UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería Informática

APLICACIÓN WEB Y MÓVIL MODERNA PARA CLÍNICAS DENTALES

MODERN WEB AND MOBILE APPLICATION FOR DENTAL CLINICS

Realizado por
Francisco Jesús Perdiguero Moreno

Tutorizado por
María Mercedes Amor Pinilla

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, diciembre de 2021

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA INFORMÁTICA

APLICACIÓN WEB Y MÓVIL MODERNA PARA CLÍNICAS DENTALES

MODERN WEB AND MOBILE APPLICATION FOR DENTAL CLINICS

Realizado por
Francisco Jesús Perdiguero Moreno

Tutorizado por
María Mercedes Amor Pinilla

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, DICIEMBRE DE 2021

Fecha defensa: Diciembre de 2021

Resumen

En base al desarrollo de una aplicación web y móvil intuitiva para clínicas dentales, este trabajo de fin de grado aborda los retos que suelen encontrar los clientes, las clínicas dentales y los desarrolladores. Además, trata de ofrecer la solución más eficiente. Los retos de este proyecto se engloban en los siguientes aspectos: calidad del servicio (QoS), en relación con los clientes que utilizan la aplicación; productividad y gestión y comunicación eficientes, en relación con los trabajadores de las clínicas dentales que utilizan la aplicación; técnicas de programación eficaces, en relación con los desarrolladores/programadores que producen la aplicación. Se ha utilizado el patrón de arquitectura de software Modelo-Vista-Controlador (MVC) para superar y resolver los desafíos. Se utilizarán tecnologías front-end simples (HTML5, CSS, JavaScript). Así como las tecnologías C#, .Net, LINQ y Microsoft SQL. La aplicación móvil se ha desarrollado para el sistema Android. Además, también se ha introducido la posibilidad de desarrollo en un futuro de aplicaciones iOS y UWP. Se han utilizado varios algoritmos avanzados y técnicas computacionales. En general, este trabajo trata de incluir los enfoques más eficaces para el desarrollo de aplicaciones web y móviles modernas y seguras, ya no solo para clínicas dentales, sino para cualquier propósito.

Palabras clave: web, móvil, aplicación, clínica dental, HTML, CSS, JavaScript, QoS, MVC, LINQ-to-SQL, Android, iOS, productividad, eficiencia, cliente, servidor.

Abstract

In light of developing a user-friendly dental clinic web and mobile application, this final degree project deals with the challenges that clients, dental clinics, and developers often encounter. It went further to providing the most efficient solution. The challenges are under the umbrella of; Quality of Service (QoS), pertaining to clients using the application; Productiveness and effective management and communication, pertaining to the dental clinics' workers using the application; Efficient programming techniques, pertaining to the developers/programmers producing the application. The Model-View-Controller (MVC) software design pattern was employed to conquer and solve the challenges. For the Model, simple front-end technologies (HTML5, CSS, JavaScript) were Introduced and applied. For the View, C# and .Net technologies came into play. A combination of C#, .Net frameworks, LINQ, and Microsoft SQL, were engaged for the Controller. Not let aside was the Android OS technology essential for the mobile application. Moreover, the possibility for developing iOS and UWP applications were also introduced. Several advanced algorithms and computational techniques were Introduced. The thesis aims to introduce the most effective approaches to developing secure modern web and mobile applications not limited to dental clinics but for general purposes.

Keywords: web, mobile, application, dental clinic, HTML, CSS, JavaScript, QoS, MVC, LINQ-to-SQL, Android, iOS, productiveness, efficiency, client, server.

Índice

1

Introducción

1.1 Motivación

Aunque el rápido crecimiento y el avance de la tecnología han tenido un impacto positivo en la comunicación entre individuos y entidades, también se han generado grandes expectativas sobre las funcionalidades de las aplicaciones utilizadas en las comunicaciones. Especialmente en esta pandemia provocada por el Covid-19, en la que la prevención del contacto físico es una prioridad absoluta, los usuarios de aplicaciones web y móviles utilizadas para comunicarse esperan una rápida operatividad de las aplicaciones, lo cual en ciertas ocasiones no ha sido posible.

En este contexto, un ejemplo típico puede ser una aplicación web y móvil de una clínica dental que se espera que optimice la comunicación de ida y vuelta entre los clientes y la propia clínica. En este sentido, los clientes desean reservar, revisar y modificar las citas e interactuar con el personal de la clínica con unos pocos clics. Asimismo, el personal de la clínica encargado de la atención a los clientes espera gestionar la información de los mismos con facilidad a través de las aplicaciones web y móviles que la clínica pone a su disposición. Además, los desarrolladores de esta aplicación esperan

acceder y gestionar sus códigos y algoritmos implementados lo más rápidamente posible para su mantenimiento.

Desgraciadamente, las expectativas mencionadas en el párrafo anterior son complicadas de alcanzar debido, generalmente, a un mal desarrollo y gestión de estas aplicaciones. Este proyecto aporta soluciones realistas y sugeiere estudios adicionales a ciertos retos para mejorar algunas de las soluciones actuales.

1.2 Objetivos

Este trabajo de fin de grado tiene como objetivo presentar un enfoque moderno y seguro para el desarrollo de aplicaciones web y móviles para la gestión de clínicas dentales. Una plataforma donde participan diversas clínicas asociadas al proyecto con funcionalidades idénticas para cada una de ellas. Para ello se desarrollará una aplicación web y móvil intuitiva y fácil de usar para los clientes y trabajadores de varias clínicas dentales, centrándose en tres principales funcionalidades:

- **Gestión de Facturación:** Este módulo permitirá el pago en remoto de las consultas y tratamientos. Los clientes no necesitarán acudir presencialmente a una clinica para pagar sus facturas. Podrían acceder a sus facturas directamente desde sus dispositivos móviles. Con un simple clic, se podrá realizar el pago usando su tarjeta bancaria o PayPal. Así, todo esto se vuelve cómodo y sencillo sin necesidad de tener papeleo de por medio.
- **Planificación de citas:** Los clientes podrán gestionar (pedir, cambiar o eliminar) y consultar sus citas. Los clientes tendrían acceso al calendario para hacerlo directamente. Además, el cliente recibirá un recordatorio de su cita vía SMS o Email.
- **Tratamiento y almacenamiento de datos:** Este módulo se encargará del almacenamiento de datos personales, prestando especial atención a su protección

y tratamiento. Para ello se recurrirá a los servicios de almacenamiento en la nube de Microsoft.

Finalmente, aparte de estos objetivos más técnicos o en lo que a funciones se refieren, también se pretenden alcanzar ciertos objetivos mediante la elaboración de este proyecto, entre ellos:

- **Eficiencia de la aplicación:** El proyecto utilizará una serie de tecnologías que permitirán resolver el problema de la latencia del sitio web. Proporcionaría tanto a los clientes como al personal de la clínica una excelente experiencia de navegación que mejorará la calidad del servicio (QoS).
- **Estrategia Mobile-first:** La navegación por Internet y la comunicación en línea a través de dispositivos móviles son inmensas, especialmente en esta situación de emergencia sanitaria en la que vivimos actualmente. Este proyecto permitiría a los clientes y los trabajadores de la clínica dental solicitar información sobre cualquier tema directamente utilizando sus dispositivos móviles. No existirán problemas debido a la latencia cuando los clientes o el personal de la clínica utilicen el móvil para navegar por el sitio web.

1.3 Tecnologías utilizadas

La eficiencia de la aplicación está impulsada y respaldada por las siguientes tecnologías que han llevado a esta aplicación a un alto nivel de excelencia:

- HTML5
- CSS
- JavaScript/jQuery
- C#
- Framework .Net
- LINQ-to-SQL

- Base de datos de Microsoft SQL
- Simulador de Android

Las tres primeras tecnologías mencionadas (HTML5, CSS y JavaScript/jQuery) han impulsado la funcionalidad del front-end (cliente). Además, se ha utilizado Bootstrap para la parte gráfica de la aplicación web y móvil. C#, framework .Net, LINQ-to-SQL y la base de datos de Microsoft SQL se han utilizado para formar el mecanismo de back-end. Un simulador de Android se ha usado para llevar a cabo el despliegue de todo el proyecto en dispositivos móviles. Por este motivo, se ha desarrollado todo el proyecto utilizando el concepto de estrategia "mobile-first".

1.4 Metodología de trabajo

La metodolía utilizada para realizar este proyecto ha sido la metodología iterativa. Para ello, el proyecto se divide en diversas tareas agrupadas en lo que se denominan iteraciones. Estas iteraciones a medida que el proyecto avanza serán más y más complejas hasta llegar al resultado esperado. Cada una de estas iteraciones se corresponden con los diferentes capítulos que conforman este trabajo.

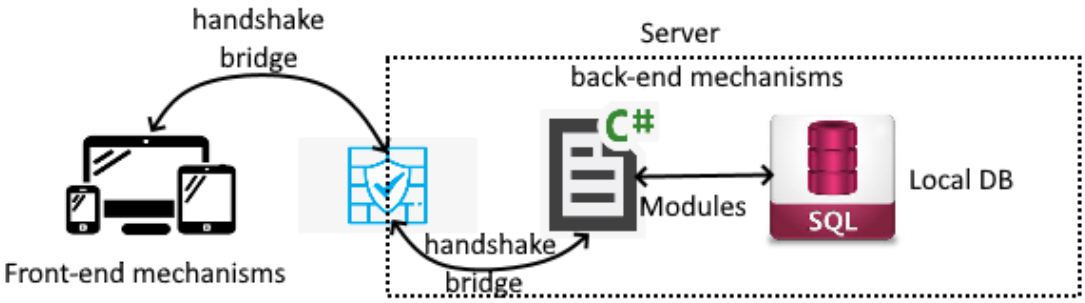
1.5 Estructura de la memoria

Este documento se organizará en diferentes capítulos para presentar todo el proyecto de forma clara, sencilla e intuitiva. La memoria contará con los siguientes capítulos:

- En el capítulo 2 se analizará la importancia de las aplicaciones web y móviles en la cultura actual y como pueden ayudar a solucionar los retos que encuentran hoy en día muchas clínicas dentales. Además, se analizarán las principales tecnologías utilizadas para desarrollar este proyecto.

- El capítulo 3 se centrará en el análisis de los requisitos del proyecto, tanto a nivel funcional como no funcional. Además, se desarrollarán algunos de los principales casos de uso del sistema.
- El capítulo 4 describirá el diseño y el modelo de datos del sistema. En él, se describirá el diseño jerárquico del proyecto, el modelo de almacenamiento de la base de datos, las dependencias del proyecto y sus diversos componentes.
- El capítulo 5 explicará el desarrollo y la construcción de los mecanismos de front-end, back-end y handshake. Además, se describirá de forma detallada la comunicación entre estos mecanismos, incluyendo la gestión de la seguridad de todas las solicitudes. Adicionalmente, se describirá la solución aportada para la creación de la aplicación móvil.

Figura 1: Mecanismo de conexión entre front-end y back-end



Fuente: Elaboración propia

- El capítulo 6 describirá todas las fases de pruebas del proyecto, centrándose principalmente en las pruebas funcionales y no funcionales del sistema.
- El capítulo 7 se centrará en la conclusión del sistema desarrollado, incluyendo la discusión sobre futuras mejoras. En este capítulo también se analizarán los objetivos alcanzados y no alcanzados en este trabajo.

2

Aplicación web y móvil: Tecnologías

Este capítulo analiza el significado de aplicación web y móvil y su importancia en la cultura actual. Además, se estudia la importancia del uso de estas aplicaciones en clínicas dentales y una serie de retos actuales a los que se enfrentan estas clínicas. Para alcanzar una solución a estos retos, se analizará una serie de tecnologías donde se explicarán porque son las optimas para este tipo de proyecto.

2.1 ¿Qué es una aplicación web y móvil?

Aunque las aplicaciones web y móviles son dos tecnologías diferentes impulsadas por Internet, ambas utilizan metodologías web estándar para realizar tareas a través de Internet. Por ejemplo, tanto las aplicaciones web como las móviles presentan el contenido en HTML simple; transmiten comunicaciones y eventos a través de funciones y peticiones sencillas de JavaScript; y dan estilo a los contenidos mostrados con la ayuda de CSS. En los siguientes subpárrafos se definirá y discutirá ambas tecnologías por separado.

2.1.1 Aplicación web

Gibb (2016) define una aplicación web como "a computer program that utilizes web browsers and web technology to perform tasks over the Internet" (para. 1). En otras palabras, podemos decir que una aplicación web es un software diseñado para ejecutarse en sistemas informáticos personales con el objetivo de acceder a un servicio en la red. En la cultura actual, miles de empresas y millones de personas acceden a internet a través de aplicaciones web; éstas sirven como un medio de comunicación económico entre clientes y empresarios. Podemos afirmar que todas las personas del mundo que acceden a Internet lo hacen a través de una aplicación web o móvil. En enero de 2021, se observaron 4.660 millones de usuarios activos de Internet en todo el mundo, aproximadamente el 59,5% de la población mundial (Johnson, 2021). Y este número sigue aumentando con el paso del tiempo.

Figura 2: Aplicación web

Fuente: QODE (2015)

Ejemplos de una aplicación web típica, por mencionar algunos:

- Formularios en línea.
- Tiendas online.
- Programas de correo electrónico como Gmail, Yahoo y AOL.
- Aplicaciones de almacenamiento en la nube como Google Drive o Microsoft Onedrive.
- Sitios web dinámicos e interactivos como WhatsApp Web y Telegram Web.

2.1.2 Aplicación móvil

A diferencia de las aplicaciones web diseñadas para ejecutarse en ordenadores, las aplicaciones móviles, a veces denominadas "apps", están diseñadas para ejecutarse en dispositivos móviles como smartphones, tablets y smartwatches. Las aplicaciones móviles suelen servir para proporcionar a los usuarios/clientes servicios similares a los que ofrecen los ordenadores normales pero de una forma muy accesible. Hoy en día, prácticamente una gran cantidad de personas en todo el mundo dispone de dispositivos móviles con los que buscan información, trabajan o interactúan con otras personas. Acceder a servicios de internet a través de las aplicaciones móviles se ha convertido en el método más rápido y sencillo en los últimos tiempos.

Figura 3: Aplicación móvil

Fuente: Rose (2019)

Los ejemplos de una aplicación móvil típica incluyen las siguientes categorías:

- Aplicaciones de juegos, como Candy Crush,
- Aplicaciones de productividad, como la aplicación de una clínica dental, una aplicación bancaria, una aplicación de reserva de hoteles y aplicaciones de compras.
- Aplicaciones de redes sociales, como Whatsapp, Facebook o TikTok.

2.2 ¿Por qué son importantes las aplicaciones web y móviles?

El avance de la tecnología ha dado lugar a un rápido crecimiento de la comunicación a través de Internet. Las aplicaciones web y móviles (AWM) proporcionan estas facilidades de comunicación rápida. La importancia de estas aplicaciones ha sido destacable durante la pandemia en la que nos tocó vivir. A pesar del parón que experimentó el mundo, la comunicación a través de la web y las aplicaciones móviles no se paralizó; la gente de todo el mundo pudo seguir comunicándose gracias a Internet.

A partir de ahora, este trabajo utilizará el acrónimo AWM, para referirse a una aplicación web y móvil en algunas situaciones.

La importancia de las AWM en la vida y las actividades cotidianas es innegable. La razón de esto es la enorme transformación de los dispositivos de telecomunicación gracias al avance de la tecnología. Los ordenadores y los dispositivos móviles de hoy en día ya no son el dispositivo de comunicación convencional que se utilizaban en las décadas pasadas. Las AWM se han convertido en el principal punto de atención para los individuos y las empresas por igual, gracias a las diversas e increíbles características y oportunidades que ofrecen. El progreso de la tecnología móvil, la disponibilidad, el acceso a Internet de alta velocidad y las agradables interfaces de estos dispositivos dan lugar a nuevas experiencias innovadoras en la programación de AWM (Oza, 2017).

Hoy en día, la disponibilidad de las AWM respaldada por el avance de la tecnología ha cambiado la forma de las actividades diarias del ser humano. No se limita sólo a la comunicación, sino que ha cambiado nuestra forma de sentir y experimentar la informática. Hace décadas, era complejo y tedioso para las organizaciones (los principales destinatarios de los dispositivos informáticos) consultar el simple correo electrónico a través de los ordenadores debido a la baja velocidad de Internet. Hoy en

día, incluso los niños de la escuela primaria pueden navegar por la web y consultar sus correos electrónicos con un simple dispositivo móvil.

Las AWM tienen un uso ilimitado en todos y cada uno de los ámbitos de nuestra vida. El uso de las AWM es evidente en todas partes, como la comunicación, la educación, las redes sociales, las compras, los negocios y la banca.

Figura 4: Tipos de aplicaciones para el día a día

Fuente: https://topnet.com.my/

2.3 ¿Por qué las clínicas dentales necesitan aplicaciones web y móvil?

Las AWM se han convertido en un componente esencial de los negocios en el mundo actual. Las empresas ahora pueden desarrollarse y ser más directas utilizando las AWM y alcanzar sus objetivos rápidamente. Las AWM ayudan a dirigirse a numerosos clientes y consumidores simultáneamente. Las organizaciones están aprovechando rápidamente el avance de la tecnología y el Internet de alta velocidad para difundir su presencia, servicios y productos a través del desarrollo de AWM con la ayuda de los desarrolladores para satisfacer sus demandas de negocio. Entonces, ¿por qué las clínicas dentales serían una excepción? Las AWM son esenciales para las clínicas dentales por varias razones:

- **Publicidad y marca**: Ya no es posible que las empresas vean crecer su cuota de mercado si no cuentan con una AWM adecuada. Mientras que las empresas más destacadas pueden permitirse el lujo de contar con sus equipos de desarrollo para estos fines, las empresas más pequeñas subcontratan el trabajo a empresas de desarrollo web para obtener la misma ventaja a un coste reducido. Ayuda a las organizaciones a llegar a nuevos clientes y darles a conocer la organización y sus servicios.

  Las AWM desempeñan un papel crucial en el proceso de creación de marca. Con su ayuda, es más fácil mantener un canal de comunicación adecuado entre los clientes potenciales y las empresas. Gracias a ellas se pueden aumentar las oportunidades de vender los servicios o productos. Al mismo tiempo, la popularidad de la organización y la aparición de nuevos clientes potenciales crece. Con la ayuda de los desarrolladores de comercio electrónico, una empresa puede conseguir un mercado totalmente nuevo para realizar ventas.
- **Atención al cliente**: Las AWM también ofrecen opciones para mejorar la atención al cliente. Hoy en día, estas aplicaciones pueden convertirse en la principal vía de contacto entre los clientes potenciales y la empresa. Lo mejor de las AWM es que se pueden acceder a ellas en cualquier momento. Incluso la ubicación ya no es una limitación. Por supuesto, sólo una empresa de desarrollo web de alta calidad puede garantizar tal facilidad en una aplicación.
- **Ventaja competitiva**: El mundo empresarial actual se ha vuelto tan intensamente competitivo que se ha vuelto fundamental contar con AWM específicas para cada empresa. Estas aplicaciones pueden convertirse en herramientas esenciales para la captación de clientes. Con la ayuda del desarrollo de aplicaciones para iOS y Android, las empresas pueden utilizar los teléfonos

inteligentes para comercializar sus productos con una importante ventaja competitiva.

2.4 Desafíos actuales en las clínicas dentales

Aunque la mayoría de las clínicas dentales cuentan con AWM en la actualidad, hay retos que superar y mejoras que deben considerarse. Las AWM de la mayoría de las clínicas dentales no proporciona algunas funciones vitales. Algunas de ellas son:

- **La capacidad de gestionar las citas**: Los datos demuestran que la mayoría de las clínicas dentales no ofrecen la posibilidad de personalizar las citas. Por ejemplo, en el AWM de muchas clínicas dentales, sólo existe la posibilidad de utilizar formularios de contacto o hacer una llamada telefónica directa para pedir una cita.
- **La capacidad de gestionar un sistema de facturación dinámico**: Según algunos resultados de la investigación, la gestión de las facturas es otro reto o componente esencial que falta en la mayoría de las AWM de las clínicas dentales. Por ejemplo, los clientes suelen tener problemas para obtener las facturas pagadas de visitas anteriores a través de la AWM. Las AWM de las clínicas no permiten a los clientes ver todas sus facturas e incluso descargarlas cuando lo necesitan. Las facturas tienen que ser recogidas en las instalaciones físicas o enviadas por correo electrónico cuando lo solicitaban los clientes.
- **La capacidad de procesar el pago en línea**: Otra tendencia de desafío es que la mayoría de las AWM de las clínicas dentales no ofrecen la funcionalidad de pago en línea para que los clientes puedan pagar los servicios directamente desde las aplicaciones. Por ejemplo, en algunas clínicas dentales populares de la provincia de Málaga, el pago del servicio se realiza en la clínica de forma física.

El principal objetivo de este TFG es proporcionar soluciones eficientes a los retos mencionados anteriormente.

2.5 Descripción de las tecnologías empleadas

Para aportar soluciones reales y eficientes antes los retos mencionados anteriormente, será necesario utilizar una serie de tecnolgías para el desarrollo de estas aplicaciones. En esta sección se analizarán cada una de ellas y se demostrará por qué estas tecnologías destacan sobre otras y son las óptimas para este proyecto.

2.5.1 HTML5

HTML5 es la quinta y última versión del lenguaje de marcado de hipertexto (HTML). Es la tecnología estándar recomendada por el World Wide Web Consortium (W3C) para estructurar y presentar contenidos en la World Wide Web (W3C, 2014). La tecnología es mantenida hoy en día por el Web Hypertext Application Technology Working Group (WHATWG). WHATWG se trata de un consorcio de los principales proveedores de navegadores como Google, Microsoft, Mozilla y Apple.

Como su nombre indica, HTML no es un lenguaje de programación. Es un lenguaje de marcado que se utiliza para organizar el contenido que se muestra en nuestras pantallas desde Internet. Los documentos HTML5 se componen de elementos HTML, normalmente denominados etiquetas, escritos dentro de llaves angulares para categorizarlos.

Los documentos HTML5 se alojan en servidores web. Desde el navegador se realiza una petición al servidor. Después el servidor recupera de su disco duro esa página, la devuelve al navegador y la página se renderiza. Uno de los principales objetivos de HTML5 es introducir información en un documento HTML5 de forma que sea semántico y no visual, es decir, todos los aspectos de diseño son gestionados por el lenguaje CSS del cual hablaremos en apartados posteriores.

HTML5 evolucionó a partir de HTML4, que también evolucionó de HTML. HTML4 y versiones previas fueron los estándares del W3C durante 15 años y todavía son utilizados por muchos desarrolladores. Sin embargo, se mejoró a HTML5 para apoyar una experiencia de usuario consistente y más fluida para los usuarios y desarrolladores web.

HTML5 es el resultado de una colaboración entre el W3C y el WHATWG; se unieron en 2006 para reducir dependencias con plugins, mejorar la gestión de errores y sustituir largas secuencias de comandos por más elementos HTML, lo que solía ser un problema con antiguas versiones.

2.5.1.1 HTML5 vs HTML4 y versiones anteriores

Algunos desarrolladores sostienen que la diferencia más significativa entre HTML5 y sus homólogos más antiguos es que permite la integración de audios y vídeos en las especificaciones del lenguaje. Sin embargo, según Arsenault (2017), existen otras diferencias más relevantes entre HTML5 y versiones anteriores:

- Las reglas de análisis sintáctico mejoradas permiten un análisis sintáctico más flexible en comparación con el análisis sintáctico fijo y restringido de las versiones anteriores.
- Añade compatibilidad con nuevos atributos de entrada, como el correo electrónico, URLs, fechas y horas, que no están presentes directamente en las otras versiones.
- La tecnología HTML5 ofrece soporte para el almacenamiento en caché fuera de línea y mecanismos de arrastrar y soltar, que la contraparte no soporta.
- Es compatible con los gráficos vectoriales y puede prescindir de plugins API externos como Flash o Silverlight.
- También es compatible con MathML para una mejor visualización de las notaciones matemáticas, cosa que las versiones anteriores no hacen.

- Incluye soporte para que JavaScript se ejecute en segundo plano cuando se muestra el documento.

2.5.1.2 Ventajas de HTML5

Aunque HTML4 y versiones anteriores tienen algunas ventajas, las ventajas que ofrece HTML5 para los usuarios son más importantes. Algunas de ellas son (Arsenault, 2017):

- HTML5 permite el almacenamiento de datos en el dispositivo del usuario. En otras palabras, las aplicaciones pueden seguir funcionando sin una conexión a Internet adecuada.
- Proporciona una gama más amplia de colores, sombras y otros efectos que las páginas web pueden mostrar.
- Permite medios interactivos, como los juegos, que pueden ejecutarse en los navegadores web sin necesidad de plugins adicionales.
- La reproducción de vídeo y audio tampoco requiere un plugin adicional.
- En los lenguajes de programación modernos, HTML5 permite incrustar directamente el código en el documento (codificación y renderización del lado del servidor)

Éstas son, por mencionar sólo algunas de las más relevantes, las razones por las que el desarrollo de aplicaciones web y móviles con HTML5 es extremadamente esencial.

