Organizador de Garaje

Miguel Ángel Orgaz Salcedo
Grado en Ingeniería Informática

Caballe Llobet, Santi
Oller Arcas, Antoni

Junio del 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	Organizador de Garaje
Nombre del autor:	Miguel Ángel Orgaz Salcedo
Nombre del consultor:	Antoni Oller Arcas
Fecha de entrega (mm/aaaa):	06/2022
Área del Trabajo Final:	JavaEE
Titulación:	*Grado en Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

Este Trabajo de fin de grado comprende el análisis, diseño e implementación de una aplicación web, utilizando la tecnología Java EE. Para ello, se abarcan los distintos conocimientos que se han conseguido obtener a lo largo del grado en Ingeniería Informática.

La aplicación desarrollada, consistí en un organizador de garaje, pues suele ser habitual tener muchas cajas sin catalogar, en las distintas estanterías del mismo. Y esto hace que sea difícil saber lo que se tiene almacenado y hace que la tarea de encontrar algo requiera mucho tiempo. Además, debido a que se ha tenido en cuenta que hay quien cuenta también con este mismo problema en distintas estancias de la casa, se ha ampliado la aplicación, para que se puedan gestionar diferentes habitaciones.

Se ha utilizado SpringBoot para realizar la aplicación, lo que hace que, a la hora de implementar, el programador se pueda centrar en dicha tarea, pues los despliegues quedan configurados y cuenta con su propio servidor para arrancar la aplicación.

Abstract (in English, 250 words or less):

This final degree work includes the analysis, design and implementation of a web application, using Java EE technology. For this, the different knowledge that have been obtained throughout the degree in Computer Engineering are covered.

The application developed, consisted of a garage organizer, because it is common to have many uncategorized boxes on the shelves of the garage. And this makes it difficult to know what is stored and makes the task of finding something time consuming. In addition, because it has been taken into account that some people also have this same problem in different rooms of the house, the application has been extended, so that different rooms can be managed.

SpringBoot has been used to make the application, which means that at the time of implementation, the programmer can focus on this task, because the deployments are configured and has its own server to start the application.

Translated with www.DeepL.com/Translator (free version)

Palabras clave (entre 4 y 8):
JavaEE, Spring, SpringBoot, SpringSecurity, Thymeleaf, MySQL, MVC, Organizador

Índice

1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 1
1.3 Enfoque y método seguido 2
1.4 Planificación del Trabajo 2
1.5 Breve sumario de productos obtenidos 4
1.6 Breve descripción de los otros capítulos de la memoria 4
2. Modelo de caso de uso y actores 5
3. Fichas de caso 6
4. Diagramas de clases principales 12
5. Diseño relacional de la base de datos 13
6. Diagrama de arquitectura 13
7. Prototipo 30
8. Implementación 38
9. Instrucciones de despliegue 44
10. Conclusiones 47
11. Glosario 48
12. Anexos 49

Lista de figuras

Ilustración 1 - Planificación 3
Ilustración 2 - Diagrama casos de usos 5
Ilustración 3 - Diagrama de clases 12
Ilustración 4 - Diagrama entidad/relación 13
Ilustración 5 - Punto de vista de la computación 14
Ilustración 6 - Interfaces Profile 15
Ilustración 7 - Interfaces Administration 16
Ilustración 8 - Interfaces ShelfManagement 17
Ilustración 9 - Interfaces ObjectManagement 18
Ilustración 10 - ProfilePresentacion Diagrama Componentes Inicial 19
Ilustración 11 - Diagrama componentes ProfilePresentation 19
Ilustración 12 - Diagrama componentes AdministrationPresentation 20
Ilustración 13 - Diagrama componentes ShelfManagementPresentation 21
Ilustración 14 - Diagrama componentes ObjectManagementPresentation 22
Ilustración 15 - Capa de integración 23
Ilustración 16 - Estructura Spring MVC 24
Ilustración 17 - Capa presentación JEE Administration 24
Ilustración 18 - Capa presentación JEE Profile 25
Ilustración 19 - Capa presentación JEE ShelfManagement 26
Ilustración 20 - Capa presentación JEE ObjectManagement 27
Ilustración 21 - Capa negocio JEE Profile 28
Ilustración 22 - Capa negocio JEE Administration 28
Ilustración 23 - Capa negocio JEE ShelfManagement 28
Ilustración 24 - Capa negocio JEE ObjectManagement 29
Ilustración 25 - Capa integración JEE 29
Ilustración 26 - Prototipo - Registro 30
Ilustración 27 - Prototipo - Login 30
Ilustración 28 - Prototipo - Estancias 31
Ilustración 29 - Prototipo - Nueva estancia 31
Ilustración 30 - Prototipo - Estanterías 31
Ilustración 31 - Prototipo - Nueva estantería 32
Ilustración 32 - Prototipo - Detalle Estantería 32
Ilustración 33 - Prototipo - Cajas sin colocar 33
Ilustración 34 - Prototipo - Nueva caja 33
Ilustración 35 - Prototipo - Categorías 34
Ilustración 36 - Prototipo - Nueva categoría 34
Ilustración 37 - Prototipo - Cajas 34
Ilustración 38 - Prototipo - Detalle de caja 35
Ilustración 39 - Prototipo - Objetos 35
Ilustración 40 - Prototipo - Insertar objeto en caja 36
Ilustración 41 - Prototipo - Nuevo objeto 36
Ilustración 42 - Prototipo - Modificar datos usuario 36
Ilustración 43 - Prototipo - Reactivar usuarios 37
Ilustración 44 - Implementación - Spring initializr 38
Ilustración 45 - Implementación - MySQL Workbench 39
Ilustración 46 - Implementación - Estructura 41

Ilustración 47 - Implementación - Controller 42
Ilustración 48 - Implementación - Security 42
Ilustración 49 - Implementación - Services 43
Ilustración 50 - Código QR 44
Ilustración 51 - Instrucciones - Importar 45
Ilustración 52 - Instrucciones - Ejecutar 46
Ilustración 53 - Instrucciones - Login 46

1. Introducción

1.1 Contexto y justificación del Trabajo

En el mundo contemporáneo en el que vivimos, la sociedad es cada vez más y más consumista, por lo que la gente va comprando más bienes materiales que acaban almacenados en los distintos rincones de los que se dispone, y que, por norma general, aquellos que tienen un garaje, sótano, buhardilla o trastero, suelen tenerlos llenos de una gran cantidad de objetos de los que no se suelen acordar hasta el día en el que se necesitan. Y es entonces cuando comienza la ardua tarea de encontrar aquello entre las distintas cajas que hay en las múltiples estanterías.

Es por ello por lo que surge la necesidad de poder abordar esta problemática, y por lo que se ha decidido desarrollar una aplicación web que permitirá tener organizado digitalmente todo aquello que se tenga, tanto en cajas, como en las distintas baldas de las estanterías o armarios de los que se disponga. Para posteriormente poder realizar una búsqueda de un objeto concreto o visualizar lo que existe en estos lugares sin tener que desplazarse hasta ellos y tener que abrir caja por caja hasta encontrar aquel objeto que se estuviera buscando. Así se evitará tener que estar apuntando en papel el contenido de cada caja, o el deber tener una gran memoria para acordarse de todo lo que se posee.

1.2 Objetivos del Trabajo

Con todo esto, y para realizar el TFG, y poder poner en práctica todo lo aprendido durante estos años de estudios del grado en ingeniería informática, tendremos una aplicación JEE, donde nuestro sistema tendrá un apartado para registrar a los distintos usuarios, que posteriormente podrán hacer uso de la aplicación, donde cada uno de los usuarios podrá ver únicamente sus bienes. Además, cada usuario podrá crear las habitaciones que necesite, ya que podría darse el caso de que una persona tenga estanterías en el garaje, pero también en la buhardilla, y dentro de estas habitaciones se tendrán almacenadas las distintas estanterías de las que se disponga, teniendo un identificador único para ellas que servirá para que cada uno pueda incorporar dicho identificador a su estantería física.

Además, cada una de las estanterías, estará formada por distintas baldas, en las que se guardarán las cajas, que contendrán uno o varios objetos. Así, estas cajas tendrán un identificador, con el que el usuario podría generar unas pegatinas para poner en las cajas y así consultarlas posteriormente. En un principio se creará un código QR que se podrá visualizar al ver el detalle de la caja.

Los distintos objetos se van a categorizar, por lo que existirán distintas categorías, que tendrán un nombre único, y éstas, a su vez, podrán tener subcategorías. Así podríamos tener la categoría de moda y complementos, y dentro de ésta podríamos tener calzado, ropa, bolsos, … Tendremos una serie de categorías que el administrador creará y que serán visibles para todos los usuarios, sin poder modificarse por parte de éstos, aunque sí podrán hacerlas no visibles, y además, cada usuario tendrá la posibilidad de crear sus propias categorías y subcategorías.

Para cada uno de los objetos, guardaremos un nombre, descripción y fotografía del mismo si se deseara. Al igual que podemos incorporar una foto para nuestros objetos, podremos hacer lo mismo para poder visualizar el interior de las cajas.

1.3 Enfoque y método seguido

Para este TFG se desarrolla un producto nuevo, pues el objetivo es poner en práctica lo que se ha ido aprendiendo a lo largo de estos cursos del grado en ingeniería informática. Aprendiendo así a desarrollar una aplicación desde cero basándonos en la tecnología JEE, que es la materia de este trabajo.

1.4 Planificación del Trabajo

Podríamos diferencias cuatro componentes en nuestra aplicación, que nos ayudarán a la hora de separar las distintas tareas y poder planificar mejor.

• Administración: donde los administradores gestionarán las categorías por defecto que se mostrarán a todos los usuarios. Y además podrán gestionar los usuarios, ya que podría darse el caso de que un usuario estuviera bloqueado y se necesitara reactivar.

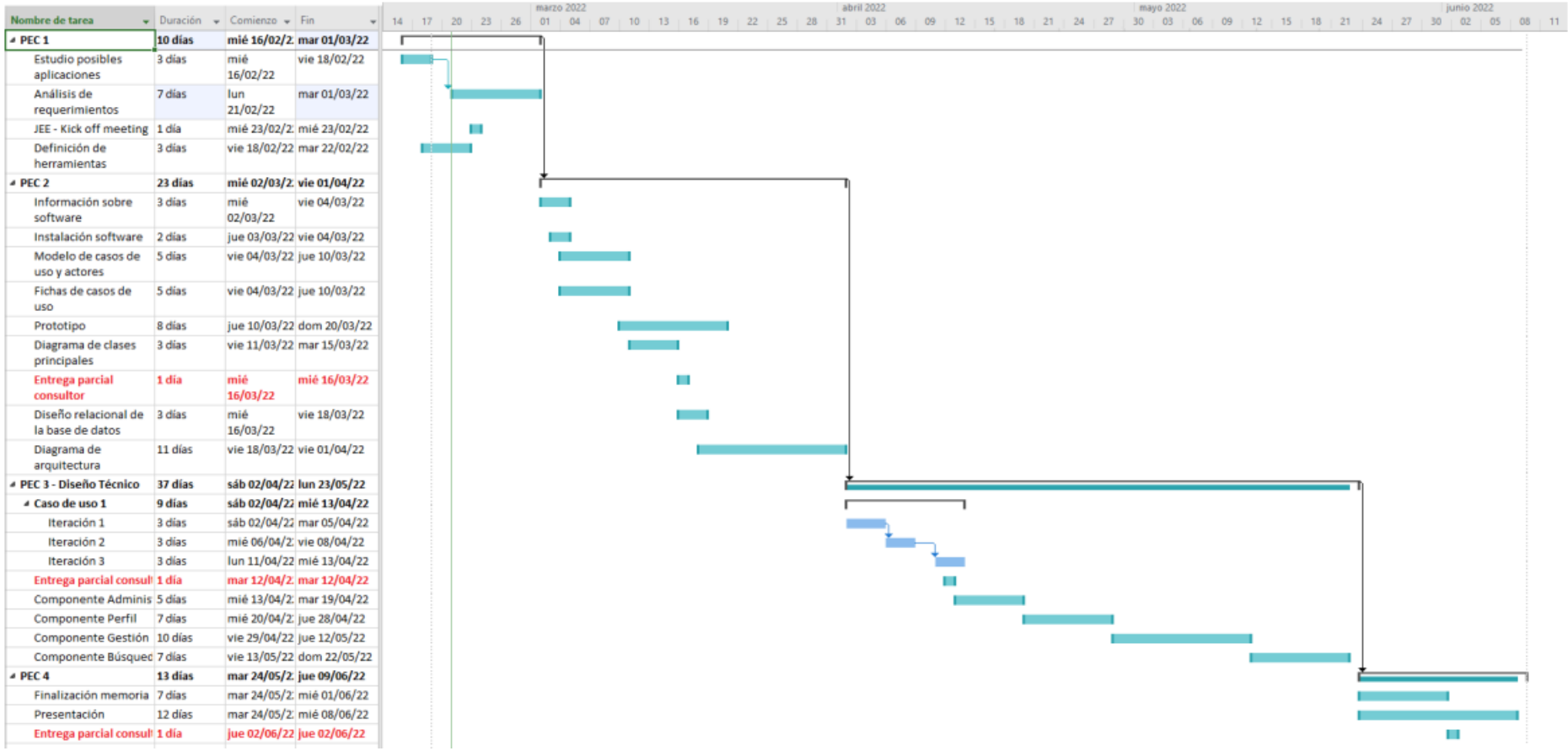
• Perfil: que contendrá el registro y acceso a la aplicación, y donde los usuarios accederán a sus datos para poder modificarlos. También se gestionará las categorías propias de cada usuario.

• Gestión: Se gestionan las habitaciones, las estanterías que en ellas se encuentra, y se crearán las cajas y los bienes que el usuario tiene en cada una de ellas o en las propias estanterías.

• Búsqueda: Donde se podrá realizar la búsqueda de una caja en concreto, o buscar un objeto para poder localizarlo entre las estanterías del usuario.

La planificación se ha realizado siguiendo las fechas de entrega de las distintas PEC.

Ilustración 1 - Planificación



Nombre de tarea	Duración	Comienzo	Fin
PEC 1	**10 días**	**mié 16/02/2**	**mar 01/03/22**
Estudio posibles aplicaciones	3 días	mié 16/02/22	vie 18/02/22
Análisis de requerimientos	7 días	lun 21/02/22	mar 01/03/22
JEE - Kick off meeting	1 día	mié 23/02/2	mié 23/02/22
Definición de herramientas	3 días	vie 18/02/22	mar 22/02/22
PEC 2	**23 días**	**mié 02/03/2**	**vie 01/04/22**
Información sobre software	3 días	mié 02/03/22	vie 04/03/22
Instalación software	2 días	jue 03/03/22	vie 04/03/22
Modelo de casos de uso y actores	5 días	vie 04/03/22	jue 10/03/22
Fichas de casos de uso	5 días	vie 04/03/22	jue 10/03/22
Prototipo	8 días	jue 10/03/22	dom 20/03/22
Diagrama de clases principales	3 días	vie 11/03/22	mar 15/03/22
Entrega parcial consultor	1 día	mié 16/03/22	mié 16/03/22
Diseño relacional de la base de datos	3 días	mié 16/03/22	vie 18/03/22
Diagrama de arquitectura	11 días	vie 18/03/22	vie 01/04/22
PEC 3 - Diseño Técnico	**37 días**	**sáb 02/04/22**	**lun 23/05/22**
Caso de uso 1	**9 días**	**sáb 02/04/22**	**mié 13/04/22**
Iteración 1	3 días	sáb 02/04/22	mar 05/04/22
Iteración 2	3 días	mié 06/04/2	vie 08/04/22
Iteración 3	3 días	lun 11/04/22	mié 13/04/22
Entrega parcial consult	1 día	mar 12/04/2	mar 12/04/22
Componente Adminis	5 días	mié 13/04/2	mar 19/04/22
Componente Perfil	7 días	mié 20/04/2	jue 28/04/22
Componente Gestión	10 días	vie 29/04/22	jue 12/05/22
Componente Búsqued	7 días	vie 13/05/22	dom 22/05/22
PEC 4	**13 días**	**mar 24/05/2**	**jue 09/06/22**
Finalización memoria	7 días	mar 24/05/2	mié 01/06/22
Presentación	12 días	mar 24/05/2	mié 08/06/22
Entrega parcial consult	1 día	jue 02/06/22	jue 02/06/22

1.5 Breve sumario de productos obtenidos

Se ha obtenido una aplicación web desarrollada con la tecnología JEE, que nos permite realizar una gestión de las estanterías que tenemos en nuestros hogares. En los siguientes capítulos se entrará más en detalle sobre la misma.

1.6 Breve descripción de los otros capítulos de la memoria

Los siguientes ocho capítulos de la memoria, tienen el análisis, diseño e implementación de la aplicación que se ha ido desarrollando.

En cada uno de estos capítulos se irá entrando en detalle de las decisiones tomades para poder realizar el trabajo de la mejor forma posible.

