Universitat Oberta de Catalunya
uoc.edu

StayFit!

Francisco Lifante Rivera
Grado en Ingeniería Informática
Java Enterprise Edition (JEE)

Consultor: Vicenç Font Sagristà

Profesor responsable de la asignatura: Santi Caballé Llobet

12 de junio de 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*StayFit!*
Nombre del autor:	*Francisco Lifante Rivera*
Nombre del consultor/a:	*Vicenç Font Sagristà*
Nombre del PRA:	*Santi Caballé Llobet*
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	*Grado en Ingeniería Informática*
Área del Trabajo Final:	*Java EE*
Idioma del trabajo:	*Castellano*
Palabras clave	*Java EE, TFG, WebApp*

Resumen del Trabajo

El presente proyecto tiene como objetivo principal la creación de una aplicación web para poner en práctica los conocimientos adquiridos durante el Grado, así como el itinerario escogido de desarrollo de *software*.

La idea de esta aplicación surge de la motivación personal de llevar un estilo de vida saludable y realizar ejercicio físico. De esta manera, la aplicación va dirigida a facilitar al usuario la gestión de su alimentación.

La metodología empleada para el desarrollo del proyecto ha sido estándar. Se ha comenzado con una planificación temporal, para proceder con la recogida de requisitos y posterior análisis y diseño de los diferentes componentes que conforman la aplicación. Por último, se procede a la implementación de los citados componentes.

El resultado obtenido ha sido una aplicación web que cumple con los requisitos y objetivos planteados al inicio del proyecto.

En conclusión, la realización de este TFG me ha servido para aprender a gestionar un proyecto por mí mismo y a mejorar en el desarrollo de *software*.

Abstract

The main aim of this project is the creation of a web application to put into practice the knowledge acquired during the Degree, as well as the chosen software development itinerary.

The idea of this application arises from the personal motivation of leading a healthy lifestyle and doing physical exercise. In this regard, the application is aimed at the user to facilitate the management of his own diet.

The methodology used for the project development has been the standard one. We have started with a temporal planning, to proceed with the collection of requirements and subsequent analysis and design of the different components that form the application. Finally, the implementation of these components is carried out.

The result obtained has been a web application that meets the requirements and objectives set at the beginning of the project.

In summary, the realization of this TFG has served me to learn on how to manage a project by myself and to improve in the software development.

ÍNDICE

Lista de ilustraciones

Lista de tablas

1. Introducción

La realización del presente TFG sirve para completar mis estudios en el Grado de Ingeniería Informática, concretamente en el itinerario de Ingeniería del Software. Con motivo de este itinerario, se realizará un proyecto de desarrollo en el que se va a proceder a la creación de una plataforma web que permitirá a un usuario calcular sus necesidades calóricas y la creación de un menú diario para ajustar su alimentación a estas necesidades, previamente calculadas por la propia aplicación.

Para la realización de este proyecto se va a utilizar la tecnología Java Enterprise Edition (Java EE), que ya se ha estudiado a lo largo de varias asignaturas en mi itinerario durante el Grado, y me permitirá aplicar muchos de los conocimientos que he ido adquiriendo a lo largo de estos años al realizar mis estudios de Ingeniería Informática.

1.1. Contexto y justificación del trabajo

La idea de realizar este tipo de aplicación surge de mi afición y necesidad de realizar deporte, llevar un estilo de vida saludable y una buena alimentación, ya que esto repercutirá en la salud y en el bienestar propio.

La necesidad que se pretende cubrir es poder conocer fácilmente y de manera precisa la cantidad de calorías que necesita un individuo según sus condiciones particulares como: género, edad, peso, altura y actividad física. Con una sencilla aplicación es posible conocer estas cantidades recomendadas, ya que hay mucha desinformación y se suele sobreestimar nuestras necesidades diarias. De esta manera, podemos llevar un control en la ingesta de alimentos.

Considero la alimentación y el ejercicio un tema relevante y de bastante actualidad, ya que en los últimos años ha proliferado un culto por el físico. Como muestra, podemos observar el gran auge de los gimnasios y en particular del espectacular boom del Crossfit. ¿Quién no tiene un conocido en su círculo de amigos hoy en día que lo practique? Además, cada día hay más productos que nos aportan un extra de proteínas, convirtiéndose ya en algo habitual (y casi una obsesión), el tomar esta clase de alimentos si queremos mantenernos en forma o perder algún kilo extra.

Por lo tanto, este proyecto pretende ser una herramienta para aquellas personas que no sepan muy bien cómo llevar una alimentación controlada, y puedan comenzar a conocer sus necesidades energéticas y ver lo que consumen a diario.

1.2. Objetivos del trabajo

El objetivo principal de este TFG es desarrollar una aplicación web empleando Java EE, que será accesible a través de Internet. En concreto, se trata de una aplicación para gestionar la cantidad de alimentos consumidos por los usuarios a lo largo del día.

Un objetivo desde el punto de vista de los usuarios es el de crear una aplicación que satisfaga ciertas necesidades. En el caso de StayFit, trata de controlar el peso de un usuario a lo largo del tiempo y mejorar su condición física y, por ende, su salud.

Como objetivo personal, me planteo afianzar y ampliar los conocimientos adquiridos a lo largo del Grado, así como aplicarlos a la realización de un proyecto completo.

1.3. Enfoque y método seguido

Este trabajo pretende crear una nueva aplicación similar a otras alternativas que existen en el mercado, como MyFitnessPal o FatSecret. Por lo tanto, se puede decir que sería adaptar un producto existente a mi propia versión. Lógicamente, será un desarrollo de una única persona sin experiencia previa, por lo que obviamente no podré conseguir el mismo nivel en la aplicación que las citadas anteriormente. No obstante, servirá como experiencia para aplicar los conocimientos adquiridos a lo largo del Grado.

1.4. Planificación del Trabajo

A continuación, presentaré la planificación elaborada para llevar a cabo la realización del proyecto y la consecución de los objetivos en el tiempo. Así como el diagrama de Gantt que confeccioné para disponer de una guía que sirviera de referencia para poder gestionar el tiempo. Además, presentaré una lista de recursos que han sido imprescindibles para realizar el trabajo.

1.4.1. Planificación temporal del trabajo

La planificación inicial del trabajo se ha dividido en base a las sucesivas entregas que se debían de realizar como parte de la evaluación continua de la asignatura y, es la siguiente:

Tabla 1. Planificación temporal del trabajo

DOCUMENTO	INICIO	ENTREGA	ACTIVIDAD
PEC 1	20/02/2020	06/03/2020	Plan de trabajo
PEC 2	07/03/2020	10/04/2020	Requerimientos, análisis y diseño
PEC 3	11/04/2020	29/05/2020	Implementación
Memoria y presentación	30/05/2020	12/06/2020	Memoria y presentación

1.4.2. Diagrama de Gantt

Para lograr los objetivos y gestionar la planificación y ejecución de las diferentes fases de proyecto, se elaboró el siguiente diagrama de Gantt, utilizando para ello el software en línea de Teamgantt:

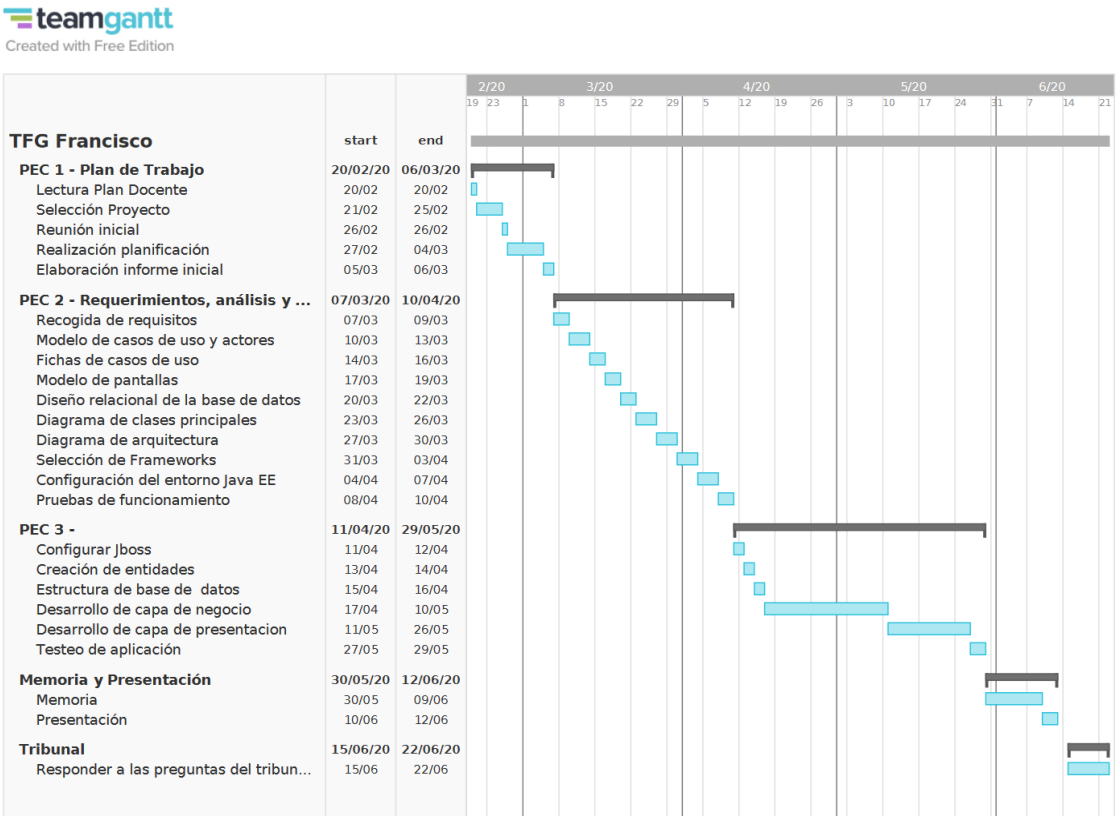


Ilustración 1. Diagrama de Gantt

1.4.3. Recursos empleados

Los recursos que he empleado para la realización del proyecto han sido los siguientes:

Tabla 2. Recursos empleados

RECURSO	VERSIÓN
PC sobremesa	Intel Core I7-5820k, 32 GB RAM
Sistema Operativo	Windows 10 Pro
Entorno de desarrollo (IDE)	Eclipse Java EE, versión 2018-09 (4.9.0)
Java Enterprise Edition	JSF, Managed Bean, JPA
Sistema gestor de base de datos	PostgreSQL 10, versión 10.10.1
Aplicación de gestión de base de datos	pdAdmin 4, versión 4.5
Servidor de aplicaciones	Wildfly 14.0.1
Automatización proceso de compilación	Apache Ant
Automatización descarga dependencias	Maven
Herramienta mapeo ORM	Hibernate (JPA 2.1)
Framework CSS para estilos	Bootstrap
Biblioteca de componentes para JSF	Primefaces, versión 6.2
Herramienta diagrama Gantt	Teamgantt
Herramienta de diseño de software	MagicDraw 18.5

1.5. Breve sumario de productos obtenidos

Los productos obtenidos durante el desarrollo del presente TFG han sido:

- **Plan de trabajo:** objetivo del proyecto, requisitos funcionales, tecnologías propuestas, planificación temporal del proyecto.

- **Documentación de requerimientos, análisis y diseño:** modelo de casos de uso y actores, fichas de casos de uso, prototipo de la interfaz de usuario, diseño relacional de la base de datos, diagrama de clases y diagrama de arquitectura.

- **Proyecto Java EE:** implementación de la aplicación desarrollada.

- **Manual de despliegue:** documento para explicar puesta en funcionamiento
- del proyecto para su prueba.

- **Presentación:** documento de presentación del proyecto en PowerPoint y vídeo de presentación virtual de la aplicación.

- **Memoria:** el presente documento.

1.6. Breve descripción de los otros capítulos de la memoria

En esta sección voy a describir el resto de los capítulos que conforman este documento:

- **Análisis:** en este capítulo se describen las necesidades del proyecto, requisitos no funcionales, requisitos funcionales y casos de uso.
  Este apartado es la base que cimienta la futura aplicación web, ya que contiene partes fundamentales para la construcción de este proyecto.

- **Diseño:** este apartado contiene: actores, entidades y objetos de valor, diagramas de clases, diagrama relacional de base de datos y diagrama de arquitectura en tres capas.

- **Implementación:** en esta sección se incluye: una breve explicación de las diferentes capas de la aplicación, así como el resultado final de la implementación a través de las distintas pantallas de la aplicación.

- **Valoración económica y viabilidad del producto:** este capítulo contiene una estimación del posible coste económico del desarrollo y un breve análisis de su viabilidad en caso de su lanzamiento al mercado.

- **Trabajo futuro y posibles mejoras:** en este apartado se detallan posibles mejoras y adecuaciones que se podrían implementar en un
  futuro para la aplicación desarrollada.

- **Conclusiones:** este capítulo contiene una valoración y reflexión personal de lo que ha supuesto el desarrollo de este TFG, así como su relación con las diferentes asignaturas que conforman el Grado de Ingeniería Informática.

- **Glosario:** en esta sección se recoge la definición de términos utilizados en la confección del presente documento.

- **Bibliografía:** este capítulo contiene la bibliografía utilizada para el desarrollo del presente TFG.

- **Anexos:** en este apartado se incluye el tutorial de instalación del entorno necesario para el despliegue de la aplicación.

2. Análisis

2.1. Requisitos no funcionales

- Portabilidad de la aplicación, ya que será ejecutable en cualquier navegador web independientemente del sistema operativo del usuario.
- Será una aplicación *user-friendly*, es decir, sencilla de utilizar, intuitiva y amigable de cara a cualquier tipo de usuario.
- La aplicación se ha construido siguiendo el patrón MVC (Modelo Vista Controlador). Dicha arquitectura separa los datos y la lógica de negocio de la aplicación de su representación. Se basa en tres componentes: modelo, vista y controlador.
- La aplicación utilizará una base de datos PostgreSQL.
- La aplicación se desplegará mediante el servidor de aplicaciones, Wildfly.

2.2. Requisitos funcionales

- Los usuarios no registrados deben poder registrarse en el sistema.
- Los usuarios deben poder identificarse en el sistema (iniciar sesión).
- Los administradores deben poder identificarse en el sistema (iniciar sesión).
- Los usuarios deben poder salir del sistema (cerrar sesión).
- Los administradores deben poder salir del sistema (cerrar sesión).
- Los administradores pueden dar de alta un alimento.
- Los administradores pueden modificar un alimento.
- Los administradores pueden eliminar un alimento.
- Los administradores pueden consultar la ficha de un alimento.
- Los administradores pueden consultar el listado de alimentos.
- Los administradores pueden dar de alta una categoría de alimento.
- Los administradores pueden modificar una categoría de alimento.
- Los administradores pueden eliminar una categoría de alimento.
- Los administradores pueden consultar la ficha de una categoría de alimento.
- Los administradores pueden consultar el listado de categorías de alimentos.
- Los administradores pueden gestionar las cuentas de administrador.
- Los usuarios pueden introducir/modificar su peso.
- Los usuarios pueden introducir/modificar su altura.
- Los usuarios pueden introducir/modificar su edad.
- Los usuarios pueden introducir/modificar su sexo.
- Los usuarios pueden introducir/modificar el nivel de actividad física que realizan.
- Los usuarios pueden consultar la ficha de un alimento.
- Los usuarios pueden consultar el listado de alimentos.
- Los usuarios pueden gestionar (agregar y eliminar) alimentos a un menú diario.
- Los usuarios pueden ver un resumen diario con las calorías, proteínas, grasas y carbohidratos ingeridos hasta el momento (conforme introduce alimentos consumidos a lo largo del día se va actualizando el resumen).
- Los usuarios pueden consultar los menús que han introducido a lo largo del tiempo.
- Los usuarios pueden consultar un histórico con la evolución de su peso.

2.3. Casos de uso

2.3.1. Diagrama de casos de uso

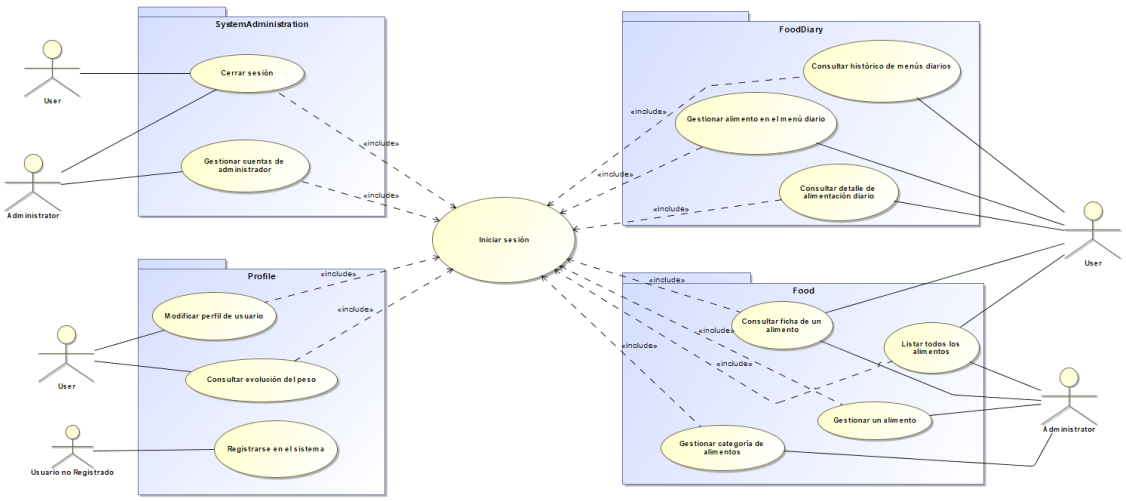


Ilustración 2. Diagrama de casos de uso

2.3.2. Fichas de casos de uso

Tabla 3. Caso de uso 01

Caso de uso	01 - Registrarse en el sistema
Actor principal	Usuario no registrado
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	Usuario
***Stakeholders* e intereses**	Usuario: el usuario desea poder acceder al sistema y debe registrarse para ello
Precondición	No estar previamente registrado
Garantías en caso de éxito	El sistema dará de alta al usuario y le permitirá acceder al mismo
Escenario principal de éxito	1) El usuario no registrado en el sistema pulsa sobre el botón de registro 2) El usuario rellena los campos del formulario de registro 3) El sistema confirma el registro de usuario

Tabla 4. Caso de uso 02

Caso de uso	02 – Iniciar sesión en el sistema
Actor principal	Usuario / Administrador
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	Usuario
***Stakeholders* e intereses**	Usuario y Administrador: ambos tienen un interés común. Se trata de autenticarse en el sistema y poder hacer uso de este
Precondición	No estar autenticado previamente
Garantías en caso de éxito	Quedar autenticado en el sistema
Escenario principal de éxito	1) El usuario/administrador accede a la plataforma web 2) El usuario/administrador introduce sus credenciales en el campo habilitado para ello 3) El sistema comprueba las credenciales y si son correctas, permite al usuario o administrador acceder al sistema con sus datos
Escenarios alternativos	3.1) El sistema no halla el nombre de usuario o contraseña introducido y no permite el acceso al usuario/administrador

Tabla 5. Caso de uso 03

Caso de uso	03 – Cerrar sesión en el sistema
Actor principal	Usuario / Administrador
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	Usuario
***Stakeholders* e intereses**	Usuario y Administrador: ambos tienen un interés común. Se trata de cerrar sesión en el sistema y poder hacer uso de este
Precondición	Estar autenticado en el sistema
Garantías en caso de éxito	El usuario o administrador cierra sesión en el sistema
Escenario principal de éxito	1) El usuario o administrador pulsa en el botón habilitado para cerrar la sesión actual 2) El sistema cierra la sesión del usuario o administrador y lleva de vuelta a la página principal de la plataforma

Tabla 6. Caso de uso 04

Caso de uso	04 – Gestionar un alimento
Actor principal	Administrador
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Administrador: es el encargado de gestionar los alimentos que hay en el sistema, por lo que debe poder crear, modificar y eliminar los diferentes productos almacenados en la base de datos Usuario: le interesa que haya multitud de alimentos disponibles en el sistema para la conformación de su menú diario y así disponer de una mayor cantidad de opciones
Precondición	Ser Administrador y estar autenticado en el sistema
Garantías en caso de éxito	Se habrá creado un nuevo alimento, se habrá modificado algún alimento ya existente o se eliminará un alimento de la base de datos
Escenario principal de éxito	1) El administrador navega a la sección de alimentos 2) El administrador pulsa sobre el botón de añadir alimento 3) El sistema le muestra la ficha para rellenar los datos de un alimento 4) El administrador completa los datos del alimento a crear y pulsa sobre el botón de añadir 5) El sistema crea el alimento en la base de datos y se completa el caso de uso
Escenarios alternativos	2.a) El administrador pulsa sobre el botón de modificar alimento 2.a.1) El sistema le muestra la ficha del alimento con los campos disponibles para modificar los datos 2.a.2) El administrador modifica los datos que necesita del alimento y pulsa en el botón de guardar 2.a.3) El sistema guarda los datos en base de datos y se completa el caso de uso 2.b) El administrador pulsa sobre el botón de eliminar alimento

	2.b.1) El sistema le pregunta al administrador si realmente quiere eliminar el alimento seleccionado 2.b.2) El administrador pulsa sobre el botón “Si” para eliminar el producto 2.b.3) El sistema elimina el alimento de la base de datos y se completa el caso de uso 2.c.2) El administrador pulsa sobre el botón “No” porque no quiere eliminar ese producto realmente y el caso de uso termina

Tabla 7. Caso de uso 05

Caso de uso	05 – Gestionar una categoría de alimento
Actor principal	Administrador
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Administrador: es el encargado de gestionar las categorías de alimentos que hay en el sistema, por lo que debe poder crear, modificar y eliminar las diferentes categorías en la base de datos Usuario: le interesa que los diferentes alimentos tengan una categoría para poder ordenar los productos por dicha categoría y poder organizar la información disponible
Precondición	Ser Administrador y estar autenticado en el sistema
Garantías en caso de éxito	Se habrá creado una nueva categoría de alimento, se habrá modificado alguna categoría ya existente o se eliminará una categoría de la base de datos
Escenario principal de éxito	1) El administrador navega a la sección de categoría de alimentos 2) El administrador pulsa sobre el botón de añadir nueva categoría 3) El sistema le muestra la ficha para rellenar los datos de categoría 4) El administrador completa los datos de la categoría a crear y pulsa sobre el botón de añadir 5) El sistema crea dicha categoría en la base de datos y se completa el caso de uso

Escenarios alternativos	2.a) El administrador pulsa sobre el botón de modificar categoría de alimento 2.a.1) El sistema le muestra la ficha de categoría con los campos disponibles para modificar los datos 2.a.2) El administrador modifica los datos de dicha categoría y pulsa en el botón de guardar 2.a.3) El sistema guarda los datos en base de datos y se completa el caso de uso 2.b) El administrador pulsa sobre el botón de eliminar categoría de alimento 2.b.1) El sistema le pregunta al administrador si realmente quiere eliminar la categoría seleccionada 2.b.2) El administrador pulsa sobre el botón "Si" para eliminar la categoría escogida 2.b.3) El sistema elimina la categoría de la base de datos y se completa el caso de uso 2.c.2) El administrador pulsa sobre el botón "No" porque no quiere eliminar esa categoría realmente y el caso de uso termina

Tabla 8. Caso de uso 06

Caso de uso	06 – Gestionar cuentas de administrador
Actor principal	Administrador
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Administrador: el administrador es el responsable de la administración y organización del sistema. En este caso, se ocupa de dar de alta a otros administradores o eliminar alguno de los ya existentes
Precondición	Ser administrador y estar autenticado en el sistema
Garantías en caso de éxito	El administrador podrá dar de alta o eliminar a otros administradores
Escenario principal de éxito	1) El administrador navega a la sección de gestión de administradores 2) El sistema le presenta el listado de administradores registrados en el mismo

	3) El administrador elige una de las opciones disponibles (añadir, modificar o eliminar administradores) 4) Una vez el administrador elige una de las opciones, se le muestra la pantalla correspondiente a la acción escogida que le permite realizar la tarea 5) El sistema guarda los cambios producidos y se termina el caso de uso

Tabla 9. Caso de uso 07

Caso de uso	07 – Modificar perfil de usuario
Actor principal	Usuario
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Usuario: su interés radica en poder guardar sus datos para que la aplicación calcule sus necesidades basales y le pueda guiar para conseguir sus objetivos de reducción, mantenimiento o aumento de peso
Precondición	Ser usuario registrado y estar autenticado en el sistema
Garantías en caso de éxito	El usuario podrá modificar sus datos de usuario y sus características para el cálculo basal
Escenario principal de éxito	1) El usuario navega a la sección de perfil para poder introducir o modificar las diferentes características (peso, altura, edad, sexo, nivel de actividad física) 2) El sistema le permite introducir o modificar dichos datos 3) El usuario modifica los diferentes atributos y confirma los cambios introducidos 4) El sistema guarda los datos y finaliza el caso de uso

Tabla 10. Caso de uso 08

Caso de uso	08 – Consultar ficha de alimento
Actor principal	Usuario
Actores de apoyo	Administrador
Ámbito	Sistema
Nivel de objetivo	General

***Stakeholders* e intereses**	Usuario: el interés radica en que el usuario busca conocer la información nutricional del alimento Administrador: el administrador consultará la ficha de alimento para comprobar que el alimento ha sido dado de alta correctamente
Precondición	Ser usuario registrado y estar autenticado en el sistema
Garantías en caso de éxito	El usuario puede consultar la ficha de un alimento con su información
Escenario principal de éxito	1) El usuario navega a la sección de alimentos, busca el alimento que quiere consultar y pulsa sobre el mismo 2) El sistema le muestra la ficha de alimento, donde se ven todos los datos relativos al mismo 3) El usuario puede observar la información del alimento seleccionado y el caso de uso finaliza

Tabla 11. Caso de uso 09

Caso de uso	09 – Gestionar alimento en el menú diario
Actor principal	Usuario
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Usuario: el interés del usuario en conformar un menú diario con los diferentes alimentos que ha consumido en el día y así conocer las calorías consumidas
Precondición	Ser usuario registrado y estar autenticado en el sistema
Garantías en caso de éxito	El usuario tendrá registrados todos los alimentos que ha consumido en el día y conocerá el total de calorías que le aportan
Escenario principal de éxito	1) El usuario se sitúa en la pantalla del diario de comidas, busca el alimento que va a consumir (o ha consumido) y lo agrega a una de las 4 comidas del día (desayuno, comida, cena o snacks) 2) El sistema registra el alimento que el usuario ha agregado en la hora del día correspondiente 3) El usuario puede observar el alimento agregado y el resumen de calorías que le aporta

Escenarios alternativos	1.a) El usuario se sitúa en la pantalla de diario de comidas y selecciona un alimento del que quiere modificar la cantidad consumida 1.a.1) El sistema le muestra la ficha del alimento con la cantidad escogida como consumición y le permite al usuario modificarla 1.a.2) El usuario modifica la cantidad a consumir del alimento y pulsa el botón de guardar 1.a.3) El sistema devuelve al usuario a la pantalla anterior y guarda los datos en base de datos. Volvemos al paso 3) 2.b) El usuario pulsa sobre el botón de eliminar alimento del diario de comidas 2.b.1) El sistema le pregunta al usuario si realmente quiere eliminar el alimento seleccionado 2.b.2) El usuario pulsa sobre el botón “Si” para eliminar el producto 2.b.3) El sistema elimina el alimento del diario y se completa el caso de uso 2.c.2) El usuario pulsa sobre el botón “No” porque no quiere eliminar ese producto realmente y el caso de uso termina

Tabla 12. Caso de uso 10

Caso de uso	10 – Consultar detalle de alimentación diario
Actor principal	Usuario
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Usuario: el usuario está interesado en conocer los detalles de la comida que habrá consumido en un día concreto
Precondición	Ser usuario registrado y estar autenticado en el sistema
Garantías en caso de éxito	El usuario podrá ver el resumen de lo que ha comido a lo largo del día
Escenario principal de éxito	1) El usuario consulta desde el diario de comidas en la parte superior el resumen completo de las calorías, proteínas, grasas y carbohidratos consumidos con todos los alimentos introducidos ese día en el diario de comidas

Tabla 13. Caso de uso 11

Caso de uso	11 – Consultar histórico de menús diarios
Actor principal	Usuario
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Usuario: el usuario para poder contrastar los diferentes menús que ha ido guardando a lo largo del tiempo
Precondición	Ser usuario registrado y estar autenticado en el sistema
Garantías en caso de éxito	El usuario podrá consultar el menú guardado en fechas anteriores
Escenario principal de éxito	1) El usuario desde la pantalla de diario de alimentos dispone de una opción para consultar otros días. Cuando quiera consultar un día, introduce la fecha y pulsa el botón 2) El sistema carga el menú del día seleccionado y muestra al usuario la información del diario del día escogido 3) El caso de uso finaliza

Tabla 14. Caso de uso 12

Caso de uso	12 – Consultar evolución del peso
Actor principal	Usuario
Actores de apoyo	Ninguno
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Usuario: el usuario tiene interés en conocer la evolución a lo largo del tiempo de su peso y así comprobar si le está resultando positivo el uso de la plataforma para controlar sus ingestas y conseguir el objetivo que se haya propuesto
Precondición	Ser usuario registrado y estar autenticado en el sistema
Garantías en caso de éxito	El usuario puede consultar la evolución de su peso a lo largo del tiempo
Escenario principal de éxito	1) El usuario pulsa sobre el menú de histórico de peso 2) El sistema le muestra al usuario la pantalla con la información de su peso

	3) El usuario tiene la opción de introducir su nuevo peso en caso de que haya cambiado 4) El caso de uso finaliza
Escenarios alternativos	3.a) El usuario pulsa el botón para introducir un nuevo peso 3.a.1) El sistema le muestra al usuario una pantalla para cambiar su peso 3.a.2) El usuario registra su nuevo peso. Se vuelve al paso 4

Tabla 15. Caso de uso 13

Caso de uso	**13 – Listar todos los alimentos**
Actor principal	Usuario
Actores de apoyo	Administrador
Ámbito	Sistema
Nivel de objetivo	General
***Stakeholders* e intereses**	Usuario: el usuario tiene interés en conocer los diferentes alimentos que están a su disposición para poder agregarlos en la confección de su menú diario Administrador: el administrador podrá listar los alimentos para comprobar que se han agregado correctamente
Precondición	Ser usuario registrado y estar autenticado en el sistema
Garantías en caso de éxito	El usuario puede consultar la evolución de su peso a lo largo del tiempo
Escenario principal de éxito	1) El usuario pulsa sobre el menú de alimentos 2) El sistema le muestra al usuario la pantalla donde aparece un listado de los diferentes alimentos disponibles en la base de datos 3) El usuario obtiene el listado de alimentos y puede consultar los macronutrientes y calorías de este 4) El caso de uso finaliza

3. Diseño

En este capítulo se describen las entidades y objetos de valores, junto con el diagrama de clases que modelan la estructura del sistema mostrando sus atributos y las relaciones entre los objetos. Además, se incluye el diseño relacional de la base de datos, junto con un esquema visual. Finalmente, se describe la arquitectura de la aplicación.

