Appoint

Trabajo de fin de grado
Memoria

Santi Otín Marco

Universidad Politécnica de Cataluña
Facultad de Informática de Barcelona
Especialización en Ingeniería del Software
Claudia Ayala - Directora
Joan Subirats Soler - Tutor de GEP
18 de enero de 2021

Resumen

Este documento es la memoria del Trabajo de Fin de Grado de la Universidad Politécnica de Cataluña para el grado de Ingeniería Informática con especialidad en Ingeniería del Software. El trabajo consiste en el análisis, diseño e implementación de un prototipo de aplicación móvil que permita a clientes de pequeños negocios poder pedir cita previa en estos a través de internet. El sistema realizado está basado en una arquitectura en capas y utiliza tanto tecnologías tratadas en el grado como las tecnologías más utilizadas en la industria móvil. El objetivo de esta memoria es el de documentar todo el proceso de realización del trabajo.

Resum

Aquest document és la memòria del Treball de Final de Grau de la Universitat Politècnica de Catalunya pel grau d'Enginyeria Informàtica amb especialitat en Enginyeria del Software. El treball consisteix en l'anàlisi, disseny i implementació d'un prototip d'aplicació mòbil que permeti als clients de petits negocis poder demanar cita prèvia en aquests a través d'internet. El sistema realitzat està basat en una arquitectura en capes i utilitza tant tecnologies tractades en el grau com les tecnologies més utilitzades en la indústria mòbil. L'objectiu d'aquesta memòria és el de documentar tot el procés de realització del treball.

Abstract

This document is the report of the Degree Final Project of the Polytechnic Faculty of Catalonia on the Bachelor Degree of Informatics Engineering, on the major Software Engineering. The work consists of the analysis, design and implementation of a mobile application prototype that allows clients in small businesses to request online appointments. The system is based on a layered architecture and uses technologies learned in the bachelor degree and the most used in the mobile industry. The aim of this report is to document the whole process of carrying out the work.

Índice de contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

1.1 Contextualización

El trabajo de fin de grado (TFG) es una asignatura obligatoria de la carrera de Ingeniería Informática, impartida por la Facultad de Informática de Barcelona (FIB). Tiene el objetivo de poner en práctica todos los conocimientos adquiridos a lo largo del grado, así como las competencias específicas de la especialidad escogida, en mi caso Ingeniería del Software. Estos conocimientos y habilidades se ponen en práctica a través de la realización de un proyecto de carácter profesional. Este trabajo de fin de grado se ha hecho dentro de la modalidad A, es decir, es un proyecto realizado en la UPC.

Actualmente, las necesidades de comunicación del mundo actual están cada vez más sesgadas hacia internet. Las personas tienden a interactuar cada vez más mediante este sistema y no con las tradicionales llamadas telefónicas, afectando de forma significativa a los negocios, sobre todo a los que cuyo modelo económico se basa en la venta de servicios o productos al cliente o consumidor final. Estos negocios están sujetos a las necesidades y comportamientos de las personas, teniéndose que adaptar a estos nuevos comportamientos. Hay sectores cuyos negocios se han sabido adaptar a las demandas tecnológicas de sus clientes, algunos ejemplos son el sector de alojamientos y el de restauración. Según un estudio de Google España realizado en 2017 [1], el 73% de las personas que hicieron reservas de alojamientos y vuelos lo hicieron online. No es de extrañar entonces que la mayoría de estos negocios ofrezcan la posibilidad de realizar reservas de forma online y que además existan herramientas que permitan reservar en distintos negocios desde el mismo sitio, como podría ser Booking [2].

Figura 1: Porcentaje de reservas de hoteles y vuelos hechas en España en 2017. Fuente: Google España.

Recientemente, se le ha añadido otro factor determinante en esta transición hacia internet. La pandemia mundial que se está viviendo está acelerando mucho esta transición en todos los sectores, haciendo que esta necesidad de transformación digital no solo se vuelva crucial, sino un requisito en diversas etapas o fases de pandemia. Un estudio realizado por ElTenedor [3] afirma que en el verano de 2020 las reservas online en restaurantes han crecido un 63%. Pero ya no se trata únicamente de algo que los usuarios quieran, sino que durante la pandemia del 2020 ha habido momentos que desde las instituciones gubernamentales se ha prohibido a todos los negocios dar servicios sin cita previa.

Ya sea por un motivo de cambio en los usuarios, que prefieren internet como forma de interacción predeterminada, o por un motivo legal debido a las situaciones excepcionales que se han empezado a vivir, los negocios deben adaptarse a esta forma de funcionar. Y si bien ya hay muchos sectores muy bien adaptados, hay muchos negocios, mayoritariamente pequeños, que por unas u otras circunstancias aún no ofrecen interacciones a través de internet.

1.2 Términos y conceptos

En este apartado se ha redactado una lista con diferentes términos con el fin de especificar el significado que se les va a dar dentro de sus múltiples interpretaciones para no confundir al lector a lo largo de todo el trabajo. Además, también se van a explicar ciertos conceptos relacionados con el tema que es objeto de estudio.

- **Negocio**: Cuando se hable de negocio, se estará hablando de una entidad creada con el fin de obtener un beneficio económico, generalmente a través de la realización de actividades de producción de productos, comercialización de productos o prestación de servicios, que benefician a las personas o a otros negocios. Ejemplos de negocio serían las peluquerías o las clínicas dentales.

- **Servicio**: Se entiende como servicio un conjunto de actividades que buscan satisfacer las necesidades de un cliente. Siguiendo el ejemplo anterior, un corte de pelo o una limpieza bucal serían ejemplos de servicios realizados por los negocios anteriores.

- ***Marketplace***: Un *marketplace* es una plataforma de distribución donde los distintos negocios ofrecen sus productos y servicios, del mismo modo que lo hacen los centros comerciales offline con productos y servicios de las tiendas físicas.

- **Gestor de reservas:** Sistema encargado de almacenar las reservas realizadas, así como otras funciones relacionadas con su gestión, como podría ser la creación o eliminación.

1.3 Descripción del problema

Incluso hoy en día existen muchos negocios de distintos sectores que aún no disponen de un canal online para que sus clientes puedan reservar cita previa en ellos. Esta situación es debida a que estos negocios siguen funcionando con métodos tradicionales para la gestión de las citas, utilizando una libreta (o varias) para llevar un control de la reserva de horas de los servicios que proporcionan, método incompatible con el mundo online. Estos negocios suelen ser de tamaño pequeño y es por eso por lo que aún no se han podido permitir dar este salto tecnológico, sin embargo, este cambio cada vez va siendo más necesario debido a los problemas que existen con el modelo tradicional.

El problema principal de llevar una gestión manual y por lo tanto no poder ofrecer la posibilidad de pedir cita online es que solo se deja a los clientes la posibilidad de hacerlo de las siguientes dos maneras. La primera es ir hasta el sitio para pedirla presencialmente, con la consecuente pérdida de tiempo que supone esto en función de lo lejos que esté el sitio.

Además, no se puede ir a cualquier hora, sino que se ha de buscar primero el horario del sitio para asegurarse de que no está cerrado.

La otra opción es llamar al negocio, una forma más cómoda pero que también tiene sus contras, como que hay que buscar el teléfono del sitio y hay que esperar a que un trabajador coja la llamada. Para cualquier imprevisto que sufra el cliente y tenga que modificar la hora reservada o incluso cancelarla, tendrá que volver a realizar el proceso de llamar o ir otra vez.

Los negocios tampoco obtienen muchas ventajas de la gestión manual de las citas previas, ya que o bien los empleados del negocio han de estar pendientes de atender las llamadas y a las personas que se presentan ahí o bien se ha de contratar a personas expresamente para hacer este trabajo. Cualquiera de estos dos casos es perjudicial para el negocio, ya que en uno estará haciendo que sus trabajadores sean menos productivos y en el otro estará teniendo un gasto extra en la contratación de una o varias personas más para hacer ese trabajo.

A este problema se le suma que la gestión de reservas en una libreta es muy limitada, ya que impide al negocio recabar toda la información que se podría obtener, además de dificultar la capacidad de análisis de las reservas al estar escritas en papel. Mucho más complicada se hace esta gestión cuando se dispone de más de una libreta (o la combinación de libretas y hojas de Excel), porque en ese caso la información no estará unificada en un mismo sitio y puede darse el caso de que se asignen varias reservas a la misma hora. Otra gran desventaja a mencionar es que al depender de una o varias personas, las reservas solo se pueden hacer cuando estas estén, y en el mejor de los casos se podrán hacer mientras el negocio se encuentre abierto.

A todo lo mencionado anteriormente, ahora se le añade la pandemia del covid, que ha provocado que muchos negocios se hayan tenido que adaptar a las nuevas normativas instauradas. Algunas de estas normas limitan el acceso a los negocios en función de la gente que haya dentro, para evitar así las aglomeraciones de personas. Esto perjudica a los negocios, ya que posibles clientes pueden dejar de entrar debido a esta nueva situación. Además, en muchos sitios se impide la entrada a todo aquel que no tenga cita previa, forzando a los clientes a avisar con antelación para que el negocio pueda gestionar los espacios sin incumplir ninguna normativa.

Resumiendo, el método manual de gestión de citas previas que usan los pequeños negocios origina el problema de no poder ofrecer citas previas online, además de acarrear muchos otros problemas.

1.4 *Stakeholders*

Una vez definido el problema y sus orígenes, se va a hablar de los *stakeholders.* Los *stakeholders* son las denominadas partes interesadas en el proyecto y que por lo tanto son afectadas por la existencia o la actividad de este. Estas partes son de gran importancia para el proyecto, ya que nos ayudarán a definir los requisitos del sistema a través de sus demandas. Por lo tanto, en función de lo bien que se identifiquen estas partes interesadas, encontraremos los requisitos más o menos adecuados para el proyecto.

También es importante mencionar que no solo se ha de identificar bien estas partes, sino intentar mantenerlas involucradas en el proyecto, porque puede que sus necesidades cambien y en ese caso se deberán introducir o quitar requisitos que ellos nos ayudarán a identificar.

Pequeños negocios

Los pequeños negocios existentes son uno de los *stakeholders* más importantes de este proyecto. Concretamente, aquellos negocios que no dispongan de un *software* de gestión de reservas o que dispongan de uno muy básico.

Empleados de los negocios

Los empleados de los negocios que adopten la solución que se desarrollará se verán directamente afectados, ya que esto implica cambios tanto en la forma de trabajar como en la de organizarse.

Clientes

Otro *stakeholder* muy importante van a ser los clientes de los negocios, ellos van a poder beneficiarse de las mejoras que se van a introducir en el proceso de realización de citas previas, así como van a obtener otra forma de conocer nuevos negocios.

Potenciales clientes

Las personas que no sean clientes de estos pequeños negocios también van a ser afectados por este proyecto debido a que la nueva solución y sus mejoras para los pequeños negocios puede que sea determinante para que estas personas empiecen a ir a estos pequeños negocios.

Competencia

Las plataformas que ya ofrecen soluciones para este problema o similares se verán afectadas debido a que entrará un nuevo participante en el mercado, creando más competencia y teniendo que esforzarse más para distinguirse.

Tutora del TFG

La tutora de este trabajo de final de grado, Claudia Ayala, es uno de los actores interesados. Está implicada tanto con la dirección y asesoramiento del proyecto como con la correcta finalización de este.

Realizador del TFG

El autor de este trabajo también es uno de los principales *stakeholders* de este proyecto, ya que será el encargado de realizarlo de la mejor forma posible porque de ello depende la superación de este trabajo y la finalización de la carrera. Además, este trabajo influirá en su aprendizaje y también en el mundo laboral posterior.

2. Justificación

2.1 Análisis de mercado

Una vez ya hemos hablado de la necesidad que creemos que tiene el mercado acerca de una herramienta para gestionar las citas previas, vamos a analizar qué tipo de herramientas existen relacionadas con este tema, así como sus funcionalidades. Para empezar, hay que diferenciar entre los llamados gestores de reservas y los *marketplaces* de citas previas.

Los primeros suelen ser *softwares* encargados de la gestión de reservas desde el lado del negocio, es decir, su tarea es proporcionar el servicio de cita previa para el negocio, así como cualquier tipo de gestión de esta reserva. Estos *softwares* suelen integrarse con la página web del servicio para que el cliente pueda desde ahí reservar la hora que quiera. Una vez reservada queda registrada y al servicio le queda constancia, pudiendo interactuar con la reserva de la manera apropiada. Para entenderlo más fácilmente tenemos que pensar que una libreta donde se apuntan las reservas es el gestor de reservas más básico, en este caso un software gestor de reservas podría ser una libreta digital.

Por otro lado, están los *marketplaces* de reservas, plataformas que incluyen distintos negocios permitiendo al usuario visualizarlos todos y pedir cita en cualquiera de ellos desde el *marketplace*. Al englobar muchos negocios, estas plataformas pueden ofrecer otras funcionalidades, como la búsqueda según ciertas características, ya sea por tipo de negocio, proximidad, valoraciones, etc.

Podemos observar que la diferencia entre ambos reside en que los *marketplaces* de reservas engloban muchos negocios y permiten que el usuario pueda verlos todos y reservar en cualquiera de ellos, mientras que los gestores de reservas son los encargados de mantener un control de las reservas que le llegan a un negocio, ya sea a través de la web, de las redes sociales, de un *marketplace*, etc. Sin embargo, puede darse que un *marketplace*, además de hacer de canal de reservas para los distintos negocios que están en él, también funcione como gestor de reservas para estos mismos negocios. Si no es el caso, cada vez que se haga una reserva de un negocio desde un *marketplace*, el *marketplace* será el encargado de enviarle esta reserva al gestor de reservas que use el negocio para poderla gestionar. Para poder participar en un *marketplace*, es necesario que el negocio tenga un software gestor de citas previas, ya sea propio o del *marketplace* al que pertenece, ya que si no lo tiene y funciona con un gestor de reservas manual (libreta) el software del *marketplace* no sabría leer ni apuntar en la libreta las reservas que se van haciendo.

Como aún no se sabe si la solución adecuada a los problemas descritos es la creación de un software gestor de citas previas individual o de un *marketplace* que a su vez es gestor, es conveniente analizar las herramientas que existen actualmente en ambos campos.

2.1.1 Gestores de reservas

Para escoger los gestores de reservas de los que se van a hacer un análisis, se ha buscado cuáles son los que tienen más búsquedas en Google, que también coincide con los que tienen más usuarios activos en el momento de realización de este trabajo.

Setmore

Setmore [4] es uno de los gestores de reservas más populares que existen. Tiene una de las mejores interfaces del sector, facilitando mucho el acceso para gestionar las reservas que se han realizado de la forma más intuitiva posible. Como todo gestor de reservas, permite introducir horarios, trabajadores, servicios ofrecidos, entre otras más cosas. Además, no solo cuenta con versión web, sino que también hay versión disponible para Android y iOS, permitiendo que los administradores y los trabajadores del servicio puedan consultar las reservas que los clientes han hecho desde cualquier dispositivo móvil.

Podemos decir que el apartado de integraciones de Setmore es el más completo que hay, pudiendo conectarlo con casi cualquier software relacionado. Puede añadirse como módulo a cualquier página web para que desde esta los clientes puedan realizar reservas, siendo compatible tanto a nivel nativo como a través de *plugins* para Wordpress o Wix o incluso con otras tecnologías como Drupal. Por la parte de redes sociales, admite conexión con Facebook, haciendo mucho más sencillo el proceso de reserva, ya que no hay que salir de la red social en cuestión, admite conexión con Instagram, para transmitir fotos directamente a la página de reservas, y también admite conexión con Slack. Es compatible tanto con calendarios de Google como Office, y tiene muchas otras integraciones comerciales disponibles como con Google Analytics, MailChimp o Zendesk.

SimplyBook

SimplyBook [5] es otro de los motores de reservas más conocidos del sector. Además de su función principal, la de gestionar reservas, está centrado mucho en la integración y sobre todo en la personalización. Permite analizar tanto las reservas recibidas como los clientes que las han hecho, pudiendo conocer más profundamente a quien se está llegando y qué gente es la que está interesada en el negocio. Una vez hecho esto, permite la aplicación de ofertas, cupones, venta de productos y tarjetas regalo que se integran a la perfección con el motor de reservas.

Este software es muy adecuado para grandes servicios, ya que se puede centralizar toda su gestión desde un mismo panel, pudiendo mover trabajadores de un servicio a otro con facilidad, así como aplicar actualizaciones como promociones y ofertas en masa para todos los servicios. También admite sincronización con calendarios de Google para que tanto los trabajadores como los administradores de los servicios estén siempre actualizados con la última información.

Por último, SimplyBook le da mucho poder al usuario, permitiendo las cancelaciones según las condiciones del servicio, permitiendo las reservas en grupo incluso las reservas periódicas. Así mismo, una vez acabada la reserva podrán dejar una reseña en la web con su experiencia.

Reservio

Reservio [6] no es tan popular como los anteriores *softwares,* pero sí que es muy funcional. No solo ofrece una gestión de reservas que se puede integrar tanto con la web del servicio como con la mayoría de calendarios, sino que además también ofrece otras funciones como la administración de clientes, recordatorios y asistencia a tu negocio.

Cuenta con un plan gratuito en el que se permiten hacer hasta 40 reservas al mes, pero si se quieren más Reservio ofrece 3 planes distintos de pago por 8, 15 y 30 €. También dispone de una serie de tutoriales para formarse en esta plataforma, tanto en la implementación como el uso de ella, así como un programa de afiliados que otorga dinero por cada cliente que se traiga a Reservio.

Otros

Aparte de los tres servicios mencionados anteriormente, también se han analizado Timify y Bookitit. Sin embargo, no se ha querido hacer un análisis detallado, ya que no se han encontrado características relevantes que no se hayan mencionado con los anteriores.

2.1.2 *Marketplace* de reservas

En este apartado se han analizado dos *marketplace* de negocios para realizar reservas de cita previa. Se han escogido estos dos debido a que el primero es el único *marketplace* de negocios de España y el otro es el *marketplace* de negocios más popular que existe hasta la fecha.

miCita

MiCita [7] es una plataforma española disponible solo en formato aplicación que recopila distintos servicios y permite al usuario hacer reservas en cada uno de ellos. Esta aplicación es muy reciente y podemos observar que es un producto con sus funcionalidades básicas, pero aún no está completamente desarrollado.

La funcionalidad principal de la app es poder realizar reservas en los distintos establecimientos que hay dentro de ella. Actualmente no se encuentran demasiados y los pocos que hay parecen ser de prueba. Se puede ver poca información de cada establecimiento, algunas fotos, el horario y el código QR con el que podemos agregarlo a la lista de favoritos. Un inconveniente que le encontramos es que solo permite la búsqueda por nombre o población y que para hacer una reserva de un servicio hay que añadir primero el negocio a la lista de favoritos.

Otro factor interesante que tiene esta app es el sistema de puntos por negocio. Cada vez que se hace una reserva en un negocio se le otorgan al usuario una serie de puntos de ese negocio que luego puede canjear por productos del mismo. Sin embargo, esta funcionalidad aún no está operativa.

Booksy

Booksy [8] es un *marketplace* para pedir cita previa en negocios de belleza y salud. A diferencia del anterior, Booksy posee tanto versión web como versión app, y tiene disponibles una gran cantidad de negocios. Esta plataforma nació como una *startup* en Estados Unidos y se ha convertido en la primera app de reserva de cita previa en ese país. Este último año se ha empezado a expandir por Europa, empezando por Inglaterra, Francia y España.

Al ser ya una app con recorrido e instaurada en ciertos mercados, todas sus funcionalidades funcionan a la perfección. Por cada negocio disponible podemos ver información detallada, imágenes, reseñas, localización y horario, así como comprar sus productos. Además, esta aplicación permite que, una vez realizada la reserva del servicio en cuestión, se le pueda cobrar al usuario directamente desde la app si así lo prefiere, evitando intermediarios. Por otro lado, también se le puede aplicar una tasa de cancelación si no cumple las políticas de cancelación del negocio. Booksy también ofrece a los negocios la posibilidad de hacer ofertas y promociones en el perfil del negocio de la app. Esta funcionalidad es muy interesante y aporta mucho valor al usuario, sin embargo, tiene la limitación de que si el usuario no entra en el perfil del negocio no verá estas promociones.

Otra de las mejores funcionalidades que tiene es el filtro de búsqueda de negocios, pudiendo buscar por nombre, lugar, día o categoría. Hay que destacar que todos los negocios son o bien de belleza o bien de salud, y las peluquerías forman la mayor parte de estos negocios.

2.2 Conclusiones

A continuación, se presenta de forma resumida en una tabla, una comparación de las características más importantes a destacar entre las distintas plataformas analizadas.

Marketplaces	miCita	Booksy
Está disponible en formato app como en web	No (solo app)	Sí
Cantidad de negocios disponibles en España	Menos de 100	1350 aprox.
Supone un precio fijo para los negocios	Si	No
Posibilidad de pagar a través de la app	No	Sí
Posibilidad de hacer promociones disponibles para todos los usuarios	Sí	No
Posibilidad de hacer promociones a usuarios específicos	No	No
Posibilidad de añadir negocios mediante códigos QR	Sí	No
Tiene un sistema de puntos para obtener descuentos	Sí	No

Tabla 1: Comparación de funcionalidades entre marketplaces. Fuente: Elaboración propia.

Gestores de reservas	Setmore	SimplyBook	Reservio
Artículos y vídeos de soporte	Sí	No	No
Gestión de distintos establecimientos	No	Sí	No
Número de integraciones con otros servicios	+10	+5	+5
Contiene una versión app	Sí	No	No
Proporciona un análisis del negocio	No	Sí	Sí
Reservas grupales	Sí	Sí	No
Posibilidad de aceptar pagos en línea	Sí	Sí	No
Programa de afiliación	No	No	Sí
Posibilidad de personalización	Sí	Sí	Sí

Tabla 2: Comparación de funcionalidades entre gestores de reservas. Fuente: Elaboración propia.

Se puede ver que el estado actual es que existen muchos gestores de reservas muy completos, pero la mayoría demasiado complejos y de excesivo coste. Por otro lado, el mercado de *marketplaces* es más escaso y está dominado por uno en particular (Booksy). Sin embargo, este carece de algunas funcionalidades clave para llegar a los clientes además de no estar disponible para todo tipo de negocios pequeños. Por lo tanto, se concluye que lo que hace falta en el mercado es un gestor de reservas simple, con las funcionalidades básicas y esenciales al alcance de los pequeños negocios, y a su vez un *marketplace* donde estos pequeños negocios (de cualquier ámbito) puedan anunciarse y llegar con más facilidad a los clientes, que saldrán beneficiados al tener una plataforma desde donde poder manejar cualquier tipo de reservas. No hace falta reinventar la rueda, sino desarrollar una solución que encaje mejor con las necesidades y vacíos actuales junto con las funcionalidades clave que actualmente nadie está ofreciendo.

3. Alcance del proyecto

El sistema completo consistirá en una plataforma que funcione como *marketplace* y como gestor de cada negocio registrado en ella. Por un lado, el *marketplace* permitirá a los clientes encontrar negocios, pedir citas previas online en ellos y así poder centralizar en un mismo sitio todas sus citas. Por otro lado, los negocios contarán con un gestor que les ayudará no solo a digitalizar sus citas previas sino también a poder ofrecer citas online, así como otras funcionalidades como la obtención de estadísticas, gestión de trabajadores, etc. De esta manera el sistema aportará valor tanto a clientes como a negocios solucionando problemas a las dos partes implicadas.

Figura 2: Esquema del sistema completo. Fuente: Elaboración propia.

Esta plataforma es muy ambiciosa, compleja y abarca muchos aspectos y funcionalidades. Es por eso que se ha decidido dividir en dos sistemas distintos: el sistema *marketplace* (para los clientes) y el sistema gestor (para los negocios), tal como puede verse en la figura 2. El sistema *marketplace* se centrará en el cliente y en la centralización de todas sus citas, mientras que el subsistema gestor de negocio contendrá todas las funcionalidades para gestionar a este. Sin embargo, a pesar de que estos dos sistemas funcionen independientemente, lo que si que harán será la compartición de datos, ya que los dos trabajarán sobre los mismos.

Teniendo en cuenta que la dedicación para el TFG es de aproximadamente 540 horas (30 horas por crédito), este trabajo se centrará en el sistema *marketplace*, abordando su análisis, diseño e implementación de un prototipo, dejando para más adelante la elaboración del sistema gestor de negocio. De aquí en adelante, todo lo que se trate (requisitos, arquitectura, diseño, etc.) hará referencia a este sistema.

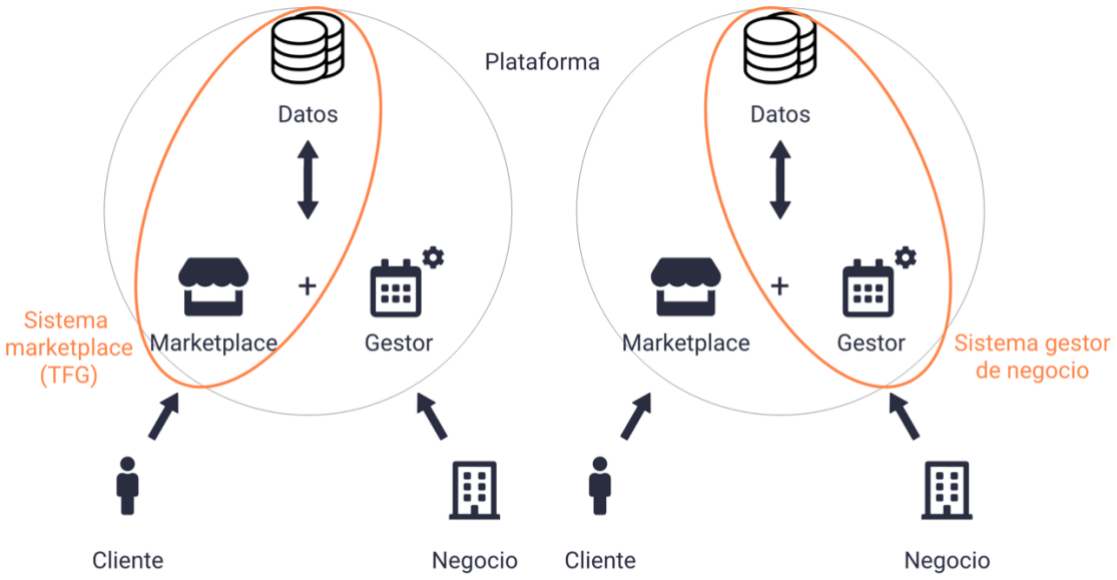


Figura 3: Esquema de los dos subsistemas. Fuente: Elaboración propia.

3.1 Sistema *marketplace*

Una de las decisiones que se han tomado desde el inicio de este trabajo es que la forma con la que el cliente va a interactuar con el sistema *marketplace* va a ser a través de una app para dispositivos móviles. Esta decisión ha sido tomada ya que al ser un sistema orientado al cliente una app móvil ofrece una mayor portabilidad y flexibilidad para este. Otras alternativas, como podrían ser una web o una aplicación de escritorio, limitan mucho al cliente, ya que han de tener a mano dispositivos más grandes y ese no suele ser el caso en este contexto. Una app móvil permite al cliente poder utilizar el sistema siempre que tenga un móvil, que es la mayor parte del tiempo, y así tener la posibilidad de pedir citas previas en cualquier momento de necesidad.

Otro motivo por el cual se ha tomado esta decisión es debido a las preferencias del autor, ya que quiere especializarse en este tipo de tecnologías y quiere aprovechar esta oportunidad para ganar conocimiento tanto en el desarrollo de apps para móviles como en las distintas tecnologías que se utilizan en este ámbito.

3.2 Objetivos y sub-objetivos

El objetivo principal de este trabajo es el de abordar el análisis, diseño y desarrollo de un prototipo de app móvil para el sistema *marketplace*. Para cumplir con este objetivo se han de cumplir distintos sub-objetivos descritos a continuación:

- **Proporcionar un servicio de reservas siempre disponible:** Esta herramienta ha de permitir la realización de reservas online por parte de los clientes en cualquier momento del día según la disponibilidad del negocio.

- **Buscar negocios:** El cliente ha de poder encontrar todos los negocios que tenga el sistema, pudiendo filtrarlos según distintos parámetros.

- **Centralizar reservas del cliente:** Ya sean reservas hechas en negocios como reservas apuntadas manualmente, el cliente ha de ser capaz de tenerlas todas juntas y poder gestionarlas.

- **Mejorar la comunicación con el negocio:** Creando no solo un canal para poder hacer reservas online sino también una forma de comunicación más personalizada, ya sea dando la posibilidad al cliente de recibir notificaciones, ofertas, promociones o guardarse sus negocios favoritos.

Además de estos objetivos funcionales, con la realización de este proyecto el autor también quiere poder alcanzar los siguientes objetivos técnicos:

- **Familiarizarse con el desarrollo multiplataforma:** A pesar de que el autor ya tiene conocimientos sobre el desarrollo nativo, quiere iniciarse en el desarrollo de aplicaciones multiplataforma.

