administrador o el empleado tanto administrativo como el especialista en salud bucal, lo que permitirá una mejor comprensión del flujo de la aplicación. Además, se destacan las acciones más importantes que puede realizar el administrador, como la creación de citas o la edición de datos de los empleados.

Una vez establecido el diagrama, se dispone a presentar los modelos de la página principal, de los datos de registro del cliente, solicitud de cita, entre otras.

En la Ilustración 14 se muestra la página de inicio a la que tienen acceso los usuarios registrados. Se puede observar una breve descripción de lo que la aplicación ofrece y cuáles son sus principales objetivos y misión. También contiene la solicitud de ingreso de los datos para los usuarios ya registrados que les permite iniciar sesión.

Ilustración 14: *Interfaz de inicio de usuario registrado*

En la página de solicitar cita, Ilustración 15, se presenta un formulario en el que es necesario ingresar los datos solicitados como la fecha y la hora.

Ilustración 15: *Interfaz de inicio de solicitar cita*

4. IMPLEMENTACIÓN

4.1. Arquitectura

En este apartado la descripción de la arquitectura está basada en cliente/servidor. El servidor estaría compuesto por Flask y MySQL, mientras que el cliente sería cualquier usuario que acceda a la aplicación web mediante un navegador web, Ilustración 16.

En el lado del servidor, Flask actúa como el *framework* de aplicación web y gestiona las solicitudes HTTP enviadas por los clientes. Flask es un *framework* de Python que proporciona herramientas para crear aplicaciones web, desde enrutamiento hasta la conexión con bases de datos.

Por su lado, Jinja2 es la herramienta de plantillas utilizada para crear las páginas HTML y renderizar los datos de la base de datos. Por último, MySQL sería la base de datos utilizada para almacenar la información de la aplicación.

En cuanto a la comunicación entre el cliente y el servidor, se utiliza el protocolo HTTP para transferir la información entre ambos. El cliente envía solicitudes HTTP al servidor, que responde con las páginas HTML correspondientes.

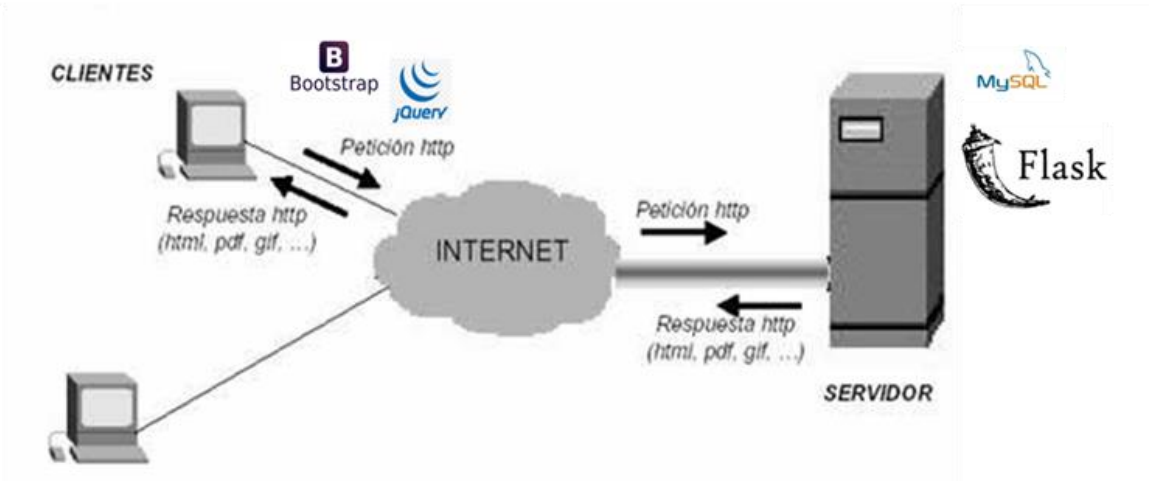


Ilustración 16*: Diagrama de arquitectura MVC*

4.2. Detalles de la implementación

En esta sección se proporciona una visión un poco más detallada de cómo se ha implementado la aplicación, destacando algunos aspectos claves de la parte de programación y uso de base de datos, así como la comunicación entre el cliente y el servidor.

4.2.1. Modelo de framework

El modelo presentado en la Ilustración 17, pertenece al modelo que se utiliza en el *framework* de Python llamado Flask.

En el caso específico de Flask, se utiliza un ORM (Object Relational Mapping) llamado SQLAlchemy para manejar la comunicación con la base de datos, que es este caso muestra el modelo Empleado, siendo esta una clase que hereda de la clase db_Model de SQLAlchemy, lo que indica que esta clase representa una tabla en la base de datos.

Además, se utiliza la clase UserMixin de Flask-Login para agregar funcionalidades de autenticación y autorización de usuarios a través de los atributos “id” y “password_hash”.

```
class Empleado(db.Model, UserMixin):
    __tablename__ = 'empleados'

    ID = db.Column(INTEGER(255), nullable=False, unique=True)
    DNI = db.Column(VARCHAR(9), primary_key=True)
    Nick = db.Column(VARCHAR(128), nullable=False)
    Apellido1 = db.Column(VARCHAR(128), nullable=False)
    Apellido2 = db.Column(VARCHAR(128), nullable=False)
    Nombre = db.Column(VARCHAR(128), nullable=False)
    Roles = db.Column(VARCHAR(128), nullable=False, comment='roles separados por coma')
    email = db.Column(VARCHAR(128), nullable=True)
    password_hash = db.Column(VARCHAR(128), nullable=False)

    def __repr__(self):
        return '<Empleado {}>'.format(self.Nick)

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)

    def get_id(self):
        return "e" + str((self.ID))

    def get_reset_password_token(self, expires_in=600):
        return jwt.encode(
            {'reset_password': self.ID, 'exp': time() + expires_in},
            current_app.config['SECRET_KEY'], algorithm='HS256').decode('utf-8')

    @staticmethod
    def verify_reset_password_token(token):
        try:
            id = jwt.decode(token, current_app.config['SECRET_KEY'],
                algorithms=['HS256'])['reset_password']
        except:
            return
        return Empleado.query.filter_by(ID=id).first()
```

Ilustración 17: *Modelo de framework*

4.2.2. Función crear citas

Para almacenar los datos de las citas, se utiliza una base de datos. Para acceder y manipular la base de datos, se utiliza un sistema de gestión de bases de datos. Estos sistemas proporcionan una interfaz para crear, modificar y consultar la base de datos desde el servidor.

Una vez que se ha establecido la base de datos, se programa el código para crear una cita, Ilustración 17, en particular, la línea "try: db.session.add(cliente)" se utiliza para agregar un registro a la base de datos, "db.session" es un objeto que representa la sesión de la base de datos y "add(cliente)" se utiliza para agregar el registro del cliente a la base de datos.

Cabe destacar que la sentencia "try" es una construcción en Python que indica que se intentará agregar el registro, y si ocurre algún error, se manejará con la excepción correspondiente.
Finalmente, la línea "@bp.router(´/crearcitas…, se utiliza para establecer la ruta de la aplicación web que se utilizará para crear citas. "@bp.router" es un decorador en Flask.

