CanSport. Gestión de reservas

Manuel Andrés Mora del Pozo
Grado de Ingeniería Informática
JavaEE

Consultor
Vicenç Font Sagrista
Santi Caballe Llobet

05/2020

Agradecimientos

Trabajando en este trágico periodo de pandemia y confinamiento solo he tenido ejemplos de trabajo duro y superación.
A mi pareja, que como sanitaria estuvo cada día en primera línea, por su energía y vocación al trabajo a pesar de llegar el día anterior agotada.
A mi hijo, valiente y decidido, que realiza estudios universitarios a más de 2.000 km.
A mi familia y amigos, que teniéndolos tan cerca no pude acercarme.
Y en memoria de los que ya no están y no pude despedirme.

Gracias.

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	***CanSport**. Gestión de reservas*
Nombre del autor:	*Manuel Andrés Mora del Pozo*
Nombre del consultor/a:	*Vicenç Font Sagrista*
Nombre del PRA:	*Santi Caballe Llobet*
Fecha de entrega (mm/aaaa):	05/2020
Titulación::	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*JavaEE*
Idioma del trabajo:	*Castellano*
Palabras clave	*JavaEE, JSF, Primefaces*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La pequeña empresa necesita el apoyo de la universidad para vencer una brecha en la información y aplicación de las nuevas tecnologías. Son muchas las historias de usuarios idénticas en estas entidades, pero no disponen de recursos y acceso para actualizar métodos de gestión que dificultan el acceso de los clientes.

Este proyecto aportará una solución particular y sencilla para un pequeño negocio de residencia canina. Ofrecerá al usuario accesibilidad a su información y las reservas de estancias, como ya lo realiza para sus propias vacaciones. De bajo coste y mantenimiento.

La metodología y propia gestión del proyecto es consecuencia del itinerario formativo que la universidad nos brinda. Pautas para la planificación, patrones de diseño y desarrollo, reutilización de componentes y programación en la infraestructura que hemos venido practicando en los laboratorios.

La práctica que proporciona el TFG permite la inmersión en todos los roles involucrados en la realización de proyectos. La experiencia real de trabajo en equipo, en ocasiones te limita y especializa en ciertos aspectos del ciclo de vida del desarrollo.

En definitiva, el producto obtenido ofrece un canal de recursos para el cliente, que demanda acceso directo e inmediato para gestionar su necesidades diarias. Ahora dispone de la opción para programar estancias para sus mascotas desde su ordenador o dispositivo móvil con acceso a internet.

Abstract (in English, 250 words or less):

Small businesses need the support of the university to overcome a gap in the information and application of new technologies. There are many stories of common users, but they do not have the resources and accessibility to update management methods that facilitate customer access.

This project will provide a particular and simple solution for a small dog-sitting business to offer the user accessibility to their information and reservation of stays, as they already do for their own holidays. Low cost and maintenance.

The methodology and management of the project is a consequence of the training itinerary that the university offers us. Guidelines for planning, design and development patterns, reuse of components and programming in the infrastructure that we have been practicing in the laboratories.

The practice provided by the TFG (Trabajo Fin de Grado) allows immersion in all the roles involved in carrying out projects. Real teamwork experience sometimes limits and specializes you in certain aspects of the development life cycle.

To summarize, the product obtained offers a channel of resources for the client that demands direct and immediate access to manage their daily needs. Now you have the option to schedule stays for your pets from your computer or mobile device with internet access.

Índice

1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 1
1.2.1 Objetivos generales y específicos 2
1.2.2 Herramientas y *frameworks* 2
1.3 Enfoque y método seguido 3
1.4 Planificación del Trabajo 4
1.5 Breve sumario de productos obtenidos 6
1.6 Breve descripción de los otros capítulos de la memoria 7
2 Análisis y diseño 8
2.1 Requisitos 8
2.1.1 Roles 8
2.1.2 Requisitos funcionales 8
2.1.3 Módulos del sistema 9
2.1.4 Requisitos no funcionales 9
2.2 Casos de uso 10
2.2.1 Diagrama de casos de uso 11
2.2.2 Fichas de casos de uso 12
2.3 Modelo de pantallas 26
2.3.1 Diagrama de navegación 26
2.3.2 Prototipos de pantallas 27
2.4 Diseño relacional de BBDD 38
2.5 Diagrama de clases 39
2.6 Diagrama de arquitectura 42
2.7 Arquitectura orientada a Java Enterprise Edition 44
2.7.1 Capa de presentación 44
2.7.2 Capa de negocio 45
2.7.3 Capa de integración 46
2.7.4 Servidor de aplicaciones 46
2.8 Diseño y modelado de la arquitectura JavaEE 47
2.8.1 Capa de presentación 47
2.8.2 Capa de negocio 52
2.8.3 Capa de integración 54
3 Implementación 56
3.1 Producto obtenido 57
3.2 Mejoras sugeridas 62
4 Conclusiones 63
5 Glosario 64
6 Bibliografía 65
7 Anexos 66
7.1 Implantación 66
7.1.1 Base de datos 66
7.1.2 Desplegando la aplicación 66

Lista de figuras

Ilustraciones 1 Ranking lenguajes de programación más usados [2] 3
Ilustración 2 Planificación del Trabajo y calendario del plan docente 4
Ilustración 3 Planificación de PEC1 5
Ilustración 4 Planificación de PEC2 5
Ilustración 5 Planificación de PEC3 5
Ilustración 6 Planificación de Memoria y presentación del TFG 6
Ilustración 7 Diagrama de uso. Módulo de administración 11
Ilustración 8 Diagrama de uso. Módulo de clientes 11
Ilustración 9 Diagrama de uso. Módulo de reservas 11
Ilustración 10 Diagrama de navegación 26
Ilustración 11 Prototipos de pantalla. Login 27
Ilustración 12 Prototipo de cabecera de pantallas 27
Ilustración 13 Prototipo de cabecera de pantallas con menú completamente desplegado 27
Ilustración 14 Prototipo de plantilla de pantallas 28
Ilustración 15 Prototipos de pantalla. Consulta de administradores 28
Ilustración 16 Prototipos de pantalla. Elimando administrador 29
Ilustración 17 Prototipos de pantalla. Edición de administrador 29
Ilustración 18 Prototipos de pantalla. Consulta de estancias 30
Ilustración 19 Prototipos de pantalla. Eliminando estancia 30
Ilustración 20 Prototipos de pantalla. Edición de estancia 31
Ilustración 21 Prototipos de pantalla. Edición de datos residencia 31
Ilustración 22 Prototipos de pantalla. Edición de cliente 32
Ilustración 23 Prototipos de pantalla. Consulta de clientes 32
Ilustración 24 Prototipos de pantalla. Eliminando cliente 33
Ilustración 25 Prototipos de pantalla. Consulta de reservas de cliente 33
Ilustración 26 Prototipos de pantalla. Consulta de mascotas 34
Ilustración 27 Prototipos de pantalla. Edición de mascota 34
Ilustración 28 Prototipos de pantalla. Alta reserva. disponibilidad 35
Ilustración 29 Prototipos de pantalla. Alta reserva 35
Ilustración 30 Prototipos de pantalla. Consulta de reservas 36
Ilustración 31 Prototipos de pantalla. Eliminando reserva 36
Ilustración 32 Prototipos de pantalla. Aviso de error 37
Ilustración 33 Prototipos de pantalla. Aviso de advertencia 37
Ilustración 34 Prototipos de pantalla. Mensaje informativo 37
Ilustración 35 Entidad relación del proyecto 38
Ilustración 36 Diagrama de clases 39
Ilustración 37 Diseño general. Punto de vista de la computación 42
Ilustración 38 Dependencias componentes y entidades 44
Ilustración 39 Diseño general. Capa presentación 44
Ilustración 40 Diseño en detalle. Capa presentación 45
Ilustración 41 Diseño detalle JavaEE. Capa presentación. Administración 49
Ilustración 42 Diseño detalle JavaEE. Capa presentación. Clientes 50
Ilustración 43 Diseño detalle JavaEE. Capa presentación. Reservas 51
Ilustración 44 Diseño detalle JavaEE. Capa negocio. Administración 52

Ilustración 45 Diseño detalle JavaEE. Capa negocio. Clientes 53
Ilustración 46 Diseño detalle JavaEE. Capa negocio. Reservas 53
Ilustración 47 Diseño detalle JavaEE. Capa integración. Administración 54
Ilustración 48 Diseño detalle JavaEE. Capa integración. Clientes 55
Ilustración 49 Diseño detalle JavaEE. Capa integración. Reservas 55
Ilustración 51 Menú para cliente 57
Ilustración 50 Menú para administrador 57
Ilustración 52 Consulta de clientes 58
Ilustración 53 Administración de datos de cliente 58
Ilustración 54 mantenimiento de datos de contacto de la organización 59
Ilustración 55 Consulta de administradores 59
Ilustración 56 Edición de datos de administrador 59
Ilustración 57 Consulta de estancias de la residencia 59
Ilustración 58 Edición de la estancia 60
Ilustración 59 Consulta de mascotas 60
Ilustración 60 Edición de datos de mascota 60
Ilustración 61 Consulta de reservas 61
Ilustración 62 Crear nueva reserva 61
Ilustración 63 Pantalla general. usuario administrador 61
Ilustración 64 Ejemplo de edición propiedad de carpeta de despliegue 66

1. Introducción

1.1 Contexto y justificación del Trabajo

Es sorprendente, que tras recorrer una quinta parte de este siglo XXI, vestimos complementos que son realmente verdaderos avances de la tecnología pero observemos como pequeños negocios recurran al uso de una agenda tradicional para organizarse. Aún utilizan el papel para registrar reservas y los contactos de sus clientes.

El problema de aquellos sistemas de gestión anclados en el pasado no está en la falta de información, es más bien el exceso de esta y el tiempo requerido para filtrarla y estar siempre al día. Esta es la evidencia de una brecha digital y cultural en el pequeño negocio. Muchas empresas se crean y crecen innovando, pero otras permanecen fieles a métodos tradicionales. Desconocen opciones gratuitas o de bajo coste que les facilitaría gestionarse.

El proyecto **CanSport** aborda y ofrece una sencilla solución para sustituir aquella acumulación de agendas saturadas de anotaciones, de contactos de dueños (clientes) difíciles de mantener. Además, proporcionará a los clientes una ventana al sistema, permitiendo gestionar y actualizar los datos de contacto. Conocer la disponibilidad de plazas en tiempo real, pudiendo reservar y consultar el historial.

Para la residencia canina, proporcionará una herramienta de gestión de reservas y una base de datos de contactos siempre disponible online. Evitará las constantes llamadas perdidas para solicitar estancias y que se producían durante la realización de las actividades del recinto. Llamadas perdidas al no poder atender el teléfono, o únicamente atender a una sola llamada en periodos de alta demanda.

En definitiva, encontraremos en este proyecto una propuesta desarrollada como solución simple, orientada a la necesidad de aplicar herramientas TIC para mejorar en la organización y la toma de decisiones. Atendiendo a un aspecto fundamental de un pequeño negocio, pero con una inversión baja para su implantación y mantenimiento. Su desarrollo se ha realizado con productos exentos de licencias costosas.

1.2 Objetivos del Trabajo

Como objetivo principal del trabajo de fin de grado nos comprometemos a la realización de un sistema de información para la gestión de reservas y contactos de clientes, accesible desde una aplicación en entorno web.

El proyecto proporciona un escenario general para el aprendizaje y práctica en el desarrollo de una aplicación web mediante *javaEE*, *JSF*, *frameworks* de componentes *Primefaces*, *PostgreSQL* y *WildFly* (*JBoss*).

1.2.1 Objetivos generales y específicos

1. Registro de clientes y sus mascotas
   1.1. Agenda de clientes
   1.2. Registro de mascotas
   1.3. Historial de reservas
2. Registro de estancias
   2.1. Mantenimiento de estancias
   2.2. Anulación de disponibilidad
   2.3. Historial de reservas
3. Gestión de reservas de estancias disponibles
   3.1. Alta de reservas
   3.2. Anulación de reservas
4. Consulta de disponibilidad de estancias
   4.1. Número de estancias disponibles
   4.2. Consulta de estancias disponibles

1.2.2 Herramientas y *frameworks*

Para conseguir los objetivos del proyecto de ingeniería de software utilizaremos los elementos de software libre establecidos en el tutorial de instalación de laboratorio de asignaturas de la carrera [1] que citamos en el apartado de referencias. A través de este documento obtenemos una guía de introducción e instalación a las herramientas en sus versiones estables en la fecha de su edición.

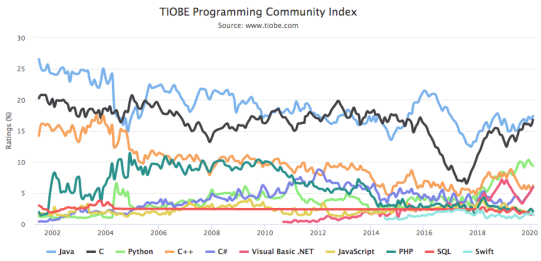
Software:

- JDK 1.8 (*build* 1.8.0_221-b11)
- *PostgreSQL* versión 10.10-1
- *Connector Java PostgreSql* 9.4.1209
- *WildFly*, versión 14.0.1.Final (*JBoss*)
- *Eclipse Java EE IDE for Web Developers*. Versión: 2018-09 (4.9.0)
- *Apache Maven 3*.6.3
- *JBoss Tools* 4.9.0.Final
- *Eclipse EGit* 5.4.2
- *GitLab Enterprise Edition* 12.9.0-pre
- *Primefaces 7.0*
- *Trello Atalasian Cloud*
- *Gantt Project (free edition)*
- *Magic Draw*, versión 18.5 PE
- *LucidChart* (licencia corporativa)

La elección de java como lenguaje de programación para el desarrollo del proyecto, viene promovida por ser el lenguaje principal utilizado en las prácticas de laboratorio de las asignaturas del itinerario de ingeniería de software. Es tendencia y práctica habitual, en empresas de desarrollo de aplicaciones, que utilicen JavaEE para aplicaciones web con diseño responsivo, permitiendo una visualización adaptada para distintos dispositivos móviles.

Podemos comprobar en las siguientes tablas comparativas extraídas del índice TIOBE de febrero de 2020, donde apreciamos el ranking y la tendencia en un amplio periodo de tiempo. Apreciamos especialmente significado a la cita: "*The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system*" [2] (2020 TIOBE Software BV).

Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.358%	+1.48%
2	2		C	16.766%	+4.34%
3	3		Python	9.345%	+1.77%
4	4		C++	6.164%	-1.28%
5	7	^	C#	5.927%	+3.08%
6	5	v	Visual Basic .NET	5.862%	-1.23%
7	6	v	JavaScript	2.060%	-0.79%
8	8		PHP	2.018%	-0.25%
9	9		SQL	1.526%	-0.37%



Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	22	21	-	-
C++	4	4	4	3	2	1	2	12
C#	5	5	5	8	8	-	-	-
Visual Basic .NET	6	10	-	-	-	-	-	-
JavaScript	7	8	8	9	6	-	-	-
PHP	8	6	3	4	27	-	-	-
SQL	9	-	-	97	-	-	-	-

Ilustraciones 1 Ranking lenguajes de programación más usados [2]

1.3 Enfoque y método seguido

El enfoque se basará en la realización de documentos del proyecto en distintas fases, indicadas en el plan docente a lo largo del desarrollo de la asignatura y creando una aplicación desde cero. Los objetivos se obtienen de las necesidades particulares de una empresa local. El objetivo general será el diseño básico de un módulo de gestión de reservas. Esta podría ser solución general para un gran número de pequeños negocios de este tipo.

La disciplina y evolución de la asignatura marcará unas pautas de entregables, que prácticamente nos llevan a un sistema de desarrollo en cascada. Sin embargo, el continuo contacto con el tutor de la asignatura nos permitirá interactuar de forma activa, pudiendo integrar elementos de metodologías agiles a través del *feedback* continuo. Seguiremos una planificación inicial que nos facilitará la organización del trabajo, atendiendo al cumplimiento de estas entregas. De forma complementaria, utilizaremos herramientas de *planning Kanban* para priorizar y desgranar en tareas más detalladas y sencillas.

1.4 Planificación del Trabajo

La planificación de desarrollo y evolución del proyecto viene acotado por el plan docente de la asignatura, y finalizará tras la evaluación del tribunal. Por tanto, se verá condicionado a formalizarse en fases cuya fecha de cumplimiento van asociadas aproximadamente a los plazos de entrega:

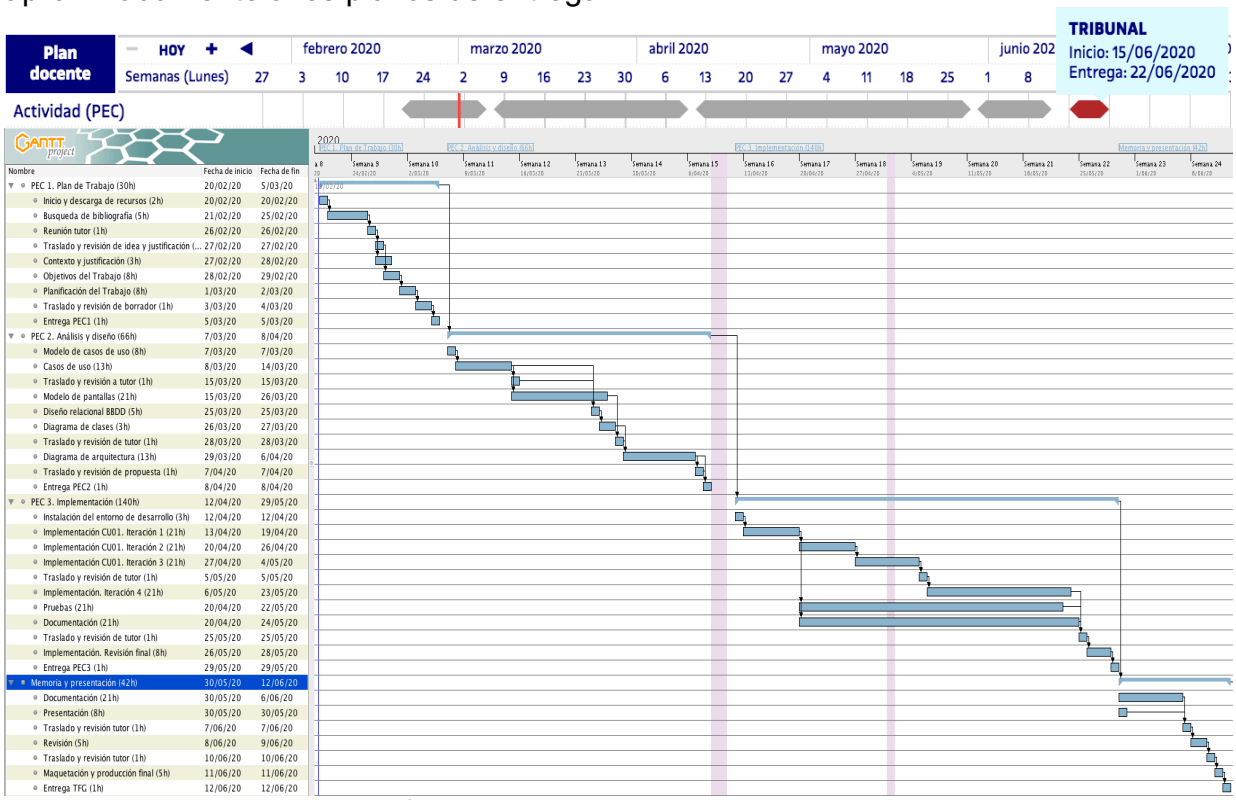


Ilustración 2 Planificación del Trabajo y calendario del plan docente

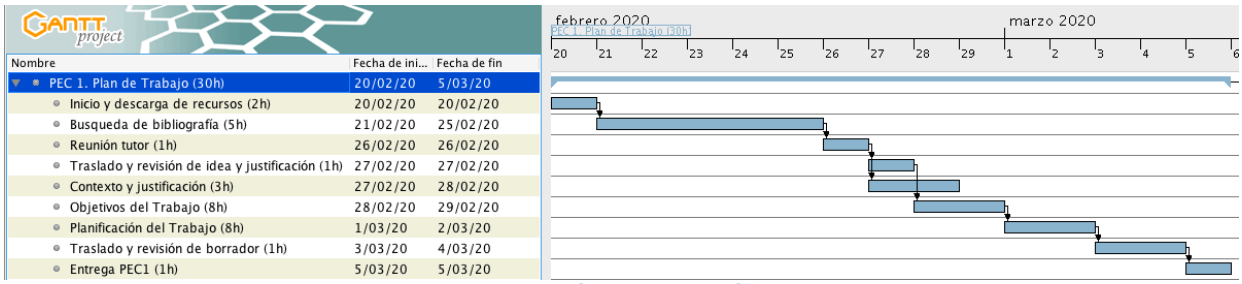
Nota: Para la asignación de horas a las subtareas de cada fase se ha seguido la sucesión de Fibonacci.

A continuación, veremos con más detalle cada fase de la planificación del trabajo:

PEC1. Plan de trabajo

En esta fase iniciaremos el contacto con el tutor del proyecto, recopilaremos recursos didácticos y describiremos el contexto y justificación. Se establecerán los objetivos de la aplicación y estableceremos una planificación detallada a través de tareas y actividades en un calendario ajustado a los periodos de cada PEC.

El principal producto que obtendremos en este periodo inicial, será un elaborado plan de trabajo. Sin embargo, en esta primera etapa aún no podremos desgranar los componentes y detallar subtareas. Por tanto, quedan reflejadas en un alto nivel y ajustaremos las fechas para cumplir con los propósitos descritos en la asignatura, incluyendo entregas parciales y validando la dirección tomada en las tareas intermedias.

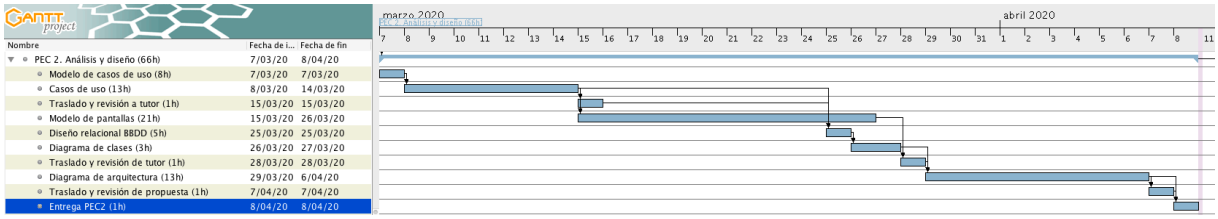


Nombre	Fecha de ini...	Fecha de fin
PEC 1. Plan de Trabajo (30h)	20/02/20	5/03/20
Inicio y descarga de recursos (2h)	20/02/20	20/02/20
Busqueda de bibliografía (5h)	21/02/20	25/02/20
Reunión tutor (1h)	26/02/20	26/02/20
Traslado y revisión de idea y justificación (1h)	27/02/20	27/02/20
Contexto y justificación (3h)	27/02/20	28/02/20
Objetivos del Trabajo (8h)	28/02/20	29/02/20
Planificación del Trabajo (8h)	1/03/20	2/03/20
Traslado y revisión de borrador (1h)	3/03/20	4/03/20
Entrega PEC1 (1h)	5/03/20	5/03/20

Ilustración 3 Planificación de PEC1

PEC2. Análisis y diseño

A través de los requisitos crearemos un modelo de casos de uso, describiremos los actores e identificaremos los casos de uso y los componentes que compondrán nuestro sistema. Completaremos el diseño del modelo con la descripción en ficha de dichos casos y construiremos un modelo de navegación y prototipos donde reflejar los escenarios expuestos. Completaremos esta fase con un modelo relacional y diagrama de clases. Finalizando con el diagrama de arquitectura.



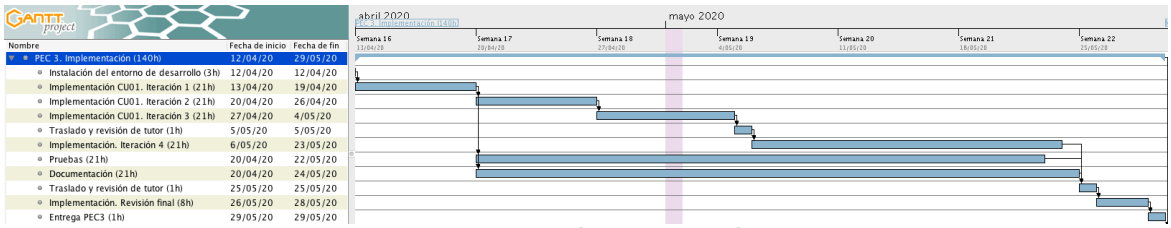
Nombre	Fecha de i...	Fecha de fin
PEC 2. Análisis y diseño (66h)	7/03/20	8/04/20
Modelo de casos de uso (8h)	7/03/20	7/03/20
Casos de uso (13h)	8/03/20	14/03/20
Traslado y revisión a tutor (1h)	15/03/20	15/03/20
Modelo de pantallas (21h)	15/03/20	26/03/20
Diseño relacional BBDD (5h)	25/03/20	25/03/20
Diagrama de clases (3h)	26/03/20	27/03/20
Traslado y revisión de tutor (1h)	28/03/20	28/03/20
Diagrama de arquitectura (13h)	29/03/20	6/04/20
Traslado y revisión de propuesta (1h)	7/04/20	7/04/20
Entrega PEC2 (1h)	8/04/20	8/04/20

Ilustración 4 Planificación de PEC2

PEC3. Implementación

Llegados a este periodo, instalaremos y revisaremos el entorno creando un repositorio donde mantener el proyecto. Iniciaremos la implementación según la documentación previo y estableceremos varias iteraciones para completar un caso de uso significativo. Esto nos permitirá un primer contacto con los criterios establecidos limitándonos a un desarrollo simple. En una segunda iteración incorporaremos usabilidad y gestión de errores. Y en una tercera, completaremos y afinaremos la implementación añadiendo validaciones.

