UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Máster Universitario en Ingeniería Web

Trabajo Fin de Máster

Aplicación web para la gestión de comidas diarias e insumos necesarios en su preparación.

Autor

Ricardo Miguel Vente Montes

Tutor

Luis Fernández Muñoz

julio de 2024

AGRADECIMIENTOS

A Dios por estar siempre presente en mi vida.

A mi familia, en especial a mis padres, Elsa y Ricardo, por su apoyo incondicional.

A mi asesor, Luis Fernández Muñoz, por su tiempo y dedicación en este proyecto.

RESUMEN

Este proyecto tiene como objetivo unificar y consolidar el conocimiento adquirido a lo largo de todo del Máster de Ingeniería Web .

El proyecto desarrolla una aplicación web, Merkado, que permita a los usuarios gestionar las comidas que prepara a diario y la lista de compra de insumos necesarios en su preparación .

De esta manera, se busca facilitar parte del proceso de hacer mercado en el hogar, permitiendo a los usuarios listar un grupo de comidas con los ingredientes y cantidades necesarias en su preparación, luego planificar un menú diario para un periodo de tiempo y tener la posibilidad de obtener la lista total de insumos que necesitaría comprar.

Se utilizará Angular para el desarrollo de la capa de presentación, Spring para la capa de negocio y MySql para la capa de datos.

Durante todo el proceso de desarrollo se plantean las diferentes etapas de trabajo de la metodología de desarrollo RUP (Rational Unified Process)

En el primer capítulo se detalla el objetivo de este proyecto y se presenta el modelo de dominio del sistema.

En el segundo capítulo, requisitos, se presentarán los diagramas y especificación de casos de uso y el prototipo de interfaz de usuario del sistema.

El tercer capítulo detalla el análisis, aquí se presentarán los diagramas de clases, análisis de casos de uso y diagrama de secuencias correspondientes al análisis.

En el cuarto capítulo se detalla el diseño del sistema mediante el diagrama de la arquitectura del sistema, diagrama de clases, diagrama general de casos de uso, diagrama de paquetes y diagrama de datos.

En el quinto capítulo se presenta la implementación y pruebas del sistema.

PALABRAS CLAVE

RUP, Rational Unified Process, Spring, Angular, MySQL, Comida, Insumos, Planificador.

ABSTRACT

This project's main goal is to unify and consolidate the knowledge acquired throughout the Master's in Web Engineering.

The project involves developing a web application, Merkado, that allows users to manage the meals they prepare daily and the shopping list of the ingredients needed for their preparation.

In this way, the goal is to simplify part of the household shopping process by enabling users to list a group of meals with the necessary ingredients and quantities for their preparation, then plan a daily menu for a period of time and have the ability to obtain the total list of ingredients they would need to buy..

Angular will be used for developing the presentation layer, Spring for the business layer, and MySQL for the data layer.

Throughout the development process, different stages of the Rational Unified Process (RUP) development methodology will be used.

The first chapter details the objective of this project and presents the domain model of the system.

In the second chapter, requirements, details of the use case diagrams and specifications, as well as the user interface prototype of the system, will be presented.

The third chapter details the analysis and design of the system. Here, class diagrams, use case analysis, and sequence diagrams corresponding to the analysis will be presented. For the design part, the system architecture diagram, class diagram, general use case diagram, package diagram, and data diagram will be presented.

The fourth chapter presents the implementation of the system.

Finally, the development of tests is explained, the organization for conducting them is outlined, and code coverage graphs are shown.

KEYWORDS

RUP, Rational Unified Process, Spring, Angular, MySQL, Meal, Ingredients, Planner

Tabla de Contenidos

Contenido

OBJETIVOS

El objetivo principal de este proyecto es unificar y consolidar el conocimiento adquirido a lo largo de todo del Máster de Ingeniería Web. Específicamente busca consolidar los conocimientos aprendidos en las siguientes asignaturas:

- Ingeniería Web Visión General
- Arquitectura y Patrones para Aplicaciones Web
- Front-end para Navegadores Web
- Back-end con Tecnologías de Código Abierto
- Metodologías de Desarrollo Web

Para ello se desarrollará una aplicación web utilizando la metodología RUP (Rational Unified Process).

La capa de negocio se hará utilizando Spring, la capa de presentación usará Angular y la capa de datos MySQL.

La aplicación a desarrollar para el proyecto de fin de Máster, Merkado, permitirá a los usuarios gestionar las comidas que preparan a diario y la lista de insumos necesarios en su preparación .

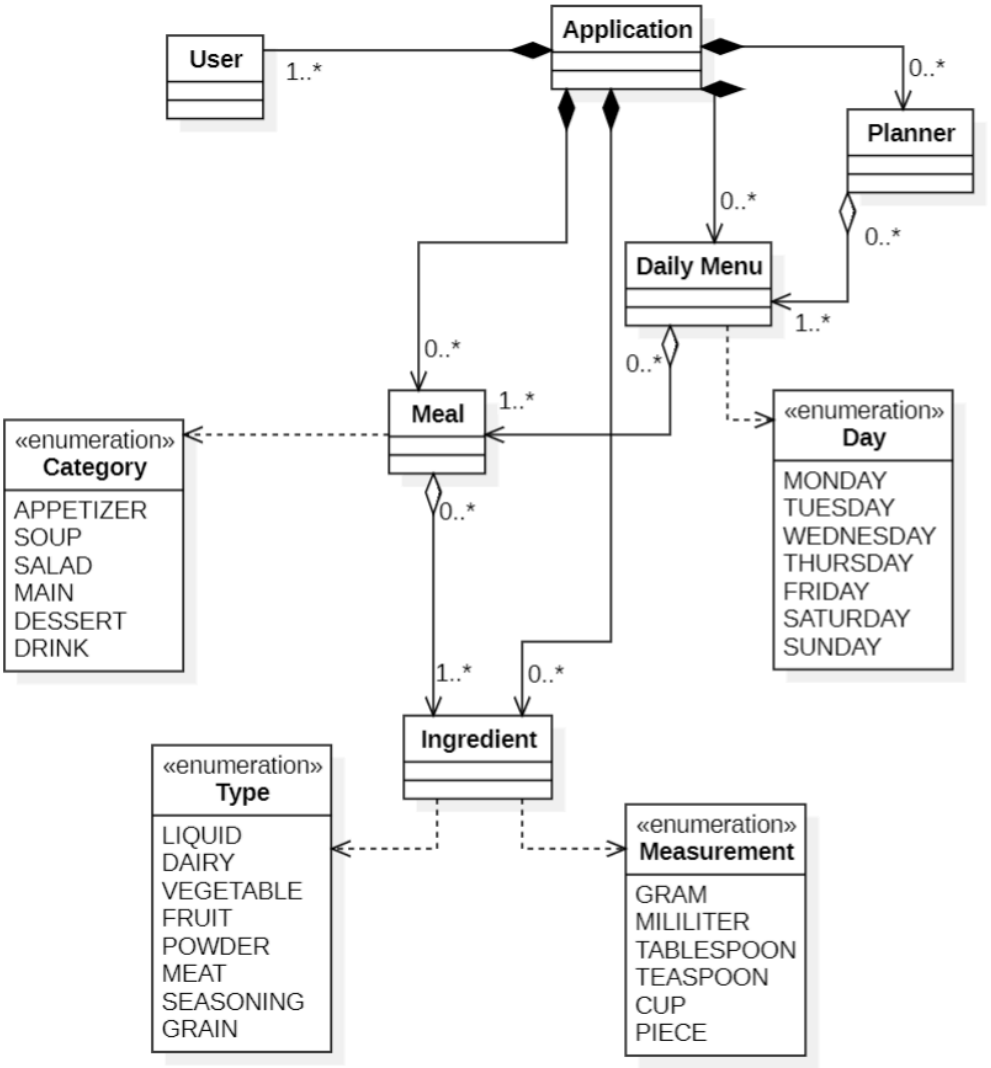
De esta manera, se busca facilitar parte del proceso de hacer mercado en el hogar, permitiendo a los usuarios listar un grupo de comidas con los ingredientes y cantidades necesarias en su preparación, luego planificar un menú diario para un periodo de tiempo y tener la posibilidad de obtener la lista total de insumos que necesitaría comprar.

La aplicación al ser reducida para este tipo de proyectos contempla un solo actor, el usuario, y 23 casos de uso. Esto permitirá centrarnos en la mayoría de actividades de la metodología RUP. La fase de gestión no se desarrollará ya que el objetivo del proyecto es tratar de unificar el conocimiento adquirido en las distintas materias cursadas a lo largo del Máster, por lo que estará enfocado en el análisis, diseño e implementación de la solución.

Capítulo 1

Modelo de dominio

A continuación se detalla el diagrama de modelo de dominio, el cual describe el concepto general del sistema. Este modelo tiene como objetivo comprender la solución y establecer un vocabulario común entre usuarios, desarrolladores y otros implicados.



Capítulo 2

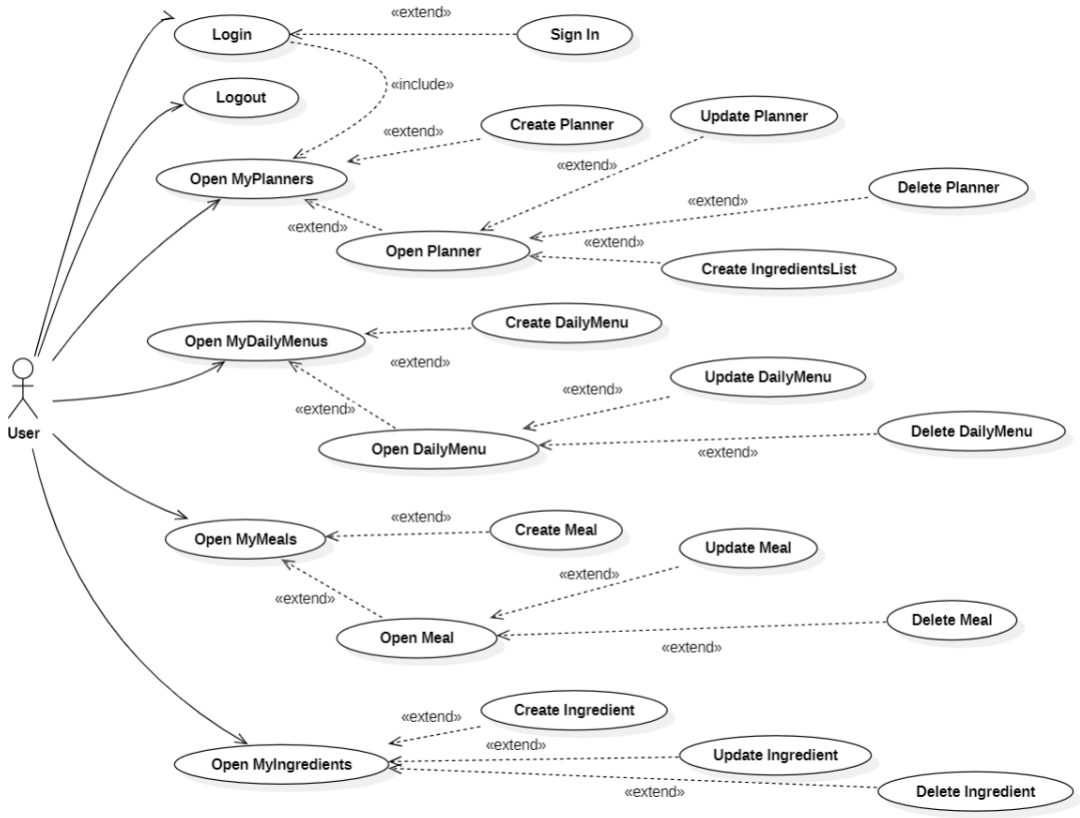
Requisitos

En éste capítulo se describen los requisitos del sistema. El objetivo es proveer a los desarrolladores una mejor comprensión de los requisitos del sistema y establecer un acuerdo con todos los participantes del desarrollo.

