UNIVERSIDADE
DE VIGO

ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Memoria do Traballo de Fin de Grao que presenta

D. Francisco López Alonso

para a obtención do Título de Graduado en Enxeñaría

Informática **Gestión de Dietas y Entrenamientos**

Febreiro, 2021

Traballo de Fin de Grao Nº:

Titor/a: Miguel Reboiro Jato
Área de coñecemento: Linguaxes e Sistemas Informáticos
Departamento: Informática

Contenido

Índice de Tablas

Índice de Ilustraciones

1. Introducción

¿Por qué la mayoría de la gente que se plantea objetivos físicos y alimenticios fracasa en el intento? Casi todo el mundo que se ha propuesto bajar o subir de peso ha tenido muchas dudas de como estructurar su alimentación o qué cantidades de alimentos injerir para poder cumplir con el propósito final. Siempre que empezamos una dieta, pensamos que con comer sano es suficiente, sin embargo, influyen muchos más factores. Algunos de ellos son: la cantidad de alimentos que utilizamos en cada comida, nuestra actividad física, nuestro metabolismo basal o la calidad de la comida que injerimos.

Uno de los errores más comunes que comete la gente es aplicar los hábitos alimenticios de otras personas en su día a día. Esto es así, dado que lo que le pueda funcionar a una persona, no tiene porque funcionarle a otra. Como hemos mencionado, el metabolismo basal es un factor importante a tener en cuenta para saber cuáles son las necesidades calóricas de una persona. Además, no todo el mundo tiene el mismo ritmo de vida y actividad física lo que influye ostensiblemente en el éxito de nuestra dieta. Por lo tanto, debemos entender que una dieta debe ajustarse a los hábitos de cada persona y a su constitución física.

Otro factor a tener en cuenta es el ejercicio físico. A la hora de estructurar un entrenamiento, la complejidad del mismo varía en función de los objetivos físicos. No es lo mismo una persona cuyo único objetivo es bajar de peso respecto a otra que quiere ganar masa muscular y mantener al mismo tiempo un cierto grado de definición física. Por lo tanto, el entrenamiento físico será otra parte fundamental para alcanzar el propósito final.

En conclusión, en el presente documento se pretende exponer el desarrollo de una aplicación web que permita gestionar y estructurar la alimentación de los usuarios en base a su metabolismo basal y preferencias alimenticias, además de gestionar entrenamientos físicos en base a sus preferencias deportivas. Con ello, los usuarios de la plataforma dispondrán, por un lado, de una guía sobre qué comer y en qué cantidades y, por otro lado, una guía sobre qué ejercicios realizar y la frecuencia de cada uno de ellos.

2. Objetivos

El objetivo de este trabajo es, por un lado, la creación de una aplicación web que permita gestionar la alimentación de un usuario a través de la creación de dietas, y por otro lado, la creación de entrenamientos en base a sus preferencias deportivas.

En primer lugar, la aplicación debe disponer de funcionalidades que permitan calcular las calorías objetivo (totales y por macronutriente) del cliente. De esta manera, se partirá de una base numérica para poder controlar la cantidad de calorías que lleva la dieta a medida que se va añadiendo a la misma los alimentos y comidas disponibles en sistema. A partir de aquí, se deduce que la plataforma debe proporcionar información alimenticia de los alimentos/comidas que queremos añadir a la dieta. Si queremos añadir un alimento/comida que no está disponible en la plataforma, podremos añadirlo.

En segundo lugar, la creación de los entrenamientos tendrá una misma base operativa. Sin embargo, será el usuario entrenador el encargado de crearlos en base a las preferencias deportivas del usuario cliente. Para ello, la plataforma incluye la posibilidad de dar de alta nuevos ejercicios y poder añadirlos al entrenamiento del usuario. Dicho esto, para añadir el ejercicio serán necesarios datos como el número de series, las repeticiones por serie y el descanso entre cada una de las series.

3. Resumen

3.1. Solución aportada

Para la solución aportada se ha hecho uso del marco de desarrollo Django. Django es un framework web de alto nivel desarrollado en Python. Una de sus virtudes, es la alta seguridad que proporciona al sitio web que estamos desarrollando. Así, Django nos ayuda a no cometer los errores más comunes a la hora de programar, pudiendo centrarnos en el desarrollo de la aplicación y la escritura de código.

Además de Django, se hace uso de JQuery para el manejo de los elementos que componen el archivo HTML que carga el navegador, JavaScript para algunas validaciones y el manejo de campos de formulario, HTML y CSS para el despliegue de la aplicación en el navegador y su modelaje a nivel de front-end y Bootstrap para mejorar el diseño del sitio web.

3.2. Metodología de Desarrollo de Software

La metodología de software utilizada en el desarrollo de este proyecto es el Proceso Unificado (Unified Process). El Proceso unificado en un marco de desarrollo de software que se basa en el refinamiento del sistema a través de iteraciones. Esta metodología se caracteriza por estar compuesta de 4 fases, donde cada fase tiene una serie de iteraciones. Cada iteración completada representa un nuevo incremento en el sistema, y por lo tanto, una evolución en el mismo. En la Figura 1 se representa el desarrollo del Proceso Unificado. Las 4 fases son:

1. **Inicio**: Es la fase más corta del proyecto. En esta primera etapa, se planifica el desarrollo del proyecto, su alcance, se desarrolla el diagrama de casos de uso y el diagrama de Gantt y se estiman los costes y los riesgos asociados. La fase no puede prolongarse mucho; si lo hace, es indicativo de que los objetivos del proyecto no están claros.
2. **Elaboración**: El objetivo de esta fase es capturar todos los requisitos del sistema para estimar el riesgo y coste del proyecto. Además, se crea el diseño y la arquitectura del sistema mediante el modelaje de diagramas. De esta forma se consigue una hoja de ruta más nítida sobre los hitos y objetivos a conseguir.
3. **Construcción**: Usando como base los artefactos generados en la fase de elaboración, en la fase de construcción se finaliza y refina el diseño del sistema. Con diferencia, es la fase más larga del sistema y está dividida en múltiples iteraciones
4. **Transición**: Fase final del proyecto. En la fase de transición de entrega el sistema a sus usuarios finales.

Finalmente, también se hace uso de UML (Unified Modeling Language). UML es un lenguaje de modelado de sistemas que se usa para la visualización, especificación, construcción y documentación del sistema.

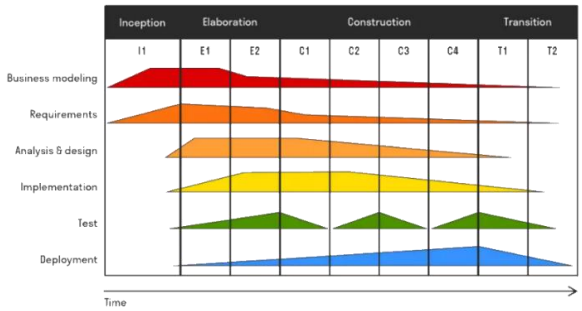

Ilustración 1. Fases del proceso unificado

4. Planificación y Seguimiento

4.1. Planificación

Se ha elaborado una planificación que tiene en cuenta posibles retrasos debido a poca formación para completar una fase. La duración del proyecto será de 128 días con una media de 3h y media por día, de lunes a domingo.

En la Tabla 1 se expone la planificación del proyecto.

Inicio – 6 días 1 iteración	1. Planificar recursos 2. Estimar coste y tiempo 3. Planificación proyecto 4. Manejo del riesgo 5. Desarrollo del diagrama casos de uso y el diagrama de Gantt
Elaboración – 10 días 1 iteración	1. Análisis del dominio del problema 2. Implementación de la arquitectura inicial
Construcción – 57 días 4 iteraciones	1º Iteración – 10 días: • Implementación estructura aplicación • Implementación consulta de tarifas • Implementación sistema de pago • Modelaje ficha de usuario • Pruebas 2º Iteración – 7 días: • Implementación gestión de comidas • Implementación gestión de alimentos • Implementación asignación de entrenadores • Pruebas 3º Iteración – 17 días: • Implementación gestión de dietas • Implementación gestión de entrenamientos • Pruebas 4º Iteración – 21 días: • Implementación chat • Implementación gestión de permisos • Pruebas
Transición – 51 días 1 iteración	1. Revisión de las historias de usuario 2. Documentación 3. Manual de Usuario

Tabla 1. Planificación del proyecto

Así mismo, en la Figura 2 se muestra una visión general de la planificación del proyecto:

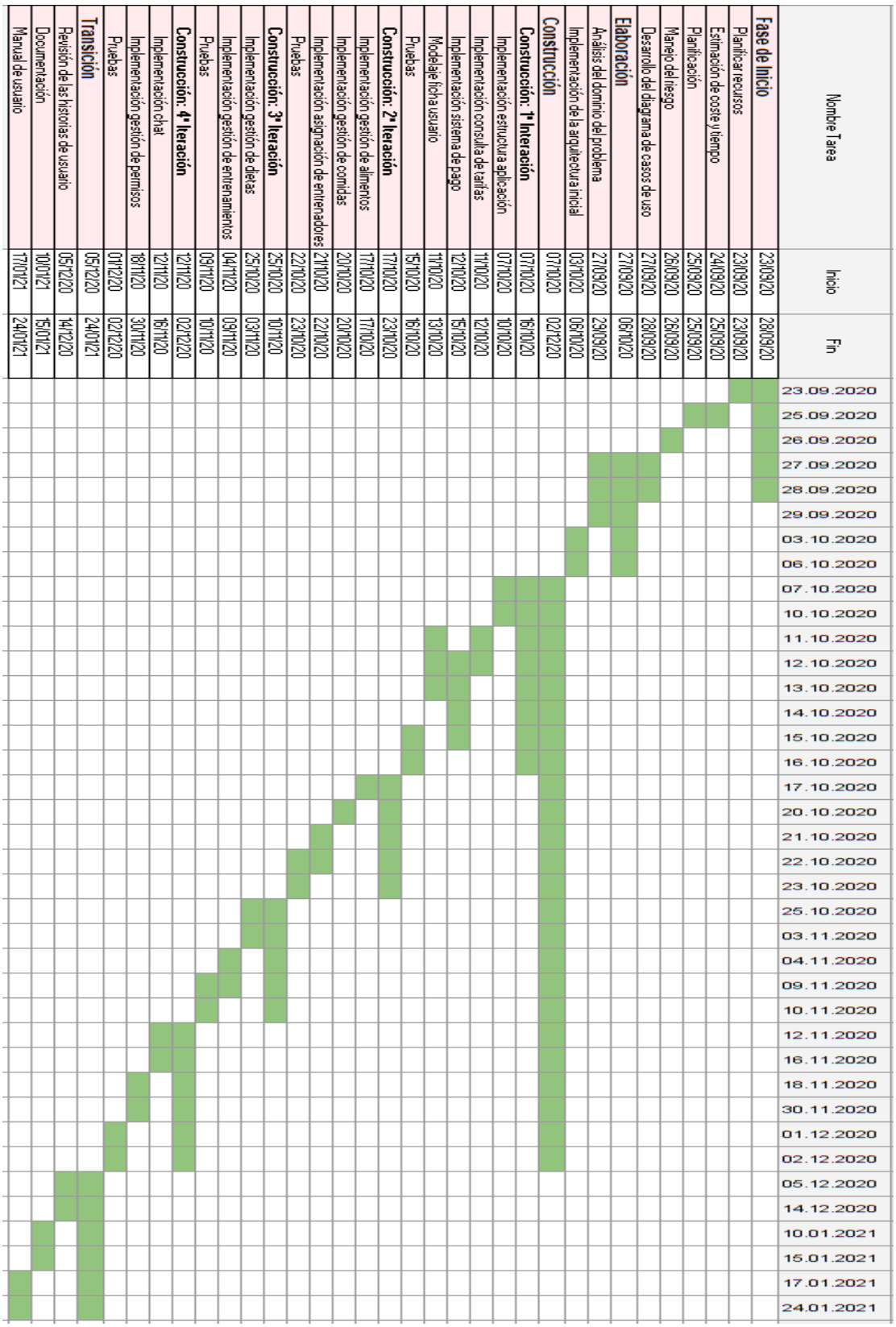

Ilustración 2. Diagrama Gantt de Planificación

4.2. Seguimiento

He cometido errores de novato en el diseño de algunos artefactos que retrasaron el inicio de algunas fases. En la siguiente tabla se muestra el proceso de desarrollo real:

Inicio – 17 días 1 iteración	1. Planificar recursos 2. Estimar coste y tiempo 3. Planificación proyecto 4. Manejo del riesgo 5. Desarrollo del diagrama casos de uso y el diagrama de Gantt
Elaboración – 2 días 1 iteración	1. Análisis del dominio del problema 2. Implementación de la arquitectura inicial
Construcción – 64 días 4 iteraciones	1º Iteración – 10 días: • Implementación estructura aplicación • Implementación consulta de tarifas • Implementación sistema de pago • Modelaje ficha de usuario • Pruebas 2º Iteración – 7 días: • Implementación gestión de comidas • Implementación gestión de alimentos • Implementación asignación de entrenadores • Pruebas 3º Iteración – 17 días: • Implementación gestión de dietas • Implementación gestión de entrenamientos • Pruebas 4º Iteración – 21 días: • Implementación chat • Implementación gestión de permisos • Pruebas
Transición – 50 días 1 iteración	1. Revisión de las historias de usuario 2. Documentación 3. Manual de Usuario

Tabla 2. Seguimiento del proyecto

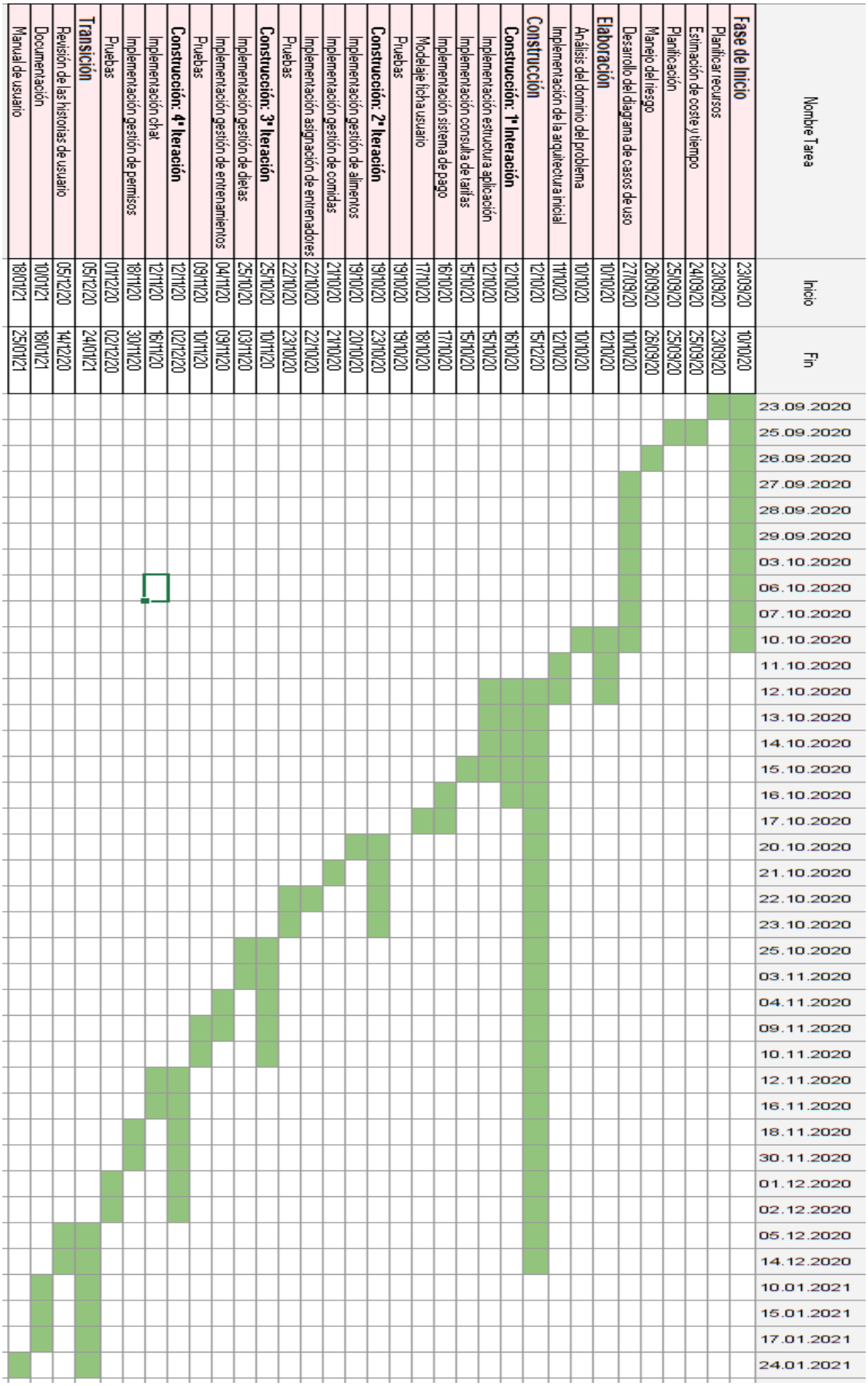

Ilustración 3. Diagrama Gantt de Seguimiento

5. Arquitectura

Django es un framework basado en la arquitectura MVC (Model View Controller). Existe una variación en relación a este paradigma como el MVT (Model View Template) que también se asocia a este marco de desarrollo.

En la Figura 3 se representan los 3 componentes de la arquitectura MVT:

1. Modelo: Se define la estructura lógica de nuestro programa. En el se definen las clases y ciertos métodos que serán manejados en la Vista.
2. View: Se manejan las instancias de las clases y se realizan las consultas a la BD con la finalidad de obtener los datos correspondientes a una petición del cliente.
3. Template: Se muestran los datos que son enviados a través de la vista.

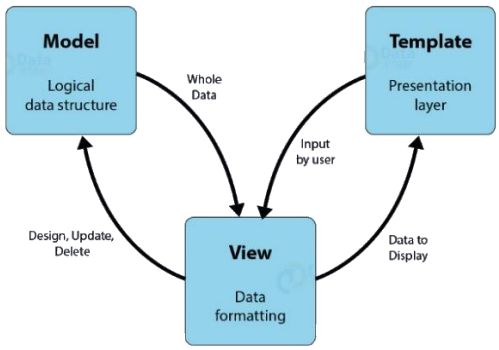

Ilustración 4. Componentes de la arquitectura MVT

Para entender de una mejor manera cómo funciona el flujo de datos entre los elementos que componen esta arquitectura, en la Figura 4 se muestra el diagrama de la arquitectura de Django.