Nos servirá de base para trasladar al resto de casos de uso, basándonos en la experiencia adquirida. Reduciendo así, la repetición de errores y permitiendo reutilizar componentes. Durante el proceso iniciaremos la documentación y pruebas del sistema que continuaremos en cada iteración hasta la revisión final previa a la entrega del software producido.

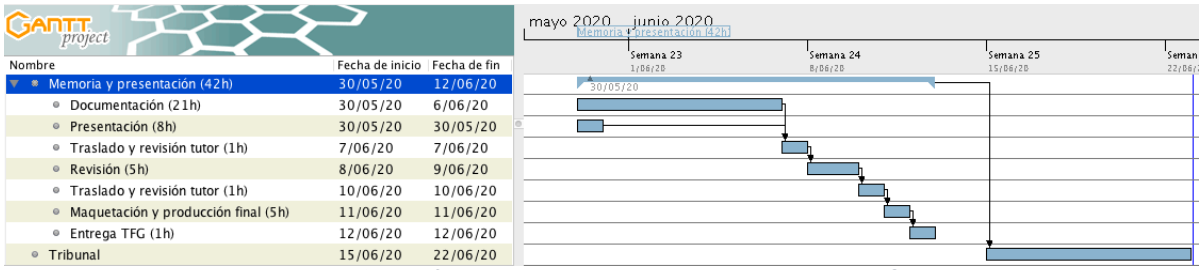


Nombre	Fecha de inicio	Fecha de fin
PEC 3. Implementación (140h)	12/04/20	29/05/20
Instalación del entorno de desarrollo (3h)	12/04/20	12/04/20
Implementación CU01. Iteración 1 (21h)	13/04/20	19/04/20
Implementación CU01. Iteración 2 (21h)	20/04/20	26/04/20
Implementación CU01. Iteración 3 (21h)	27/04/20	4/05/20
Traslado y revisión de tutor (1h)	5/05/20	5/05/20
Implementación. Iteración 4 (21h)	6/05/20	23/05/20
Pruebas (21h)	20/04/20	22/05/20
Documentación (21h)	20/04/20	24/05/20
Traslado y revisión de tutor (1h)	25/05/20	25/05/20
Implementación. Revisión final (8h)	26/05/20	28/05/20
Entrega PEC3 (1h)	29/05/20	29/05/20

Ilustración 5 Planificación de PEC3

Memoria y presentación

La fase final será completada con la documentación y revisión de la memoria y manual de instalación y una presentación para el tribunal que evaluará el producto terminado.



Nombre	Fecha de inicio	Fecha de fin
Memoria y presentación (42h)	30/05/20	12/06/20
Documentación (21h)	30/05/20	6/06/20
Presentación (8h)	30/05/20	30/05/20
Traslado y revisión tutor (1h)	7/06/20	7/06/20
Revisión (5h)	8/06/20	9/06/20
Traslado y revisión tutor (1h)	10/06/20	10/06/20
Maquetación y producción final (5h)	11/06/20	11/06/20
Entrega TFG (1h)	12/06/20	12/06/20
Tribunal	15/06/20	22/06/20

Ilustración 6 Planificación de Memoria y presentación del TFG

1.5 Breve sumario de productos obtenidos

Plan de trabajo

Punto de inicio fundamental para la organización de las tareas a desarrollar para lograr los objetivos del proyecto. Encontramos descrito en el capítulo anterior la distribución de las fases y tareas principales en una escala temporal. Comprobamos las dependencias y posibles conflictos.

Diseño

Análisis de los requisitos y diseño de propuestas a los casos de uso. Toma de decisiones sobre la infraestructura, prototipado de los interfaces de usuario, diagrama de navegación y diseño de la arquitectura.

Implementación

Desarrollo de la aplicación a través varias de iteración de periodos cortos, avanzando primero en un caso de uso y creando componentes de valor que han podido reutilizarse en los siguientes casos de uso en iteraciones posteriores.

Presentación

En esta última etapa se compone la memoria a partir de los documentos generados en las fases anteriores y se prepara una presentación.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos de la memoria encontraremos los detalles de análisis y diseño. Enumerando y describiendo los requisitos basados en las historias de usuario. Se introducirá al diseño propuesto de una forma general hasta completar detalles del diseño.

Finalizaremos el capítulo sobre el diseño definiendo la propuesta a la tecnología elegida para la elaboración del proyecto. Recorriendo una evolución del diseño general de componentes, hasta detallar con los prototipos de interfaz y diseño de la arquitectura.

El capítulo dedicado a la implementación describirá la experiencia, cumplimiento de los objetos marcados, sugerencia de mejoras, aspectos del desarrollo y organización obtenidos a lo largo de ese periodo y la experiencia como itinerario de aprendizaje en la realización de proyectos y uso de componentes de software.

Finalizaremos la memoria con una muestra del resultado visual de los formularios como muestra de una navegación entre las opciones, que según el rol del usuario, tendrá disponibles. Tras incorporar un capítulo de conclusiones finales, se añaden capítulos de glosario básico, referencias y un anexo para informar de detalles de despliegue y puesta en marcha.

2 Análisis y diseño

2.1 Requisitos

A partir de los objetivos establecidos previamente elaboraremos la documentación necesaria para abordar nuestro proyecto. En este apartado enumeraremos y describiremos los perfiles, casos de uso, módulos y entidades del sistema de información.

2.1.1 Roles

- **Administrador** (*Administrator*). Este perfil es compartido por los usuarios con la responsabilidad de Introducir y mantener la información básica para el funcionamiento del sistema.

- **Cliente** (*Customer*). Este rol permite acceder al sistema para realizar reservas y mantener los datos de las mascotas e información de contacto.

- **Usuario no registrado** (*NoCustomer*). Será cada usuario que aún no está registrado en el sistema. Siempre tendrá acceso al formulario de alta de clientes para formalizar su relación con la entidad y posteriormente tener acceso como cliente.

2.1.2 Requisitos funcionales

Atendiendo a las historias de usuario recogidas podremos agrupar y establecer unos requisitos funcionales por perfil.

- **Datos del centro**. Mantenimiento básico con los datos descriptivos y de contacto del centro (Administrador).

- **Gestión de estancias**. Registro de las estancias y mantenimiento del estado (Administrador).

- **Alta de clientes**. Registro de datos personales (usuario no registrado, Administrador).

- **Gestión de clientes**. Consulta filtrada para acceder/editar los datos de los clientes (Cliente, Administrador).

- **Gestión de mascotas**. Registro y mantenimiento de los datos de las mascotas de los clientes (Cliente, Administrador).

- **Consulta de mascotas**. Consulta filtrada para acceder/editar los datos de las mascotas (Administrador).

- **Consulta disponibilidad entre fechas**. Consultar si existen estancias disponibles entre una fecha de inicio y fin (Cliente, Administrador).

- **Reserva de estancia entre fechas**. Asignación reserva entre una fecha de inicio y finalización (Cliente, Administrador).

- **Eliminación de reserva**. Eliminar reserva entre una fecha de entrada y salida (Cliente, Administrador).

- **Consulta historial de reservas de cliente**. Relación de reservas realizadas por un cliente. (Cliente, Administrador).

2.1.3 Módulos del sistema

A partir de los requisitos podemos establecer una división funcional en módulos. Esto nos permitirá organizar, distribuir las tareas y mejorar la planificación del desarrollo del trabajo a realizar para alcanzar nuestros objetivos.

- **Administración del sistema** (*Administration*). Agrupando los casos de uso relativos al registro y mantenimientos generales de información básica.

- **Gestión de usuarios y mascotas** (*Customer*). Todo lo relativo a la gestión de datos de clientes, sus mascotas y consulta de historial.

- **Gestión de reservas** (*Booking*). Módulo para la consulta de disponibilidad y confirmación o eliminación de reservas.

2.1.4 Requisitos no funcionales

- Interfaz de usuario sencilla y de fácil interpretación.
- Accesibilidad a través de internet.
- Acceso seguro a través de usuario y contraseña.
- Tiempos de respuesta adecuados.
- Libre de licencias de pago.

2.2 Casos de uso

Acceso y administración

Generales

CUA.1. Identificarse en el sistema.
CUA.2. Salir del sistema.
CUA.3. Editar datos del centro canino.

Gestionar estancias

CUA.4. Alta de estancia.
CUA.5. Consulta de estancia.
CUA.6. Anulación de estancia.
CUA.7. Modificación de estancia.

Gestión de administradores

CUA.8. Alta de administrador.
CUA.9. Consulta de administrador.
CUA.10. Eliminación de administrador.
CUA.11. Modificación de administrador.

Clientes

Gestionar clientes

CUC.1. Alta de cliente.
CUC.2. Consulta de cliente.
CUC.3. Eliminar cliente
CUC.4. Anulación de cliente.
CUC.5. Modificación de cliente.

Gestionar mascotas

CUC.5. Alta de mascota.
CUC.6. Consulta de mascota.
CUC.7. Anulación de mascota.
CUC.8. Modificación de mascota.

Reservas

Generales

CUB.1 Consultar disponibilidad entre fechas

Gestión de reservas

CUB.2 Alta de reserva entre fechas.
CUB.3 Consulta de reserva.
CUB.4 Eliminar reserva.
CUB.5 Consulta de historial de reservas por cliente.

2.2.1 Diagrama de casos de uso

Administración

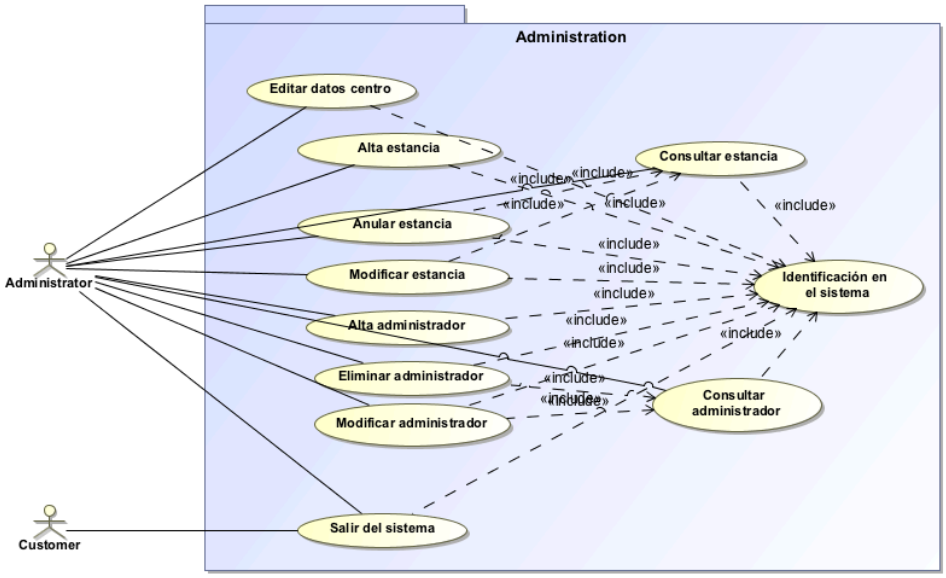


Ilustración 7 Diagrama de uso. Módulo de administración

Clientes

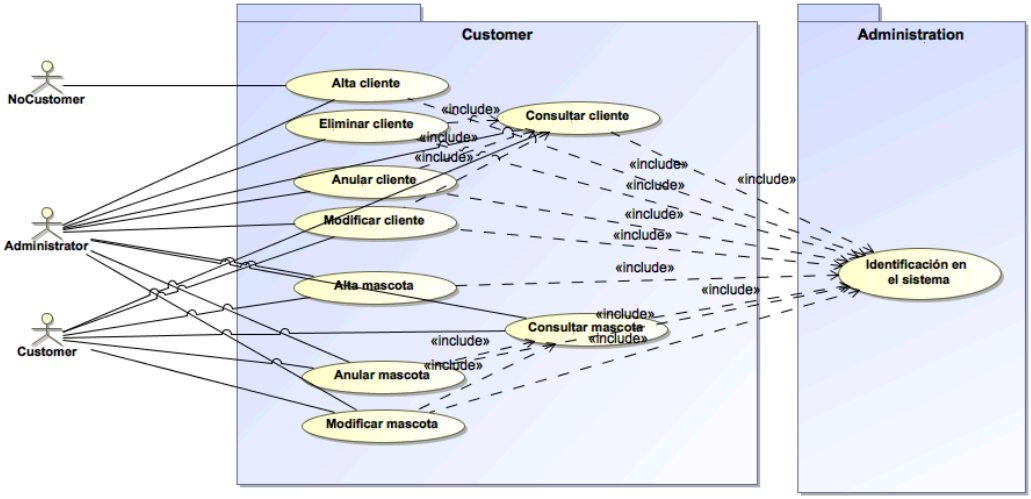


Ilustración 8 Diagrama de uso. Módulo de clientes

Reservas

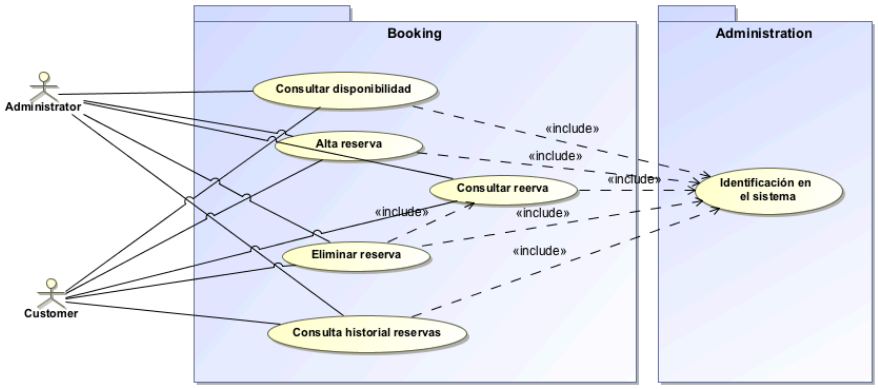


Ilustración 9 Diagrama de uso. Módulo de reservas

2.2.2 Fichas de casos de uso

Acceso y administración

CUA.1 Identificarse en el sistema	
Actor principal	Administrador, cliente
Objetivo	Acceso al sistema
Precondiciones	Usuario registrado y no autentificado
Postcondiciones	Usuario autentificado
Escenario principal	1. El sistema muestra el formulario de *login*, se solicita usuario y contraseña de forma obligatoria. 2. El usuario (administrador o cliente) introducen su usuario, la contraseña de acceso y pulsa sobre botón Acceder. 3. El sistema valida que exista el usuario con esa contraseña en la base de datos del sistema. 4. Se accede a la página principal de la aplicación.
Escenario alternativo	3a. El sistema indica que el usuario no existe 3b. El sistema indica que la contraseña no es válida 3c. El sistema indica que usuario tiene restringido el acceso porque el usuario está anulado)

CUA.2 Salir del sistema	
Actor principal	Administrador, cliente
Objetivo	Cerrar sesión
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Usuario no autentificado
Escenario principal	1. El sistema muestra botón de salida del sistema. 2. El usuario (administrador o cliente) pulsan sobre la opción de cerrar sesión. 3. El sistema cierra la sesión. 4. Se muestra la pantalla de acceso al sistema.
Escenario alternativo	

CUA.3 Editar datos del centro	
Actor principal	Administrador
Objetivo	Administrar datos de la residencia canina
Precondiciones	Usuario registrado y autentificado como administrador
Postcondiciones	Datos del centro modificados en la base de datos
Escenario principal	1. Administrador accede a formulario de edición de datos del centro. 2. El sistema muestra el formulario con los datos de contacto e información del centro. 3. El usuario modifica los datos editables del formulario:

	• Nombre • Dirección • Código postal • Población • Provincia • Teléfonos • Email y pulsa sobre Guardar. 4. El sistema modifica con éxito los datos en la base de datos.
Escenario alternativo	3a. El usuario pulsa sobre cancelar. 4a. El sistema muestra la página principal. 3b. El usuario deja algún campo obligatorio en blanco y pulsa sobre Guardar. 4b. El sistema muestra mensaje indicando la obligatoriedad del campo.

CUA.4 Alta de Estancia	
Actor principal	Administrador
Objetivo	Registro de estancias que podrán ser reservadas
Precondiciones	Usuario registrado y autentificado como administrador
Postcondiciones	Estancia registrada en el sistema
Escenario principal	1. Administrador accede a la opción de alta de estancia nueva. 2. El sistema muestra el formulario de alta de estancia. 3. El usuario indica: • Código • Descripción • Observaciones • Estado y pulsa sobre Guardar. 4. El sistema guarda los datos con éxito en la base de datos.
Escenario alternativo	3a. El usuario indica un identificador ya existente y pulsa sobre Guardar. 4a. El sistema advierte que el identificador ya está en uso. 3b. El usuario no indica identificador y pulsa sobre Guardar. 4b. El sistema advierte que debe introducir los campos obligatorios.

CUA.5 Consulta de Estancia	
Actor principal	Administrador
Objetivo	Consulta de estancia
Precondiciones	Usuario registrado y autentificado como administrador

Postcondiciones	Datos de la estancia mostrada en pantalla
Escenario principal	1. Usuario administrador accede a la opción de consulta de estancias. 2. El sistema muestra las estancias registradas y opciones de filtrado y ordenado: • Código • Descripción • Estado 3. El usuario indica un criterio de filtrado por código y/o estado. 4. El sistema muestra el resultado de la consulta.
Escenario alternativo	2a. No existen estancias registradas y el sistema muestra el mensaje: No existen estancias registradas.

CUA.6 Anulación de Estancia	
Actor principal	Administrador
Objetivo	Anular la estancia.
Precondiciones	Usuario registrado y autentificado como administrador Estancia seleccionada en la consulta
Postcondiciones	Dato estado de la estancia asignada como anulada en base de datos
Escenario principal	1. El usuario administrador selecciona la estancia en la consulta de estancias y pulsa sobre Anular. 2. El sistema solicita confirmación: ¿Desea anular la estancia? 3. El usuario confirma pulsando sobre Si. 4. El sistema registra la estancia con estado anulado.
Escenario alternativo	3a. El usuario pulsa sobre No. 4a. El sistema no realiza ninguna acción.

CUA.7 Modificación de Estancia	
Actor principal	Administrador
Objetivo	Modificar datos de la estancia
Precondiciones	Usuario registrado y autentificado como administrador Estancia seleccionada en la consulta
Postcondiciones	Datos de la estancia editada y guardado los cambios en base de datos
Escenario principal	1. El usuario administrador selecciona la estancia en la consulta de estancias y pulsa sobre Editar. 2. El sistema muestra un cuadro de dialogo con la información editable de la estancia. 3. El usuario edita los campos del formulario: • Código

	• Descripción • Observaciones • Estado y pulsa sobre Guardar. 4. El sistema modifica con éxito los datos en la base de datos.
Escenario alternativo	3a. El usuario pulsa sobre cancelar. 4a. El sistema se sale del cuadro de dialogo sin cambiar datos. 3b. El usuario deja algún campo obligatorio en blanco y pulsa sobre Guardar. 4b. El sistema muestra mensaje indicando la obligatoriedad del campo.

CUA.8 Alta de Administrador	
Actor principal	Administrador
Objetivo	Registro de administrador
Precondiciones	Usuario registrado y autentificado como administrador
Postcondiciones	Usuario administrador añadido en el sistema
Escenario principal	1. Administrador accede a la opción de alta de administrador nuevo. 2. El sistema muestra el formulario de alta de administrador. 3. El administrador indica los datos del usuario: • Usuario • Clave • Nombre • Apellidos • Teléfonos • Email • Observaciones • Estado y pulsa sobre Guardar. 4. El sistema guarda los datos con éxito en la base de datos.
Escenario alternativo	3a. El usuario indica un identificador ya existente y pulsa sobre Guardar. 4a. El sistema advierte que el identificador ya está en uso. 3b. El usuario no indica identificador y pulsa sobre Guardar. 4b. El sistema advierte que debe introducir los campos obligatorios.

CUA.9 Consulta de Administrador	
Actor principal	Administrador
Objetivo	Consulta de administrador

Precondiciones	Usuario registrado y autentificado como administrador
Postcondiciones	Datos del administrador mostrador en pantalla
Escenario principal	1. Usuario administrador accede a la opción de consulta de administradores. 2. El sistema muestra los administradores registradas y opciones de filtrado por: • Apellidos • Nombre • Teléfono • Estado 3. El usuario indica un criterio de filtrado. 4. El sistema muestra el resultado de la consulta. 5. El usuario pulsa dos veces sobre el registro del administrador o pulsa sobre Editar el registro seleccionado. 6. El sistema muestra un cuadro de dialogo con los datos del administrador.
Escenario alternativo	

CUA.10 Eliminar Administrador	
Actor principal	Administrador
Objetivo	Eliminar el administrador del sistema
Precondiciones	Usuario registrado y autentificado como administrador Administrador seleccionado en la consulta
Postcondiciones	Administrador eliminado del sistema.
Escenario principal	1. El usuario administrador selecciona un administrador diferente al usuario actual en la consulta de administradores y pulsa sobre Eliminar. 2. El sistema solicita confirmación: ¿Desea eliminar el administrador? 3. El usuario confirma pulsando sobre Si. 4. El sistema elimina el administrador de la base de datos
Escenario alternativo	1a. El usuario selecciona su propio usuario 4a. El sistema advierte que no puede eliminarse su propio usuario y no realiza ninguna acción. 3b. El usuario pulsa sobre No. 4b. El sistema no realiza ninguna acción.

CUA.11 Modificación de Administrador	
Actor principal	Administrador
Objetivo	Modificar datos del administrador
Precondiciones	Usuario registrado y autentificado como administrador Administrador seleccionado en la consulta
Postcondiciones	Datos del administrador editados y guardado los cambios en base de datos

Escenario principal	1. El usuario administrador selecciona el registro en la consulta de administradores y pulsa sobre Editar. 2. El sistema muestra un cuadro de dialogo con la información editable del administrador. 3. El usuario edita los campos: • Usuario • Clave • Nombre • Apellidos • Teléfonos • Email • Observaciones • Estado y pulsa sobre Guardar. 4. El sistema modifica con éxito los datos en la base de datos.
Escenario alternativo	3a. El usuario pulsa sobre cancelar. 4a. El sistema se sale del cuadro de dialogo sin cambiar datos. 3b. El usuario deja algún campo obligatorio en blanco y pulsa sobre Guardar. 4b. El sistema muestra mensaje indicando la obligatoriedad del campo.

Clientes

CUC.1 Alta de Cliente	
Actor principal	Administrador, usuario no registrado
Objetivo	Dar de alta un nuevo cliente
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Datos del nuevo cliente guardado en la base de datos
Escenario principal	1. El usuario accede a la opción de nuevo cliente 2. El sistema le muestra un formulario para el alta de nuevo cliente. 3. El usuario introduce los datos: • Usuario • Clave • Identificador • Nombre • Apellidos • Teléfonos • Email • Dirección • Código postal • Población • Provincia • Nombre del veterinario de contacto • Teléfono del veterinario • Estado • Observaciones y pulsa sobre Guardar. 4. El sistema guarda los datos con éxito.
Escenario alternativo	1a. El usuario accede desde la opción Registrase de la pantalla de acceso a la aplicación. 3b. El usuario introduce un nombre de usuario existente. 4b. El sistema advierte que el nombre del usuario está en uso. 3c. El usuario no introduce alguno de los campos obligatorios. 4c. El sistema advierte que debe introducir los campos obligatorios.

CUC.2 Consulta de Cliente	
Actor principal	Administrador, cliente
Objetivo	Consultar los datos de un cliente
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Consulta de datos del cliente
Escenario principal	1. El usuario accede a la opción de consulta de clientes

	2. El sistema le muestra un formulario para la consulta de clientes con campos de filtrado y ordenación: • Apellidos • Nombre • Teléfono • Estado 3. El usuario indica un criterio de filtrado. 4. El sistema muestra el resultado de la consulta. 5. El usuario pulsa dos veces sobre el registro consultado o pulsa sobre botón Editar 6. El sistema muestra un formulario con los datos del cliente.
Escenario alternativo	2a. El sistema muestra únicamente los datos del cliente si el usuario es el cliente

CUC.3 Eliminación de Cliente	
Actor principal	Administrador
Objetivo	Eliminación de datos del cliente ajustándose a ley protección de datos.
Precondiciones	Usuario registrado y autentificado como administrador
Postcondiciones	Cliente, sus mascotas e historial de reservas eliminados de la base de datos
Escenario principal	1. El usuario accede a la opción de consulta de clientes 2. El sistema le muestra un formulario para la consulta de clientes con campos de filtrado 3. El usuario indica un criterio de filtrado. 4. El sistema muestra el resultado de la consulta. 5. El usuario selecciona el registro y pulsa sobre Eliminar. 6. El sistema solicita confirmación: ¿Desea eliminar el Cliente? 7. El usuario confirma pulsando sobre Si. 8. El sistema elimina el administrador de la base de datos
Escenario alternativo	7a. El usuario pulsa No 8a. El sistema no realiza cambios

CUC.4 Anulación de Cliente	
Actor principal	Administrador
Objetivo	Anulación del cliente para evitar acceso al sistema
Precondiciones	Usuario registrado y autentificado como administrador
Postcondiciones	Cliente editado y guardado con éxito
Escenario principal	1. El usuario accede a la opción de consulta de clientes

	2. El sistema le muestra un formulario para la consulta de clientes con campos de filtrado 3. El usuario indica un criterio de filtrado. 4. El sistema muestra el resultado de la consulta. 5. El usuario pulsa sobre botón Editar 6. El sistema muestra un formulario con los datos del cliente. 7. El usuario edita el campo estado asignándolo como desactivado y pulsa sobre Guardar. 8. El sistema realiza los cambios con éxito.
Escenario alternativo	7a. El usuario pulsa sobre Cancelar. 8a. El sistema no realiza cambios en los datos del cliente y muestra la pantalla de consulta.