2.5.2 Bootstrap

Bootstrap es una herramienta que contiene un conjunto de hojas de estilo en cascada (CSS) acompañados de algunos módulos especiales de JavaScript (JS) que mejoran el diseño y el comportamiento de las páginas web, los temas, la tipografía, los botones, la navegación y los formularios, por mencionar algunos de los muchos componentes de las interfaces gráficas de usuario (GUI) (Otto, 2012). Por lo tanto, podemos considerar a Bootstrap como una biblioteca CSS y JS que ayuda a simplificar el desarrollo de

páginas web interactivas. Gracias a esta herramienta, se ha podido seguir de forma rápida y eficiente la estrategia *mobile-first*.

Bootstrap se llamó originalmente Twitter Blueprint y fue desarrollado por Mark Otto y Jacob Thornton en Twitter. Se desarrolló como un framework para mejorar la coherencia entre las herramientas internas de Twitter. Inicialmente, se utilizaron varias bibliotecas para el desarrollo de la interfaz gráfica, lo que posteriormente provocó incoherencias en los diferentes dispositivos que usaban Twitter. Según Otto (2012):

> *Un grupo súper pequeño de desarrolladores y yo nos reunimos para diseñar y construir una nueva herramienta interna y vimos la oportunidad de hacer algo más. A través de ese proceso, nos vimos haciendo algo mucho más sustancial que otra herramienta interna. Meses después, terminamos con una primera versión de Bootstrap para documentar y compartir patrones de diseño y activos comunes dentro de la empresa* (párrafo 2).

El resultado del desarrollo fue funcionalmente robusto. Tras unos meses de desarrollo conjunto, un grupo de desarrolladores de Twitter (incluido Mark Otto) empezó a contribuir al proyecto como parte de una semana al estilo de un hackathon para el equipo. Se cambió el nombre de Twitter Blueprint a Bootstrap antes de lanzarlo como proyecto de código abierto a finales de agosto de 2011 (Otto, 2012). En la actualidad, sigue siendo mantenido por Mark Otto, Jacob Thornton y una gran comunidad de colaboradores.

2.5.2.1 Bootstrap vs CSS estándar

Al igual que el CSS de Bootstrap, el CSS estándar es un lenguaje de hojas de estilo utilizado para describir la presentación de un documento HTML o XML. En otras palabras, CSS describe cómo deben presentarse los elementos o los medios en la pantalla de un dispositivo (Mozilla Developer, 2021a). A diferencia de Bootstrap, el CSS estándar no es una colección de paquetes de módulos CSS y JavaScript.

Joshi (2020) diferencia explícitamente el CSS estándar de Boostrap en la siguiente tabla:

Tabla 1: Comparación entre CSS estándar y Bootstrap

CSS estándar	**Bootstrap CSS**
El CSS estándar no ofrece soporte para un sistema de cuadrículas.	Bootstrap, por otro lado, se basa en un sistema de cuadrículas.
El CSS estándar no proporciona directamente soporte para páginas o sitios web responsive. Los desarrolladores tienen que escribir líneas de código adicionales para ello.	En Bootstrap, podemos diseñar un sitio o página web responsive sin necesidad de escribir código extra para la funcionalidad
El CSS estándar es algo complejo porque no hay clases ni diseños predefinidos	Bootstrap es relativamente sencillo de entender y dispone de muchas clases prediseñadas, como las clases para alineación, colores de fondo, tamaños de las fuentes, ancho y alto de elementos,...
El uso de CSS estándar implica que para toda la funcionalidad de visualización requerida, los desarrolladores tienen que escribir todo el código de estilo desde cero.	En Bootstrap, podemos añadir dinámicamente las clases predefinidas en el código sin necesidad de escribir líneas de código adicionales.

Fuente: Elaboración propia basado en Joshi (2020)

2.5.2.2 Ventajas de Bootstrap

Aunque la tabla anterior ya expresaba algunas de las ventajas de Bootstrap sobre el CSS estándar, se listará de forma más detallada algunas de sus principales ventajas:

- **Flexibilidad**: Bootstrap permite a los diseñadores aplicar toda su creatividad de forma rápida y sencilla. Es un framework/kit estructurado en CSS con clases predefinidas. Además, permite el uso de Javascript para su control.

- **Adaptabilidad (responsive):** El avance de la tecnología dio lugar a la aparición de dispositivos móviles inteligentes, las tabletas y los ordenadores de pantalla táctil con diferentes tamaños y resoluciones de pantalla. Bootstrap proporciona los módulos necesarios para el desarrollo de sitios web adaptables a todo tipo de pantallas.
- **Módulos de Javascript**: Bootstrap engloba una colección de módulos JavaScript que podemos utilizar para inyectar comportamientos inteligentes a algunos componentes del documento HTML. Por ejemplo, los menús desplegables, los deslizadores y los botones pueden hacerse inteligentes gracias a los distintos módulos de JavaScript que posee Bootstrap.
- **Abundante documentación y una comunidad de ayuda**: Bootstrap proporciona una documentación detallada de forma excepcional y es apoyado y mantenido por numerosas comunidades potentes. La documentación ofrece un excelente apoyo en el aprendizaje de las implementaciones y la estructura del framework.

Estos son algunos de los beneficios por los que el desarrollo de aplicaciones web y móviles con Bootstrap es extremadamente esencial.

2.5.3 Javascript/JQuery

JavaScript (JS) es un lenguaje de programación ligero con funciones de primera clase. Es popularmente conocido como el lenguaje por excelencia para la programación dinámica de páginas web en el lado del cliente (Mozilla Developer, 2021b). Sin embargo, no se limita al desarrollo de páginas web. Muchas otras herramientas, como Node.js, Apache CouchDB y Adobe Acrobat, también lo utilizan. JavaScript es un lenguaje dinámico que admite la programación orientada a objetos, imperativa, funcional y declarativa.

JQuery es una biblioteca de JavaScript pequeña, rápida y con muchas funciones (jQuery, 2021). Permite la manipulación dinámica de los componentes de los documentos HTML, como el manejo de eventos de animación o el uso de Ajax para peticiones. JQuery ha cambiado la forma en que millones de personas escriben JavaScript.

2.5.3.1 Javascript/JQuery vs otros lenguajes

Actualmente, Javascript y sus diferentes frameworks son la única alternativa para la ejecución de código del lado del cliente, puesto que se trata del único lenguaje que los navegadores web son capaces de procesar en la actualidad. Sin embargo, hoy en día existe otra corriente en auge donde se centran en la ejecución de código del lado del servidor.

Los scripts del lado del cliente son ejecutados por los navegadores web. El código fuente se transfiere del servidor web al ordenador del usuario a través de Internet y se ejecuta directamente en los navegadores. Los scripts del lado del servidor son ejecutados por los servidores web. Se utilizan básicamente para crear páginas dinámicas. También puede acceder al sistema de archivos que reside en el servidor web.

Las principales diferencias de ambas tendencias pueden resumirse en la siguiente tabla:

Tabla 2: Comparación entre script del lado del cliente y del lado del servidor

Scripts del lado del cliente	**Scripts del lado del servidor**
El código fuente es visible por el usuario.	El código fuente no es visible para el usuario porque lo que devuelve el servidor es una página HTML.
Normalmente depende del navegador y de su versión.	En este caso se puede utilizar cualquier tecnología del lado del servidor y no depende del cliente.
Se ejecuta en el ordenador del usuario.	Se ejecuta en el servidor.
Hay muchas ventajas relacionadas con esto, como tiempos de respuesta más rápidos, una aplicación más interactiva.	La principal ventaja es su capacidad para personalizar en gran medida, los requisitos de respuesta y los derechos de acceso en función del usuario.
No proporciona seguridad a los datos.	Proporciona más seguridad para los datos.
Es una técnica utilizada en el desarrollo web en la que los scripts se ejecutan en el navegador del cliente.	Es una técnica que utiliza scripts en el servidor web para producir una respuesta personalizada para cada petición del cliente.
Se utiliza HTML, CSS y Javascript.	Se utilizan PHP, Python, Java, C#, etc.

Fuente: Elaboración propia

Todos los componentes del cliente y del servidor que construyen una página web dinámica se denominan aplicación web. Las aplicaciones web gestionan las interacciones de los usuarios, el estado, la seguridad y el rendimiento. Por eso, el uso de una combinación de scripts del lado del cliente y del lado del servidor es la mejor opción. Se trata de una técnica de desarrollo de aplicaciones web para intercambiar contenidos de forma dinámica. Primero, se envía peticiones al servidor para obtener datos. El

servidor devuelve los datos solicitados que luego son procesados por un script del lado del cliente. Esta técnica puede reducir el tiempo de carga del servidor porque el cliente no solicita que el servidor envíe toda la página web; sólo se transmite el contenido que va a cambiar.

2.5.3.2 Ventajas de Javascript/JQuery

Dado que Jquery es una biblioteca avanzada de JavaScript, su ventaja también reside en los beneficios de JavaScript. Algunos de estos beneficios son los siguientes:

- **Popularidad**: Jquery es una biblioteca de JavaScript popular con millones de desarrolladores que la utilizan para el desarrollo de aplicaciones web y móviles. La popularidad se traduce en un sinfín de recursos, fragmentos de código, publicaciones en blogs, tutoriales de calidad y documentación, que son fuentes cruciales de aprendizaje e investigación
- **Compatibilidad entre navegadores**: Cuando una aplicación web puede ejecutarse en diferentes navegadores y presentar un aspecto idéntico, podemos decir que la aplicación web es compatible con todos los navegadores. JavaScript/Jquery permite la manipulación en el script para lograr este propósito puesto que la mayor parte de navegadores poseen un intérprete de código Javascript.
- **Interoperabilidad**: JavaScript se integra eficazmente con otros lenguajes y puede utilizarse en una gran variedad de aplicaciones.
- **Velocidad**: El uso de JavaScript en el lado del cliente es muy rápido porque puede ejecutarse directamente en el navegador del usuario. A menos que se necesiten recursos externos, JavaScript no se ve obstaculizado por las peticiones a un servidor.

Estas son sólo algunas de las ventajas de desarrollar aplicaciones web y móviles con JS/Jquery. En la fase de desarrollo de la aplicación se verán más ventajas de esta tecnología.

2.5.4 C#

C#, pronunciado "si 27ipos" en inglés, es un lenguaje de programación moderno, orientado a objetos y con seguridad de tipos (Wagner et al., 2021). Microsoft lo desarrolló bajo la dirección de Anders Hejlsberg y su equipo dentro de la iniciativa .NET y fue aprobado por la European Computer Manufacturers Association (ECMA) y la International Organization for Standardization (ISO). C# permite a los desarrolladores crear un sinfín de aplicaciones seguras y robustas que se ejecutan con el framework .Net. C# tiene sus raíces en la familia de lenguajes C y resulta muy familiar para los programadores de C, C++, Java y JavaScript.

2.5.4.1 C# vs Python

Al igual que C#, Python es un lenguaje de programación portátil y de propósito general. Está a la altura de C y Java en la mayoría de sus características y tiene capacidades de programación de alto nivel. Aunque C# y Python se encuentran entre las tendencias actuales de los lenguajes de programación más populares, el segundo es un lenguaje de interpretación mientras que el primero no lo es. Sin embargo, ambos se basan en los conceptos de la programación orientada a objetos (POO), son fáciles de aprender y codificar, y ofrecen un desarrollo rápido y un buen rendimiento. Para demostrar por qué C# es la mejor opción para este proyecto, es necesario comparar ambos lenguajes. Shankar (2021) diferencia ambos lenguajes de la siguiente forma:

Tabla 3: Comparación entre C# y Python

C#	Python
Sintaxis y formato más organizados y consistentes.	Código sencillo y fácil de leer, no contiene demasiados símbolos ni formatos.
Se trata de un lenguaje de tipado estático. Los errores de tipificación son detectados antes y la ejecución del programa es más eficiente y segura.	Tipificación dinámica. Es más flexible a pesar de ejecutarse más lentamente y ser más propenso a contener errores de programación. No hay necesidad de declaraciones de variables.
Gracias al framework Common Language Infrastructure (CLI), C# es más rápido y ofrece un mejor rendimiento.	El desarrollo en este lenguaje es más rápido, pero comparando el rendimiento con C#, es ligeramente inferior.
El multithreading es relativamente fácil de llevar a cabo con el framework .NET.	Debido al intérprete que utiliza (Global Interpreter Lock), el multithreading es más complejo.

Fuente: Elaboración propia basado en Shankar (2021)

No hay duda de que C# tiene una estructura más organizada, velocidad y naturaleza multihilo que Python (Shankar, 2021).

2.5.4.2 Ventajas de C#

Además de las ventajas mencionadas en la tabla anterior, podemos presentar una serie de beneficios del uso de C#:

- C# es muy eficaz en la gestión del sistema. La liberación de memoria de los objetos (Garbage Collector) se realiza automáticamente y de forma eficiente.
- No hay problemas de fugas de memoria en C# debido a su alta reserva de memoria.
- El coste de mantenimiento es menor y es más seguro en comparación con otros lenguajes.

- El código de C# se compila en un lenguaje intermedio (.NET), un lenguaje estándar, independiente del sistema operativo y la arquitectura de destino.

2.5.5 Framework .NET

El framework .NET ("dot Net"), desarrollado por Microsoft, es una plataforma de desarrollo de software. Sirve para crear y ejecutar aplicaciones de Windows. Abarca herramientas para desarrolladores, lenguajes de programación y bibliotecas para crear aplicaciones de escritorio y web. También se utiliza para crear sitios web, servicios web y juegos (Thompson, 2021).

El framework estaba principalmente destinado a crear aplicaciones que se ejecutaran en la plataforma Windows. La primera versión (.Net 1.0) fue lanzada en el año 2002. Desde entonces, el framework ha recorrido un largo camino, y la versión actual es .Net 4.8. Con el paso de los años, ha permitido el desarrollo de aplicaciones no solo para Windows, sino para muchas otras plataformas (Linux o IOS).

El framework también es perfectamente compatible con varios lenguajes de programación como Visual Basic y C#. Así, los desarrolladores pueden elegir y seleccionar el lenguaje para desarrollar la aplicación.

2.5.5.1 Framework .NET vs Oracle Java SE

La plataforma Java Standard Edition (Java SE) permite el desarrollo y despliegue de aplicaciones Java en ordenadores de sobremesa y servidores. Esta capacidad hace que Java sea un claro competidor del framework .Net.

Aunque Java ofrece una rica interfaz de usuario, un buen rendimiento, versatilidad, portabilidad y seguridad que requieren las aplicaciones de hoy en día, para este proyecto es preferible la tecnología .Net por estas dos sencillas razones:

- El lenguaje de programación del backend elegido para este proyecto es C# y, por lo tanto, se pretende utilizar todas las características y ventajas significativas que proporciona .Net.
- El uso de Oracle Java SE supone un reto en la gestión de la memoria, cosa que no ocurre con .Net.

2.5.5.2 Ventajas del framework .Net

Algunas de las principales ventajas sobre el uso del framework .Net en este proyecto son las siguientes:

- **Naturaleza orientada a objetos**: .Net se basa en el modelo de programación orientada a objetos. La POO es un modelo de desarrollo que abarca la descomposición del software en piezas más pequeñas fácilmente controlables. El módulo de POO hace el código manejable, respondiendo a problemas recurrentes más fáciles de detectar. Además, .NET permite reutilizar los componentes y el código, con lo que se ahorra tiempo y costes de desarrollo.
- **Interoperabilidad**: El framework .Net es compatible con versiones anteriores. Con cada lanzamiento de nuevas versiones, Microsoft se asegura de que las versiones anteriores del framework se combinen bien con las últimas versiones.
- **Compatibilidad multiplataforma**: El framework .Net es un mecanismo de desarrollo de aplicaciones multiplataforma, lo que significa que permite que el código se ejecute en cualquier versión de Windows y recientemente en otras plataformas como IOS o Linux. Tiene un código totalmente abierto, lo que garantiza que una amplia comunidad de ingenieros pueda contribuir a su desarrollo.
- **Flexibilidad, despliegue y mantenimiento**: Una de las características esenciales y más beneficiosas de .NET es la flexibilidad de despliegue. Puede instalarse como parte de la aplicación desarrollada o por separado. El diseño estructural modular permite incluir todas las dependencias necesarias de forma

rápida y sencilla. Además, posee herramientas que permiten empaquetar fácilmente las aplicaciones. Estos paquetes se distribuyen y se encargan de instalar de forma automática la aplicación.

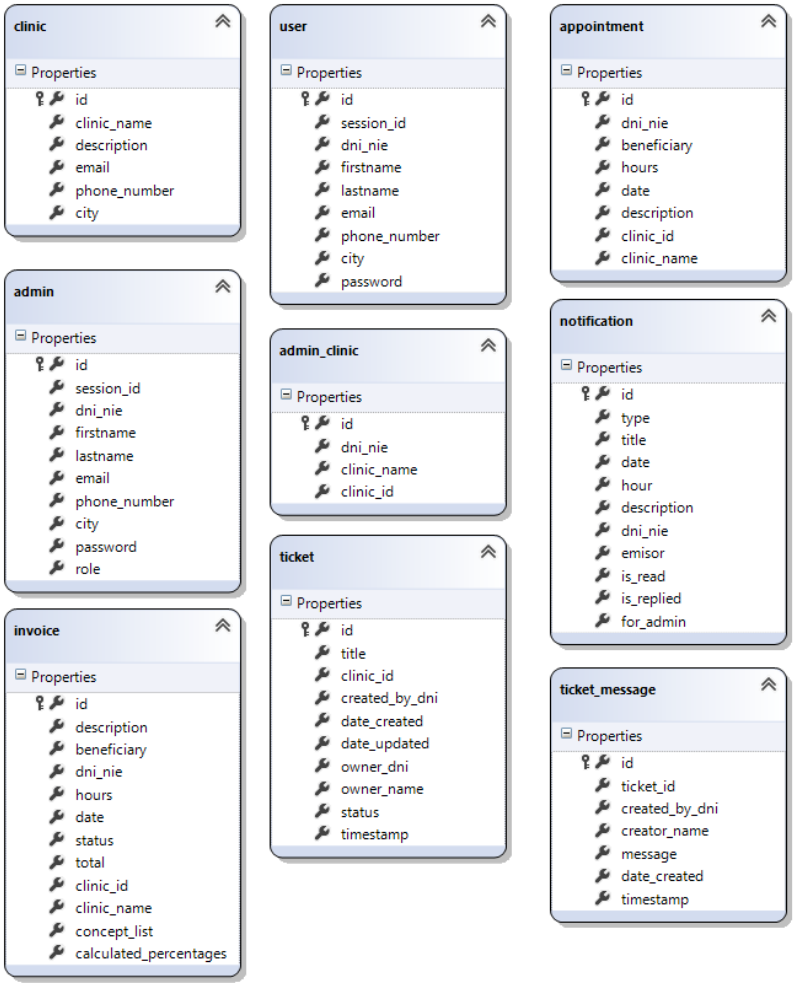
- **Seguridad**: El framework .NET posee un buen mecanismo de seguridad. El mecanismo de seguridad incorporado ayuda tanto a la validación como a la verificación de las aplicaciones. Cada aplicación puede definir explícitamente su mecanismo de seguridad. Cada mecanismo de seguridad se utiliza para conceder al usuario el acceso al código o al programa en ejecución. En el capítulo de desarrollo se profundizará en este tema.

2.5.6 LINQ-to-SQL

LINQ-to-SQL (LINQ) es un módulo de la versión 3.5 y superior del framework .NET que ofrece una infraestructura en tiempo de ejecución para gestionar datos relacionales como objetos (LeBLanc et al., 2021). En LINQ-to-SQL, el modelo de datos de la base de datos relacional se asigna a un modelo de objetos establecido en el lenguaje de programación del desarrollador, en este caso, C#. Cuando la aplicación se ejecuta, LINQ-to-SQL interpreta y traduce las consultas integradas en el lenguaje del modelo de objetos (C#) a SQL y las envía a la base de datos para su ejecución. Cuando la base de datos devuelve los resultados, LINQ-to-SQL los traduce de nuevo a objetos con los que el desarrollador puede trabajar según el lenguaje de programación indicado.

Es común entre los desarrolladores que usan Visual Studio utilizar la herramienta Relational Object Designer (ROD), la cual proporciona una interfaz de usuario agradable para implementar muchas de las características de LINQ-to-SQL desde el propio IDE. Un ejemplo de esta utilidad se muestra en la siguiente imagen tomada del proyecto:

Figura 5: Ejemplo de esquema de tablas en LINQ-to-SQL



Fuente: Elaboración propia

2.5.6.1 Linq-to-SQL vs. Dapper

Dapper es un simple mapeador de objetos para .NET y a menudo es calificado como el rey de los mapeadores relacionales de objetos (ORM) en términos de velocidad. Un ORM se encarga de realizar el mapeo entre la base de datos y el lenguaje de programación.

Uno puede preguntarse por qué este proyecto se utiliza LINQ-to-SQL en lugar de Dapper ya que este último es más rápido. La respuesta es simple: este proyecto se decantó por precisión y facilidad para realizar consultas en lugar de velocidad irrelevante. Al fin y al cabo, sólo es en la primera compilación donde LINQ-to-SQL

parece ser algunos milisegundos más lento que Dapper, y después de eso, la velocidad de ejecución de LINQ-to-SQL supera a Dapper (Jones, 2019).

La naturaleza implícita de Dapper en la sentencia de consulta para realizar consultas a la base de datos la hace débil comparada con Linq-to-SQL. Para entenderlo mejor, vamos a poner un ejemplo con ambas herramientas:

- Si Dapper quiere realizar una consulta a la base de datos para obtener todas las clínicas a las que pertenece un determinado trabajador, el enfoque de Dapper es emplear una cadena en texto plano, como se muestra a continuación:

Figura 6: Ejemplo de uso de Drapper

```
var dni = dic["dni"].ToString();

string raw_sql_query = "select * from admin_clinics where dni_nie !=null and dni_nie==dnie;";

using (var connection = new SqlConnection(Database.fran_tfg_db))
{
    var admin_clinic_list = connection.Query(raw_sql_query).ToList();
}
```

Fuente: Elaboración propia

  - La primera línea del código declara la identidad del trabajador (dni).
  - La segunda línea declara débilmente la consulta utilizando el formato crudo y tradicional de SQL.
  - La tercera línea es un bloque de código que implementa la conexión a la base de datos y luego obtiene la lista de clínicas en cuestión.

  Aunque el enfoque parece bueno, existe la posibilidad de que se produzcan errores tipográficos (33ipos). En el ejemplo anterior, si miramos detenidamente a la cadena *raw_sql_query*, hay un error tipográfico en la última entrada en *dni_nie=dnie*. El "*dnie*" es una errata causada por un error humano.
- Del mismo modo, LINQ-to-SQL, quiere consultar la base de datos para obtener una lista de todas las clínicas en las que trabaja un determinado trabajador. En

ese caso, la consulta es explícita, y la sintaxis para el nombre del atributo no tiene errores, como se muestra a continuación:

Figura 7: Ejemplo de uso de LINQ

```
var dni = dic["dni"].ToString();

fran_tfg_dbDataContext db = new fran_tfg_dbDataContext();

var admin_clinic_list = db.admin_clinics.Where(c => c.dni_nie != null && c.dni_nie == dni).ToList();
```

Fuente: Elaboración propia

- La primera línea del código declara la identidad del trabajador (dni).
- La segunda línea declara explícitamente el contexto de conexión a la base de datos.
- La tercera línea es la consulta que obtiene la lista de clínicas a las que pertenece el trabajador.

Este enfoque es sencillo y sin errores de sintaxis o de atributos en comparación con Dapper. La razón es que LINQ proporciona explícitamente los nombres de todos los atributos que pertenecen a esa tabla consultada. Por lo tanto, dado que nuestro objetivo es centrarnos en consultas precisas y sencillas de ejecutar, nuestra mejor opción es utilizar LINQ-to-SQL.

2.5.6.2 Ventajas de LINQ-to-SQL

LINQ proporciona un enfoque cómodo y de fácil aprendizaje para modificar y consultar datos. Admite potencialmente la consulta de diversas fuentes y tipos de datos, no solo a datos relacionales, sino también a datos XML y estructuras de datos en memoria. Según Shankar (2013), una versión resumida de algunas de las ventajas de LINQ son las siguientes:

- LINQ ofrece una sintaxis estándar para consultar diversas fuentes de datos.
- En segundo lugar, trata de mezclar los enfoques basados en datos relacionales y los orientados a objetos.

- LINQ agiliza el tiempo de desarrollo al detectar errores en tiempo de compilación e incluye soporte de IntelliSense y depuración.
- Las expresiones LINQ son de tipado fuerte, lo que permite evitar muchos errores.