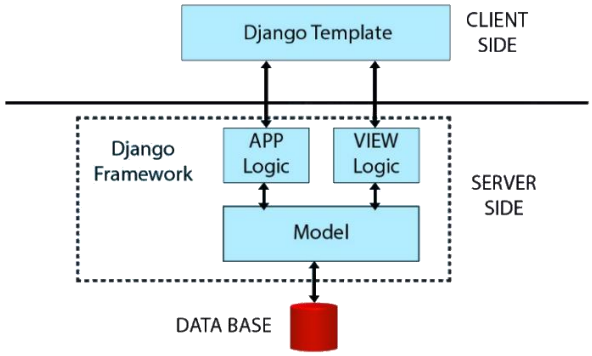

Ilustración 5. Arquitectura de Django: Servidor y Cliente

En la Figura 4, vemos que el framework maneja la lógica de la aplicación a través del modelo y la vista. El modelo es el encargado de realizar las consultas

a la BD; una vez que el modelo obtiene los datos requeridos por la petición, los envía a la vista. La vista es el componente encargado de manejarlos y hacer las operaciones que se requieran sobre los mismos. Finalmente, la solicitud realizada por el cliente se muestra a través del template.

Además, Django basa su operativa en el mapeo de rutas. Al crear un proyecto, el framework proporciona una estructura predefinida para tener una base con la que comenzar el desarrollo. Cuando deseamos renderizar una vista para mostrar los datos solicitados por el cliente, lo primero que hace el framework es comprobar que la URL solicitada está almacenada como ruta. Si lo está, utiliza los datos pasados como parámetro a la función que renderiza el template, y redirecciona al cliente a la página solicitada, mostrando los datos correspondientes.

5.1. Estructura del Proyecto

A continuación, se detallan los componentes Django que configuran la estructura del proyecto, proporcionando así un marco de desarrollo base para la codificación de sus funcionalidades.

En este proyecto, se ha seguido con el standard de desarrollo ideal propuesto por Django. El standard se basa en que el desarrollo de la aplicación tiene como resultado un conjunto de apps con interdependencia entre estas, donde cada una se caracteriza por proporcionar un conjunto de características. La ventaja de trabajar con aplicaciones es que estas pueden ser reusadas en proyectos diferentes ya que son estructuras de código independientes con total capacidad de integración en cualquier proyecto.

La estructura de aplicaciones o carpetas que define nuestro proyecto es la siguiente:

La estructura básica de componentes cada módulo creado es la siguiente:

1. Ficheros:
    a. admin.py: configuración del panel de administración de Django
    b. apps.py: configuración de cualquier característica de la aplicación
    c. models.py: definición de las clases y métodos que dan forma a la aplicación
    d. urls.py: fichero en el que se definen las rutas de la app.
    e. views.py: se definen las vistas que componen nuestra app.
2. Carpetas:
    a. migrations: cada versión de nuestra BD genera un nuevo archivo .py con la modificación realizada. En esta carpeta se almacenan las diferentes versiones de la BD.

b. templates: los archivos HTML que se renderizan a través de las vistas se almacenan en esta carpeta

Podríamos seguir definiendo más componentes en base al proyecto a realizar y el procedimiento de desarrollo utilizado. Todo depende del desarrollador y sus preferencias. Por ejemplo, en el desarrollo de esta aplicación se ha creado un archivo forms.py por cada componente de la aplicación que contiene la estructura de todos los formularios que se despliegan en la aplicación

A nivel general, la aplicación tiene más ficheros que almacenan la configuración general del proyecto e integran el marco de desarrollo. Los principales son:

1. settings.py: contiene toda la configuración de nuestra aplicación.
2. asgi.py: comunica el código con la aplicación servidor.
3. wsgi.py: funciona igual que asgi.py

5.2. Entorno Virtual

El uso del entorno virtual es fundamental en el desarrollo de cualquier proyecto en Python. En nuestro caso, nos ayuda a mantener las dependencias de un proyecto.

Si, por ejemplo, usamos Django 1.9 para un proyecto, y Django 3.1 para otro, existe un conflicto en nuestro sistema al no saber qué versión de Django utilizar. A partir de este momento, el uso de un entorno virtual es fundamental. Nos ayuda a diferenciar sobre qué versión de Django trabajar, consiguiendo aislar una versión respecto a la otra.

Una vez creado el entorno virtual, podremos instalar en el mismo las versiones de las dependencias que usaremos. De esta manera, no tendremos todas las dependencias instaladas en nuestro sistema. Será más que suficiente con tener la dependencia instalada en nuestro entorno virtual.

6. Tecnologías y productos de terceros

En esta sección, se enumeran las tecnologías que se han utilizado para el desarrollo del proyecto.

6.1. Tecnologías para la codificación del proyecto

1. Django: Framework desarrollado en Python basado en el paradigma MVT que proporciona el entorno y las funcionalidades necesarias para el desarrollo de la aplicación. En esta aplicación se hace uso de los módulos para facilitar la implementación de ciertas funcionalidades. Los más destacados son:

a) Django-ckeditor: proporciona una interfaz más funcional para ciertos campos de formulario
b) Django-crispy-forms: interfaz más amigable a nivel de formularios tanto en la parte de la aplicación como en la parte del cliente
c) Pillow: Hace posible la gestión de imágenes en el intérprete de Python
d) Six: Suaviza las diferencias entre Python 2 y Python 3+
e) Pylint: busca errores de código y sugiere cambios al programador

Estos son algunos de los módulos que se han utilizado. Los demás se encuentran en el Anexo I de esta documentación.

2. MySQL: Sistema de gestión de bases de datos para administrar los datos de la aplicación
3. JavaScript: Lenguaje de programación orientado a objetos, débilmente tipado y dinámico. Se usa para validar ciertos campos a nivel de formulario
4. JQuery: Biblioteca de JavaScript que permite simplificar las interacciones de los elementos del DOM.

6.2. Tecnologías para el desarrollo del proyecto

1. Windows 10: S.O sobre el que se ha desarrollado el proyecto
2. Google Chrome y Mozilla FireFox: Navegadores sobre los que se ha desplegado la aplicación
3. Microsoft Word: Elaboración de la documentación
4. Microsoft Excel: Elaboración de algunas de las tablas de la documentación
5. SmartSheet: Elaboración del Diagrama de Gantt
6. GitHub Desktop: Herramienta para registrar y notificar las versiones del proyecto
7. GitHub: Herramienta para almacenar el repositorio remoto que contiene el proyecto
8. Visual Paradigm 16.2: Desarrollo de los artefactos UML.
9. Stripe: Realización de pagos a través de la red

7. Especificación y Análisis de Requisitos

En este apartado se detallan los requisitos funcionales y no funcionales, así como los actores y casos de uso que interactúan en el sistema.

7.1. Actores

Hay 3 actores en el sistema:

- Cliente: Usuario que utiliza la aplicación de acuerdo a su objetivo.
- Administrador: Usuario que gestiona el funcionamiento de la aplicación corrigiendo cualquier incidencia
- Entrenador: Usuario que gestiona a los clientes que le han sido asignados. Es el encargado de crear las dietas y entrenamientos, así como ayudar al cliente en cualquier duda o problema que tenga.

7.2. Diagrama de Casos de Uso

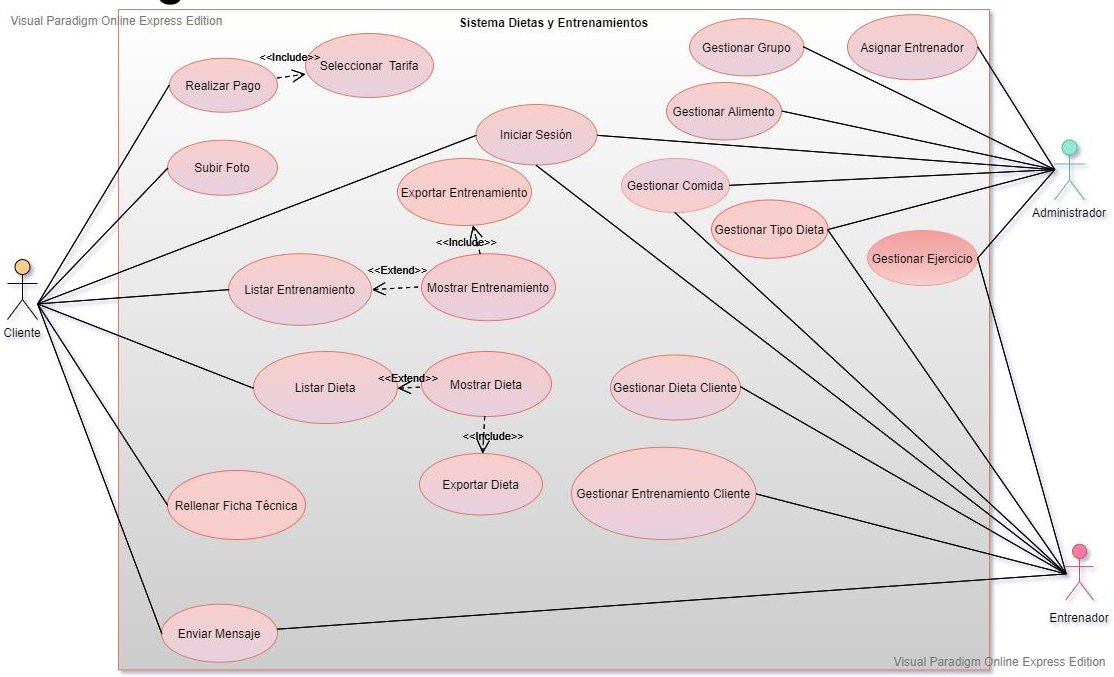

Ilustración 6. Diagrama de casos de uso

7.3. Descripción de los casos de uso

7.3.1. Caso de uso listar entrenamiento

Nombre	**Listar Entrenamiento**
Actor	Cliente
Precondiciones	El actor debe haber iniciado sesión y mantener vigente su suscripción de pago
Postcondiciones	La aplicación muestra la lista de entrenamientos asociados al actor
Flujo Principal	
Actor	**Sistema**
1. Accede al listado de entrenamientos	
	2. La aplicación muestra los entrenamientos del actor

Tabla 3. Descripción del caso de uso listar entrenamiento

7.3.2. Caso de uso listar dieta

Nombre	**Listar Dieta**
Actor	Cliente
Precondiciones	El actor debe haber iniciado sesión y mantener vigente su suscripción de pago
Postcondiciones	La aplicación muestra la dieta asignada al actor
Flujo Principal	
Actor	**Sistema**
1. Accede al listado de dietas	
	2. La aplicación muestra las dietas del actor

Tabla 4. Descripción del caso de uso listar dieta

7.3.3. Caso de uso mostrar entrenamiento

Nombre	**Mostrar Entrenamiento**
Actor	Cliente
Precondiciones	El actor debe haber iniciado sesión y mantener vigente su suscripción de pago
Postcondiciones	La aplicación muestra el entrenamiento en detalle
Flujo Principal	
Actor	**Sistema**
1. Accede al entrenamiento	
	2. La aplicación muestra el entrenamiento

Tabla 5. Descripción del caso de uso mostrar entrenamiento

7.3.4. Caso de uso mostrar dieta

Nombre	**Mostrar Dieta**
Actor	Cliente
Precondiciones	El actor debe haber iniciado sesión y mantener vigente su suscripción de pago
Postcondiciones	La aplicación muestra la dieta en detalle
Flujo Principal	
Actor	**Sistema**
1. Accede a la dieta	
	2. La aplicación muestra la dieta

Tabla 6. Descripción del caso de uso mostrar dieta

7.3.5. Caso de uso enviar mensaje

Nombre	**Enviar Mensaje**
Actor(es)	Cliente, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma y mantener vigente su suscripción de pago
Postcondiciones	La aplicación muestra un chat que permite la interacción entre los actores
Flujo Principal	
Actor	**Sistema**
1. Accede al chat	
	2. La aplicación despliega el chat
3. El actor origen selecciona al actor destino y le envía un mensaje	
	4. La aplicación envía el mensaje al actor destino
	5. La aplicación envía un e-mail de notificación al actor destino de que ha recibido un mensaje

Tabla 7. Descripción del caso de uso enviar mensaje

7.3.6. Caso de uso subir foto

Nombre	**Subir Foto**
Actor	Cliente
Precondiciones	El actor debe haber iniciado sesión y mantener vigente su suscripción de pago
Postcondiciones	La aplicación muestra las fotos del actor
Flujo Principal	
Actor	**Sistema**
1. Accede a la galería	
	2. La aplicación muestra las fotos subidas
3. El actor añade una nueva foto	
	4. La aplicación muestra la galería de imágenes con la nueva foto subida y un mensaje de confirmación
Flujo Alternativo [1]	
3.1. El actor selecciona un archivo con formato incorrecto	
	4.1 La aplicación muestra un mensaje de error y no sube el archivo

Tabla 8. Descripción del caso de uso subir foto

7.3.7. Rellenar ficha técnica

Nombre	**Rellenar Ficha Técnica**
Actor	Cliente
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Es conducido a la vista de tarifas
Flujo Principal	
Actor	**Sistema**
1. El usuario inicia sesión en la plataforma	
	2. La aplicación comprueba que no ha rellenado la ficha técnica de usuario
3. El actor rellena la ficha técnica y envía los datos	
	4. La aplicación muestra un mensaje de confirmación y las tarifas disponibles

Tabla 9. Descripción del caso de uso rellenar ficha técnica

7.3.8. Caso de uso realizar pago

Nombre	**Realizar Pago**
Actor	Cliente
Precondiciones	El actor debe haber iniciado sesión y completado la ficha técnica
Postcondiciones	La aplicación muestra el portal de inicio
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona una tarifa	
	2. La aplicación muestra las opciones de pago
3. El actor selecciona una opción	
	4. La aplicación redirecciona la petición a Stripe
	5. La aplicación informa que la transacción ha sido realizada con éxito y redirecciona al actor al portal de inicio
Flujo Alternativo [1]	
3.1. El usuario selecciona una opción	
	4.1. La aplicación muestra un mensaje de que no se ha podido completar la transacción y vuelve a mostrar las opciones de pago

Tabla 10. Descripción del caso de uso realizar pago

7.3.9. Caso de uso asignar entrenador

Nombre	**Asignar Entrenador**
Actor	Administrador
Precondiciones	Debe haber actores en el sistema
Postcondiciones	El actor administrador asigna al actor cliente un actor entrenador
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona al actor cliente	
	2. La aplicación muestra en detalle los datos del actor cliente
3. En actor realiza la asignación	
	4. La aplicación muestra un mensaje de confirmación

Tabla 11. Descripción del caso de uso asignar entrenador

7.3.10. Caso de uso asignar grupo usuario

Nombre	**Gestionar Usuarios Grupo**
Actor	Administrador
Precondiciones	Debe haber usuarios registrados en el sistema
Postcondiciones	El usuario pertenece a otro grupo
Flujo Principal	
Actor	**Sistema**
1. El administrador accede al usuario	
	2. La aplicación un formulario de edición con los datos de usuario
3. El administrador modifica al usuario asignándole un nuevo grupo	
	4. La aplicación procesa la petición y muestra un mensaje de confirmación

Tabla 12. Descripción del caso de uso asignar grupo usuario

7.3.11. Caso de uso gestionar alimento

7.3.11.1. Añadir alimento

Nombre	**Añadir Alimento**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Hay un nuevo alimento en el sistema
Flujo Principal	
Actor	**Sistema**

1. El actor selecciona la opción para agregar un nuevo alimento	
	2. La aplicación muestra la vista correspondiente
3. El actor añade el alimento	
	4. La aplicación muestra un mensaje de confirmación
Flujo Alternativo[1]	
3.1. El actor añade el alimento	
	4.1 La aplicación detecta error de formato y muestra un mensaje indicando dónde está el error
Flujo Alternativo[2]	
3.2. El actor añade el alimento	
	4.2 La aplicación detecta campos vacíos y muestra un mensaje indicando dónde está el error

Tabla 13. Descripción del caso de uso añadir alimento

7.3.11.2. Eliminar alimento

Nombre	**Eliminar Alimento**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	El alimento es eliminado del sistema
Flujo Principal	
Actor	**Sistema**
1. El actor elimina el alimento	
	2. La aplicación elimina el ítem seleccionado y muestra un mensaje de confirmación

Tabla 14. Descripción del caso de eliminar alimento

7.3.11.3. Modificar alimento

Nombre	**Editar Alimento**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	El alimento es modificado
Flujo Principal	
Actor	**Sistema**

1. El actor selecciona un alimento	
	2. La aplicación muestra la vista de edición
3. El actor modifica el alimento	
	4. La aplicación modifica el alimento y muestra un mensaje de confirmación
Flujo Alternativo[1]	
3.1. El actor modifica el alimento	
	4.1 La aplicación detecta error de formato y muestra un mensaje indicando dónde está el error
Flujo Alternativo[2]	
3.2. El actor modifica el alimento	
	4.1 La aplicación detecta campos vacíos y muestra un mensaje indicando dónde está el error

Tabla 15. Descripción del caso de uso modificar alimento

7.3.12. Caso de uso gestionar comida

7.3.12.1. Añadir comida

Nombre	**Añadir Comida**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Hay una nueva comida en el sistema
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona la opción para agregar una nueva comida	
	2. La aplicación muestra la vista correspondiente
3. El actor añade la comida	
	4. La aplicación muestra un mensaje de confirmación
Flujo Alternativo [1]	
3.1. El actor añade la comida	
	4.1 La aplicación detecta error de formato y muestra un mensaje indicando dónde está el error
Flujo Alternativo [2]	
3.2. El actor añade la comida	

	4.2 La aplicación detecta campos vacíos y muestra un mensaje indicando dónde está el error

Tabla 16. Descripción del caso de uso añadir comida

7.3.12.2. Eliminar comida

Nombre	**Eliminar Comida**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se elimina la comida
Flujo Principal	
Actor	**Sistema**
1. El actor elimina la comida	
	2. La aplicación muestra un mensaje de confirmación

Tabla 17. Descripción del caso de uso eliminar comida

7.3.12.3. Editar comida

Nombre	**Editar Comida**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se actualiza la comida
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona la opción modificar una comida	
	2. La aplicación muestra la vista correspondiente
3. El actor modifica la comida	
	4. La aplicación muestra un mensaje de confirmación
Flujo Alternativo [1]	
3.1. El actor modifica la comida	
	4.1 La aplicación detecta error de formato y muestra un mensaje indicando dónde está el error
Flujo Alternativo [2]	
3.2. El actor modifica la comida	

	4.2 La aplicación detecta campos vacíos y muestra un mensaje indicando dónde está el error

Tabla 18. Descripción del caso de uso editar comida

7.3.12.4. Consultar comida

Nombre	**Consultar Comida**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se muestra la comida en detalle
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona la comida	
	2. La aplicación muestra la vista en detalle de la comida