CUC.5 Modificación de Cliente	
Actor principal	Administrador, cliente
Objetivo	Edición de datos del cliente
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Cliente editado y guardado con éxito
Escenario principal	1. El usuario accede a la opción de consulta de clientes 2. El sistema le muestra un formulario para la consulta de clientes con campos de filtrado 3. El usuario indica un criterio de filtrado. 4. El sistema muestra el resultado de la consulta. 5. El usuario pulsa sobre botón Editar 6. El sistema muestra un formulario con los datos del cliente. • Usuario • Clave • Identificador • Nombre • Apellidos • Teléfonos • Email • Dirección • Código postal • Población • Provincia • Estado • Nombre del veterinario de contacto • Teléfono del veterinario • Observaciones 7. El usuario edita los campos editables y pulsa sobre Guardar. 8. El sistema realiza los cambios con éxito.
Escenario alternativo	7a. El usuario pulsa sobre Cancelar.

	8a. El sistema no realiza cambios en los datos del cliente y muestra la pantalla de consulta.

CUC.6 Alta de Mascota	
Actor principal	Administrador, cliente
Objetivo	Dar de alta una nueva mascota
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Datos del nueva mascota de cliente guardados en la base de datos
Escenario principal	1. El usuario accede a la opción de nueva mascota 2. El sistema le muestra un formulario para el alta de nueva mascota. 3. El usuario introduce los datos: • Nombre • Raza • Sexo • Fecha de nacimiento • Identificador microchip • Localización del microchip • Estado • Observaciones y pulsa sobre Guardar. 4. El sistema guarda los datos con éxito.
Escenario alternativo	3a. El usuario no introduce alguno de los campos obligatorios. 4a. El sistema advierte que debe introducir los campos obligatorios.

CUC.7 Consulta de Mascota	
Actor principal	Administrador, cliente
Objetivo	Consultar los datos de una mascota
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Consulta de datos de la mascota
Escenario principal	1. El usuario accede a la opción de consulta de mascotas 2. El sistema le muestra un formulario para la consulta de mascotas con campos de filtrado 3. El usuario indica un criterio de filtrado. 4. El sistema muestra el resultado de la consulta. 5. El usuario pulsa dos veces sobre el registro consultado o pulsa sobre botón Ver 6. El sistema muestra un formulario con los datos de la mascota.
Escenario alternativo	2a. El sistema muestra únicamente los datos de sus macotas si el usuario es el cliente

CUC.8 Anulación de Mascota	
Actor principal	Administrador, cliente
Objetivo	Dar de baja una mascota
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Datos del nuevo cliente guardado en la base de datos
Escenario principal	1. El usuario accede a la opción de consulta de mascotas 2. El sistema le muestra un formulario para la consulta de mascotas con campos de filtrado 3. El usuario indica un criterio de filtrado. 4. El sistema muestra el resultado de la consulta. 5. El usuario pulsa sobre botón Editar 6. El sistema muestra un formulario con los datos de la mascota. • Nombre • Raza • Sexo • Fecha de nacimiento • Identificador microchip • Localización del microchip • Estado • Observaciones 7. El usuario edita el campo estado asignándolo como desactivado y pulsa sobre Guardar. 8. El sistema realiza los cambios con éxito.
Escenario alternativo	7a. El usuario pulsa sobre Cancelar. 8a. El sistema no realiza cambios en los datos y muestra la pantalla de consulta.

CUC.9 Modificación de Mascota	
Actor principal	Administrador, cliente
Objetivo	Modificar datos de mascota
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Datos editados de mascota guardados en la base de datos
Escenario principal	1. El usuario accede a la opción de consulta de mascotas 2. El sistema le muestra un formulario para la consulta de mascotas con campos de filtrado 3. El usuario indica un criterio de filtrado. 4. El sistema muestra el resultado de la consulta. 5. El usuario pulsa sobre botón Editar 6. El sistema muestra un formulario con los datos de la mascota. 7. El usuario edita los campos editables y pulsa sobre Guardar. 8. El sistema realiza los cambios con éxito.

Escenario alternativo	7a. El usuario pulsa sobre Cancelar. 8a. El sistema no realiza cambios en los datos del cliente y muestra la pantalla de consulta.

Reservas

CUB.1 Consultar disponibilidad entre fechas	
Actor principal	Administrador, cliente
Objetivo	Obtener confirmación de disponibilidad entre una fecha de inicio y fecha de fin
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Información sobre la disponibilidad de estancia entre unas fechas
Escenario principal	1. El usuario accede a la opción de consulta de disponibilidad entre fechas 2. El sistema le muestra un formulario de consulta de disponibilidad 3. El usuario introduce la fecha de inicio y la de salida y pulsa sobre Consultar 4. El sistema consulta disponibilidad de estancias para estas fechas
Escenario alternativo	3a. El usuario indica una fecha de salida anterior a la de entrada. 4a. El sistema indica el error advirtiendo que la fecha de salida no puede ser anterior a la de entrada

CUB.2 Alta de reserva entre fechas	
Actor principal	Administrador, cliente
Objetivo	Registrar una reserva de estancia para la mascota de un cliente
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Datos de reserva guardados en la base de datos
Escenario principal	1. El usuario accede a la opción de consulta de disponibilidad entre fechas 2. El sistema le muestra un formulario de consulta de disponibilidad 3. El usuario introduce la fecha de inicio y la de salida y pulsa sobre Consultar 4. El sistema consulta disponibilidad de estancias para estas fechas 5. El usuario confirma la reserva indicando su mascota, observaciones y pulsando Reservar 6. El sistema registra la reserva en la base de datos asignando una estancia disponible a la mascota entre las fechas indicadas.

Escenario alternativo	3a. El usuario indica una fecha de salida anterior a la de entrada. 4a. El sistema indica el error advirtiendo que la fecha de salida no puede ser anterior a la de entrada

CUB.3 Consulta de reserva	
Actor principal	Administrador, cliente
Objetivo	Consultar fecha y estancia de una reserva
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Datos de la reserva consultados
Escenario principal	1. El usuario accede a la opción de consulta de reservas. 2. El sistema le muestra una consulta de reservas donde puede introducir criterios. Un criterio activado es la opción de reservas pendientes. En caso de ser un cliente ya tendrá establecido este criterio 3. El usuario puede establecer criterios de consulta 4. El sistema devuelve los resultados mostrando las reservas que cumplan dicho criterio.
Escenario alternativo	

CUB.4 Eliminar reserva	
Actor principal	Administrador, cliente
Objetivo	Eliminar una reserva de un cliente
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Reserva eliminada de la base de datos
Escenario principal	1. El usuario accede a la opción de consulta de reservas. 2. El sistema le muestra una consulta de reservas donde puede introducir criterios. Un criterio activado es la opción de reservas pendientes. En caso de ser un cliente ya tendrá establecido este criterio 3. El usuario puede establecer criterios de consulta 4. El sistema devuelve los resultados mostrando las reservas que cumplan dicho criterio. 5. El usuario pulsa sobre Eliminar. 6. El sistema pide confirmación para eliminar la reserva. 7. El usuario pulsa Sobre Si para confirmar la eliminación. 8. El sistema asigna elimina la reserva con éxito.
Escenario alternativo	7a. El usuario pulsa sobre No para cancelar eliminación

	8a. El sistema no realiza ninguna acción.

CUB.5 Consulta de historial de reservas por cliente	
Actor principal	Administrador, cliente
Objetivo	Consulta de historial de reservas de un cliente
Precondiciones	Usuario registrado y autentificado
Postcondiciones	Información de las reservas de un cliente
Escenario principal	1. El usuario accede a la consulta de clientes 2. El sistema le muestra la consulta de clientes, si el usuario es cliente únicamente verá sus datos 3. El usuario accede a la consulta del cliente mediante la opción Ver 4. El sistema le muestra la pantalla de consulta de los datos del cliente con la opción de consulta de las reservas realizadas y opciones de filtrado para realizas consultas 5. El usuario establece criterios de consulta 6. El sistema le muestra el resultado del filtrado obtenido a través de los criterios establecidos.
Escenario alternativo	

2.3 Modelo de pantallas

2.3.1 Diagrama de navegación

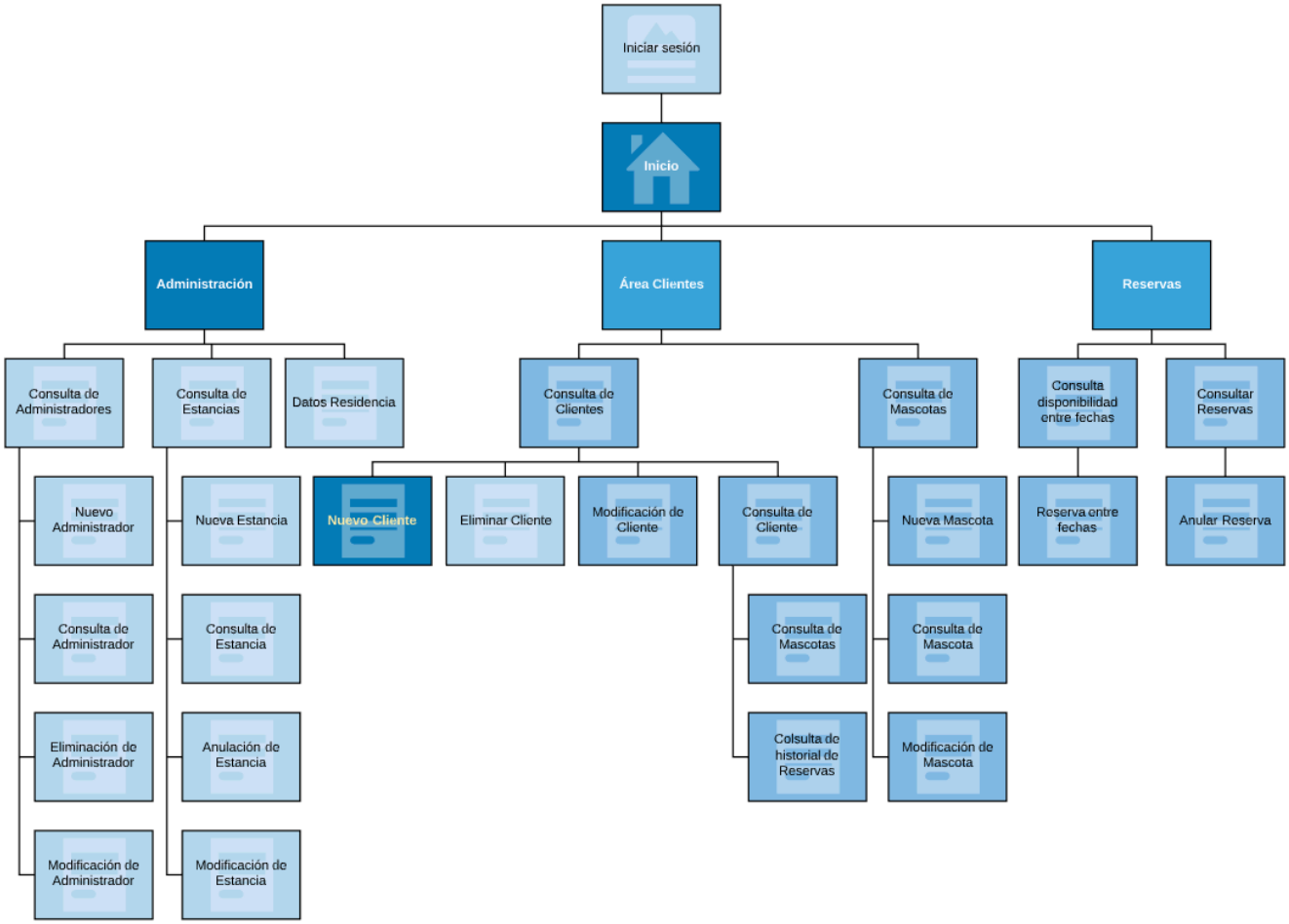


Ilustración 10 Diagrama de navegación

A través de este diagrama podemos observar la distribución de las pantallas de la aplicación. Este será el mapa de navegación donde podemos encontrar cubiertos los objetivos y el recorrido de los casos de usos. Sin embargo, esta ilustración representa el perfil completo, recordemos que los usuarios con rol Cliente solo podrán acceder a las pantallas del Área de Clientes y Reservas. Los usuarios no registrados accederían únicamente a la pantalla de Nuevo Cliente para registrarse en el sistema.

En el siguiente apartado se mostrarán prototipos para estas pantallas, serán el complemento para seguir los pasos descritos en los casos de uso. A lo largo del diseño intentaremos añadir reutilización de componentes y formularios Lo que supondrá un ahorro de esfuerzos para reducir el tiempo de implementación. Por tanto, ciertos casos de uso compartirán pantalla, o bien, se resolverán con cuadros de diálogo en algún formulario. Por ejemplo, se usará la pantalla de consulta de datos para incorporar la funcionalidad de eliminar, resolviendo en un cuadro de dialogo emergente, la advertencia de la acción y para solicitar confirmación.

2.3.2 Prototipos de pantallas

Acceso y Administración

PALG. Iniciar sesión
Caso de uso: CUA.1

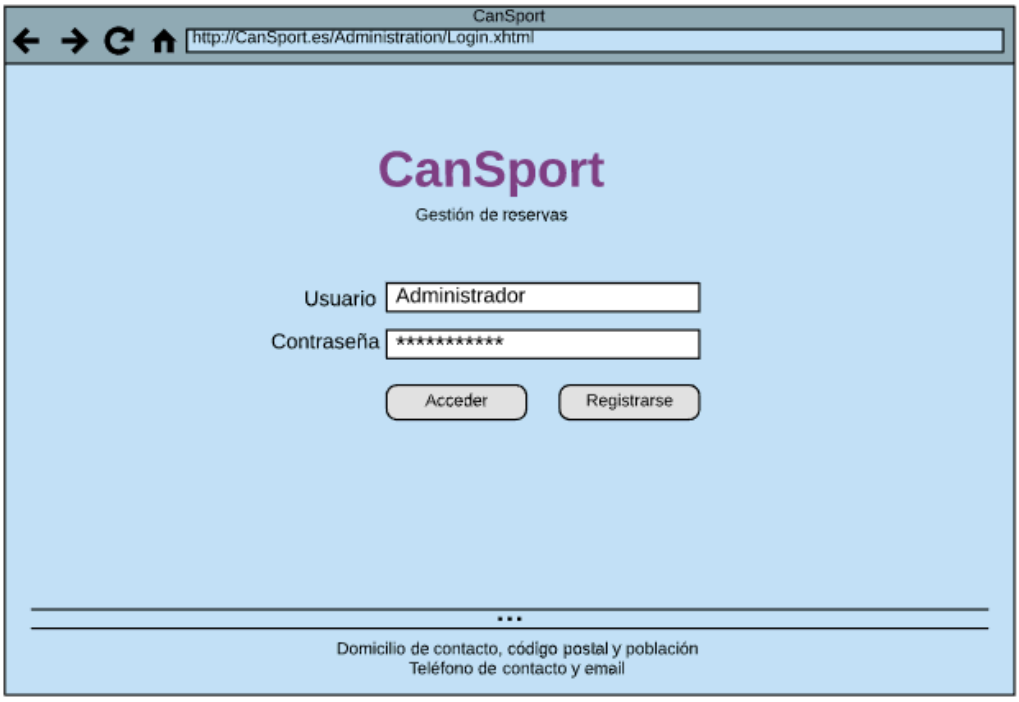


Ilustración 11 Prototipos de pantalla. Login

Este formulario es el que permite acceso al sistema o acceder al registro de nuevos clientes.

PA00. Plantilla general

Ilustración 12 Prototipo de cabecera de pantallas

Utilizaremos una plantilla general que formará parte del cabecero de todos las pantallas (excepto la de acceso al sistema). En ella encontraremos el nombre de la residencia y las opciones de menú, que permitirán la navegación por la aplicación y la opción de salir del sistema. Recordemos que las opciones de menú se habilitarán según el rol del usuario. Desplegadas todas, serían como se muestra en el siguiente prototipo:

Ilustración 13 Prototipo de cabecera de pantallas con menú completamente desplegado

En los siguientes prototipos se añadirá la plantilla superior con la opción de menú que se ha seleccionado para llegar a la descrita.

PA01. Pantalla de inicio

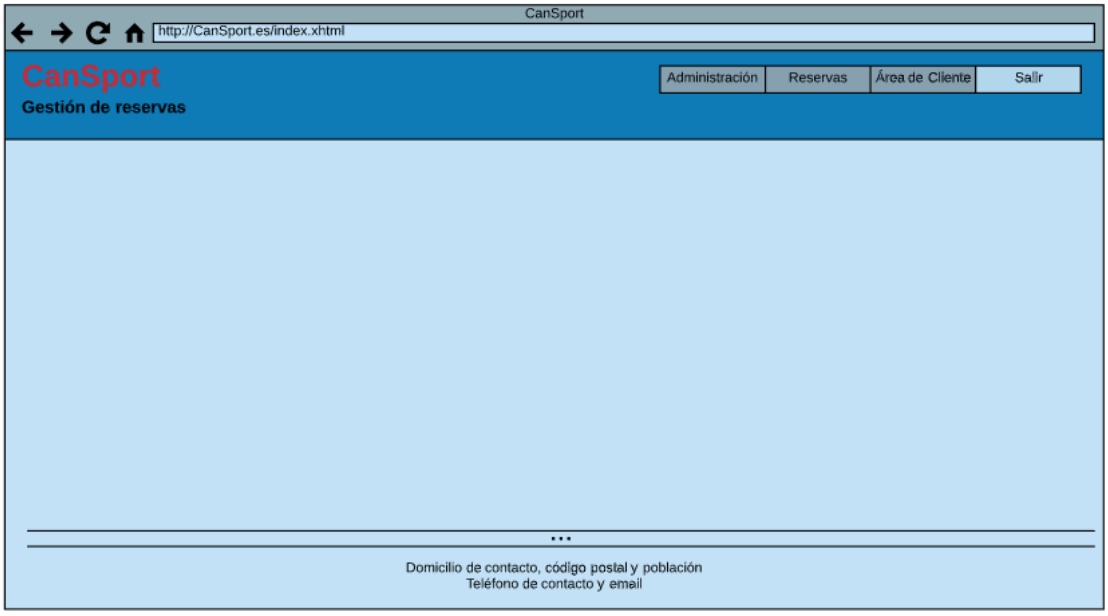


Ilustración 14 Prototipo de plantilla de pantallas

PA02. Consulta de Administradores

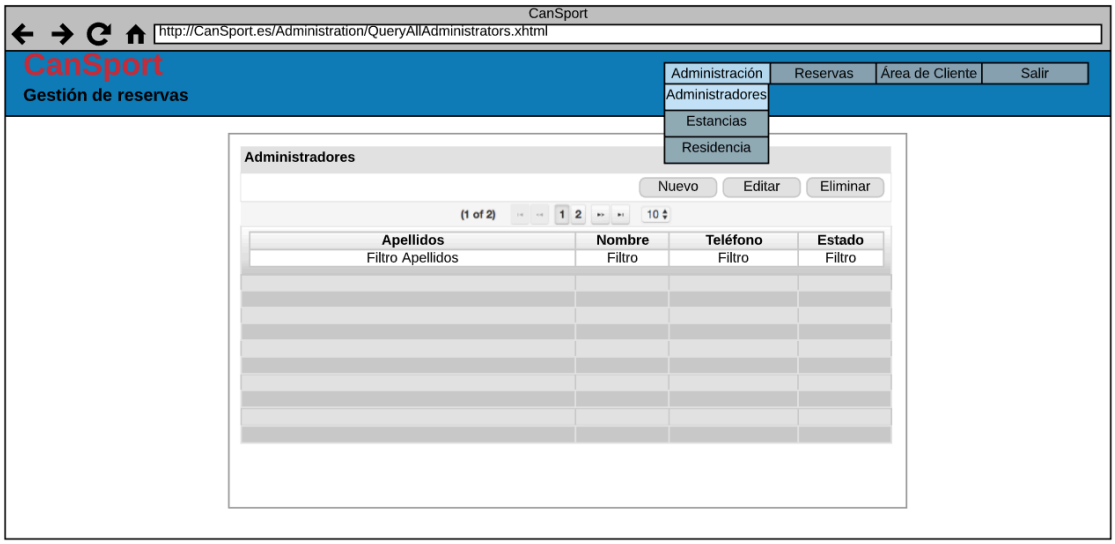


Ilustración 15 Prototipos de pantalla. Consulta de administradores

Desde esta consulta de Administradores podremos ordenar, filtrar, repaginar por un número de elementos y tendremos acceso a las opciones de dar de alta, editar y eliminar un registro de Administrador. Esta operatoria se repetirá forma similar en cada uno de los formularios CRUD que se mantienen en el sistema.

PA02. Eliminar Administrador
Caso de uso: CUA.10

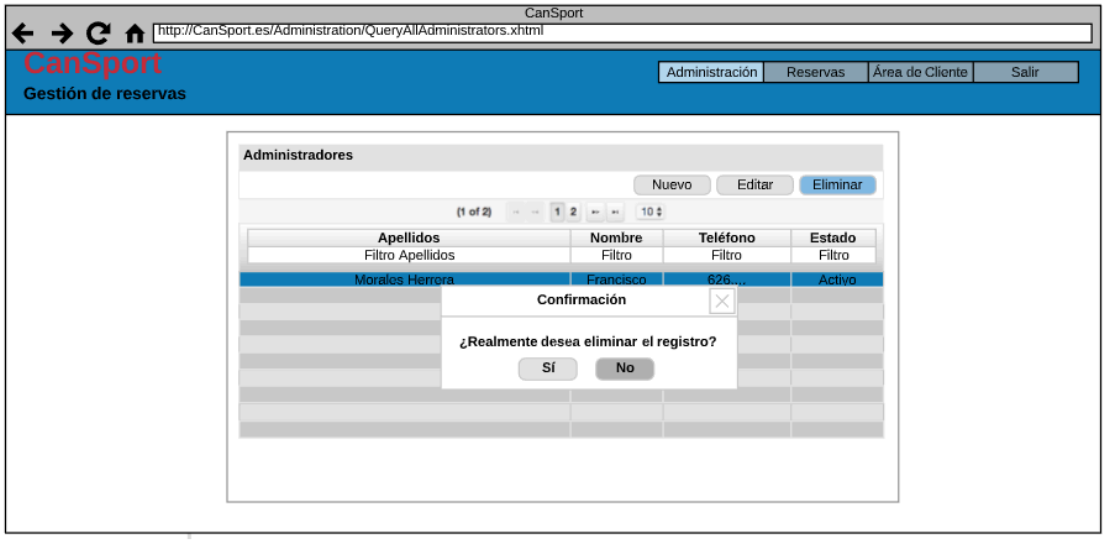


Ilustración 16 Prototipos de pantalla. Elimando administrador

En el propio formulario de consulta de todos los Administradores implementamos la opción de eliminación de Administrador seleccionado.

PA03. Consulta/Modificar/Alta Administrador
Casos de uso: CUA.8, CUA.9, CUA.11

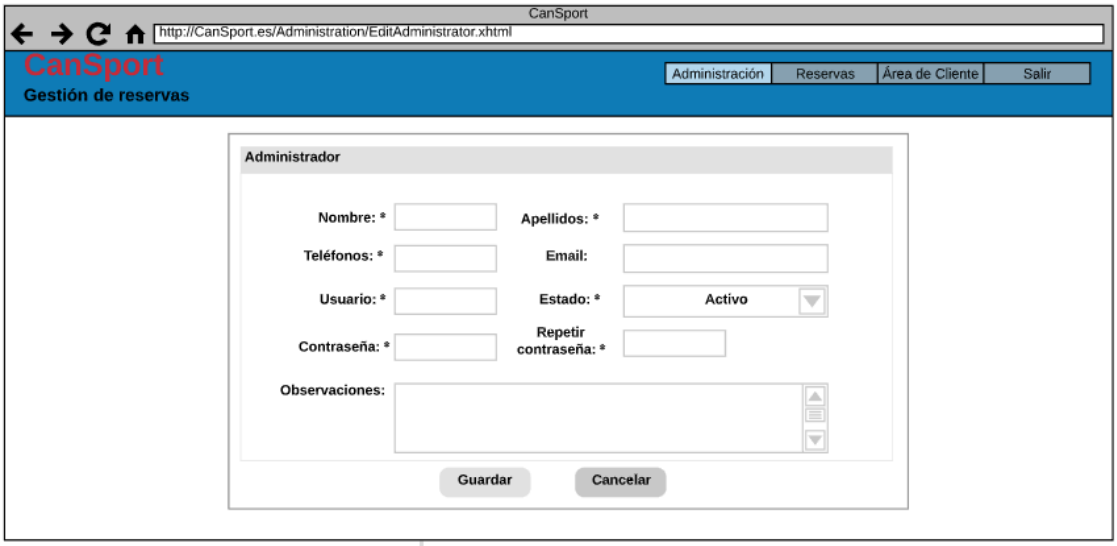


Ilustración 17 Prototipos de pantalla. Edición de administrador

En este formulario podremos realizar el alta de un nuevo Administrador o su edición. En el caso de alta nueva, la edición de los datos se realiza sobre un objeto inicializado, en caso de modificación, el objeto habrá recuperado los datos del registro seleccionado mostrándose y permitiendo la edición.