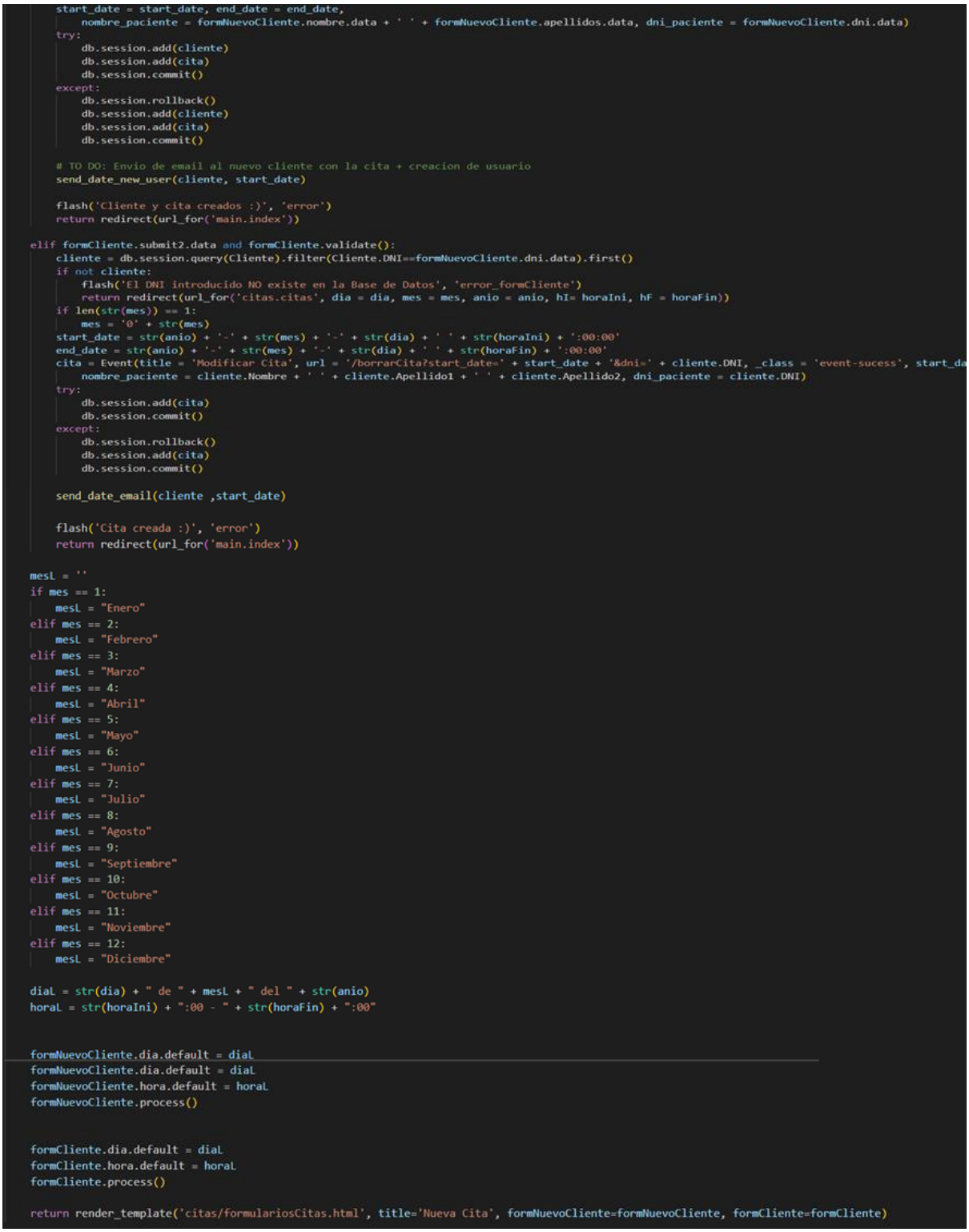
Esto último se usa para modificar el comportamiento de una función en particular, es decir, para definir las rutas o URLs de la aplicación web y para especificar los métodos HTTP que son aceptados por cada ruta, permitiendo así agregar funcionalidades a la aplicación sin tener que modificar la función subyacente.

```python
@bp.route('/crearCitas', methods=['GET', 'POST'])
@has_loged()
@has_permissions('dentista')
def citas():
    dia = request.args.get('dia',-1, type=int)
    mes = request.args.get('mes',-1, type=int)
    anio = request.args.get('anio',-1, type=int)
    horaIni = request.args.get('hI',-1, type=int)
    horaFin = request.args.get('hF',-1, type=int)

    # Comprobamos que los argumentos que pide esta funcion se han pasado correctamente (en caso contrario, redirigimos a la página principal)

    if dia == -1 or mes == -1 or anio == -1 or horaIni == -1 or horaFin == -1:
        flash('Ocurrió un error al intentar acceder a esta página.', 'error')
        return redirect(url_for('main.index'))

    formNuevoCliente = CitaNuevoCliente()
    formCliente = CitaCliente()

    if formNuevoCliente.submit1.data and formNuevoCliente.validate():
        cliente = db.session.query(Cliente).filter(Cliente.DNI==formNuevoCliente.dni.data).first()
        if cliente:
            flash('El DNI introducido ya existe en la Base de Datos', 'error_formNuevoCliente')
            return redirect(url_for('citas.citas', dia = dia, mes = mes, anio = anio, hI= horaIni, hF = horaFin))
        apellidos = formNuevoCliente.apellidos.data.split()
        cliente = Cliente(DNI=formNuevoCliente.dni.data, Nick = '', Apellido1 = apellidos[0], Apellido2 = apellidos[1] if len(apellidos) > 1 else '' ,
        Nombre = formNuevoCliente.nombre.data, password_hash = '', email = formNuevoCliente.email.data)
        if len(str(mes)) == 1:
            mes = '0' + str(mes)
        start_date = str(anio) + '-' + str(mes) + '-' + str(dia) + ' ' + str(horaIni) + ':00:00'
        end_date = str(anio) + '-' + str(mes) + '-' + str(dia) + ' ' + str(horaFin) + ':00:00'
        cita = Event(title = 'Modificar Cita', url = '/borrarCita?start_date=' + start_date + '&dni=' + cliente.DNI, _class = 'event-sucess',
```

Ilustración 18: *Función crear citas*

En este caso, la ruta "/crearcitas", y los métodos que se pueden utilizar para hacer solicitudes a esa ruta son "GET" y "POST".