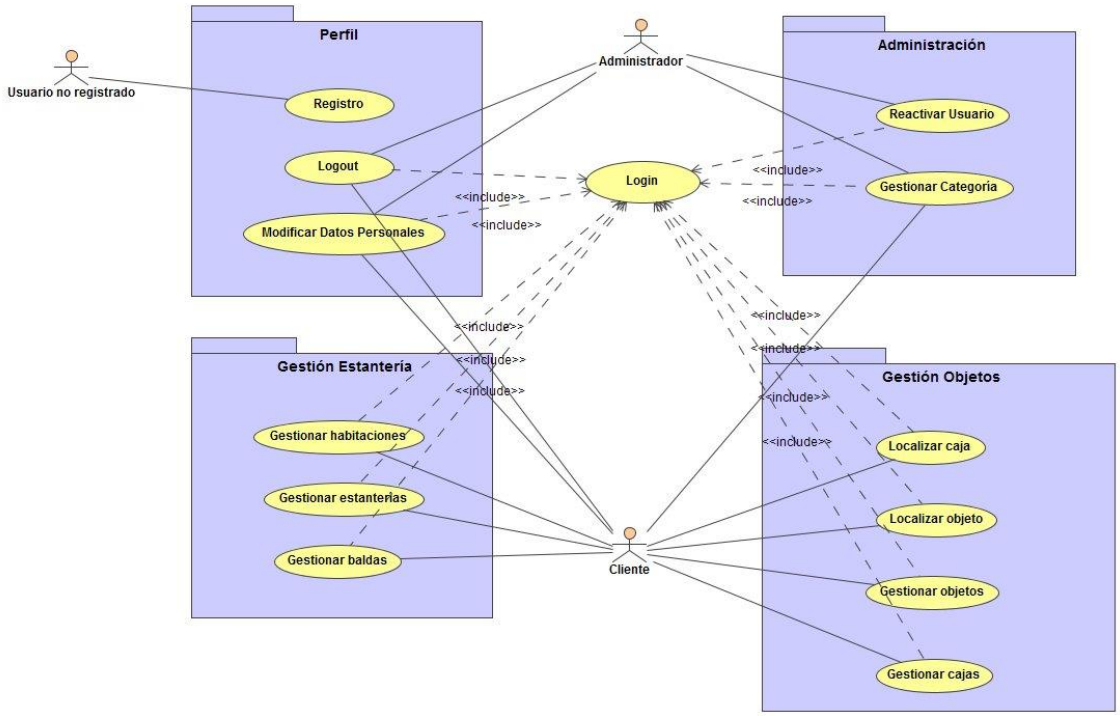
Además de estos capítulos, hay un capítulo de conclusiones en el que comento cómo ha sido la experiencia de crear una aplicación desde cero, así como un pequeño apartado de qué coses se han dejado para continuar mejorando la aplicación en un futuro.

Finalmente, hay un capítulo con un glosario con los términos más importantes de la memoria.

2. Modelo de caso de uso y actores

Tras un estudio de las distintas funcionalidades con las que debe contar nuestra aplicación de organizador de garajes, se ha creado el siguiente diagrama de casos de usos que detallaremos a continuación:

Ilustración 2 - Diagrama casos de usos



Como podemos observar, tendremos 3 actores principales

- El usuario no registrado, que lo único que conseguirá hacer en nuestra aplicación es registrarse.

- El cliente, que es un usuario que se ha registrado para poder utilizar la aplicación y gestionar su garaje.

- El administrador, que se encargará de administrar ciertas partes de la aplicación, así como de solventar problemas con las cuentas de los clientes.

Se han agrupado las funcionalidades de listar, consultar, crear, modificar y eliminar, de las distintas entidades, y por eso se ha puesto simplemente “Gestionar x”. En el capítulo posterior, tenemos un detalle de los distintos casos de uso que tenemos.

3. Fichas de caso

Caso de uso: Registrar usuario

Actor principal: Usuario no registrado.

Precondición: Es un usuario que no tiene cuenta en el sistema

Garantías en caso de éxito: El usuario quedaría registrado en el sistema, por lo que desde ese momento podría acceder con el usuario y contraseña creado.

Flujo principal:
1. El usuario pulsa sobre el botón de registro.
2. El sistema muestra el formulario para el registro.
3. El usuario rellena el email, el nombre de usuario y su contraseña y confirma el registro.
4. El sistema informa de que se ha registrado y vuelve a la pantalla principal para logarse.

Flujo alternativo:
4.a. Se produce algún error durante el proceso de registro.
4.a.1 El sistema informa del error.
4.a.2 Volvemos al paso 3.

Caso de uso: Login

Actor principal: Usuario no autenticado.

Precondición: Es un usuario que se había registrado en el sistema

Garantías en caso de éxito: El usuario quedaría logado en el sistema, teniendo acceso a todos los casos de uso del actor cliente.

Flujo principal:
1. El usuario accede a la página principal.
2. El sistema muestra el formulario para el login.
3. El usuario rellena el nombre de usuario y contraseña y pulsa en aceptar.
4. El sistema muestra la pantalla de la gestión de habitaciones

Flujo alternativo:
4.a. Se produce algún error durante el proceso de login.
4.a.1 El sistema informa del error.
4.a.2 Volvemos al paso 3.
4.b. El usuario ha introducido mal los datos 5 veces.
4.b.1 El sistema informa del error bloqueando al usuario
4.b.2 El sistema vuelve al paso 1.

Caso de uso: Modificar datos personales

Actor principal: Cliente y administrador.

Precondición: Es un usuario que se había registrado en el sistema, o el propio administrador

Garantías en caso de éxito: El sistema permite modificar los datos del cliente.

Flujo principal:
1. El usuario accede a la gestión del usuario.
2. El sistema muestra el formulario con los datos del usuario, donde permite modificar el email y la contraseña.
3. El usuario rellena los datos que desea modificar y pulsa en aceptar
4. El sistema muestra un aviso indicando que los datos se han modificado correctamente

Flujo alternativo:
1.a. Es un administrador el que accede a la gestión de usuarios.
1.a.1 El sistema muestra un listado con los clientes existentes
1.a.2 El administrador selecciona uno de los clientes.
1.a.3 Vamos al paso 2.

Caso de uso: Reactivar usuario

Actor principal: Administrador.

Precondición: Es un administrador quien entra al sistema y existen usuarios con su usuario bloqueado

Garantías en caso de éxito: El sistema desbloquea el acceso del usuario

Flujo principal:
1. El administrador accede a Reactiva Usuarios
2. El sistema muestra un listado con aquellos usuarios que están bloqueados
3. El administrador pulsa en desbloquear sobre los clientes que desea.
4. El sistema informa de que se han desbloqueado los usuarios

Caso de uso: Gestionar habitaciones

Actor principal: Cliente.

Precondición: El usuario/cliente debe haberse logado en el sistema.

Garantías en caso de éxito: El sistema permite consultar las habitaciones que posee, así como crear nuevas habitaciones, editar y eliminar las existentes.

Flujo principal:
1. El usuario accede a Gestión de habitaciones
2. El sistema muestra un listado con las habitaciones disponibles
3. El usuario pulsa en crear una nueva habitación
4. El sistema muestra el formulario de alta de habitaciones.
5. El usuario rellena el nombre de la habitación y la descripción y confirma el alta.
6. El sistema informa del alta y muestra de nuevo el listado de habitaciones.
7. El usuario pulsa sobre una habitación
8. El sistema muestra información sobre la habitación.
9. El usuario pulsa sobre modificar habitación.
10. El sistema muestra el formulario para modificar el nombre y la descripción de la habitación.
11. El cliente rellena los datos y confirma la modificación.
12. El sistema informa de que se ha modificado
13. El usuario pulsa en el botón eliminar habitación.
14. El sistema pide confirmación.
15. El usuario confirma.
16. El sistema informa de la eliminación de la habitación y muestra el listado de habitaciones actualizado.

Flujo alternativo:
7.a. El usuario pulsa el icono modificar habitación.
7.a.1 Vamos al paso 10.
15.a. El usuario cancela la operación.
15.a.1 El sistema continúa mostrando el listado de habitaciones.

Caso de uso: Gestionar estanterías

Actor principal: Cliente.

Precondición: El usuario/cliente debe haberse logado en el sistema y ha seleccionado una habitación.

Garantías en caso de éxito: El sistema permite consultar las estanterías que posee, así como crear nuevas estanterías, editar y eliminar las existentes.

Flujo:
El caso de uso de gestionar estanterías sería prácticamente igual al de Gestionar habitaciones.

Caso de uso: Gestionar baldas

Actor principal: Cliente.

Precondición: El usuario/cliente debe haberse logado en el sistema y ha seleccionado una estantería.

Garantías en caso de éxito: El sistema permite consultar las baldas que posee la estantería, así como crear nuevas baldas, editar y eliminar las existentes.

Flujo:
El caso de uso de gestionar baldas sería prácticamente igual al de Gestionar habitaciones, solo que, en este caso, a la hora de crear una nueva balda, no se introducirán datos sobre la balda, sino que lo único que se hace es añadir una nueva balda a una estantería. Y la edición consiste en indicar si está llena o no.

Caso de uso: Gestionar objetos

Actor principal: Cliente.

Precondición: El usuario/cliente debe haberse logado en el sistema y haber seleccionado una estantería o una caja de una balda.

Garantías en caso de éxito: El sistema permite consultar los objetos que hay en la balda de una estantería o en una caja, así como crear nuevos objetos, editar y eliminar los existentes.

Flujo:
El caso de uso de gestionar objetos sería prácticamente igual al de Gestionar baldas, aunque en esta ocasión, los datos de entrada serán: nombre del objeto, marca, descripción, categoría a la que pertenece, y la posibilidad de introducir una imagen.

Caso de uso: Gestionar cajas

Actor principal: Cliente.

Precondición: El usuario/cliente debe haberse logado en el sistema.

Garantías en caso de éxito: El sistema permite consultar las cajas que hay en una balda de una estantería, así como crear nuevas cajas, editar y eliminar los existentes.

Flujo:
El caso de uso de gestionar objetos sería prácticamente igual al de Gestionar baldas, aunque en esta ocasión, los datos de entrada serán: descripción de la caja y la posibilidad de introducir una imagen.

Caso de uso: Gestionar categorías

Actor principal: Cliente y administrador

Precondición: El usuario/cliente debe haberse logado en el sistema.

Garantías en caso de éxito: El sistema permite consultar las categorías que posee, así como crear nuevas categorías, editar y eliminar las existentes.

Flujo principal:
1. El usuario accede a Gestión de categorías
2. El sistema muestra un listado con las categorías disponibles para el cliente
3. El usuario pulsa en crear una nueva categoría
4. El sistema muestra el formulario de alta de categorías.
5. El usuario rellena el nombre, categoría padre, si procede, y si será visible, y acepta.
6. El sistema informa del alta y muestra de nuevo el listado de categorías.
7. El usuario pulsa sobre una categoría existente
8. El sistema muestra información sobre la categoría.
9. El usuario pulsa sobre modificar categoría.
10. El sistema muestra el formulario para modificar el nombre y si es visible.
11. El cliente rellena los datos y confirma la modificación.
12. El sistema informa de que se ha modificado
13. El usuario pulsa en el botón eliminar categoría.
14. El sistema pide confirmación.
15. El usuario confirma.
16. El sistema informa de la eliminación de la categoría y muestra el listado de categorías actualizado.

Flujo alternativo:
3.a. El usuario selecciona una categoría existente.
3.a.1 El sistema muestra los datos de la categoría.
3.a.2 El usuario pulsa en crear subcategoría.
3.a.3 Vamos al paso 4.
15.a. El usuario cancela la operación.
15.a.1 El sistema continúa mostrando el listado de habitaciones.
16.b Existen objetos asociados a la categoría
16.b.1 El sistema informa de que existen objetos asociados a la categoría y que por ese motivo no se puede eliminar.
16.b.2 Vamos al paso 2

Caso de uso: Localizar caja

Actor principal: Cliente.

Precondición: El usuario/cliente debe haberse logado en el sistema y existen cajas creadas.

Garantías en caso de éxito: El sistema permite localizar una caja.

Flujo principal:
1. El usuario accede a la gestión de cajas
2. El sistema muestra un formulario para introducir los campos de búsqueda
3. El usuario puede rellenar el id, la descripción de la caja, la habitación, la estantería y la balda, y pulsa en Buscar.
4. El sistema muestra la caja o las cajas encontradas, indicando en qué habitación/estantería/balda se encuentra.

Caso de uso: Localizar objeto

Actor principal: Cliente.

Precondición: El usuario/cliente debe haberse logado en el sistema y existen objetos creados.

Garantías en caso de éxito: El sistema permite localizar un objeto.

Flujo principal:
1. El usuario accede a la gestión de objetos
2. El sistema muestra un formulario para introducir los campos de búsqueda
3. El usuario puede rellanar el identificador del objeto o de la caja, la marca, el nombre, la habitación, estantería y balda en la que se puede encontrar y pulsa en Buscar.
4. El sistema muestra el objeto u objetos encontrados, indicando en qué habitación/estantería/balda/caja se encuentra.

Caso de uso: Insertar caja en balda

Actor principal: Cliente.

Precondición: El usuario/cliente debe haberse logado en el sistema y existen cajas y estanterías en el sistema. El cliente ha entrado en una de las estanterías

Garantías en caso de éxito: El sistema permite insertar una caja dentro de una balda.

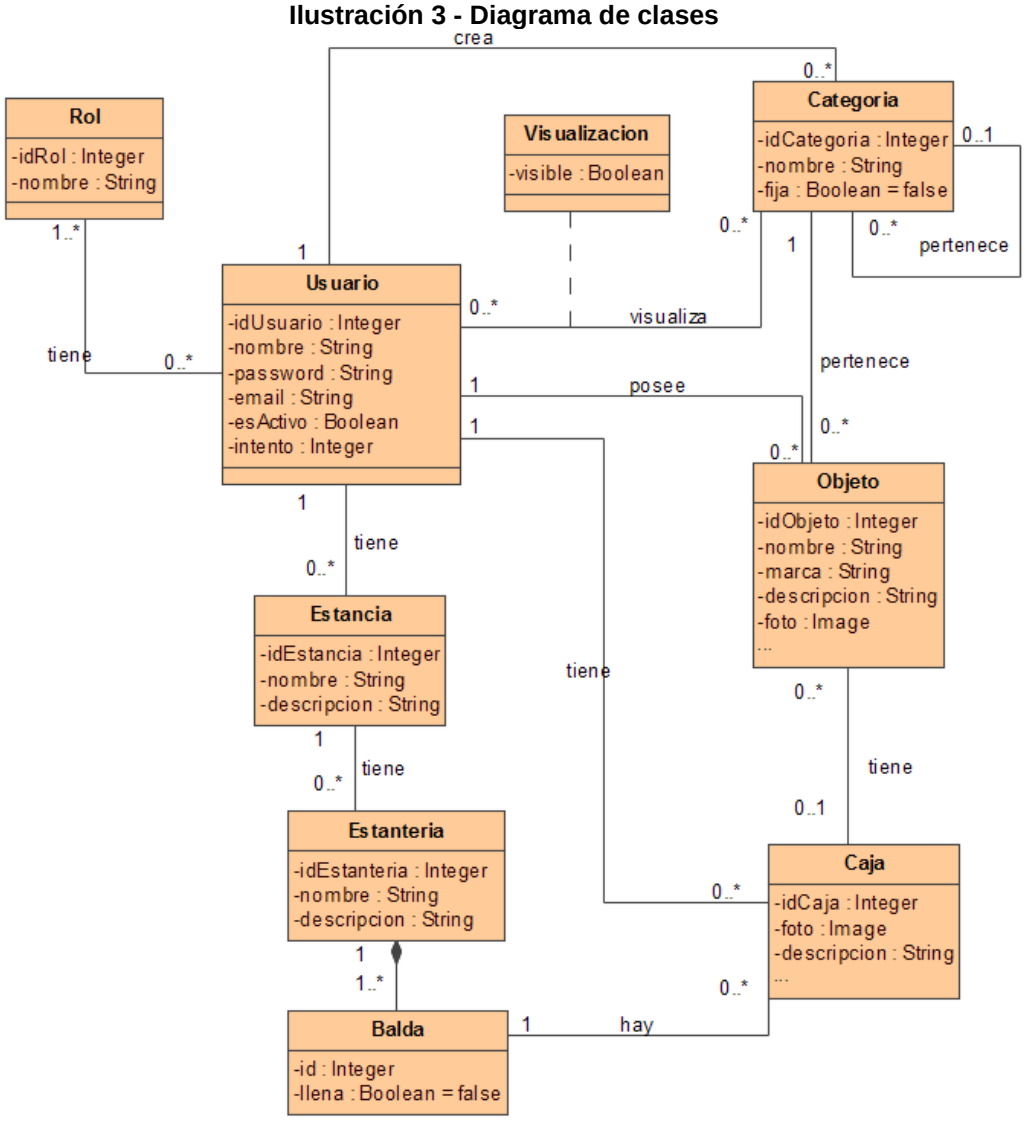
Flujo principal:

1. El usuario pulsa en un hueco libre de una balda
2. El sistema muestra un listado de cajas que no se encuentran en ninguna balda
3. El usuario pulsa en una de las cajas para insertarla en la balda
4. El sistema informa que el objeto se ha insertado en la caja y vuelve a la pantalla que muestra la estantería, donde ahora aparece la caja insertada en el hueco que se había seleccionado.

4. Diagramas de clases principales

Partiendo de los requisitos que se han fijado para la aplicación, y que vimos en los distintos casos de usos que se han descrito anteriormente, se ha podido crear el siguiente esquema invariante.