- **Familiarizarse con servicios externos:** Aprender a usar servicios externos en combinación de las apps móviles ya que ofrecen muchas ventajas sobretodo en términos de escalabilidad, además de ser muy usados en esta industria.

Finalmente, los objetivos no funcionales están listados a continuación:

- **Importancia de la apariencia:** Intentar desarrollar una app con una gran apariencia, que atraiga y guste visualmente a los usuarios.

- **Priorización de la escalabilidad y la integración:** Al ser un proyecto grande, con visión de futuro y con intención de convertirlo en un negocio, la escalabilidad es un factor muy importante. Además, al tener que juntarse con otro sistema en un futuro, siempre hay que tener en cuenta la forma en que se integraran estos dos sistemas.

3.3 Riesgos

Durante el periodo de desarrollo de este proyecto, hay que ser consciente de que pueden darse situaciones que alteren la planificación hecha al comienzo. Para minimizar el impacto, es primordial conocer de antemano los riesgos existentes que pueden hacer que el desarrollo del proyecto se vea afectado. Estos riesgos son:

- **Inexperiencia en las tecnologías necesarias:** A la hora de comenzar el proyecto, el autor no tiene conocimientos en algunas de las tecnologías que va a necesitar para desarrollar ciertos apartados de la plataforma. Por lo tanto, va a ser necesario un periodo previo de aprendizaje. En caso de que sea mayor de lo esperado, esto afectará de manera sustancial a la planificación al tener que dedicar más horas de las previstas en el estudio de estas tecnologías, y por lo tanto habrá que reducir horas en otros apartados del proyecto.

- **Bugs:** Independientemente del uso o no de tecnologías de las que no se posee un avanzado conocimiento, a la hora de desarrollar software siempre existe la posibilidad de aparición de errores. En función de cuando se detecten estos errores serán más o menos fáciles de solucionar, ya que si son encontrados en el momento de desarrollo tendrán una solución más sencilla que si estos son detectados al final del proyecto, teniendo que hacer más modificaciones. Es por eso que hay que tener muy en cuenta que la detección es clave en este riesgo, y esto ha de estar reflejado en la planificación.

- **Calendario cerrado:** Como ya se ha mencionado, este proyecto es un trabajo de final de grado, y por lo tanto ha de seguir las pautas y tiempos que la universidad dictamina. Es por eso que la fecha de entrega ya está fijada y a pesar de que la planificación cuente con ello, si ocurre algún imprevisto esta fecha no va a poder ser modificada, teniendo que recortar en funcionalidades a modo de poder entregar el trabajo a tiempo. Este riesgo hace que la planificación cobre una gran importancia, y que tenga que ser revisada usualmente para detectar y corregir cuanto antes las desviaciones en ella.

4. Metodología

La elección de metodología es una de las decisiones más condicionantes de los proyectos, ya que marcará la forma de trabajar siendo determinante a la hora de alcanzar los objetivos previamente descritos. Esta no es una decisión sencilla, hay que tener muy claro los requisitos del proyecto, así como las ventajas e inconvenientes de cada metodología para poder ver cuál es la que se adecúa más al proyecto. A pesar de que existen una gran cantidad de metodologías, las dos más utilizadas son *Waterfall* y *Agile* [9].

Waterfall o también denominada "en cascada", es la metodología utilizada tradicionalmente en el desarrollo de proyectos. Consiste en la ejecución secuencial de las fases de análisis, diseño, testeo y producción. *Agile* en cambio es una filosofía con un enfoque más iterativo, aportando más flexibilidad al proceso de desarrollo para adaptarse a un entorno que puede tener más cambios.

En este caso se ha escogido utilizar una metodología *Agile* ya que, al no tener un cliente específico, este proyecto está sujeto a las necesidades del mercado, pudiendo cambiar estas en cualquier momento. Este tipo de metodología es más adecuada para este tipo de situaciones, pudiendo adaptarse a los cambios que vayan surgiendo e ir revisando el estado del producto gracias al desarrollo continuo de este.
Dentro de *Agile* hay distintas metodologías, siendo *Scrum* y *Kanban* [10] las principales. A pesar de que las dos siguen la filosofía *Agile*, cada una la adapta a su manera. En *Scrum* un proyecto se ejecuta en ciclos temporales cortos de duración fija (iteraciones que duran aproximadamente 2 semanas) y los distintos roles del equipo desempeñan una función muy importante para el correcto funcionamiento de esta metodología. Por su parte, *Kanban* se basa en el desarrollo y entrega continuos, abordando un pequeño número de tareas de forma fluida y simultánea. Para ello, se utilizan herramientas de planificación visual para saber en qué estado se encuentran las historias de usuario. Debido a que este proyecto solo será desarrollado por una persona, se ha optado por la utilización de *Kanban* en vez de *Scrum*, ya que a pesar de ser menos popular no le da tanta importancia a los distintos roles del equipo.

Este proyecto no va a seguir estrictamente *Kanban*, sino que, partiendo de la idea principal, esta metodología va a adaptarse a las necesidades del trabajo que se ha de realizar. Con la herramienta visual que será explicada posteriormente, se va a poder generar un tablero *kanban* que va a ser actualizado cada vez que haya un cambio de estado en una historia de usuario. Cada vez que se termine una historia, se va a comprobar la planificación inicial para ver si se están cumpliendo los plazos y poder tomar medidas en caso de haber desviaciones. Debido a su gran utilidad, se va a incorporar el concepto de *Sprint Review* de la metodología *Scrum* en este proyecto, en este caso al no tener *Sprints* se hará cada vez que se terminen todas las tareas de un mismo apartado (más adelante en este documento se especificarán las tareas y a que apartado corresponden). Estos *Reviews* no se utilizarán como en *Scrum*, sino que servirán para comprobar, mediante tests de integración, que todo el bloque de tareas finalizado funciona correctamente. Además, también se aprovechará para acabar de redactar la documentación relacionada con ese bloque de tareas.

4.1 Taiga

Se usará la herramienta Taiga [11] para llevar un control de las historias de usuario y poderlas visualizar en un tablero de forma que en todo momento podamos conocer su estado. Para ello, el tablero se dividirá en 4 columnas (*To-Do*, *In Progress, Testing* y *Done*) y las historias de usuario se irán moviendo por estas columnas a medida que vayan habiendo cambios. Cada una de estas historias tendrá su propia descripción, así como una etiqueta que servirá para identificar a qué ámbito pertenece.

4.2 Git

Para la gestión de versiones del código generado usaremos Git [12], un sistema de control de versiones muy utilizado en el mundo del software. Este sistema permite llevar un control exhaustivo de todas las modificaciones que se van realizando en el código, permitiendo así a los desarrolladores tener un control de los cambios y poder navegar entre ellos. Para llevar este control, Git permite organizar el código en repositorios y a su vez estos permiten la utilización de ramas, que simbolizan distintas versiones del mismo código. Estos repositorios de código pueden ser gestionados por servicios como GitHub [13], que permiten subir los repositorios a la nube para que estén disponibles en cualquier momento, así como otras muchas funcionalidades.

Antes de empezar a trabajar con Git, es conveniente definir qué flujo de trabajo seguiremos. En este caso, vamos a trabajar con dos ramas principales, la *master* y la *development*. La rama master contendrá el código ya testeado y listo para producción y la rama *development* será la rama donde se irán añadiendo las funcionalidades que vayan siendo finalizadas. Para cada historia de usuario se abrirá una nueva rama desde *development* y se desarrollará ahí la historia de usuario. Una vez finalizada la historia y testeado el código, se añadirá la rama a *development*. Esto se hará para cada historia a implementar, y cuando ya se lleven unas cuantas historias realizadas se generará una nueva rama que partirá de *development* donde se comprobará que no existe ningún error y finalmente se juntará esta rama con la rama *master*. Esto se hará porque así evitaremos que la rama *master* esté demasiado desactualizada. Este flujo de trabajo que seguiremos es una adaptación de GitFlow [14] para nuestro proyecto en específico.

5. Descripción de las tareas

Las tareas de este proyecto están divididas en 3 apartados: gestión del proyecto, desarrollo y documentación. Dentro del apartado de desarrollo están las tareas relacionadas con la parte técnica del proyecto, que han sido agrupadas en distintos bloques según a qué gran funcionalidad pertenecen. Todas las tareas contienen una breve explicación para entender su significado, así como una estimación de las horas que va a tomar su realización y las tareas que han de ser finalizadas antes de poder empezar la tarea actual. La fecha de inicio de este proyecto fue el 1 de septiembre de 2020, teniendo que finalizar antes de finales de enero de 2021, fecha en la cual se tendrá que presentar.

5.1 Gestión del proyecto

- **GP1 - Contextualización y alcance:** Se elaborará un documento contextualizando el proyecto, analizando la competencia y justificando la elección de la solución, definiendo el alcance y explicando las metodologías que se van a seguir. Esta tarea durará 20 horas.
- **GP2 - Planificación:** Se elaborará un documento describiendo las tareas que se van a realizar, se harán estimaciones sobre la duración de estas tareas y se expondrán en un diagrama de *Gantt*. Además, se hará una gestión de los riesgos. Esta tarea durará 15 horas. Dependencias: GP1.
- **GP3 - Gestión económica y de sostenibilidad:** Se elaborará un documento con la estimación del presupuesto de costes y de gestión, así como se realizará un informe de sostenibilidad. Esta tarea durará 15 horas. Dependencias: GP2.
- **GP4 - Integración final del documento:** Se elaborará un documento que integrará los contenidos de todos los documentos anteriormente realizados. Esta tarea durará 20 horas. Dependencias: GP3.

5.2 Desarrollo

5.2.1 *Inception*

- **I1 - Especificación de los requisitos:** Se identificarán y especificarán los requisitos del sistema, así como sus funcionalidades. Para ello, entre varias cosas que se harán, se examinará la competencia en busca de carencias y se hablará con dueños de pequeños negocios para entender mejor sus necesidades. Esta tarea durará 20 horas.
- **I2 - Preparación del entorno:** Se descargarán todos los programas necesarios para la gestión, diseño y programación. Algunos de estos ejemplos son el sistema de control de versiones, los *IDEs* necesarios, etc. Esta tarea durará 5 horas y es necesario que I1 esté acabada.
- **I3 - Diseño de las interfaces de la app:** Antes de empezar a programar, se hará un diseño gráfico de la apariencia de la aplicación que servirá como guía para posteriormente programar las interfaces. Esto es lo que se suele hacer en metodologías ágiles para aumentar la productividad. Esta tarea durará 20 horas y es necesario que I2 esté acabada.

- **I4 - Diseño de las bases de datos:** Siguiendo las pautas de las metodologías *Agile*, también se hará un diseño de la estructura de las bases de datos que tendrán que ser creadas para tener una idea clara de la arquitectura general, así como lo que se ha de programar en términos de *Backend*. Esta tarea durará 5 horas y es necesario que I2 esté acabada.

Las siguientes tareas, al ser de programación y siguiendo las pautas de las metodologías *Agile*, van a constar cada una de una fase de programación y otra fase de *testing*. Se hará de esta forma y no un testeo general de todo al final para poder evitar que se arrastren errores de las tareas hasta el final del proyecto. Las horas de duración totales son las compuestas por las horas de programación y las horas de *testing*. Como ya se comentó en el apartado de metodologías, también se harán test de integración cada vez que se finalice un bloque de tareas.

5.2.2 Gestión de Usuario

- **US1 - Creación de cuenta de usuario:** Se creará la base de datos de usuarios, una interfaz en la app para poder crear una cuenta y la conexión entre ambas. Además, se programará un sistema de verificación por email. Esta tarea durará 15 horas y es necesario que todas las tareas de *Inception* estén acabadas.
- **US2 - *Login*:** Se creará una interfaz para poder autenticarse en la app que estará vinculada con la base de datos de usuarios. Esta tarea durará 20 horas y es necesario que US1 esté acabada.
- **US3 - Contraseña olvidada:** Se programará el sistema de recuperación de contraseña en caso de que se haya olvidado. Esta tarea durará 5 horas y es necesario que US2 esté acabada.
- **US4 - Edición de información:** Se creará una interfaz para poder editar toda la información del usuario, así como las preferencias de la app. Esta interfaz se vinculará con la base de datos para guardar los cambios. Esta tarea durará 20 horas y es necesario que US3 esté acabada.

5.2.3 Gestión de citas previas en negocios

- **NEG1 - Información de los negocios:** Se creará la base de datos de los negocios y una interfaz en la aplicación para poder ver la información relacionada con ellos. Esta tarea durará 30 horas y es necesario que todas las tareas de *Inception* estén acabadas.
- **NEG2 - Reserva de cita previa:** Se programará un sistema para poder reservar citas en los negocios. Este sistema constará de una interfaz, así como una lógica y conexión con la base de datos de reservas que se tendrá que crear. Esta tarea durará 80 horas y es necesario que US1 y NEG1 estén acabadas.
- **NEG3 - Visualización y cancelación cita previa:** Se creará una interfaz que estará vinculada con la base de datos para poder visualizar las reservas que se han hecho, así como la función de cancelación de estas. Esta tarea durará 20 horas y es necesario que NEG2 esté acabada.

5.2.4 Gestión de citas personales

- **PER1 - Creación y visualización de citas personales:** Se programará un sistema para la creación de citas que constará de una interfaz y de su base de datos que también tendrá que ser creada. Además, para su visualización, se aprovechará la interfaz de visualización de citas de los negocios y solo se tendrá que crear la conexión con la base de datos. Esta tarea durará 30 horas y es necesario que NEG3 esté acabada.
- **PER2 - Edición y eliminación citas personales:** Se habilitará la edición de la información de este tipo de citas a través de una interfaz y su conexión con la base de datos. Además, la cita también podrá ser eliminada a través de esta interfaz. Esta tarea durará 20 horas y es necesario que PER1 esté acabada.

5.2.5 Búsqueda de negocios

- **BUS1 - Búsqueda por nombre:** Se programará la búsqueda de negocios por nombre. Esta búsqueda constará de una interfaz y su correspondiente conexión con la base de datos. Esta tarea durará 30 horas y es necesario que NEG1 y todas las tareas de US estén acabadas.
- **BUS2 - Búsqueda por categoría:** Para la búsqueda por categoría se aprovechará la interfaz anterior y se programará la conexión con la base de datos para poder obtener negocios en función de su categoría. Esta tarea durará 10 horas y es necesario que BUS1 esté acabada.
- **BUS3 - Búsqueda por localidad:** Para la búsqueda por localidad se aprovechará la interfaz anterior y se programará la conexión con la base de datos para poder obtener negocios en función de su localidad. Esta tarea durará 10 horas y es necesario que BUS2 esté acabada.
- **BUS4 - Mapa:** Se programará una interfaz en forma de mapa para mostrar los resultados de las búsquedas anteriores. Esta tarea durará 50 horas y es necesario que BUS3 esté acabada.

5.2.6 Gestión de negocios favoritos

- **FAV1 - Creación y eliminación de negocios favoritos:** Se programará la opción de creación y eliminación de negocios favoritos para los usuarios. Esta tarea constará de la creación de la base de datos de negocios favoritos, así como un botón en la interfaz de los negocios y su conexión con la base de datos. Esta tarea durará 5 horas y es necesario que NEG1 y todas las tareas de US estén acabadas.
- **FAV2 - Visualización de negocios favoritos:** Se creará una interfaz para visualizar los negocios favoritos, así como su correspondiente conexión con la base de datos. Esta tarea durará 5 horas y es necesario que FAV1 esté acabada.

5.2.7 Notificaciones y promociones

- **NP1 - Recepción y visualización de notificaciones:** Se programará la recepción de notificaciones, así como una interfaz de visualización y su correspondiente base de datos. Esta tarea durará 20 horas y es necesario que todas las tareas de US, NEG, y PER estén acabadas.
- **NP2 - Recepción y visualización de promociones:** Se programará la recepción de promociones, así como una interfaz de visualización. Esta tarea durará 20 horas y es necesario que NP1 esté acabada.

5.3 Documentación y comunicación

- **DC 1 - Seguimiento:** Se irán apuntando todos los eventos que se vayan realizando con el fin de llevar un seguimiento y posteriormente incluirlos en la documentación. Esta tarea se realizará continuamente y es por ello que no tiene una duración específica.
- **DC2 - Documentación:** Se dedicará tiempo a la elaboración de toda la documentación relativa al desarrollo, así como a la mejora de la documentación previamente hecha. Esta tarea durará 80 horas.
- **DC2 - Comunicación:** Se realizarán reuniones con la tutora del proyecto para analizar el trabajo realizado, comentar la progresión, las dudas y los problemas que surjan. Esta tarea durará 20 horas.

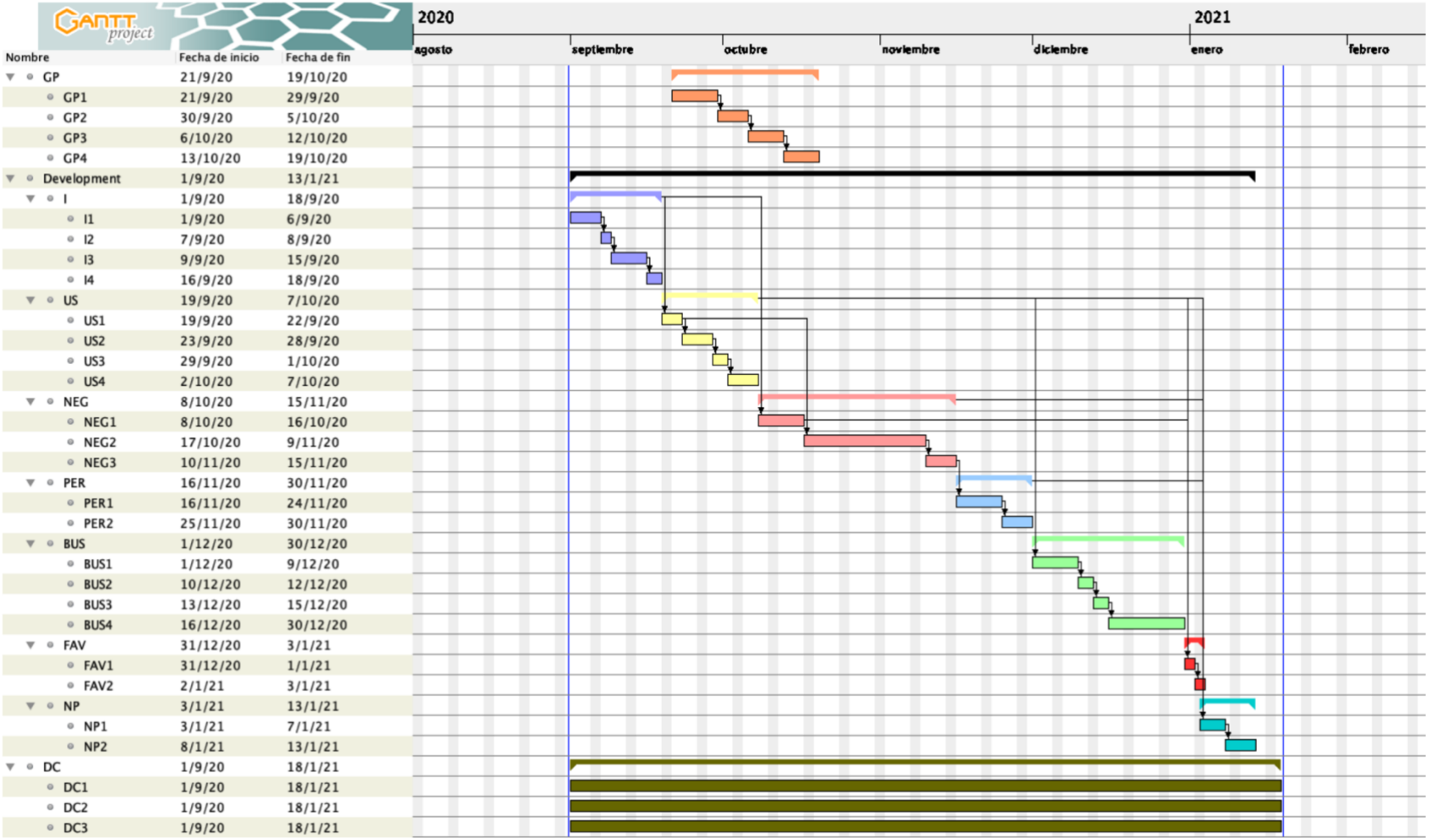
6. Estimaciones y Gantt

6.1 Estimaciones

Código	Tarea	Duración (h)	Dependencias	Recursos
GP	GESTIÓN DEL PROYECTO	Total: 70		
GP1	Contextualización y alcance	20		PC, Drive, Word
GP2	Planificación	15	GP1	PC, Drive, Word
GP3	Gestión económica y sostenibilidad	15	GP2	PC, Drive, Word, Excel
GP4	Integración final del documento	20	GP3	PC, Drive, Word
	DESARROLLO	Total: 440		
I	*Inception*	Total: 50		
I1	Especificación de requisitos	20		PC, Word
I2	Preparación del entorno	5	I1	PC, Android Studio, Xcode, Git
I3	Diseño de las interfaces de la app	20	I2	PC, Adobe XD, Drive
I4	Diseño de las bases de datos	5	I2	PC, Word
US	Gestión de Usuario	Total: 60		
US1	Creación de cuenta de usuario	15	I	PC, Android Studio, Xcode, Git
US2	*Login*	20	US1	PC, Android Studio, Xcode, Git
US3	Contraseña olvidada	5	US2	PC, Android Studio, Xcode, Git
US4	Edición de información	20	US3	PC, Android Studio, Xcode, Git
NEG	Gestión de citas previas en negocios	Total:130		
NEG1	Información de los negocios	30	I	PC, Android Studio, Xcode, Git
NEG2	Reserva de cita previa	80	NEG1, US1	PC, Android Studio, Xcode, Git
NEG3	Visualización y cancelación de cita previa	20	NEG2	PC, Android Studio, Xcode, Git
PER	Gestión de citas personales	Total: 50		
PER1	Creación y visualización de citas personales	30	NEG3	PC, Android Studio, Xcode, Git
PER2	Edición y eliminación de citas personales	20	PER1	PC, Android Studio, Xcode, Git
BUS	Búsqueda de negocios	Total: 100		
BUS1	Búsqueda por nombre	30	NEG1, US	PC, Android Studio, Xcode, Git
BUS2	Búsqueda por categoría	10	BUS1	PC, Android Studio, Xcode, Git
BUS3	Búsqueda por localidad	10	BUS2	PC, Android Studio, Xcode, Git
BUS4	Mapa	50	BUS3	PC, Android Studio, Xcode, Git
FAV	Gestión de negocios favoritos	Total: 10		
FAV1	Creación y eliminación de negocios favoritos	5	NEG1, US	PC, Android Studio, Xcode, Git
FAV2	Visualización de negocios favoritos	5	FAV1	PC, Android Studio, Xcode, Git
NP	Notificaciones y promociones	Total: 40		
NP1	Recepción y visualización de notificaciones	20	US, NEG, PER	PC, Android Studio, Xcode, Git
NP2	Recepción y visualización de promociones	20	NP1	PC, Android Studio, Xcode, Git
	DOCUMENTACIÓN Y COMUNICACIÓN	Total: 100		
DC1	Seguimiento	-		PC, Taiga, Word
DC2	Documentación	80		PC, Taiga, Drive, Word
DC3	Comunicación	20		PC, Meet, Gmail
	Total	610		

Tabla 3: Estimación de horas, recursos y dependencias entre tareas. Fuente: Elaboración propia.

6.2 Gantt



Nombre	Fecha de inicio	Fecha de fin
GP	21/9/20	19/10/20
GP1	21/9/20	29/9/20
GP2	30/9/20	5/10/20
GP3	6/10/20	12/10/20
GP4	13/10/20	19/10/20
Development	1/9/20	13/1/21
I	1/9/20	18/9/20
I1	1/9/20	6/9/20
I2	7/9/20	8/9/20
I3	9/9/20	15/9/20
I4	16/9/20	18/9/20
US	19/9/20	7/10/20
US1	19/9/20	22/9/20
US2	23/9/20	28/9/20
US3	29/9/20	1/10/20
US4	2/10/20	7/10/20
NEG	8/10/20	15/11/20
NEG1	8/10/20	16/10/20
NEG2	17/10/20	9/11/20
NEG3	10/11/20	15/11/20
PER	16/11/20	30/11/20
PER1	16/11/20	24/11/20
PER2	25/11/20	30/11/20
BUS	1/12/20	30/12/20
BUS1	1/12/20	9/12/20
BUS2	10/12/20	12/12/20
BUS3	13/12/20	15/12/20
BUS4	16/12/20	30/12/20
FAV	31/12/20	3/1/21
FAV1	31/12/20	1/1/21
FAV2	2/1/21	3/1/21
NP	3/1/21	13/1/21
NP1	3/1/21	7/1/21
NP2	8/1/21	13/1/21
DC	1/9/20	18/1/21
DC1	1/9/20	18/1/21
DC2	1/9/20	18/1/21
DC3	1/9/20	18/1/21

Figura 4: Diagrama de Gantt. Fuente: Elaboración propia.

7. Gestión de riesgos

Como ya se ha mencionado en apartados anteriores, pueden darse situaciones que conlleven una modificación de la planificación realizada. Para cada uno de estos riesgos se ha calculado la probabilidad de que ocurra, así como las horas que se perderían aproximadamente para darle solución.

Riesgos	Probabilidad	Horas aprox.
Inexperiencia en las tecnologías utilizadas	Media	50
Bugs	Media	30
Calendario cerrado	Baja	10

Tabla 4: Por cada riesgo, probabilidad de aparición y estimación de horas de desviación en caso de aparición. Fuente: Elaboración propia.

Con el fin de minimizar el impacto, seguidamente se expondrán las acciones a realizar y los planes alternativos que se han considerado en caso de que alguna o varias situaciones expuestas anteriormente ocurran.

7.1 Inexperiencia en las tecnologías utilizadas

A la hora de empezar el proyecto, a pesar de que se tengan algunos conocimientos básicos de desarrollo de aplicaciones nativas en Android, se tienen muy pocos conocimientos de la realización de aplicaciones multiplataforma. Además, tampoco se tienen amplios conocimientos de tecnologías de *Backend*. Esta situación puede influir mucho en la planificación si la curva de aprendizaje no es la esperada, pudiendo determinar de manera incorrecta las horas que se necesitarán para realizar las tareas de desarrollo cuando se hace la planificación. El plan alternativo que se ha utilizado para minimizar este riesgo es informarse sobre la dificultad de las tecnologías, específicamente en su implementación en aquellos procesos relacionados con las tareas especificadas anteriormente, y la sobreestimación de horas de las tareas que involucran tecnologías de las cuales se posee poco conocimiento para así contar con un margen de error. En caso de extensión, no se necesitarán recursos adicionales.

7.2 *Bugs*

Un problema que suele aparecer en la programación y más cuando se está aprendiendo una tecnología nueva son los *bugs*. Este riesgo se minimiza con la introducción de tests en cada una de las tareas para asegurarse de que no existe ningún *bug* o de que si existe y es un *bug* importante poderlo solucionar antes de seguir con el desarrollo y hacer más complicada su solución. La etapa de *debugging* y *testing* en cada tarea es muy importante precisamente para evitar estas situaciones. Igualmente, en caso de que aparezca un *bug* en tareas ya finalizadas se adaptará la planificación creando una nueva tarea exclusivamente para solucionar este error.

Las horas de las tareas que se crearán serán contabilizadas como horas de solución de *bugs* mostradas en la tabla de arriba. En caso de extensión, no se necesitarán recursos adicionales.

7.3 Calendario cerrado

Este riesgo tiene una probabilidad muy baja debido a que es una situación que se conoce desde el inicio del proyecto. Toda la planificación se ha hecho teniendo en cuenta este factor y se ha adaptado el alcance al tiempo disponible para la realización del trabajo. Igualmente, puede haber desviaciones inesperadas y por ello se dedican 10 horas a la solución de estas. En caso de extensión, no se necesitarán recursos adicionales.

8. Presupuesto

8.1 Identificación y estimación de los costes

Después de haber hecho una planificación exhaustiva y siendo conscientes de todas las tareas que hay que realizar para completar el proyecto, en este apartado se van a especificar todos los costes que conlleva este trabajo. Para ello, se van a presentar dos casos distintos de costes en función del número de personas que trabajan en el proyecto. En el primer caso, el caso real, la única persona que trabajará en este proyecto será el autor de este. En el caso número dos, se calcularán todos los costes suponiendo que este trabajo es realizado por un equipo de personas con distintos roles. El equipo del caso 2 lo formarán 6 personas, una para cada rol expuesto más adelante.

Para poder calcular los costes de los recursos humanos primero hay que especificar cuál es el salario por hora de cada uno de los roles. En el primer caso, el salario que se le asignará al autor de este proyecto es de 9€ netos la hora, siendo este el salario mínimo que exige la UPC a las empresas que contratan a sus estudiantes para realizar prácticas. En el caso dos, el salario por hora de los distintos roles se determinará usando la web Indeed [15], una de las plataformas más utilizadas en España para buscar trabajo. El resultado de esta búsqueda está expuesto en la Tabla [5].