"GET" se utiliza para solicitar información de la base de datos, mientras que "POST" se utiliza para enviar información a la base de datos.



```
        start_date = start_date, end_date = end_date,
            nombre_paciente = formNuevoCliente.nombre.data + ' ' + formNuevoCliente.apellidos.data, dni_paciente = formNuevoCliente.dni.data)
        try:
            db.session.add(cliente)
            db.session.add(cita)
            db.session.commit()
        except:
            db.session.rollback()
            db.session.add(cliente)
            db.session.add(cita)
            db.session.commit()

        # TO DO: Envio de email al nuevo cliente con la cita + creacion de usuario
        send_date_new_user(cliente, start_date)

        flash('Cliente y cita creados :)', 'error')
        return redirect(url_for('main.index'))

    elif formCliente.submit2.data and formCliente.validate():
        cliente = db.session.query(Cliente).filter(Cliente.DNI==formNuevoCliente.dni.data).first()
        if not cliente:
            flash('El DNI introducido NO existe en la Base de Datos', 'error_formCliente')
            return redirect(url_for('citas.citas', dia = dia, mes = mes, anio = anio, hI= horaIni, hF = horaFin))
        if len(str(mes)) == 1:
            mes = '0' + str(mes)
        start_date = str(anio) + '-' + str(mes) + '-' + str(dia) + ' ' + str(horaIni) + ':00:00'
        end_date = str(anio) + '-' + str(mes) + '-' + str(dia) + ' ' + str(horaFin) + ':00:00'
        cita = Event(title = 'Modificar Cita', url = '/borrarCita?start_date=' + start_date + '&dni=' + cliente.DNI, _class = 'event-sucess', start_da
            nombre_paciente = cliente.Nombre + ' ' + cliente.Apellido1 + ' ' + cliente.Apellido2, dni_paciente = cliente.DNI)
        try:
            db.session.add(cita)
            db.session.commit()
        except:
            db.session.rollback()
            db.session.add(cita)
            db.session.commit()

        send_date_email(cliente ,start_date)

        flash('Cita creada :)', 'error')
        return redirect(url_for('main.index'))

    mesL = ''
    if mes == 1:
        mesL = "Enero"
    elif mes == 2:
        mesL = "Febrero"
    elif mes == 3:
        mesL = "Marzo"
    elif mes == 4:
        mesL = "Abril"
    elif mes == 5:
        mesL = "Mayo"
    elif mes == 6:
        mesL = "Junio"
    elif mes == 7:
        mesL = "Julio"
    elif mes == 8:
        mesL = "Agosto"
    elif mes == 9:
        mesL = "Septiembre"
    elif mes == 10:
        mesL = "Octubre"
    elif mes == 11:
        mesL = "Noviembre"
    elif mes == 12:
        mesL = "Diciembre"

    diaL = str(dia) + " de " + mesL + " del " + str(anio)
    horaL = str(horaIni) + ":00 - " + str(horaFin) + ":00"


    formNuevoCliente.dia.default = diaL
    formNuevoCliente.dia.default = diaL
    formNuevoCliente.hora.default = horaL
    formNuevoCliente.process()


    formCliente.dia.default = diaL
    formCliente.hora.default = horaL
    formCliente.process()

    return render_template('citas/formulariosCitas.html', title='Nueva Cita', formNuevoCliente=formNuevoCliente, formCliente=formCliente)
```

Ilustración 19: *Creación cita y envío confirmación a email usuario*

En la Ilustración 19, se observa como ocurre el proceso de generar una respuesta HTTP que será enviada al cliente a través de la web (renderizar) a partir de los datos y plantillas que tiene el servidor.

Se muestra como a través del código return render_template('citas/formularios.html', title='Nueva cita', formnuevocliente=formnuevocliente, formcliente=formcliente) la función render_template es una función proporcionada por el framework Flask que toma una plantilla en formato HTML (en este caso, 'citas/formularioscitas.html') y rellena los espacios en blanco en la plantilla con los datos que se le pasan como parámetros.

Los parámetros que se pasan a render_template son información que se mostrará en la página web, como el título de la página ('Nueva cita') y los formularios (formnuevocliente y formcliente), que serán llenados por el usuario para crear una nueva cita.

Finalmente, la respuesta HTTP generada por render_template se devuelve como resultado de la función mediante la palabra clave return. Así, cuando el usuario accede a la URL correspondiente a esta vista en el servidor, se le mostrará una página web dinámica con los datos y formularios necesarios para crear una nueva cita.

4.2.3. Función login

En la Ilustración 20, se muestra la codificación del acceso y cierre de sesión del usuario a través de una función de Flask que se encarga de manejar la autenticación de usuarios.

La función render_template se utiliza para renderizar la plantilla HTML que muestra un formulario de inicio de sesión. En este caso, la plantilla es 'auth/login.html'.
Los valores que se pasan como argumentos en la función render_template se utilizan para personalizar la plantilla, en este caso se le está pasando el título de la página y un objeto del formulario denominado 'form'.

Para el cierre de sesión la función pop() se utiliza para eliminar una clave del diccionario de la sesión del usuario. La clave que se está eliminando es obtenida a través de la función os.environ.get(), que se utiliza para acceder a una variable de entorno llamada "auth_token_key". Si no existe esa variable de entorno, se utilizará el valor por defecto de False. El valor de la clave eliminada se establece en None.

```python
# Login y redirecion a la pagina principal
@bp.route('/login', methods=['GET', 'POST'])
def login():
    # time.sleep( delay )
    session.permanent = True

    if current_user.is_authenticated:
        return redirect(url_for('main.index'))

    if is_logged_in():
        email_ = get_user_info()['email']
        print(email_)
        user = Empleado.query.filter_by(email=email_).first()
        if user is None:
            user = Cliente.query.filter_by(email=email_).first()
        if not user is None:
            login_user(user, remember=False)
            return redirect(url_for('main.index'))
        else:
            session.pop(os.environ.get("AUTH_TOKEN_KEY", default=False), None)
            session.pop(os.environ.get("AUTH_STATE_KEY", default=False), None)

            flash('No se puede conectar al servidor, usuario incorrecto', 'error_logpass')
            return redirect(url_for('auth.login'))

    form = LoginForm()
    if form.validate_on_submit():
        user = Empleado.query.filter_by(Nick=form.username.data).first()
        if user is None:
            user = Cliente.query.filter_by(Nick=form.username.data).first()
        if not user is None and user.check_password(form.password.data):
            login_user(user, remember=form.remember_me.data)
            return redirect(url_for('main.index'))
        else:
            flash('No se puede conectar al servidor, compruebe su usuario y contraseña', 'error_logpass')
            return redirect(url_for('auth.login'))
    return render_template('auth/login.html', title='Iniciar sesion', form=form)
```

Ilustración 20: *Codificación del acceso y cierre de sesión del usuario*

5. PRUEBAS

Las pruebas de sistemas son una parte fundamental del proceso de desarrollo de software, y en el caso de la gestión de citas para una clínica odontológica, no es la excepción. En este proyecto, las pruebas de sistemas se realizaron después de cada Sprint para validar las historias de usuario implementadas y asegurar que cumplían con los criterios de aceptación establecidos.

En apartados anteriores se definieron las historias de usuario, que son descripciones detalladas de las funcionalidades que debe tener el software. Cada historia de usuario tiene sus criterios de aceptación, que son los requisitos que debe cumplir para considerarse implementada de manera satisfactoria. Estos criterios de aceptación son la base para las pruebas de sistemas, ya que son la guía que se sigue para comprobar que cada historia de usuario funciona como se espera.

La primera historia de usuario definida fue “Iniciar sesión”, que permite al usuario acceder a la aplicación. Para esta historia, se establecieron tres criterios de aceptación: el inicio de sesión exitoso, el inicio de sesión fallido y la expiración de sesión. En la prueba correspondiente, se comprobó que el usuario pudiera acceder a la aplicación de manera correcta con los datos requeridos, que el sistema respondiera adecuadamente ante un inicio de sesión fallido y que la sesión expirará correctamente.

Otra historia de usuario fue “Dar de alta cliente”, que permite al usuario crear nuevos clientes en el sistema. Para esta historia, se establecieron dos criterios de aceptación: la creación exitosa y la creación fallida. En la prueba correspondiente, se comprobó que el usuario pudiera crear clientes correctamente al rellenar el formulario y que el sistema respondiera adecuadamente ante errores como datos faltantes, datos incorrectos o la creación de usuarios ya existentes.

La historia de usuario “Cambiar contraseña” permite al usuario modificar su contraseña. Para esta historia, se establecieron dos criterios de aceptación: el cambio exitoso y el cambio fallido. En la prueba correspondiente, se comprobó que el usuario pudiera cambiar su contraseña y que el sistema respondía adecuadamente ante un cambio fallido.

La historia de usuario “Añadir cita” permite al usuario agregar nuevas citas al sistema. Para esta historia, se estableció un único criterio de aceptación: la creación exitosa. En la prueba correspondiente, se comprobó que el usuario pudiera agregar citas correctamente al completar los datos del formulario.