Tabla 19. Descripción del caso de uso consultar comida

7.3.13. Caso de uso gestionar dieta cliente

7.3.13.1. Añadir dieta

Nombre	**Añadir Dieta**
Actor(es)	Entrenador
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Dieta creada
Flujo Principal	
Actor	**Sistema**
1. El actor entrenador accede a la sección de dietas	
	2. La aplicación muestra los usuarios que le han sido asignados
3. El actor selecciona un usuario	
	4. La aplicación muestra un formulario con sus datos físicos y su tasa de metabolismo basal (método Harris Benedict)
5. El actor selecciona el índice de actividad física	
	6. La aplicación calcula la normocaloría del usuario y la muestra
7. El actor introduce el ajuste calórico (déficit o superávit) a realizar	
	8. La aplicación muestra las calorías necesarias para comenzar la dieta

9. El actor selecciona el siguiente paso	
	9. La aplicación muestra los tipos de dieta con la distribución por macronutrientes
10. El actor selecciona un tipo de dieta	
	11. La aplicación calcula la distribución de sus calorías por macronutriente y muestra el panel con la asignación
12. El actor envía la dieta	
	13. Muestra un mensaje de confirmación con la dieta creada
Flujo Alternativo [1]	
7.1. El actor introduce un formato incorrecto	
	8.1 La aplicación detecta error de formato y vuelve a solicitar el dato
Flujo Alternativo [2]	
7.2. El actor deja el campo de ajuste calórico vacío	
	8.2. La aplicación muestra un de "Campo Vacío"

Tabla 20. Descripción del caso de uso añadir dieta

7.3.13.2. Eliminar dieta

Nombre	**Eliminar Dieta**
Actor	Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se elimina la dieta
Flujo Principal	
Actor	**Sistema**
1. El actor elimina la dieta	
	2. La aplicación muestra un mensaje de confirmación

Tabla 21. Descripción del caso de uso eliminar dieta

7.3.13.3. Editar dieta

Nombre	**Editar Dieta**
Actor	Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se actualiza la dieta
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona la opción para modificar una dieta	

	2. La aplicación muestra la vista correspondiente
3. El actor modifica la dieta	
	4. La aplicación muestra un mensaje de confirmación
Flujo Alternativo [1]	
3.1. El actor modifica la dieta	
	4.1 La aplicación detecta que el intervalo de fecha es erróneo
Flujo Alternativo [2]	
3.2. El actor modifica la dieta	
	4.2 La aplicación detecta campos vacíos y muestra un mensaje indicando dónde está el error

Tabla 22. Descripción del caso de uso editar dieta

7.3.13.4. Añadir comida a dieta

Nombre	**Añadir Comida a Dieta**
Actor	Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se añade una comida a la dieta
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona la dieta	
	2. La aplicación muestra los días que abarca la dieta
3. El actor selecciona un día	
	4. La aplicación muestra una tabla con la información calórica del día
5. El actor selecciona un comida, indica sus gramos y añade unos comentarios	
	6. La aplicación añade la comida a la dieta y actualiza la tabla con la información calórica

Tabla 23. Descripción del caso de uso añadir comida a dieta

7.3.13.5. Añadir alimento a dieta

Nombre	**Añadir Alimento a Dieta**
Actor	Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se añade un alimento a la dieta
Flujo Principal	
Actor	**Sistema**

1. El actor selecciona la dieta	
	2. La aplicación muestra los días que abarca la dieta
3. El actor selecciona un día	
	4. La aplicación muestra una tabla con la información calórica del día
5. El actor selecciona un alimento, indica sus gramos y añade unos comentarios	
	6. La aplicación añade el alimento a la dieta y actualiza la tabla con la información calórica
Flujo Alternativo [1]	
3.1. El actor introduce un formato gramos incorrecto	
	4.1 La aplicación muestra un error de formato

Tabla 24. Descripción del caso de uso añadir alimento a dieta

7.3.14. Caso de uso exportar dieta

Nombre	**Exportar a PDF**
Actor	Cliente
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Se genera un archivo PDF con la dieta
Flujo Principal	
Actor	**Sistema**
1. El usuario selecciona la opción para exportar la dieta	
	2. La aplicación genera el PDF y muestra un mensaje de confirmación.

Tabla 25. Descripción del caso de uso exportar dieta

7.3.15. Caso de uso gestionar ejercicio

7.3.15.1. Añadir ejercicio

Nombre	**Añadir Ejercicio**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Hay un nuevo ejercicio en el sistema
Flujo Principal	
Actor	**Sistema**
1. El actor accede a la vista	
	2. La aplicación muestra la vista correspondiente
3. El actor añade el ejercicio	

	4. La aplicación muestra un mensaje de confirmación
Flujo Alternativo[1]	
3.1. El actor añade el ejercicio	
	4.1 La aplicación detecta error de formato y muestra un mensaje indicando dónde está el error
Flujo Alternativo[2]	
3.2. El actor añade el ejercicio	
	4.2 La aplicación detecta campos vacíos y muestra un mensaje indicando dónde está el error

Tabla 26. Descripción del caso de uso añadir ejercicio

7.3.15.2. Editar ejercicio

Nombre	**Editar Ejercicio**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se actualiza el ejercicio
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona el ejercicio	
	2. La aplicación muestra la vista correspondiente
3. El actor modifica el ejercicio	
	4. La aplicación muestra un mensaje de confirmación
Flujo Alternativo [1]	
3.1. El actor modifica el ejercicio	
	4.1 La aplicación detecta error de formato y muestra un mensaje indicando dónde está el error
Flujo Alternativo[2]	
3.2. El actor modifica el ejercicio	
	4.2 La aplicación detecta campos vacíos y muestra un mensaje indicando dónde está el error

Tabla 27. Descripción del caso de uso editar ejercicio

7.3.15.3. Eliminar ejercicio

Nombre	**Eliminar Ejercicio**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se elimina el ejercicio
Flujo Principal	
Actor	**Sistema**
1. El actor elimina el ejercicio	
	2. La aplicación muestra un mensaje de confirmación

Tabla 28. Descripción del caso de uso eliminar ejercicio

7.3.15.4. Consultar ejercicio

Nombre	**Consultar Ejercicio**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Se muestra el ejercicio en detalle
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona el ejercicio	
	2. La aplicación muestra la vista en detalle del ejercicio

Tabla 29. Descripción del caso de uso consultar ejercicio

7.3.16. Caso de uso gestionar entrenamiento cliente

7.3.16.1. Crear entrenamiento

Nombre	**Añadir Entrenamiento**
Actor(es)	Entrenador
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Entrenamiento Creado
Flujo Principal	
Actor	**Sistema**
1. El actor accede a la vista de los usuarios que tiene asignados	
	2. La aplicación muestra el listado de usuarios
3. El actor selecciona a un usuario	
	4.La aplicación muestra los entrenamientos de ese usuario
5. El actor accede a la vista de inserción, rellena el formulario y envía los datos	
	6. La aplicación muestra un mensaje de confirmación

Flujo Alternativo[1]	
5.1. El actor introduce un formato incorrecto	
	6.1 La aplicación detecta y muestra el error de formato y vuelve a solicitar los datos
Flujo Alternativo[2]	
5.2. El actor introduce un rango de fechas no válido	
	6.2 La aplicación detecta y muestra el error de fecha y vuelve a solicitar los datos

Tabla 30. Descripción del caso de uso crear entrenamiento

7.3.16.2. Editar entrenamiento

Nombre	**Editar Entrenamiento**
Actor(es)	Entrenador
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Entrenamiento Creado
Flujo Principal	
Actor	**Sistema**
1. El actor accede a la vista de los usuarios que tiene asignados	
	2. La aplicación muestra el listado de usuarios
3. El actor selecciona a un usuario	
	4. La aplicación muestra los entrenamientos de ese usuario
5. El actor accede a la vista de edición de un entrenamiento, modifica el formulario y envía los datos	
	6.La aplicación muestra un mensaje de confirmación
Flujo Alternativo[1]	
5.1. El actor introduce un formato incorrecto	
	6.1 La aplicación detecta y muestra el error de formato y vuelve a solicitar los datos
Flujo Alternativo[2]	
5.2. El actor introduce un rango de fechas no válido	
	6.2 La aplicación detecta y muestra el error de fecha y vuelve a solicitar los datos

Tabla 31. Descripción del caso de uso editar entrenamiento

7.3.16.3. Eliminar entrenamiento

Nombre	**Eliminar Entrenamiento**
Actor	Entrenador
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	El entrenamiento es eliminado
Flujo Principal	
Actor	**Sistema**
1. El actor accede a la vista de los usuarios que tiene asignados	
	2. La aplicación muestra los entrenamientos de ese usuario
3. El actor elimina un entrenamiento	
	4. La aplicación muestra un mensaje de confirmación

Tabla 32. Descripción del caso de uso eliminar entrenamiento

7.3.16.4. Añadir ejercicio a entrenamiento

Nombre	**Añadir ejercicio a entrenamiento**
Actor(es)	Entrenador
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Ejercicio agregado
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona en entrenamiento	
	2. La aplicación muestra el rango de fechas que abarca el entrenamiento
3. El actor selecciona una fecha	
	4. La aplicación muestra un formulario de inserción
5. El actor rellena el formulario y envía los datos	
	6.La aplicación muestra un mensaje de confirmación
Flujo Alternativo[1]	
5.1. El actor deja campos vacíos incorrecto	
	6.1 La aplicación detecta, muestra el error y vuelve a solicitar los datos

Tabla 33. Descripción del caso de uso añadir ejercicio a entrenamiento

7.3.16.5. Eliminar ejercicio de entrenamiento

Nombre	**Eliminar ejercicio de entrenamiento**
Actor(es)	Entrenador
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Ejercicio eliminado

Flujo Principal	
Actor	**Sistema**
1. El actor selecciona en entrenamiento	
	2. La aplicación muestra el rango de fechas que abarca el entrenamiento
3. El actor accede a la vista en detalle de una fecha	
	4. La aplicación muestra la tabla de entrenamientos para esa fecha en concreto
5. El actor elimina el ejercicio	
	6.La aplicación muestra un mensaje de confirmación

Tabla 34. Descripción del caso de uso eliminar ejercicio de entrenamiento

7.3.17. Caso de uso exportar entrenamiento

Nombre	**Exportar entrenamiento**
Actor	Cliente
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Se genera un archivo PDF
Flujo Principal	
Actor	**Sistema**
1. El usuario accede al entrenamiento y exporta el entrenamiento	
	2. La aplicación procesa la petición y genera el PDF

Tabla 35. Descripción del caso de uso exportar entrenamiento

7.3.18. Caso de uso seleccionar tarifa

Nombre	**Seleccionar Tarifa**
Actor	Cliente
Precondiciones	Debe haber iniciado sesión en la plataforma
Postcondiciones	Es conducido a la pasarela de pago
Flujo Principal	
Actor	**Sistema**
1. El usuario inicia sesión en la plataforma	
	2. La aplicación comprueba que sí ha rellenado la ficha técnica y si no tiene vigencia en pagos ya pasados
3. El usuario selecciona la tarifa	
	4. La aplicación muestra los métodos de pago

Tabla 36. Descripción del caso de uso seleccionar tarifa

7.3.19. Caso de uso gestionar tipo dieta

7.3.19.1. Añadir tipo dieta

Nombre	**Añadir Tipo Dieta**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	Hay un nuevo tipo de dieta en el sistema con su respectiva distribución de macronutrientes
Flujo Principal	
Actor	**Sistema**
1. El actor selecciona accede a la vista de inserción	
	2. La aplicación muestra la vista
3. El actor rellena la vista y envía los datos	
	4. La aplicación añade el tipo de dieta y muestra un mensaje de confirmación
Flujo Alternativo [1]	
3.1. El actor rellena la vista y envía los datos	
	4.1 La aplicación detecta error de formato y muestra un mensaje de error
Flujo Alternativo [2]	
3.2. El actor rellena la vista y envía los datos	
	4.2 La aplicación detecta campos vacíos y muestra un mensaje de error

Tabla 37. Descripción del caso de uso añadir tipo dieta

7.3.19.2. Eliminar tipo dieta

Nombre	**Eliminar Tipo Dieta**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	El tipo de dieta es eliminada del sistema
Flujo Principal	
Actor	**Sistema**
1. El actor elimina el tipo de dieta	
	2. La aplicación elimina el tipo de dieta y muestra un mensaje de confirmación

Tabla 38. Descripción del caso de uso eliminar tipo dieta

7.3.19.3. Editar tipo dieta

Nombre	**Editar Tipo Dieta**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	El tipo de dieta se actualiza
Flujo Principal	
Actor	**Sistema**
1. El actor accede a la vista de edición	
	2. La aplicación muestra la vista de edición
3. El actor modifica la vista y envía los datos	
	4. La aplicación modifica el tipo de dieta y muestra un mensaje de confirmación
Flujo Alternativo [1]	
3.1. El actor modifica la vista y envía los datos	
	4.1 La aplicación detecta error de formato y muestra el mensaje de error correspondiente
Flujo Alternativo [2]	
3.2. El actor modifica la vista y envía los datos	
	4.1 La aplicación detecta campos vacíos y muestra el mensaje de error correspondiente

Tabla 39. Descripción del caso de uso editar tipo dieta

7.3.19.4. Consultar tipo dieta

Nombre	**Eliminar Tipo Dieta**
Actor	Administrador, Entrenador
Precondiciones	El actor debe haber iniciado sesión en la plataforma
Postcondiciones	El tipo de dieta es mostrada en detalle
Flujo Principal	
Actor	**Sistema**
1. El actor accede a la dieta	
	2. La aplicación muestra la dieta en detalle

Tabla 40. Descripción del caso de uso consultar tipo dieta

8. Diseño del Software

En relación con el diseño del software, destacamos dos perspectivas: diseño estático de la aplicación y diseño dinámico

8.1. Diseño estático

La siguiente ilustración de Data Flair sobre la configuración inicial que proporciona Django nos va a ayudar a entender el framework:

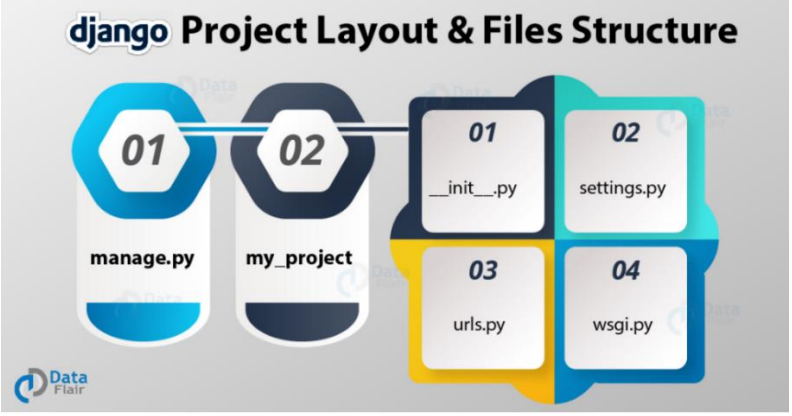

Ilustración 7. Diseño del proyecto y estructura de archivos

Cuando creamos el proyecto, Django nos proporciona automáticamente la estructura de directorios representada en la figura. Los archivos representados en la figura cumplen las siguientes funciones:

1. *manage.py*: Su función principal es poner en marcha en servidor. Además, es el encargado de migrar y controlar el proyecto a través de la línea de comandos
2. *my_project*: El el nombre que le damos al proyecto cuando lo creamos mediante el comando *python django-admin startproject* .

   Contiene los siguientes archivos:

   a) ***__init__.py***: Es un archivo sin contenido que le indica al intérprete de Python que este directorio es un paquete.
   b) ***settings.py***: Es el archivo principal del proyecto. Se indica lo siguiente: las aplicaciones creadas, las preferencias del desarrollador y la información middleware instalada en el proyecto, entre otras.
   c) ***urls.py***: Almacena las rutas del proyecto. En nuestro caso, se ha optado por crear un archivo urls por cada app creada. Este archivo almacena una referencia a cada archivo urls creado, almacenando así, las urls de cada app.
   d) ***wsgi.py***: Django utiliza el servidor WSGI para el desarrollo web. No es un archivo muy usado en el desarrollo del proyecto.

En definitiva, estos archivos son los que se crean por defecto cuando iniciamos cualquier proyecto en Django. Su función principal es proporcionar el backend de la aplicación y resolver los problemas de conectividad.