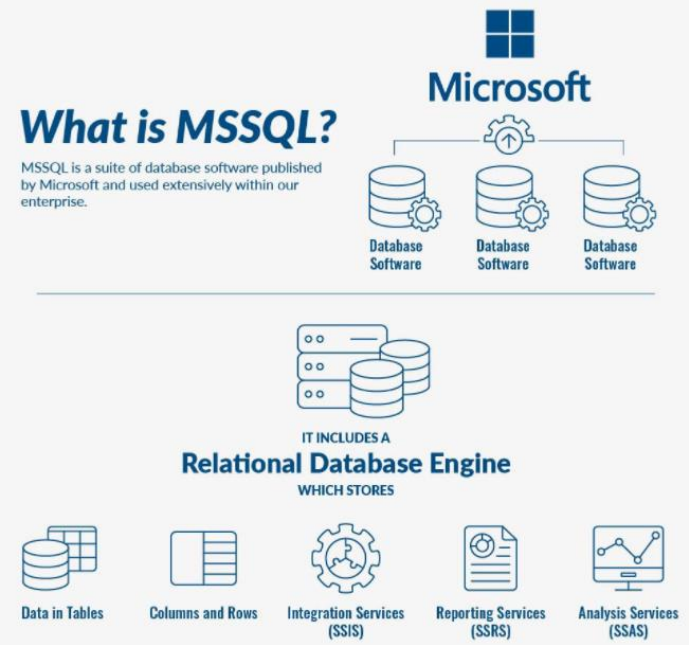
2.5.7 MSSQL Server

MSSQL es el acrónimo de Microsoft Structured Query Language, un sistema de gestión de bases de datos relacionales (RDBMS) mantenido y desarrollado por Microsoft. Fue lanzado por primera vez en 1989 y ahora está disponible en muchas versiones con diferentes características, como las ediciones Enterprise, Standard, Community y Express.

Su función principal es el almacenamiento y la recuperación de datos. Las aplicaciones que utilizan MSSQL se ejecutan en el mismo ordenador o en un servidor a través de Internet. Incluye un motor de bases de datos relacionales, el cual almacena los datos en tablas, columnas y filas. Además, también presenta una serie de herramientas:

- **Integration Services (SSIS)**, que es una herramienta de movimiento de datos para importar, exportar y transformar datos.
- **Reporting Services (SSRS)**, que se utiliza para crear informes y presentarlos a los usuarios finales.
- **Analysis Services (SSAS)**, una base de datos multidimensional que se utiliza para consultar datos desde el motor de la base de datos central.

Figura 8: ¿Qué es MSSQL?



Fuente: Microsoft (2021)

MSSQL se utiliza ampliamente en muchas implementaciones empresariales. Se trata de una plataforma de datos escalable que incluye varias herramientas de extracción, transformación y carga de datos y de elaboración de informes. (Toprek, 2020).

2.5.7.1 MSSQL vs MySQL

MySQL se trata también de una solución de gestión de bases de datos relacionales (RDBMS) de código abierto. Oracle lo compró en 2008 y desde entonces se encarga de su mantenimiento. Está detrás de algunos de los sitios web más populares del mundo, como Facebook y Twitter.

Aunque MySQL y MSSQL Server son RDBMSs muy populares, existen ciertas razones por la que este proyecto usa MSSQL en su lugar. Algunas de ellas son:

- Aunque MySQL y MSSQL Server presentan un amplio rendimiento y capacidad de escalado, ciertos datos demuestran que MSSQL Server tiene una mayor ventaja de ejecución. Según Amlanjyoti et al.(2015), MSSQL Server supera a

MySQL en pruebas que implican grandes volúmenes de consultas SELECT, INSERT, UPDATE y DELETE.

- En lo que respecta a este TFG, MSSQL es totalmente funcional con los componentes de los frameworks .NET, ya que es un producto de Microsoft. MySQL, en cambio, introduciría un requisito adicional para ser incorporado con los frameworks .NET.

Sin duda, MSSQL es la mejor opción para este proyecto sobre todo debido a la eficiencia en almacenamiento de datos y recuperación de información.

2.5.7.1 Ventajas de MSSQL Server

Toprek (2020) resume las ventajas de MSSQL en los siguientes puntos:

- **Fácil de instalar**: Microsoft SQL es fácil de usar y puede instalarse mediante un asistente de instalación. A diferencia de otros servidores de bases de datos que requieren extensas configuraciones de línea de comandos, MSSQL ofrece una interfaz de instalación fácil de usar. Además, el proceso de instalación con un solo clic viene con una interfaz gráfica de usuario junto con muchas instrucciones.
- **Rendimiento mejorado**: Gracias a las funciones integradas de compresión y encriptación transparente de datos, MSSQL ofrece un mayor rendimiento. Para asegurar y cifrar los datos, los usuarios no necesitan modificar los programas. Proporciona herramientas eficaces de gestión de permisos con controles de acceso diseñados para ayudar a los usuarios a conectar la información sensible.
- **Varias ediciones de MSSQL Server**: MSSQL se presenta en varias ediciones para satisfacer las necesidades de las empresas y de los usuarios. Las distintas ediciones varían en cuanto a características y precio. Por lo tanto, las empresas

pueden elegir la versión adecuada a sus necesidades operativas. Cabe mencionar que este TFG utiliza la edición Express.

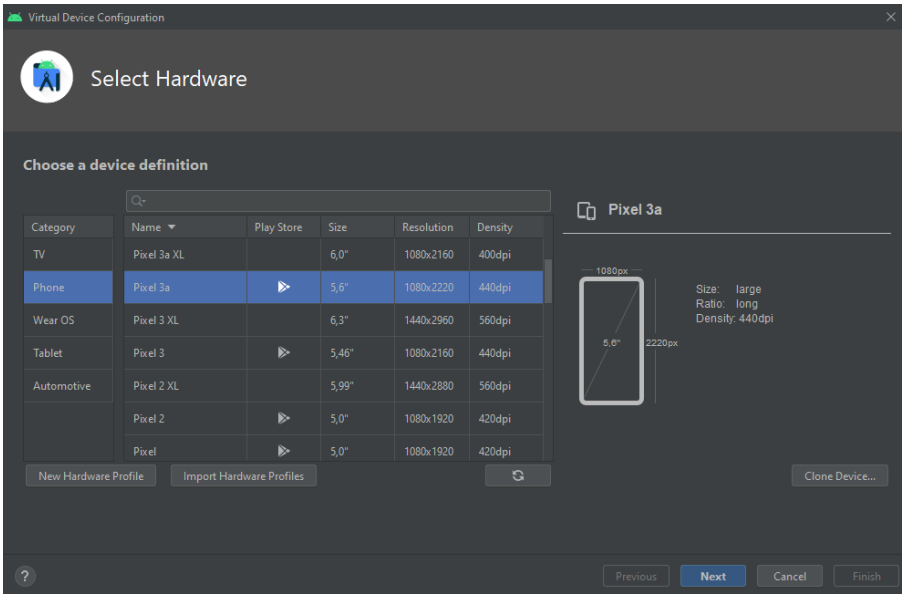
- **Altamente seguro**: MSSQL es altamente seguro y utiliza sofisticados algoritmos de encriptación, lo que hace prácticamente imposible romper las capas de seguridad. Fue desarrollado para ser una base de datos relacional comercial con características de seguridad adicionales para reducir el riesgo de ataques.
- **Excelente mecanismo de triple R**: MSSQL cuenta con un inmejorable mecanismo de recuperación, restauración y recuperación de datos (Triple-R). Consta de varias funciones sofisticadas que ayudan a restaurar y recuperar los datos perdidos, dañados o eliminados inesperadamente. Con la ayuda de herramientas de recuperación avanzadas, como ApexSQL, es posible recuperar la base de datos completa. El componente principal de SQL Server, el motor de base de datos, controla el almacenamiento de datos y ayuda a ejecutar las demandas y consultas de los usuarios, incluyendo transacciones, archivos e índices. Las grandes organizaciones suelen utilizar estas facilidades de SQL Server.

2.5.8 Simulador de Android

Un emulador de Android es una aplicación de software que permite a un dispositivo emular las características del sistema operativo Android. Se utiliza principalmente para fines de depuración.

Aunque hay muchas herramientas de depuración con el objetivo de emular el sistema operativo Android y depurar aplicaciones móviles, este TFG utiliza el simulador de Android incorporado en Android Studio.

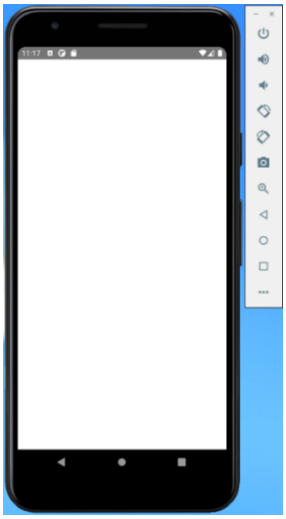
Figura 9: Selección del emulador de Android



Fuente: Elaboración propia

Con el gestor de dispositivos virtuales Android (AVD), podemos configurar el hardware de la aplicación móvil necesaria o desplegar o depurar. Un ejemplo es el hardware del Smartphone Pixel 3a, cuyo aspecto es el que se muestra a continuación:

Figura 10: Emulador Pixel 3a



Fuente: Elaboración propia

2.5.8.1 Ventajas del emulador de Android (Emulador AVD)

Para este TFG, la principal ventaja de usar este emulador es tener una depuración práctica, rápida y sencilla de la aplicación móvil y facilidad para desplegar el APK.

3

Análisis del sistema

Tras haber realizado una introducción al proyecto, estableciendo los objetivos globales, estudiar la importancia de las aplicaciones web y móvil en nuestra cultura actual y analizar las tecnologías empleadas, en este capítulo se establecerán las ideas del sistema y los requisitos necesarios para la consecución de los objetivos establecidos.

4.1 Definición del sistema

Este proyecto se encuentra dentro de un marco plenamente académico. Esto hace que el propósito del mismo se entienda desde su propia creación. En cambio, si se tratase de un proyecto en un ámbito profesional, su propósito se vería determinado mediante el análisis de una serie encuestas de usuarios, conversaciones con clientes y estudios más profundos.

Este proyecto tiene como principal propósito el desarrollo de una aplicación web y móvil segura y eficiente centradas en la gestión de los principales servicios administrativos de clínicas dentales.

4.2 Introducción al análisis de los requisitos

En esta sección de la memoria, es necesario entender una serie de requisitos que nos servirán como base para el diseño y desarrollo de nuestra aplicación. En los procesos de desarrollo de software, se trata de una de las etapas más importantes de todo el proyecto, puesto que un error en esta fase, se irá arrastrando e incrementando a medida que se desarrolla todo el proyecto.

Al realizar un análisis de requisitos, principalmente buscamos una respuesta para la siguiente pregunta: ¿qué debe hacer nuestro sistema? La respuesta será disponer de un conjunto de propiedades y funcionalidades que deben ser llevadas a cabo en las siguientes fases del proyecto cuya finalidad sea permitir alcanzar todos y cada uno de los objetivos establecidos inicialmente.

Estos requisitos podemos clasificarlos en dos grandes bloques principales:

- **Requisitos funcionales**: Estos son los requisitos que el usuario final exige específicamente como facilidades básicas que debe ofrecer el sistema. Se representan o declaran en forma de entrada que debe darse al sistema, la operación que se realiza y la salida que se espera. Básicamente, son los requisitos declarados por el usuario que se pueden ver directamente en el producto final.
- **Requisitos no funcionales**: Definen el comportamiento del sistema, las funciones y las características generales que afectan a la experiencia del usuario. La calidad de la definición y ejecución de los requisitos no funcionales determina la facilidad de uso del sistema y se utiliza para juzgar su rendimiento.

4.2.1 Requisitos funcionales

Para definir los requisitos funcionales del sistema, estos se han decidido dividir para cada uno de los usuarios de la aplicación. En nuestro caso, disponemos de dos tipos de usuarios: clientes y trabajadores.

- Requisitos para *clientes*:
    - **RF-01**. **Inicio de sesión**: La aplicación necesita de un sistema de registro e inicio de sesión (mediante DNI/NIE y contraseña) para identificar cada uno de los usuarios del sistema y disponer de un panel privado para cada uno de ellos.
    - **RF-02. Registro**: El sistema ofrece la posibilidad a los usuarios de registrarse como clientes del sistema.
    - **RF-03. Reserva de citas:** El sistema debe disponer de un mecanismo para crear, revisar, actualizar e incluso eliminar una cita en cualquier clínica dental (CRUD) con simples clics.
    - **RF-04. Visualización de facturas**: El cliente debe ser capaz de visualizar todas las facturas generadas por las clínicas dentales.
    - **RF-05. Descarga de facturas**: El sistema debe ser capaz de generar y permitir la descarga de facturas en formato pdf.
    - **RF-06. Sistema de pago en línea**: Mediante el uso de PayPal o tarjeta de crédito, el cliente puede pagar de forma online todas sus facturas pendientes.
    - **RF-07. Sistema de notificaciones**: Los clientes reciben notificaciones de recordatorio de una próxima cita o de pagos vencidos a través de SMS, correo electrónico y la propia aplicación.
    - **RF-08. Sistema de beneficiarios**: Un cliente puede añadir beneficiarios a los servicios que ofrece la clínica en su nombre.

    - **RF-09. Sistema de ayuda**: El sistema debe ofrecer la posibilidad de interaccionar con el personal de cualquier clínica a través de un sistema de tickets.

- Requisitos para *trabajadores*:
    - **RF-10. Inicio de sesión**: La aplicación necesita de un sistema de inicio de sesión (mediante DNI/NIE y contraseña) independiente de los clientes para identificar cada uno de los trabajadores del sistema.
    - **RF-11. Gestión de citas**: Gestión las citas de todos los clientes, lo que incluye crear, revisar, actualizar e incluso eliminar la cita (CRUD) con simples clics.
    - **RF-12. Gestión de facturas**: Generación y actualización de facturas de los clientes.
    - **RF-13. Gestión de cuentas de clientes**: El sistema debe ofrecer la posibilidad de actualizar toda la información relativa a los clientes por parte de los trabajadores.
    - **RF-14. Sistema de ayuda**: El sistema debe ofrecer la posibilidad de interaccionar con los clientes de cualquier clínica a través de un sistema de tickets.

4.2.2 Requisitos no funcionales

Los requisitos no funcionales podemos clasificarlos en los siguientes apartados:

- **Usabilidad**
    - **RNF-01** Mostrar de forma clara el lugar donde se encuentra el usuario.
    - **RNF-02** Barra lateral para una navegación rápida y sencilla.
    - **RNF-03** Diseño responsive para garantizar la adecuada visualización de la aplicación en diferentes dispositivos.
    - **RNF-04** Muestra de errores informativos y orientados al usuario.

- **Seguridad**
    - **RNF-06** Gestión de las diferentes sesiones de usuario de forma segura.
    - **RNF-07** Sistema de manejo de peticiones de clientes al servidor de forma segura.
    - **RNF-08** Los campos de las contraseñas se ocultarán tras asteriscos para protegerla.
    - **RNF-09** Los permisos de acceso al sistema podrán ser cambiados únicamente por el administrador de acceso a datos.
    - **RNF-10** Asegurar que los datos estén protegidos del acceso no autorizado
- **Rendimiento**
    - **RNF-11** Soportar conexiones simultáneas al sistema.
    - **RNF-12** Disponer de una respuesta rápida para la carga de recursos en el menor tiempo posible.
- **Fiabilidad**
    - **RNF-13** Sistema de notificaciones de errores dirigidas al usuario.
    - **RNF-14** Uso de la metodología try-catch para el manejo de cualquier tipo de error y evitar la inoperatividad del sistema.

4.3 Principales casos de uso

El modelado de casos de uso se utiliza en el análisis de sistemas para identificar, aclarar y organizar los requisitos establecidos. Un caso de uso está formado por un conjunto de posibles secuencias de acciones entre los usuarios y el sistema en un entorno concreto, todo ello con un objetivo determinado.

Todo caso de uso contiene tres elementos esenciales:

- Actor: El usuario del sistema, que puede ser una sola persona o un grupo de personas que interactúan con el proceso. En este proyecto disponemos de dos actores: clientes y trabajadores.
- Objetivo: El resultado final que completa el proceso.
- Sistema: El proceso y los pasos dados para alcanzar el objetivo final, incluyendo los requisitos funcionales necesarios y sus comportamientos previstos.

Debido a la complejidad que presenta todo el proyecto, vamos a desarrollar los casos de uso más importantes del sistema:

Tabla 5: Caso de uso – Nueva cita

Caso de uso – Nueva cita	
Actores principales	Cliente, Trabajador
Precondiciones	1. El cliente o trabajador están registrados en el sistema. 2. El cliente o trabajador han iniciado sesión en el sistema.
Secuencia normal	1. En el panel de usuario (dashboard), el cliente o trabajador accede al apartado de Calendario de citas. 2. El cliente selecciona la fecha y hora deseadas junto con una descripción, el beneficiario y la clínica deseada y obtiene una cita. El trabajador, además de esto, puede seleccionar al cliente que desea sacar una nueva cita. 3. El sistema comprueba que la cita es válida y envía confirmación al usuario.
Postcondiciones	El cliente, a través de su panel de usuario o a través de un trabajador, obtiene una nueva cita en una determinada clínica.

Fuente: Elaboración propia

Tabla 4: Caso de uso – Nueva cita rápida

Caso de uso – Nueva cita rápida	
Actores principales	Cliente
Precondiciones	Ninguna
Secuencia normal	1. El cliente accede a la página principal. 2. Rellena el formulario para la solicitud de cita rápida con todos sus datos. 3. Posteriormente, comprueba si los datos introducidos se corresponden con algún usuario registrado. a. Si el usuario no está registrado, se genera una cuenta con el DNI y contraseña aleatoria que se notifica por email. El sistema inicia sesión al usuario y se genera una nueva cita. b. Si el usuario está registrado, se redirige a la página de inicio de sesión y se guardan los datos de la cita solicitada. Cuando el usuario inicia sesión, se genera una nueva cita.
Postcondiciones	El cliente, registrado previamente o no, pasa a iniciar sesión en el sistema y se genera una nueva cita de forma sencilla y rápida.

Fuente: Elaboración propia

Tabla 6: Caso de uso – Nueva factura

Caso de uso – Nueva factura	
Actores principales	Trabajador
Precondiciones	1. El trabajador está registrado en el sistema. 2. El trabajador ha iniciado sesión en el sistema.
Secuencia normal	1. En el panel de usuario (dashboard), el trabajador accede al apartado de Facturas. 2. El trabajador rellena todos los campos necesarios para generar una factura como beneficiario, cliente, clínica, conceptos, descuentos,... 3. El trabajador acepta la factura y el sistema comprueba que la cita es válida y notifica al cliente.
Postcondiciones	El cliente, a través de su panel de usuario, recibe una nueva factura pendiente de pago y notificación.

Fuente: Elaboración propia

Tabla 7: Caso de uso – Cancelar cita

Caso de uso – Cancelar cita	
Actores principales	Cliente, Trabajador
Precondiciones	1. El cliente o trabajador están registrados en el sistema. 2. El cliente o trabajador han iniciado sesión en el sistema.
Secuencia normal	1. En el panel de usuario (dashboard), el cliente o trabajador accede al apartado de Calendario de citas. 2. El sistema muestra la lista de citas de un determinado cliente o trabajador. 3. El cliente o trabajador pueden cancelar de la lista una determinada cita pulsando el botón de la papelera.
Postcondiciones	El sistema elimina una determinada cita del listado completo de citas y notifica al cliente.

Fuente: Elaboración propia

Tabla 8: Caso de uso – Modificar una factura

Caso de uso – Modificar una factura	
Actores principales	Trabajador
Precondiciones	1. El trabajador está registrado en el sistema. 2. El trabajador ha iniciado sesión en el sistema.
Secuencia normal	1. En el panel de usuario (dashboard), el trabajador accede al apartado de Facturas. 2. El trabajador visualiza todas las facturas que pertenecen a sus clínicas. 3. El trabajador puede editar la mayor parte de los campos de la factura (Descripción, beneficiario, clínica a la que pertenece, estado,...) haciendo click en el botón de editar.
Postcondiciones	El cliente recibe una notificación tras la modificación de los campos de una factura por parte del trabajador.

Fuente: Elaboración propia

Tabla 9: Caso de uso- Pagar una factura

Caso de uso – Pagar una factura	
Actores principales	Cliente
Precondiciones	1. El cliente está registrado en el sistema. 2. El cliente ha iniciado sesión en el sistema. 3. El cliente dispone de facturas pendientes.
Secuencia normal	1. En el panel de usuario (dashboard), el cliente accede al apartado de Facturas. 2. El cliente visualiza todas sus facturas. 3. El cliente puede pagar una factura pendiente haciendo clic al botón de Pagar ahora. 4. La sistema abre un popup con los diferentes métodos de pago (En este caso, únicamente Paypal). 5. El cliente inicia sesión en su cuenta de Paypal y realizar todo el proceso de pago.
Postcondiciones	El cliente recibe una notificación tras el correcto o incorrecto pago de la factura. Si todo es correcto, el sistema modifica el estado de la factura a Pagado.

Fuente: Elaboración propia

4

El diseño del proyecto

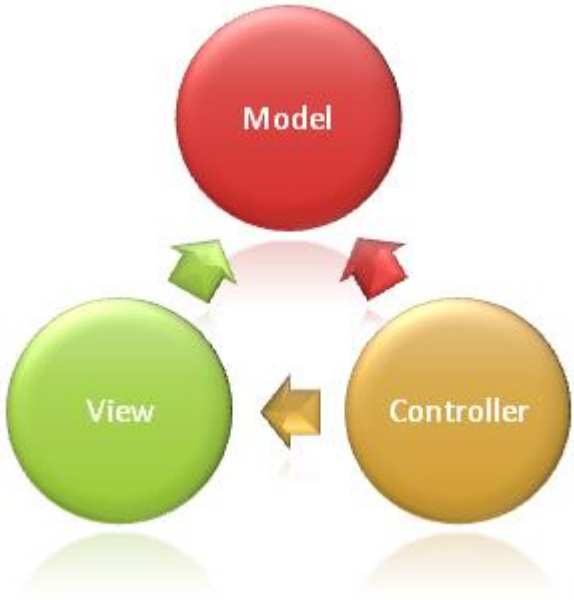
En este capítulo se explicará el diseño del sistema. Para comprender todo el diseño, se utilizarán gráficos para mostrar y especificar de forma sencilla los componentes que conforman la solución completa del proyecto.

4.1 Arquitectura del proyecto

La arquitectura principal del sistema se basa en el patrón Modelo-Vista-Controlador (MVC).

El patrón Modelo-Vista-Controlador permite separar una aplicación en tres grandes grupos de componentes: Modelos, Vistas y Controladores. Con este patrón, las solicitudes del usuario se enrutan a un controlador que se encarga de trabajar con el modelo para realizar las acciones del usuario o recuperar los resultados de consultas. El controlador elige la vista para mostrar al usuario y proporciona cualquier dato de modelo que sea necesario (Microsoft, 2021).

Figura 11: Patrón Modelo-Vista-Controlador



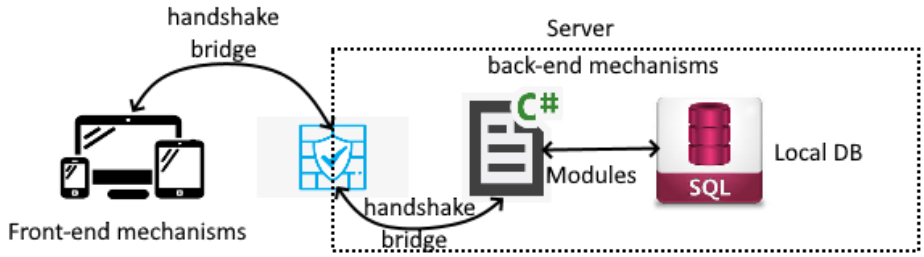
Fuente: Microsoft (2021)

Microsoft (2021) define específicamente cada uno de estos componentes de la siguiente forma:

- El modelo en una aplicación de MVC representa el estado de la aplicación y cualquier lógica de negocios u operaciones que esta deba realizar. La lógica de negocio debe encapsularse en el modelo, junto con cualquier lógica de implementación para conservar el estado de la aplicación.
- Las vistas se encargan de presentar el contenido a través de la interfaz de usuario.
- Los controladores actúan como un puente entre el modelo y la vista para procesar toda la lógica de negocio y las peticiones entrantes, manipular los datos utilizando el modelo e interactuar con las vistas para renderizar la salida final.

En la siguiente figura, podemos ver a alto nivel cada uno de estos componentes:

Figura 12: Mecanismo de conexión entre front-end y back-end

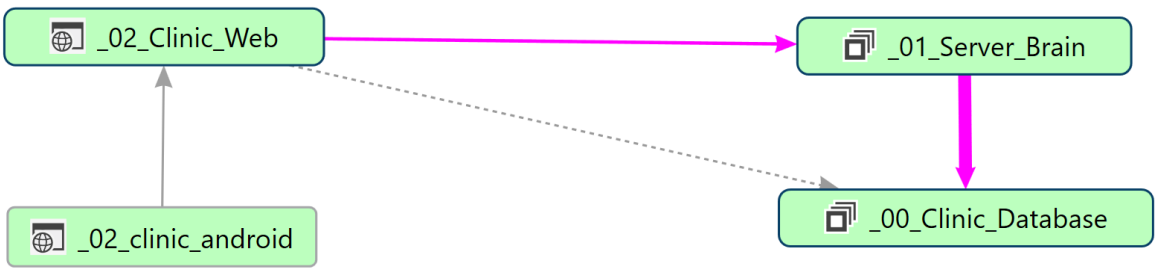


Fuente: Elaboración propia

- Los mecanismos del front-end representan a las vistas del sistema y funcionalidades del lado del cliente.
- El puente de conexión (handshake) entre el front-end y el back-end representa al controlador.
- Los mecanismos del back-end representan el modelo de la aplicación.