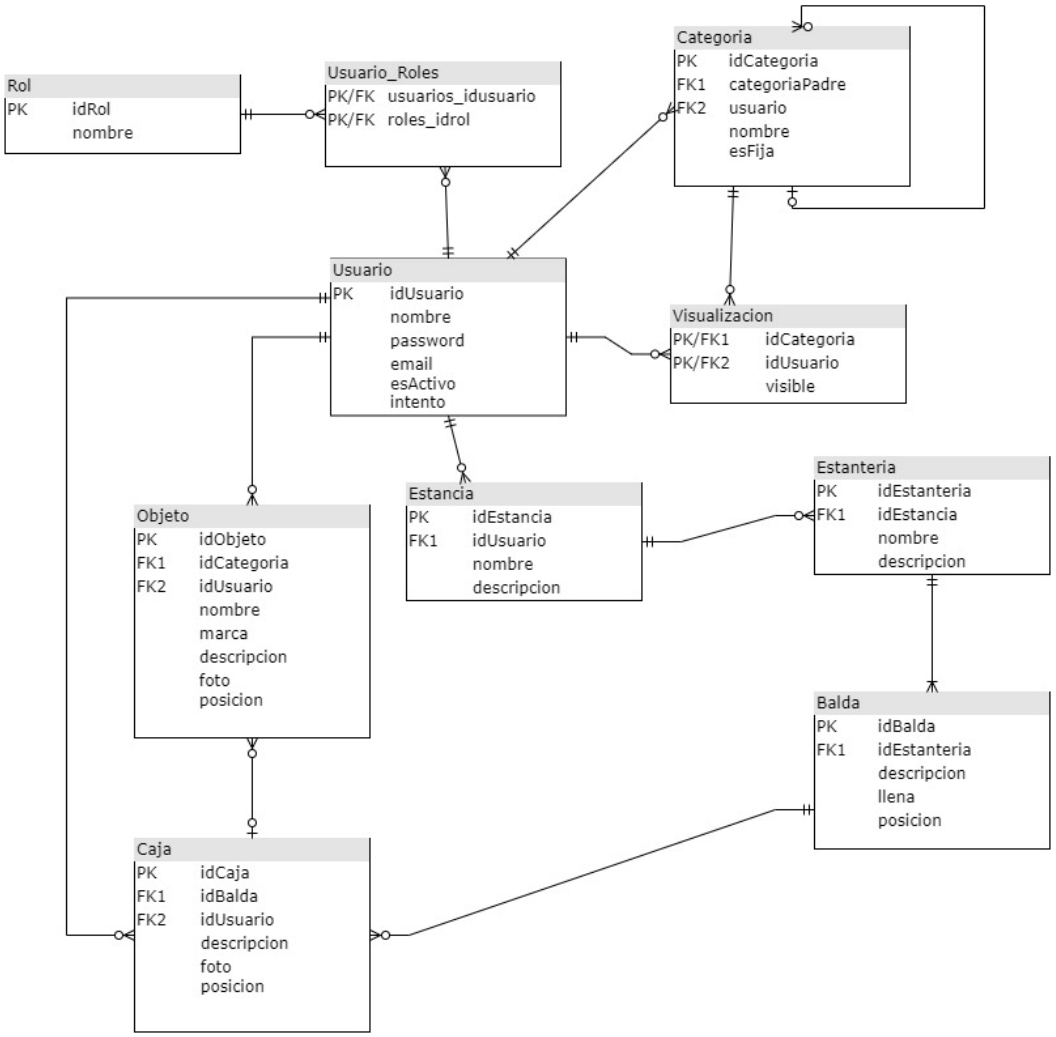
Ilustración 3 - Diagrama de clases



5. Diseño relacional de la base de datos

Una vez tenemos nuestro diagrama de clases, procedemos a realizar el diagrama entidad/relación, para mostrar las tablas de las que dispondrá nuestro sistema.

Ilustración 4 - Diagrama entidad/relación

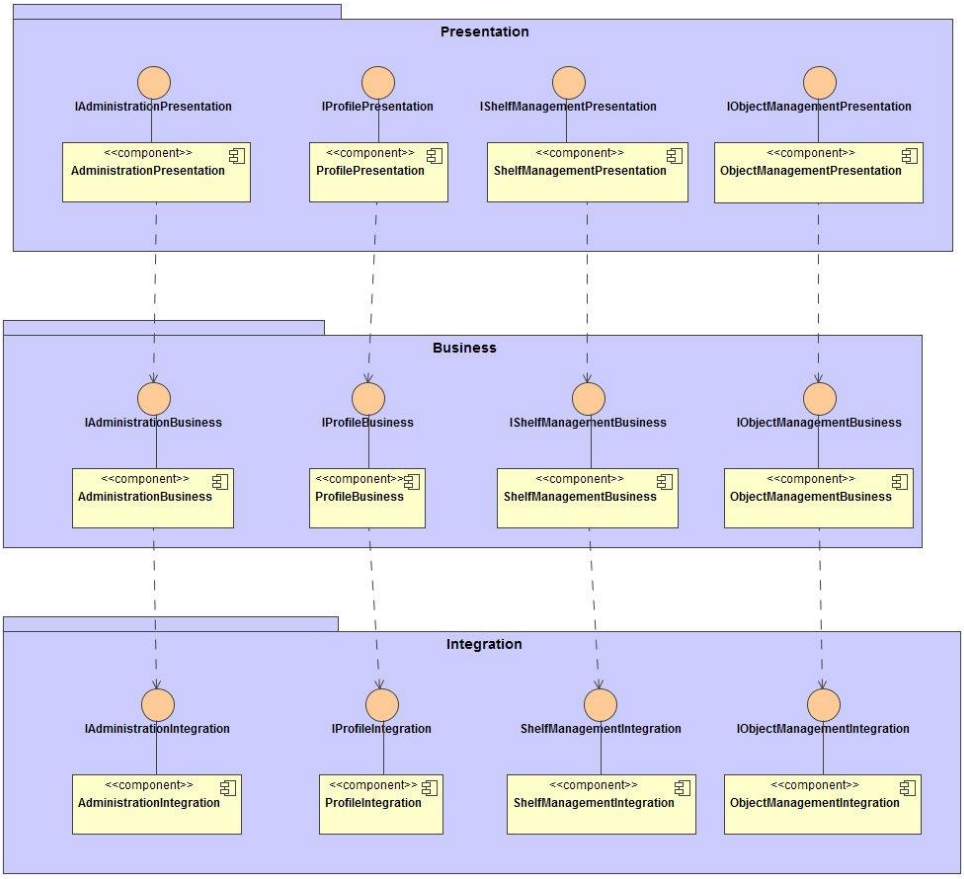


6. Diagrama de arquitectura

He decidido usar una arquitectura en capas, que nos permite descomponer nuestra aplicación en varios sistemas independientes, lo que nos permitirá mayor reutilización, en caso de que fuera necesario, y una mayor libertad para poder modificar parte de nuestra aplicación sin que el resto se vea afectado, por si en un futuro se quisiera realizar una aplicación móvil, tener que cambiar solamente la parte visual de nuestro sistema.

A continuación, se muestra el punto de vista de la computación, donde se puede ver cómo interactúan los componentes del sistema.

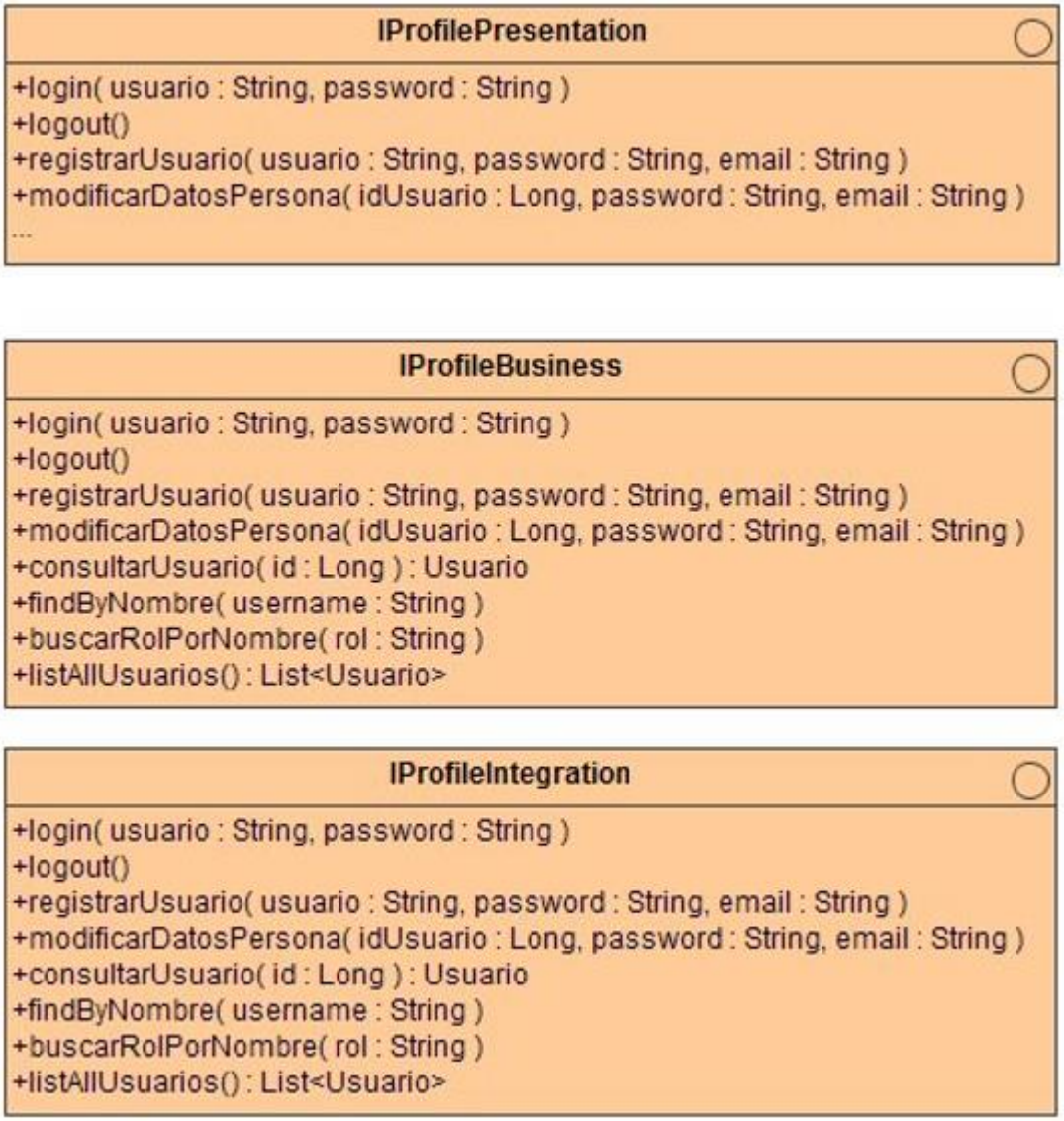
Ilustración 5 - Punto de vista de la computación



De forma detallada, especificando lo que contiene cada una de las interfaces que se ven en el diagrama anterior, tenemos lo siguiente:

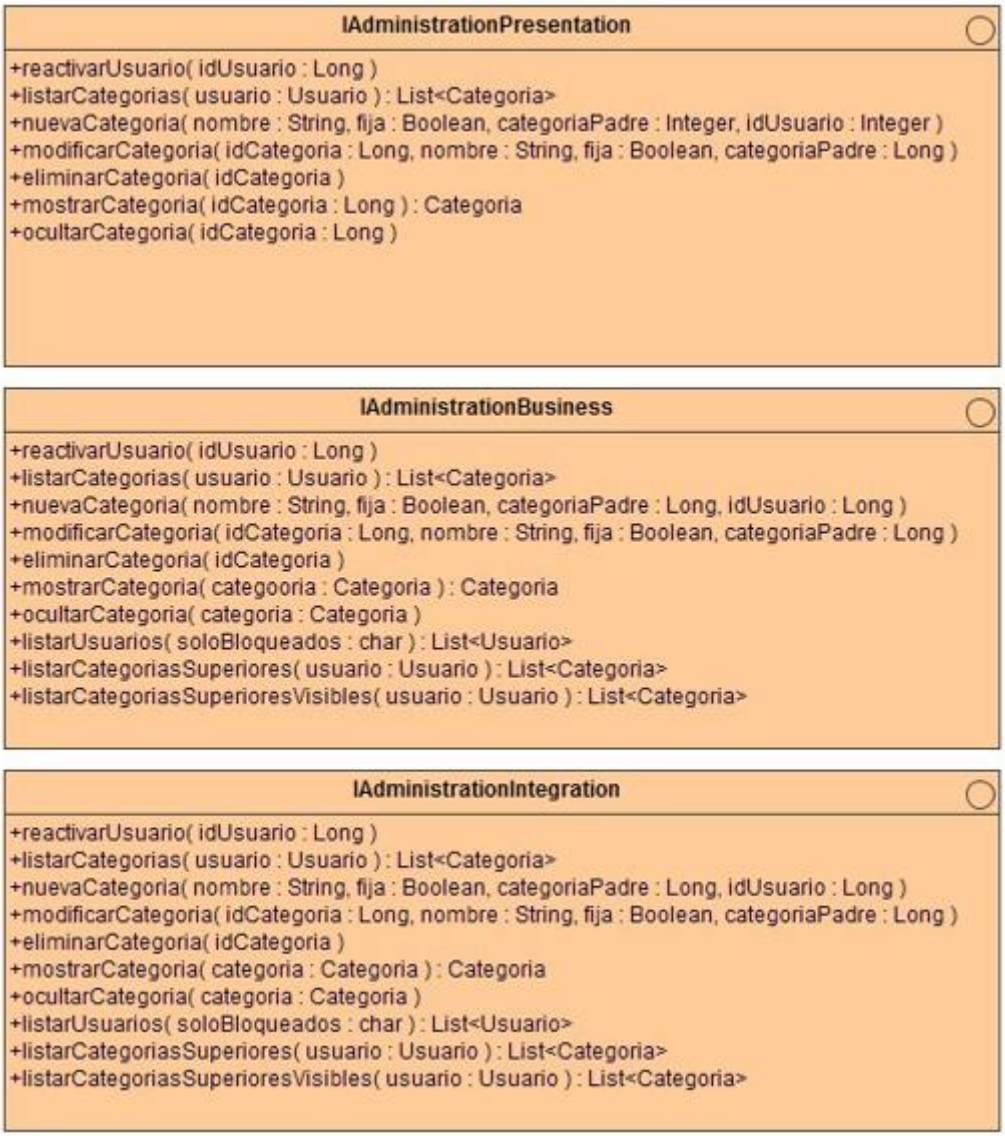
Detalle de las interfaces del elemento Profile:

Ilustración 6 - Interfaces Profile



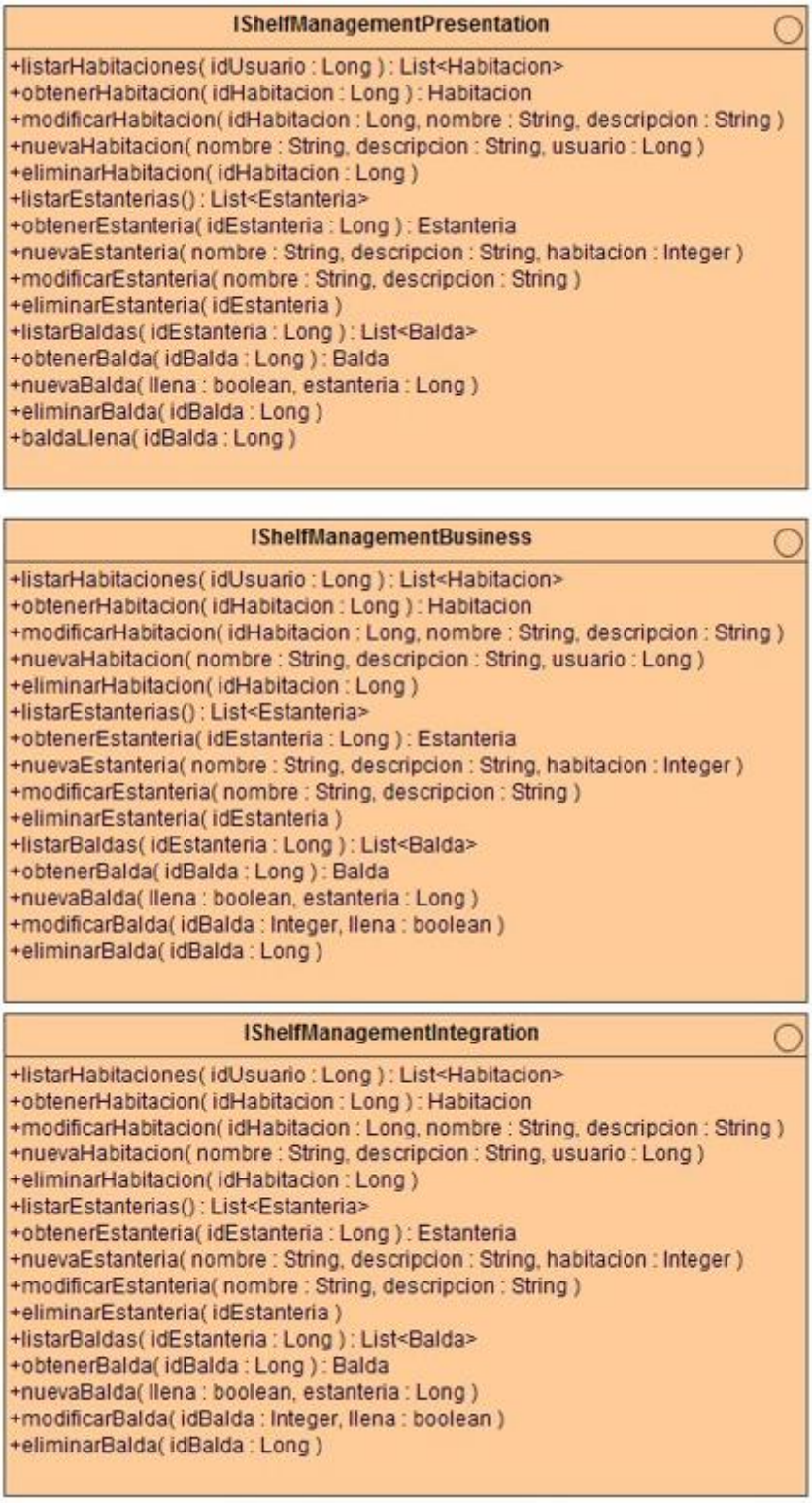
Detalle de las interfaces del elemento Administration:

Ilustración 7 - Interfaces Administration



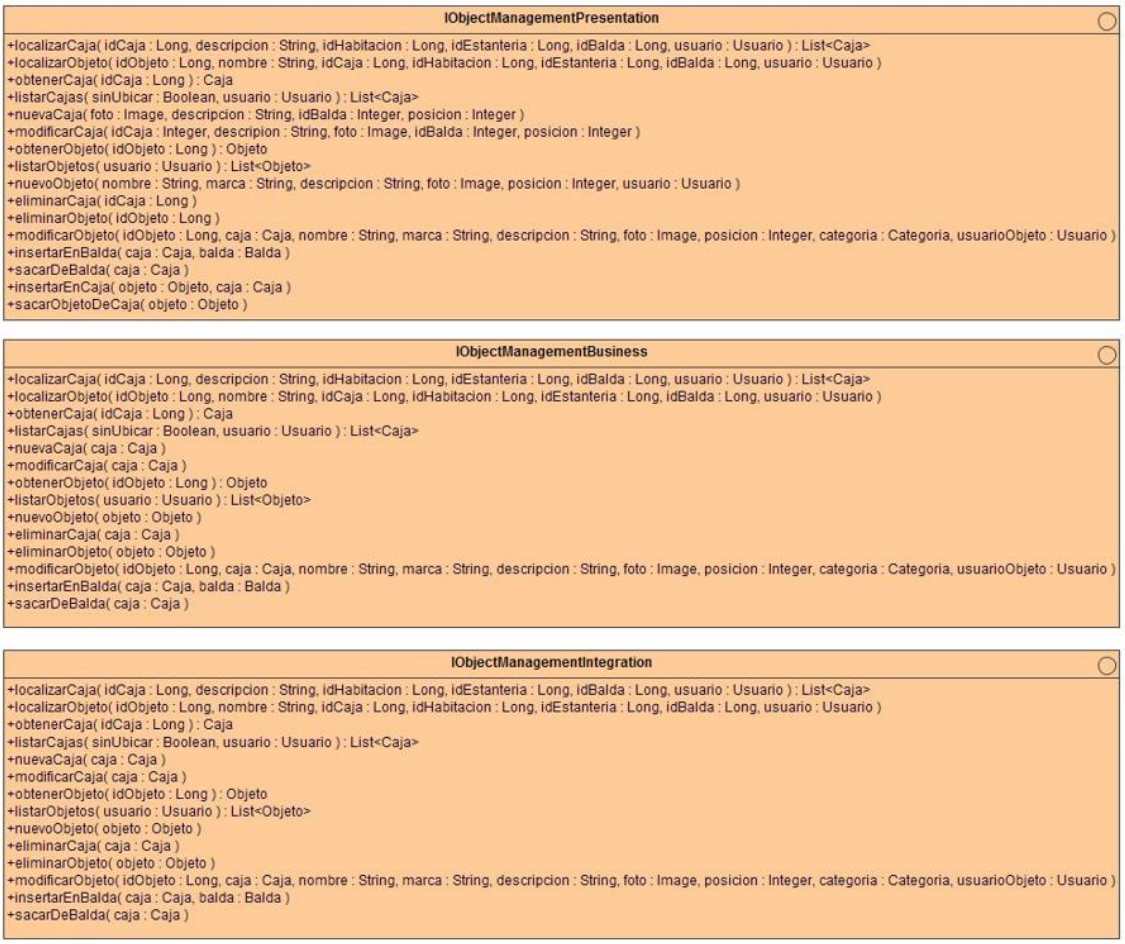
Detalle de las interfaces del elemento ShelfManagement:

Ilustración 8 - Interfaces ShelfManagement



Detalle de las interfaces del elemento ObjectManagement:

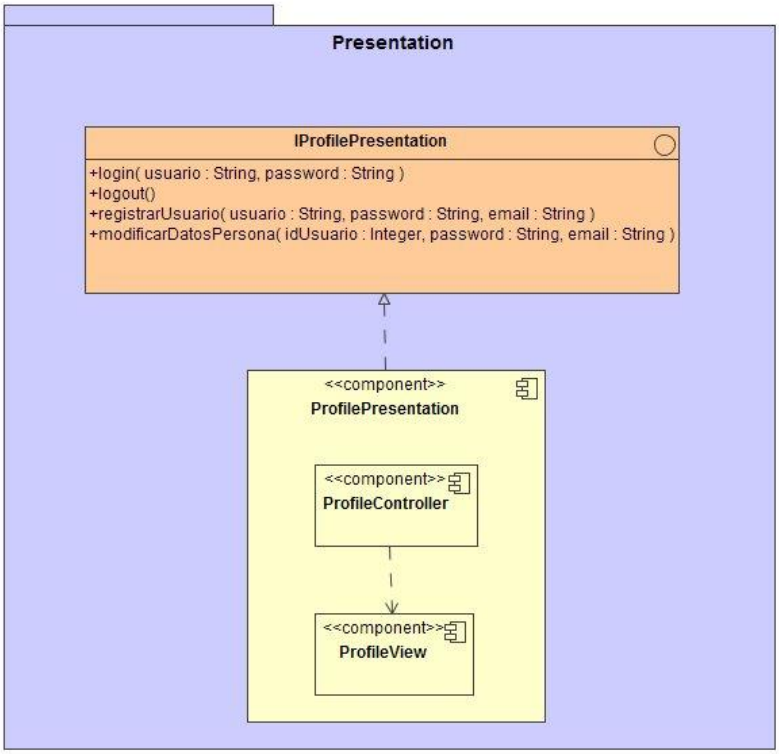
Ilustración 9 - Interfaces ObjectManagement



Para la capa de presentación se ha optado por aplicar la estructura Modelo-2, usando el patrón MCV, donde el modelo se implementará en el componente de negocio. Así que dividimos los distintos componentes que veíamos en la imagen anterior (ProfilePresentation, AdministrationPresentation…) en un componente que hará de controlador y en otro que será la vista.

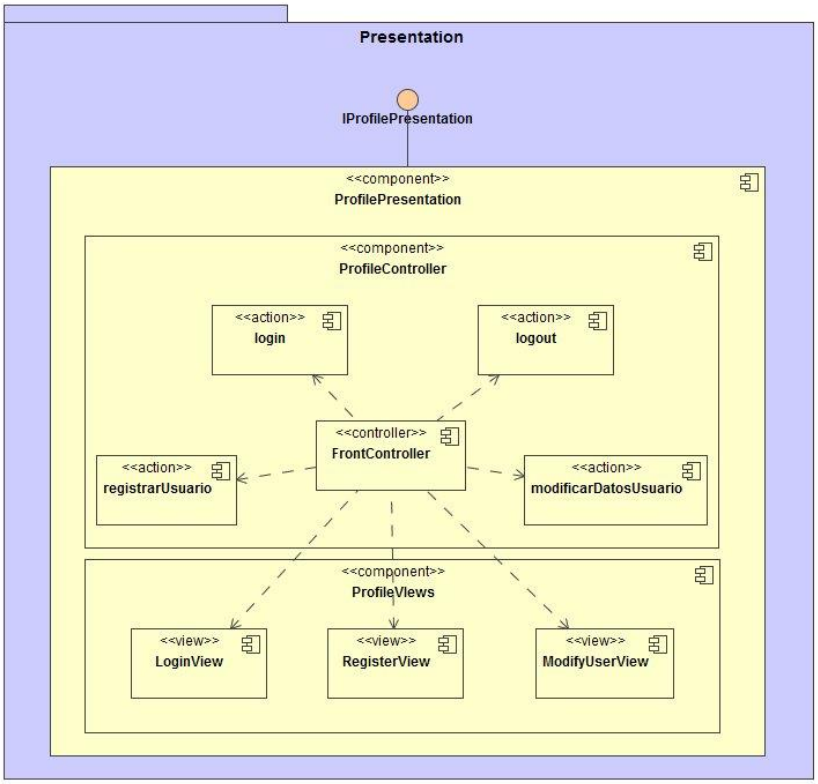
Pongo el ejemplo del componente ProfilePresentation, pues el resto serían exactamente iguales, aunque para el resto sí que se pondrá el diagrama de componentes final. Tendremos en el controlador las distintas acciones, y éstas tendrán una determinada vista.