8.2. Diseño dinámico

8.2.1. Diagrama de clases

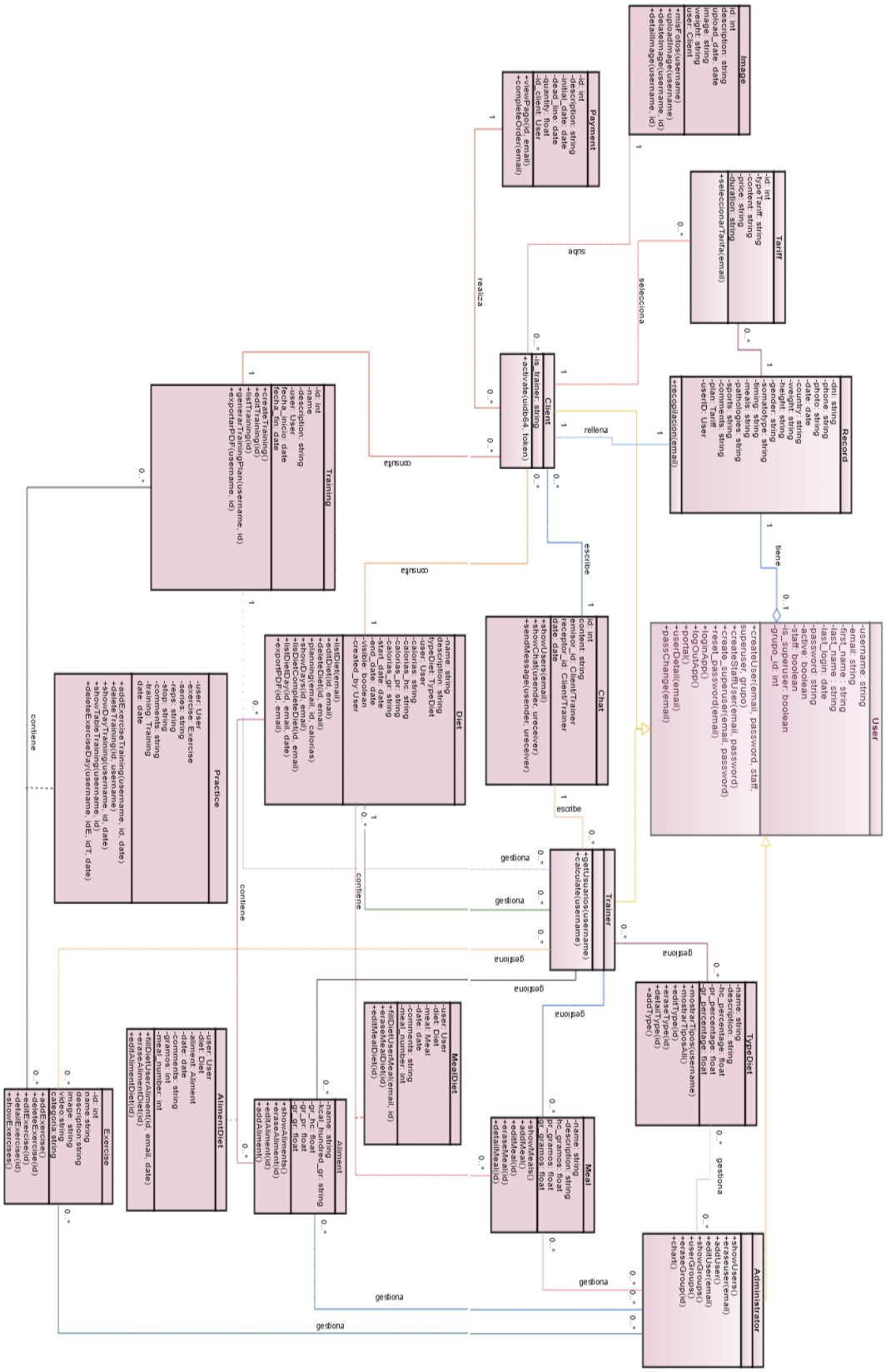

Ilustración 8. Diagrama de clases

8.2.2. Diagramas de secuencia del sistema

A continuación, se muestran los diagramas de secuencia más relevantes del sistema:[1]

8.2.2.1. Añadir ejercicio a entrenamiento

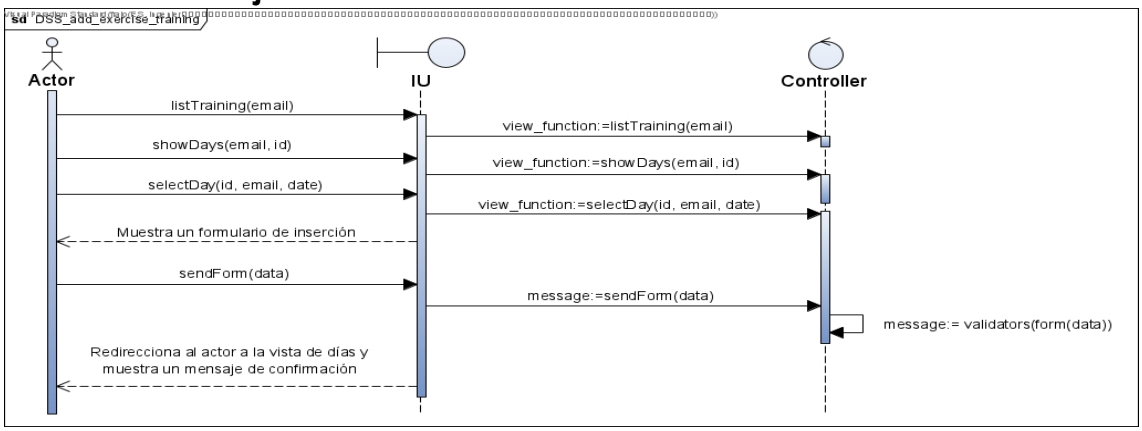

Ilustración 9. DSS añadir ejercicio a entrenamiento

8.2.2.2 Añadir alimento a dieta

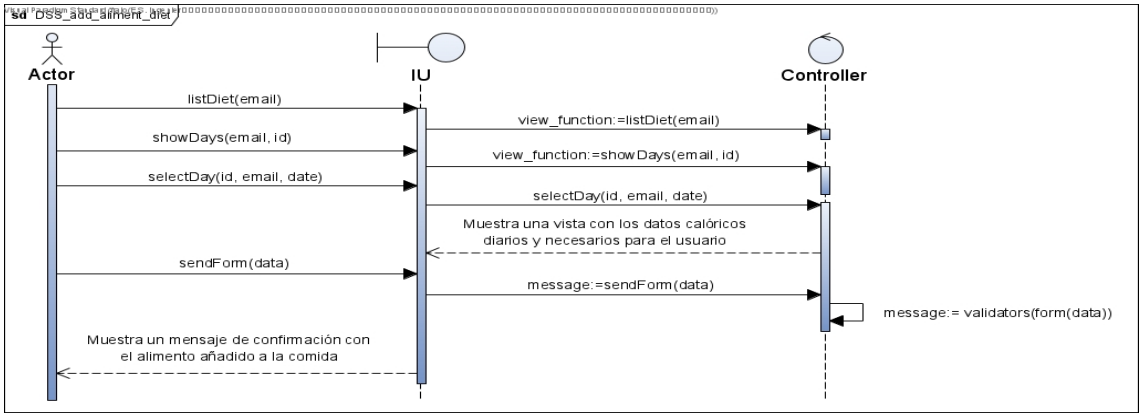

Ilustración 10. DSS Añadir alimento a dieta

8.2.2.3. Editar dieta

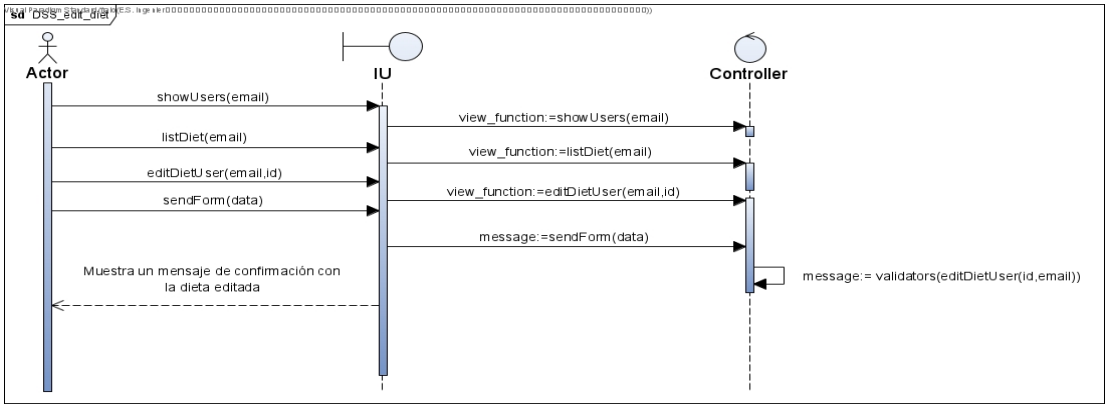

Ilustración 11. DSS editar dieta

[1] Consultar el ANEXO para ver el resto de los diagramas de secuencia.

8.2.2.4. Añadir comida a dieta

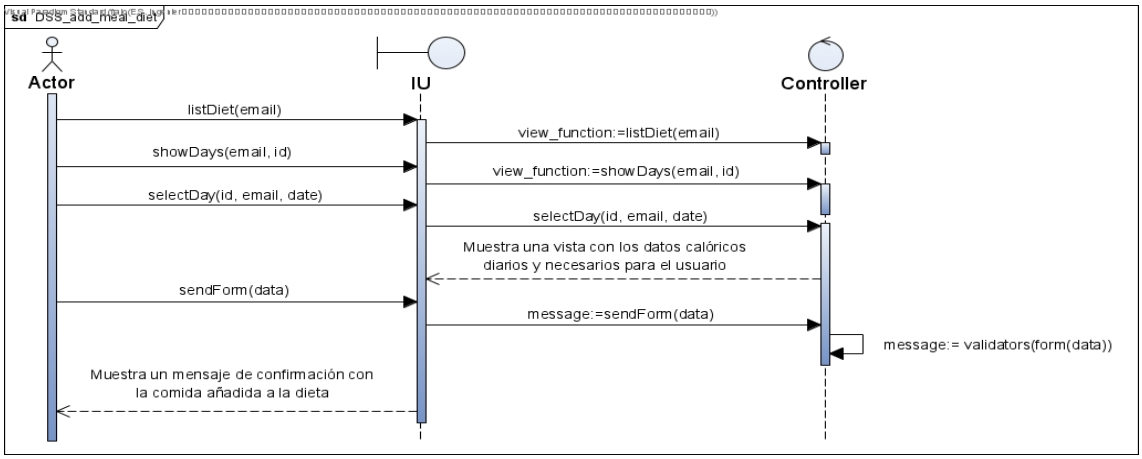

Ilustración 12. DSS añadir comida a dieta

8.2.2.5. Crear entrenamiento

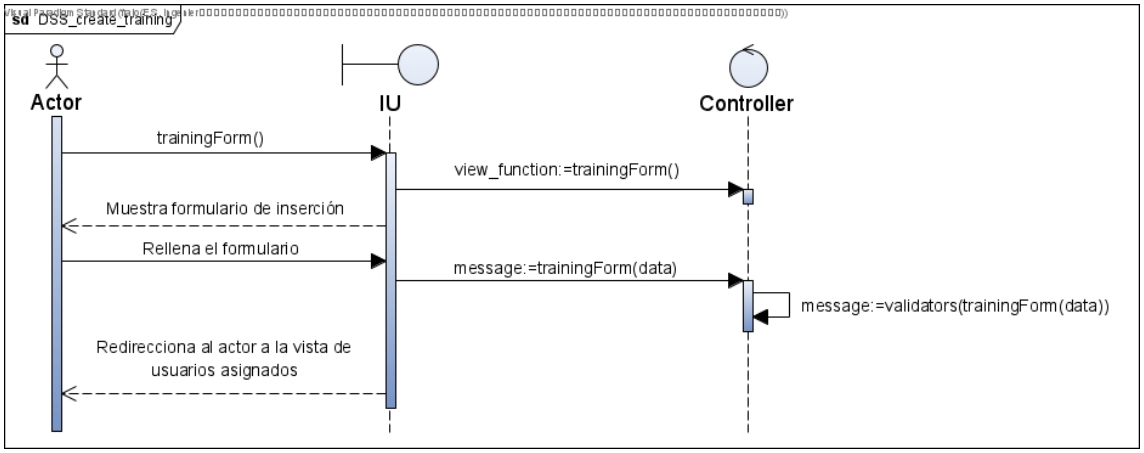

Ilustración 13. DSS crear entrenamiento

8.2.2.6. Crear dieta

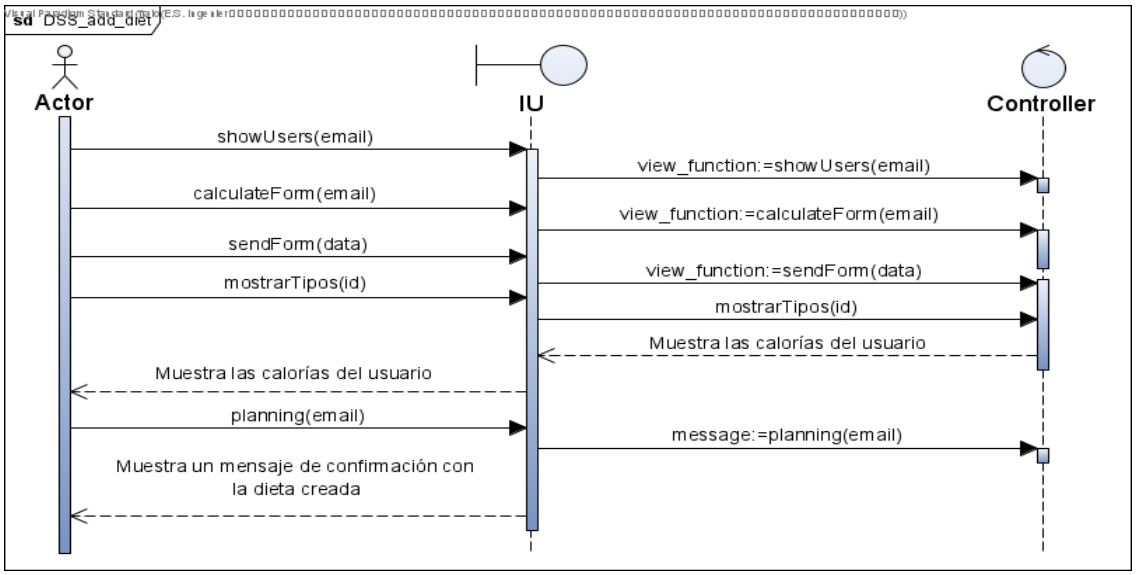

Ilustración 14. DSS crear dieta

9. Gestión de datos e información

En esta sección, se detalla la información que almacena el sistema y la estructura que contiene la información

9.1. Modelo Entidad Relación Lógico

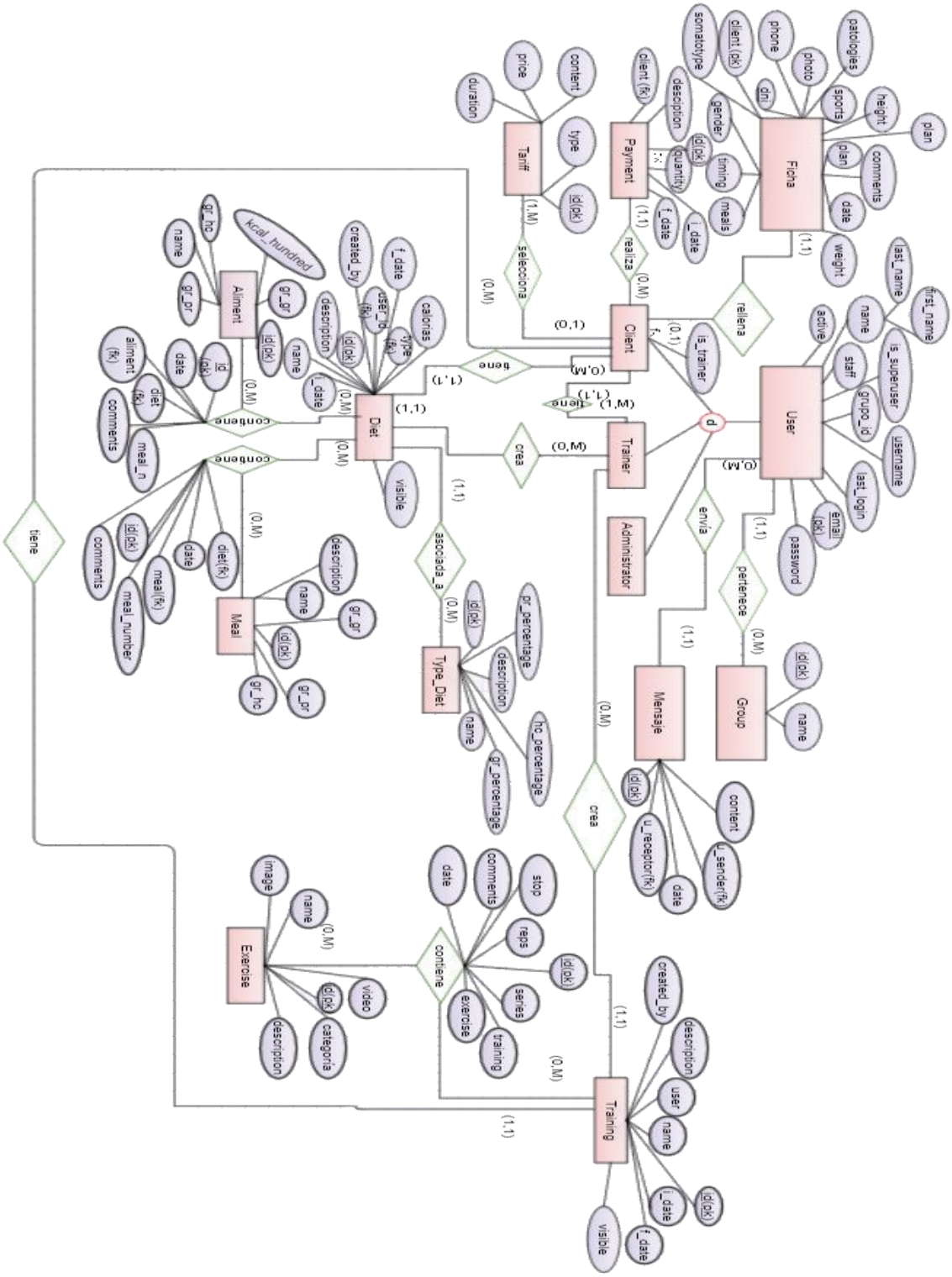

Ilustración 15. MERE lógico

9.2. Datos MERE

9.2.1. Entidad User

Representa los usuarios almacenados en el sistema. Los datos que guarda esta entidad son:

- email: código identificador
- username: login de usuario (único)
- password: contraseña encriptada del usuario
- first_name: nombre del usuario
- last_name: apellido usuario
- active: el usuario es o no activo
- superuser: el usuario es o no superusario
- staff: el usuario e o no staff
- last_login: fecha y hora de la última conexión del usuario
- grupo: grupo al que pertenece el usuario
- trainer: entrenador asociado al usuario cliente

9.2.2. Entidad Record

Representa tanto los datos personales del usuario como los físicos. Los datos que guarda esta entidad son:

- dni: único para cada usuario
- pone: número de teléfono del usuario
- photo: foto de perfil del usuario
- date: fecha de nacimiento del usuario
- country: residencia del usuario
- weight: peso en kg del usuario
- height: altura en cm del usuario
- somatotype: somatotipo del usuario
- timing: horarios de interés del usuario (para cuadrar número de comidas por día)
- meals: número de comidas que desea consumir el usuario
- sports: deportes que son de interés para el usuario
- patologies: patologías del usuario que puedan ser de interés desde un punto de vista físico
- comments: comentarios adicionales que quiera añadir el usuario
- userID: identificador que muestra a quien pertenece el registro

9.2.3. Entidad Payment

Almacena los pagos que ha realizado cada usuario dentro del sistema. Los datos que guarda son:

- id: identificador del pago
- description: almacena la tarifa seleccionada
- i_date: fecha de inicio de la suscripción
- f_date: fecha fin de la suscripción
- quantity: cantidad que se ha pagado

- client: cliente asociado al pago

9.2.4. Entidad Tariff

Almacena las tarifas que ofrece el sistema al cliente. Los datos que guarda son:

- id: identificador de la tarifa
- type: tipo de tarifa
- content: información adicional de la tarifa
- Price: precio de la tarifa
- duration: duración en meses de la tarifa

9.2.5. Entidad Diet

Almacena los registros de todas las dietas creadas. Los datos que guarda son:

- id: identificador de la dieta
- description: descripción de la dieta
- i_date: fecha de inicio de la dieta
- f_date: fecha de fin de la dieta
- name: nombre de la dieta
- user: usuario asociado a esa dieta
- created_by: entrenador que ha creado la dieta
- type: tipo de dieta asociada a la dieta creada
- visible: indica si la dieta está visible al usuario final

9.2.6. Entidad Type_Diet

Almacena los tipos de dieta que ofrece el sistema. Los datos que guarda son los siguientes:

- id: identificador del tipo de dieta
- name: nombre del tipo de dieta
- description: descripción asociada a la dieta
- hc_percentage: % de carbohidratos que caracteriza a este tipo de dieta
- pr_percentage: % de proteínas que caracteriza a este tipo de dieta
- gr_percentage: % de grasas que caracteriza a este tipo de dieta

9.2.7. Entidad Meal

Almacena los registros correspondientes a las comidas dadas de alta en el sistema. Los datos que guarda son los siguientes:

- id: identificador de cada comida
- name: nombre de la comida
- description: descripción de la comida
- gr_hc: gramos de carbohidratos de la comida
- gr_pr: gramos de proteínas de la comida
- gr_gr: gramos de grasas de la comida

9.2.8. Entidad Aliment

Almacena los registros correspondientes a los alimentos dados de alta en el sistema. Los datos que guarda son los siguientes:

- id: identificador del alimento
- name: nombre del alimento
- description: descripción del alimento
- gr_hc: gramos de carbohidratos del alimento
- gr_pr: gramos de proteínas del alimento
- gr_gr: gramos de grasas del alimento

9.2.9. Entidad Meal_Diet

Almacena los registros de las comidas asociadas a una dieta. Los datos que guarda son los siguientes:

- id: identificador de la comida añadida a la dieta
- diet: dieta asociada a la comida añadida
- meal: identificador de la comida
- date: fecha de la comida
- meal_number: número de comida
- comments: comentarios adicionales sobre la comida añadida

9.2.10. Entidad Aliment_Diet

Almacena los registros de los alimentos asociados a una dieta. Los datos que guarda son los siguientes:

- id: identificador del alimento añadido a la dieta
- diet: dieta asociada al alimento añadido
- aliment: identificador del alimento
- date: fecha de la comida
- meal_number: número de comida
- comments: comentarios adicionales sobre el alimento añadido

9.2.11. Entidad Training

- id: identificador del entrenamiento
- description: descripción del entrenamiento
- i_date: fecha de inicio del entrenamiento
- f_date: fecha de fin del entrenamiento
- name: nombre del entrenamiento
- user: usuario asociado a ese entrenamiento
- created_by: entrenador que ha creado el entrenamiento
- visible: indica si el entrenamiento está visible al usuario final

9.2.12. Entidad Exercise

- id: identificador del ejercicio
- decription: descripción del ejercicio
- image: imagen del entrenamiento

- video: video del entrenamiento
- categoría: categoría a la que pertenece el ejercicio

9.2.13. Entidad Training_Exercise

- id: identificador del ejercicio asociado al entrenamiento
- exercise: ejercicio asociado al entrenamiento
- training: entrenamiento que contiene el ejercicio registrado
- series: series del ejercicio
- reps: repeticiones por serie
- date: fecha del ejercicio para el entrenamiento registrado
- stop: descanso entre series
- comments: comentarios adicionales

9.2.14. Entidad Message

- id: identificador del mensaje
- content: contenido del mensaje
- date: fecha de envío del mesaje
- usender: usuario emisor del mensaje
- ureceiver: usuario receptor del mensaje

9.2.15. Entidad Group

- id: identificador del grupo
- name: nombre del grupo

10. Pruebas

El desarrollo de un caso de uso tiene que proporcionar un beneficio concreto al usuario. Por ello, se ha optado por desarrollar un apartado de pruebas para verificar que cada caso de uso cumple con las restricciones establecidas. A continuación, se detallan las pruebas realizadas:

10.1. Gestionar tipo de dieta

Precondiciones	Se introduce un formato incorrecto en los campos que almacenan la distribución de macronutrientes
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error sobre los campos erróneos
Validación	Prueba correcta

Tabla 41. Pruebas gestionar tipo de dieta

10.2. Gestionar alimento

Precondiciones	Se introduce un formato incorrecto en los campos que almacenan los gramos de macronutrientes
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario

Resultado	Muestra el error sobre los campos erróneos
Validación	Prueba correcta

Tabla 42. Pruebas gestionar alimento

10.3. Gestiona comida

Precondiciones	Se introduce un formato incorrecto en los campos que almacenan la distribución de macronutrientes
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error sobre los campos erróneos
Validación	Prueba correcta

Tabla 43. Pruebas gestionar comida

10.4. Gestionar dieta cliente

Precondiciones	Se introduce un intervalo de fechas erróneo o desfasado
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error sobre los campos erróneos
Validación	Prueba correcta

Tabla 44. Pruebas gestionar dieta cliente

10.5. Gestionar entrenamiento cliente

Precondiciones	Se introduce un intervalo de fechas erróneo o desfasado
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error sobre los campos erróneos
Validación	Prueba correcta

Tabla 45. Pruebas gestionar entrenamiento cliente

10.6. Rellenar ficha técnica

Precondiciones	Se introduce DNI erróneo
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error el campo DNI
Validación	Prueba correcta

Precondiciones	Se introduce teléfono erróneo
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error sobre el campo teléfono
Validación	Prueba correcta

Precondiciones	Se introduce nombre de usuario ya existente
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error de usuario ya existe
Validación	Prueba correcta

Precondiciones	Se introduce un archivo que no es una imagen
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error sobre el campo que solicita la imagen
Validación	Prueba correcta

Tabla 46. Pruebas rellenar ficha técnica

10.7. Crear ejercicio

Precondiciones	Se introduce un archivo que no es una imagen
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error sobre el campo que solicita la imagen
Validación	Prueba correcta

Tabla 47. Pruebas crear ejercicio

10.9. Subir foto

Precondiciones	Se introduce un archivo que no es una imagen
Ejecución	El módulo Validators detecta el error y redirecciona al usuario de nuevo al formulario
Resultado	Muestra el error sobre el campo que solicita la imagen
Validación	Prueba correcta

Tabla 48. Pruebas subir foto

11. Manual de usuario

A continuación, se exponen tanto el manual para desarrolladores como para clientes. En particular, el manual de desarrollador está pensado para un sistema operativo Windows.

11.1. Manual para desarrolladores

Para hacer uso de la aplicación el desarrollador debe configurar su proyecto como se indica en los siguientes apartados. El proyecto está contenido en el fichero *fitness_project* desde el cual se ejecutarán los comandos necesarios.

11.1.1. Creación base de datos y volcado

Modificar la directiva DATABASES situada en el archivo fitness_project>fitness_project>settings.py[2] especificando USER, PASSWORD, HOST Y PORT correspondientes para poder establecer la conexión con la BD.

Volcar el archivo DB.sql para la creación de la base de datos y el volcado de datos.[3]

11.1.2. Creación, activación y uso del entorno virtual

1. Debemos tener Python 3.9 en nuestro sistema y la dependencia virtualenv. Para instalar la dependencia, ejecutamos el siguiente comando: *pip install virtualenv*
2. Dentro del directorio *>fitness_project* ejecutamos lo siguiente: *python -m venv env*. De esta manera, creamos el entorno virtual.
3. Para activar el entorno ejecutamos desde el directorio raíz *>fitness_project*: *"env/Scripts/activate"* entre comillas dobles. Como resultado veremos al comienzo del Shell de Windows la directiva (env) indicativo de que el entorno está activado.
4. Ahora es necesario instalar las dependencias almacenadas en *fitness_project>requirements.txt*. Antes de indicar como hacerlo, el usuario debe crear una carpeta en su disco principal llamada "paquetes" (C:\\paquetes) y almacenar la dependencia *Twisted-20.3.0-cp39-cp39-win_amd64.whl* que está contenida en el directorio raíz del proyecto[4]
5. Para instalar las dependencias en nuestro entorno virtual hacemos: *pip install -r requirements.txt*.
6. WeasyPrint es un motor de renderizado para HTML y CSS que se instala juntamente con las demás dependencias contenidas en el archivo *requirements.txt*. Dicho esto, hace uso de una biblioteca de

[2] Línea 107 a 119

[3] Se sugiere utilizar la herramienta MySQLWorkbench. Se deja en manos del desarrollador la configuración de usuario que utilizará.

[4] Una vez instaladas las dependencias en el entorno virtual, ya no hace falta la carpeta "paquetes". El archivo requirements.txt está escrito para encontrar el archivo. whl en C:\.

componentes gráficos llamada GTK. Para su instalación, seguir el paso 4 de la siguiente documentación: https://weasyprint.readthedocs.io/en/stable/install.html#step-4-install-the-gtk-libraries

7. Una vez finalizada la instalación, escribimos desde el mismo directorio raíz (>fitness_project) la siguiente instrucción: *python manage.py runserver*.[5][6]
8. El servidor ya está operativo.

11.1.3. Creación de la cuenta de Stripe y almacenamiento de tarifas

Para configurar Stripe es necesario tener una cuenta en la plataforma. Una vez hayamos completado el registro, debemos hacer lo siguiente:

1. En el menú lateral izquierdo del Dashboard, ir a la pestaña "Desarrolladores" y acceder a "claves API". A continuación, se muestran dos claves que se copiarán y pegarán dentro del archivo *settings.py* situado en fitness_project>fitness_project. La clave pública y privada se copian en las directivas STRIPE_PUBLIC_KEY y STRIPE_PRIVATE_KEY, respectivamente.[7]
2. A continuación, debemos crear las tarifas. Para ello, accedemos a la pestaña "Productos". Para añadir una tarifa, accedemos a "añadir producto" e indicamos nombre, tipo de tarifa[8] y cuantía. Debemos crear 3 productos cuyos nombres serán Trimestral, Semestral y Anual. Y sus cuantías serán 479.88, 299.94 y 179.97, respectivamente.
3. Cada tarifa creada tiene un ID de API. Este ID de API debe ir en el campo *typeTariff* sustituyendo la directiva *API_tarifa* almacenada como dato en el campo *typeTariff* de la tabla user_tariff[9]. Este ID de API se debe almacenar de forma que la tupla que representa la tarifa anual se corresponda con el ID de API de la tarifa anual creada en la cuenta de Stripe y así sucesivamente para el resto de las tarifas creadas.

11.1.4. Ajustes finales

Debemos hacer unas configuraciones finales en el archivo settings.py situado en fitness_project>fitness_project para que la aplicación funcione:

1. En la línea 29, modificar la directiva DEBUG de True a False.

[5] Antes de ejecutar la aplicación, se sugiere comprobar que los datos de conexión con la BD sean los correctos.
[6] No ejecutar dicha instrucción cuando la directiva DEBUG del archivo Settings.py está a False (sistema en producción). Tras ejecutar el comando, se puede cambiar su valor a False.
[7] Líneas 176 y 177 del archivo settings.py.
[8] Pago único.
[9] Implica una modificación de la tupla en cuestión.

2. Modificar las directivas EMAIL_HOST_USER e EMAIL_HOST_PASSWORD de las líneas 168 y 169. En la primera se solicita el email emisor de los correos y, en la segunda, la contraseña del email emisor.[10]

11.1.4. Desactivación del entorno virtual

1. CRTL-BREAK para detener el servidor
2. Para desactivar el entorno virtual escribimos desde el directorio raíz (fitness_project>) la directiva entre comillas dobles: "env/Scripts/deactivate".

11.2. Manual para clientes

11.2.1. Vista administrador

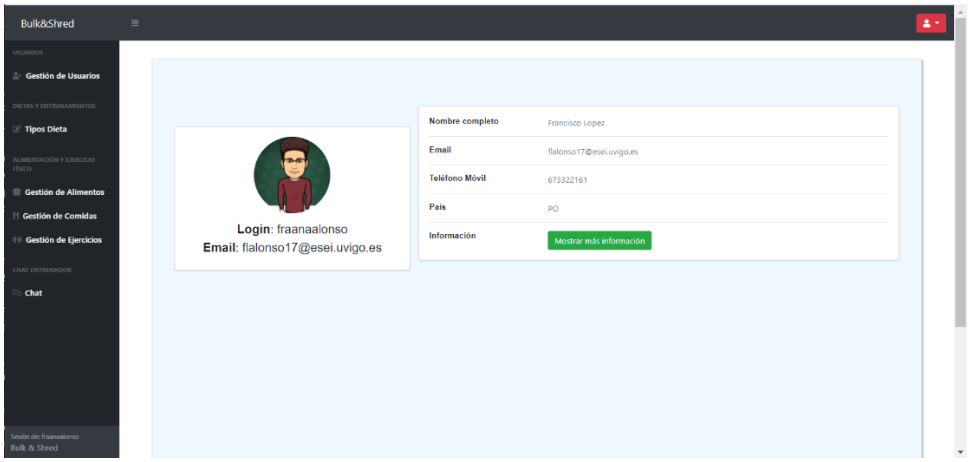

Ilustración 16. Vista administrador

11.2.1. Vista cliente

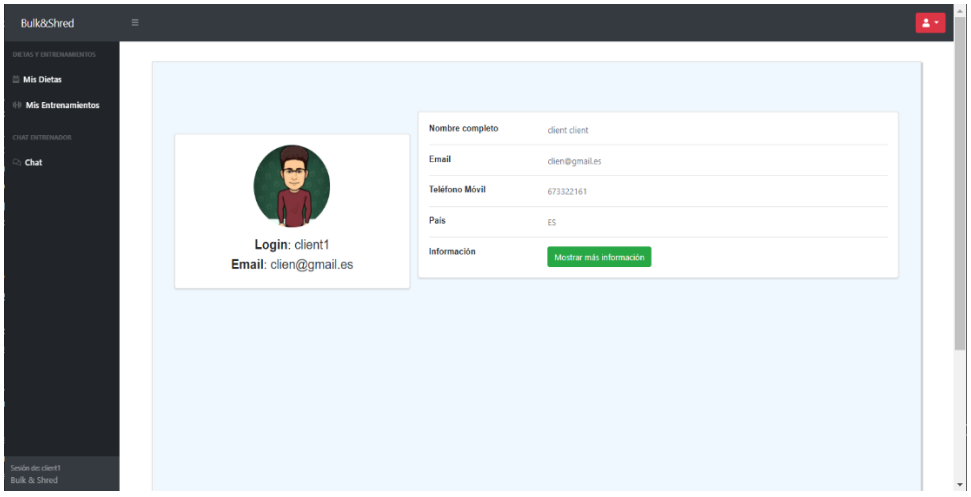

Ilustración 17. Vista cliente

[10] La cuenta Gmail emisora de los correos, debe permitir el acceso a terceros.

11.2.3. Vista entrenador

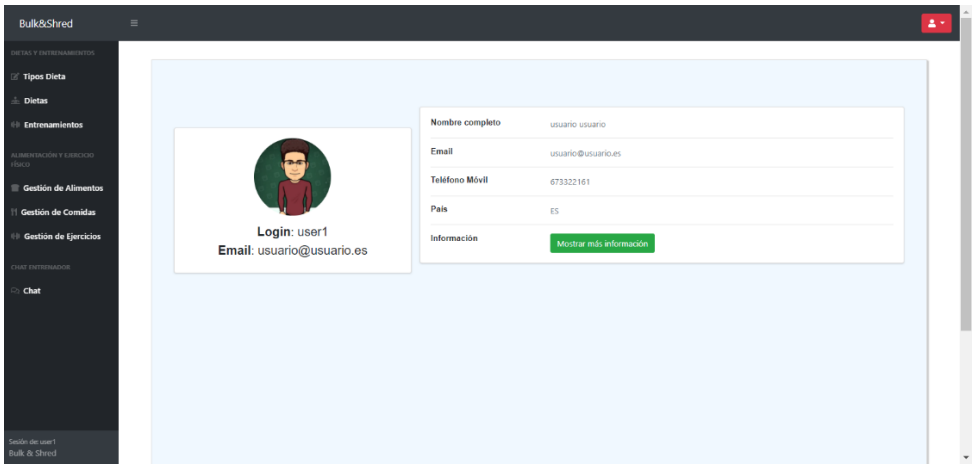

Ilustración 18. Vista entrenador

12. Principales aportaciones

La aportación principal de este proyecto es proporcionar al usuario un mecanismo para automatizar el proceso de creación de dietas y entrenamientos. Por una parte, en lo que se refiere a la gestión de dietas, al entrenador se le facilita el cálculo de los macronutrientes necesarios para un usuario; el proceso es intuitivo. Y por otra, el cliente puede disponer de una dieta desarrollada en profundidad (gramos por alimento a consumir y descripción por comida y alimentos añadidos) y que se ajusta a sus necesidades metabólicas.

En lo que se refiere a entrenamientos, el usuario tiene detallado lo que debe hacer en cada ejercicio. Es decir, nombre del ejercicio, series, repeticiones y descanso entre series.

13. Conclusiones

13.1. Conclusiones técnicas

Django es un marco de desarrollo que permite al programador centrarse en la codificación y desarrollo de los casos de uso sin tratar aspectos claves para una aplicación como pueda ser la seguridad. En el caso de Django, para evitar ataques como puedan ser el SQL Injection o el Cross Site Forgery, utiliza directivas que están destinadas para ese propósito. Hacer uso de esas directivas, garantiza que este tipo de ataques no pongan en peligro el sistema desarrollado por el usuario. Técnicamente, me parece un punto muy positivo ya que el programador puede centrarse en desarrollar un código más legible, eficiente y fácil de entender.

Un punto negativo que veo a nivel técnico es la capacidad de respuesta cuando da servicio a muchos clientes. Aplicaciones como Instagram que empezaron a dar su servicio con base en Django, tuvieron que migrar su entorno a otros lenguajes porque Django no era capaz de soportar múltiples peticiones. En el caso de Instagram, cuando ya se hizo popular en varios estados de los EE.UU, el cambio fue necesario.

13.2. Conclusiones personales

A nivel personal, estoy muy satisfecho con el trabajo realizado. Cuando he empezado el proyecto no conocía Django y a medida que iba desarrollando la aplicación iba descubriendo algunas de las facilidades que proporcionaba. Creo que ir aprendiendo sobre el framework a medida que desarrollaba el proyecto ha sido muy revelador.

El hecho de no conocer bien el framework, me ha obligado a explorar otras tecnologías que podían resolver los problemas que me iban surgiendo. Desde el comienzo, este proceso requería investigación y, en consecuencia, he mejorado mis conocimientos en lenguajes como JQuery o JavaScript.