Actores

- User
- Administrator

Entidades

- User
- Administrator
- Food
- FoodCategory
- ServingSize (donde se registra el tamaño de una porción de alimento consumida)
- FoodDiary (el diario de alimentos donde se guardan las porciones consumidas por un usuario en un día concreto)
- BasalMetabolicRateCalculator (clase que calcula las necesidades basales del usuario en base a sus características)
- HistoricalWeight (clase que registra los diferentes pesajes de un usuario a lo largo del tiempo)

Objetos de valores

- Email
- Password
- Roles
- PhysicalActivityType
- SexType
- MealsType

3.1. Diagrama de clases

El diagrama de clases que he obtenido ha sido el siguiente:

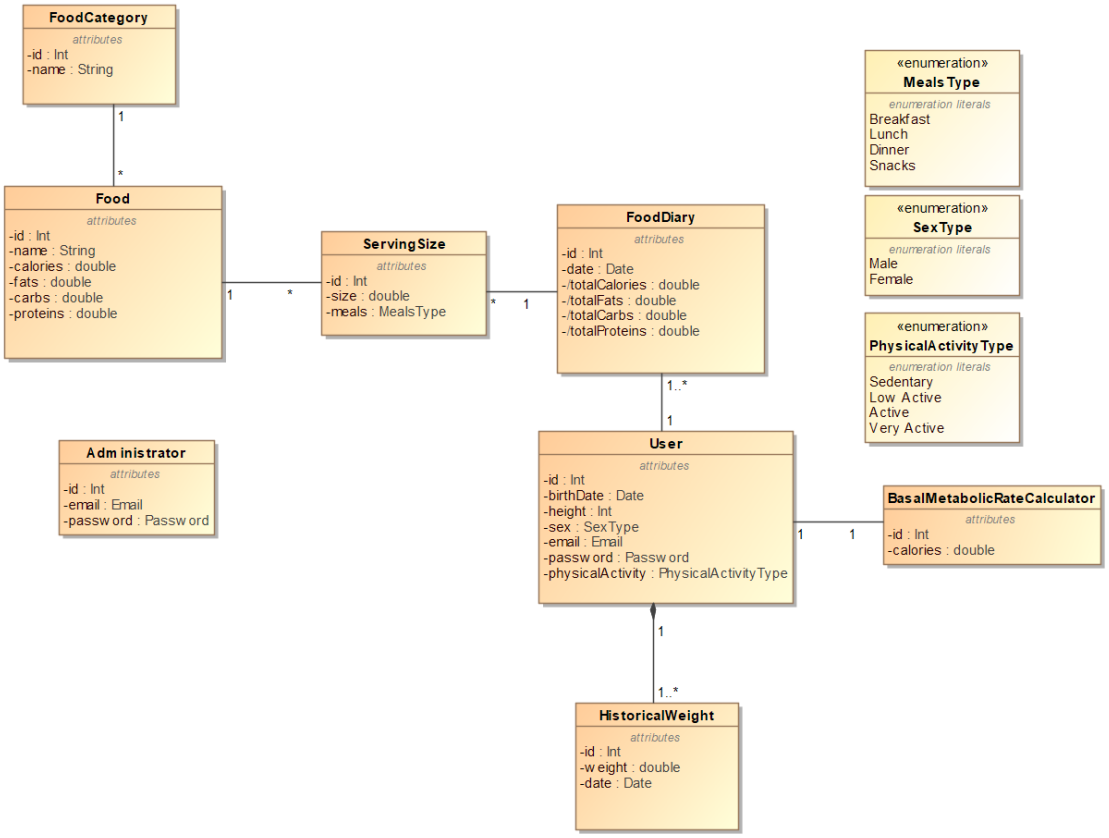


Ilustración 3. Diagrama de clases

Ciertos aspectos que aclarar en cuanto al diseño, es que he utilizado para modelar la altura de usuario, un tipo entero, que, si bien podría haber utilizado un tipo *double* al igual que he hecho para el peso, decidí simplificar y limitar la altura a únicamente valores enteros. Aunque habría sido prácticamente igual el haber utilizado un valor con posibilidad de números decimales.

Por otro lado, he utilizado una relación 1 a 1 entre usuario y la clase que realiza el cálculo de la tasa metabólica ya que esta se va a ir actualizando conforme el usuario cambie sus características. Otra opción habría sido modelarla 1 a muchos, agregando un atributo de fecha y así poder tener un histórico de tasas metabólicas en el tiempo, pero para las necesidades este proyecto no lo creí necesario.

Respecto a las clases de tipo enumeración, he considerado que serán fijos a lo largo del tiempo, que no será necesario agregar más en el futuro y por eso se ha optado por modelizarlo de esta manera. Si se hubiera creído que podrían cambiar, se debería haber aplicado un *@Converter* de manera que pudiéramos obtener y escribir dichos atributos en la base de datos, de manera que obtendríamos la posibilidad de modificar dichos tipos enumerados y adaptarlos a las necesidades cambiantes del proyecto. En este caso, no ha sido necesario recurrir a ello, ya que serán valores inmutables.

3.2. Diseño relacional de la base de datos

El diseño relacional de la base de datos es el siguiente:

FoodCategory (id, **name**)

Food (id, **name**, **calories**, **fats**, **carbs**, **proteins, foodCategoryID**)

{foodCategoryID} is foreign key to FoodCategory

ServingSize (id, **amount, meals**, **foodID, foodDiaryID**)

{foodID} is foreign key to Food

{foodDiaryID} is foreign key to FoodDiary

FoodDiary (id, **date**, **targetCalories**, **userID**)

{userID} is foreign key to User

Administrator (id, **email**, **password**)

User (id, **birthDate**, **height**, **sex**, **email**, **password**, **physicalActivity**)

BasalMetabolicRateCalculator (id, **calories**)

{id} is foreign key to User

HistoricalWeight (id, **userID, weight**, **date**)

{userID} is foreign key to User

Nota: los atributos que son clave primaria se han subrayado. Los atributos **que no pueden ser NULOS** están en negrita.

A continuación, se muestra el esquema en forma de gráfico:

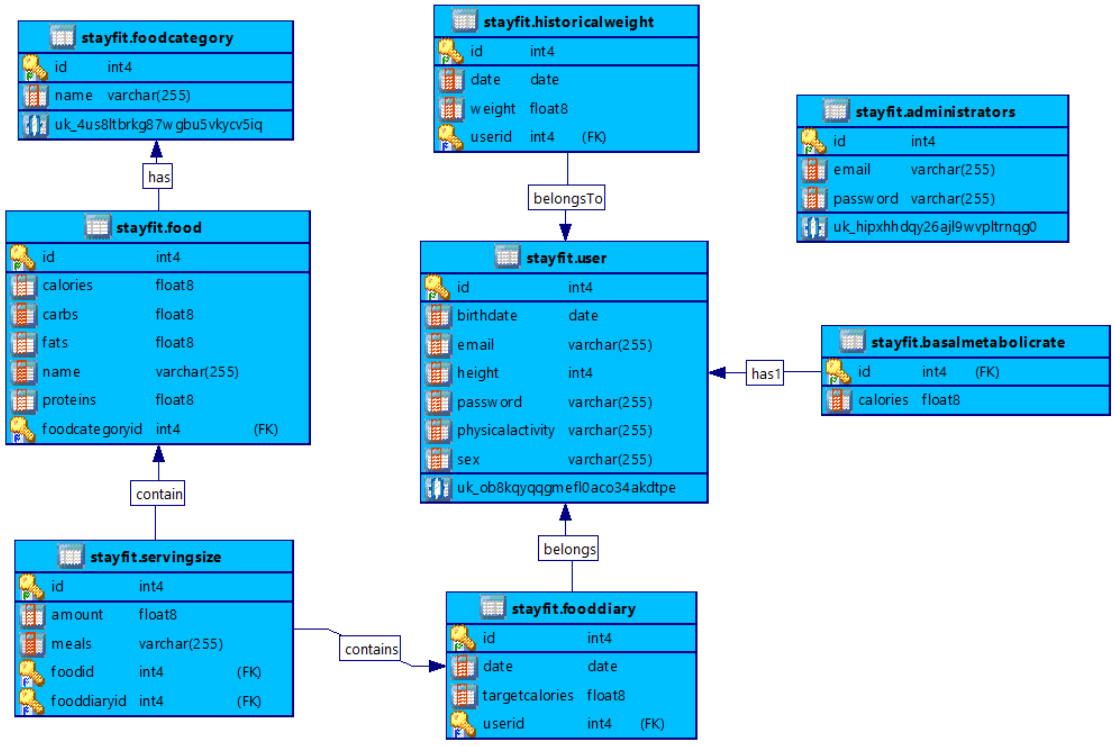


Ilustración 4. Diseño relacional de la base de datos

3.3. Diagrama de arquitectura

El proyecto emplea el patrón de arquitectura en tres capas. Las tres capas de este modelo son: Presentation (presentación), Business (negocio) e Integration (integración o persistencia).

3.3.1. Punto de vista de la computación

El siguiente esquema representa el punto de vista de la computación, donde se ha omitido el detalle de las operaciones (métodos) en las interfaces por temas de legibilidad:

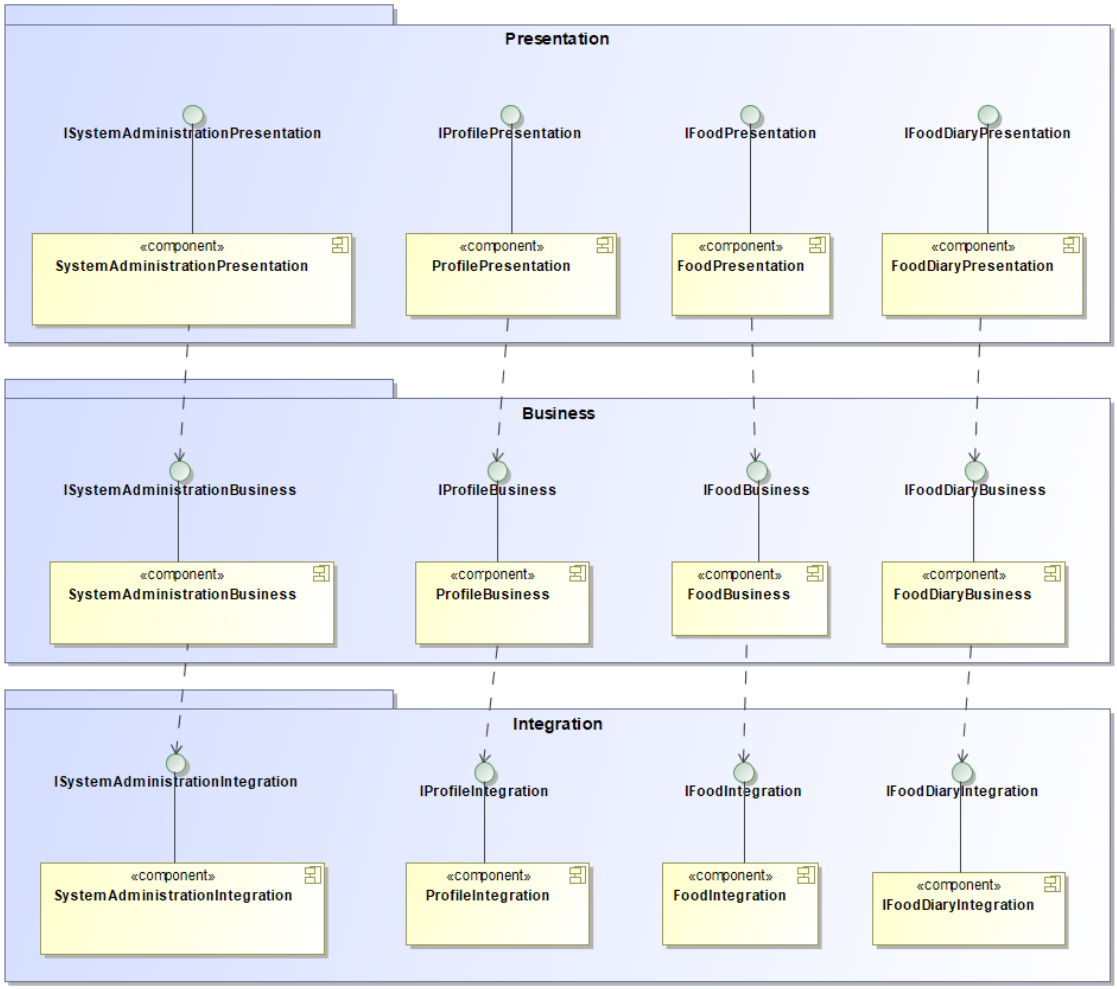


Ilustración 5. Punto de vista de la computación

Las operaciones o métodos que antes se habían colapsado con la intención de disponer de una mayor legibilidad se pueden observar en las siguientes figuras:

3.3.1.1. Capa de presentación

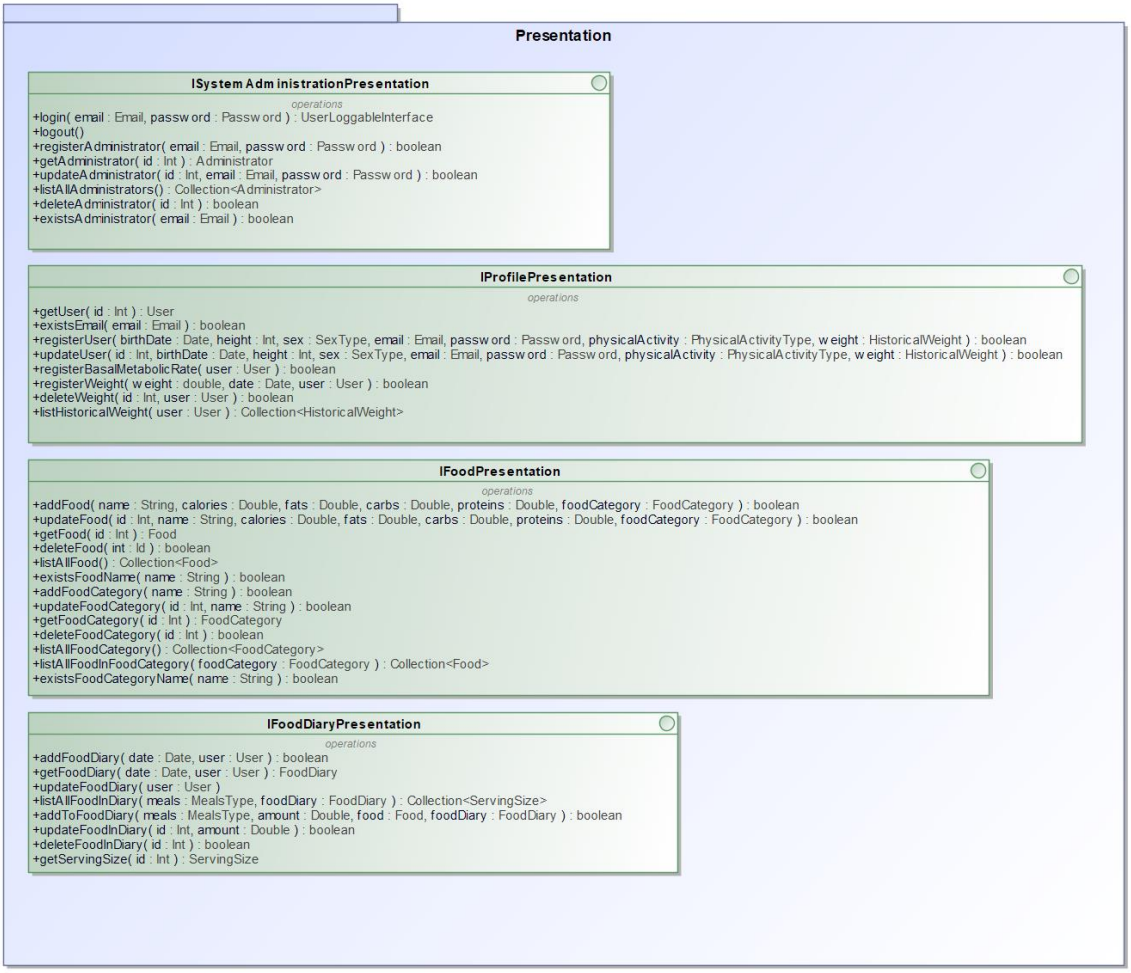


Ilustración 6. Punto de vista de la computación. Presentación.

3.3.1.2. Capa de negocio

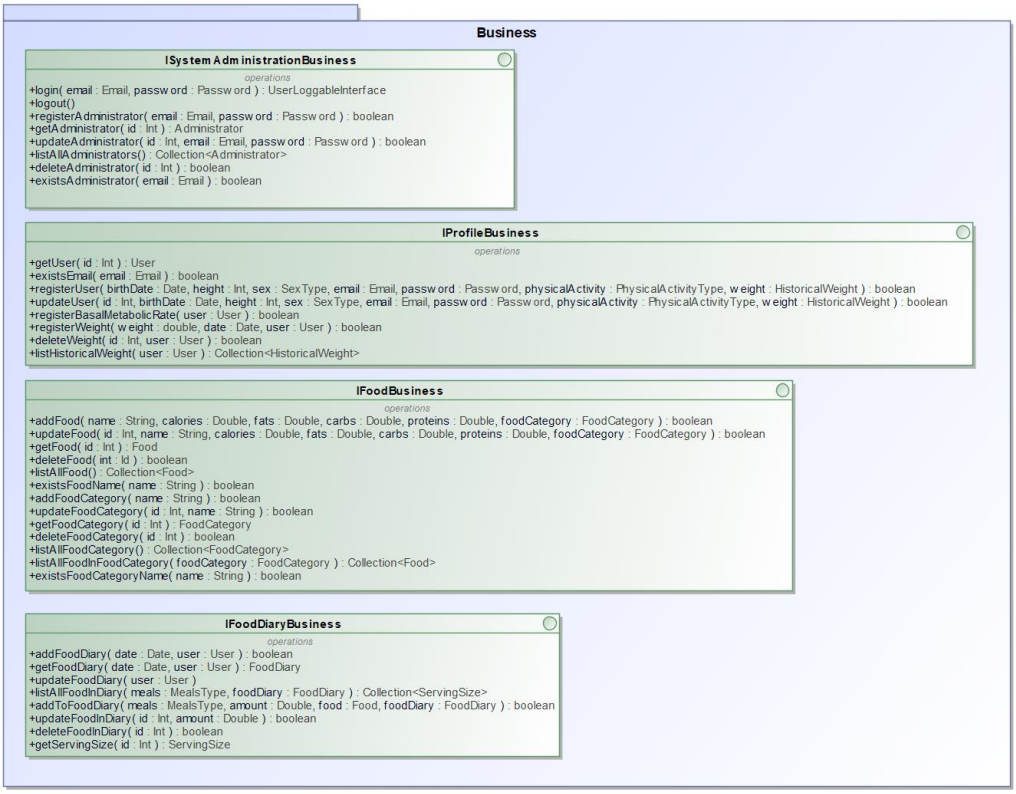


Ilustración 7. Punto de vista de la computación. Negocio.

3.3.1.3. Capa de integración (persistencia)

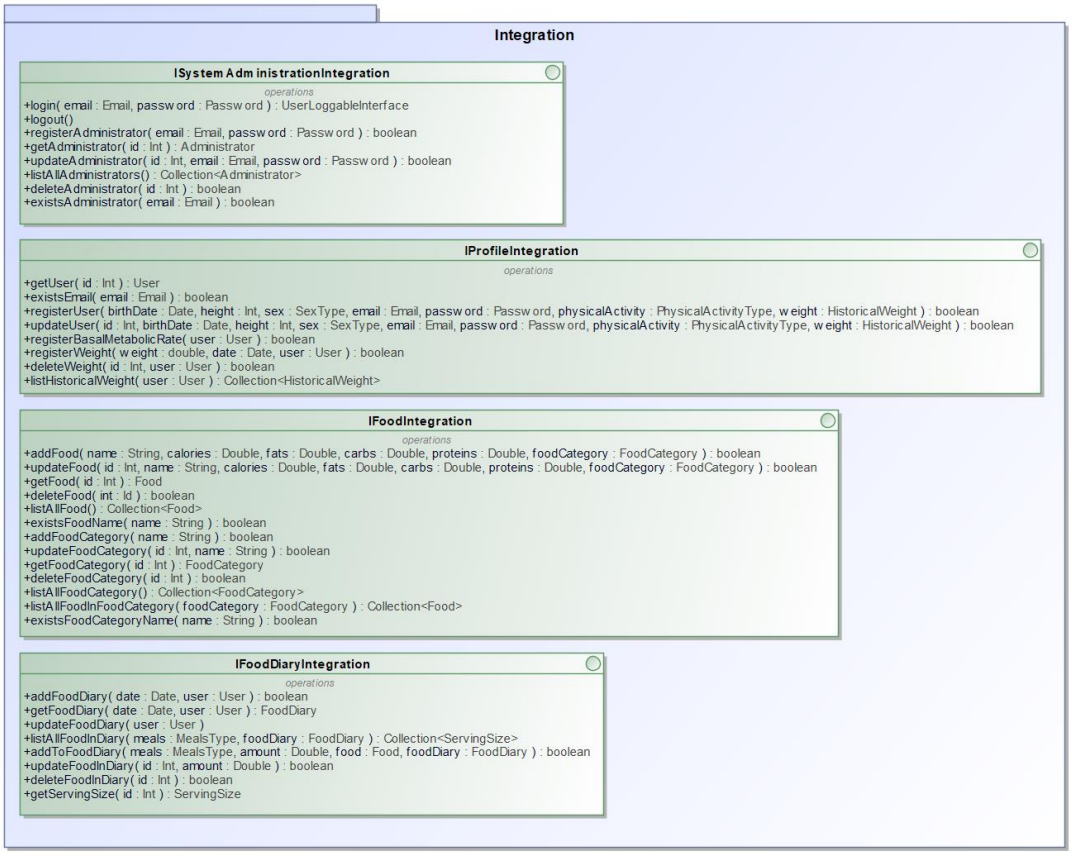


Ilustración 8. Punto de vista de la computación. Integración.

Ahora debemos de ir refinando los componentes del proyecto que se han presentado anteriormente, ya que son el punto de partida del proyecto, es decir, nuestras especificaciones iniciales. Debemos de conseguir un diagrama de componentes de *software*, por lo que, a continuación, iré presentando las diferentes refinaciones hasta conseguir dicho diagrama.

3.3.2. Refinamiento desde el punto de vista de la computación

3.3.2.1. Capa de presentación

Utilizamos el patrón MVC (Modelo-Vista-Controlador), por lo que aplicando dicho modelo y refinando la capa de presentación, obtendremos un primer refinamiento de los diferentes componentes: SystemAdministration, Profile, Food y FoodDiary, separando el componente que hará de controlador y los que harán de vistas.