A la vez ayuda a sentar las bases usadas en la planificación de los contenidos técnicos de cada iteración, además de estimar de mejor manera el esfuerzo de cada uno de los requisitos que presenta esta aplicación.

1. Casos de Uso

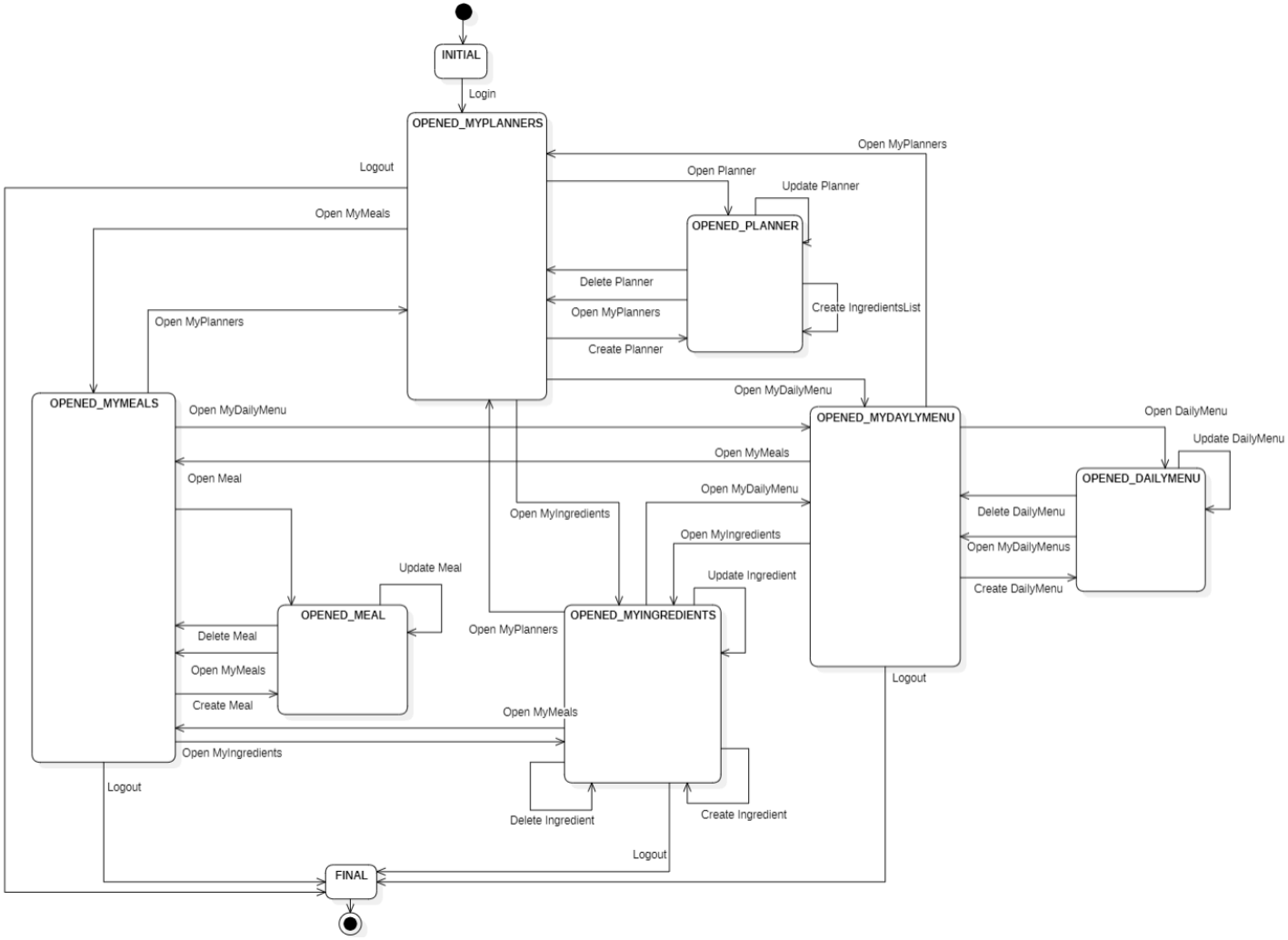
Diagrama de Casos de Uso del Sistema, en este diagrama se presenta al actor del sistema y las 23 interacciones, casos de uso, que este puede realizar y que brindan un resultado observable de interés para el actor.



2. Contexto

A continuación se presenta el diagrama de contexto de casos de uso.

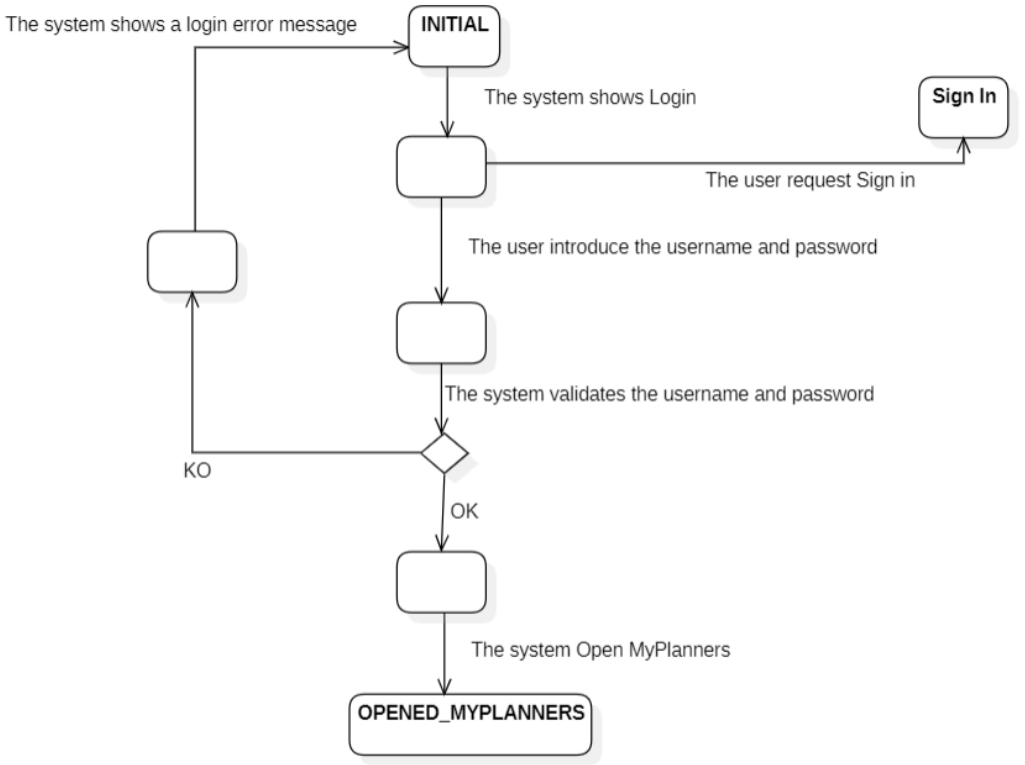
El objetivo de este diagrama es mostrar la comunicación y el flujo entre los distintos estados que existen en la aplicación. Cada flecha representa un caso de uso que conduce a un estado, el cual permite realizar otras acciones, casos de uso, que conducirán a otros estados.



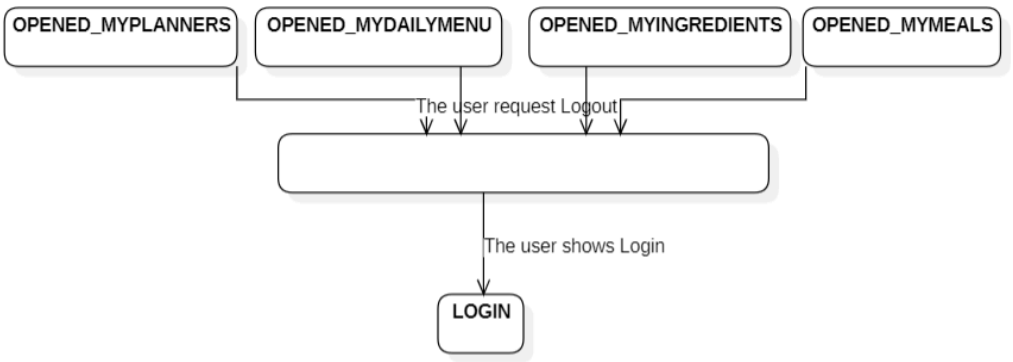
3. Especificación de casos de uso

En este apartado se presentan los diagramas de estado para cada uno de los casos de uso que presenta la aplicación. Estos diagramas representan la secuencia de interacciones entre el actor y el sistema para obtener un resultado de valor observable.