Por último, en la historia de usuario “Añadir observación”, se establecieron dos criterios de aceptación. El primer criterio es la creación exitosa, lo que significa que el usuario completa correctamente los datos del formulario. El segundo criterio se refiere a una creación fallida, en la que el usuario debe rellenar de nuevo el formulario, ya que los datos proporcionados no cumplen con los requerimientos.

Cada una de estas historias de usuario y sus criterios de aceptación fueron probadas exhaustivamente durante el proceso de pruebas de sistemas. En cada Sprint, se realizaron estas pruebas para validar las funcionalidades implementadas y asegurarse de que el software funcionará correctamente, de esta manera se consigue detectar posibles errores o problemas en la aplicación, los cuales pueden ser corregidos de manera oportuna antes de la liberación de la versión final. Además, se asegura la calidad del software y se mejora la experiencia del usuario.

6. CONCLUSIONES

En el presente proyecto, se han cumplido satisfactoriamente todos los objetivos planteados desde el inicio. En primer lugar, se llevó a cabo un estudio detallado de las necesidades y soluciones existentes en el mercado en torno al diseño e implementación de una aplicación web que cumpla con los requerimientos establecidos por el cliente.

Este análisis condujo a la elección de tecnologías avanzadas como Python, Flask, Pyramid, Django y JQuery para llevar adelante la implementación, en marcado contraste con el uso previo de Java y Spring framework en el proyecto anterior. Este cambio en la tecnología no solo permitió un enfoque más moderno y ágil en el desarrollo, sino que también se adecuó mejor a las características y necesidades del nuevo proyecto.

A partir de este análisis, se eligió el patrón de diseño MVC para llevar a cabo la implementación del proyecto. Durante el proceso de desarrollo, se ha puesto en práctica un enfoque iterativo que incorporó algunas técnicas y elementos derivados de SCRUM lo cual ha permitido gestionar de manera eficiente el tiempo y recursos disponibles, asegurando así una entrega exitosa del proyecto.

Cabe destacar que la aplicación web a pesar de contemplar las diferentes historias de usuario tanto del proyecto base como las propuestas en el inicio del proyecto actual, no todas pudieron cumplirse a cabalidad, dejando incompleto algunas historias de usuario de la épica Facturación y stock, siendo las historias de usuario no desarrolladas a partir de la historia 22 hasta la 28, debido a una limitación de tiempo no estipulada al inicio del proyecto.

Una de las historias de usuario más críticas, se centró en la implementación de un sólido sistema de autenticación y seguridad en la aplicación web. El objetivo primordial fue asegurar que el acceso a la plataforma estuviera restringido únicamente a usuarios autorizados. Esta tarea se abordó mediante la creación de mecanismos de autenticación personalizados y la aplicación de estrategias de seguridad robustas en Flask, dado que se estableció un sistema de autorización que asigna diferentes niveles de acceso y permisos según los roles de usuario, tales como administradores, clientes y personal de la aplicación.

Otra historia de usuario importante se centró en la implementación de un sistema de citas para la atención al cliente. Esto se logró mediante la creación de un formulario de registro de citas que permite a los usuarios ingresar sus datos personales y la fecha y hora en la que desean ser atendidos. Además, se incorporó la validación de la disponibilidad de la fecha y hora seleccionadas mediante una verificación en la base de datos y su posterior visualización en la interfaz de usuario.

Además, otra de las historias de usuario se enfocó en la implementación de un sistema de seguimiento y gestión del estado de sus dientes de los clientes. Esto se logró mediante la creación de una sección específica dentro de la interfaz de usuario, en la cual los clientes pueden ver el estado de sus dientes y ver la actualización del mismo.

Este aspecto más distintivo es la creación del odontograma, que permite tanto al especialista de salud bucal como al cliente, poder llevar un registro detallado de las atenciones y cuidados realizados a cada uno de los dientes especificado.

La documentación del proyecto ha sido otro aspecto clave en su éxito, ya que se ha procurado dejar claro cómo se ha realizado cada una de las partes de este.

Se ha incluido la descripción detallada de los diferentes componentes, así como también los pasos necesarios para la instalación y configuración de la aplicación. Todo esto ha permitido facilitar su mantenimiento y modificación en un futuro.

6.1. Mejoras y trabajos futuros

En el futuro, se espera que la aplicación sea probada por pacientes que tienen citas en la clínica odontológica, con el objetivo de determinar los cambios que deben hacerse y la necesidad de introducir nuevas funcionalidades que sean pertinentes.

Teniendo en cuenta que cada vez más los servicios médicos están altamente informatizados, se espera que la funcionalidad de la página web sea más interactiva. Es decir, que pueda incluirse motivo de la consulta, para que estos sean enviados inmediatamente al odontólogo responsable u otro profesional de la salud calificado, para que se haga una evaluación de la necesidad de atención inmediata o a la espera oportuna de la cita.

La reciente pandemia sufrida a nivel mundial (COVID19) resaltó la importancia de tener acceso a atención médica de calidad a distancia, por lo que se busca en esos casos una retroalimentación instantánea.

Además, sería ventajoso hacer que el módulo de la agenda de eventos sea más dinámico, creando la opción de alertas para los eventos más cercanos, funcionando como un recordatorio. De esta manera, se evitarían faltas a las citas médicas por olvido involuntario del paciente.

Se espera que el desarrollo y mejora de esta plataforma no termine con la entrega de este proyecto, ya que tiene potencial para crecer según las necesidades y el público al que se destina.

7. Bibliografía

Canal, P. (2015). *¿Qué es el diseño centrado en el usuario?* IEBS: https://www.iebschool.com/blog/diseno-centrado-en-el-usuario-analitica-usabilidad/

Couto, C. (julio de 2018). Socioantropología de la salud bucal. Aproximaciones epistemológicas. las primeras civilizaciones y la herencia grecolatina. *ODOUS CIENTIFICA*, *19*(2), 23-34.

Derming, E. (1992). Quality, Productivity and Competitive Position. Massachusetts Institute of Technology.

DigitalReport. (2023). *Digital Report 2023: Global Over~view Report*. https://datareportal.com/reports/digital-2023-global-overview-report

IBM. (2021). InfoSphere Data Architect. https://www.ibm.com/docs/es/ida/9.1.2?topic=types-domain-models

Martinez, M. (septiembre de 2018). *ESTUDIO Y DESARROLLO DE UN PROTOTIPO DE APLICACIÓN WEB PARA LA GESTIÓN DE CITAS EN UNA CLÍNICA DENTAL*. CREA: Colección de Recursos Educativos Abierto: https://crea.ujaen.es/bitstream/10953.1/14260/1/Memoria_Martinez_Lopez_Maria.pdf

MVC Framework. (s.d.). *MVC Framework - Architecture*. MVC Framework - Architecture: https://www.tutorialspoint.com/mvc_framework/mvc_framework_architecture.htm

Python software foundation. (s/f). *Tutorial Python*. Python: https://docs.python.org/es/3/tutorial/

Ramos, C. (2021). APRENDE FLASK DE PRINCIPIANTE A EXPERTO EN 2021 CON PYTHON : GUIA COMPLETA DEL DESARROLLO DE APLICACIONES WEB CON PYTHON (Edicion En Español) . Em C. Ramos.