Esta reutilización del componente del modelo vista facilita la implementación y simplifica su mantenimiento. En el caso de incorporarse nuevos atributos a la

clase *Administrator* simplemente se añadirían al formulario de alta/consulta/edición.

PA04. Consulta de Estancias

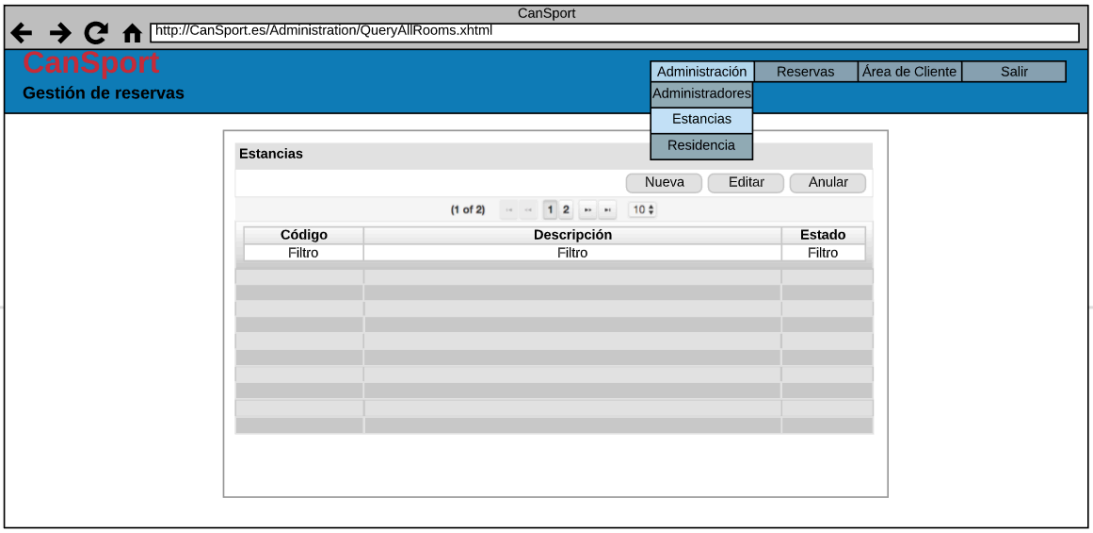


Ilustración 18 Prototipos de pantalla. Consulta de estancias

Como en el resto de formularios de consulta de entidades, incorpora botones para ofrecer la funcionalidad de añadir nuevo, editar y eliminar. La operatoria del formulario permitirá filtrar consulta, ordenar por campos y paginar el resultado.

PA04. Anular Estancia
Caso de uso: CUA.6

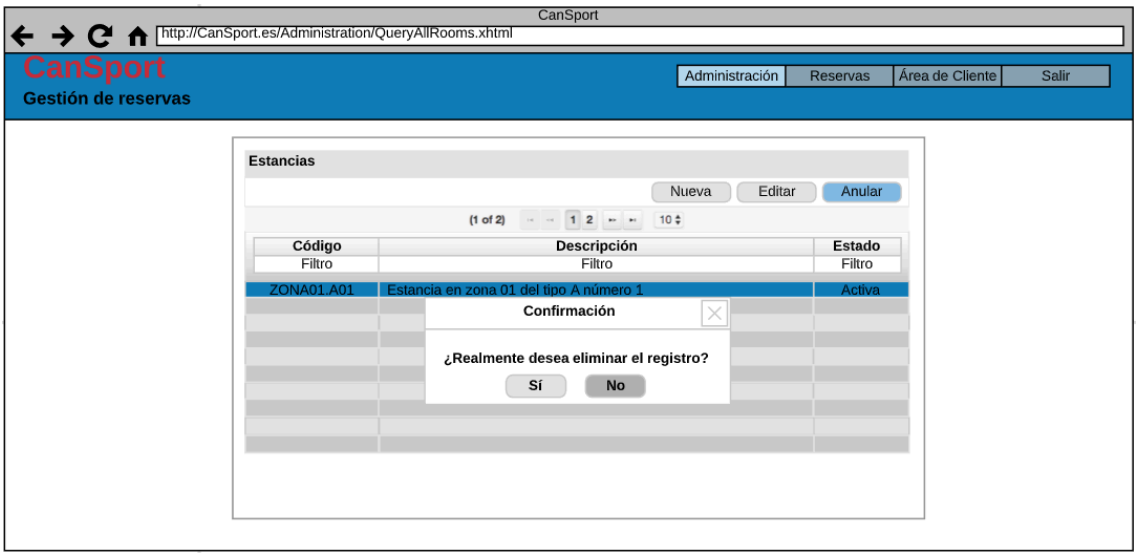


Ilustración 19 Prototipos de pantalla. Eliminando estancia

Observamos la misma operatoria descrita en prototipo similar anterior.

PA05. Consulta/Modificar/Alta Estancia
Casos de uso: CUA.4, CUA.5 y CUA.7

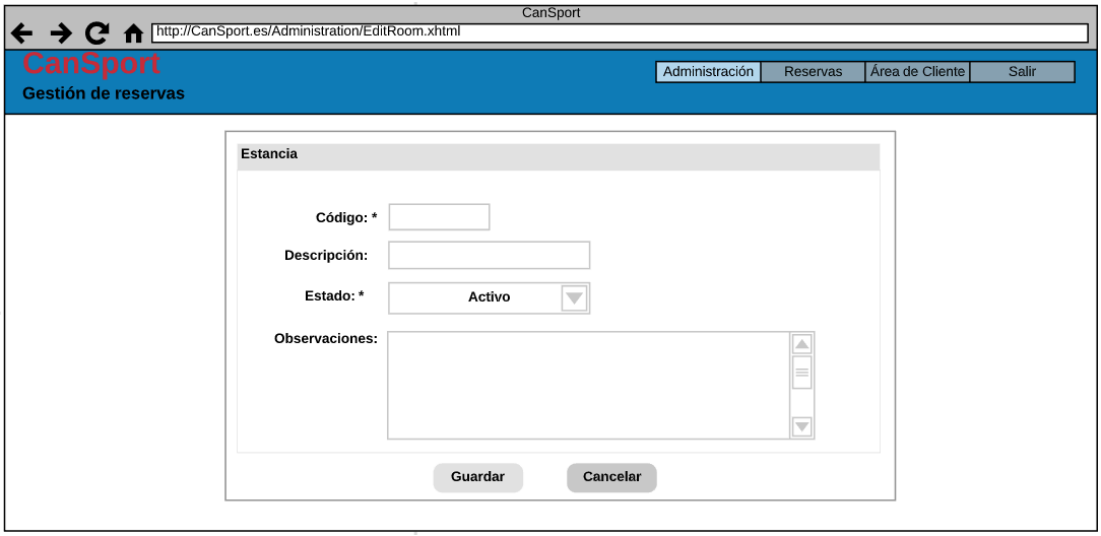


Ilustración 20 Prototipos de pantalla. Edición de estancia

Permitirá crear nuevas estancias o la edición de un registro seleccionado en la página de consulta.

PA06. Datos de Residencia
Caso de uso: CUA.3

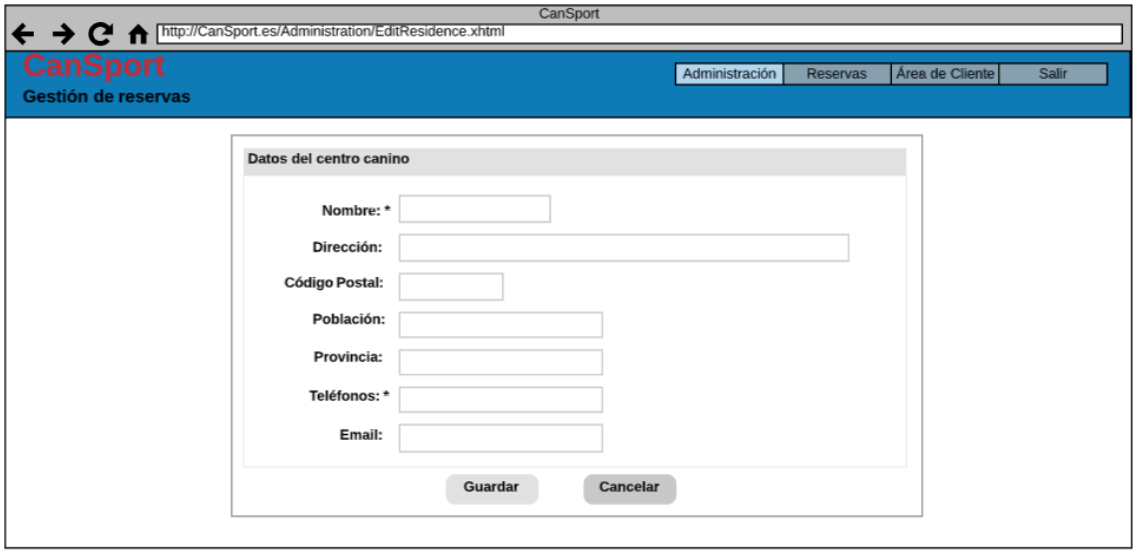


Ilustración 21 Prototipos de pantalla. Edición de datos residencia

Este formulario mantiene los datos que guardamos en la base de datos de contacto de la residencia canina y que se mostraran en la página de inicio y página de acceso al sistema o al registro de clientes.

Clientes

PC01. Nuevo Cliente/Anulación/Edición de Cliente
Casos de uso: CUC.1, CUC.4, CUC.5

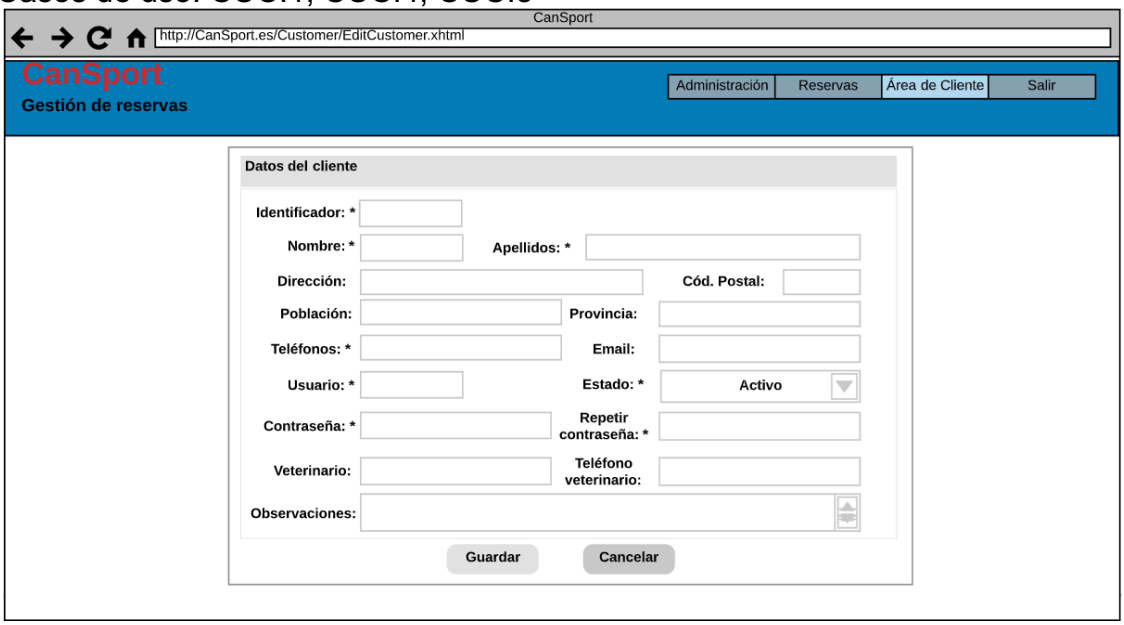


Ilustración 22 Prototipos de pantalla. Edición de cliente

PC02. Consulta de Clientes

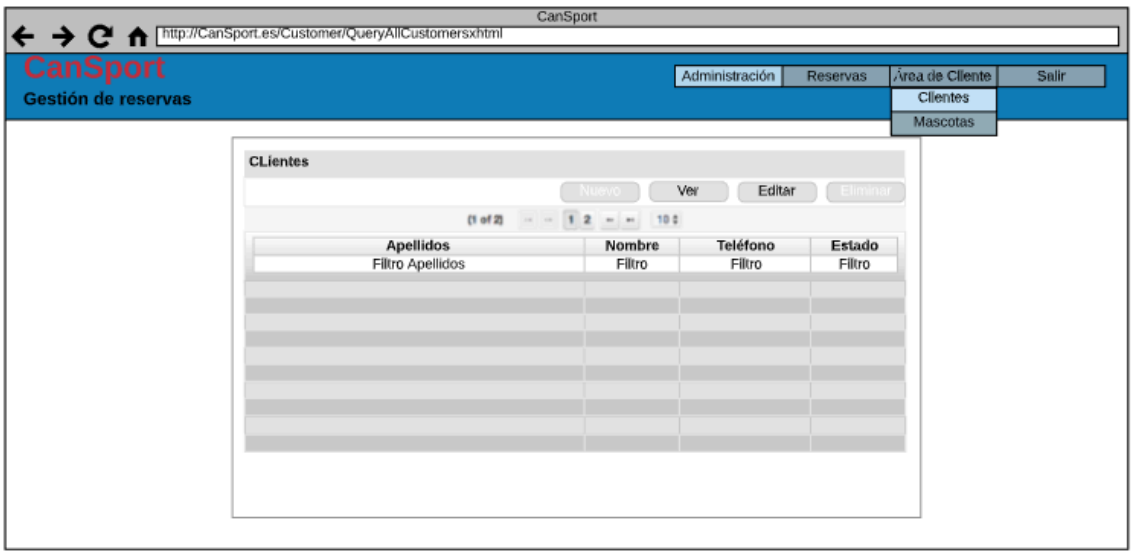


Ilustración 23 Prototipos de pantalla. Consulta de clientes

Este formulario tiene una conducta diferente según el rol que tiene acceso a él. Un Administrador tendrá disponible los botones de Nuevo y Eliminar y podrá consultar todos los clientes registrados y consultar con diferentes criterios de consulta. Un cliente solo verá su propio registro y dispondrá únicamente del botón Editar.

PC02. Eliminación de Cliente
Caso de uso: CUC.3

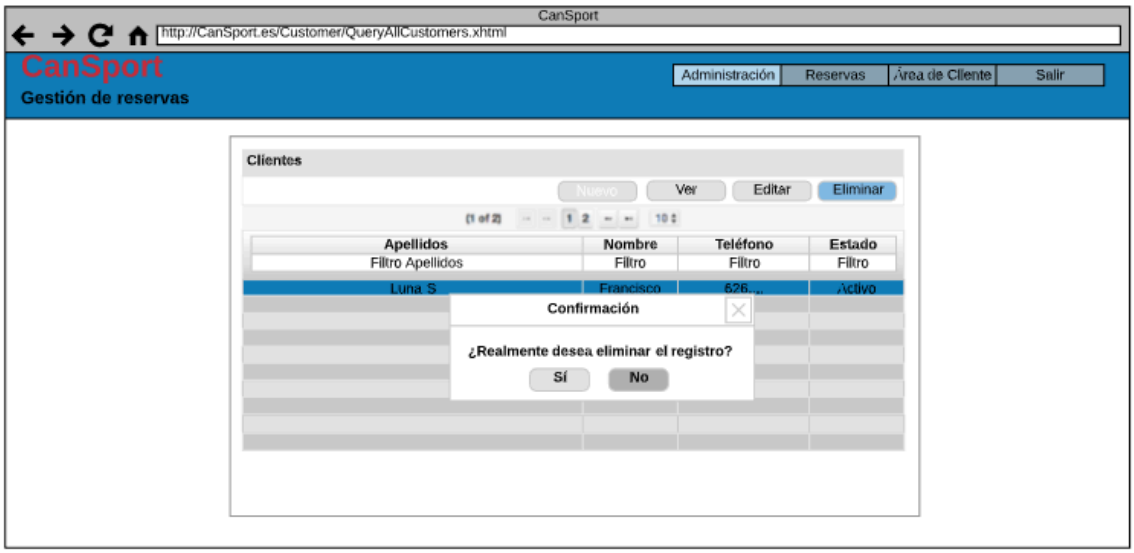


Ilustración 24 Prototipos de pantalla. Eliminando cliente

Opción únicamente disponible para un usuario Administrador, eliminará los datos del cliente y al realizarlo, eliminará también sus mascotas e historial de reservas.

PC03. Consulta de Cliente/Consulta de historial de reservas
Casos de uso: CUC.2, CUB.3, CUB.5

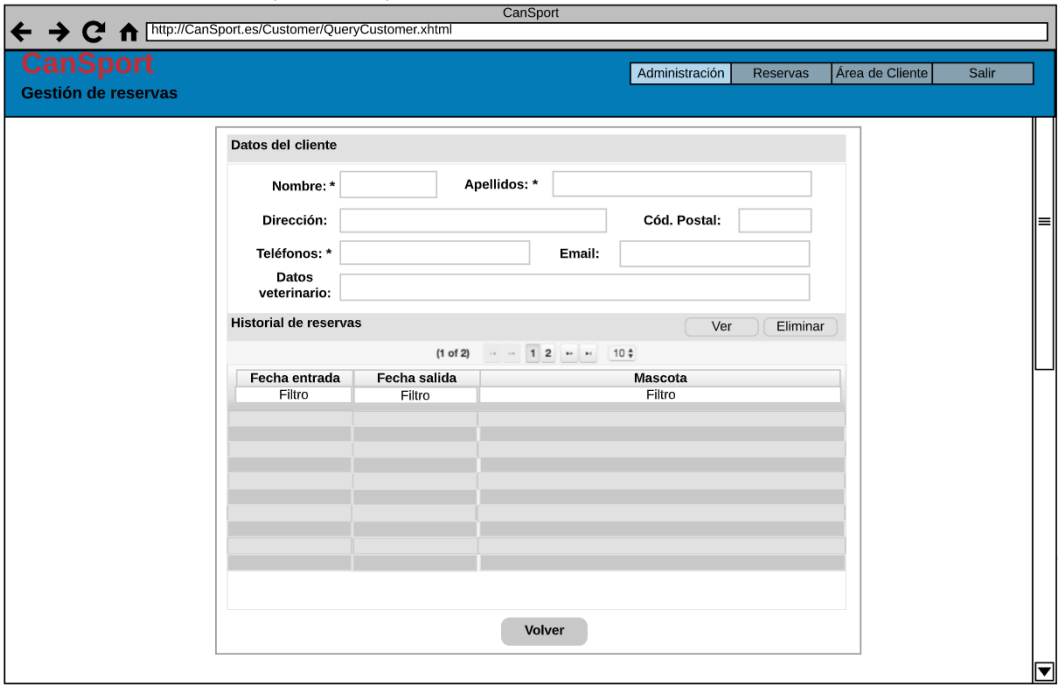


Ilustración 25 Prototipos de pantalla. Consulta de reservas de cliente

En este formulario se muestran algunos datos relevantes del cliente y un panel para la consulta del historial y detalles de las reservas, consulta reutilizada y accesible desde el menú de reservas. Esta vista oculta en la consulta el campo de cliente y lo fija como criterio de búsqueda al venir de un registro de cliente seleccionado.

PC04. Consulta de Mascotas

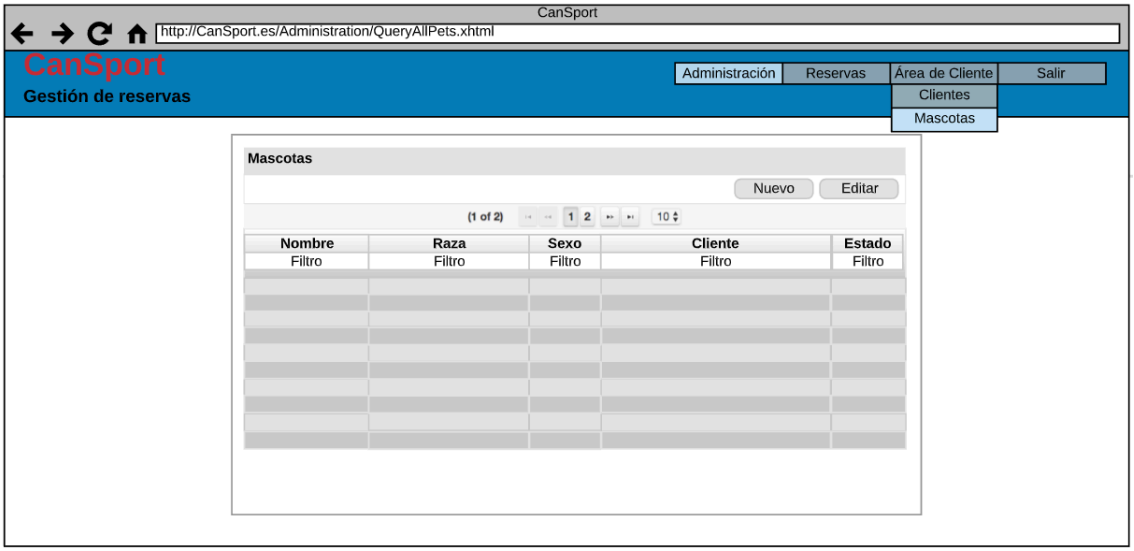


Ilustración 26 Prototipos de pantalla. Consulta de mascotas

Si el usuario es administrador podrá buscar por cualquier criterio obteniendo todos los resultados. Si el usuario es cliente, únicamente podrá consultar sus mascotas.

PC05. Consulta/Modificación/Nueva Mascota
Casos de uso: CUC.6, CUC.7, CUC.8, CUC.9

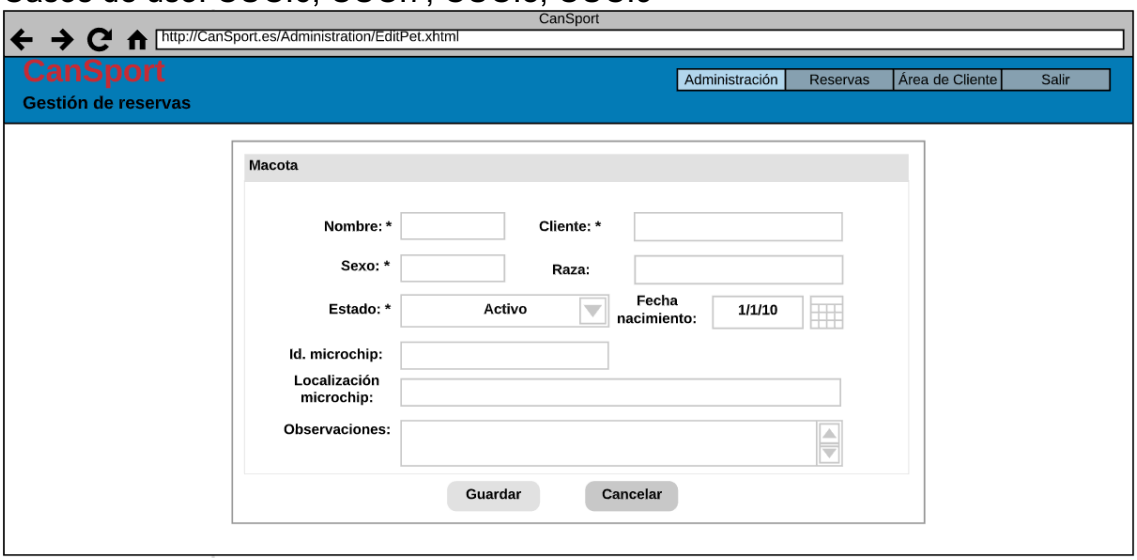


Ilustración 27 Prototipos de pantalla. Edición de mascota

Este formulario, en la misma operatoria que el resto de formularios CRUD, se reutiliza tanto para editar como para añadir nueva mascota, según la opción elegida. Si el usuario es cliente, no podrá cambiar ese dato. Por tanto, añadirá o editará sus propias mascotas en el sistema. Un usuario administrador deberá indicar el cliente al que asignarla.