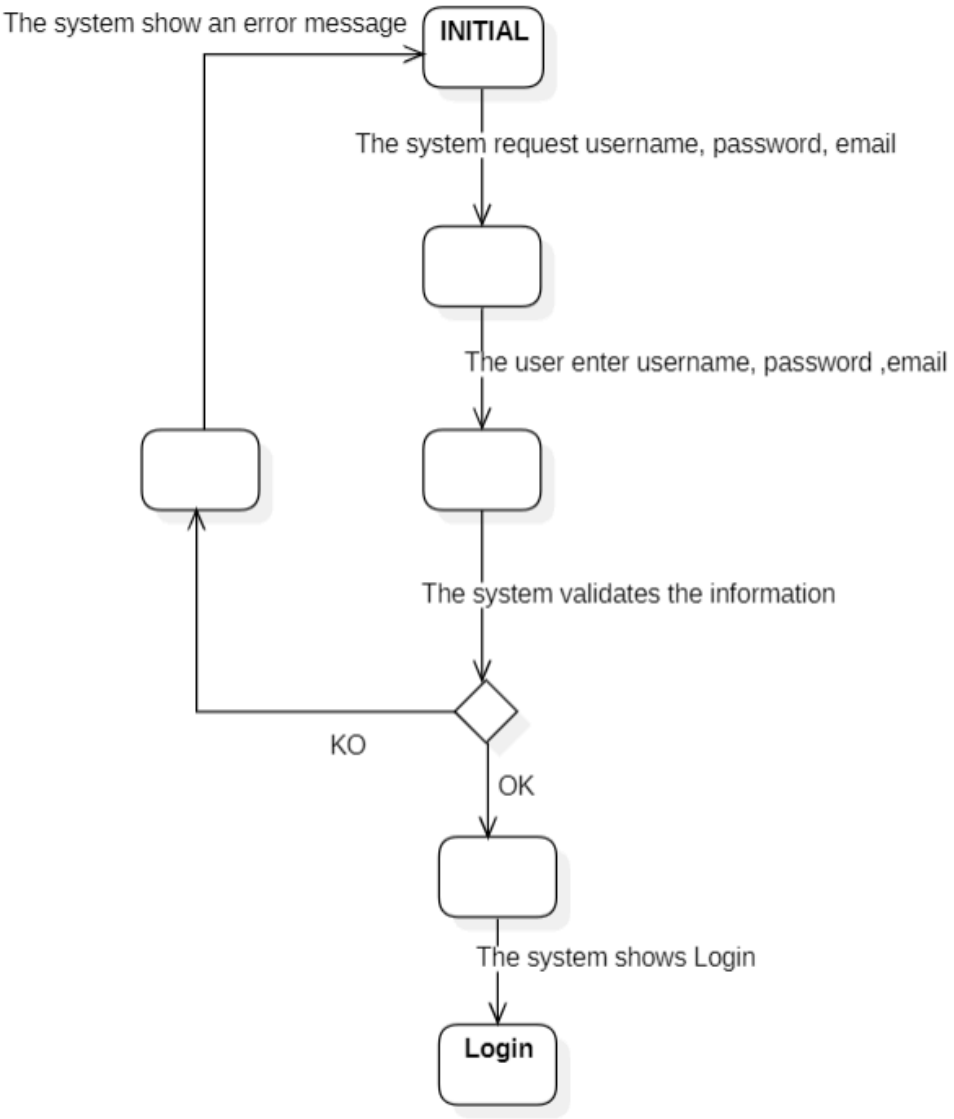
Login



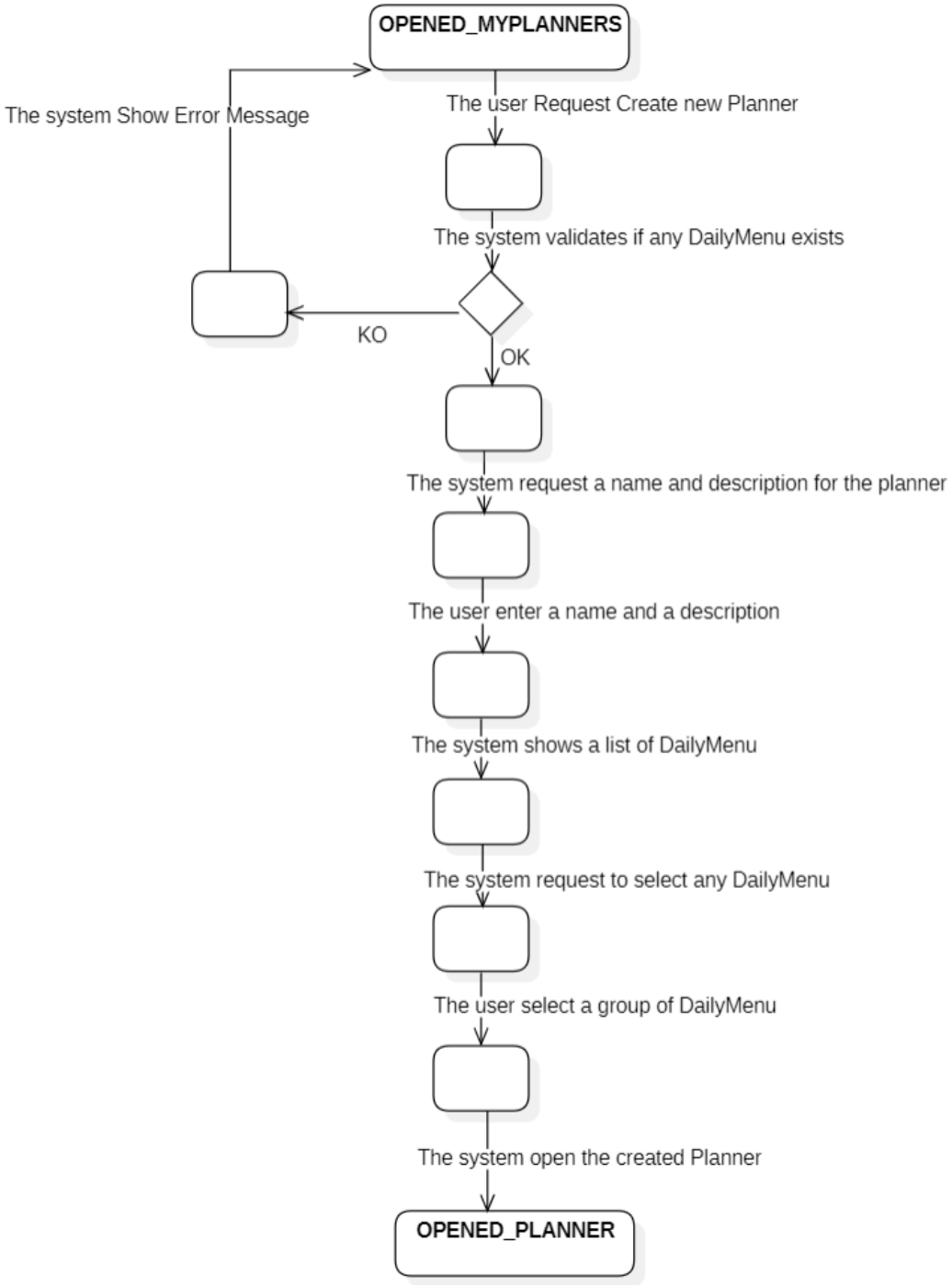
Logout



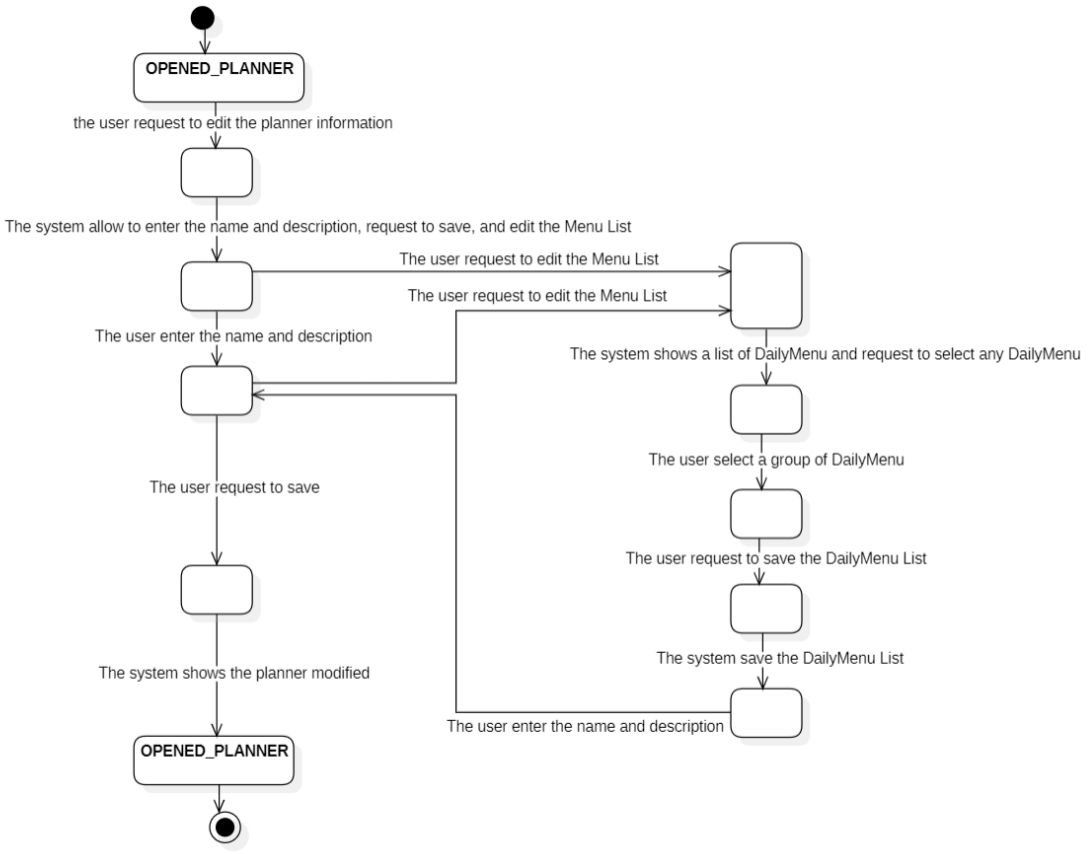
Sign In


The system show an error message
INITIAL
The system request username, password, email
The user enter username, password ,email
The system validates the information
KO
OK
The system shows Login
Login