Scrum Manager Bok. (s.d.). *INVEST*. Scrum Manager Bok: https://www.scrummanager.com/bok/index.php/INVEST

Apéndice I. Manual de instalación del sistema

A continuación, vamos a explicar en el siguiente apéndice todos los pasos necesarios para la puesta en marcha del sistema que compone este trabajo fin de grado. Los pasos que se van a describir a continuación son para su aplicación en un sistema de Windows. Para otros sistemas los pasos a seguir son similares, pero suelen variar los archivos a descargar y los comandos a utilizar, además como Windows es el sistema operativo que utilizo, he decidido realizarlo en él.

Activación de la Virtualización en la BIOS

Antes de comenzar con la instalación de Vagrant, es esencial verificar y, si es necesario, activar la virtualización en la BIOS de tu ordenador. La virtualización es fundamental para que Vagrant funcione correctamente. Los pasos suelen variar según la marca y modelo de la placa base con la que estemos trabajando, pero en general, debes seguir estos pasos:

1. Reinicia el ordenador y presiona la tecla correspondiente (generalmente, F2, F12 o Supr) para acceder a la configuración de la BIOS durante el proceso de arranque.

2. Busca una opción relacionada con la virtualización, que a menudo se llama "Intel Virtualization Technology" o "AMD-V". Activa esta opción y guarda los cambios antes de salir de la BIOS.

3. Reinicia tu ordenador nuevamente.

Instalación de XAMPP con MySQL y la Importación de una Base de Datos SQL

XAMPP es un paquete que incluye un servidor web Apache, MySQL, PHP y otros componentes útiles para el desarrollo web. Sigue estos pasos para instalar XAMPP y configurar MySQL:

1. Descarga la última versión de XAMPP desde el sitio web oficial (https://www.apachefriends.org/index.html) y sigue las instrucciones de instalación para tu sistema operativo.

2. Inicia XAMPP y asegúrate de que el servicio MySQL esté en funcionamiento.

3. Abre un navegador web e ingresa "http://localhost/phpmyadmin" en la barra de direcciones.

4. Haz clic en "Nuevo" para crear una nueva base de datos y dale un nombre significativo.

5. Selecciona la base de datos recién creada y elige la opción "Importar". Sube el archivo SQL proporcionado para la correcta instalación de la estructura de la base de datos.

6. Hace falta crear un usuario de manera manual en la tabla de empleados y darle el rol “1” para que tenga permisos de administración

Instalación de VirtualBox

Vagrant requiere un hipervisor para crear máquinas virtuales. VirtualBox es una excelente opción y es compatible con Vagrant. Sigue estos pasos para instalar VirtualBox:

1. Descarga la última versión de VirtualBox desde el sitio web oficial (https://www.virtualbox.org/) y sigue las instrucciones de instalación para tu sistema operativo.

2. Durante la instalación, asegúrate de que la opción "Interfaz de usuario de VirtualBox" esté seleccionada para instalar la interfaz gráfica de VirtualBox.

3. Una vez instalado, inicia VirtualBox para asegurarte de que funcione correctamente.

Importación de una Caja de Vagrant (.box)

Una "caja" en Vagrant es una imagen de máquina virtual preconfigurada que se utiliza como base para crear instancias de máquinas virtuales. Puedes encontrar cajas de Vagrant en el sitio web oficial de Vagrant (https://app.vagrantup.com/boxes/search). Sigue estos pasos para importar una caja:

1. Abre una terminal o línea de comandos en tu sistema.

2. Ejecuta el siguiente comando para agregar una caja de Vagrant. Reemplaza "nombre_de_la_caja" con el nombre de la caja proporcionada en este trabajo:

vagrant box add nombre_de_la_caja

3. Vagrant importará la caja y la almacenará localmente para su uso posterior.

Copia del Archivo Zip con un Código Fuente

Finalmente, hay que copiar el archivo zip que contiene el código fuente de este proyecto al entorno de Vagrant. Una vez copiada y descomprimida, tenemos que desplazarnos hasta esta carpeta a través de una instancia de Windows Powershell. Una vez en la carpeta, tenemos que utilizar los siguientes comandos:

1. **Ejecutar el comando vagrant up (Esto arranca la máquina virtual)**
2. **Una vez finalice el comando, ejecutar el comando vagrant ssh para entrar a la máquina virtual**
3. **Una vez haya conectado, ir a la carpeta del proyecto: cd /vagrant/clinicDentalWEB/**
4. **Iniciar el entorno virtual: source venv/bin/activate**
5. **Arrancar flask: flask run --host=0.0.0.0**
6. **Para entrar en la web usa la URL: localhost:5000**

Apéndice II. Manual de Usuario

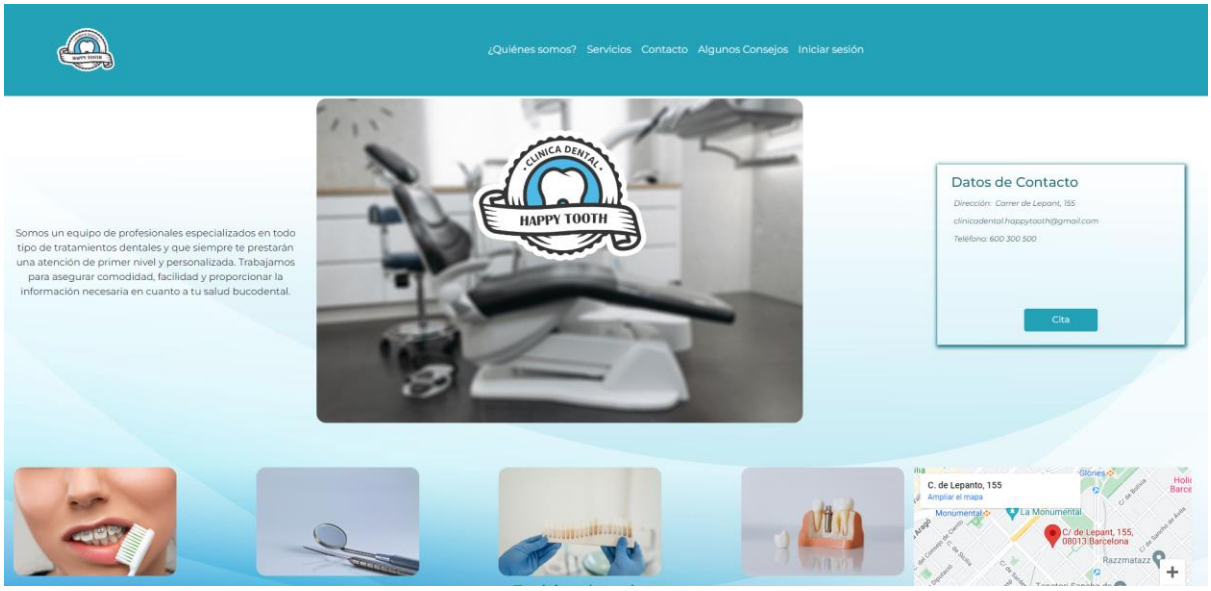
A continuación, pasamos a describir todos los detalles de la web que abarcamos en este trabajo fin de grado para un correcto funcionamiento de la misma