Reservas

PB01. Consulta de disponibilidad
Caso de uso: CUB.1

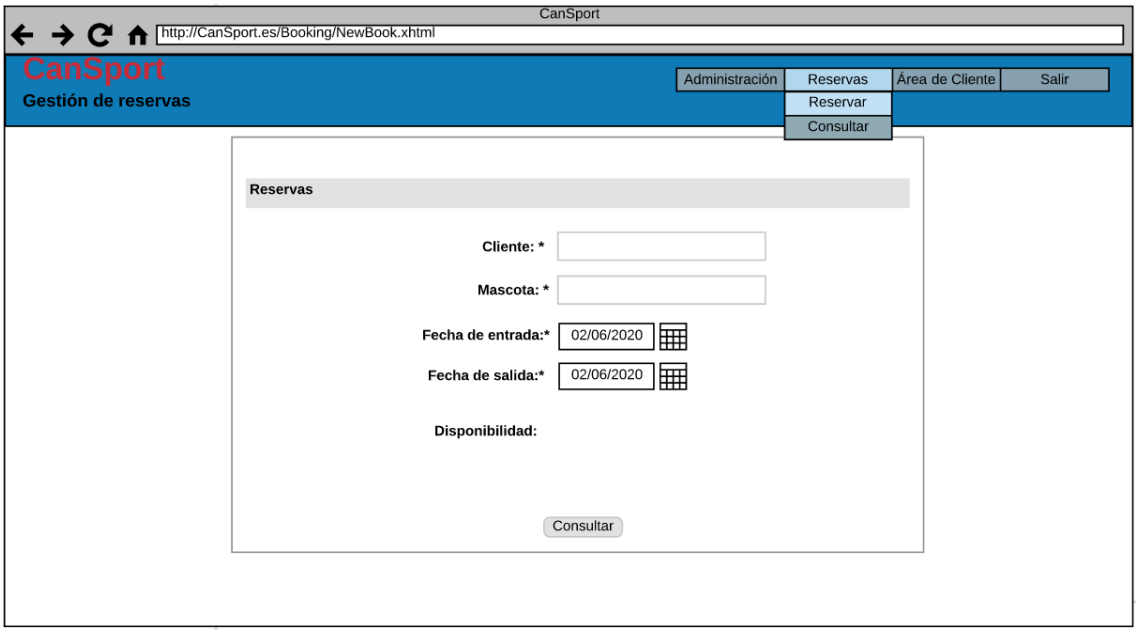


Ilustración 28 Prototipos de pantalla. Alta reserva. disponibilidad

Como precondición para reservar es la comprobación de disponibilidad. Se indica cliente y mascota para facilitar la acción.

PB01. Reserva
Caso de uso: CUB.2

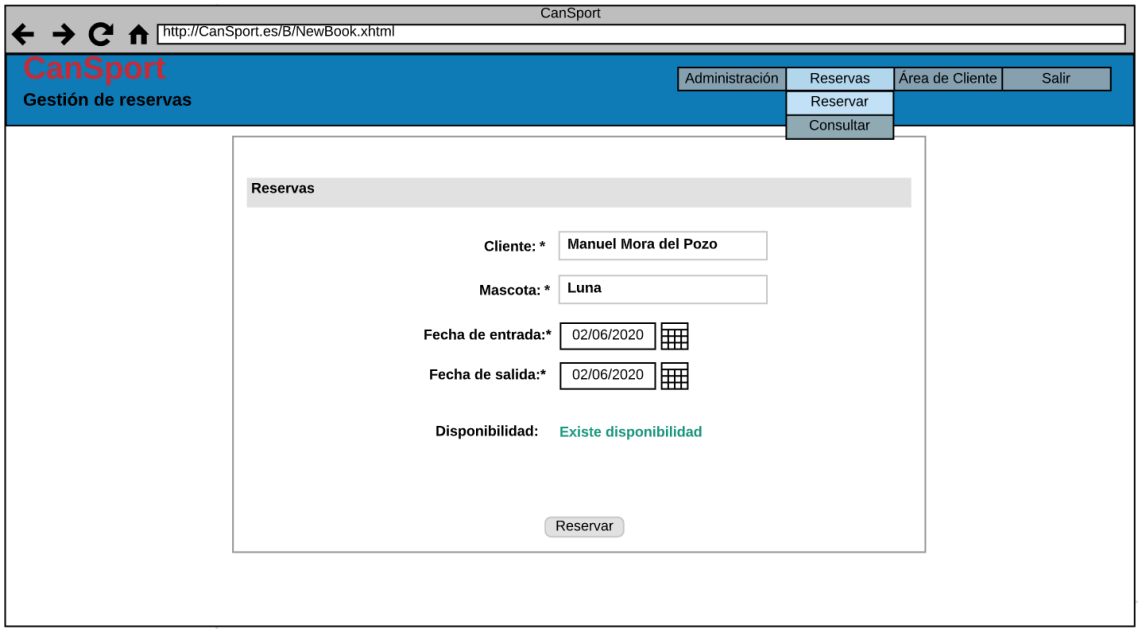


Ilustración 29 Prototipos de pantalla. Alta reserva

La reserva puede confirmarse una vez se ha comprobado la disponibilidad, pulsando sobre el botón Reservar que aparece ahora.

PB02. Consulta de Reservas
Caso de uso: CUB.3, CUB.5

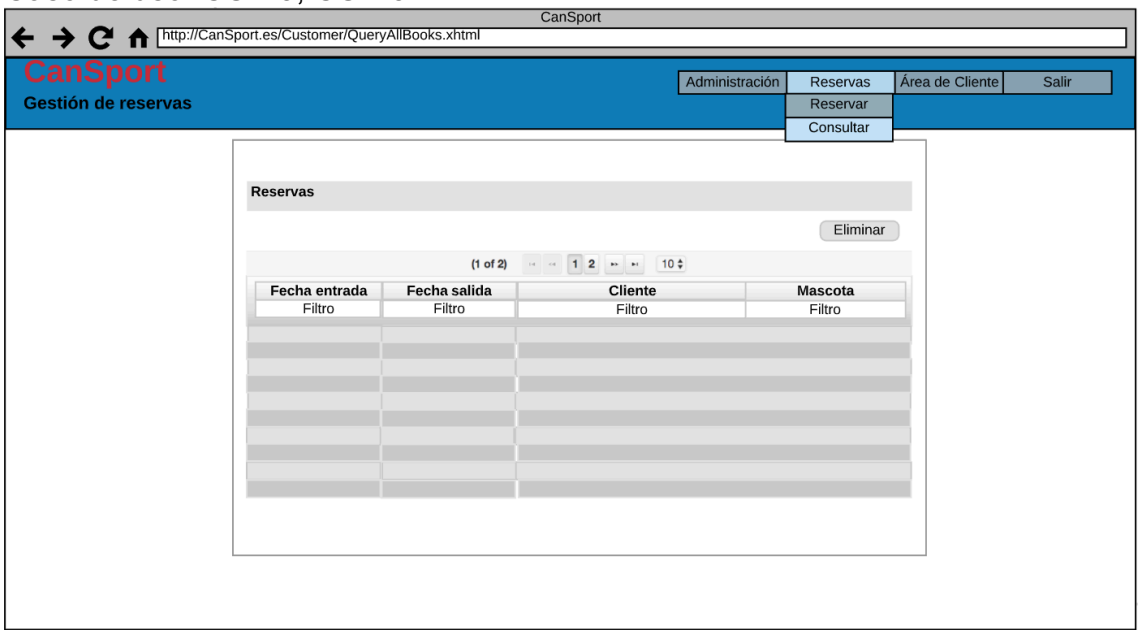


Ilustración 30 Prototipos de pantalla. Consulta de reservas

Esta consulta de reservas ya la vimos integrada en la consulta del cliente. Se reutilizaba para proporcionar el historial de reservas con el criterio del cliente establecido. En esta pantalla puede indicarlo si es administrador, en caso contrario, el cliente consultará únicamente sus movimientos.

PB02. Eliminar Reserva
Caso de uso: CUB.4

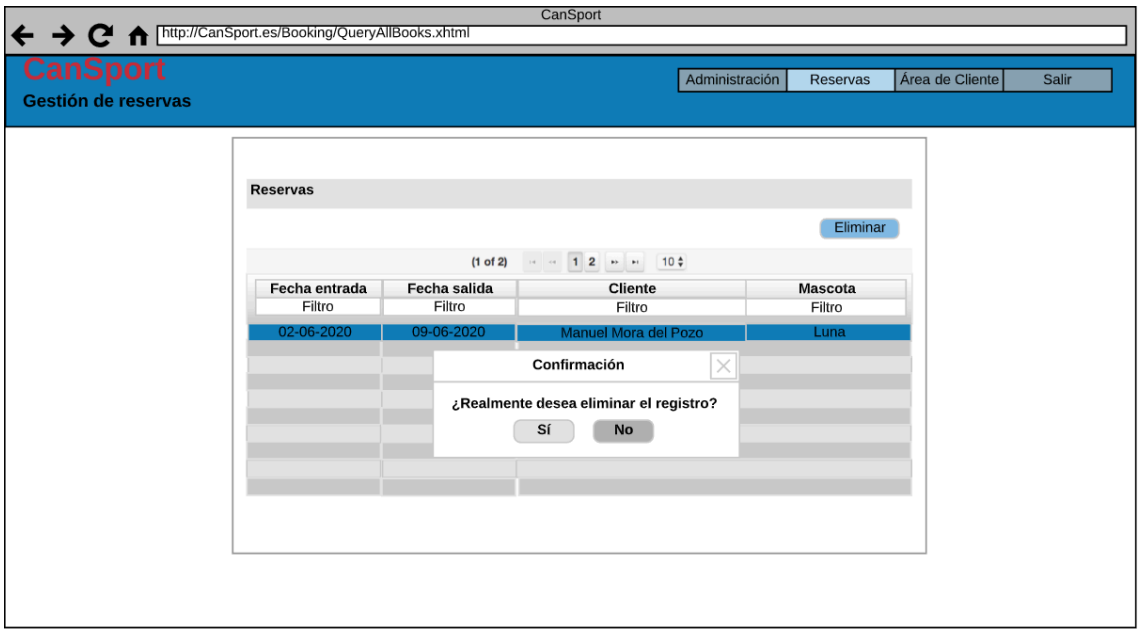


Ilustración 31 Prototipos de pantalla. Eliminando reserva

Encontramos la misma operatoria que vimos en prototipos de consulta anteriores.

Errores, advertencias e información en pantalla

Los mensajes del sistema se ofrecerán en la misma pantalla donde y cuando ocurran. Se presentaran en la zona superior del formulario donde serán visibles. Veamos algunos ejemplos:

Errores

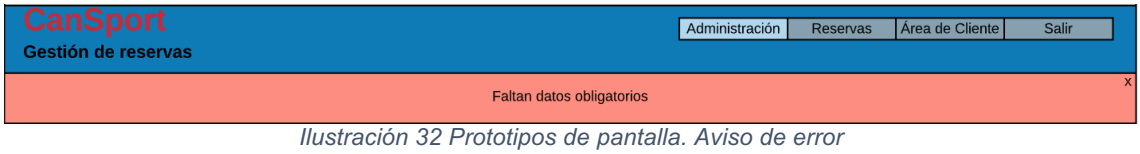


Ilustración 32 Prototipos de pantalla. Aviso de error

Advertencias

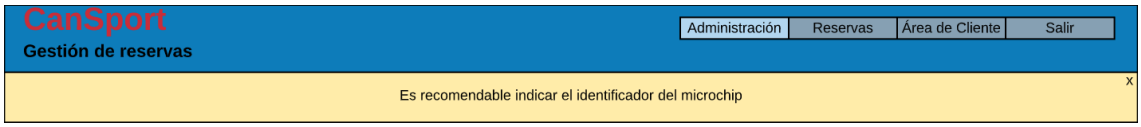


Ilustración 33 Prototipos de pantalla. Aviso de advertencia

Información

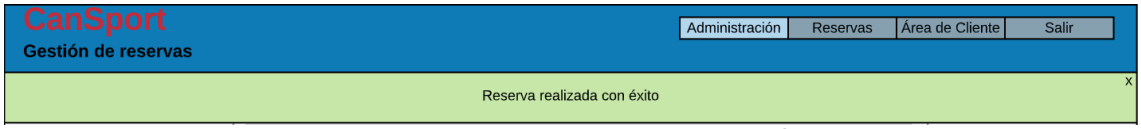


Ilustración 34 Prototipos de pantalla. Mensaje informativo

2.4 Diseño relacional de BBDD

Tras un análisis de requisitos de información se establece un diseño de entidades y aquellas relaciones que permitirán establecer reglas de integridad. Exponemos aquí este diagrama entidad-relación y detallaremos los atributos en el apartado del diagrama de clases.

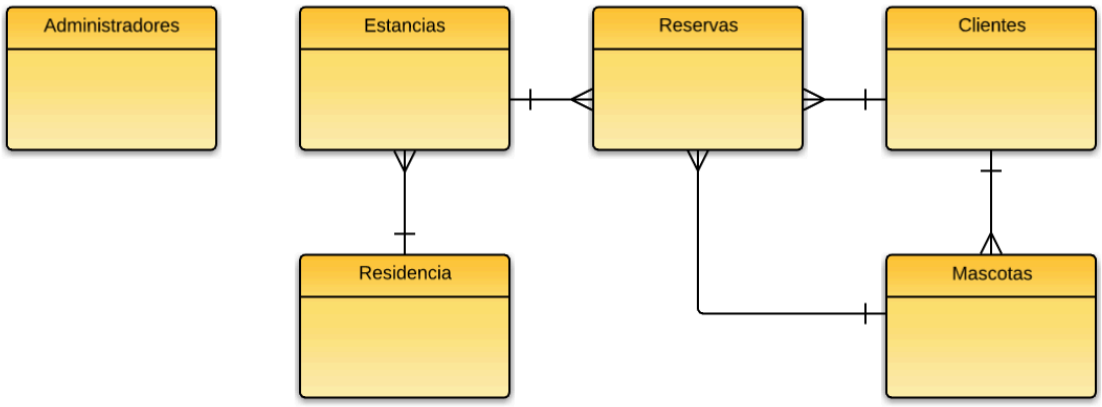


Ilustración 35 Entidad relación del proyecto

Entidades:

- **Administradores**. Donde almacenar los usuarios con privilegios para crear y mantener las estancias y los datos de la residencia canina. También podrá acceder a mantener cualquier otra dato en nombre del cliente.

- **Residencia**. Tabla donde mantener la información de contacto del centro, disponible en la página principal.

- **Estancias**. Datos de cada estancia y su estado.

- **Reservas**. Información sobre las fechas de inicio y fin. Registra la ocupación de una estancia para un cliente y la mascota entre estas fechas. Es una entidad renacida de Estancias, Clientes y Mascotas. Es consecuencia de aplicar un patrón de análisis de asociación histórica.

- **Clientes**. Usuarios del sistema interesados en realizar reservas en la residencia.

- **Mascotas**. Datos de las mascotas de los Clientes.

2.5 Diagrama de clases

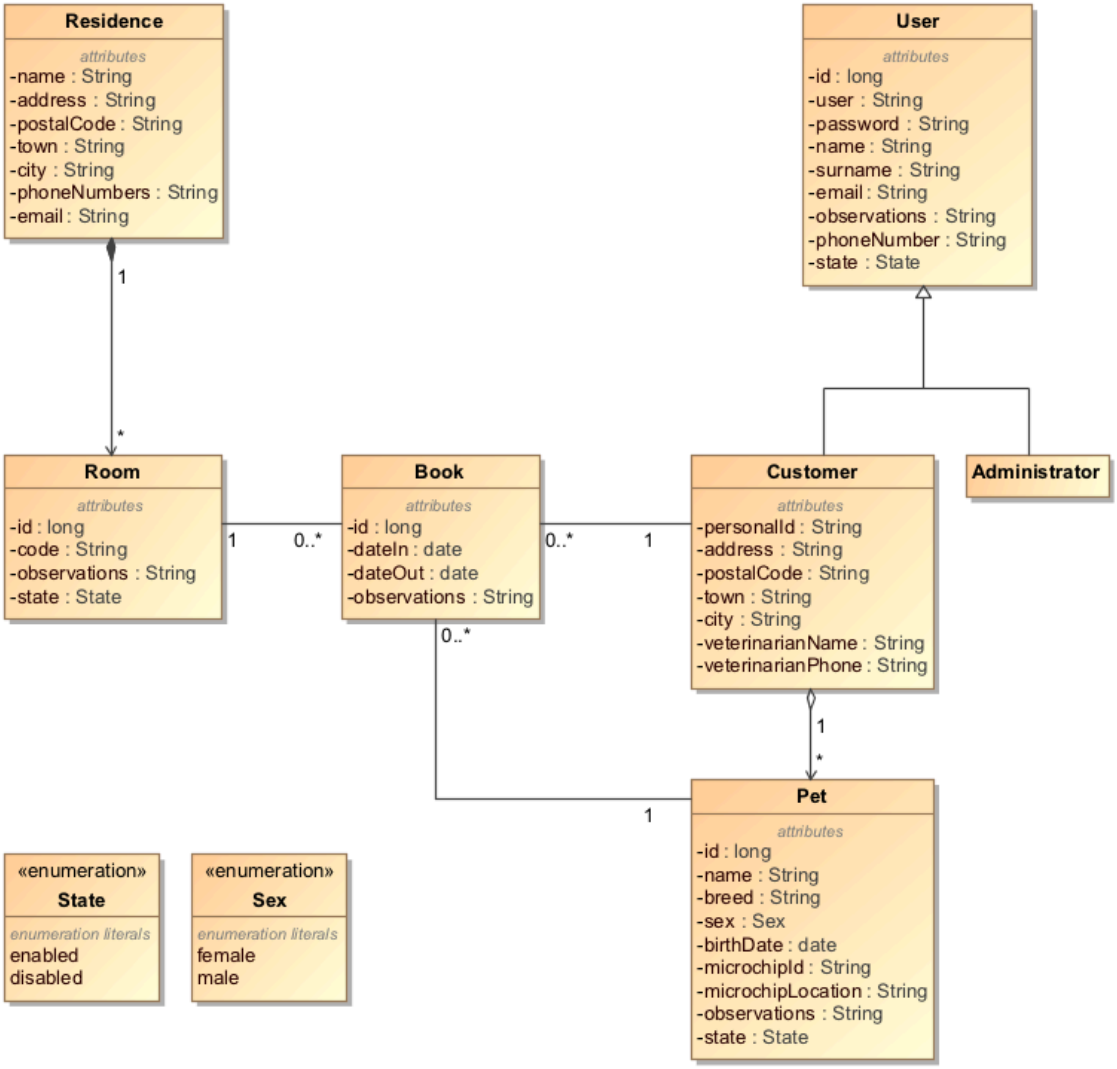


Ilustración 36 Diagrama de clases

Inventario de clases

- ***Residence***. Las propiedades de esta clase permiten configurar los datos de contacto de la residencia canina. Esta información se facilitará en pantallas de la aplicación. Unicamente tendrá un dato de residencia.
  Atributos:
  - *name:String*
    Nombre de la residencia canina. Requerido.
  - *address:String*
    Dirección postal de la residencia.
  - *postalCode:String*
    Código postal del domicilio.
  - *town:String*
    Población de la dirección.
  - *city:String*
    Provincia de la localización.
  - *phoneNumbers:String*

Teléfonos de contacto.
  - *email:String*
    Correo electrónico de contacto del centro.

- ***Room***. Clase del modelo que permite establecer estancias de la residencia que pueden ser reservadas por los clientes.
  Atributos:
  - *id:Long*
    Clave primaria. Identificador único. Requerido.
  - *code:String*
    Código identificativo de uso en la gestión y organización. Único y R requerido.
  - *observations:String*
    Observaciones sobre la estancia.
  - *state:State*
    Estado de la estancia [Activa, Anulada].

- ***User***. Generalización de los datos de acceso y contacto de clientes y administradores del sistema. *Administrator* y *Customer* heredan de *User*.
  Atributos:
  - *Id:long*
    Clave primaria. Identificador único. Requerido.
  - *user:String*
    Usuario para identificarse en el sistema. Único y requerido.
  - *password:String*
    Contraseña codificada del acceso. Requerido.
  - *name:String*
    Nombre del usuario. Requerido.
  - *surname:String*
    Apellidos del usuario. Requerido.
  - *observations:String*
    Observaciones sobre el usuario.
  - *phoneNumber:String*
    Teléfonos de contacto. Requerido.
  - *email:String*
    Correo electrónico de contacto del usuario.
  - *state:State*
    Estado del usuario [Activo, Anulado].

- ***Administrator***. Clase del modelo para los administradores que mantienen el sistema.

- ***Customer***. Clase del modelo para los clientes. Incorpora información personal e información de contacto con veterinario habitual.
  Atributos:
  - *personalId:String*
    Identificador (NIF,NIE..) del cliente. Requerido.
  - *address:String*
    Dirección postal del cliente.
  - *postalCode:String*

Código postal del domicilio.
    - *town:String*
      Población de la dirección.
    - *city:String*
      Provincia de la localización.
    - *veterinarianName:String*
      Nombre del veterinario habitual.
    - *veterinarianPhone:String*
      Teléfono de contacto del veterinario.

- ***Pet***. Clase del modelo para las mascotas de los clientes. Incorporan las propiedades necesarias para identificarlas.
  Atributos:
    - *Id:long*
      Clave primaria. Identificador único. Requerido.
    - *name:String*
      Nombre de la mascota. Requerido.
    - *breed:String*
      Raza del animal.
    - *sex:Sex*
      Sexo [Masculino, Femenino]. Requerido.
    - *birthdate:Date*
      Fecha de nacimiento. Requerido.
    - *observations:String*
      Observaciones sobre la mascota.
    - *microchipId:String*
      Código de identificación del microchip.
    - *microchipLocation:String*
      Localización del microchip implantado.
    - *state:State*
      Estado de la mascota [Activo, Anulado].

- ***Booking***. Asociación entre *Customer, Pet* y *Room*. Cumpliendo un patrón de análisis de asociación histórica, incorpora fechas de inicio y salida.
  Atributos:
    - *Id:long*
      Clave primaria. Identificador único. Requerido.
    - *dateIn:Date*
      Fecha de entrada. Requerido.
    - *dateOut:String*
      Fecha de salida. Requerido.
    - *observations:String*
      Observaciones sobre la reserva.

2.6 Diagrama de arquitectura

En base a las características de los usuarios y los objetivos de utilización, podemos concluir que el modelo arquitectónico que me mejor se ajusta a las necesidades es un modelo heterogéneo en capas o niveles y una arquitectura cliente-servidor.

Finalmente, se opta por un sistema organizado en tres capas. A través de la capa de presentación implementaríamos solución para que, a través de un cliente ligero (navegador web), los clientes accedan al sistema. Esta capa se apoyaría en la de lógica de negocio que a su vez mediaría con la de persistencia.

Diseño general. Punto de vista de la computación

A continuación podemos observar un diagrama en el que se ven representados los componentes del sistema en un modelo simplificado y de mínima granularidad. Aquí vemos la especificación de las operaciones de cada módulo en los que agrupamos la funcionalidades del proyecto.