Rol	Sueldo (€ por hora)	Sueldo + SS(€ por hora)
Jefe de proyecto	21,24	27,61
Analista	15,65	20,35
Arquitecto de Software	21,67	28,17
Diseñador UX/UI	14,36	18,67
Programador	15,48	20,12
Tester	14,45	18,79

Tabla 5: Sueldos por hora de los distintos roles en el equipo. Fuente: Elaboración propia con los datos de Indeed.

Para hacer de manera correcta los cálculos, se tomará como salario por hora el sueldo de cada rol más el importe que le corresponde pagar a la empresa de seguridad social. Este salario se calcula multiplicando el sueldo de la persona por 1,3.

8.1.1 Costes de personal por actividad

En el caso 2, para calcular los costes de personal, se va a especificar las horas que cada rol del equipo va a dedicar a cada una de las tareas, de tal forma que quede visible cuánto se va a gastar en cada persona, así como en el total de estas. En cambio, en el caso 1, al ser la misma persona quien realizará todo el trabajo simplemente hay que multiplicar el sueldo por hora por el conjunto de horas que serán llevadas a cabo.

Código	JP	AN	AS	DIS	PR	TS	Duración (h)	Coste (€)
GP							Total: 70	
GP1	20						20	552,24
GP2	15						15	414,18
GP3	15						15	414,18
GP4	20						20	552,24
Desarrollo							Total: 440	
I							Total: 50	
I1		15	5				20	446,03
I2					4,5	0,5	5	99,950
I3				20			20	373,36
I4			5				5	140,85
US							Total: 60	
US1		1,5	1,5		10,5	1,5	15	312,25
US2		2	2		14	2	20	416,33
US3		0,5	0,5		3,5	0,5	5	104,08
US4		2	2		14	2	20	416,33
NEG							Total:130	
NEG1		3	3		21	3	30	624,50
NEG2		8	8		56	8	80	1665,3
NEG3		2	2		14	2	20	416,33
PER							Total: 50	
PER1		3	3		21	3	30	624,50
PER2		2	2		14	2	20	419,01
BUS							Total: 100	
BUS1		3	3		21	3	30	624,50
BUS2		1	1		7	1	10	208,16
BUS3		1	1		7	1	10	208,16
BUS4		5	5		35	5	50	1040,8
FAV							Total: 10	
FAV1		0,5	0,5		3,5	0,5	5	104,08
FAV2		0,5	0,5		3,5	0,5	5	104,08
NP							Total: 40	
NP1		2	2		14	2	20	416,33
NP2		2	2		14	2	20	416,33
Documentación							Total: 100	
DC1	-	-	-		-	-	-	
DC2	50	10	10		10		80	2067
DC3	5	3	3	3	3	3	20	456,33
Total Horas	125	67	62	23	290,5	42,5	610	
Total Euros C2	3452	1363,1	1747	429	5846	798		13635
Total Euros C1								5490

Tabla 6: Horas dedicadas por cada persona a cada tarea, total de dinero necesario por persona y total de dinero del proyecto en los casos uno y dos. Fuente: Elaboración propia.

8.1.2 Costes generales

Los costes generales son aquellos que se calculan globalmente, sin calcular su imputación a nivel de actividad. En este caso, de recursos materiales solo hará falta un ordenador, ya que todo el software utilizado es gratuito. El ordenador escogido es el usado por el autor del proyecto y su amortización se ha calculado usando la siguiente fórmula:

$$\frac{Coste\ (euros)}{Vida\ útil\ (años) \times Días\ laborables\ al\ año \times Dedicación\ diaria\ (horas)} \times Duración\ del\ proyecto\ (horas)$$

Donde la dedicación diaria es de 5 horas, los días laborables al año son 220 y la duración del proyecto es de 610 horas.

Recurso	Precio (€)	Vida Útil (años)	Amortización (€)
Macbook Pro 13"	2500	4	346,6

Tabla 7: Recurso material y su amortización. Fuente: Elaboración propia.

Por otro lado, también es necesario un área de trabajo, así como su correspondiente electricidad. Para facilitar el cálculo de estos servicios, se ha decidido buscar el precio que tiene un espacio de *coworking* en una zona cercana a donde vive el autor del trabajo. Se han consultado los tres portales más famosos para alquilar este tipo de espacios (Cwork [16], Meetbcn [17] y OneCoWork [18]) para determinar su precio.

Recurso	Precio por mes (€)	Meses	Total (€)
Workspace	295	4	1185

Tabla 8: Recurso de espacio y electricidad. Fuente: Elaboración propia con datos externos.

En el primer caso, al solo estar trabajando una persona en el proyecto solo se necesita un ordenador y un espacio en el área de *coworking*. Por otro lado, para el equipo de 6 personas se necesitarán 6 ordenadores y un espacio de trabajo para cada una de ellas.

Casos	Nº Ordenadores	Nº Personas *Workspace*	Total CG
C1	1	1	1532
C2	6	6	9190

Tabla 9: Número de ordenadores y de espacios de coworking con sus respectivos precios. Fuente: Elaboración propia.

8.1.3 Coste de contingencia

El coste de contingencia es dinero que se añade al presupuesto para aquellos imprevistos que no se han anticipado. Este coste se calcula como un porcentaje del valor total del presupuesto. El porcentaje suele venir determinado por el sector y el nivel de detalle del presupuesto. En proyectos informáticos este porcentaje suele estar entre el 10 y el 20%, siendo 15% el margen escogido para este proyecto.

Casos	Total CPA (€)	Total CG (€)	Contingencia (%)	Contingencia (€)
C1	5490	1532	15	1053
C2	13635	9190	15	3424

Tabla 10: Costes de contingencia para el caso uno y dos. Fuente: Elaboración propia.

8.1.4 Coste de imprevistos

En el apartado de gestión de riesgos se han especificado los riesgos que se han detectado, así como las acciones a tomar para solucionarlos en caso de que aparezcan. Estas acciones tendrán un impacto económico si ocurren, es por eso que se ha de calcular cuánto dinero hay que añadir al presupuesto destinado a costear las acciones a realizar en caso de riesgo. Como se puede observar en la Tabla 7, en el caso 2 no se ha incluido el riesgo 1, ya que este riesgo está vinculado con los conocimientos del autor del trabajo y en caso de tener que contratar a gente ya se buscaría gente con esos conocimientos.

Casos	Riesgo	Probabilidad (%)	Tiempo (h)	Coste (€)
C1	R1	66	50h Personales	297
	R2	66	30h Personales	178,2
	R3	33	20h Personales	59,4
	Total			535
C2	R2	66	30h Programador	398,46
	R3	33	20h Programador	132,82
	Total			531

Tabla 11: Coste por riesgo en los casos uno y dos. Fuente: Elaboración propia.

En caso de que ocurra solo uno de estos imprevistos, el dinero destinado a los distintos riesgos irá todo al imprevisto en cuestión. En el mejor de los casos, si no ocurre ningún imprevisto, todo el dinero se destinará a más horas de programación y *testing* con el fin de mejorar la experiencia de usuario en las distintas funcionalidades de la aplicación.

8.1.5 Presupuesto final

Finalmente, el presupuesto final viene determinado por la suma de todos los costes anteriores. El precio final está redondeado al alza.

Casos	Total CPA (€)	TOTAL CG (€)	Contingencia (€)	Imprevistos (€)	Final (€)
C1	5490	1532	1053	535	8610
C2	13635	9190	3424	531	26780

Tabla 12: Presupuesto final de los casos 1 y 2. Fuente: Elaboración propia.

8.2 Control de gestión

La planificación y la estimación de los costes es igual de importante que llevar un buen control del dinero que se está gastando. De nada sirve un buen presupuesto si luego no se sigue o no se sabe si se está cumpliendo o no. Es por eso que hay que establecer unos mecanismos de control para conocer en todo momento si las predicciones que se han hecho se están cumpliendo o está ocurriendo alguna desviación inesperada. Estos mecanismos e indicadores van a ser consultados y actualizados para así poder observar las desviaciones que van ocurriendo a medida que avanza el proyecto.

- **Desviación coste de horas por tarea:**
  (Horas estimadas - Horas reales) * Coste real

- **Desviación de los costes en recursos humanos por tarea:**
  (Coste estimado - Coste real) * Horas reales

- **Desviación total costes personales por actividad:**
  CPA estimado - CPA real

- **Desviación total costes generales:**
  Coste general estimado - coste general real

- **Desviación total costes imprevistos:**
  Coste imprevisto estimado - coste imprevisto real

- **Desviación total horas:**
  Horas totales estimadas - Horas totales reales

- **Desviación total costes:**
  Coste total estimado - Coste total real

Para llevar este control se utilizará una hoja de Excel con todos los valores estimados y las fórmulas descritas anteriormente, así se facilita la visualización de las desviaciones que puedan ocurrir. A continuación, se presenta un trozo de la hoja de Excel en cuestión:

Caso 2

Código	Coste estimado (h)	Coste estimado (€)	Coste Real (horas)	Coste Real (€)	Desviación (horas)	Desviación (€)
GP	Total: 70					
GP1	20	552,24	20	552,24	0	0
GP2	15	414,18	15	414,18	0	0
GP3	15	414,18	15	414,18	0	0
GP4	20	552,24	20	552,24	0	0
Desarrollo	Total: 440					
I	Total: 50					
I1	20	446,03	19,5	434,84925	-0,5	-11,15
I2	5	99,9505	5	99,9505	0	0
I3	20	373,36	20	373,36	0	0
I4	5	140,855	4,5	126,7695	-0,5	-14
US	Total: 60					
US1	15	312,2535	16,5	343,47885	1,5	31
US2	20	416,338	20	416,338	0	0
US3	5	104,0845	5	104,0845	0	0
US4	20	416,338	20	416,338	0	0
NEG	Total:130					
NEG1	30	624,507	30	624,507	0	0
NEG2	80	1665,352				
NEG3	20	416,338				
PER	Total: 50					
PER1	30	624,507				
PER2	20	419,016				
BUS	Total: 100					
BUS1	30	624,507				
BUS2	10	208,169				
BUS3	10	208,169				

Casos	Total CPA Estimado (€)	Total CPA Real (€)	Desviación (€)
C1	5490		
C2	13634,96		

Casos	Total CG Estimado (€)	Total CG Real (€)	Desviación (€)
C1	1531,6		
C2	9189,6		

Casos	Total Contingencia Estimado (€)	Total Contingencia Real (€)	Desviación (€)
C1	1053,24		
C2	3423,684		

Casos	Total Imprevistos Estimados (€)	Total Imprevistos Real (€)	Desviación (€)
C1	534,6		
C2	531,28		

Casos	Total Costest Finales Estimados (€)	Total Costest Finales Real (€)	Desviación (€)
C1	8609,44		
C2	26779,524		

Figura 5: Parte de la hoja de Excel que se utilizará como método de control de desviaciones. Fuente: Elaboración propia.

9. Sostenibilidad

Al realizar la encuesta de EDINSOST sobre el nivel de conocimiento en el ámbito de la sostenibilidad, se ha podido reflexionar sobre el nivel de conocimientos en este ámbito. La opinión personal del autor es que posee los conocimientos necesarios para analizar en profundidad el impacto en la dimensión económica de la sostenibilidad. Esto se hace a partir de cálculos de presupuesto, dando importancia a todos los recursos que se utilizan y lo que se espera generar. Sin embargo, no tiene tan claro como analizar el impacto medioambiental, sobre todo por la falta de conocimientos en indicadores que ayuden a medir y comparar los resultados en estas dimensiones. Por otro lado, a pesar de que el impacto social es difícil de medir siempre hay que tener en mente la realización de software ético, transparente y accesible para todo el mundo. En resumen, el aspecto medioambiental es el que más hay que mejorar, sobre todo en la obtención de indicadores para poder medir los resultados y compararlos para saber en qué nivel medioambiental está el proyecto que se está realizando.

9.1 Dimensión Ambiental

¿Se ha estimado el impacto ambiental que tendrá la realización del proyecto?
Al no ser un producto físico sino una solución software, el producto final no tiene un impacto medioambiental grande. Sin embargo, también se ha estimado el impacto que tendrán los recursos utilizados, como por ejemplo el ordenador utilizado o la electricidad consumida. Si bien el ordenador es de uso personal y ya se lleva utilizando muchos años, habiendo evitado comprar uno para la ocasión, la electricidad consumida proviene de energías no renovables y este es un aspecto donde se podría mejorar.

¿Se ha planteado minimizar el impacto, por ejemplo, reutilizando recursos?
Este factor se ha tenido en cuenta, se puede hacer asegurándose de utilizar energías renovables a la hora del desarrollo del proyecto y también adquiriendo menos recursos materiales, como podría ser la obtención de 5 ordenadores en vez de 6 y compartir los 5 de tal manera que se optimice su utilización sin perjudicar la productividad.

¿Se ha cuantificado el impacto ambiental de la realización del proyecto? ¿Qué medidas se han tomado para reducir el impacto? ¿Se ha cuantificado esta reducción?
El impacto ambiental de la realización de este proyecto si se ha cuantificado, y principalmente viene debido a la utilización del ordenador, así como de la energía eléctrica para hacerlo funcionar y para el lugar de trabajo del autor. A pesar de que no se tiene control sobre el reciclaje de los recursos materiales, para reducir el impacto ambiental el autor llevará el dispositivo a un punto de reciclaje para que se recicle debidamente.

¿Si se hiciera de nuevo el proyecto, podría realizarse con menos recursos?
Es poco probable poder realizar este proyecto utilizando menos recursos, ya que inicialmente ya se han utilizado muy pocos. Los recursos humanos solo han sido de una persona y los recursos materiales solo han supuesto la utilización de un ordenador (el ordenador que usa diariamente el autor) y se ha usado la vivienda personal como lugar de trabajo. Es por eso que la reducción de recursos es muy difícil.

¿Cómo se resuelve actualmente el problema que se quiere abordar (estado del arte)?
Actualmente este problema es solucionado utilizando múltiples libretas de notas o un ordenador con el programa Excel instalado. Algunos negocios ya están empezando a usar gestores de reserva o apuntando a *marketplaces* pero aún son muy pocos y la mayoría son grandes negocios que se lo pueden permitir y no los pequeños negocios.

¿En qué mejorará ambientalmente la solución a las existentes?
La solución propuesta, junto con las soluciones ya existentes de gestores de reserva y *marketplace*, permite reducir sustancialmente el uso de libretas de papel, así como utensilios para la escritura. Además, al facilitar la reserva de citas previas a través de internet, se reducirán los desplazamientos que algunas personas hacen hacia los negocios únicamente para pedir cita, contribuyendo así a la reducción de la contaminación medioambiental que producen estos desplazamientos.

¿Qué recursos se estiman que se usarán durante la vida útil del proyecto? ¿Cuál será el impacto ambiental de estos recursos?
El principal recurso que se usará durante la vida útil de este proyecto es un servidor. Este servidor será compartido para que su impacto sea menor y solo se contratarán los recursos necesarios para el proyecto. Además, los ordenadores utilizados para mantener el proyecto deberán ser debidamente amortizados y reciclados una vez dejen de servir.

¿El proyecto permitirá reducir el uso de otros recursos? ¿Globalmente, el uso del proyecto mejorará o empeorará la huella ecológica?
Los pequeños negocios, al utilizar esta herramienta software como método de gestión de citas previas, estarán reduciendo en gran cantidad el uso tanto de libretas como de bolígrafos ya que no necesitarán de ellos para apuntar las citas previas. Por lo tanto, globalmente el proyecto mejorará la huella ecológica al reducir no solo el gasto papel, sino también en la producción de estos artículos.

¿Podrían producirse escenarios que hiciesen aumentar la huella ecológica del proyecto?
El escenario que podría hacer aumentar la huella ecológica de este proyecto es el de aumentar recursos exageradamente sin ningún tipo de necesidad, como podría ser la compra de ordenadores nuevos mucho más potentes de lo necesario, la utilización de oficinas que consuman mucho, etc. Para ello, hay que comprometerse a utilizar únicamente lo necesario, hacer un buen uso de ello y reutilizarlo y reciclarlo debidamente. Además, preocuparse también de la energía que se usa, tanto directa como indirectamente, priorizando siempre energías renovables.

9.2 Dimensión Económica

¿Se ha estimado el coste de la realización del proyecto (recursos humanos y materiales)?
Se ha elaborado un presupuesto exhaustivo con todos los recursos ya sean humanos, materiales y generales para poder estimar los costes de la realización del proyecto.

¿Se ha cuantificado el coste (recursos humanos y materiales) de la realización del proyecto? ¿Qué decisiones se han tomado para reducir el coste? ¿Se ha cuantificado el ahorro de estas decisiones?
Tanto el coste de recursos humanos como el de materiales fue cuantificado en la planificación inicial. Aunque no fue una decisión, el hecho de realizar este proyecto solo una persona y no un equipo entero ha reducido mucho los costes en recursos humanos. Los costes materiales también se han visto reducidos, especialmente el relacionado con el sitio de trabajo, ya que a pesar de que el presupuesto se hizo contando que el proyecto se realizaría en un área de trabajo compartido (con los costes que eso supone), la totalidad del proyecto se ha realizado desde la casa del autor, eliminando el coste de la estancia de ese *workspace* y los desplazamientos a este.

¿Se ha ajustado el coste previsto al coste final? ¿Se han justificado las diferencias?
El coste final dista muy poco del coste previsto, debido a las desviaciones que se han producido en algunas tareas. Sin embargo, el desvío es muy pequeño y entra dentro de los gastos asignados a imprevistos y contingencia, así que la planificación inicial fue bastante buena.

¿En qué mejorará económicamente la solución a las existentes?
La solución provee a los pequeños negocios de una plataforma para las reservas online con coste por uso, es decir, si estos pequeños negocios no reciben reservas a través de la solución no se les va a cobrar. Además, mejorará la productividad de los empleados, ya que no tendrán que gastar tanto tiempo en la toma de reservas, afectando positivamente a la economía del pequeño negocio.

¿Qué coste se estima que tendrá el proyecto durante su vida útil? ¿Se podría reducir este coste para hacerlo más viable? ¿Se ha tenido en cuenta el coste de los ajustes/actualizaciones/reparaciones durante la vida útil del proyecto?
El coste del proyecto durante su vida útil viene determinado por el uso de los servidores del *backend*. Para poder reducir este coste, es conveniente la utilización de servidores compartidos. Además, gracias a las tecnologías utilizadas, las actualizaciones tanto del servidor como de la app son más rápidas y sencillas, pudiendo así reducir costes en este aspecto.

¿Podrían producirse escenarios que perjudicasen la viabilidad del proyecto?
Después de hacer un profundo análisis de la viabilidad de este proyecto, se cree que es muy improbable que se produzca algún escenario que perjudique la viabilidad. Sin embargo, todo es posible y la solución para ello es estar constantemente analizando el mercado y las necesidades para dar respuesta en todo momento y avanzarse a los cambios de tendencias para así pivotar adecuadamente y no quedarse atrás.

9.3 Dimensión Social

¿Qué crees que va a aportar a nivel personal la realización de este proyecto?
Gracias a la realización de este proyecto, el autor ganará conocimientos y experiencia en el desarrollo de aplicaciones para móviles en las tecnologías usadas.

Además, se entenderá mejor cómo funcionan los pequeños negocios y cuál es su organización, así como sus dificultades (mayormente económicas) a la hora de adaptarse a las nuevas corrientes tecnológicas.

¿La realización de este proyecto ha implicado reflexiones significativas a nivel personal, profesional o ético de las personas que han intervenido?
Este proyecto ha implicado todo tipo de reflexiones para el autor, principalmente la de estar siempre pensando si el software va a poder ser usado por una gran variedad de personas, ya que este proyecto en concreto ha de poder ser usado por todo el mundo. También se ha reflexionado sobre el impacto económico que tendrá sobre las personas y negocios, y quien de estos ha de asumir el coste.

¿Existe una necesidad real del proyecto?
Desde el punto de vista del autor de este proyecto, sí que existe una necesidad real. Esta necesidad viene dada a la migración inevitable hacia las nuevas tecnologías que ya tienen muchísima presencia y aún van a tenerla más. A todo esto, se suma la actual pandemia mundial que está acelerando este proceso y que está haciendo que la necesidad de este proyecto se acentúe.

¿Quién se beneficiará del uso del proyecto? ¿Hay algún colectivo que puede verse perjudicado por el proyecto? ¿En qué medida?
Tanto los pequeños negocios como los clientes saldrán beneficiados por el proyecto. Los pequeños negocios podrán tener a su disposición un gestor de citas previas que además sirve de *marketplace* para darles a conocer, y los clientes por su lado podrán pedir citas previas online de manera más fácil y rápida desde un mismo sitio.

El colectivo que podría verse perjudicado es aquel que no está familiarizado con el uso de la tecnología, como podrían ser las personas mayores. Sin embargo, esto ya se contempló y se dio la funcionalidad al software para que si esta gente fuera a pedir cita presencialmente o por otras vías al pequeño negocio este pudiera añadir la cita al gestor. De esta forma se incluyen todos los colectivos.

¿En qué medida soluciona el proyecto el problema planteado inicialmente?
Creo que este proyecto soluciona en gran medida la necesidad de los clientes de tener una forma más fácil, rápida y cómoda de pedir cita previa en negocios que no tienen disponible un sistema de citas previas, que hoy en día siguen siendo muchos. Además, también es una forma de potenciar el negocio local y el descubrimiento de nuevos negocios.

¿Podría crear el proyecto algún tipo de dependencia que dejase a los usuarios en posición de debilidad?
Al ser un proyecto orientado tanto a negocios como a clientes, es difícil que cree algún tipo de dependencia, ya que se regulan entre ellos. Sin embargo, en un futuro y si su utilización crece muchísimo, sobretodo en términos de clientes, puede que los negocios si que sufrieran algún tipo de dependencia. En ese caso, hay que comprometerse como empresa a no abusar de esto. En cambio, la dependencia de los clientes está descartada, ya que debido a las funcionalidades del software ellos pueden seguir utilizando las vías tradicionales para pedir cita previa sin usar esta solución.

10. Legislación

La privacidad y la seguridad son dos derechos fundamentales que cualquier sistema software ha de garantizar a todos sus usuarios. Para asegurarse de ello, tanto autoridades nacionales como internacionales han establecido normativas y leyes que se han de cumplir en relación a estos derechos. El incumplimiento de las legislaciones puede derivar en sanciones muy importantes tanto económicamente como penalmente. Es por eso que a la hora de desarrollar un software se ha de ser muy consciente de la legislación vigente y las medidas que han de ser tomadas y garantizadas con el fin de no incumplir ninguna de estas leyes. En este capítulo se va a hablar de las 2 legislaciones que atañen a este proyecto: LOPD y GRPD.

10.1 Reglamento general de protección de datos (RGPD)

El Reglamento General de Protección de Datos (RGPD) es una ley europea que busca proteger los datos personales y la forma en la que las empresas y organizaciones los procesan, almacenan y destruyen. No solo eso, sino que esta ley también dictamina las sanciones que pueden ser impuestas a aquellas personas u empresas que no cumplan con este reglamento. Al ser una normativa a nivel europeo, cualquier empresa o persona de la Unión Europea ha de cumplirla.

El RGPD establece ocho derechos relativos al tratamiento de datos personales que han de ser garantizados por la persona, empresa u organización que los trata:

- Derecho a estar informado
- Derecho al acceso
- Derecho a la rectificación
- Derecho al olvido
- Derecho a restringir el procesamiento
- Derecho a la portabilidad de datos
- Derecho a objetar
- Derecho sobre la toma de decisiones

En el caso de incumplimiento de este reglamento, pueden imponerse alguna de las siguientes sanciones en función de la gravedad de la infracción:

- Una advertencia por escrito en los casos de incumplimiento previo e intencional
- Auditorías periódicas de protección de datos
- Una multa de hasta 10 000 000 de euros o hasta el 2 % del volumen de negocios mundial anual del ejercicio anterior
- Una multa de hasta 20 000 000 de euros o hasta el 4 % del volumen de negocios anual del año financiero anterior

10.2 Ley Orgánica de Protección de Datos y Garantía de Derechos Digitales (LOPDGDD)

La Ley Orgánica de Protección de Datos de Carácter Personal y Garantía de Derechos digitales (LOPDGDD), es una ley orgánica española que tiene como objeto regular el tratamiento de datos de carácter personal en España. Anteriormente existía la LOPD, sin embargo, esta ley quedó obsoleta debido a la RGPD, la ley europea mencionada anteriormente. Es por eso que surgió la necesidad de crear una nueva ley española de protección de datos que se adaptara al reglamento europeo.

Esta ley afecta a toda persona, empresa u organismo que opere en territorio español y que tenga en su posesión datos de carácter personal. La obliga a cumplir y establecer una serie de normas y medidas de seguridad para el tratamiento y almacenamiento de estos datos. De forma resumida, la LOPDGDD establece tres principios básicos:

Básicamente pueden quedar establecidos en tres principios:

- Exactitud de los datos: por lo tanto, estos han de ser actualizados periódicamente.
- Deber de confidencialidad: para todas aquellas personas involucradas en el tratamiento de estos datos.
- Consentimiento: es necesario la autorización del individuo para poder recabar y tratar los datos. Por lo tanto, este ha de ser informado en todo momento de lo que se le pide y qué se va a hacer con los datos.

El titular de los datos personales debe ser informado sobre todos los medios disponibles para ejercer sus derechos respecto a estos datos. Estos derechos son los recogidos por la RGPD mencionada anteriormente. Además, las sanciones de esta ley son las mismas que la ley europea.

11. Requisitos

11.1 Justificación de los requisitos

Para desarrollar un sistema antes hace falta especificar las condiciones que ha de satisfacer. Estas condiciones son los llamados requisitos. En función de los requisitos que se especifiquen, el software satisfará unas condiciones u otras, y por extensión se cubrirán unas necesidades concretas del cliente. Es por eso que la especificación de requisitos es una de las partes más importantes de cualquier proyecto, ya que si no se hace bien puede ocurrir que no se satisfagan parcial o completamente las necesidades del cliente.

Este proyecto en concreto no tiene un cliente específico y por lo tanto está sujeto a las necesidades del mercado y a sus clientes. El grado de conocimiento del mercado determinará lo bien que se especifiquen los requisitos y consecuentemente, lo bien que se satisfagan las necesidades. A pesar de que el autor del proyecto sea un futuro cliente de la solución propuesta y por lo tanto conozca en cierta medida las necesidades actuales, no podemos afirmar que estas necesidades coincidan plenamente con las necesidades del mercado en general. Para conocer en profundidad el mercado en cuestión y sus clientes, se ha decidido hacer dos encuestas, una a los posibles clientes y otra a los posibles negocios de la solución propuesta. Cada encuesta tiene la finalidad de conocer mejor a cada tipo de cliente, ya sea cliente o negocio, y averiguar cuáles son sus necesidades para posteriormente poder hacer una especificación de requisitos. Estas dos encuestas y sus resultados pueden ser encontrados como anexos de este documento.

Una vez hechas las encuestas y analizado la competencia en capítulos anteriores, podemos afirmar que se conocen mejor las necesidades de los clientes en cuestión y que la especificación de requisitos ha ido acorde con estos resultados.

11.2 Esquema conceptual

Con el fin de especificar el dominio del problema, se presenta a continuación un esquema conceptual realizado en lenguaje UML [19].

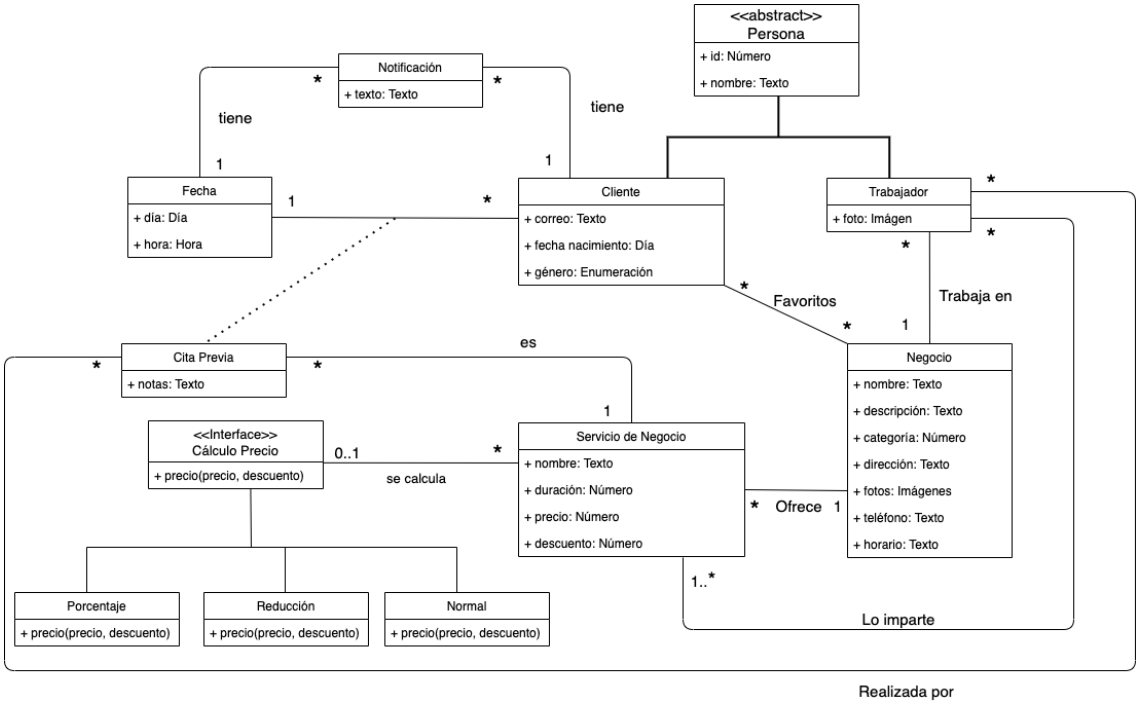


Figura 6: Esquema conceptual en UML. Fuente: Elaboración propia.