Acceso a la Plataforma

1. **Usuarios no Logueados:**

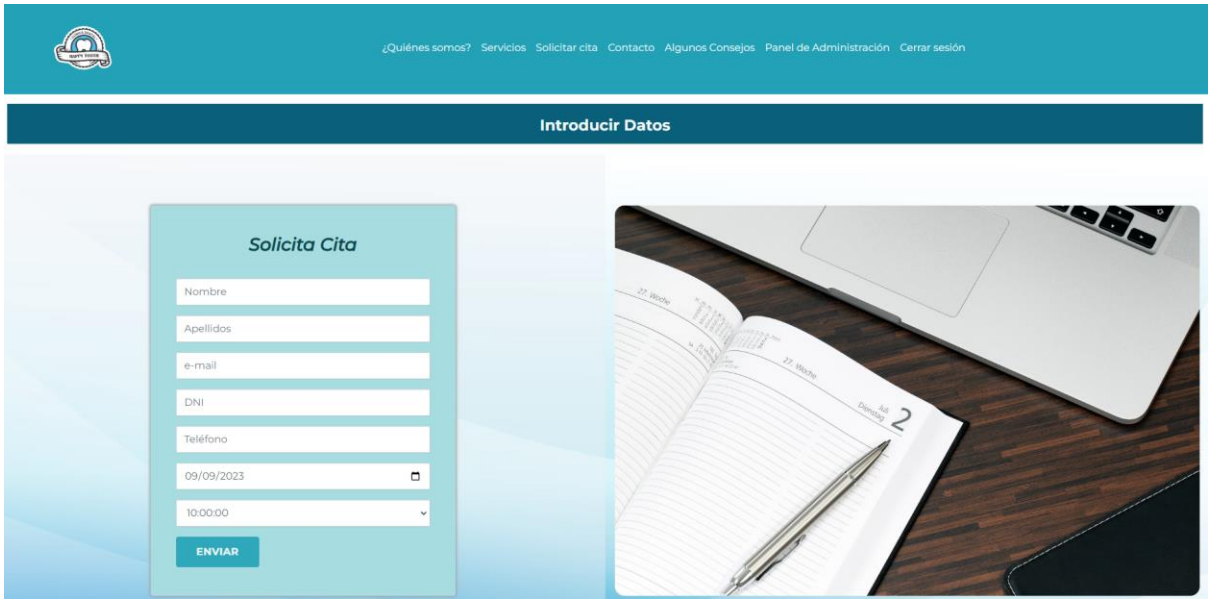
Cuando accedes a la plataforma sin iniciar sesión, verás la página de inicio que muestra el logotipo de la clínica dental, información de contacto y un mapa con la ubicación de la clínica. Además, tienes acceso a las siguientes opciones de navegación en la barra de navegación:

- ¿Quiénes Somos?: Información sobre la clínica y su equipo.
- Servicios: Detalles de los servicios dentales ofrecidos.
- Contacto: Formulario de contacto y detalles de ubicación.
- Algunos Consejos: Consejos de cuidado dental.
- Iniciar Sesión: Para acceder a funcionalidades avanzadas.

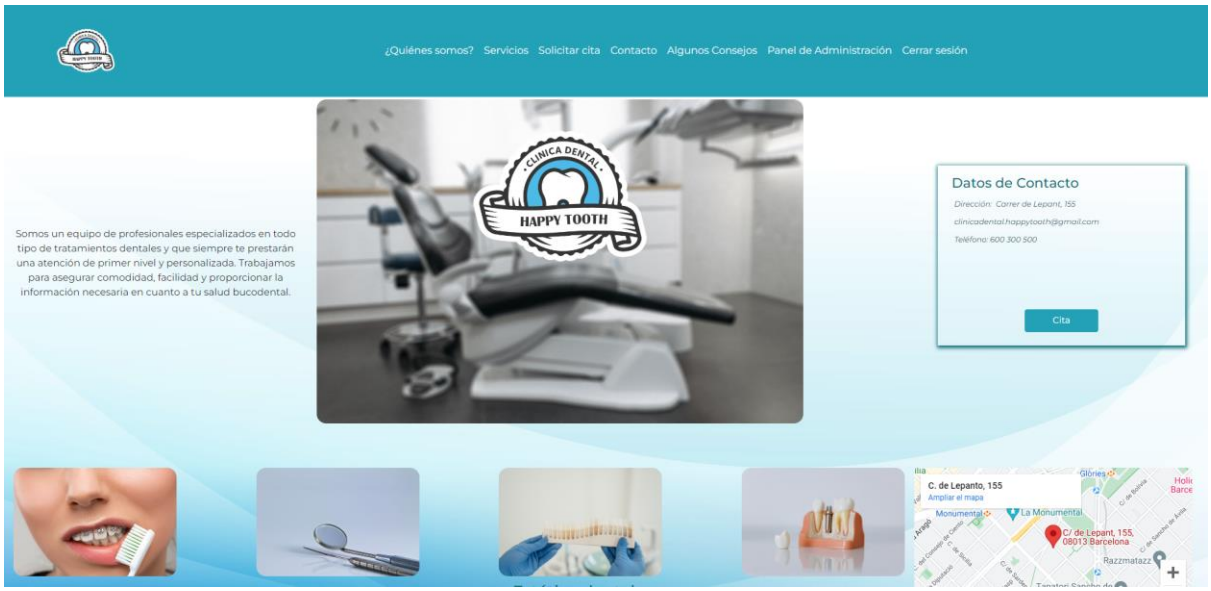


2. **Usuarios Logueados**:

- Clientes:
- Además de las opciones anteriores, los clientes tienen acceso a la opción "Pedir Cita" en la barra de navegación para programar citas dentales.
- En lugar de "Iniciar Sesión", verán la opción "Cerrar Sesión" para finalizar su sesión.

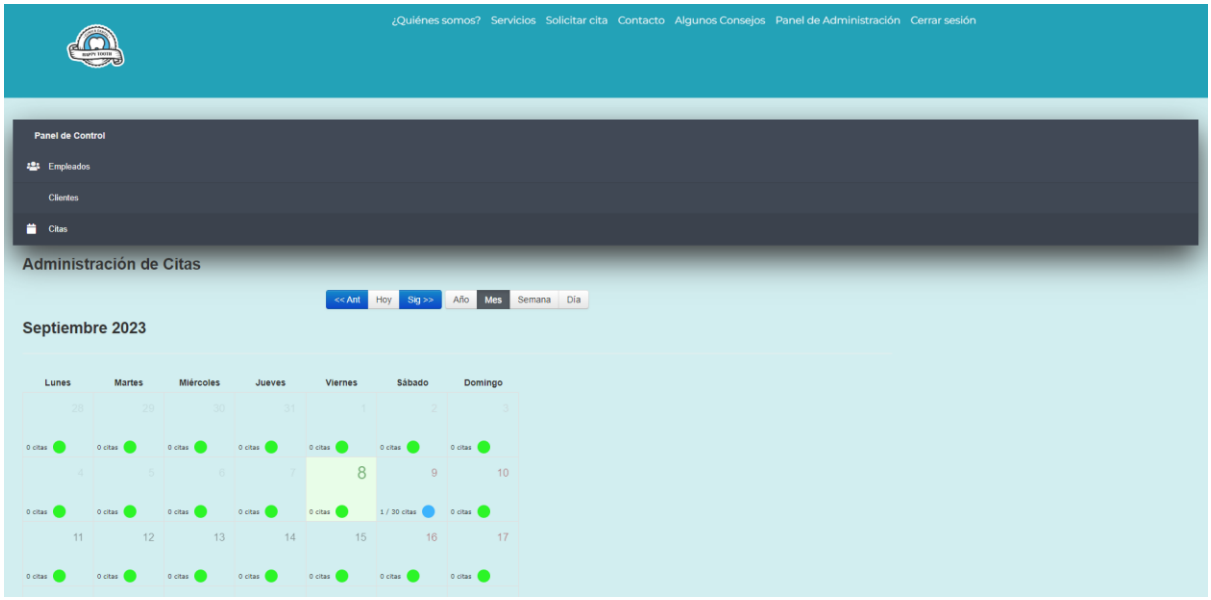


- Dentistas y Administradores (rol “2” y “1”):
- Acceso a un panel de administración con opciones adicionales.