Ilustración 10 - ProfilePresentacion Diagrama Componentes Inicial



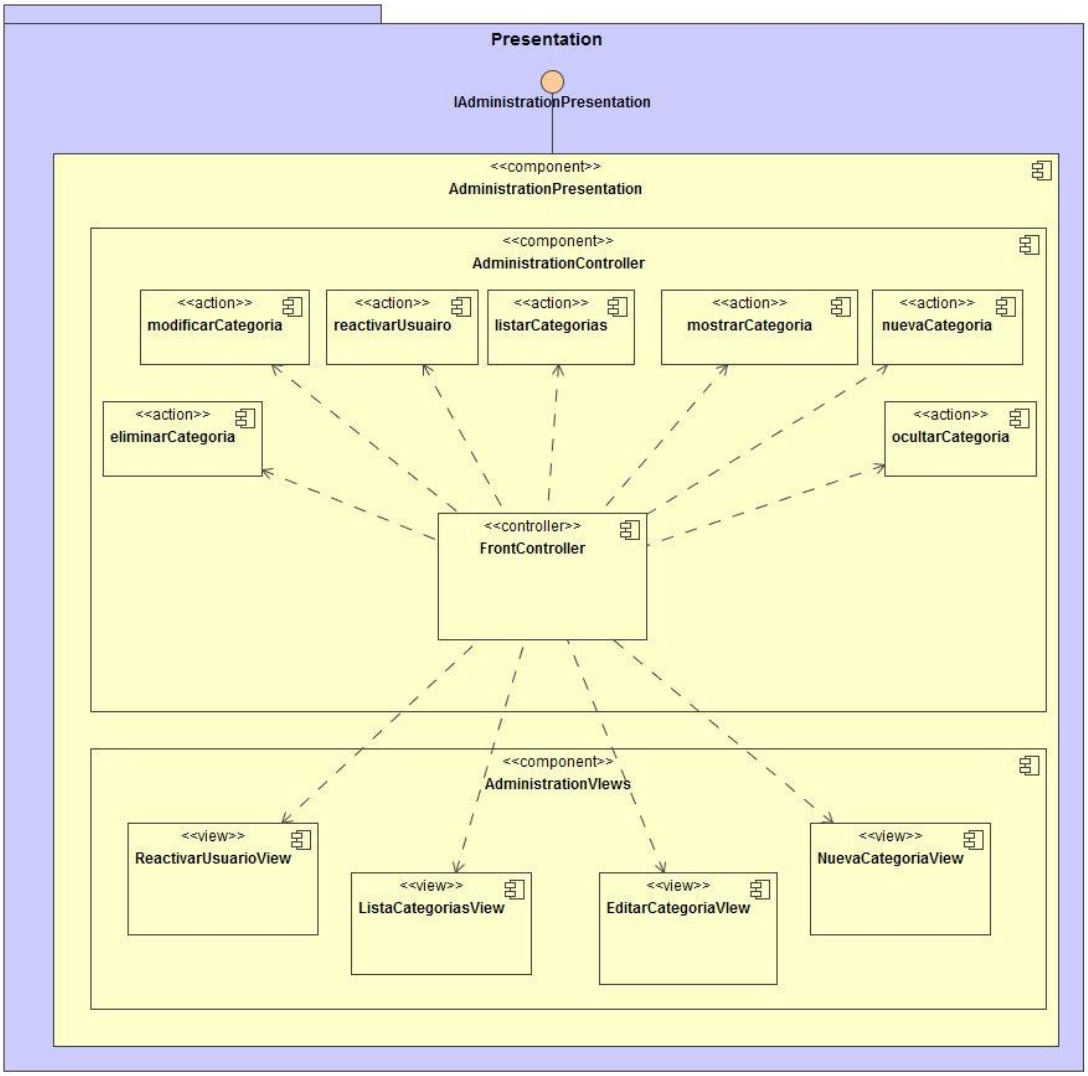
Profile

Ilustración 11 - Diagrama componentes ProfilePresentation



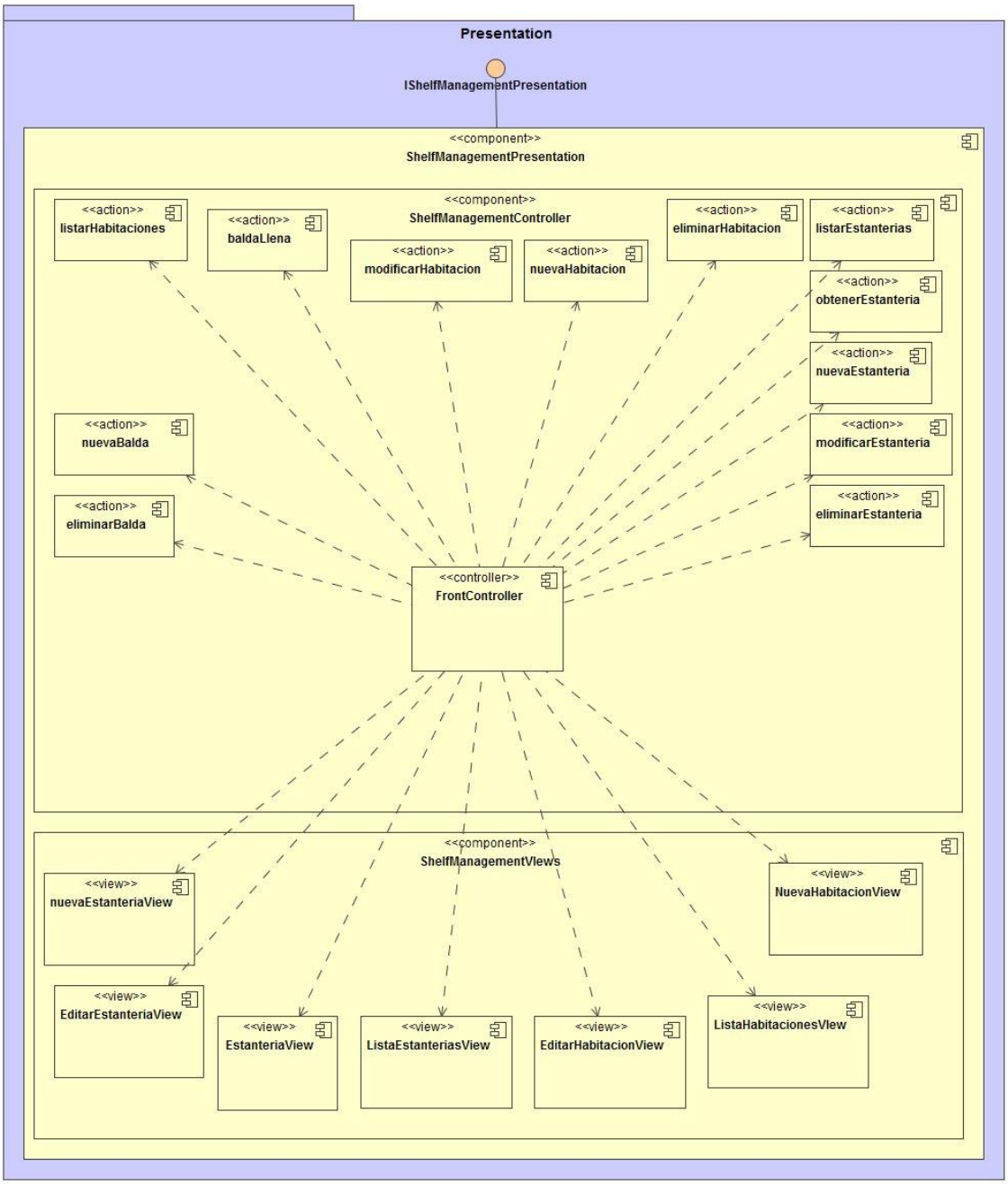
Administration

Ilustración 12 - Diagrama componentes AdministrationPresentation



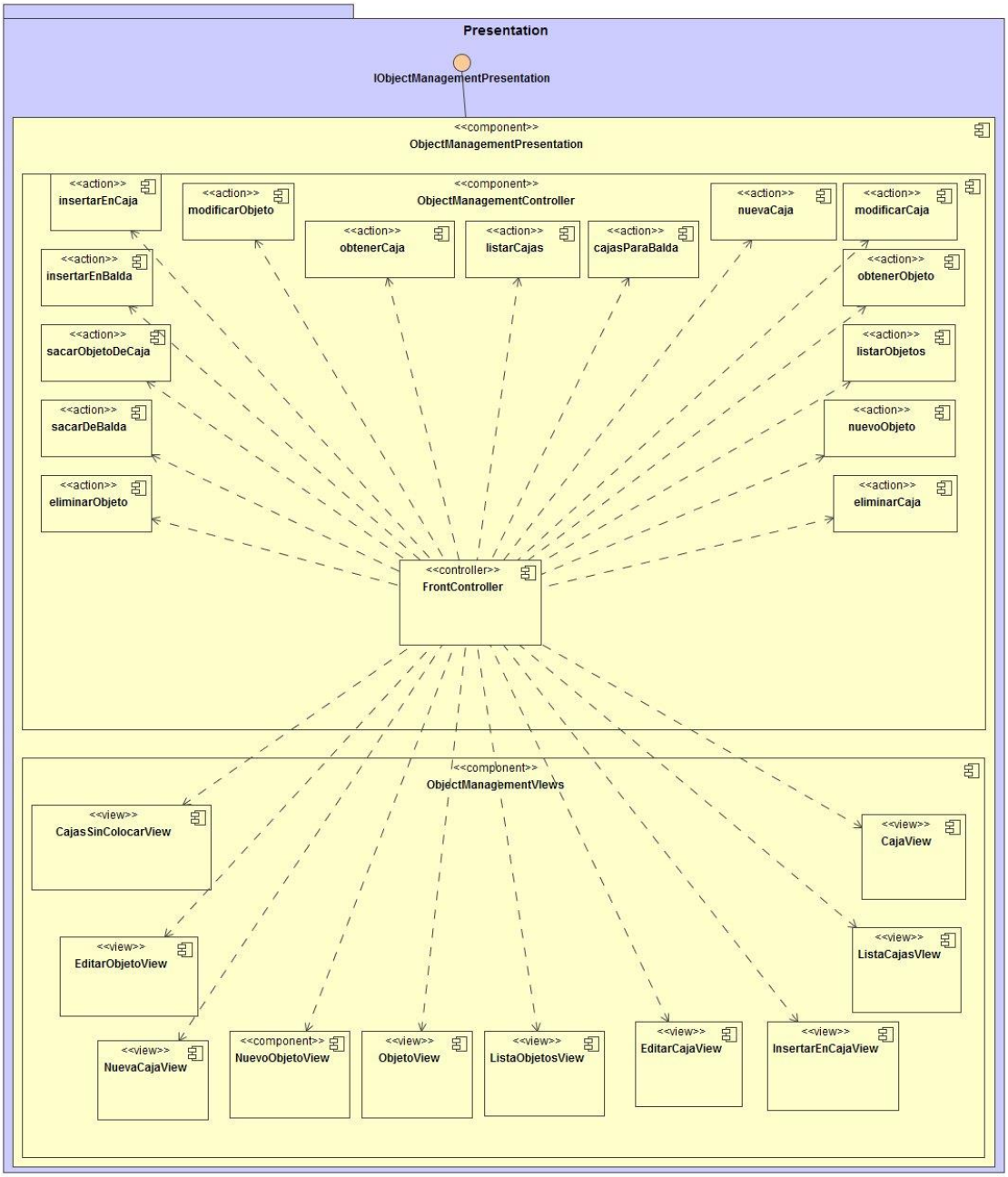
ShelfManagement

Ilustración 13 - Diagrama componentes ShelfManagementPresentation



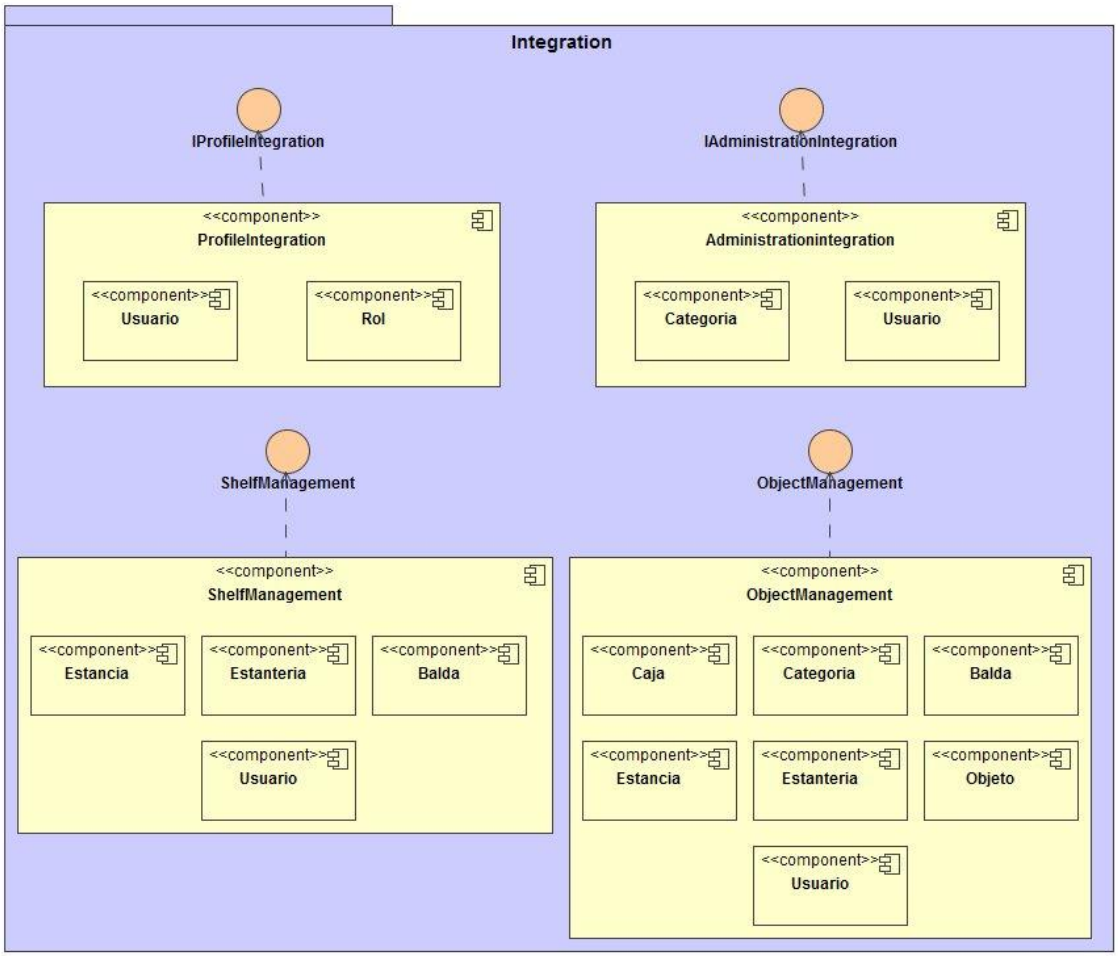
ObjectManagement

Ilustración 14 - Diagrama componentes ObjectManagementPresentation



La capa de integración quedará de la siguiente forma:

Ilustración 15 - Capa de integración



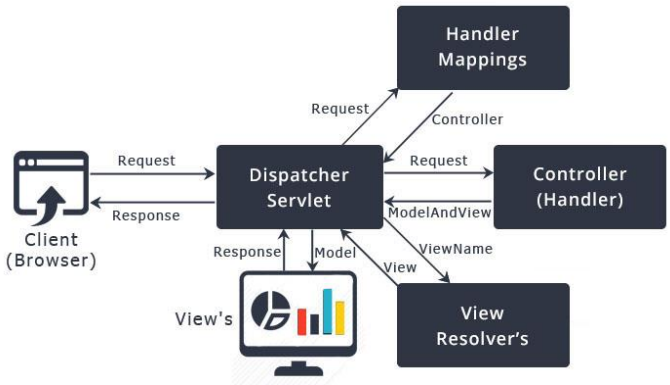
Debido a que la tecnología de implementación es JEE, haremos el diseño aplicando el perfil JEE a los diagramas obtenidos anteriormente.

Para la capa de presentación se ha optado por Spring MVC que, resumiendo, lo que tiene es un controlador, el Front Controller (en el caso de SpringMVel servlet DispatcherServlet) que atenderá las peticiones que llegan, para buscar a qué Controller tendría que llamar para poder servir esta petición.

El Controller será quien consulte la lógica de negocio, obteniendo lo que necesite en cada caso y devolviendo, mediante un Model, dicho resultado al servlet, así como la vista que se deberá mostrar. El DispatcherServlet se encargará de redirigir al usuario a la vista correspondiente, donde se podrán ver los resultados de la petición realizada. Para las vistas usaremos html.

Con esto tendremos la siguiente estructura.

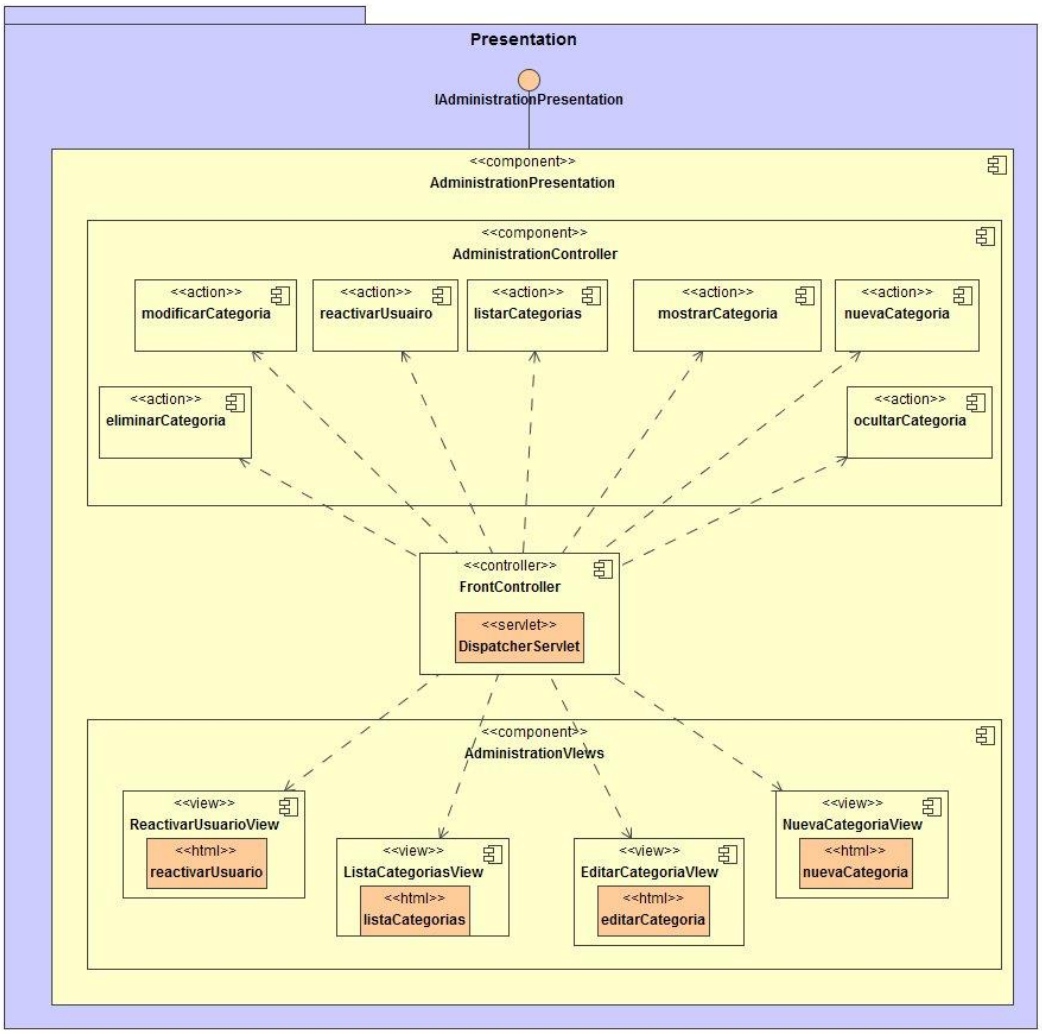
Ilustración 16 - Estructura Spring MVC



Con esto, nuestros diseños de la capa de presentación quedarán de la siguiente forma:

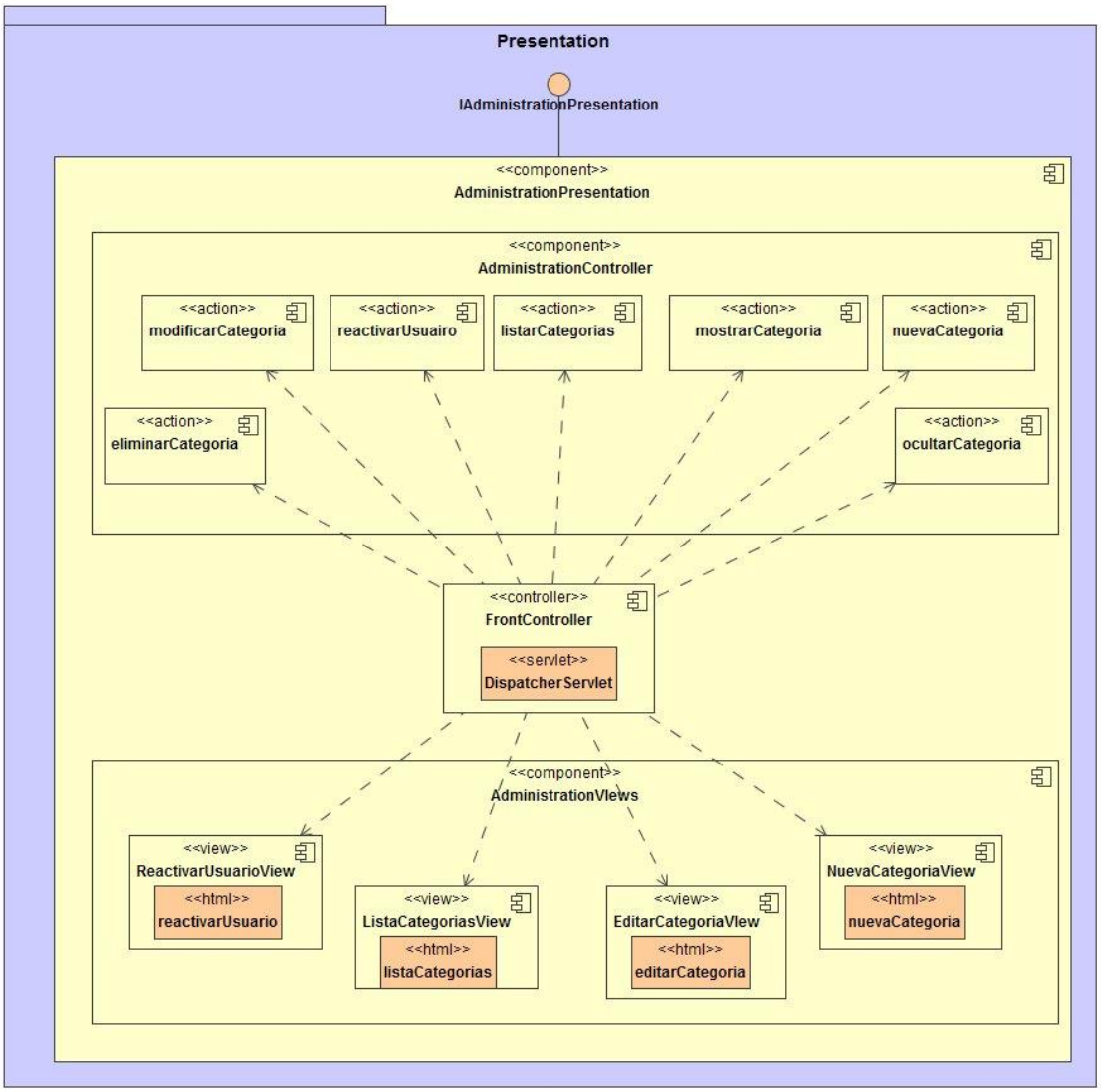
Administration

Ilustración 17 - Capa presentación JEE Administration



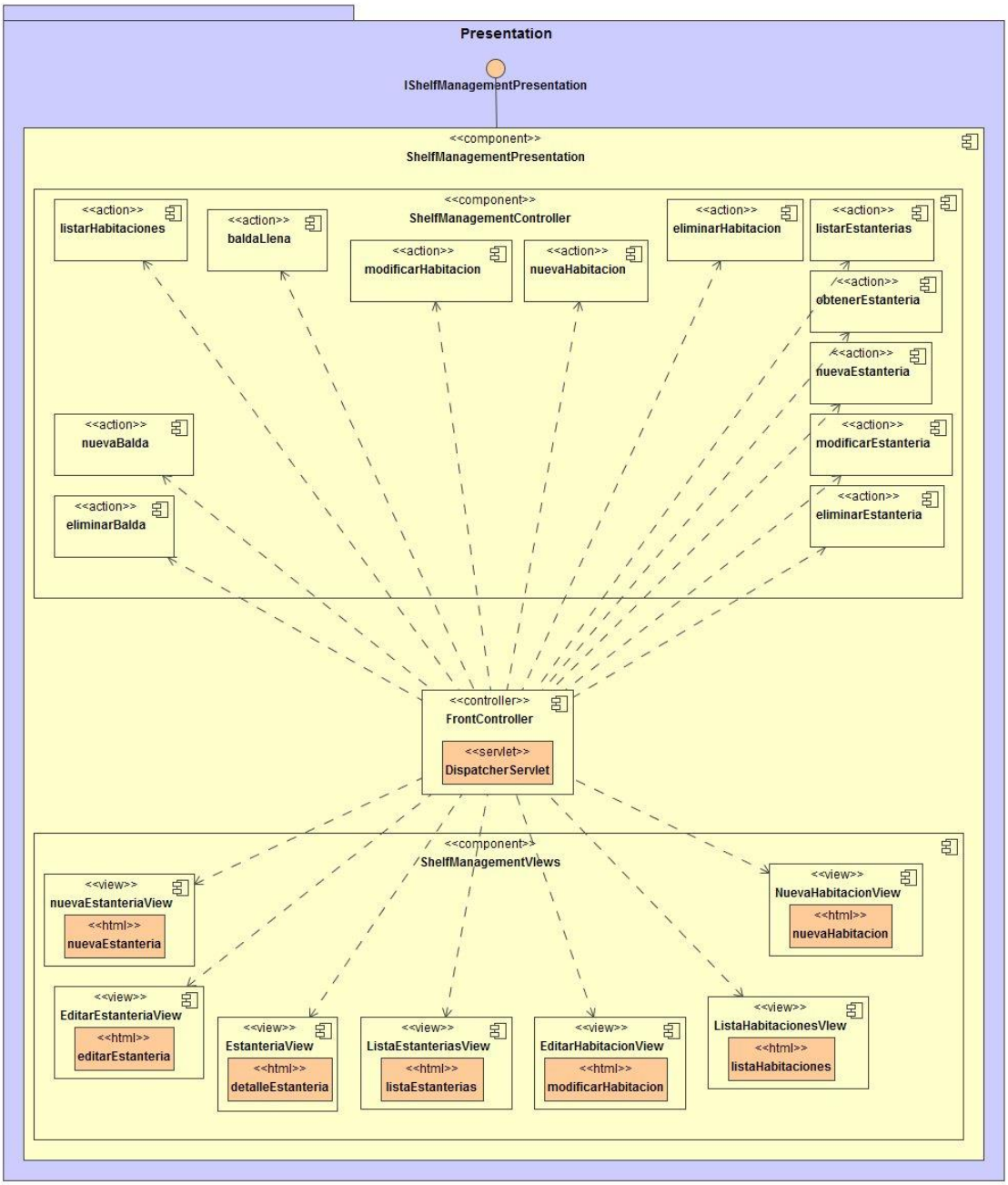
Profile:

Ilustración 18 - Capa presentación JEE Profile



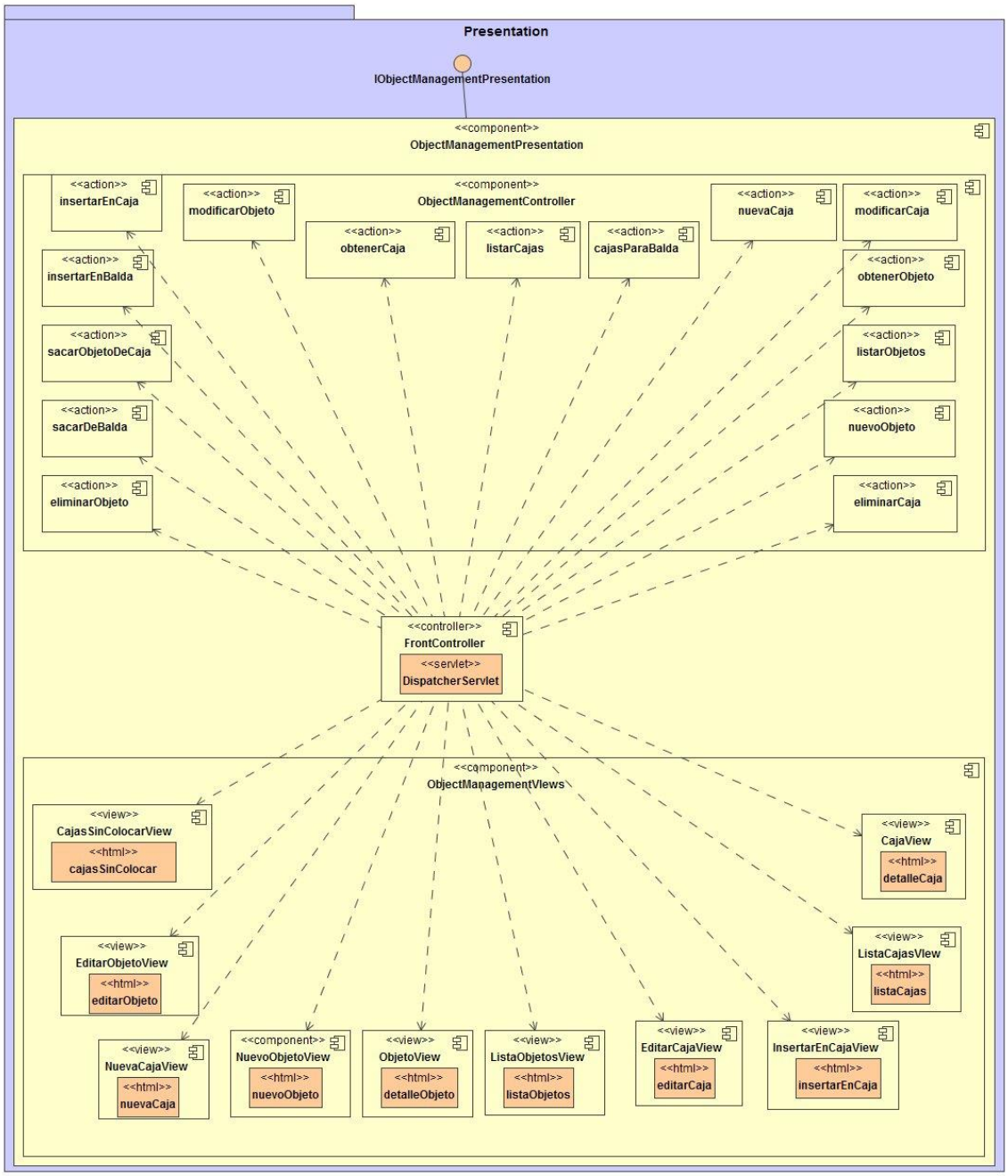
ShelfManagement:

Ilustración 19 - Capa presentación JEE ShelfManagement



ObjectManagement:

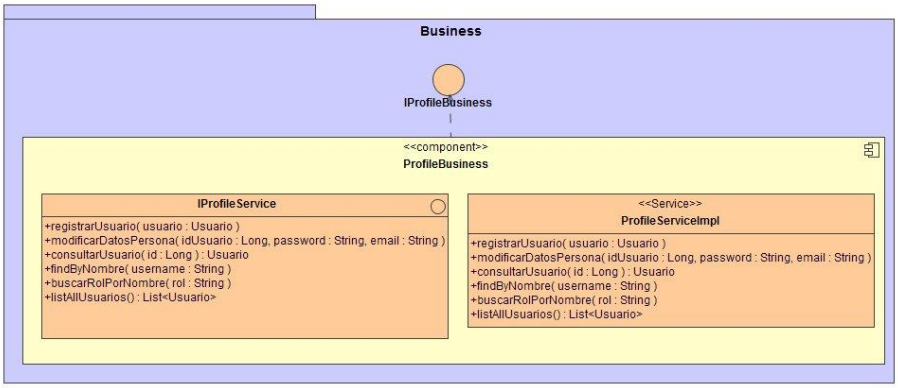
Ilustración 20 - Capa presentación JEE ObjectManagement



En la capa de negocio haremos uso del patrón fachada para tener un único punto de acceso a diferentes DAO. Como haremos uso de Spring, utilizamos un Service como fachada.

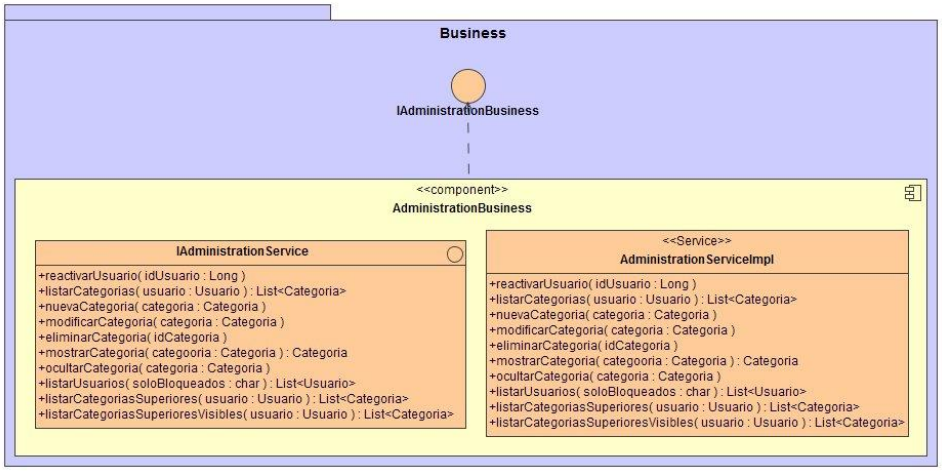
Profile:

Ilustración 21 - Capa negocio JEE Profile



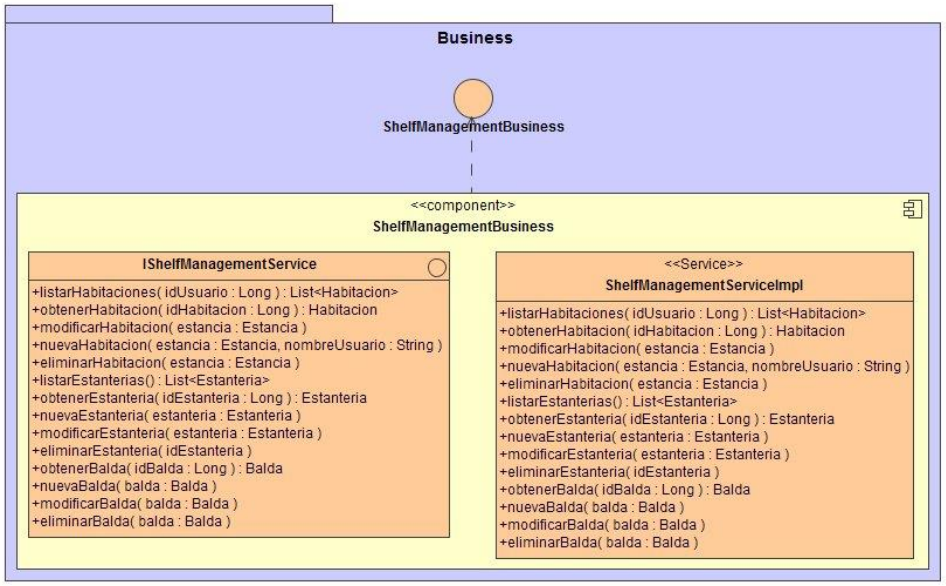
Administration:

Ilustración 22 - Capa negocio JEE Administration



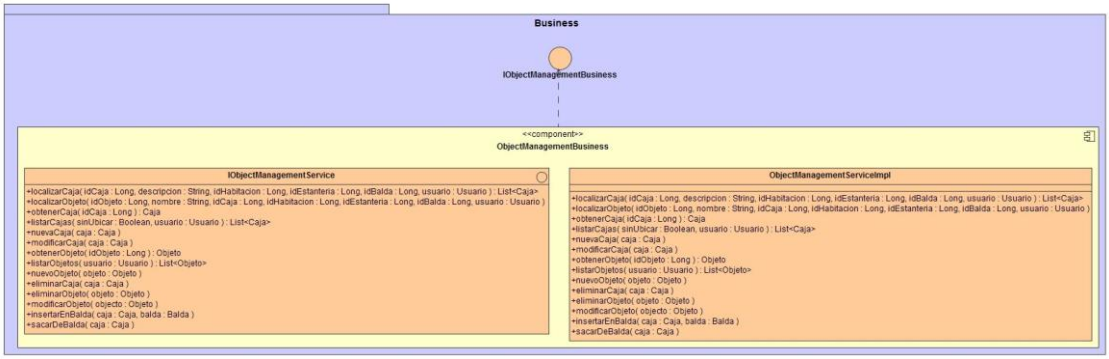
ShelfManagement:

Ilustración 23 - Capa negocio JEE ShelfManagement



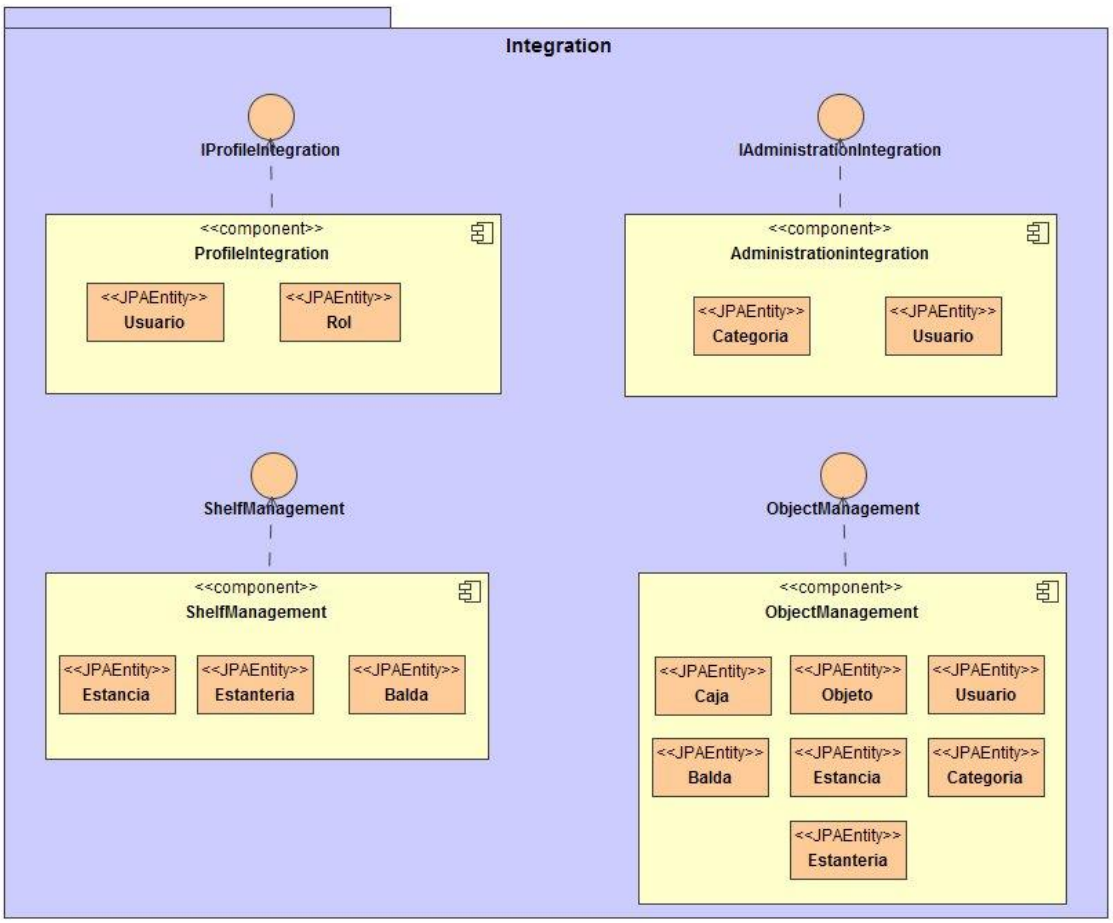
ObjectManagement:

Ilustración 24 - Capa negocio JEE ObjectManagement



En la capa de integración, cada módulo tiene un componente de control, donde usamos los componentes JPA para las entidades que utilizan.

Ilustración 25 - Capa integración JEE



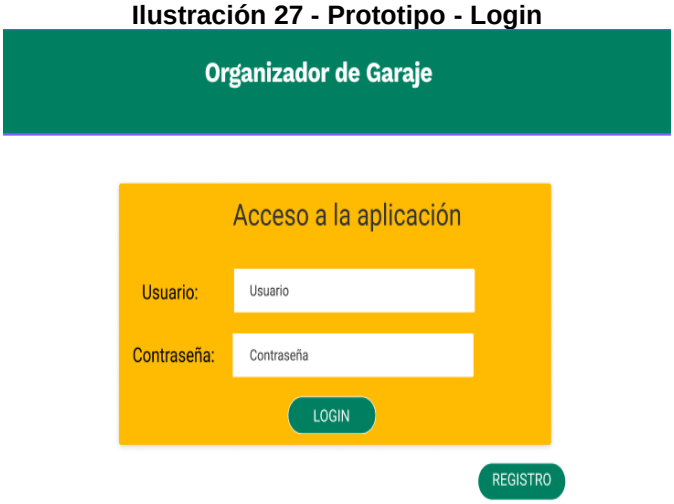
7. Prototipo

A continuación, se muestra un prototipo de las distintas pantallas que componen la aplicación Organizador de Garaje. Estos prototipos se han realizado con la aplicación Figma, que es una de las que pude ver y utilizar en la asignatura de “Interacción persona ordenador”.

Página principal, donde se realizará el login de los clientes y administradores, así como el registro de los usuarios no registrado:

Ilustración 26 - Prototipo - Registro

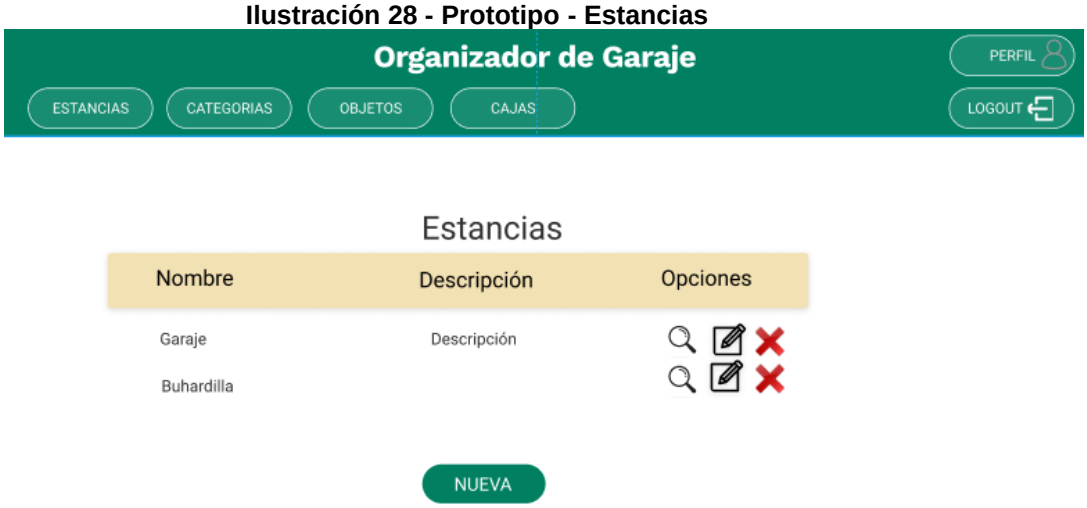
Ilustración 27 - Prototipo - Login



Cualquiera de los dos botones de la página de registro de usuario, llevará de vuelta a la página principal para poder realizar el login.

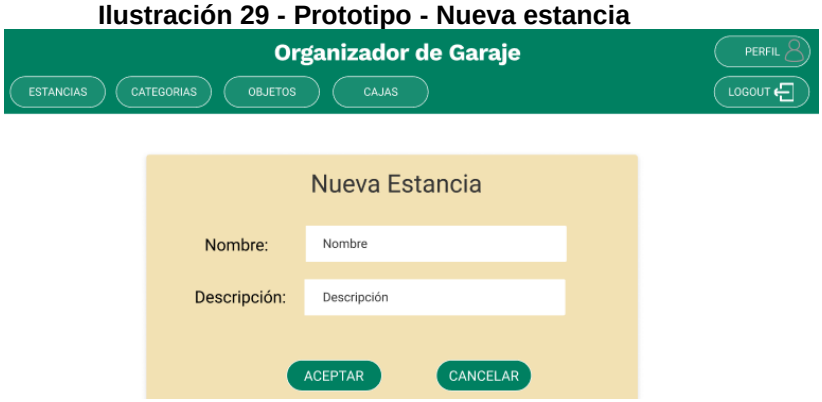
Una vez se ha rellenado el usuario y contraseña de forma correcta, se llevará a la página que contiene las habitaciones que se hubieran creado anteriormente y donde se podrán dar de alta nuevas habitaciones.

Ilustración 28 - Prototipo - Estancias



La pantalla de nueva estancia será prácticamente igual que la de modificación y la de consulta de estancia. Aunque en esta última no se podrán modificar datos.

Ilustración 29 - Prototipo - Nueva estancia



Todas las pantallas de modificar serán iguales a la de crear, pero aparecerán rellenos los datos del elemento seleccionado.

Al pulsar sobre la lupa de una de las estancias disponibles, se mostrarán las estanterías de las que se dispone en la habitación, y se podrán dar de alta nuevas habitaciones. Se muestra el nombre de la estancia en la que se encuentra y en la que se creará la estantería.

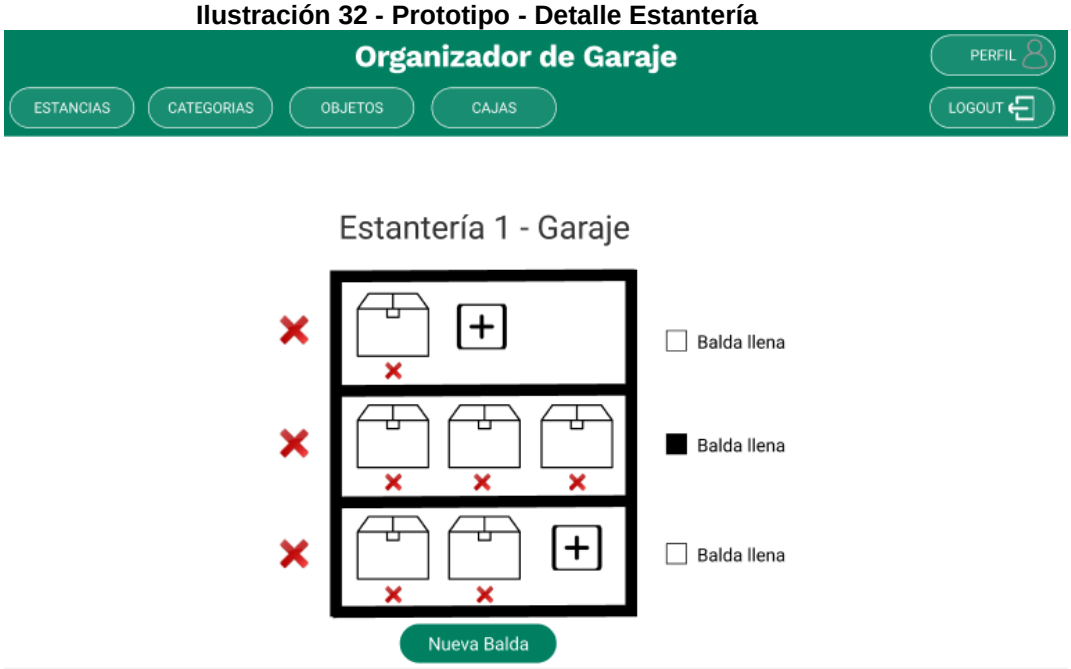
Ilustración 30 - Prototipo - Estanterías

Al igual que con las estancias, la pantalla de nueva estantería será prácticamente igual que la de modificación. Aunque en este caso, la página de consulta es totalmente diferente.

Ilustración 31 - Prototipo - Nueva estantería

A continuación, se muestra la pantalla de consulta de una estantería concreta:

Ilustración 32 - Prototipo - Detalle Estantería



En esta pantalla se verán las baldas que tiene la estantería, con las cajas que se tiene en cada una de ellas. Además, se podrá pulsar sobre las cajas para ver el detalle de éstas, así como agregar una nueva caja a la balda. Cuando se pulse sobre el botón de nueva caja, se mostrará un listado con las cajas que actualmente el usuario no tiene colocadas en ninguna balda.

Ilustración 33 - Prototipo - Cajas sin colocar

Al pulsar en el “+” se asignará la caja en la balda seleccionada. Y al pulsar sobre nueva caja, se mostrarán los datos que se pueden guardar, que es la descripción y una imagen de la caja.

Ilustración 34 - Prototipo - Nueva caja

Esta es la pantalla de gestión de categorías, donde se muestran todas las categorías, y sus subcategorías, así como si son visibles o no. Ya que las que no son visibles, no se podrán seleccionar en el desplegable de la inserción/modificación de objetos.

Ilustración 35 - Prototipo - Categorías

Ilustración 36 - Prototipo - Nueva categoría

En la gestión de cajas, se podrá realizar la búsqueda de una determinada caja, así como rellenar una serie de filtros para poder acotar la búsqueda de las cajas. Ya que en esta pantalla aparecerán todas las cajas que hubiera creado el cliente. Algo similar pasará con la gestión de objetos.

Ilustración 37 - Prototipo - Cajas

Al pulsar en la lupa de las cajas, tengo que destacar que además de visualizar el detalle de la caja, así como la imagen que pudiera tener, y el listado de objetos que se tuviera dentro. También se mostrará un código QR con la dirección a la visualización del detalle de la caja.

Esta captura se incorporó posteriormente, cuando ya se estaba realizando la implementación.

Ilustración 38 - Prototipo - Detalle de caja

Ilustración 39 - Prototipo - Objetos

Cuando se pulsa sobre el botón de la caja, si el objeto ya estaba en una caja se preguntará si se quiere sacar de ahí, y si no está en ninguna caja nos aparecerá la siguiente pantalla, en la que nos aparecerán las cajas disponibles para poder asociarla a una de ellas.

Ilustración 40 - Prototipo - Insertar objeto en la caja

La pantalla de crear un nuevo objeto mostrará los campos que se pueden rellenar. La pantalla del detalle de un objeto será similar a ésta, con los datos rellenos, sin poder editar, y donde se mostrará la imagen que se hubiera insertado anteriormente.

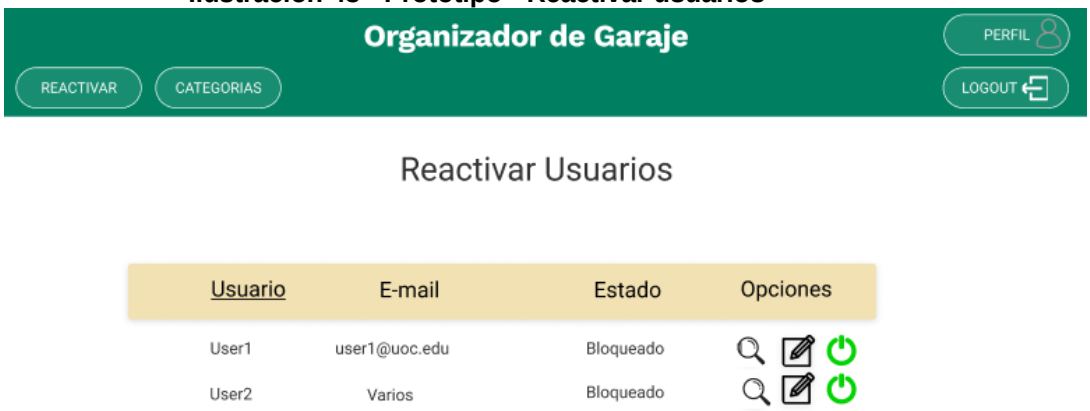
Ilustración 41 - Prototipo - Nuevo objeto

La pantalla de modificación de datos personales, donde lo único que no se podrá modificar es el usuario:

Ilustración 42 - Prototipo - Modificar datos usuario

Cuando entra un usuario administrador, las opciones de menú que aparecerán serán solamente la de gestión de categorías, modificación de datos personales y reactivar usuarios.

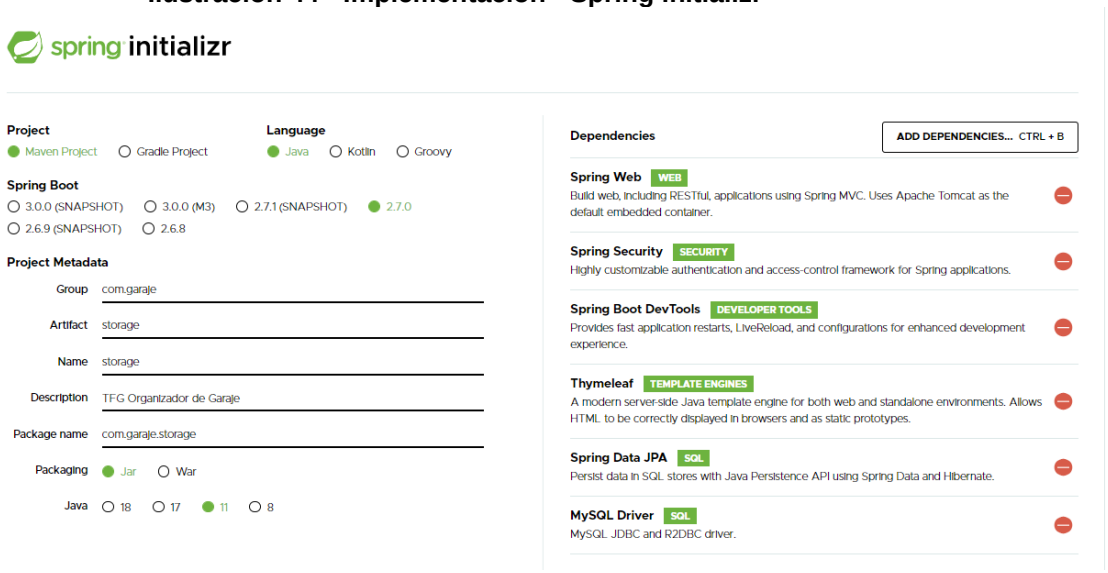
Ilustración 43 - Prototipo - Reactivar usuarios



8. Implementación

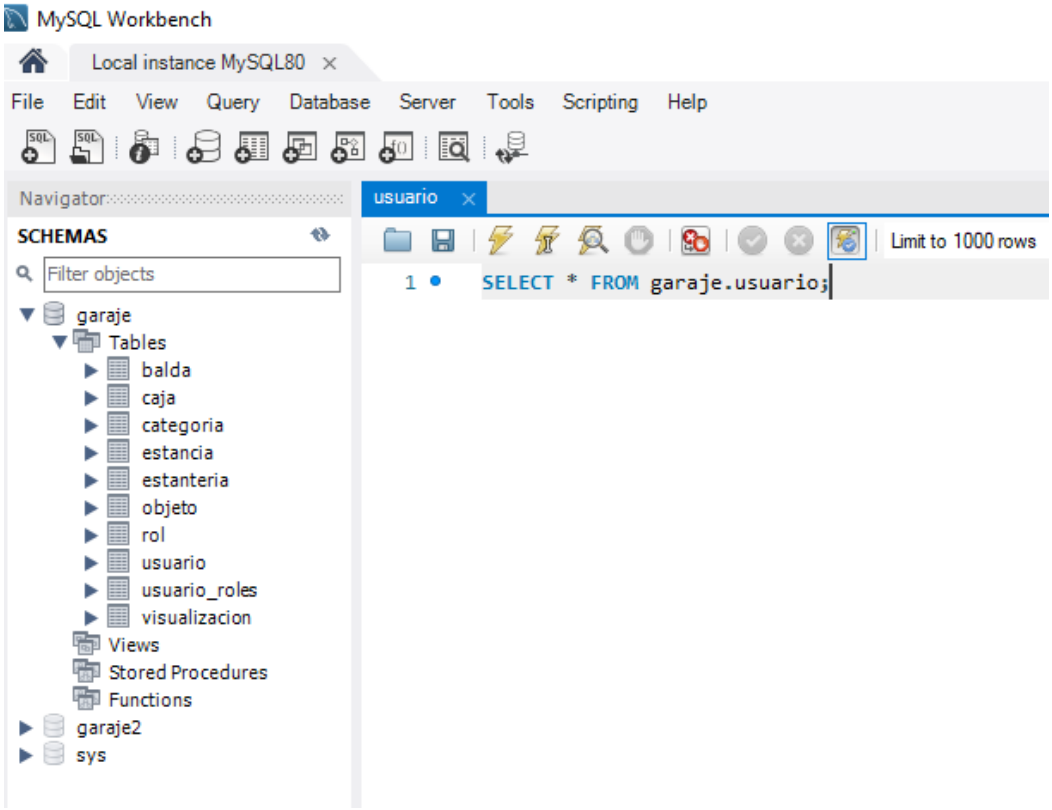
A la hora de realizar la implementación de la aplicación, he decidido hacer uso de SpringBoot para poder centrarme en el propio desarrollo, en lugar de tener que desviarme en el tema de las configuraciones y despliegue, utilizando Spring Boot internamente un servidor de aplicaciones Apache Tomcat. El esqueleto de la aplicación se ha hecho haciendo uso de Spring Initialzr, donde se indicó que la aplicación sería un proyecto Maven, que utilizaría la versión 11 de Java, 2.6.6 de SpringBoot (aunque en la captura se aprecie 2.7.0, ya que cuando comencé el proyecto era la versión más actual), y donde se añadieron las distintas dependencias que se utilizarían a lo largo del proyecto.

Ilustración 44 - Implementación - Spring initializr



Además, el IDE utilizado ha sido Spring Tool Suite 4.
Tengo que destacar el driver MySQL, pues, aunque en un principio tenía intención de haber usado PostgreSQL, decidí que era una buena oportunidad para probar con un gestor de base de datos diferente al que había usado en otras asignaturas e incluso en el mundo laboral. Por esto, hice uso de MySQL Server, haciendo uso de la herramienta visual MySQL Workbench.

Ilustración 45 - Implementación - MySQL Workbench



Además, para la parte de frontal, he decidido utilizar Thymeleaf, que es un motor de plantillas que permite prototipar de una forma más sencilla las pantallas utilizando código HTML al que se puede añadir algunos atributos nuevos, por lo que es mucho más legible. Además, quería otro reto de probar cosas nuevas y útiles para continuar aprendiendo.

Pude comprobar que Thymeleaf tiene bloque de código, al que llaman fragmentos, que puede ser reutilizado en las demás páginas de la aplicación. Así que me decidí a crear un HTML que incluía dos fragmentos.

El primero es el fragmento header, que incluye los estilos que se utilizan en las distintas páginas, así como los js necesarios para poder utilizar jQuery y Bootstrap (que se ha usado para obtener sus estilos), como algunos propios para ordenar las tablas que aparecen en varias partes de la aplicación.

```
<head th:fragment="header">
      <meta charset="utf-8">
      <title th:text="${tituloPagina}"></title>
      <link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/boo
tstrap.min.css"
            integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
            crossorigin="anonymous">
      <link href="/css/main.css" rel="stylesheet">
```

```
      <link href="/css/ordenarTabla.css" rel="stylesheet">
    <script src="/webjars/jquery/3.6.0/jquery.min.js"
type="text/javascript"></script>
    <script src="/js/ordenarTabla.js" type="text/javascript"></script>
      <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-
7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB"
crossorigin="anonymous"></script>
      <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-
QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13"
crossorigin="anonymous"></script>
</head>
```

Además de ese fragmento, en el mismo HTML se incluye otro fragmento para poder ser utilizado, que es el que contiene la cabecera de la aplicación, con los distintos botones disponible según el usuario que esté logado en la aplicación.

```
<div class="container header" th:fragment="menu">…</div>
```

Luego en cada una de las páginas podemos importar estos fragmentos de la siguiente forma:

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragmentsCabecera::header"/>
<body>
    <div th:replace="fragmentsCabecera::menu"></div>
```

Otro aspecto importante de la implementación ha sido la incorporación de Spring Security para la seguridad del sistema y especificar a qué partes de la aplicación se tiene acceso dependiendo del rol del usuario que esté entrando. Así, por ejemplo, con la siguiente configuración:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeHttpRequests()
    .antMatchers("/js/**","/css/**", "/register" , "/login").permitAll()
    .antMatchers("/reactivarUsuario").hasRole("ADMIN")
    .anyRequest().authenticated()
    .and()
    .formLogin().loginPage("/login")
    .usernameParameter("username")
    .failureHandler(loginFailureHandler)
    .successHandler(loginSuccessHandler)
    .and()
    .logout();
}
```

Se tiene que el acceso a los js, css, la página de registros de usuario y a la de login, se tiene acceso siempre para cualquier usuario. Sin embargo, a la de reactivar usuarios solamente podrá acceder el administrador, y al resto bastará con estar autenticado en el sistema.