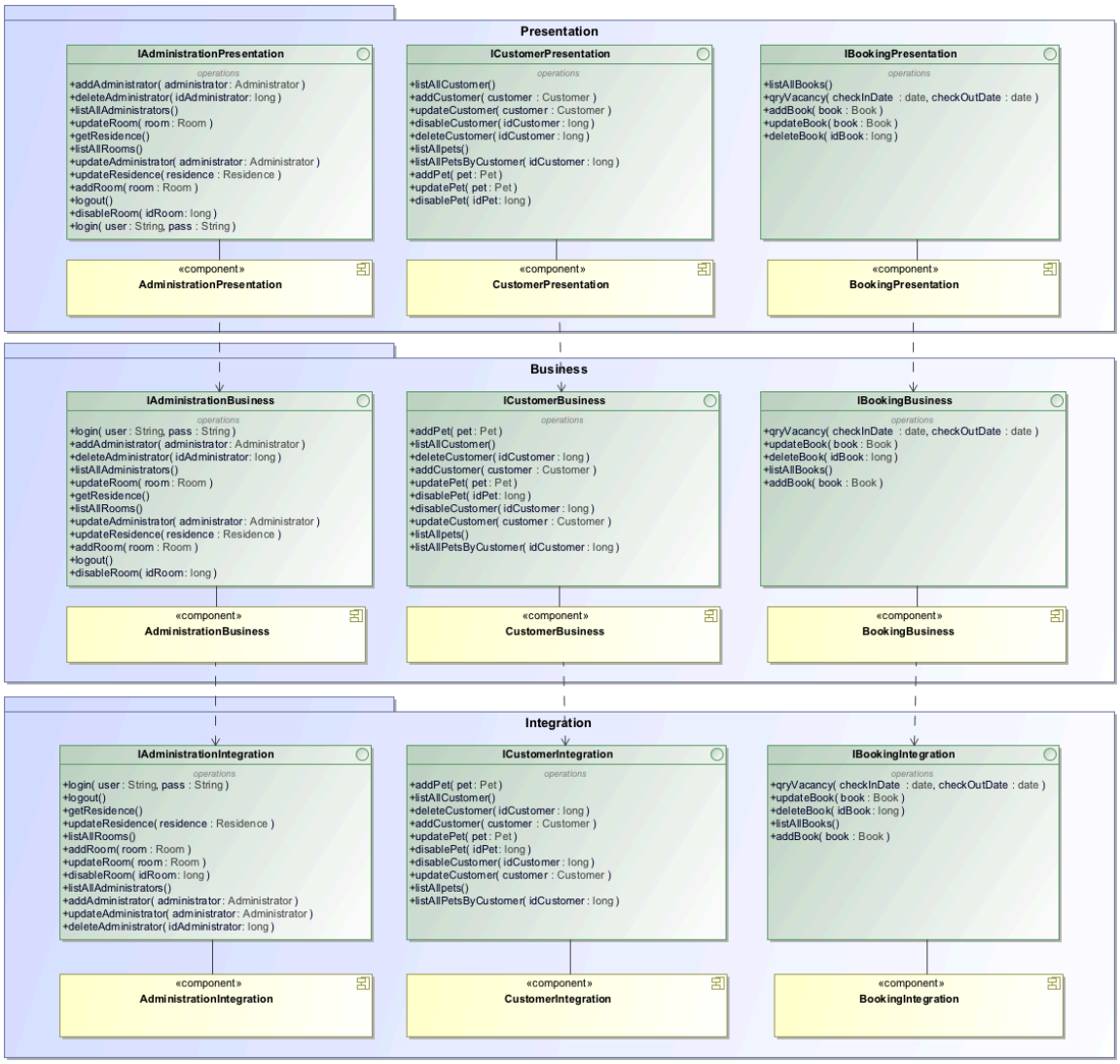


Ilustración 37 Diseño general. Punto de vista de la computación

En este diseño general aún no describimos los componentes con detalle, pero si se muestran las especificaciones para los interfaces en las que se indican los métodos ofrecidos entre capas:

Administration

- Identificación y salida del sistema:
  - *login(user : String, pass : String)*
  - *logout()*
- Consulta de los datos de la residencia canina
  - *updateResidence(residence : Residence)*
  - *getResidence()*
- Gestión de administradores:
  - *addAdministrator(administrator : Administrator)*
  - *deleteAdministrator(idAdministrator : long)*
  - *listAllAdministrators()*
  - *updateAdministrator(administrator : Administrator)*
- Gestión de estancias:
  - *listAllRooms()*
  - *addRoom(room : Room)*
  - *updateRoom(room : Room)*
  - *disableRoom(idRoom : long)*

Customer

- Gestión de clientes:
  - *listAllCustomer()*
  - *addCustomer(customer : Customer)*
  - *updateCustomer(customer : Customer)*
  - *disableCustomer(idCustomer : long)*
  - *deleteCustomer(idCustomer : long)*
- Gestión de mascotas:
  - *listAllpets()*
  - *listAllPetsByCustomer(idCustomer : long)*
  - *addPet(pet : Pet)*
  - *updatePet(pet : Pet)*
  - *disablePet(idPet : long)*

Booking

- Gestión de reservas:
  - *listAllBooks()*
  - *qryVacancy(checkInDate : date, checkOutDate : date)*
  - *addBook(book : Book)*
  - *updateBook(book : Book)*
  - *deleteBook(idBook : long)*

A continuación vemos representadas las dependencias de los componentes de la capa de integración con las clases de nuestro diseño del sistema de información.

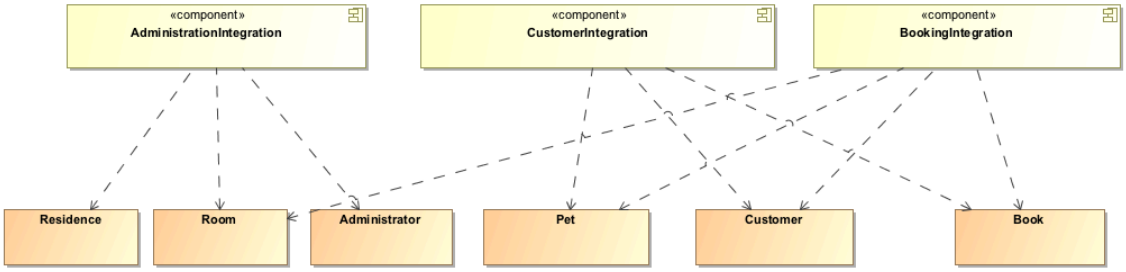


Ilustración 38 Dependencias componentes y entidades

2.7 Arquitectura orientada a Java Enterprise Edition

Para el desarrollo de nuestro proyecto se opta por la utilización de *Java Enterprise Edition,* siendo la versión 8 la disponible en la fechas de confección del documento. Esta elección permite la implementación de una estructura multicapa y el uso de clientes ligeros en distinta tipología de dispositivos, mientras tengan un navegador con opción de acceso a internet.

2.7.1 Capa de presentación

Aún con un modelo sin detallar, podemos avanzar en nuestro diseño incorporando una propuesta en nuestra capa de presentación. Añadiremos dos tipos de componentes por cada módulo que se dividirán las responsabilidades:

- ***Vistas***. Interface con el usuario.
- ***Driver***. Componente que dará soporte a las vistas. Proporcionará toda la funcionalidad y mediación entre las vistas y la lógica de negocio del sistema.

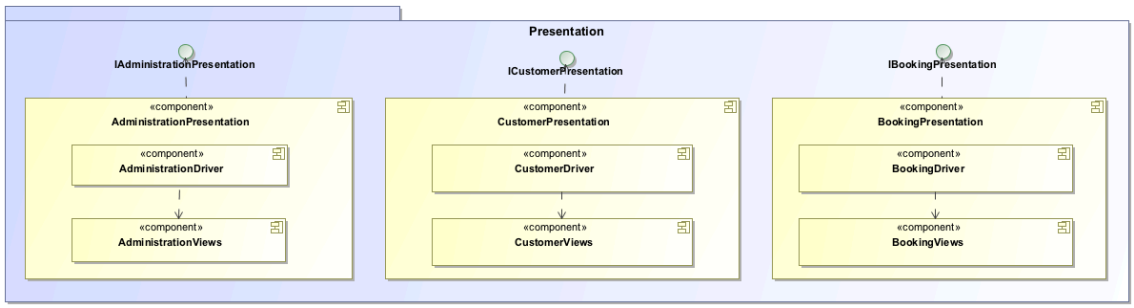


Ilustración 39 Diseño general. Capa presentación

Para la implementación de esta capa utilizaremos la combinación de *Java Server Faces* y el *framework Primefaces Community Edition*. Esta asegura que la aplicación proporcione al usuario una experiencia de fácil interpretación y accesibilidad mediante componentes ampliamente probados y verificados por una amplia comunidad de desarrolladores. Cumple con los objetivos de obtener

un producto libre de licencias de pago. Dedicaremos un próximo capítulo a detallar esta estructura.

Si avanzamos en este diseño veríamos desgranar el modelo en una especificación más completa. El siguiente diagrama introduce el desglose de las vistas propuesto e incorpora en los componentes de tipo *driver,* los elementos de tipo *action* y un *frontControler* para cada módulo. Veamos brevemente las funciones de estas:

- ***Action*** (*Managed Bean*). Proporcionaran un interface que interactuará con las acciones del usuario y soporte lógico a las vistas.
- ***FrontController*** (*Servlet*). Mediación con la lógica de las acciones y las peticiones del usuario. Ejecutará y coordinará estas acciones sin implementarlas, proporcionando desacople e independencia.

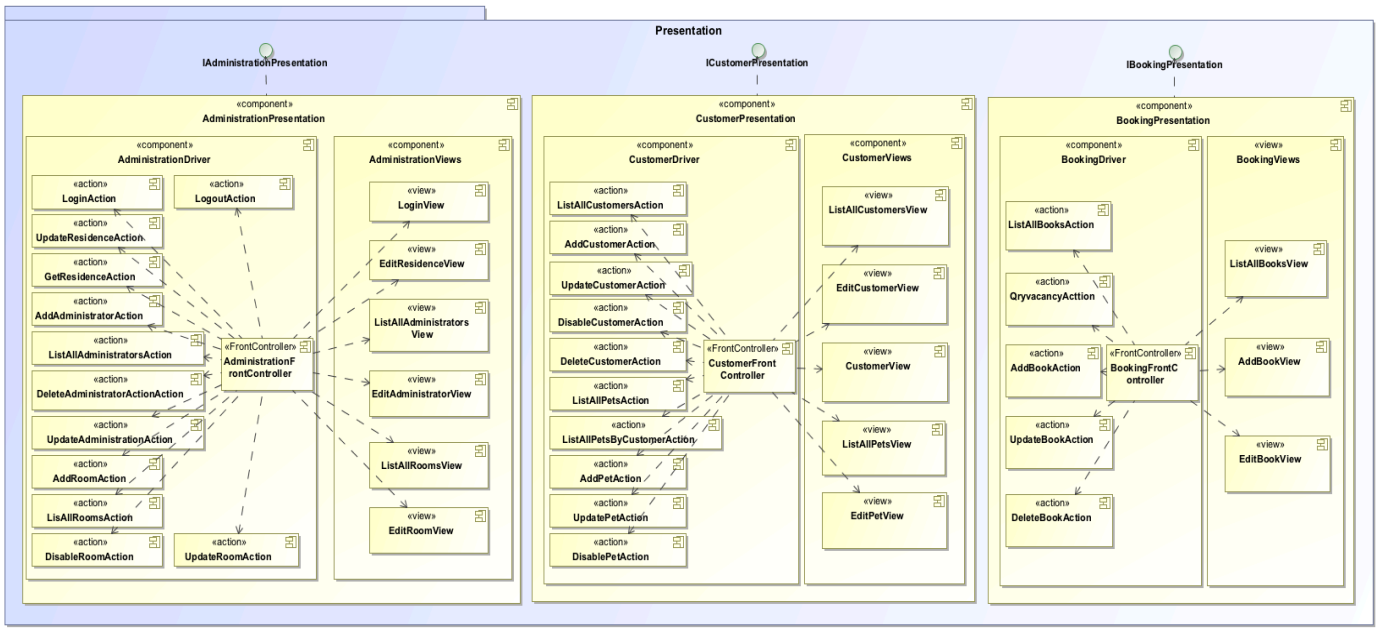


Ilustración 40 Diseño en detalle. Capa presentación

2.7.2 Capa de negocio

Continuando con el modelo general, en este apartado añadimos especificaciones a nuestro diseño desde el punto de vista de la computación. Nos sirve la representación actual, vista en el capítulo anterior. La implementación de los componentes de la capa de negocio será mediante *Enterprise Java Beans* (EJB), garantizando sencillez, escalabilidad y fiabilidad. Entre estas especificaciones incluimos:

- Transacciones gestionadas por el contenedor.
- Peticiones de usuario síncronas.
- Consultas del usuario sin estado.

2.7.3 Capa de integración

En esta capa se implementaran todas las tareas necesarias de persistencia y la definición del modelo conceptual y lógico, que integrará al proyecto con la base de datos. Estas tareas se simplifican por medio de *Java Persistence API* (JPA), desacoplando y delegando en el *framework* el mapeo con las tablas de datos.

El proyecto se completa con un gestor de base de datos. En nuestro caso usaremos *PostgreSQL*, continuando con nuestro objetivo de usar software libre, con la gran ventaja de la distribución gratuita en el producto final.

2.7.4 Servidor de aplicaciones

Para el despliegue y ejecución de la aplicación usaremos un servidor de aplicaciones Java EE de código abierto: *Wildfly* (JBOSS). Se ha usado en las asignaturas prácticas de laboratorio a lo largo del grado y no requiere una compleja configuración para su puesta en marcha. Tiene un amplio soporte por una gran comunidad y continuas actualizaciones.

2.8 Diseño y modelado de la arquitectura JavaEE

Si estudiamos detenidamente los objetivos del proyecto, podremos comprobar que se trata de una sencilla aplicación cuya gestión se basa en el mantenimiento de datos y el registro de reservas de estancias. Por tanto, la mayoría de la interacción con los usuarios es a través de operaciones CRUD, incluido la realización de una reserva, que incorpora un método adicional para comprobar si podrá añadirse. La estructura de las pantallas de cada entidad es de operatoria similar y la única excepción la encontramos en la relación cliente y mascotas, que requiere un maestro detalle.

A partir de la especificación de las operaciones que se han descrito en los diagramas anteriores, comprobamos la similitud y repetición de métodos para listar todos sus elementos y añadir, modificar, anular o eliminar un elemento de la clase. Todos estas son susceptibles de generalizar y es así como los incorporaremos a la propuesta, simplificando y reutilizando código. Este diseño incorpora sencillez y mayor fiabilidad al tener menos exposición a posibles fallos.

2.8.1 Capa de presentación

Decidida la tecnología de implementación se adoptan decisiones de diseño:

- Uso de *Java Server Faces* (JSF) para la capa de presentación.
- Las vistas utilizaran componentes de *Primefaces* con *Facelets*.
- El propio JSF proporciona el controlador *servlet*: *Faces Servlet*.
- Las acciones se implementan con *Managed Bean*.

Todas las peticiones http del usuario serán recogidas por el controlador *servlet* (en nuestro caso *Faces Servlet*) que actualizará la información en los *managed beans* y vistas. Lanzaran la lógica asociada a eventos y los métodos asociados a las acciones. *Primefaces*[3] proporciona componentes de interfaz que se vinculan al modelo implementado en los manejadores, simplificando la implementación y diseño de los formularios, reduciendo la complejidad mediante *frameworks* verificados y estables.

En el módulo anterior del proyecto hemos elaborado un diseño detallado, incorporando a cada componente *driver* las vistas y las acciones necesarias. Para agrupar y simplificar la implementación, se reunirán estas acciones en *managed beans* orientados a reducir la lógica de control y mediación con la capa de negocio. E interactuar con las vistas mejorando la experiencia del usuario. De nuevo, podremos aplicar el concepto de generalización agrupando las operaciones similares dependientes de la entidad a administrar.

Y de esta forma, describimos a continuación esa agrupación de acciones en los *managed beans* propuestos. A continuación, en los siguientes apartados, detallaremos el diseño por módulos.

Administration

Detallamos a continuación los elementos y diagramas del diseño, desgranados para la implementación del módulo de administración:

- AdministrationDriver:
    - LoginMBean
        - Login. Operación para acceder al sistema.
    - LogoutMbean
        - Logout. Operación para salir del sistema.
    - ResidenceMbean
        - Get. Acción para obtener los datos de la residencia.
        - Update. Acción para actualizar los datos editados.
    - ListAdministratorsMbean
        - ListAllAdministrators. Consulta todos los administradores.
        - DeleteAdministrator. Elimina administrador del sistema.
    - AdministratorMBean
        - AddAdministrator. Añade nuevo administrador.
        - UpdateAdministrator. Modifica información del administrador.
    - ListRoomsMBean
        - ListAllRooms. Consulta todos las estancias.
        - DisableRoom. Anula estancia en el sistema.
    - RoomMBean
        - AddRoom. Añade nueva estancia.
        - UpdateRoom. Modifica información de la estancia.

- AdministrationViews:
    - LoginView. Pantalla de acceso al sistema.
    - EditResidence. Edición de los datos de la residencia canina.
    - ListAdministrators. Consulta de administradores.
    - EditAdministrator. Consulta, alta y edición de administrador.
    - ListRooms. Consulta de estancias.
    - EditRoom. Consulta, alta y edición de estancia.

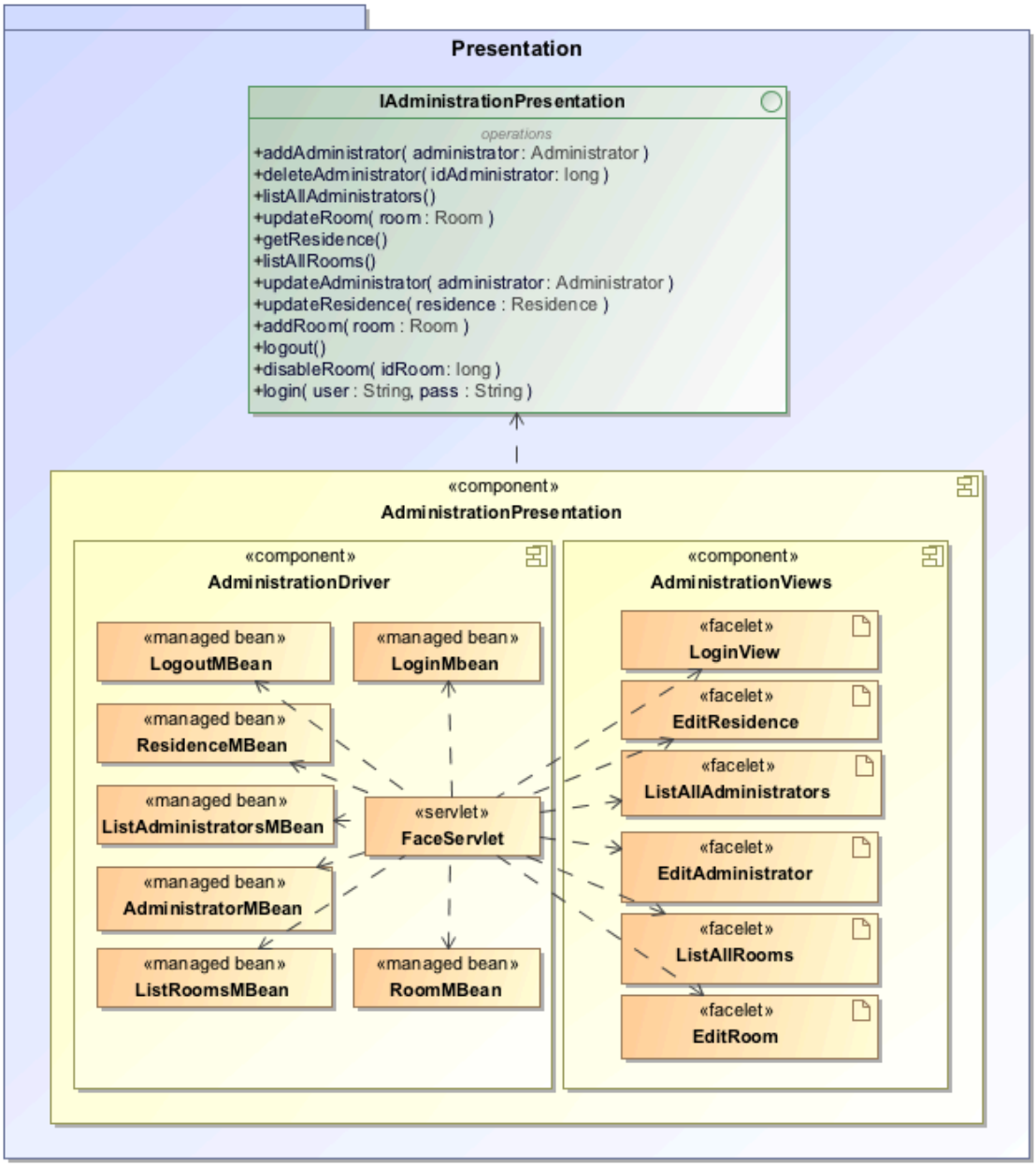


Ilustración 41 Diseño detalle JavaEE. Capa presentación. Administración

Customer

- CustomerDriver:
  - ListCustomersMbean
    - ListAllCustomers. Consulta todos los clientes.
    - DeleteCustomer. Elimina cliente del sistema (solo administrador).
    - DisableCustomer. Anula cliente en el sistema.
  - CustomerMBean
    - AddCustomer. Añade nuevo cliente.
    - UpdateCustomer. Modifica información del cliente.
  - ListPetsMBean

        - ListAllPets. Consulta todos las mascotas.
        - DisablePet. Anula mascota en el sistema.
    - PetMBean
        - AddPet. Añade nueva mascota.
        - UpdatePet. Modifica información de la mascota.

- CustomerViews:

    - ListCustomers. Consulta de clientes.
    - EditCustomer. Consulta, alta y edición de cliente.
    - ViewCustomer. Consulta de cliente.
    - ListPets. Consulta de mascotas.
    - EditPet. Consulta, alta y edición de mascota.

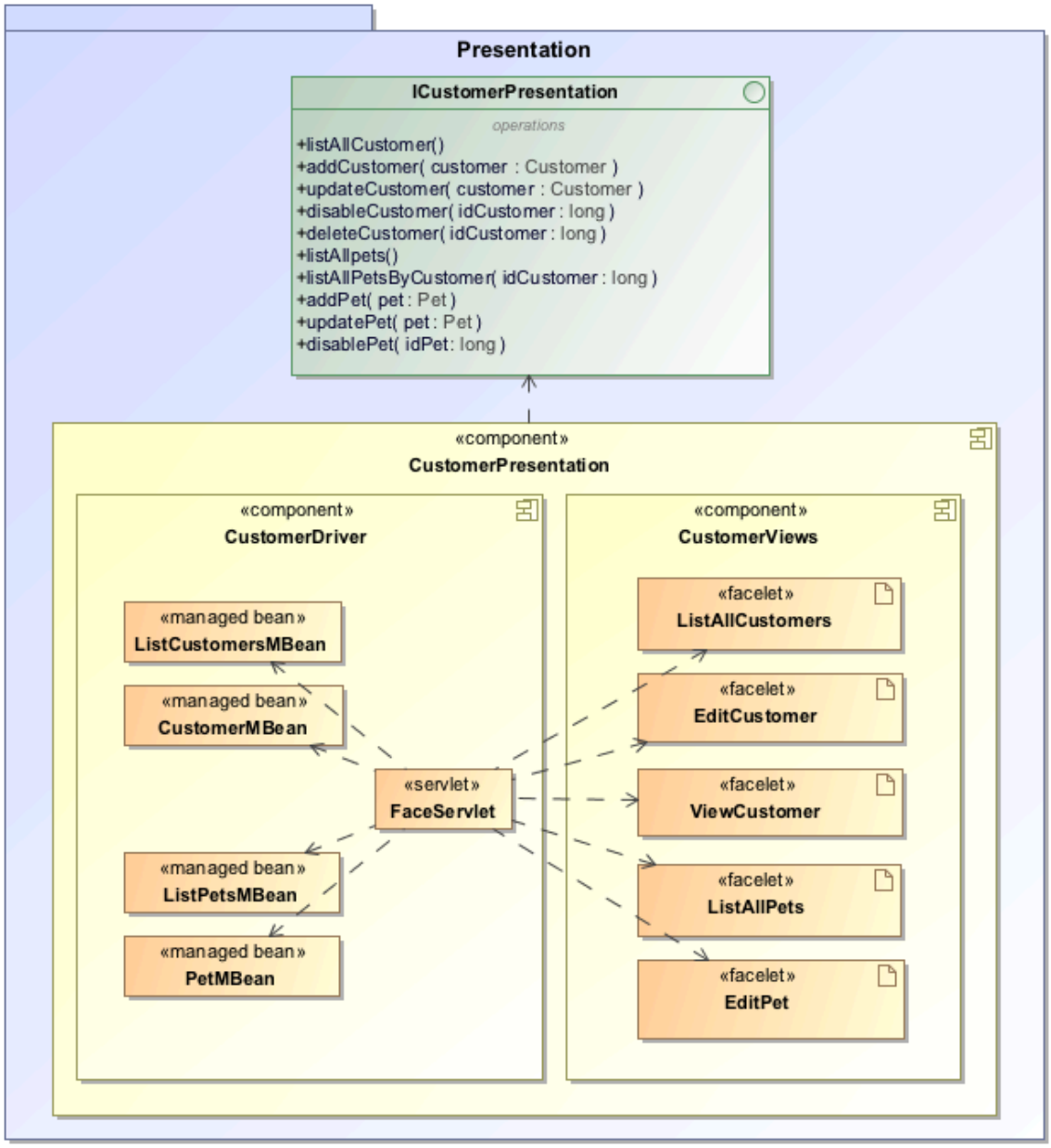


Ilustración 42 Diseño detalle JavaEE. Capa presentación. Clientes

Booking

- BookingDriver:
  - ListBooksMbean
    - ListAllBooks. Consulta todos las reservas.
    - DeleteBook. Elimina reserva del sistema.
  - BookMBean
    - QryVacance. Comprueba disponibilidad.
    - AddBook. Añade nueva reserva.
    - UpdateBook. Modifica información de la reserva.
- BookingViews:
  - ListBooks. Consulta de reservas.
  - EditBook. Consulta y edición de reserva.
  - AddBook. Consulta disponibilidad y alta de reserva.