11.3.1 Restricciones textuales

1. Claves externas de las clases no asociativas: (Persona, id), (Cliente, id), (Trabajador, id), (Negocio, nombre + dirección), (Servicio de Negocio, nombre + Negocio), (Fecha, día + hora), (Notificación, texto + Cliente)
2. La fecha de una cita previa ha de ser posterior a la fecha actual.
3. Un cliente no puede tener dos o más citas superpuestas por fecha.
4. Un trabajador no puede tener dos o más citas superpuestas por fecha
5. Un trabajador no puede impartir un servicio que no es ofrecido por el negocio donde trabaja.
6. El trabajador de una cita previa automática ha de impartir el servicio de la cita.

11.3.2 Explicación del esquema conceptual

En este esquema conceptual, principalmente existen dos entidades importantes, los clientes (también llamados usuarios) y los negocios. Por un lado, los clientes representan a aquellas personas que quieren pedir citas previas y su información personal está expresada como atributos.

Por otro lado, los negocios representan a los negocios que están disponibles en el sistema, con su información también como atributos. Estos negocios tienen trabajadores que trabajan ahí y ofrecen distintos servicios. Estos servicios hacen uso de una interfaz para calcular su precio (ya sea porque tienen un descuento expresado en porcentaje, un descuento de una cantidad fija de dinero o simplemente el precio normal).

Cuando el cliente pide una cita previa en un negocio, esta queda asociada a una hora, un servicio y un trabajador.

11.4 Requisitos funcionales

Los requisitos funcionales definen los efectos funcionales que el software ha de tener en su entorno. Este tipo de requisitos serán listados a continuación como casos de uso. Un caso de uso es un conjunto de acciones realizadas por actores, que producen un resultado observable que es valioso para uno o más actores del sistema. Cada caso de uso contiene los siguientes atributos:

- **Actor principal**: actor que realizará el caso de uso.
- **Precondiciones**: todo lo que se ha de cumplir antes de realizar el caso.
- **Disparador**: evento o acción que desencadenará el inicio del caso.
- **Escenario de éxito**: pasos que deberá seguir el cliente para la correcta realización de ese caso.
- **Extensiones**: posibles errores o circunstancias que pueden aparecer durante la ejecución del caso.

Se supone que los clientes poseen una conexión a internet estándar desde su dispositivo en todo momento de tal forma que puedan acceder a todas las funcionalidades de la plataforma. Esta es una precondición de todos los requisitos expuestos a continuación, así como una extensión de todos ellos, ya que si en algún momento se pierde esta conexión el cliente no podrá continuar usando la plataforma.

En este sistema, a parte de él solo existe un actor más: el cliente. Es por eso que todos los casos de uso que se presentan a continuación pertenecen a este actor.

Caso de uso 1	**Registro**
Actor principal	Cliente
Precondiciones	El cliente no está autenticado
Disparador	El cliente cuando selecciona la opción de registrarse en el sistema.
Escenario de éxito	1. Al seleccionar esta opción, el cliente es redirigido al formulario de registro. 2. El cliente rellena todos los campos de forma correcta. 3. El sistema valida los campos y añade al cliente en la base de datos. 4. El sistema envía un correo al cliente para que valide su cuenta.
Extensiones	**Campos incorrectos** El cliente a la hora de rellenar los campos lo hace de forma incorrecta o no los rellena todos. En este caso, el sistema al validar los campos avisa al cliente el error que ha cometido. **Cliente existente** El cliente ya se ha registrado previamente en el sistema. El sistema detecta este hecho y no añade otra vez al cliente en la base de datos.

Tabla 13: Caso de uso 1 - Registro. Fuente: Elaboración propia.

Caso de uso 2	***Log In***
Actor principal	Cliente
Precondiciones	El cliente se ha registrado previamente en el sistema. El cliente ha validado su cuenta. El cliente aún no se ha autenticado.
Disparador	El cliente cuando quiere autenticarse en el sistema.
Escenario de éxito	1. El cliente introduce sus credenciales privadas (correo y contraseña) de forma correcta. 2. El cliente es redirigido hacia la pantalla principal.
Extensiones	**Credenciales erróneas** El cliente introduce unas credenciales que no existen en la base de datos. En ese caso se le informa de que esas credenciales no son correctas.

Tabla 14: Caso de uso 2 - Login. Fuente: Elaboración propia.

Caso de uso 3	**Contraseña olvidada**
Actor principal	Cliente
Precondiciones	Ninguna
Disparador	El cliente cuando selecciona la opción de contraseña olvidada.
Escenario de éxito	1. El cliente selecciona esta opción y es redirigido a un formulario donde ha de introducir su correo 2. El cliente introduce de manera correcta su correo. 3. El sistema comprueba que el cliente existe y envía un correo al cliente con un link para cambiar su contraseña.
Extensiones	**Cliente inexistente** El cliente introduce un correo que no está asociado a ningún cliente. El sistema detecta esto y no envía ningún correo de cambio de contraseña.

Tabla 15: Caso de uso 3 - Contraseña olvidada. Fuente: Elaboración propia.

Caso de uso 4	***Log Out***
Actor principal	Cliente
Precondiciones	El cliente ya se ha autenticado en el sistema.
Disparador	El cliente cuando quiere salir de su cuenta.
Escenario de éxito	1. El cliente selecciona la opción de salir de la cuenta. 2. El sistema borra el *token* de sesión del cliente. 3. El cliente es echado del sistema y redirigido al *log in*.
Extensiones	Ninguna.

Tabla 16: Caso de uso 4 - Log Out. Fuente: Elaboración propia.

Caso de uso 5	**Consultar información de negocio**
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema. El negocio existe en el sistema.
Disparador	El cliente cuando entra en la pantalla de un negocio.
Escenario de éxito	1. El cliente navega hasta la pantalla de un negocio para consultar su información.
Extensiones	Ninguna.

Tabla 17: Caso de uso 5 - Consultar información de negocio. Fuente: Elaboración propia.

Caso de uso 6	**Agregar negocio a favoritos**
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema. El negocio existe en el sistema.
Disparador	El cliente cuando selecciona la opción de agregar/quitar un negocio a/de favoritos
Escenario de éxito	1. El cliente navega hasta el negocio. 2. El negocio en cuestión no es favorito para el cliente. 3. El cliente selecciona la opción de agregar/quitar el negocio a/de favoritos. 3. El sistema pide confirmación al cliente para agregar el negocio. 4. El cliente da su confirmación y el sistema añade el negocio como favorito para el cliente.
Extensiones	**Denegación de confirmación** El cliente deniega la confirmación al sistema y este no realiza ninguna acción. **Negocio ya añadido** El cliente ya tiene este negocio como favorito. En este caso, el sistema preguntaría al cliente si quiere quitar el negocio de sus favoritos y empezaría el caso de uso 7.

Tabla 18: Caso de uso 6 - Agregar negocio a favoritos. Fuente: Elaboración propia.

Caso de uso 7	**Quitar negocio de favoritos**
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema. El negocio existe en el sistema.
Disparador	El cliente cuando selecciona la opción de agregar/quitar un negocio a/de favoritos
Escenario de éxito	1. El cliente navega hasta el negocio. 2. El negocio en cuestión es favorito para el cliente. 3. El cliente selecciona la opción de quitar de favoritos. 3. El sistema pide confirmación al cliente para quitar el negocio. 4. El cliente da su confirmación y el sistema quita el negocio de los negocios favoritos para el cliente.
Extensiones	**Denegación de confirmación** El cliente deniega la confirmación al sistema y este no realiza ninguna acción. **Negocio no favorito** El cliente no tiene este negocio como favorito. En este caso, empezaría el caso de uso 6 para agregar el negocio a favoritos.

Tabla 19: Caso de uso 7 - Quitar negocio de favoritos. Fuente: Elaboración propia.

Caso de uso 8	**Consultar cita previa**
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando entra en el calendario del sistema.
Escenario de éxito	1. El cliente navega hasta el calendario del sistema. 2. El cliente selecciona el día que quiera para ver las citas previas que tiene ese día. 3. El cliente selecciona la cita que quiere ver y el sistema le muestra los detalles de esa cita previa.
Extensiones	**Ningún resultado** Si no existen citas previas se muestra un calendario vacío y un mensaje explicando esta situación.

Tabla 20: Caso de uso 8 - Consultar cita previa. Fuente: Elaboración propia.

Caso de uso 9	**Creación manual de cita previa**
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando inicia la creación manual de una cita previa.
Escenario de éxito	1. El cliente selecciona la opción de agregar una cita previa manual al sistema. 2. El sistema lo redirige a un formulario de creación de citas previas manuales. 3. El cliente rellena todos los campos y de manera correcta. 4. Cuando el cliente selecciona agregar el sistema añade esta cita previa con los datos introducidos en los campos.
Extensiones	**Campos vacíos, incorrectos o no válidos** El cliente no rellena todos los campos o lo hace introduciendo valores incorrectos o no válidos. **Cita solapada** El sistema avisará al cliente de que la cita que quiere crear se solapa con una cita previamente creada.

Tabla 21: Caso de uso 9 - Creación de cita previa manual. Fuente: Elaboración propia.

Caso de uso 10	Edición de cita previa manual
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema. La cita previa manual existe en el sistema.
Disparador	El cliente cuando quiere editar una cita previa manual.
Escenario de éxito	1. El cliente selecciona la cita previa que quiere editar. 2. El cliente selecciona la opción de editar esa cita. 3. El cliente modifica de manera correcta la información de esa cita previa. 4. El sistema pide confirmación al cliente y una vez confirmado modifica la información de esa cita en la base de datos.
Extensiones	**Campos vacíos, incorrectos o no válidos** El cliente no rellena todos los campos o lo hace introduciendo valores incorrectos o no válidos. **Cita solapada** El sistema avisará al cliente de que con los nuevos valores la cita que quiere modificar se solapa con una cita previamente creada, ya sea manual o automática. **Denegación de confirmación** El cliente deniega la confirmación al sistema y este no realiza ninguna acción.

Tabla 22: Caso de uso 10 - Edición de cita previa manual. Fuente: Elaboración propia.

Caso de uso 11	Eliminación de cita previa manual
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema. La cita previa manual existe en el sistema.
Disparador	El cliente cuando quiere eliminar una cita previa manual.
Escenario de éxito	1. El cliente selecciona la cita previa manual que quiere eliminar. 2. El cliente selecciona la opción de editar esa cita y posteriormente selecciona la opción de eliminar la cita previa manual. 3. El sistema pide confirmación y cuando el cliente se la da el sistema borra la cita previa manual del cliente.
Extensiones	**Denegación de confirmación** El cliente deniega la confirmación al sistema y este no realiza ninguna acción.

Tabla 23: Caso de uso 11 - Eliminación de cita previa manual. Fuente: Elaboración propia.

Caso de uso 12	Pedir cita previa en negocio
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema. El negocio existe en el sistema. El negocio tiene servicios disponibles. El negocio tiene trabajadores.
Disparador	El cliente cuando selecciona la opción de pedir cita previa en el negocio.
Escenario de éxito	1. El cliente entra en la pantalla del negocio en cuestión. 2. El cliente selecciona el servicio del cual quiere pedir cita previa. 3. El cliente selecciona el trabajador que quiere que le haga el servicio. 4. El cliente selecciona la fecha en la que quiere recibir el servicio y finalmente rellena otra información adicional. 5. El sistema muestra los datos recabados junto con una confirmación que el cliente ha de aceptar. 6. El cliente revisa los datos introducidos y acepta la confirmación. 7. El sistema crea la cita previa con los datos introducidos y esta pasa a estar disponible en el apartado de citas del cliente.
Extensiones	**Trabajador no ofrece servicio** El sistema impedirá que el cliente seleccione trabajadores que no ofrezcan el servicio previamente seleccionado. **Cita solapada** El sistema avisará al cliente de que la cita que quiere crear se solapa con una cita previamente creada, ya sea manual o automática y no dejará crearla. **Fecha anterior** El sistema impedirá que el cliente seleccione una fecha anterior a la actual. **Denegación de confirmación** El cliente deniega la confirmación al sistema y este no realiza ninguna acción.

Tabla 24: Caso de uso 12 – Pedir cita previa en negocio. Fuente: Elaboración propia.

Caso de uso 13	Cancelar cita previa automática
Actor principal	Cliente.
Precondiciones	El cliente está autenticado en el sistema. La cita previa automática existe en el sistema.
Disparador	El cliente cuando quiere cancelar una cita previa automática.
Escenario de éxito	1. El cliente selecciona la cita previa automática que quiere cancelar. 2. El cliente selecciona la opción de cancelar esa cita previa. 3. El sistema pide confirmación al cliente y una vez dada se cancela esta cita.
Extensiones	**Denegación de confirmación** El cliente deniega la confirmación al sistema y este no realiza ninguna acción.

Tabla 25: Caso de uso 13 - Cancelar cita previa automática. Fuente: Elaboración propia.

Caso de uso 14	Buscar negocios por nombre
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando quiere buscar negocios en función de su nombre.
Escenario de éxito	1. El cliente selecciona la opción de búsqueda de negocios. 2. El cliente selecciona la opción de búsqueda por nombre e introduce el nombre a buscar. 3. El sistema muestra al cliente una lista de todos los negocios que tienen ese nombre.
Extensiones	**Ningún resultado** Si no existen negocios con este nombre se muestra una lista vacía y un mensaje indicando esta situación.

Tabla 26: Caso de uso 14 - Buscar negocios por nombre. Fuente: Elaboración propia.

Caso de uso 15	Buscar negocios por ciudad
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando quiere buscar negocios en función de su ciudad.
Escenario de éxito	1. El cliente selecciona la opción de búsqueda de negocios. 2. El cliente selecciona la opción de búsqueda por ciudad. 3. El sistema muestra el conjunto de ciudades disponibles y el cliente selecciona una. 3. El sistema muestra al cliente una lista de todos los negocios que están en esa ciudad.
Extensiones	**Ningún resultado** Si no existen negocios en esta ciudad se muestra una lista vacía y un mensaje indicando esta situación.

Tabla 27: Caso de uso 15 - Buscar negocios por ciudad. Fuente: Elaboración propia.

Caso de uso 16	Buscar negocios por categoría
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando quiere buscar negocios en función de su categoría.
Escenario de éxito	1. El cliente selecciona la opción de búsqueda de negocios. 2. El cliente selecciona la opción de búsqueda por categoría. 3. El sistema muestra el conjunto de categorías disponibles y el cliente selecciona una. 3. El sistema muestra al cliente una lista de todos los negocios que son de esa categoría.
Extensiones	**Ningún resultado** Si no existen negocios de esta categoría se muestra una lista vacía y un mensaje indicando esta situación.

Tabla 28: Caso de uso 16 - Buscar negocios por categoría. Fuente: Elaboración propia.

Caso de uso 17	Cambiar visualización de los resultados
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando quiere cambiar la visualización de los resultados de las búsquedas.
Escenario de éxito	1. El cliente selecciona la opción de búsqueda de negocios. 2. El cliente selecciona la opción de cambio de visualización de los resultados. 3. En caso de estar en lista, el modo de visualización pasa a mapa, en caso de estar en mapa pasa a lista.
Extensiones	Ninguna

Tabla 29: Caso de uso 17 - Cambiar visualización de los resultados. Fuente: Elaboración propia.

Caso de uso 18	Consultar siguiente cita
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando está en la pantalla principal
Escenario de éxito	1. El cliente navega a la pantalla principal. 2. El cliente es capaz de ver su próxima cita previa que tiene. 3. El cliente selecciona la cita para ver sus detalles
Extensiones	**Ningún resultado** Si no existe una próxima cita para el cliente se muestra un mensaje indicando esta situación.

Tabla 30: Caso de uso 18 - Consultar siguiente cita. Fuente: Elaboración propia.

Caso de uso 19	Consultar negocios favoritos
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando está en la pantalla principal
Escenario de éxito	1. El cliente navega a la pantalla principal. 2. El cliente es capaz de ver una lista de sus negocios favoritos.
Extensiones	**Ningún resultado** Si el cliente no tiene negocios favoritos se muestra una lista vacía y un mensaje indicando esta situación.

Tabla 31: Caso de uso 19 - Consultar negocios favoritos. Fuente: Elaboración propia.

Caso de uso 19	Consultar promociones
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando está en la pantalla principal.
Escenario de éxito	1. El cliente navega a la pantalla principal. 2. El cliente es capaz de ver una lista de las promociones que actualmente existen en el sistema.
Extensiones	**Ningún resultado** Si no existen promociones en el sistema se muestra una lista vacía y un mensaje indicando esta situación.

Tabla 32: Caso de uso 20 - Consultar promociones. Fuente: Elaboración propia.

Caso de uso 20	Editar información personal
Actor principal	Cliente
Precondiciones	El cliente está autenticado en el sistema.
Disparador	El cliente cuando quiere editar su información personal
Escenario de éxito	1. El cliente navega hasta la pantalla de información de la cuenta. 2. El cliente selecciona la opción de editar información personal. 3. El cliente edita los campos que cree necesarios. 4. El sistema valida los campos y pide confirmación al cliente. 5. El cliente confirma la edición y el sistema cambia los valores en la base de datos.
Extensiones	**Campos vacíos, incorrectos o no válidos** El cliente no rellena todos los campos o lo hace introduciendo valores incorrectos o no válidos. **Denegación de confirmación** El cliente deniega la confirmación al sistema y este no realiza ninguna acción.

Tabla 33: Caso de uso 21 - Editar información personal. Fuente: Elaboración propia.

11.5 Requisitos no funcionales

Aparte de los requisitos funcionales, también se exponen los requisitos no funcionales que serán necesarios para el buen funcionamiento de la plataforma. Cada requisito tendrá su nombre, tipo (según la clasificación de Volere [20]), descripción, justificación de por qué se ha decidido que es necesario y finalmente la condición de satisfacción, condición objetiva que el sistema ha de cumplir para considerar ese requisito como satisfecho.

Cabe mencionar que durante las explicaciones de los requisitos no funcionales se menciona a los *beta testers*. Este grupo de clientes son personas cercanas al autor que junto con él probarán los prototipos que se vayan generando de la plataforma y darán *feedback* de su experiencia des del punto de vista de futuro cliente de la plataforma.

Requisito no funcional 1	**Apariencia**
Tipo	10 A.
Descripción	La plataforma ha de ser atractiva y ha de gustar visualmente a los clientes que la utilicen.
Justificación	La plataforma ha de ser atractiva y sencilla debido a que se quiere que los clientes la utilicen diariamente para su planificación y organización. Para ello, hay que facilitar su uso y hacer que el cliente se sienta cómodo usando la plataforma, ya que sino no la utilizará.
Condición de satisfacción	Se dará por alcanzado este requisito si durante el desarrollo de la plataforma se han utilizado los patrones y estilos definidos para los sistemas de Android y iOS.

Tabla 34: Requisito no funcional 1 - Apariencia. Fuente: Elaboración propia.

Requisito no funcional 2	**Usabilidad**
Tipo	11 A.
Descripción	La plataforma tiene que ser fácil de utilizar para todos los distintos tipos de persona, roles y perfiles. Se espera que la edad no sea un inconveniente para utilizarla.
Justificación	La plataforma está destinada a ser utilizada por un gran número de personas con perfiles muy distintos: conocimientos distintos, objetivos distintos, edades distintas, etc. Es por eso que es muy importante que la plataforma sea intuitiva y fácil de utilizar, ya que sino mucha gente podría renunciar a su uso.
Condición de satisfacción	Se dará por alcanzado este requisito si durante las fases de *testing* de las distintas funcionalidades, la plataforma es considerada intuitiva y fácil de utilizar. El autor será el encargado de valorar este aspecto junto con la ayuda de otros clientes que formarán un grupo de beta *testers*.

Tabla 35: Requisito no funcional 2 - Usabilidad. Fuente: Elaboración propia.

Requisito no funcional 3	**Fiabilidad y disponibilidad**
Tipo	12 D.
Descripción	La plataforma ha de estar operativa en cualquier momento y hay que evitar que ocurran caídas que puedan afectar a su funcionamiento.
Justificación	Es muy importante que todos los clientes puedan hacer uso de esta app en cualquier momento del día, ya que esto es precisamente uno de los objetivos de la plataforma. También es importante que no ocurran fallos de forma reiterada porque lo que se pretende es que los clientes utilicen de forma activa la plataforma y esta situación podría impedir esto.
Condición de satisfacción	Se dará por alcanzado este requisito si durante las fases de *testing* de las distintas funcionalidades, así como las fases de *testing* de integración, la plataforma se mantiene activa un 95% del tiempo de pruebas.

Tabla 36: Requisito no funcional 3 - Fiabilidad y Disponibilidad. Fuente: Elaboración propia.

Requisito no funcional 4	**Aprendizaje**
Tipo	11 C.
Descripción	La plataforma no tiene que ser muy complicada para conseguir un aprendizaje rápido y fácil de todas las funcionalidades que ofrece.
Justificación	Como se ha mencionado anteriormente, la plataforma está destinada a ser utilizada por un gran número de personas durante su día a día. Por lo tanto, si no se familiarizan rápido y no descubren el potencial de todas las funcionalidades lo más probable es que dejen de utilizarla, ya que la plataforma será más difícil que los métodos tradicionales.
Condición de satisfacción	Se dará por alcanzado este requisito si durante las fases de *testing* de las distintas funcionalidades, la plataforma es considerada de aprendizaje rápido y sencillo. El autor y un grupo de beta *testers* serán los encargados de valorar este aspecto.

Tabla 37: Requisito no funcional 4 - Aprendizaje. Fuente: Elaboración propia.

Requisito no funcional 5	Internacionalización
Tipo	11 B
Descripción	La plataforma tiene que ser internacional, es decir, no solo estar disponible en castellano sino también en inglés.
Justificación	A pesar de que sea una plataforma pensada para España, si desde un principio se piensa en la internacionalización es mucho más fácil incluirla en vez de hacerlo una vez ha sido acabada. Además, esta característica ayuda a que pueda ser usada por un mayor número de personas.
Condición de satisfacción	Se dará por alcanzado este requisito si una vez acabado el desarrollo, todos los textos de la plataforma están traducidos tanto al inglés como al español.

Tabla 38: Requisito no funcional 5 - Internacionalización. Fuente: Elaboración propia.

Requisito no funcional 6	Velocidad y latencia
Tipo	12 A.
Descripción	La plataforma tiene que tener una velocidad y latencia adecuadas a las tecnologías usadas y el presupuesto invertido.
Justificación	La velocidad y latencia han de satisfacer las expectativas del cliente para que así pueda tener una experiencia satisfactoria con la plataforma. Ningún cliente usaría una plataforma que resolviera su problema de forma más lenta que la manera actual de resolverlo.
Condición de satisfacción	Se dará por alcanzado este requisito si durante el *testing* los resultados de las pruebas de velocidad y latencia son satisfactorios en un 95% de los casos. Estas pruebas consistirán en la repetición de situaciones donde el rendimiento de velocidad y latencia sea importante, anotando los resultados obtenidos y comparándolos con los valores estándar de la industria de las apps.

Tabla 39: Requisito no funcional 6 - Velocidad y Latencia. Fuente: Elaboración propia.

Requisito no funcional 7	**Capacidad**
Tipo	12 F.
Descripción	La plataforma tiene que ser capaz de gestionar todo el tráfico de datos de los clientes sin que se produzcan errores ni inconvenientes.
Justificación	El sistema ha de estar preparado para gestionar grandes cantidades de información de todos los clientes, soportando subidas y bajadas del tráfico de datos.
Condición de satisfacción	Se dará por alcanzado este requisito si durante el *testing* el sistema soporta el tráfico de datos simultáneo de todos los beta *testers*. Además, se ha de contar con un plan para el aumento en la capacidad de la plataforma y sobretodo la base de datos tanto en términos de almacenamiento como tráfico.

Tabla 40: Requisito no funcional 7 - Capacidad. Fuente: Elaboración propia.

Requisito no funcional 8	**Escalabilidad**
Tipo	12 G.
Descripción	La plataforma tiene que estar preparada para soportar un incremento constante de clientes y de negocios.
Justificación	La plataforma está dirigida a negocios y clientes, por lo tanto, si se hace popular y cada vez más negocios y clientes quieren usarla, esta ha de estar preparada para soportar este incremento en el volumen de datos.
Condición de satisfacción	Se dará por alcanzado este requisito si durante el desarrollo se tiene en mente esta circunstancia, se prevé este aumento y se toman decisiones en base a esta situación. Además, en el *testing* se harán pruebas donde todos los *beta testers* interactuarán simultáneamente.

Tabla 41: Requisito no funcional 8 - Escalabilidad. Fuente: Elaboración propia.

Requisito no funcional 9	Adaptabilidad
Tipo	14 C.
Descripción	La plataforma tiene que ser capaz de funcionar correctamente en cualquier clase de dispositivo móvil independientemente de su tamaño o de si su sistema operativo es Android o iOS.
Justificación	Debido a que una gran cantidad de gente utilizará esta plataforma, ha de estar disponible en la mayor variedad de dispositivos móviles. Esto engloba a los dos grandes sistemas operativos Android y iOS, que respectivamente están presentes en dispositivos móviles con una gran variedad de dimensiones de pantalla.
Condición de satisfacción	Se dará por alcanzado este requisito si el prototipo final de la plataforma funciona correctamente en cualquier clase de dispositivo móvil independientemente de su tamaño o de si su sistema operativo es Android o iOS.

Tabla 42: Requisito no funcional 9 - Adaptabilidad. Fuente: Elaboración propia.

Requisito no funcional 10	Seguridad y privacidad
Tipo	15 A y 15 C.
Descripción	La plataforma tiene que asegurarse de que todas las funcionalidades y datos sean accesibles solo para aquellos clientes autorizados a utilizarlas.
Justificación	La plataforma tendrá datos privados de los clientes, por eso es sumamente importante que se mantenga un alto nivel tanto de privacidad como de seguridad.
Condición de satisfacción	Se dará por alcanzado este requisito si durante el desarrollo de la plataforma y de la base de datos se han cumplido todas las leyes y se ha comprobado que los datos de los clientes son inaccesibles para cualquier cliente que no esté autorizados a verlos.

Tabla 43: Requisito no funcional 10 - Seguridad y Privacidad. Fuente: Elaboración propia.

Requisito no funcional 11	Legislación
Tipo	17 A.
Descripción	La plataforma debe cumplir con todas las leyes de protección de datos, tanto las españolas como las europeas (LOPD y RGPD).
Justificación	La plataforma ha de ser desarrollada teniendo en cuenta las leyes españolas y europeas, ya que sino puede haber consecuencias de carácter penal y criminal.
Condición de satisfacción	Se dará por alcanzado este requisito si todos los datos tratados son transferidos y guardados de manera segura, siguiendo la normativa vigente.

Tabla 44: Requisito no funcional 11 - Legislación. Fuente: Elaboración propia.

12. Arquitectura y Diseño

12.1 Arquitectura

Una vez analizado el sistema *marketplace* y sus requisitos, la arquitectura que se ha decidido implementar ha sido una arquitectura basada en capas. Esta es un tipo de arquitectura muy utilizada en sistemas de información, ya que estos sistemas son muy cambiantes y de esta manera se consigue un gran nivel de flexibilidad, para poder adaptarse a los cambios, así como un gran nivel de reusabilidad y mantenimiento. Esta arquitectura, ya validada en este tipo de sistemas, tiene como objetivo principal la separación (desacoplamiento) de las partes principales que forman un sistema software.

Figura 7: Arquitectura en capas. Fuente: Elaboración propia.