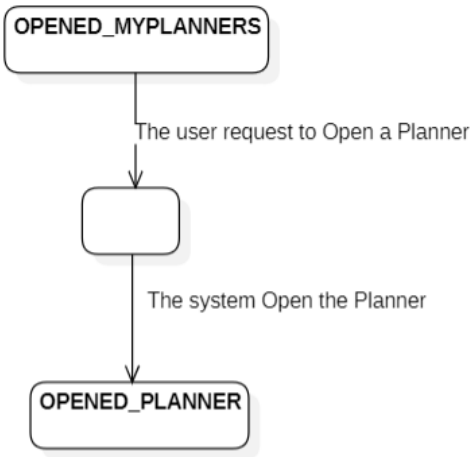
Create Planner



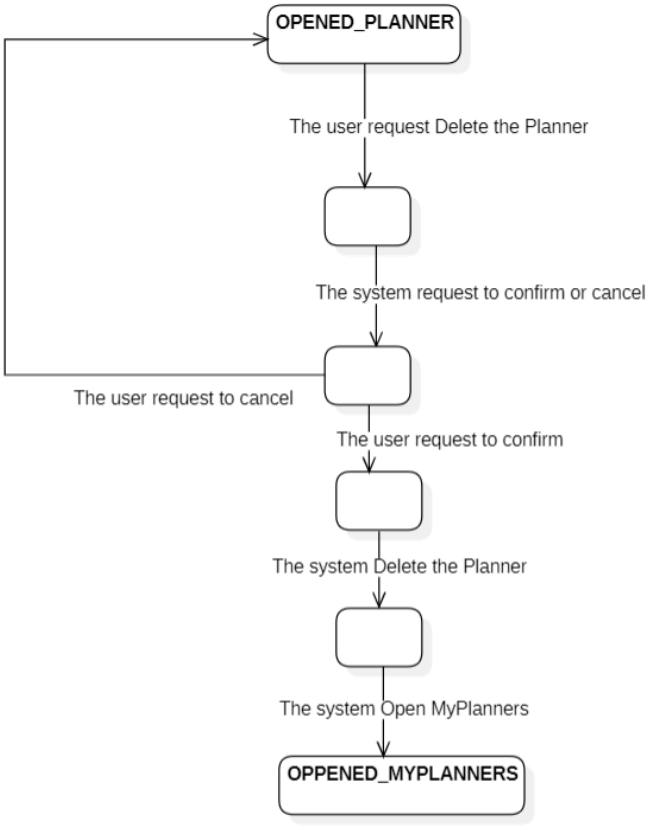
Update Planner



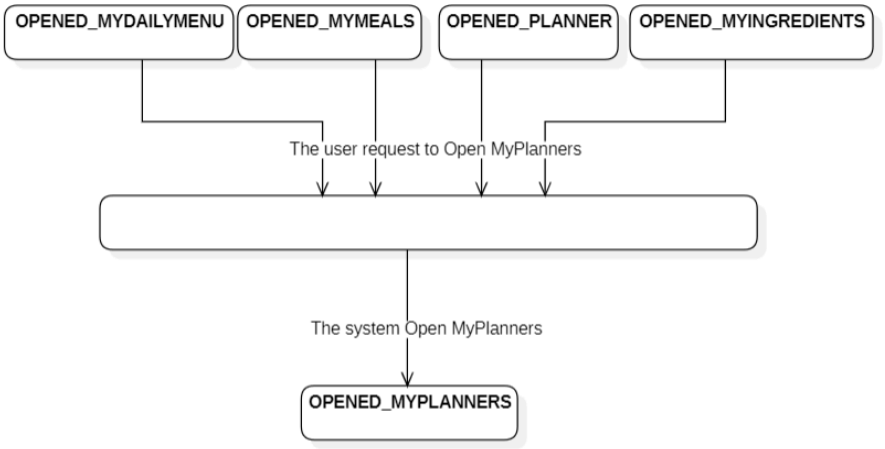
Open Planner



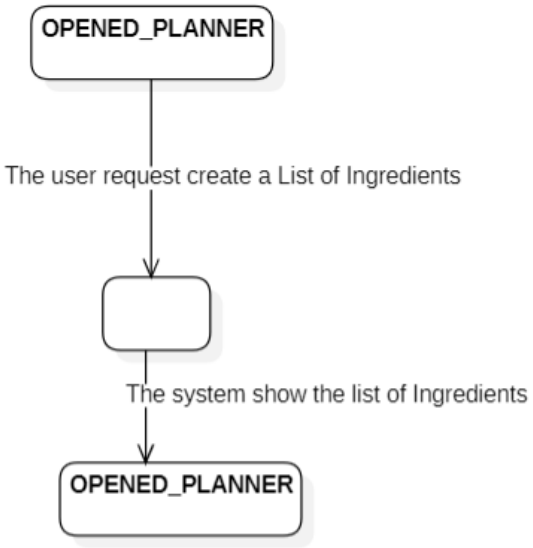
Delete Planner



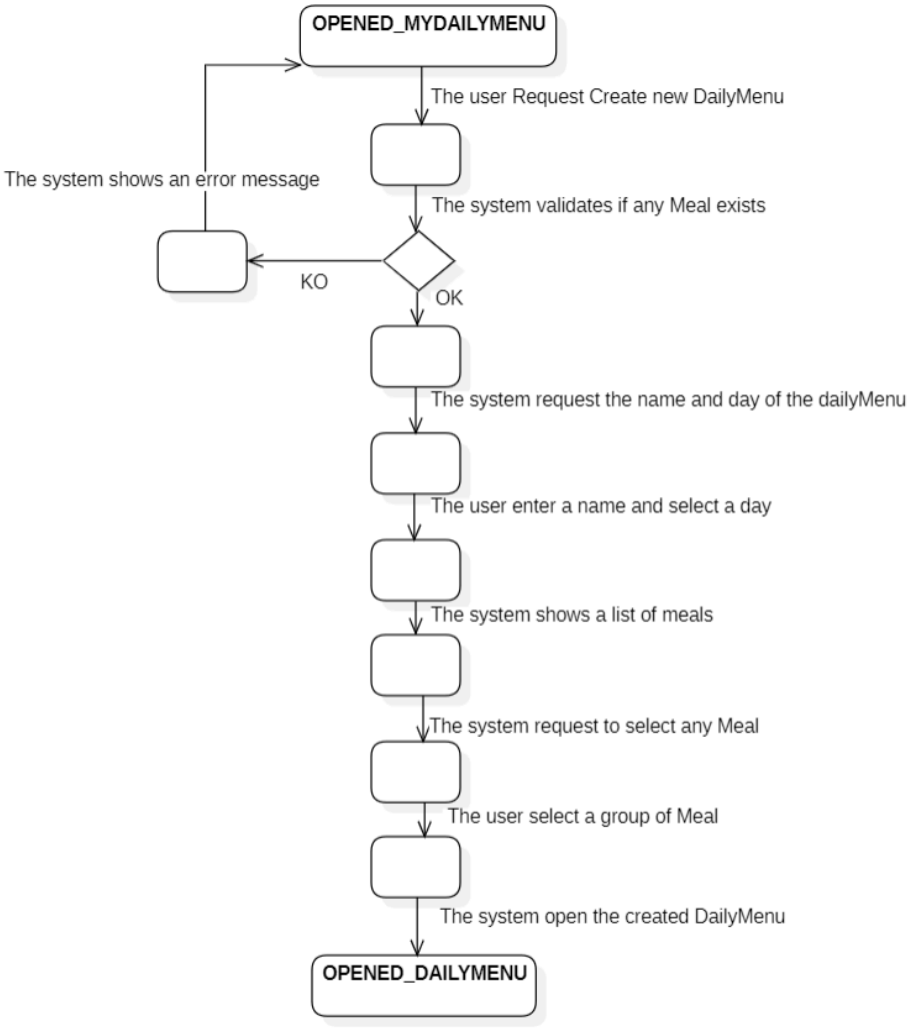
Open My Planners



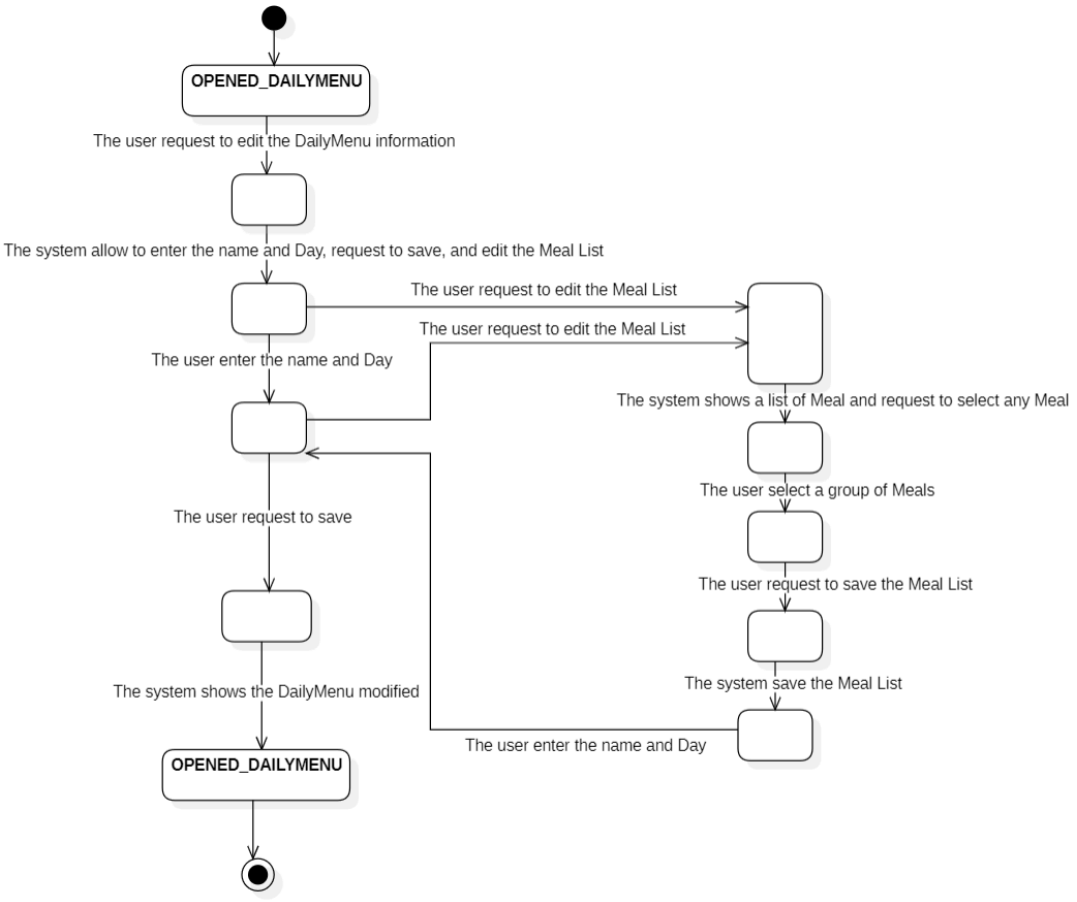
Create Ingredient List



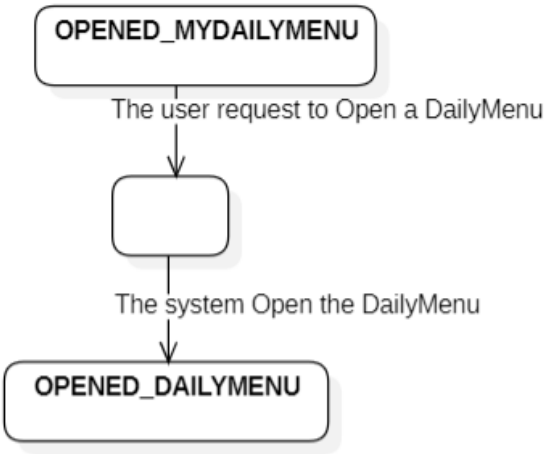
Create Daily Menu



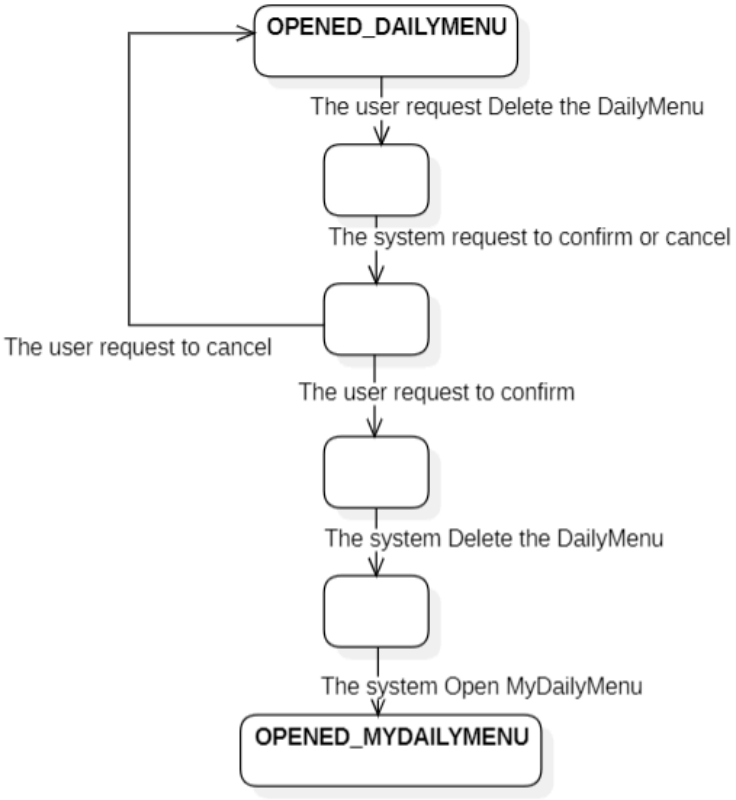
Update Daily Menu



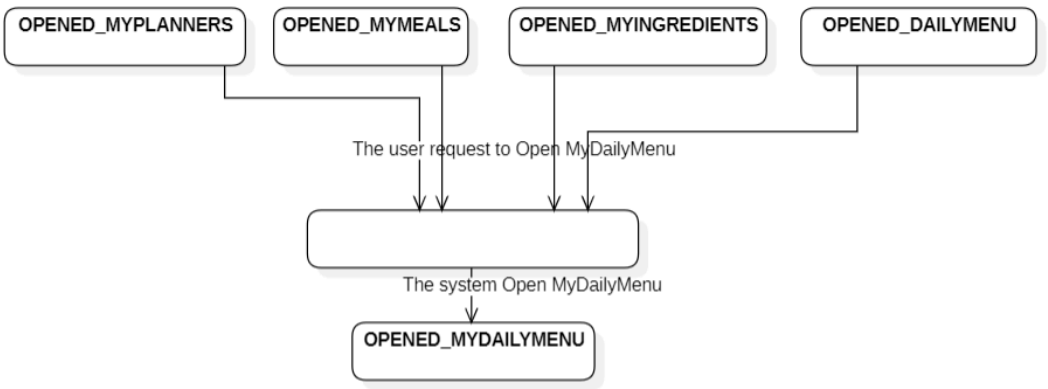
Open Daily Menu



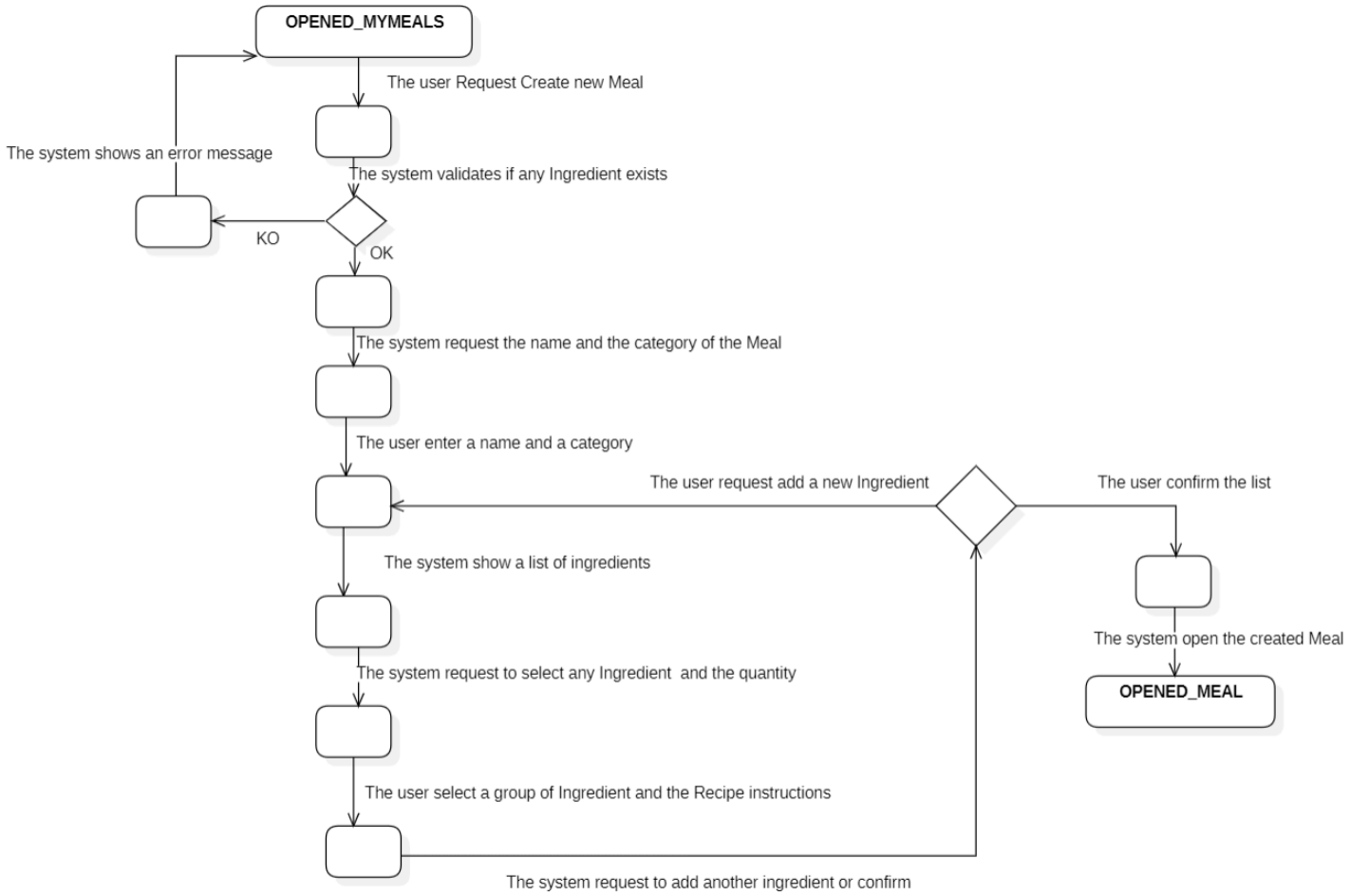
Delete Daily Menu



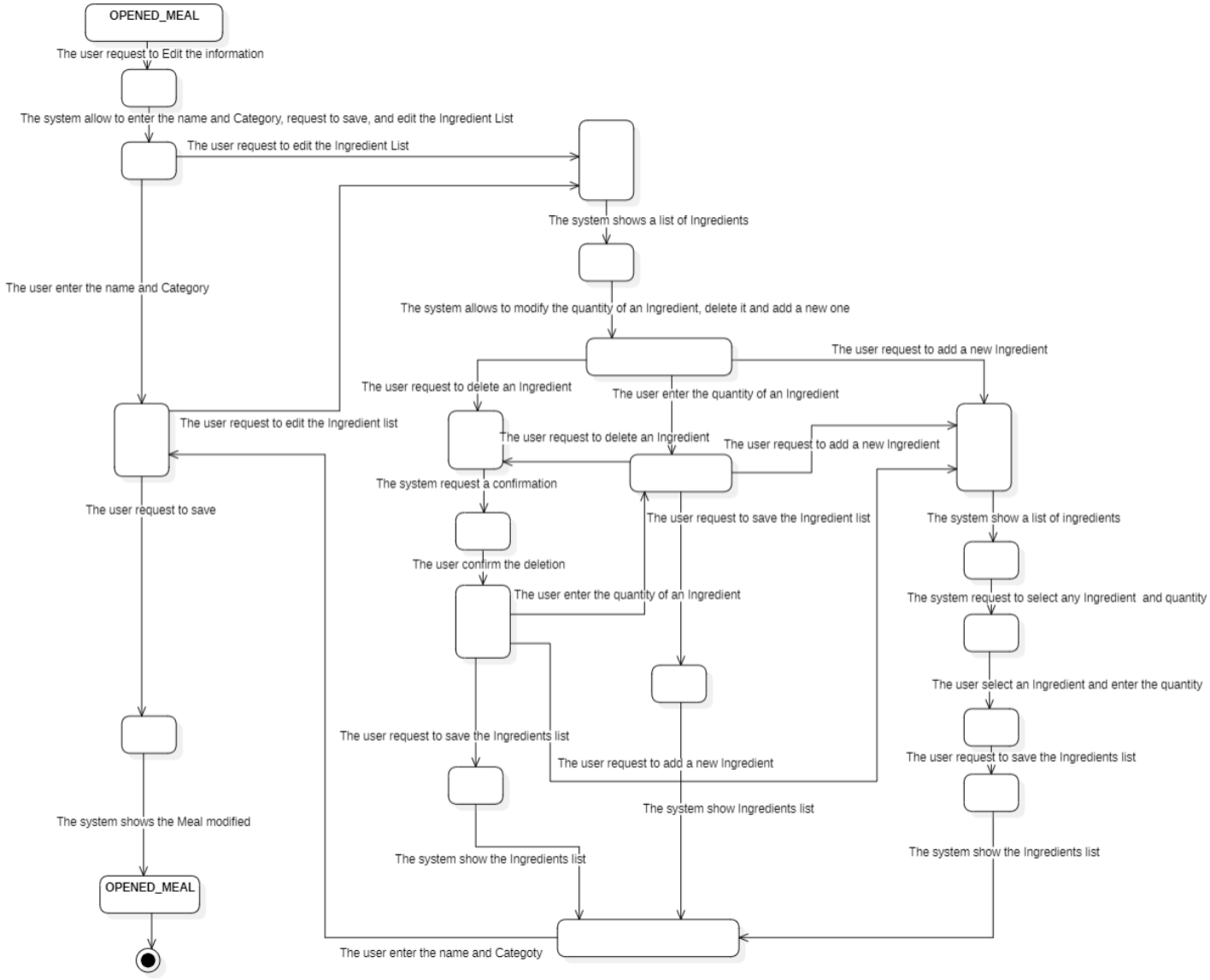
Open My Daily Menus



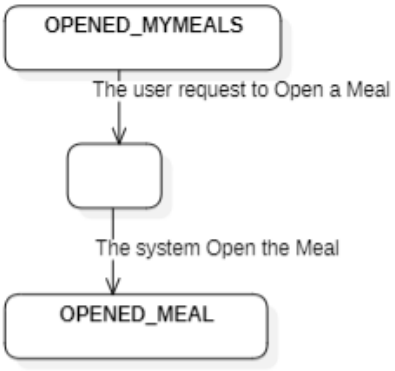
Create Meal


OPENED_MYMEALS
The user Request Create new Meal
The system shows an error message
The system validates if any Ingredient exists
KO
OK
The system request the name and the category of the Meal
The user enter a name and a category
The user request add a new Ingredient
The user confirm the list
The system show a list of ingredients
The system open the created Meal
OPENED_MEAL
The system request to select any Ingredient and the quantity
The user select a group of Ingredient and the Recipe instructions
The system request to add another ingredient or confirm