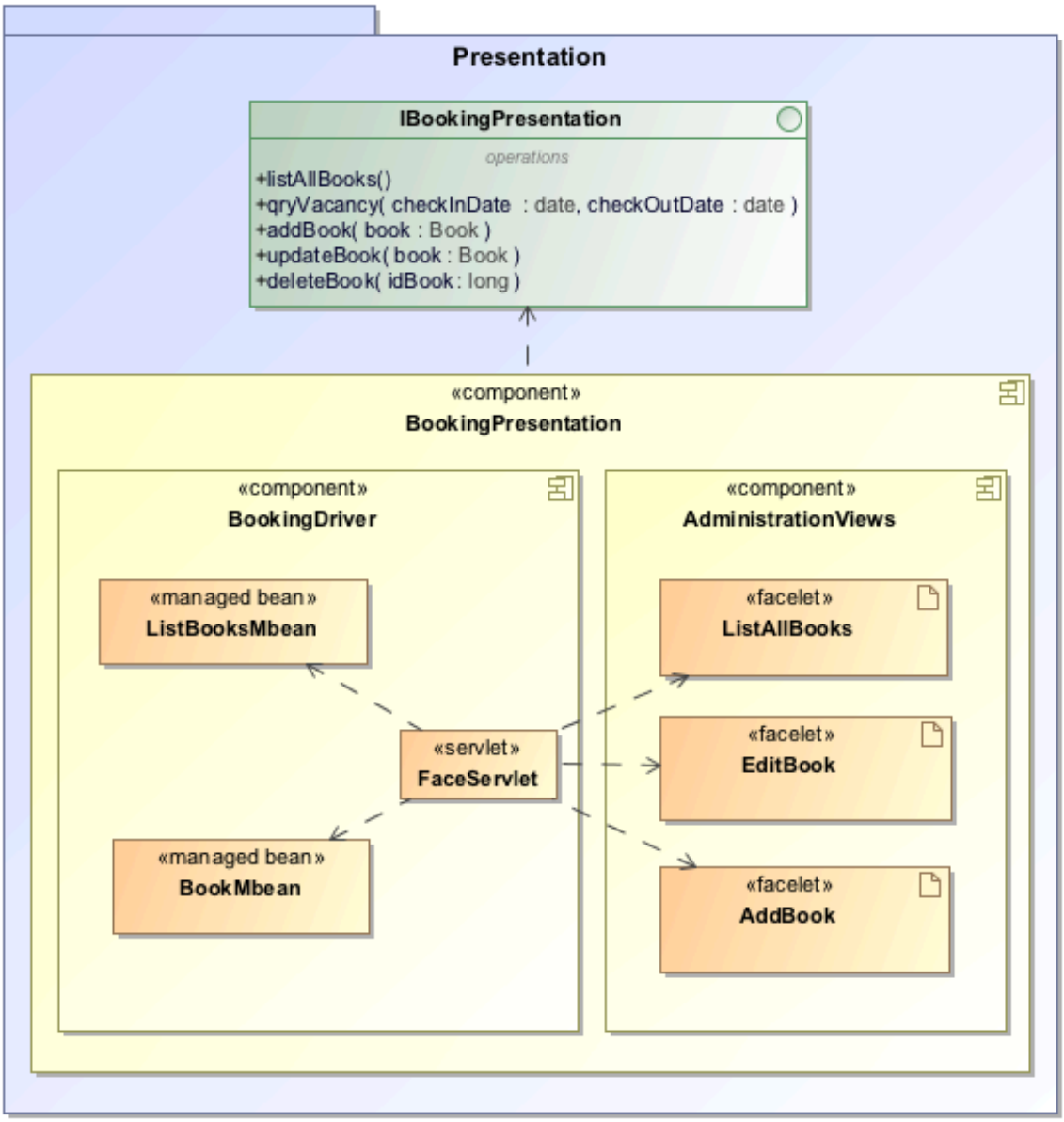


Ilustración 43 Diseño detalle JavaEE. Capa presentación. Reservas

2.8.2 Capa de negocio

La arquitectura *Enterprise Java Beans* es la tecnología utilizada para implementar el uso de componentes distribuidos. Se despliegan en un contenedor EJB del servidor de aplicaciones y permiten gran reutilización. Aplicando el perfil Java EE al diseño debemos dar acceso tanto remoto como local a los componentes, por lo que a nuestro diseño actual de la capa se le incorpora un interface para tipo de acceso. Enumeraremos los principios que incorporamos a nuestro sistema en esta arquitectura:

- Despliegue en contenedores (servidor de aplicaciones).
- Interceptación de las llamadas por el contenedor y pasan el control al componente.
- El propio contenedor gestionará la seguridad y resolverá la concurrencia.
- Puede establecerse una configuración de *pool* de conexiones mejorando así el rendimiento.
- El contenedor gestionará el ciclo de vida de las instancias de los componentes.
- Disponibilidad y escalabilidad. Los servidores de aplicaciones aseguran alta disponibilidad y opciones de ampliación.

Administration

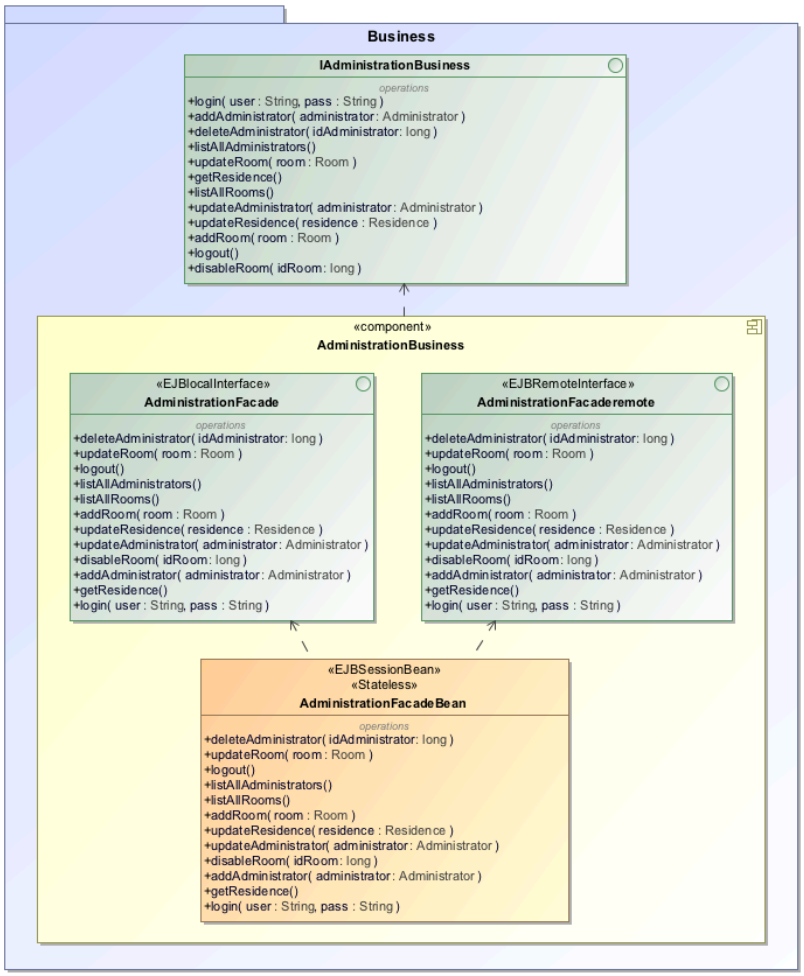


Ilustración 44 Diseño detalle JavaEE. Capa negocio. Administración

Customer

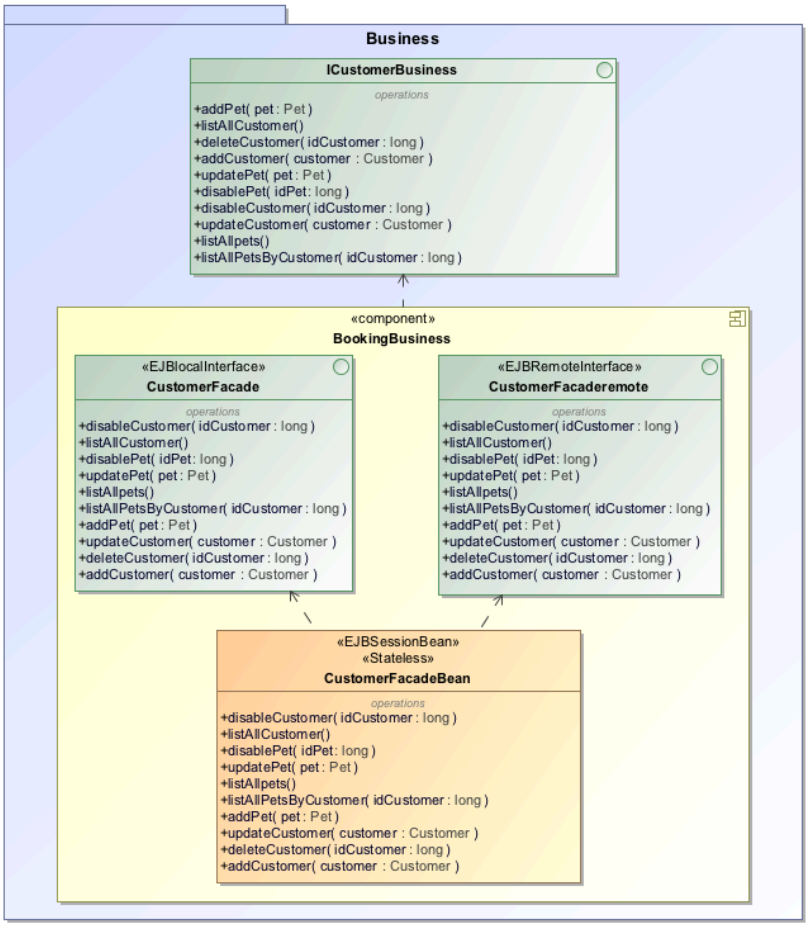


Ilustración 45 Diseño detalle JavaEE. Capa negocio. Clientes

Booking

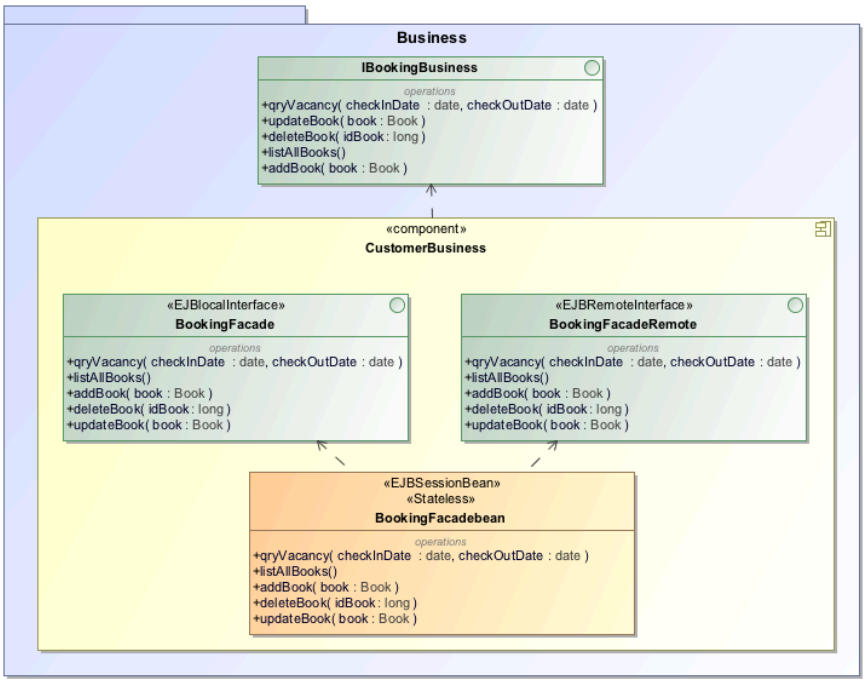


Ilustración 46 Diseño detalle JavaEE. Capa negocio. Reservas

2.8.3 Capa de integración

El acceso a la capa de datos será local y se almacenará la información del sistema en una base de datos relacional. La solución propuesta consiste en el uso de persistencia mediante *Java Persistence API* (JPA). Esta alternativa automatiza y nos aísla de los procesos de creación, lectura y modificación de la información.

La configuración del mapeo entre las clases de entidad del modelo y las tablas del gestor de base de datos puede realizarse mediante anotaciones o mediante un fichero de configuración de persistencia XML. Esto especifica un interfaz para las operaciones. Mediante lenguaje JPQL se podrán establecer consultas específicas.

Cada entidad JPA representa una tabla o vista de la base de datos, una instancia de dicha clase representa una fila de los datos y cada atributo, una columna.

Administration

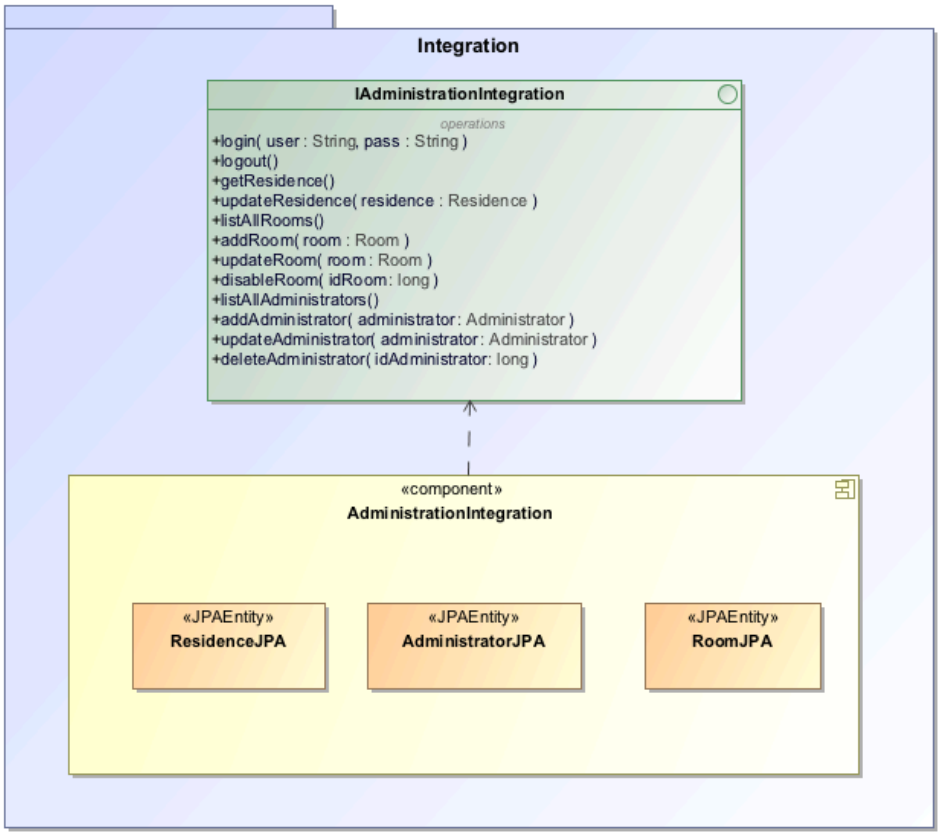


Ilustración 47 Diseño detalle JavaEE. Capa integración. Administración

Customer

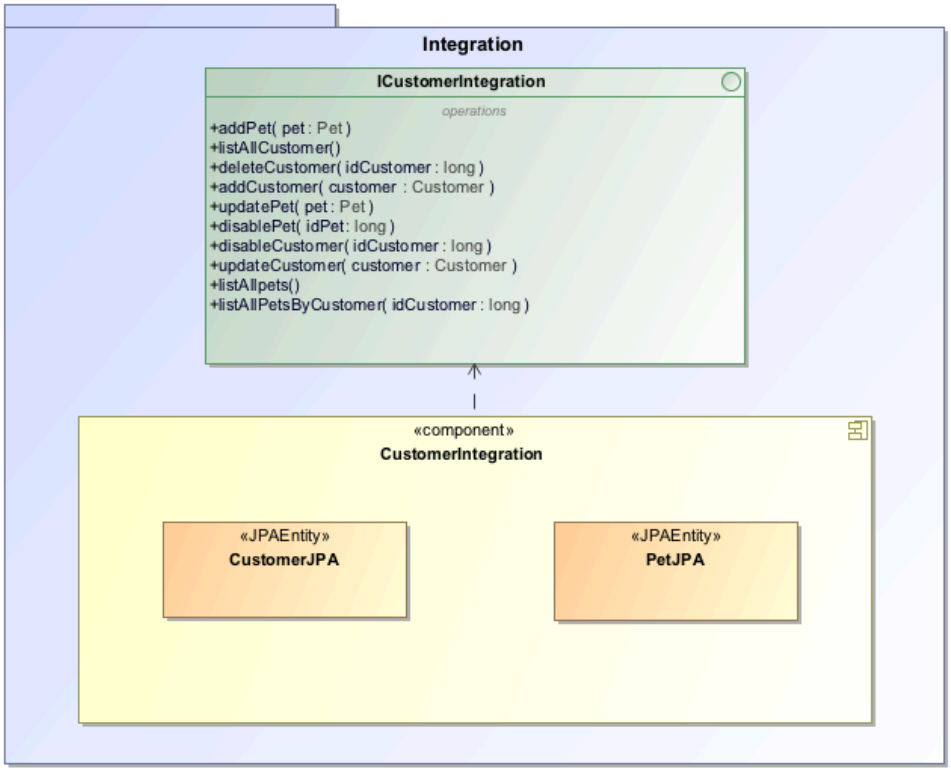


Ilustración 48 Diseño detalle JavaEE. Capa integración. Clientes

Booking

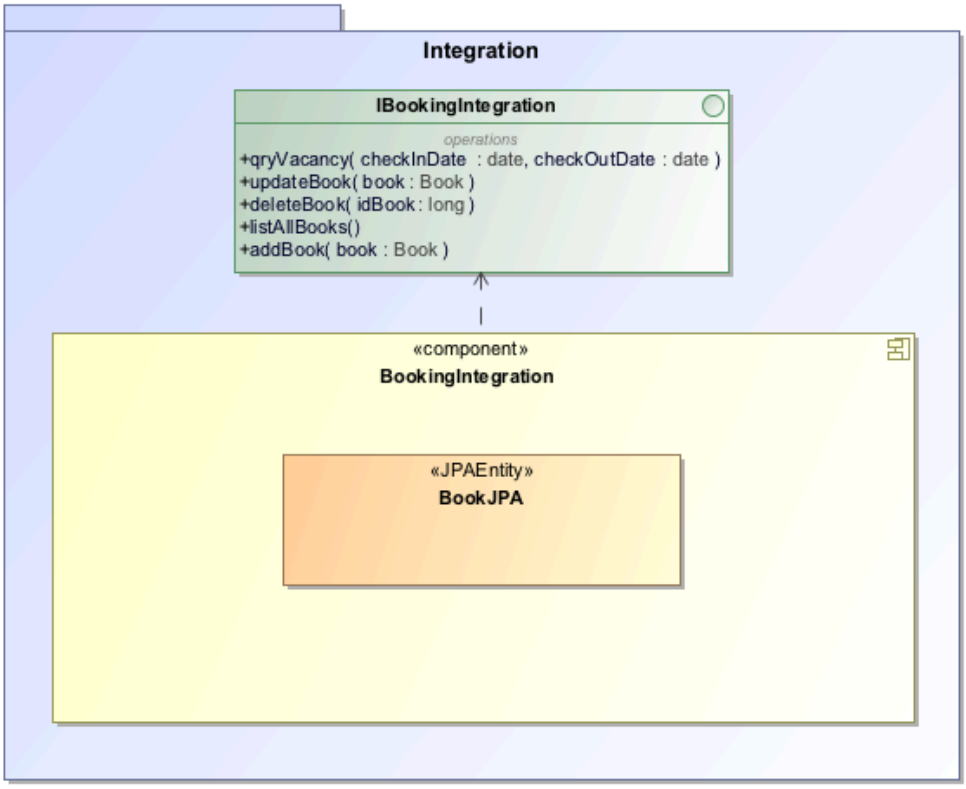


Ilustración 49 Diseño detalle JavaEE. Capa integración. Reservas

3 Implementación

Con la base de la infraestructura de desarrollo (véase capítulo 1.2.2) y una planificación general basada en iteraciones, establecidas en el capítulo 1.4, el periodo de implementación se inicia desgranando los siguientes tareas a realizar. Estas se centrarán en la gestión de los clientes, siendo este el módulo base de la gestión de reservas y a la veremos enriquecer con más funcionalidad.

A raíz de la consecución de las pequeñas metas fijadas en esa primera etapa y evolucionando en el recorrido del aprendizaje, es posible comprobar en el propio código como se han ido aplicando técnicas y generalizaciones que han permitido facilitar una base de lógica estable y reutilizable para las siguientes iteraciones basadas en los siguientes casos de uso.

En cada periodo, se han establecido objetivos fácilmente alcanzables aportando valor. Finalizada cada iteración, ha facilitado la consecución de la siguientes. Las pruebas y refinamiento han proporcionado una revisión constante y una mejora continua sin necesidad de comenzar desde el principio. Y es quizás, en el periodo final donde se han practicado distintos métodos para conseguir fines similares, lo que proporcionará en este proyecto elementos de referencia en proyecto futuros.

Todo producto acabado que cumpla con la funcionalidad esperada es, a menudo, valorado por su estética y facilidad para su uso. Este este fue el motivo fundamental para la elección de utilizar componentes Primefaces[3]. Básicamente proporciona un *framework* cuyos componentes proporcionan estabilidad y facilidad para configurar cualquiera de los temas proporcionados. La aplicación de estos influye en el aspecto y operatoria en todo el entorno sin necesidad de variaciones individuales. El tema elegido para este proyecto es *humanity*.

Se han incluido vistas para capturar los errores típicos en este entorno de visualización. Error 404 para advertir cuando un usuario quiere navegar a una vista no implementada. Y el error 500, en previsión de un error en tiempo de ejecución no controlado. Las advertencias y mensajes informativos se integran en los propios formularios, de esta forma la interacción con el usuario es más ágil y transparente. Se diferencian fácilmente el nivel de indicaciones de error, advertencia e información a través del color de las etiquetas emergentes.

Como referencias futuras de uso e integración, se han practicado diferentes métodos para la navegación entre pantallas y paso de parámetros. Son evidentes que algunos métodos resultan más dinámicos en la navegación. Dependiendo del volumen de información en pantalla y de la complejidad añadida, se han practicado desde los cuadro de diálogos emergentes hasta el salto a otros formularios totalmente independientes y siempre accesibles de las opciones de menú, por lo que se ha evitado incorporar en algún caso la funcionalidad de volver a pantalla anterior cuando se han reutilizado estas vistas.

Reutilizamos formularios limitando funcionalidad y acceso según este perfil. Comprobaremos que también se reutilizan para la edición y el alta de nuevos elementos sin incorporar excesiva complejidad en la lógica codificada. Nos

hemos apoyado en la revisión constante y advertencias del producto Sonarlint[6]. Este producto evalúa ciertos aspectos de la codificación en base a un repositorio de reglas, evaluando aspectos como buenas prácticas en la simplicidad, e incluso, la complejidad.

Excepto el acceso al formulario de alta de nuevo cliente para usuarios no registrados, se ha establecido un modelo de plantilla con información común y relacionada con la vista actual en pantalla. Así, encontramos interpretado las propuestas que encontramos en los prototipos, estableciendo un estándar de uso cada una de las pantallas del diseño. Una zona superior, con posibilidades de mejora y particularización de la entidad. Una zona inferior, con información de contacto de la residencia. Y en los laterales, pudiendo ser minimizados, a la izquierda las opciones de menú según el rol y a la derecha grandes posibilidades para indicar ayudas e instrucciones al usuario.

3.1 Producto obtenido

La funcionalidades, visibles desde el menú, son diferentes según el perfil de usuario de la sesión. Se han incorporado validaciones en el código que evitarían que un cliente pudiera acceder a la misma información que un administrador. Este no podrá acceder a datos de otros clientes y un administrador, sin embargo, podrá realizar acciones en nombre de los clientes que lo demanden.

Comprobemos en las siguiente ilustración como un cliente al acceder tiene menos opciones en su menú:

Ilustración 50 Menú para cliente

Ilustración 51 Menú para administrador

Comprobamos el aspecto visual que proporciona el tema configurado: *humanity*.

Como pantalla principal, accederemos directamente a la consulta de clientes. De esta podremos navegar a las distintas opciones desde la propia vista de edición. Si accedemos al sistema como administrador podremos seleccionar uno y administrarlo. Al acceder podremos crear uno nuevo, y si seleccionamos un registro, veremos las opciones disponibles a través de los botones que ahora será visibles:

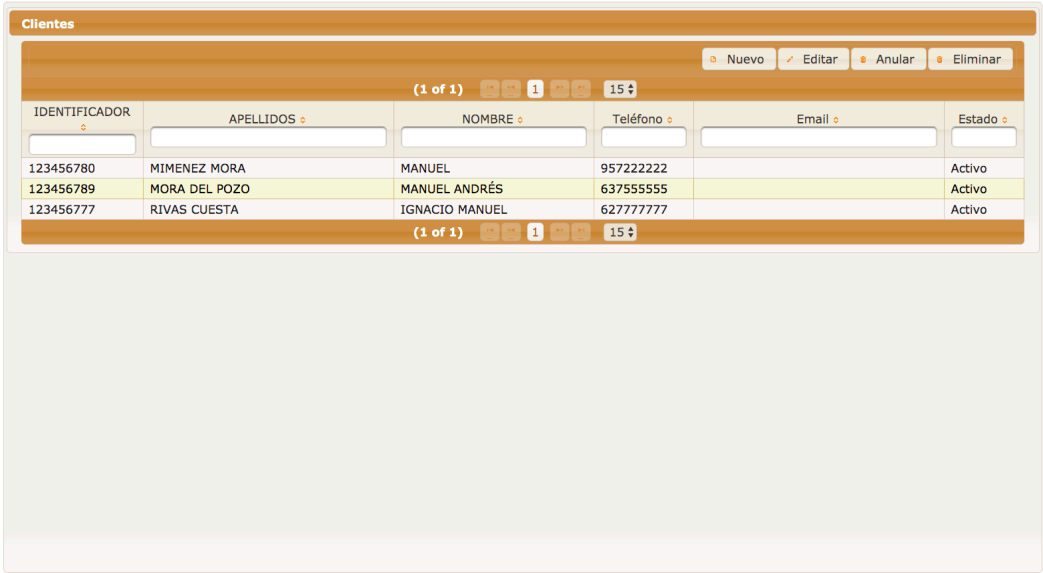


Ilustración 52 Consulta de clientes

Comprobaremos, en esta y resto de pantallas de consulta, el componente utilizado. Integra opciones de ordenación, filtrado y paginación. Únicamente un administrado puede eliminar un cliente y sus datos. El sistema le advertirá y solicitará confirmación. Si optamos por la edición, comprobaremos un formulario complejo con varios bloques de información y funcionalidades accesibles desde esta opción. Si el cliente dispone de reservas futuras no tendrá activa la opción de cambiar el estado.