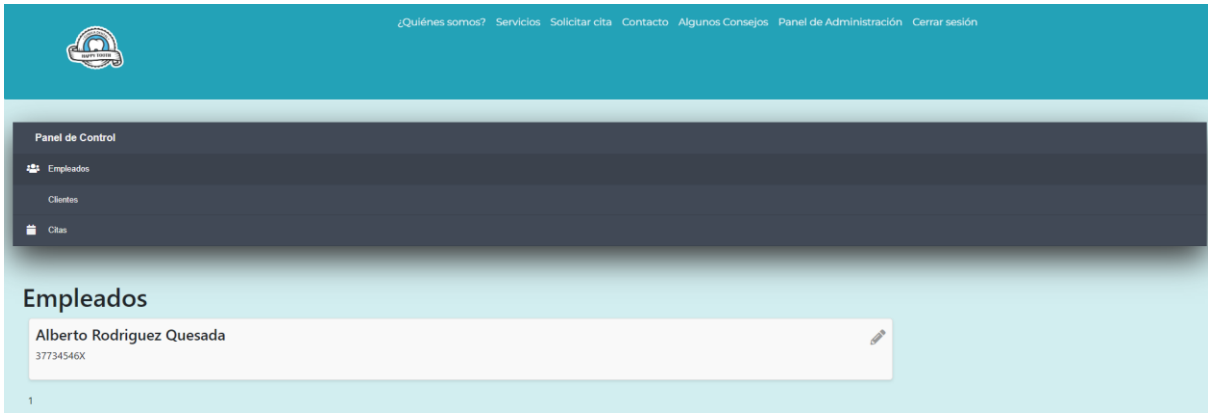


Panel de Administración (Dentistas y Administradores)

1. Gestión de Citas:
   - Crear nuevas citas dentales seleccionando la fecha y hora disponibles.
   - Borrar citas programadas (solo si son responsables de la cita).
   - Consultar las citas pendientes y confirmadas.
   - Imprimir facturas de las citas

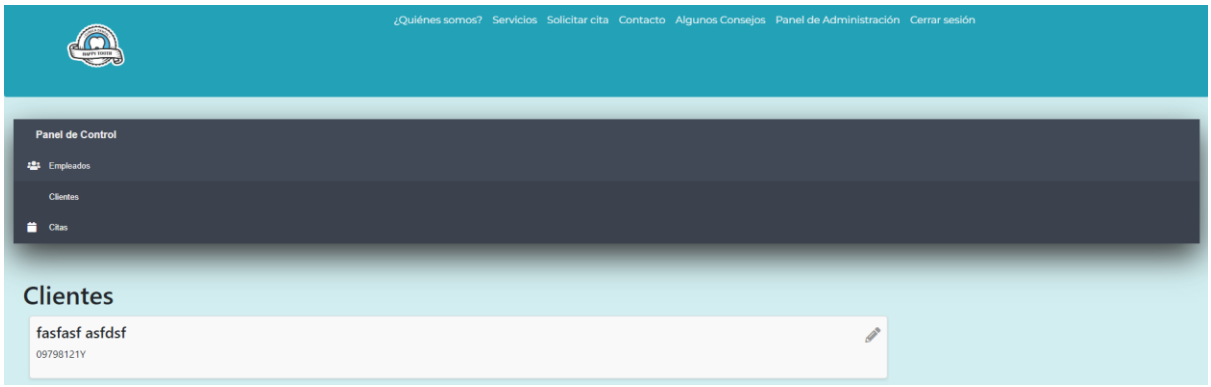


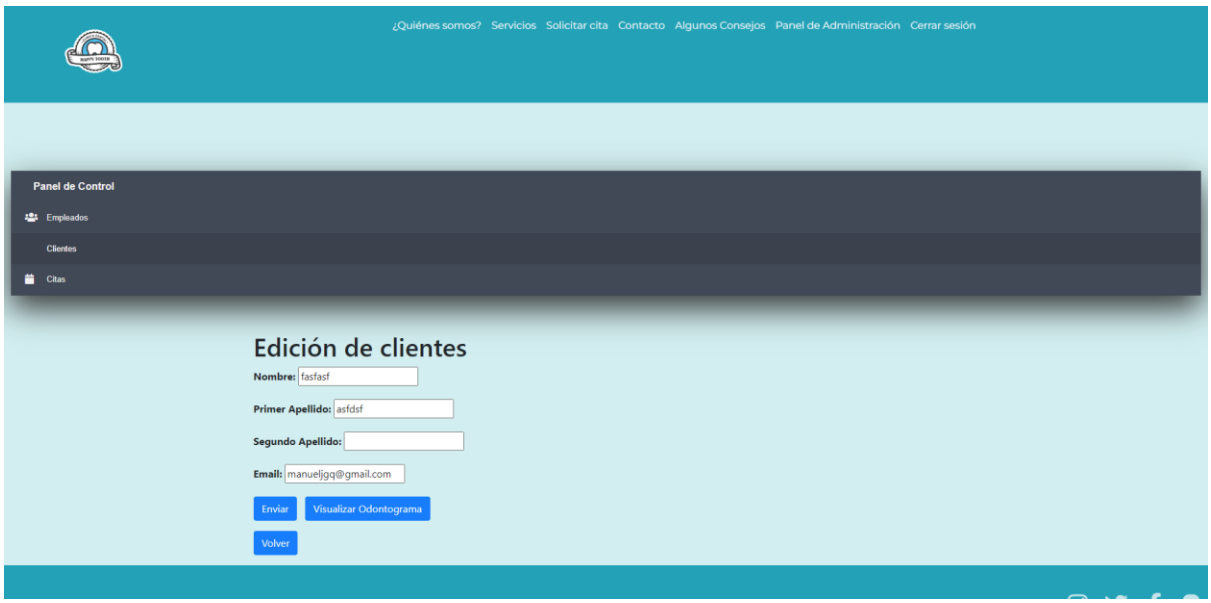
2. Gestión de Empleados:
   - Acceso a datos de empleados, incluyendo información personal y de contacto.
   - Posibilidad de editar información de empleados.