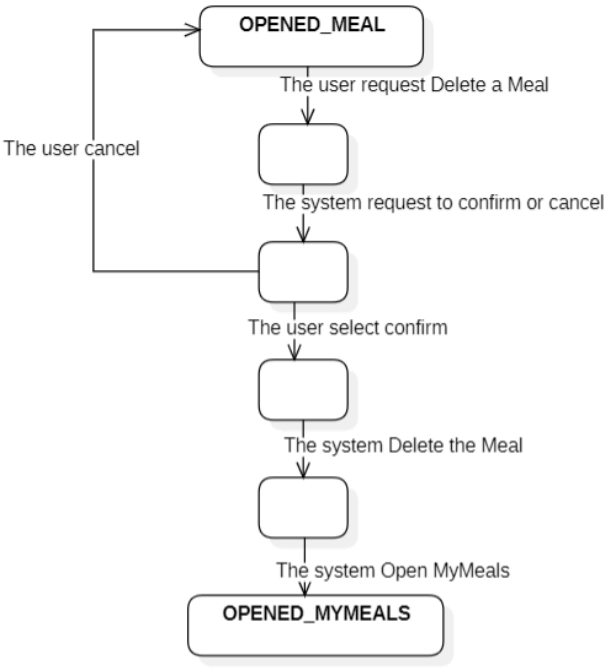
Update Meal


OPENED_MEAL
The user request to Edit the information
The system allow to enter the name and Category, request to save, and edit the Ingredient List
The user request to edit the Ingredient List
The system shows a list of Ingredients
The user enter the name and Category
The system allows to modify the quantity of an Ingredient, delete it and add a new one
The user request to add a new Ingredient
The user request to delete an Ingredient
The user enter the quantity of an Ingredient
The user request to edit the Ingredient list
The user request to delete an Ingredient
The user request to add a new Ingredient
The system request a confirmation
The user request to save
The user request to save the Ingredient list
The system show a list of ingredients
The user confirm the deletion
The user enter the quantity of an Ingredient
The system request to select any Ingredient and quantity
The user select an Ingredient and enter the quantity
The user request to save the Ingredients list
The user request to save the Ingredients list
The user request to add a new Ingredient
The system show Ingredients list
The system shows the Meal modified
The system show the Ingredients list
The system show the Ingredients list
OPENED_MEAL
The user enter the name and Categoty