Componente SystemAdministration

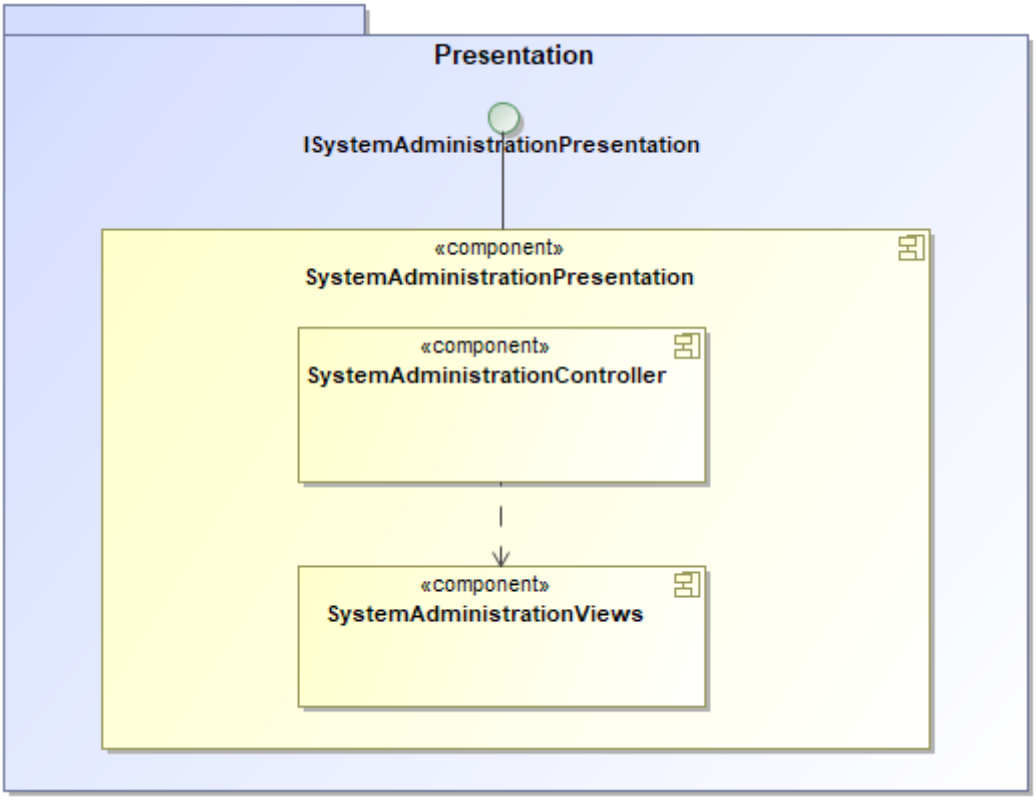


Ilustración 9. Patrón MVC. Presentación. Componente SystemAdministration

Componente Profile

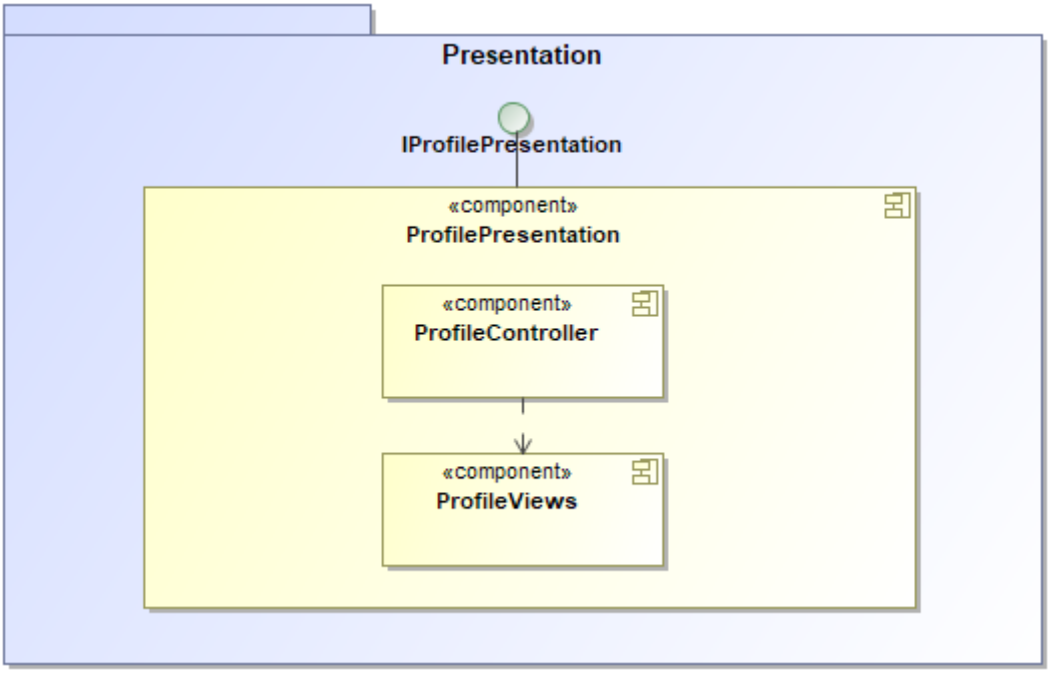


Ilustración 10. Patrón MVC. Presentación. Componente Profile

Componente Food

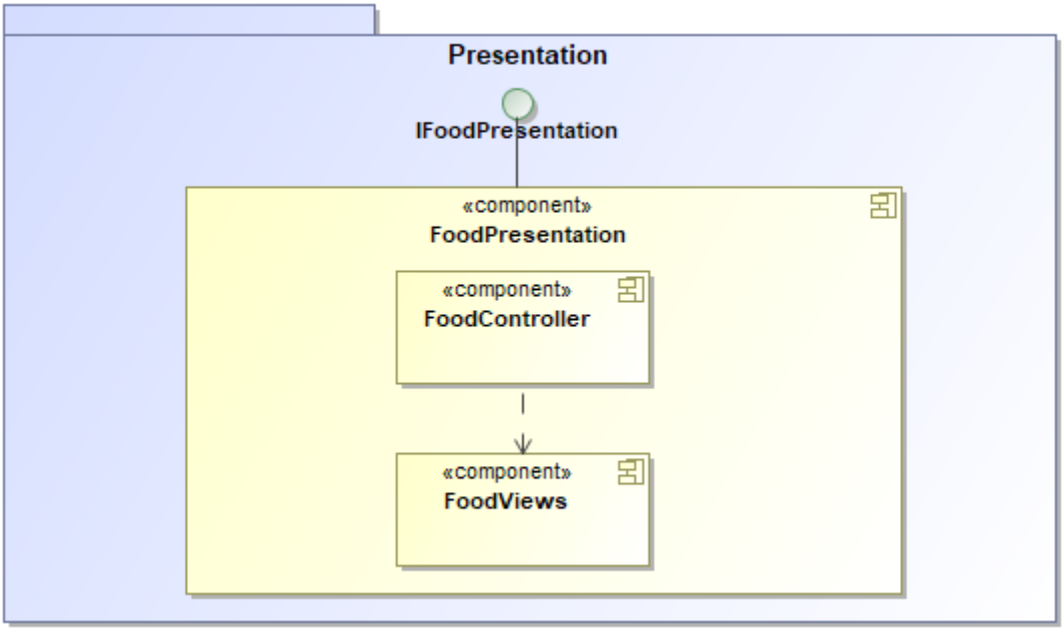


Ilustración 11. Patrón MVC. Presentación. Componente Food

Componente FoodDiary

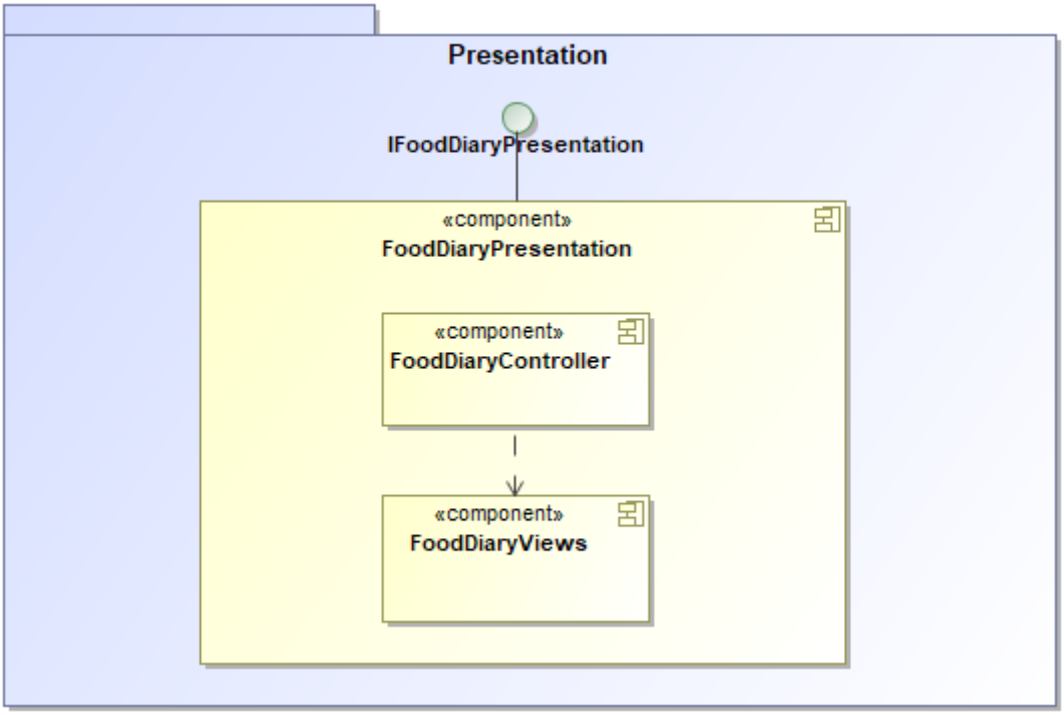


Ilustración 12. Patrón MVC. Presentación. Componente FoodDiary

A partir del refinamiento anterior, tomamos las siguientes decisiones de diseño:

- Habrá un único controlador para todas las operaciones de un componente (utilizamos el patrón *Front Controller*). Las operaciones no se implementarán dentro del controlador, sino que se utilizará un esquema basado en el patrón *Command*. El controlador será responsable de coordinar todo el proceso, pero la implementación estará desacoplada. Cada una de las diferentes acciones que el usuario puede realizar por medio de la interfaz de usuario se mapeará con una relación 1:1 con una vista y el controlador simplemente ejecutará estas acciones.

Por lo que, al aplicar los patrones citados, tendremos un nuevo diagrama con el refinamiento aplicado:

Componente SystemAdministration

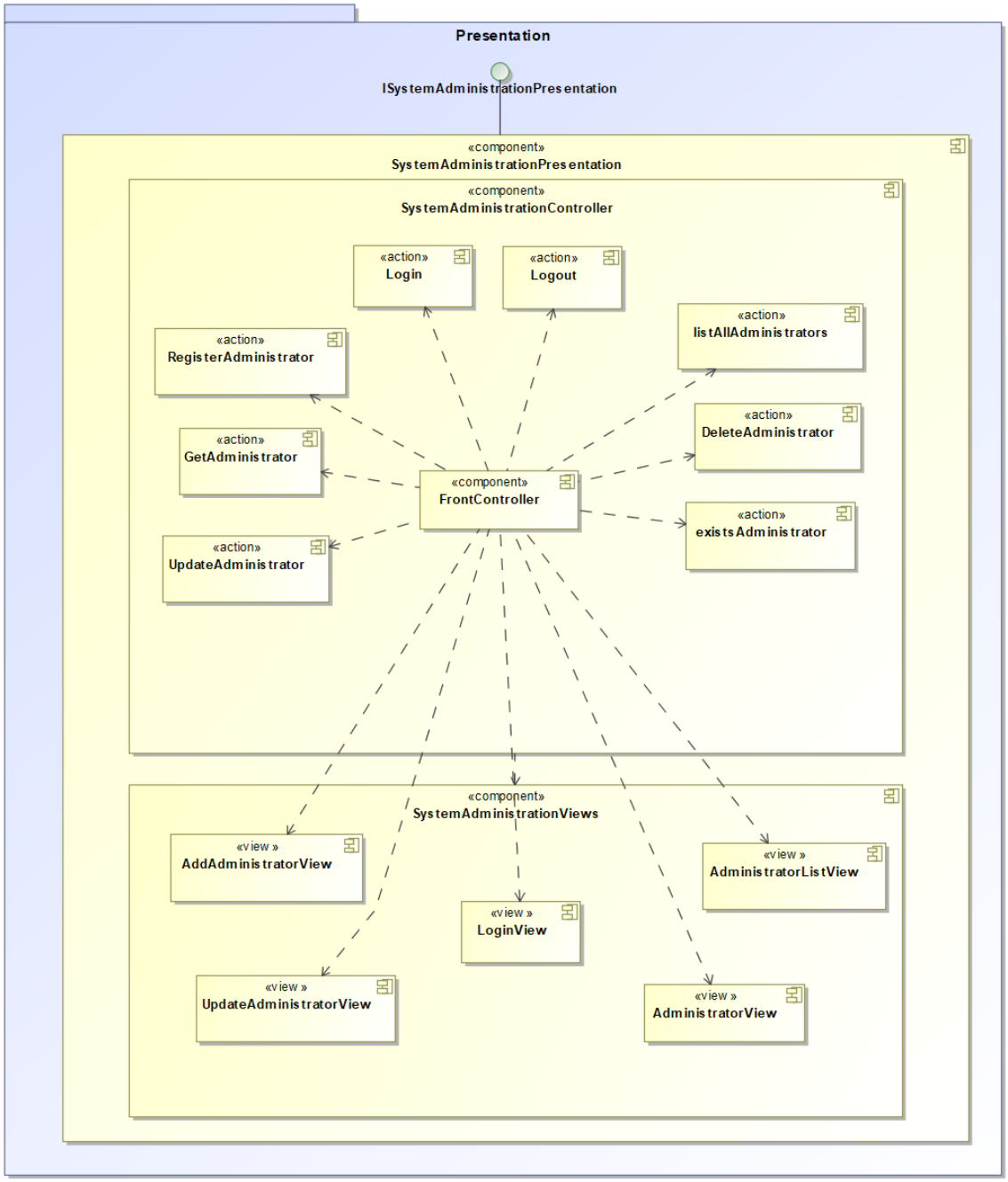


Ilustración 13. Refinamiento MVC. Presentación. Componente SystemAdministration

Componente Profile

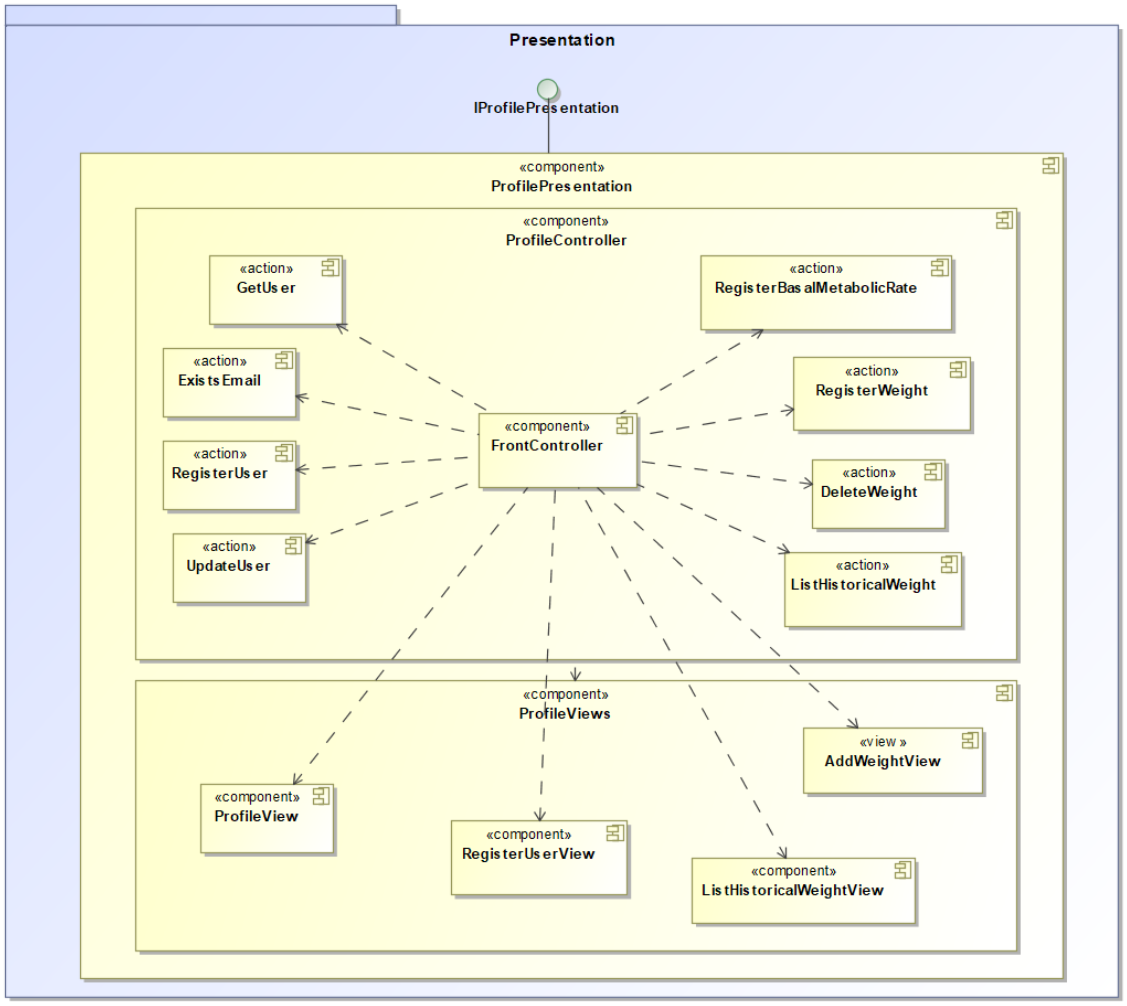


Ilustración 14. Refinamiento MVC. Presentación. Componente Profile

Componente Food

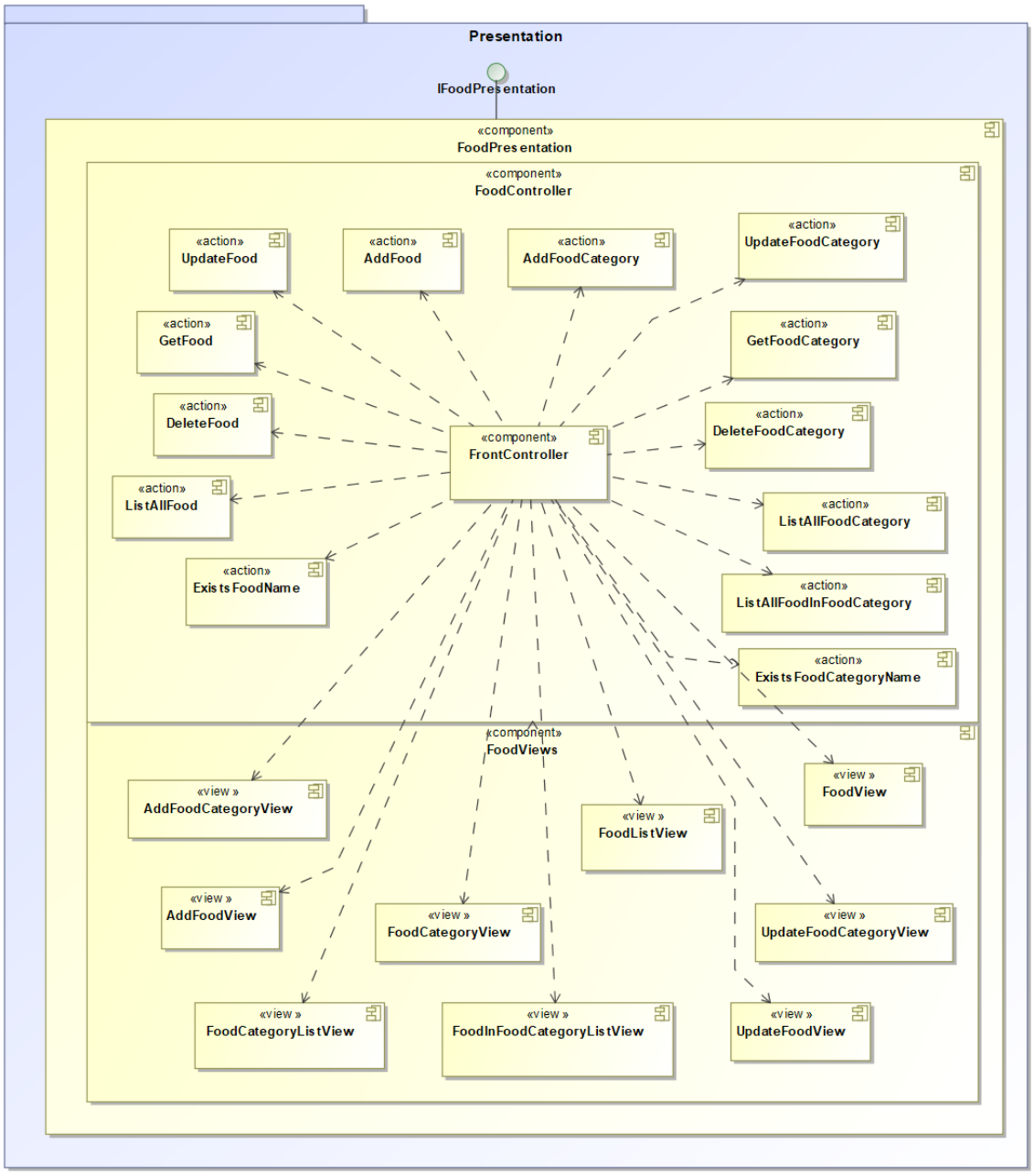


Ilustración 15. Refinamiento MVC. Presentación. Componente Food

Componente FoodDiary

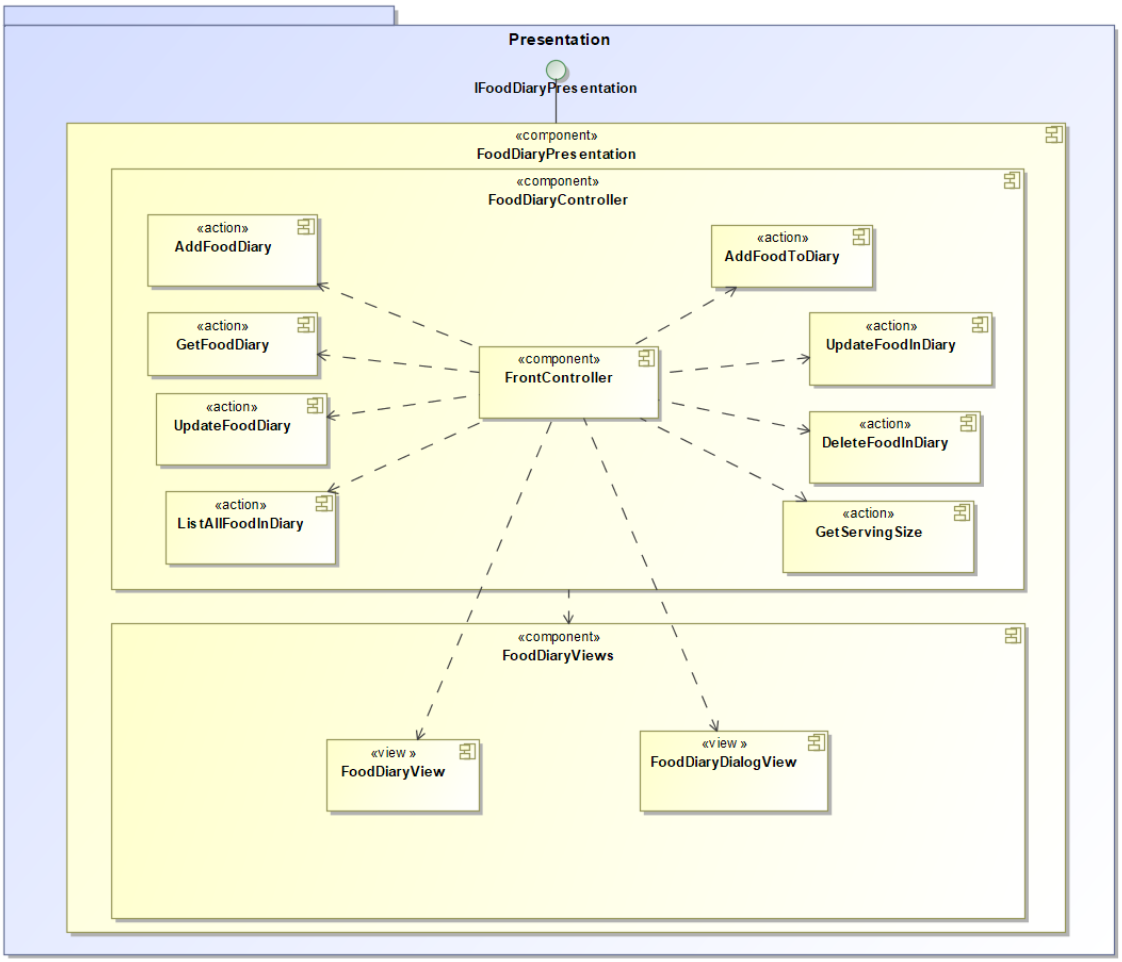


Ilustración 16. Refinamiento MVC. Presentación. Componente FoodDiary

Por lo que la capa de presentación ya la tenemos suficientemente refinada. Ya solo nos queda aplicar el perfil de Java EE a dicha capa.

3.3.2.2. Capa de negocio

Para la capa de negocio no hará falta realizar tantos refinamientos como para la capa anterior. Por lo que obtenemos los siguientes diagramas para los distintos componentes:

Componente SystemAdministration

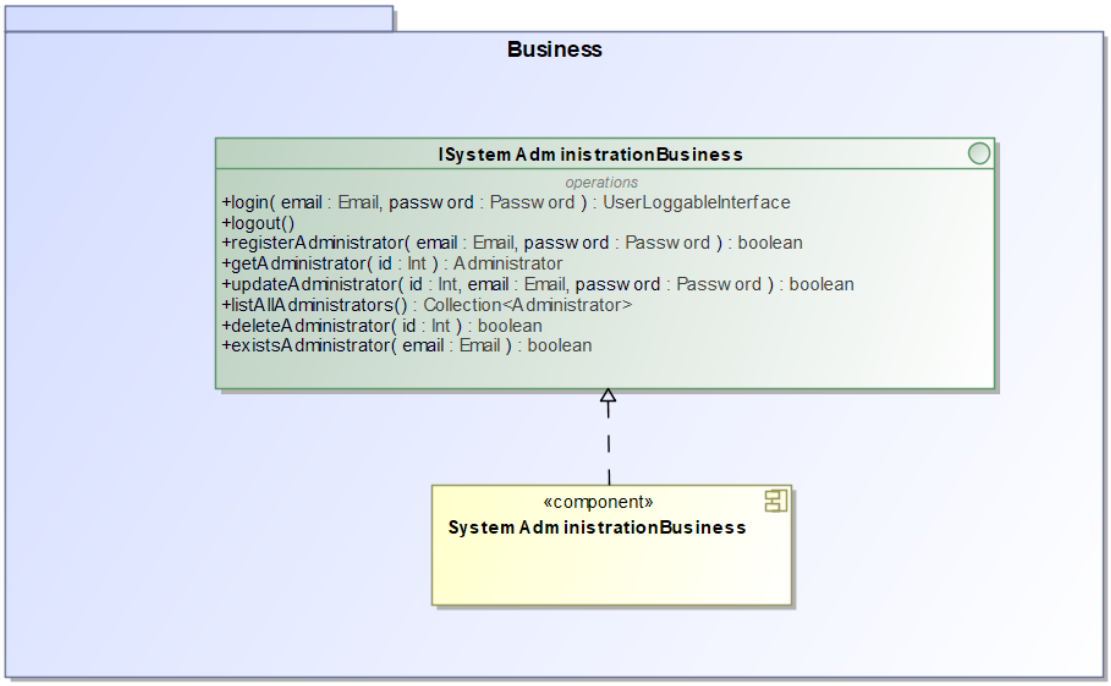


Ilustración 17. Patrón MVC. Negocio. Componente SystemAdministration

Componente Profile

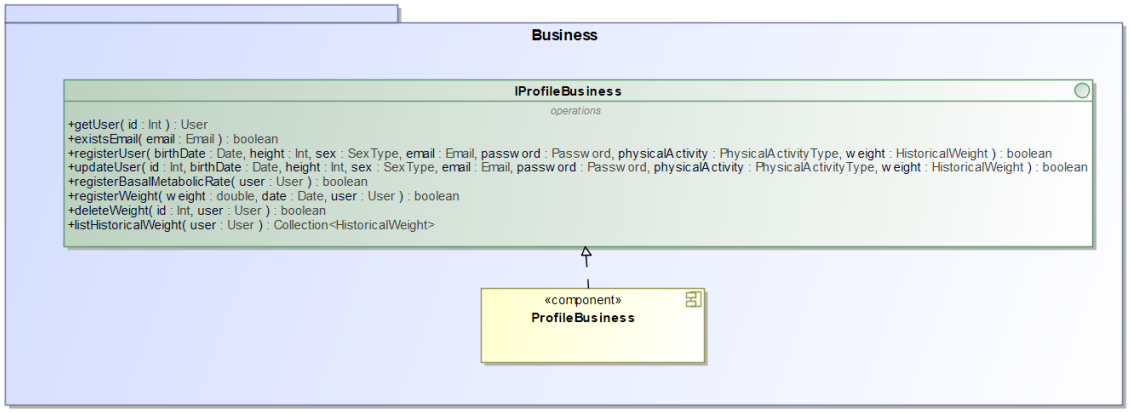


Ilustración 18. Patrón MVC. Negocio. Componente Profile

Componente Food

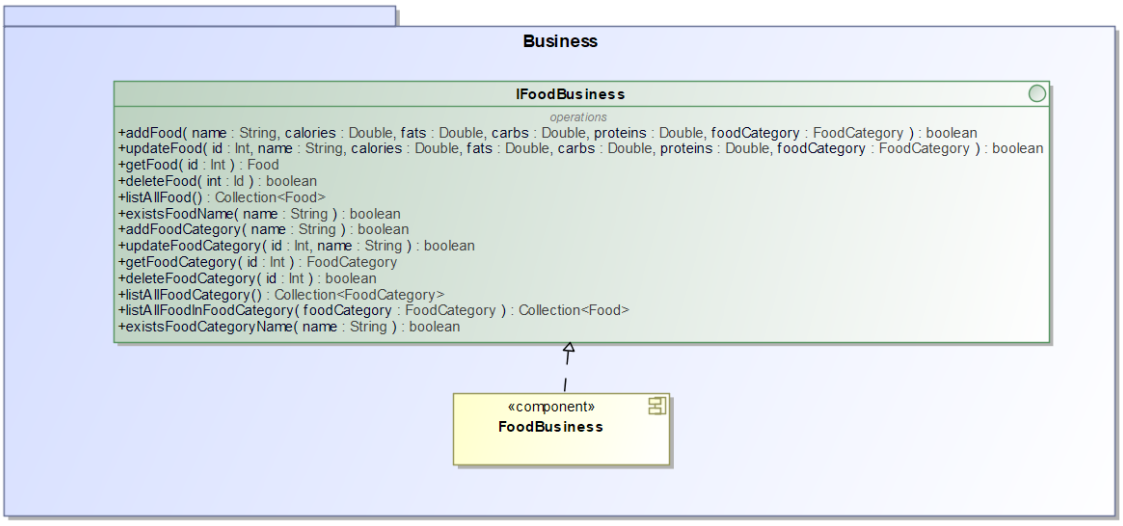


Ilustración 19. Patrón MVC. Negocio. Componente Food

Componente FoodDiary

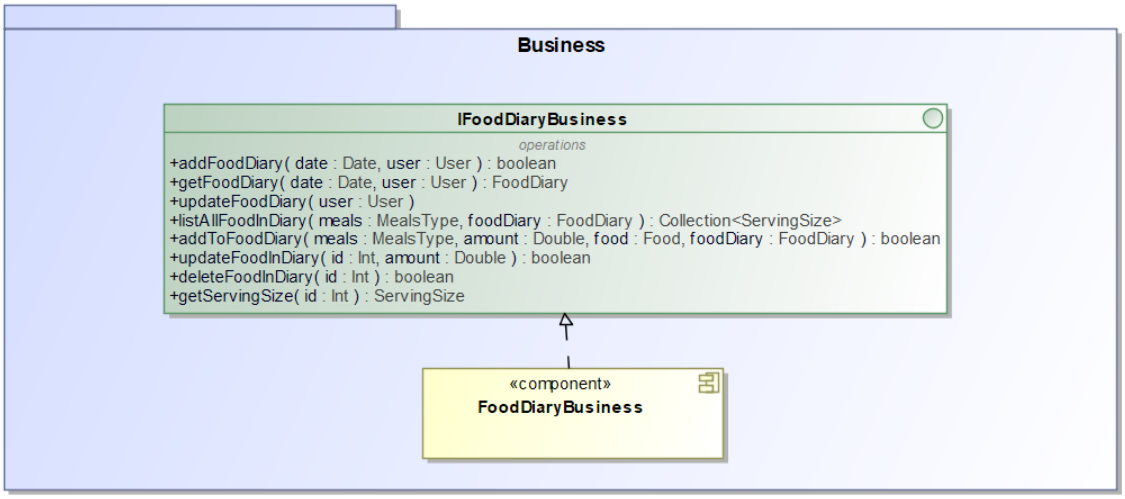


Ilustración 20. Patrón MVC. Negocio. Componente FoodDiary

3.3.2.3. Capa de integración (persistencia)

Para la capa de integración, al igual que para la capa de negocio, el refinamiento es más simple. Por lo tanto, nos quedan los siguientes diagramas para los distintos componentes:

Componente SystemAdministration

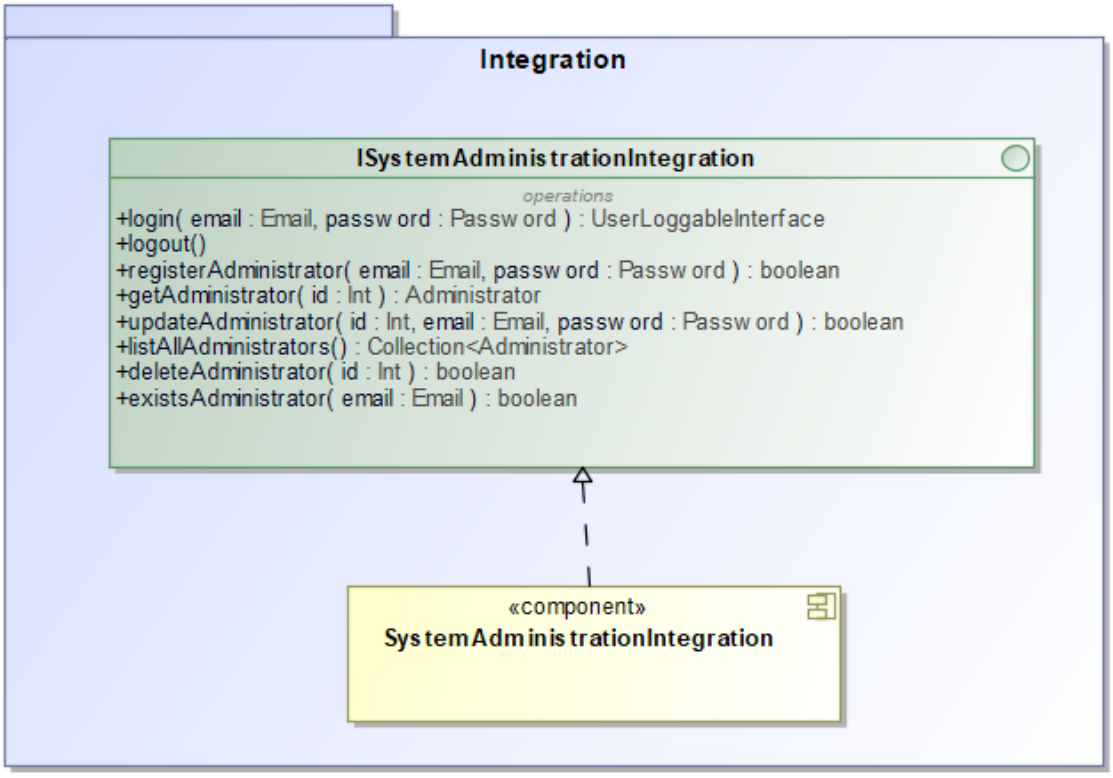


Ilustración 21. Patrón MVC. Integración. Componente SystemAdministration

Componente Profile

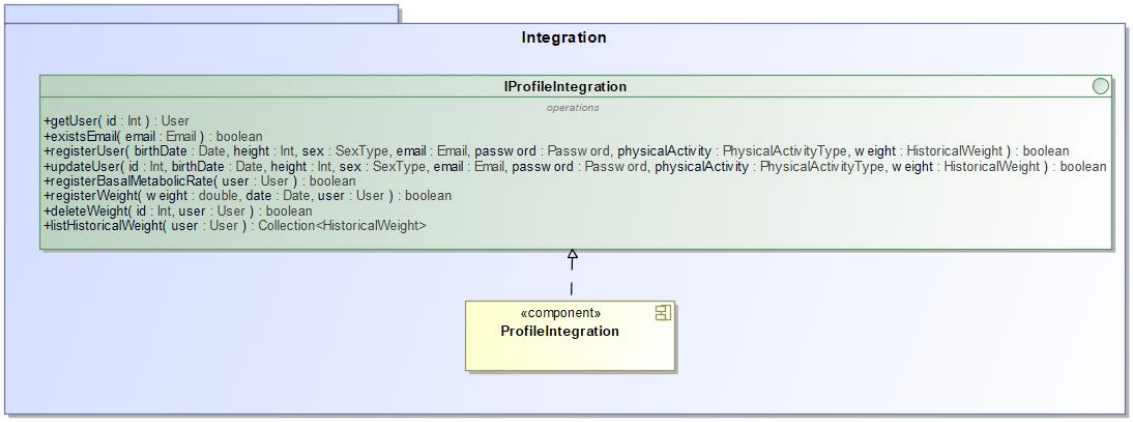


Ilustración 22. Patrón MVC. Integración. Componente Profile

Componente Food

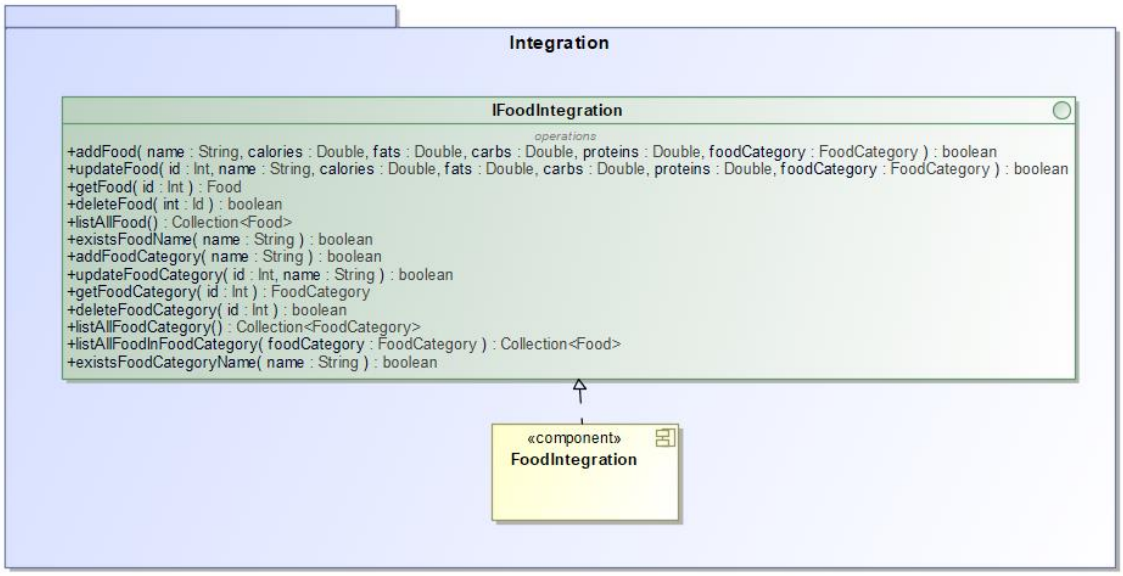


Ilustración 23. Patrón MVC. Integración. Componente Food

Componente FoodDiary

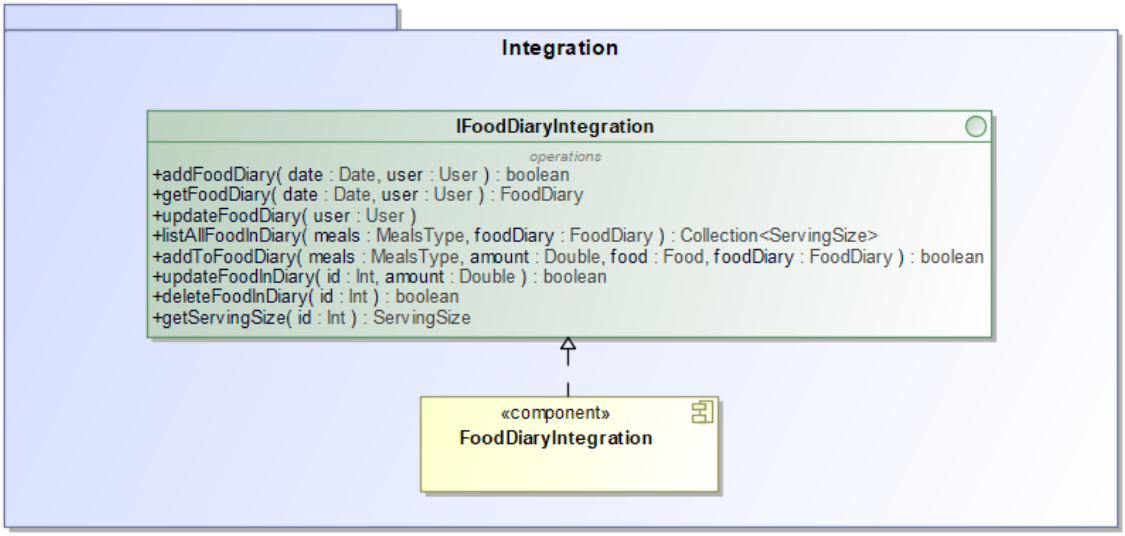


Ilustración 24. Patrón MVC. Integración. Componente FoodDiary

3.3.3. Aplicación del perfil Java Enterprise Edition a los diagramas de componentes

Ahora aplicaremos el perfil Java EE a nuestro diseño. Utilizaremos las siguientes decisiones de diseño:

- Para la capa de presentación utilizaremos el *framework Java Server Faces* (JSF).
- El controlador no será necesario implementarlo, ya que será el *servlet Faces Servlet* que ya tenemos disponible al utilizar JSF.
- Las acciones que se corresponden a los *Commands* estarán definidas mediante *Managed Beans*.
- Las vistas se implementarán con *Facelets*.

3.3.3.1. Capa de presentación

Componente SystemAdministration

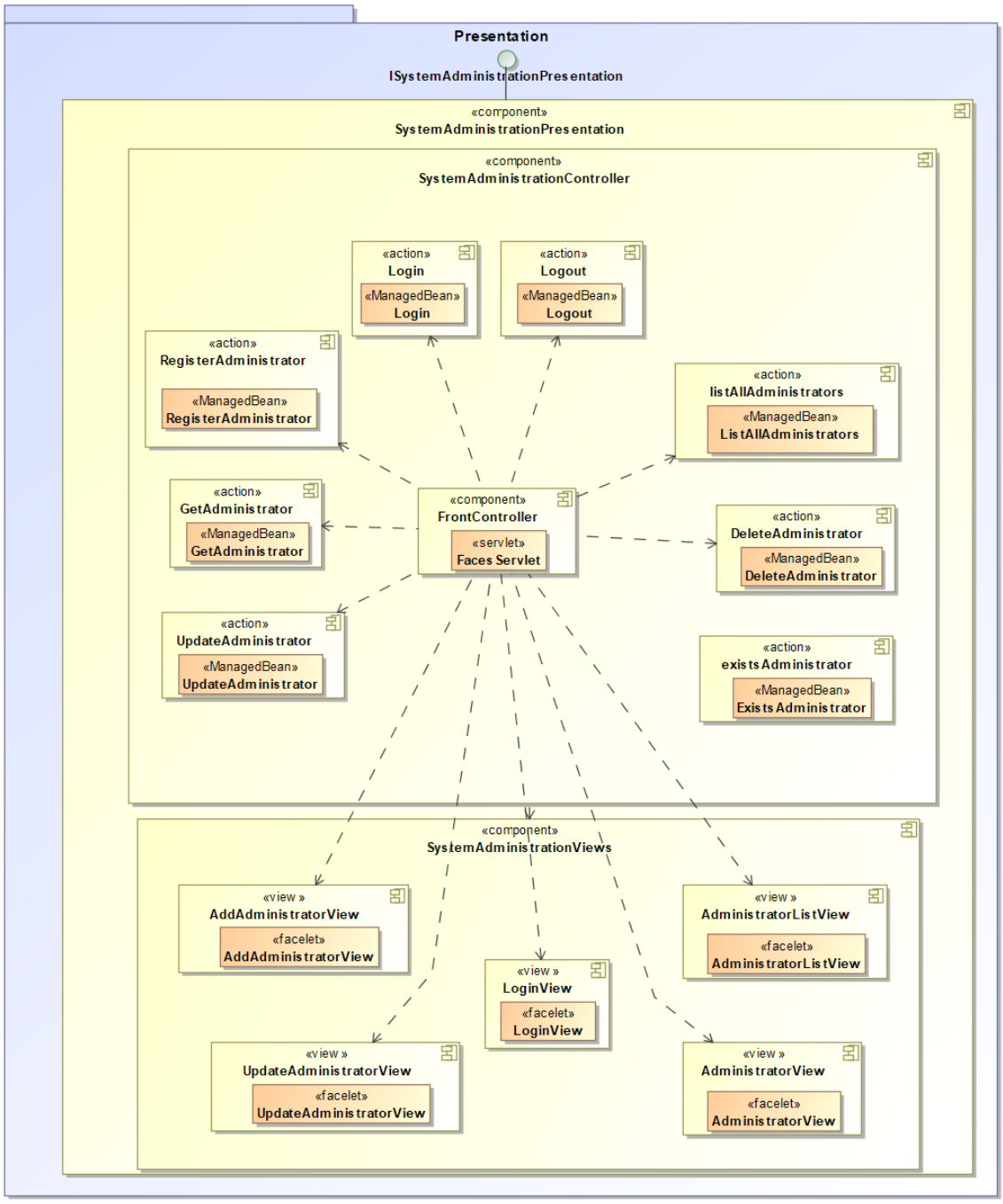


Ilustración 25. Perfil Java EE. Presentación. Componente SystemAdministration

Componente Profile

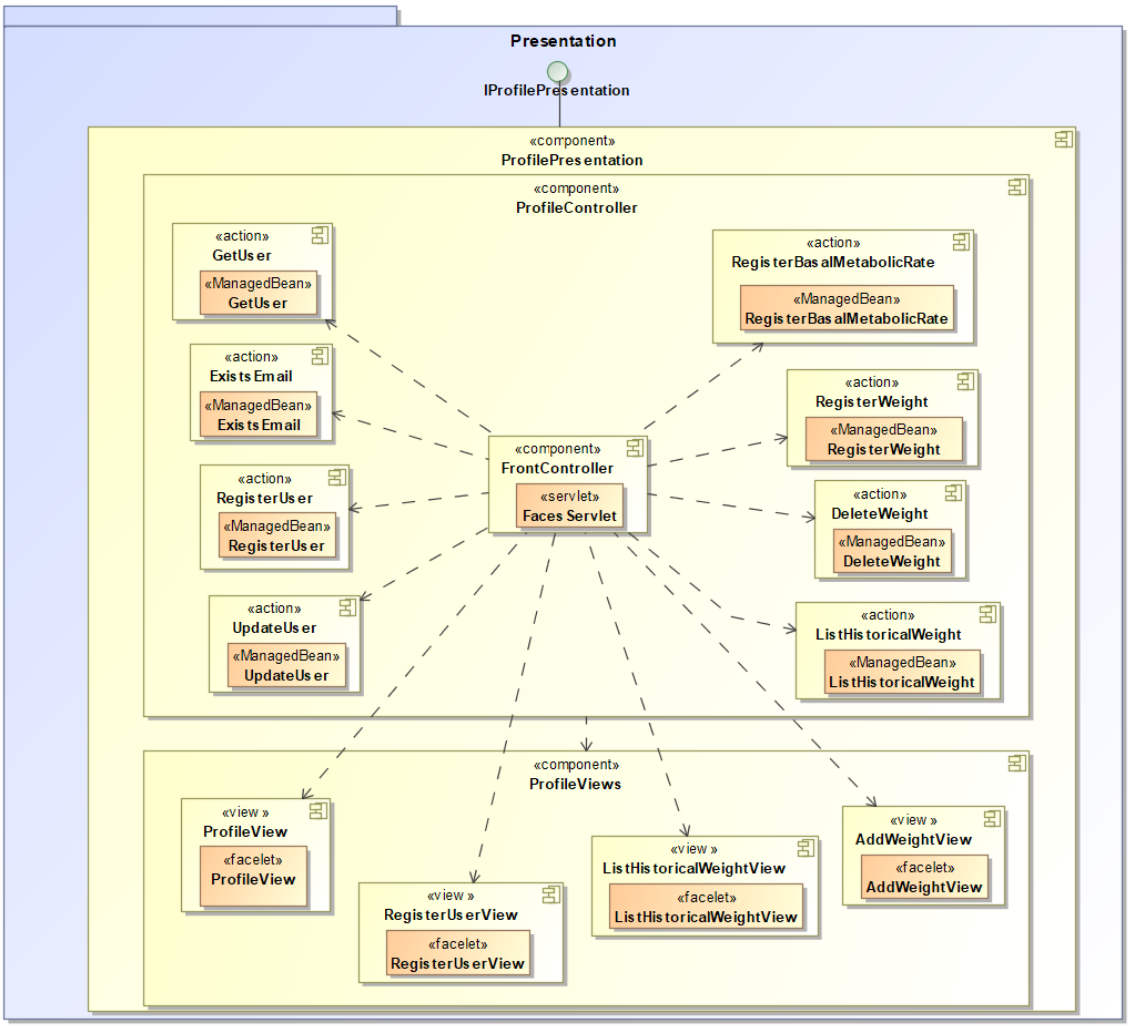


Ilustración 26. Perfil Java EE. Presentación. Componente Profile

Componente Food

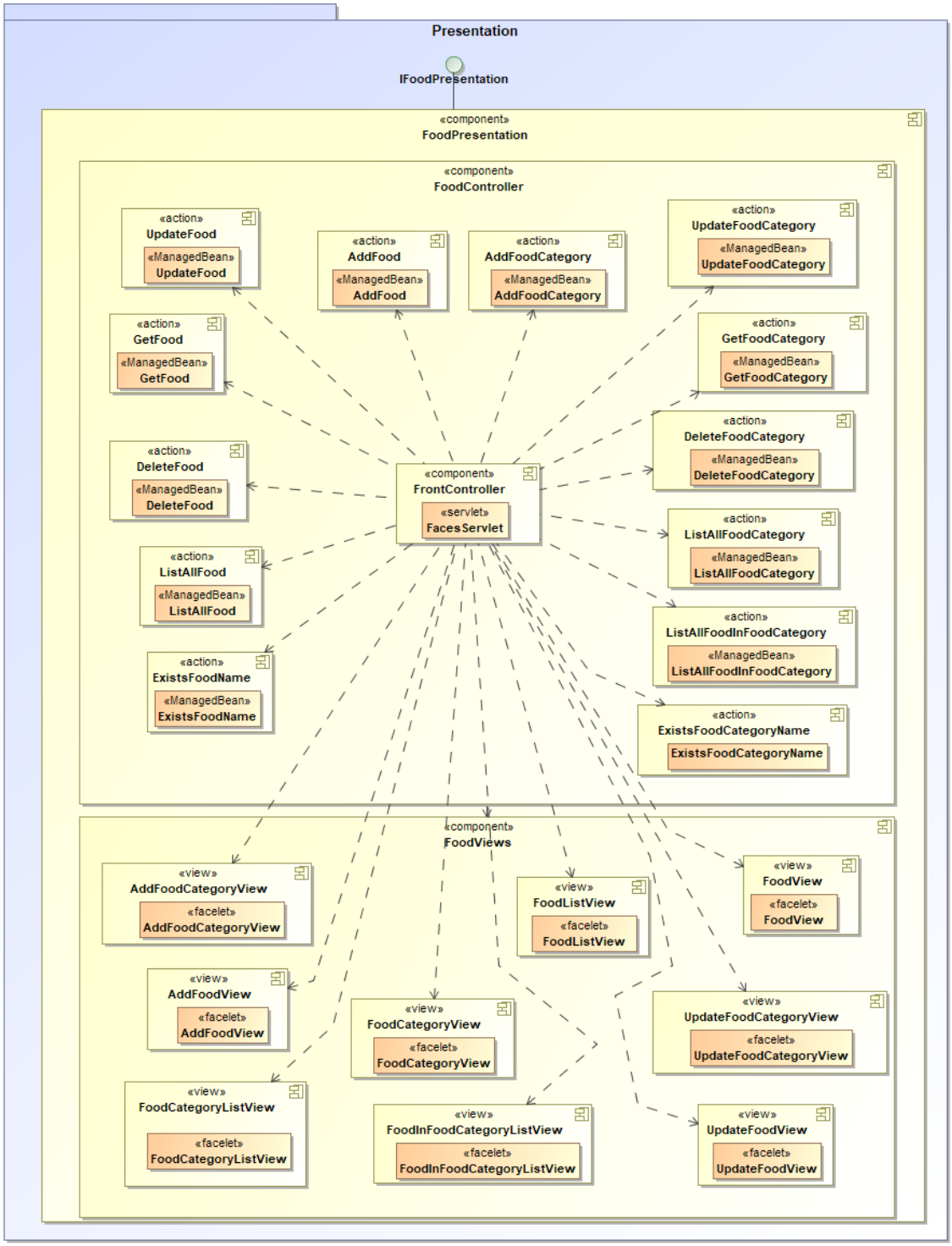


Ilustración 27. Perfil Java EE. Presentación. Componente Food

Componente FoodDiary

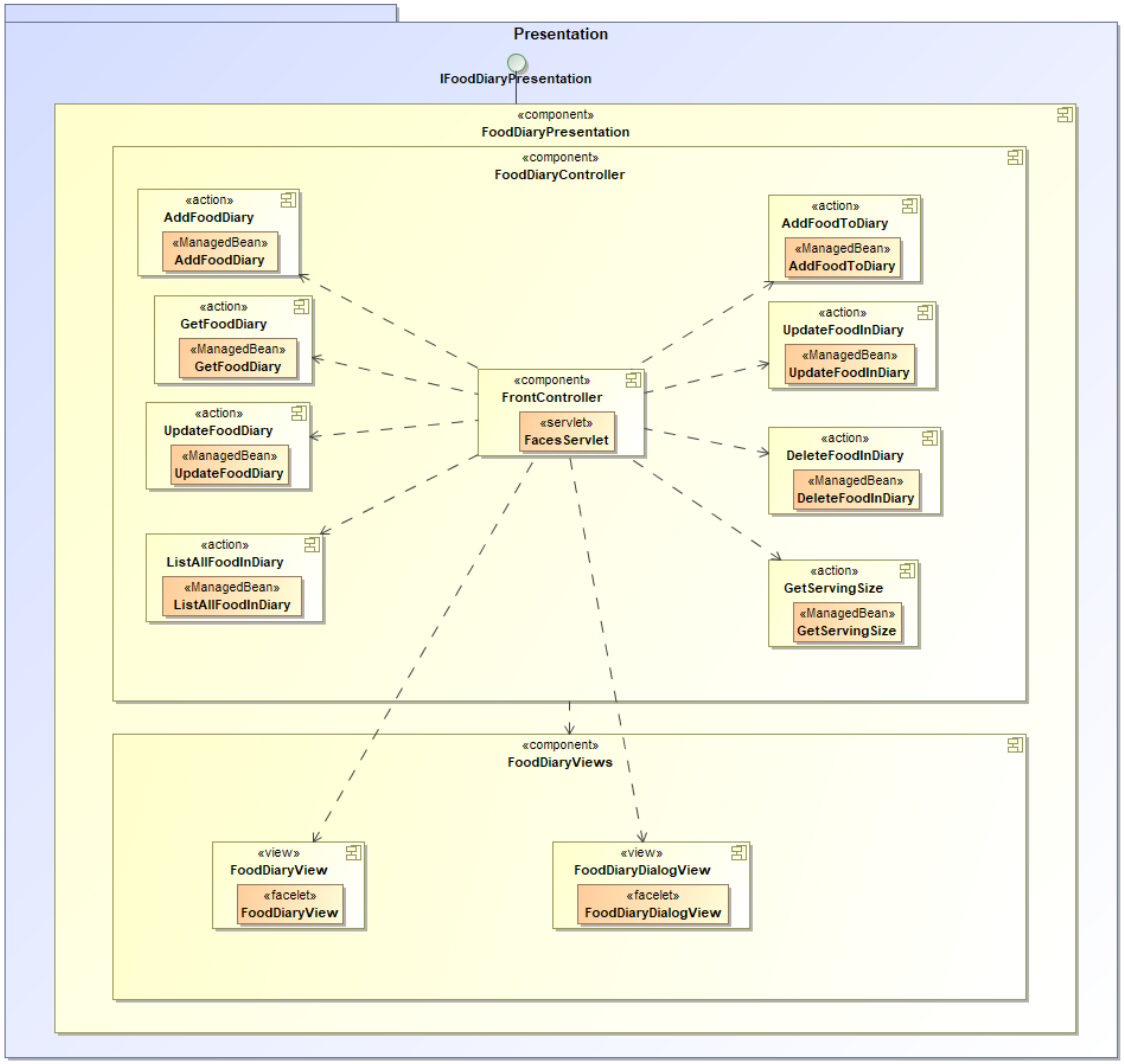


Ilustración 28. Perfil Java EE. Presentación. Componente FoodDiary

3.3.3.2. Capa de negocio

Para implementar la capa de negocio utilizaremos el patrón Fachada. Daremos acceso tanto remoto como local a los componentes de esta capa.