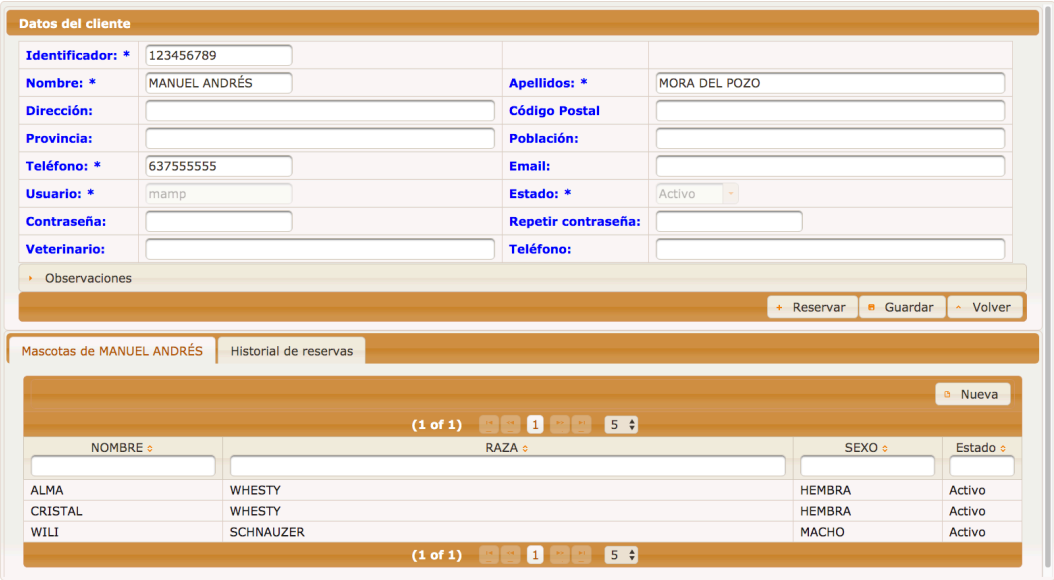


Ilustración 53 Administración de datos de cliente

Desde una sesión con usuario que disponga rol de administrador podremos mantener la información de contacto visible de la residencia:

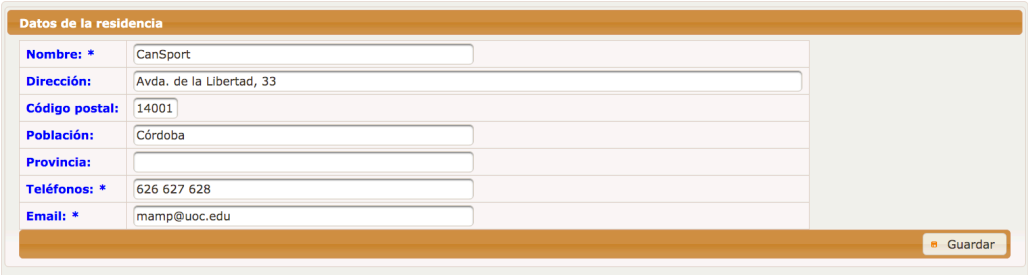


Ilustración 54 mantenimiento de datos de contacto de la organización

La consulta de administradores tiene una estética similar a la de clientes.

Ilustración 55 Consulta de administradores

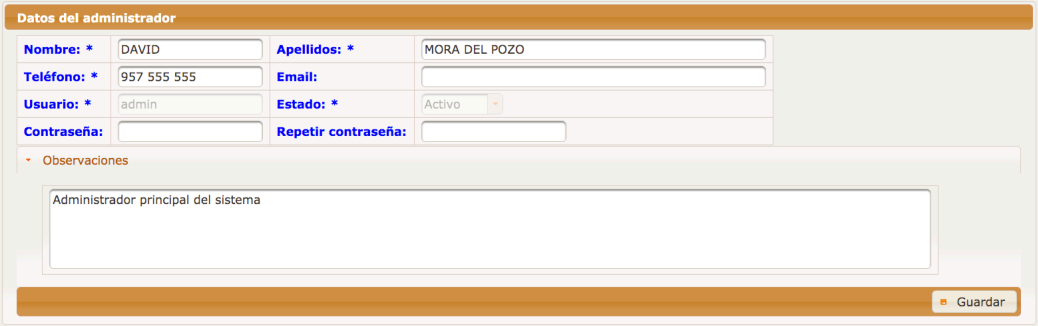


Ilustración 56 Edición de datos de administrador

Las estancias, entidad con pocas propiedades, se incorpora una variación que permite ver en pantalla las observaciones sin necesidad de acceder a la edición de los datos.

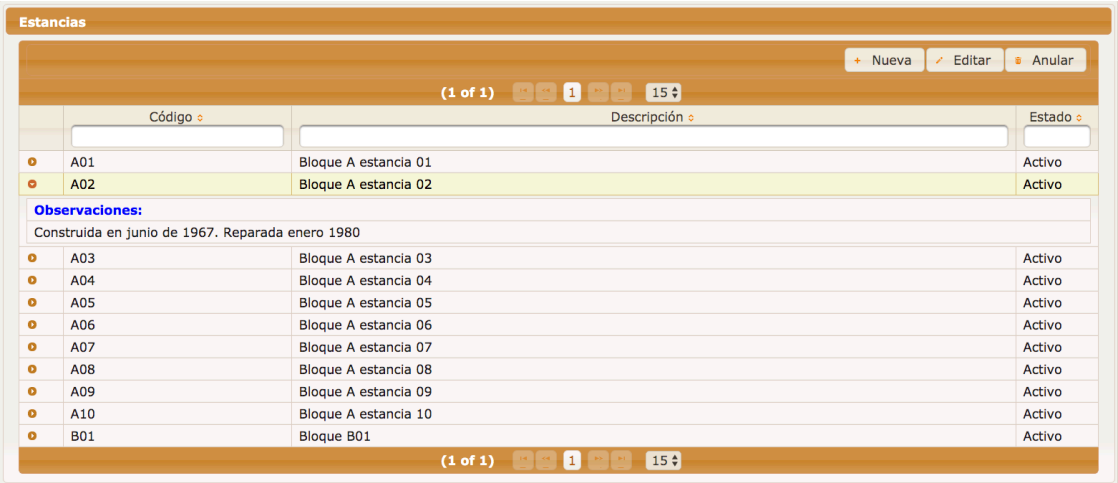


Ilustración 57 Consulta de estancias de la residencia

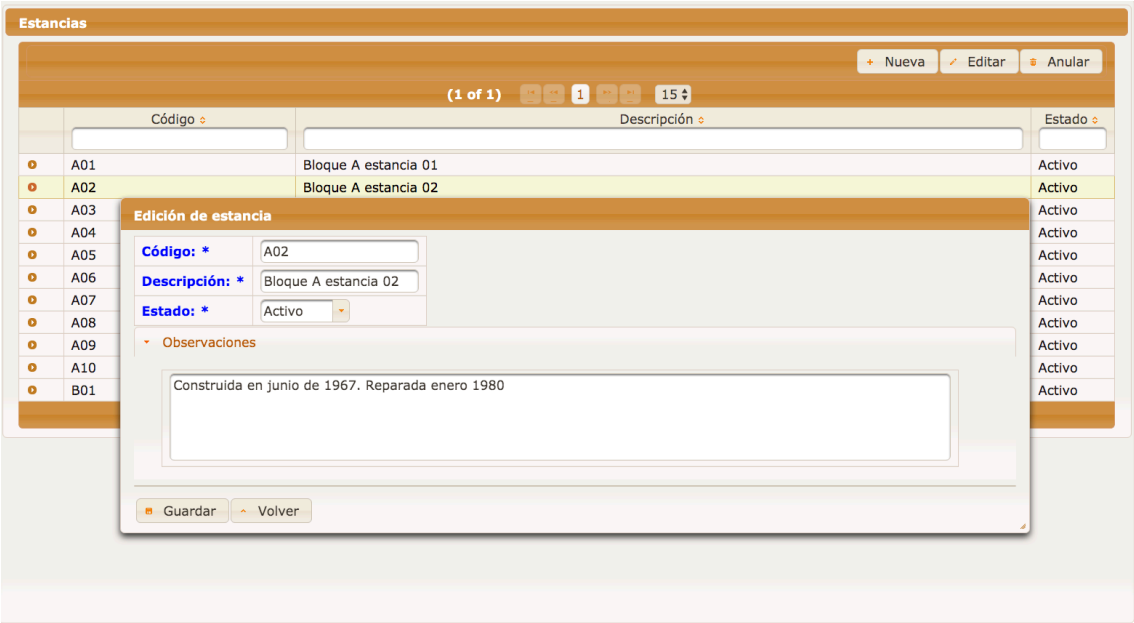


Ilustración 58 Edición de la estancia

A continuación, comprobaremos la similitud advertida en la consulta de mascotas. Cuando el usuario interactúe con alguna pantalla de la aplicación reconocerá rápidamente las opciones disponibles en las siguientes.

Ilustración 59 Consulta de mascotas

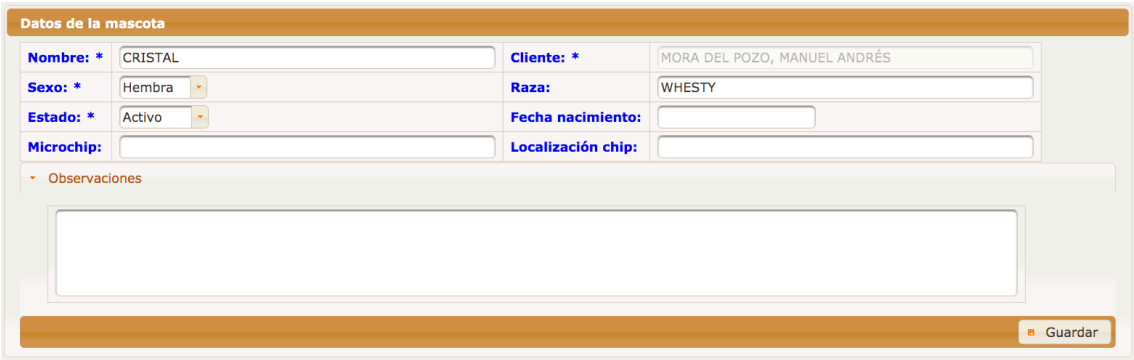


Ilustración 60 Edición de datos de mascota

De forma similar, la consulta de las reservas se ajustará al perfil. En caso de la consulta realizada por un usuario cliente, solo tendrá accesibles las suyas. Un administrador, sin embargo, tendrá acceso a todas. Se ve en la siguiente captura:

Ilustración 61 Consulta de reservas

Para simplificar el producto obtenido, la operatoria indicada será eliminar una reserva si se desea cambiar las fechas. Por tanto, la edición solo permitirá incorporar unas observaciones. En el caso de crear una nueva podrá seleccionar la mascota, en caso de disponer de varias.

Ilustración 62 Crear nueva reserva

El botón para reservar únicamente estará disponible si se cumplen las reglas de validación: Fechas consecutivas y superiores al día de hoy, no disponer de reserva en dichas fechas para la mascota seleccionada y existir disponibilidad de una estancia activa para las fechas indicadas. La reserva quedará confirmada en cuanto se confirmen los datos en pantalla.

A continuación tenemos una visión general del formulario de edición de cliente y su encuadre dentro de la plantilla general de la aplicación. Pueden ajustarse, en cada vista, la información visible en cada bloque:

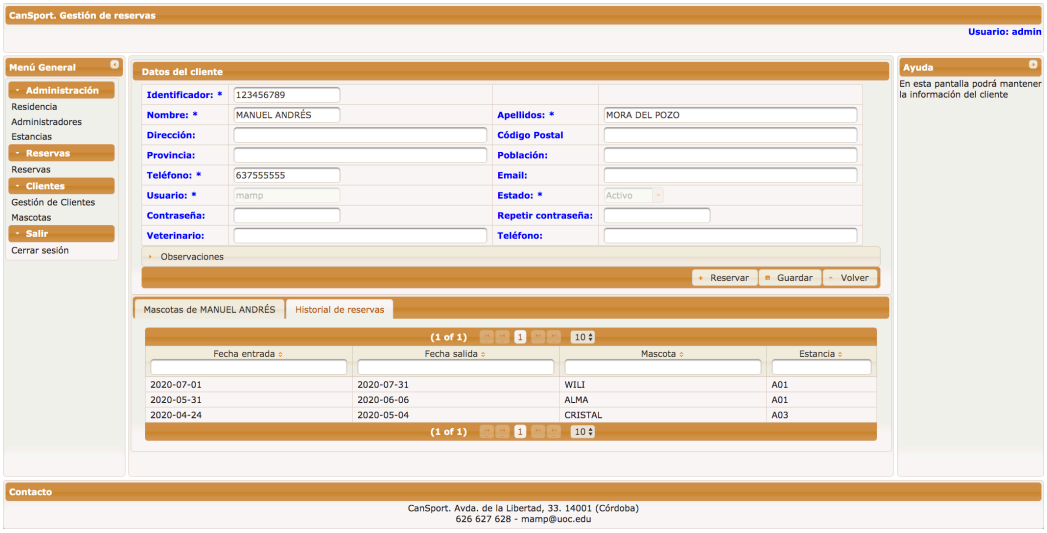


Ilustración 63 Pantalla general. usuario administrador

3.2 Mejoras sugeridas

El proyecto desarrollado cumple con los requisitos obtenidos en las historias de usuario pero está abierto para muchas posibles mejoras. Desde la incorporación de soluciones de comunicación online o en diferido, o bien, la visualización de calendarios en pantalla o la integración con las aplicaciones comerciales más habituales para organización y recordatorios.

Como gestor de reservas, definimos un módulo funcional que puede incorporarse a una suite, que agrupe soluciones a un sistema de información más ambicioso. Incluyendo una cartera mayor de servicios y un sistema común de facturación online. Un sistema adaptado a las necesidades integrales de cada organización con intereses similares y con un sistema de seguridad y acceso común.

Se puede aportar mayor personalización del interfaz a través de la incorporación de un configurador del aspecto visual a través de varios temas que proporciona Primefaces de forma gratuita, o incluso contratando alguno de pago. Incluso alternando estilos si se integra con algunos modos de configuración global de algunos sistemas, por ejemplo, el modo nocturno.

4 Conclusiones

El desarrollo de una aplicación a partir de una idea inicial siempre es una oportunidad de aplicar conocimientos en una alternativa más real que los escenarios encontrados en los temarios. Añade una motivación extra para aprender sobre elementos de software de libre distribución existentes y consultar el amplio foro de profesionales que comparten conocimientos y experiencias.

Sin embargo, si tuviéramos que destacar lecciones aprendidas en este periodo serían las relativas a la organización y la gestión del tiempo. Una planificación ágil, de tareas basadas en los siguientes pasos sencillos de realizar. Y fundamental, como no podría ser de otro modo, un trabajo constante y centrado en el cumplimiento de estas tareas.

Los objetivos planteados no eran excesivamente ambiciosos y la solución propuesta se originó en base a franjas temporales de la asignatura. Los periodos de cada PEC marcaron el calendario de realización en cada etapa. Y la planificación inicial se estimó para que fueran objetivos cumplibles en condiciones normales, sin sospechar que se convertirían en retos personales.

Quizás quepa destacar la situación vivida en este cuatrimestre de 2020. Circunstancias que afectaron a todos y en todos los aspectos que rodean el estudio y concentración necesarios para aportar la calidad que requiere un trabajo final de Grado. Sin embargo, superados los momentos más críticos de la pandemia vivida de cerca, se priorizaron los objetivos funcionales sobre las posibles mejoras relativas al interfaz y su estética.

La planificación, y su cumplimiento como objetivo personal, ha supuesto las marcas de tiempo y tareas fundamentales. Unas se ha realizado sin dificultad y su rápida finalización ha supuesto una ventaja temporal y alivio para otras que han requerido tiempo extra para adquirir conocimientos y seguridad en su aplicación.

Se han seguido las iteraciones incrementales sugeridas y se han observado las rutinas repetitivas o con opciones de generalización. Por lo que parte del trabajo realizado centrado en un primer caso de uso han significado un avance y adelanto en posterior desarrollo de los siguientes.

Se han descrito ciertas mejoras que podrían aportar al producto verdaderas ventajas competitivas con otros productos del mercado. Y se le pueden añadir opciones de seguridad y otras funcionalidades que proporcionan terceros sin incurrir en la necesidad de adquirir licencias. Y por supuesto, las relativas a la mejora de la interfaz para que sea más usable y responsiva en dispositivos móviles.

En definitiva, la aplicación supone un gran salto tecnológico para organizaciones de métodos obsoletos de gestión y de bajo prepuesto. Se incorpora al cliente como un usuario del sistema con acceso a información y el recurso sencillo de reserva en el momento.

5 Glosario

Arquitectura cliente servidor. Aplicado al diseño de software, básicamente disponemos de un elemento cliente que realiza peticiones y uno o varios proveedores de servicios dan respuesta.

Cliente ligero. En este caso hace referencia al software utilizado por el cliente para acceder al sistema. Permite a los usuarios interactuar con el servidor sin necesidad de capacidades altas de procesamiento.

Formularios CRUD. Siendo CRUD el acrónimo de crear, leer, actualizar y borrar (*create, read, update and delete,* en inglés) hace referencia a formularios que resuelven estas acciones. Muy comunes y de funcionalidades básicas similares en los mantenimientos de datos.

Framework. Usado en el documento, en referencia a componentes desarrollados por terceros y que incorporan funcionalidades resueltas. Por ejemplo, la visualización de una consulta de registros con opciones de filtrado y ordenación, o los propios campos editables en pantalla.

JPQL (*Java Persistence Query Language*). Es un lenguaje de consulta basado en SQL que usa la caché de persistencia para hacer consultas en una base de datos.

Metodología ágil. En referencia al desarrollo de software con un enfoque dinámico, donde los requisitos y las soluciones aportadas evolucionan según las necesidades del proyecto. Se establecen periodos para una planificación a cortos plazos, orientados a conseguir producto de valor al finalizarlos. Permite una gran organización del trabajo y de equipos involucrados.

Reglas de integridad. Es una característica de las bases de datos relacionales. Proporcionan seguridad cuando se definen relaciones entre tablas de información. En dichas propiedades se pueden establecer formas de actuar cuando existe un alta, modificación o eliminado de un dato referenciado.

Responsive. Se usa para indicar la adaptabilidad del diseño a la pantalla donde se muestra. Esta característica nos permite ver de forma óptima en dispositivos de pantalla móvil, con menor tamaño o diferente forma. Adaptando la posición y tamaño de la información mostrada.

Tema. El uso de este término en la documentación hace referencia a la capacidad de agrupar una serie de configuraciones de fuentes de texto, colores y efectos visuales en los componentes de software (*frameworks*). Más información en https://www.primefaces.org/showcase/index.xhtml

6 Bibliografía

[1] UOC. 2019. **PROFESORADO DE INFORMÁTICA Y MULTIMEDIA.** TUTORIAL DE LABORATORIO DE PDS: *PROYECTO DE DESARROLLO DE SOFTWARE. LABORATORIO VERSIÓN 8.0E. SEPTIEMBRE 2019*

[2] **TIOBE INDEX FOR FEBRUARY 2020.** [Consultado marzo de 2020] https://www.tiobe.com/tiobe-index/

[3] **PRIMETEK.** [Consultado mayo de 2020]. **PRIMEFACES SHOWCASE.** HTTPS://WWW.PRIMEFACES.ORG/SHOWCASE/INDEX.XHTML

[4] **STACKOVERFLOW.COM.** [Consultado mayo de 2020].. **QUESTIONS.** HTTPS://STACKOVERFLOW.COM/QUESTIONS

[5] **ORACLE HELP CENTER.** [Consultado mayo de 2020]. **ECLIPSELINK, JPA, AND ORACLE COHERENCE** HTTPS://DOCS.ORACLE.COM/EN/MIDDLEWARE/OSA/12.2.1.3/APPLICATIONS-DEVELOP/ECLIPSELINK-JPA-AND-ORACLE-COHERENCE.HTML#GUID-DCB2AFE4-3ABA-4912-806A-E6C2B4780CAB

[6] SonarSource. [Consultado mayo de 2020]. **FIX ISSUES BEFORE THEY EXIST** https://www.sonarlint.org/

7 Anexos

7.1 Implantación

7.1.1 Base de datos

La base de datos utilizada en el desarrollo es PostGreSQL. Se facilitan en la carpeta *sql* los ficheros necesarios para crear esquema, tablas y datos iniciales de ejemplo. También pueden encontrarse en las carpeta del subproyecto *CanSport-EJB*: */src/main/resources/META-INF/sql/*

El orden correcto de ejecución es:

1. Crear el esquema: ***cansport.create_schema.sql***
2. Creación de secuencias y tablas: ***cansport.create_tables.sql***
3. Inserción de datos de ejemplo: ***cansport.example_data.sql***

7.1.2 Desplegando la aplicación

El proyecto está compuesto por varios subproyectos atendiendo al diseño multicapa. Se ha utilizado Maven para la configuración del proyecto, de dependencias e instrucciones para su despliegue.

Si hemos importado el proyecto y deseamos que nuestro IDE realice el despliegue del fichero *ear* a nuestro servidor, deberemos indicarle donde realizarlo. Si editamos el fichero pom.xml del subproyecto CanSport-EAR deberemos buscar el tag server.deploy.dir dentro de las propiedades (esquema properties) e indicar el path de nuestra configuración local.

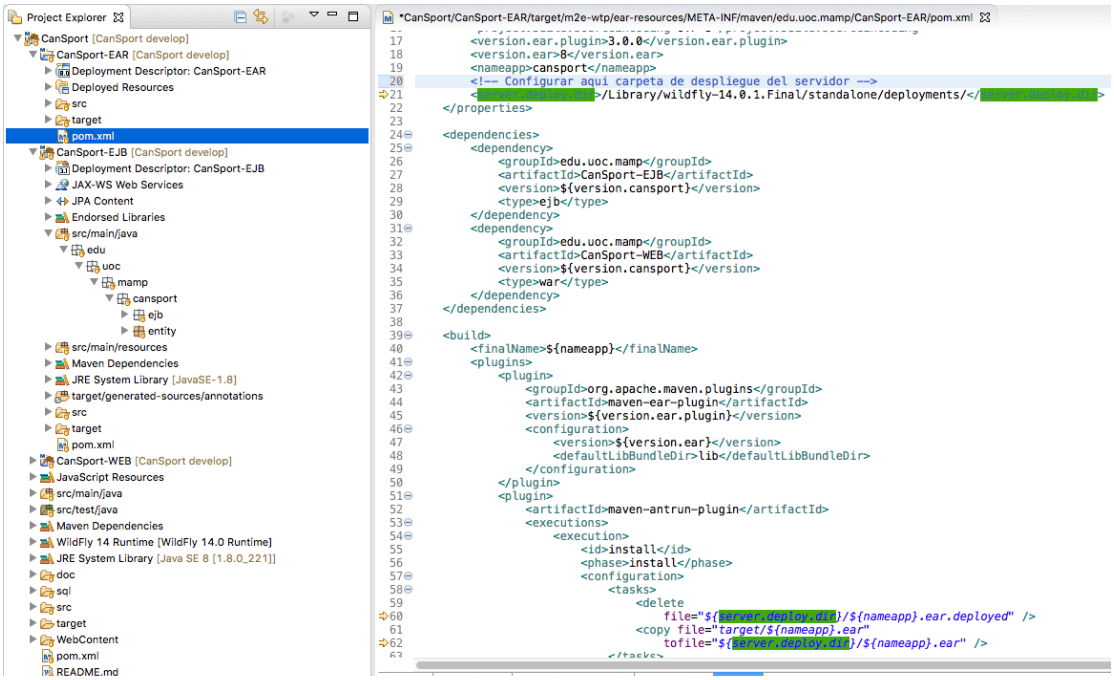


Ilustración 64 Ejemplo de edición propiedad de carpeta de despliegue

Observamos en la imagen, como ejemplo, la configuración en la instalación del entorno del equipo donde se ha desarrollado. Otro, sería si lo tuviéramos instalado en la unidad D, sería algo como: D:\wildfly-14.0.1.Final\standalone\deployments.

Una vez completada la configuración, ejecutaremos “maven install” (botón derecho de ratón sobre el proyecto, opción “run as”) para el compilado y puesta en producción en el servidor WildFly (JBoss) en la ubicación que hemos editado.

Podemos comprobar el correcto despliegue en servidor local a través de la consola de administrador:
http://127.0.0.1:9990/console/index.html#deployments;path=deployment~dply-cansportear

Accederemos a la aplicación a través de la *URL* local:
http://127.0.0.1:8080/CanSport-WEB/

Podemos acceder con perfil de **administrador** con las credenciales: admin/admin, y como **cliente** de prueba: mamp/mamp.