Gracias a esta separación, no solo se pueden realizar cambios en una capa sin que afecte a las otras, sino que distintos grupos de personas pueden trabajar simultáneamente en distintas capas sin problema alguno. Esta característica es primordial cuando se trabaja en proyectos grandes, y a pesar de que no es el caso en este momento debido a que solo hay un desarrollador en este trabajo, una buena organización inicial hará posible el crecimiento de la plataforma. Otra de sus muchas ventajas es que, al tener una gran separación y desacoplamiento entre componentes, hace mucho más sencilla la reutilización de estos por el sistema y su testeo. Las capas en las que se ha dividido este sistema son:

Capa de presentación: Esta capa tiene dos grandes funcionalidades. Por un lado, es la encargada de presentarle al usuario todos los elementos gráficos para que pueda interactuar con el sistema, así como también ha de mostrarle el resultado de las interacciones y los datos requeridos por el usuario. Esta sería la funcionalidad más visual. Por otro lado, esta capa también es la encargada de no solo mostrar elementos gráficos, sino de capturar las interacciones que hace el usuario con ellos para poder trasladarlas a la capa de dominio. Aquí se utiliza orientación a objetos.

Capa de dominio (lógica de negocio): Esta capa es la responsable de la implementación de todas las funcionalidades del sistema. Todos aquellos procesos, reglas, validaciones, cálculos, etc., que el sistema ha de hacer para su correcto funcionamiento residen en esta capa. Además, es la capa encargada de conectar tanto con la capa de presentación, para enviarle los datos que hay que presentar y para recibir las acciones que realiza el usuario, como con la capa de datos, para poder obtener o introducir los datos necesarios. Aquí se utiliza orientación a objetos.

Capa de datos: La capa de datos es la encargada de interactuar y proporcionar acceso a los datos. Esta capa recibe solicitudes de datos, provenientes de la capa de lógica, y responde a estas solicitudes con los datos requeridos gracias a que sabe cómo comunicarse con el sistema gestor del almacenamiento de datos.

12.2 Diseño

Explicada la arquitectura, seguidamente se va a detallar el diseño de cada una de las capas explicadas anteriormente, así como el diseño de las conexiones entre ellas.

12.2.1 Diseño de la capa de presentación

La capa de presentación es la encargada tanto de interactuar con el usuario como con la capa de dominio, es por eso que su diseño puede dividirse en estas dos grandes categorías:

12.2.1.1 Diseño externo

El diseño externo consiste en la definición de los elementos visuales con los que el usuario podrá interactuar con el sistema. El uso de mapas navegacionales ayuda a llevar a cabo el diseño externo, ya que nos permiten definir el flujo de acción entre los distintos elementos visuales a utilizar.

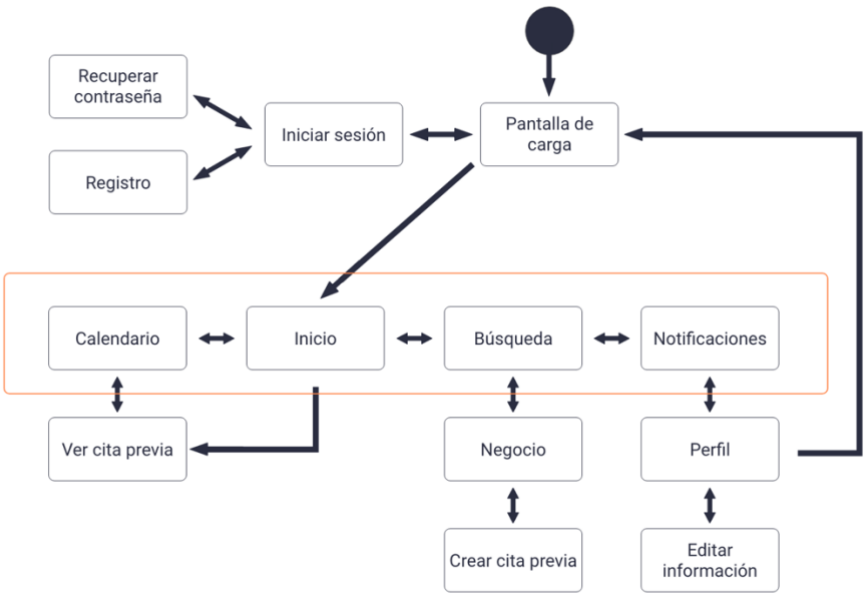


Figura 8: Mapa navegacional del sistema. Fuente: Elaboración propia.

Las interfaces que podemos ver en este mapa navegacional son independientes de la plataforma en la que se implementen (aplicación móvil, web, etc.). Además, las interfaces que se encuentran dentro del cuadrado naranja permiten libre navegación entre ellas.

12.2.1.2 Diseño interno

Dentro de las interfaces definidas en el mapa navegacional, habrá elementos que permitirán al usuario interaccionar con el sistema. Estas interacciones que son sencillas y la capa de presentación puede hacer sola (como cambiar de pantalla), sin embargo, habrá otras que serán más complejas (como la solicitud o presentación de datos).

Es ahí donde entra el concepto de diseño interno, encargado de toda esta gestión de la interacción entre usuario y sistema, pudiendo ser resumida en 4 etapas:

- Recepción de la interacción
- Gestión de la interacción
- Comunicación de la interacción a la capa de dominio
- Presentación de la información proveniente de la capa de dominio

La recepción de la interacción se hará mediante los elementos visuales del diseño externo ya explicado. Estos elementos serán modelados como objetos y podrán capturar eventos realizados sobre ellos. Sin embargo, una vez se ha recibido la interacción, es ahí donde empieza la parte más compleja del diseño de la capa de presentación.

Cuando una interacción es recibida, puede que necesite ser tratada de distintas formas dependiendo de dónde proviene esta interacción (desde que interfaz proviene). Pasa lo mismo con el resultado de esta interacción, ya que si se reciben unos datos desde la capa de dominio deberán ser presentados de una forma concreta en función de la interfaz que tenga que presentarlos. De ahí la necesidad de tener una lógica de presentación, que gestione las peticiones que reciben los elementos visuales y se encargue de adaptar los datos resultantes para que puedan ser presentados de forma correcta.

Con el fin de que la capa de presentación esté correctamente modularizada, haciendo que sea más fácil su modificación y reutilización, cumpliendo así con los principios y buenas prácticas del diseño software, se ha decidido utilizar el patrón Model-View-ViewModel (MVVM).

Este patrón nos permite desacoplar la presentación de la información, la lógica de presentación y la lógica de negocio (dominio), de tal forma que resulta mucho más sencillo modificar o reutilizar vistas o lógicas de presentación. El patrón MVVM es un patrón arquitectónico que se basa en 3 componentes: la vista, el modelo y el modelo de vista.

- **Modelo:** El modelo representa la capa de dominio y la capa de datos, devolviendo los datos que son solicitados por la lógica de presentación.

- **Modelo de Vista (ViewModel):** Este es un actor intermediario entre el modelo y la vista, encargado de gestionar las interacciones que se reciben desde la vista, solicitando al modelo los datos necesarios y encargado de manipular estos datos para que puedan ser presentados por las vistas (lógica de presentación).

- **Vista:** La vista es la encargada de representar la información que proviene del modelo de vista. Además, también contiene los elementos especificados en el diseño externo que reciben las interacciones que han de ser pasadas al modelo de vista.

Figura 9: Esquema del patrón MVVM. Fuente: Elaboración propia.

La razón más importante por la cual se ha decidido utilizar este patrón de entre otros similares (como podrían ser MVC o MVP) es el particular enlace de la vista y su lógica de presentación. Este enlace es reactivo, es decir, no se controlan manualmente los cambios, sino que la vista siempre está observando los datos del modelo de vista, y si estos cambian automáticamente cambia la vista. De esta manera, se consigue un desacoplamiento por parte del modelo de vista hacia la vista, ya que el modelo de vista no es consciente de qué vistas dependen de él.

En conclusión, la capa de presentación contiene la vista y el modelo de vista, ya que el modelo solo es una representación de la capa de dominio y la de presentación. Desde el punto de vista del diseño externo, los patrones utilizados están relacionados con los principios del diseño de interfaces para usuarios, mientras que, desde el punto de vista del diseño interno, se usa el patrón MVVM y los principios de la arquitectura orientada a objetos.

12.2.2 Diseño de la capa de dominio

La capa de dominio es la capa intermedia entre la de presentación y la de datos. Esta capa contiene toda la lógica de negocio, es decir, es la encargada de implementar todas las funcionalidades y procesos que ha de realizar el sistema. Cuando recibe una petición que proviene de la capa de presentación, la capa de dominio solicita los datos necesarios a la capa de datos, los procesa, realizando todas las acciones necesarias sobre ellos, y una vez completado este proceso, los envía a la capa de presentación para que estos sean mostrados al usuario.

La capa de dominio puede usar distintos patrones con el fin de implementar la lógica de negocio, siendo los dos más populares el patrón modelo *Domain Model* y el *Transaction Script*. Para este proyecto se ha decidido utilizar el patrón *Domain Model,* ya que hace uso del paradigma orientado a objetos para trabajar sobre diagrama de clases que representan el sistema software. El diagrama de clases especifica atributos y relaciones entre objetos, y gracias a la colaboración de las instancias de estos objetos, pueden ser implementadas todas las funcionalidades y procesos del sistema.

Teniendo en cuenta los requisitos de este sistema, se ha especificado el siguiente diagrama de clases con las mismas restricciones que el modelo conceptual:

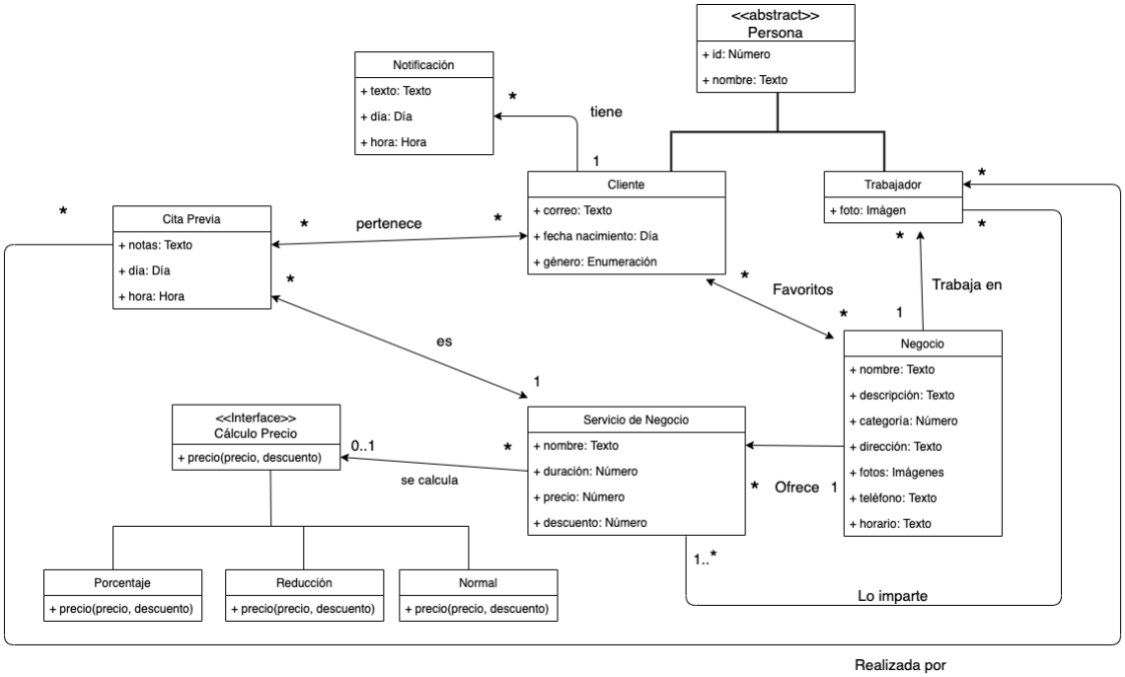


Figura 10: Diagrama de clases. Fuente: Elaboración propia.

12.2.2.3 Conexión con la capa de datos

Otro aspecto a tener en cuenta en la capa de dominio, es la manera en la que se enlazará con la capa de datos. Estas dos capas han de estar lo más desacopladas posible, haciendo que las modificaciones realizadas en una de ellas no impliquen modificaciones en la otra. Para ello, se ha decidido utilizar el patrón *Repository.* Este patrón consiste en la creación de una interfaz que se sitúa entre la capa de dominio y la de datos, llamada repositorio, que traduce las peticiones que hace la lógica en peticiones que entiende la capa de datos. De esta forma, si en algún momento se cambia la implementación de la capa de datos, simplemente hay que cambiar la interfaz repositorio, sin que la lógica de negocio se vea afectada.

Por otro lado, los objetos presentes en el modelo no tienen por qué estructurar su información de igual manera que la capa de datos. Además, algunas propiedades, como el polimorfismo, no pueden ser expresados en las bases de datos. Sin embargo, es necesario poder obtener la información de la base de datos y convertirla en objetos para tratar con ella, al igual que es necesario poder almacenar la información de los objetos. Es por eso que se va a utilizar también un patrón llamado Data Mapper, que consiste en una capa software que permite la transformación de los objetos a elementos de la base de datos y a la inversa. De esta manera, el esquema de la base de datos no conoce los objetos que se usan y el modelo tampoco conoce el esquema de la base de datos, creando un menor acoplamiento entre capas.

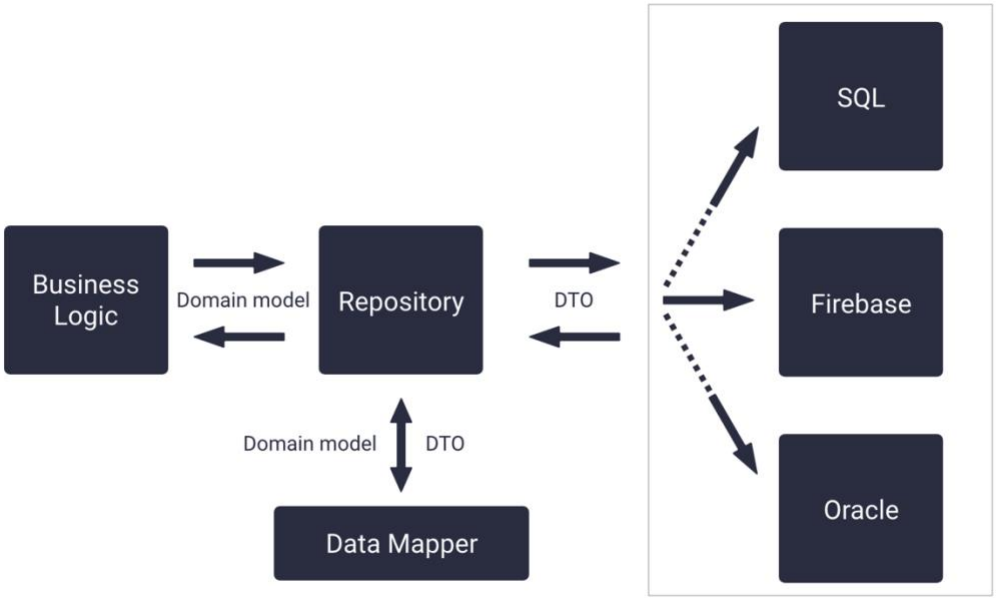


Figura 11: Patrón repositorio y DataMapper. Fuente: Elaboración propia.

12.2.3 Diseño de la capa de datos

El diseño de esta capa, a diferencia de las anteriores, va a ser un diseño basado en servicios. La gran ventaja de este formato es aprovechar el potencial que ofrecen servicios desarrollados por terceros, ya sea en términos de seguridad, escalabilidad o rendimiento.

Para este proyecto se van a utilizar 2 servicios externos: uno para la base de datos y otro para el almacenamiento de archivos multimedia. Para poder trabajar con estos dos servicios externos se utilizará un framework (explicado en el capítulo de implementación) encargado de hacer los accesos necesarios a estos servicios. De esta forma, el framework utilizado recibirá las peticiones que provengan de la capa de dominio y hará los accesos correspondientes a los dos servicios para obtener los datos que se están pidiendo desde el dominio y así poder enviárselos.

12.2.3.1 Servicio de base de datos

El primer servicio del que se hablará es el servicio de base de datos, encargado del almacenamiento y persistencia de estos. Este servicio permite diseñar una base de datos para poder almacenar los datos de la forma deseada. Sin embargo, la tecnología que use este servicio será determinante para la estructura de la base de datos, es por eso que primero hay que decidir qué tipo de estructura se quiere que permita el servicio antes de especificar el diseño de la base de datos.

12.2.3.1.1 Estructura no relacional

Principalmente existen dos tipos de estructuras para el almacenamiento de datos: relacional y no relacional. Para la elección de estructura se han tenido muy en cuenta los requisitos no funcionales del proyecto, principalmente los requisitos de velocidad y latencia, escalabilidad y usabilidad.

La estructura que más se adecua a este tipo de sistema y a estos requisitos es la no relacional, de ahí que se busque un servicio que permita la creación de bases de datos no relacionales. Esta estructura, muy utilizada en sistemas en tiempo real, aplicaciones móviles y sobretodo redes sociales, prioriza la rapidez y la velocidad de respuesta a la hora de obtener los datos, haciendo esperar el menor tiempo posible al usuario y mejorando la usabilidad del sistema. Para ello, los sistemas no relacionales tienen mucha redundancia de datos, y se hace mucho más complejo la gestión y actualización de los datos almacenados. Sin embargo, gracias a esta redundancia, no hace falta hacer un gran número de consultas para obtener los datos necesarios, incrementando la rapidez del sistema. Además, es fácilmente escalable y no baja su rendimiento.

Existen distintos tipos de bases de datos no relacionales (NoSQL), y su elección depende mucho del programador, ya que sus características son muy similares y solo difiere la forma en que se organizan los datos. Debido a los conocimientos que tiene el autor sobre este tipo de bases de datos, se ha elegido una no relacional basada en documentos, el tipo más popular de base de datos *NoSQL*. Este tipo de bases de datos se basa en colecciones y documentos. Los documentos son los encargados de almacenar información de distinto tipo, como por ejemplo texto, listas, diccionarios, números, puntos geográficos, etc. Estos documentos suelen ser de tipo JSON, ya que es un modelo de datos eficiente e intuitivo para los desarrolladores. Por otro lado, están las colecciones, que permiten agrupar una cierta cantidad de documentos dentro de ellas. Cabe destacar que cada documento tiene un identificador único dentro de su colección y que aparte de la información se pueden tener también sub-colecciones. Las sub-colecciones son colecciones localizadas dentro de documentos, dando más flexibilidad a la base de datos. Se pueden anidar tantas colecciones y documentos como se quiera, pero nunca se puede meter una colección dentro de otra, ni un documento dentro de otro, siempre ha de ser colección, documento, colección, documento, etc.

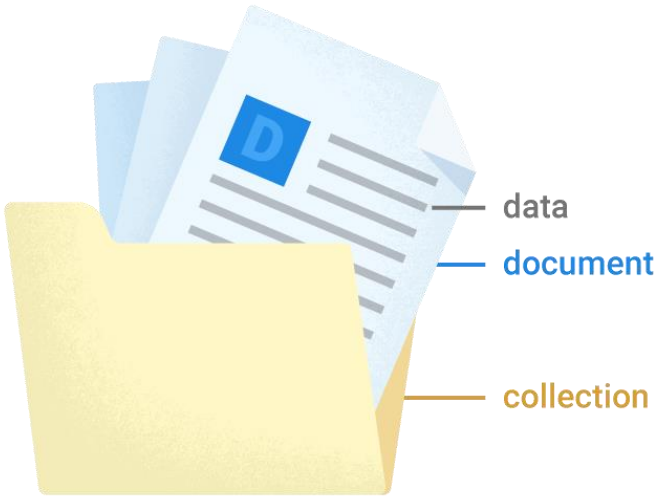


Figura 12: Estructura de una base de datos no relacional. Fuente: Firebase Cloud Storage.

12.2.3.1.2 Diseño de la base de datos

Visto el formato que ha de seguir una base de datos no relacional basada en documentos, a continuación se presenta el diseño de la base de datos de este proyecto partiendo del modelo conceptual mostrado anteriormente:

1) **Nombre de la colección:** Businesses
   Atributos de los documentos de esta colección**:**
   - Address
   - Category
   - City
   - Cover Photo
   - Description
   - Latlong
   - Name
   - NumServices
   - Phone
   - Photos
   - Schedule

   1.1) **Nombre de la subcolección:** Appointments
   Atributos de los documentos de esta colección:
   - Date
   - Duration
   - ServiceId
   - UserId
   - WorkerId

   1.2) **Nombre de la subcolección:** Services
   Atributos de los documentos de esta colección:
   - Currency
   - Duration
   - Name
   - Price
   - Discount
   - PromotionType
   - PromotionText
   - WorkersId

   1.3) **Nombre de la subcolección:** Workers
   Atributos de los documentos de esta colección:
   - Name
   - Photo
   - Services

2) **Nombre de la colección:** Users
   Atributos de los documentos de esta colección:
   - Birthday
   - Email
   - Gender
   - Name

2.1) **Nombre de la subcolección:** Appointments
Atributos de los documentos de esta colección:

- Address
- BusinessId
- Category
- Date
- Duration
- IsManual
- Name
- Notes
- ServiceName
- WorkerName

2.2) **Nombre de la subcolección:** Favorite Businesses,
Atributos de los documentos de esta colección:

- Address
- Category
- CoverPhoto
- Name

2.3) **Nombre de la subcolección:** Notifications
Atributos de los documentos de esta colección:

- Date
- Text

3) **Nombre de la colección:** Cities
Atributos de los documentos de esta colección:

- Name
- Postal code

4) **Nombre de la colección:** Promotions
Atributos de los documentos de esta colección:

- PromotionName
- PromotionType
- Discount
- ServiceName
- ServiceId
- BusinessName
- BusinessId
- BusinessPhoto

La colección de usuarios contiene un documento por cada usuario registrado en el sistema, y este contiene toda la información personal del usuario. Este documento es creado en el momento del registro, cuando el usuario da de alta una cuenta o se autentica por primera vez. Además de la información personal, cada documento de usuario contiene dos sub-colecciones: negocios favoritos y reservas. Estas dos sub-colecciones almacenan en documentos tanto la información de los negocios favoritos del usuario como la de sus reservas.

Por otro lado, tenemos la colección de negocios, donde cada documento guarda la información relativa a cada uno de ellos. Además, cada negocio tiene tres sub-colecciones, las reservas que se han hecho, los servicios que se dan y los trabajadores que tienen. Esta configuración de base de datos nos permite que la mayoría de funcionalidades de la app solo tengan que hacer una consulta para obtener la información que hay que mostrarle al usuario. Hay que tener en cuenta un factor que suele ocurrir mucho en las bases de datos no relacionales, que es la repetición de la información. Por poner un ejemplo, la información que hay en un documento dentro de la sub-colección de negocios favoritos de un usuario es prácticamente igual a la que contiene el documento de ese negocio en la colección de negocios. Esto no es un fallo ni un error, sino que está hecho para poder obtener esta información de manera más rápida. Si en vez de duplicar la información solo se guardara una referencia (como pasa en las bases de datos relacionales) cada vez que un usuario quisiera ver sus negocios favoritos habría que hacer dos consultas, una para obtener las referencias de esos negocios y otra para obtener su información. De este modo solo una consulta es necesaria para completar esta funcionalidad (simplemente hay que obtener los documentos de la sub-colección “negocios favoritos” dentro de un documento de usuario). Por el contrario, este hecho hace que se tenga que poner mucho cuidado a la hora de actualizar la información, ya que va a tener que ser actualizada en distintos sitios para no tener incongruencias en la base de datos. Lo mismo pasa con las reservas, ya que una reserva hecha por un usuario en un negocio está tanto dentro de la sub-colección “reservas” de ese negocio como dentro de la sub-colección “reservas” del usuario. Se podría haber decidido crear una colección principal llamada reservas que albergara todas las reservas hechas en la aplicación y tanto los usuarios como los negocios tener referencias a estas reservas, sin embargo, volvería a pasar lo mismo, para obtener las reservas de un usuario o negocio concreto se deberían hacer 2 consultas en vez de una.

También está la colección de ciudades, encargada de almacenar el nombre y el código postal de las ciudades dónde están los negocios del sistema. Si no se tuviera esta colección, cada vez que se quisiera filtrar por ciudad habría que examinar todos los documentos de los negocios en busca de las diferentes ciudades, de ahí su utilidad.

Como última colección principal, está la de promociones, encargada de almacenar en documentos la información de la promoción hecha por cada servicio de negocio. Esta información vuelve a estar repetida, ya que dentro de cada servicio ya está la información de su promoción. Sin embargo, para no tener que estar haciendo muchas consultas cada vez que se quieren mostrar las promociones existentes, cuando una promoción es creada se crea en esta colección y también en el documento del servicio en cuestión.

12.2.3.1 Servicio de almacenamiento multimedia

Al escoger una base de datos no relacional en formato de documentos JSON, el almacenamiento de archivos como imágenes no es posible hacerlo en la misma base de datos, ya que no son soportados en formato JSON. Es por eso que para las imágenes de los negocios y de los trabajadores es necesario utilizar otro servicio externo. Este servicio proporcionará un almacenamiento de imágenes en la nube y con una estructura basada en clave-valor, es decir, cada imagen tendrá por clave el identificador del negocio o trabajador al que pertenece. Al tenerlas almacenadas en este servicio, a cada imagen se le otorga una URL de acceso para que pueda ser visualizada.

Sin embargo, cada vez que se quisiera una imagen el servicio gestor de la capa de datos tendría que hacer dos accesos, uno sal servicio de base de datos para coger los datos del negocio o trabajador y otro al servicio de almacenamiento multimedia para coger las URLs de sus respectivas imágenes. Para reducir el número de acceso, se ha decidido guardar la URL de acceso de cada imagen en la base de datos no relacional en formato JSON, junto con la otra información relacionada con los negocios y trabajadores. De esta manera, al hacer una consulta a la base de datos no relacional y obtener la información de un negocio o trabajador, esta información ya incluye las URLs para visualizar las imágenes relacionadas con lo que se está obteniendo, ya sea un negocio o un trabajador (las únicas dos entidades que tienen imágenes).

12.2.4 Diagrama de secuencia

Una vez visto el diseño de las tres capas de este sistema, a continuación se presenta el diagrama de secuencia del proceso de creación de una cita previa en un negocio por parte de un cliente:

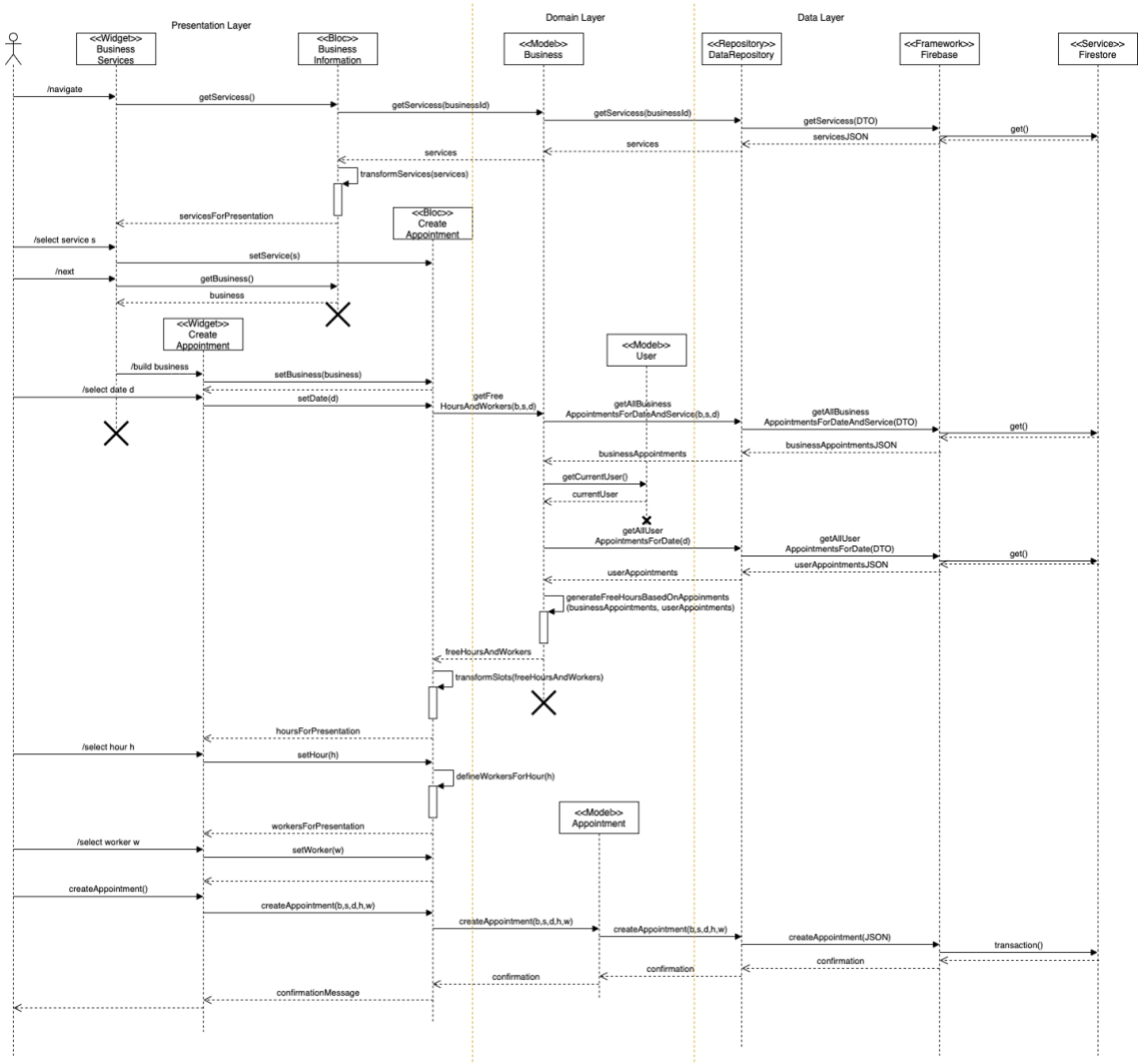


Figura 13: Diagrama de secuencia de la funcionalidad de cita previa. Fuente: Elaboración propia.