14. Vías de trabajo futuras

14.1. Mejoras a nivel de contenido

1. Posibilidad de incluir recetas en las dietas. Esto es, que el entrenador pueda crear comidas proporcionando los requerimientos para hacerlo.
2. Incluir un seguimiento gráfico del peso en función de las fotos subidas. Es decir, en cada foto del usuario subida debe indicar su peso. Si existe más de 1, es viable hacer un tracking de la evolución del peso y analizar si está cumpliendo los objetivos
3. Mejorar el front-end de la aplicación incluyendo fotos de las comidas.
4. Crear comidas a partir de los alimentos añadidos.

14.2. Mejoras a nivel técnico

1. Hacer la aplicación más dinámica. En lugar de tener que crear un HTML para editar una instancia, sería interesante hacer las ediciones desde la propia tabla.
2. Desarrollar los formularios sólo con directivas de Django. Los campos hacen uso de la clase ModelForm o Form, según sea el caso. No obstante, el formulario entero puede hacer uso de las funcionalidades de Django que, entre la cuales, destacan sus directivas HTMl y CSS para estilizarlo desde la propia clase del formula.

15.Bibliografía

[1] *Django documentation | Django documentation | Django*. (2020). Django documentation. https://docs.djangoproject.com/en/3.1/

[2] JS Foundation - js.foundation. (2020). *jQuery API Documentation*. jQuery API Documentation. https://api.jquery.com/

[3] *JavaScript | MDN*. (2020, 23 noviembre). JavaScript MDN Web DOCS. https://developer.mozilla.org/es/docs/Web/JavaScript

[4] Otto, M. J. T. (2020). *Bootstrap*. Bootstrap. https://getbootstrap.com/

[5] *Managing Application Dependencies — Python Packaging User Guide*. (2020). Managing aplication dependencies. https://packaging.python.org/tutorials/managing-dependencies/

[6] *Stripe API Reference*. (2020). Stripe API Reference. https://stripe.com/docs/api

[7] Alonso, F. (2019). *fraanaalonso/Padel_Web*. GitHub. https://github.com/fraanaalonso/Padel_Web

[8] *DevDocs — CSS documentation*. (2020). CSS documentation. https://devdocs.io/css/

[9] *DevDocs — HTML documentation*. (2020). HTML documentation. https://devdocs.io/html/

[10] Freitas, V. (2016, 24 agosto). *How to Create a One Time Link*. Simple is Better Than Complex. https://simpleisbetterthancomplex.com/tutorial/2016/08/24/how-to-create-one-time-link.html

[11] *Master en Python: Aprender Python 3, Django, Flask y Tkinter*. (2020). Udemy. https://www.udemy.com/course/master-en-python-aprender-python-django-flask-y-tkinter/

[12] *W3Schools Online Web Tutorials*. (2021). W3Schools Online Tutorials. https://www.w3schools.com/

[13] *SB Admin 2 - Free Bootstrap Admin Theme*. (2014). Start Bootstrap. https://startbootstrap.com/theme/sb-admin-2

[14] *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. (2002). Ideal Modeling & Diagramming. https://www.visual-paradigm.com/

[15] Smartsheet. (2006). *Welcome back.* https://es.smartsheet.com/welcome-customers-home

[16] Team, D. (2019, 19 marzo). Django Project Layout and Different Files Structure in Root Directory. DataFlair. https://data-flair.training/blogs/django-project-layout/

[17] Osis, Donins, J. U. (2017). *Unified Process - an overview | ScienceDirect Topics*. Unified Process. https://www.sciencedirect.com/topics/computer-science/unified-process

[18] FatSecret. (2021). *Alimentos y Marcas en EspaÃ±a*. https://www.fatsecret.es/calor%C3%ADas-nutrici%C3%B3n/

[19] *Attention Required! | Cloudflare*. (2021). PixaBay. https://pixabay.com/es/

[20] *Icons - Materialize*. (2021). Icons Materialize. https://materializecss.com/icons.html