Para realizar una implementación rápida, sencilla y eficiente de cara a futuras mejoras, el proyecto completo se divide en diferentes subproyectos. La relación entre estos permite el correcto funcionamiento del sistema. Siguiendo este patrón y estructura, en la siguiente imagen se representan cada uno de los componentes que conforman el sistema:

Figura 13: Componentes y relaciones del sistema



Fuente: Elaboración propia

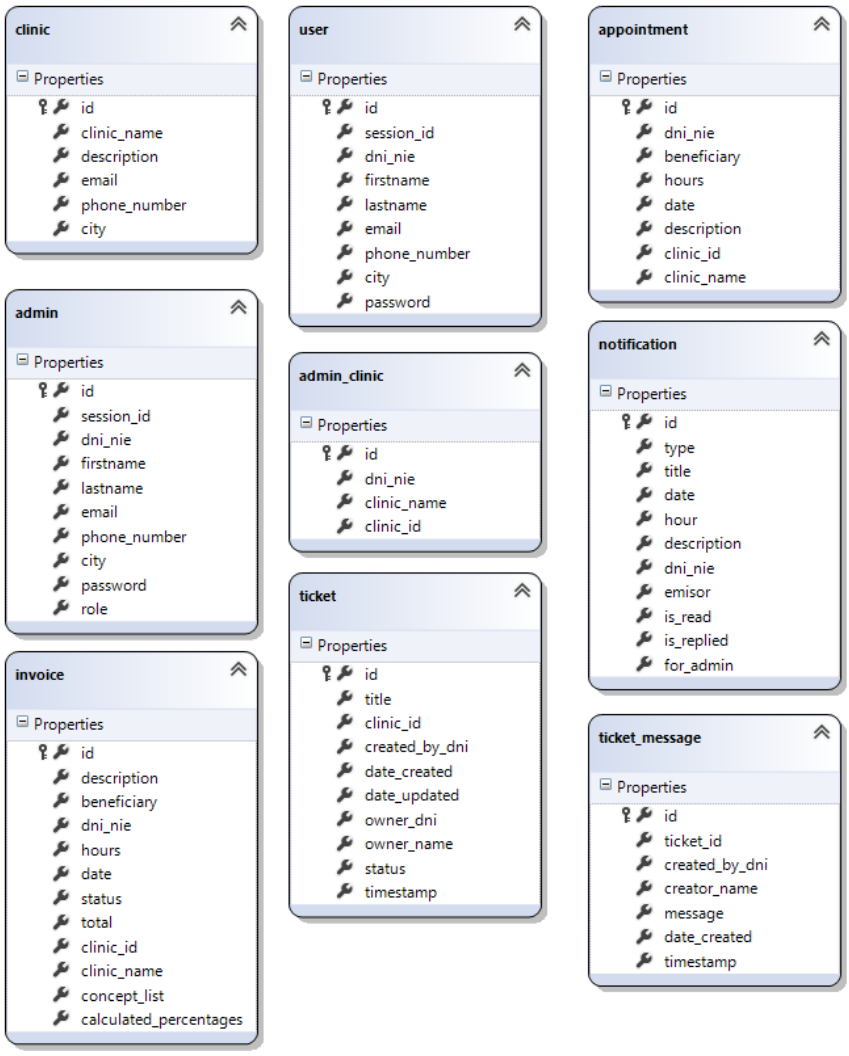
A continuación vamos a explicar cada uno de estos proyectos y la relaciones que existen entre ellos.

4.2 El proyecto 00_Clinic_Database

El proyecto 00_Clinic_Database alberga la base de datos relacional utilizada para el almacenamiento de datos. Este proyecto no depende del resto. Se trata de uno de los componentes que conforman el modelo del sistema.

La base de datos relacional del sistema contiene una serie de tablas (relaciones) que permiten el correcto almacenamiento de datos del sistema. Gracias al uso del generador de diagramas de la base de datos de LINQ, podemos representar todas estas tablas en la siguiente imagen:

Figura 14: Modelo de tablas de la base de datos



Fuente: Elaboración propia

Para entender cada una de estas tablas, es importante realizar una breve descripción de cada una de estas:

- Tabla ***clinic***: Contiene todos los datos relacionados con las clínicas dentales del sistema.
- Tabla ***user***: Almacena toda la información relativa a los clientes. Gracias al DNI y la contraseña, un cliente puede acceder al sistema.
- Tabla ***appointment***: Está formada por los campos que representan a una cita de una clínica determinada para un usuario concreto.
- Tabla ***admin***: Almacena toda la información relativa a los trabajadores. Gracias al DNI y la contraseña, el trabajador puede acceder al sistema.
- Tabla ***admin_clinic***: Contiene la información sobre las clínicas dentales a las que pertenece un trabajador determinado.
- Tabla ***notification***: Guarda toda la información relativa a la notificación realizada a clientes y trabajadores dentro del sistema y a través de correo electrónico y SMS.
- Tabla ***invoice***: Almacena la información relativa a las facturas del sistema, desde el usuario receptor de la factura hasta el listado de conceptos a facturar y porcentajes de descuentos e impuestos.
- Tabla ***ticket***: Contiene la información relativa a los tickets de ayuda que se generan en el sistema. Se ordenan por fecha de creación.
- Tabla ***ticket_message***: Representa cada uno de los mensajes que se encuentran dentro de un ticket en concreto. Se ordenan por fecha de creación.

Todas estas tablas (relaciones) permiten representar perfectamente los datos necesarios para el sistema.

5.2.1 ¿Por qué decimos que una tabla es una relación?

Según Chapple(2021), cabe mencionar que una relación de base de datos no es lo mismo que una base de datos relacional. No implica una relación entre tablas, a pesar de su nombre. Más bien, una relación de base de datos se refiere a una tabla individual en una base de datos relacional. En una base de datos relacional, la tabla es una relación porque almacena la relación entre los datos en su formato columna-fila. Las columnas son los atributos de la tabla y las filas representan los registros de datos.

4.3 El proyecto 01_Server_Brain

El proyecto 01_Server_Brain, como su nombre indica, es el cerebro de todo el proyecto. En él se encuentran las distintas clases y módulos que contienen los algoritmos esenciales para el buen funcionamiento de la aplicación. Este proyeto se encarga de realizar todas las conexiones a la base de datos para recuperar y actualizar información dependiendo de las diferentes peticiones del usuario. Cabe mencionar que todas estas peticiones del lado del cliente suelen requerir una o varias de operaciones CRUD (Create, Read, Update, Delete) en la base de datos, que implican técnicas de recuperación y manipulación de la información. Este proyecto también forma parte del modelo del sistema.

Como se muestra en la siguiente figura, la estructura completa de este proyecto comprende 11 clases, las cuales podemos explicar brevemente:

Figura 15: Diagrama de clases del proyecto 01_Server_Brain

Fuente: Elaboración propia

4.3.2 La clase *Authentication*

Como su nombre indica, esta clase contiene las funciones relacionadas con la autenticación de los usuarios. Tiene módulos que verifican a los usuarios (clientes), registran y actualizan sus datos. Esta clase también posee las mismas funciones para los *trabajadores* (denominados *administradores* en el código).

4.3.3 La clase *Calendar*

Este TFG utiliza el término *Calendar* de forma abstracta para hacer referencia a todo el mecanismo relacionado con las citas. Esta clase contiene los algoritmos necesarios para la gestión de las citas de los clientes, desde su creación hasta su eliminación.

4.3.4 La clase *Client*

Esta clase maneja los principales algoritmos para obtener todos los usuarios e información sobre sus actividades así como número de citas, facturas y clínicas a las que pertenece un determinado cliente.

4.3.5 La clase *Clinics*

Esta clase contiene los principales métodos para obtener un listado de todas las clínicas dependiendo del tipo de usuario que la requiere (trabajador o clientes).

4.3.6 La clase *Emailer*

La clase Emailer, como su nombre indica, maneja todo tipo de mecanismos de envío de correo electrónico basados en el tipo notificación que se desea llevar a cabo.

4.3.7 La clase *Invoices*

Invoices es la clase que contiene los algoritmos relacionados con la creación, lectura, actualización, y eliminación de las facturas del sistema así como el control del sistema de pago.

4.3.8 La clase *Notificacions*

La clase de notificación, como su nombre indica, contiene todos los mecanismos de CRUD de todo el proyecto. Hay diez tipos de notificaciones en esta clase.

4.3.9 La clase *Post_connect_controller*

Esta clase es una de las más importantes de todo el proyecto. Contiene la función que establece y controla el canal de comunicación seguro entre clientes y servidor. Más adelante discutiremos esta clase en detalle en el capítulo 6.

4.3.10 La clase *SessionDetails*

Esta clase gestiona los detalles de la sesión de los usuarios y genera nuevas claves de sesión para nuevos usuarios. Además, se encarga de comprobar las sesiones actuales en el sistema.

4.3.11 La clase *SMS_Sender*

Como su nombre indica, esta clase maneja las metodologías y algoritmos de envío de notificaciones SMS en función de los tipos de notificación.

4.3.12 La clase *Tickets*

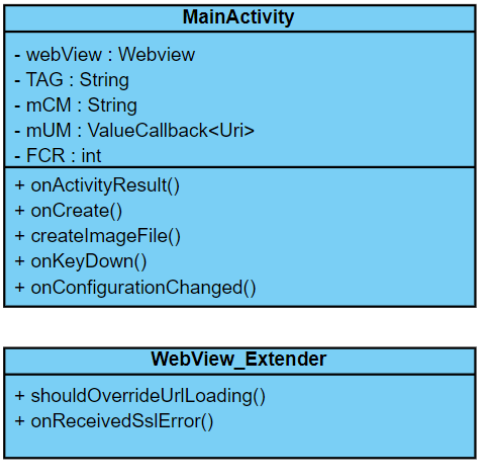
La clase Ticket contiene todas las funciones de CRUD para la gestión del sistema de tickets (sistema de ayuda). Los usuarios pueden generar estos tickets para una determinada clínica. Estos tickets están formados por diversos mensajes ordenados por fecha de creación.

4.4 El proyecto 02_Clinic_Android

El proyecto de la aplicación Android se trata de un proyecto Java generado desde el IDE de Android Studio. Contiene las funciones necesarias para la generación de la aplicación móvil. La idea principal de este proyecto es la reutilización de las vistas, diseño y funciones de la aplicación web con el objetivo de generar una aplicación móvil en Android sin tener que reescribir todo el proyecto desde cero.

Para conseguir esto, será necesaria la creación de una serie de clases que implementen la funcionalidad de la aplicación y un Webview, lo que permitirá lograr nuestro objetivo con esta aplicación móvil. Para conseguir esto, serán necesarias las siguientes clases:

Figura 16: Diagrama de clases del proyecto 02_Clinic_Android



Fuente: Elaboración propia

Más adelante, en el capítulo de desarrollo, se comentarán más detalles acerca del componente WebView y su implementación.

4.5 El proyecto 02_Clinic_Web

Este proyecto contiene los archivos necesarios para la creación de la aplicación web. Se trata de uno de los principales proyectos ya que contiene los componentes esenciales que conforman el núcleo de la aplicación. Este proyecto depende principalmente de los proyectos 00_Clinic_Database y 01_Server_Brain, es decir, el modelo del sistema. Además, este proyecto contiene las vistas y controladores del sistema.

La aplicación web gestiona las peticiones del lado del cliente y necesita consultar la base de datos para la recuperación de datos con relativa frecuencia. Asimismo, depende directamente de las funcionalidades del back-end para ejecutar el algoritmo necesario que realiza estas peticiones a la base de datos.

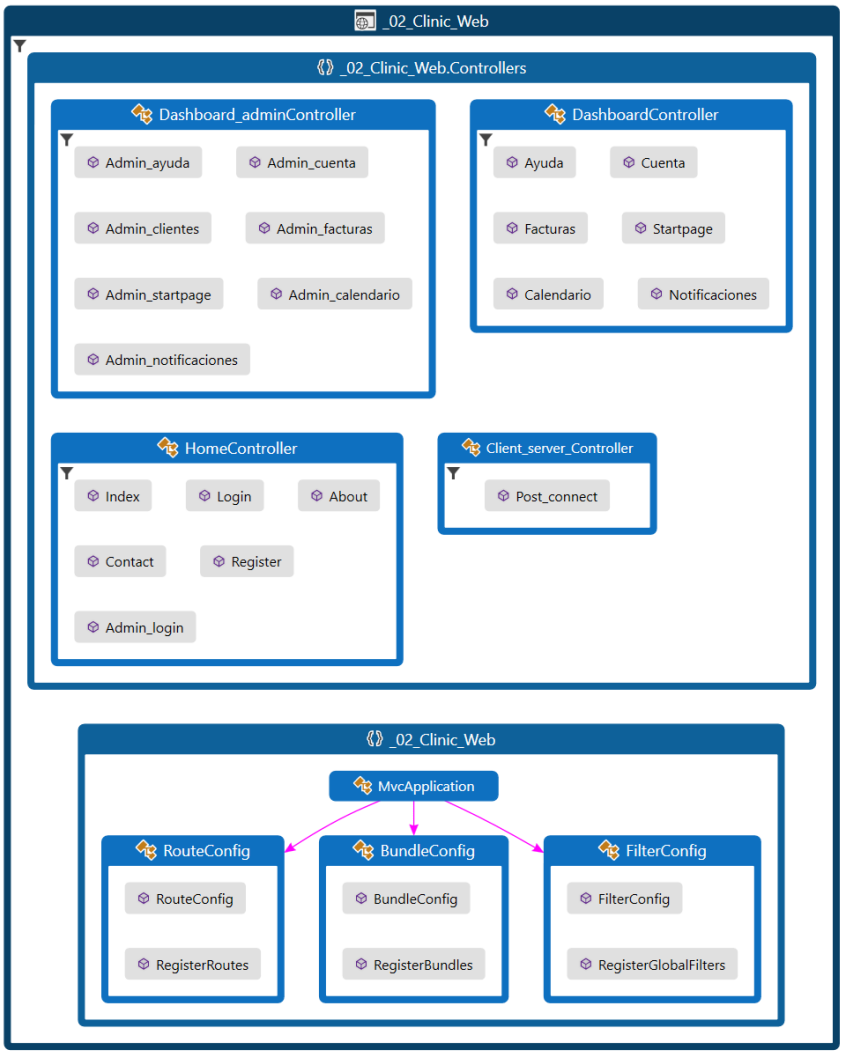
El sistema constará de tres bloques de vistas principales:

- **Home**: Contiene las vistas generales para cualquier usuario:
  - **Página de inicio**: Información general del sistema.
  - **Página de inicio de sesión de clientes**: Inicio del sesión para clientes.
  - **Página de inicio de sesión de trabajadores**: Inicio de sesión para trabajadores.
  - **Página de registro**: Registro para nuevos clientes.
- **Dashboard**: Contiene las vistas del panel del cliente cuando inicia sesión en el sistema**:**
  - **Página de inicio del panel**: Resumen del listado de citas, facturas y otros indicadores del sistema relativos al cliente.
  - **Página de facturas**: Listado de todas las facturas del cliente.
  - **Página de citas**: Calendario de citas y listado de las citas reservadas para el cliente.
  - **Página de notificaciones**: Listado de todas las notificaciones relativas al cliente.
  - **Página de ayuda**: Listado de los tickets de ayuda abiertos por el cliente.
  - **Página de preferencias**: Página donde el cliente puede editar sus datos personales.
- **Dashboard_admin:** Contiene las vistas del panel del trabajador cuando inicia sesión en el sistema**:**
  - **Página de inicio del panel**: Resumen del listado de citas, facturas y otros indicadores del sistema relativos a las clínicas donde pertenece el trabajador.
  - **Página de facturas**: Listado de las facturas de todos los clientes de las clínicas donde pertenece el trabajador..

- **Página de citas**: Calendario de citas y listado de las citas reservadas para todos los clientes de las clínicas donde pertenece el trabajador.
- **Página de notificaciones**: Listado de todas las notificaciones relativas al trabajador.
- **Página de ayuda**: Listado de los tickets de ayuda abiertos por el cliente relativos a una determinada clínica donde pertenece el trabajador..
- **Página de preferencias**: Página donde el cliente puede editar sus datos personales.

Cada una de estas vistas presentan una funcionalidad determinada. Gracias a estas funcionalidades, los usuarios pueden interactuar con el sistema y enviar peticiones al back-end. Para ello, este proyecto dispone de los controladores necesarios para el renderizado de vistas y envío y recepción de peticiones al back-end (handshake):

Figura 17: Estructura del controlador del proyecto



Fuente: Elaboración propia

Los controladores *Dashboard_adminController*, *DashboardController* y *HomeController* son los encargados de renderizar cada una de las diferentes vistas de cada uno de los distintos bloques de vistas cuando un usuario accede a una determinada página de la aplicación. El controlador principal *Client_server_Controller* se encarga de gestionar todas las peticiones entre el front-end y back-end.

Este proyecto es el núcleo de esta aplicación web. El correcto funcionamiento de todos y cada uno de los diferentes componentes de este proyecto junto con los proyectos del servidor permiten el correcto funcionamiento del sistema.

5

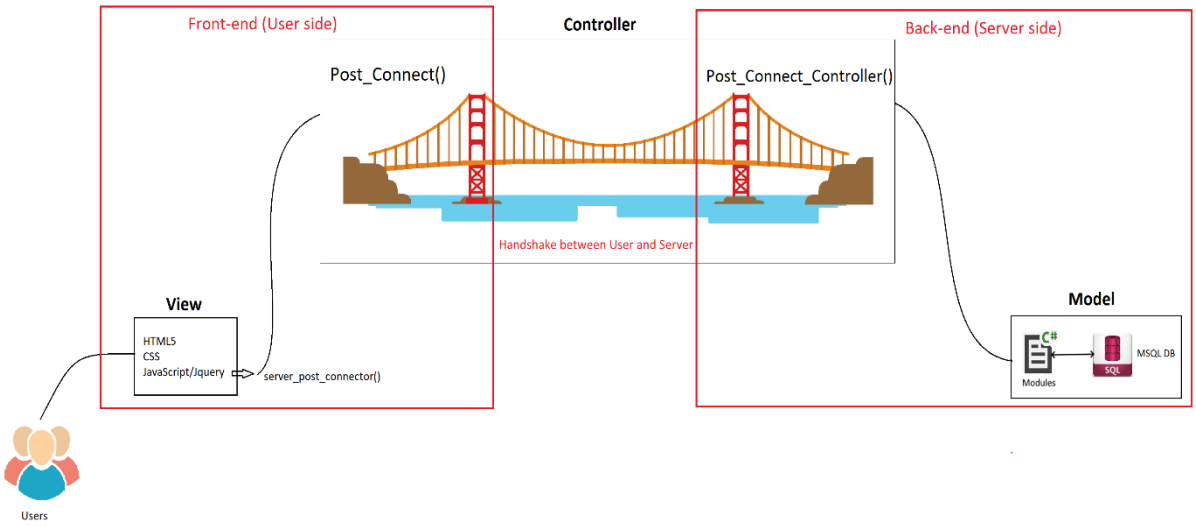
El desarrollo del proyecto

En este capítulo se detallará de forma clara las estrategias de desarrollo del front-end, el back-end y el handshake seguro entre el cliente y el servidor de la aplicación web y móvil. Además, se explicará como se conectan cada una de estas partes entre si. En este capítulo también se comentarán algunos de los retos encontrados durante el desarrollo y cómo se resolvieron.

5.1 Introducción al desarrollo del sistema

Como hemos comentado en el capítulo anterior, el proyecto sigue el patrón Modelo-Vista-Controlador (MVC). Para explicar el desarrollo del proyecto, es necesario discutir el desarrollo de cada uno de los componentes de este sistema. Para ello, vamos a mostrar un esquema sencillo que representa la arquitectura general del sistema:

Figura 18: Esquema sencillo de la arquitectura del sistema



Fuente: Elaboración propia

En la figura superior podemos analizar los siguientes elementos:

- El puente central (handshake) representa el camino entre las peticiones y respuestas del front-end y back-end. Esta conexión es llevada a cabo por el controlador.
- La parte izquierda del puente representa al desarrollo front-end del sistema (Vistas, estilos y funciones).
- La parte derecha del puente representa el desarrollo back-end del sistema (Lógica del modelo y datos).

Para entender detenidamente cada una de estas secciones, vamos a separar la explicación del desarrollo en tres partes:

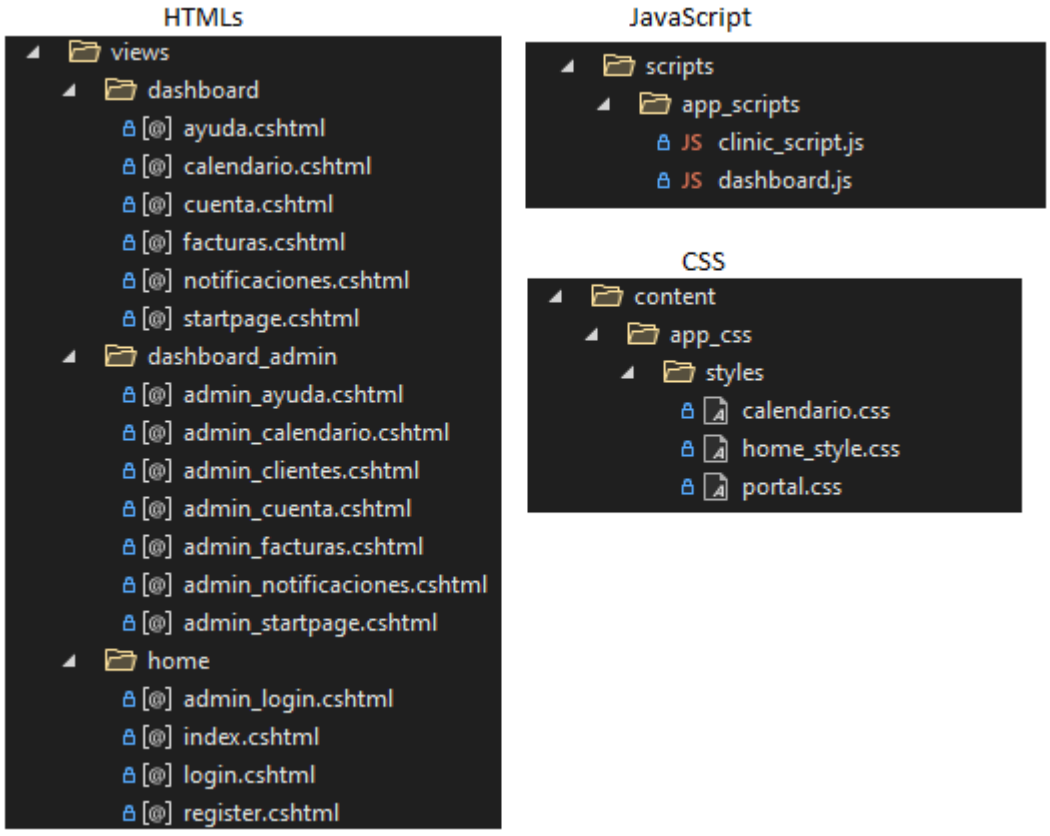
- Desarrollo del front-end
- Desarrollo del back-end
- Desarrollo del puente cliente-servidor (Handshake)

5.1 El desarrollo del front-end.

El desarrollo front-end, a veces llamado desarrollo del lado del cliente, implica la participación de HTML, CSS y JavaScript para producir una aplicación web o móvil donde los usuarios vean e interactúen con el sistema. En otras palabras, el desarrollo front-end es el desarrollo de la interfaz gráfica de usuario (GUI) de una aplicación web o móvil.