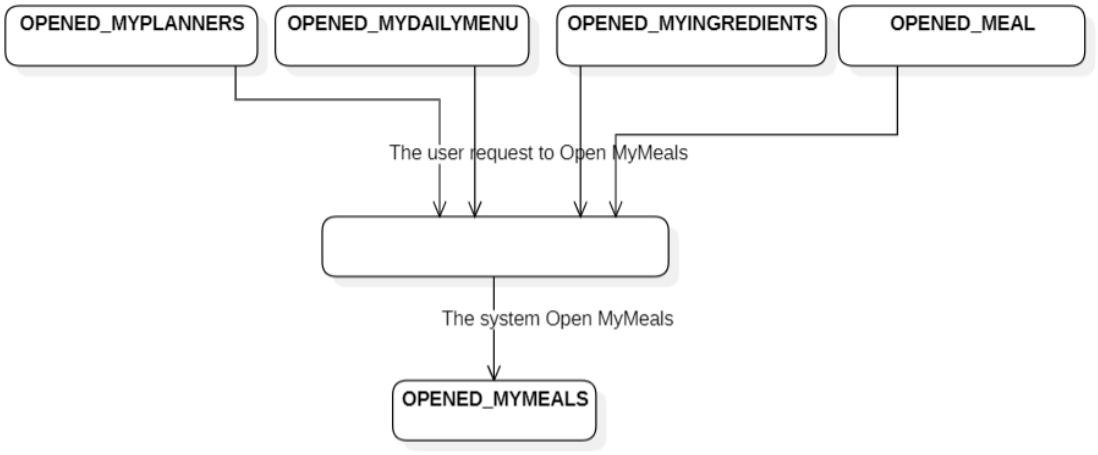
Open Meal



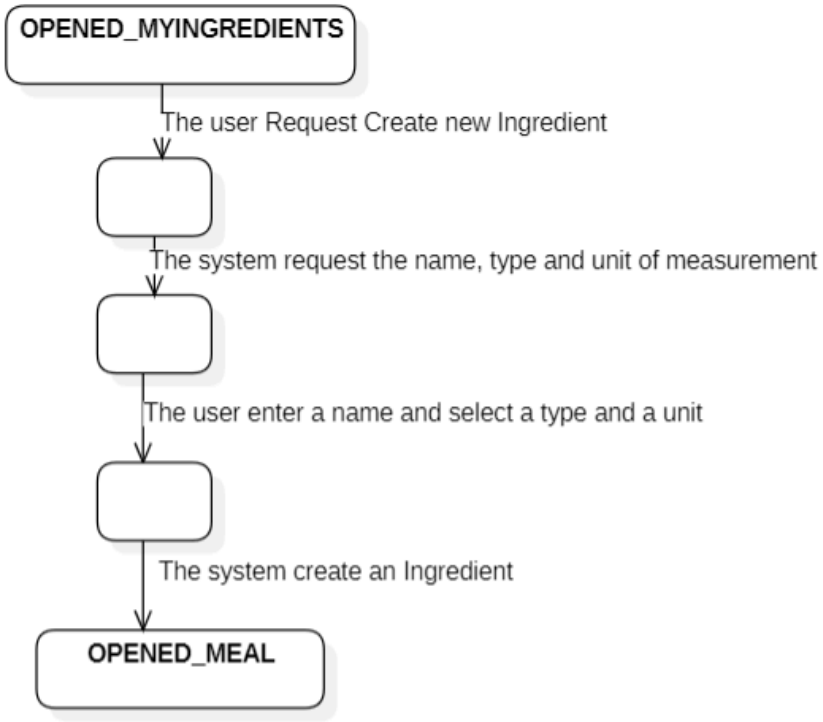
Delete Meal



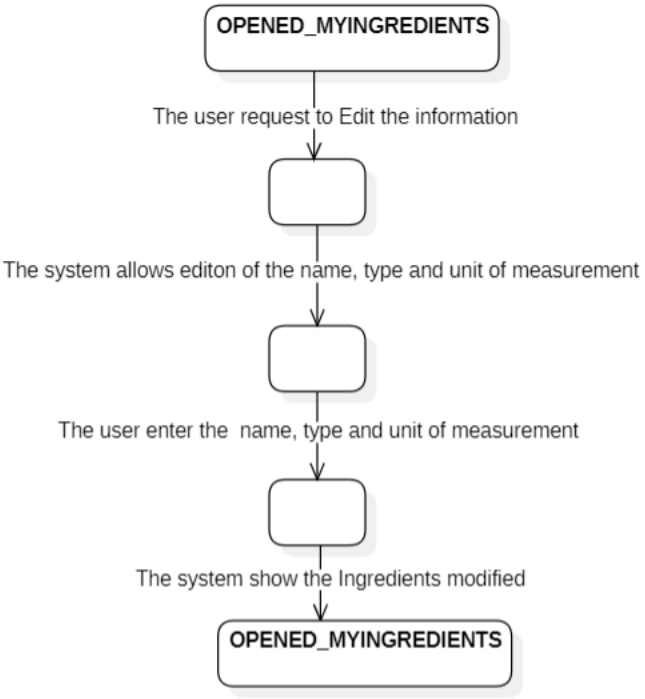
Open My Meals



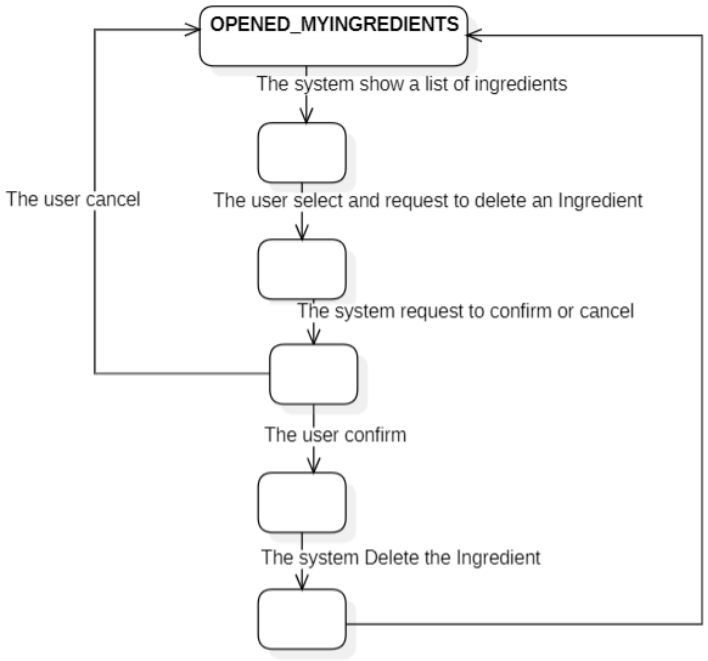
Create Ingredient



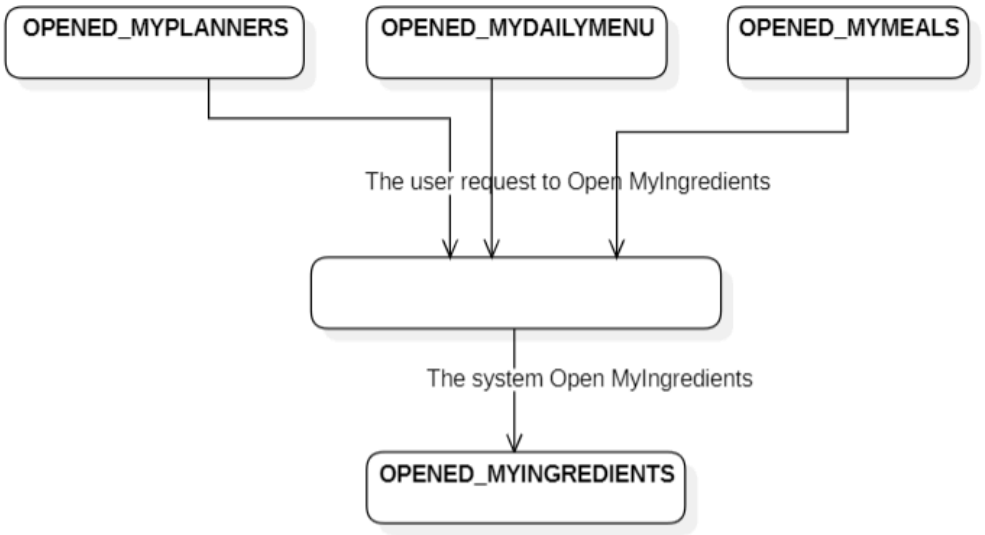
Update Ingredient



Delete Ingredient



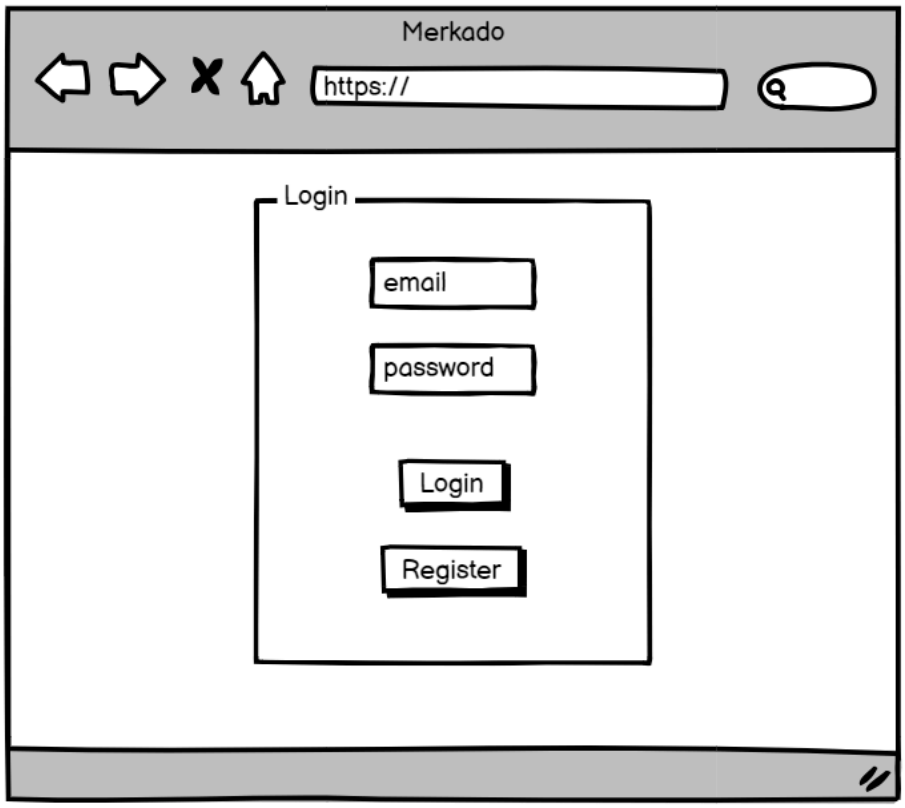
Open My Ingredients



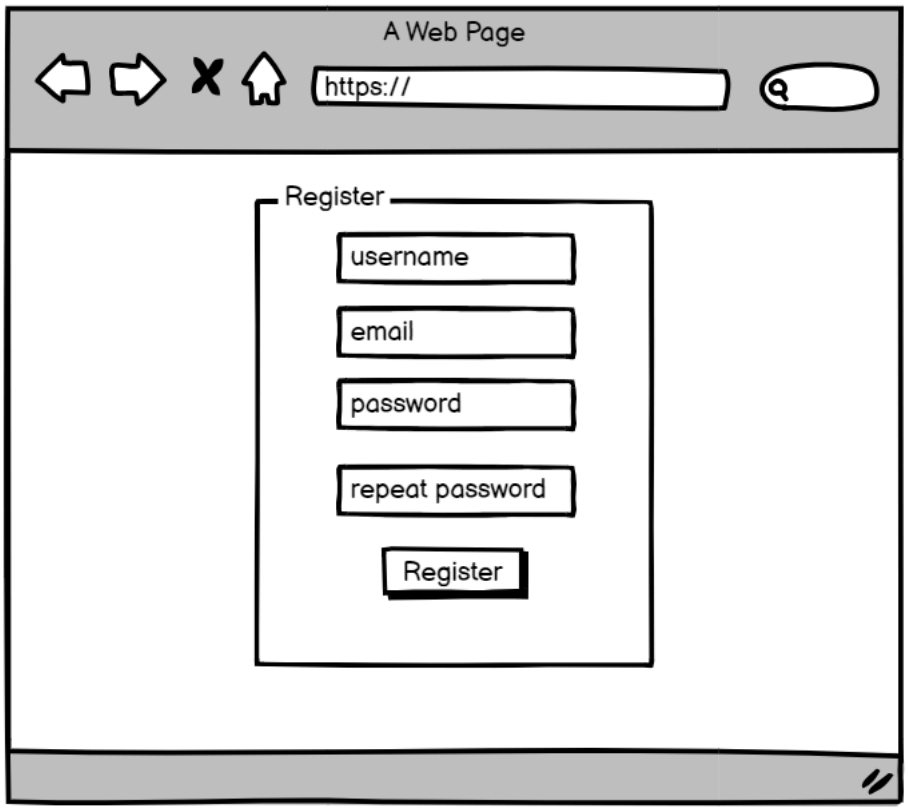
4. Prototipo de Interfaz de Usuario

A continuación se presenta el diseño físico que tendrá la interfaz de usuario de la aplicación. Estos bocetos reflejan la información requisitada en la especificación de casos de uso.