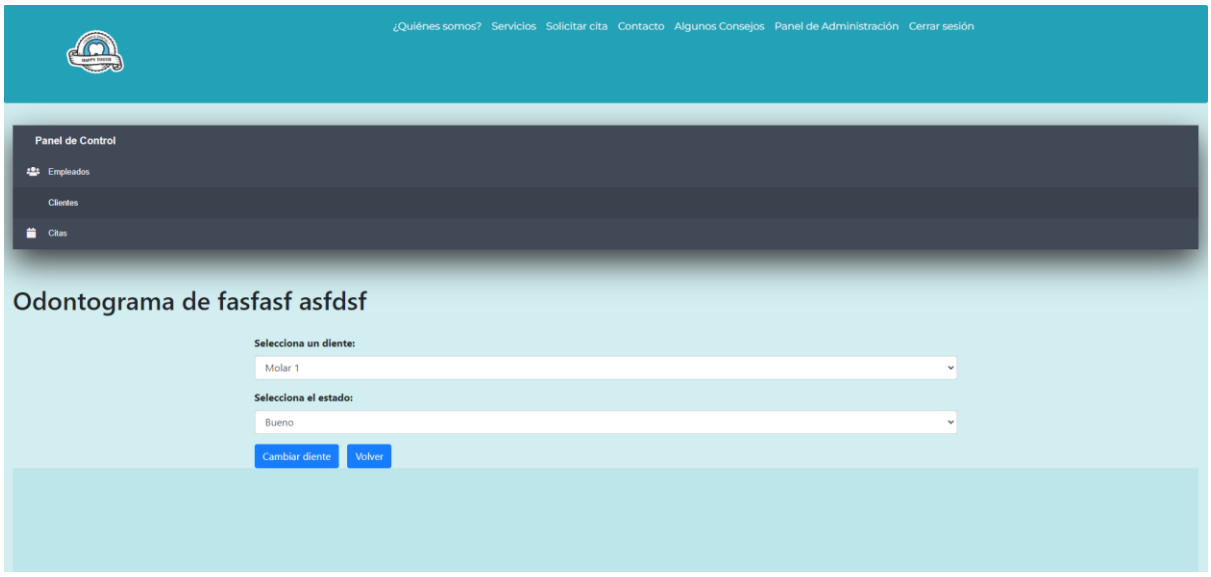


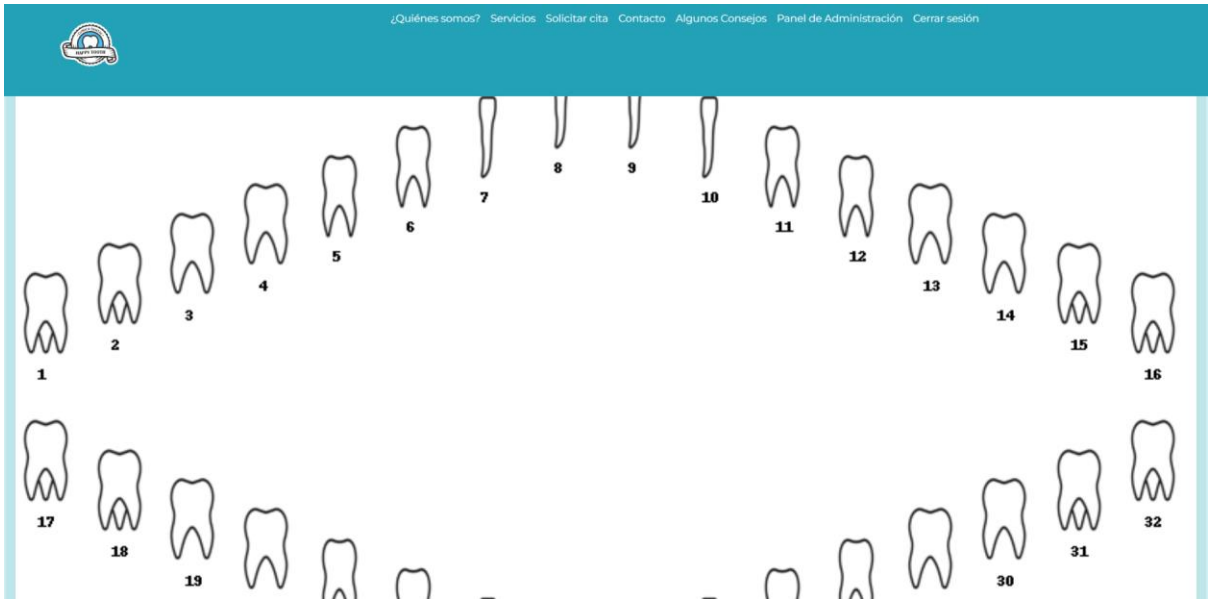
3. Gestión de Clientes:

- Acceso a datos de clientes, incluyendo información personal y de contacto.
- Consultar y editar el odontograma de los clientes, donde se registran las condiciones de los dientes (sano, regular-sucio, malo-enfermo).



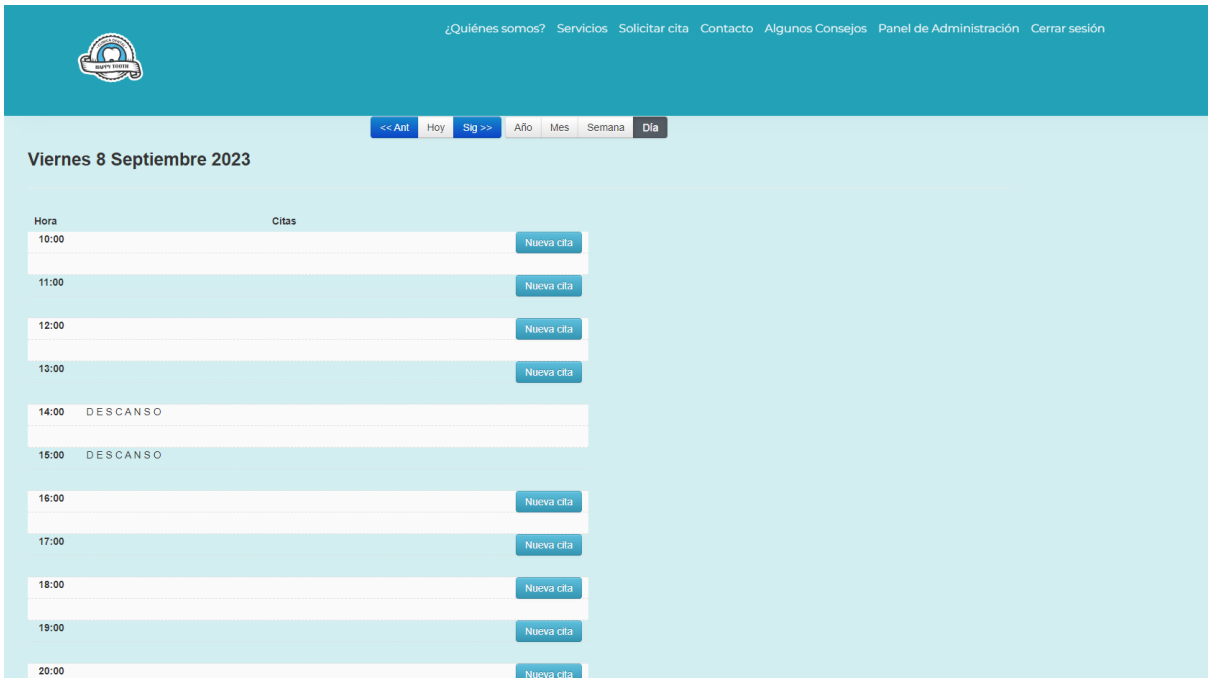






Solicitud de Citas

- Las citas se pueden programar en franjas horarias de 9:00 a 21:00, con un descanso durante la hora de comer.
- Solo se permiten 3 citas por cada hora disponible, asegurando una programación eficiente.



Cierre de Sesión

- En cualquier momento, los usuarios pueden cerrar sesión para garantizar la seguridad de su cuenta y datos personales.

Preguntas y Ayuda

- Si tienes alguna pregunta o necesitas asistencia, utiliza la página de "Contacto" para enviar un mensaje al equipo de la clínica dental.

Apéndice III. Descripción de contenido suministrados

En este apéndice se pasa a detallar todo el contenido necesario para el correcto funcionamiento del proyecto. Junto al archivo comprimido con el código del proyecto se incluye un archivo .zip con los siguientes ficheros:

- .env: en él viene todos los datos clave de configuración para el proyecto, como el string de conexión a la base de datos y la configuración para el smtp de google para el envío de emails y el oauth para la pasarela de inicio de sesión de google
- dentalclinicweb.box: la caja donde se ha trabajado en el proyecto. Facilita tener que tener que instalar de 0 toda la máquina con python, flask y todas las librerías necesarias
- dentalclinicweb.sql: estructura sql de la base de datos para su importación en mysql y poder empezar a funcionar correctamente