En este proyecto, los componentes del desarrollo del front-end que construyen la interfaz del usuario y funcionalidades del lado del cliente se encuentran en los siguientes directorios del proyecto 02_Clinic_Web:

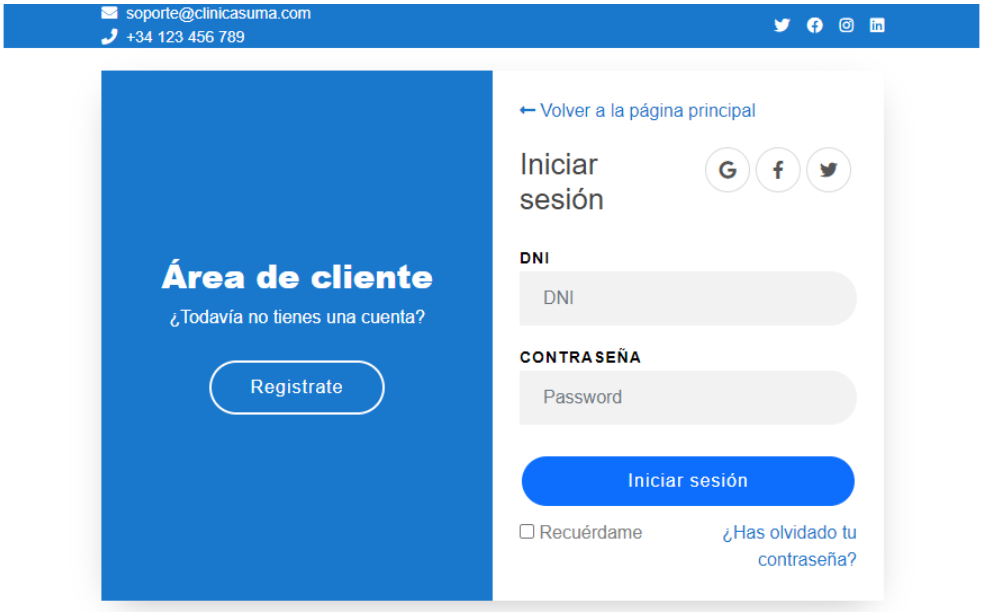
Figura 19: Estructura de carpetas del proyecto 02_Clinic_Web



Fuente: Elaboración propia

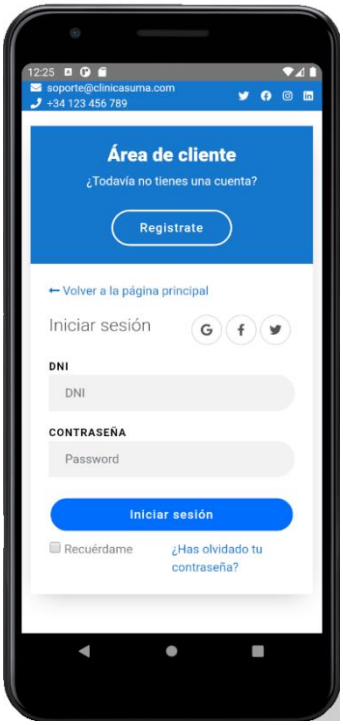
Estos componentes construyen la estructura, diseño y funcionalidades que componen la vista del sistema. Por ejemplo, en las siguientes imágenes se muestra una de las muchas vistas que conforman la aplicación:

Figura 20: Vista de la página de inicio de sesión (Versión web)



Fuente: Elaboración propia

Figura 21: Vista de la página de inicio de sesión (Versión móvil)



Fuente: Elaboración propia

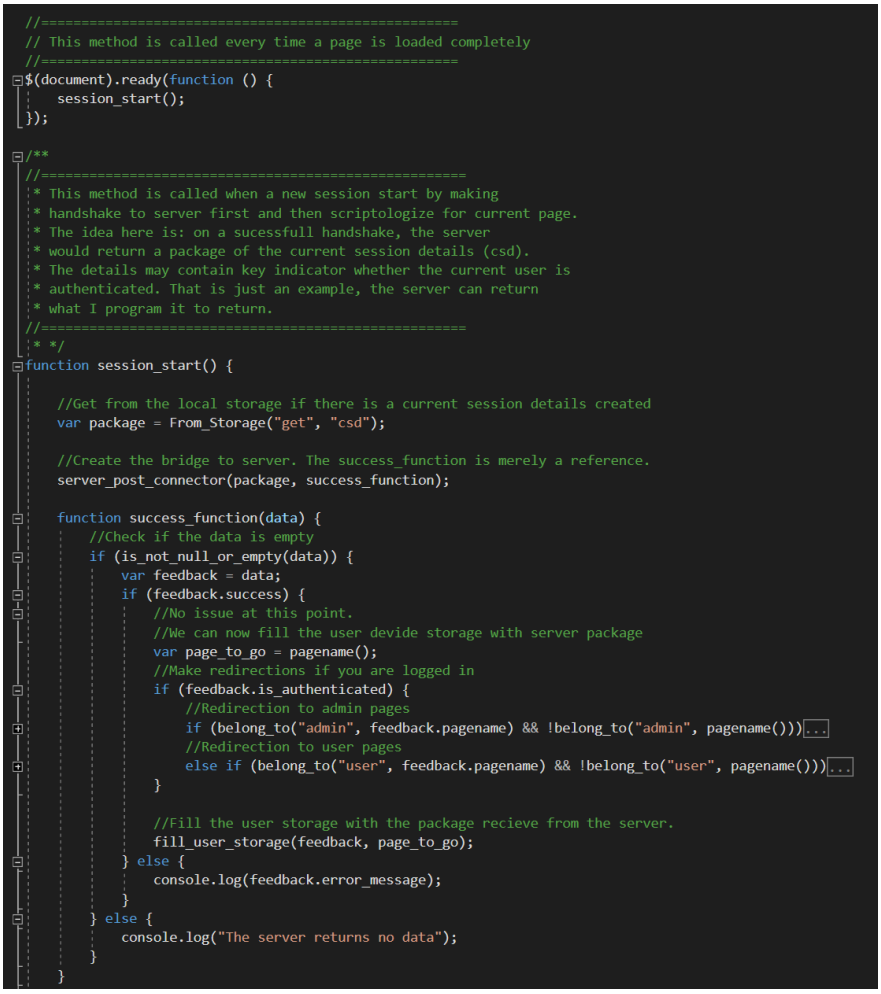
5.1.1 Funcionamiento principal del front-end

Inicialmente, todas y cada una de las funciones del lado del cliente fueron implementadas utilizando un script por página existente. Esto provocó una gran sobrecarga al sistema cuando se deseaba acceder a alguna de las páginas puesto el sistema se encontraba constantemente cargando scripts. Además, la gestión de todos estos scripts se volvía desordenada y prácticamente imposible de gestionar.

Debido a esto, se tomó la decisión de utilizar un único script (*clinic_script.js*) para realizar la gestión del lado del cliente. Este script presenta una estructura muy peculiar y eficiente que permite establecer todos los mecanismos de interacción entre usuario-cliente y cliente-servidor. Esta estructura es la siguiente:

- Cada vez que un usuario accede a una de las páginas, se da comienzo a una sesión. Primero, se lleva a cabo un handshake entre el cliente y servidor. Si todo ocurre de forma exitosa, el servidor devuelve un paquete con los detalles de la sesión actual. Por ejemplo, entre estos detalles se incluye información clave para conocer si el usuario se encuentra autenticado o no, entre otros.

Figura 22: Establecimiento de la sesión del usuario



```
//=================================================
// This method is called every time a page is loaded completely
//=================================================
$(document).ready(function () {
    session_start();
});

/**
//=================================================
* This method is called when a new session start by making
* handshake to server first and then scriptologize for current page.
* The idea here is: on a sucessfull handshake, the server
* would return a package of the current session details (csd).
* The details may contain key indicator whether the current user is
* authenticated. That is just an example, the server can return
* what I program it to return.
//=================================================
* */
function session_start() {

    //Get from the local storage if there is a current session details created
    var package = From_Storage("get", "csd");

    //Create the bridge to server. The success_function is merely a reference.
    server_post_connector(package, success_function);

    function success_function(data) {
        //Check if the data is empty
        if (is_not_null_or_empty(data)) {
            var feedback = data;
            if (feedback.success) {
                //No issue at this point.
                //We can now fill the user devide storage with server package
                var page_to_go = pagename();
                //Make redirections if you are logged in
                if (feedback.is_authenticated) {
                    //Redirection to admin pages
                    if (belong_to("admin", feedback.pagename) && !belong_to("admin", pagename())) ...
                    //Redirection to user pages
                    else if (belong_to("user", feedback.pagename) && !belong_to("user", pagename())) ...
                }

                //Fill the user storage with the package recieve from the server.
                fill_user_storage(feedback, page_to_go);
            } else {
                console.log(feedback.error_message);
            }
        } else {
            console.log("The server returns no data");
        }
    }
```

Fuente: Elaboración propia

- Tras generar una nueva sesión o recuperar una sesión previamente establecida, el script detecta la página en la que nos encontramos actualmente y se ejecutan únicamente los métodos definidos para esa página en concreto. Así, en un mismo script se encuentra estructurada las funcionalidades que cada página requiere de una forma visual y sencilla de controlar. Esta estructura permite la detección rápida de errores y la escalabilidad del sistema.

Figura 23: Estructura del script

```
/**
//=================================================
* This function checks the page we are currently on and
* calls the functions used in that specific page.
//=================================================
* */
function prepare_scripts() {
    switch (pagename()) {
        case "home":
            use_home_script();
            use_calendar_script();
            break;
        case "login":
            use_login_script();
            break;
        case "register":
            use_register_script();
            break;
        default: otherpages(); break;
    }

    //Admin Dashboard pages or Dashboard pages
    function otherpages() {

        //Admin Dashboard Scripts
        if (pagename().includes("admin_"))... else { //User Dashboard script
            use_dashboard_script();
            switch (pagename()) {
                case "startpage":
                    use_invoices_script();
                    use_calendar_script();
                    check_unread_notifications();
                    break;
                case "calendario":
                    use_calendar_script();
                    check_unread_notifications();
                    break;
                case "cuenta":
                    use_account_script();
                    check_unread_notifications();
                    break;
```

Fuente: Elaboración propia

Las funciones de cada página establecen las funcionalidades base para la creación de eventos como clics a botones, inyección de código HTML, modificación del diseño, etc. Además, se encargan de realizar las peticiones al servidor:

- Cada una de las peticiones que el cliente desea realizar al back-end se realizan mediante la función de la figura 24, la cual se encarga de empaquetar todos los datos y los envía utilizando el controlador.

Figura 24: Función para el envio de peticiones del lado del cliente

```
/**
//=======================================
* This function connect the clients request to the server
* Every client request is packed in a formdata unproccessed,
* in a dictionary format, passed in as parameter.
* The success parameter is a reference to a function that is
* excuted after a successful connect.
* The processing takes place in the server side.
* I intend to use only one action name for the mvc action result.
* =======================================================
* @param {Dictionary} dictionary_parameters : the parameters in dictionary format
* @param {Function} success_function : the function trigger that is executed on success
*/
function server_post_connector(dictionary_parameters, success_function)...
```

Fuente: Elaboración propia

En la sección que se centra en el desarrollo del puente cliente-servidor se explicarán más detalles sobre esta importante función.

5.1.2 Aplicación móvil y adaptabilidad del sistema

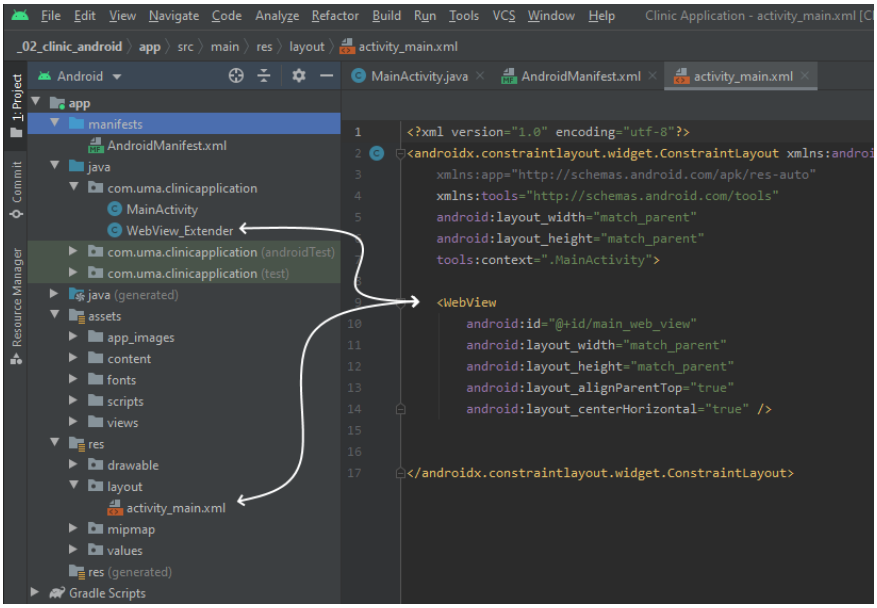
La idea principal de esta aplicación consiste en compartir todas y cada una de las vistas y archivos auxiliares del proyecto web para que el comportamiento y apariencia de la aplicación web se mantengan en la aplicación Android. Con este componente, no se necesita desarrollar todo el contenido de la aplicación de forma repetida, solo es necesario desarrollarlo una única vez.

Esto se consigue principalmente gracias al uso de un Webview. Android WebView es un componente del sistema con la tecnología del navegador Chromium que permite que las aplicaciones de Android puedan mostrar contenido web.

Para que todo este sistema funcione adecuadamente, se han realizado las siguientes acciones:

- Construimos la implementación de la Webview en el directorio /res/layout del proyecto Android y se añaden la referencia del Webview en la construcción de la aplicación, como se muestra a continuación.

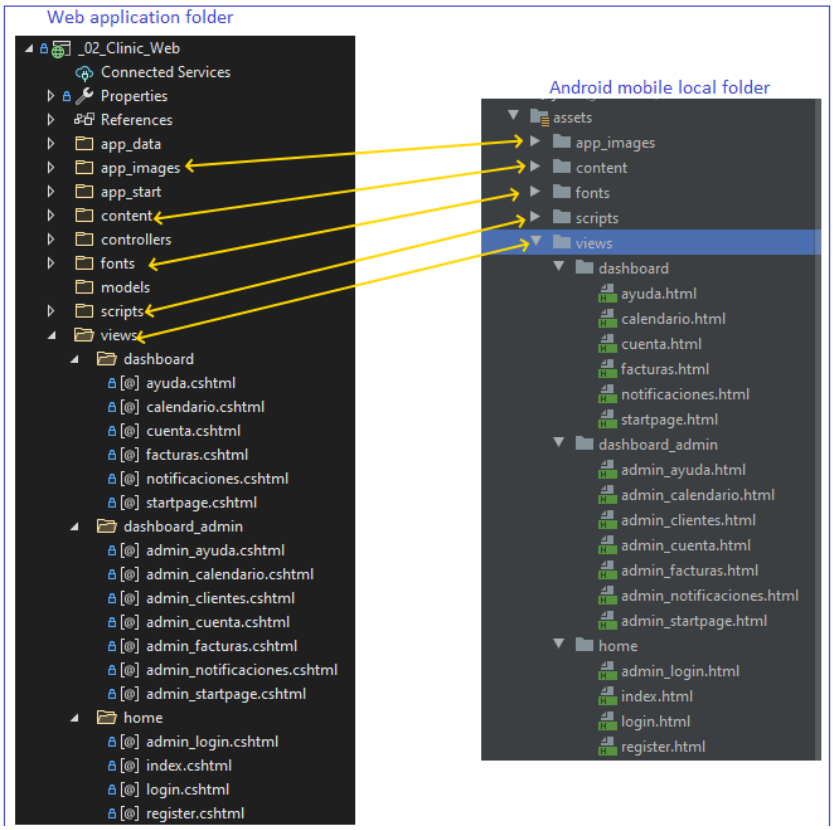
Figura 25: Desarrollo del layout de la aplicación Android



Fuente: Elaboración propia

- Los archivos fuente de la WebView en la aplicación móvil se encuentran en la carpeta */assets*. Esta carpeta contiene una copia de todos los archivos HTML fuente (las vistas), los archivos de scripts, los archivos de *app_images*, las fuentes y los archivos de la carpeta *content* del proyecto web.

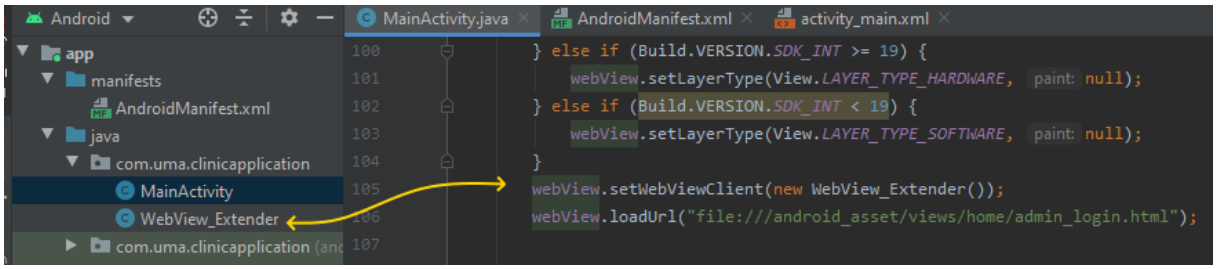
Figura 26: Estructura de carpetas del proyecto 02_Clinic_Android



Fuente: Elaboración propia

- La principal función de la aplicación es cargar un archivo HTML5 ubicado localmente en el paquete de la aplicación. Este archivo HTML5 contiene las referencias a las diferentes hojas de estilo y scripts en la aplicación Android. En este caso, como se muestra a continuación, el WebView carga la página de inicio de sesión del trabajador.

Figura 27: Inicialización del WebView



Fuente: Elaboración propia

El principal reto detrás del desarrollo del front-end en este proyecto se debe principalmente a una característica importante de las aplicaciones móviles: la adaptabilidad del sistema. Hoy en día, debido al avance de la tecnología, los dispositivos cambian constantemente de tamaño. En este sentido, durante el desarrollo del proyecto aparecieron numerosos problemas relacionados con el diseño responsive del sitio web utilizando diferentes tamaños de pantalla. Este problema se debe principalmente al uso incorrecto de las hojas de estilo en cascada (CSS) elaboradas manualmente.

Finalmente, se decidió llevar a cabo el uso de la librería Bootstrap junto con un uso correcto de estilos CSS (uso de media queries) y la manipulación de elementos mediante el uso de Javascript/JQuery. Esto solucionó el problema inicial. Un ejemplo claro se encuentra en el diseño del evento de desplazamiento automático en diferentes páginas:

Figura 28: Ejemplo de CSS y JavaScript para el control del diseño adaptable

CSS

```css
@media (min-width: 1024px) {
    #initial {
        background-attachment: fixed;
    }
}

@media (max-width: 992px) {
    #initial {
        margin-bottom: 0;
        height: 100vh;
        background-position-x: 55% !important;
    }

        #initial .container {
            padding-bottom: 63px;
        }

        #initial h1...

        #initial h2...
}

@media (max-height: 600px) {
    #initial {
        height: 110vh;
    }
```

Script

```javascript
$(document).on("scroll", function () {
    let position = window.scrollY + 200
    navbarlinks.forEach(navbarlink => {
        if (!navbarlink.hash) return
        let section = document.querySelector(navbarlink.hash)
        if (!section) return
        if (position >= section.offsetTop
            && position <=
            (section.offsetTop + section.offsetHeight)) {
            navbarlink.classList.add('active_nav_link')
        } else {
            navbarlink.classList.remove('active_nav_link')
        }
    })
});
```

Fuente: Elaboración propia

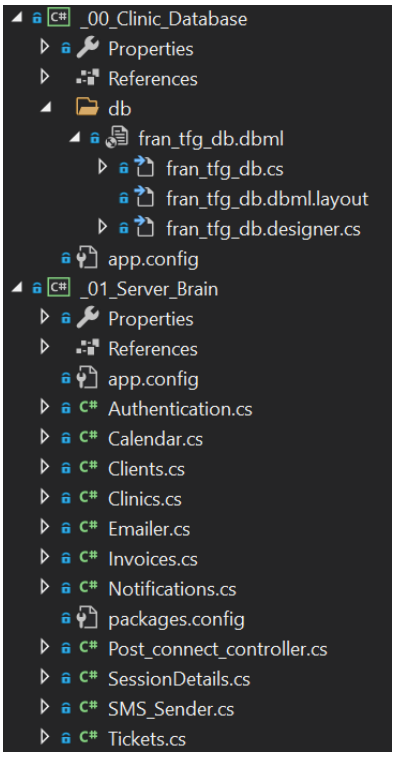
5.3El desarrollo del back-end

El desarrollo del backend también se conoce como desarrollo del lado del servidor. Se centra principalmente en las bases de datos, la lógica del backend, las API y los servidores. El backend de un sitio web es una combinación de servidores, aplicaciones y bases de datos. Principalmente, el código escrito por los desar rolladores de backend

ayuda a los navegadores a comunicarse y a almacenar, leer, actualizar y eliminar la información de la base de datos.

Como hemos mencionado en el capítulo de diseño, los componentes del desarrollo del backend que conforman la lógica del modelo de datos y la base de datos se encuentran en dos proyectos diferentes:

Figura 29: Estructura de los proyectos 00_Clinic_Database y 01_Server_Brain



Fuente: Elaboración propia

5.3.1 Implementación de la base de datos

Para generar la base de datos y las tablas (relaciones), es necesario elaborar una serie de scripts donde se indican todos los componentes que necesitamos. Para controlar la base de datos de una forma sencilla y clara, se ha utilizado la herramienta Microsoft SQL Server Management Studio, la cual ofrece una interfaz para manipular toda la base de datos.

Para llevar a cabo estas acciones, se utilizan varios scripts empleando la sintaxis de Transact-SQL (extensión de MSSQL que permite realizar consultas mediante sentencias declarativas).

5.3.1.1 Código Transact-SQL (T-SQL): base de datos *fran_tfg_db*

El código T-SQL que este TFG utiliza para crear la base de datos se muestra en el siguiente fragmento de código. Además, se crea una base de datos denominada *fran_tfg_db_log* para registrar todas las acciones llevadas a cabo en la base de datos.

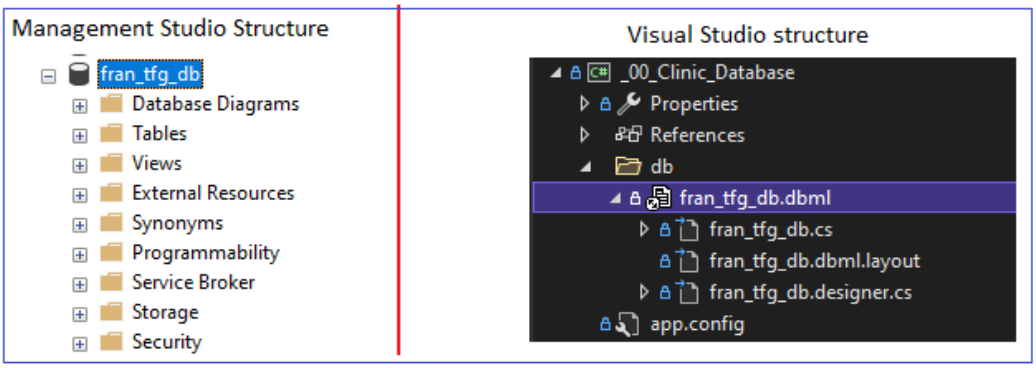
Figura 30: Creación de la base de datos

```
1
2  use [master] -- // use the master database on server
3  go
4      --// create the
5      create database [fran_tfg_db] on  primary
6      ( name = N'fran_tfg_db',
7      filename = N'E:\Repos\fran_tfg_db.mdf' ,
8      --// the {E:\Repos} must be a valid file directory
9      size = 167872kb ,
10     maxsize = unlimited,
11     filegrowth = 16384kb )
12
13      log on
14
15     ( name = N'fran_tfg_db_log',
16     filename = N'E:\Repos\fran_tfg_db_log.ldf' ,
17     --// the {E:\Repos} must be a valid file directory
18     size = 2048kb ,
19     maxsize = 2048gb ,
20     filegrowth = 16384kb )
21 go
22
```

Fuente: Elaboración propia

Tras ejecutar este código, la estructura de la base de datos en SQL Management Studio y en el IDE de Visual Studio tiene el siguiente aspecto:

Figura 31: Estructura de SQL Server Management Studio y Visual Studio



Fuente: Elaboración propia

5.3.1.2 Código Transact-SQL (T-SQL): tablas

Aunque en este TFG se han utilizado nueve tablas para la base de datos y, ya que todas las tablas se generan de forma prácticamente idéntica, vamos a poner como ejemplo la creación de las tablas Usuarios y Citas.

Figura 32: Creación de las tablas *users* y *appointments*

```
Use fran_tfg_db

-- check if user table exists
if object_id('dbo.users', 'u') is not null drop table users

-- create users table
-- please not that the users in this scenario implies the Clients
create table users(
    id bigint identity primary key,
    session_id nvarchar(50),
    dni_nie nvarchar(20),
    firstname nvarchar(50),
    lastname nvarchar(50),
    email nvarchar(100),
    phone_number nvarchar(50),
    city nvarchar(100),
    password nvarchar(50)
);

--============== appointments table
-- check if appointment table exists
if object_id('dbo.appointments', 'u') is not null drop table appointments

-- create appointment table
create table appointments(
    id bigint identity primary key,
    dni_nie nvarchar(20),
    beneficiary nvarchar(50),
    hours nvarchar(10),
    date nvarchar(20),
    description nvarchar(max),
    clinic_id nvarchar(50),
    clinic_name nvarchar(max)
);
```

Fuente: Elaboración propia

Aunque el código en la imagen anterior se explica por sí mismo, una rápida explicación puede ser útil:

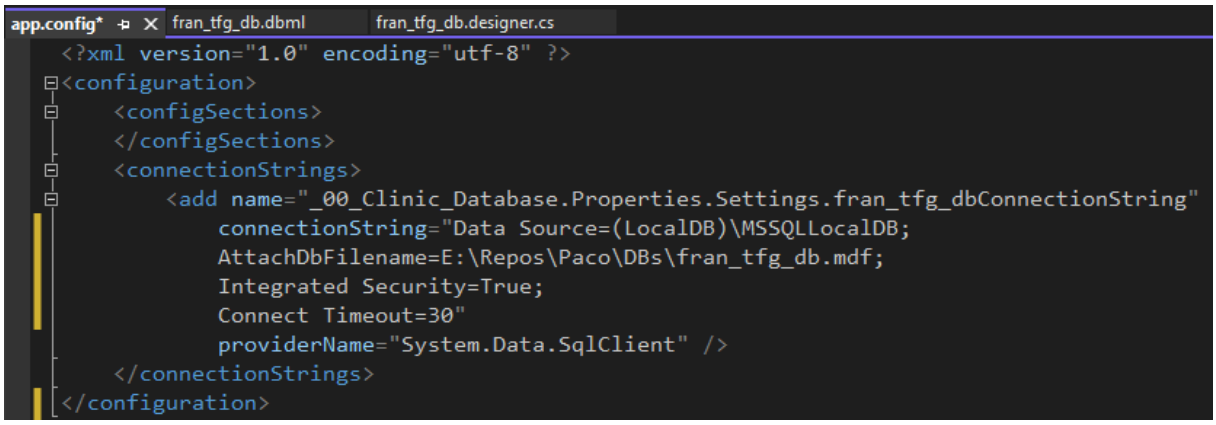
- La línea 2 se asegura que el código T-SQL posterior se ejecutará en la base de datos especificada (*fran_tfg_db*, que es la base de datos que creamos previamente).
- Las líneas 5 y 23 garantizan que si la tabla ya existe, el sistema debe eliminar la tabla.
- Las líneas 9 a 19 y 26 a 35 crean las tablas Users y Appointment (relaciones).