Componente SystemAdministration

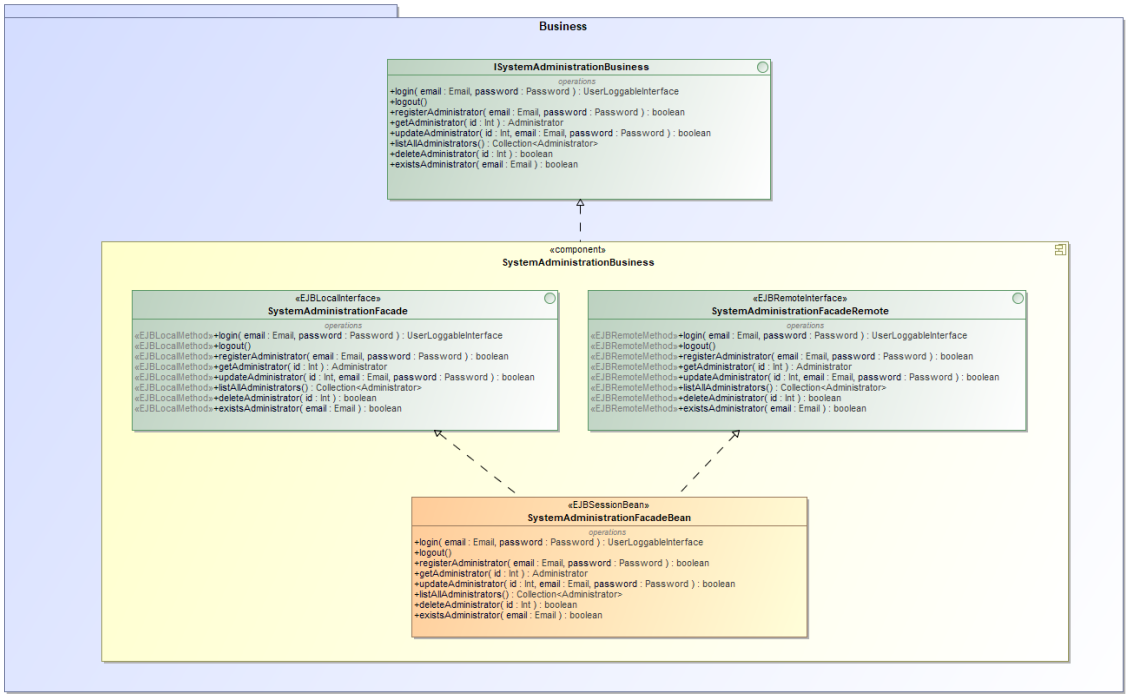


Ilustración 29. Perfil Java EE. Negocio. Componente SystemAdministration

Componente Profile

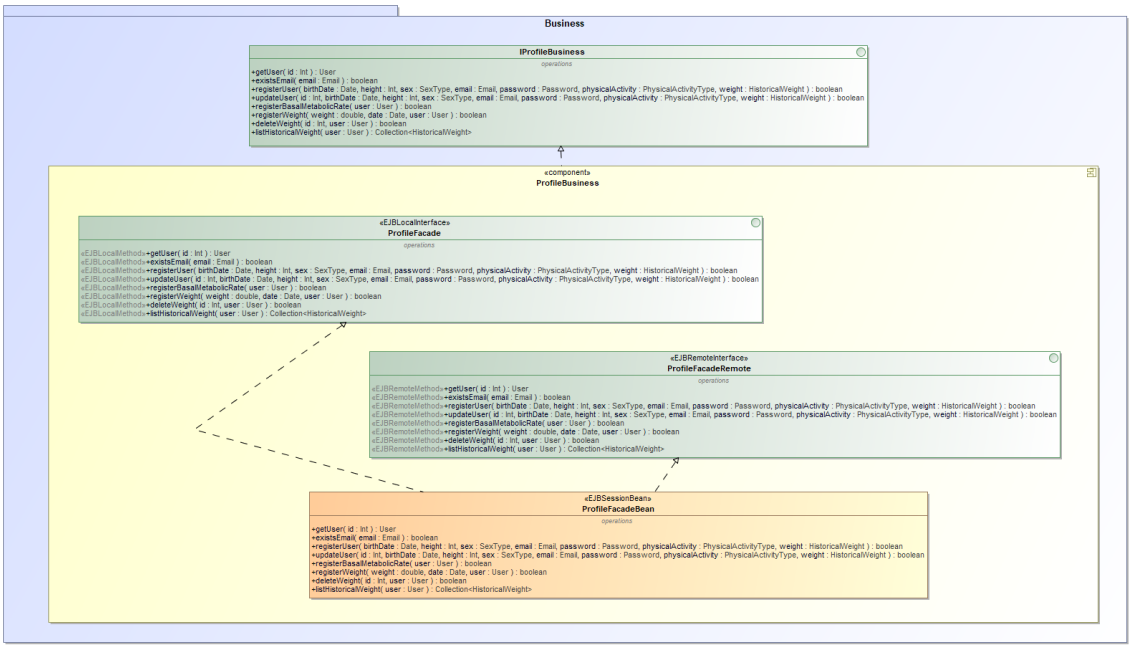


Ilustración 30. Perfil Java EE. Negocio. Componente Profile

Componente Food

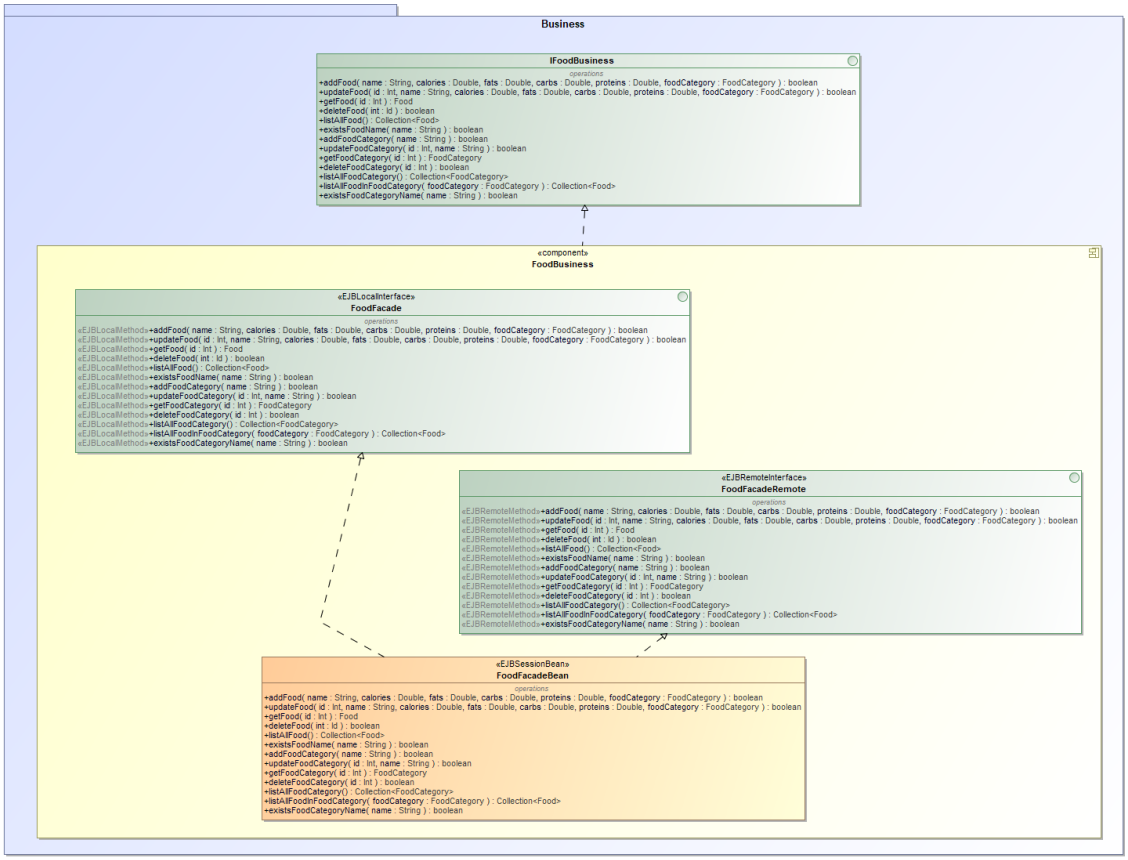


Ilustración 31. Perfil Java EE. Negocio. Componente Food

Componente FoodDiary

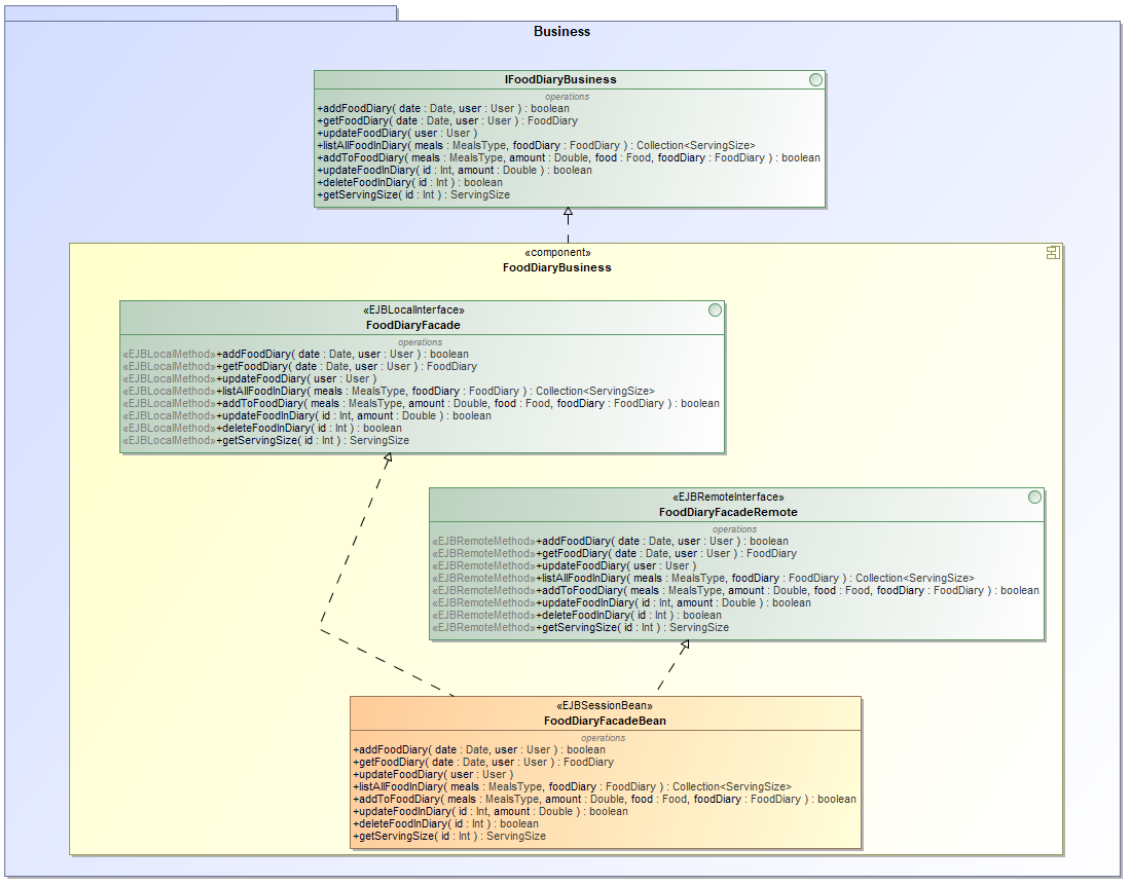


Ilustración 32. Perfil Java EE. Negocio. Componente FoodDiary

3.3.3.3. Capa de integración (persistencia)

Los datos se almacenarán en una base de datos relacional y el acceso a la capa de integración desde la capa de negocio será local. Implementaremos los componentes con JPA. Así, aplicando Java EE a los diagramas que hemos hecho anteriormente, obtenemos los siguientes diagramas de componentes:

Componente SystemAdministration

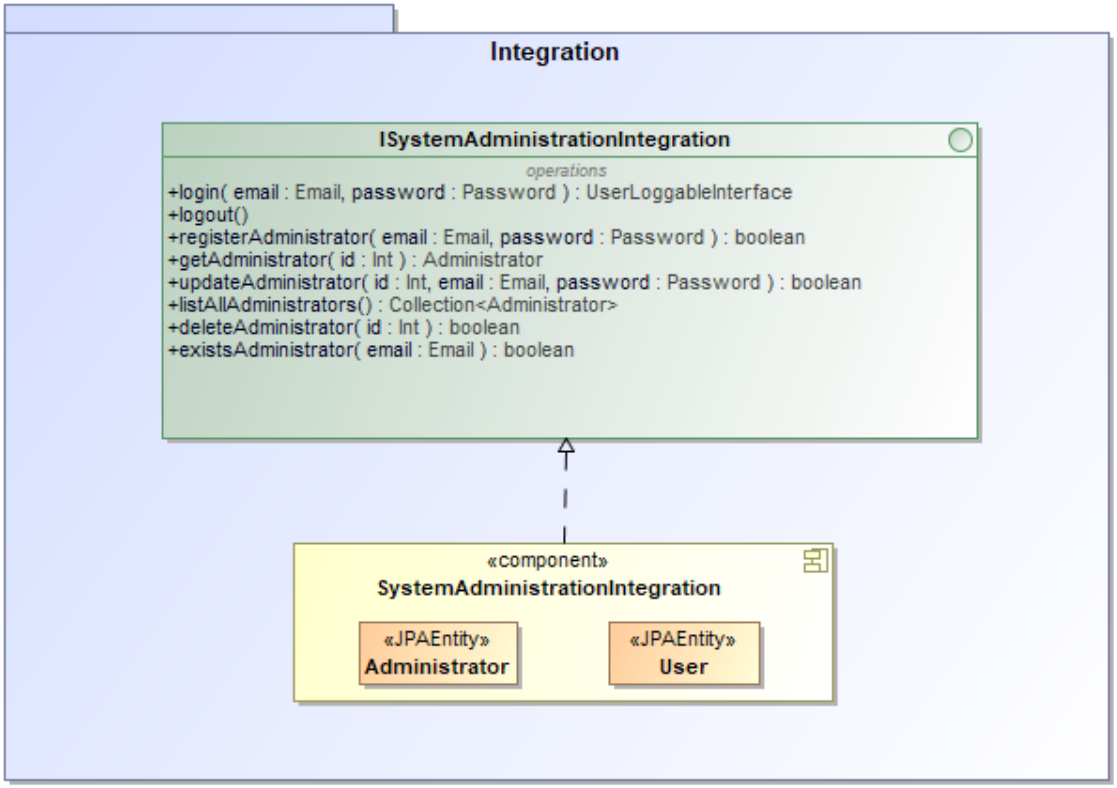


Ilustración 33. Perfil Java EE. Integración. Componente SystemAdministration

Componente Profile

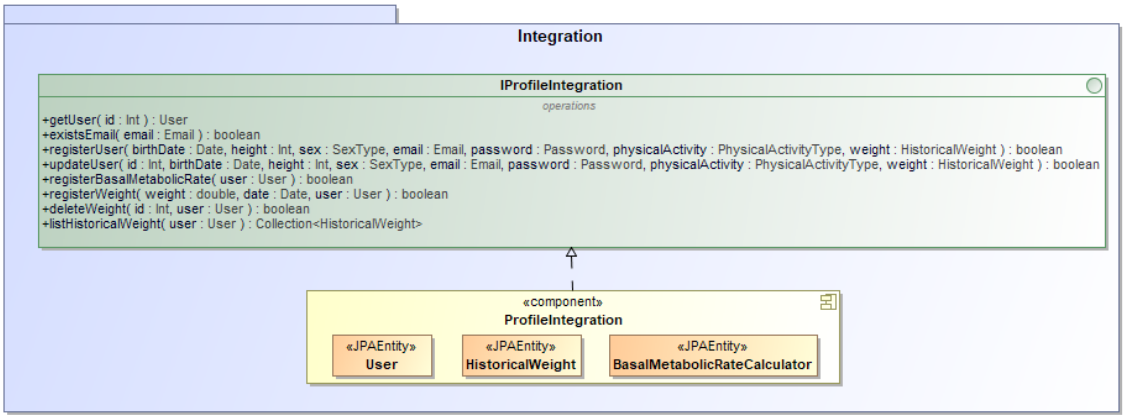


Ilustración 34. Perfil Java EE. Integración. Componente Profile

Componente Food

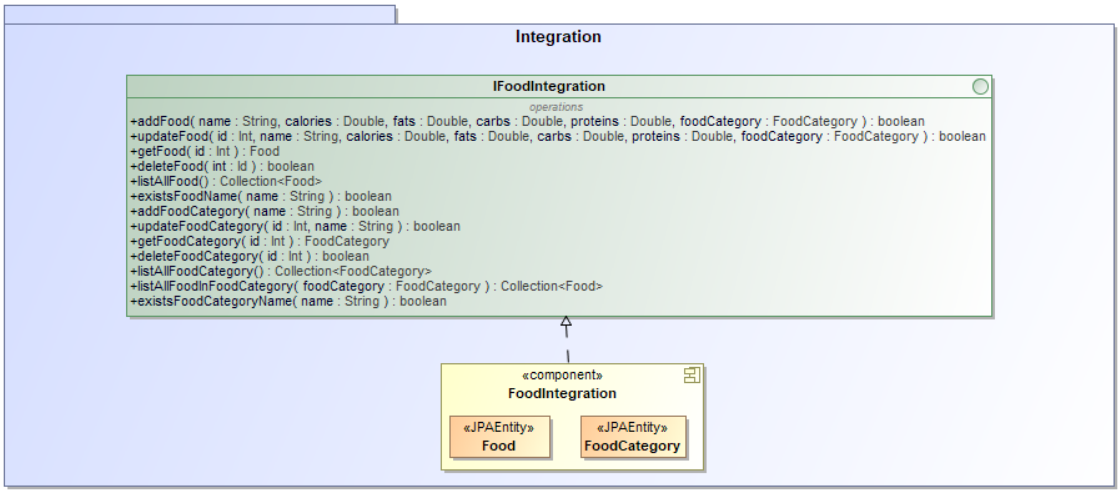


Ilustración 35. Perfil Java EE. Integración. Componente Food

Componente FoodDiary

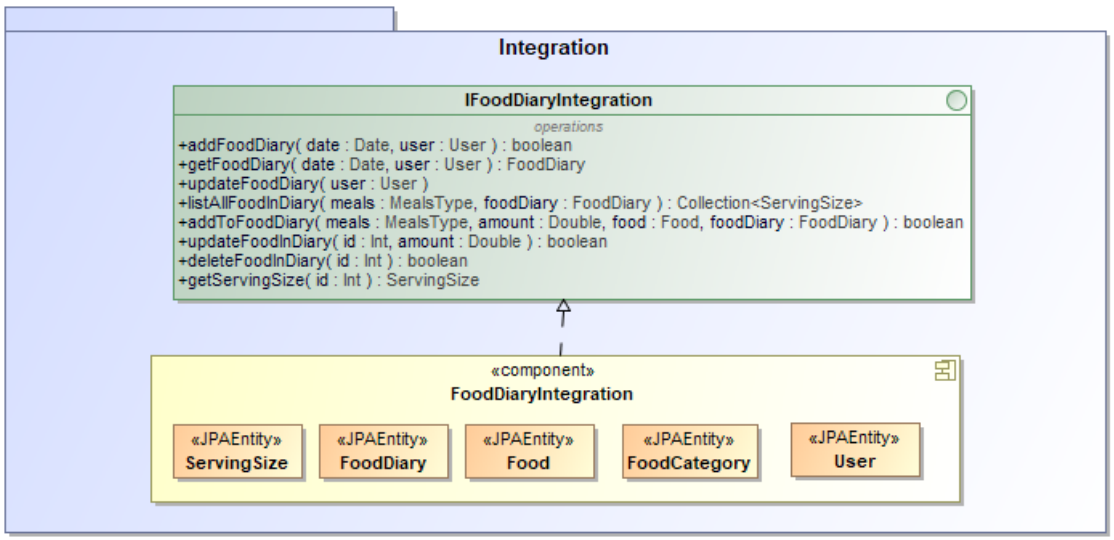


Ilustración 36. Perfil Java EE. Integración. Componente FoodDiary

4. Implementación

Este capítulo contiene la información y detalles que conforman la fase de implementación del proyecto realizado para el presente TFG.

4.1. Herramientas de *software* empleadas

StayFit es una aplicación desarrollada en Java EE, por lo que he utilizado para su construcción las siguientes herramientas:

- **IDE:** Eclipse Java EE IDE for Web Developers, versión 2018-09 (4.9.0). Se ha empleado la versión Java 8.
- **Base de datos:** se ha utilizado PostgreSQL, versión 10.10.1. Para gestionar la base de datos, se ha utilizado la aplicación PGAdmin 4, en su versión 4.5.
- **Servidor de aplicaciones:** se ha empleado Wildfly 14.0.1 (anteriormente JBoss) como servidor para desplegar la aplicación y poder poner a prueba la aplicación web desarrollada.

En los anexos al final del presente documento se encuentra un tutorial de instalación de los componentes necesarios, así como de una breve explicación para el despliegue y puesta en funcionamiento del proyecto para su testeo.

4.2. Análisis de la capa de presentación

Para implementar la capa de presentación, es decir, las vistas de nuestra aplicación, se han utilizado 3 *frameworks:*

- **JavaServer Faces (JSF):** el *framework* recomendado por Java EE para desarrollar la capa de presentación. Implementa un patrón MVC, tal como he comentado anteriorment, en el que el Controlador lo proporciona dicho *framework* y las vistas y acciones del modelo se implementan a través de páginas .xhtml.
- **Primefaces:** este *framework* está desarrollado sobre el estándar JSF y facilita la creación de las vistas y el desarrollo de determinadas acciones.
- **Bootstrap:** este conocido *framework* también se ha utilizado para la creación de las vistas y poder darle un aspecto más bonito y presentable a las distintes pantalles de la aplicación.

4.3. Análisis de la capa de negocio

La capa de negocio se ha implementado mediante el uso de los conocidos Enterprise Java Beans (EJB), que son el estándar recomendado por Java EE. Además, de ser una tecnologia estudiada en diverses assignatures del itinerario de desarrollo de *software*.

Cada componente en los que se ha dividido el proyecto, está formado por una interfaz local, una interfaz remota y un EJB de sesión sin estado.

Las interfaces contienen la firma de los métodos que exponen los componentes y que son invocados por otros componentes. Sin embargo, es en el EJB donde se implementan los diferentes métodos y donde se desarrolla la lógica de negocio.

4.4. Análisis de la capa de integración

Dicha capa se ha implementado utilizando el *framework* de persistencia de datos Java Persistence Api (JPA) junto con la herramienta de mapeo objeto-relacional para Java, Hibernate.

4.5. Pantallas

4.5.1. Pantalla inicial de la aplicación e inicio de sesión

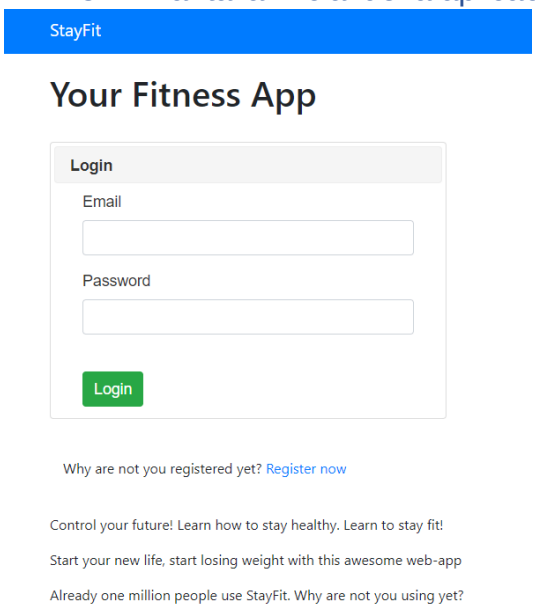


Ilustración 37. Pantalla inicio

4.5.2. Pantalla de registro

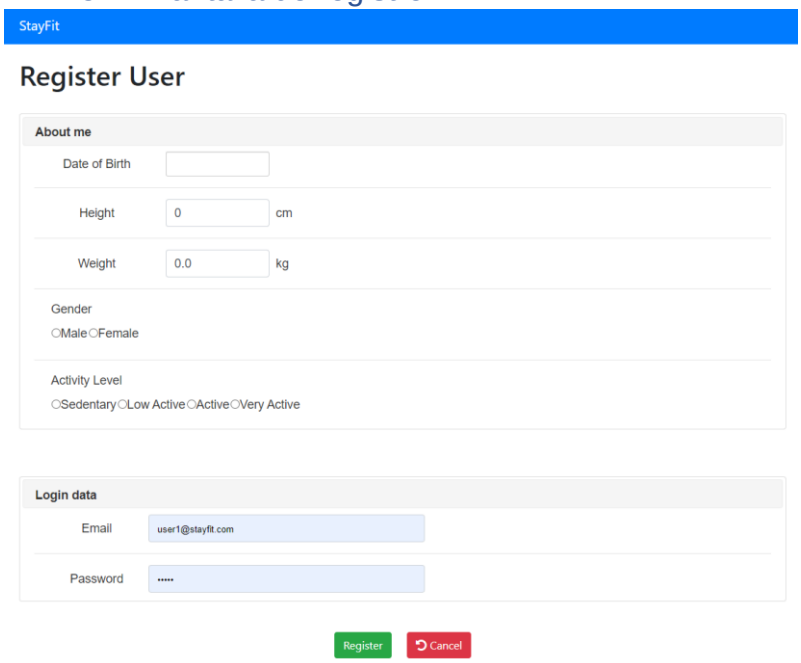


Ilustración 38. Pantalla registro

4.5.3. Pantalla inicio Usuario. Diario de comidas

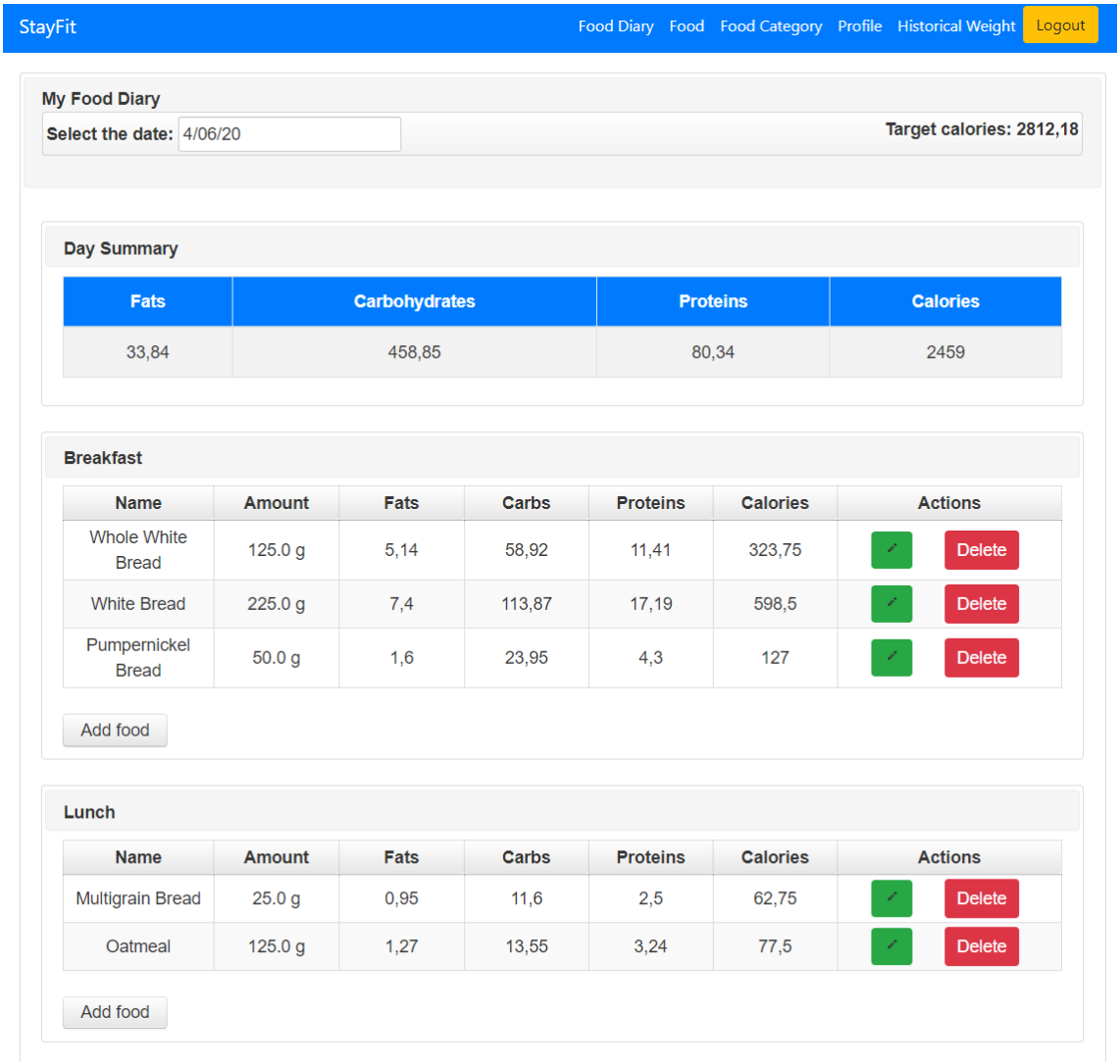


Ilustración 39. Pantalla diario de comidas

4.5.4. Pantalla de añadir alimento al diario

Ilustración 40. Pantalla añadir alimento a diario

4.5.5. Pantalla perfil de Usuario

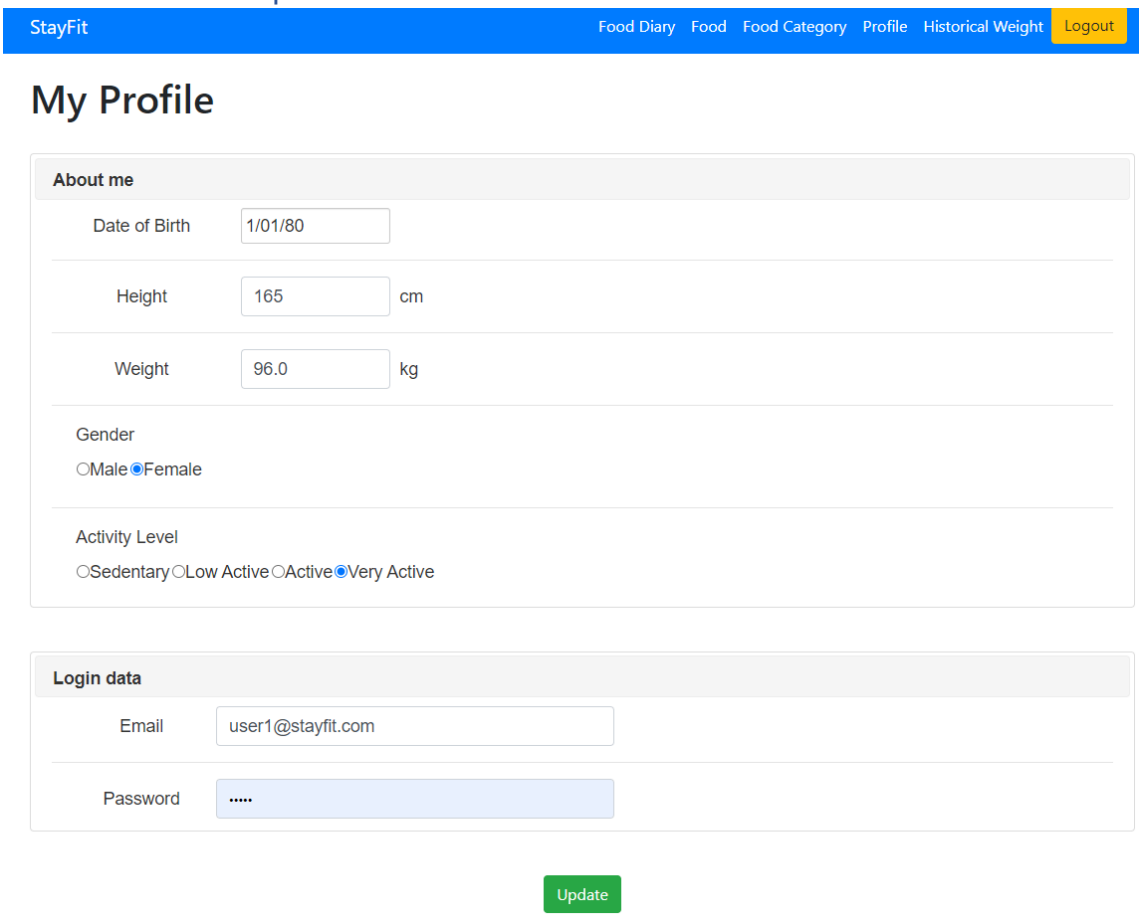


Ilustración 41. Pantalla perfil de usuario

4.5.6. Pantalla de histórico de peso

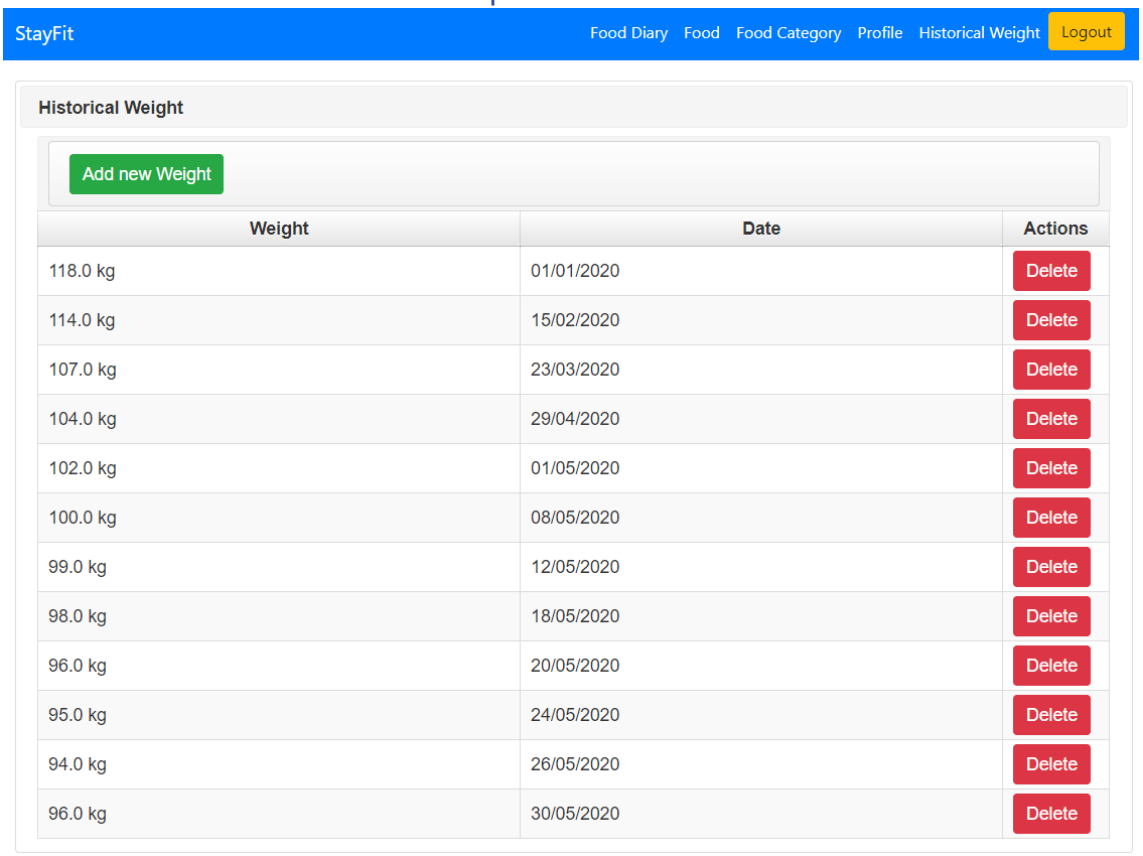


Ilustración 42. Pantalla histórico de peso

4.5.7. Pantalla de alimentos, vista administrador

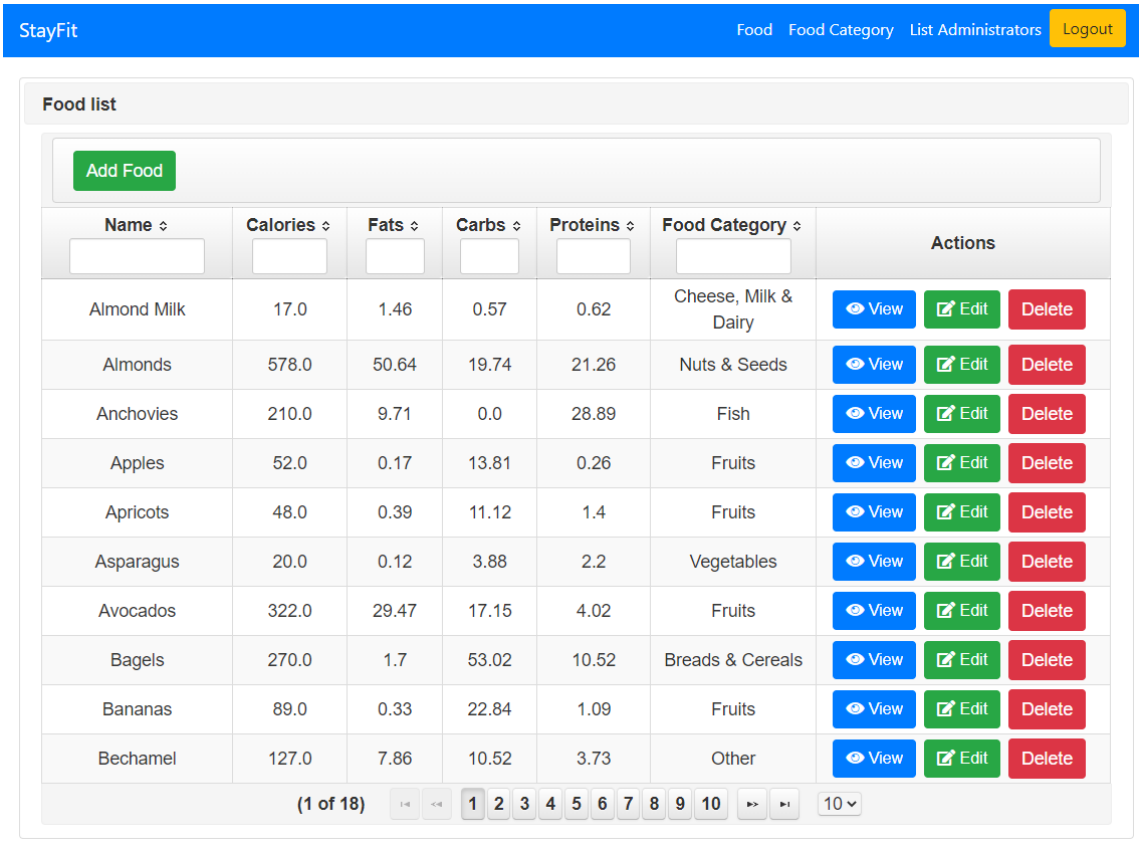


Ilustración 43. Pantalla de alimentos, vista administrador

4.5.8. Pantalla de detalle de alimento

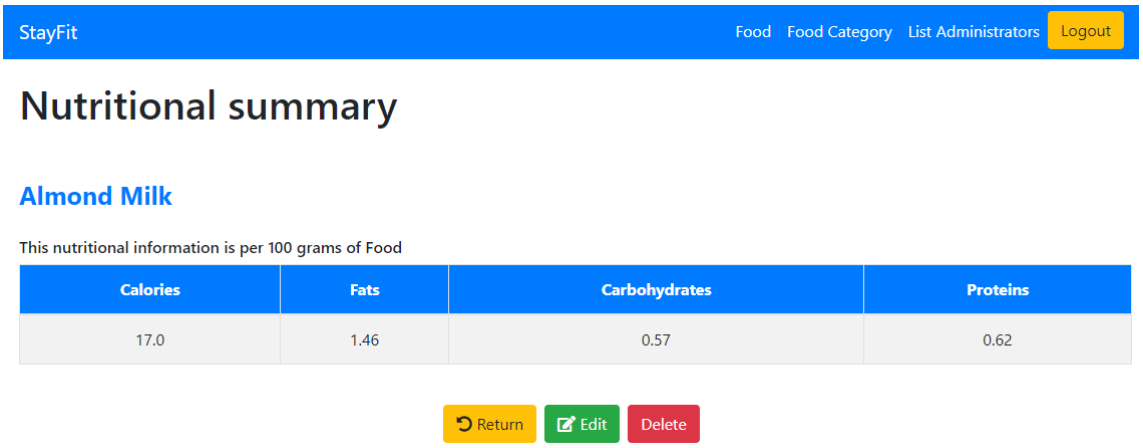


Ilustración 44. Pantalla de detalle de alimento

4.5.9. Pantalla de categorías de alimentos, vista administrador

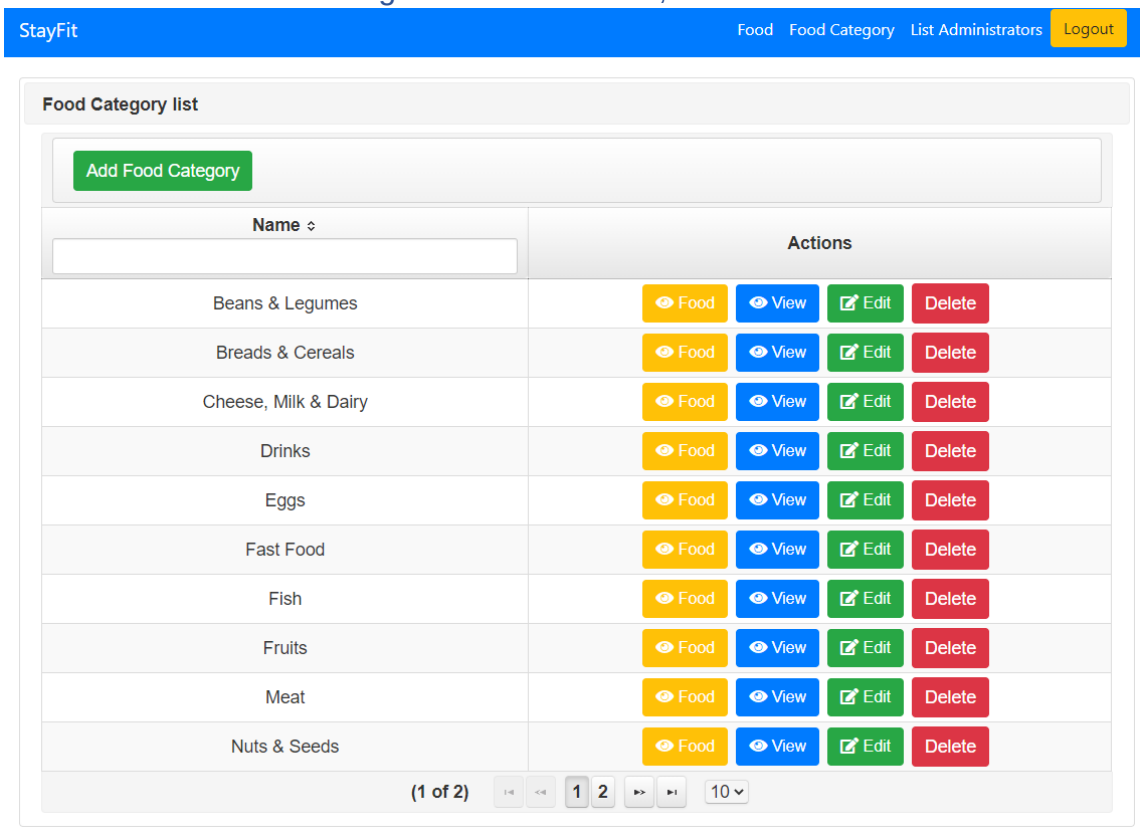


Ilustración 45. Pantalla de categorías de alimentos, vista administrador

4.5.10. Pantalla de gestión de administradores

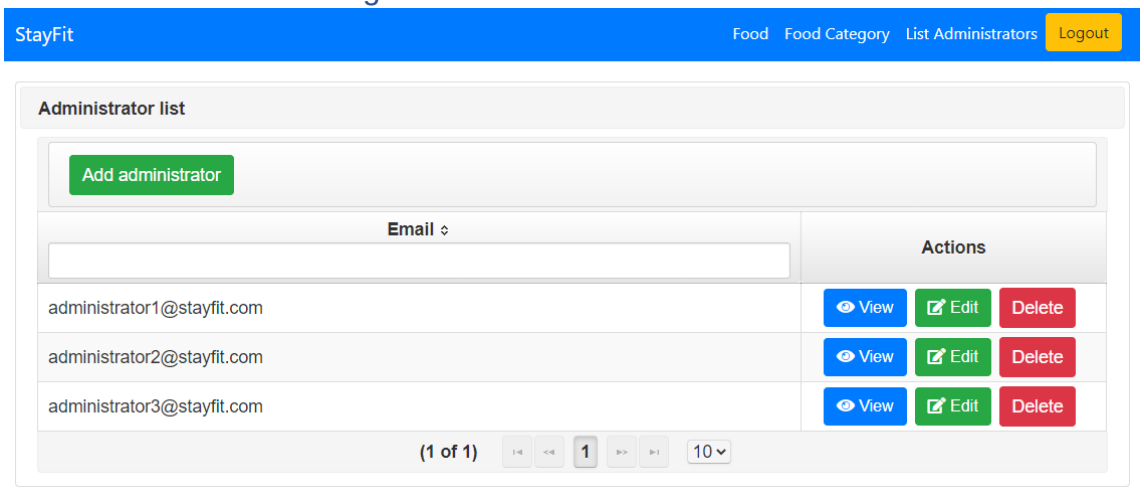


Ilustración 46. Pantalla de gestión de administradores

5. Valoración económica y viabilidad del producto

En este apartado valoraré aproximadamente los costes que ha supuesto realizar la aplicación, así como la posible viabilidad del producto en el hipotético caso de lanzarlo al mercado.

En primer lugar, debemos hacer una estimación de los costes del análisis, desarrollo e implementación del proyecto.

- **Coste de *software*:** se ha utilizado mayormente tecnologías y aplicaciones que son consideradas *software* libre, es decir, son de uso público y gratuito. Si bien es cierto, que he utilizado la licencia de MagicDraw que la UOC nos proveía, siendo esta aplicación de pago. El coste de la licencia de MagicDraw en su versión estándar es de 300$. Por lo tanto, haciendo una conversión aproximada de 1:1, estimaremos 300€ por la licencia.

- **Coste de *hardware*:** se ha utilizado mi ordenador para la realización de este proyecto, así como mi NAS personal para ir almacenando las diferentes versiones del código fuente y disponer de copias de seguridad en caso de accidente. El precio de mi ordenador (adquirido hace 3 años) es de 3.500 € y el del servidor NAS es de 600 €, ascendiendo a un total de 4.100 €. Cabe señalar que no se ha comprado el ordenador específicamente para tal propósito. Si tenemos en cuenta que el coeficiente de amortización contable para los equipos de procesos de la información es de un máximo de 25% anual y que aún está dentro de los 4 primeros años de uso, podríamos imputar al proyecto 4 meses de dicha amortización. Por lo tanto, 4.100 € * 25 % = 1.025 €. Ahora bien, si prorrateamos la cantidad a los 4 meses de duración del semestre, imputaríamos una cantidad de 342 €.

- **Coste de Internet:** he utilizado Internet para poder buscar recursos en la red, información y resolver ciertas dudas que han surgido durante la realización del presente TFG. Por tanto, debería imputar 4 meses de conexión a un precio de 24 € mensuales. Con lo que el coste total ascendería a 96 €.

- **Coste de recursos humanos:** el coste de recursos humanos corresponde a mi sueldo, ya que he sido yo el que ha realizado todas las tareas relacionadas con el proyecto. No he calculado las horas que habré destinado a la realización de este, pero si estimamos que esta asignatura son 12 ECTS y que equivalen a unas 25 horas por crédito, obtenemos un total de 300 horas. Si consideramos un sueldo de 1.200 € mensuales para un programador Junior a 40 horas semanales, tendríamos que aproximadamente, redondeando, habría llevado dos meses de sueldo de un programador cubrir esas 300 horas. Por lo tanto, estimaré unos 2.400 € de coste de recursos humanos.

Finalmente, presento un resumen de las diferentes partidas de gasto relacionadas con la realización del proyecto:

Concepto	Valor
Coste de *software*	300 €
Coste de *hardware*	342 €
Coste de Internet	96 €
Coste de recursos humanos	2.400 €
Coste TOTAL	3.138 €

Respecto a la viabilidad de lanzar el producto al mercado, podríamos utilizar un hosting de la web: https://www.cyberneticos.com/ y contratar un dominio a través de este mismo proveedor.

Los costes del hosting, si seleccionamos, por ejemplo, su plan Hosting Personal, con 4GB de almacenamiento SSD y 40 GB de transferencia mensual, para comenzar sería suficiente. Por lo tanto, podríamos estimar un coste de 6’69 € mensuales, más un coste de 7’54 € por el dominio: www.stayfit.com.es.

Finalmente, tendremos un coste anual aproximado de 90 €. Si cobrásemos una suscripción anual de 2 € por usuario, con una base de 60 usuarios ya tendríamos cubierto el coste de funcionamiento de la web. Por lo que perfectamente es una aplicación viable económicamente.

6. Trabajo futuro y posibles mejoras

En primer lugar, a pesar de contar con una aplicación funcional que puede realizar las tareas inicialmente planteadas, siempre se pueden mejorar ciertos aspectos. Sinceramente, no creo que exista todavía la aplicación perfecta que satisfaga las necesidades de todos sus usuarios.

En primer lugar, una de las características que podría implementar, es algo tan básico como permitir la selección de idioma de la aplicación. Actualmente se ha desarrollado en inglés, pero podría ampliarse a castellano y catalán, por ejemplo. Es un añadido bastante sencillo de realizar, que mejoraría la experiencia de usuario, permitiendo escoger un idioma con el que se sienta más cómodo.

En segundo lugar, otra de las mejoras que realizaría y, que me habría gustado implementar si hubiera contado con más tiempo para el desarrollo, sería crear listas personalizadas de alimentos. De tal forma que un usuario pueda crear por ejemplo “Desayuno con tostadas” e introducir varios alimentos, para que cuando quiera introducir un desayuno completo, pueda agregar ese listado de 4 o 5 elementos que va a ingerir en dicha comida. De este modo, el usuario ahorraría tiempo.

Otro aspecto que mejorar sería las pantallas de alimentos y categorías, sustituyendo el listado por un menú con imágenes que sean más agradable a la vista. Esta es una tarea pendiente que podría mejorar, en general, la interfaz de usuario de la aplicación.

Una característica bastante deseable hoy en día, sería adaptar la aplicación web a dispositivos móviles. Ya que he utilizado *Bootstrap*, no debería ser demasiado complicado realizar su portabilidad a entornos móviles. De esta manera, se puede mejorar el *software* para que los usuarios dispongan de un mayor abanico de opciones, y así poder utilizarlo y conseguir ampliar el público objetivo, ya que hoy en día la mayoría de personas utiliza un dispositivo móvil.

Otra mejora a tener en cuenta sería desarrollar una sección donde podamos introducir los ejercicios y actividades que hemos realizado a lo largo del día, y así controlar el gasto calórico de los usuarios.

7. Conclusiones

La realización de este Trabajo Final de Grado supone cerrar una bonita, y a la vez dura, etapa de mi vida. Durante este tiempo he tenido que compaginar mi vida académica con la laboral, la familiar y con una importante pérdida de un ser querido en el camino.

Así mismo, he tenido la suerte de poder adquirir los conocimientos básicos y formarme en una nueva carrera laboral en el ámbito de las nuevas tecnologías, concretamente en la informática, la cual es una de mis pasiones desde que era pequeño, cuando mis padres me regalaron mi primera videoconsola.

Durante las diferentes etapas del proyecto he podido aplicar muchos conocimientos adquiridos a lo largo del Grado de Ingeniería Informática, destacando, quizás, por haber tenido un mayor peso en el presente TFG, las siguientes asignaturas:

- **Fundamentos de programación, Prácticas de programación, Diseño y programación orientada a objetos:** las dos primeras asignaturas me sirvieron para aprender a programar y construir los cimientos de mis escasos (aunque espero aprender más con el paso del tiempo) conocimientos de programación. La tercera, por haberme introducido al mundo de la programación orientada a objetos, en concreto al lenguaje de programación Java. Gracias a estas asignaturas y a los conocimientos aprendidos en ellas, me hicieron escoger el itinerario de desarrollo de software.

- **Ingeniería del software, Ingeniería de requisitos, Ingeniería del software de componentes y sistemas distribuidos:** estas asignaturas más teóricas me han enseñado bastante sobre los procesos y metodologías que se necesitan para la creación de un proyecto de tal magnitud, ya que me enseñaron a recolectar los requisitos iniciales, analizar las necesidades de la aplicación, a realizar diseños UML tanto de clases, como diagramas de casos de uso, diagrama de componentes, etc. Han sido asignaturas de gran utilidad para la realización de este proyecto.

- **Uso de bases de datos, Diseño de bases de datos:** dichas asignaturas me introdujeron en el mundo de las bases de datos, de la persistencia. Me han enseñado a diseñar una base de datos y a comprenderlas. Asignaturas fundamentales ya que prácticamente la totalidad de las aplicaciones hoy en día necesitan persistir los datos y para poder consultarlos o extraer información para su explotación.
- **Análisis y diseño con patrones:** asignatura que me ha enseñado a diseñar mejor ciertos aspectos de la aplicación, ya que son muy útiles a la hora de estandarizar el lenguaje entre programadores, nos evita perder tiempo en buscar soluciones a problemas que ya han sido resueltos y son conocidos y a crear código reutilizable.

- **Proyecto de desarrollo de software:** esta asignatura me pareció bastante interesante y de gran utilidad, ya que nos planteaba realizar un proyecto junto

a 4 compañeros más, con las dificultades que conlleva el trabajar en equipo y sincronizar a un grupo de personas desconocidas en la elaboración de una aplicación en común. También ha sido la primera vez que desarrollaba una aplicación completa.

En cuanto a las tecnologías utilizadas en el proyecto, he escogido el conjunto de herramientas ya descrito en anteriores apartados, debido a que las he estudiado y utilizado en diferentes asignaturas. Desarrollar la aplicación con este conjunto de tecnologías me ha servido para afianzar mejor ciertos conceptos y crecer como futuro profesional del sector, ya que me gustaría introducirme en el desarrollo de aplicaciones, en particular de Java.

Considero que ha sido un reto, poder desarrollar desde cero una aplicación, pasando por sus diferentes fases. Desde la recogida de requisitos, análisis y diseño, hasta la implementación final. Me ha gustado la experiencia y seguramente intentaré realizar las mejoras propuestas en el apartado anterior para seguir creciendo y mejorando.

Por otro lado, me habría gustado disponer de más tiempo para haber escogido otro lenguaje de programación y otro conjunto de tecnologías. Pero debido al acotado tiempo disponible para realizar el Trabajo Final de Grado, opté por hacerlo de esta manera. Habrá tiempo en el futuro de seguir aprendiendo, esto no es sino el comienzo de una nueva etapa en mi carrera profesional.

8. Glosario

IDE: *Integrated Development Environment* o Entorno de desarrollo integrado, es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de *software.*

Java EE: *Java Enterprise Edition*. Plataforma de programación para desarrollar y ejecutar *software* de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya en componentes de *software* modulares, ejecutándose sobre un servidor de aplicaciones.

JPA: *Java Persistence API.* Es un *framework* del lenguaje de programación Java que maneja datos relacionales en aplicaciones utilizando la Plataforma Java en sus ediciones *Standard y Enterprise*.

Framework: es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

JSF: *Java Server Faces* es una tecnología y *framework* para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF utiliza Java Server Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías.

JSP: *Java Server Pages* es una tecnología que ayuda a los desarrolladores de *software* a crear páginas web dinámicas basadas en HTML y XML. Es similar al lenguaje PHP, pero utilizando el lenguaje de programación Java.

EJB: *Enterprise Jaba Beans* son una de las interfaces de programación de aplicaciones (API) que forman parte del estándar de construcción de aplicaciones empresariales Java EE. El objetivo de estos EJB es dotar al programador de un modelo que le permita abstraerse de los problemas de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí.

NAS: *Network Attached Storage* o Almacenamiento conectado en red, es un dispositivo de almacenamiento conectado a una red que permite almacenar y recuperar datos en un punto centralizado para los usuarios en dicha red. Es como disponer de una nube privada en casa u oficina.

9. Bibliografía

[1] M. Keith and M. Schnicariol, *Pro JPA 2. Mastering the Java Persistence API.* Editorial Apress.

[2] M. Çaliskan and O. Varaksin, *PrimeFaces Coobbook*. Second edition. Editorial Packt Publishing.

[3] Materiales de la asignatura “Ingeniería de Software de Componentes y Sistemas Distribuidos”, Universitat Oberta de Catalunya.

[4] Materiales de la asignatura “Proyecto de Desarrollo de Software”, Universitat Oberta de Catalunya.

[5] Entorno de desarrollo integrado. En *Wikipedia.* https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

[6] Java Enterprise Edition. En *Wikipedia*. https://es.wikipedia.org/wiki/Java_EE

[7] Java Persistence Api. En *Wikipedia*. https://es.wikipedia.org/wiki/Java_Persistence_API

[8] Framework. En *Wikipedia*. https://es.wikipedia.org/wiki/Framework

[9] Java Server Facs. En *Wikipedia.* https://es.wikipedia.org/wiki/JavaServer_Faces

[10] Java Server Pages. En *Wikipedia.* https://es.wikipedia.org/wiki/JavaServer_Pages

[11] Enterprise Java Beans. En *Wikipedia.* https://es.wikipedia.org/wiki/Enterprise_JavaBeans

[12] Conversor de tipo de atributo con Java Persistence API. [En línea]. Disponible en: https://www.oracle.com/technetwork/es/articles/java/convertidor-java-persistence-api-2906142-esa.html

[13] Persisting Enums in JPA. [En línea]. Disponible en: https://www.baeldung.com/jpa-persisting-enums-in-jpa

[14] The best way to map One-To-Many and Many-To-One association with JPA and Hibernate. [En línea]. Disponible en: https://vladmihalcea.com/the-best-way-to-map-a-onetomany-association-with-jpa-and-hibernate/

[15] Hibernate Exception – Simultaneously Fetch Multiple Bags. [En línea]. Disponible en: https://blog.eyallupu.com/2010/06/hibernate-exception-simultaneously.html

[16] Por qué no deberíamos utilizar getSingleResult() en JPA. [En línea]. Disponible en: http://alejandroayala.solmedia.ec/?p=947

[17] Tutorial Bootstrap. [En línea]. Disponible en: https://www.w3schools.com/bootstrap4/default.asp

[18] Ecuación de Harris-Benedict. En *Wikipedia*. https://es.wikipedia.org/wiki/Ecuaci%C3%B3n_de_Harris-Benedict

10. Anexos

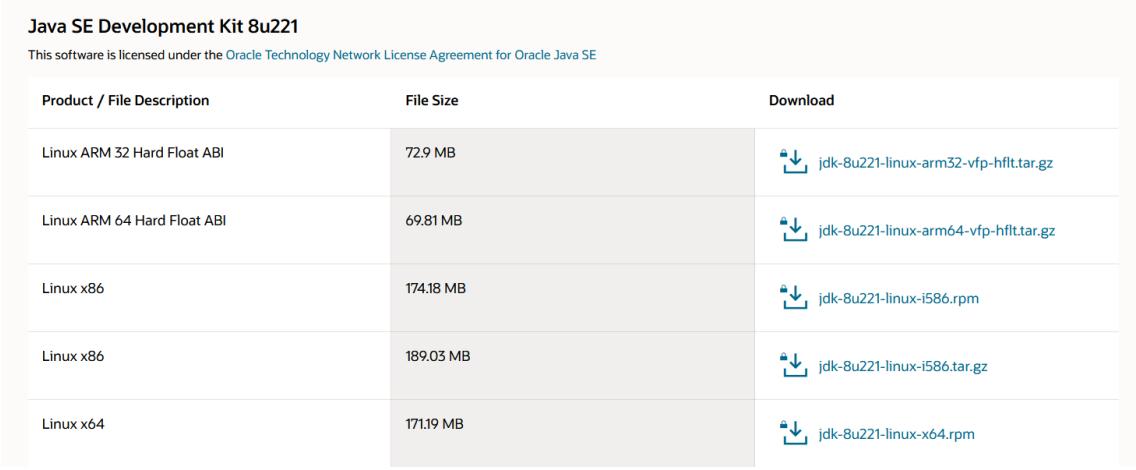
10.1. Tutorial instalación

En este capítulo voy a explicar cómo preparar el entorno para la compilación y ejecución del proyecto realizado para poder ejecutar mi Trabajo Final de Grado.

10.1.1. Java JDK

Descarga del software

En primer lugar, debemos de instalar Java Development Kit (JDK). Para ello iremos a su web de origen y descargaremos Java SE 8u221:



Java SE Development Kit 8u221

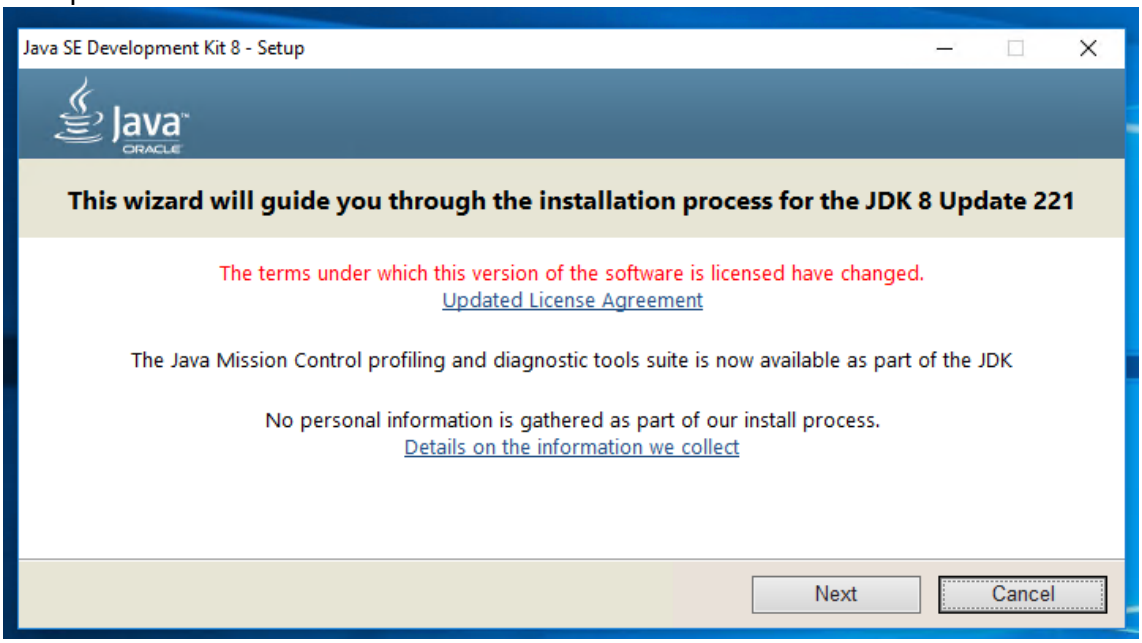
This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.9 MB	jdk-8u221-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.81 MB	jdk-8u221-linux-arm64-vfp-hflt.tar.gz
Linux x86	174.18 MB	jdk-8u221-linux-i586.rpm
Linux x86	189.03 MB	jdk-8u221-linux-i586.tar.gz
Linux x64	171.19 MB	jdk-8u221-linux-x64.rpm

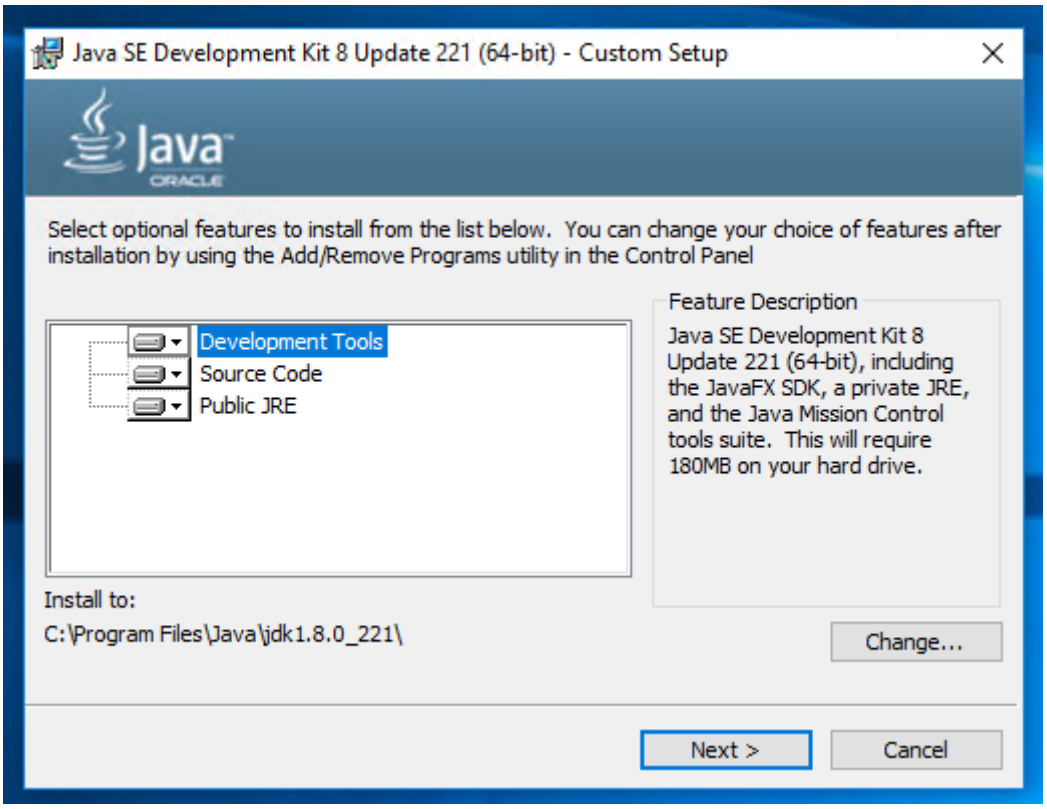
Seleccionaremos la instalación del producto acorde a nuestro sistema operativo.

Instalación de JDK

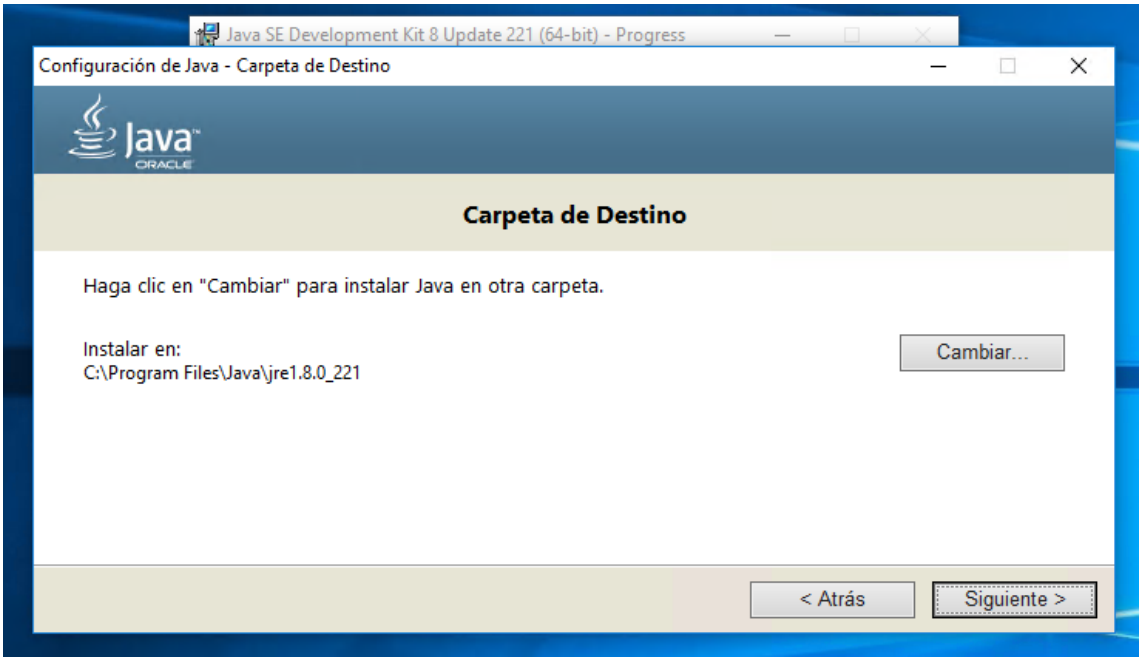
1. Ejecutamos el programa de instalación que hemos descargado previamente.
2. Aparece la ventana de instalación:



Pulsamos en Next. En la siguiente ventana podemos dejar las opciones por defecto y pulsar el botón Next de nuevo:

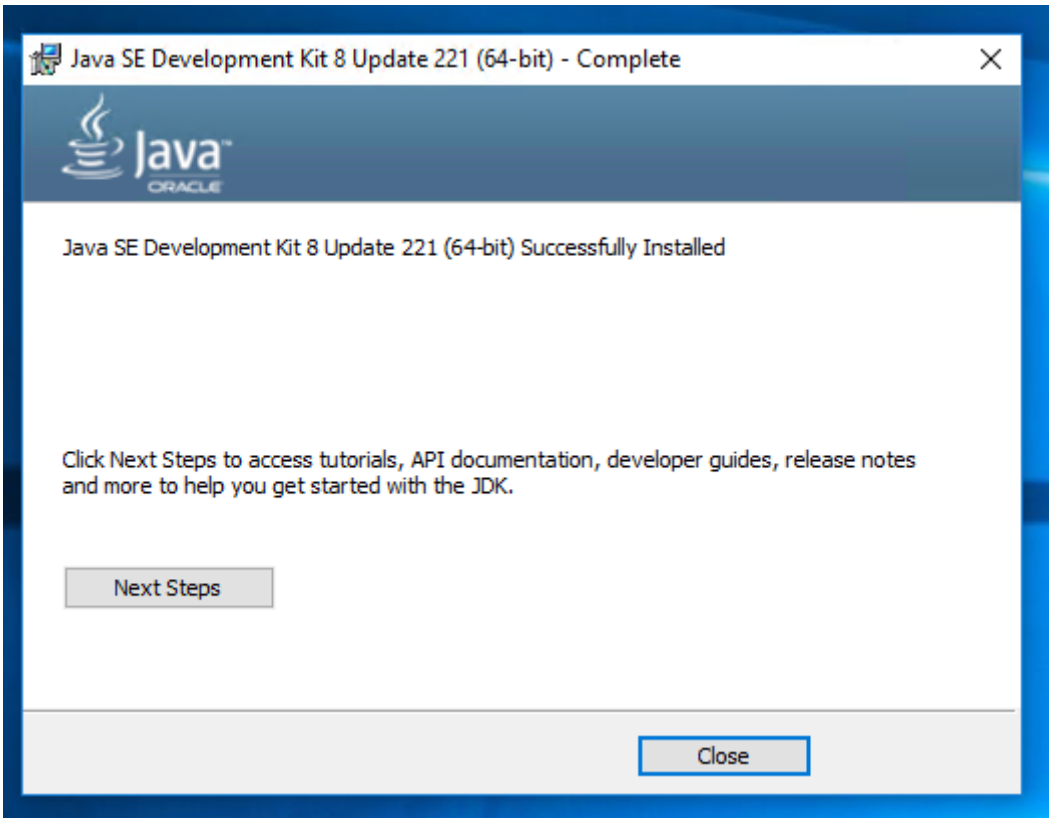


El programa de instalación trabajará y nos preguntará al cabo de unos segundos si queremos cambiar la carpeta de destino:



Pulsamos en Siguiente y el proceso de instalación continuará:

Una vez finalizada la instalación, pulsaremos sobre el botón Close:

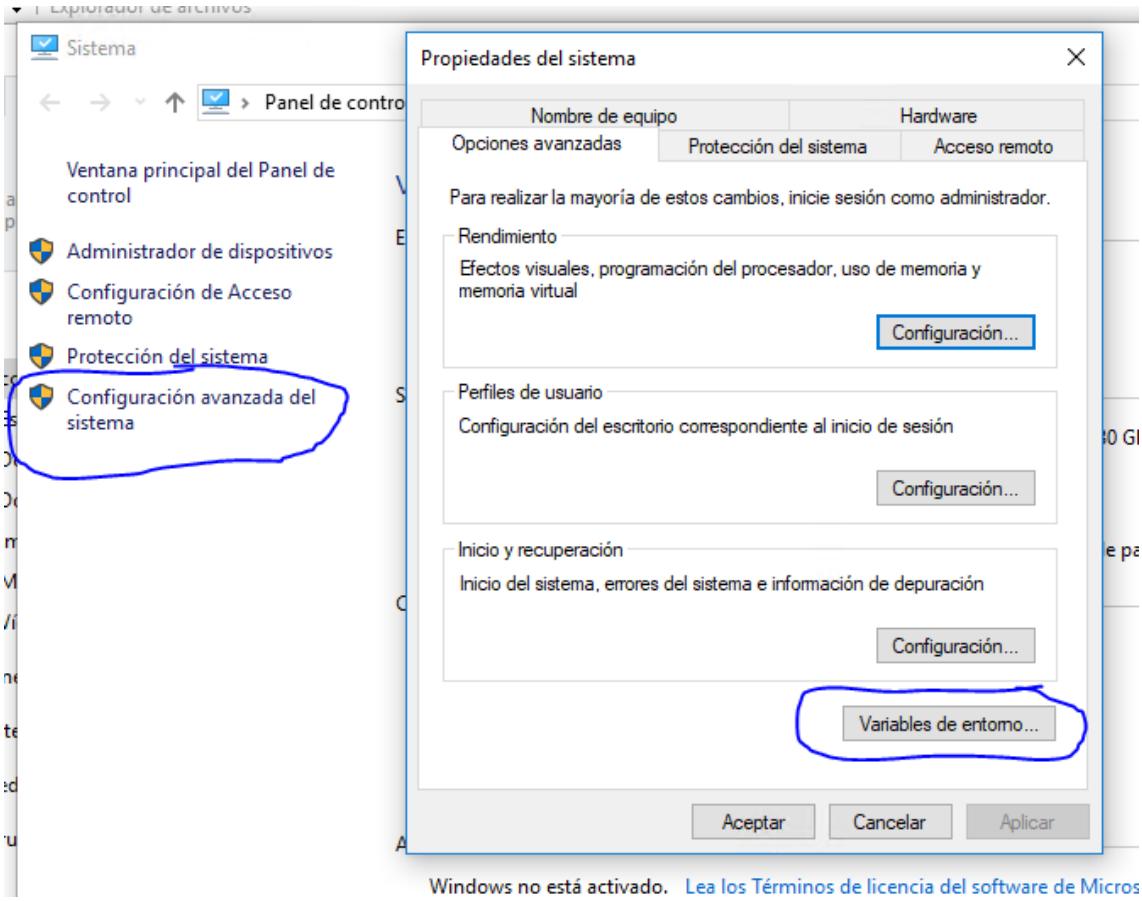


Ya tenemos JDK instalado.

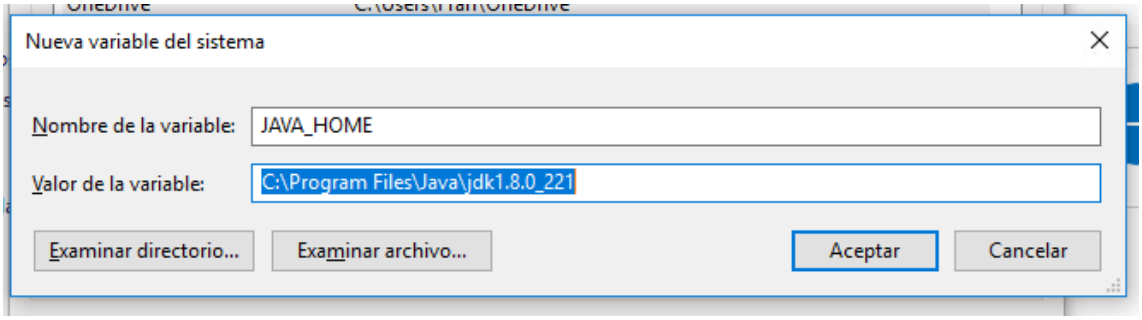
Configuración de JDK

Procedemos a configurar las variables de Entorno.

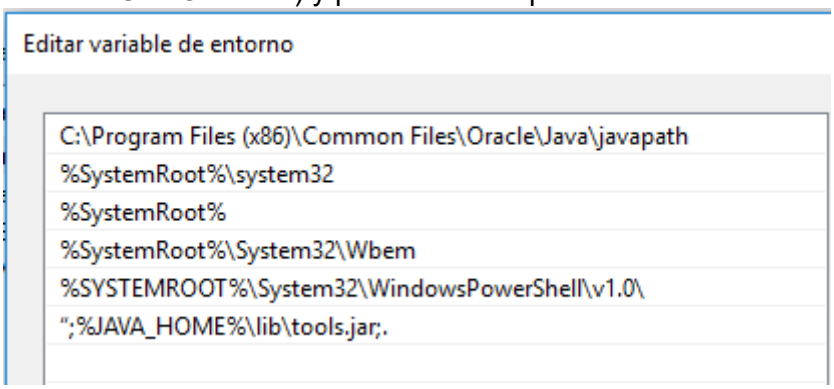
1. Pulsamos con el botón derecho sobre “Mi PC” y seleccionamos “Propiedades”.
2. Seleccionamos en el menú lateral izquierdo “Configuración avanzada del sistema” y pulsamos en el botón “Variables de entorno”.



3. En la ventana de variables de entorno crearemos la variable de Sistema: “JAVA_HOME”. Como valor ponemos el directorio donde se encuentra JAVA y pulsamos el botón aceptar para crearla:



4. Buscamos la variable llamada “PATH”, hacemos click sobre ella y pulsamos el botón “Editar”. En la nueva ventana pulsamos sobre el botón “Nuevo” y añadimos: “;%JAVA_HOME%\lib\tools.jar;.” (No olvidar el PUNTO al final) y pulsamos “Aceptar”.

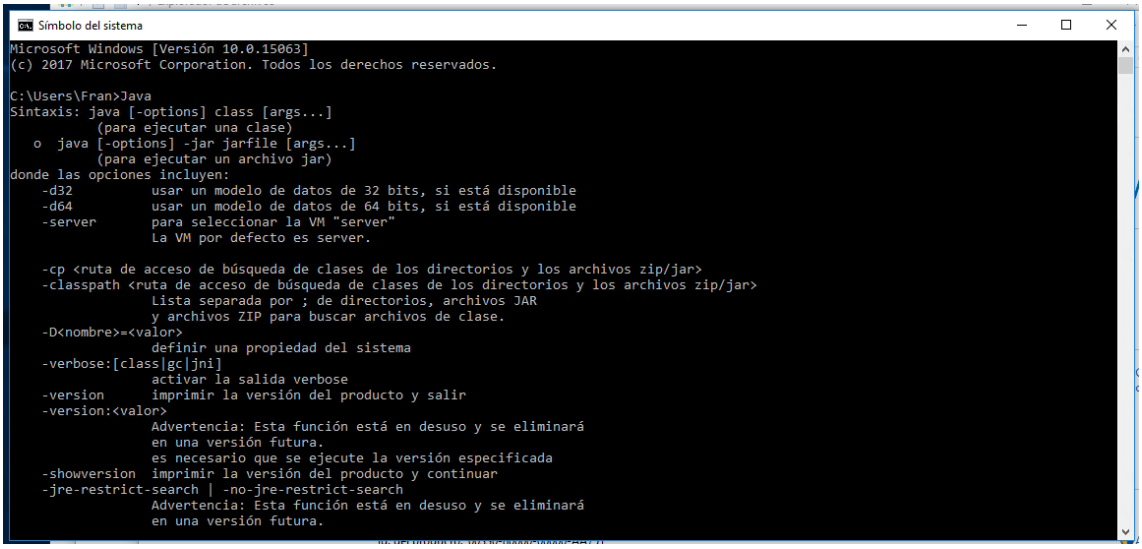


5. Pulsamos aceptar en la ventana de variables de entorno para guardar los cambios realizados.
6. Pulsamos aceptar en la ventana de propiedades del sistema para guardar también los cambios.

Para comprobar el correcto funcionamiento, abrimos la “Consola del Sistema” (podemos poner “cmd” en el botón de ejecutar, por ejemplo).

Una vez abierta la consola, escribimos: Java

Si aparece un mensaje como este:



Significa que está correctamente configurado. En caso contrario, algo habrá fallado en la configuración de las variables de entorno. Debemos revisarlas por si no son correctas.

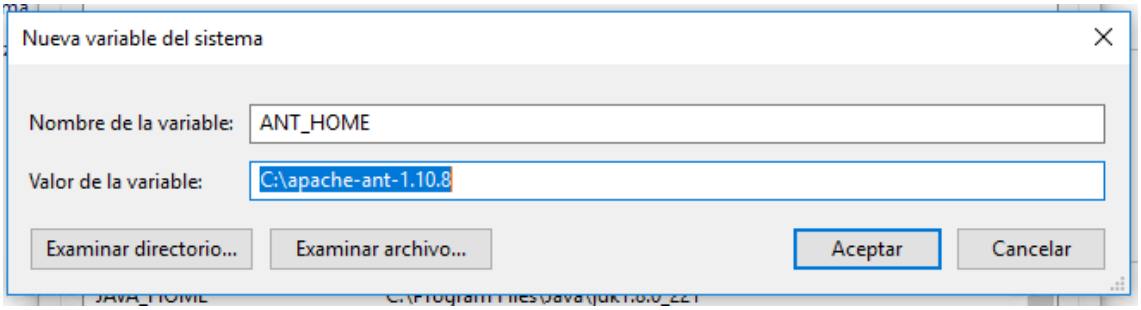
10.1.2. Apache Ant

Descarga del software

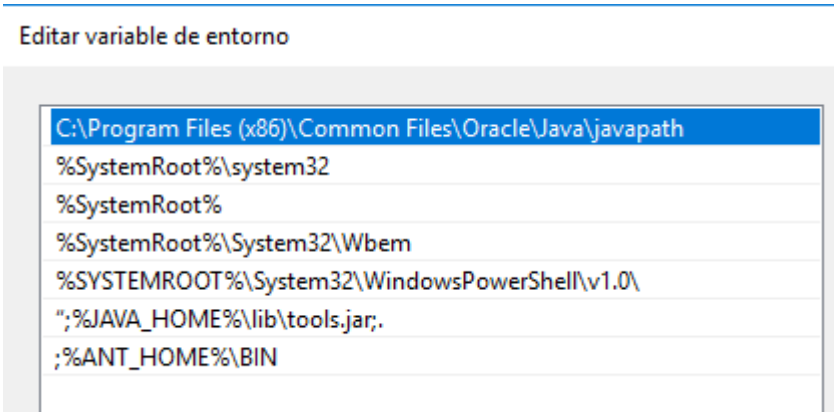
Descargamos el programa desde https://ant.apache.org/bindownload.cgi . He utilizado la versión Ant 1.10.5, pero en la web encontramos la 1.10.8. La descargamos.

Instalación de Ant

1. Descomprimimos el fichero ZIP en la raíz del disco duro.
2. Creamos la variable del Sistema con el nombre ANT_HOME (los pasos son los mismos que para crear la configuración de JAVA_HOME que hemos realizado en el anterior paso de este tutorial). Como valor establecemos el directorio donde está instalado Ant, en la raíz de nuestro disco duro:



3. Buscamos la variable “PATH”, y añadimos al final: “;%ANT_HOME%\BIN” (si existe esta variable tanto cómo variable del sistema cómo de usuario, realizamos la misma operación en ambos). Quedará la variable “PATH” así:



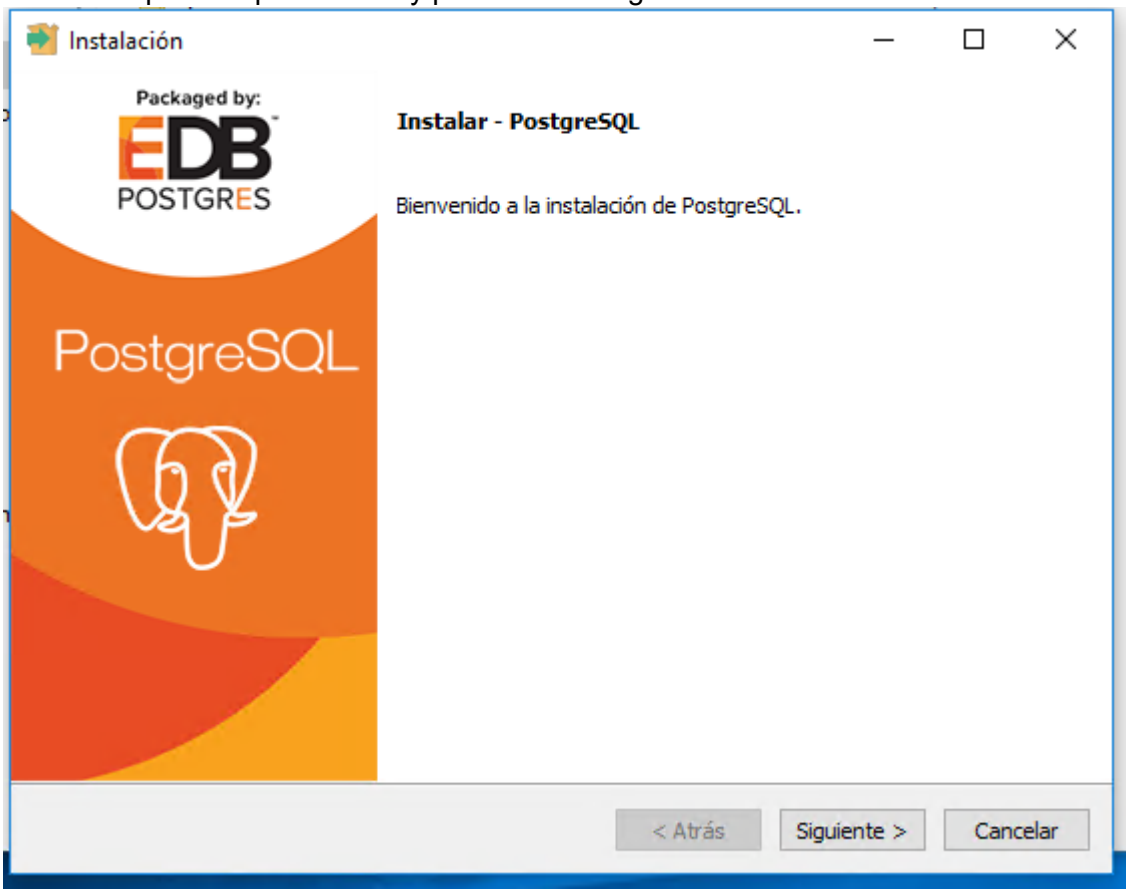
10.1.3. PostgreSQL

Descarga del software

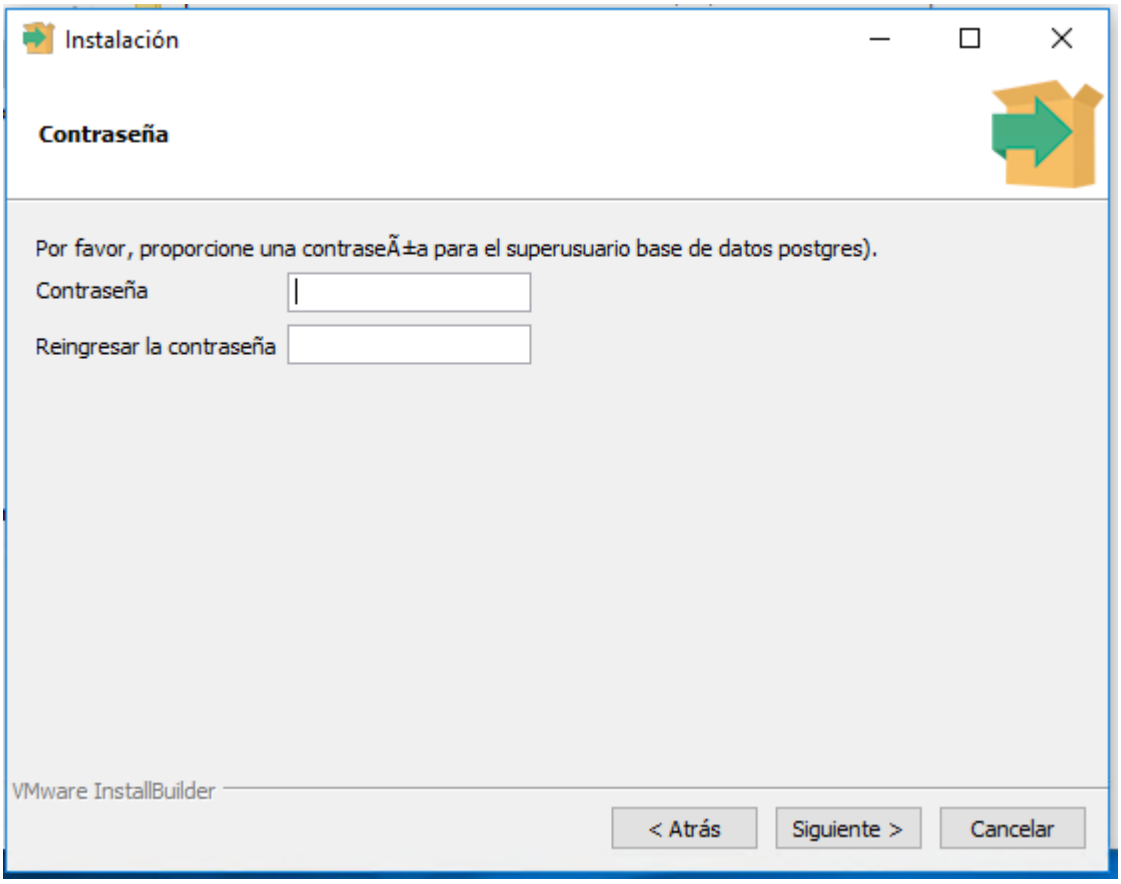
Descargamos el programa desde su web: https://www.postgresql.org/download/ . En mi proyecto he utilizado la versión PostgreSQL 10.10.1

Instalación de PostgreSQL

1. Ejecutamos el fichero que hemos descargado (hacerlo como usuario administrador).
2. Cuando nos aparezca la ventana de instalación del programa dejaremos todas las opciones por defecto y pulsaremos “Siguiente”:



Llegado el momento, nos pedirá que configuremos una contraseña para el superusuario de la base de datos:



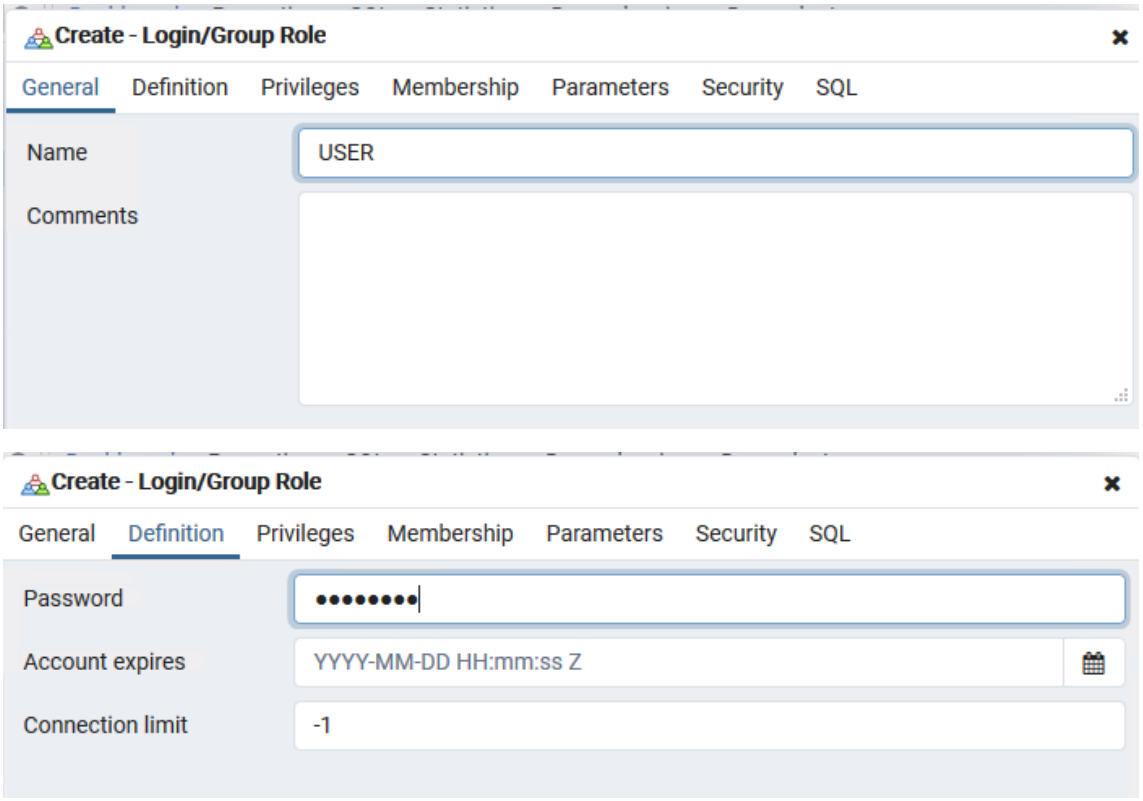
Introducimos “PASSWORD” por ejemplo.

3. Cuando aparezca una ventana con selección de puerto, dejaremos por defecto el que viene asignado. Continuaremos dejando todas las opciones por defecto hasta completar la instalación.

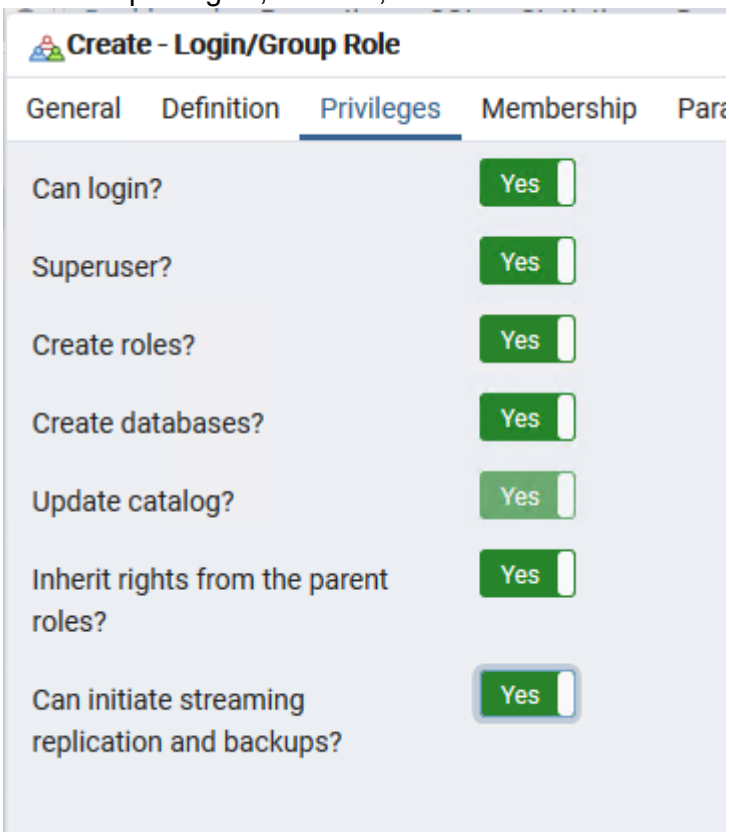
Configurar PostgreSQL

Ahora preparamos PostgreSQL para que pueda ser utilizado por mi proyecto.

1. Accedemos a PostgreSQL mediante la aplicación pgAdmin 4 (pulsamos en el botón de inicio de Windows y buscamos el programa).
2. Al ser la primera vez que lo ejecutamos nos pedirá una contraseña maestra para el usuario pgAdmin. Ponemos “PASSWORD”.
3. Nos conectamos al servidor PostgreSQL haciendo click sobre el (o boton derecho y pulsando en connect).
4. Ahora crearemos un nuevo Rol. Pulsamos botón derecho sobre la sección “Login/Group Roles” y lo creamos con los datos: USER y PASSWORD para el usuario y contraseña, respectivamente. Debemos de hacerlo en la primera pestaña “General” y en “Definition”, como vemos a continuación en las siguientes capturas:

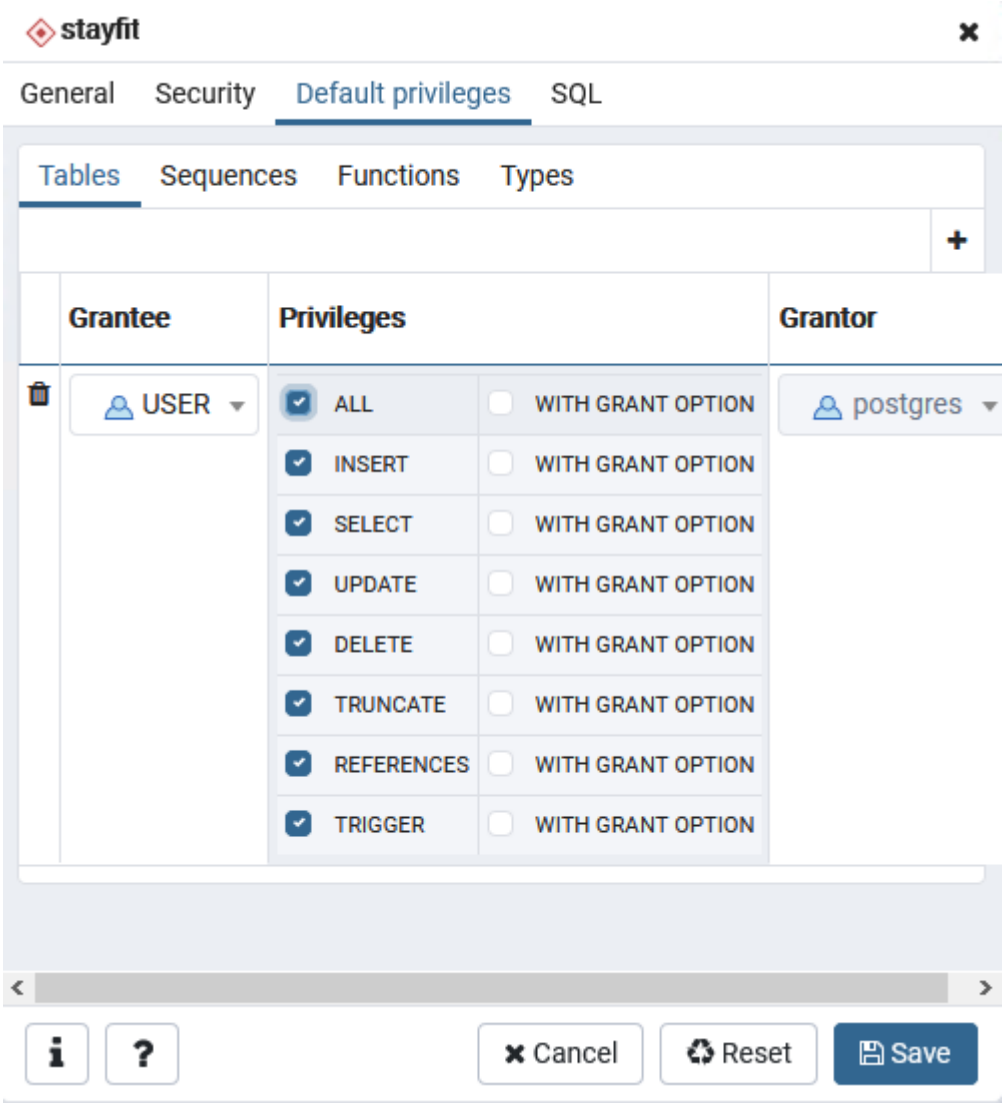


5. Ahora, sin cerrar la ventana de roles, en la sección “Privileges” le daremos todos los privilegios, es decir, marcamos todo “Yes”:



6. Pulsamos sobre el botón de “Save” para guardar los cambios realizados.
7. Ahora creamos un nuevo esquema con el nombre “stayfit”. Seleccionamos como “Owner” USER y con privilegios “ALL”:

8. Crearemos un nuevo esquema con el nombre “stayfit” y asignamos como owner USER con privilegios “ALL”:



Pulsamos en guardar y ya lo tenemos. Aunque este paso no es estrictamente necesario, ya que he realizado un script para que se cree automáticamente, no está de más crearlo ahora, ya que también hace un DROP CASCADE en caso de existir.

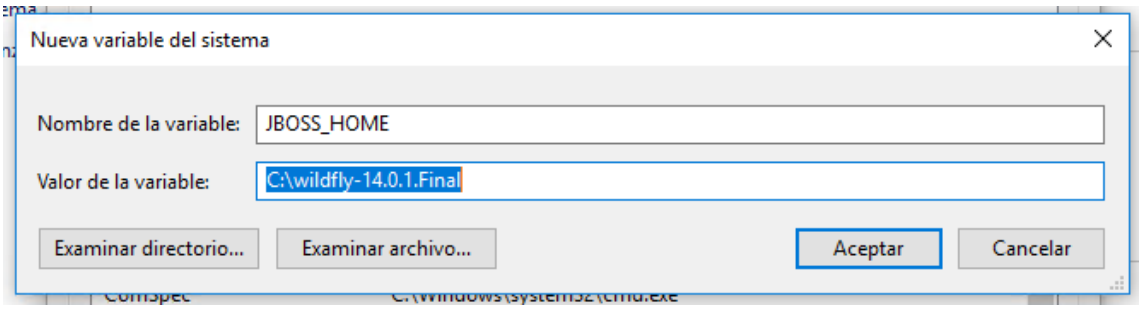
10.1.4. Wilfdly (o JBoss)

Descarga del software

Descargaremos el programa desde su web: https://wildfly.org/downloads/ . He utilizado la versión 14.0.1, por lo que lo mejor es que descarguemos dicha versión: https://download.jboss.org/wildfly/14.0.1.Final/wildfly-14.0.1.Final.zip .

Instalación de Wildfly

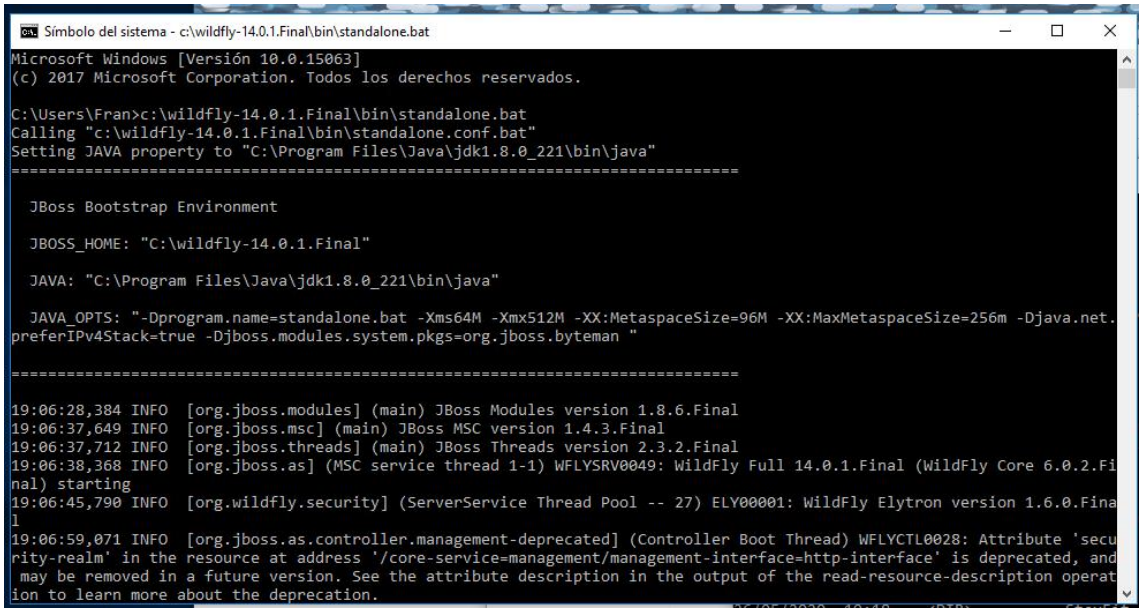
1. Una vez descargado el fichero .zip de la página, simplemente tendremos que descomprimirlo en la raíz de nuestro disco duro, quedando así: "C:\Wildfly-14.0.1.Final" y dentro quedara la aplicación.
2. Crearemos una variable de Sistema y de Usuario con el nombre JBOSS_HOME y como valor seleccionaremos la carpeta donde está instalado Wildfly:



3. Agregamos también a la variable "PATH" (como hemos hecho antes con Ant y Java): ";%JBOSS_HOME%\bin".
4. Aceptamos esa y el resto de las ventanas para confirmar los cambios realizados.

Ejecución de Wildfly

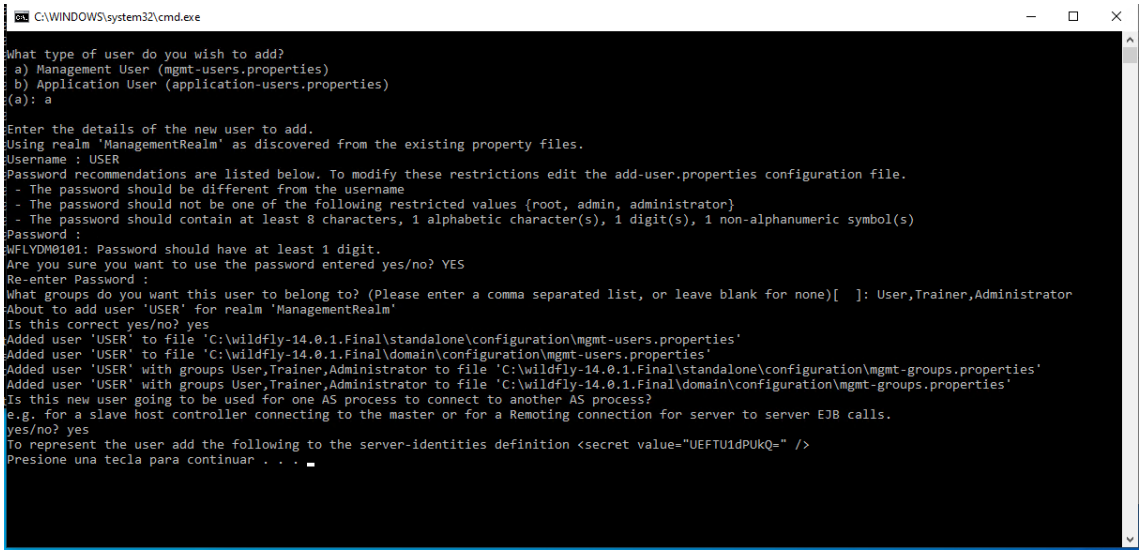
Podemos comprobar que funciona todo correctamente escribiendo la ruta hacia el fichero "standalone.bat" del directorio donde hemos instalado Wildfly, que debería ser: "C:\wildfly-14.0.1.Final\bin\standalone.bat". Lo haremos mediante consola:

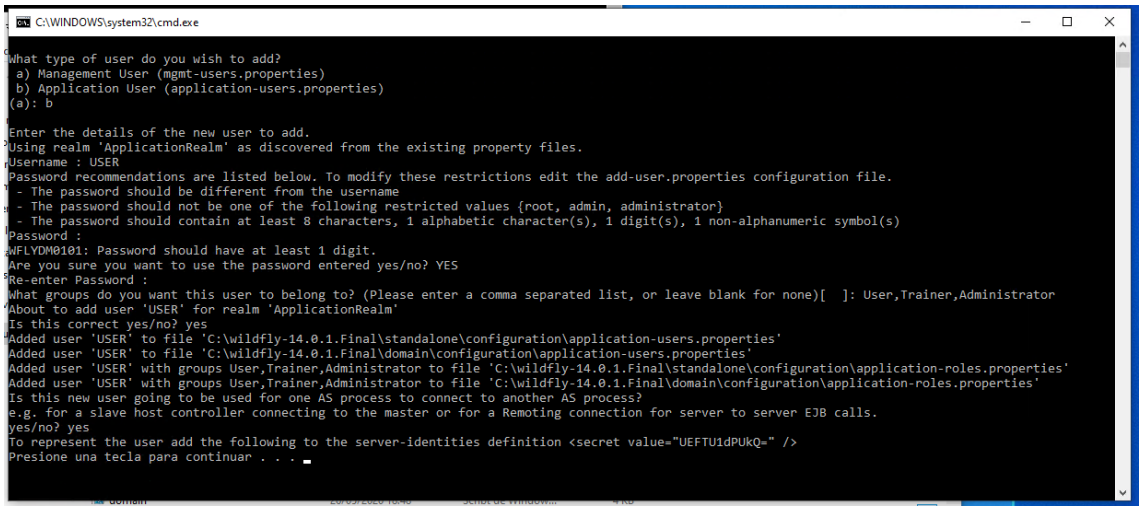


Luego podemos comprobar que funciona abriendo el navegador de Internet preferido y escribir: http://localhost:8080 . Aparecerá lo siguiente:

Para obtener acceso a la consola de JBoss, y para poder acceder de forma remota a los componentes de JBoss, es necesario dar de alta usuarios tipo ManagementRealm y ApplicationRealm, por lo que deberemos de realizar el siguiente paso para cada uno de ellos.

Utilizaremos la herramienta add-user.bat que se encuentra en el directorio \bin dentro de la carpeta de Wildfly:





Se ha utilizado como Username: USER y como Password: PASSWORD. En caso de utilizar otro nombre y clave, será necesario modificar el código de los ejemplos en el lugar donde se piden estos valores.

Preparación del entorno de trabajo de Wildfly

Finalmente, debemos tener las siguientes variables definidas:

- JAVA_HOME: debe contener el directorio donde hemos instalado el JDK.
- ANT_HOME: debe contener el directorio donde se ha instalado Ant.
- JBOSS_HOME: debe contener el directorio donde se ha instalado Wildfly.
- PATH: debe contener el subdirectorio bin de todo el software que utilizamos: %JAVA_HOME%\bin;%ANT_HOME%\bin;%JBOSS_HOME%\bin;%PATH% .
- CLASSPATH: debe contener el fichero tolos.jar del JDK.

10.1.5. Instalación del conector Jaba JDBC PostgreSQL en Wildfly

1. Nos descargaremos el conector Java JDBC de PostgreSQL desde la web: https://jdbc.postgresql.org/ , concretamente, este: https://jdbc.postgresql.org/download/postgresql-9.4.1209.jar

2. Crearemos la estructura: \postgresql\main en la carpeta Wildfly-14.0.1.Final\modules\system\layers\base\org, así la estructura quedará: Wildfly-14.0.1.Final\modules\system\layers\base\org\postgresql\main.
3. Copiamos el .jar postgresql-9.4.1209.jar, correspondiente al controlador JDBC de PostgreSQL que hemos descargado, en la carpeta Wildfly-14.0.1.Final\modules\system\layers\base\org\postgresql\main.
4. En la misma carpeta, creamos un fichero con el nombre “module.xml” y con el siguiente texto:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="org.postgresql">
<resources>
<resource-root path="postgresql-9.4.1209.jar"/>
</resources>
<dependencies>
<module name="javax.api"/>
<module name="javax.transaction.api"/>
```

```
</dependencies>
</module>
```

5. Para que Wildfly reconozca este datasource es necesario añadir en el xml Wildfly\standalone\configuration\standalone.xml, concretamente colgando de la etiqueta <datasources>, el siguiente texto:

```
<datasource jta="false" jndi-name="java:jboss/postgresDS" pool-name="postgresDS" enabled="true" use-
java-context="true" use-ccm="false">
   <connection-url>jdbc:postgresql://localhost:5432/postgres</connection-url>
   <driver-class>org.postgresql.Driver</driver-class>
   <driver>postgresql</driver>
   <pool/>
   <security>
            <user-name>USER</user-name>
            <password>PASSWORD</password>
   </security>
   <statement/>
</datasource>
```

USER y PASSWORD corresponden al usuario y contraseña creados en PostgreSQL en el apartado que he explicado la configuración del mismo en este mismo documento. El 5432 corresponde al puerto donde escucha PostgreSQL (si hemos cambiado el puerto en la instalación, habrá que cambiarlo por el puerto que hayamos escogido).

Así, colgando de la etiqueta <drivers>:

```
<driver name="postgresql" module="org.postgresql">
  <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
</driver>
```

Así, la etiqueta <datasources> debe quedar:

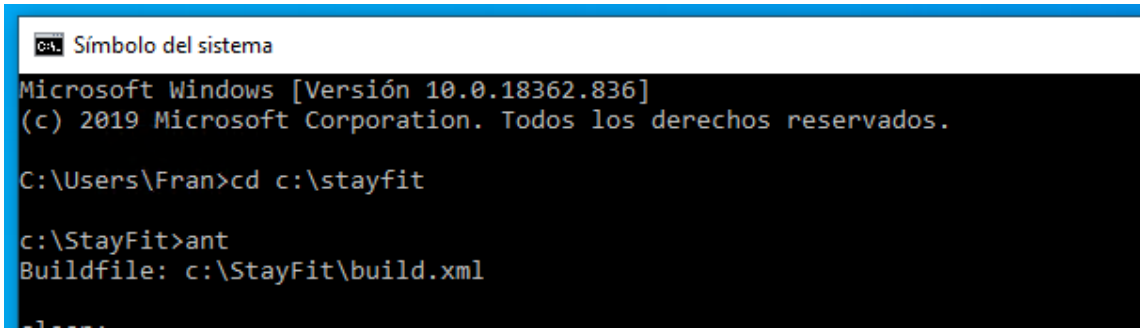
```
<datasources>
    <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS"    enabled="true"
    use-java-context="true">
      <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
      <driver>h2</driver>
      <security>
            <user-name>sa</user-name>
            <password>sa</password>
   </security>
  </datasource>
  <datasource jta="false" jndi-name="java:jboss/postgresDS" pool- name="postgresDS" enabled="true" use-
java-context="true" use-ccm="false">
   <connection-url>jdbc:postgresql://localhost:5432/postgres</connection-url>
   <driver-class>org.postgresql.Driver</driver-class>
   <driver>postgresql</driver>
   <pool/>
   <security>
            <user-name>USER</user-name>
            <password>PASSWORD</password>
   </security>
   <statement/>
  </datasource>
  <drivers>
   <driver name="h2" module="com.h2database.h2">
     <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
   </driver>
   <driver name="postgresql" module="org.postgresql">
     <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource- class>
   </driver>
  </drivers>
</datasources>
```

6. Fin de la configuración del servidor de aplicaciones.

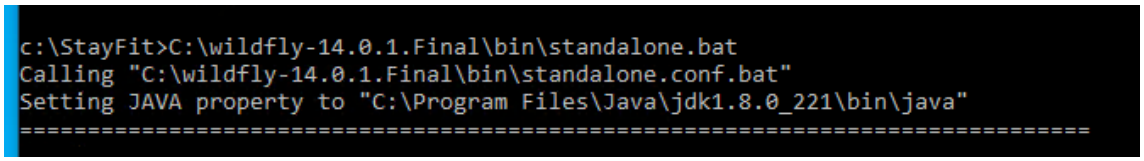
10.2. Ejecución de la aplicación StayFit

Ahora ya tenemos todo lo necesario para compilar y desplegar el proyecto para poder probarlo y hacer uso de este.

Abriremos la consola y navegamos hasta el directorio donde está el proyecto StayFit, en mi caso lo tengo en la raíz del disco duro, así que bastará con escribir la ruta hacia la misma y después escribiremos: ant.



Ahora tenemos el proyecto compilado. Falta ejecutar el servidor de aplicaciones Wildfly para desplegar el proyecto:



Ejecutando el fichero "standalone.bat" dentro de C:\wildfly-14.0.1.Final\bin\standalone.bat, ya tendremos el servidor ejecutándose y la aplicación en funcionamiento.

Ahora solo falta ir a la web: http://localhost:8080/StayFit y ¡probar la aplicación!

Para poder utilizar los usuarios y administradores que hay registrados en la aplicación se debe de utilizar las siguientes credenciales:

Nombre de usuario (email)	Contraseña
administrator1@stayfit.com	fuertaco
administrator2@stayfit.com	fuertaco
administrator3@stayfit.com	fuertaco
user1@stayfit.com	dieta
user2@stayfit.com	dieta
user3@stayfit.com	dieta