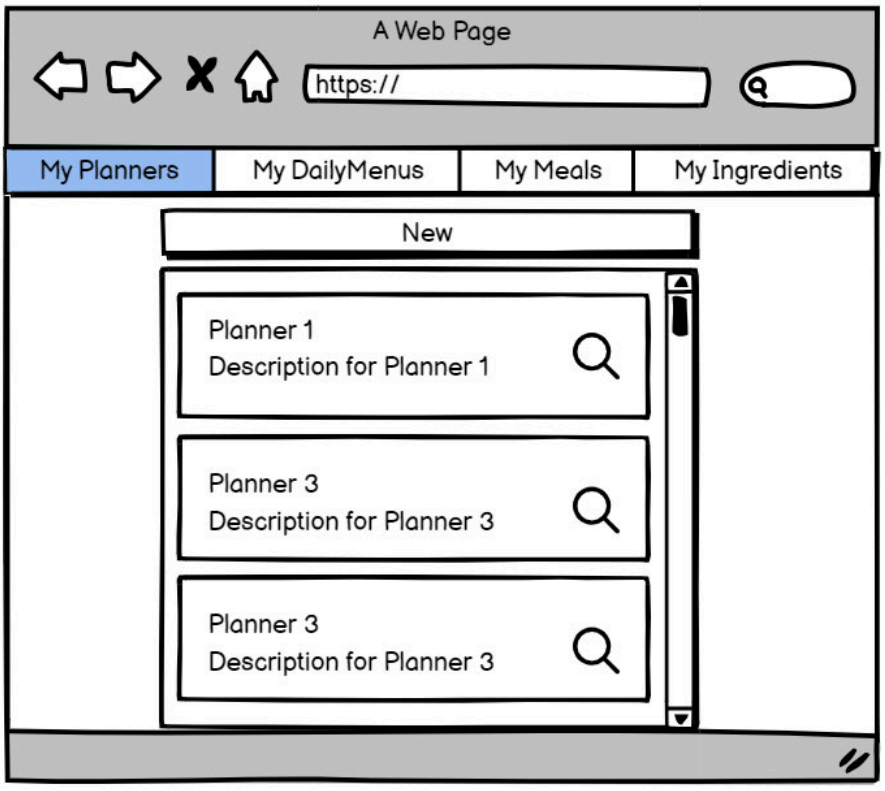
Login



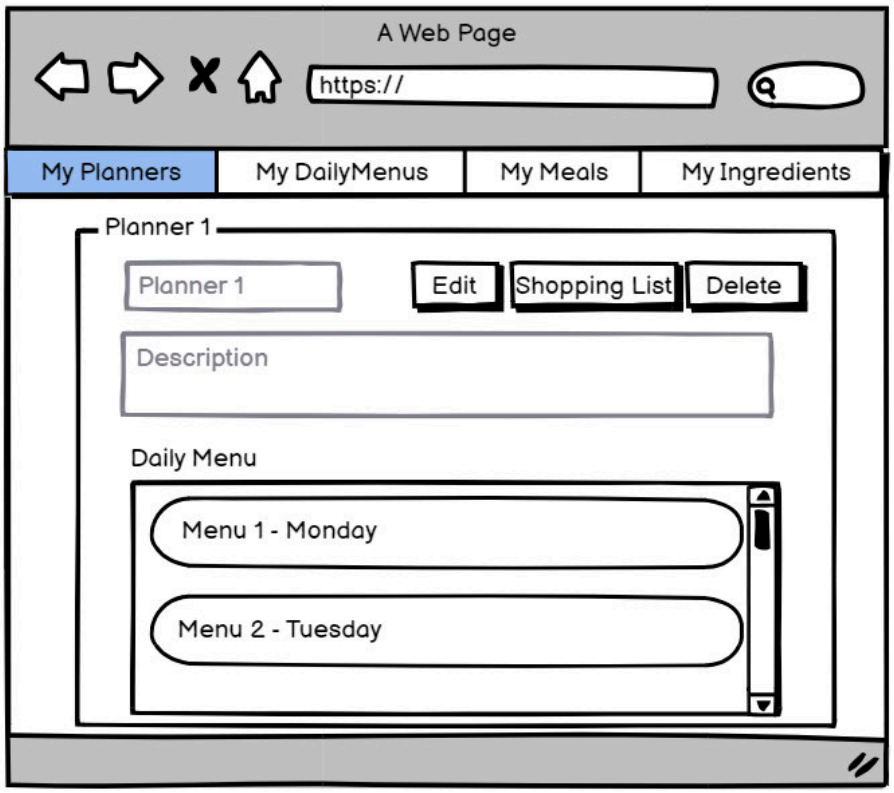
Register



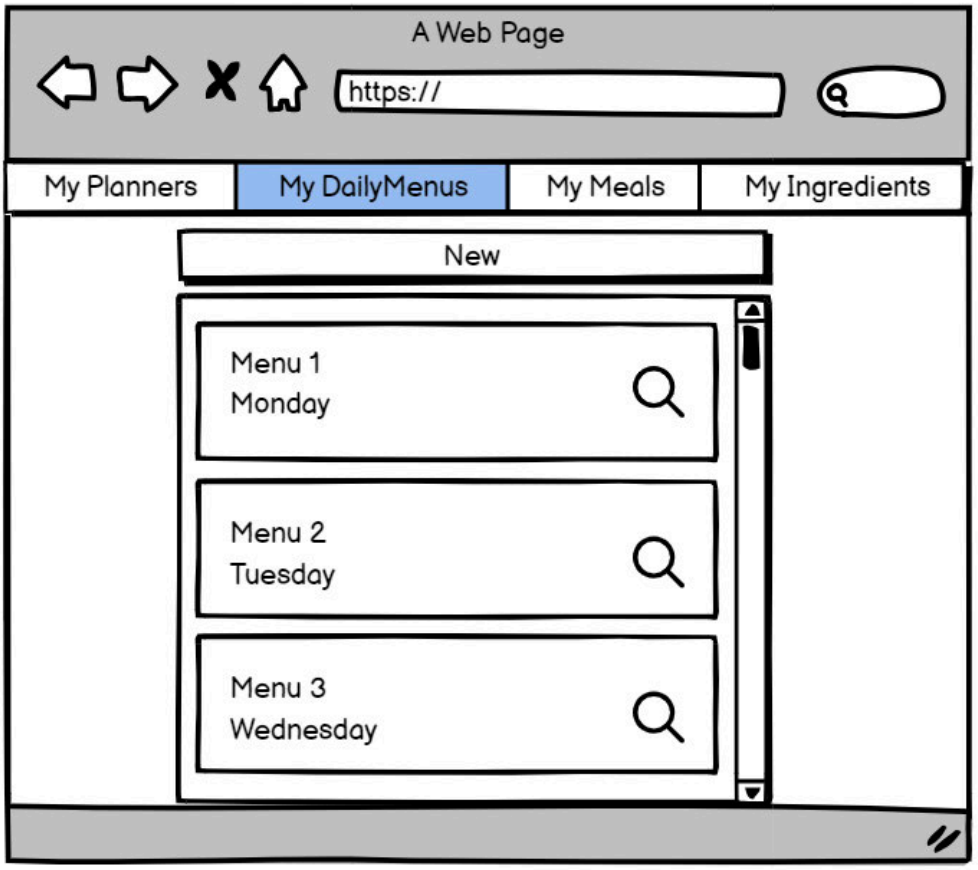
Open My Planners



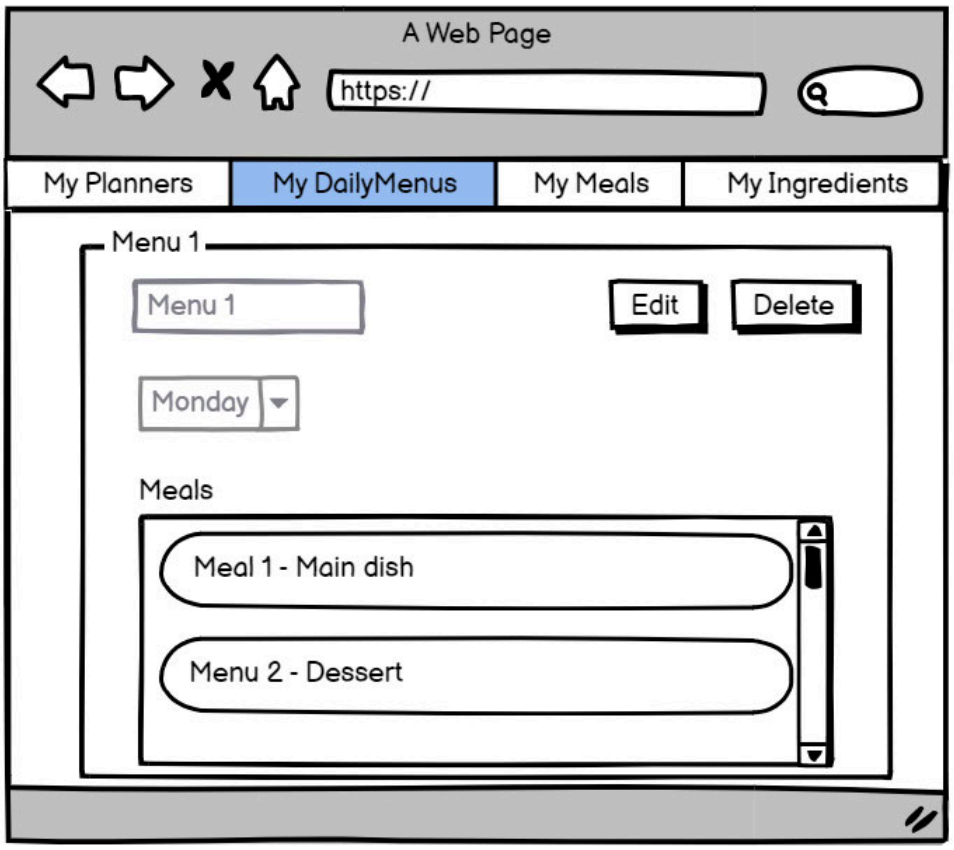
Open Planner



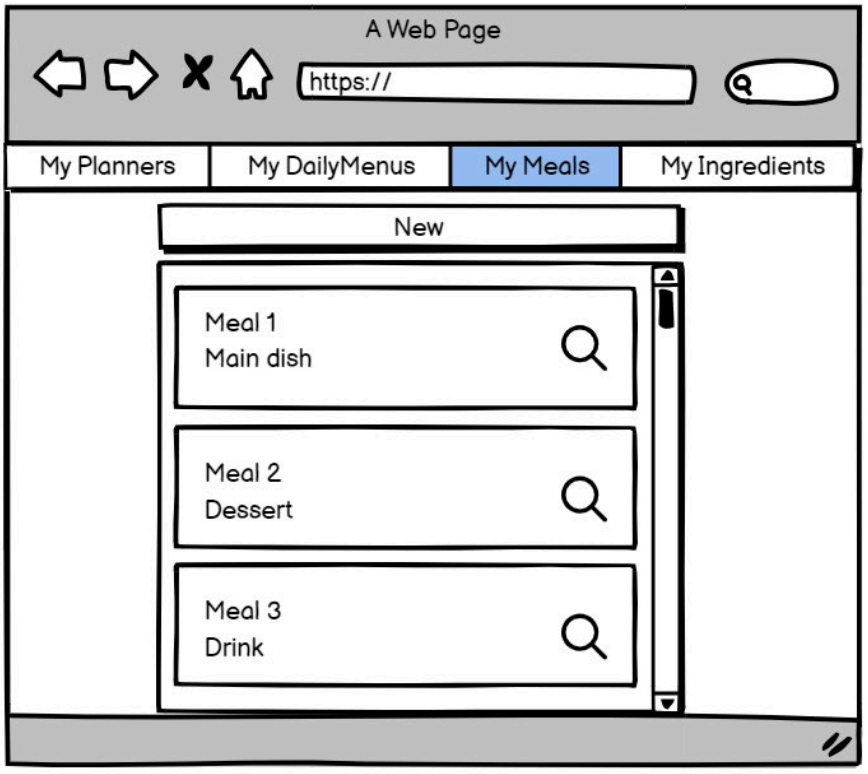
Open My Daily Menus



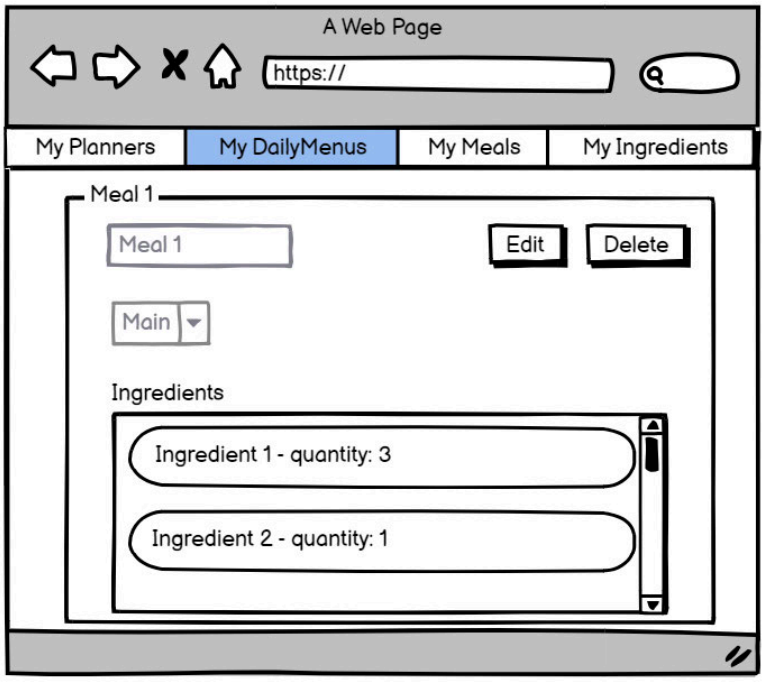
Open Daily Menu



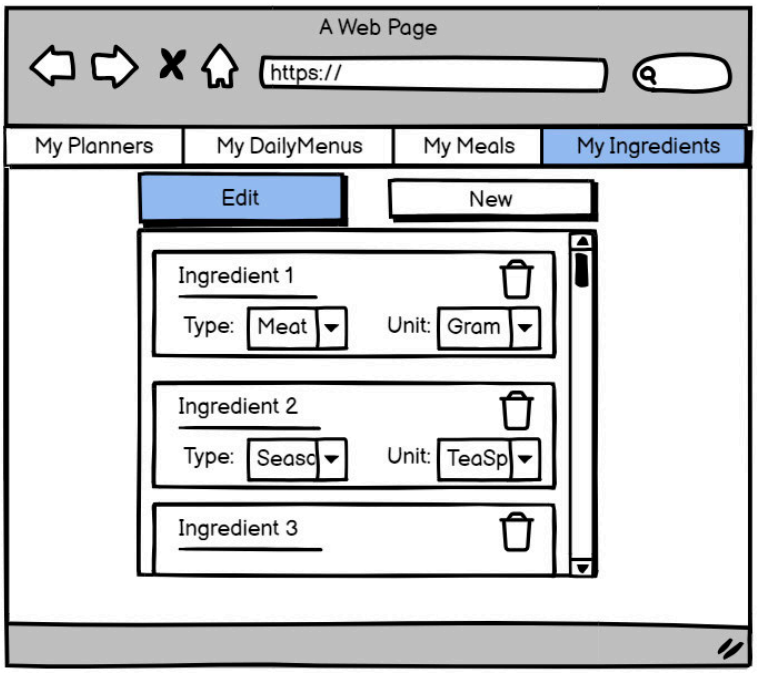
Open My Meals



Open Meal



Open My Ingredients