Una vez creadas todos estos componentes esenciales, es muy importante generar la cadena de conexión a la base de datos.

Una cadena de conexión es una línea de texto que establece la información sobre una fuente de datos y los medios de conexión. En otras palabras, es una cadena que pasa instrucciones de conexión a un controlador para iniciar la conexión con la base de datos.

Para este proyecto, la cadena de conexión a una base de datos local es la siguiente:

Figura 33: Cadena de conexión a la base de datos local



Fuente: Elaboración propia

Sin embargo, si la base de datos se encuentra en un servidor online, la cadena de conexión en el archivo de configuración de la aplicación tendrá el siguiente aspecto:

Figura 34: Cadena de conexión a la base de datos online

```
<configuration>
    <configSections>
    </configSections>
    <connectionStrings>
    <add name="_00_Clinic_Database.Properties.Settings.fran_tfg_dbConnectionString"
         connectionString="Data Source=www.clinicasuma.com;
         database=fran_tfg_db;
         uid=[redacted];
         pwd=[redacted]"
        providerName="System.Data.SqlClient" />
    </connectionStrings>
</configuration>
```

Fuente: Elaboración propia

El "*uid*" y el "*pwd*" hacen referencia al nombre de usuario y contraseña del servidor, respectivamente, los cuales se han ocultado a propósito para esta documentación.

5.3.2 Funcionamiento principal del back-end

Para entender como funciona el back-end, primero es necesario introducir algunas características sobre el mecanismo del servidor en este apartado:

- La base de datos se encuentra en el proyecto 00_Clinic_Database y toda su lógica se encuentra en el proyecto 01_Server_Brain. Este es el proyecto que procesa todas las peticiones del servidor.
- Todo este proyecto se caracteriza por ser un proyecto de librería de clases. Esto significa que está formado por un conjunto de clases.
- Cada clase está relacionada con cada una de las funciones implementadas. Las funciones internas de cada una de las clases se denominan módulos.
- Un módulo puede ser público (si se accede desde el mecanismo de intercambio seguro) o privado (si se llama internamente desde otros módulos).
- Cada módulo devuelve siempre un diccionario con el resultado de la petición solicitada. Por ejemplo, como se muestra a continuación, el módulo que se

encarga de verificar las credenciales del usuario acepta la solicitud y devuelve un diccionario.

Figura 35: Ejemplo de módulo

```
1 reference | Francisco Perdiguero Moreno, 58 days ago | 1 author, 1 change
public static Dictionary<string, object> Verify_User(Dictionary<string, object> dic)
{
    try
    {

        var dni = dic["dni"].ToString();
        var pass = dic["password"].ToString();


        Dictionary<string, object> response_dic = new Dictionary<string, object>()
        {
            { "success", true},
            { "feedback", Check_credentials(dni, pass, true)}
        };
        return response_dic;
    }
    catch (Exception x)
    {
        return new Dictionary<string, object>() {
            { "success", false},
            { "error_message", x.Message},
        };
    }
}
```

Fuente: Elaboración propia

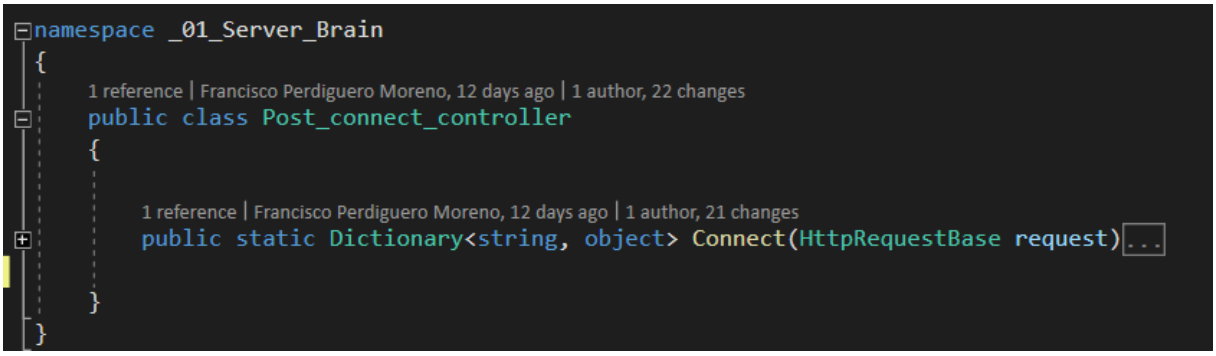
- El algoritmo de cada módulo emplea la metodología try-catch para gestionar las excepciones. Ante cualquier excepción, el paquete devuelto devuelve una clave *success* con valor *false* y otra clave *error_message* con un valor que contiene el mensaje de excepción del servidor.

Es importante mencionar que cada algoritmo desarrollado en los módulos tiene un nombre coherente, lo que permite a los lectores y desarrolladores entender rápidamente el objetivo y principal funcionamiento del código.

La característica principal que presenta este proyecto es la estructura de las peticiones recibidas y enviadas. Los paquetes recibidos son unos diccionarios que contienen una clave principal que indica la acción solicitada por parte del cliente junto con una serie de datos. Posteriormente, el servidor procesa la petición y devuelve otro diccionario a modo de respuesta que el mecanismo del lado del cliente pueda decodificarlo.

El servidor recibe la petición del cliente a través del controlador. La función que se encarga de recibir las peticiones en el lado del servidor es la siguiente:

Figura 36: Función para la recepción de peticiones en el servidor



Fuente: Elaboración propia

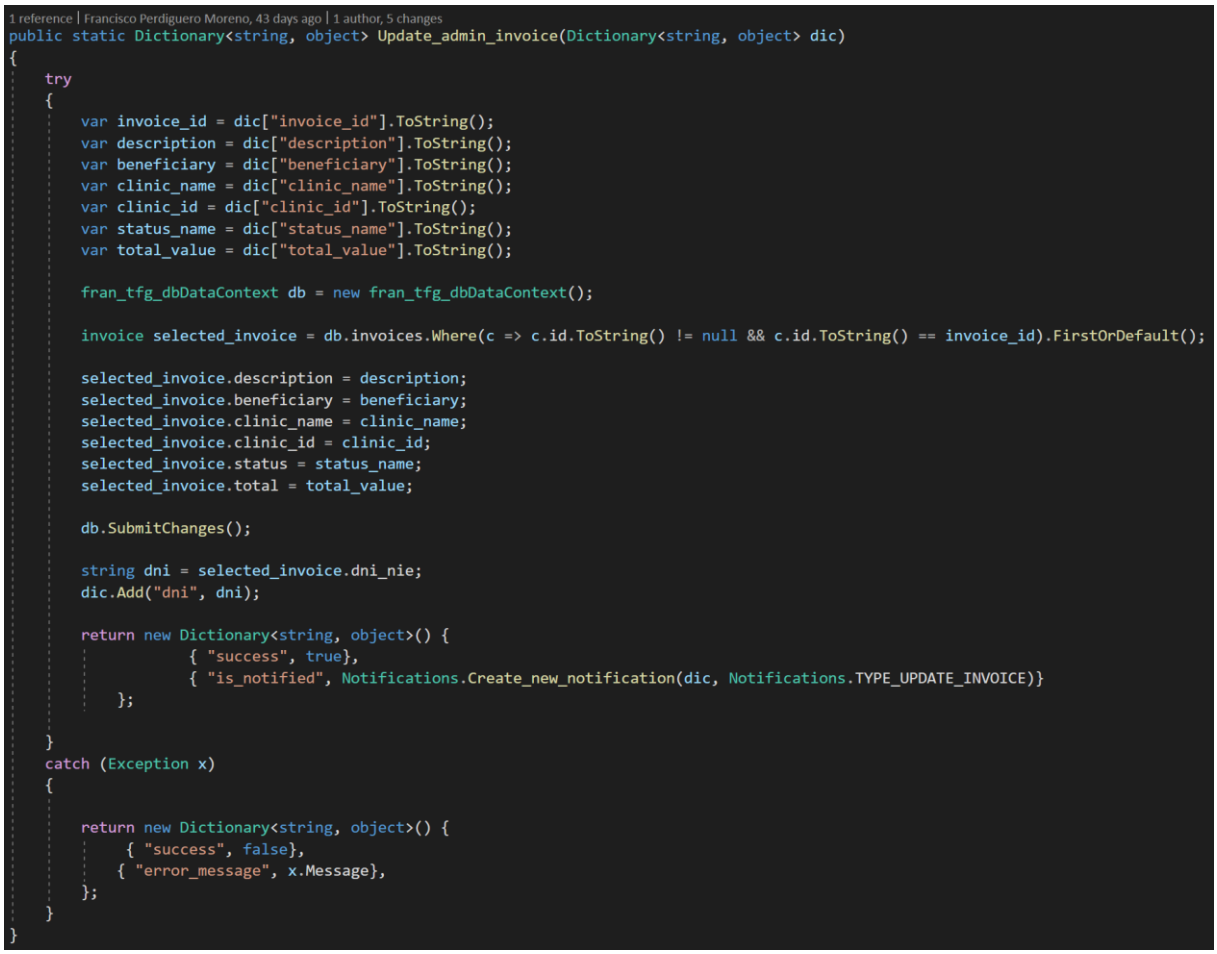
Esta función es uno de los núcleos del sistema puesto que implemente el mecanismo de seguridad del sistema. Esta función se detallará más adelante en la explicación del desarrollo del puente cliente-servidor.

Cuando una petición llega al servidor a través del controlador, esta función es la encargada de seleccionar el módulo necesario para resolver la petición.

Tras identificar el módulo encargado de resolver la petición, el servidor se encarga de procesarla y devuelve la respuesta al cliente. En estos módulos, existen una gran cantidad de funciones. Para hacerlo más simple, no vamos a entrar en todos sus detalles. Sin embargo, podemos destacar las principales funciones que realizan:

- **Conexión con la base de datos**: Todas las clases del servidor deben iniciar su conexión con la base de datos relacional puesto que todas las peticiones que el cliente realiza se basan en la manipulación de los datos (CRUD). En este ejemplo, podemos ver su funcionamiento:

Figura 37: Ejemplo de conexión a la base de datos



```
1 reference | Francisco Perdiguero Moreno, 43 days ago | 1 author, 5 changes
public static Dictionary<string, object> Update_admin_invoice(Dictionary<string, object> dic)
{
    try
    {
        var invoice_id = dic["invoice_id"].ToString();
        var description = dic["description"].ToString();
        var beneficiary = dic["beneficiary"].ToString();
        var clinic_name = dic["clinic_name"].ToString();
        var clinic_id = dic["clinic_id"].ToString();
        var status_name = dic["status_name"].ToString();
        var total_value = dic["total_value"].ToString();

        fran_tfg_dbDataContext db = new fran_tfg_dbDataContext();

        invoice selected_invoice = db.invoices.Where(c => c.id.ToString() != null && c.id.ToString() == invoice_id).FirstOrDefault();

        selected_invoice.description = description;
        selected_invoice.beneficiary = beneficiary;
        selected_invoice.clinic_name = clinic_name;
        selected_invoice.clinic_id = clinic_id;
        selected_invoice.status = status_name;
        selected_invoice.total = total_value;

        db.SubmitChanges();

        string dni = selected_invoice.dni_nie;
        dic.Add("dni", dni);

        return new Dictionary<string, object>() {
                    { "success", true},
                    { "is_notified", Notifications.Create_new_notification(dic, Notifications.TYPE_UPDATE_INVOICE)}
                };

    }
    catch (Exception x)
    {

        return new Dictionary<string, object>() {
            { "success", false},
            { "error_message", x.Message},
        };
    }
}
```

Fuente: Elaboración propia

En él podemos observar como la función recibe el diccionario que contiene la petición del cliente. Posteriormente, se crea un objeto de tipo *fran_tfg_dbDataContext* el cual es generado por LINQ-to-SQL y permite acceder a la base de datos con las funcionalidades que ofrece este framework. Con este simple objeto se puede realizar cualquier tipo de consulta CRUD a la base de datos de una forma muy sencilla.

- **Envío de notificaciones a través del email y SMS**: Las clases Emailer y SMS_Sender son un poco diferentes al resto. Estas clases contienen las funciones necesarias para realizar el envío de notificaciones a través del email y SMS.

Figura 38: Envío de notificaciones a través de Email y SMS

```
9 references | Francisco Perdiguero Moreno, 7 days ago | 1 author, 4 changes
public static void Send_notification(Dictionary<string, object> dic, int type, bool can_send_sms = false)
{

    if (type == TYPE_SIGN_UP)
    {
        //Use parallel programming technique
        Thread notification_thread = new Thread(new ThreadStart(Send_email));
        notification_thread.Start();

        void Send_email()
        {
            Emailer.Send_sign_up_email(dic);
        }


    }
    else if (type == TYPE_REMOVE_INVOICE)
    {
        //Use parallel programming technique
        Thread notification_email_thread = new Thread(new ThreadStart(Send_email));
        notification_email_thread.Start();

        void Send_email()
        {
            Emailer.Send_remove_invoice_email(dic);
        }
    }
    else if (type == TYPE_UPDATE_INVOICE)
    {
        //Use parallel programming technique
        Thread notification_email_thread = new Thread(new ThreadStart(Send_email));
        notification_email_thread.Start();

        void Send_email()
        {
            Emailer.Send_update_invoice_email(dic);
        }

        //SMS
        //Use parallel programming technique
        Thread notification_sms_thread = new Thread(new ThreadStart(Send_sms));
        notification_sms_thread.Start();

        void Send_sms()
        {
            SMS_Sender.Send_update_invoice_sms(dic);
        }
    }
    else if (type == TYPE_NEW_INVOICE)
```

Fuente: Elaboración propia

La clase *Notifications* se encarga de llamar a cada uno de los métodos necesarios dependiendo si se necesita enviar email o SMS.

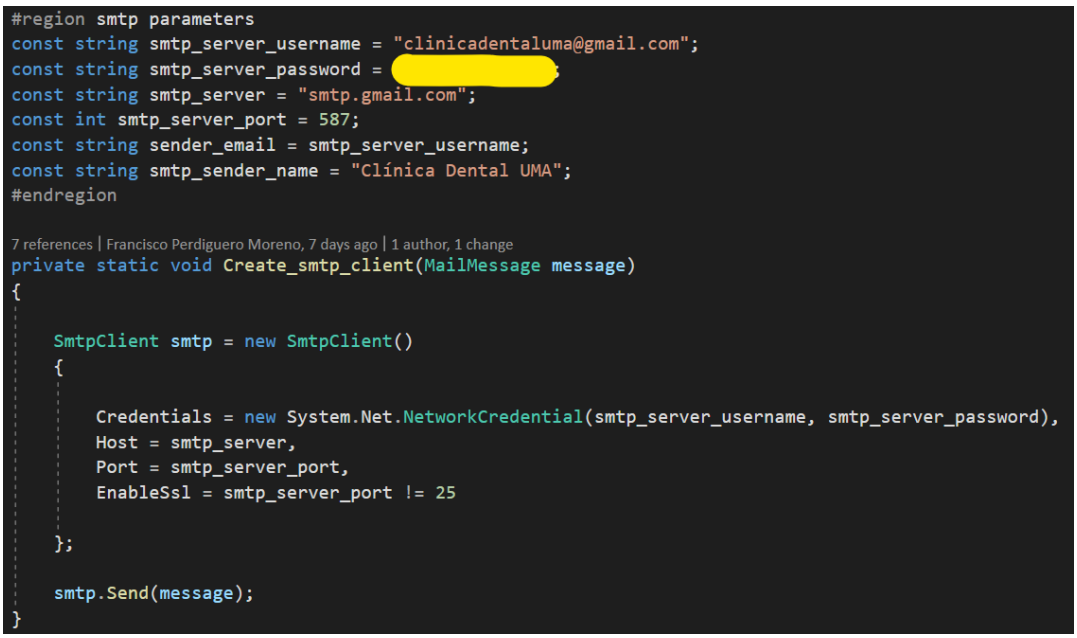
Inicialmente, el flujo principal de ejecución continuaba por cada uno de los módulos encargados del envío de la notificación. Esto provocó uno de los grandes problemas encontrados durante todo el desarrollo: cuando uno de estos módulos generaba un error, provocaba que todo el sistema se bloquease y fallase al completo.

Como podemos ver en la imagen superior, gracias a la creación de un objeto Thread por cada módulo, el envío de notificación es independiente del flujo principal de ejecución, puesto como hemos visto, el sistema de notificaciones debe ser independiente del funcionamiento global del sistema.

Tras comprender el flujo del sistema de notificaciones, podemos pasar a analizar cómo funcionan estos módulos poniendo algunos ejemplos:

- **Envío de email**

Figura 39: Envío de notificaciones a través de Email

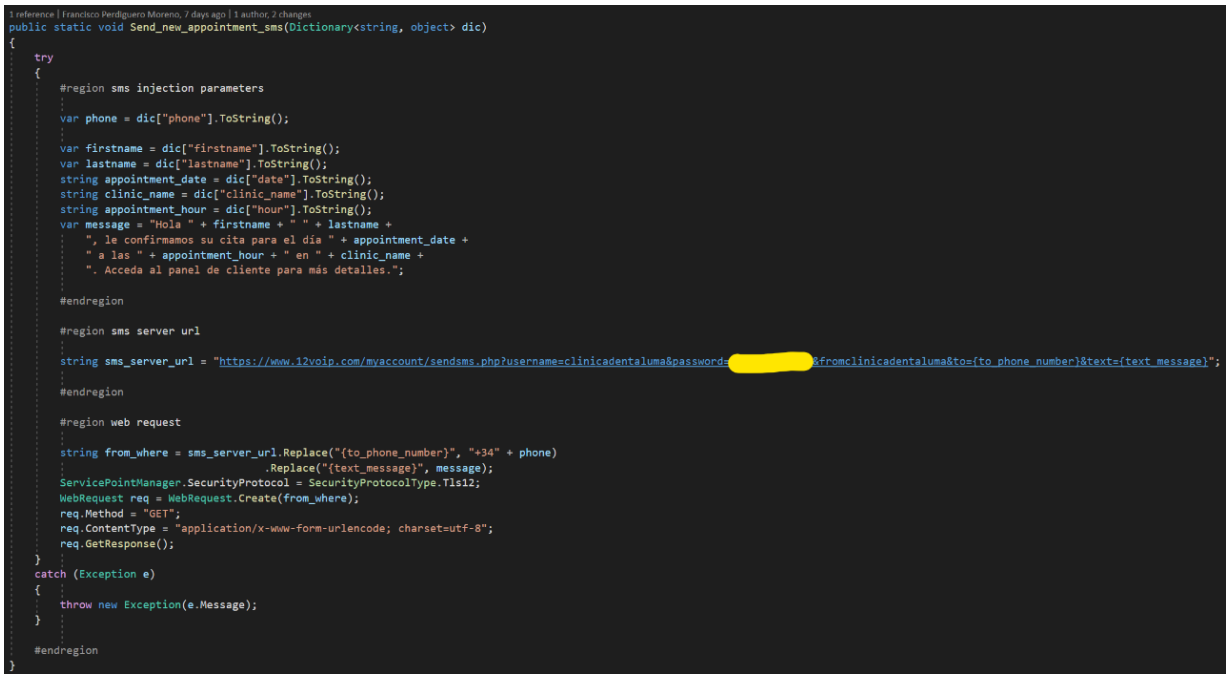


Fuente: Elaboración propia

Para realizar el envío de emails, se necesitan una serie de parámetros iniciales. Estos son los parámetros necesarios para la creación de un cliente SMTP. Gracias a estos parámetros, mediante la función Create_smtp_client() y un mensaje distinto para cada tipo de notificación, se inicia un cliente SMTP el cual se encarga de enviar el email.

- **Envío de SMS**

Figura 40: Envío de notificaciones a través de SMS



```
1 reference | Francisco Perdiguero Moreno, 7 days ago | 1 author, 2 changes
public static void Send_new_appointment_sms(Dictionary<string, object> dic)
{
    try
    {
        #region sms injection parameters

        var phone = dic["phone"].ToString();

        var firstname = dic["firstname"].ToString();
        var lastname = dic["lastname"].ToString();
        string appointment_date = dic["date"].ToString();
        string clinic_name = dic["clinic_name"].ToString();
        string appointment_hour = dic["hour"].ToString();
        var message = "Hola " + firstname + " " + lastname +
            ", le confirmamos su cita para el día " + appointment_date +
            " a las " + appointment_hour + " en " + clinic_name +
            ". Acceda al panel de cliente para más detalles.";

        #endregion

        #region sms server url

        string sms_server_url = "https://www.12voip.com/myaccount/sendsms.php?username=clinicadentaluma&password=[redacted]&fromclinicadentaluma&to={to_phone_number}&text={text_message}";

        #endregion

        #region web request

        string from_where = sms_server_url.Replace("{to_phone_number}", "+34" + phone)
                                          .Replace("{text_message}", message);
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
        WebRequest req = WebRequest.Create(from_where);
        req.Method = "GET";
        req.ContentType = "application/x-www-form-urlencode; charset=utf-8";
        req.GetResponse();
    }
    catch (Exception e)
    {
        throw new Exception(e.Message);
    }

    #endregion
}
```

Fuente: Elaboración propia

Para realizar el envío de SMS, este proyecto utiliza la aplicación *12voip.com* la cual ofrece una API para el envío de mensajes. Para ello, se genera el mensaje deseado con un máximo de 150 caracteres. Tras esto, se envía una petición HTTP a la URL que nos ofrece 12voip donde indicamos los datos de la cuenta desde donde se envían los mensajes, el número de teléfono al que queremos enviarlo y el mensaje.

Aunque existen otras alternativas que mejorarían el envío y reducirían el coste de este servicio, este proyecto se ha decantado por esta solución debido a la facilidad de su implementación.

5.3 El desarrollo del puente cliente-servidor (Handshake)

El desarrollo del puente cliente-servidor es una de las partes más importantes del sistema. Gracias a este conector podemos enviar peticiones al servidor y responderlas al cliente de forma rápida y bastante segura.

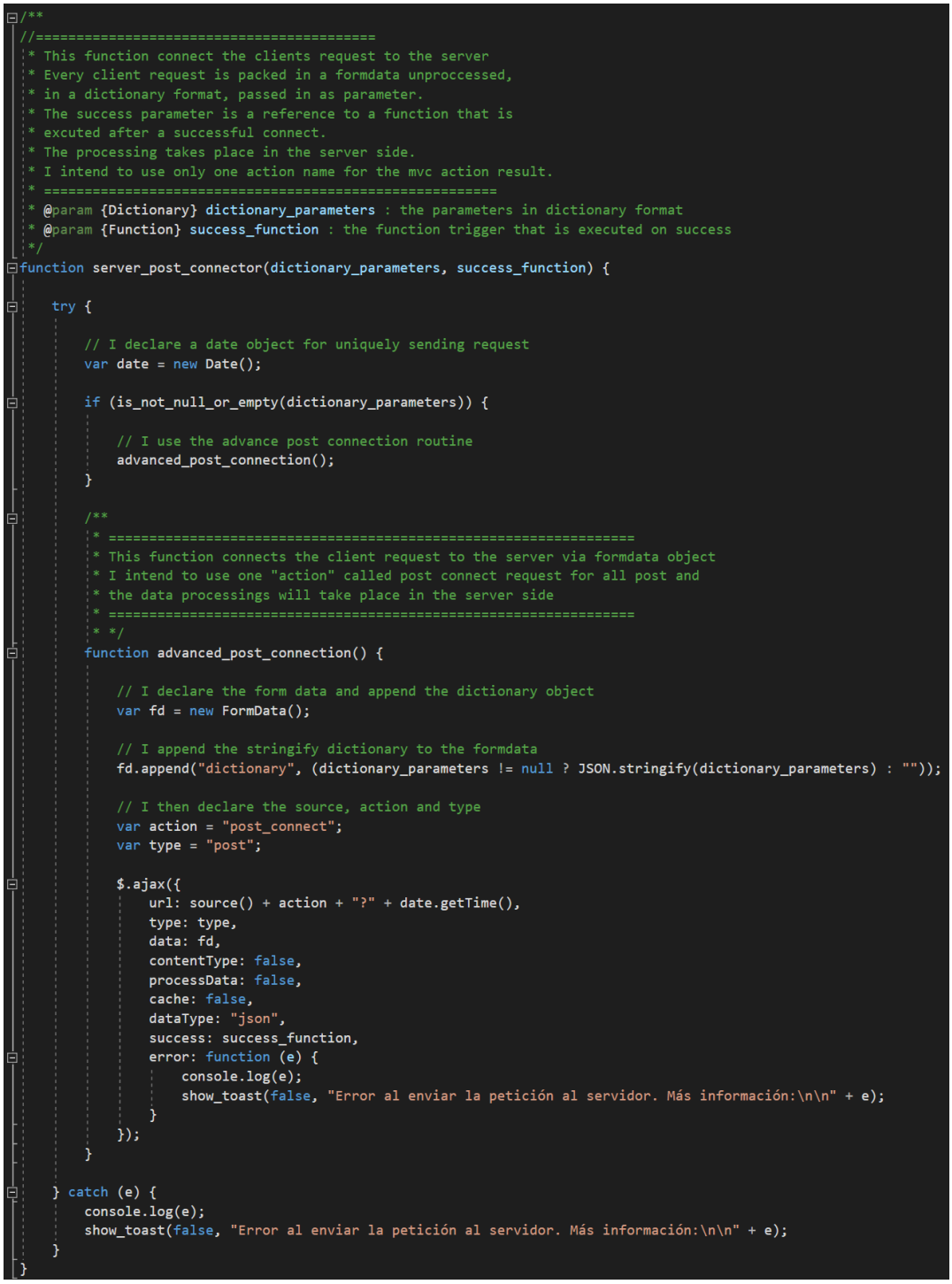
Este desarrollo podemos dividirlo en tres grandes secciones:

- Envío de la petición al controlador del cliente.
- Envío de la petición al controlador cliente-servidor
- Recepción de la petición en el lado del servidor.