ANEXO

Diagramas de secuencia del sistema restantes

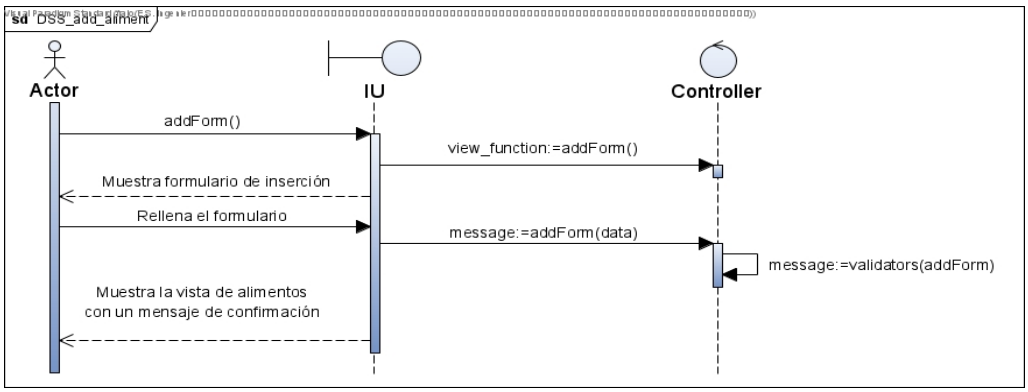

Ilustración 19. DSS añadir alimento

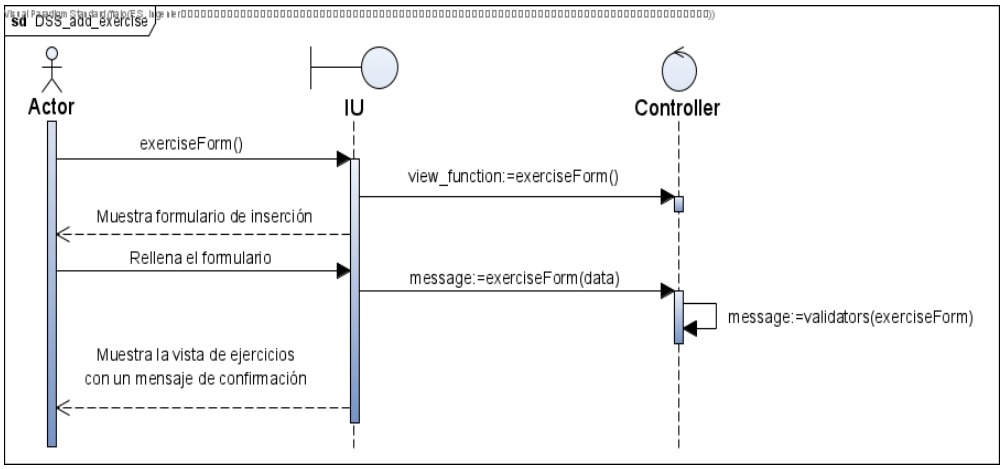

Ilustración 20. DSS añadir Ejercicio

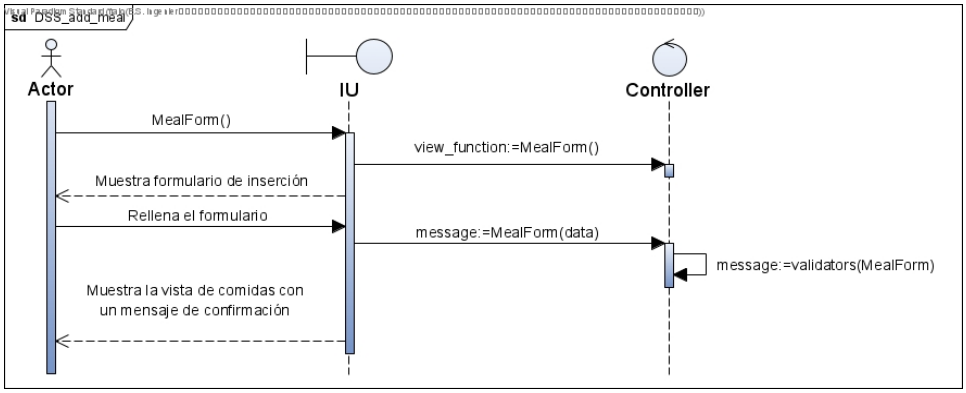

Ilustración 21. DSS añadir comida

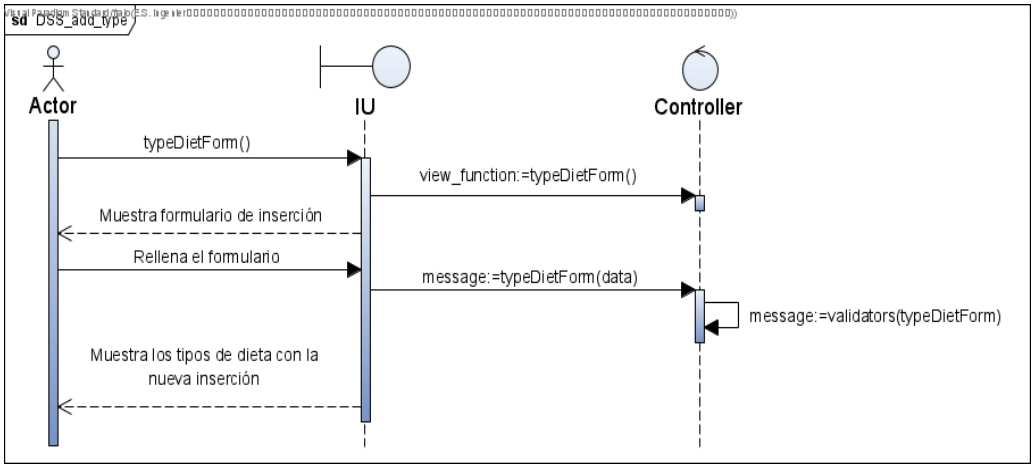

Ilustración 22. DSS añadir tipo de dieta

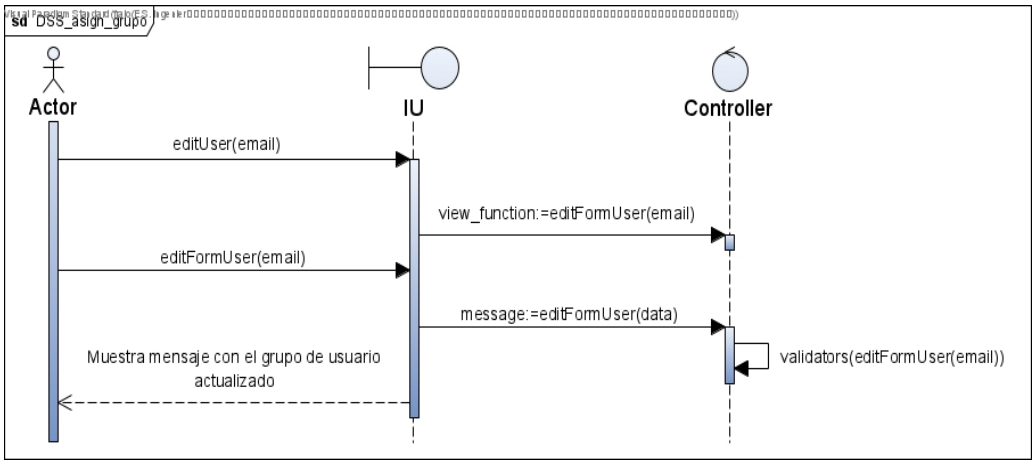

Ilustración 23. DSS asignar grupo

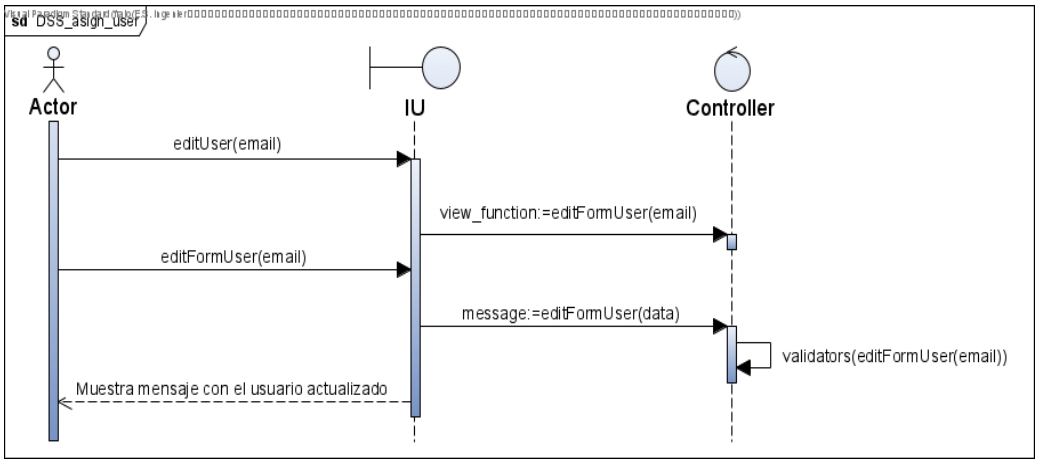

Ilustración 24. DSS asignar entrenador

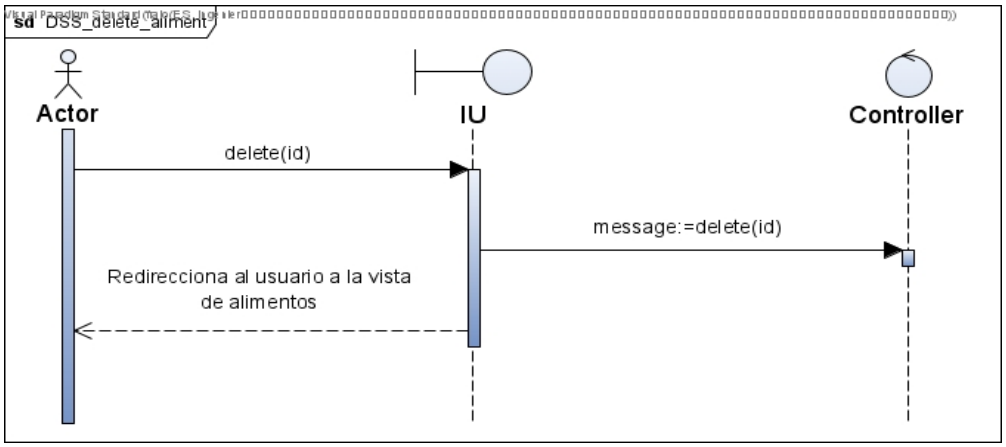

Ilustración 25. DSS borrar alimento

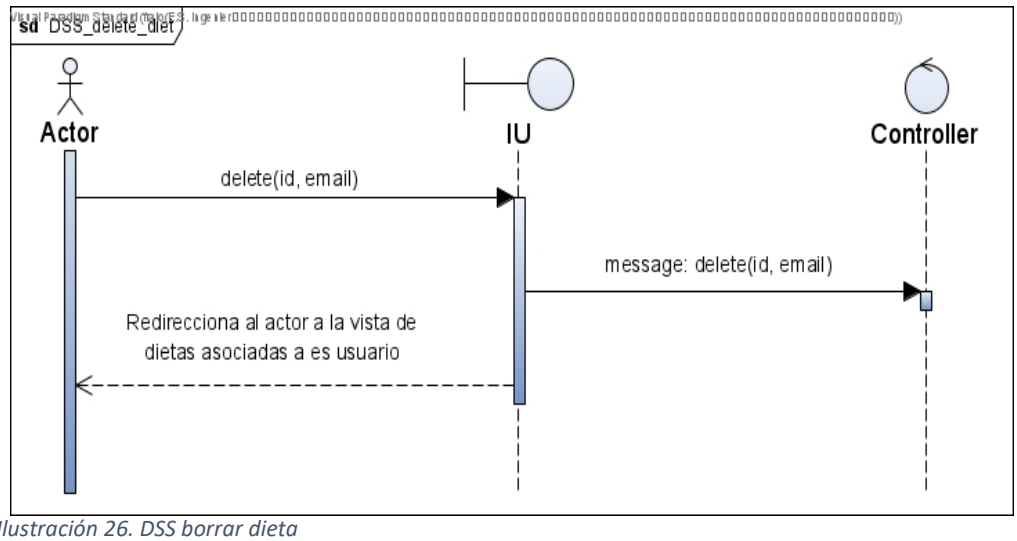

Ilustración 26. DSS borrar dieta

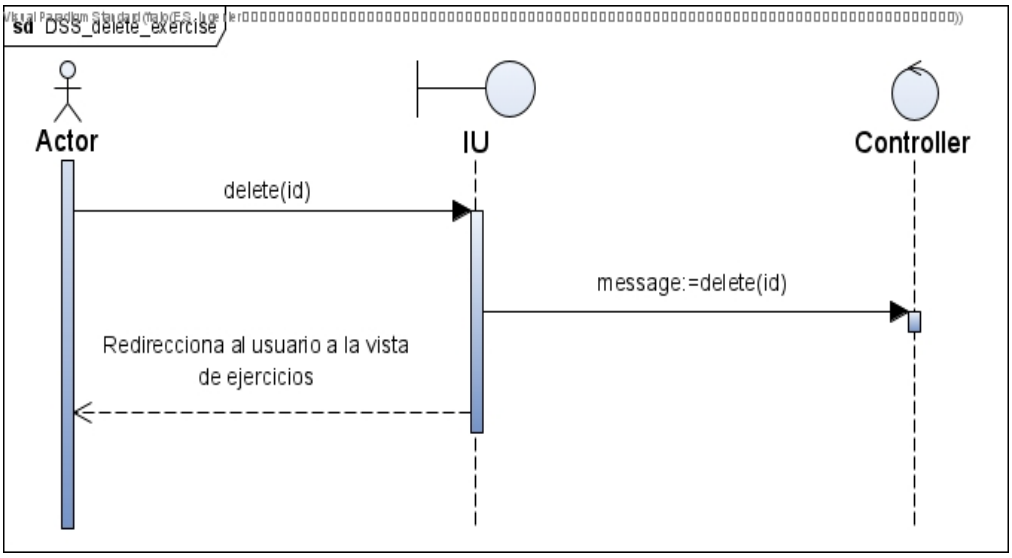

Ilustración 27. DSS borrar ejercicio

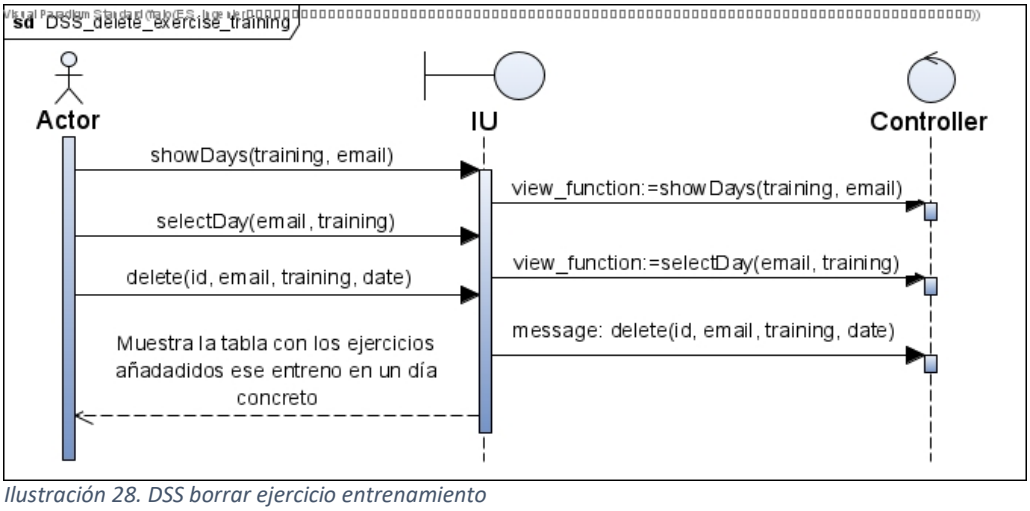

Ilustración 28. DSS borrar ejercicio entrenamiento

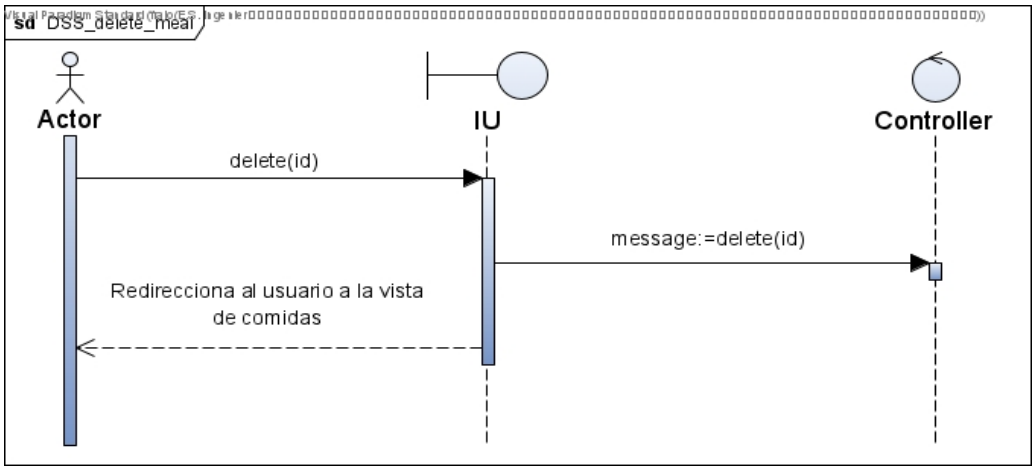

Ilustración 29. DSS borrar comida

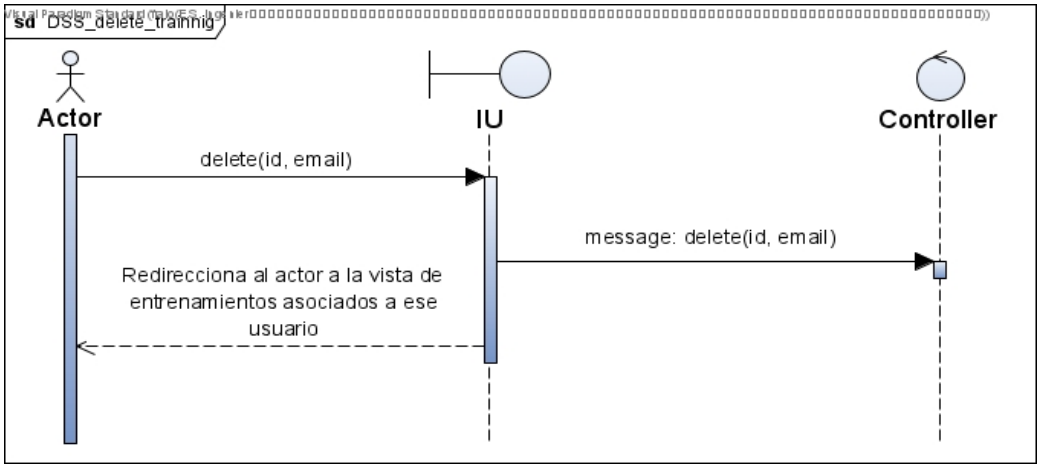

Ilustración 30. DSS borrar entrenamiento

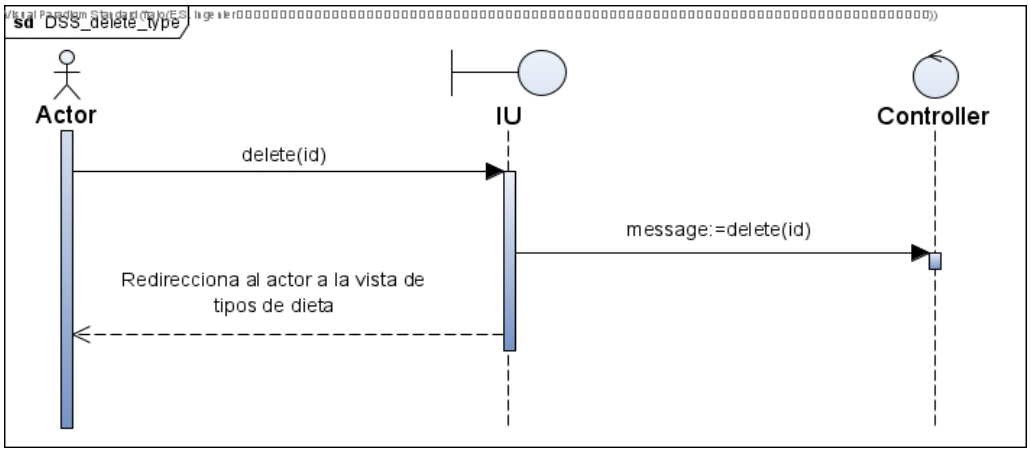

Ilustración 31. DSS borrar tipo de dieta

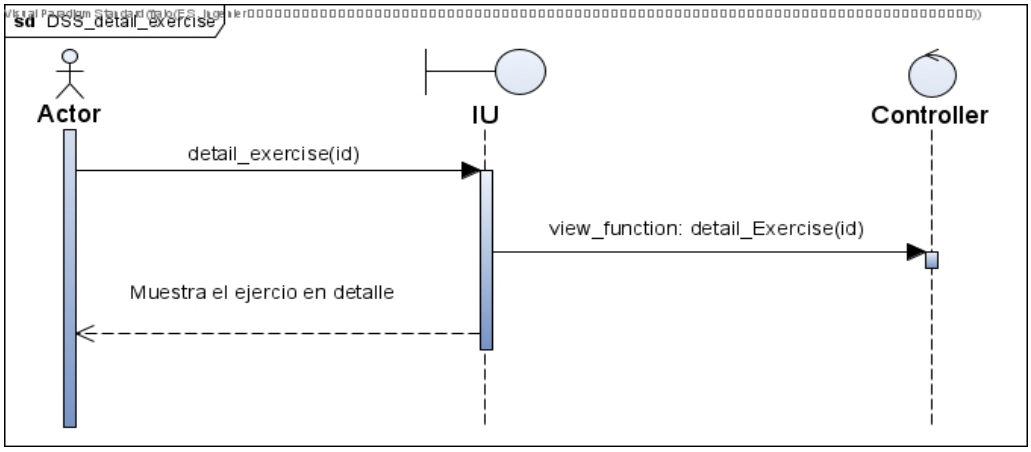

Ilustración 32. DSS ejercicio en detalle

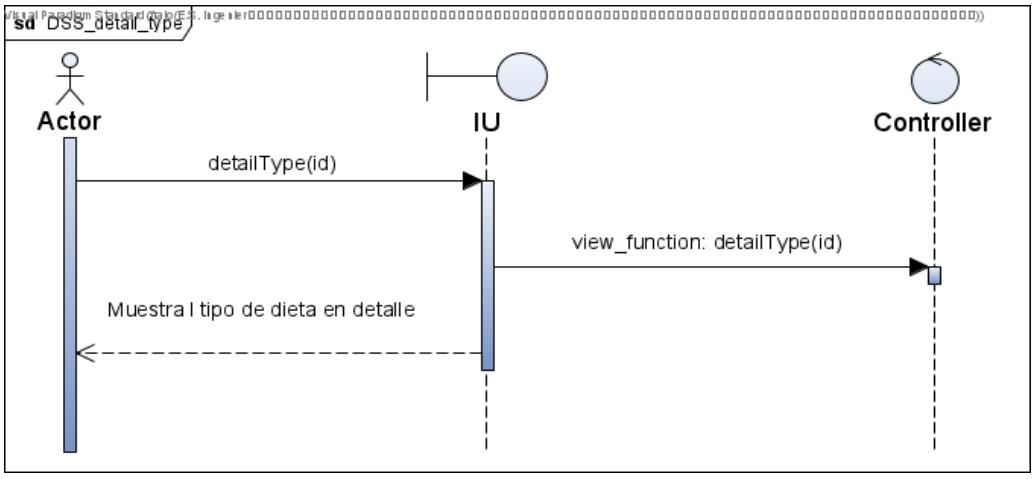

Ilustración 33. DSS tipo de dieta en detalle

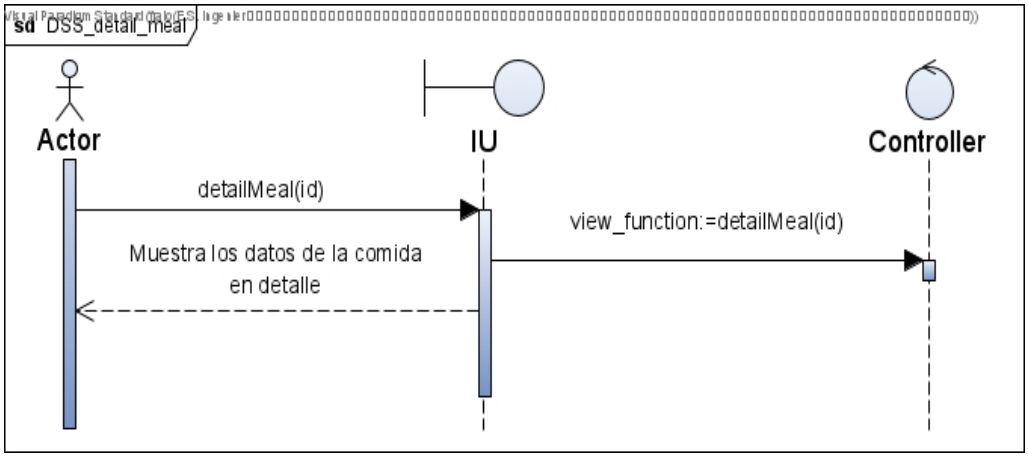

Ilustración 34. DSS comida en detalle