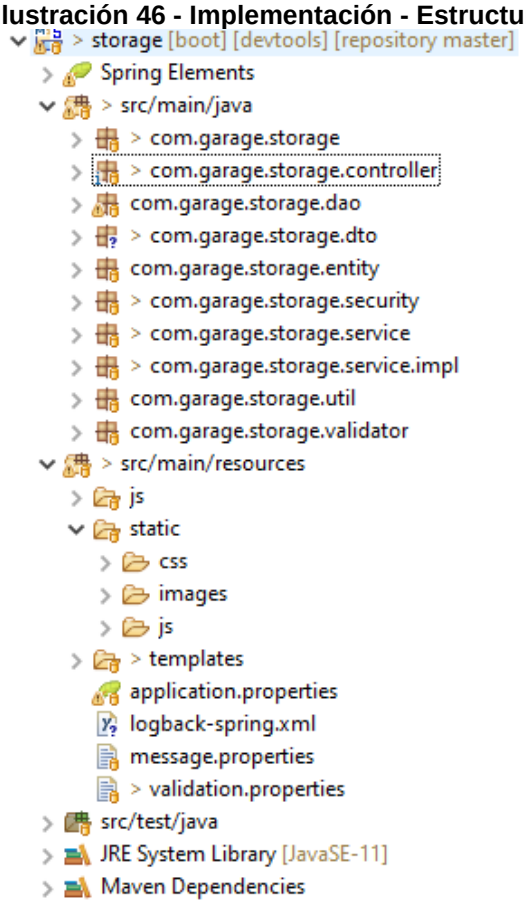
Además, he implementado qué pasaría en caso de login fallido o satisfactorio, pues he querido que cuando un usuario haga un login fallido 5 veces, su usuario se quede bloqueado. Además, si el login va bien, se resetean los intentos fallides que llevara, por si el usuario se hubiera confundido algunes veces al logarse.

Otra de las herramientas utilizadas en la implementación de la aplicación, ha sido el uso del plugin de Eclipse SonarLint para comprobar la calidad del código. Esto sacó a la luz que, en el controlador, en los método que tenían las anotaciones RequestMapping no deberían tener entre los parámetros, los Entity, pues podría verse comprometido por un ataque malicioso en el que se modificaran campos que no debería ser posible modificar.

Por esa razón, en el controlador, en lugar de usar como parámetros los Entity (ya vinculados con la base de datos), se hace uso de DTO, realizando posteriormente una transformación de DTO a Entity y viceversa, gracias a la librería ModelMapper.

La aplicación ha terminado teniendo la siguiente estructura:

Ilustración 46 - Implementación - Estructura

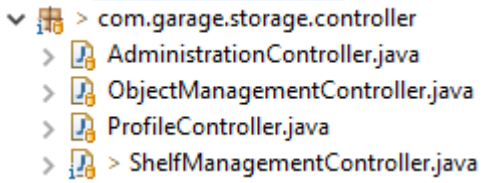


Donde dentro de “storage” tenemos el método principal que arrancará la aplicación.

```java
package com.garage.storage;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class StorageApplication {


    public static void main(String[] args) {
        SpringApplication.run(StorageApplication.class, args);
    }

}
```

En controller se encuentran los 4 controladores que ya se especificaron anteriormente, con los métodos encargados de recoger los elementos de las vistas para llamar a la capa de negocio, así como los métodos encargados de llamar a las vistas con lo necesario para el buen funcionamiento.

Ilustración 47 - Implementación - Controller



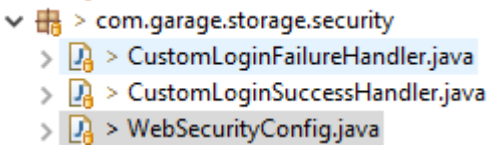
Por otro lado, tenemos los DAO implementados con Spring Data, de los que solo se necesita la interfaz, aunque se ha probado a crear un par de ellos con JPA para poder probar las dos formas.

Tenemos los DTO que ya comenté anteriormente que fueron creados para no tener los Entity en los RequestMapping.

En “entity” tenemos todas las clases que están identificadas con la anotación de JPA @Entity para enlazarlas con las distintas tablas de la base de datos.

En “security” se han incorporado las 3 clases que se han añadido para SpringSecurity.

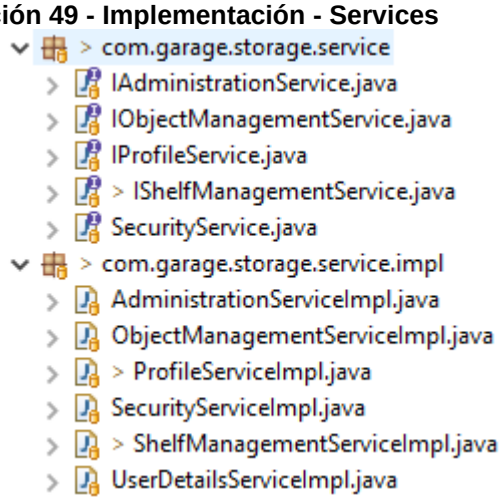
Ilustración 48 - Implementación - Security



Donde la primera extiende de SimpleUrlAuthenticationFailureHandler, para poder ir incrementando los intentos de login que realiza un usuario, y bloquear el usuario si fuera necesario, por haber superado el número máximo de intentos (5). La segunda clase, extiende de SimpleUrlAuthenticationSuccessHandler, y sirve para resetear los intentos fallidos cuando se realiza el login correctamente. Y finalmente, la última clase tiene la configuración vista anteriormente para decidir a qué funcionalidades se tiene acceso según el rol del usuario, o si no se hubiera logado en el sistema.

En “service” y “service.impl” tenemos lo servicios definidos en la capa de negocio, más un par de ellos necesarios para la seguridad vista antes, así como para comprobar si el usuario puede consultar la caja, objeto, estantería, etc., que estuviera intentando consultar, modificar o eliminar.

Ilustración 49 - Implementación - Services



Finalmente, en “util", se ha añadido una clase con algunas constantes, y una clase Utils con un par de métodos utilizados en distintas partes de la aplicación, como es el saber si una cadena es nula o vacía.
Y en “validator”, se han añadido varias clases para validar los datos de las vistas cuando llegaban al controlador, para comprobar que los campos cumplieran con los requisitos de cada una de las pantallas. Por ejemplo, que ciertos campos se tengan que rellenar o que un adjunto solo pueda ser de un determinado tipo y tamaño máximo.

En “resources” tenemos los distintos js, css, imágenes utilizadas en la aplicación, así como los templates, que son los html utilizados para las distintas páginas.

En “js”, además de mis propios javascript creados para ordenar las tablas o eliminar las imágenes guardadas, he incorporado una librería opensource para la generación de códigos QR, llamada QRCode.

Con esto se ha modificado la pantalla de obtener detalle de las cajas, para que ahora también se muestre el código QR generado, con la dirección de la página para visualizar el detalle de la caja. Así cuando la aplicación esté subida a un servidor, se podrá consultar fácilmente.

Ilustración 50 - Código QR

Ahora mismo, al estar en local, la dirección asociada sería: http://localhost:8080/obtenerCaja/{idCaja}

Cuando se escanee desde un móvil, por ejemplo, se irá a esa dirección, que mostrará el detalle de la caja, siempre y cuando el usuario estuviera logado en la aplicación y esa caja fuera de su propiedad, ya que de lo contrario se mostrará la página de inicio.

9. Instrucciones de despliegue

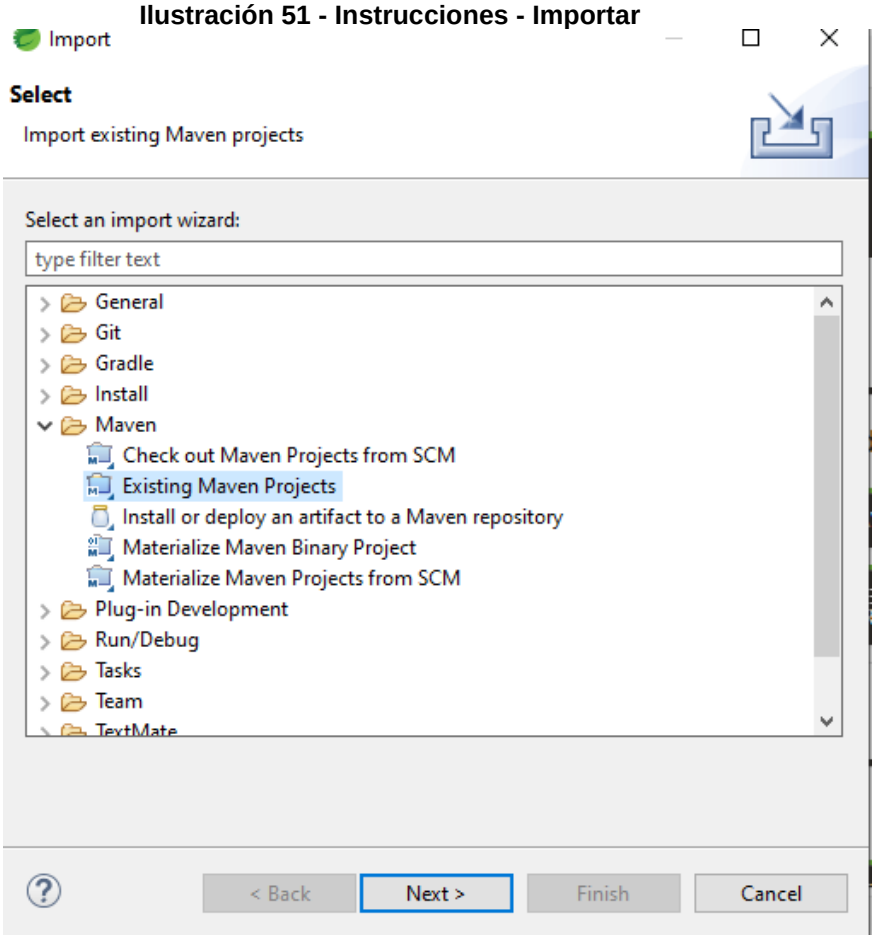
Lo primero que habría que hacer, es obtener el código de la aplicación, que se podría hacer de dos formas distintas.

Se podría clonar el repositorio: https://github.com/MiguelAngelOrgaz/garagestorage2

O se podría descargar todo el código para posteriormente importarlo.

Para ello, se pulsaría sobre importar y se seleccionaría un proyecto Maven existente desde Eclipse:

Ilustración 51 - Instrucciones - Importar



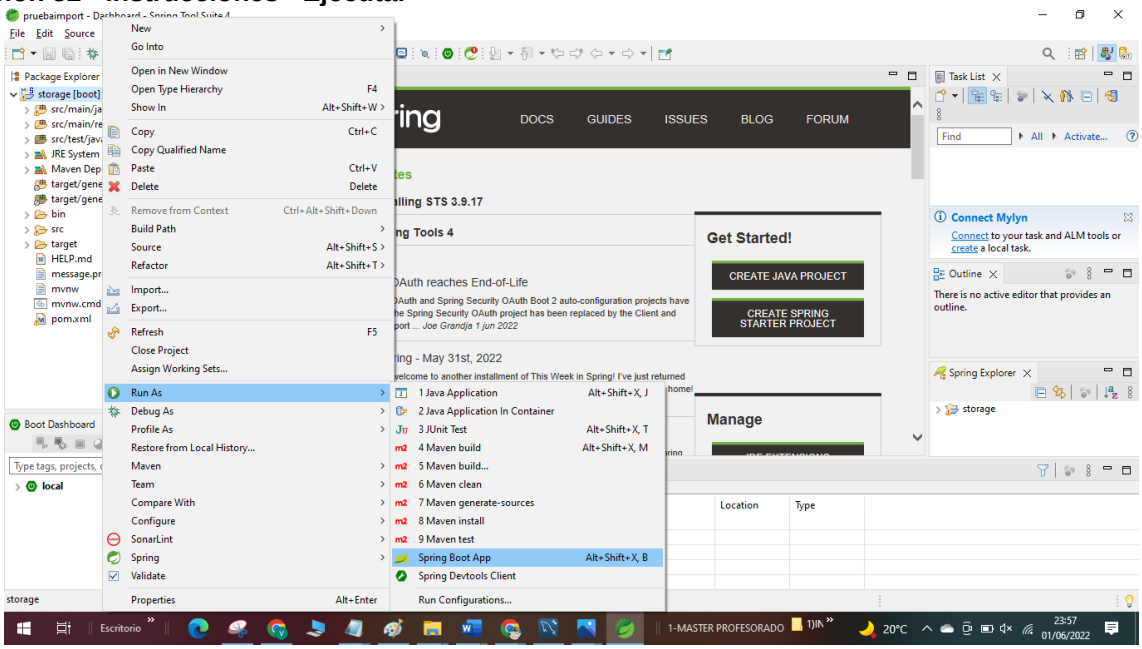
Tras seleccionar el proyecto que habíamos descargado, se cargará en eclipse.

A continuación, se creará el esquema de la base de datos MySQL.
Abrid MySQL Workbench y ejecutar las sentencias que hay en el fichero README de GitHub.

https://github.com/MiguelAngelOrgaz/garagestorage2/blob/master/README.md

Una vez creado el esquema y ejecutados los scripts de base de datos, para arrancar la aplicación, hacer clic derecho sobre el proyecto en Eclipse y pulsar en run as spring boot app

Ilustración 52 - Instrucciones - Ejecutar



Así se iniciará la aplicación sin necesidad de configurar ningún servidor de aplicaciones, pues ya tiene embebido un apache Tomcat.

Ahora bastaría con abrir el navegador y poner: http://localhost:8080/
Y nos aparecerá la página principal de la aplicación:

Ilustración 53 - Instrucciones - Login

Se podría entrar en la aplicación con el usuario y contraseña: admin / 12345678. O registrar un nuevo usuario cliente.

10. Conclusiones

Gracias a este trabajo de fin de grado, he tenido la oportunidad de crear una aplicación JEE desde 0, desde su toma de requisitos, aunque no hubiera usuarios a excepción de mí mismo, hasta su implementación, pasando por un análisis exhaustivo de las distintas funcionalidades que la componen.

Al haber elaborado una planificación realista, con las entregas que se habían especificado en cada una de las PEC del TFG, y sabiendo del tiempo que disponía para la elaboración de toda la documentación y la implementación de la aplicación, he podido realizar todas las tareas que me había propuesto.

He conseguido cumplir los objetivos iniciales que no solamente eran crear una aplicación desde cero, sino aprovechar la realización de este TFG para poder seguir adquiriendo nuevos conocimientos, como ha sido la utilización de Spring Security, que nunca antes había visto y que me ha parecido muy interesante el cómo se implementa la seguridad con Spring, y lo fácil que resulta una vez se configura todo correctamente.

Y se han cumplido los objetos iniciales del TFG, que consistían en realizar un trabajo en el que se abordaran las distintas tecnologías existentes relacionades con JavaEE, tal y como se puede apreciar en el capítulo de implementación visto anteriormente en la memoria.

Además, una espinita que he tenido siempre ha sido la parte visual, que no hacía más que usar JSP para crear las páginas. Y aquí he podido aprender a utilizar Thymeleaf, el cual me ha sorprendido gratamente.

Esa planificación realista que se hizo, también me hizo tener en cuenta varios aspectos que no abordé desde un principio, y que se podrían tener en mente para poder realizarlos en un futuro.

Por ejemplo, para un trabajo futuro, la aplicación se podría mejorar para hacerla compatible para poder trabajar de forma amigable con ella mediante cualquier dispositivo móvil. Incluso hasta se podría pensar en crear alguna aplicación móvil.

Además, la parte visual se podría mejorar, aunque para este trabajo de fin de grado, prefería centrarme en el desarrollo de la aplicación JEE y no tanto en el aspecto visual de la misma, debido al tiempo del que podría disponer.

11. Glosario

DTO: “Data Transfer Objet”, u objeto de transferencia de datos, es un objeto que indica cómo se envían los datos.

Entity: Es un objeto del dominio de persistencia que representa una tabla de una base de datos, con sus campos y relaciones y donde cada instancia que se tuviera de una entidad sería una fila de la tabla con la que está relacionada.

Figma: Es una herramienta de prototipado web, que fue usada en la asignatura de “Interacción persona ordenador” y que por su facilidad de uso he decidido usarla para realizar el prototipo de la aplicación.

JavaEE: Java Enterprise Edition es una plataforma para desarrollar aplicaciones en lenguaje Java.

JPA: “Java Persistent API”, es una especificación, con definición y regles que nos permiten construir la capa de persistencia.

ModelMapper: Es una librería java que nos permite copiar los datos de un objeto en otro. Se ha utilizado para poder pasar de un DTO a un Entity y viceversa.

MVC: Es el modelo vista controlador, que es una modelo arquitectónico de software para separar la interfaz y lógica de nuestra aplicación en tres partes. El modelo, con la lógica de negocio, la vista con la parte visual de la aplicación y el controlador que hace de intermediario entre los otros dos.

MySQL: Es un gestor de base de datos relacionales que utiliza el lenguaje SQL.

Spring: Es un framework para el desarrollo de aplicaciones Java de forma más sencilla.

Spring Boot: Es una tecnología que nos permite centrarnos en la implementación de nuestra aplicación, sin tener que centrarnos en configuraciones ni en el despliegue. Pues nos crea un contenedor en el que se incluye el servidor Tomcat para poder desplegar.

Spring Security: Es un módulo de Spring centrado en la seguridad de las aplicaciones. Incluyendo la autenticación y autorización para el acceso a las distintas partes de la aplicación, entre otras.

12. Anexos

El código fuente de la aplicación, se encuentra en un repositorio de GitHub público, para que pueda ser accedida por cualquiera que esté visualizando esta memoria:

https://github.com/MiguelAngelOrgaz/garagestorage2