5.3.1 Envío de la petición al controlador del cliente.

Cada vez que un usuario necesita solicitar al servidor cualquier tipo de información de la base de datos, es necesario generar un paquete de petición. Como hemos mencionado anteriormente, cada una de las peticiones que el cliente desea realizar al back-end se realizan mediante la función server_post_connector() en JavaScript.

Figura 41: Función para el envio de peticiones del lado del cliente



```
/**
 *=======================================
 * This function connect the clients request to the server
 * Every client request is packed in a formdata unproccessed,
 * in a dictionary format, passed in as parameter.
 * The success parameter is a reference to a function that is
 * excuted after a successful connect.
 * The processing takes place in the server side.
 * I intend to use only one action name for the mvc action result.
 * ======================================================
 * @param {Dictionary} dictionary_parameters : the parameters in dictionary format
 * @param {Function} success_function : the function trigger that is executed on success
 */
function server_post_connector(dictionary_parameters, success_function) {

    try {

        // I declare a date object for uniquely sending request
        var date = new Date();

        if (is_not_null_or_empty(dictionary_parameters)) {

            // I use the advance post connection routine
            advanced_post_connection();
        }

        /**
         * ================================================================
         * This function connects the client request to the server via formdata object
         * I intend to use one "action" called post connect request for all post and
         * the data processings will take place in the server side
         * ================================================================
         * */
        function advanced_post_connection() {

            // I declare the form data and append the dictionary object
            var fd = new FormData();

            // I append the stringify dictionary to the formdata
            fd.append("dictionary", (dictionary_parameters != null ? JSON.stringify(dictionary_parameters) : ""));

            // I then declare the source, action and type
            var action = "post_connect";
            var type = "post";

            $.ajax({
                url: source() + action + "?" + date.getTime(),
                type: type,
                data: fd,
                contentType: false,
                processData: false,
                cache: false,
                dataType: "json",
                success: success_function,
                error: function (e) {
                    console.log(e);
                    show_toast(false, "Error al enviar la petición al servidor. Más información:\n\n" + e);
                }
            });
        }

    } catch (e) {
        console.log(e);
        show_toast(false, "Error al enviar la petición al servidor. Más información:\n\n" + e);
    }
}
```

Fuente: Elaboración propia

Esta función recibe como parámetros un diccionario con los datos necesarios por el servidor y una referencia a la función que se ejecuta cuando la petición se ha enviado correctamente.

Este diccionario se comprime y se empaqueta como un objeto *FormData*, necesario para el envío de datos en peticiones HTTP. Tras esto, mediante el uso de AJAX, se realiza una petición HTTP a una ruta que el controlador del cliente establece. En la siguiente sección entenderemos su funcionamiento.

5.3.2 Envío de la petición entre el controlador cliente-servidor.

Inicialmente, el TFG utilizaba el enfoque tradicional MVC donde toda la implementación de la lógica del servidor es gestionada por el controlador. Por ejemplo, la siguiente imagen muestra como es este enfoque convencional:

Figura 42: Enfoque convencional para la implementación del controlador

```
public class AppointmentController : Controller
{
    private readonly appointment _context;

    public UsersController(appointment context)
    {
        _context = context;
    }

    // GET: /people and index page we serve the appointment model
    public async Task Index()
    {
        return View(await _context.Appomtment.ToListAsync());
    }

    // GET: /people/details/5
    public async Task Details(int id)
    {
        var the_appointment= await _context.Appointment.Find(id);

        if (the_appointment == null)
        {
            return NotFound();
        }

        return View(the_appointment);
    }
}
```

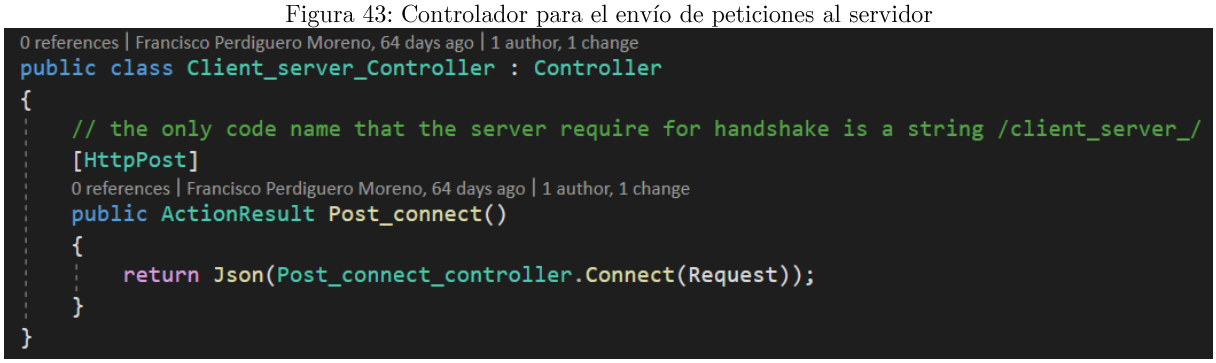
Fuente: Elaboración propia

Tras varias funcionalidades implementadas de esta forma, el código se vuelve demasiado ambiguo y difícil de mantener, por lo que modificamos la definición de controlador y modelo y ahora toda la lógica del sistema se encuentra gestionada por el modelo. Además, Microsoft considera que siempre que el controlador disponga de una gran cantidad de responsabilidades y antes de que la lógica del controlador se vuelva

muy compleja, es recomendable sacar la lógica fuera y llevarla al modelo. Por este motivo, convertimos el controlador para que se encargue únicamente del envío de las peticiones del cliente al modelo (back-end). El modelo gestiona todos los procesos de datos y los devuelve al controlador.

El controlador encargado de recibir las peticiones del cliente se denomina Client_server_Controller. En él, se establece una función pública denominada Post_connect(). Esta función es la encargada del manejo de todas las peticiones:

Figura 43: Controlador para el envío de peticiones al servidor



Fuente: Elaboración propia

Como podemos observar en la figura superior, la función Post_connect() está declarada con un atributo de enrutamiento "[HttpPost]". Gracias a este atributo, cualquier petición POST que se realice a una determinada URL, ejecutará las acciones que se encuentren en su interior.

Esta URL se define de la siguiente forma: **{nombre del controlador}/{nombre de la función}**. En este caso, cualquier petición que se realice a la dirección "**client_server_/post_connect**" será enrutada hacia esta función y ejecutará su contenido.

En concreto, cuando llega una petición (*Request*), se llama a la función que controla todas y cada una de las peticiones recibidas por el servidor. Además, es la función

encargada de ofrecer la seguridad al sistema. En la siguiente sección explicaremos su funcionamiento.

5.3.3 Recepción de la petición en el lado del servidor

Cuando el controlador recibe la petición, este se encarga de llamar a una función que se encuentra en el modelo del sistema. Esta función se encuentra dentro de la clase Post_connect_controller y se denomina Connect(). Este es el punto de entrada de todas y cada una de las peticiones que envía el cliente. En este proyecto se ha decidido denominar alternativamente esta función como "Cámara de seguridad".

La cámara de seguridad sirve como puerta de entrada de cualquier solicitud procedente del lado del cliente al servidor. Imagínese la cámara de seguridad como un puesto de control en la frontera de cada país, donde el oficial de seguridad tiene que comprobar la originalidad del pasaporte de todo el mundo. Esa es precisamente una versión resumida de lo que hace la cámara de seguridad.

Así es como funciona internamente:

- Al recibir cualquier petición del controlador, la cámara de seguridad deserializa el paquete y luego busca la clave "*action*" del diccionario contenido en la petición.
- A continuación, la cámara utiliza el método *switch* para procesar las acciones.
- Si la acción es conocida (es decir, auténtica), entonces la cámara pasa el paquete al módulo respectivo responsable de procesar ese paquete. La respuesta de ese módulo se devuelve entonces como un paquete a la máquina del cliente que hace la solicitud, todo de vuelta a través del controlador.
- Si la acción no es conocida, entonces la cámara de seguridad devuelve null. Esto hace sospechar que se trata de una petición realizada por algún cibercriminal.

Figura 44: Descripción de la función Connect() en el backend

```
1 reference | Francisco Perdiguero Moreno, 13 days ago | 1 author, 22 changes
public class Post_connect_controller
{

    1 reference | Francisco Perdiguero Moreno, 13 days ago | 1 author, 21 changes
    public static Dictionary<string, object> Connect(HttpRequestBase request)
    {

        //SECURITY CHAMBER

        //Deserialize because dictionary is Stringify
        Dictionary<string, object> dic = JsonConvert.DeserializeObject<Dictionary<string, object>>(request["dictionary"]);
        string action = dic["action"].ToString();

        switch (action)
        {
            case "session_details":
                return SessionDetails.Get_session_details(dic);
            case "sign_in":
                return Authentication.Verify_User(dic);
            case "sign_in_admin":
                return Authentication.Verify_admin(dic);
            case "sign_up":
                return Authentication.Sign_up(dic);
            case "get_clinics":
                return Calendar.Get_clinics(dic);
            case "get_appointments":
                return Calendar.Get_all_appointments(dic);
            case "create_appointment":
                return Calendar.Create_appointment(dic);
            case "get_user_appointments":
                return Calendar.Get_user_appointment(dic);
            case "remove_appointment":
                return Calendar.Remove_user_appointment(dic);
            case "get_user_details":
                return Authentication.Get_user_details(dic);
            case "change_user_details":
                return Authentication.Change_user_details(dic);
            case "get_user_invoices":
                return Invoices.Get_user_invoices(dic);
            case "get_admin_clinics":
                return Clinics.Get_admin_clinics(dic);
            case "get_all_clinics":
                return Clinics.Get_all_clinics(dic);
            case "update_admin_invoice":
                return Invoices.Update_admin_invoice(dic);
            case "remove_admin_invoice":
                return Invoices.Remove_admin_invoice(dic);
            case "create_new_invoice":
                return Tickets.Add_ticket_message(dic);
            case "update_ticket":
                return Tickets.Update_ticket(dic);
            case "delete_ticket":
                return Tickets.Delete_ticket(dic);
            case "own_ticket":
                return Tickets.Own_ticket(dic);
            default:
                break;
        }


        //For security porposes, send this request to the cybersecurity team directly.
        return null;

    }
```

Fuente: Elaboración propia

Aquí es donde necesitamos más mejoras. Antes de devolver *null*, podemos pasar el HttpRequestBase completo a otro módulo que lo entregue a un futuro departamento de ciberseguridad. Es aquí donde se puede identificar la IP, la firma y el nombre de la máquina del ciberdelincuente, entre otros detalles.

5.4 Despliegue de la aplicación

Tras analizar todo el desarrollo en específico de cada una de las partes del proyecto, se ha llevado a cabo el despliegue de la aplicación. Inicialmente, la aplicación estaba centrada en el uso de un servidor local para su funcionamiento. Sin embargo, para llevar la aplicación a un mayor nivel y realizar pruebas reales con muchos usuarios al mismo tiempo, se ha decidido desplegar la aplicación en un servidor online. En el Apéndice A se explica más detalles sobre como llevar a cabo este despliegue mediante el manual de instalación del proyecto.

La aplicación web se aloja en un servidor web en la siguiente dirección:
https://www.clinicasuma.com.

Además, una APK se genera con la aplicación móvil compilada y lista para ser instalada en los dispositivos Android.

6

Pruebas

Tras explicar el desarrollo del sistema, en este capítulo se llevarán a cabo una serie de pruebas para comprobar que las soluciones aportadas cumplen los requisitos especificados inicialmente.

6.1 Introducción a la fase de pruebas

La fase de desarrollo de pruebas es una de las más importantes en todo el ciclo de desarrollo de software. En esta parte, se analizan y se identifican todos los problemas y errores que pueden haber existido durante el desarrollo del sistema. Para ello, será necesario comprobar que todos los requisitos establecidos inicialmente cumplen su función.

Como hemos comentado anteriormente, se ha decidido desplegar la aplicación en un servidor online con el objetivo de llevar a cabo las pruebas mediante el uso de diferentes usuarios simultáneamente.

Las principales pruebas llevadas a cabo en el sistema se resumen en los siguientes casos de prueba:

Tabla 10: Prueba 1 – Inicio de sesión

Prueba 1 – Inicio de sesión	
Condiciones previas	El cliente o trabajador deben de estar registrados en el sistema.
Acciones	El usuario introduce sus datos en el formulario de inicio de sesión correspondiente para clientes o trabajadores y pulsa el botón de Iniciar Sesión.
Resultado esperado	El sistema valida los datos introducidos por el usuario y los redirige al panel de cliente o trabajador.

Fuente: Elaboración propia

Tabla 11: Prueba 2 - Registro

Prueba 2 – Registro	
Condiciones previas	El cliente no está registrado en el sistema.
Acciones	El usuario introduce sus datos en el formulario de registro correspondiente y pulsa el botón de Registrar.
Resultado esperado	El sistema valida los datos introducidos por el usuario, los añade al sistema si todo está correcto y le redirige al panel de cliente.

Fuente: Elaboración propia

Tabla 12: Prueba 3 – Reserva de citas desde el panel

Prueba 3 – Reserva de citas desde el panel	
Condiciones previas	El cliente o trabajador han iniciado sesión en el sistema.
Acciones	El cliente o trabajador accede al apartado de Calendario de citas, introduce todos los datos requeridos y reserva la cita deseada pulsando el botón Aceptar.
Resultado esperado	El sistema comprueba que la cita no existe, la añade a la base de datos y confirma la cita al usuario a través de notificaciones.

Fuente: Elaboración propia

Tabla 13: Prueba 4 – Cancelar citas desde el panel

Prueba 4 – Cancelar citas desde el panel	
Condiciones previas	El cliente o trabajador han iniciado sesión en el sistema.
Acciones	El cliente o trabajador accede al apartado de Calendario de citas y cancela la cita pulsando el icono de la papelera al lado de la cita en cuestión.
Resultado esperado	El sistema elimina la cita de la base de datos y notifica al cliente sobre esta acción.

Fuente: Elaboración propia

Tabla 14: Prueba 5 – Visualización de facturas

Prueba 5 – Visualización de facturas	
Condiciones previas	El cliente o trabajador han iniciado sesión en el sistema.
Acciones	El cliente o trabajador accede al apartado de Facturas.
Resultado esperado	El sistema muestra una lista con todas las facturas para ese cliente o todas las facturas de todos los clientes de un determinado trabajador.

Fuente: Elaboración propia

Tabla 15: Prueba 6 – Descarga de facturas

Prueba 6 – Descarga de facturas	
Condiciones previas	El cliente o trabajador han iniciado sesión en el sistema.
Acciones	El cliente o trabajador accede al apartado de Facturas. Expande más información relacionada con una determinada fatura. Finalmente, pulsa el botón de descargar la factura.
Resultado esperado	El sistema genera un archivo pdf con todos los datos relacionados con la factura en cuestión y se descarga en el sistema.

Fuente: Elaboración propia

Tabla 16: Prueba 7- Procedimiento de pago

Prueba 7 – Procedimiento de pago	
Condiciones previas	El cliente ha iniciado sesión en el sistema.
Acciones	El cliente accede al apartado de Facturas. Si dispone de facturas pendientes de pago, hace clic en el botón de Pagar Ahora, selecciona el método de pago deseado y completa el proceso de pago.
Resultado esperado	El sistema recibe el resultado de la transacción y actualiza la factura en caso de pago correcto o notifica de cualquier error de pago al cliente.

Fuente: Elaboración propia

Tabla 17: Prueba 8 – Generación de notificaciones

Prueba 8 – Generación de notificaciones	
Condiciones previas	El cliente o trabajador pueden haber iniciado sesión en el sistema o no.
Acciones	Cualquier acción reacionada con el registro, la gestión de citas y facturas del sistema y comunicaciones entre cliente y trabajador.
Resultado esperado	El sistema detecta el tipo de acción realidada por el cliente o trabajador y genera una notificación por Email y/o SMS y/o a través de la propia web.

Fuente: Elaboración propia

Tabla 18: Prueba 9 – Generación de facturas

Prueba 9 – Generación de facturas	
Condiciones previas	El trabajador ha iniciado sesión en el sistema.
Acciones	El trabajador accede al apartado de Facturas. Tras esto, rellena todos los campos necesarios para una nueva factura, seleccionando al cliente que desea emitírsela.
Resultado esperado	El sistema recibe la petición de creación de una nueva factura con todos los datos necesarios, la añade a la base de datos y el cliente en cuestión obtiene una nueva factura y recibe una notificación.

Fuente: Elaboración propia

Tabla 19: Prueba 10 – Actalización de facturas

Prueba 10 – Actualización de facturas	
Condiciones previas	El trabajador ha iniciado sesión en el sistema.
Acciones	El trabajador accede al apartado de Facturas. Tras esto, pulsa el botón de Editar y modifica los campos de la factura deseados. Posteriormente, pulsa el botón Aceptar.
Resultado esperado	El sistema recibe la petición de actualización de una factura con todos los datos necesarios, la actualiza en la base de datos y el cliente recibe una notificación.

Fuente: Elaboración propia

Tabla 20: Prueba 11- Modificación de cuentas de clientes

Prueba 11 – Modificación de cuentas de clientes	
Condiciones previas	El trabajador ha iniciado sesión en el sistema.
Acciones	El trabajador accede al apartado de Mis Clientes. Tras esto, selecciona más detalles acerca de un cliente determinado. Pulsa el botón de Editar y modifica los datos del cliente. Posteriormente, pulsa el botón Aceptar.
Resultado esperado	El sistema recibe la petición de actualización de los datos de un cliente y los actualiza en la base de datos.

Fuente: Elaboración propia

Todas estas pruebas anteriormente descritas se centran en la funcionalidad del sistema.

Los siguientes casos de prueba se centraran en los principales requisitos no funcionales:

Tabla 21: Prueba 12 – Muestra de errores

Prueba 12 – Muestra de errores	
Condiciones previas	Ninguna
Acciones	Cualquier acción errórea por parte de cualquier usuario.
Resultado esperado	El sistema notifica del error al usuario mediante popups informativos.

Fuente: Elaboración propia

Tabla 22: Prueba 13 – Diseño responsive

Prueba 13 – Diseño responsive	
Condiciones previas	Ninguna
Acciones	Ninguna
Resultado esperado	Todas y cada una de las vistas del sistema se presentan con un estilo totalmente adaptable a diferentes tamaños de pantalla de dispositivos.

Fuente: Elaboración propia

Tabla 23: Prueba 14 – Soportar conexiones simultáneas

Prueba 14 – Soportar conexiones simultáneas	
Condiciones previas	Ninguna
Acciones	Diversos usuarios acceden al mismo tiempo al sistema.
Resultado esperado	El sistema gestiona a la perfección todas las peticiones que cualquier usuario realice dentro del sistema.

Fuente: Elaboración propia

Tabla 24: Prueba 15 – Bajo tiempo de respuesta

Prueba 15 – Bajo tiempo de respuesta	
Condiciones previas	Ninguna
Acciones	Un usuario accede al sistema.
Resultado esperado	El sistema responde y gestiona todas las peticiones para cada una de las vistas en segundos.

Fuente: Elaboración propia

6.2 Resultados de las pruebas

La siguiente tabla recoge los resultados de cada una de las pruebas descritas previamente. Un total de 10 usuarios simultáneos han accedido a realizar estas pruebas en concreto. Estas pruebas se han llevado a cabo tanto en la aplicación web como en la aplicación móvil:

Tabla 25: Resultados obtenidos en cada una de las pruebas

Nº Prueba	Aplicación web	Aplicación móvil
1	Resultado esperado	Resultado esperado
2	Resultado esperado	Resultado esperado
3	Resultado esperado	Resultado esperado
4	Resultado esperado	Resultado esperado
5	Resultado esperado	Resultado esperado
6	Resultado esperado	Resultado esperado
7	Resultado esperado	Resultado esperado
8	Resultado esperado	Resultado esperado
9	Resultado esperado	Resultado esperado
10	Resultado esperado	Resultado esperado
11	Resultado esperado	Resultado esperado
12	Resultado esperado	Resultado esperado
13	Resultado esperado	Resultado esperado
14	Resultado esperado	Resultado esperado
15	Resultado esperado. El tiempo de respuesta se encuentra entre 2 segundos y 5 segundos (con conexión estable y normal).	Resultado esperado. El tiempo de respuesta se encuentra entre 2 segundos y 5 segundos (con conexión estable y normal).

Fuente: Elaboración propia

Es importante mencionar que todas estas pruebas se han llevado a cabo mediante la aplicación tanto en despliegue local como en su despliegue online. Para la aplicación móvil se ha utilizado el emulador de Android Studio y la aplicación instalada en diferentes dispositivos de forma completa.

Los resultados obtenidos muestran un claro éxito a la hora del desarrollo del sistema y nos permiten afirmar que se cumplen los requisitos principales establecidos para este proyecto.

7

Conclusiones y futuras mejoras

Tras el diseño y desarrollo de este proyecto, hemos obtenido una aplicación web y móvil para clínicas dentales que cumple con todos y cada uno de los objetivos establecidos inicialmente para este trabajo. Una aplicación con aspecto profesional que presenta una solución eficiente a los retos y deficiencias que poseen otras aplicaciones del mismo ámbito. Este sistema permite unificar la gestión de un conjunto de clínicas dentales en una sola plataforma.

Este proyecto demuestra con éxito un enfoque eficiente para el desarrollo de una aplicación web y móvil que no se limita a clínicas dentales, sino que permite aplicar este enfoque para cualquier tipo de aplicación. El personal de las clínicas dentales podría gestionar eficazmente sus preferencias, las citas de los clientes y las facturas. Del mismo modo, los clientes podrían gestionar sus facturas, beneficiarios, citas, cuentas y preferencias de notificación. Tanto el personal de la clínica como los clientes podían comunicarse entre sí y utilizar un ordenador de sobremesa o un smartphone Android

para acceder a la aplicación de la clínica dental. Además, uno de los enfoques llevados a cabo durante el desarrollo de la misma ha sido facilitar el mantenimiento de la aplicación y la escalabilidad para futuras mejoras.

Durante todo el desarrollo, se han utilizado una gran cantidad de tecnologías punteras como C#, MSSQL o el patrón MVC. Los conocimientos obtenidos por mi parte en la Universidad de Málaga no han sido sobre estas tecnologías en específico, sino que me han facilitado la comprensión y estudio de cualquier nueva tecnología gracias al conocimiento de las bases de cualquier aplicación. Por supuesto, esto no ha sido fácil y ha requerido horas y horas delante de la pantalla para lograr desarrollar este proyecto tan complejo y grande. La mayor parte de estas tecnologías se encuentran hoy en día en el top de tecnologías más demandadas por las empresas de desarrollo Fullstack.

Además, este proyecto me ha permitido cumplir mi objetivo personal que era ofrecer una aplicación que aporte soluciones reales a un determinado ámbito y, si todo va bien, continuar con su desarrollo y ofrecerla a diversas empresas interesadas. Además, he conseguido dominar todas las tecnologías elegidas en este proyecto como reto para prepararme de cara al mundo profesional.

A pesar del éxito en la implementación y el desarrollo de la aplicación web y móvil, se podrían añadir algunas mejoras para aumentar la funcionalidad y la productividad de la aplicación. El modo en que accedemos y utilizamos las aplicaciones web y móviles hoy en día ha cambiado nuestra forma de vida. Por ello, es necesario seguir ofreciendo nuevas funcionalidades que nos faciliten aun más nuestra forma de comunicarnos y llevar a cabo tareas contidianas:

- **Mejora en el sistema de pagos**

Este proyecto ha implementado el sistema de pago mediante PayPal. Aunque este sistema es funcional en la aplicación, hay algunas nuevas tendencias de sistemas de

pago que los clientes han conocido en los últimos años y serían recomendables para añadirlos. Algunos ejemplos son:

- **Bizum**
- **PaysafeCard**
- **Google Pay**

Figura 45: Métodos de pago alternativos

Fuente: Google.com

- **Mejoras en el sistema de notificaciones**

Este trabajo desarrolló con éxito un sistema de notificación dinámico, que proporciona un flujo de comunicación fluido entre los clientes y los trabajadores de la clínica dental. Sin embargo, el enfoque de la comunicación puede mejorarse añadiendo las tendencias actuales de comunicación, como:

- **WhatsApp**
- **Telegram**

Figura 46: Aplicaciones de mensajería instantánea

Fuente: Google.com

Además de estas nuevas plataformas, será necesario mejorar el diseño de las notificaciones recibidas a través de email y en la propia aplicación que permitan leer las notificaciones de forma rápida y sencilla utilizando diferentes colores e imágenes agradables para el usuario.