Se puede ver que hay dos funciones que pertenecen a los blocs utilizados que no se han especificado. Estas dos funciones son "*transformServices()"* y "*transformSlots()",* encargadas de transformar los datos que provienen de los modelos en datos que los widgets sepan presentar al usuario. Otra de las funciones más importantes es generateFreeHoursBasedOnAppointments(). Esta función que toma como parámetros las citas previas de un negocio para una fecha y un servicio y las citas previas del usuario actual es la encargada de generar las horas libres que ese negocio y ese usuario tienen para esa fecha, con tal de que el usuario no pueda coger una hora que se solape con alguna cita ya sea suya o del negocio.

En la capa de datos tenemos el framework (Firebase) y el servicio de base de datos (Firestore). El primero es el encargado de hacerle peticiones al segundo para coger los datos necesarios. La última acción hecha por el framework es "transaction()", esta acción es muy importante debido a su diferencia con un "create()". Mientras que en el segundo simplemente se están añadiendo datos a la base de datos, el segundo es una serie de acciones que han de ejecutarse como un solo bloque sin interrupciones. Esto ha de hacerse así porque puede pasar cierto tiempo entre que se piden los datos a la base de datos y se añade la cita previa a esta, pudiendo ocurrir que entre medio otro usuario decida crear una cita previa que se solape con la que se está creando. Utilizando una transacción, antes de crear definitivamente la cita previa se vuelve a comprobar que es posible crearla y en caso afirmativo se crea, todo sin interrupciones.

13. Implementación

En el capítulo 3, el alcance del proyecto, ya se mencionó que el sistema a desarrollar se implementaría en formato app para dispositivos móviles. En este capítulo se discutirán las diferentes tecnologías para ello, así como la implementación de cada capa.

13.1 Tecnologías

El desarrollo de una app móvil puede hacerse de dos formas distintas: nativamente o multiplataforma. Con el desarrollo nativo únicamente se puede producir una app nativa, en cambio si se hace un desarrollo multiplataforma o bien se puede producir una app multiplataforma, o distintas apps nativas para cada una de las distintas plataformas.

Las aplicaciones nativas son aquellas que pueden ser instaladas únicamente en un sistema operativo concreto (ya que su código es código nativo, hecho específicamente para la plataforma en la que va a funcionar). De entre sus ventajas podemos destacar que, al ser específicas para un sistema operativo en concreto, permiten aprovechar mucho mejor sus características y funcionalidades. Suelen ser más eficientes y tienen un acceso más profundo en el sistema, tanto a nivel de componentes como a nivel de control de este. Por otro lado, las aplicaciones multiplataforma son aquellas que pueden ser instaladas y usadas en más de un sistema operativo. La ventaja de estas aplicaciones es que solo hay que mantener un código, ya que este sirve para todas las plataformas. Sin embargo, la característica de multiplataforma hace que estas apps tengan que ser más generales y no puedan profundizar tanto en los sistemas operativos, y por lo tanto no tener tanto control sobre ellos. Pueden acceder a las funcionalidades y componentes generales de los dispositivos (como la cámara, por ejemplo), pero no a características más específicas, como podrían ser funcionalidades de *machine learning* o un componente de red neuronal presente en algún dispositivo concreto. Sin embargo, hoy en día existen métodos de desarrollo multiplataforma que su resultado son distintas apps nativas, teniendo las ventajas de solo mantener un único código y al exportarlo poderlo convertir en código nativo para los sistemas operativos a los que se quiere llegar. Este tipo de desarrollo multiplataforma, a pesar de que al final se exportan apps en código nativo, no ofrece el grado de personalización que podemos obtener con el desarrollo nativo.

La herramienta que se quiere desarrollar pretende llegar al máximo número de personas, siendo este un gran condicionante para su desarrollo. Actualmente, los dos grandes sistemas operativos que dominan el mercado de dispositivos móviles son Android y iOS, cubriendo un 99,8% de mercado en España [21]. Si se quisiera hacer un desarrollo nativo, sería obligatorio desarrollar (y mantener) dos códigos distintos, uno para cada plataforma que derivaría en distintas aplicaciones. Sin embargo, con un desarrollo multiplataforma solo se necesitaría desarrollar un mismo código para las dos versiones de la app, haciendo mucho más económico en términos de tiempo (y de dinero) este desarrollo. Es verdad que, como hemos mencionado antes, esta forma de hacerlo tiene sus inconvenientes. Sin embargo, los requisitos de la aplicación permiten que la aplicación pueda ser genérica en términos de desarrollo, ya que no se necesita acceso a componentes específicos o funcionalidades muy concretas. Esto, junto con el tiempo que se tiene disponible, justifica la elección de una aplicación multiplataforma.

Frameworks multiplataforma

Actualmente las aplicaciones multiplataforma son desarrolladas con la ayuda de *frameworks*. Un *framework*, o entorno de trabajo, es un conjunto de conceptos, prácticas y criterios estandarizados a seguir. Los *frameworks* también nos proporcionan una serie de funciones ya desarrolladas, que se encuentran a nuestra disposición para usarse en los proyectos que realicemos. Suelen ser funciones comunes a la mayoría de proyectos, como por ejemplo la funcionalidad de *login* con correo y contraseña, presente en muchas webs y apps. Hoy en día los *frameworks* son muy utilizados debido a la complejidad que supondría la realización de un proyecto (como podría ser una web o una app) dónde hay que implementar todas las funciones de forma manual. Es por esto que la aplicación que se va a desarrollar se hará utilizando un *framework* que permita el desarrollo multiplataforma. A día de hoy los *frameworks* más populares con este propósito son los siguientes tres:

- **React native [22]:** Este es un *framework* creado por Facebook que se utiliza para desarrollar aplicaciones para Android y iOS. Está basado en la librería de JavaScript React, que es usada para el desarrollo Web. Esto hace que sea uno de los *frameworks* más utilizados, ya que no es nuevo para los desarrolladores web al basarse en JavaScript. Gracias a esta librería, el *framework* convierte los componentes desarrollados en componentes nativos para las plataformas de Android y iOS, haciendo que las aplicaciones resultantes estén en lenguaje nativo propio de la plataforma destino.

- **Ionic [23]:** Este framework fue desarrollado por Max Lynch, Ben Sperry y Adam Bradley en 2013. A diferencia de React Native, Ionic fue construido sobre Angular. Además, el funcionamiento de las aplicaciones creadas con este *framework* es distinto al de React, ya que no son transformadas en aplicaciones nativas, sino que las aplicaciones resultantes son aplicaciones web que se ejecutan como una aplicación de móvil.

- **Flutter [24]:** Flutter es el *framework* más nuevo de todos. Fue creado por google en 2015 y poco a poco ha ido ganando más popularidad. Siendo similar a React Native, las aplicaciones hechas con Flutter son convertidas a aplicaciones nativas cuando se quieren exportar a otras plataformas.

A pesar de que los 3 *frameworks* expuestos podrían servir perfectamente para hacer la aplicación, hay algunas características que los diferencian. Primero de todo, podemos distinguir entre dos grupos, los que sus aplicaciones resultantes funcionan en código nativo de las plataformas destino (como en el caso de React Native y Flutter) y los que sus aplicaciones resultantes son de tipo web ejecutadas en móvil (como Ionic). La ventaja de los primeros es que hacen que sus aplicaciones se integren mejor con la plataforma donde son ejecutadas, dando mejores resultados en términos de eficiencia y velocidad. También es más fácil para ellas acceder a propiedades de los sistemas operativos de Android y iOS.

Entre React Native y Flutter podemos decir que los dos tienen una gran compañía detrás que les da soporte, siendo eso un factor importante en términos de continuidad. Es verdad que React Native es más popular y usado que Flutter, mayormente por su antigüedad; sin embargo, Google está apostando cada vez más por Flutter haciendo que poco a poco se vuelva más popular.

Al funcionar de forma similar y tener características muy parecidas hace que la decisión entre estos dos *frameworks* sea tomada en función de las preferencias del autor, que decide utilizar Flutter, ya que cree tiene mucho potencial y es una buena oportunidad para iniciarse en su aprendizaje. Flutter es un *framework* muy centrado en el diseño e interfaz, pudiendo conseguir muy buenos diseños y buena experiencia de usuario, siendo este un requisito importante en este proyecto. Otra de las grandes ventajas de este *framework* es que se integra muy bien con sistemas de datos como Firebase, hecho también por Google. Gracias a esta integración se pueden desarrollar aplicaciones que reaccionen a eventos de la base de datos (programación reactiva), consiguiendo una experiencia mucho más completa.

13.2 Implementación de la capa de presentación

Flutter proporciona un kit de herramientas de interfaz de usuario poder construir de una forma más fácil y rápida las interfaces de una aplicación. Este framework considera a todo elemento gráfico como un *widget*, y las interfaces se construyen anidando *widgets* unos dentro de otros y modificando sus propiedades.

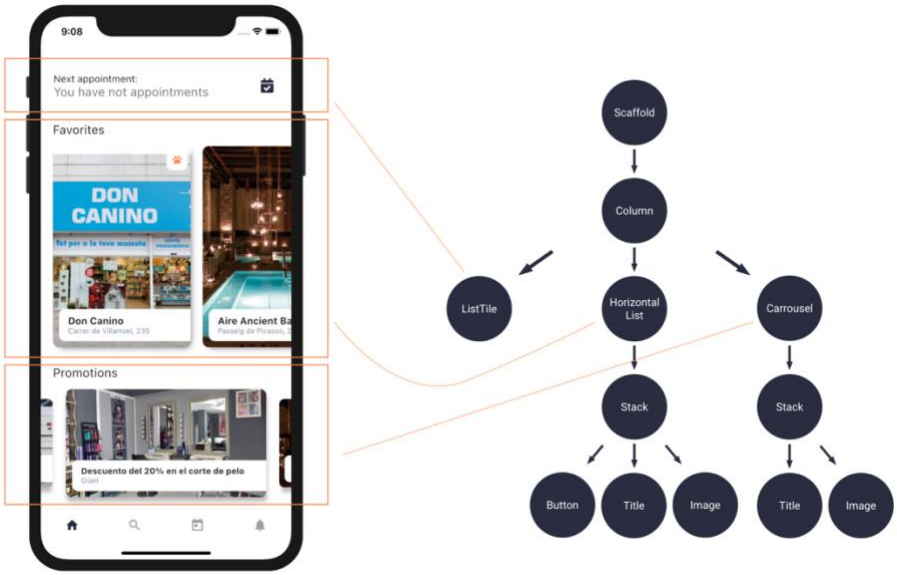


Figura 14: Comparativa entre interfaz (izquierda) y su "widget tree" (derecha). Fuente: Elaboración propia.

Como se puede observar en la imagen superior, a la izquierda está una interfaz de la aplicación y a la derecha su estructura en *widgets*. Se puede ver que la anidación de *widgets*, con sus distintas propiedades, acaban creando toda la interfaz. Estos *widgets* proporcionados por Flutter tienen la particularidad de que no modifican su aspecto independientemente de la plataforma donde se vaya a ejecutar la aplicación (Android o iOS), haciendo que se comparta un mismo aspecto entre plataformas. Existen una gran cantidad de *widgets* dentro de este framework, sin embargo, al ser de código abierto todo el mundo puede ser capaz de crear sus propios *widgets*. Existen sitios web donde se pueden encontrar *widgets* creados por distintas personas que pueden ser usados por cualquiera, únicamente importándolos como librería al proyecto.

Los *widgets* se utilizan para crear las interfaces de la aplicación, sin embargo, también incorporan cierta lógica en su interior, pueden reaccionar a eventos al interactuar con ellos y pueden mantener un estado. El caso más simple es un *widget* de botón, que no solo permite introducir un botón en la interfaz de la manera y las características deseadas, sino que además también incorpora eventos y escuchas para poder reaccionar cuando ha sido pulsado. Además, también puede guardar un estado, pudiendo así cambiar sus características en función de este estado.

Sin embargo, para tener una mejor separación entre las interfaces, lógica de presentación de estas interfaces y la lógica de negocio que pertenece a la capa de dominio, se va a implementar el patrón BLoC. Como ya se ha comentado en el apartado de arquitectura y diseño, se ha dicho que se usaría el patrón MVVM para conectar tanto la capa de presentación como la de dominio. BLoC (Business Logic of Component) es una implementación de MVVM específica para Flutter, ya que aprovecha ciertas características de este framework. La implementación sería la siguiente:

Figura 15: Implementación del patrón BLoC. Fuente: Elaboración propia.

Podemos ver que entre la UI y el modelo tenemos una capa intermedia llamada BLoC. Esta capa está formada por distintos componentes (llamados BLoC), cada uno de ellos encargado de recibir unos datos concretos y gestionar su lógica de presentación. Cada BLoC está encargado de la lógica de presentación de un apartado o funcionalidades concretas de la app, siendo el gestor de los datos relacionados con este apartado y encargado de transformarlos para que las interfaces puedan presentarlos. Además, también tiene todas las funciones necesarias para interactuar con el tipo de datos que el BLoC gestiona.

Estos componentes tienen la particularidad de estar vinculada de una forma especial con las interfaces, ya que el envío de datos no es mediante peticiones, sino que las vistas siempre están observando a sus respectivos BLoC en busca de cambios. Este método, llamado *polling*, tiene la gran ventaja de que cada BLoC particular, es decir, el encargado de gestionar una lógica de presentación concreta, no sabe qué interfaces dependen de él. Esto es gracias a que son las interfaces las que son conscientes de qué BLoC están escuchando. Por lo tanto, el BLoC simplemente se encarga de recibir datos y crear los estados correspondientes, y una vez estos datos y estados sean modificados, todas las interfaces que estén observando estos datos y estados cambiarán acorde a las nuevas modificaciones. Con este sistema se consigue que la programación sea reactiva y a tiempo real, ya que no hay que estar constantemente haciendo peticiones de datos desde las interfaces hacia los BLoC. Siguiendo con el ejemplo anterior y para que quede un poco más claro, cuando en la interfaz de login el usuario pulsa el botón iniciar sesión, esta interfaz llama a la función de iniciar sesión del BLoC encargado de la autenticación, enviándole los parámetros necesarios. La función del BLoC es la encargada de enviar a la lógica de negocio (Modelo) los datos recibidos para que se hagan los procesos necesarios de autenticación.

Una vez se obtiene una respuesta, el BLoC transforma estos datos en distintos estados, como podría ser el estado de usuario autenticado o no autenticado por error, y una vez creado este estado, las interfaces que están observando reaccionan según el resultado. El BLoC es exactamente lo mismo que el ModelView dentro del patrón MVVM. Cabe aclarar que el BLoC no realiza ninguna tarea de lógica de negocio, sino que simplemente es el encargado de transformar los datos que se le pasan desde la capa de dominio en un formato que sea entendible para las interfaces de usuario.

13.3 Implementación de la capa de dominio

Flutter usa Dart como lenguaje de programación, un lenguaje creado por Google que, a pesar de que tiene más usos, Flutter lo ha convertido en su principal.

Dart fue diseñado con el objetivo de agilizar y hacer más cómodo y sencillo el desarrollo de aplicaciones en general. Consta de un extenso conjunto de herramientas para ello, como su propio gestor de paquetes, así como un analizador y compilador. Además, su función de Just-in-Time permite visualizar al instante todos los cambios que se van realizando en los programas. El código generado en Dart es compilado en el lenguaje nativo de la plataforma en la que se quiera ejecutar, como podría ser Android o iOS.
De esta forma, con un mismo código se obtienen 2 aplicaciones compiladas en código nativo. Posteriormente, se amplió para que también soportara el desarrollo web, transpilando su código a JavaScript. Dart tiene una leve barrera de entrada, ya que es muy similar a lenguajes como Java, JavaScript y C++, por lo que si se conoce alguno de estos lenguajes no es muy difícil iniciarse en Dart.

Toda la capa de dominio ha sido programada en Dart, de tal forma que pueda complementarse de la mejor forma con Flutter, ya que este framework también usa Dart para la declaración de los *widgets*. Como ya se ha explicado anteriormente, la capa de dominio se conecta con la de presentación a través de los BLoC, y a la capa de datos a través del repositorio.

13.4 Implementación de la capa de datos

En el capítulo de arquitectura y diseño ya se ha explicado que esta capa funcionará con 2 servicios de terceros. Para este proyecto, se ha tomado la decisión de usar los servicios de Firebase, una plataforma creada por Google en el 2014. Ubicada en la nube, esta plataforma proporciona servicios tales como bases de datos, posibilidad de ejecución de código, servicio de autenticación, integración con otros servicios de Google, etc. La primera ventaja es que esta plataforma proporciona todos los distintos servicios que el proyecto necesita, haciendo más fácil su conexión entre estos distintos servicios. Además, al tener detrás una gran empresa como Google, nos asegura una alta continuidad y calidad de los servicios. Los servicios de Firebase tienen una muy fácil integración tanto con frameworks para aplicaciones móvil (Flutter) como con frameworks para web y servidores, así como una documentación extensa para su integración.

Para el servicio de base de datos se usará Cloud Firestore, un servicio de Firebase que permite la creación de bases de datos no relacionales basadas en documentos y colecciones. Este servicio es muy flexible y escalable, y mantiene los datos sincronizados entre apps clientes a través de agentes de escucha en tiempo real. Además, este servicio permite definir reglas sobre las bases de datos creadas para limitar su acceso con el fin de incrementar la seguridad.

Para el servicio de almacenamiento de archivos multimedia se utilizará Cloud Storage, un servicio de Firebase de almacenamiento de objetos potente, simple y también con una gran escalabilidad. Este servicio permite la subida y la descarga de archivos, y de entre otras muchas ventajas está la de que se adapta mucho a la velocidad de conexión del cliente, permitiendo a este retomar la operación si en algún momento hay una caída de la conexión, ahorrando ancho de banda y tiempo a los usuarios.

Finalmente, se utilizará también Cloud Functions, un framework de Firebase con una arquitectura *serverless*, que permite ejecutar código de forma automática en respuesta a solicitudes HTTPS. Con este tipo de arquitectura no es necesario administrar ni escalar servidores, ya que de esto se ocupa Firebase. Gracias a que este framework es de la misma empresa que los otros servicios, la integración de estos tres componentes es total, pudiendo así comunicarse eficazmente y de forma segura tanto con Cloud Firestore como con Cloud Storage. Es por eso que se ha escogido este framework como encargado de hacer de puente entre la capa de dominio y los datos.

13.5 Ubicación de las capas

La distribución final de las capas lógicas en dispositivos físicos es la siguiente:

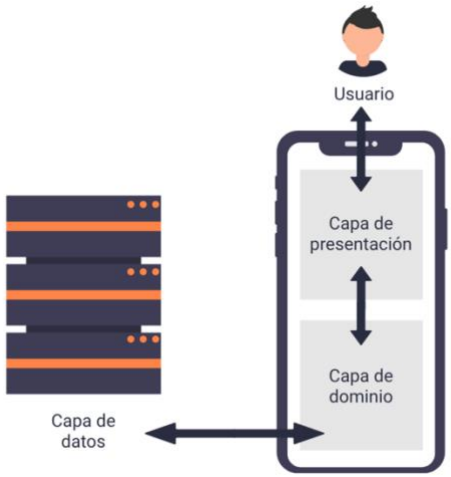


Figura 16: Ubicación de las capas. Fuente: Elaboración propia.

Como se puede observar en la figura 16, tanto la capa de presentación como la capa de dominio residen en la app móvil, mientras que la capa de datos reside en servidores externos.

La capa de datos ha de estar ubicada en un servidor externo debido a que todos los usuarios han de poder acceder a los datos, y si por el contrario estos estuvieran repartidos dentro de las apps móviles de los usuarios sería muy difícil y complejo gestionar todas las conexiones entre sí. Es por eso que se centralizan estos datos en un servidor externo para que desde la app móvil se pueda acceder a ellos. Además, la utilización de servicios de terceros en la capa de datos hace que estos servidores externos sean gestionados por las entidades propietarias de estos servicios, localizándolos donde ellos quieren dentro de la red.

Por otro lado, tanto la capa de datos como la capa de presentación residen en la app móvil dentro del dispositivo del usuario. Es lógico que la capa de presentación resida ahí, ya que es con la que interactúa el usuario, sin embargo, la capa de dominio podría estar ahí como podía localizarse en un servidor centralizado al que todos los usuarios se conectaran. Se ha decidido poner la capa de dominio también en la app móvil por dos razones en concreto: la primera es que al tener la capa de presentación y la de dominio juntas en el mismo dispositivo físico se consigue una mayor eficiencia que se traduce en una mejor experiencia de usuario al utilizar el sistema. La segunda es que, gracias a las tecnologías utilizadas, la comunicación entre la capa de dominio y la capa de datos, a pesar de que estén en dos dispositivos físicos diferentes, es muy rápida y está muy optimizada, pudiendo conseguir información en tiempo real sin necesidad de estar pidiendo información todo el tiempo a la capa de datos y sin que los usuarios tengan que estar refrescando la aplicación para obtener la información más actualizada. No todo son ventajas, ya que el hecho de tener la capa de dominio en la app móvil hace que la app ocupe más espacio en los dispositivos del usuario y también será necesario actualizar la app cada vez que se haga un cambio en la capa de dominio.

Gracias a esta configuración y las tecnologías utilizadas, la capa de dominio puede reaccionar a eventos que ocurran en la capa de datos sin necesidad de que el usuario desde la capa de presentación haya pedido esos datos, consiguiendo así un sistema en tiempo real que no está basado en peticiones, mejorando mucho la experiencia de usuario, velocidad y apariencia. Hay que mencionar también que nunca se rompe el patrón arquitectónico basado en 3 capas, la de presentación únicamente se conecta con la de dominio, la de dominio con presentación y datos y la capa de datos solo con la de dominio.

14. Testing

La realización de test es una parte muy importante en el desarrollo de software. Esta práctica puede ahorrar muchos problemas en un futuro que inicialmente no se había encontrado.

Durante el desarrollo de este proyecto se han ido realizando test cada vez que se acababa de desarrollar una funcionalidad. Estos test iniciales han sido realizados de forma manual, aunque en algunas funcionalidades concretas o partes del código más críticas ha sido posible realizar test automatizados. De esta forma, se podía comprobar que la funcionalidad acabada funcionaba como se esperaba y no había ningún error en ella. Este tipo de test han sido muy parecidos a lo que comúnmente se denomina test unitarios.

Sin embargo, no ha sido posible realizar grandes test hasta las fases finales del desarrollo. Una vez se tenían acabadas la mayoría de las funcionalidades, no solo se ha vuelto a testear funcionalidades concretas, sino que se han llevado a cabo test de integración. Los test de integración han permitido comprobar el funcionamiento de distintos módulos y funcionalidades que trabajan conjuntamente, así como poder testear casos de uso completos. En este caso ha sido muy difícil automatizar estos test, ya que era necesario comprobar la interfaz de usuario, los *widgets* usados y su comportamiento, distintas funcionalidades de distintas capas trabajando juntas, etc. Es por eso que la mayoría de estos test han sido realizados de forma manual.

Para ello, no solo el autor de este proyecto los ha realizado, sino que ha contado con un grupo de *beta testers* que han ayudado a realizar estos test. Este grupo se ha compuesto por 6 personas cuyo rango de edades es: 2 personas entre 20 y 30, 2 personas entre 45 y 55 y dos personas entre 70 y 80. Además, 4 de ellos usan frecuentemente aplicaciones móviles y sistemas de reservas (de hoteles y restaurantes) sin embargo las otras dos no están acostumbradas a hacerlo.

Se han realizado test a todos los casos de uso, para comprobar el funcionamiento tanto en el escenario de éxito como en los otros escenarios. Los resultados de estos test son binarios, ya que el caso de uso puede realizarse o no realizarse. En el caso de los test para los requisitos no funcionales se ha usado la condición de satisfacción para comprobar si ha sido satisfecho o no cada requisito. Tanto el autor como los distintos *beta testers* han participado en la valoración de los requisitos no funcionales.

14.1 Test de requisitos funcionales

Para cada escenario de cada caso de uso se expone el objetivo del test, el input recibido, el resultado obtenido y los comentarios que se han decidido añadir. En algunos casos no es necesario input o comentarios y se ha indicado con un guion.

Test del caso de uso 1: Registro

Escenario: Éxito	
Objetivo	Verificar que se puede crear una cuenta en el sistema introduciendo datos correctos.
Input	Nombre: Prueba nombre, correo: santiotin98@gmail.com, contraseña: 1234Qwerty, fecha: 01/01/1990, género: null
Resultado	Correcto
Comentarios	La cuenta es creada correctamente y se le envía un correo al usuario para que confirme su cuenta.

Tabla 45: Requisito funcional 1, test escenario de éxito. Fuente: Elaboración propia.

Escenario: Campos incorrectos	
Objetivo	Verificar que no se puede crear una cuenta con campos incorrectos
Input	Nombre: null, correo: hola, contraseña: 12, fecha: null, género: hombre.
Resultado	Correcto
Comentarios	La aplicación avisa al usuario de que existen errores en los campos de nombre, correo y contraseña.

Tabla 46: Requisito funcional 1, test escenario campos incorrectos. Fuente: Elaboración propia.

Escenario: Usuario existente	
Objetivo	Verificar que un usuario con una cuenta no puede crear otra
Input	Nombre: Prueba nombre, correo: santiotin98@gmail.com, contraseña: 1234Qwerty, fecha: 01/01/1990, género: null
Resultado	Correcto
Comentarios	La cuenta no es creada y no se le envía ninguna confirmación al usuario

Tabla 47: Requisito funcional 1, test escenario usuario existente. Fuente: Elaboración propia.

Test del caso de uso 2: Login

Escenario: Éxito	
Objetivo	Verificar si el usuario puede iniciar sesión en la app con sus credenciales.
Input	Correo: santiotin98@gmail.com, contraseña: 1234Qwerty
Resultado	Correcto
Comentarios	El usuario es redireccionado a la pantalla inicial de la aplicación.

Tabla 48: Requisito funcional 2, test escenario éxito. Fuente: Elaboración propia.

Escenario: Credenciales erróneas	
Objetivo	Verificar que, si las credenciales no son correctas, el usuario no puede entrar en la app
Input	Correo: santiotin98@gmail.com, password: hola
Resultado	Correcto
Comentarios	El usuario no puede iniciar sesión y la app le muestra un mensaje de credenciales erróneas.

Tabla 49: Requisito funcional 2, test escenario credenciales erróneas. Fuente: Elaboración propia.

Test del caso de uso 3: Contraseña olvidada

Escenario: Éxito	
Objetivo	En caso de contraseña olvidada, el usuario podrá crear otra
Input	Correo: santiotin98@gmail.com
Resultado	Correcto
Comentarios	El usuario recibe un correo con un enlace donde podrá crear una nueva contraseña

Tabla 50: Requisito funcional 3, test escenario éxito. Fuente: Elaboración propia.

Escenario: Usuario inexistente	
Objetivo	Si la dirección de correo no está vinculado a ningún usuario, no se mandará nada.
Input	Correo: hola@hola.com
Resultado	Correcto
Comentarios	El sistema detecta que este correo no pertenece a ningún usuario y no manda nada.

Tabla 51: Requisito funcional 3, test escenario usuario inexistente. Fuente: Elaboración propia.

Test del caso de uso 4: Log out

Escenario: Éxito	
Objetivo	Comprobar que el usuario actual puede salir de su cuenta y no puede volver a entrar sin introducir de nuevo sus credenciales.
Input	-
Resultado	Correcto
Comentarios	El usuario sale de su cuenta y para volver a entrar a de introducir nuevamente sus credenciales.

Tabla 52: Requisito funcional 4, test escenario éxito. Fuente: Elaboración propia.

Test del caso de uso 5: Consultar información

Escenario: Éxito	
Objetivo	El usuario es capaz de ver toda la información relacionada con un negocio concreto.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 53: Requisito funcional 5, test escenario éxito. Fuente: Elaboración propia.

Test del caso de uso 6: Agregar negocio a favoritos

Escenario: Éxito	
Objetivo	El usuario es capaz de añadir un negocio a sus negocios favoritos.
Input	-
Resultado	Correcto
Comentarios	El usuario añade un negocio y este pasa automáticamente a poder verse en el apartado de la app de negocios favoritos.

Tabla 54: Requisito funcional 6, test escenario éxito. Fuente: Elaboración propia.

Escenario: Denegación de confirmación	
Objetivo	Comprobar que si el usuario no confirma la acción el negocio no es añadido
Input	-
Resultado	Correcto
Comentarios	-

Tabla 55: Requisito funcional 6, test escenario denegación de confirmación. Fuente: Elaboración propia.

Escenario: Negocio ya añadido	
Objetivo	Si el negocio ya es favorito, empezará el caso de uso 7
Input	-
Resultado	Correcto
Comentarios	-

Tabla 56: Requisito funcional 6, test escenario negocio ya añadido. Fuente: Elaboración propia.

Test del caso de uso 7: Quitar negocio de favoritos

Escenario: Éxito	
Objetivo	Comprobar que, si el negocio ya es favorito, este es quitado de la lista.
Input	-
Resultado	Correcto
Comentarios	Después de la confirmación, automáticamente el negocio pasa a no estar disponible como negocio favorito

Tabla 57: Requisito funcional 7, test escenario éxito. Fuente: Elaboración propia.

Escenario: Denegación de confirmación	
Objetivo	Si el usuario no confirma la acción, el negocio no es quitado de favoritos
Input	-
Resultado	Correcto
Comentarios	-

Tabla 58: Requisito funcional 7, test escenario denegación de confirmación. Fuente: Elaboración propia.

Escenario: Negocio no favorito	
Objetivo	Si el negocio no es favorito, empieza el caso de uso 6
Input	-
Resultado	Correcto
Comentarios	-

Tabla 59: Requisito funcional 7, test escenario negocio no favorito. Fuente: Elaboración propia.

Test del caso de uso 8: Consultar cita previa

Escenario: Éxito	
Objetivo	El usuario puede consultar las citas previas que ha pedido.
Input	-
Resultado	Correcto
Comentarios	Se puede consultar la información adyacente a cada cita previa.

Tabla 60: Requisito funcional 8, test escenario éxito. Fuente: Elaboración propia.

Escenario: Ningún resultado	
Objetivo	Informar al usuario que no tiene ninguna cita previa.
Input	-
Resultado	Correcto
Comentarios	Se le muestra al usuario una ilustración para que entienda mejor que aún no tiene citas previas y no piense que la app no funciona.

Tabla 61: Requisito funcional 8, test escenario ningún resultado. Fuente: Elaboración propia.

Test del caso de uso 9: Creación de cita previa de forma manual

Escenario: Éxito	
Objetivo	Comprobar que el usuario es capaz de crear una cita previa de un negocio no existente en el sistema
Input	Nombre: Dentista, Categoría: Dentista, Fecha: 24/01/2021, Hora: 19:00, Duración: 30 min, Servicio: Limpieza, Dirección: Calle Aribau 308, Notas: ninguna
Resultado	Correcto
Comentarios	La cita previa es creada y puede ser consultada por el usuario.

Tabla 62: Requisito funcional 9, test escenario éxito. Fuente: Elaboración propia.

Escenario: Campos vacíos, incorrectos o no válidos	
Objetivo	El sistema no deja al usuario crear una cita previa con campos vacíos, incorrectos o no válidos
Input	Nombre: null, Categoría: Peluquería, Fecha: 01/2021, Hora: hola, Duración: null, Servicio: Corte de Pelo, Dirección: Calle Londres 12, Notas: ninguna
Resultado	Correcto
Comentarios	El sistema avisa al usuario de los campos incorrectos y obligatorios, en este caso nombre, fecha, hora y duración estarían mal.

Tabla 63: Requisito funcional 9, test escenario campos vacíos. Fuente: Elaboración propia.

Escenario: Cita solapada	
Objetivo	El sistema no deja al usuario crear una cita previa de forma manual que se solapa con otra existente.
Input	Nombre: Pelu, Categoría: Peluquería, Fecha: 24/01/2021, Hora: 19:15, Duración: 45 min, Servicio: Corte de Pelo, Dirección: Calle Londres 12, Notas: ninguna.
Resultado	Correcto
Comentarios	El sistema avisa al usuario de este hecho y le recomienda que anule la otra cita antes de crear una nueva.

Tabla 64: Requisito funcional 9, test escenario cita solapada. Fuente: Elaboración propia.

Test del caso de uso 10: Edición de cita previa manual

Escenario: Éxito	
Objetivo	El usuario ha de poder editar los atributos de una cita previa que se ha creado de forma manual.
Input	Nombre: Dentista, Categoría: Dentista, Fecha: 24/01/2021, Hora: 19:30, Duración: 45 min, Servicio: Limpieza, Dirección: Calle Aribau 308, Notas: ninguna.
Resultado	Correcto
Comentarios	El usuario modifica algunos o todos los atributos de la cita previa manual.

Tabla 65: Requisito funcional 10, test escenario éxito. Fuente: Elaboración propia.

Escenario: Campos vacíos, incorrectos o no válidos	
Objetivo	En el caso de que haya errores en los nuevos campos, la cita no ha de poder ser editada.
Input	Nombre: Dentista, Categoría: null, Fecha: 24/01/2021, Hora: 19, Duración: 30 min, Servicio: null, Dirección: Calle Aribau 308, Notas: ninguna.
Resultado	Correcto
Comentarios	El sistema muestra un mensaje de que hay campos que no son correctos.

Tabla 66: Requisito funcional 10, test escenario campos vacíos. Fuente: Elaboración propia.

Escenario: Cita solapada	
Objetivo	Si la nueva cita resultante de la edición se solapa con otra existente no ha de poder ser editada.
Input	Nombre: Dentista, Categoría: Dentista, Fecha: 24/01/2021, Hora: 19:30, Duración: 45 min, Servicio: Limpieza, Dirección: Calle Aribau 308, Notas: ninguna.
Resultado	Correcto
Comentarios	El sistema avisa al usuario de esta situación para que pueda ser corregida.

Tabla 67: Requisito funcional 10, test escenario cita solapada. Fuente: Elaboración propia.

Escenario: Denegación de confirmación	
Objetivo	En caso de denegación de confirmación, la cita ha de permanecer como estaba antes.
Input	-
Resultado	Correcto
Comentarios	La cita no se ha modificado.

Tabla 68: Requisito funcional 10, test escenario denegación de confirmación. Fuente: Elaboración propia.

Test del caso de uso 11: Eliminación de cita previa

Escenario: Éxito	
Objetivo	Eliminar la cita previa seleccionada.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 69: Requisito funcional 11, test escenario éxito. Fuente: Elaboración propia.

Escenario: Denegación de confirmación	
Objetivo	En caso de denegación de confirmación, la cita no ha de ser eliminada.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 70: Requisito funcional 11, test escenario denegación de confirmación. Fuente: Elaboración propia.

Test del caso de uso 12: Pedir cita previa en negocio

Escenario: Éxito	
Objetivo	El usuario pide una cita previa en un negocio existente en el sistema.
Input	Negocio: Aire Ancient Baths, Servicio: Acceso a piscinas, Fecha: 07/01/2021, Hora: 20:00, Profesional: Elena
Resultado	Correcto
Comentarios	El sistema crea una cita previa para este usuario en el negocio concreto.

Tabla 71: Requisito funcional 12, test escenario éxito. Fuente: Elaboración propia.

Escenario: Trabajador no ofrece servicio	
Objetivo	El sistema no permite seleccionar un trabajador que no ofrece el servicio seleccionado.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 72: Requisito funcional 12, test escenario trabajador no ofrece servicio. Fuente: Elaboración propia.

Escenario: Cita solapada	
Objetivo	El sistema no permite seleccionar una fecha que se solape con otra cita previa, ni del usuario ni del negocio
Input	-
Resultado	Correcto
Comentarios	El sistema comprueba tanto las citas del negocio en cuestión como las del usuario.

Tabla 73: Requisito funcional 12, test escenario cita solapada. Fuente: Elaboración propia.

Escenario: Fecha anterior	
Objetivo	El sistema no permite seleccionar al usuario una fecha anterior a la actual.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 74: Requisito funcional 12, test escenario fecha anterior. Fuente: Elaboración propia.

Escenario: Denegación de confirmación	
Objetivo	Si se deniega la confirmación, la cita previa no es creada.
Input	-
Resultado	Correcto
Comentarios	Esta confirmación es usada para que el usuario pueda asegurarse de que no se ha equivocado y quiere pedir esta cita previa.

Tabla 75: Requisito funcional 12, test escenario denegación de confirmación. Fuente: Elaboración propia.

Test del caso de uso 13: Cancelar cita previa en negocio

Escenario: Éxito	
Objetivo	El usuario cancela una cita previa que ha creado previamente.
Input	-
Resultado	Correcto
Comentarios	La cita previa es cancelada y pasa a no estar disponible ni para el negocio ni para el usuario.

Tabla 76: Requisito funcional 13, test escenario éxito. Fuente: Elaboración propia.

Escenario: Denegación de confirmación	
Objetivo	Si se deniega la confirmación, la cita previa no es cancelada.
Input	-
Resultado	Correcto
Comentarios	Esta confirmación es usada para que el usuario pueda asegurarse de que no se ha equivocado y quiere cancelar esta cita previa.

Tabla 77: Requisito funcional 13, test escenario denegación de confirmación. Fuente: Elaboración propia.

Test del caso de uso 14: Buscar negocios por nombre

Escenario: Éxito	
Objetivo	El usuario puede buscar por palabras los negocios que hay en el sistema.
Input	Nombre: Aire
Resultado	Correcto
Comentarios	Se muestran los negocios que contienen alguna palabra de las buscadas.

Tabla 78: Requisito funcional 14, test escenario éxito. Fuente: Elaboración propia.

Escenario: Ningún resultado	
Objetivo	Mostrar al usuario un mensaje de que no se han encontrado resultados.
Input	-
Resultado	Correcto
Comentarios	Esto se hace para que el usuario no piense que la app no está funcionando al no mostrar ningún resultado.

Tabla 79: Requisito funcional 14, test escenario ningún resultado. Fuente: Elaboración propia.

Test del caso de uso 15: Buscar negocios por ciudad

Escenario: Éxito	
Objetivo	El usuario puede buscar por ciudad los negocios que hay en el sistema.
Input	Ciudad: Barcelona
Resultado	Correcto
Comentarios	Se muestran los negocios del sistema que están en la ciudad seleccionada.

Tabla 80: Requisito funcional 15, test escenario éxito. Fuente: Elaboración propia.

Escenario: Ningún resultado	
Objetivo	Mostrar al usuario un mensaje de que no se han encontrado resultados.
Input	-
Resultado	Correcto
Comentarios	Esto se hace para que el usuario no piense que la app no está funcionando al no mostrar ningún resultado.

Tabla 81: Requisito funcional 15, test escenario ningún resultado. Fuente: Elaboración propia.

Test del caso de uso 16: Buscar negocios por categoría

Escenario: Éxito	
Objetivo	El usuario puede buscar por categoría los negocios que hay en el sistema.
Input	Categoría: Peluquería
Resultado	Correcto
Comentarios	Se muestran los negocios del sistema que tienen la misma categoría que la seleccionada.

Tabla 82: Requisito funcional 16, test escenario éxito. Fuente: Elaboración propia.

Escenario: Ningún resultado	
Objetivo	Mostrar al usuario un mensaje de que no se han encontrado resultados.
Input	-
Resultado	Correcto
Comentarios	Esto se hace para que el usuario no piense que la app no está funcionando al no mostrar ningún resultado.

Tabla 83: Requisito funcional 16, test escenario ningún resultado. Fuente: Elaboración propia.

Test del caso de uso 17: Cambiar visualización de los resultados

Escenario: Éxito	
Objetivo	Los resultados han de poder ser vistos tanto en lista como en un mapa.
Input	-
Resultado	Correcto
Comentarios	El mapa muestra marcadores en los lugares donde haya un negocio que forma parte del resultado.

Tabla 84: Requisito funcional 17, test escenario éxito. Fuente: Elaboración propia.

Test del caso de uso 18: Consultar siguiente cita

Escenario: Éxito	
Objetivo	El usuario podrá ver en la pantalla principal cuál es su próxima cita.
Input	-
Resultado	Correcto
Comentarios	De esta forma, el usuario sabe rápidamente que cita e la siguiente que tiene.

Tabla 85: Requisito funcional 18, test escenario éxito. Fuente: Elaboración propia.

Escenario: Ningún resultado	
Objetivo	El usuario verá un mensaje que dice que no tiene próximas citas.
Input	-
Resultado	Correcto
Comentarios	Es necesario el mensaje, ya que si hubiera un espacio en blanco podría confundirse con un mal funcionamiento.

Tabla 86: Requisito funcional 18, test escenario ningún resultado. Fuente: Elaboración propia.

Test del caso de uso 19: Consultar negocios favoritos

Escenario: Éxito	
Objetivo	El usuario podrá ver en la pantalla principal todos sus negocios favoritos.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 87: Requisito funcional 19, test escenario éxito. Fuente: Elaboración propia.

Escenario: Ningún resultado	
Objetivo	El usuario verá un mensaje si no tiene ningún negocio favorito.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 88: Requisito funcional 19, test escenario ningún resultado. Fuente: Elaboración propia.

Test del caso de uso 20: Consultar promociones

Escenario: Éxito	
Objetivo	El usuario podrá ver en la pantalla principal todas las promociones actuales de los negocios.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 89: Requisito funcional 20, test escenario éxito. Fuente: Elaboración propia.

Escenario: Ningún resultado	
Objetivo	El usuario verá un mensaje si no hay ninguna promoción activa.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 90: Requisito funcional 20, test escenario ningún resultado. Fuente: Elaboración propia.

Test del caso de uso 21: Editar información personal

Escenario: Éxito	
Objetivo	El usuario podrá editar los atributos de su cuenta personal.
Input	Nombre: Santi Otín, Fecha de nacimiento: 12/06/1998, Género: hombre.
Resultado	Correcto
Comentarios	-

Tabla 91: Requisito funcional 21, test escenario éxito. Fuente: Elaboración propia.

Escenario: Campos vacíos, incorrectos o no válidos	
Objetivo	Si los campos no son correctos no se podrán editar los datos
Input	Nombre: null, Fecha de nacimiento: 1998, Género: hombre.
Resultado	Correcto
Comentarios	Se muestra un mensaje al usuario indicando que campos son incorrectos.

Tabla 92: Requisito funcional 21, test escenario campos vacíos. Fuente: Elaboración propia.

Escenario: Denegación de confirmación	
Objetivo	Si el usuario no confirma la edición, los campos permanecen como estaban anteriormente.
Input	-
Resultado	Correcto
Comentarios	-

Tabla 93: Requisito funcional 21, test escenario denegación de confirmación. Fuente: Elaboración propia.

14.2 Test de requisitos no funcionales

Como ya se ha dicho anteriormente, los requisitos funcionales han sido comprobados por el autor y por un grupo de *beta testers*. Estas comprobaciones no solo han sido hechas de forma manual, sino que las decisiones tomadas a lo largo del proyecto han influido en la satisfacción de estos requisitos.

Para la presentación de los resultados se expone, por cada requisito no funcional, su condición de satisfacción, el resultado binario de haberla satisfecho o no y comentarios en caso necesario. La explicación de cada requisito puede encontrarse en el capítulo 11.

Requisito no funcional 1: Apariencia	
Condición de satisfacción	Se dará por alcanzado este requisito si durante el desarrollo de la aplicación se han utilizado los patrones y estilos definidos para los sistemas de Android y iOS.
Resultado	Satisfecho
Comentarios	Se ha utilizado tanto material design para Android como la guía de diseño de Apple para la realización de las interfaces de la aplicación.

Tabla 94: Test requisito no funcional 1. Fuente: Elaboración propia.

Requisito no funcional 2: Usabilidad	
Condición de satisfacción	Se dará por alcanzado este requisito si durante las fases de *testing* de las distintas funcionalidades, la aplicación es considerada intuitiva y fácil de utilizar.
Resultado	Satisfecho
Comentarios	Tanto el autor como los *beta testers* han considerado que la aplicación es intuitiva y fácil de utilizar para cualquier rango de edad y conocimiento.

Tabla 95: Test requisito no funcional 2. Fuente: Elaboración propia.

Requisito no funcional 3: Fiabilidad y Disponibilidad	
Condición de satisfacción	Se dará por alcanzado este requisito si durante las fases de *testing* de las distintas funcionalidades, así como las fases de *testing* de integración, la aplicación se mantiene activa un 95% del tiempo de pruebas.
Resultado	Satisfecho
Comentarios	Durante todos los tests y el desarrollo el sistema ha estado activo un 99% del tiempo aproximadamente.

Tabla 96: Test requisito no funcional 3. Fuente: Elaboración propia.

Requisito no funcional 4: Aprendizaje	
Condición de satisfacción	Se dará por alcanzado este requisito si durante las fases de *testing* de las distintas funcionalidades, la aplicación es considerada de aprendizaje rápido y sencillo.
Resultado	Satisfecho
Comentarios	Todos los usuarios ajenos al proyecto que han podido utilizar el sistema no ha hecho falta explicar nada para que lo usaran.

Tabla 97: Test requisito no funcional 4. Fuente: Elaboración propia.

Requisito no funcional 5: Internacionalización	
Condición de satisfacción	Se dará por alcanzado este requisito si una vez acabado el desarrollo, todos los textos de la aplicación están traducidos tanto al inglés como al español.
Resultado	Satisfecho
Comentarios	El sistema admite tanto el inglés como el español y decide en función del idioma del dispositivo del usuario.

Tabla 98: Test requisito no funcional 5. Fuente: Elaboración propia.

Requisito no funcional 6: Velocidad y Latencia	
Condición de satisfacción	Se dará por alcanzado este requisito si durante el *testing* los resultados de las pruebas de velocidad y latencia son satisfactorios en un 95% de los casos. Estas pruebas consistirán en la repetición de situaciones donde el rendimiento de velocidad y latencia sea importante, anotando los resultados obtenidos y comparándolos con los valores estándar de la industria de las apps.
Resultado	Satisfecho
Comentarios	Los valores de velocidad y latencia entran dentro del estándar de la industria.

Tabla 99: Test requisito no funcional 6. Fuente: Elaboración propia.

Requisito no funcional 7: Capacidad	
Condición de satisfacción	Se dará por alcanzado este requisito si durante el *testing* el sistema soporta el tráfico de datos simultáneo de todos los beta *testers*. Además, se ha de contar con un plan para el aumento en la capacidad de la aplicación y sobretodo la base de datos tanto en términos de almacenamiento como tráfico.
Resultado	Satisfecho
Comentarios	El sistema ha soportado a todos los *beta testers* simultáneamente. Además, sobretodo en la arquitectura y diseño se han tomado las decisiones basadas en la capacidad que podría llegar a obtener.

Tabla 100: Test requisito no funcional 7. Fuente: Elaboración propia.

Requisito no funcional 8: Escalabilidad	
Condición de satisfacción	Se dará por alcanzado este requisito si durante el desarrollo se tiene en mente esta circunstancia, se prevé este aumento y se toman decisiones en base a esta situación.
Resultado	Satisfecho
Comentarios	En el capítulo de arquitectura y diseño se explican las decisiones tomadas para garantizar este requisito no funcional.

Tabla 101: Test requisito no funcional 8. Fuente: Elaboración propia.

Requisito no funcional 9: Adaptabilidad	
Condición de satisfacción	Se dará por alcanzado este requisito si el prototipo final de la aplicación funciona correctamente en cualquier clase de dispositivo móvil independientemente de su tamaño o de si su sistema operativo es Android o iOS.
Resultado	Satisfecho
Comentarios	El sistema funciona correctamente tanto en Android como en iOS y para una gran variedad de tamaños de pantalla.

Tabla 102: Test requisito no funcional 9. Fuente: Elaboración propia.

Requisito no funcional 10: Seguridad y privacidad	
Condición de satisfacción	Se dará por alcanzado este requisito si durante el desarrollo de la aplicación y de la base de datos se han cumplido todas las leyes y se ha comprobado que los datos de los usuarios son inaccesibles para cualquier usuario que no esté autorizados a verlos.
Resultado	Satisfecho
Comentarios	En todas las capas, especialmente en la de datos, se ha garantizado esta seguridad y privacidad. La base de datos cumple con todas las normativas en este aspecto.

Tabla 103: Test requisito no funcional 10. Fuente: Elaboración propia.

Requisito no funcional 11: Legislación	
Condición de satisfacción	Se dará por alcanzado este requisito si todos los datos tratados son transferidos y guardados de manera segura, siguiendo la normativa vigente.
Resultado	Satisfecho
Comentarios	No solo se guardan y se transfieren de forma segura, sino que en el capítulo 10 se ha analizado la legislación actual para estar seguro de que no se infringen las normas.

Tabla 104: Test requisito no funcional 11. Fuente: Elaboración propia.

15. Informe final

Una vez finalizado el trabajo, es importante analizar cómo ha ido el desarrollo del proyecto y compararlo con las panificaciones que se habían hecho al inicio de este. Los resultados de esta comparativa permitirán actualizar la planificación y ser conscientes de lo ocurrido para así aplicar esta experiencia a la planificación de futuros proyectos.

Seguidamente se expondrán las desviaciones que han ocurrido a lo largo del desarrollo de este proyecto y cómo estas han afectado a la planificación inicial. Finalmente se concluirá con la justificación de las competencias que se han trabajado y las futuras acciones que se van a tomar para continuar con el proyecto del cual forma parte este trabajo.

15.1 Desviaciones

Durante la realización de este trabajo se han dado ciertas situaciones que han producido desvíos en la planificación inicial. Aunque estas desviaciones no han afectado sustancialmente a la ejecución del trabajo, hay que tenerlas en cuenta para reestructurar el trabajo de acuerdo a las acciones implementadas, así como en un futuro realizar mejores planificaciones iniciales.

Código	Duración estimada (h)	Duración real (h)
GP	70	70
Desarrollo	440	430
I	50	50
US	60	60
NEG	130	160
PER	50	50
BUS	100	80
FAV	10	10
NP	40	20
Documentación	100	120
Total	610	620

Tabla 105: Desviación de horas. Fuente: Elaboración propia.

Como se puede ver en la imagen superior, las tareas de gestión de citas previas en negocios (NEG) han llevado más horas de lo esperado (30 horas más). Esto es debido a que su complejidad ha sido mayor de lo esperado, sobretodo a la hora de tener en cuenta todas las limitaciones que se han de aplicar para que no hay inconsistencia entre citas a la hora de crearlas.

Por otro lado, las tareas relacionadas con la búsqueda de negocios (BUS) se han podido hacer más rápido de lo esperado (20 horas menos). Como ya se dijo en el apartado de gestión de riesgos, algunas de las tareas fueron sobreestimadas como método de prevención, y en este caso ha servido para compensar (o casi) las horas que se habían perdido.

Otra desviación en la planificación inicial ha sido el orden de realización de ciertas tareas. En concreto, se ha realizado primero la búsqueda de negocios antes que la gestión de citas previas en negocios. Esto se ha hecho debido a que para poder realizar una cita previa en un negocio primero se debía encontrar ese negocio, y para no simular esta llegada de otra forma y perder tiempo codificando algo que luego sería borrado se ha decidido alternar el orden. Sin embargo, esta desviación no ha influido en las horas.

Finalmente, la documentación del proyecto ha sido más costosa de lo esperado, teniendo que involucrar 20 horas más de lo esperado. Esto hace que se hayan gastado 30 horas más (50 perdidas más 20 ganadas) que han tenido que ser compensadas usando una estrategia que ya se planteó en la gestión de riesgos. Como ya se explicó en ese punto, las dos últimas funcionalidades que se dejaron para el final no eran determinantes para el propósito principal del sistema y podían ser modificadas si existía una desviación importante. Para compensar estas horas de desviación, se ha decidido no desarrollar en profundidad la funcionalidad de notificaciones, dejándola solo como una funcionalidad más básica de lo que se había planeado inicialmente y así poder reducir en 20 horas su desarrollo. De esta forma se ha podido compensar el desvío, teniendo que realizar solo 10 horas más de las planeadas y con la funcionalidad de notificaciones algo más simple que lo planeado.

15.1.2 Desviaciones económicas

Las desviaciones en la planificación explicadas anteriormente han tenido consecuencias en las planificaciones económicas. Se pueden observar en las siguientes tablas:

Código	Coste Estimado S1 (€)	Coste Real S1 (€)
GP	630	630
Desarrollo	3960	3870
I	450	450
US	540	540
NEG	1170	1440
PER	450	450
BUS	900	720
FAV	90	90
NP	360	180
Documentación	900	1080
Total	5490	5580

Tabla 106: Desviación económica en el escenario 1. Fuente: Elaboración propia.

Código	Coste Estimado S2 (€)	Coste Real S2 (€)
GP	1932	1932
Desarrollo	9181	8973
I	1060	1060
US	1249	1249
NEG	2706	3330
PER	1043	1043
BUS	2081	1665
FAV	208	208
NP	832	416
Documentación	2523	3028
Total	13635	13934

Tabla 107: Desviación económica en el escenario 2. Fuente: Elaboración propia.

Como se puede observar, en los dos escenarios ha habido un aumento de costes de CPA, ya que se han hecho ligeramente más horas (10 más). Sin embargo, este aumento es menor que los costes que se destinaron a contingencia e imprevistos, por lo tanto, el proyecto puede ser asumido con el presupuesto inicial que se hizo.

15.2 Competencias

CES1.1: Desarrollar, mantener y evaluar sistemas y servicios software complejos i/o críticos. Trabajado en profundidad.

La complejidad de este proyecto viene determinada tanto por sus requisitos como por el marco al que pertenece. Por una parte, no había ningún cliente específico y el mercado era el que determinaba los requisitos, siendo especialmente importante y complejo el análisis de estos. Además, para satisfacer estos requisitos la app ha tenido que contar con funcionalidades elaboradas. Pero al formar todo de un proyecto más grande, en el que en un futuro va a tener que ser integrado un sistema gestor de negocios, ha sido imprescindible hacer la elaboración de este sistema con esta circunstancia en mente. Esto ha condicionado mucho las decisiones tomadas y los diseños realizados, añadiéndole un grado más de complejidad a todo el proceso de desarrollo.

Al ser un sistema basado en información en tiempo real, también ha sido muy importante tener siempre en mente la integridad de los datos en todos los procesos, así como en las reacciones a las acciones realizadas en el sistema, ya que estas han de ser propagadas de tal forma que el usuario vea cómo el sistema mantiene siempre una coherencia e integridad en sus datos.

Por estos motivos se considera que esta competencia ha sido trabajada en profundidad. Cabe destacar que la mayoría de conocimientos han sido adquiridos especialmente de la asignatura de **IES** y sobretodo **AS**.

CES1.2: Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles. Trabajado bastante.

Como ya se ha comentado en la competencia anterior, el sistema *marketplace* desarrollado forma parte de una plataforma dónde también existirá un sistema de gestión de negocio. Los datos de estos dos sistemas van a ser compartidos y esto hace que tanto la estructuración de los datos como la integración de estos dos sistemas sea un factor muy importante. Esta circunstancia no solo afecta a la arquitectura y el diseño, sino que también hay que tener en cuenta las tecnologías que se van a usar ya que la combinación de éstas puede dar lugar a mejores o peores implementaciones. Por lo tanto, en todas las decisiones técnicas ha sido necesario tener en cuenta esta futura integración. Además, dentro de la app desarrollada se ha optado por la utilización de servicios de terceros, teniéndolos que integrar de la mejor forma posible en el sistema para sacar el máximo rendimiento de ellos y obtener una gran eficacia.

CES1.4: Desarrollar, mantener y evaluar servicios y aplicaciones distribuidas con soporte de red. Trabajado en profundidad.

El sistema desarrollado se apoya mucho en la conexión a la red, ya que es un sistema que pretende conectar en tiempo real tanto a clientes como a negocios. Esto ha hecho que se hayan buscado tecnologías que no solo faciliten esta conexión, sino que también la sepan explotar de tal forma que se consigan grandes resultados. Un ejemplo muy claro es la elección tanto de tecnología (Flutter) como los servicios de terceros (Firebase), que combinados permiten la reactividad del sistema, dando al usuario una mayor experiencia debido a que no ha de estar todo el rato refrescando la app para obtener los nuevos datos, sino que cuando estos cambian el usuario recibe los cambios automáticamente. Los conocimientos impartidos en **SOAD** y **ASW** han sido muy útiles para esta competencia.

CES1.5: Especificar, diseñar, implementar y evaluar bases de datos. Trabajado bastante.

Para la capa de datos ha sido necesario diseñar un sistema de persistencia de datos. Se ha razonado la necesidad de que esta base de datos sea de tipo no relacional y después de evaluar este hecho se ha implementado con la ayuda de un servicio externo. La especificación y el diseño pueden encontrarse en el capítulo de arquitectura y diseño y su implementación en el de implementación. Gracias a las asignaturas de **DB** y **DBD** se conocían previamente las distintas bases de datos que podían ser usadas y este proyecto ha servido para profundizar mucho en el uso de las no relacionales.

CES1.7: Controlar la calidad y diseñar pruebas en la producción de software. Trabajado un poco.

Esta competencia se ha cubierto con la creación y realización de test automáticos. El control de calidad de un software es esencial tanto durante las fases iniciales del desarrollo de este como en las fases finales.

A pesar de no haber podido crear tantos test automatizados como se hubiera querido debido a su dificultad y limitación de tiempo, todos los requisitos tanto funcionales como no funcionales han sido testeados de forma manual para asegurar que el software desarrollado cumple con lo esperado. Asignaturas como **AS** y **PES** han sido útiles sobretodo para la creación de los test automatizados.

CES2.1: Definir y gestionar los requisitos de un sistema software. Trabajado en profundidad.

Al no tener un cliente concreto, el mercado era el cliente de este proyecto, teniendo que analizar tanto a la competencia como a los pequeños negocios y usuarios de estos negocios. Se han realizado encuestas (ver Anexo) para conocer las necesidades reales de unos y otros, y se han analizado en profundidad todas las herramientas relacionadas que dan algún tipo de servicio relacionado con los problemas a resolver. Gracias a toda la información que se ha recabado, se han podido identificar y especificar de forma correcta los requisitos de este proyecto. Para esta competencia se han puesto en práctica todos los conocimientos dados en **ER**.

CES1.9: Demostrar comprensión en la gestión y el gobierno de los sistemas software. Trabajado un poco.

Aunque el alcance de este TFG solo incluya la app del *marketplace*, forma parte de una plataforma más grande. Para ello, ha sido necesario tener una comprensión global de esta plataforma y de su gestión, con el fin de definir correctamente los requisitos y arquitectura del *marketplace* y poderlo integrar y gobernar en el contexto de la plataforma global.

Gracias a la comprensión de las particularidades de este proyecto ha sido posible tomar las decisiones adecuadas, no solo a nivel de planificación sino sobretodo a nivel de desarrollo, pudiendo así satisfacer ampliamente los requisitos del proyecto a la vez que se conseguía una gran experiencia en su utilización.

15.3 Acciones futuras

Antes de empezar con el desarrollo del sistema gestor de negocios, es conveniente tener completamente finalizado el sistema *marketplace* tratado en este trabajo. Por lo tanto, el siguiente paso a tomar es la finalización del desarrollo de la tarea de notificaciones, que debido a las desviaciones presentadas en este capítulo no ha sido posible completar. Después de esto, la creación de más test automatizados sería lo siguiente. Esta tarea es muy importante ya que los test automatizados serán los encargados de confirmar que nada de lo desarrollado hasta la fecha deja de funcionar cuando se empiece a integrar con el otro sistema. Cuando se haya acabado lo descrito, se pretende volver a seguir todos los pasos de este trabajo esta vez para el sistema gestor de negocio, con la intención de obtener un prototipo funcional de la plataforma global y poder lanzar la primera versión al mercado.

Referencias

[1] "Comportamiento de los viajeros españoles en 2017: desde la inspiración hasta el destino." Think with Google, https://www.thinkwithgoogle.com/intl/es-es/insights/tendencias-de-consumo/comportamiento-de-los-viajeros-espa%C3%B1oles-en-2017-desde-la-inspiraci%C3%B3n-hasta-el-destino/. Accedido el 15 sept. 2020.

[2] "Booking.Com: The Largest Selection of Hotels, Homes, and Vacation Rentals." Booking.Com, https://www.booking.com/. Accedido el 15 sept. 2020.

[3] "Las reservas online de restaurantes crecen un 63% en verano." El Tenedor, 3 sept. 2020, https://blog.thefork.com/es/las-reservas-online-de-restaurantes-crecen-un-63-en-verano/. Accedido el 15 sept. 2020.

[4] *Calendario de Programación de citas online gratis Software | Setmore*. https://www.setmore.com/. Accedido el 20 sept. 2020.

[5] *SimplyBook.Me - Free Appointment Scheduling Software*. https://simplybook.me/en/. Accedido el 20 sept. 2020.

[6] *Reservio - Free Online Appointment Scheduling Software*. https://www.reservio.com/. Accedido el 20 sept. 2020.

[7] miadmweb. "La App para reserva de citas en peluquerías, barberías, clínicas - miCita." *La App para reserva de citas en peluquería, barbería, estética...*, https://micita.app/. Accedido el 22 sept. 2020.

[8] Łukasz. "Booksy - Hair Stylists, Barbers, Beauticians... Book Appointments Online!" *Booksy*, https://booksy.com/en-us/. Accedido el 23 sept. 2020.

[9] "¿Cuál es la metodología más adecuada para tu proyecto?" *Deloitte Spain*, https://www2.deloitte.com/es/es/pages/technology/articles/waterfall-vs-agile.html. Accedido el 25 sept. 2020.

[10] "Kanban vs. Scrum." *Deloitte Spain*. https://www2.deloitte.com/es/es/pages/technology/articles/kanban-vs-scrum.html. Accedido el 25 sept. 2020.

[11] "Taiga.Io." *Taiga - Love Your Projects*, https://taiga.io/. Accedido el 26 sept. 2020.

[12] *Git*. https://git-scm.com/. Accedido el 26 sept. 2020.

[13] "Build Software Better, Together." *GitHub*, https://github.com. Accedido el 26 sept. 2020.

[14] "A Successful Git Branching Model." *Nvie.Com*, http://nvie.com/posts/a-successful-git-branching-model/. Accedido el 26 sept. 2020.

[15] *Ofertas de Trabajo, Bolsa de Trabajo | Buscar Empleo En Indeed España*. https://es.indeed.com/. Accedido el 7 oct. 2020.

[16] CWork. “cwork | Coworking Barcelona | Oficinas Compartidas.” *cwork*, https://cwork.cat/. Accedido el 9 oct. 2020.

[17] Isabel. “Servicios y Tarifas de Meet BCN, tu espacio Coworking en Barcelona.” *Meet BCN - Coworking Barcelona*, https://meetbcn.com/coworking-barcelona/. Accedido el 9 oct. 2020.

[18] *Espacios Coworking Barcelona - Oficinas Compartidas | OneCoWork*. https://onecowork.com/es. Accedido el 9 oct. 2020.

[19] *What Is UML | Unified Modeling Language*. https://www.uml.org/what-is-uml.htm. Accedido el 14 nov. 2020.

[20] “Volere Requirements.” Volere Requirements, https://www.volere.org/. Accedido el 25 nov. 2020.

[21] Fernández, Samuel. “Android supera el 90% de cuota en España mientras que iOS cae por debajo del 9%, según Kantar.” Xataka Móvil, 16 Apr. 2019, https://www.xatakamovil.com/mercado/android-supera-90-cuota-espana-ios-cae-debajo-9-kantar. Accedido el 4 dic. 2020.

[22] React Native. https://reactnative.dev/. Accedido el 10 dic. 2020.

[23] Ionic. “Ionic - Cross-Platform Mobile App Development.” Ionic Framework, https://ionicframework.com/. Accedido el 10 dic. 2020.

[24] Flutter - Beautiful Native Apps in Record Time. https://flutter.dev/. Accedido el 10 sept. 2020.

[25] “Dart Packages.” Dart Packages, https://pub.dev/. Accedido el 10 sept. 2020.

[26] Bloc Library. https://bloclibrary.dev/#/. Accedido el 10 sept 2020.

[27] “Firebase.” Firebase, https://firebase.google.com/?hl=es. Accedido el 10 sept. 2020.

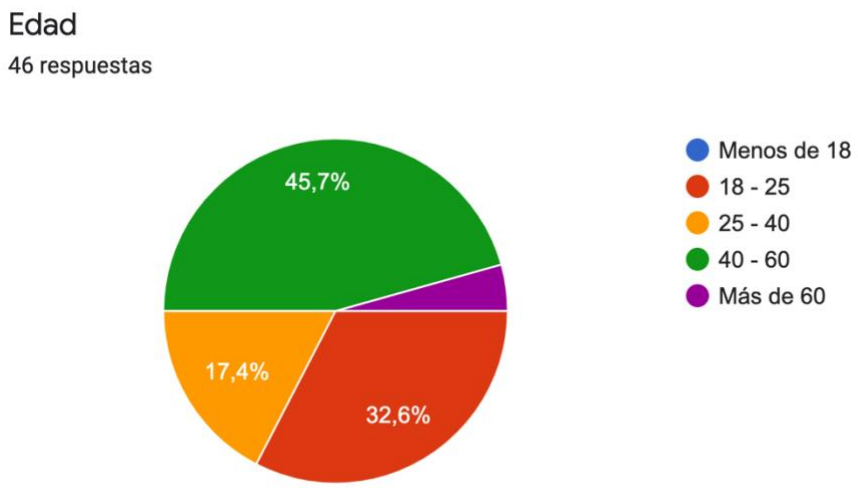
[28] “Stack Overflow - Where Developers Learn, Share, & Build Careers.” Stack Overflow, https://stackoverflow.com/. Accedido 20 sept. 2020.

Anexo

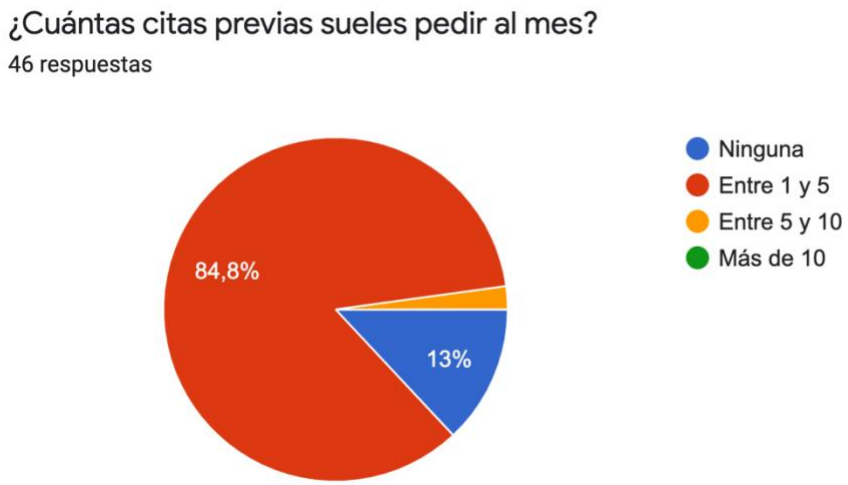
Encuesta realizada a clientes potenciales

Esta encuesta ha sido realizada por un total de 46 personas que cumplen el perfil de clientes potenciales de la herramienta desarrollada en este proyecto.

Pregunta 1



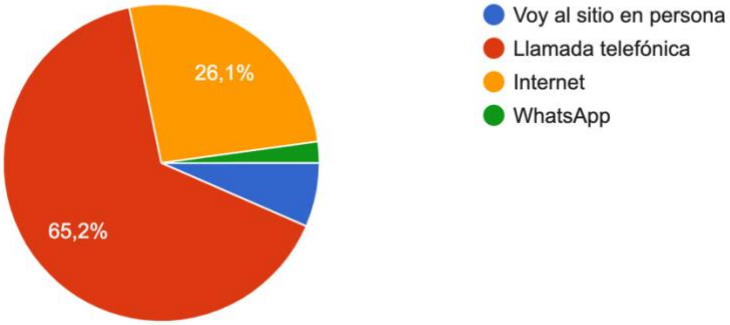
Pregunta 2



Pregunta 3

¿Qué método usas para pedir citas previas?

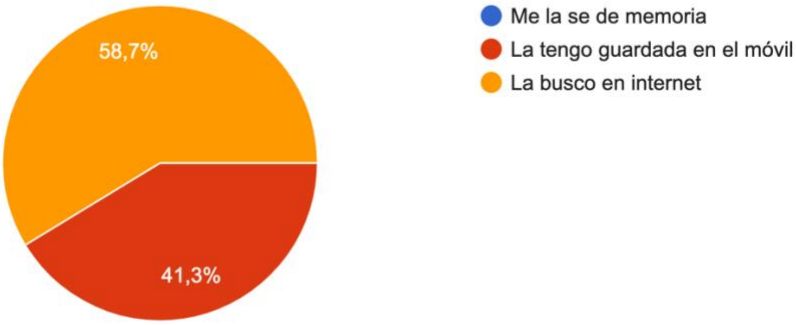
46 respuestas



Pregunta 4

En caso de ir al sitio o llamar, ¿dónde buscas su información de contacto?

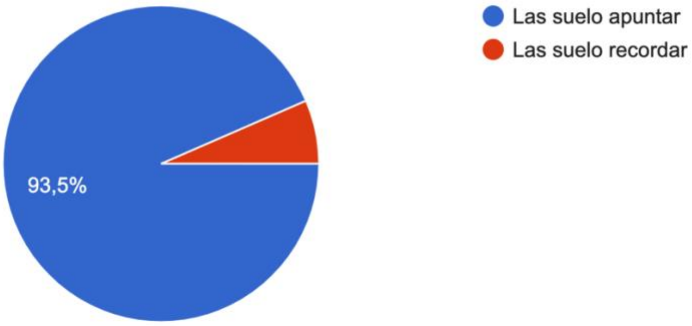
46 respuestas



Pregunta 5

¿Sueles apuntar tus citas previas o las recuerdas de memoria?

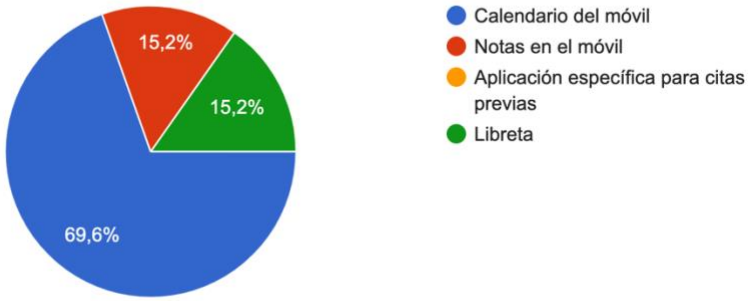
46 respuestas



Pregunta 6

En caso de apuntarlas, ¿dónde lo haces?

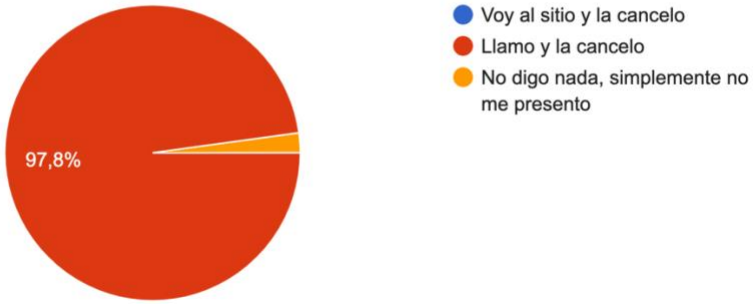
46 respuestas



Pregunta 7

En caso de que no puedas ir a una cita previa, ¿cómo la cancelas?

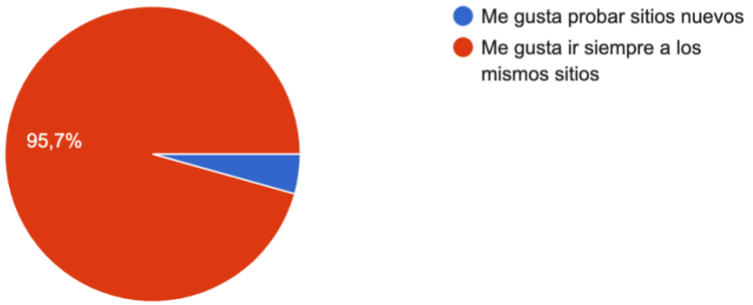
46 respuestas



Pregunta 8

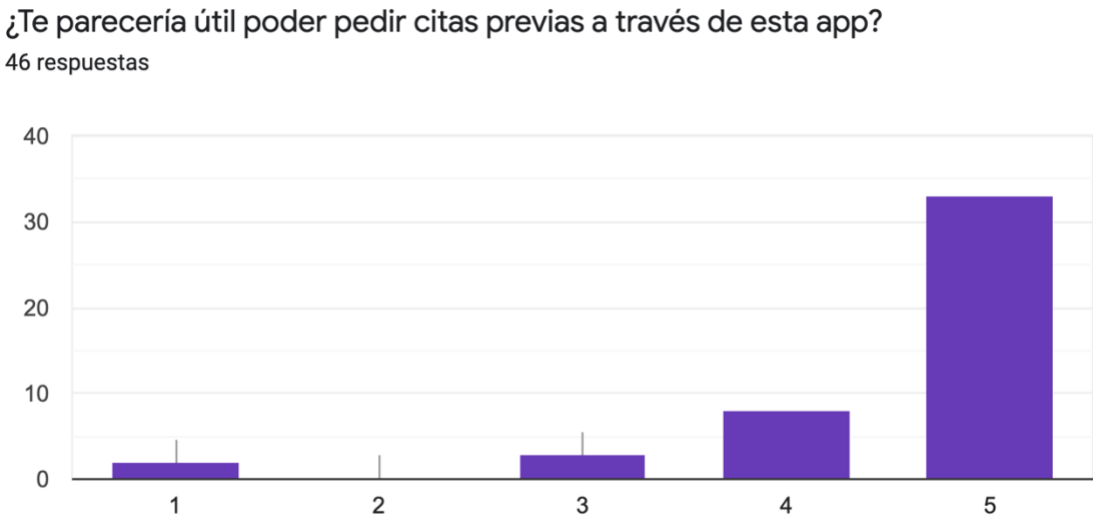
A la hora de pedir citas previas, ¿frecuentas siempre los mismos sitios o te gusta probar sitios distintos?

46 respuestas

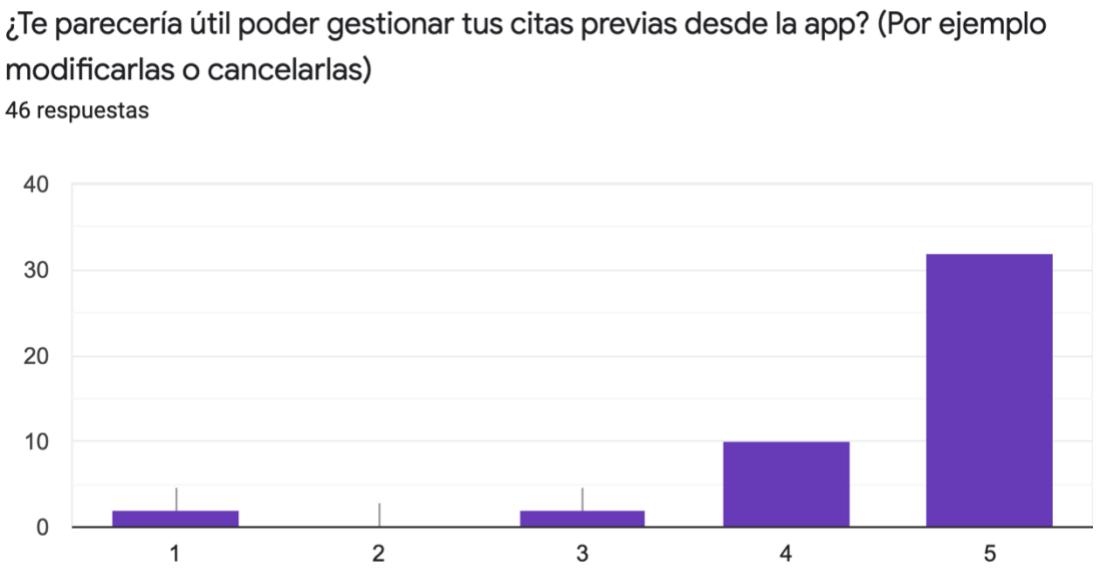


Suponiendo que existiera una app móvil donde estuvieran todos los establecimientos a los que vas (peluquería, clínicas, centros de estética, etc.) contesta a las siguientes preguntas:

Pregunta 9



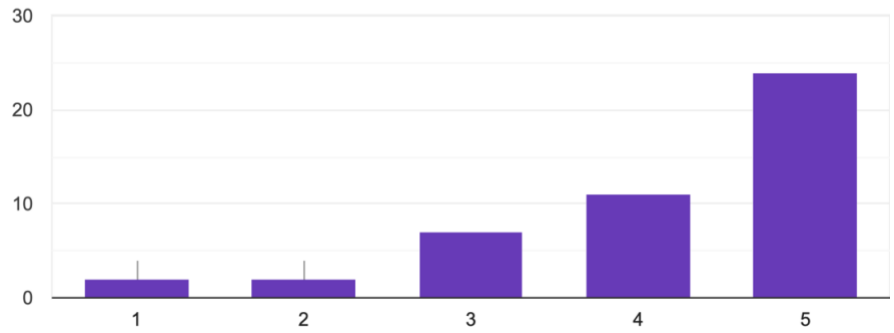
Pregunta 10



Pregunta 11

¿Crees que sería útil poder recibir ofertas, promociones o descuentos de los negocios que frecuentas?

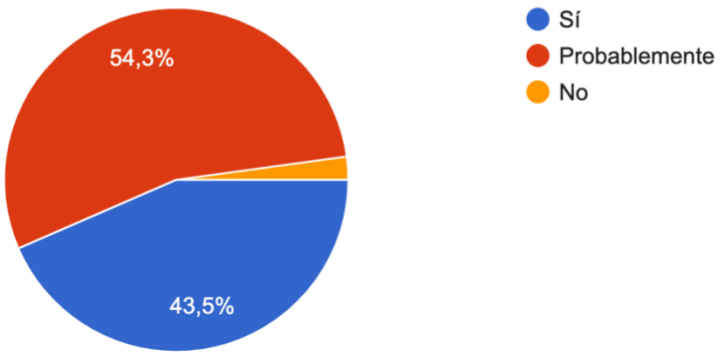
46 respuestas



Pregunta 12

¿Usarías la app para encontrar otros sitios a los que ir, ya sea por proximidad, precio u otras preferencias?

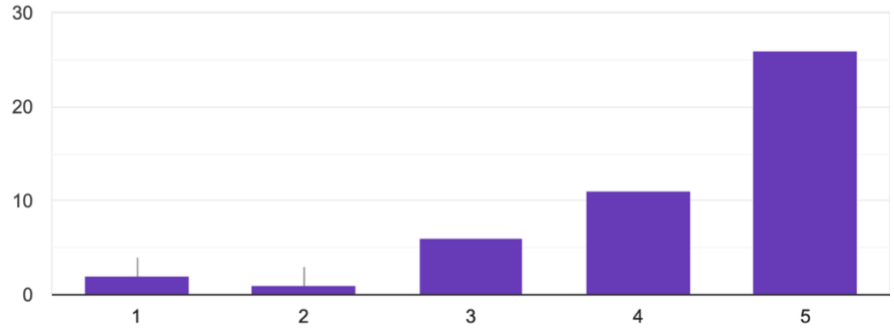
46 respuestas



Pregunta 13

En el caso de un negocio no estuviera en la app, ¿crees que sería útil poder apuntar en la app de forma manual la cita...das tus citas previas en un mismo sitio?

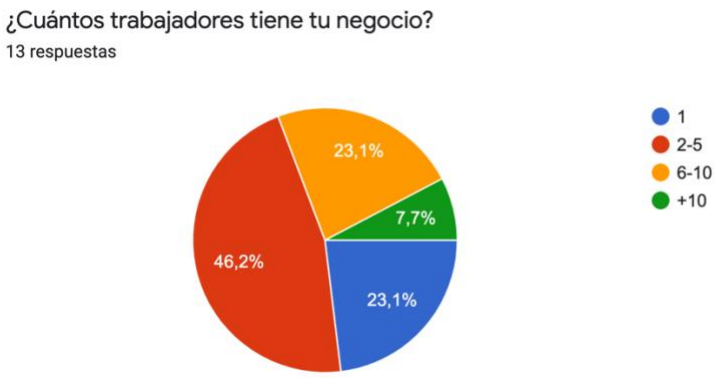
46 respuestas



Encuesta realizada a pequeños negocios

Esta encuesta ha sido realizada por un total de 12 personas que cumplen el perfil de pequeños negocios que podrían ser incluidos en la herramienta desarrollada en este proyecto.

Pregunta 1



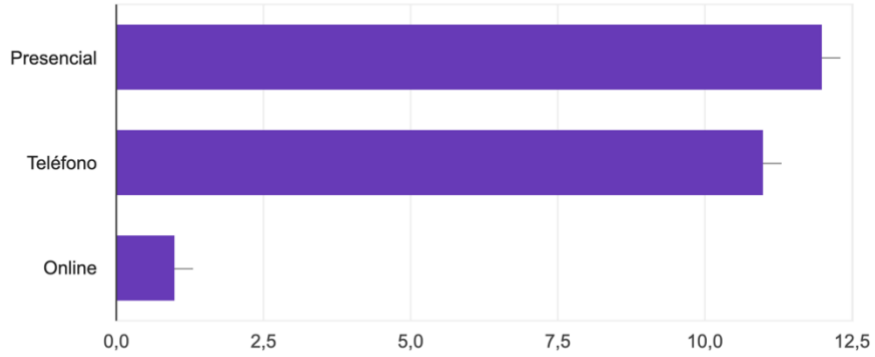
Pregunta 2

Pregunta 3

Pregunta 4

¿Qué métodos utilizáis para recibir citas previas?

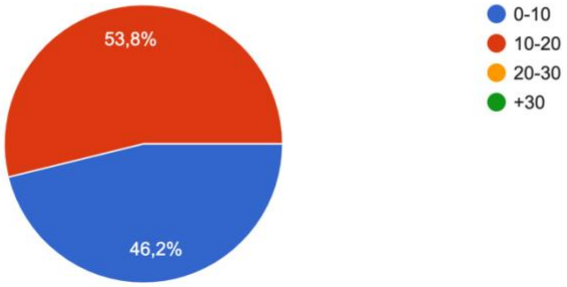
13 respuestas



Pregunta 5

¿Cuántas personas al mes no se presentan a las citas previas?

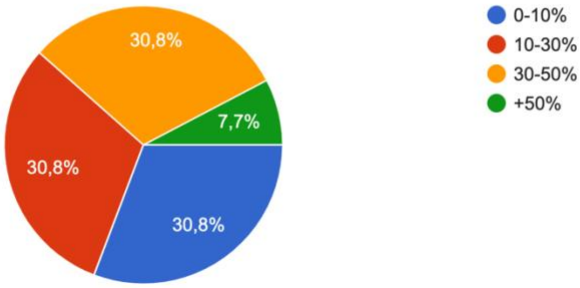
13 respuestas



Pregunta 6

De entre estas personas ¿cuántas no avisan?

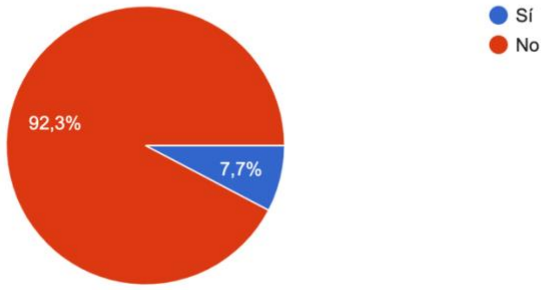
13 respuestas



Pregunta 7

¿Recoges datos y/o estudias el comportamiento de tus clientes?

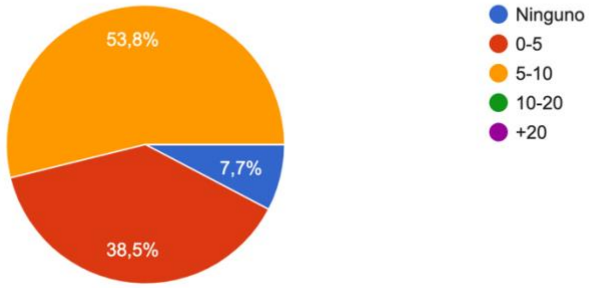
13 respuestas



Pregunta 8

Aproximadamente, ¿cuántos clientes nuevos tienes al mes?

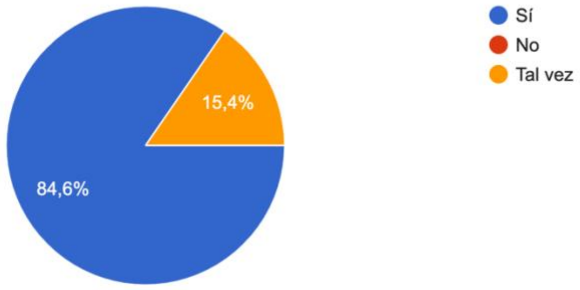
13 respuestas



Pregunta 9

¿Crees que te ayudaría un canal más directo para comunicarte periódicamente con tus clientes?

13 respuestas



Pregunta 10

En caso de en un futuro utilizar un gestor de citas previas, ¿preferirías pagar por uso o una tarifa fija?

13 respuestas