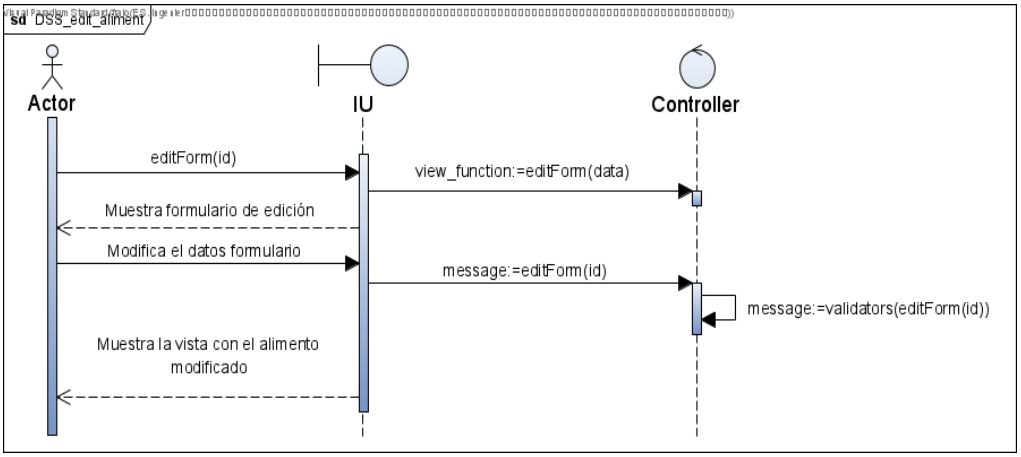

Ilustración 35. DSS editar alimento

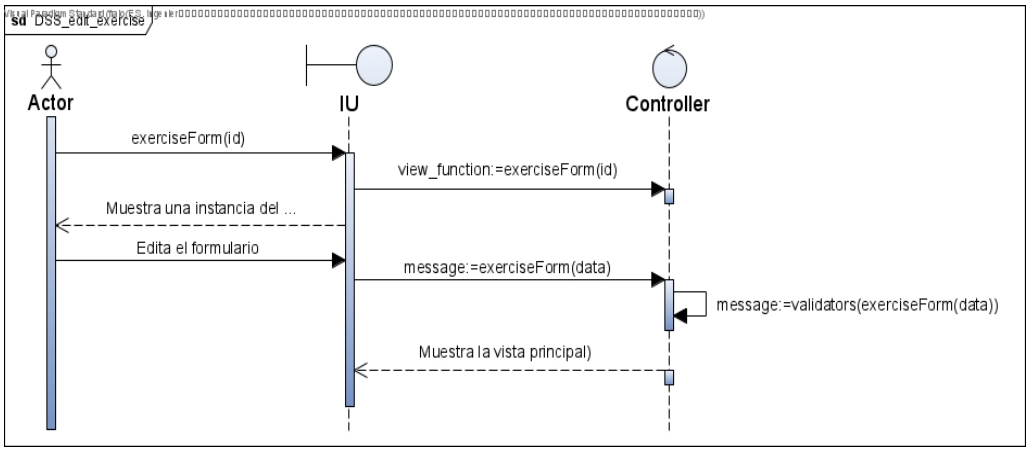

Ilustración 36. DSS editar ejercicio

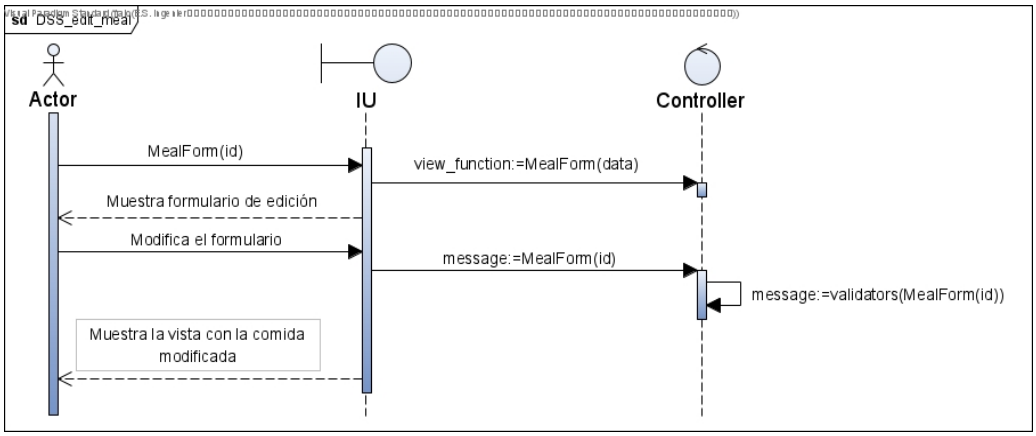

Ilustración 37. DSS editar comida

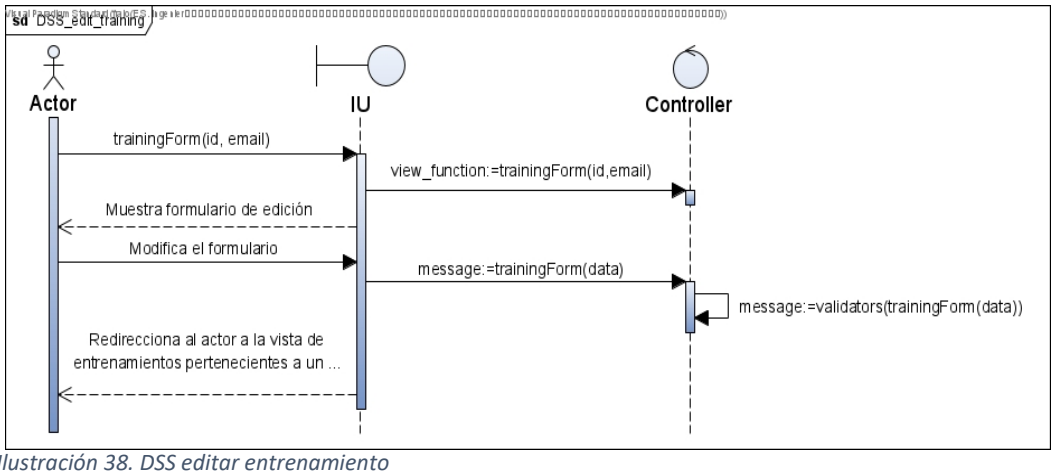

Ilustración 38. DSS editar entrenamiento

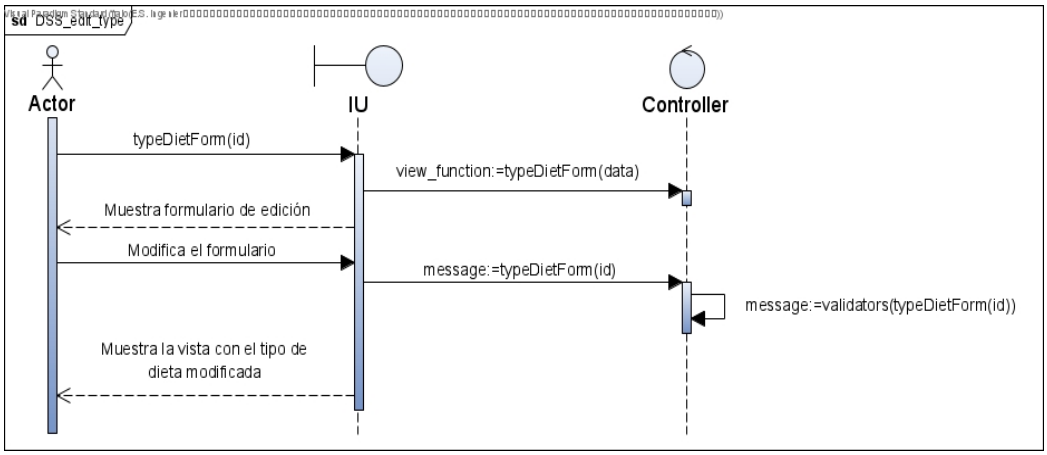

Ilustración 39. DSS editar tipo de dieta

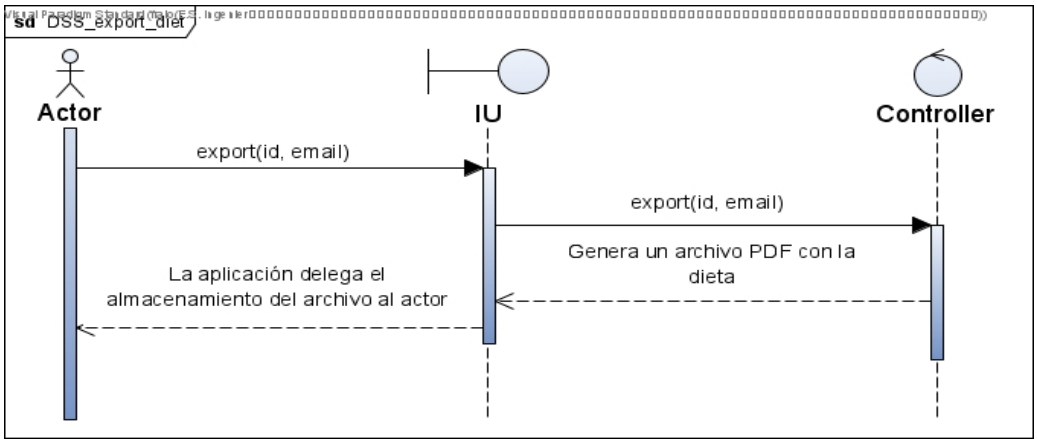

Ilustración 40. DSS exportar dieta

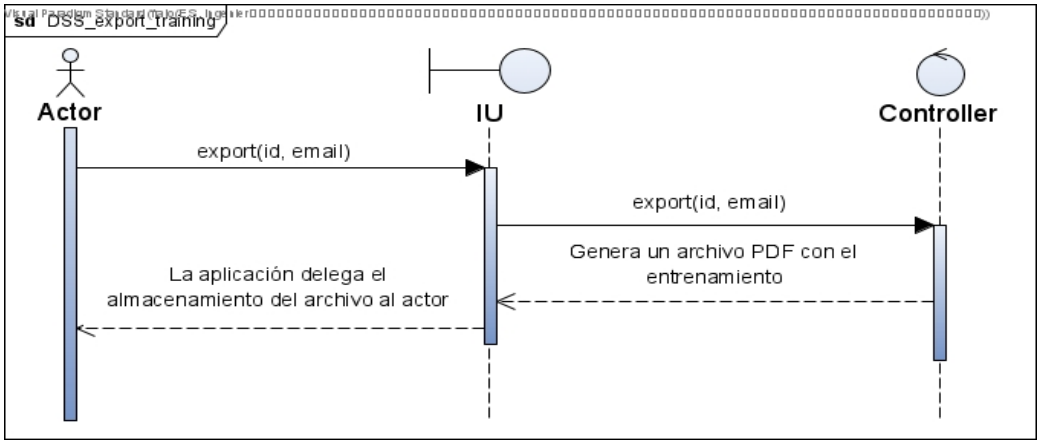

Ilustración 41. DSS exportar entrenamiento

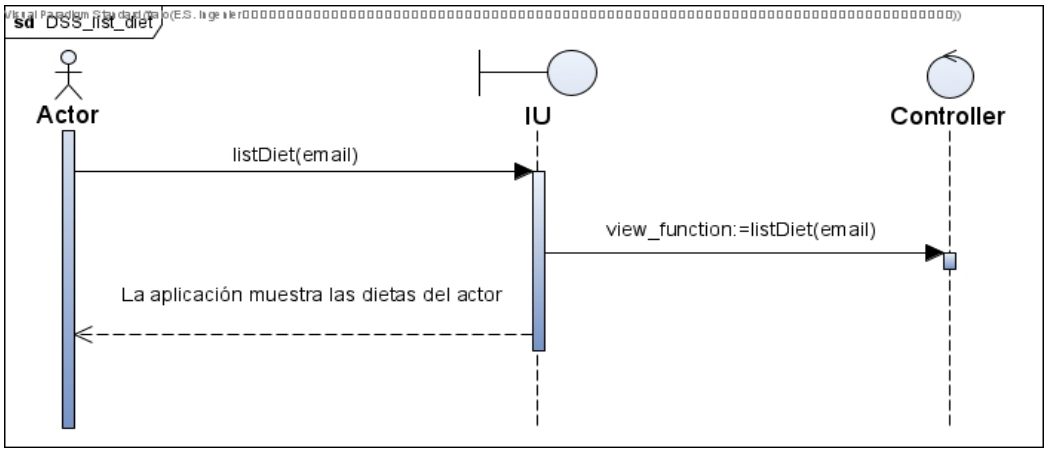

Ilustración 42. DSS listar dieta

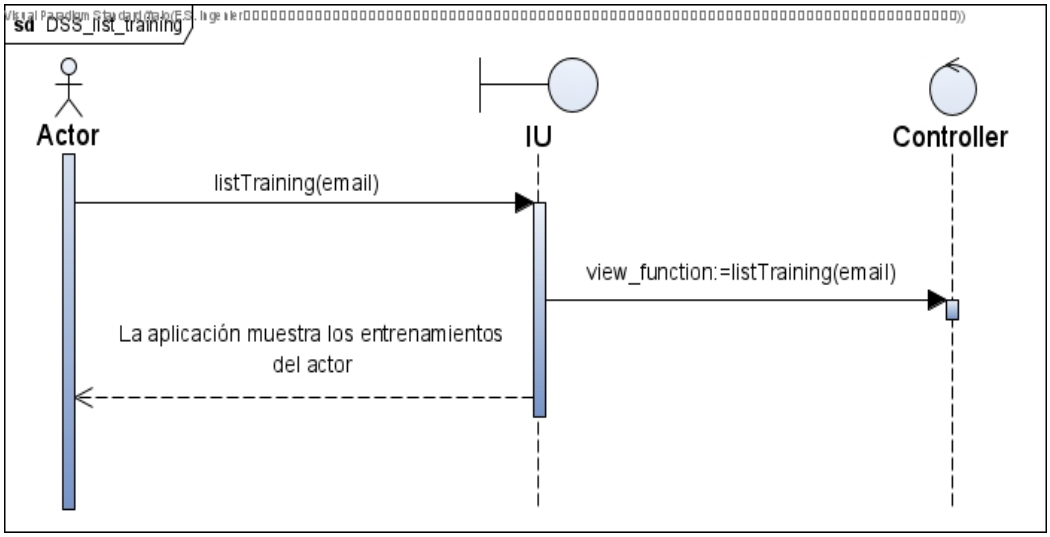

Ilustración 43. DSS listar entrenamiento

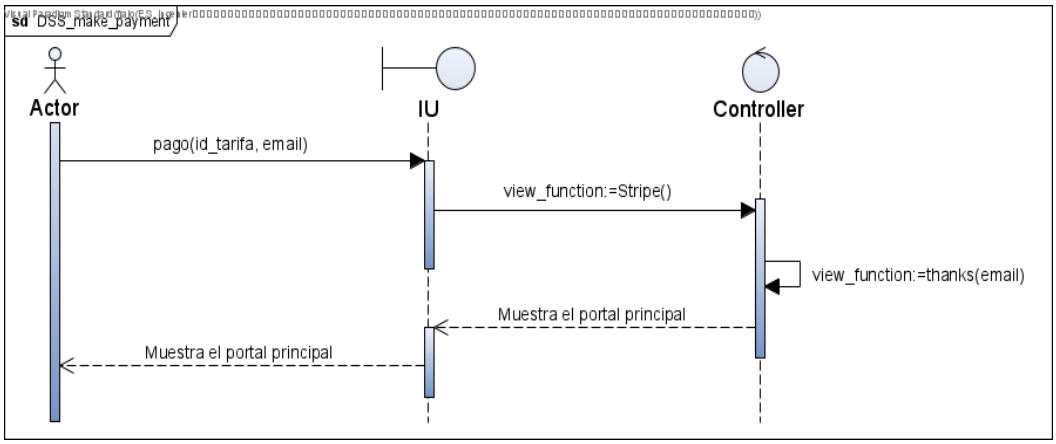

Ilustración 44. DSS realizar pago

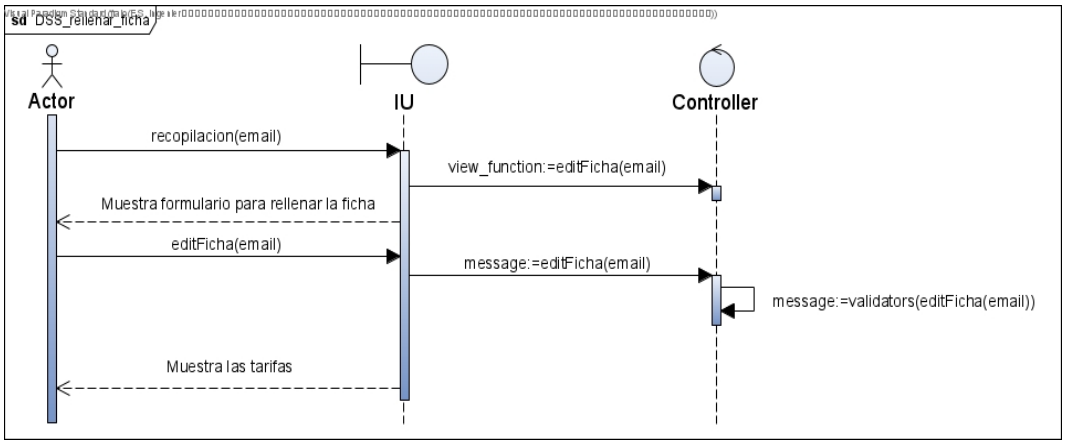

Ilustración 45. DSS rellenar ficha técnica

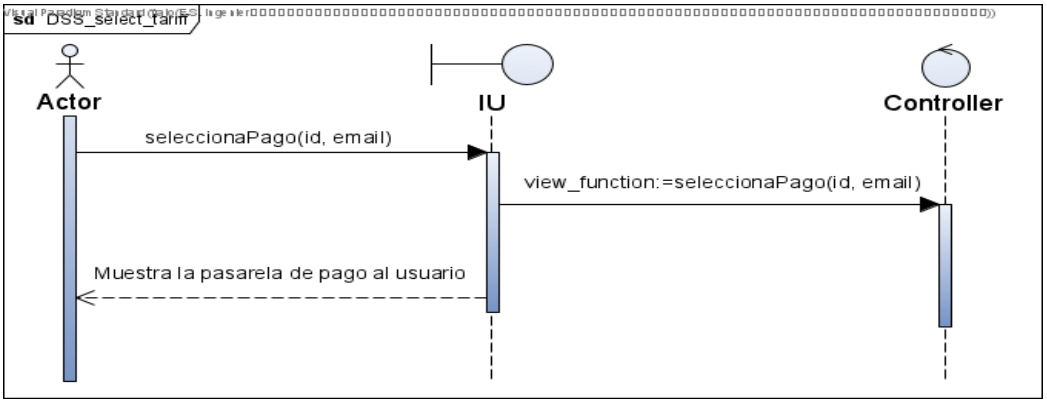

Ilustración 46. DSS seleccionar tarifa

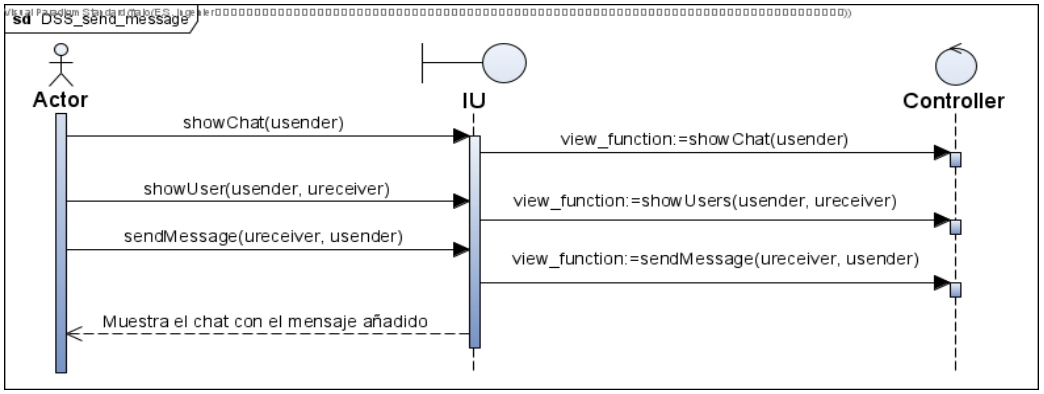

Ilustración 47. DSS enviar mensaje

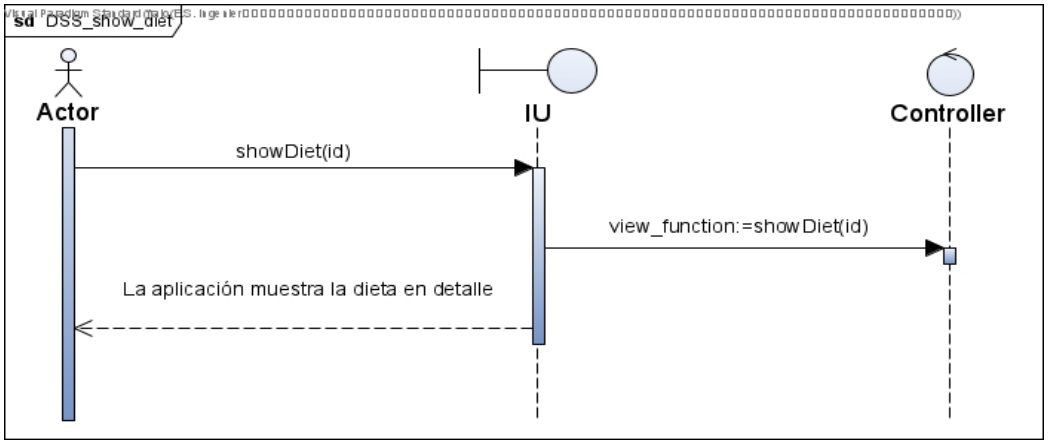

Ilustración 48. DSS mostrar dieta

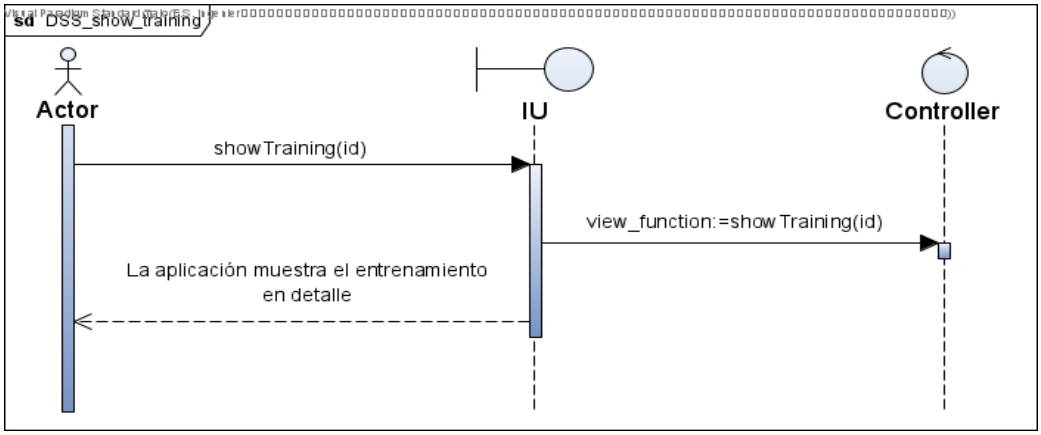

Ilustración 49. DSS mostrar entrenamiento

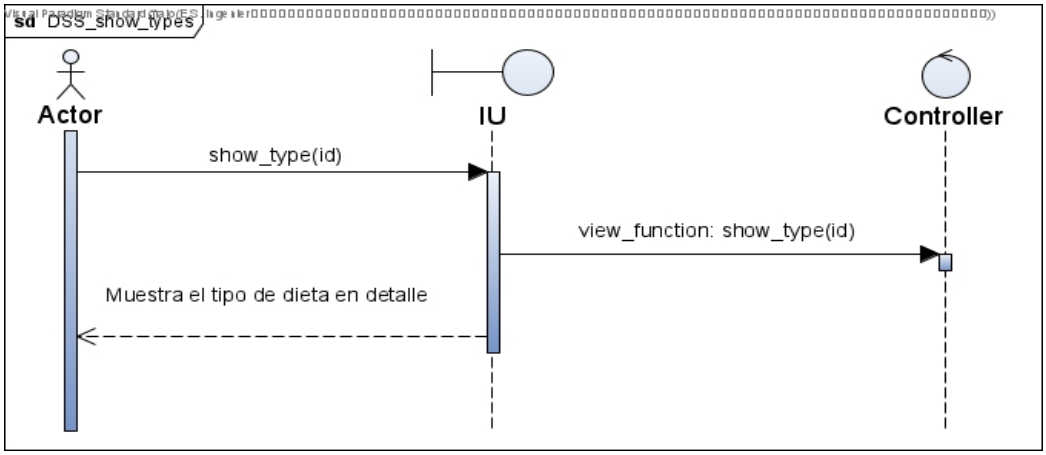

Ilustración 50. DSS mostrar tipo de dieta

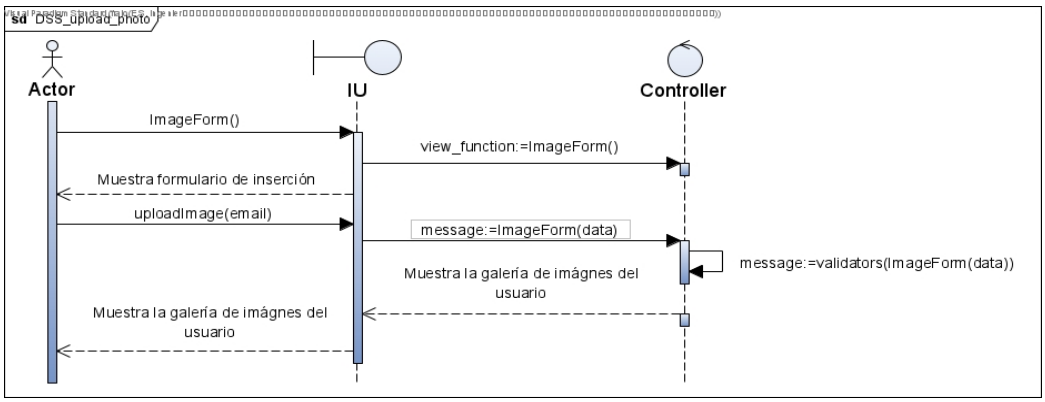

Ilustración 51. DSS subir foto