- **Selección de diferentes lenguajes**

Este proyecto ha sido desarrollado utilizando una única preferencia de idioma: el español. En el futuro sería importante permitir otros idiomas como el alemán y el inglés puesto que mejoraría la productividad de la aplicación. La razón es que Málaga, por ejemplo, es una provincia multicultural con muchos habitantes que no hablan español. Estos habitantes son clientes potenciales de las clínicas dentales.

- **Aplicar un conjunto de pruebas intensivas**

Aunque en este proyecto se han llevado a cabo diversas pruebas de funcionamiento y rendimiento, sería recomendable en un futuro llevar a cabo un conjunto de pruebas más complejas y profundas que permitan analizar la estabilidad y funcionamiento del sistema ante situaciones difíciles, incluso ante posibles ataques informáticos, llevando a cabo un estudio sobre el nivel de exposición a las amenazas de la aplicación.

- **Desarrollo en otras plataformas**

Aunque hoy en día los ordenadores de sobremesa y los dispositivos móviles Android son los más utilizados por usuarios en todo el mundo, no podemos olvidarnos de aquellos usuarios que utilizan dispositivos IOS o prefieren utilizar una aplicación de escritorio (WPF) en vez de usar el navegador web. Estas posibilidades están añadidas dentro del funcionamiento del sistema y solo sería necesario replicar la idea utilizada en el desarrollo de la aplicación Android para el resto de plataformas.

- **Mejora del sistema de privilegios**

Para establecer un mayor control y añadir nuevas funcionalidades en el sistema, es importante establecer un mejor sistema de privilegios donde se detallen más permisos para cada uno de los usuarios de la aplicación. Por ejemplo, añadir permisos para los propietarios de la clínica, los cuales en podrían añadir nuevos trabajadores al sistema.

Referencias

Amlanjyoti, S., Sherin, J., Dhondup, D., & Roseline, M. R (2015). *Comparative Performance Analysis of MySQL and SQL Server Relational Database Management Systems in Windows Environment*. IJARCCE. Recuperado 2 de noviembre de 2021, de https://www.ijarcce.com/upload/2015/march-15/IJARCCE%2039.pdf

Arsenault, C. (2017). *HTML vs HTML5 - What Is the Difference?* Key CDN. Recuperado 15 de octubre de 2021, de https://www.keycdn.com/blog/html-vs-html5/

Author, G. (2015). *Why is a web application a must in today's business?* Your Story. Recuperado 15 de octubre de 2021, de https://yourstory.com/2015/07/web-application/amp

Chapple, M. (2021). *Definition of Database Relation*. Lifewire. Recuperado 2 de noviembre de 2021, de https://www.lifewire.com/relation-definition-1019260

Gibb, R. (2016). *What is a Web Application?* StackPath. Recuperado 15 de octubre de 2021, de https://blog.stackpath.com/web-application/

Gray Grids Inc.(2021). *The 7 Great Advantages and Reasons to Use Bootstrap as a Front-end CSS Framework*. GrayGrids. Recuperado 16 de octubre de 2021, de https://graygrids.com/advantage-reasons-use-bootstrap-front-end-css-framework/

Johnson, J. (2021). *Global digital population as of January 2021*. Statista. Recuperado 15 de octubre de 2021, de https://www.statista.com/statistics/617136/digital-population-worldwide/

Joshi, A. (2020). *Difference Between CSS and Bootstrap*. GeeksforGeeks. Recuperado 16 de octubre de 2021, de https://www.geeksforgeeks.org/difference-between-css-and-bootstrap/

JQuery. (2020). *What is jQuery?* JQuery. Recuperado 16 de octubre de 2021, de https://jquery.com/

Jones, M. (2019). *Dapper vs. EF Core Query Performance Benchmarking.* Recuperado 16 de octubre de 2021, de https://exceptionnotfound.NET/dapper-vs-entity-framework-core-query-performance-benchmarking-2019/

LeBLanc, B., Schonning, N., Coulter, D., Sherer, T., George, A. D. et al. (2021). *LINQ to SQL.* Microsoft Docs. Recuperado 16 de octubre de 2021, de https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/linq/

Microsoft MVC. (2021). *ASP.NET MVC Pattern.* Microsoft. Recuperado 8 de noviembre de 2021, de https://dotnet.microsoft.com/apps/aspnet/mvc

Mozilla Developer. (2021a). *CSS: cascading style sheets.* MDN Web Doc. Recuperado 15 de octubre de 2021, de https://developer.mozilla.org/en-US/docs/Web/CSS

Mozilla Developer. (2021b). *JavaScript.* MDN Web Doc. Recuperado 16 de octubre de 2021, de https://developer.mozilla.org/en-US/docs/Web/JavaScript

Otto, M. (2012). *Building Twitter Bootstrap.* Hyperlink Infosystem. Recuperado 16 de octubre de 2021, de https://www.hyperlinkinfosystem.com/blog/the-importance-of-mobile-applications-in-everyday-life

Oza, H. (2017). *The Importance Of Mobile Applications In Everyday Life!.* A-List Apart. Recuperado 15 de octubre de 2021, de https://alistapart.com/article/building-twitter-bootstrap/

QODE.(2015). *What is a Web App?* QODE. Recuperado 15 octubre de 2021, de https://www.qode.pro/blog/what-is-a-web-app/

Rose, P. (2019). *The Importance of Mobile App Development For Businesses.* Nex Pixel. Recuperado 16 de octubre de 2021, de https://www.nexpixel.com/the-importance-of-mobile-app-development-for-businesses/

Shankar, M. (2013). *Using LINQ in .NET.* CSharp Corner. Recuperado 5 de noviembre de 2021, de https://www.c-sharpcorner.com/UploadFile/c5c6e2/using-linq-in-net/

Shankar, R. (2021). *C# vs. Python.* Blog post. Recuperado 16 de octubre de 2021, de https://hackr.io/blog/c-sharp-vs-python

Thompson, B. (2020). *What is .NETFramework? Explain Architecture & Components*. Guru 99. Recuperado 16 de octubre de 2021, de https://www.guru99.com/net-framework.html
Toprek, M. (2020). *What is MSSQL?* Medium. Recuperado 16 octubre de 2021, de https://medium.com/@toprak.mhmt/what-is-mssql-9a152d7d4ed0

Wagner, B., Sharkey, k., Coulter, D., Newcomb, B., Warren, G. et al. (2019). *A tour of the C# language*. Microsoft Docs. Recuperado 16 de octubre de 2021, de https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

World Wide Web Consortium[W3C]. (2014). *HTML5 IS A W3C RECOMMENDATION*. W3. Recuperado 15 de octubre de 2021, de https://www.w3.org/blog/news/archives/4167

Apéndice A

Manual de Instalación

Existen dos principales formas para la instalación de este proyecto en un sistema: instalación local, despliegue productivo y instalación móvil. Cada uno de estos despliegues requiere de una serie de requisitos y pasos que pasaremos a explicar. Debido a las diferentes configuraciones que un sistema puede presentar y a la complejidad de los requisitos, este manual ofrecerá las principales ideas de instalación.

1. Instalación Local

Requerimientos

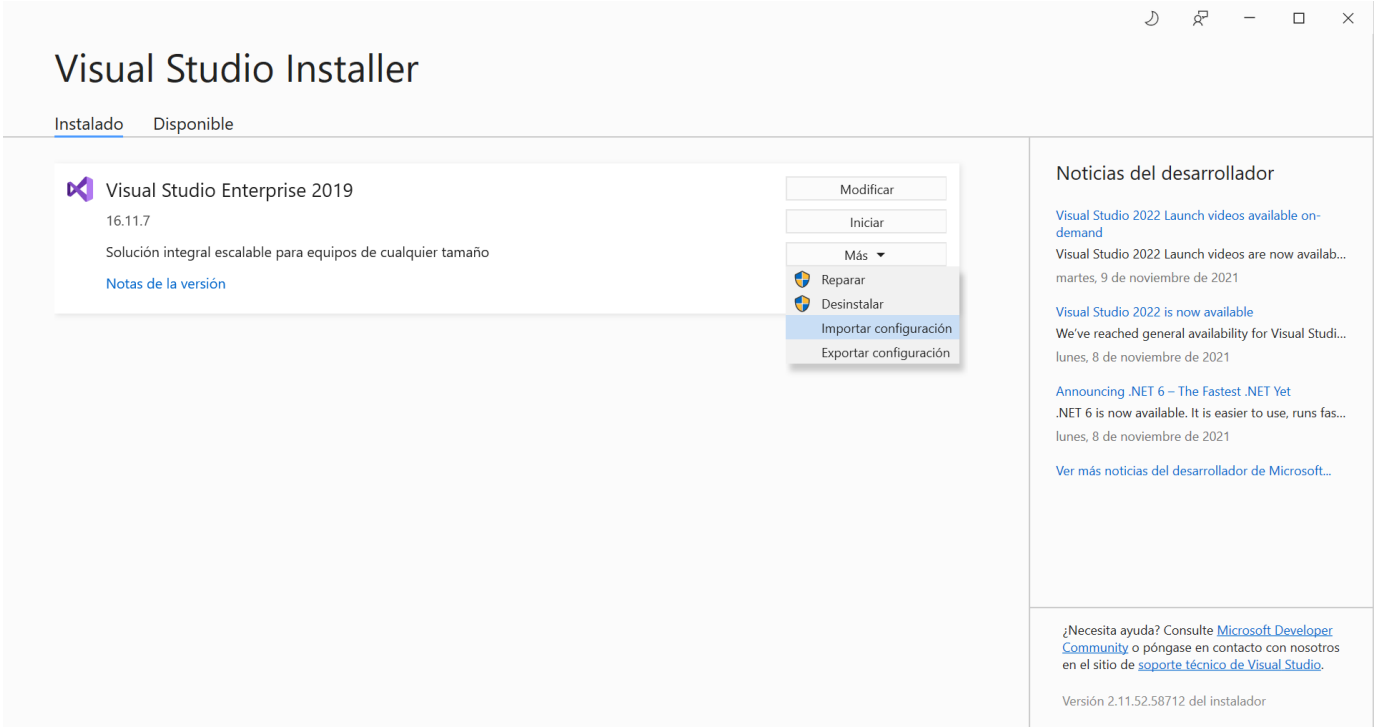
- Visual Studio 2019 en cualquiera de sus versiones (Community o Enterprise)
- Visual Studio 2019 Enterprise
- Visual Studio 2019 Community
- Microsoft SQL Server Express
  https://www.microsoft.com/es-es/sql-server/sql-server-downloads
- Microsoft SQL Server Management Studio
- https://aka.ms/ssmsfullsetup

Debido a la gran cantidad de componentes usados, se recomienda utilizar el IDE de Visual Studio para ejecutar la aplicación de forma local e instalar todas las dependencias.

En primer lugar, será necesario llevar a cabo la instalación de Visual Studio 2019. Tras abrir el instalador, será necesario dejar los paquetes por defecto y esperar a su instalación. Antes de continuar, será necesario importar la configuración de Visual Studio con todos los componentes necesarios. Para ello:

a. Abrimos Visual Studio Installer y seleccionamos el Importar configuración en el apartado Más.

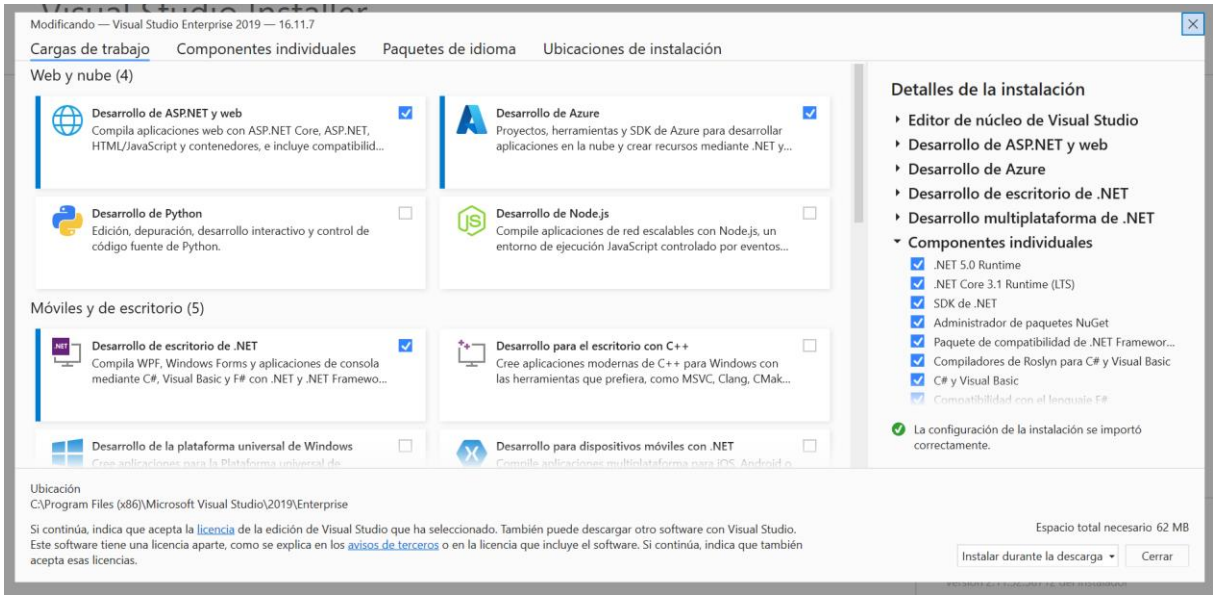
Figura 47: Visual Studio Installer



Fuente: Elaboración propia

b. Buscamos la ubicación del archivo **config.vsconfig** contenido dentro del fichero *.zip* inicial. Después le damos a *Revisar detalles*.

Figura 48: Visual Studio Installer – Importar configuración



Fuente: Elaboración propia

Finalmente será necesario hacer clic en *Instalar durante la descarga*.

Mientras Visual Studio se descarga, podemos descargar primeramente Microsoft SQL Server Express y Microsoft SQL Server Management Studio. Para proceder con la instalación de ambos, será necesario seguir el siguiente tutorial para la creación de una base de datos local y acceder a través de SQL Server Management Studio: https://www.jasoft.org/Blog/post/que-es-sql-server-localdb-como-instalarlo-usarlo-y-actualizarlo

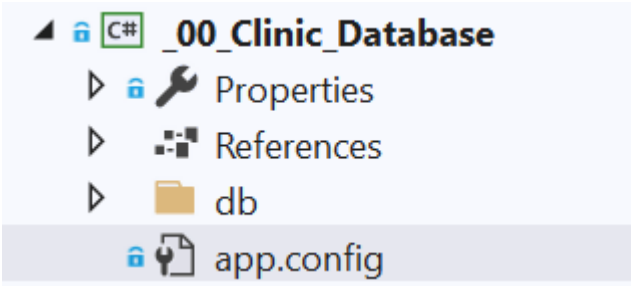
Para que la aplicación se conecte a la base de datos, será necesario establecer la cadena de conexión específica. Esto dependerá de tu SQL Server. Si has seguido el tutorial anterior, la cadena de conexión de la base de datos es la siguiente:

```
Data Source=(LocalDB)\MSSQLLocalDB;Integrated
Security=true;AttachDbFileName=E:\Datos\MiBDD.mdf
```

Para establecer la cadena de datos, debemos primeramente importar el proyecto a Visual Studio.

Tras importarlo, debemos ir al proyecto **00_Clinic_Database** y modificar el fichero **app.config**.

Figura 49: Fichero de configuración de la base de datos



Fuente: Elaboración propia

Aquí es donde estableceremos nuestra cadena de conexión a la base de datos local de en el campo ***connectionString***:

Figura 50: Cambio de la cadena de conexión a la BBDD

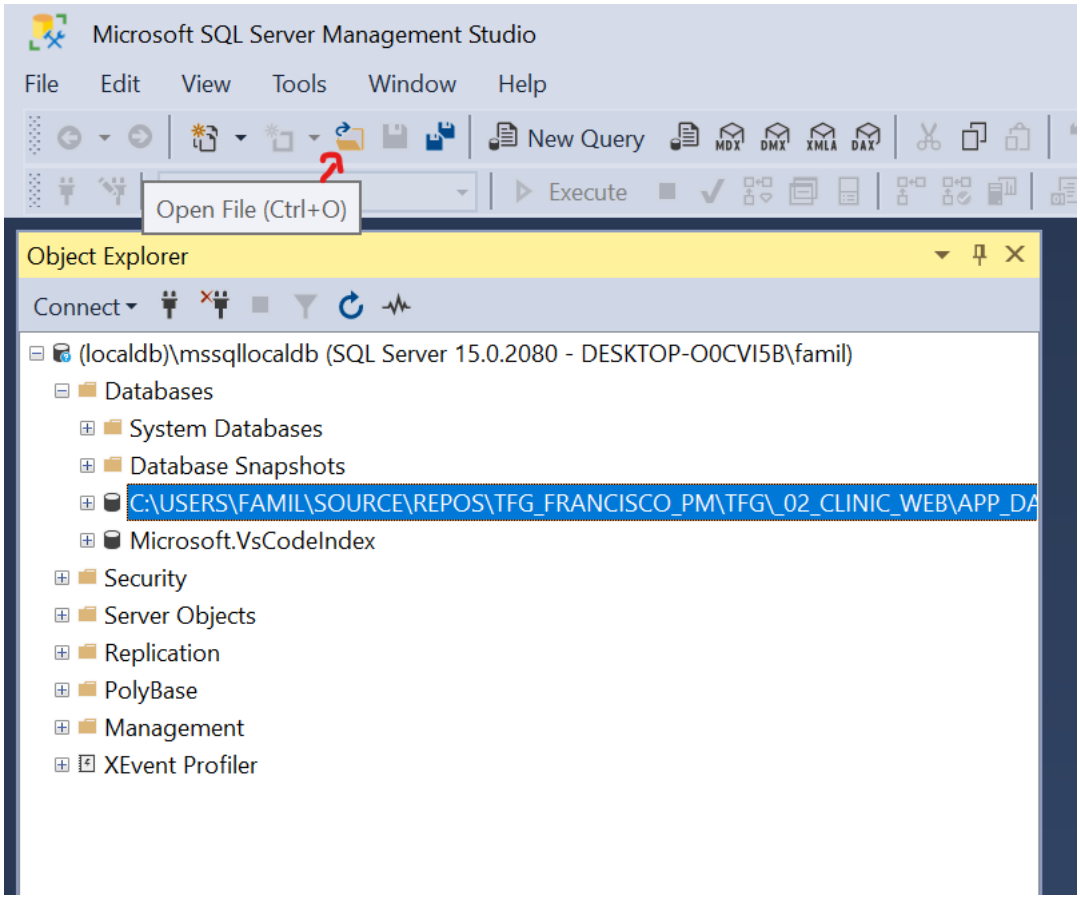
```
app.config*
1  <?xml version="1.0" encoding="utf-8" ?>
2  <configuration>
3    <configSections>
4    </configSections>
5    <connectionStrings>
6      <!--Local db-->
7      <add name="_00_Clinic_Database.Properties.Settings.fran_tfg_dbConnectionString"
8          connectionString="CONNECTION_STRING"
9          providerName="System.Data.SqlClient" />
10   </connectionStrings>
11 </configuration>
```

Fuente: Elaboración propia

Para popular la base de datos, será necesario acceder a esta a través de SQL Server Management Studio y ejecutar el **SQLQuery1.sql** incluido en los archivos del proyecto.

Cuando abrimos la base de datos con SQL Server Management Studio, debemos abrir el script neesario a ejecutar:

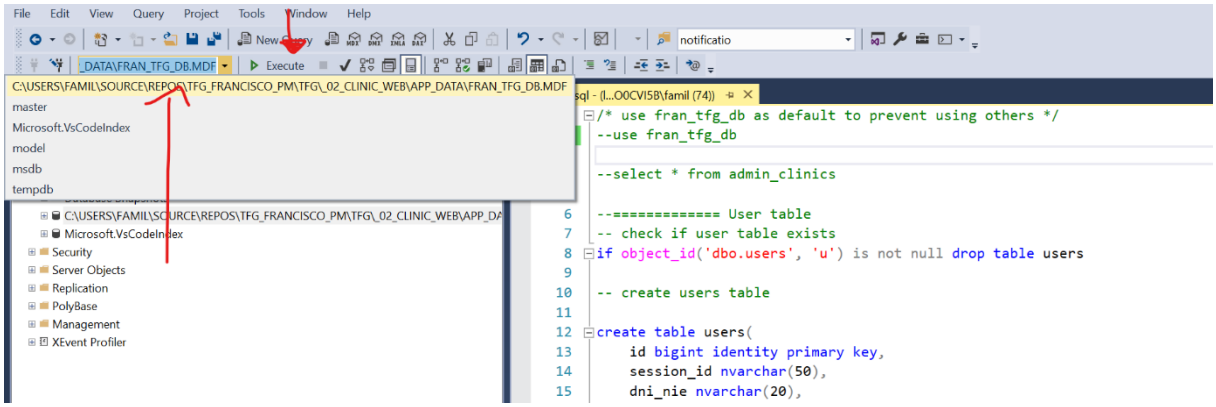
Figura 51: Abrir script de creación de datos



Fuente: Elaboración propia

Finalmente seleccionamos la base de datos donde deseamos añadir los datos y pulsamos el botón ***Execute***:

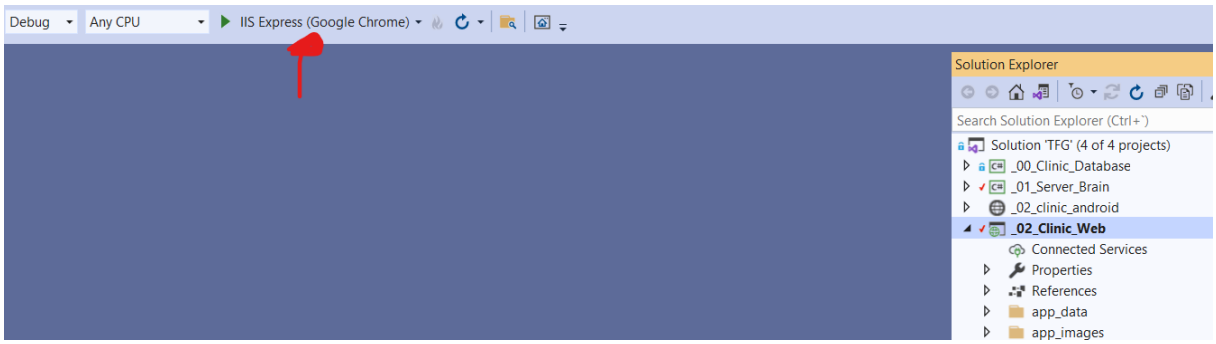
Figura 52: Selección de la base de datos y ejecución del script



Fuente: Elaboración propia

Para iniciar el proyecto, será necesario volver a Visual Studio y ejecutar el proyecto 02_Clinic_Web. Tras esto, se nos abrirá nuestro navegador por defecto para acceder a la aplicación:

Figura 53: Ejecución de la aplicación



Fuente: Elaboración propia

Figura 54: Aplicación en ejecución

Fuente: Elaboración propia

2. Depliegue productivo

Requerimientos

- Comprar un servidor Microsoft Windows Server.
- Microsoft SQL Server Express
  https://www.microsoft.com/es-es/sql-server/sql-server-downloads
- Microsoft SQL Server Management Studio
- https://aka.ms/ssmsfullsetup

Para desplegar la aplicación en línea será necesario comprar un servidor Windows e instalar todos los componentes que requiere la aplicación (visita el instalador de Visual Studio importando la configuración manual para ver todos estos componentes).

Tras esto, será necesario crear la base de datos y popularla de la misma forma que se hace en el despliegue local pero ahora desde el servidor. Será también necesario configurar los puertos para que la base de datos sea accesible a través de internet.

Para importar el proyecto al servidor, será necesario seguir el siguiente tutorial donde nos explican paso a paso como importar un proyecto ASP .NET en un servidor IIS:
https://anexsoft.com/como-configurar-y-publicar-en-el-iis-un-proyecto-asp-net
El paso más importante será configurar nuestro servidor para que exponga públicamente el puerto donde se encuentra el servidor y sea accesible por cualquier persona.

Opcionalmente, es posible comprar un domino para que cualquiera que desee acceder a la aplicación, pueda accerlo a través del dominio. Existen numerosas páginas que permiten comprar dominios y configurarlos a su gusto. Este proyecto ha utilizado **Namecheap**.

3. Instalación móvil

Para llevar a cabo la instalación en dispositivos Android será necesario una serie de pasos:

1. Descargar el fichero ***.apk*** en el dispositivo móvil donde se desea instalar la aplicación.
2. Activar los **orígenes desconocidos** en el móvil

Esta opción se activa de forma diferente en cada uno de los dispositivos móviles. Es por ello que para saber como instalar esta funcionalidad, se recomienda que busque a través de internet el nombre de su dispositivo junto con "activar orígenes desconocidos".

3. Tras activar esta opción, podemos abrir el archivo .apk almacenado en nuestro dispositivo y seguir los pasos de instalación.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA