Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería de Software

Desarrollo de una aplicación web para la gestión de reservas en un centro de cuidado de mascotas

---

Alumna: Amanda Morera Pérez

Tutora: Yania Crespo González-Carvajal

...

Agradecimientos

A mi familia, en especial a mi madre y mi hermana que han sido un pilar fundamental, y a mis amigos por haberme apoyado durante toda esta etapa, incluso en los momentos más difíciles.

A los compañeros de carrera, tanto los que siguieron como los que se quedaron por el camino, por formar parte de este proceso.

A mi tutora Yania, por ayudarme a realizar este proyecto para poner el broche final a esta etapa. Gracias por tu esfuerzo y dedicación con todos nosotros.

Resumen

El proyecto consiste en el desarrollo de una aplicación web que permite a los usuarios realizar reservas de servicios, exclusivamente por horas, para sus mascotas, ofreciendo una solución para aquellos momentos en los que no puedan llevarlos consigo. Permite la posibilidad de solicitar servicios adicionales durante la estancia, adaptándose mejor a las necesidades individuales. Además, integra funcionalidades adicionales, como perfiles de mascotas e historial de reservas.

El proyecto se ha desarrollado utilizando Angular como framework para el frontend empleando TypeScript, Express.js para el backend empleando JavaScript junto con MongoDB como base de datos no relacional.

Abstract

This project involves the development of a web application that allows users to make reservations for pet services only for a few hours, bringing a solution for those moments when they can't take their pets with them. It allows the possibility to request additional services during their stay, better adapting to individual needs. In addition, there are additional functionalities integrated, such as pet profiles and reservation history.

This project has been developed using Angular as a framework for the frontend using TypeScript, Express.js for the backend and using JavaScript along with MongoDB as a non-relational database.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

En la actualidad, la mayoría de las personas conviven con una o varias mascotas en sus hogares, según el siguiente artículo [12], el porcentaje de personas de personas en España que conviven con una mascota ha aumentado hasta el 49 % y 8 de cada 10 personas las consideran un miembro más de la familia. Es por ello que cada vez se ve más a los dueños queriendo llevarlos consigo en las tareas cotidianas o a lugares de ocio como pueden ser centros comerciales con supermercados. Esto también ha provocado un impulso claro para este sector que se encuentra en expansión, en el 2023 la actividad económica de este sector llegó a alcanzar los 5,770 millones de euros.

Hoy en día, la disponibilidad de lugares donde poder dejar mascotas sólo por un número de horas es escasa, siendo lo más común estancias mínimas de una noche o por períodos más largos. Además, suelen estar ubicados a las afueras de la población por lo que obligan a disponer de un medio para llegar hasta allí.

Aunque actualmente se permite la entrada a los centros comerciales, en aquellos lugares que disponen de secciones de alimentación tienen prohibida la entrada por lo que si existe la necesidad de entrar al supermercado y hay dos personas como mínimo, deben turnarse para poder acceder ambos lo cuál puede resultar hasta incómodo. Si sólo hay una persona, no queda más remedio que prescindir de entrar.

Por todo esto surge la oportunidad de crear un negocio que ofrezca un servicio de cuidado de mascotas por horas ubicados en áreas urbanas. Este servicio daría solución al problema de no encontrar alternativas para dejar a las mascotas en un establecimiento seguro por unas horas mientras que los dueños realizan sus actividades diarias, evitando la incomodidad de tener que turnarse o la imposibilidad de realizarlas. Posteriormente, surge la necesidad de dar soporte tecnológico a este negocio y para ello crear una plataforma para poder gestionar estas reservas.

Este proyecto ha sido desarrollado en el contexto de la asignatura "Trabajo de Fin de Grado" en la mención de Ingeniería de Software del Grado en Ingeniería Informática [44].

1.2. Motivación

La idea para este proyecto surge, por parte del estudiante, debido a la necesidad propia de tener un lugar dónde poder dejar a sus mascotas durante unas horas en lo que realiza algunas tareas en lugares donde no los puede llevar consigo.

Para ello, se ha dado forma a la idea de un negocio que permita el cuidado de mascotas por horas, y se creará una aplicación web que permita a los usuarios poder registrar a sus mascotas y solicitar reservas para cada una de ellas, y al gerente o comercial poder gestionar dichas reservas para su establecimiento.

1.3. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación web que cumpla con las funcionalidades imprescindibles para ser un producto mínimo viable dentro del contexto del proyecto académico del que se trata. Este producto mínimo viable está pensado inicialmente para un único establecimiento pero con el propósito de que en un futuro, si crece y tiene éxito, pueda utilizarse para varios.

Los objetivos principales de la mencionada aplicación serían los siguientes:

- Permitir a los clientes gestionar sus mascotas.
- Permitir a los clientes solicitar una reserva para cada una de sus mascotas.
- Permitir al comercial o gerente gestionar las reservas solicitadas por el cliente.
- Permitir al comercial o gerente gestionar la información relativa al establecimiento a través de la aplicación web.

Por otro lado, la estudiante se plantea una objetivos académicos o personales:

- Aprender a utilizar una tecnología nueva, Node JS con Express y MongoDB, e integrarlo con Angular.
- Mejorar en la realización de tests unitarios con Jasmine.
- Investigar y aprender una arquitectura adecuada para el despliegue.
- Mejorar en la toma de decisiones para aplicar una arquitectura adecuada teniendo en cuenta las limitaciones de tiempo.

1.4. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: Este capítulo describe el contexto y de dónde surge la idea para realizar este proyecto, así como sus objetivos.

Capítulo 2 Requisitos y planificación: Este capítulo describe el marco de trabajo utilizado en el proyecto, los requisitos del mismo, el análisis de los riesgos, su presupuesto y la planificación.

Capítulo 3 Análisis Este capítulo describe los resultados de las tareas de análisis realizadas en diferentes momentos del proyecto a partir de los requisitos.

Capítulo 4 Tecnologías utilizadas Este capítulo describe las tecnologías utilizadas tanto para la gestión del proyecto, como para su desarrollo.

Capítulo 5 Diseño Este capítulo describe las decisiones de diseño realizadas en el desarrollo del proyecto, diseño del despliegue, los patrones arquitectónicos utilizados, principios y patrones de diseño, así como el diseño inicial de la interfaz de usuario.

Capítulo 6 Implementación y pruebas Este capítulo describe cómo se han implementado la funcionalidad más relevante y cómo se han desarrollado las pruebas.

Capítulo 7 Seguimiento del proyecto Este capítulo describe cómo se ha ido ejecutando la planificación del proyecto, los objetivos alcanzados o no, y los problemas que hayan podido surgir y cómo se han resuelto.

Capítulo 8 Conclusiones Este capítulo describe las conclusiones a las que se han llegado tras acabar el proyecto y una reflexión sobre las líneas de trabajo futuras.

Anexo A Manuales: Incluye manuales de instalación, despliegue, y de uso.

Anexo B Resumen de enlaces adicionales: Incluye enlaces de interés sobre el proyecto, como el repositorio de código y el enlace a la aplicación desplegada.

Capítulo 2

Requisitos y Planificación

En este capítulo se va a detallar el marco de trabajo elegido para este proyecto, el análisis de los riesgos realizado, la planificación realizada por Sprints del proyecto y la calendarización de los eventos del mismo, la elaboración de un presupuesto y se van a dejar recopilados los requisitos del mismo.

2.1. Scrum

Scrum es un marco de referencia para un proceso de desarrollo ágil. Es un proceso empírico, donde las decisiones se basan en la observación, la experiencia y la experimentación [41]. Scrum tiene tres pilares [26]:

- **Transparencia**: El proceso y el trabajo emergentes deben ser visibles para quienes realizan el trabajo, así como para quienes reciben el trabajo.
- **Inspección**: Los artefactos de Scrum y el progreso hacia los objetivos acordados deben inspeccionarse con frecuencia y diligencia para detectar variaciones o problemas potencialmente indeseables.
- **Adaptación**: Si algún aspecto de un proceso se desvía fuera de los límites aceptables o si el producto resultante es inaceptable, se debe ajustar el proceso que se está aplicando o los materiales que se están produciendo. El ajuste debe hacerse lo antes posible para minimizar una mayor desviación.

Los valores de Scrum también son fundamentales para que los equipos Scrum se adhieran, ya que ayudan a guiar la forma en que trabajas y a impulsar la confianza. Los valores [13] son:

- **Compromiso**: Cada miembro del equipo debe aceptar comprometerse a realizar tareas que puedan completar y no comprometerse en exceso.

- **Enfoque**: En el corazón del flujo de trabajo para los equipos de Scrum está el sprint, un período de tiempo enfocado y especificado en el que el equipo completa una cantidad determinada de trabajo.
- **Apertura**: La daily fomenta una apertura que permite a los equipos hablar abiertamente sobre el trabajo en progreso y los bloqueadores. A menudo hacemos que nuestros equipos se enfrenten a estas preguntas: ¿En qué trabajé ayer?, ¿en qué estoy trabajando hoy?, ¿qué problemas me están bloqueando?.
- **Respeto**: Celebrar los logros de los demás y ser respetuosos unos con otros, con el Product Owner, los stakeholders y el Scrum Master.
- **Coraje**: La valentía para cuestionar el status o cualquier cosa que obstaculice su capacidad de tener éxito.

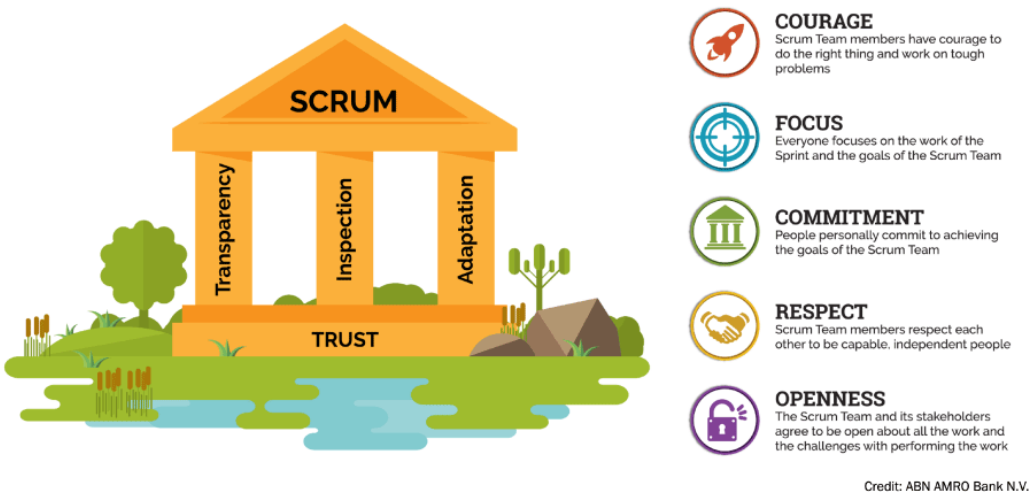


Figura 2.1: Valores de Scrum. Tomada de [41].

2.1.1. Equipo en Scrum

Un equipo de Scrum es un equipo pequeño y ágil dedicado a ofrecer incrementos de productos comprometidos. El tamaño de un equipo de Scrum suele ser pequeño, con alrededor de 10 personas, pero es lo suficientemente grande como para completar una cantidad sustancial de trabajo dentro de un Sprint. Un equipo de Scrum necesita tres roles específicos: Product Owner, Scrum Master y el equipo de desarrollo. Y debido a que los equipos de Scrum son multifuncionales, el equipo de desarrollo incluye testers, diseñadores, especialistas en UX, además de desarrolladores [13, 26].

- **Product Owner**: Responsable de maximizar el valor del producto resultante del trabajo del equipo. También es responsable de la gestión efectiva del Product Backlog,

que incluye: desarrollar y comunicar explícitamente el Product Goal, crear y comunicar claramente los items del Product Backlog, ordenar los items del Product Backlog y asegurar que el registro de productos sea transparente, visible y comprensible.

Figura 2.2: Product Owner. Tomada de [13].

- **Scrum Master**: Son verdaderos líderes que sirven al equipo Scrum y a la organización en general. Incluyendo: entrenar a los miembros del equipo en la autogestión y la funcionalidad cruzada, ayudar al equipo Scrum a centrarse en crear incrementos de alto valor que cumplan con Definition of Done, eliminar los impedimentos para el progreso del equipo Scrum, y asegurarse de que todos los eventos de Scrum tengan lugar y sean positivos, productivos y se mantengan dentro de la planificación.

Figura 2.3: Scrum Master. Tomada de [13]

- **Equipo de desarrollo**: Personas del equipo Scrum que se comprometen a crear cualquier aspecto de un Increment utilizable en cada Sprint.Los desarrolladores siempre son responsables de: creación de un plan para el Sprint, el Sprint Backlog, inculcar calidad adhiriéndose a una Definition of Done, adaptar su plan cada día hacia el objetivo de Sprint, y responsabilizarse mutuamente como profesionales.

Figura 2.4: Development Team. Tomada de [13].

2.1.2. Eventos en Scrum

Cada evento en Scrum es una oportunidad formal para inspeccionar y adaptar los artefactos de Scrum. Estos eventos están diseñados específicamente para permitir la transparencia requerida. La falta de operación de cualquier evento según lo prescrito da lugar a la pérdida de oportunidades para inspeccionar y adaptarse. Los eventos se utilizan en Scrum para crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum.[41]. Se distinguen los siguientes eventos:

- **Sprint**: es el latido del corazón de Scrum, donde las ideas se convierten en valor. El Sprint es el evento Scrum que abarca todos los demás eventos Scrum. Es el período de tiempo real en el que el equipo de scrum trabaja juntos para terminar un Increment. Dos semanas es una duración bastante típica para un sprint, aunque algunos equipos encuentran que es más fácil y se puede abarcar o en un mes se puede entregar un Increment valioso.
- **Sprint Planning**: El trabajo que se realizará (scope) durante el Sprint actual es planificado durante esta reunión por todo el equipo de desarrollo. Esta reunión está dirigida por el Scrum master y es donde el equipo decide el objetivo del Sprint. A continuación, se añaden historias de usuarios específicas al sprint desde el trabajo atrasado del Product Backlog.
- **Daily Scrum**: reunión diaria breve que tiene lugar a la misma hora (generalmente por las mañanas). Muchos equipos intentan que la duración de la reunión sea de unos 15 minutos, como mucho de 30. El objetivo de esta es que todos los miembros del equipo estén alineados y que tengan planificadas las tareas a realizar a lo largo de la jornada.
- **Sprint Review**: al final de cada sprint, el equipo se reune de manera informal para revisar el resultado del Sprint. Normalmente, para un Sprint con una duración de un mes esta reunión puede durar como máximo cuatro horas.
- **Sprint Retrospective**: se trata de una reunión que tiene lugar al final de cada Sprint, en la que el equipo se reúne para documentar y discutir que funcionó y que no funcionó en el sprint, en el propio proyecto, con las personas del equipo o incluso con las herramientas utilizadas.

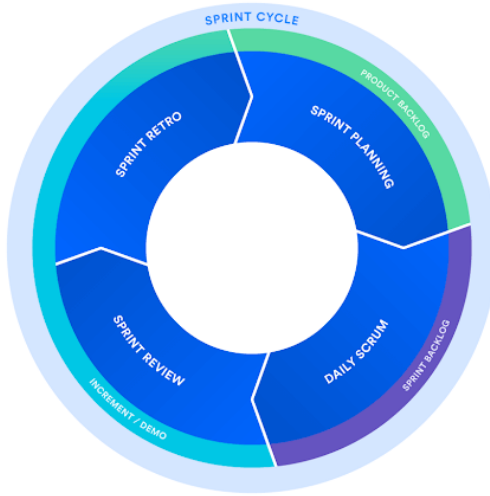


Figura 2.5: Scrum - Eventos y artefactos. Tomada de [13]

2.1.3. Artefactos en Scrum

- **Product Backlog**: es una lista ordenada de lo que se necesita para mejorar el producto. Los elementos de esta lista que pueden ser abordados dentro de un Sprint por el equipo se consideraran listos para seleccionar en la Sprint Planning. Los desarrolladores son responsables de controlar el tamaño de esta lista, y el Product Owner puede influenciar en los desarrolladores ayudandoles a comprender y seleccionar las compensaciones.
- **Sprint Backlog**: está compuesto por el Sprint Goal, los elementos del Product Backlog seleccionados para ese Sprint y un plan accionable para entregar el Increment. Es un plan de y para los desarrolladores. Es una imagen a tiempo real del trabajo que los desarrolladores planean realizar durante el Sprint para lograr el objetivo del Sprint. Por lo que, se actualiza a lo largo del Sprint a medida que se aprende.
- **Increment**: es un paso concreto hacia el Product Goal. Cada incremento se suma a los anteriores y está verificado, lo que garantiza que todos los incrementos funcionen juntos. Para proporcionar valor, el incremento debe ser utilizable. La suma de los incrementos se presenta en la Sprint Review. El trabajo realizado no puede considerarse parte de un incremento a menos que cumpla con Definition of Done.
- **Product Goal**: proporciona contexto del Product Backlog, describe un estado futuro del producto que sirve como objetivo para que el equipo de Scrum se planifique. Es el objetivo a largo plazo del equipo de Scrum. Es compromiso del Product Owner para el Product Backlog.
- **Sprint Goal**: es el objetivo único del Sprint. Se crea durante el Sprint Planning por

el equipo de Scrum y después se añade al Sprint Backlog. Es compromiso de los desarrolladores para el Sprint Backlog.

- **Definition of Done**: es una descripción formal del estado del Increment cuando cumple con las medidas de calidad requeridas para el producto. Una vez que se ha cumplido la Definition of Done, el Increment se hace y se puede entregar.

2.1.4. Scrum en el proyecto

El marco escogido para este proyecto es Scrum, definido y explicado en detalle en la secciones 2.1, 2.1.1, 2.1.2 y 2.1.3, aunque debido a las limitaciones del contexto y a las características del mismo se han tomado las decisiones detalladas a continuación para realizar una adaptación de Scrum al contexto del proyecto.

- Con respecto a los pilares y valores de Scrum se van a mantener tal y cómo se detallan en la Sección 2.1.
- En relación al equipo en Scrum, detallado en la Sección 2.1.1, la estudiante adoptará los papeles de Product Owner y del Equipo de desarrollo, y la tutora adoptará el papel de Scrum Master.
- En cuanto a los eventos, detallados en la Sección 2.1.2, los Sprints se mantendrán tal cuál con una duración de dos semanas. Las Sprint Planning, Sprint Review y Sprint Retrospective se mantendrán al final de cada Sprint por lo que se realizarán cada dos semanas. Por último, las Daily Scrum se han modificado y en lugar de realizarlas cada día se realizarán cada semana, pasando a denominarse Weekly Scrum.

2.1.5. Stakeholders y usuarios directos

Dadas las características del proyecto, entre los stakeholders se pueden distinguir tres tipos de usuarios directos dentro del mismo, A continuación se enumeran los stakeholders, comenzando por los usuarios directos:

- **Usuarios no registrados**: son personas que simplemente verán la información acerca del establecimiento y los servicios que ofrece.
- **Usuarios registrados**: son personas que realizarán reservas para sus mascotas.
- **Gerente**: es la persona encargada de gestionar las reservas realizadas por los usuarios registrados.
- **Equipo de desarrollo**: son personas encargadas de realizar la implementación y el desarrollo de la aplicación web.

2.2. Riesgos

Un riesgo se define como un evento o actividad que tiene la fuerza o el potencial de poner en peligro el éxito del desarrollo de un proyecto software. Es la posibilidad de experimentar pérdidas, y la exposición total al riesgo en un proyecto específico tendrá en cuenta tanto la probabilidad como la magnitud de la pérdida.

En los proyectos de desarrollo de software un riesgo puede ser de dos tipos: interno, está bajo control del manager del proyecto y externo, no está bajo el control del manager del proyecto.

La gestión de los riesgos se lleva a cabo realizando las siguientes actividades: identificación de los riesgos, análisis y priorización de los riesgos, planificación de los riesgos y monitorización de los riesgos.

A continuación se clasifican los riesgos de este proyecto en dos grupos: en primer lugar, los riesgos generales de cualquier proyecto en este contexto, apartado 2.2.1, y en segundo lugar, los riesgos particulares de este proyecto dadas las características del mismo y las condiciones particulares de la alumna, apartado 2.2.2.

Para calcular el nivel de riesgos de cada uno de los riesgos, se ha utilizado la siguiente matriz 2.1 en la que se calcula a partir de la probabilidad y el impacto.

PROB/IMP	BAJA	MEDIA	ALTA
BAJO	BAJO	BAJO	MEDIO
MEDIO	BAJO	MEDIO	ALTO
ALTO	MEDIO	ALTO	ALTO

Tabla 2.1: Matriz nivel de riesgo basada en la probabilidad y el impacto

2.2.1. Riesgos generales

En este apartado se detallan los riesgos generales dados en el proyecto, de la Tabla 2.2 a la Tabla 2.7.

2.2.2. Riesgos particulares

En este apartado se detallan los riesgos particulares encontrados, dadas las características propias del mismo, de la Tabla 2.8 a la Tabla 2.9.

Id	RSK001
Nombre	Equipo muy reducido
Descripción	Si el estudiante se pone enfermo un cierto período de tiempo en el desarrollo del TFG, no podría realizar tareas durante el mismo.
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Acciones de mitigación	Planificar el TFG con algún sprint de margen para esta situación. Mantener, al menos, durante este período de tiempo un estilo de vida saludable sin malos hábitos.
Acciones correctivas	Aumentar la carga de trabajo para intentar cumplir con las tareas planificadas en ese sprint, y si no utilizar los sprints de margen. Si fuera necesario, acudir al médico para que el tiempo de recuperación sea el menor posible

Tabla 2.2: - Riesgo 001

Id	RSK002
Nombre	Formación escasa
Descripción	Algunas de las tecnologías a utilizar no son conocidas o no se tienen la suficiente experiencia.
Probabilidad	Media
Impacto	Bajo
Nivel de riesgo	Bajo
Acciones de mitigación	Realizar una formación previa de aquellas tecnologías que no se dominen.
Acciones correctivas	Consultar toda la documentación necesaria sobre la tecnología en concreto. Consultar a personas con conocimientos extensos o mucha experiencia. Si fuera necesario, hacer uso de los sprints de margen.

Tabla 2.3: - Riesgo 002

2.3. Planificación

En la planificación inicial se ha establecido que la duración total del proyecto sea de diez Sprints. Cada uno de ellos con una duración de dos semanas, a excepción del Sprint

Id	RSK003
Nombre	Disponibilidad del equipo de trabajo
Descripción	Se pueden producir problemas con el equipo de trabajo en el que se está desarrollando, lo cuál imposibilita seguir con el desarrollo.
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Acciones de mitigación	Subir los cambios que se vayan realizando en el proyecto a algún repositorio en la nube. Disponer de algún equipo físico en backup
Acciones correctivas	Adquirir algún equipo físico para continuar con el desarrollo.

Tabla 2.4: - Riesgo 003

Id	RSK004
Nombre	Difícil comunicación con la tutora
Descripción	La tutora no dispone de tiempo para poder atender presencialmente.
Probabilidad	Baja
Impacto	Bajo
Nivel de riesgo	Bajo
Acciones de mitigación	Planificar con antelación suficiente las reuniones con la tutora. Disponer de medios alternativos para poder realizar las reuniones.
Acciones correctivas	Continuar con el desarrollo del proyecto.

Tabla 2.5: - Riesgo 004

0 o inicial, que tendrá una duración de doce días hasta dar por finalizadas las actividades anteriores al inicio de los Sprints y el último Sprint, que realmente no es un Sprint al uso sino que es margen para mitigar posibles retrasos en el proyecto y tendrá una posible duración de hasta diez días.

También se ha planificado que el tiempo invertido por semana sea aproximadamente de veinte horas, teniendo en cuenta el tiempo disponible de la alumna.

Este tiempo puede variar según los acontecimientos que tengan lugar o si tiene lugar algunos de los riegos expuestos en la Sección 2.2, que pueden condicionar a que aumente o disminuya el tiempo invertido. Se espera que el tiempo invertido en total por Sprint sea aproximadamente de cuarenta horas.

La Tabla 2.11 resume la calendarización inicial planificada para este proyecto, detallando las fechas en las que tienen lugar los eventos adaptados y explicados en la Sección 2.1.4.

Id	RSK005
Nombre	Cambios continuos en los requisitos
Descripción	Si estas modificaciones requieren un tiempo mayor al planificado podría provocar un reajuste, o incluso invalidar el trabajo realizado anteriormente.
Probabilidad	Media
Impacto	Medio
Nivel de riesgo	Medio
Acciones de mitigación	Documentar claramente los requisitos / objetivos y el alcance inicial.
Acciones correctivas	Aumentar la carga de trabajo para intentar cumplir con los cambios, y si no utilizar los sprints de margen.

Tabla 2.6: - Riesgo 005

Id	RSK006
Nombre	Estimación errónea
Descripción	Si no se realiza una gestión eficaz del tiempo, podría provocar un aumento del esfuerzo necesario con respecto a la estimación que se había realizado.
Probabilidad	Baja
Impacto	Medio
Nivel de riesgo	Bajo
Acciones de mitigación	Establecer una planificación del proyecto detallada con las fechas límite claras. Seguir estrictamente la planificación y gestión eficaz del tiempo.
Acciones correctivas	Aumentar la carga de trabajo para intentar cumplir con la planificación, y si no utilizar los sprints de margen.

Tabla 2.7: - Riesgo 006

Id	RSK007
Nombre	Compatibilidad prácticas
Descripción	Si se produce algún contratiempo y se necesita dedicar más tiempo en las prácticas extracurriculares, esto llevaría a invertir menos horas en el proyecto.
Probabilidad	Baja
Impacto	Medio
Nivel de riesgo	Bajo
Acciones de mitigación	Seguir la planificación y realizar una gestión eficaz del tiempo.
Acciones correctivas	Aumentar la carga de trabajo para intentar cumplir con la planificación, y si no utilizar los sprints de margen.

Tabla 2.8: - Riesgo 007

Id	RSK008
Nombre	Falta de conocimiento en la tecnología de despliegue
Descripción	Al ser la primera vez que se va a utilizar una tecnología de este tipo, será necesario invertir más tiempo y sobre todo al principio ya que se cometerán más errores.
Probabilidad	Media
Impacto	Medio
Nivel de riesgo	Medio
Acciones de mitigación	Realizar formación previa a la realización de las tareas. Seguir la planificación y realizar una gestión eficaz del tiempo.
Acciones correctivas	Aumentar la carga de trabajo para intentar cumplir con la planificación, y si no utilizar los sprints de margen.

Tabla 2.9: - Riesgo 008

Id	RSK009
Nombre	Falta de conocimiento en la tecnología para backend
Descripción	Al ser la primera vez que se va a utilizar una tecnología de este tipo, será necesario invertir más tiempo y sobre todo al principio ya que se cometerán más errores.
Probabilidad	Media
Impacto	Medio
Nivel de riesgo	Medio
Acciones de mitigación	Realizar formación previa a la realización de las tareas. Seguir la planificación y realizar una gestión eficaz del tiempo.
Acciones correctivas	Aumentar la carga de trabajo para intentar cumplir con la planificación, y si no utilizar los sprints de margen.

Tabla 2.10: - Riesgo 009

	Fecha inicio	Fecha fin
Sprint 0	04/02/2023	16/02/2023
Sprint 1	17/02/2023	01/03/2023
Sprint 2	02/03/2023	15/03/2023
Sprint 3	16/03/2023	29/03/2023
Sprint 4	30/03/2023	12/04/2023
Sprint 5	13/04/2023	26/04/2023
Sprint 6	27/04/2023	10/05/2023
Sprint 7	11/05/2023	24/05/2023
Sprint 8	25/05/2023	07/06/2023
Sprint 9	08/06/2023	17/06/2023

Tabla 2.11: Calendarización inicial por Sprints

Sprint 0	Fecha
Weekly Scrum	09/02/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	16/02/2023

Tabla 2.12: Calendarización Sprint 0

Sprint 1	Fecha
Weekly Scrum	23/02/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	01/03/2023

Tabla 2.13: Calendarización Sprint 1

Sprint 2	Fecha
Weekly Scrum	08/03/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	15/03/2023

Tabla 2.14: Calendarización Sprint 2

Sprint 3	Fecha
Weekly Scrum	22/03/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	27/03/2023

Tabla 2.15: Calendarización Sprint 3

Sprint 4	Fecha
Weekly Scrum	05/04/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	12/04/2023

Tabla 2.16: Calendarización Sprint 4

Sprint 5	Fecha
Weekly Scrum	19/04/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	26/04/2023

Tabla 2.17: Calendarización Sprint 5

Sprint 6	Fecha
Weekly Scrum	03/05/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	10/05/2023

Tabla 2.18: Calendarización Sprint 6

Sprint 7	Fecha
Weekly Scrum	17/05/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	24/05/2023

Tabla 2.19: Calendarización Sprint 7

Sprint 8	Fecha
Weekly Scrum	31/05/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	07/06/2023

Tabla 2.20: Calendarización Sprint 8

Sprint 9	Fecha
Weekly Scrum	14/06/2023
Weekly Scrum Sprint Planning Sprint Review Sprint Retrospective	17/06/2023

Tabla 2.21: Calendarización Sprint 9

2.4. Presupuesto

En este apartado se explicarán dos tipos diferentes de presupuestos para este proyecto. En primer lugar, un presupuesto simulado, apartado 2.4.1, que se daría en un ámbito más profesional. En segundo lugar, un presupuesto real, que sería el que realmente ha tenido este proyecto con sus circunstancias reales, apartado 2.4.2.

2.4.1. Presupuesto simulado

Para realizar esta estimación de los costes internos de desarrollo del proyecto, es decir, no será el que finalmente sería entregado al cliente, en primer lugar hay que tener en cuenta el sueldo del empleado. En este caso, se trataría del sueldo de un Full Stack Junior Developer que está de media en 22.000€ [19], lo que supone una media de 1833,33€ al mes. Teniendo en cuenta que la cotización en la seguridad social es de un 23 %, supondría a mayores un gasto de 421,67€ al mes, y finalmente 2.255€ al mes. Si hacemos el cálculo del gasto que supone por hora ya que este sueldo es para 40 horas semanales,y serían 160 horas al mes, saldría a 14,09€ la hora.

Trasladando lo anterior al proyecto, el cuál está planificado para 10 sprints lo que supone una duración total de 300 horas, se obtiene un gasto final de 4227€.

Otro aspecto a considerar sería el lugar de trabajo que necesitará el empleado para cumplir con un mínimo del 30 % de presencialidad. Para ello se alquilará una mesa fija en un espacio de trabajo compartido, ya que dispone de lo necesario para poder realizar las tareas. El precio de este es de 100 € al mes. Por lo que si lo necesitaremos durante unos cinco meses, supondría un gasto de 500 €.

Además, necesitamos tener en cuenta el equipo que usará el empleado. Para ello, se le proporcionará un Dell Latitude 5430 con 16 GB de RAM, un i5-1335U y 512GB SSD, cuyo coste es de 1158,92€ [35], lo que supone un gasto mensual de 231,79 €. Y un monitor Dell P2422H de 23.8", cuyo coste es de 167,98€, lo que supone un gasto mensual de 33,6€ [36].

A continuación, habría que añadir al presupuesto el coste de las licencias, programas y servicios que se vayan a utilizar. En primer lugar, se utilizará un paquete básico de Microsoft 365 básico, que tendrá un coste de 7,40 € al mes [1]. Por lo que el coste sería de 37 €. Lo siguiente será la licencia de Astah Professional, que tiene un coste mensual de 8,99 € [37], por lo que el coste sería de 44,95 €. Como entorno de desarrollo se usará Visual Studio Code, cuya licencia es gratuita. Por último, para los servicios usaremos Amazon Web Services, se detallarán a continuación: Amazon Simple Storage Service(S3), para disponer de un servicio de almacenamiento y cuyo coste inicial es de 0,065€ y mensualmente de 2,40€, y finalmente un coste de 7,465€; Amazon API Gateway, para disponer de un servicio de solicitudes API REST y cuyo coste mensual es de 151,14€, y finalmente un coste de 755,7€; Amazon DynamoDB, para disponer de una base de datos, cuyo coste inicial es de 188,03€ y cuyo coste mensual es de 31,34€, y finalmente un coste de 344,73€; y por último Amazon EC2, para disponer de máquinas virtuales dónde poder ejecutar la aplicación, y cuyo coste mensual es de 16,84€, y finalmente un coste de 84,2€.

A todo esto, se le añadirá un incremento del 20 % y así tener una reserva disponible para que en el caso de que algún riesgo se materialice poder hacerle frente. En Tabla 2.22 se muestra un resumen con todo lo descrito anteriormente y el cálculo del presupuesto final.

	Gasto / mes	Gasto total
Sueldo	845,4€	4227€
Oficina	100€	500€
Equipo empleado	265,39€	1326,9€
Licencias, servicios	218,11€	1274,05€
	Gasto total	7.327,95€
	Incremento por riesgos	1.465,59€
	Gasto total + incremento	8.793,54€

Tabla 2.22: Presupuesto simulado de los costes de desarrollo.

2.4.2. Presupuesto en el contexto real

En este caso, dado el contexto real de este proyecto, mucho de los costes descritos en el apartado anterior desaparecerán ya que no se le paga ningún sueldo a la alumna, no necesita alquilar ningún espacio de trabajo ya que lo realizará mayoritariamente en el domicilio y algunas de las licencias son gratuitas al tratarse de licencias académicas.

Lo primero a tener en cuenta es el equipo de trabajo del que dispone la alumna, se trata de un iMac Chip M1 2021 8GB RAM y 256GB SSD [22] lo que supone un gasto de 1799,99 €, y sería de 360 €, al mes y de un Acer Aspire A315-56, i5 1035G1, 12GB, 512GB SSD [11] lo que supone un gasto de 499 € y sería 99,8 €, al mes.

Lo siguiente que tendríamos que tener en cuenta son los gastos propios que tendrá la alumna a la hora de realizar el trabajo en el domicilio. En este caso tendría gastos de luz principalmente ya que es necesario para poner el equipo en funcionamiento, lo siguiente es el gasto en disponer de una conexión Wifi y por último el gasto relativo al gas que necesitará para utilizar la calefacción ya que la mitad del proyecto se realizará en meses de invierno.

Para calcular el gasto aproximado relativo al consumo eléctrico que se tendrá durante el desarrollo del proyecto, se tendrá en cuenta que un Mac consume cuando está inactivo 40W y en un nivel máximo de CPU consume 70 W [3] y se tomará un valor medio en este caso sería 55W. Si tenemos en cuenta el precio actual del kwH en España, de media 0,07936 €/kWh [43] y que el tiempo que se va a emplear para el proyecto son 400 horas, consumirá unos 22 kWh lo que supone un gasto de 1,75 €. Si le sumamos el gasto medio del portátil [39] que es de 16kWh al mes, supondría un gasto total de 6,35 €. A continuación, teniendo en cuenta que el gasto por mes relativo a Internet es aproximadamente de 20 € al mes, supondría un gasto total de 100 €.

En cuanto al gasto relativo al gas consumido para utilizar la calefacción [42], teniendo en cuenta que se utilizará una tarifa en el mercado regulado cuya tarifa es 3,85 €, al mes relativo al término fijo y 0.0511 € al mes relativo al término variable. Si de media se tiene un gasto al mes de 30 €, supondría un gasto de 0,042 €, a la hora lo que finalmente supondría un gasto en el proyecto, teniendo en cuenta que se utilizará unos dos meses aproximadamente, de 60 € .

En último lugar, hay que tener en cuenta el gasto que supone para la alumna el desplazamiento hasta el centro académico donde se realizan las Weekly Scrum, Sprint Review, Sprint Planning y Sprint Retrospective. Teniendo en cuenta que el billete ida y vuelta de Palencia hasta Valladolid en autobús [38] cuesta 8,45 €, y que se realizará un viaje a la semana supondría un gasto de 33,8 €, al mes y en total un gasto de 169 €.

En la siguiente tabla 2.23, se mostrará un resumen con todo lo descrito anteriormente y el cálculo del presupuesto final.

	Gasto / mes	Gasto total
Equipo empleado	459,8€	2298,99€
Luz	8,1€	40,5€
Internet	20€	100€
Gas	30€	60€
Desplazamiento	33,8	169€
	Gasto total	2.668,49€

Tabla 2.23: Presupuesto de los costes de desarrollo en el contexto real.

2.5. Épicas para Product Backlog Inicial

Siguiendo el marco de trabajo en el que se ha basado este proyecto, los requisitos se expresan o se detallan como historias de usuario de la siguiente forma : "Como **stakeholder1** quiero **x** para **y**".
En este apartado se describe el Product Backlog Inicial, en el que se detallan las épicas del proyecto, tal y cómo se puede ver en la Tabla 2.24. Las épicas son historias de usuario de mayor tamaño y que no pueden abordarse en una sola sino que necesitan descomponerse posteriormente en varias historias de usuario.

ID	Descripción
E1	Como cliente no registrado quiero poder ver la información pública de la empresa, así como poder registrarme en la plataforma.
E2	Como cliente registrado, quiero gestionar mi cuenta y la información almacenada en ella.
E3	Como cliente registrado, quiero gestionar las mascotas vinculadas a mi perfil, y la información almacenada sobre ellas.
E4	Como cliente registrado, quiero gestionar las solicitudes de reserva para cada una de mis mascotas.
E5	Como gerente, quiero gestionar las reservas pendientes realizadas por los clientes y la información sobre los establecimientos.

Tabla 2.24: Épicas para Product Backlog Inicial

2.6. Requisitos

Tal y cómo se ha detallado en el apartado anterior, los requisitos se expresan como historias de usuario y estas se obtienen de la descomposición de las épicas del Product Backlog Inicial 2.24 en historias de usuario más pequeñas que se puedan llevar a cabo y completar en un Sprint.

2.6.1. Requisitos funcionales expresados como historias de usuario

Una vez explicado como vamos a obtener los requisitos funcionales, lo siguiente sería cómo se va a estimar el esfuerzo requerido para completar las historias de usuario. Se utilizará como métrica los puntos de historia, que son una forma de calcular la cantidad de esfuerzo necesario para completar una historia de usuario en la lista de trabajo pendiente del producto, además se va a utilizar el Planning Poker, de normal se hace en un equipo con mínimo 3 desarrolladores ya que cada uno expondrá la estimación según su opinión y luego llegarán a un consenso aunque en este caso lo realice un único integrante, tomando como referencia la escala de Fibonacci [5], de menos a más complejidad: 0, 0.5, 1, 2, 3, 5, 8, 13 (como máximo).

Se utilizará la siguiente nomenclatura para definir los requisitos funcionales: `HU_IDÉPICA_IDHU`, siendo `HU` las iniciales de historia de usuario, `IDÉPICA` el ID correspondiente a la épica de la que se ha obtenido e `IDHU`, el propio ID del requisito funcional expresado como historia de usuario.

A continuación, se muestran como quedarían las historias de usuario tras la descomposición de las épicas del Product Backlog Inicial, de la Tabla 2.25 a 2.29.

ID	Descripción	Estimación
HU_01_01	Como cliente no registrado quiero poder ver información sobre la empresa y los servicios que ofrece.	2
HU_01_02	Como cliente no registrado quiero poder ver información acerca de la ubicación de la empresa.	2
HU_01_03	Como cliente no registrado quiero poder ver información de contacto de la empresa.	2
HU_01_04	Como cliente no registrado quiero poder registrarme en la plataforma.	3
HU_01_05	Como cliente no registrado quiero poder ver información acerca de las tarifas que ofrece la empresa.	3

Tabla 2.25: Historias de usuario - Épica 1

ID	Descripción	Estimación
HU_02_01	Como cliente registrado quiero poder iniciar sesión en la plataforma.	8
HU_02_02	Como cliente registrado quiero poder cerrar sesión en la plataforma.	2
HU_02_03	Como cliente registrado quiero poder ver la información detallada de mi perfil.	2
HU_02_04	Como cliente registrado quiero poder editar la información detallada de mi perfil.	3
HU_02_05	Como cliente registrado quiero poder eliminar mi cuenta.	3

Tabla 2.26: Historias de usuario - Épica 2

ID	Descripción	Estimación
HU_03_01	Como cliente registrado quiero poder ver todas las mascotas asociadas a mi perfil.	2
HU_03_02	Como cliente registrado quiero poder añadir una nueva mascota a mi perfil.	5
HU_03_03	Como cliente registrado quiero poder ver la información detallada de cada una de mis mascotas.	2
HU_03_04	Como cliente registrado quiero poder editar la información detallada de mi mascota.	3
HU_03_05	Como cliente registrado quiero poder eliminar una mascota asociada a mi perfil.	3
HU_03_06	Como cliente registrado quiero poder asociar mis mascotas a otro dueño.	5

Tabla 2.27: Historias de usuario - Épica 3

ID	Descripción	Estimación
HU_04_01	Como cliente registrado quiero poder solicitar una nueva reserva.	5
HU_04_02	Como cliente registrado quiero poder ver todas las reservas asociadas a mi perfil.	3
HU_04_03	Como cliente registrado quiero poder ver la información detallada de cada reserva asociada a mi perfil.	2
HU_04_04	Como cliente registrado quiero poder cancelar una reserva asociada a mi perfil.	3
HU_04_05	Como cliente registrado quiero poder filtrar las reservas según su estado.	3
HU_04_06	Como cliente registrado quiero poder filtrar las reservas según su fecha, de más reciente a menos reciente.	3

Tabla 2.28: Historias de usuario - Épica 4

ID	Descripción	Estimación
HU_05_01	Como gerente quiero poder iniciar sesión en la plataforma.	8
HU_05_02	Como gerente quiero poder cerrar sesión en la plataforma.	2
HU_05_03	Como gerente quiero poder ver todas las reservas pendientes.	3
HU_05_04	Como gerente quiero poder confirmar o cancelar cada reserva pendiente.	5
HU_05_05	Como gerente quiero poder ver todas las reservas confirmadas.	3
HU_05_06	Como gerente quiero poder modificar los datos de un establecimiento.	3
HU_05_07	Como gerente quiero poder modificar las tarifas de un establecimiento.	3

Tabla 2.29: Historias de usuario - Épica 5

2.6.2. Requisitos no funcionales expresados como historias de usuario

En este apartado se detallarán los requisitos no funcionales expresados como historias de usuario siguiendo la siguiente nomenclatura, análoga a la del apartado previo: `HUNFIDHU`, siendo `HUNF` las iniciales de historia de usuario no funcional e `IDHU`, el propio ID del requisito no funcional expresado como historia de usuario. Se definen en la siguiente Tabla 2.30:

Id	Descripción
HUNF01	Como usuario quiero que la interfaz de la aplicación sea intuitiva y fácil de usar podiendo realizar las funciones principales con un porcentaje de error inferior al 5 %.
HUNF02	Como usuario quiero que se utilicen medidas de seguridad para proteger la información sensible.
HUNF03	Como usuario quiero que la aplicación tenga un tiempo de respuesta en los casos más importantes inferior a cinco segundos.
HUNF04	Como usuario quiero que la aplicación este disponible de lunes a viernes durante 18 horas diarias, y durante los fines de semana, 24 horas.
HUNF05	Como ingeniero quiero que la aplicación se pueda usar desde cualquier navegador, utilizando en esta primera versión el navegador Chrome, siendo responsive para los siguientes tamaños de pantala: .
HUNF06	Como ingeniero quiero la documentación sea clara y completa, incluyendo manuales de usuario.
HUNF07	Como ingeniero quiero que la aplicación sea escalable cuando se produzca un incremento en el número de usuarios o de establecimientos.

Tabla 2.30: Historias de usuario - Requisitos no funcionales

2.6.3. Requisitos de información expresados como historias de usuario

En este apartado se describen los requisitos de información expresados como historias de usuario, siguiendo la siguiente la siguiente nomenclatura, también análoga a la de los dos apartados anteriores: `HUIDHU`, siendo `HUI` las iniciales de historia de usuario de información `IDHU`, el propio ID del requisito de información expresado como historia de usuario. Se definen en la siguiente tabla 2.31:

Id	Descripción
HUI01	Como gerente quiero almacenar la información sobre un establecimiento: nombre, dirección, teléfono de contacto, correo electrónico de contacto y sus tarifas asociadas.
HUI02	Como gerente quiero almacenar la información sobre un usuario: rol, alias, nombre, apellidos, correo electrónico, teléfono y contraseña.
HUI03	Como gerente quiero almacenar la información sobre una mascota: tipo, nombre, edad, raza, tamaño, alimentación y si tiene alguna enfermedad.
HUI04	Como gerente quiero almacenar la información sobre una reserva: fecha, estado, horas contratadas, extras solicitados y precio.

Tabla 2.31: Historias de usuario - Requisitos de información

Capítulo 3

Análisis

En este capítulo se va a detallar los modelos resultantes de las actividades de análisis realizadas mayoritariamente en el Sprint inicial. Estos modelos se han ido refinando en los Sprints posteriores a medida que se iba desarrollando la aplicación.

3.1. Modelo de dominio inicial

En la Figura 3.1, se puede ver el resultado del modelo de dominio inicial realizado a partir de los requisitos, dónde se puede ver las relaciones entre las clases y los atributos que las caracterizan. Este modelo se ha ido refinando durante los sprints siguientes hasta obtener este resultado.

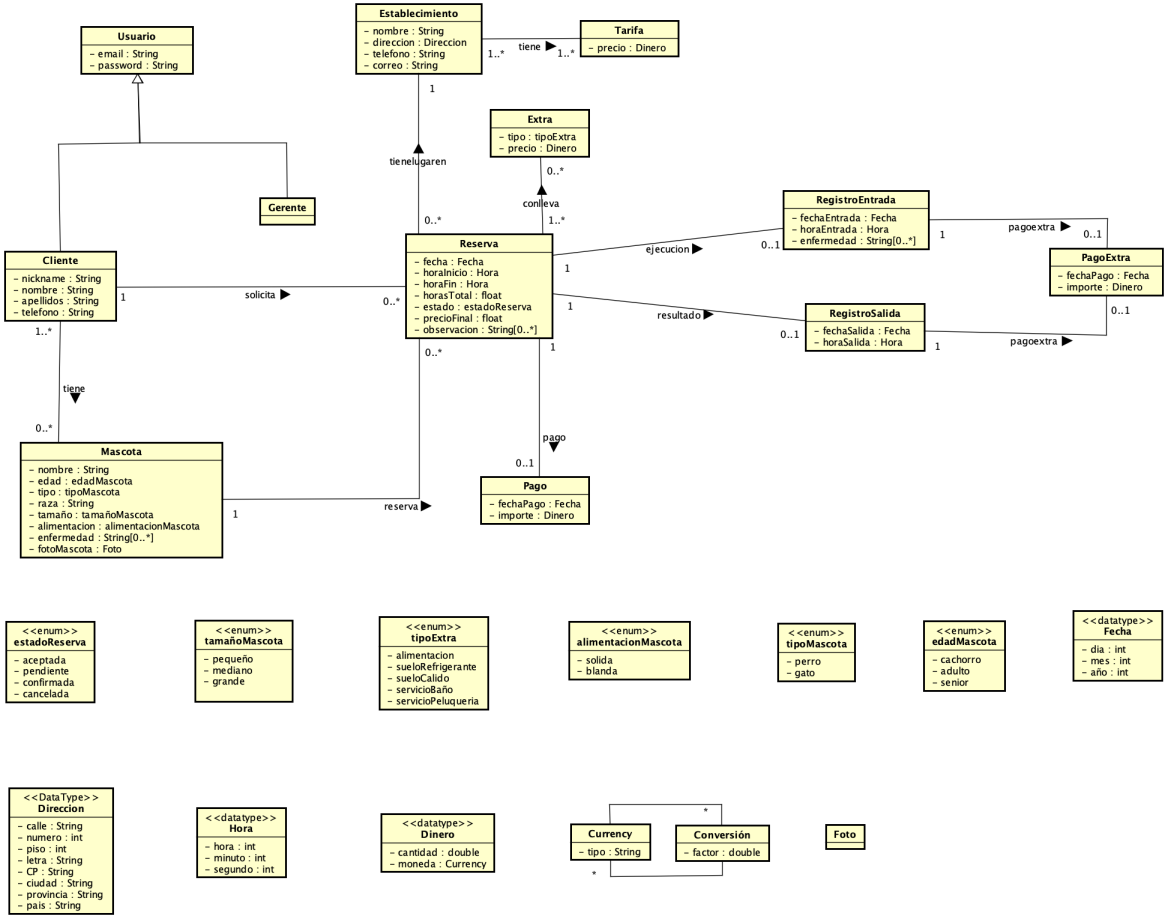


Figura 3.1: Modelo de dominio inicial

3.2. Modelo de proceso de negocio

En la Figuras 3.2 y 3.3 se puede ver el modelo de proceso de negocio representado por estos dos diagramas de actividades. En el primer caso, se ve todo el proceso detallado desde que un usuario solicita una reserva hasta que se realiza el pago y se confirma o, por el contrario, se cancela. En el segundo caso, se muestra todo el proceso cuando se añade un extra a la reserva en el día de la reserva hasta que finalmente se acepta y se modifica la reserva.

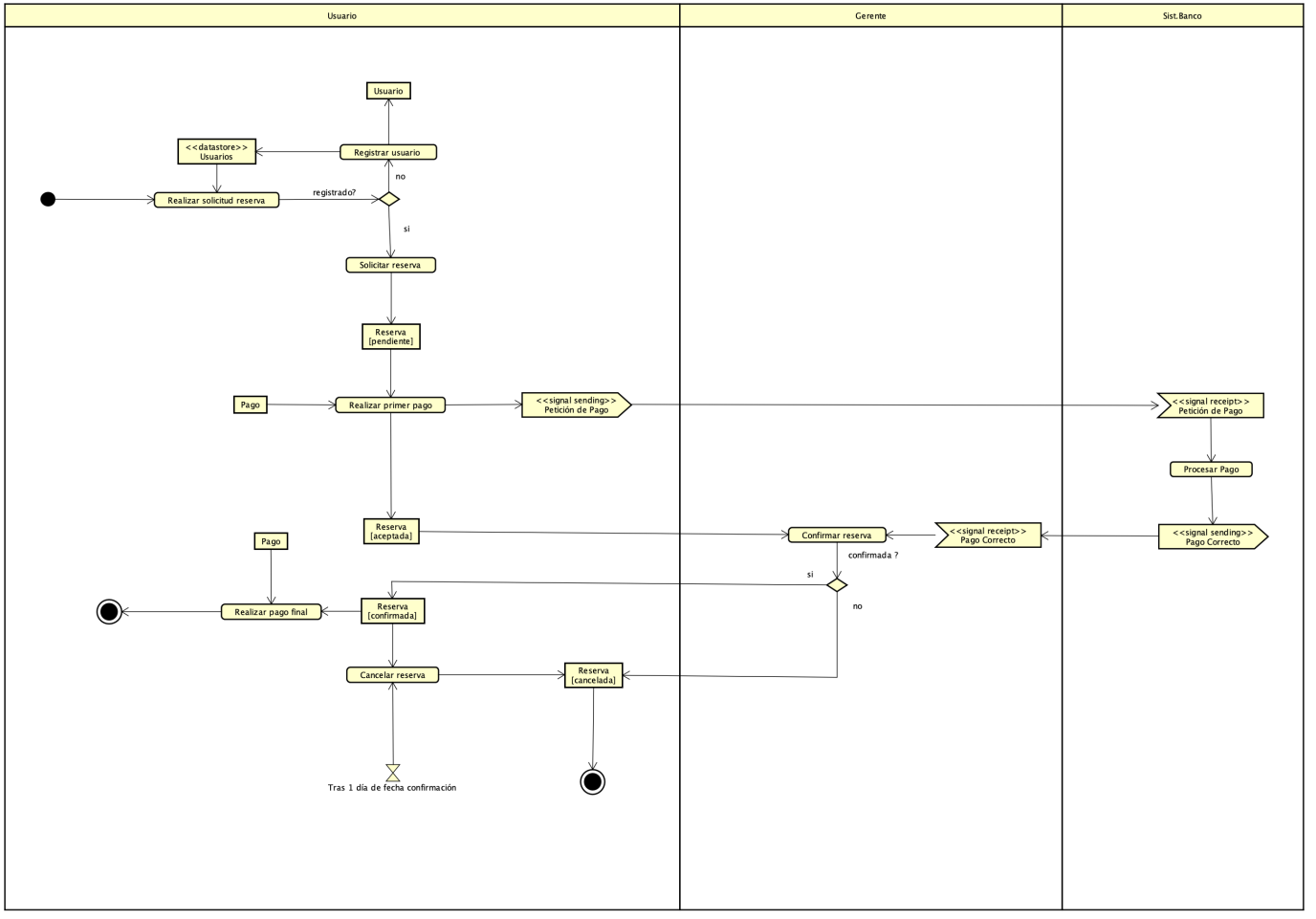


Figura 3.2: Modelo de proceso del negocio (1), el proceso de una reserva

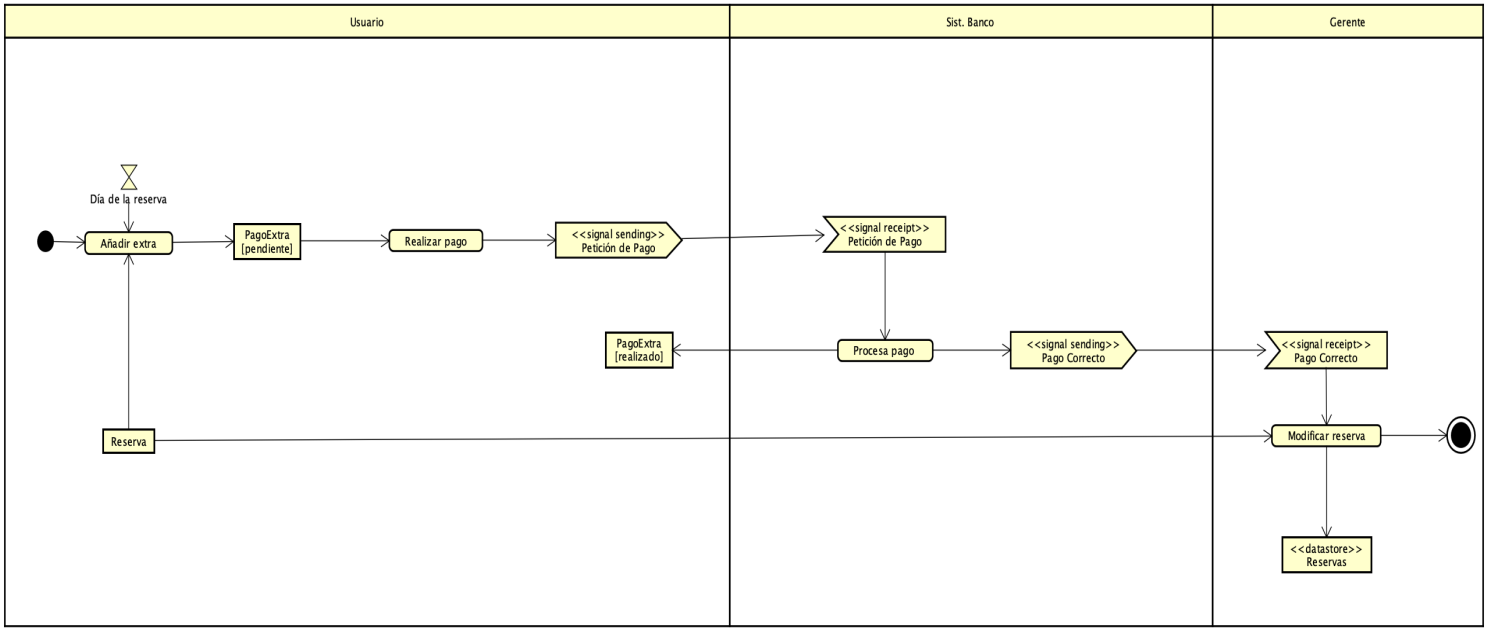


Figura 3.3: Modelo de proceso del negocio (2), el proceso de añadir servicios extras a una reserva.

3.3. Análisis de objetos que pasan por diferentes estados

En esta sección se va a realizar un análisis de aquellos objetos que, a partir de los modelos de proceso anteriores, se analiza que cambian de estado por lo que se considera estudiarlos cómo máquinas de estados.

En el modelo del proceso de negocio que se muestra en la Figura 3.2, se ve que el único objeto que va cambiando de estado es la reserva. En la Figura 3.4, se analiza la reserva como una máquinas de estados, resumiendo los estados por los que pasa la reserva y los eventos que provocan las transiciones de un estado a otro.

En el modelo de proceso que se muestra en la Figura 3.3, se observa cómo el objeto que cambia de estado es el pago extra que se realiza el día de la reserva. En la Figura 3.5, se analiza los estados por los que pasa el pago extra y los eventos que provocan las transiciones.

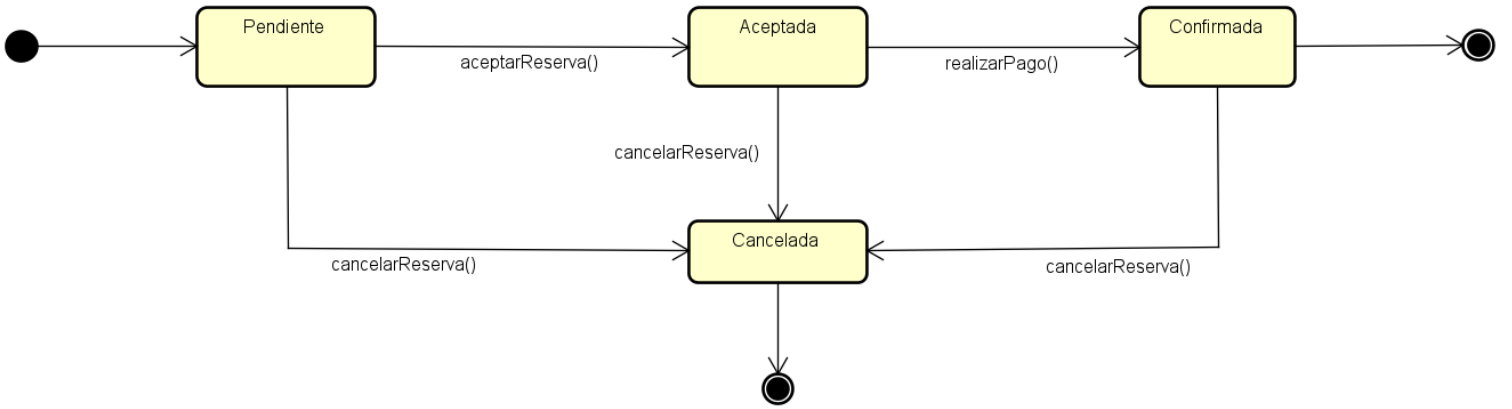


Figura 3.4: Estados de una reserva

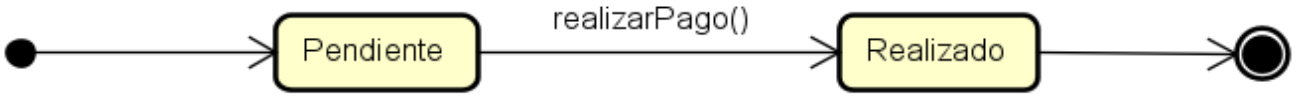


Figura 3.5: Estados de un pago extra

Capítulo 4

Tecnologías utilizadas

En este capítulo se presenta un resumen de las tecnologías utilizadas tanto para la gestión y control del proyecto, para el análisis y documentación del proyecto, para el desarrollo del proyecto como para la comunicación.

4.1. Tecnologías para el análisis y la documentación del proyecto

4.1.1. Astah Professional

Astah [6] es un software de realización de diagramas muy potente que permite la realización de diagramas UML, ER, de flujo y de flujo de datos. Permite la realización fácilmente de diagramas para visualizar sistemas complejos y de manera consistente y permite el desarrollo de una explicación clara del diseño software para el equipo.

En este caso se ha utilizado para la elaboración de los diagramas realizados en los capítulos de análisis y diseño.

4.1.2. Overleaf

Overleaf [45] es un editor LaTeX colaborativo basado en la nube que se utiliza para escribir, editar y publicar documentos. Esta herramienta permite realizar cambios en tiempo real, mantener un historial con los últimos cambios realizándose incluso realizar comentarios por las personas con las que se haya compartido.

En este caso se ha utilizado para la redacción de este documento.

4.1.3. Figma

Figma [47] es un editor de gráficos vectorial y una herramienta de generación de prototipos, principalmente basada en la web. El conjunto de funciones de Figma se enfoca en el uso de la interfaz de usuario y el diseño de experiencia de usuario, con énfasis en la colaboración en tiempo real.

En este caso se ha utilizado para la realización del diseño de interfaz de usuario inicial, en el capítulo de diseño.

4.2. Tecnologías para la gestión y control del proyecto

4.2.1. Jira Atlassian

Jira [7] es la herramienta de gestión de proyectos *Agile* más usada por los equipos para planificar, seguir, lanzar y dar soporte al software. Ofrece las herramientas necesarias para estimar, informar y medir la velocidad con flujos de trabajo diseñados para poder adaptarse. Proporciona tableros personalizables, para administrar el trabajo y líneas de tiempo para planificar y realizar un seguimiento del trabajo para garantizar que el equipo se mantenga sincronizado.

En el proyecto se han utilizado los siguientes elementos que proporciona Jira:

- Cronograma: proporciona una línea del tiempo para tener una visión general de las épicas que se han ido desarrollando en cada sprint, se puede ver por días, semanas, meses y trimestres.
- Backlog: muestra el listado general, desde el Sprint actual hasta el último, de las tareas e historias de usuario definidas para cada Sprint, su estimación, su estado y la persona a la que se le asigna. Permite ver el detalle de cada tarea donde, entre otras opciones, permite imputar el número de horas que se han empleado para realizarla.
- Tablero: sirve para tener una visión más general del Sprint actual, ya que proporciona tres columnas en las que agrupan las tareas e historias de usuario según su estado: por hacer, en curso o finalizado.
- Incidencias: muestra la lista de todas las tareas, historias de usuario y épicas que se han definido en todos los sprints, permite aplicar diferentes filtros a las mismas: según el tipo, el estado, la persona a la que se le haya asignado e incluso personalizados.

4.2.2. Google Calendar

Google Calendar [49] es una agenda y calendario electrónico desarrollado por Google. Permite sincronizarlo con los contactos de Gmail de manera que podamos invitarlos y compartir eventos.

En este proyecto se ha utilizado para que la alumna guardase los eventos en dicho calendario y le avisase siempre con antelación previa al mismo.

4.2.3. Git

Git [8] es un sistema de control de versiones muy utilizado ya que es compatible con una gran variedad de sistemas operativos y Entornos de Desarrollo Integrados (IDEs). Es un ejemplo de sistema de control de versiones distribuido (DVCS, por sus siglas en inglés). En lugar de tener un único espacio para todo el historial de versiones del software, la copia de trabajo del código de cada desarrollador es un repositorio local que alberga el historial completo de todos los cambios.

Una de las mayores ventajas que presenta es su capacidad para ramificación, ya que facilita el flujo de trabajo. Estas ramas proporcionan un entorno de trabajo aislado para cada cambio que se quiera realizar. Para ello se crea una nueva rama en la que trabajar y así se asegura de no modificar nada en la rama principal.

Los comandos de Git que se han utilizado para este proyecto han sido los siguientes:

- `git init`: al ejecutar este comando, se crea un subdirectorio .git en el directorio actual de trabajo, que contiene todos los metadatos necesarios para el nuevo repositorio. Estos metadatos incluyen subdirectorios de objetos, referencias y archivos de plantilla. También se genera un archivo HEAD que apunta a la confirmación actualmente extraída.

- `git commit`: captura una instantánea de los cambios preparados en ese momento del proyecto. Las instantáneas confirmadas pueden considerarse como versiones "seguras" de un proyecto, capturan el estado de un proyecto en ese determinado momento y siempre se confirman en el repositorio local. Al ejecutar este comando se le pueden añadir las opciones -a, que confirma una instantánea de todos los cambios del directorio de trabajo, solo incluye las modificaciones a los archivos con seguimiento. La opción -m, que crea inmediatamente una confirmación con un mensaje de confirmación usado. Finalmente, la opción que se ha usado en el proyecto ha sido la combinación de ambas, -am que crea inmediatamente una confirmación de todos los cambios preparados y aplica un mensaje de confirmación insertado.

- `git add`: Antes de ejecutar `git commit`, se utiliza este comando para pasar o "preparar" los cambios en el proyecto que se almacenarán en una confirmación.

- `git push <remote> <branch>`: se usa sobre todo para publicar y cargar cambios locales a un repositorio central. Después de modificar el repositorio local, se ejecuta un envío para compartir las modificaciones con los miembros remotos del equipo. En este caso no ha sido necesario ya que sólo hay una persona en el equipo de desarrollo, pero en el caso de proyecto dónde hay más de una persona, antes de hacer un `git push` se debe de hacer un `git pull`, después de haber hecho `git commit`, para bajar los cambios que hubiese en el repositorio remoto. También se puede utilizar `git fetch`

antes del `git pull` que lista todas las ramas y los cambios que no se hayan bajado del repositorio.

- `git checkout <branch>`: se utiliza para desplazarse entre las ramas creadas. Se ha utilizado con frecuencia `git checkout -b <new-branch>`, para crear una nueva rama y cambiar a ella al instante.

- `git merge`: se ha utilizado para fusionar los cambios de una rama inferior a una superior. En concreto se ha usado siempre la opción, `git merge --no-ff <branch>`, fusiona la rama especificada en la rama actual, pero siempre genera una confirmación de fusión (incluso si se trata de una fusión con avance rápido), lo cuál es útil para documentar todas las fusiones que tienen lugar en tu repositorio.

A continuación, se van a detallar las distintas ramas que se han utilizado durante el desarrollo del proyecto junto con una breve explicación de su contenido o finalidad:

- `main`: rama principal, creada en el primer commit contiene la versión final de la aplicación.

- `develop`: creada a partir de main, contiene las versiones intermedias que se van obteniendo al final de cada Sprint.

- `sprint<number>`: se crea a partir del contenido de la rama develop del sprint anterior ya se que se realiza al inicio de cada sprint. Contendrá la implementación de las historias de usuario correspondientes a dicho sprint.

- `feature/HU-<epic number>-<number>` : se crea a partir de la rama del Sprint actual. Contiene la implementación de la funcionalidad de dicha historia de usuario.

- `feature/test`: se crea a partir de develop. Contiene la implementación de todos los tests unitarios realizados para la aplicación.

4.3. Tecnologías para el desarrollo del proyecto

4.3.1. Visual Studio Code

Visual Studio Code [27] es un editor de código fuente liviano pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (como C++, C#, Java, Python, PHP, Go, .NET).

Se ha elegido para este proyecto debido a que es el IDE en el que más experiencia se tiene por proyectos anteriores, además de que en las prácticas extracurriculares se seguía utilizando.

4.3.2. MongoDB

MongoDB [21] es un sistema de gestión de bases de datos no relacionales y de código abierto, que utiliza documentos flexibles en lugar de tablas y filas para procesar y almacenar varias formas de datos. No requiere un sistema de gestión de bases de datos relacionales (RDBMS), por lo que proporciona un modelo de almacenamiento de datos elástico que permite a los usuarios almacenar y consultar fácilmente tipos de datos multivariados. Esto no solo simplifica la gestión de la base de datos para los desarrolladores, sino que también crea un entorno altamente escalable para aplicaciones y servicios multiplataforma.

Los documentos o colecciones de documentos de MongoDB son las unidades básicas de datos. Estos documentos, con formato JSON binario, pueden almacenar varios tipos de datos y distribuirse en varios sistemas. El diseño de esquema dinámico de MongoDB da a los usuarios una flexibilidad para crear registros de datos, consultar colecciones de documentos a través de la agregación de MongoDB y analizar grandes cantidades de información.

En este proyecto se ha utilizado en el backend de la aplicación.

4.3.3. Node.js v16.14.0

Node.js [53] es un entorno de ejecución de JavaScript multiplataforma, de código abierto y gratuito que permite a los desarrolladores crear servidores, aplicaciones web, herramientas de línea de comandos y scripts.

En este proyecto se ha utilizado para la parte del backend de la aplicación.

4.3.4. NPM 8.3.1

NPM [4] es una herramienta fundamental en el desarrollo con JavaScript. Corresponde con las siglas de Node Package Manager y consiste en un administrador de paquetes que permite a los desarrolladores de JavaScript trabajar con lo que llamamos dependencias. Las dependencias son todas aquellas librerías que usamos al desarrollar un programa y que nos permiten evitar hacer todo el trabajo desde cero.

En este proyecto se ha utilizado para el backend de la aplicación.

4.3.5. Express.js

Express.js [9] es el framework backend más popular para Node.js. Está diseñado para crear aplicaciones web híbridas, de una sola página y de varias páginas; también se ha convertido en el estándar para desarrollar aplicaciones backend con Node.js y es la parte backend de algo conocido como pila MEAN. La pila MEAN es una pila de software JavaScript gratuita y de código abierto para crear sitios web y aplicaciones web dinámicos.

En este proyecto se ha utilizado para el backend de la aplicación, se detalla más en la sección 5.3.

4.3.6. Angular 16.2.6

Angular [16] es una plataforma de desarrollo, construida sobre TypeScript.Como plataforma, Angular incluye:

- Un marco basado en componentes para crear aplicaciones web escalables
- Una colección de bibliotecas bien integradas que cubren una amplia variedad de funciones, incluido el enrutamiento, la administración de formularios, la comunicación cliente-servidor y más.
- Un conjunto de herramientas de desarrollo para ayudarle a desarrollar, crear, probar y actualizar su código

Con Angular, se aprovecha una plataforma que puede escalar desde proyectos de un solo desarrollador hasta aplicaciones de nivel empresarial.

En este proyecto se ha utilizado para el frontend de la aplicación.

4.3.7. Bootstrap 4.6.2

Bootstrap [46] es un framework multiplataforma o conjunto de herramientas, de código abierto, para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, sólo se ocupa del desarrollo front-end.

En este proyecto se ha utilizado para el frontend de la aplicación.

4.3.8. Jasmine

Jasmine [17] es un framework para probar código JavaScript. No depende de ningún otro marco de JavaScript. Se ejecuta en navegadores y en Node.js. Y tiene una sintaxis limpia y obvia para poder escribir pruebas fácilmente.

En el proyecto se ha utilizado para la implementación de test unitarios.

4.3.9. Karma

Karma [55] es una herramienta que genera un servidor web que ejecuta código fuente contra código de prueba para cada uno de los navegadores conectados. Los resultados de cada prueba en cada navegador se examinan y se muestran a través de la línea de comando al desarrollador para que pueda ver qué navegadores y pruebas pasaron o fallaron.

En el proyecto se ha utilizado para la realización de test unitarios.

4.3.10. Google Fonts

Google Fonts [50] es un servicio de distribución de fuentes tipográficas propiedad de Google. Incluye una amplia biblioteca de fuentes de código abierto, un directorio web interactivo para navegar por la biblioteca y una API para usar las fuentes a través de CSS y Android.

En el proyecto se ha utilizado para añadir algunas estilos a los textos a través de los archivos SCSS, que son los archivos que contienen las hojas de estilos.

4.3.11. Microsoft Copilot

Copilot [51] es un asistente virtual de inteligencia artificial, desarrollado por Microsoft, que está disponible en Windows 11. Utiliza un modelo de lenguaje de gran tamaño para generar texto, traducir idiomas, escribir diferentes tipos de contenido creativo y responder a preguntas con un enfoque informativo.

En el proyecto se ha utilizado para generar los iconos de la aplicación y los iconos de las mascotas con la función de Designer, y para ayudar a ahorrar tiempo en la depuración de errores, en su mayoría sintácticos.

4.3.12. Ngx-paypal

Ngx-paypal [30] es una biblioteca de Angular que se basa en el SDK de Javascript de Paypal. Proporciona la interfaz de usuario propia para realizar los pagos a través de Paypal y conecta directamente con dicha plataforma.

En el proyecto se ha utilizado para realizar los pagos simulados de las reservas aceptadas por el gerente, se detallara más en la sección 6.1.2.

4.3.13. Dashboard Developer Paypal

Dashboard Developer Paypal [34] proporciona un sandbox, es decir, un entorno de prueba virtual autónomo que simula el entorno de producción de Paypal. Este entorno actúa de paso

previo antes de llevarlo a producción en un entorno real, ya que los procesos actúan igual en ambos entornos. Cuando se inicia un pago a través de una cuenta sandbox, Paypal crea un pago simulado. Al utilizar cuentas ficticias con sus credenciales de autenticación asociadas en sus llamadas a la API de PayPal, se puede probar sin hacer referencia a ningún usuario real de PayPal o cuentas reales de PayPal.

En el proyecto se ha utilizado para obtener las credenciales proporcionadas por el sandbox de Paypal para simular los pagos de las reservas aceptadas por el gerente. Se explica más detalladamente en la sección 6.1.2.

4.3.14. Node-cron

Node-cron [31] es un pequeño programador de tareas en JavaScript para Node.js, permite que se ejecuten una serie de tareas, las que se le indique, al iniciar el servidor.

En el proyecto se ha utilizado para programar una tarea que se ejecute cada vez que se inicie el servidor y cancele aquellas reservas en las que haya transcurrido un día o más desde que el gerente las acepto.

4.3.15. MongoDB Atlas

MongoDB Atlas [28] proporciona una base de datos como servicio (DBaaS), lo que permite configurar, implementar y escalar una base de datos sin preocuparse por el hardware físico local. Se trata de una base de datos en la nube que maneja la complejidad de implementar y administrar sus implementaciones en el proveedor de servicios en la nube de su elección (AWS, Azure y GCP).

En el proyecto se ha utilizado para obtener el clúster donde se alojará la base de datos se utilizará para el despliegue de la aplicación.

4.3.16. Vercel

Vercel [40] es una plataforma unificada en la nube que permite a los desarrolladores desplegar, gestionar y escalar sus aplicaciones y sitios web. Vercel proporciona una amplia gama de funciones, como despliegues automatizados, dominios personalizados y una potente CLI. También ofrece una amplia gama de integraciones con servicios populares, como GitHub, Slack y Zapier.

En el proyecto se ha utilizado para realizar el despliegue de la aplicación, por separado el frontend y el backend, a partir del repositorio de Git.

4.3.17. GitHub

GitHub [48] es una plataforma para desarrolladores que permite crear, almacenar, administrar y compartir su código. Utiliza software Git, que proporciona control de versiones distribuidas de Git más control de acceso, seguimiento de errores, solicitudes de funciones de software, gestión de tareas, integración continua y wikis para cada proyecto. Se utiliza comúnmente para albergar proyectos de desarrollo de software de código abierto.

En el proyecto se ha utilizado para el desarrollo del despliegue de la aplicación, para alojar el repositorio con el código de la aplicación para el despliegue.

4.4. Tecnologías para la comunicación

4.4.1. Microsoft Teams

Microsoft Teams [52] es una plataforma unificada de comunicación y colaboración que combina chat persistente en el lugar de trabajo, reuniones de vídeo, almacenamiento de archivos (incluida la colaboración en archivos) e integración de aplicaciones. El servicio se integra con el paquete de productividad de Office por suscripción y presenta extensiones que pueden integrarse con productos que no son de Microsoft.

En el proyecto se ha utilizado para mantener la comunicación con la tutora, y para realizar los eventos que no podían realizarse de manera presencial.

Capítulo 5

Diseño

En este capítulo se va a detallar el diseño realizado de la aplicación y su arquitectura junto con las decisiones tomadas para ello.

5.1. Decisiones iniciales de diseño

En este apartado se detallarán las decisiones que se han tomado para el desarrollo de la aplicación:

- Se tomará como referencia una arquitectura cliente-servidor
- Para el desarrollo del backend se utilizará Node con Express.js
- Para el almacenamiento de datos, se utilizara MongoDB como base de datos no relacional.
- Para el desarrollo del frontend se utilizarán Angular
- Para ayudar a la implementación de los estilos de la aplicación se utilizará Boootstrap

5.2. Arquitectura de referencia

En este apartado se detallará la arquitectura inicial de la aplicación que se irá refinando posteriormente con las decisiones de diseño que se tomen a lo largo del proyecto.
Se tomará como arquitectura inicial, o sobre la que se va a basar la aplicación y se va a ir evolucionando según las decisiones que se tomen, una arquitectura del tipo cliente-servidor.

Se puede definir como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente. El cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras. A continuación, se detalla con algo más de detalle ambos roles: [24]

- Cliente: proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término front-end.
  Normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además cliente-servidor de acceder a los servicios distribuidos en cualquier parte de una red.
  Las funciones que lleva a cabo el proceso cliente se resumen en: administrar la interfaz de usuario,interactuar con el usuario,hacer validaciones locales y recibir y formatear resultados del servidor.

- Servidor: encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Se le conoce con el término back-end.
  El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.
  Las funciones que lleva a cabo se resumen en: aceptar los requerimientos de bases de datos que hacen los clientes, procesar requerimientos de bases de datos, formatear datos para trasmitirlos a los clientes, y procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

En el caso de esta aplicación, esta arquitectura se puede ver reflejada entre la comunicación del cliente, en este caso Angular, con el servidor, en este caso Node.js y Express.js. El cliente, Angular, realizará solicitudes HTTP al servidor, Node.js y Express.js, para obtener o enviar datos, y este responderá devolviendo la respuesta al cliente.

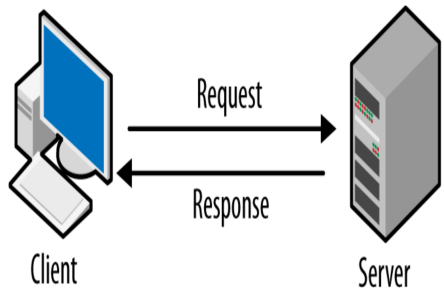


Figura 5.1: Cliente Servidor. Tomada de [15]

5.3. MEAN Stack Architecture

MEAN es un acrónimo de **M**ongoDB, **E**xpressJS, **A**ngular y **N**ode.js. Tiene un funcionamiento de la siguiente forma [29][20][10]:

1. Cuando el cliente hace una petición primero es procesado por Angular, en el lado del cliente.
2. Después la petición entra en la fase 2, que es NodeJS en el lado del servidor.
3. Después la petición entra en la fase 3, dónde ExpressJS hace la petición a la base de datos.
4. Después de que MongoBD reciba los datos y devuelve la respuesta a ExpressJS.
5. Entonces ExpressJS devuelve una respuesta a NodeJS y NodeJS se la devuelve a Angular y este muestra el resultado.

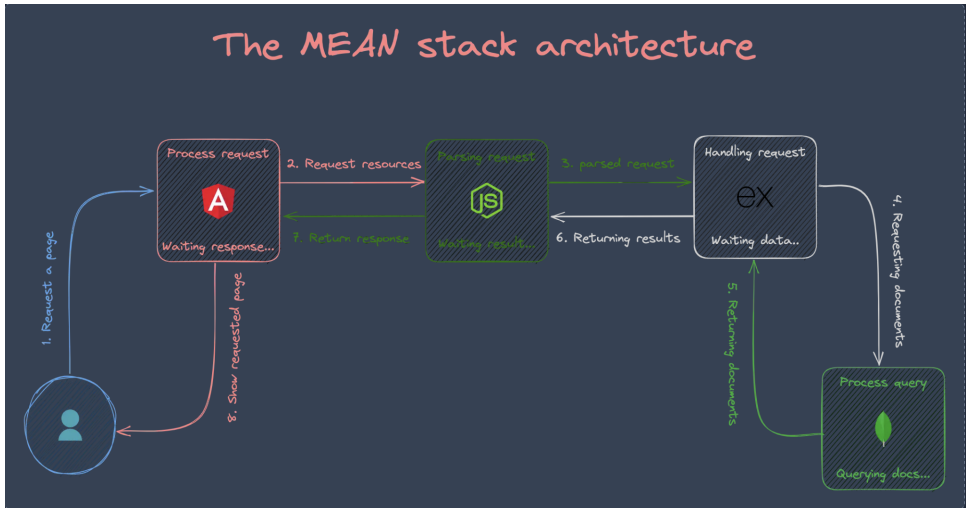


Figura 5.2: Arquitectura MEAN. Tomada de [10].

Si trasladamos esto a la arquitectura inicial o de referencia quedaría de la siguiente forma:

- Nivel de presentación - Angular : el nivel de presentación se encuentra en el lado del cliente y se encarga de la interfaz de usuario y la presentación de datos. Para ello se utiliza Angular como framework principal y el patrón sobre el que se basa será descrito en la siguiente sección 5.4.1. Se encarga de la interacción del usuario a través de eventos y acciones.
- Nivel de aplicación - Node.js y Express.js : el nivel de aplicación, en el lado del servidor, se encarga de la interacción con la capa de datos y del procesamiento de las solicitudes HTTP. Para ello se utiliza Node.js y Express.js.

- Nivel de datos - MongoDB: facilita la interacción con la base de datos MongoDB para realizar las operaciones CRUD, se encarga de definir los modelos que representan la estructura de los datos y ejecutar las operaciones necesarias según las solicitudes del nivel de aplicación.

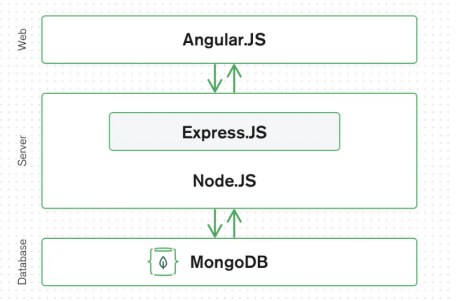


Figura 5.3: Pila MEAN. Tomada de [29]

5.4. Patrones arquitectónicos

5.4.1. Patrón MVVM

Se trata de de una variación del patrón MVC, resultado de una evolución del mismo.
En el caso de Angular, el modelo tiene mucha relación con la vista. Esto se debe al concepto base de two-way data binding, ya que la forma de sincronizar los datos entre la vista y el modelo-vista es totalmente dependiente, es decir, en la vista se puede modificar el modelo y en el modelo se puede modificar la vista. Esto hace que la independencia que se produce en el MVC clásico aquí no se produce, y por eso pasa a denominarse modelo vista vista modelo.[32]
A continuación, se detallan los tres componentes principales:

- **Model**: representa los datos y la lógica de negocio de la aplicación.
- **View**: es responsable de la presentación visual y la interfaz de usuario.
- **ViewModel**: actúa como capa intermedia entre el View y el Model. Se encarga de la lógica de presentación, gestionando los datos del Model para que puedan ser fácilmente utilizados por el View. También controla las interacciones del usuario y puede contener enlaces de datos bidireccionales para mantener sincronizar los datos del Model y el View.

En el caso de esta aplicación, como ya se ha mencionado anteriormente se de en el cliente, Angular, en el nivel de presentación. Angular introduce el concepto de componentes. Un componente es una combinación de View y ViewModel. El View está representado por el

template del componente, y la lógica del componente está representada por el ViewModel, gestionando la lógica de la presentación y la interacción con el Model.
El Model se encuentra representado principalmente por las interfaces en archivos TypeScript (.ts), el ViewModel se encuentra representado por el archivo TypeScript (.ts) de cada componente, y El View se encuentra representado por el archivo .html de cada componente.

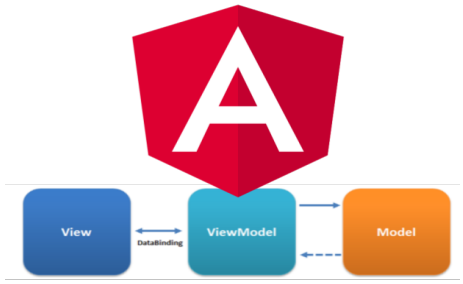


Figura 5.4: Patrón MVVM. Tomada de [14]

5.5. Principios de diseño

En este apartado se detallarán los principios de diseño, en este caso los principios SOLID [54], que se han tomado como referencia y sobre los que se va a basar el proyecto.[18]

5.5.1. Single Responsibility Principle

Cada clase, módulo, o función en un programa debería tener una única responsabilidad u objetivo en el programa. Comúnmente se utiliza la definición, cada clase debería tener una única razón para cambiar.
En el caso de esta aplicación, este principio se puede ver en que cada módulo o clase tendrá una única responsabilidad, es decir, una única razón. En un ejemplo práctico, se dispondrá de un servicio para las reservas que deberá tener responsabilidades relacionadas sólo con la gestión de reservas.
Esto facilita el mantenimiento y la compresión del código al dividir las funcionalidades en unidades cohesivas.

5.5.2. Open Closed Principle

Este principio establece que las entidades de software deberían estar abiertas para extensión, pero cerradas para modificaciones. Esto implica que estas entidades, clases o funciones deberían ser creadas de manera que sus funcionalidades principales puedan ser extendidas a otras entidades sin alterar el código fuente original de la entidad.

En el caso de esta aplicación, este principio se puede ver al permitir añadir nuevas funcionalidades a las ya existentes sin tener que modificar el código existente.

5.6. Patrones de diseño

En este apartado se detallarán los patrones de diseño que se han tomado como referencia y sobre los que se ha basado el proyecto.

5.6.1. Singleton Pattern

Este patrón se asegura de que al crear una instancia de una clase, esta sea única para ello se debe impedir que los usuarios creen nuevas instancias. También, hace que sea accesible de manera global. Esto se logra declarando el constructor como privado, lo que significa que sólo el propio código del singleton puede instanciarse a si mismo y asegura que el mismo objeto llega al usuario. Si ya existe la instancia, no se crea ninguna nueva.
En el caso de esta aplicación, la conexión con la base de datos en el nivel de datos se podría hacer utilizando este patrón para asegurar que se crea una única instancia de conexión.[23]

5.6.2. Observer Pattern

Este patrón se basa en la idea de que un objeto, llamado sujeto o sujeto observable, tiene una lista de observadores que quieren ser informados ante cualquier cambio en su estado. Cuando el sujeto cambia, notifica a todos los observadores que tenga registrados y les proporciona los detalles para que actualicen su estado o realicen cualquier acción necesaria en función de la notificación recibida.
Aunque se ha dado algo de detalles, tiene los siguientes componentes clave [25]:

- Sujeto observable: mantiene una lista de observadores y notifica a estos cuando cambia su estado. Puede haber más de un sujeto y cada uno puede tener su propia lista de observadores.
- Observadores: objetos que quieren ser notificados ante cualquier cambio en el sujeto observable. Cada observador implementa una interfaz o clase abstracta que define un método de actualización. Este método se llama automáticamente cuando el sujeto notifica un cambio.

En el caso de esta aplicación, se podrá apreciar el uso de este patrón en la parte del cliente, Angular, donde los componentes podrán observar y reaccionar a cambios. Un ejemplo práctico, podría ser cuando se solicita una reserva, los componentes relativos podrían ser notificados para actualizar la interfaz de usuario.

5.7. Diseño Inicial UX

A continuación, se muestra el diseño realizado inicialmente de la interfaz de usuario, mostrando las diferentes pantallas por las que se podría navegar y con algunas anotaciones con aclaraciones a tener en cuenta en la fase de implementación y desarrollo.

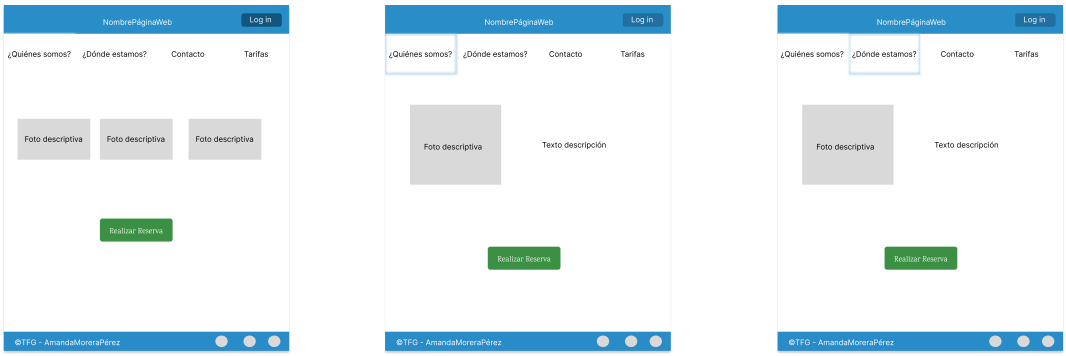


Figura 5.5: Diseño UX - Cliente no registrado inicio (1)

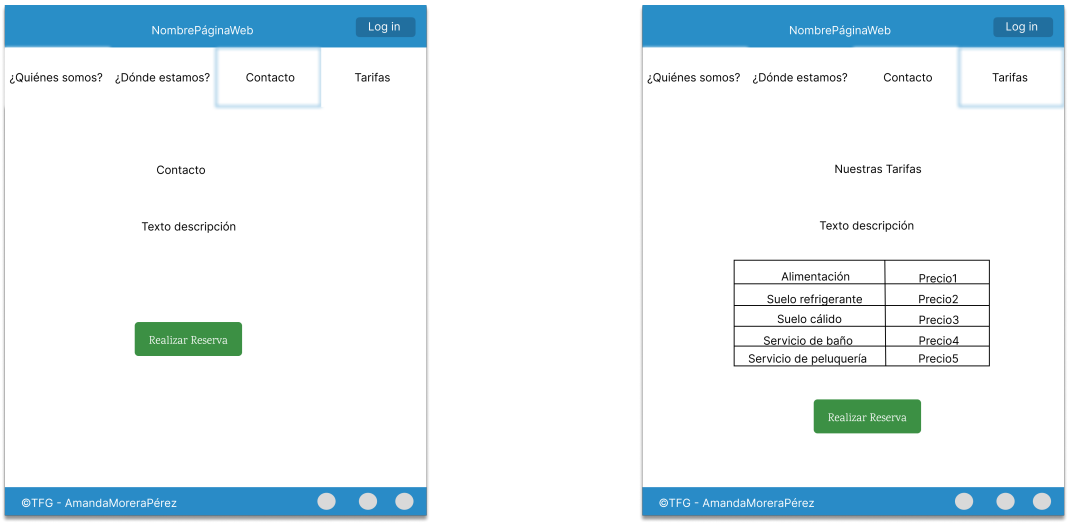


Figura 5.6: Diseño UX - Cliente no registrado inicio (2)

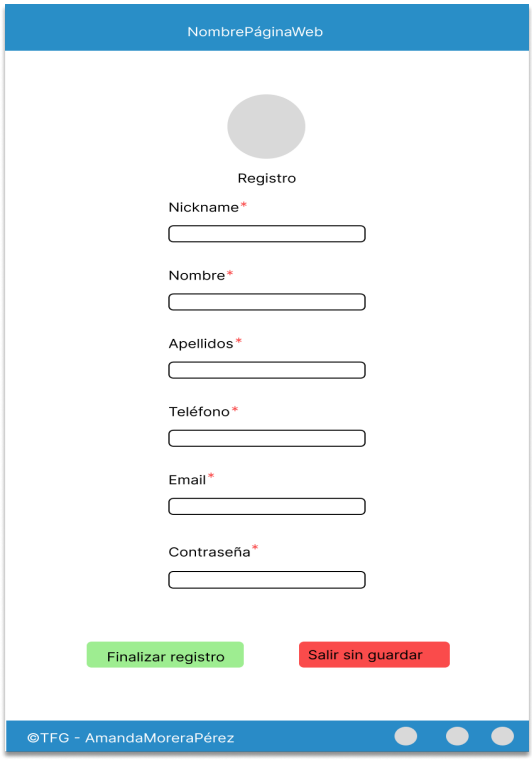
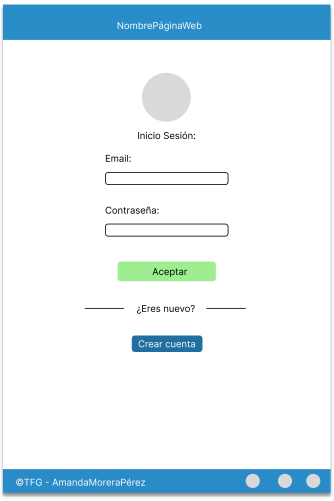
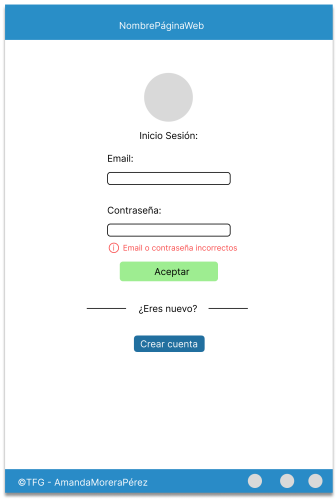


Figura 5.7: Diseño UX - Formulario registro





Aparecera en el caso de que el email o la contraseña introducidos sean incorrectos

Figura 5.8: Diseño UX - Inicio de sesión

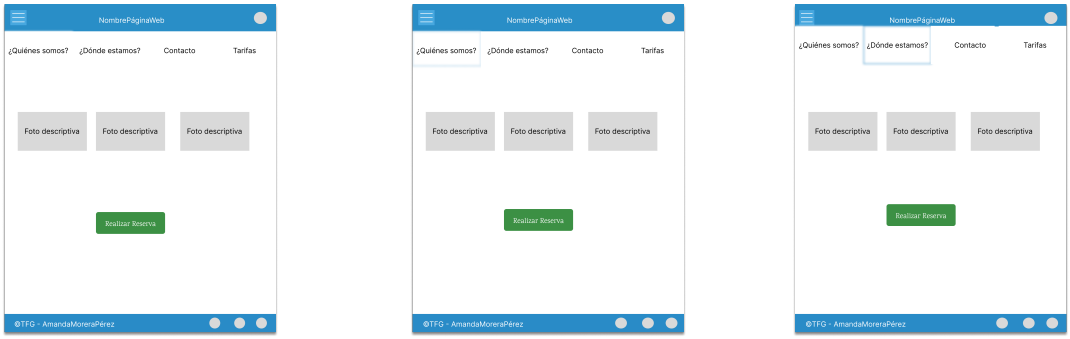


Figura 5.9: Diseño UX - Cliente registrado inicio(1)

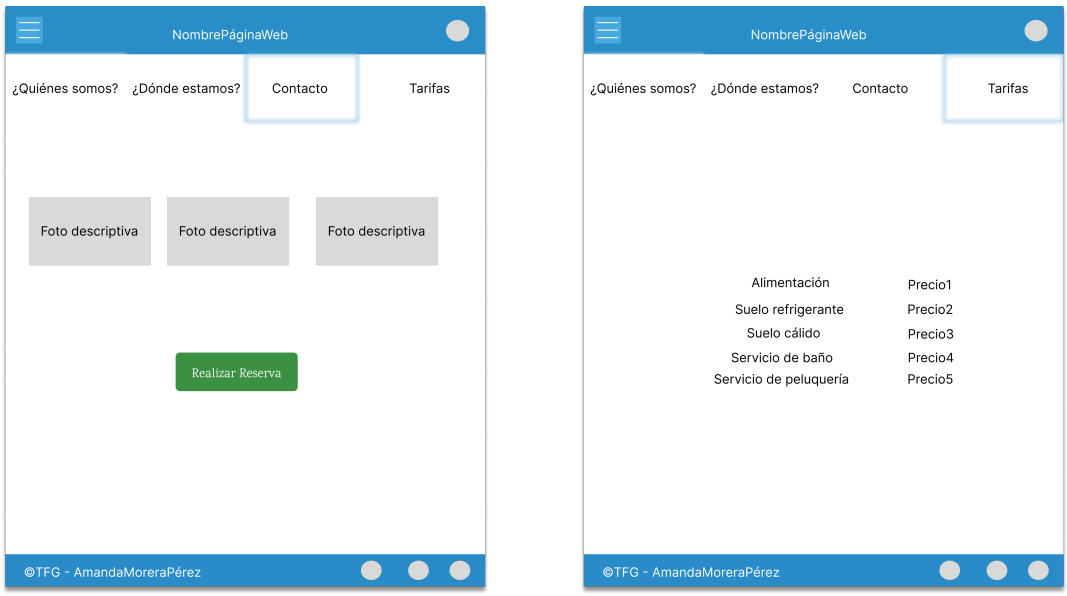


Figura 5.10: Diseño UX - Cliente registrado inicio(2)

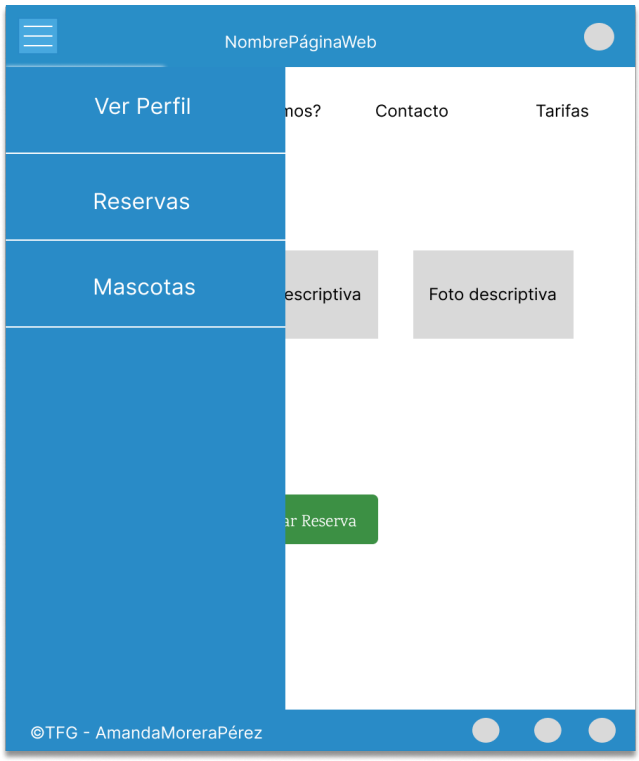


Figura 5.11: Diseño UX - Cliente registrado menú desplegable

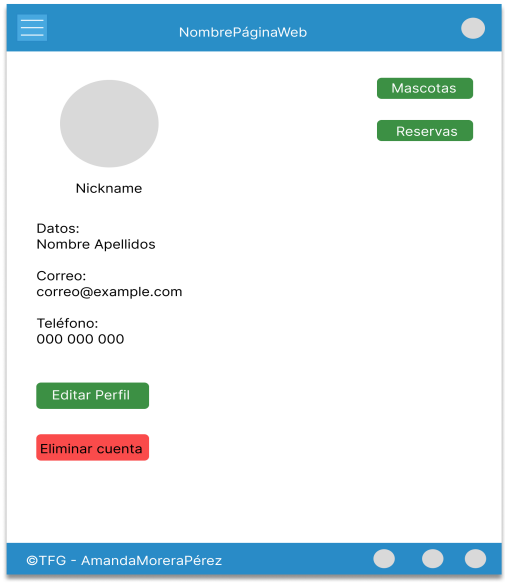


Figura 5.12: Diseño UX - Cliente registrado apartado perfil

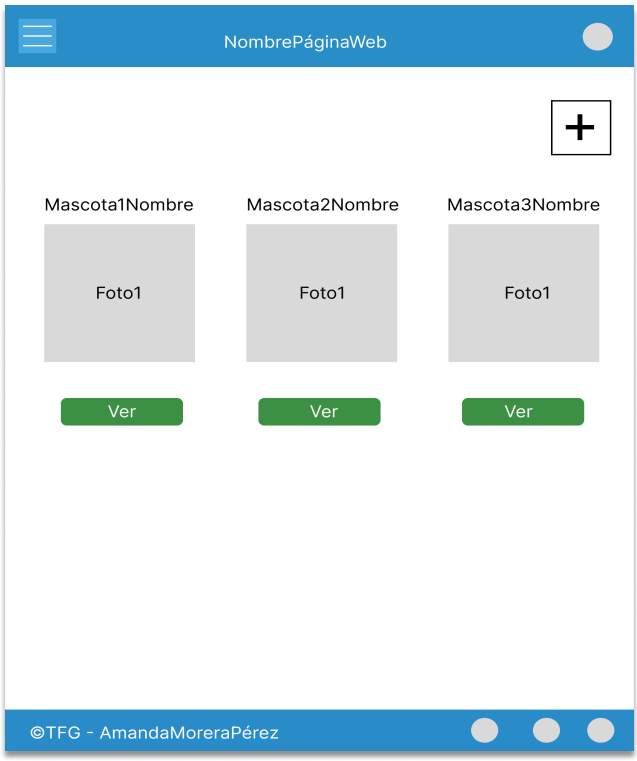


Figura 5.13: Diseño UX - Cliente registrado apartado mascotas

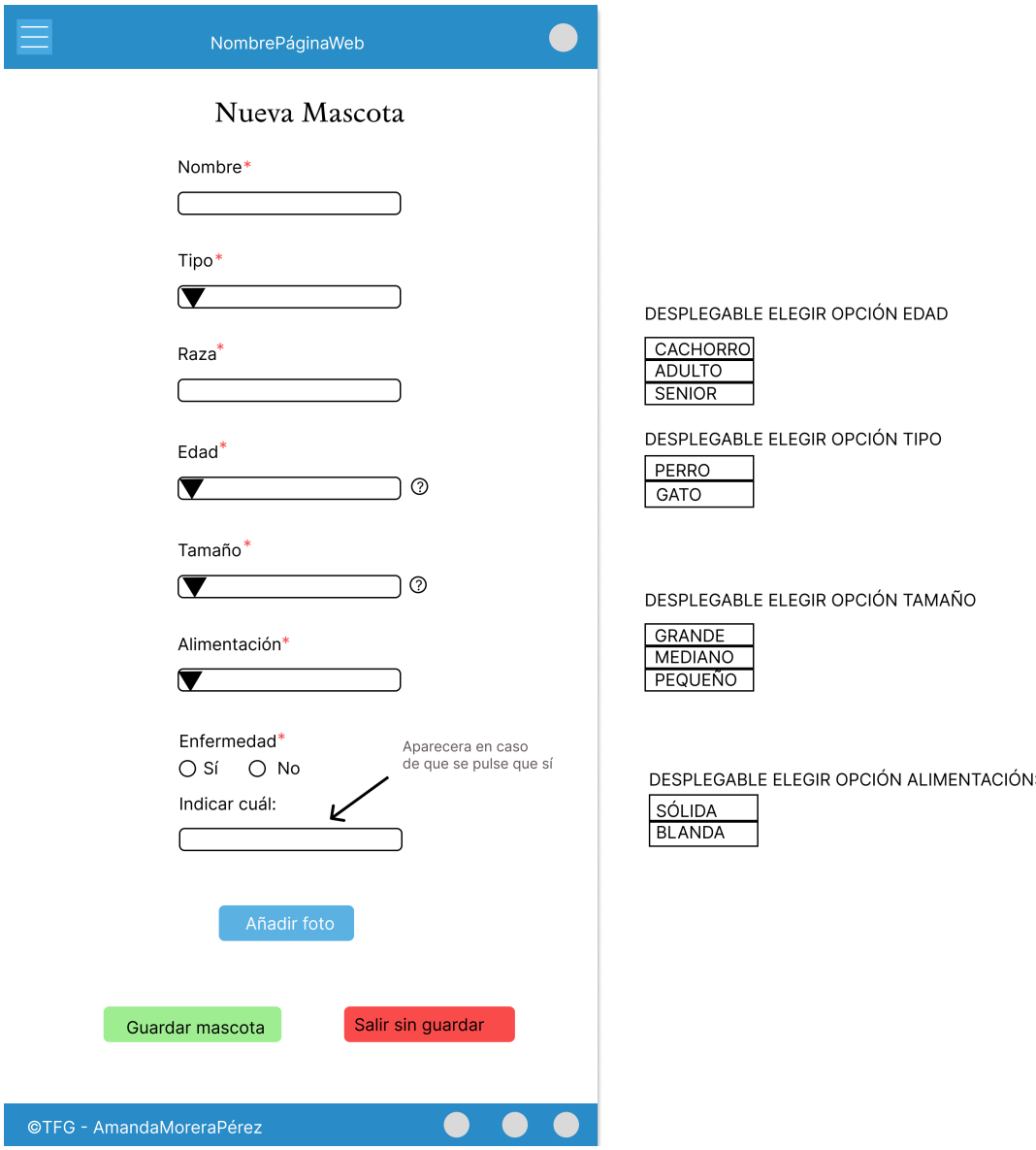


Figura 5.14: Diseño UX - Formulario nueva mascota(1)

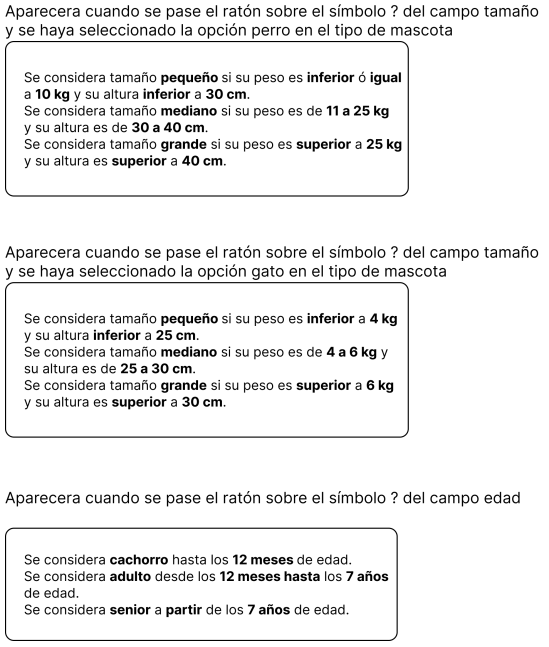


Figura 5.15: Diseño UX - Formulario nueva mascota(2)

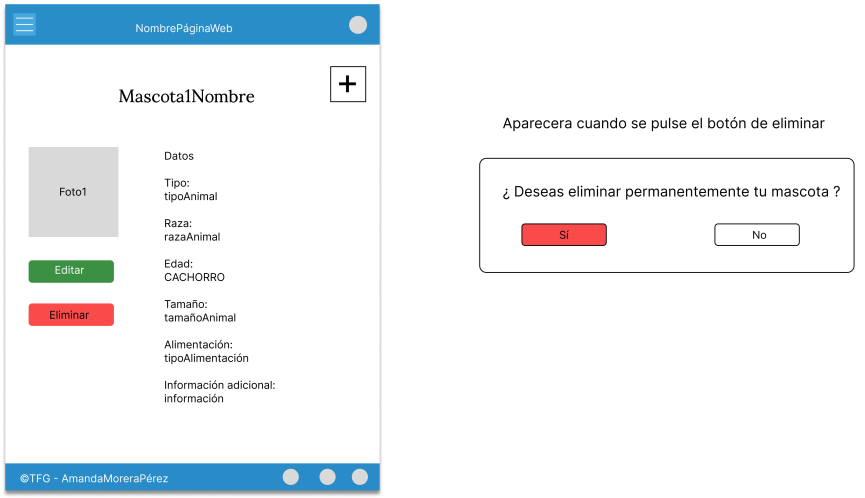


Figura 5.16: Diseño UX - Detalle mascota, modal eliminar

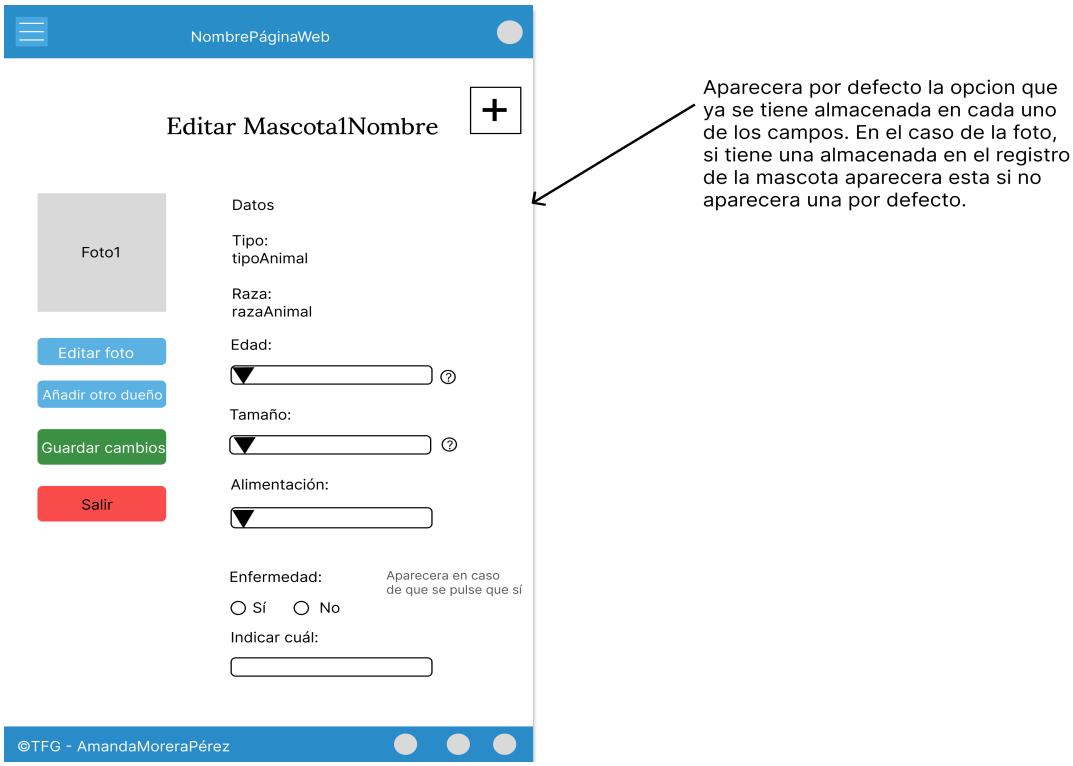


Figura 5.17: Diseño UX - Editar mascota

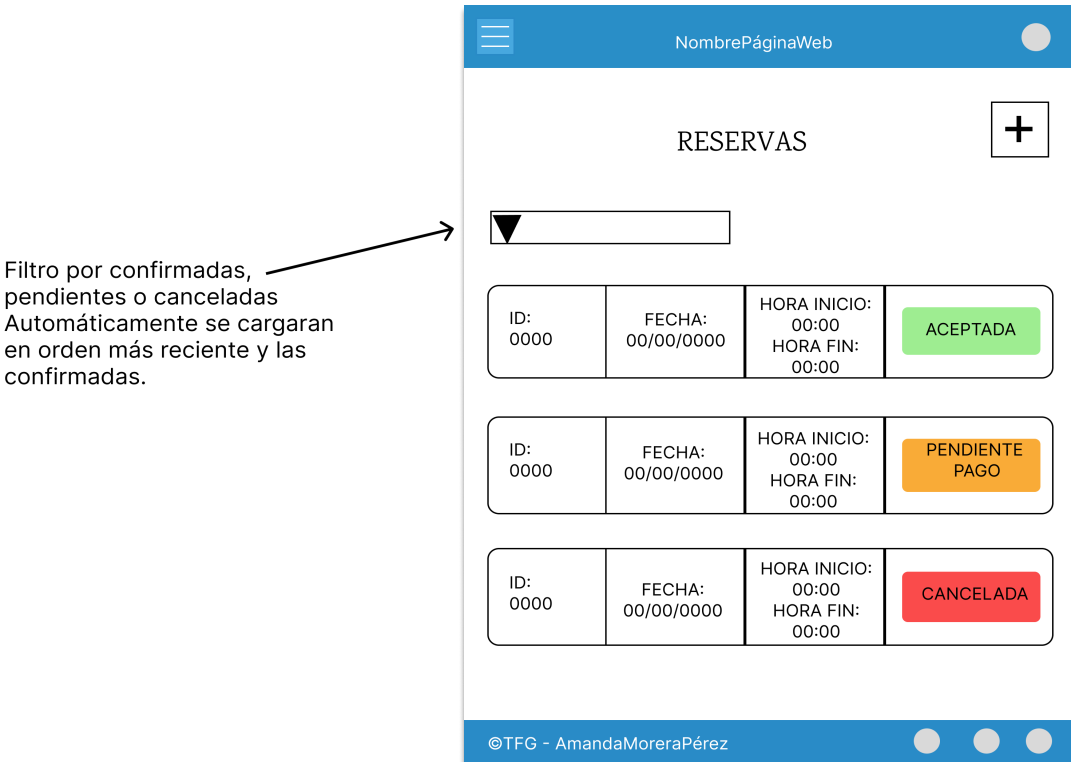


Figura 5.18: Diseño UX - Apartado reservas

Figura 5.19: Diseño UX - Detalle reserva

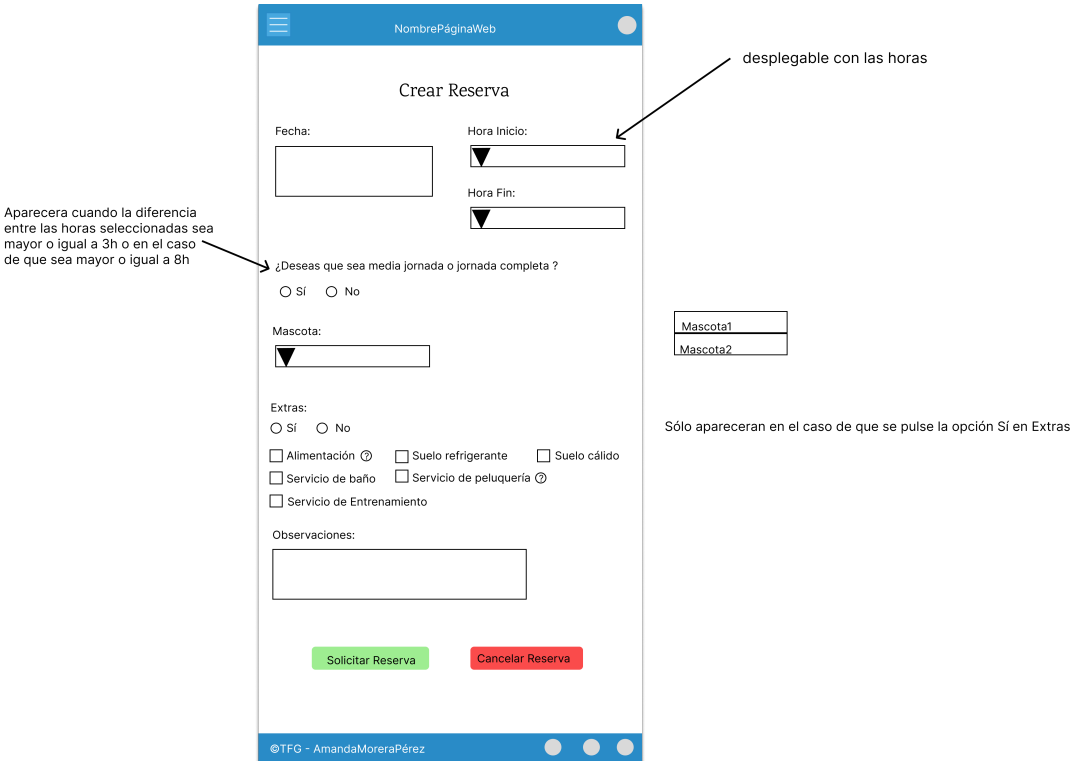


Figura 5.20: Diseño UX - Formulario solicitar reserva(1)

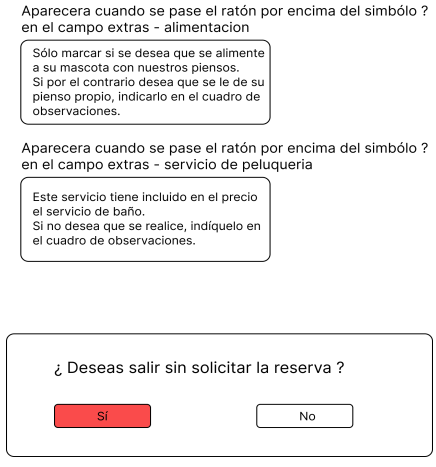


Figura 5.21: Diseño UX - Formulario solicitar reserva(2)

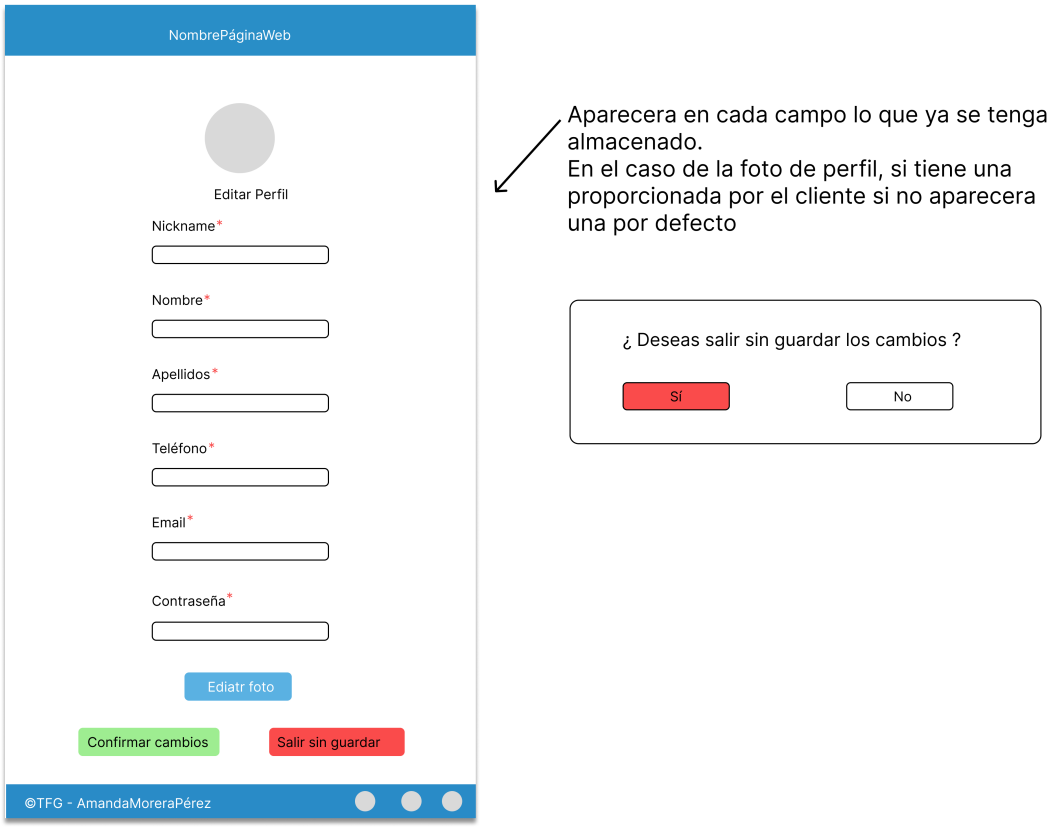


Figura 5.22: Diseño UX - Formulario editar perfil

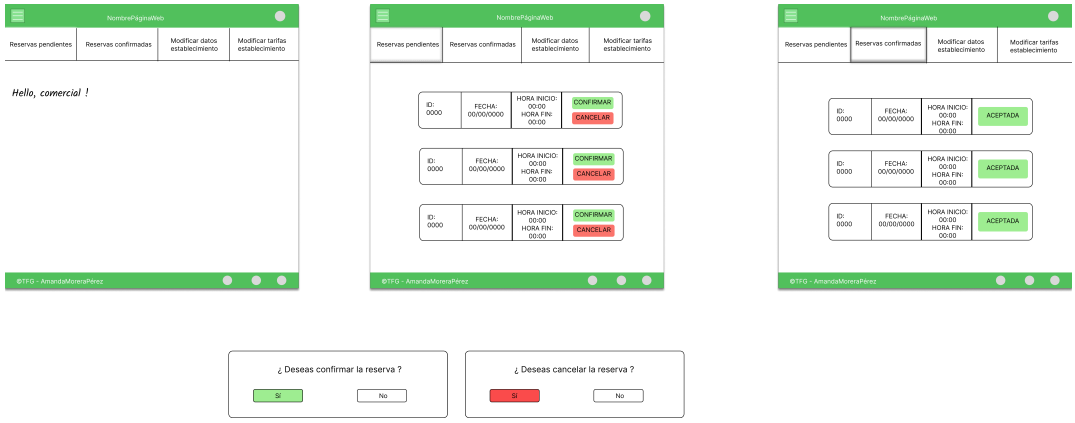
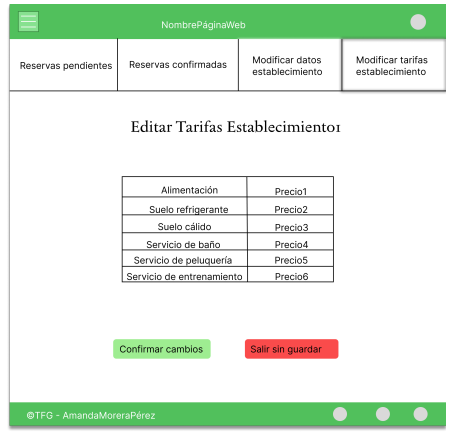
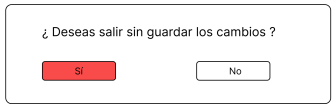


Figura 5.23: Diseño UX - Gerente(1)





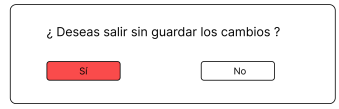


Figura 5.24: Diseño UX - Gerente(2)

5.8. Diseño basado en componentes

En Angular, un componente controla la vista de la pantalla. La lógica de aplicación de un componente se define dentro de una clase y esta interactúa con la vista a través de una API. Angular crea, actualiza y destruye componentes a medida que se va navegando por la aplicación. [2]

Dentro de Angular, para identificar una clase como un componente se pone el decorador @Component al principio de la misma y se especifican sus metadatos. Estos le dicen a Angular dónde obtener los bloques de construcción principales que necesita para crear y presentar el componente y su vista. En particular, asocia una plantilla con el componente, ya sea directamente con el código en línea o por referencia. Juntos, el componente y su plantilla describen una vista.

En cuánto a las plantillas, se encargan de definir las vistas de un componente y está formada por un HTML que le dice a Angular cómo tiene que renderizar el componente. Normalmente, las vistas se organizan jerárquicamente, lo que permite modificar, mostrar u ocultar secciones o páginas enteras de la pantalla. En la siguiente imagen 5.25 se puede ver cómo sería dicha jerarquía.

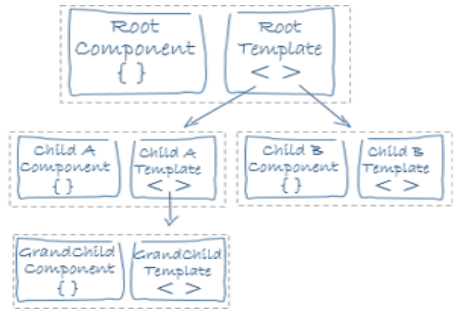


Figura 5.25: Jerarquía de componentes Angular

5.8.1. Componentes creados para el frontend

En la imagen 5.26, se pueden ver los componentes desarrollados para el frontend de la aplicación, y cómo están relacionados entre ellos.

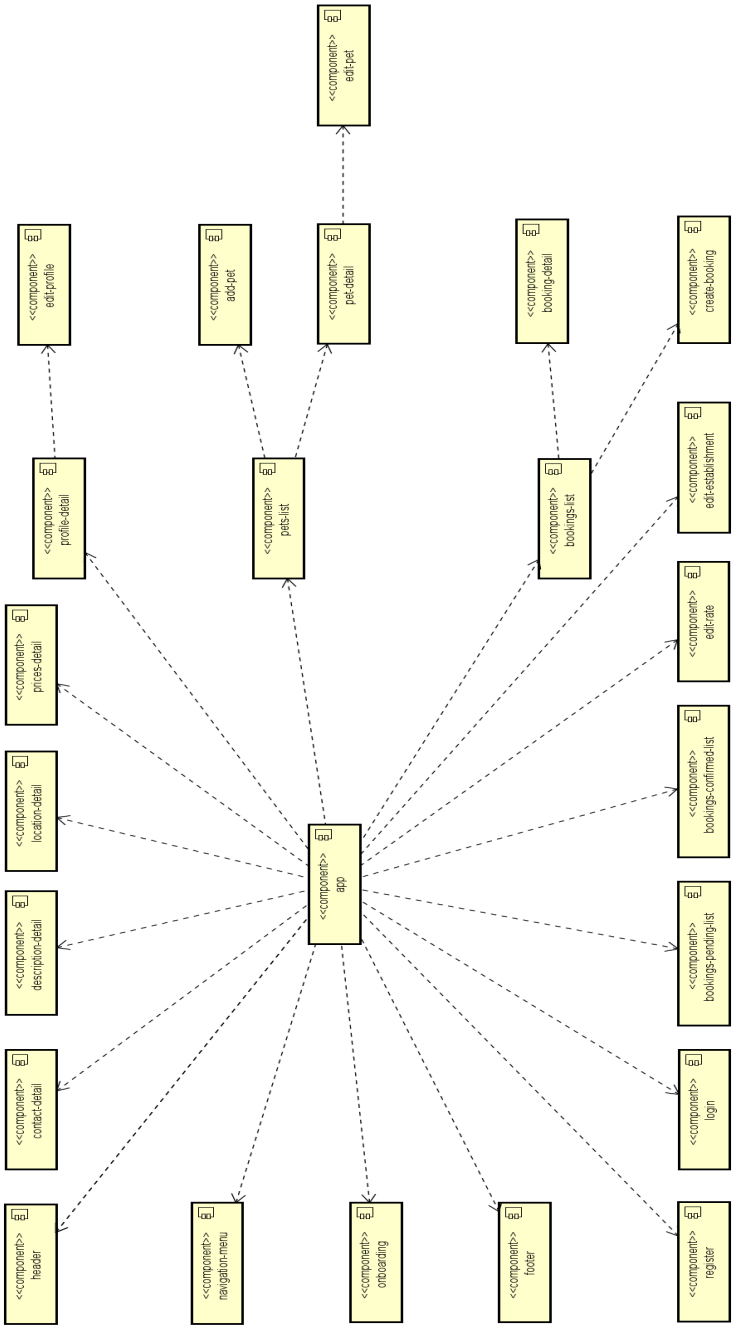


Figura 5.26: Diagrama componentes creados en el frontend

5.9. Diseño del despliegue

En este apartado, se detallará el diseño del despliegue realizado para el proyecto. En primero lugar, tal y cómo se ve en la figura 5.27, tenemos el diagrama de despliegue en el entorno local. Desde un navegador. en este caso Google Chrome, se realizarán peticiones a la aplicación que se ejecuta en un servidor de Node, conectada a la API ejecutandose en Express y esta última conectada a la base de datos en MongoDB.

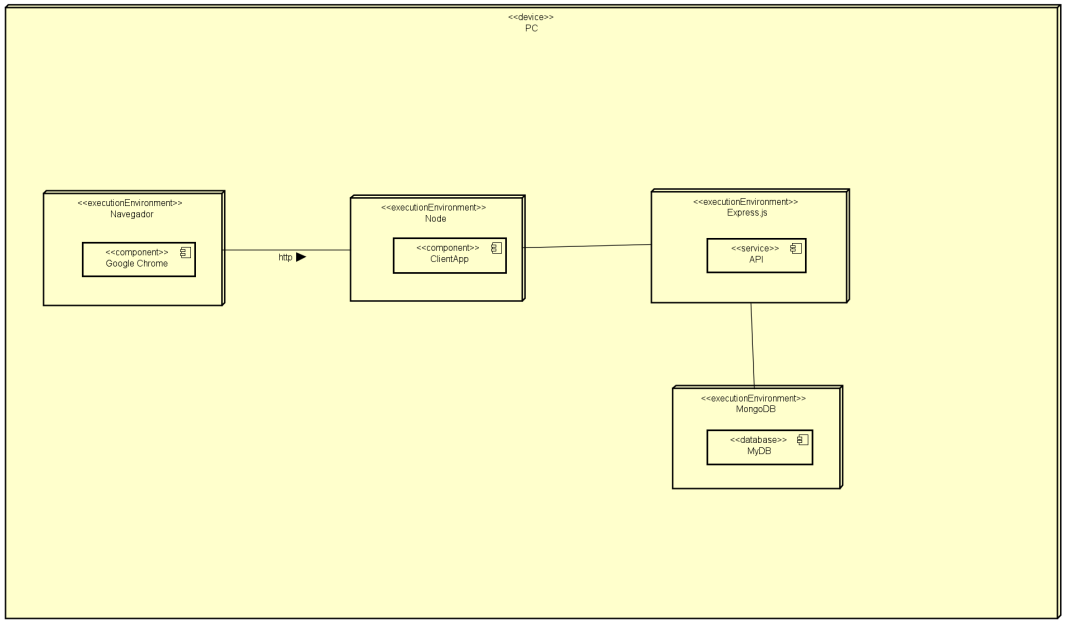


Figura 5.27: Diagrama despliegue en local

En segundo lugar, tenemos el diseño de despliegue en el entorno de producción, tal y cómo se ve en la figura 5.28, tendremos de nuevo un navegador desde el que se realizarán peticiones https al contenedor de Vercel, que es el entorno elegido para el despliegue, este contiene la aplicación en dos contenedores, uno para el frontend de la aplicación y otro para el backend de la aplicación. Este se comunicará con otro contenedor, MongoDB Atlas, que proporciona un clúster en AWS que contiene la base de datos, testDB.

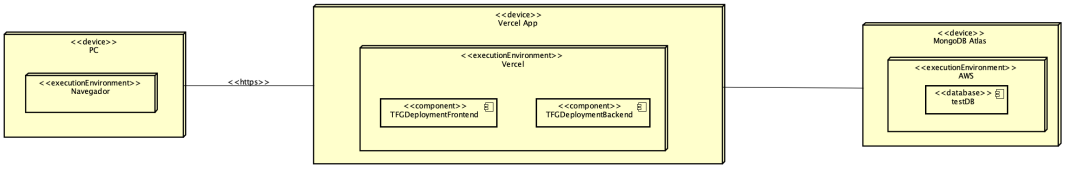


Figura 5.28: Diagrama despliegue

Capítulo 6

Implementación y pruebas

En este capítulo se va a detallar algunas cuestione sobre lo relacionado con la implementación y con las pruebas realizadas durante el desarrollo. En particular se presenta cómo se ha organizado el código, cómo se han realizado los tests unitarios y, si se ha encontrado algún fallo, cómo se ha solucionado.

6.1. Implementación

En esta sección se va a detallar cómo se ha organizado el código, detallando como se ha estructurado la jerarquía de directorios y explicando brevemente el contenido de cada uno centrándose en los archivos más importantes o que tienen más relevancia para la funcionalidad de la aplicación, tanto en el lado del cliente como en el lado del servidor.

6.1.1. Organización del código

En Visual Studio, se han distinguido dos directorios dentro del general del proyecto. En primer lugar, uno denominado client-app para la parte del frontend de la aplicación y, en segundo lugar, uno denominado server para la parte del backend de la aplicación.

Organización del frontend

A continuación, se va a detallar cómo se ha organizado el frontend dentro del proyecto en Visual Studio Code. En las figuras 6.1 y 6.2 se puede ver cómo quedaria estructura del proyecto.

Dentro del directorio, client-app, encontramos:

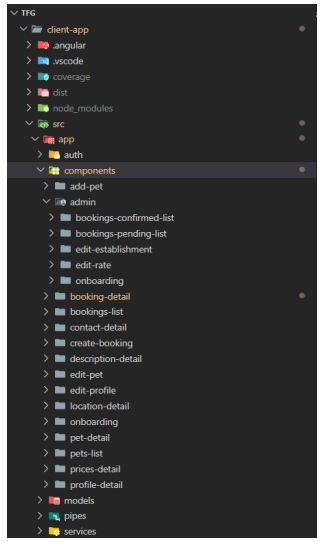


Figura 6.1: Organización del código del frontend (1)

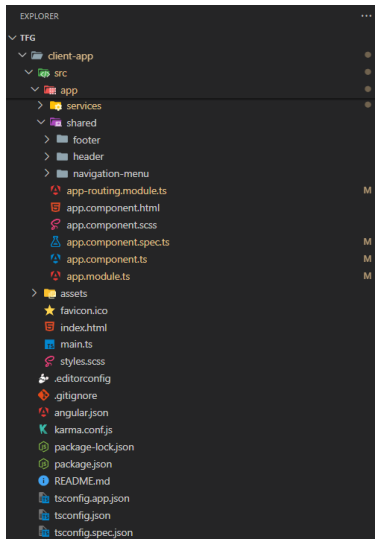


Figura 6.2: Organización del código del frontend (2)

- El directorio coverage, que contiene el resultado de la ejecución de los tests.
- El directorio dist, que se genera al hacer un build del proyecto.
- El directorio node_modules que contiene las dependencias instaladas.
- El directorio auth, que contiene todo lo relativo a la lógica para la autenticación. Dentro de este, encontramos otros dos directorios para los componentes de login y

register. Cada uno con su archivo .html, .ts, .scss y .spect.ts . También se encuentran los siguientes archivos:

  - El archivo auth.guard.ts, para definir cómo se van a restringir las rutas y limitar el acceso a ellas, dependiendo de si se ha iniciado sesión o no.
  - El archivo auth.interceptor.ts, para definir el token y los headers de autorización de las peticiones HTTP.
  - El archivo auth.service.spec.ts, para definir los tests que serán lanzados para probar el servicio.
  - El archivo auth.service.ts, que contiene las funciones que realizaran las solicitudes HTTP al backend, para iniciar sesión, registrarse o cerrar sesión.
  - El archivo storage.service.spec.ts, para definir los tests que serán lanzados para probar el servicio.
  - El archivo storage.service.ts, que contiene la lógica que se encarga de almacenar la información del usuario en la sesión del navegador, obtener esa información, borrar esa información de la sesión y comprobar si alguna información almacenada

- El directorio components, que contiene la implementación de los componentes detallados, cada uno con su propio directorio y sus correspondientes archivos .html, .ts, .scss y .spec.ts .
- El directorio models, que contiene las clases del modelo de dominio.
- El directorio services, que contiene la implementación de los servicios que realizan las peticiones HTTP al backend.
- El directorio shared, contiene la implementación de aquellos componentes que se repiten varias veces y se reutilizan. Dentro de este, se encuentran los componentes footer, navigation-menu y header.
- El directorio assets, que contiene las imágenes y los iconos que se utilizarán en la aplicación.
- app component, que es el componente principal de la aplicación ya que actúa como punto entrada y contiene la lógica y estructura para renderizar otros componentes Sus archivos principales son:

  - app-routing.module.ts: es el módulo de enrutamiento principal de la aplicación, dentro de este archivo se definen las rutas y la configuración de navegación entre los componentes. También se pueden indicar las guardas que van a permitir que se navegue o no a los distintos componentes.
  - app.component.html : es el arcchivo que contiene la plantilla html del componente, dentro de este se definen los elementos y otros componentes que se van a renderizar.
  - app.component.scss : es el archivo que contiene las reglas de los estilos que van a aplicar al componente.
  - app.component.spec.ts: es el archivo que contiene la lógica para las pruebas unitarias del componente.

  - app.component.ts : es el archivo principal del componente en el que se define la lógica y su comportamiento. Contiene la clase del componente con sus propiedades, sus métodos y el ciclo de vida.
  - app.module.ts: es el módulo raíz de la aplicación, en este archivo se definen los componentes, servicios y otros módulos que utiliza la aplicación. También se configura dentro de este archivo la inyección de dependencias.

- El archivo index.html,es el archivo html principal de la aplicación, que sirve como base sobre la cuál se carga la aplicación ya que es el primer archivo que carga el navegador al iniciar la aplicación. Contiene el elemento <app-root>, que es el componente raíz de la aplicación. También sirve para incluir dependencias.
- El archivo main.ts, es el punto de entrada para compilar y arrancar la aplicación. Realiza la configuración inicial y carga el módulo principal, el app.module, iniciando la aplicación.
- El archivo styles.scss, para configurar los estilos de la aplicación.
- El archivo .gitignore, es un archivo de texto que le indica a Git qué archivos o directorios deben ser ignorados y no rastrearlos en el control de versiones.
- El archivo angular.json, que contiene las configuraciones propias de Angular.
- El archivo karma.conf.js, que contiene la configuración para el lanzamiento y realización de los tests.
- El archivo package-lock.json, que se genera automáticamente al instalar las dependencias.
- El archivo package.json, que contiene las dependencias que serán instaladas en el proyecto y que generarán el directorio node_modules

Organización del backend

A continuación, se va a detallar cómo se ha organizado el backend dentro del proyecto en Visual Studio Code. En la Figura 6.3 se puede ver cómo quedaria estructura del proyecto.

Dentro del directorio, server, encontramos:

- El directorio controllers, contiene los controladores que tienen la lógica que se encarga de procesar las solicitudes HTTP entrantes, los datos si los reciben y dar la respuesta adecuada.
- El directorio middlewares, que contiene la implementación de funciones que se encargan de gestionar y procesar las solicitudes HTTP antes de que lleguen directamente al controlador correspondiente.
- El directorio models, contiene la implementación de los modelos de las clases.

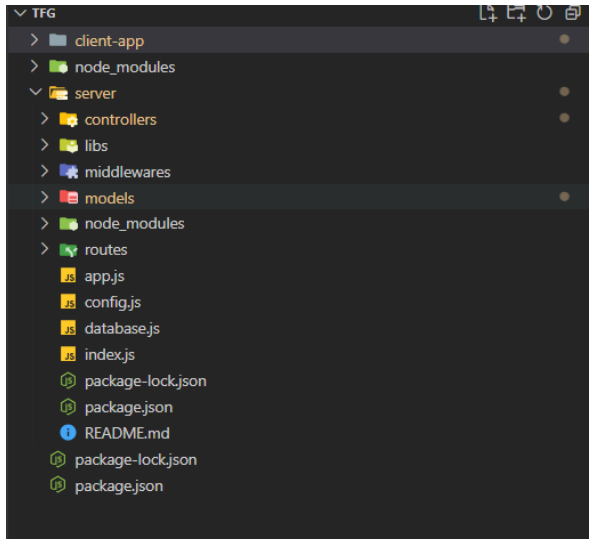


Figura 6.3: Organización del código del backend

- El directorio node_modules, contiene los directorios de las dependencias instaladas.
- El directorio routes, contiene los archivos de enrutamiento.
- El archivo app.js, contiene la implementación que se encarga de configurar algunos middlewares, la sesión de cookies y definir las rutas de la API.
- El archivo config.js, contiene la configuración de la clave secreta utilizada para JWT. Esta clave será utilizada luego para las operaciones de encriptado, autenticación o de registro.
- El archivo database.js, contiene la lógica que se encarga de establecer la conexión con la base de datos de MongoDB.
- El archivo index.js, es el principal de la parte del backend ya que contiene la lógica que inicia el servidor.
- El archivo package-lock.json, que se genera automáticamente al instalar las dependencias.
- El archivo package.json, que contiene las dependencias que serán instaladas en el proyecto y que generarán el directorio node_modules

6.1.2. Implementación de pagos

Para la implementación de los pagos se ha utilizado la API que proporciona Paypal en el modo sandbox. En el entorno de pruebas, tras crear la cuenta como desarrollador de Paypal [33], se debe instalar la librería de ngx-paypal con el comando `npm i ngx-paypal`,

situados dentro del directorio de la parte del frontend en este caso dentro del directorio client-app, y añadir el módulo en los imports del archivo app.module.ts.

En el html en el que se quiera añadir los botones para realizar el pago, en este caso en el componente de detalle de la reserva (booking-detail), se debe añadir lo siguiente `<ngx-paypal [config]="payPalConfig»</ngx-paypal>`. Con este selector se añaden dichos botones, aunque aparezcan dos, uno para realizar el pago con paypal y otro con tarjeta sólo va a estar activo y tener funcionalidad el de paypal. Como era algo que ya traía la propia librería no se podía modificar para que sólo apareciese uno.

A continuación, se implementa la función initConfig(),la cuál se debe llamar desde la función ngOnInit() del componente, que se va a encargar de implementar la interfaz propia de configuración de Paypal (IPayPalConfig) que trae la librería instalada, lo primero que se indica es el tipo de moneda y el clientId, en este apartado hay que indicar el Client ID que proporciona la API, lo más conveniente es guardarlo como una variable de entorno. Lo siguiente, es crear la transacción a realizar cuando tiene lugar dicho evento (ICreateOrderRequest), también propia de la librería. En otras se indica el tipo de moneda y el valor de la cantidad a pagar que en este caso es el precio final de la reserva.

También, destacar que cuando tiene lugar el evento en el que se autoriza al cliente (onClientAuthorization), el pago se ha realizado correctamente. En este caso se crea el nuevo pago que se almacenará en nuestra base de datos, se actualiza la reserva al estado de confirmada y se muestra la modal adecuada.

Para obtener el email y la contraseña de prueba que proporciona Paypal, se debe navegar al apartado Sandbox accounts dentro en el dashboard. Dentro de este se deben introducir los datos en la cuenta que se indica de tipo Personal. Si hace click en esta se pueden ver los datos de prueba que aparecen en la cuenta y que se observan al realizar el pago. En este caso se ha cambiado el nombre a Prueba Pago. Por último, indicar que esta cuenta de prueba de un saldo inicial de unos 5000 euros, aunque se puede modificar en este caso como era saldo suficiente no se ha modificado.

6.1.3. Despliegue de la aplicación

A continuación, se va a detallar cómo se ha llevado a cabo el despliegue de la aplicación desde cero. Para ello, lo primero que se va a realizar es la configuración del clúster que proporcionará la base de datos que se va a utilizar para el despliegue que proporciona de manera gratuita MongoDB Atlas.

Si no se dispone de una cuenta se debe crear para ello. Una vez dentro, hacer click en la opción de crear nuevo proyecto si no redirige directamente a la pantalla para crear el nuevo clúster. De las tres opciones que aparecen hacer click en la opción M0 (free), que es la opción gratuita que proporciona y se va a usar. Indicar un nombre, o dejar el que aparece ya por defecto. En provider, seleccionar AWS. En la región, indicar Ireland(eu-west-1), y en el tag no hace falta indicar nada. A continuación, pulsar el botón Create Deployment. En las figuras 6.4 y 6.5 se ve esto reflejado.

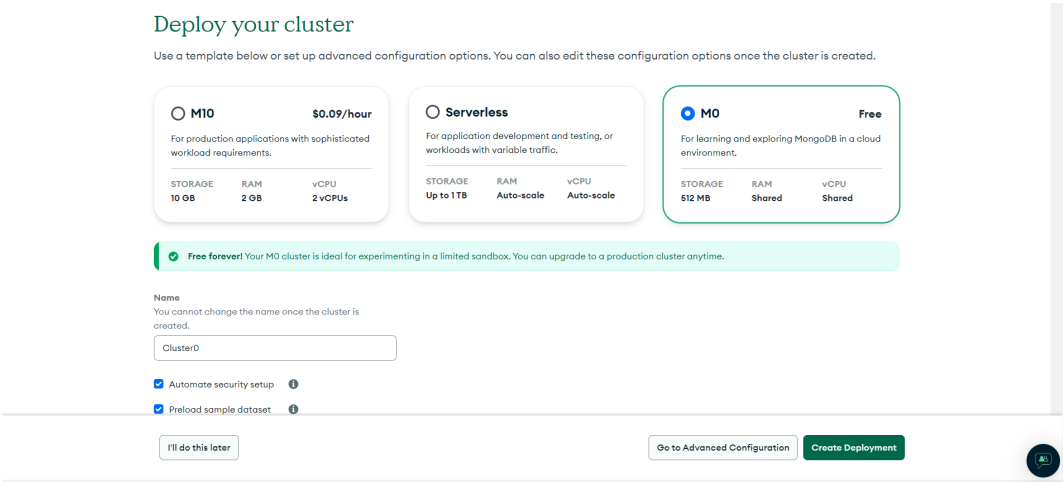


Figura 6.4: Configuración MongoDB Atlas (1)

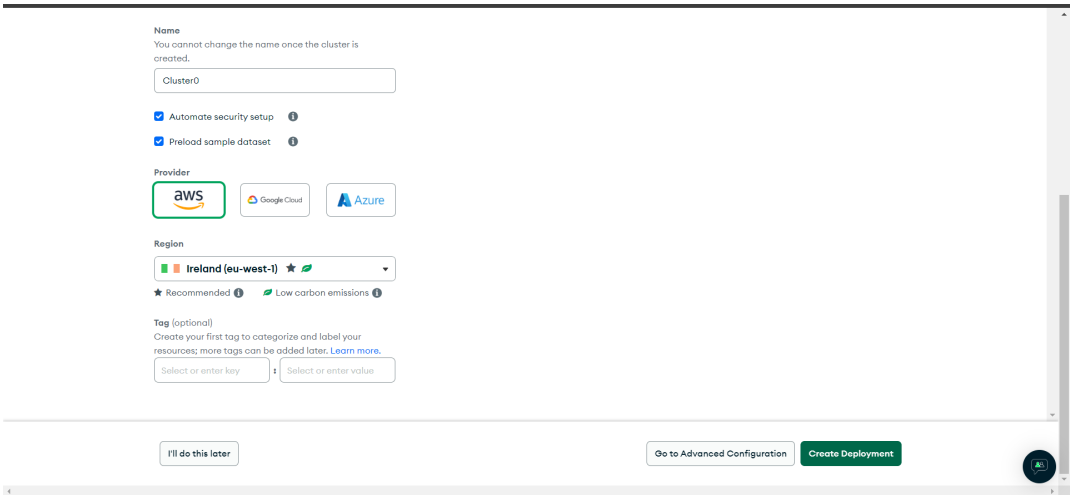


Figura 6.5: Configuración MongoDB Atlas (2)

Una vez realizado esto, redigirá a la pantalla de bases de datos y aparecerá un popup para realizar la conexión con el clúster que se acaba de crear, se recomienda guardar en alguna nota el nombre de usuario y la contraseña que aparecen por si hubiera que utilizarlo en el futuro. Después, hacer click en la opción create database user, con esto se creará la base de datos que se va a utilizar (no dejará avanzar o no se realizarán correctamente las configuraciones si no se crea). En la figura 6.6 se puede ver esto.

A continuación, hacer click en Choose a connection method. En la opción de seleccionar un driver y su versión dejar la que viene por defecto ya que las versiones que se están utilizando de node son superiores a la 5.5. El paso 2 no hace falta volverlo a realizar. En el paso 3, copiar y guardar esta url que aparece para realizar la conexión con la base de datos, ya que

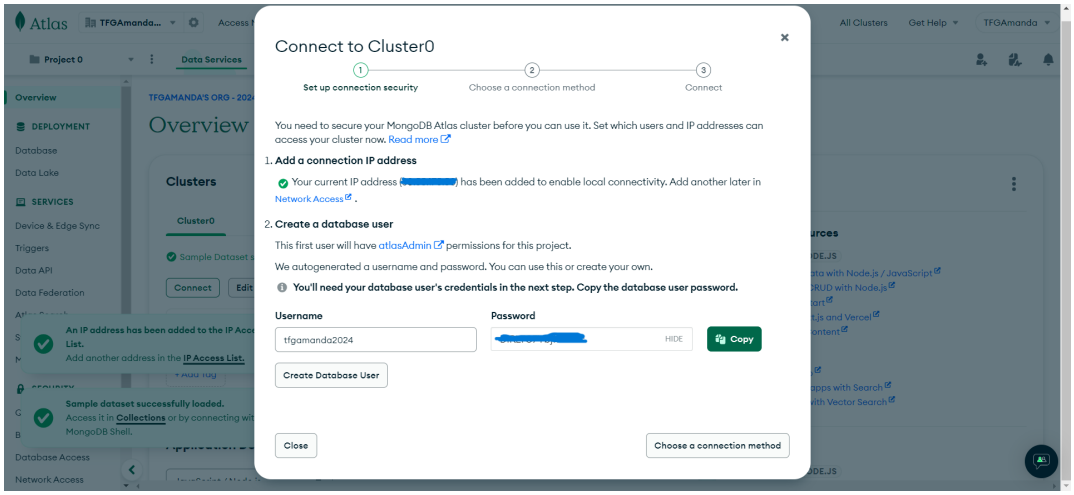


Figura 6.6: Conexión Clúster MongoDB Atlas (1)

no volverá a aparecer con la contraseña. En la figura 6.7 se puede ver esto.

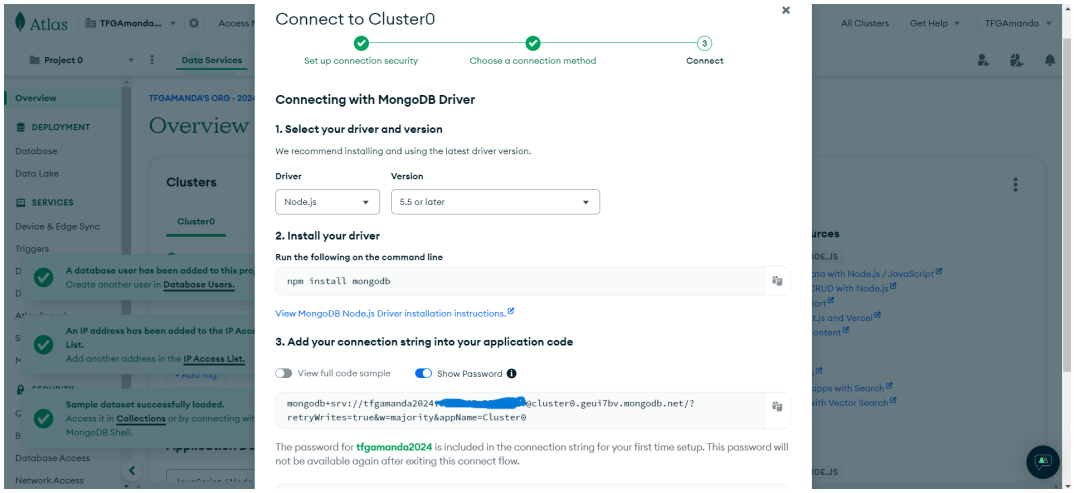


Figura 6.7: Conexión Clúster MongoDB Atlas (2)

Después, hay que realizar unas pequeñas modificaciones en el repositorio que se ha preparado solo para el despliegue. Dentro de este, en el archivo database.js, dónde se realiza la conexión con la url hay que cambiar la que había anteriormente para el despliegue en local por la que se ha guardado tal y cómo se indicaba en el párrafo anterior. Por último, se debe traer toda la lógica que hay en el archivo app.js al archivo index.js, ya que Vercel no trabaja bien si no se realizan en este último archivo. También se deberá cambiar la url indicada en el origen para que no de ningún fallo con el CORS, más adelante se indicará cuál se debe poner. El archivo database.js debe quedar asi:

```
import mongoose from "mongoose";
import User from './models/user.model.js';

// const URL = 'mongodb://localhost:27017/mydb';

const URL = 'mongodb+srv://tfg-vercel:<your-password>@tfg-vercel.hs7gbjr.
mongodb.net/?retryWrites=true&w=majority&appName=tfg-vercel';

export const connectDB = async () => {
    try {
        await mongoose.connect(URL);
        console.log(" >>> MongoDB connected to " + URL);
        await User.initRoles();
    } catch(error) {
        console.log("Error connecting to MongoDB:", error);
    }
}
```

Y el archivo index.js, debería quedar así:

```
import { connectDB } from './database.js';
import {cancelUnpaidAcceptedBookings} from './cronJobs.js';
import express from 'express';
import morgan from 'morgan';
import establishmentRoutes from './routes/establishments.routes.js';
import authRoutes from './routes/auth.routes.js';
import userRoutes from './routes/user.routes.js';
import petRoutes from './routes/pet.routes.js';
import bookingRoutes from './routes/booking.routes.js';
import paymentRoutes from './routes/payment.routes.js';
import rateRoutes from './routes/rates.routes.js';
import cors from 'cors';
import cookieSession from 'cookie-session';
import './cronJobs.js';


const app = express();

app.use(morgan('dev'));
app.use(express.json());
app.use(cors());

app.use(cors({
    //origin: 'http://localhost:4200',
    origin: 'https://tfg-amanda-morera-perez-deployment-frontend.vercel.app',
    methods: ['GET', 'POST', 'PUT', 'DELETE'],
    allowedHeaders: ['Content-Type', 'Authorization']
```

```
  }));

  app.use(
    cookieSession({
      name: "tfgamanda-session",
      keys: ["COOKIE_SECRET"],
      httpOnly: true
    })
  );

app.use('/api',establishmentRoutes);
app.use('/api',authRoutes);
app.use('/api',userRoutes);
app.use('/api',petRoutes);
app.use('/api',bookingRoutes);
app.use('/api',paymentRoutes);
app.use('/api',rateRoutes);

connectDB();
cancelUnpaidAcceptedBookings();
app.get("/", (req, res) => {
    res.json("Hello");
})
app.listen(3000);
console.log("Server running on port 3000");

export default app;
```

Se puede comprobar con la herramienta MongoDB Compass que se ha realizado correctamente, realizando una nueva conexión indicando la url generada por MongoDB Atlas, en la figura 6.8 se puede ver esto. Una vez realizada la conexión, hay que añadir los siguientes json haciendo click en add data, import JSON or CSV file.

Dentro de la base de datos test, en la colección establishments añadir el siguiente json:

```
    [{
  "_id": {
    "$oid": "65f8618a623c78b41b74c280"
  },
  "id": 1,
  "__v": 0,
  "address": "Calle DireccionPrueba1",
  "email": "contacto@example.com",
  "name": "NuevoEstablecimientoId1",
  "phone": "12456789"
}]
```

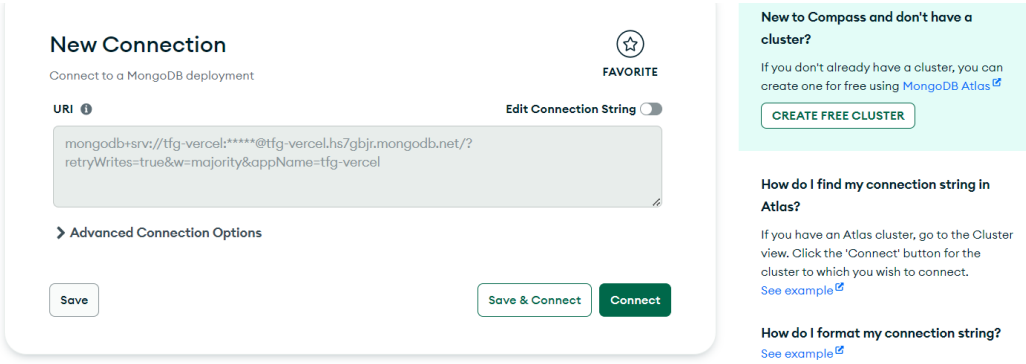


Figura 6.8: Conexión con MongoDB Compass

En la colección rates, añadir el siguiente json:

```
    [{
  "_id": {
    "$oid": "6600532f92c61db96ec59b2c"
  },
  "establecimiento_id": 1,
  "precio_alimentacion": 2,
  "precio_suelo_frio": 5,
  "precio_suelo_calido": 5,
  "precio_bano": 10,
  "precio_peluqueria": 15,
  "precio_corta_unas": 3,
  "__v": 0,
  "precio_full_time": 50,
  "precio_half_time": 25,
  "precio_hora": 8
}]
```

Y por último en la colección users, añadir el siguiente json para el usuario con el rol de admin:

```
    {
  "_id": {
    "$oid": "66278e13ff0fe0e82a0da1b4"
  },
  "email": "admin@example.com",
  "password": "$2a$12$X.xnOg4IzOKsPx58PYuXZe5QYdkOUcQMgtxYqYob5pFaCU7t4QF1G",
  "roles": [
    {
```

```
      "$oid": "660876d4aa215a91a085ac03"
    }
  ],
  "id": 1,
  "__v": 0
}
```

Nota: cambiar el campo **oid** por el valor que aparezca entre parentesis en ObjectId en el campo _id del registro con el nombre .ªdmin"de la colección roles. En la figura 6.9 se puede ver como quedaría el contenido de la colección roles, dónde se encuentra dicho campo oid y en la figura 6.10 se puede ver cómo quedaría la colección users cuando se vayan almacenando usuarios de acuerdo con los campos oid de la colección roles.

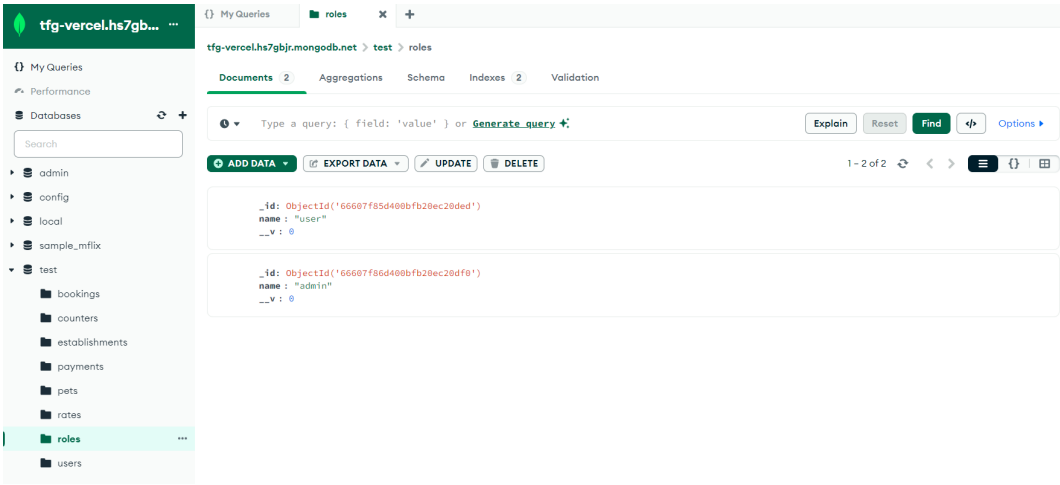


Figura 6.9: Colección roles MongoDB Compass

Una vez realizadas todas estas modificaciones y comprobaciones previas, se debe proceder a subir este repositorio para producción en GitHub ya que dentro de lo que permite Vercel, la plataforma elegida para la producción, es la opción que presenta menos problemas. Si no se dispone de una, crear la cuenta y subir el repositorio a la rama main (es la que viene por defecto). También otra ventaja al utilizar GitHub, es que al realizar cualquier cambio desde el propio editor que proporciona (no se necesitaría un IDE) y guardarlo, Vercel va a volver a realizar todo el despliegue por ella misma.

Después, se debe proceder a crear los proyectos necesarios para el despliegue en Vercel. Si no se dispone de una cuenta, se debe crear una. Una vez dentro de la cuenta, en la pantalla principal de proyectos hacer click en el botón Add new y hacer click en la opción de Project. Esto redirigirá a la pantalla para importar un repositorio de Git, iniciar sesión con nuestras credenciales de Git y seleccionar el repositorio del proyecto para el despliegue. A continuación, redirigirá a la pantalla de configuración del proyecto, en el apartado de project name se puede dejar el que viene por defecto o añadir algo como backend, server o api para hacer distinción de que este proyecto que se va a desplegar sólo contiene esta parte. En

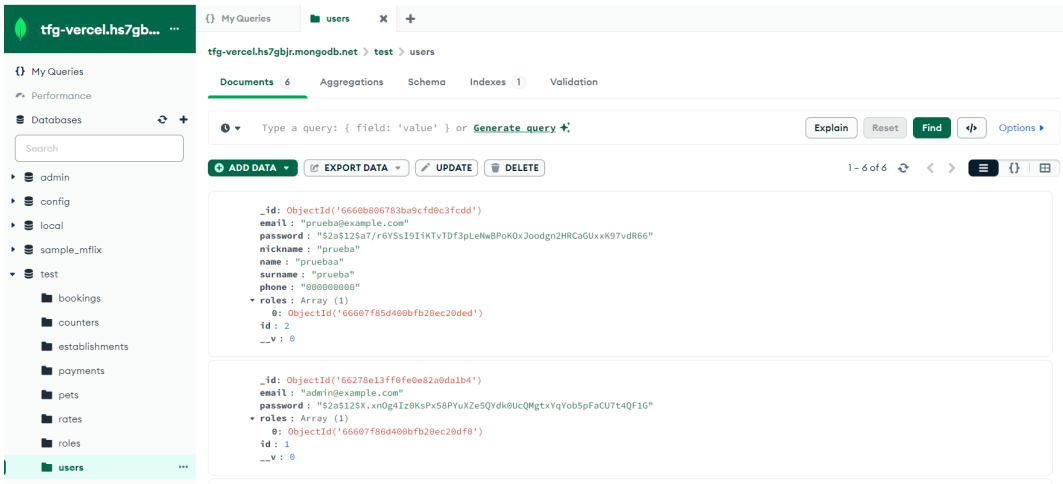


Figura 6.10: Colección users MongoDB Compass

el apartado de directorio raíz, indicar el directorio dónde se encuentre el servidor, en este caso server. En la opción de build and output settings no indicar nada y en el apartado de environment variables indicar si se tienen variables de entorno en este caso no hace falta indicar nada, en la Figura 6.11 se ve esto reflejado. Por último, hacer click en deploy y con esto si todo va bien redigirá a una página con título Congratulations, y ya estará desplegada la parte del backend del proyecto.

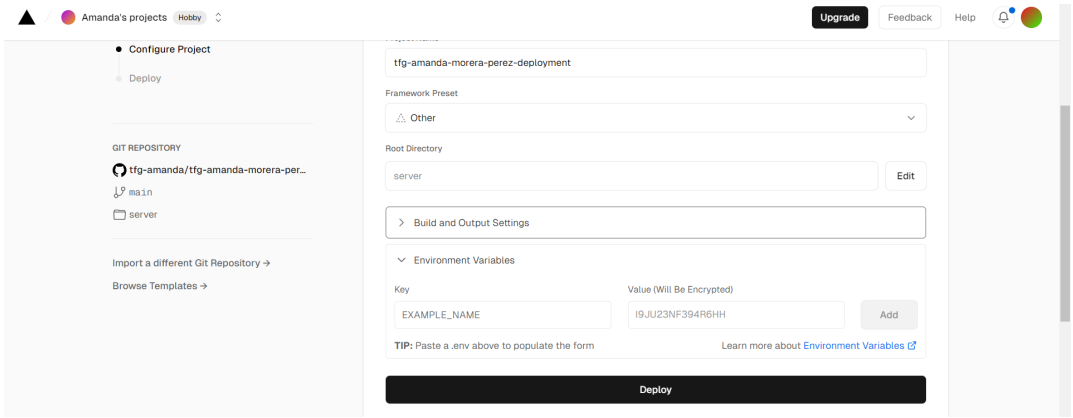


Figura 6.11: Configuración Vercel Proyecto Backend

A continuación, realizar de manera análoga a lo indicado en el párrafo anterior el despliegue de la parte del frontend de la aplicación. Cambiando el directorio raíz por el directorio dónde se encuentra el frontend, que en este caso se debe indicar client-app, en la Figura 6.12 se ve esto reflejado. Cabe destacar que este despliegue tarda más que el anterior y si todo va bien debería redirigir a la pantalla de felicitaciones. Y con esto ya estará desplegada la aplicación completa.

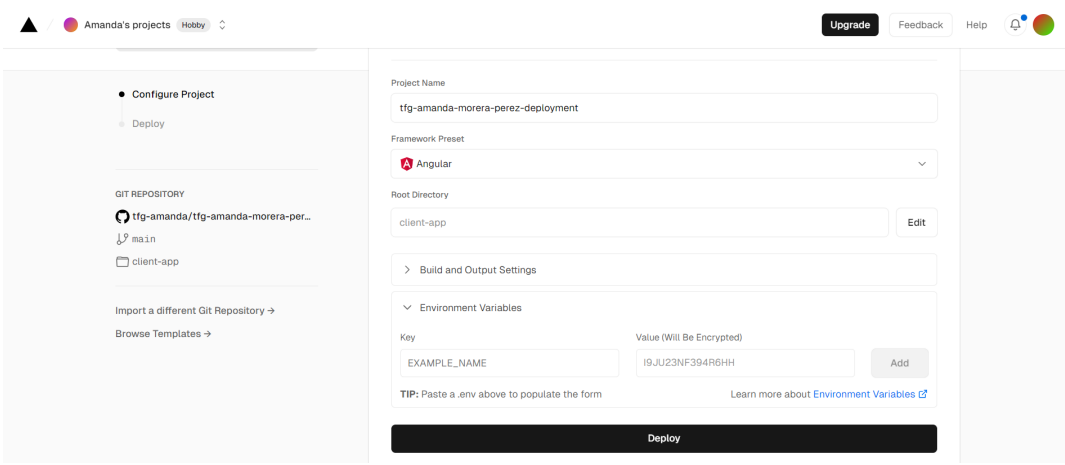


Figura 6.12: Configuración Vercel Proyecto Frontend

Para finalizar, lo último que se debe realizar para que todo funcione correctamente es cambiar las urls del despliegue en local en los servicios del frontend por la nueva url del proyecto del backend desplegado, en la Figura 6.13, se ve un ejemplo de cómo quedaría el BookingService. Esta url se obtiene haciendo click en el proyecto, lo que redirigirá al detalle del mismo y se debe indicar la url que viene en el apartado domains. En las Figuras 6.14, 6.15 y 6.16.

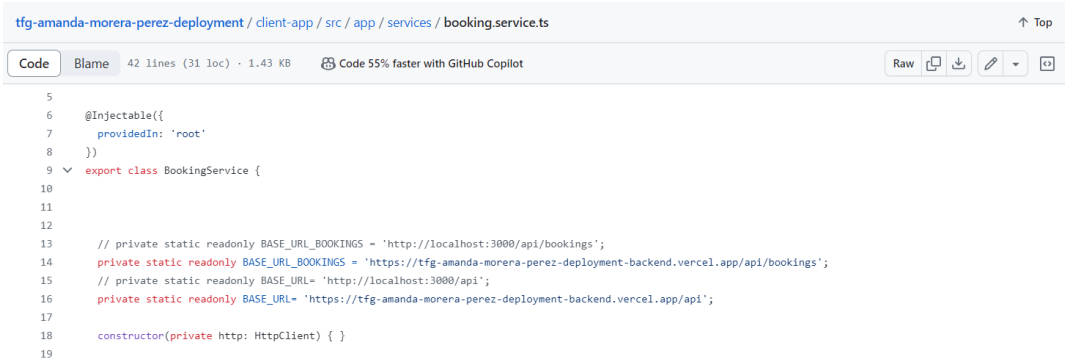


Figura 6.13: Ejemplo cambio url BookingService

Por último, cambiar la url en el archivo index.js dentro del directorio server (backend), indicando en el origin del cors la url del dominio del proyecto desplegado para el frontend, que se obtiene de manera análoga a cómo se ha indicado anteriormente para el backend, en la Figura 6.10 se puede ver como quedaría reflejado. Y ya estaría todo lo necesario para el correcto funcionamiento de la aplicación desplegada.

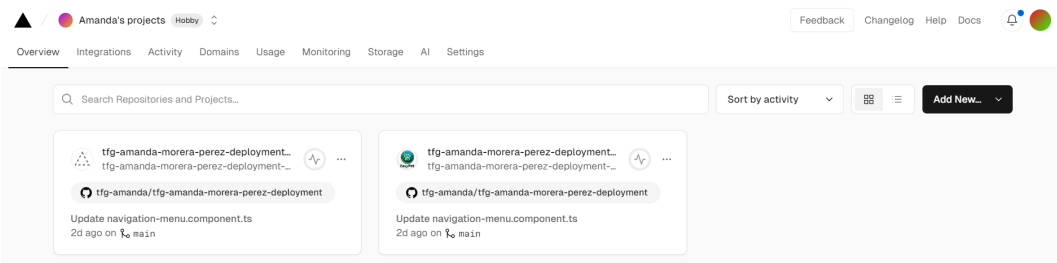


Figura 6.14: Dashboard Vercel

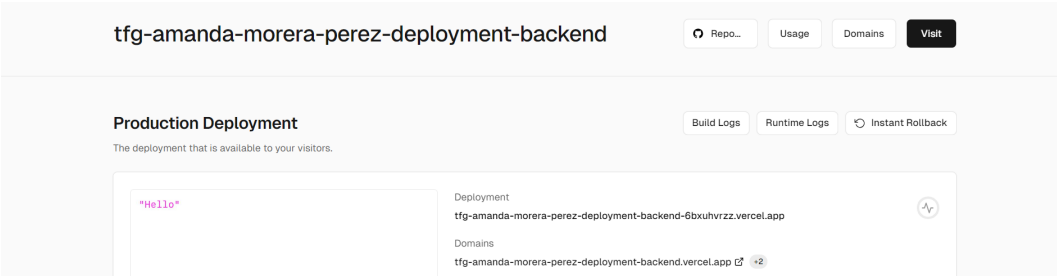


Figura 6.15: Detalle proyecto desplegado backend

Figura 6.16: Detalle proyecto desplegado frontend

6.2. Pruebas

Para llevar a cabo un control del código que se está implementando y que esté funcionando como se desea, se ha llevado a cabo la realización de pruebas unitarias. Para ello se han utilizado los frameworks Jasmine junto con Karma. En el siguiente apartado se detallará cómo se han realizado.

6.2.1. Tests unitarios

Un test unitario es es un bloque de código que verifica automáticamente la precisión de una unidad individual de código de aplicación, normalmente una función o un método. Estos tests están diseñados para verificar que el bloque de código se ejecuta según lo esperado, de acuerdo con la lógica del desarrollador. Son fundamentales en la fase de desarrollo ya que permiten detectar errores en etapas tempranas en las cuáles es más fácil corregir el error.

Cuando un test unitario involucra más de un componente o simula interacciones con dependencias externas como interacciones con servicios con el uso de mocks o elementos de prueba lo que permite centrarse en el test sin la influencia de estos elementos externos. En este caso, se pueden considerar pruebas de intergación a pequeña escala, ya que al utilizar los mocks para aislar y simular las partes del sistema que no son el foco principal de la prueba que se está realizando. Sin embargo, no se tratan de pruebas end-to-end (E2E), ya que en estas se simulan escenarios completos en el uso del software desde la perspectiva del usuario final, teniendo en cuenta la aplicación completa y su interacción con el entorno y sistemas externos.

La realización de test unitarios en Angular permite:

- Detección temprana de errores
- Mejorar la calidad del código
- Confianza en los cambios que se realicen en el código
- Ahorro de tiempo

Para poder ejecutar los tests correctamente, primero hay que generar el archivo karma.conf.js. Si aún no se ha generado, se puede crear y configurar con el siguiente comando: ng generate config karma.

Los archivos de prueba que serán ejecutados serán aquellos que se encuentran dentro del directorio o carpeta del propio componente, y se ejecutará sólo los que tengan la terminación spec.ts ya que son los que detecta la herramienta como archivos de prueba.

Para describir cómo se debería de realizar en cada componente, se va a detallar cómo se ha realizado y en qué consiste cada parte del archivo de prueba de un componente más sencillo, en este caso del App Component.

Primero, se va a mostrar el archivo app.component.ts y el app.component.html, y a continuación el archivo app.component.spec.ts, sobre los que se va a hacer referencia para explicar cómo se debe realizar un test y cómo se han realizado en este caso.

Fragmento de código 6.1: Archivo app.components.ts

```
1     import { ChangeDetectorRef, Component, OnInit, ViewChild } from '@angular/core';
2 import { MatSidenav } from '@angular/material/sidenav';
3 import { BreakpointObserver } from '@angular/cdk/layout';
4 import { Router } from '@angular/router';
```

```
5
6   @Component({
7     selector: 'app-root',
8     templateUrl: './app.component.html',
9     styleUrls: ['./app.component.scss']
10  })
11  export class AppComponent implements OnInit {
12    title = 'client-app';
13    menuDisabled:boolean = true;
14    isCollapsed = true;
15
16    @ViewChild(MatSidenav)
17    sidenav!: MatSidenav;
18
19    constructor(private observer: BreakpointObserver , private cdRef: ChangeDetectorRef
          , private router: Router) {}
20
21    ngOnInit() { }
22
23    ngAfterViewInit() {
24      this.sidenav = this.sidenav;
25      this.cdRef.detectChanges();
26    }
27
28    public navigateToProfileDetail ():void {
29      this.router.navigate(['profile']);
30    }
31
32    public navigateToBookings ():void {
33      this.router.navigate(['bookings']);
34    }
35
36    public navigateToPets ():void {
37      this.router.navigate(['pets']);
38    }
39
40  }
```

Fragmento de código 6.2: Archivo app.components.ts

```
1   describe('AppComponent', () => {
2     let component: AppComponent;
3     let fixture: ComponentFixture<AppComponent>;
4     let router: Router;
5
6     beforeEach(async () => {
7       await TestBed.configureTestingModule({
8         imports: [HttpClientTestingModule, RouterTestingModule, MatSidenavModule,
              NoopAnimationsModule, MatSidenavModule, MatListModule, MatIconModule],
9         declarations: [AppComponent, HeaderComponent, NavigationMenuComponent,
              FooterComponent]
10      }).compileComponents();
11
12      router = TestBed.inject(Router);
13      fixture = TestBed.createComponent(AppComponent);
14      component = fixture.componentInstance;
15      fixture.detectChanges();
16    });
17
18    it('should create the app', () => {
19      expect(component).toBeTruthy();
20    });
21
22    it('should navigate to profile detail on click', () => {
23      const spy = spyOn(router, 'navigate');
24      const profileLink = fixture.debugElement.query(By.css('a[mat-list-item]:nth-child
            (1)'));
25      profileLink.triggerEventHandler('click', null);
26      expect(spy).toHaveBeenCalledWith(['profile']);
27    });
```

```
28
29    it('should navigate to bookings on click', () => {
30      const spy = spyOn(router, 'navigate');
31      const bookingsLink = fixture.debugElement.query(By.css('a[mat-list-item]:
            nth-child(2)'));
32      bookingsLink.triggerEventHandler('click', null);
33      expect(spy).toHaveBeenCalledWith(['bookings']);
34    });
35
36    it('should navigate to pets on click', () => {
37      const spy = spyOn(router, 'navigate');
38      const petsLink = fixture.debugElement.query(By.css('a[mat-list-item]:nth-child(3)
            '));
39      petsLink.triggerEventHandler('click', null);
40      expect(spy).toHaveBeenCalledWith(['pets']);
41    });
42  });
```

Fragmento de código 6.3: Archivo app.components.ts

```
1     <app-header [sidenav]="sidenav" (isCollapsedChange)="isCollapsed = $event"></
          app-header>
2   <app-navigation-menu *ngIf="menuDisabled"></app-navigation-menu>
3
4   <mat-sidenav-container >
5     <mat-sidenav [opened]="!isCollapsed">
6
7       <mat-nav-list>
8         <a mat-list-item  (click)="navigateToProfileDetail()">
9           <span class="entry">
10            <mat-icon>account_circle</mat-icon>
11            <span >Ver perfil</span>
12          </span>
13        </a>
14        <a mat-list-item  (click)="navigateToBookings()">
15          <span class="entry">
16            <mat-icon>bookmarks</mat-icon>
17            <span >Reservas</span>
18          </span>
19        </a>
20        <a mat-list-item  (click)="navigateToPets()">
21          <span class="entry">
22            <mat-icon>pets</mat-icon>
23            <span >Mascotas</span>
24          </span>
25        </a>
26      </mat-nav-list>
27
28    </mat-sidenav>
29    <mat-sidenav-content>
30  <router-outlet></router-outlet>
31  </mat-sidenav-content>
32
33  </mat-sidenav-container>
34  <app-footer></app-footer>
```

Fragmento de código 6.4: Archivo app.components.ts

```
1       it('should create the app', () => {
2      expect(component).toBeTruthy();
3    });
```

Fragmento de código 6.5: Archivo app.components.ts

```
1       it('should navigate to profile detail on click', () => {
2      const spy = spyOn(router, 'navigate');
```

```
    const profileLink = fixture.debugElement.query(By.css('a[mat-list-item]:nth-child(1)'));
    profileLink.triggerEventHandler('click', null);
    expect(spy).toHaveBeenCalledWith(['profile']);
  });

  it('should navigate to bookings on click', () => {
    const spy = spyOn(router, 'navigate');
    const bookingsLink = fixture.debugElement.query(By.css('a[mat-list-item]:nth-child(2)'));
    bookingsLink.triggerEventHandler('click', null);
    expect(spy).toHaveBeenCalledWith(['bookings']);
  });

  it('should navigate to pets on click', () => {
    const spy = spyOn(router, 'navigate');
    const petsLink = fixture.debugElement.query(By.css('a[mat-list-item]:nth-child(3)'));
    petsLink.triggerEventHandler('click', null);
    expect(spy).toHaveBeenCalledWith(['pets']);
  });
```

Fragmento de código 6.6: Archivo app.components.ts

```
    import { TestBed } from '@angular/core/testing';
import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';
import { AuthService } from './auth.service';

describe('AuthService', () => {
  let service: AuthService;
  let httpMock: HttpTestingController;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [AuthService]
    });

    service = TestBed.inject(AuthService);
    httpMock = TestBed.inject(HttpTestingController);
  });

  afterEach(() => {
    httpMock.verify();
  });
```

Fragmento de código 6.7: Archivo app.components.ts

```
     it('should login and set loggedIn to true and isAdmin to false', () => {
    const mockUser = {
      id: '123',
      email: 'test@test.com',
      password: 'password',
      name: 'Test',
      surname: 'User',
      phone: '1234567890',
      roles: ['ROLE_USER']
    };

    service.login({ email: 'test@test.com', password: 'password' }).subscribe(user => {
      expect(user).toEqual(mockUser);
      service.isLoggedIn().subscribe(loggedIn => {
        expect(loggedIn).toBeTrue();
      });
      service.getIsAdmin().subscribe(isAdmin => {
        expect(isAdmin).toBeFalse();
```

```
19        });
20      });
21
22      const req = httpMock.expectOne('http://localhost:3000/api/auth/signin');
23      expect(req.request.method).toBe('POST');
24      req.flush(mockUser);
25    });
```

Lo primero que debe aparecer siempre en un fichero de pruebas es la función describe. Dentro de esta se van a crear un grupo o conjunto de pruebas, a veces también se le denomina 'suite'. Esta función tiene dos parámetros: el primero es un string que sirve para dar una descripción y una idea sobre qué se va a testear (un componente, un servicio, un pipe ...), en nuestro caso, y cómo se ve en el fragmento de código 6.1, hemos indicado 'AppComponent' ya que es el componente sobre el que vamos a implementar las pruebas.; el segundo sería el conjunto de pruebas que se explicará algo más en detalle.

Lo siguiente que se realiza es declarar una variable del propio componente, y otra para el componentFixture del componente, proporciona acceso al componente y sus métodos, propiedades y a su entorno de pruebas para poder interactuar con él y verificar su comportamiento. Después, implementamos la función beforeEach que ejecuta algunas configuraciones antes de que se ejecute cada test, con async antes para asegurar que sea asíncrono.

Dentro de este se usa el TestBed.configureTestingModule, cuya función es similar a la de NgModule, se utiliza para configurar e inicializar el entorno para los tests unitarios y proporciona los métodos para crear componentes y servicios en test unitarios, sólo que en este caso solo importamos los módulos necesarios que se vayan a usar en este componente, y se declaran otros componentes si los utiliza. En este caso, se importan el HttpClientTestingModule, que inyecta el HttpClientTestingController para esperar y eliminar solicitudes en los tests; el RouterTestingModule, que configura el router que será utilizado para los test; y, por último, los propios módulos que utiliza el componente. Se declaran los componentes que vaya a utilizar este y el propio componente en sí y por último se llama a la función compileComponents(). Después de esto, se le asigna a la variable fixture el valor devuelto tras llamar TestBed.createComponent(AppComponent), que lo que hace es crear el componente, y a la variable component se le asigna una instancia del componente. Todo esto se puede ver en el Fragmento de código 6.2.

A continuación, se procede a describir cada uno de los tests que se van a realizar o se van a probar. En este caso nos interesa comprobar que el componente se crea correctamente con todos sus elementos y que luego, cuando se haga click en cada uno de los elementos de la lista del menú lateral, se redirija correctamente a dónde corresponda. Estos elementos se pueden ver en el Fragmento de código 6.3.

Para implementar cada test, se utiliza la función it que define un único test. Debe contener uno o más expect, para indicar lo que se espera obtener en la realización del test. Si los resultados de estos expects son exitosos pasará el test y en caso contrario fallará. El primer argumento es un string que da una descripción de lo que se espera comprobar en ese test, se suele utilizar should seguido de lo que vaya a hacer.

Si se quiere que sólo se ejecute un test en concreto, se debe poner fit en lugar de it. También cabe destacar que los tests no se ejecutan en un orden secuencial según estén declarados sino

en un orden aleatorio.

En el primer test se comprueba que el componente se haya creado correctamente. Para ello se espera que el componente sea truthy, por lo que no podría ser false, 0, null, o undefined. Si se da que es alguno de estos casos, fallaría el test. Todo esto se puede ver en el Fragmento de código 6.4.

El resto de los tests de este archivo son análogos por lo que solo se va a explicar uno de ellos. La finalidad de este test es comprobar que redirige correctamente al componente del detalle del perfil, cuando se hace click. Primero, se utiliza la función spyOn (router, 'navigate'), que lo que hace es reemplazar la función navigate por un espía y por lo tanto registra cada vez que se llama a la función router.navigate. Después se procede a obtener el enlace mediante la siguiente línea: "const profileLink = fixture.debugElement.query(By.css('a[mat-list-item]:nth-child(1)'));".

En primer lugar, fixture.debugElement, es una propiedad del fixture que permite el acceso a la instancia del componente y a su DOM. Después, la función query permite buscar algún elemento del DOM del componente con lo indicado. En este caso, se ha utilizado query(By.css), que buscará el elemento que coincida con el selector CSS indicado. En este caso, 'a[mat-list-item]:nth-child(1)', selecciona el primer elemento <a>de la lista creada con mat-list-item. A continuación, se procede a simular el evento de hacer click en el enlace mediante la función triggerEventHandler, en esta línea "profileLink.triggerEventHandler('click', null);". Por último, se utiliza la función expect para comprobar que se ha llamado a la función indicada en el espía con el argumento 'profile'. Todo esto puede verse en el Fragmento de Código 6.5.

Para explicar cómo se debería proceder en el caso de que lo que se quiera probar sea un servicio, se va a explicar cómo se ha realizado un test en el servicio de autenticación para el caso en el que se hace login con el rol de User.

Tal y cómo se ve en en el Fragmento de Código 6.6, lo primero que se hace, análogo al anterior, es configurar todo lo necesario para realizar los tests. En este caso se importa el módulo de HttpClientTestingModule para las peticiones http, y el servicio de autenticación, AuthService. A continuación, se inyectan las dependencias con la función TestBed.inject, que permite obtener las instancias de los servicios necesarios para las pruebas. En este caso se inyecta el servicio que se está probando, y un mockHttp que es una versión de prueba o simulada del servicio HTTP que se utiliza para imitar y probar las solicitudes y respuestas HTTP sin realizar llamadas reales a un servidor. Finalmente, a diferencia del anterior, se define el bloque afterEach, dónde se llama a la función httpMock.verify para verificar que no queda ningún solicitud HTTP pendiente.

Tal y cómo se ve en el Fragmento de Código 6.7, para realizar el test que comprueba que el login del authService con el rol de User funciona correctamente, primero se crea un usuario de prueba, mockUser. Después, se llama a la función login y se subscribe al valor que devuelve esta llamada, que se trata del usuario. Se comprueba que el valor devuelto, user, sea igual que el usuario de prueba.

Después, primero se suscribe a los valores que devuelven los métodos isLoggedIn y getIsAdmin, para ver los cambios que se producen en estos valores y realizar las pruebas. Para

ello se utiliza la función expect() de Jasmine. En el caso de isLoggedIn se espera que el valor que devuelva sea verdadero, por eso se especifica tobeTrue(), lo que indicaría que el usuario ha iniciado sesión correctamente. En el caso de isAdmin, se espera que el valor devuelto sea falso, por eso se especifica toBeFalse(), lo que quiere decir que no ha iniciado sesión con el rol de admin. También se podía haber indicado con not.toBeTrue().

A continuación, se utiliza la función expectOne(), para comprobar que se realiza una solicitud HTTP a la url que se le indica como parámetro y se comprueba que esa solicitud sea POST. Por último, se utiliza la función flush(), para completar la llamada HTTP y simular la respuesta del servidor, devolviendo así el usuario de prueba como resultado, ya que es lo que devuelve la respuesta original.

Para ejecutar los tests implementados, bastaría simplemente con lanzar el comando ng test. En nuestro caso, como queremos que nos aparezca la cobertura que se logra con la implementación de nuestros tests, ejecutaremos ng test –code-coverage. Al hacer esto, se creará el directorio coverage en la raíz del proyecto, y el motor de pruebas, Karma, se encargará de abrir una nueva ventana en el navegador que hayamos indicado, mostrando la lista de todos los tests que haya encontrado y se hayan ejecutado correctamente o si se ha obtenido algún error, tal y cómo se puede ver en la Figura 6.17.

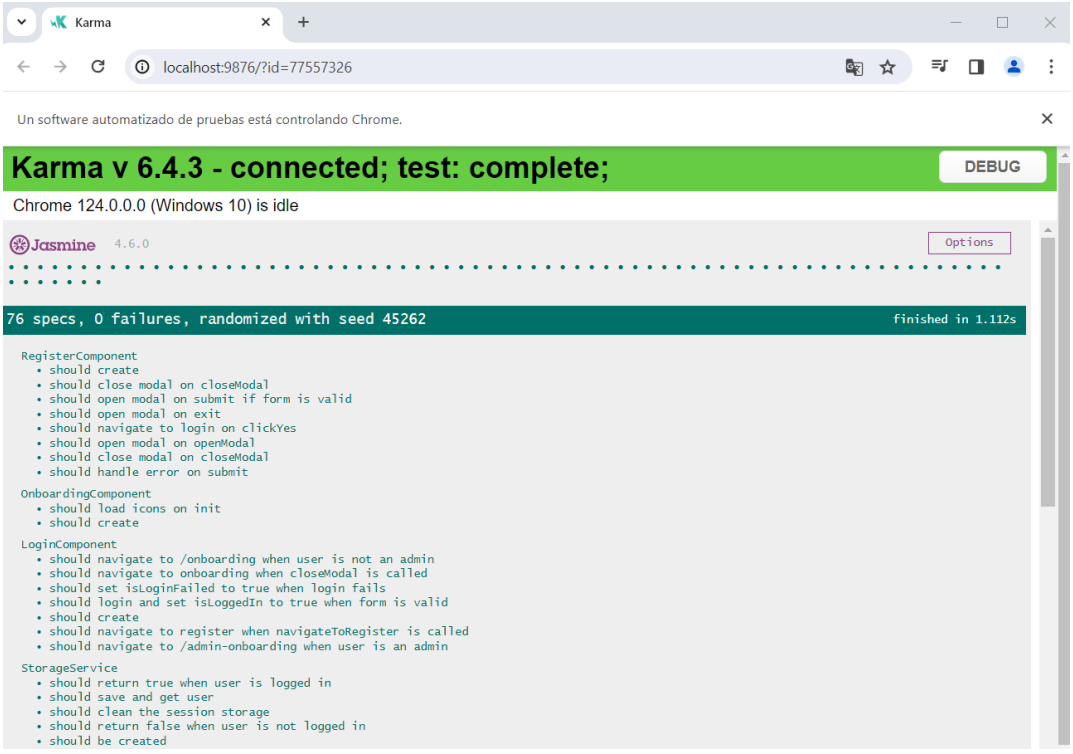
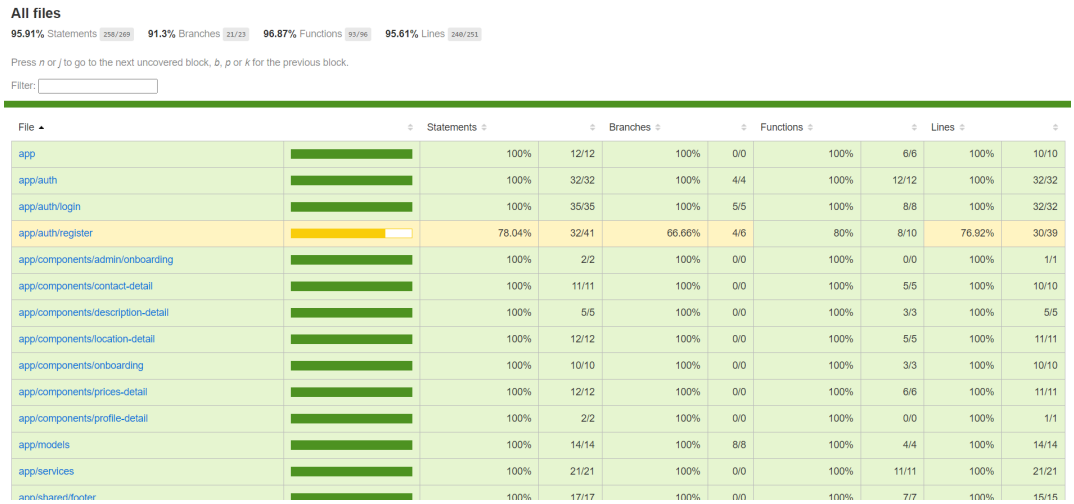


Figura 6.17: Ejecución test - Navegador Chrome

Dentro del directorio coverage creado, en el archivo index.html encontramos el informe con toda la cobertura a la que se llega con los tests y se puede comprobar que se prueba todo

correctamente. En la Figura 6.18 se puede ver el contenido del mismo, y se puede observar que aparece en verde. Es decir, que está realmente probado a partir del 80 %.

Se ha dejado intencionadamente el componente register sin probar completamente para que se aprecie como se ve en un color amarillo cuando no llega al porcentaje de cobertura esperado, entre un 50 % y menos de 80 %. Cuando está por debajo del 50 %, aparece en color rojo. Además, se puede apreciar de izquierda a derecha el porcentaje probado o que se alcanza de: declaraciones, ramas, funciones y líneas.



All files

95.91% Statements 258/269 **91.3%** Branches 21/23 **96.87%** Functions 93/96 **95.61%** Lines 240/251

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

Filter:

File	Statements		Branches		Functions		Lines	
app	100%	12/12	100%	0/0	100%	6/6	100%	10/10
app/auth	100%	32/32	100%	4/4	100%	12/12	100%	32/32
app/auth/login	100%	35/35	100%	5/5	100%	8/8	100%	32/32
app/auth/register	78.04%	32/41	66.66%	4/6	80%	8/10	76.92%	30/39
app/components/admin/onboarding	100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/components/contact-detail	100%	11/11	100%	0/0	100%	5/5	100%	10/10
app/components/description-detail	100%	5/5	100%	0/0	100%	3/3	100%	5/5
app/components/location-detail	100%	12/12	100%	0/0	100%	5/5	100%	11/11
app/components/onboarding	100%	10/10	100%	0/0	100%	3/3	100%	10/10
app/components/prices-detail	100%	12/12	100%	0/0	100%	6/6	100%	11/11
app/components/profile-detail	100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/models	100%	14/14	100%	8/8	100%	4/4	100%	14/14
app/services	100%	21/21	100%	0/0	100%	11/11	100%	21/21
app/shared/footer	100%	17/17	100%	0/0	100%	7/7	100%	15/15

Figura 6.18: Test Coverage - Archivo index.html

Capítulo 7

Seguimiento del proyecto

En este capítulo se va a detallar cómo se ha ido desarrollando el proyecto, comentando las tareas que se han llevado a cabo en cada sprint, comparándolas con las planificadas y su estimación. También se explican los cambios realizados tras los eventos realizados de Scrum, y el tiempo invertido en cada una de las tareas, comparándolo con la estimación de cada una, mediante tablas en las que se resume todo lo anterior.

7.1. Sprint 0. 04/02/2024 - 16/02/2024

En este Sprint y el siguiente se llevaron a cabo todas las tareas relacionadas con la planificación, análisis y diseño de la aplicación web. En concreto, este Sprint tuvo una duración algo menor al resto ya que duro 12 días mientras que el resto se planificaron para una duración de dos semanas.

Se empleó casi la mitad del tiempo en formación, realizando tutoriales de repaso de React, ya que se barajaban dos opciones para realizar la implementación en el frontend: Angular o React. Se tomó la decisión en este sprint, por motivos de experiencia y dado que también se iba a utilizar en las prácticas extracurriculares, de utilizar Angular como tecnología para el frontend.

Se realizó una investigación sobre qué tecnologías utilizar para el backend y finalmente se optó por utilizar Node.js con Express y MongoDB. También se realizó una investigación sobre qué usar para el despliegue de la aplicación en un principio se decidió usar AWS cómo primera opción y por ello se realizó el presupuesto basado en ello aunque se decidió tener en cuenta otras dos posibles opciones: Vercel o Azure, por si esta primera opción no fuera viable.

El trabajo de planificación realizado en este Sprint se puede ver en el apartado 2.3, en el que se detalla la duración en sprints y lo que se va a realizar en cada uno de ellos. Por último,

se realizó el diseño de la interfaz de usuario cuyo resultado se puede ver en el apartado 5.7. Se definieron y detallaron las épicas cuyo resultado se puede ver en el apartado 2.5. A partir de esto, se definieron y detallaron los requisitos funcionales expresados como historias de usuario, los requisitos no funcionales expresados como historias de usuario y los requisitos de información expresados como historias de usuario, cuyo resultado se puede ver en el apartado 2.6.

Por último, se realizó un modelo de dominio inicial y se elaboró el plan de riesgos, dejando realizados los apartados 2.1.1, 2.1.2, 2.1.3 relativos al marco de trabajo en Scrum utilizado para este proyecto.

En la Tabla 7.1 se puede ver un resumen de todas las historias de usuario realizadas, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondiente en horas, las horas realizadas finalmente y el estado de las tareas.

Cabe destacar que el tiempo empleado finalmente en la realización de las tareas fue la mitad debido a que a mitad del Sprint tuvo lugar la incorporación en la empresa, como se ha mencionado anteriormente, y, por lo tanto, no se dedicó más tiempo a estudiar la otra tecnología para el frontend que se estaba barajando ya que se decidió usar la misma que en las prácticas extracurriculares y sobre la cuál ya se tenía conocimiento. Esto también influyó en el resto de tareas ya que a todas se les dedicó menos tiempo de lo estimado inicialmente. Además se ve que la estimación inicial para la realización de estas tareas era errónea ya que se realizó una sobreestimación y finalmente se dedicó la mitad del tiempo en completar todas las tarea..

7.2. Sprint 1. 17/02/2024 - 01/03/2024

Este Sprint ya tuvo una duración como la del resto, de dos semanas. En él lo primero que se hizo fue tras la Sprint Review realizar los cambios vistos en ella, que fueron: en el modelo de dominio inicial, en los requisitos funcionales, en los diseños de las interfaces de usuario y en el plan de riesgos inicial.

Tras la Sprint Planning y, siguiendo la planificación, este Sprint se dedicaría al diseño para así poder empezar a implementar la aplicación en el siguiente.

En concreto las tareas que se llevaron a cabo fueron: terminar de redactar los apartados 2.1.4 y 2.1.5, relativos al marco de trabajo Scrum utilizado en el proyecto. También se elaboró el presupuesto del proyecto cuyo resultado se puede ver en el apartado 2.4. Además, se elaboró un modelo de proceso de negocio inicial.

Tras la Scrum Weekly, se realizaron pequeñas modificaciones en el apartado de requisitos, quedando ya el resultado final que se puede ver en el apartado 2.6, en el modelo de dominio inicial y en los riesgos iniciales quedando como resultado final lo que se puede ver en el apartado 2.2.

Por último, en este Sprint se había planificado realizar y redactar en la memoria el

apartado de la arquitectura ensi miramo el diseño junto con los seguimientos de este sprint y el anterior, el 0, pero no pudieron llevarse a cabo por lo que se trasladaron al siguiente.

En la Tabla 7.2 se puede ver un resumen de todas las historias de usuario realizadas, y las que no, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondiente en horas, las horas realizadas finalmente y el estado de las tareas al final del sprint.

Cabe destacar que el tiempo empleado fue mucho menor a la estimación inicial ya que al tratarse de los primeros Sprints y debido a la inexperencia se pensó que podían llevar más tiempo.

7.3. Sprint 2. 02/03/2024 - 15/03/2024

Durante este Sprint se completaron en primer lugar las tareas que no se realizaron en el anterior, y se empezó con la implementación de la aplicación siguiendo la planificación. En concreto se llevo a cabo la HU-01-01.

Antes de comenzar la implementación, se realizaron los cambios acordados tras la Sprint Review, siendo los siguientes: se añadieron dos nuevas historias de usuario para obtener las reservas mediante un filtrado, se puede ver en la Tabla 2.28; se realizaron los últimos cambios en el modelo de dominio inicial quedando ya como resultado final lo que se puede ver en el apartado 3.1; se realizaron los últimos cambios en el modelo de proceso de negocio relativo a realizar una reserva y se realizó un último diagrama relativo al día en el que se lleva a cabo la reserva, cuyo resultado se puede ver en el apartado 3.2.

Tras la Scrum Weekly, se llevaron a cabo las siguientes tareas nuevas: se realizó un diagrama de componentes inicial para así poder empezar mejor la implementación, se realizó un diagrama de despliegue inicial y se realizaron cambios en el apartado de arquitectura, entre ellos descartar algunos patrones y principios de diseño que no se podían realizar correctamente o no cuadraban con las características del proyecto. En concreto se descartaron el principio de inversión de dependencias y los patrones estrategia y comando.

Para completar dicha historia de usuario mencionada anteriormente, se realizaron las siguientes tareas: realización formación previa en el backend, ya que se trataba de una tecnología nueva; preparación del entorno de programación; implementación y desarrollo del backend e implementación y desarrollo del frontend.

En la Tabla 7.3 se puede ver un resumen de todas las historias de usuario realizadas, y las que no, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondiente en horas, las horas realizadas finalmente y el estado de las tareas.

Sin embargo, en este Sprint al contrario que con los anteriores, la estimación inicial fue menor que la que realmente se empleó debido a que, al empezar la implementación y todavía no tener suficiente experiencia, en la HU_01_01 se empleó el doble de tiempo de lo estimado. Aun así, tampoco se excedió mucho en la cantidad de horas planificadas por Sprint que son de unas 40 aproximadamente.

7.4. Sprint 3. 16/03/2024 - 29/03/2024

Durante este Sprint, había una pequeña probabilidad de que no se completasen todas las tareas planificadas para el mismo ya que podía materializarse el riesgo 2.2, coincidiendo con el período de vacaciones de Semana Santa. Finalmente, se pudieron completar todas las tareas planificadas ya que se dedicaron horas de trabajo a mayores aunque no se tratase de un período lectivo. También, la Sprint Review y Sprint Retrospective se realizaron junto con la primera Scrum Weekly del siguiente Sprint ya que coincidía con un día festivo.

Primero, se realizaron los cambios tras la Sprint Review, en concreto unos cambios en el frontend de la historia de usuario HU_01_01, cuya descripción se puede ver en la Tabla 2.25, y cambios en los diagramas de despliegue y componentes cuyo resultado se puede ver en las secciones 5.8 y 5.9.

Con respecto a la redacción de la memoria, se añadió el diseño UX inicial cuyo resultado se puede ver en la sección 5.7, se redactó el seguimiento del Sprint 2 cuyo resultado se puede ver en 7.3, se añadieron las tablas de los seguimientos del Sprint 0 y 1 cuyo resultado se puede ver en las secciones 7.1 y 7.2, respectivamente. Por último, se realizaron cambios en varias secciones del capítulo de arquitectura, quedando como resultado lo que se puede ver en las secciones 5.1, 5.2, 5.4 y 5.6.

En cuanto a la implementación, se completaron las historias de usuario: HU_01_02, HU_01_03 y HU_01_05 cuya descripción se puede ver en la Tabla 2.25.

En la Tabla 7.4 se puede ver un resumen de todas las historias de usuario realizadas, y las que no, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondiente en horas, las horas realizadas finalmente y el estado de las tareas.

En este Sprint, ocurrió al contrario que el anterior, ya que finalmente se empleó la mitad del tiempo estimado. Esto se debe en concreto a que el tiempo empleado para la implementación de las historias de usuario HU_01_02, HU_01_03 y HU_01_05 fue mucho menor ya que al estar relacionadas con la HU_01_01, se podían utilizar muchas de las cosas ya implementadas en la anterior.

7.5. Sprint 4. 30/03/2024 - 12/04/2024

La mayor parte del tiempo en este Sprint se dedicó a seguir con la implementación y añadiendo funcionalidad a la aplicación. En concreto, se realizaron las siguientes historias de usuario: HU_01_04, HU_02_01, HU_02_02, HU_05_01 y HU_05_02. Con esto, la primera épica quedaría finalizada.

Se lograron completar correctamente todas las historias asignadas al sprint, por lo que la planificación en cuánto al número de historias de usuario que se podían realizar era correcto. En este Sprint, ocurrió parecido al anterior, ya que finalmente se empleó menos de la mitad del tiempo estimado. Esto se debe a que la estimación realizada inicialmente era mayor de lo necesario ya que ahora se ha adquirido algo más de experiencia y no es necesario emplear

tanto tiempo para realizar las historias de usuario. En concreto, en este Sprint podemos ver como la primera historia de usuario realizada conlleva la mayor parte del tiempo mientras que en el resto, a medida que se iban realizando, el tiempo era menor.

En cuanto a la memoria, en este Sprint simplemente se añadió el seguimiento del anterior y se redactó el actual.

En la Tabla 7.5 se puede ver un resumen de todas las historias de usuario realizadas, y las que no, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondiente en horas, las horas realizadas finalmente y el estado de las tareas.

7.6. Sprint 5. 13/04/2024 - 26/04/2024

Para este Sprint se planificó realizar los tests de la implementación realizada hasta el momento. Se puede ver más detalle sobre esto en el Capítulo 6, de implementación y pruebas. Además, se planificó el desarrollo de la implementación de las historias de usuario HU_03_01 y HU_03_02, relacionadas con las mascotas de cada cliente, para seguir añadiendo funcionalidad a la aplicación web.

Debido a la coincidencia durante el Sprint con un período vacacional del estudiante, no se llegaron a completar todas las tareas planificadas para el mismo, trasladando al siguiente las tareas relativas a la redacción del documento de la memoria, en este caso las relativas a la redacción del capítulo sobre implementación y pruebas. Se podría decir que se vio materializado el riesgo 2.2, por lo que para ello se dedicarán horas extras en el siguiente Sprint para completar las tareas que no se terminaron en este. El resto de tareas relativas a la implementación de las historias de usuario se pudieron completar con éxito.

En la Tabla 7.6 se puede ver un resumen de todas las historias de usuario realizadas, y las que no, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondiente en horas, las horas realizadas finalmente y el estado de las tareas.

7.7. Sprint 6. 27/04/2024 - 10/05/2024

Para este Sprint, se planificó realizar primero las tareas que no se completaron del anterior, relativas a la redacción de los apartados del capítulo sobre implementación y pruebas. Además, se planifició desarrollar la implementación de las historias de usuario HU_03_03, HU_03_04 y HU_04_03.

Durante este Sprint, debido a que, además de no tener lugar ninguna circunstancia adversa para la estudiante, y a una menor carga durante las horas de prácticas extracurriculares, se pudieron realizar todas las tareas planificadas para este sprint incluso antes de la fecha de fin del mismo y las que se arrastraban del anterior. Cabe destacar que la realización de las historias de usuario HU_03_03, HU_03_04 consumió bastante menor tiempo del

planificado. Sin embargo, la historia de usuario HU_04_03 llevo más tiempo de lo planificado debido a problemas de maquetación y tipados que aparecieron durante la implementación.

En la Tabla 7.7 se puede ver un resumen de todas las historias de usuario realizadas, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondiente en horas, las horas realizadas finalmente y el estado de las tareas.

7.8. Sprint 7. 11/05/2024 - 24/05/2024

Para este Sprint se planificaron, en cuanto a tareas relativas a la redacción de la memoria: iniciar el capítulo 4 relativo a las tecnologías utilizadas ya que llegados a este punto no se van a añadir muchas más, y redactar algún párrafo de introducción que estaban pendientes. En cuanto a la implementación, se ha planificado realizar las historias de usuario HU_04_01 y HU_04_02

Durante este Sprint, el cuál era decisivo en cuanto a si se cumpliría con la planificación, se dedicaron una cantidad de horas algo mayor en comparación con los Sprints anteriores, completando todas las historias planificadas para el mismo. A mayores, se completó la historia de usuario HU_04_05 y se decidió suprimir la historia de usuario HU_04_06 ya que en lugar de proporcionar una opción en el filtrado para obtener las reservas ordenadas de la fecha más reciente a la más antigua, se muestran ya ordenadas siguiendo este criterio cuando se muestre la lista de todas las reservas asociadas a un usuario.

También, se realizó un reajuste en la estimación inicial ya que esta era demasiado elevada y teniendo en cuenta el avance del proyecto, el esfuerzo necesaria para llevarlas a cabo era menor. En concreto, se decidió estimar la HU_04_05 en 1,5 puntos y la HU_04_06 en 1 punto. Incluso con este reajuste cabe destacar que se realizaron en un tiempo bastante menor.

En la Tabla 7.8 se puede ver un resumen de todas las historias de usuario realizadas, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondientes en horas, las horas realizadas finalmente y el estado de las tareas.

7.9. Sprint 8. 25/05/2024 - 07/06/2024

Para este Sprint se planificó realizar todas las historias de usuarios restantes, haciendo un reajuste que se explicará en el párrafo siguiente, junto con la implementación de los tests unitarios restantes y la realización del despliegue.

El reajuste de las historias de usuario restantes, se realizó al igual que en el Sprint anterior, teniendo en cuenta el avance acumulado en el proyecto. Cabe destacar que incluso con este reajuste se realizaron en un tiempo mucho menos. En concreto se reajustaron las siguientes historias de usuario: la HU_05_05 pasó de 3 puntos a 0,5 puntos, la HU_05_03 pasó de 3 puntos a 1 punto, la HU_05_04 pasó de 5 puntos a 2 puntos, la HU_05_05 pasó de 3 puntos a 0,5 puntos, la HU_05_06 pasó de 3 puntos a 1 punto, la historia HU_05_07 pasó

de 3 puntos a 1 punto, la HU_04_04 pasó de 3 puntos a 0,5 puntos, la HU_03_05 pasó de 3 puntos a 1 punto, la HU_03_06 paso de 5 puntos a 1 punto, la HU_02_03 pasó de 2 puntos a 1 punto, la HU_02_05 pasó de 3 puntos a 0,5 puntos y la HU_02_04 pasó de 3 puntos a 1 punto.

Durante este Sprint se lograron realizar todas las tareas planificadas y se tomaron las siguientes decisiones: se decidió suprimir la opción de que el usuario pudiese editar su perfil, añadiendo una foto suya, y dejar por defecto el icono que se muestra, y de añadir una foto de la mascota al editarla, por cuestiones principalmente de tiempo y privacidad.

Algunos tests no llegan al 100 % del porcentaje de cobertura ya que utilizaban una función para recargar la página que es muy difícil de probar en las pruebas y se decidió dejar esa parte sin probar.

Se decidió que si un usuario se eliminaba, es decir, se daba de baja no se eliminasen los animales asociados a su perfil ya que posteriormente si el gerente quiere hacer un análisis de las reservas que ha hecho teniendo en cuenta las características de las mascotas para las que se había realizado necesita que estas se quedan almacenadas por lo que cuando acceda al detalle de la reserva aparecerá un mensaje de usuario dado de baja.

Además, se decidió que para los pagos extras que pudiesen surgir el día de la reserva, en lugar de tener que realizarlos por un medio como podía ser de nuevo paypal, se realicen en metálico en el propio establecimiento y lo único que interesa almacenar es el pago con su importe, la fecha y la reserva a la que va asociado. También se decidió finalmente, y tras consultar en la empresa en la que se estaban realizando las prácticas extracurriculares, utilizar la plataforma Vercel para realizar el despliegue de la aplicación en lugar de Azure como estaba planificado inicialmente.

En la Tabla 7.9 se puede ver un resumen de todas las historias de usuario realizadas, junto con sus correspondientes tareas, su estimación en puntos de historia y su correspondiente en horas, las horas realizadas finalmente y el estado de las tareas.

7.10. Sprint 9. 08/06/2024 - 17/06/2024

Este Sprint, tal y cómo estaba en la planificación, se ha dedicado para terminar de elaborar este documento, completando los apartados finales que faltasen y haciendo las correcciones necesarias. Esto también explica que la duración de este Sprint sea menor al resto, ya que tiene una duración de 9 días, teniendo en cuenta la fecha límite de entrega en convocatoria ordinaria.

También se ha terminado de probar que la aplicación funcionase correctamente en la plataforma desplegada, durante el cuál se encontró un pequeño fallo. Este consistía en que si el usuario pulsaba varias veces la tecla enter en un formulario se lanzaba ese evento varias veces, y daba lugar a que se guardase la misma mascota tres veces cuándo se añadía una nueva o la misma reserva en el caso de que se estuviese solicitando una nueva.

En la Tabla 7.10 se puede ver un resumen de todas las tareas realizadas.

7.11. Resumen de la ejecución del proyecto

En esta sección se va a realizar un análisis de la ejecución del proyecto, para ello se van a analizar las tareas que se han llevado a cabo, los costes que ha supuesto (tanto reales como simulados), y por último los riesgos que se han materializado.

7.11.1. Tareas completadas

En la Tabla 7.11, se puede ver un resumen de todas las tareas realizadas durante la ejecución del proyecto, el Sprint en el que se han realizado, la estimación de las mismas y el tiempo realmente empleado para cada una de ellas.

7.11.2. Costes finales

Para calcular cuáles han sido realmente los costes finales del proyecto, hay que tener en cuenta la diferencia de horas que se han invertido realmente de lo planificado que han sido unas 25 horas. Una vez calculados, se compararán con los presupuestos ya calculados en la Sección 2.4

Costes simulados finales

En este caso los elementos a tener en cuenta son los mismos, sólo que habría que añadirle el coste relativo al aumento de las horas planificadas inicialmente.

En la Tabla 7.12, se muestra el coste que quedaría finalmente teniendo en cuenta las horas realizadas.

Como se puede observar, el coste final simulado (7.849€) es mayor que el coste inicial simulado (7.327,95€), aunque con corrección del 20 % realizada para cubrir riesgos, se cubriría sin ningún problema.

Costes reales finales

Para el cálculo del coste real final, se tendrán en cuenta los mismos elementos. En este caso, sólo ha variado el coste relacionado con el desplazamiento hacia el centro educativo, ya que algunos eventos tuvieron lugar de forma telemática. El coste final real se puede ver en la tabla 7.13

Como se puede observar el coste real final (2.626,24€) ha sido algo menor que el coste planificado real (2.668,49€), debido al ahorro que ha supuesto no haberse desplazado todas las semanas.

7.11.3. Riesgos materializados

En este apartado, se van a listar los riesgos que se han materializado de la sección 2.2 durante la realización del proyecto junto con una breve explicación.

- RSK001 : Este riesgo se vio materializado durante el Sprint 5 ya que coincidió con un período vacacional de la alumna y se dedicaron menos horas al proyecto. Esto significó que en el siguiente Sprint tuviera que dedicarse un mayor esfuerzo, y esto implicaría un mayor número de horas.

Historia de usuario	**Tarea**	**Estimación en puntos de historia**	**Horas estimadas**	**Horas realizadas**	**Estado tarea**
Formación en distintas tecnologías	Realizar tutoriales de repaso de React	2	10	5	Finalizada
	Realizar investigación tecnologías backend	2	10	8	Finalizada
	Realizar investigación soluciones para el despliegue	2	10	6	Finalizada
Diseño	Definir épicas	0,5	2,5	0,5	Finalizada
	Definir y detallar requisitos funcionales como historias de usuario	0,5	2,5	1	Finalizada
	Definir y detallar requisitos no funcionales como historias de usuario	0,5	2,5	0,5	Finalizada
	Definir y detallar requisitos de información	0,5	2,5	0,5	Finalizada
	Realizar diseño interfaz de usuario	2	10	8	Finalizada
	Realizar modelo de dominio inicial	1,5	7,5	1,5	Finalizada
	Elaborar plan de riesgos	0,5	2,5	1,5	Finalizada
Planificación	Realizar planificación por sprints	0,5	2,5	0,5	Finalizada
	Definir qué se va a realizar en cada sprint	1,5	7,5	1,5	Finalizada
Resumen		14	70	34,5	3/3 Completadas

Tabla 7.1: Seguimiento Sprint 0

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado tarea**
Realizar modificaciones tras Sprint Review	Realizar cambios modelo dominio inicial	0,5/2,5	0,25	Finalizada
	Realizar cambios en los requisitos funcionales	0,5/2,5	0,25	Finalizada
	Realizar modificaciones en las interfaces de usuario	0,5/2,5	0,5	Finalizada
	Realizar cambios en el plan de riesgos	0,5/2,5	0,5	Finalizada
Redactar memoria realizado	Terminar apartados 2.1.4, 2.1.5	0,5/2,5	4	Finalizada
	Elaborar presupuesto	1/5	6	Finalizada
	Añadir planificación y calendarización	0,5/2,5	1	Finalizada
	Diseño arquitectura	1/5	2	En progreso
	Redactar seguimiento del proyecto Sprint 0	0,5/2,5	0	Por hacer
	Redactar seguimiento del proyecto Sprint 1	0,5/2,5	0	Por hacer
Realizar modificaciones tras la Scrum Weekly	Realizar cambios en apartado requisitos memoria	0,5/2,5	2	Finalizada
	Realizar cambios modelo dominio inicial	0,5/2,5	1	Finalizada
	Realizar cambios apartados riesgos memoria	0,5/2,5	0,4	Finalizada
Resumen		7,5/37,5	17,9	2/3 Completadas

Tabla 7.2: Seguimiento Sprint 1

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado**
	Redactar seguimiento Sprint 0	0,5/2,5	2	Completada
	Redactar seguimiento Sprint 1	0,5/2,5	2	Completada
	Diseño arquitectura	1/5	8	Completada
Realizar modificaciones tras Sprint Review	Añadir historia usuario filtrado	0,05/0,25	0,25	Completada
	Cambios modelo dominio inicial	0,05/0,25	0,25	Completada
	Cambios diagrama realizar reserva	0,05/0,25	0,25	Completada
	Nuevo diagrama día reserva	0,1/0,5	0,75	Completada
Realizar modificaciones tras Scrum Weekly	Diagrama componentes	0,25/2	2	Completada
	Diagrama despliegue	0,25/1.25	2	Completada
	Cambios apartado arquitectura	0,125/0.625	1	Completada
HU_01_01	Formación previa backend			
	Preparación entorno programación			
	Desarrollo backend			
	Desarrollo frontend			
		2/10	22,75	Completada
Resumen		4.875/26	41,25	4/4 Completadas

Tabla 7.3: Seguimiento Sprint 2

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado**
	Añadir diseño UX inicial en memoria	0,5/2,5	2	Completada
	Redactar seguimiento Sprint 2	1/5	4	Completada
	Añadir tablas tareas seguimiento Sprint 1	1/5	8	Completada
	Añadir tablas tareas seguimiento Sprint 0	0,5/2,5	2	Completada
	Añadir inicio apartado arquitectura	0,2/1	1	Completada
	Cambiar arquitectura 2 capas	0,1/0,5	0,5	Completada
	Añadir arquitectura C-S como referencia	0,3/1,5	1,5	Completada
Realizar modificaciones tras Sprint Review	Cambios front HU_01_01	0,5/2,5	2	Completada
	Cambios diagrama despliegue	0,5/2,5	1	Completada
	Cambios diagrama componentes	0,5/2,5	1	Completada
HU_01_02		2/10	5	Completada
HU_01_03		2/10	0,5	Completada
HU_01_05		3/15	2	Completada
Resumen		13,1/60,5	30,5	5/5 Completadas

Tabla 7.4: Seguimiento Sprint 3

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado**
	Redactar seguimiento Sprint 3	0,5/2,5	1,5	Completada
HU_01_04		3/15	16	Completada
HU_02_01		8/40	12	Completada
HU_02_02		2/10	3	Completada
HU_05_01		8/40	6	Completada
HU_05_02		2/10	2	Completada
Resumen		23,5/117,5	40,5	6/6 Completadas

Tabla 7.5: Seguimiento Sprint 4

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado**
	Redactar seguimiento Sprint 4	0,5/2,5	2	Completada
	Test	3/15	12	Completada
	Redacción memoria	2/10	2	En curso
HU_03_01		2/10	17	Completada
HU_03_02		5/25	3	Completada
Resumen		12,5/62,5	36	4/5 Completadas

Tabla 7.6: Seguimiento Sprint 5

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado**
	Redactar seguimiento Sprint 5	0,1/0,5	0,5	Completada
	Redactar seguimiento Sprint 6	0,1/0,5	0,5	Completada
	Redacción memoria	2/10	11,5	Completada
	Cambios tras Sprint Review	1/5	5	Completada
HU_03_03		2/10	7	Completada
HU_03_04		3/15	6	Completada
HU_04_03		2/10	13,5	Completada
Resumen		10,2/51	44	5/5 Completadas

Tabla 7.7: Seguimiento Sprint 6

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado**
	Redactar seguimiento Sprint 7	0,2/1	1,5	Completada
	Redactar capítulo 4	2,5/12,5	11	Completada
	Redacción cambios memoria	1/5	3,25	Completada
HU_04_01		5/25	24	Completada
HU_04_02		3/15	10	Completada
HU_04_05		1,5/7,5	2	Completada
Resumen		13,2/66	51,75	4/4 Completadas

Tabla 7.8: Seguimiento Sprint 7

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado**
	Redactar seguimiento Sprint 8	0,2/1	1,5	Completada
	HU_05_03	1/5	1,5	Completada
	HU_05_05	0,5/2,5	0,5	Completada
	HU_05_04	2/10	1	Completada
	HU_05_06	1/5	1,5	Completada
	HU_05_07	1/5	1	Completada
	HU_04_04	0,5/2,5	1	Completada
	HU_03_05	1/5	1	Completada
	HU_03_06	1/5	2,25	Completada
	HU_02_03	1/5	1,5	Completada
	HU_02_05	0,5/2,5	0,5	Completada
	HU_02_04	1/5	3,5	Completada
	Fixes	1/5	3,5	Completada
	Tests	4/20	20,5	Completada
	Cambios tras Scrum Weekly	2/10	10	Completada
	Despliegue Vercel	3/15	10	Completada
Resumen	Resumen	20,7/103,5	60,75	16/16 Completadas

Tabla 7.9: Seguimiento Sprint 8

Historia de usuario	**Tarea**	**Estimación / Horas**	**Horas realizadas**	**Estado tarea**
	Bug tecla enter	0,5/2,5	2	Completada
	Redacción memoria	9/45	48	Completada
Resumen		9,5/47,5	50	2/2 Completadas

Tabla 7.10: Seguimiento Sprint 9

Tarea	Sprint	Estimación	Horas
Formación React	0	2	5
Formación backend	0	2	8
Investigación despliegue	0	0,5	0,5
Definir y detallar RF como HU	0	0,5	1
Definir y detallar RNF como HU	0	0,5	0,5
Definir y detallar RI	0	0,5	0,5
Diseño UX	0	2	8
Realizar modelo dominio inicial	0	1,5	1,5
Elaborar plan riesgos	0	0,5	1,5
Realizar planificación	0	2	2
Realizar cambios tras Sprint Review	1, 2, 3	3,75	4
Redacción memoria	1, 2, 3, 4, 5, 6, 7, 8, 9	20	107,25
Realizar cambios tras Scrum Weekly	1, 8	4,125	18,4
Diseño arquitectura	2	1	8
HU_01_01	2	2	22,75
HU_01_02	3	2	5
HU_01_03	3	2	0,5
HU_01_05	3	3	2
HU_01_04	4	3	16
HU_02_01	4	8	12
HU_02_02	4	2	3
HU_05_01	4	8	6
HU_05_02	4	2	2
Tests	5, 8	7	32,5
HU_03_01	5	2	17
HU_03_02	5	5	3
HU_03_03	6	2	7
HU_03_04	6	3	6
HU_04_03	6	3	13,5
HU_04_01	7	5	24
HU_04_02	7	3	10
HU_04_05	7	1,5	2
HU_05_03	8	1	1,5
HU_05_05	8	0,5	0,5
HU_05_04	8	2	1
HU_05_06	8	1	1,5
HU_05_07	8	1	1
HU_04_04	8	0,5	1
HU_03_05	8	1	1
HU_03_06	8	1	2,25
HU_02_03	8	1	1,5
HU_02_05	8	0,5	0,5
HU_02_04	8	1	3,5
Fixes	8	1	3,5
Despliegue	8	3	10
Total		118,875	349,15

Tabla 7.11: Tareas completadas durante la ejecución

	Precio	**Cantidad**	**Total**
Sueldo	14,09 € /hora	350	4.931,5€
Oficina	100€/mes	5 meses	500€
Equipo empleado	265,39€/mes	5 meses	1.326,95€
Licencias,servicios	218,11€/mes	5 meses	1.090,55€
Total			7.849€

Tabla 7.12: Coste final simulado

	Precio	**Cantidad**	**Total**
Equipo empleado	459,8€ /mes	5 meses	2298,99€
Luz	8,1€ /mes	5 meses	40,5€
Internet	20€ /mes	5 meses	100€
Gas	30€ /mes	2 meses	60€
Desplazamiento	8,45€ /viaje	15 viajes	126,75€
Total			2.626,24€

Tabla 7.13: Coste final real

Capítulo 8

Conclusiones y líneas futuras

8.1. Conclusiones

Una vez que se ha finalizado el proyecto, se puede concluir que se ha obtenido un resultado exitoso ya que con el esfuerzo invertido en horas de trabajo se ha conseguido llegar a un producto mínimo viable que cumpliría con las expectativas de la idea de negocio planteada.

Se han cumplido tanto los objetivos tanto los propios del proyecto como personales. En cuánto a los personales, se ha conseguido lograr finalizar todo el proyecto, incluso con las dificultades y/o adversidades encontradas, dentro de los márgenes de espacio establecidos y viendo como el esfuerzo y el empeño empleados en el proyecto han dado su fruto, con la finalización exitosa del mismo. También destacar que se ha visto cómo se han puesto a prueba los conocimientos adquiridos en el grado.

En cuanto a los objetivos principales indicados en la sección 1.3, se puede decir que se han cumplido los siguientes:

- Permitir a los clientes gestionar sus mascotas.
- Permitir a los clientes solicitar una reserva para cada una de sus mascotas.
- Permitir al gerente gestionar las reservas solicitadas por el cliente.
- Permitir al gerente gestionar la información relativa al establecimiento.

En cuanto a los objetivos personales se han logrado los siguientes:

- Aprender a utilizar una tecnología nueva para el backend y base de datos, en este caso Node JS con Express y MongoDB, e integrarlo en Angular
- Aprender a realizar tests unitarios con Jasmine y Karma.

- Aprender a realizar el despliegue de la aplicación, dentro de las limitaciones que permitía la plataforma

8.2. Líneas de trabajo futuras

Como trabajo a futuro, se plantean las siguientes cuestiones, que quizás si se tratase de un proyecto a más largo plazo se hubiesen podido abordar:

- **Integrar funcionalidad con fotos** : hacer las mejoras y modificaciones necesarias en la base de datos y la implementación en la aplicación para abordar la funcionalidad relacionada con las fotos que no se ha logrado.
- **Traducciones** : implementar el servicio de traducciones con sus correspondiente archivos para que la aplicación este disponible tanto en inglés cómo en español.
- **Responsive para dispositivos móviles** : realizar las modificaciones necesarias para que la aplicación también sea responsive para dispositivos móviles.
- **Escalabilidad** : realizar las modificaciones necesarias para que la aplicación sea capaz de gestionar más de un establecimiento cuando el negocio crezca.
- **Notificaciones**: implementar la funcionalidad para recibir notificaciones push según cambie el estado de la reserva y notificaciones por correo, que no se han podido abordar.

Bibliografía

[1] Microsoft 365. Office 365 E1 EEA . https://www.microsoft.com/es-es/microsoft-365/enterprise/office365-plans-and-pricing-eea. Accessed: 2024-2-26.

[2] Angular. Architecture components. Angular Docs. https://docs.angular.lat/guide/architecture-components. Accessed: 2024-05-12.

[3] Support Apple. Consumo energético y potencia térmica de los iMac. https://support.apple.com/es-es/109513. Accessed: 2024-2-29.

[4] Arsys. ¿Qué es NPM? https://www.arsys.es/blog/que-es-npm-javascript-para-principiantes. Accessed: 2024-05-13.

[5] Asana. ¿Qué son los puntos de historia? Seis pasos sencillos para estimar el trabajo en Agile. https://asana.com/es/resources/story-points. Accessed: 2024-2-29.

[6] Astah. Astah Professional. https://astah.net/products/astah-professional/. Accessed: 2024-05-12.

[7] Atlassian. Jira Atlassian. https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software. Accessed: 2024-05-12.

[8] Atlassian. What is Git. https://www.atlassian.com/es/git/tutorials/what-is-git. Accessed: 2024-05-13.

[9] Atlassian. What is Git. https://www.atlassian.com/es/git/tutorials/what-is-git. Accessed: 2024-05-13.

[10] Ayoubkhial. Build a MEAN web app: Part 1 - The architecture. https://www.ayoubkhial.com/blog/mean-web-app-part-1-the-architecture. Accessed: 2024-3-02.

[11] Carrefour. Acer Aspire 3 A315-56, i5 1035G1, 12GB, 512GB SSD. https://www.carrefour.es/ordenador-portatil-acer-aspire-3-a315-56-i5-1035g1-12gb-512gb-ssd-156-3962-cm-w11-con-antivirus-panda-pro/VC4A-24740115/p?awc=25399_1709197783_7e749b08e7d4b7397ddca5a08601ff22&dclid=CPjiifaZ0IQDFatOQQIdcqMEEw. Accessed: 2024-2-29.

[12] Verónica Durán Castello. Artículo Prensa El Confidencial. https://www.elconfidencial.com/alma-corazon-vida/2024-03-23/datos-mundo-mascotas-espana-pet-parent_3853414/#:~:text=Un%20dato%20clave%20que%20explica,los%20hogares%20tienen%20una%20mascota. Accessed: 2024-06-08.

[13] CLAIRE DRUMOND. What is scrum and how to get started. https://www.atlassian.com/agile/scrum. Accessed: 2024-2-26.

[14] Zsolt Csik. Applying a MVVM-like pattern in Angular. https://zsolt-csik.medium.com/applying-a-mvvm-like-pattern-in-angular-part-1-ab8d93dc3a88. Accessed: 2024-5-12.

[15] Ali Darvish. Client-Server Architecture. https://darvishdarab.github.io/cs421_f20/docs/readings/client_server/. Accessed: 2024-5-12.

[16] Angular Docs. What is Angular? https://angular.io/guide/what-is-angular. Accessed: 2024-05-13.

[17] Jasmine Docs. Jasmine. https://jasmine.github.io/. Accessed: 2024-05-13.

[18] freeCodeCamp. SOLID Definition – the SOLID Principles of Object-Oriented Design Explained. https://www.freecodecamp.org/news/solid-principles-single-responsibility-principle-explained/. Accessed: 2024-3-03.

[19] Glassdoor. Sueldos de Junior Developer - Full Stack en España. https://www.glassdoor.es/Sueldos/junior-full-stack-developer-sueldo-SRCH_KO0,27.htm. Accessed: 2024-2-26.

[20] Andrew Hanks. MEAN Stack Architecture: AngularJS, NodeJS, ExpressJS and MongoDB. https://medium.com/@andrewhanks2402/mean-stack-architecture-angularjs-nodejs-expressjs-and-mongodb-adedabbf8482. Accessed: 2024-3-02.

[21] IBM. ¿Qué es MongoDB? https://www.ibm.com/es-es/topics/mongodb. Accessed: 2024-05-13.

[22] El Corte Inglés. iMac Chip M1 2021 8GB RAM y 256GB SSD. https://www.elcorteingles.es/electronica/A39591357-apple-imac-24-2021-m1-8gb-512-gb-ssd-retina-45k-24-macos-plata/?gad_source=1&gclid=EAIaIQobChMIhO_AyJjQhAMVXJSDBx18bg-EEAQYBCABEgIy5fD_BwE&gclsrc=aw.ds. Accessed: 2024-2-29.

[23] IONOS. Patron singleton: una clase propia. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/patron-singleton/. Accessed: 2024-3-03.

[24] Ittgweb. 3.5 Diseño de Software de Arquitectura Arquitectura Cliente-Servidor. https://ittgweb.wordpress.com/2016/05/28/3-5-diseno-de-software-de-arquitectura-arquitectura-cliente-servidor/. Accessed: 2024-3-29.

[25] Keepcoding. ¿Qué es el patrón Observer y cómo se usa? https://keepcoding.io/blog/patron-observer-y-como-se-usa/. Accessed: 2024-3-03.

[26] Ken Schwaber and Jeff Sutherland. The 2020 scrum guidetm. https://scrumguides.org/scrum-guide.html. Accessed: 2024-2-26.

[27] Microsoft. Getting Started - Visual Studio Code. https://code.visualstudio.com/docs. Accessed: 2024-05-13.

[28] MongoDB. MongoDB Atlas Tutorial. https://www.mongodb.com/resources/products/platform/mongodb-atlas-tutorial. Accessed: 2024-06-09.

[29] MongoDB. What is the MEAN stack? https://www.mongodb.com/mean-stack. Accessed: 2024-3-02.

[30] NPM. Ngx-Paypal. https://www.npmjs.com/package/ngx-paypal. Accessed: 2024-06-09.

[31] NPM. Node Cron. https://www.npmjs.com/package/node-cron. Accessed: 2024-06-09.

[32] OpenWebinars. ¿Qué patrón usa Angular? MVC o MVVM. https://openwebinars.net/blog/que-patron-usa-angular-mvc-o-mvvm/. Accessed: 2024-3-03.

[33] Paypal. PayPal Developer. https://developer.paypal.com/. Accessed: 2024-06-10.

[34] Paypal. Paypal Developer Dashboard. https://developer.paypal.com/tools/sandbox/. Accessed: 2024-06-09.

[35] PCComponentes. Dell Latitude 3440 Intel Core i5-1335U/16GB/512GB SSD. https://www.pccomponentes.com/dell-latitude-3440-intel-core-i5-1335u-16-gb-512gb-ssd-14?campaigntype=eshopping&campaignchannel=shopping&gad_source=1. Accessed: 2024-2-26.

[36] PCComponentes. P2422H 23.8"LED IPS FullHD. https://www.pccomponentes.com/dell-p2422h-238-led-ips-fullhd. Accessed: 2024-2-26.

[37] Astah Professional. Astah Professional Subscription License. https://cv.astah.net/hubfs/Price%20List/20240401/Astah_Updated_PriceList_20240401.pdf. Accessed: 2024-2-26.

[38] Autocares La Regional. Precio billete ida y vuelta. https://booking.autocareslaregional.com/. Accessed: 2024-2-29.

[39] Repsol. ¿Cuál es el consumo de un ordenador? https://www.repsol.es/particulares/asesoramiento-consumo/cuanto-consume-ordenador/#:~:text=Lo%20que%20gasta%20un%20PC,de%2030%20a%20140%20W. Accessed: 2024-2-29.

[40] Aplyca Tecnología SAS. Vercel: Desarrollar, Previsualizar, Enviar. https://www.aplyca.com/blog/blog-que-es-vercel-desarrollar-previsualizar-enviar. Accessed: 2024-06-09.

[41] Scrum.org. What is Scrum ? https://www.scrum.org/learning-series/what-is-scrum/. Accessed: 2024-2-26.

[42] Selectra. Precio del gas natural: Evolución en España y coste del m3. https://tarifaluzhora.es/info/gasto-medio-gas. Accessed: 2024-2-29.

[43] Selectra. Tarifa Luz Hora. https://tarifaluzhora.es/info/precio-kwh. Accessed: 2024-2-29.

[44] Universidad de Valladolid. Proyecto docente del trabajo de fin de grado 2023-2024 (Mención Ingeniería de Software). https://apps.stic.uva.es/guias_docentes/uploads/2023/545/46976/1/Documento.pdf. Accessed: 2024-2-18.

[45] Wikipedi. Overleaf. https://es.wikipedia.org/wiki/Overleaf. Accessed: 2024-05-12.

[46] Wikipedia. Bootstrap. https://es.wikipedia.org/wiki/Bootstrap_(framework). Accessed: 2024-05-13.

[47] Wikipedia. Figma. https://es.wikipedia.org/wiki/Figma. Accessed: 2024-05-12.

[48] Wikipedia. GitHub. https://en.wikipedia.org/wiki/GitHub. Accessed: 2024-06-15.

[49] Wikipedia. Google Calendar. https://es.wikipedia.org/wiki/Google_Calendar. Accessed: 2024-05-12.

[50] Wikipedia. Google Fonts. https://es.wikipedia.org/wiki/Google_Fonts. Accessed: 2024-05-13.

[51] Wikipedia. Microsoft Copilot. https://es.wikipedia.org/wiki/Microsoft_Copilot. Accessed: 2024-05-13.

[52] Wikipedia. Microsoft Teams. https://es.wikipedia.org/wiki/Microsoft_Teams. Accessed: 2024-05-13.

[53] Wikipedia. Node.js. https://es.wikipedia.org/wiki/Node.js. Accessed: 2024-06-17.

[54] Wikipedia. SOLID. https://en.wikipedia.org/wiki/SOLID. Accessed: 2024-3-03.

[55] Friedel Ziegelmayer. Karma. https://karma-runner.github.io/6.4/intro/how-it-works.html. Accessed: 2024-05-13.

Apéndice A

Manuales

A.1. Manual de despliegue

Para poder realizar el despliegue correctamente se necesitan los siguientes requisitos:

- MongoDB Compass, para ver como se almacenan los datos y hacer unas comprobaciones y configuraciones previas.

Lo primero que se debe hacer es comprobar si la base de datos está configurada correctamente, para ello en la herramienta MongoDB Compass, indicar en el cuadro la siguiente url de conexión:

```
mongodb+srv://tfg-vercel:a311vPpdV0yHBfXh@tfg-vercel.hs7gbjr.mongodb.net/
?retryWrites=true&w=majority&appName=tfg-vercel
```

Después, si no hay ningún establecimiento almacenado, se debe añadir el siguiente json:

```
    [{
  "_id": {
    "$oid": "65f8618a623c78b41b74c280"
  },
  "id": 1,
  "__v": 0,
  "address": "Calle DireccionPrueba1",
  "email": "contacto@example.com",
  "name": "NuevoEstablecimientoId1",
  "phone": "12456789"
}]
```

Si no hay ninguna tarifa, añadir el siguiente json:

```
    [{
  "_id": {
    "$oid": "6600532f92c61db96ec59b2c"
  },
  "establecimiento_id": 1,
  "precio_alimentacion": 2,
  "precio_suelo_frio": 5,
  "precio_suelo_calido": 5,
  "precio_bano": 10,
  "precio_peluqueria": 15,
  "precio_corta_unas": 3,
  "__v": 0,
  "precio_full_time": 50,
  "precio_half_time": 25,
  "precio_hora": 8
}]
```

Si no hay ningún usuario con el rol de admin, añadir el siguiente json, cambiando el valor de oid del campo roles, por el que aparezca dentro de la colección roles en admin:

```
    {
  "_id": {
    "$oid": "66278e13ff0fe0e82a0da1b4"
  },
  "email": "admin@example.com",
  "password": "$2a$12$X.xnOg4Iz0KsPx58PYuXZe5QYdk0UcQMgtxYqYob5pFaCU7t4QF1G",
  "roles": [
    {
      "$oid": "660876d4aa215a91a085ac03"
    }
  ],
  "id": 1,
  "__v": 0
}
```

Una vez realizadas esta comprobaciones previas para que funcione correctamente, simplemente acceder a la siguiente url donde se encuentra desplegada la aplicación:

```
https://tfg-amanda-morera-perez-deployment-frontend.vercel.app/
```

A.2. Manual de despliegue en local

Para poder realizar el despliegue en local correctamente se necesitan los siguientes requisitos:

- Disponer de un IDE, en este caso se recomienda Visual Studio Code, aunque no es totalmente necesario pero será más fácil realizar el despliegue desde ahí.
- Node.js, versión 16.14.0.
- npm, versión 8.3.1.
- Angular CLI 16.2.6.
- MongoDB Compass, esta herramienta tampoco es totalmente necesaria pero si se quieren visualizar los datos que se registran en la base de datos es muy útil.
- Git, para descargar el repositorio.

Una vez instalados todos los requisitos, desde la propia terminal del IDE, se debe descargar el repositorio para ello utilizar la siguiente instrucción:

```
git clone https://gitlab.inf.uva.es/amamore/tfg_morera_perez_amanda.git
```

A continuación, se debe situar en el directorio raíz del proyecto y ejecutar también en la propia terminal que proporciona el IDE, la siguiente instrucción:

```
npm i
```

Después situarnos en el directorio de la parte del cliente con la siguiente instrucción:

```
cd client-app
```

y ejecutar de nuevo la instrucción

```
npm i
```

Volver a repetir esto mismo pero situándonos en el directorio de la parte del servidor con la siguiente instrucción:

```
cd ../server
```

y ejecutar de nuevo la instrucción:

```
npm i
```

Una vez realizado todo esto, volverse a situar en el directorio raíz del proyecto y ejecutar la siguiente instrucción:

```
npm run start
```

Una vez realizado esto, el proyecto ya estará desplegado localmente y para acceder a él desde el navegador se debe hacer desde la siguiente url `http://localhost:4200`.

Por último, si se quiere tener acceso o visualizar los datos que se registran en la base de datos de MongoDB, se debe abrir MongoDB Compass y en el cuadro dónde pide a que url se quiere hacer la conexión, indicar la siguiente `mongodb://localhost:27017` y hacer click en Connect.

A.3. Manual de usuario

La primera vez que se accede a la aplicación se hace como usuario no registrado, por ello se tendrá acceso a la pantalla de inicio pero sólo se podrá ver la información sobre el establecimiento y el negocio. Si se desea realizar una reserva o acceder a cualquier otra funcionalidad se tendrá que iniciar sesión en la aplicación. En la figura A.1 se muestra la pantalla de inicio sin haber iniciado sesión.

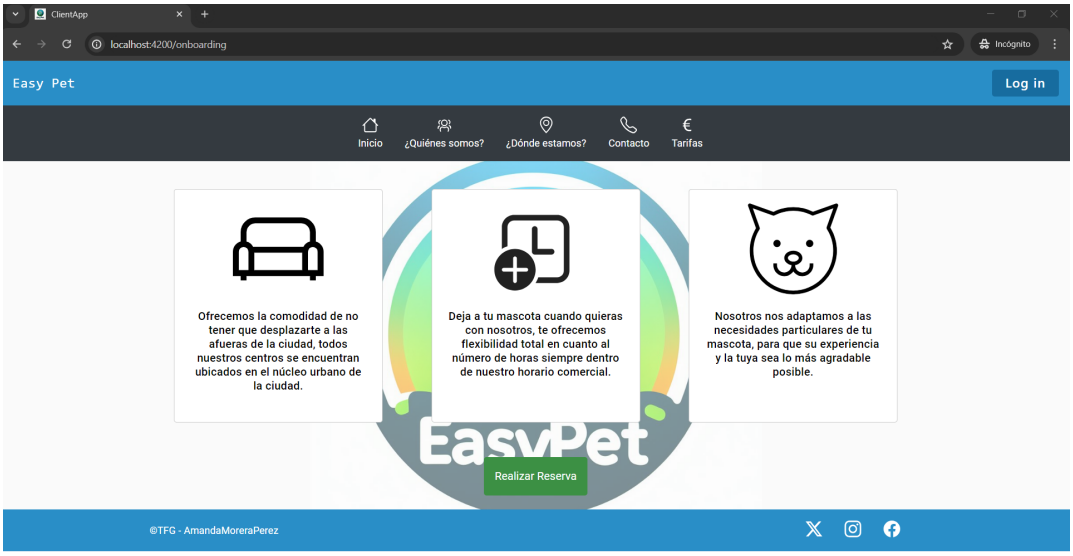


Figura A.1: Pantalla de inicio de la aplicación

A.3.1. Usuario

En la pantalla de login, si se dispone de un usuario registrado, se introducen las credenciales(ver Figura A.2) y si son correctas será redirigido a la pantalla de inicio. En caso de que no se disponga de un usuario registrado, se podrá registrar un usuario rellenando el formulario y será redirigido de nuevo al login dónde podrá iniciar sesión con el usuario que acaba de registrar. En la figura A.3 se puede ver la pantalla de registro de un usuario.

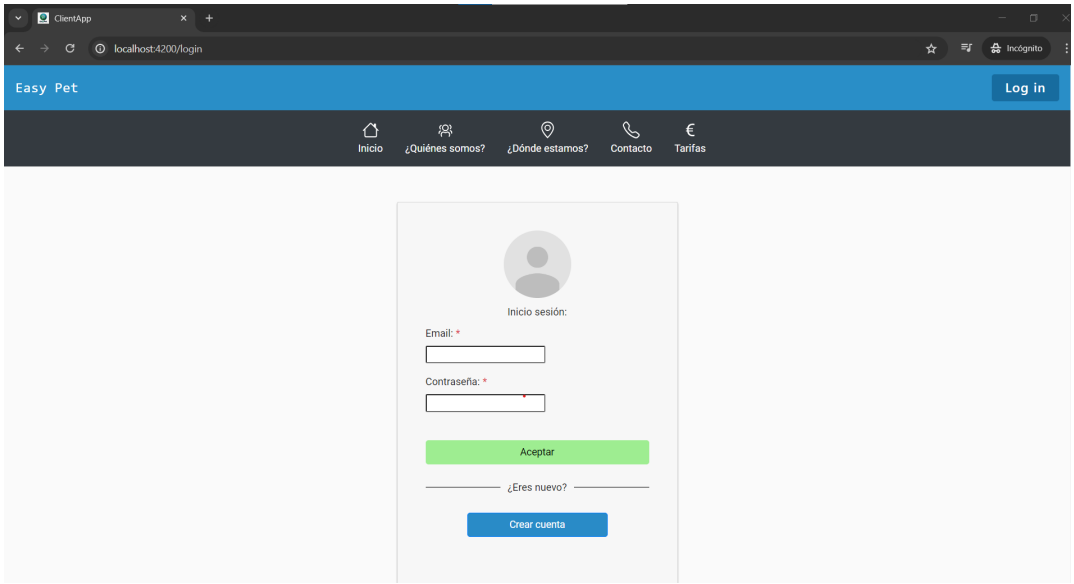


Figura A.2: Pantalla de inicio de sesión

Una vez se ha iniciado sesión con el rol de usuario correctamente, en la pantalla de inicio se puede hacer click en el menú desplegable y navegar hacia el perfil del usuario dónde se mostrará los datos del usuario, hacia las reservas dónde se mostrará el listado de todas las reservas asociadas a dicho usuario o hacia las mascotas dónde se mostrará el listado de todas las mascotas asociadas a dicho usuario. En la figura A.4, se puede ver la pantalla con el menú desplegable.

También desde la pantalla de inicio, se puede hacer click en el botón de Realizar Reserva y será redirigido al formulario, donde se rellenará el formulario y al hacer click en finalizar registro se guardará la solicitud de la reserva que quedará pendiente de que la revise el gerente y será redirigido al listado de las reservas asociadas al perfil, o por si el contrario se hace click al botón de salir sin guardar, no se almacena nada y será redirigido al listado de reservas. En la figura A.5 y A.6, se puede ver la pantalla del formulario de solicitud de la reserva.

Dentro del detalle del perfil(ver Figura A.7, se puede hacer click en el botón de editar donde será redirigido a una pantalla con un formulario con los campos que se pueden editar dónde aparecerá por defecto los datos que ya están almacenados, en las figuras A.8 y A.9 se puede ver la pantalla de editar un usuario. Una vez realizados los cambios que se consideren, si se hace click en el botón de guardar cambios, se almacenarán dichos cambios y será redirigido

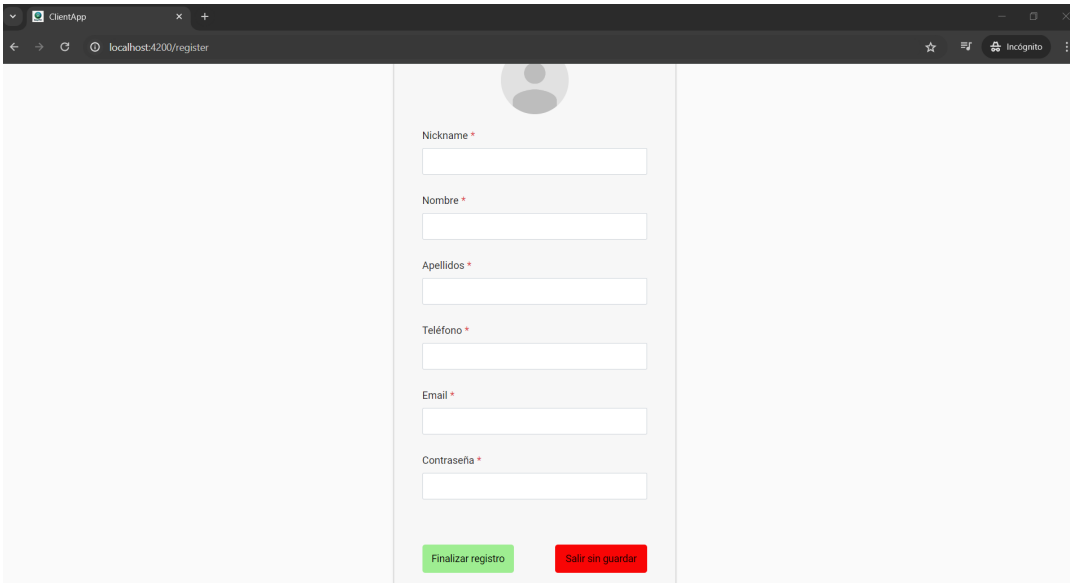


Figura A.3: Pantalla de registro de un usuario

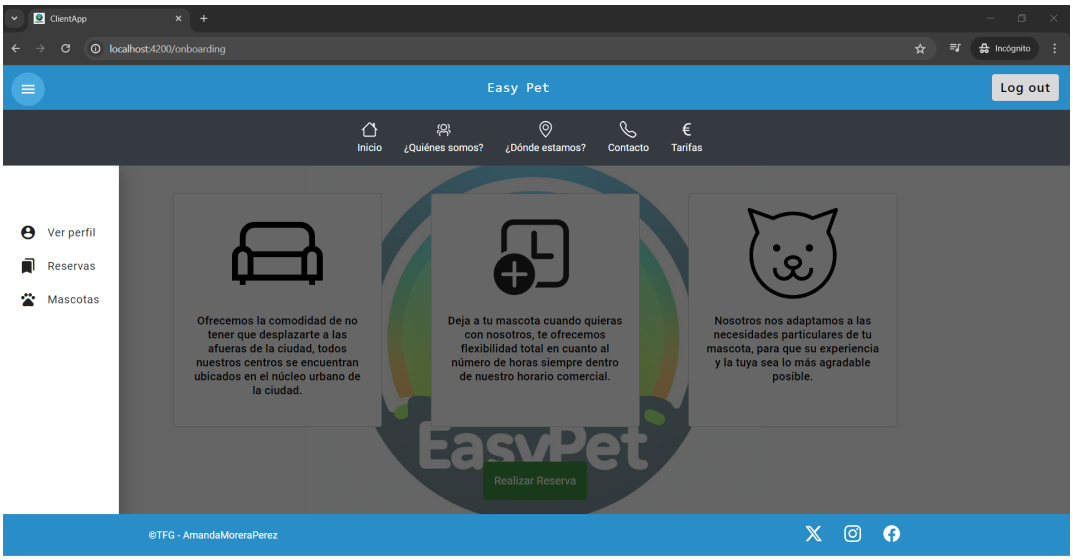


Figura A.4: Pantalla de menú desplegable de la aplicación

al detalle del perfil, o si en caso contrario se hace click en el botón de cancelar será redirigido al detalle del perfil. Si se hace click en el botón de eliminar, se eliminará dicho usuario y será redirigido a la pantalla de inicio. Si se hace click en el botón de mascotas, será redirigido al listado de mascotas asociadas a dicho usuario. Si se hace click en el botón de reservas, será redirigido al listado de las reservas asociadas a dicho usuario.

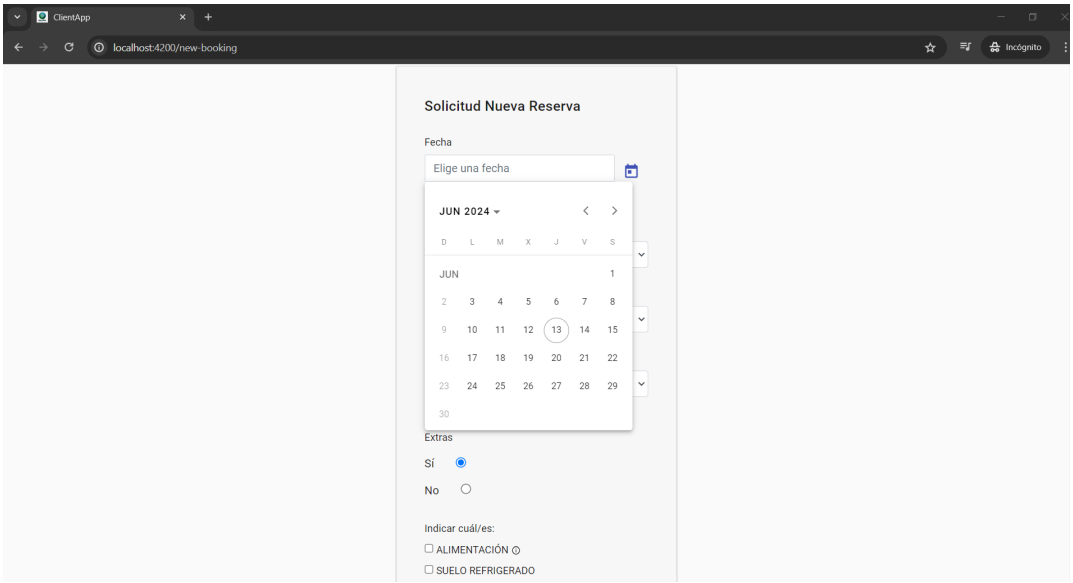


Figura A.5: Pantalla de solicitud de una reserva(1)

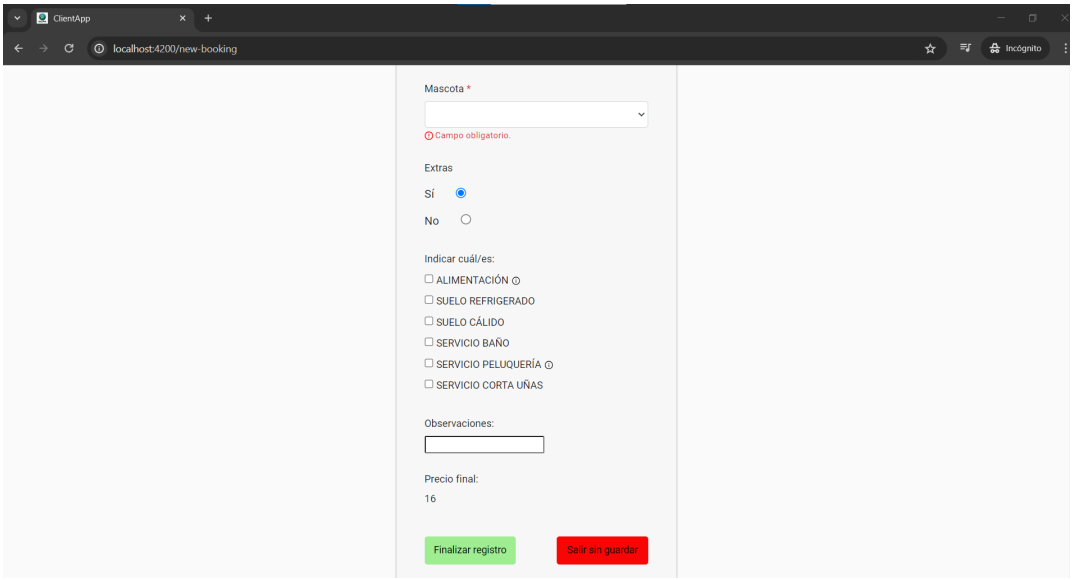


Figura A.6: Pantalla de solicitud de una reserva(2)

Dentro del listado de las reservas, ver Figura A.10, si se hace click en el botón con un + . se accederá al formulario para la solicitud de una reserva ya detallado anteriormente. También seleccionando en el filtro un estado sólo mostrará aquellas que coincidan con dicho estado, en la figura A.11, se muestra la pantalla con el funcionamiento del filtro. Si se hace click en una reserva, será redirigido al detalle de la reserva. Si se encuentra en estado pendiente (ver

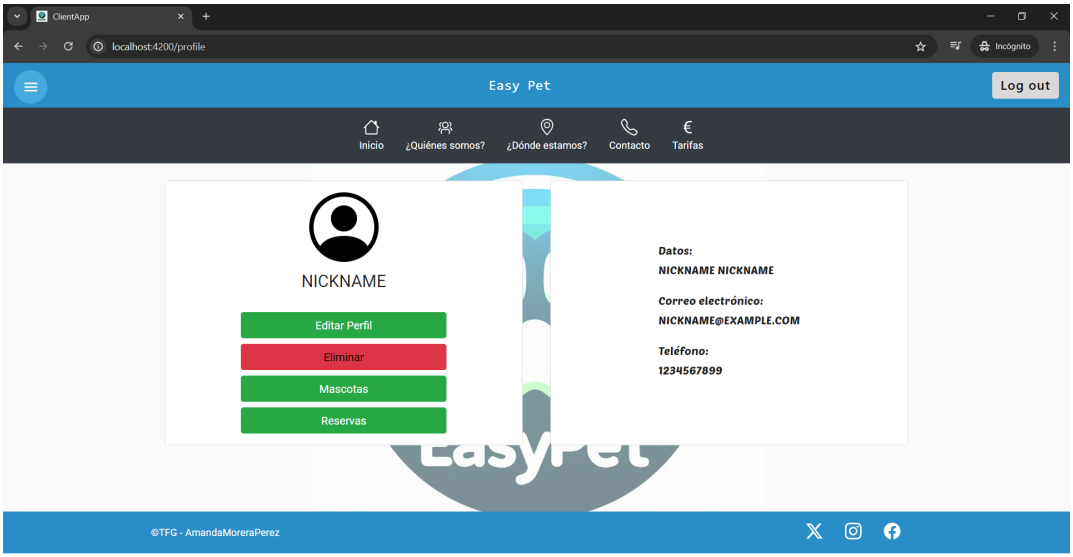


Figura A.7: Pantalla del perfil del usuario

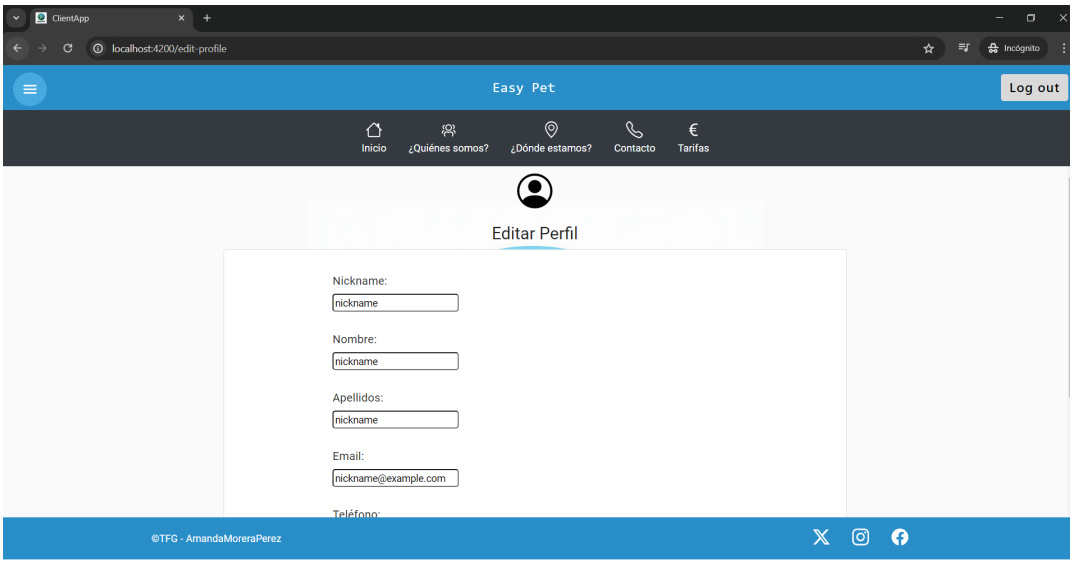


Figura A.8: Pantalla de editar usuario(1)

Figura A.12) a que la acepte el gerente, sólo aparecerá el botón de cancelar reserva que si se hace click en él, la reserva pasará al estado de cancelada y se recargará la pantalla para reflejar el cambio de estado. Si se encuentra aceptada por el gerente y pendiente de pago, (ver Figura A.13), aparecerá el botón de realizar pago por paypal y el de cancelar reserva. Si se hace click en el botón de realizar pago, será redirigido al proceso de pago y si se realiza correctamente, se almacenará el pago y se recargará la pantalla para reflejar el cambio de

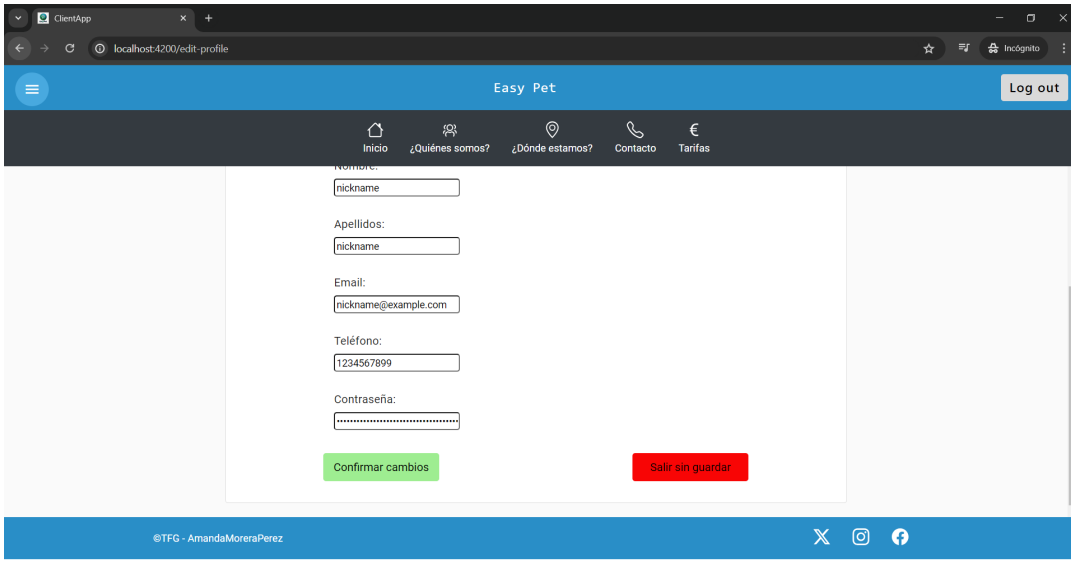


Figura A.9: Pantalla de editar usuario(2)

estado. En el caso de que ocurrá algún error al realizar el pago, no se realiza ningún cambio de estado ni se almacena ningún pago y se queda en la pantalla del detalle de la reserva. Si se encuentra en el estado de confirmada (ver Figura A.14), sólo aparecerá el botón de cancelar, en el caso de que se haga click en él seguirá el flujo descrito anteriormente. Si se encuentra en el estado de cancelada (ver Figura A.15), no mostrará ningún botón adicional.

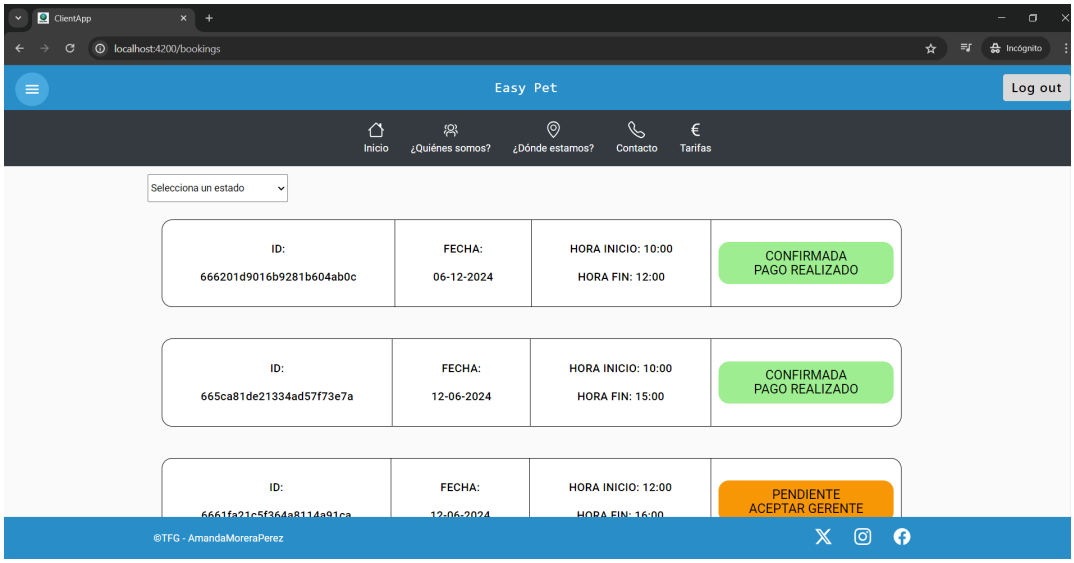


Figura A.10: Pantalla de listado de reservas

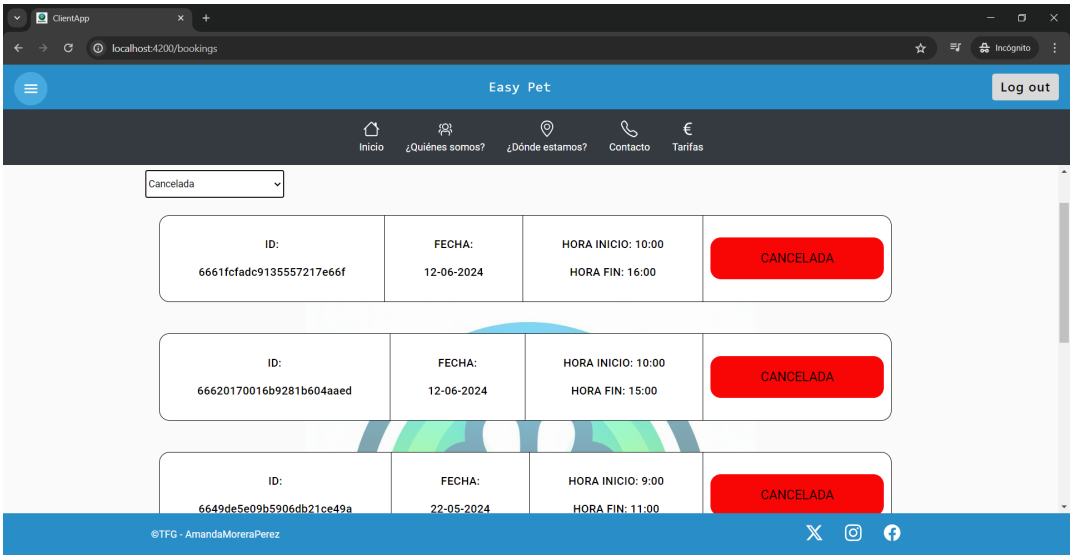


Figura A.11: Pantalla de listado de reservas con uso del filtro

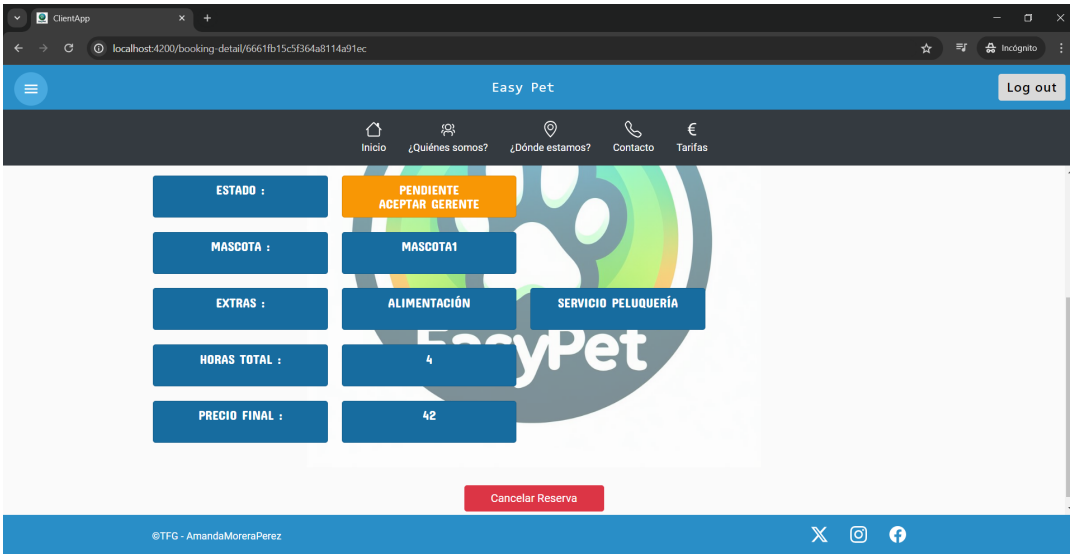


Figura A.12: Pantalla de detalle reserva pendiente

Dentro del listado de mascotas (ver Figura A.16), si se hace click en el botón de ver será redirigido a la pantalla del detalle de la mascota (ver Figura A.17). Dentro de esta, si se hace click en el botón de editar, será redirigido a la pantalla con el formulario para editar la mascota (ver Figura A.18), dónde sólo se podrán editar ciertos campos, los cuáles aparecen ya con lo almacenado hasta el momento. Cuando se realicen los cambios que se consideren, si se hace click en el botón de guardar cambios, se almacenarán dichos cambios

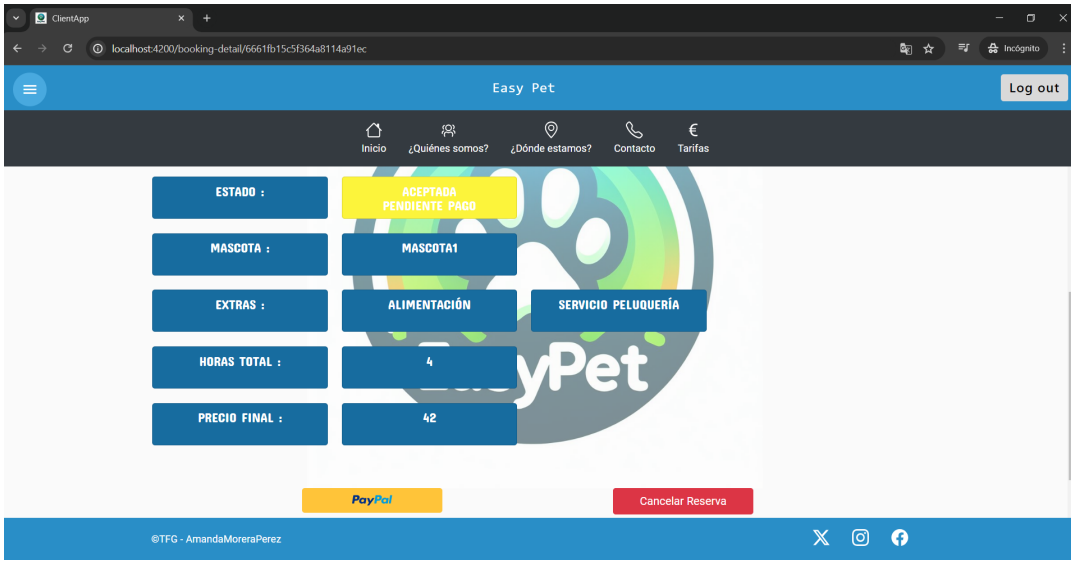


Figura A.13: Pantalla de detalle reserva aceptada pendiente pago

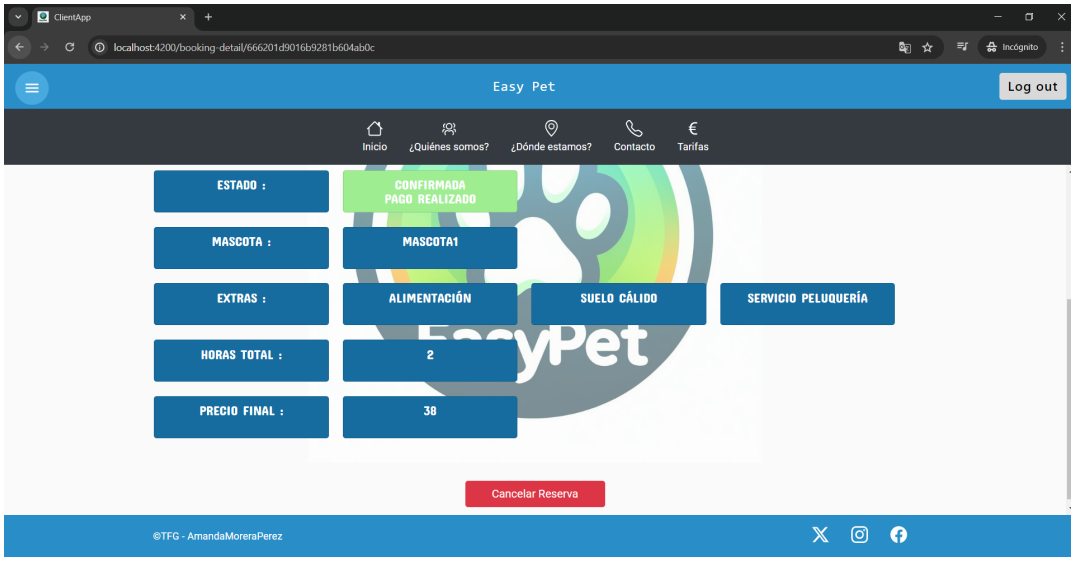


Figura A.14: Pantalla de detalle reserva confirmada

y será redirigido a la pantalla del detalle de la mascota. Si dentro de la pantalla de editar mascota, se hace click en el botón de añadir dueño (ver Figura A.19), si el correo que se introduce corresponde a un usuario existente y que no esta ya asociado a dicha mascota, se asociará a esta y será redirigido al detalle de la mascota. Si se hace click en eliminar mascota, se eliminará la mascota y será redirigido al listado de mascotas.

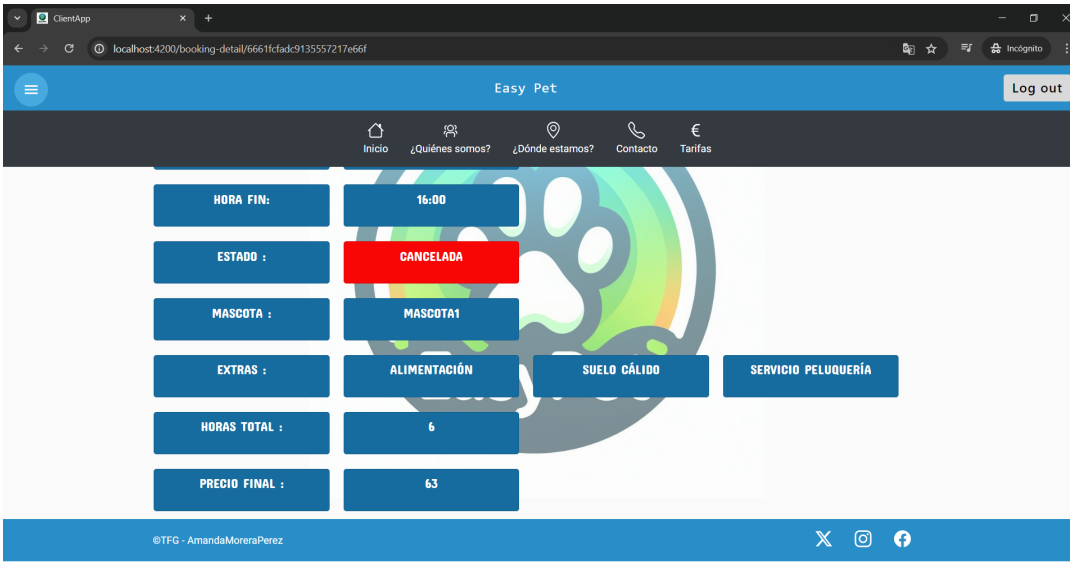


Figura A.15: Pantalla de detalle reserva cancelada

Por último, si se hace click en el botón de logout, se cerrará la sesión del usuario actual y será redirigido a la pantalla de inicio.

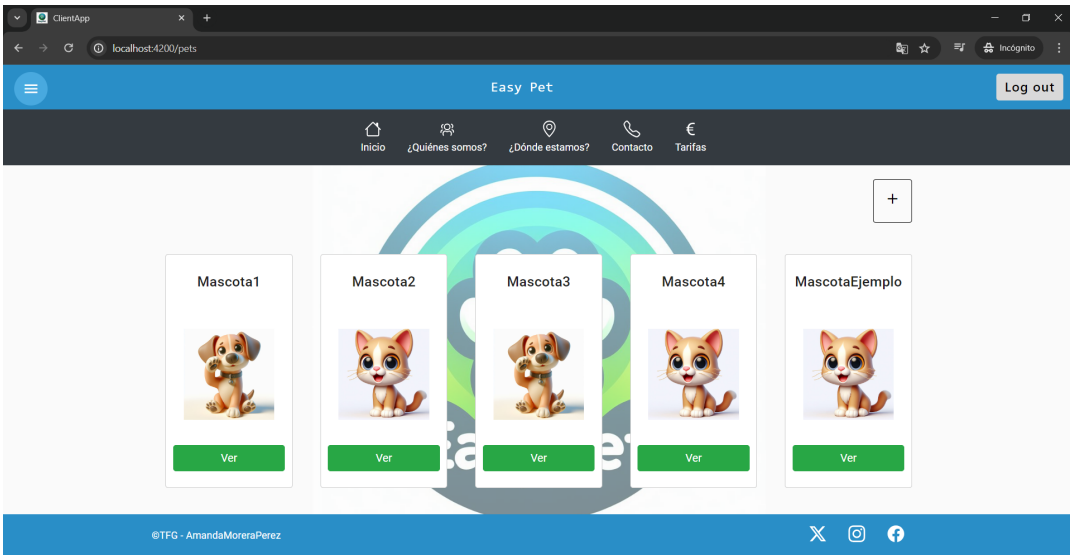


Figura A.16: Pantalla del listado de mascotas

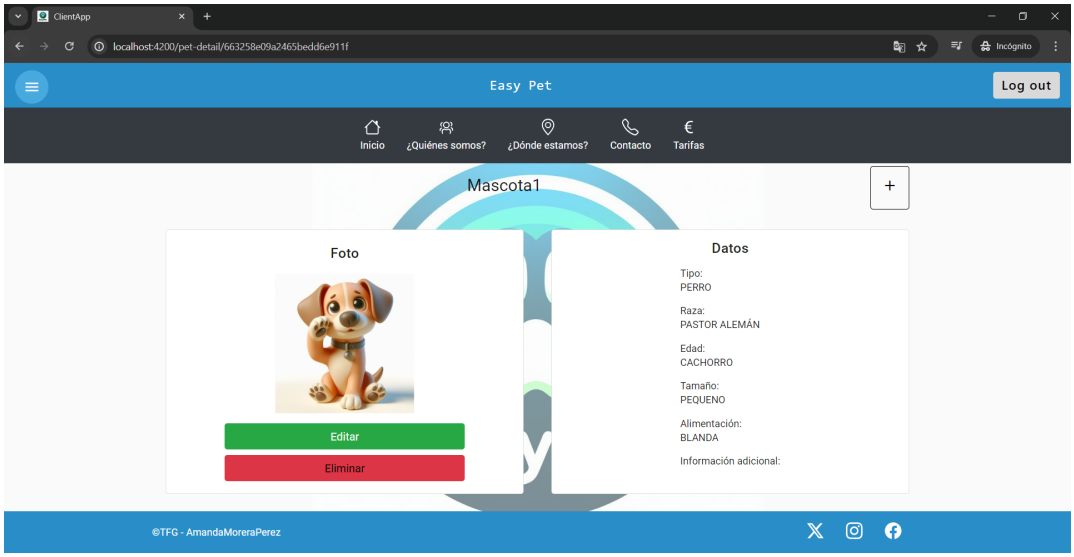


Figura A.17: Pantalla del detalle de la mascota

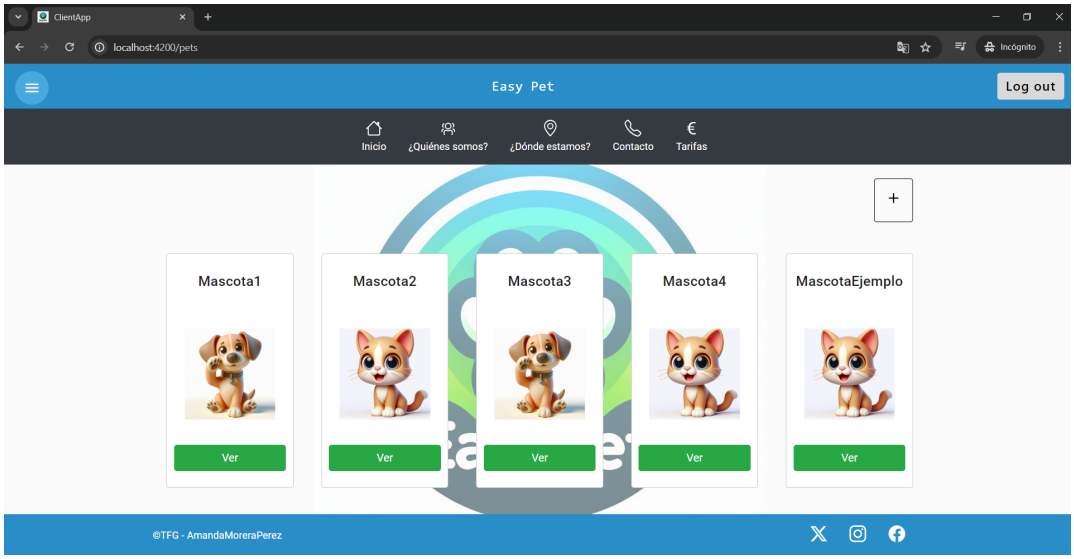


Figura A.18: Pantalla del formulario para editar mascota

A.3.2. Gerente

Cuando se inicia sesión con las credenciales con el rol de admin, será redirigido a su pantalla de inicio dónde se mostrará un mensaje de saludo, en la Figura A.20 se puede ver la pantalla de inicio del gerente.

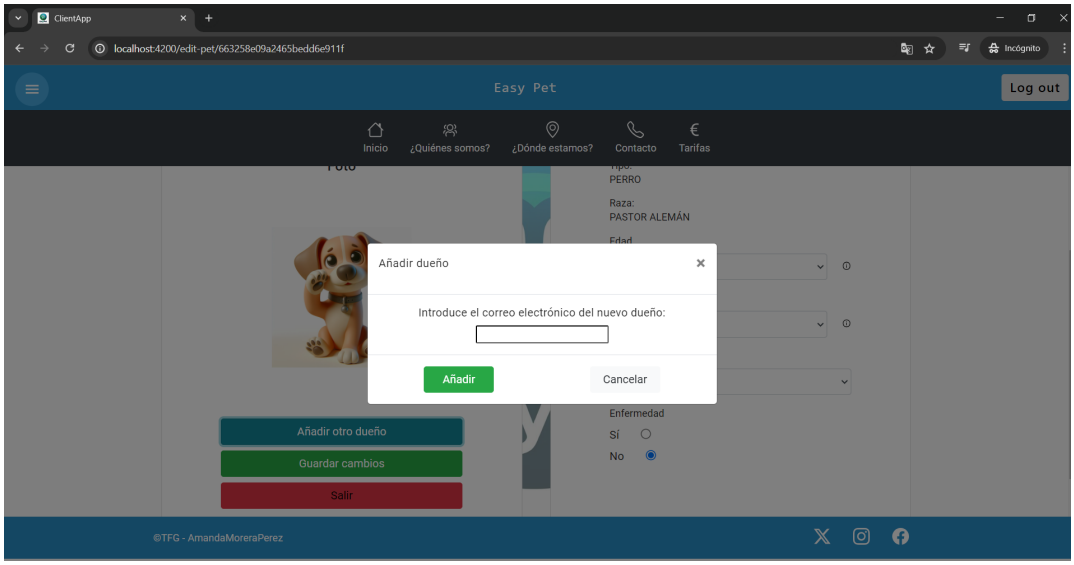


Figura A.19: Pantalla de añadir dueño a la mascota

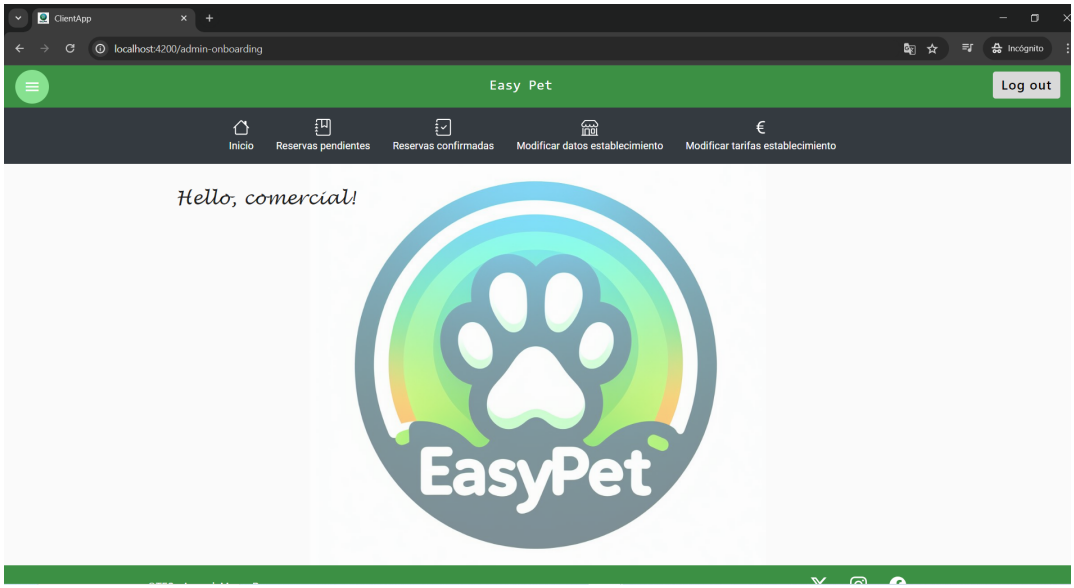


Figura A.20: Pantalla de inicio del gerente

Si se hace click en reservas pendientes, será redirigido al listado de aquellas reservas cuyo estado sea pendiente (ver Figura A.21. Dentro de esta pantalla, se puede hacer click en el botón de confirmar, en cuyo caso la reserva se actualizará al estado de aceptada y quedará pendiente del pago. Como caso contrario, se puede hacer click en el botón de cancelar, en cuyo caso se actualizará al estado de cancelada. También si se hace click en ella, se accederá

al detalle de la reserva.

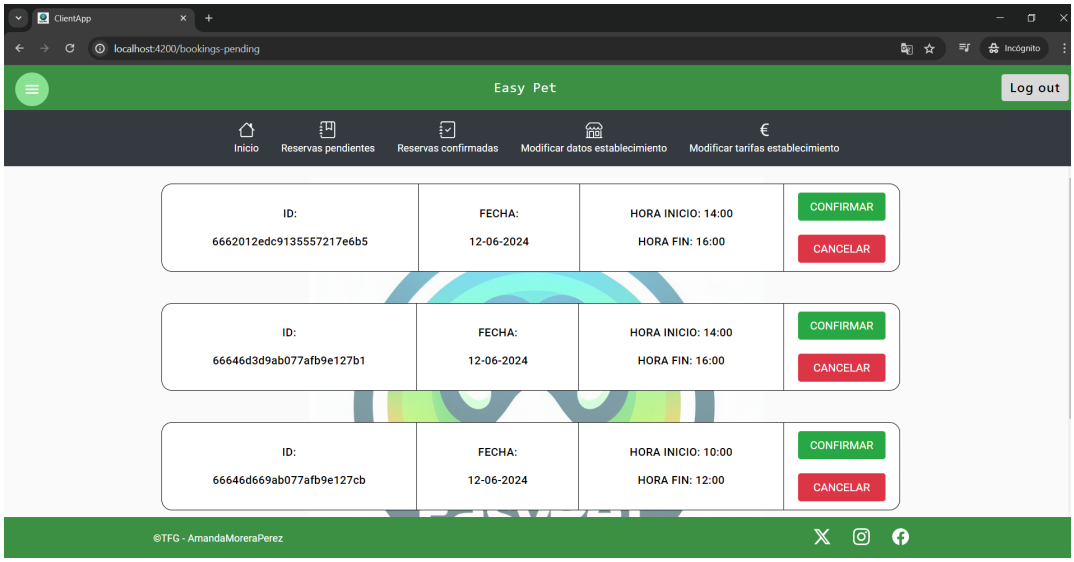


Figura A.21: Pantalla de reservas pendientes

Si se hace click en reservas confirmadas, será redirigido al listado de aquellas reservas cuyo estado sea confirmada (ver Figura A.22). Si se hace click en una de ellas, será redirigido al detalle de la misma. Dentro de esta pantalla, se puede añadir un pago adicional que tenga lugar el mismo día de la reserva que se haya hecho en metálico (ver Figura A.23) y será redirigido al listado de las reservas confirmadas, o se puede hacer click en el botón de cancelar se actualizará la reserva el estado de cancelada y se recargará la página para que se vea el detalle actualizado.

Si se hace click en modificar datos establecimiento, aparecerá un formulario dónde se rellenarán los datos que se deseen modificar (ver Figura A.24), y en el que aparecerá por defecto lo que haya almacenado. Si se pulsa en el botón de guardar cambios, se actualizará los datos del establecimiento con los cambios y se actualizará la pantalla para que se vean reflejados los cambios. Si se pulsa el botón salir sin guardar, será redirigido a la pantalla de inicio.

Si se hace click en modificar tarifas establecimiento, aparecerá un formulario dónde se modificarán las tarifas que se desee (ver Figura A.25), y aparecerá por defecto lo que ya esté almacenado. Si se pulsa el botón de guardar cambios, se actualizarán las tarifas con los cambios y se actualizará la pantalla para que se vean reflejados con los cambios. Si se pulsa el botón de salir sin guardar, será redirigido a la pantalla de inicio.

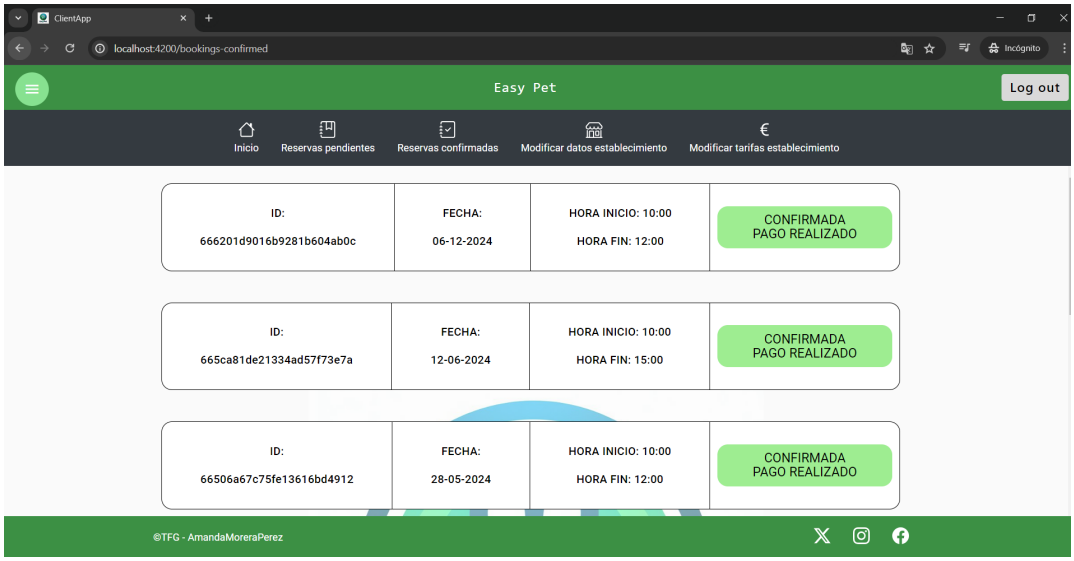


Figura A.22: Pantalla de reservas confirmadas

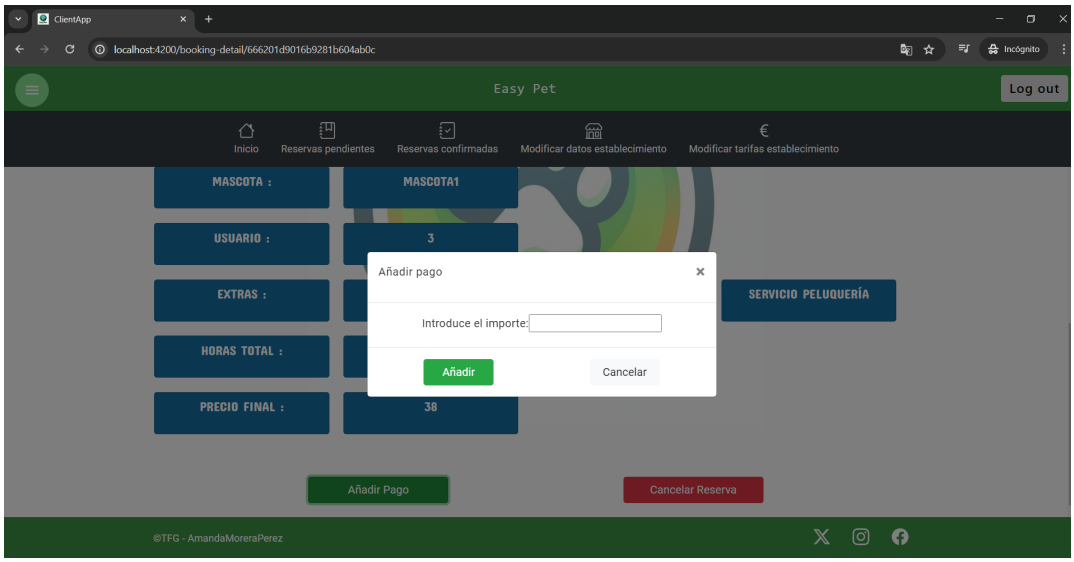


Figura A.23: Pantalla de pago adicional a la reserva

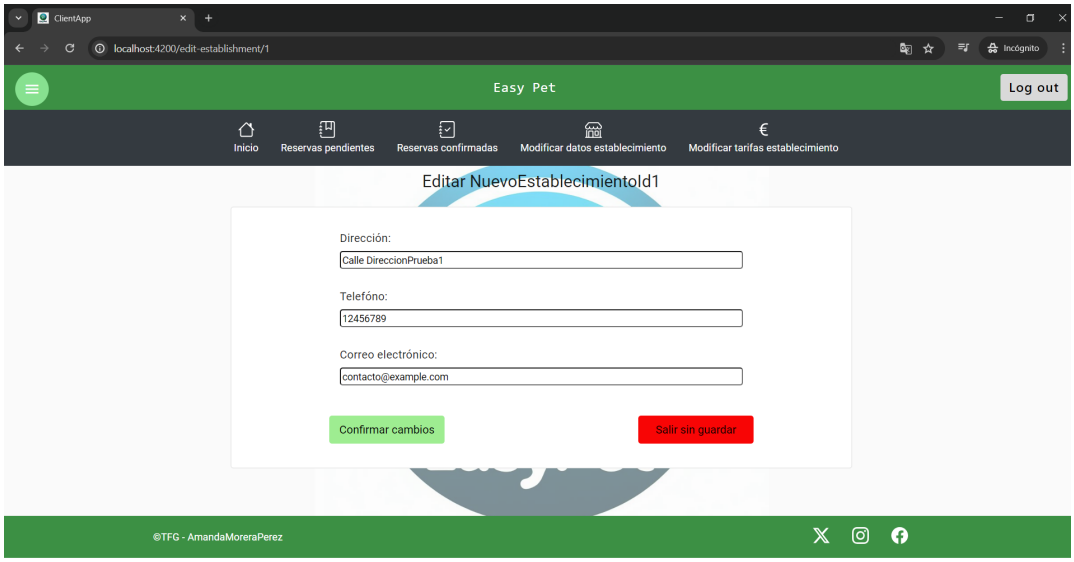


Figura A.24: Pantalla de editar establecimiento

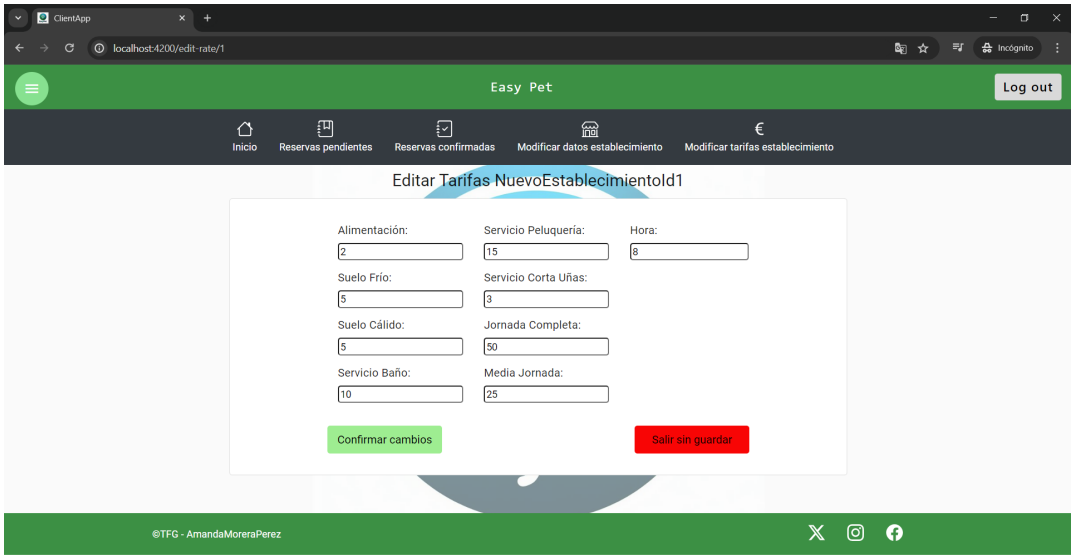


Figura A.25: Pantalla de editar tarifas

Apéndice B

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorio del código en GitLab: `https://gitlab.inf.uva.es/amamore/tfg_morera_perez_amanda`.
- Repositorio de GitHub utilizado para el despliegue de la aplicación: `https://github.com/tfg-amanda/tfg-amanda-morera-perez-deployment`
- Aplicación desplegada en Vercel: `https://tfg-amanda-morera-perez-deployment-frontend.vercel.app/`
- Diseño UX realizado en Figma: `https://www.figma.com/design/3wEprNEG5LS7SgRMySC3aD/TFGAmandaMoreraPÃ‰rez?node-id=0-1`
- Despliegue backend aplicación en Vercel: `https://tfg-amanda-morera-perez-deployment-backend.vercel.app/`