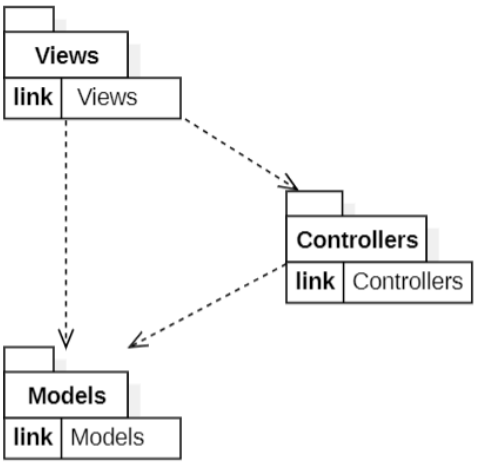
CAPÍTULO 3

Análisis

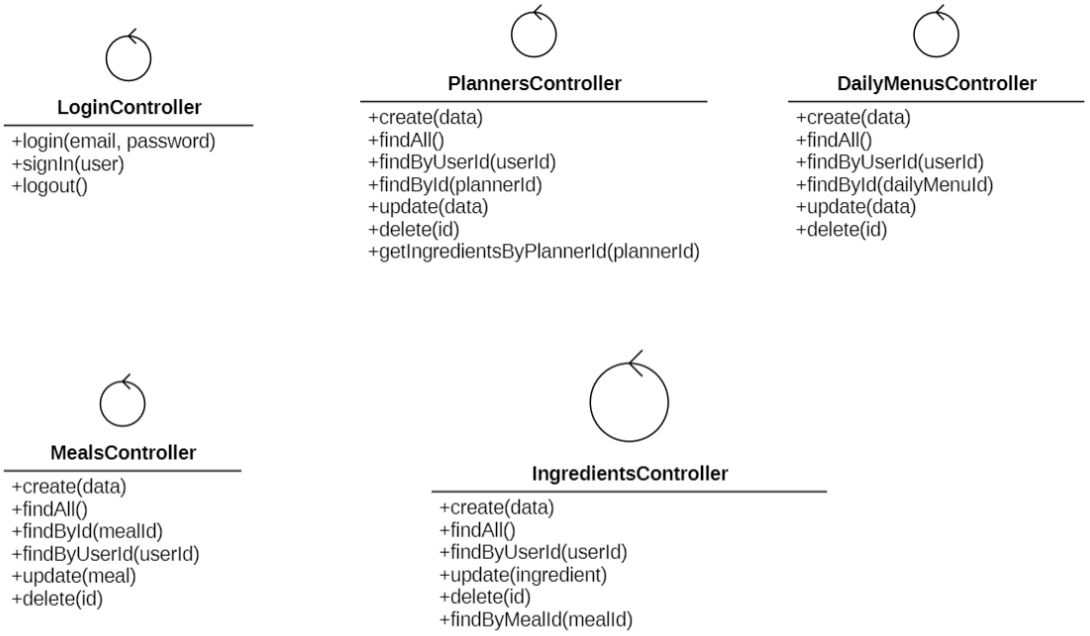
El objetivo de este capítulo es descomponer y estructurar los requisitos capturados para lograr una comprensión más precisa y detallada. Esto permitirá una interpretación clara de las necesidades del usuario y las partes interesadas, estableciendo una base sólida para las siguientes fases del desarrollo de la solución.

1. Análisis de la arquitectura

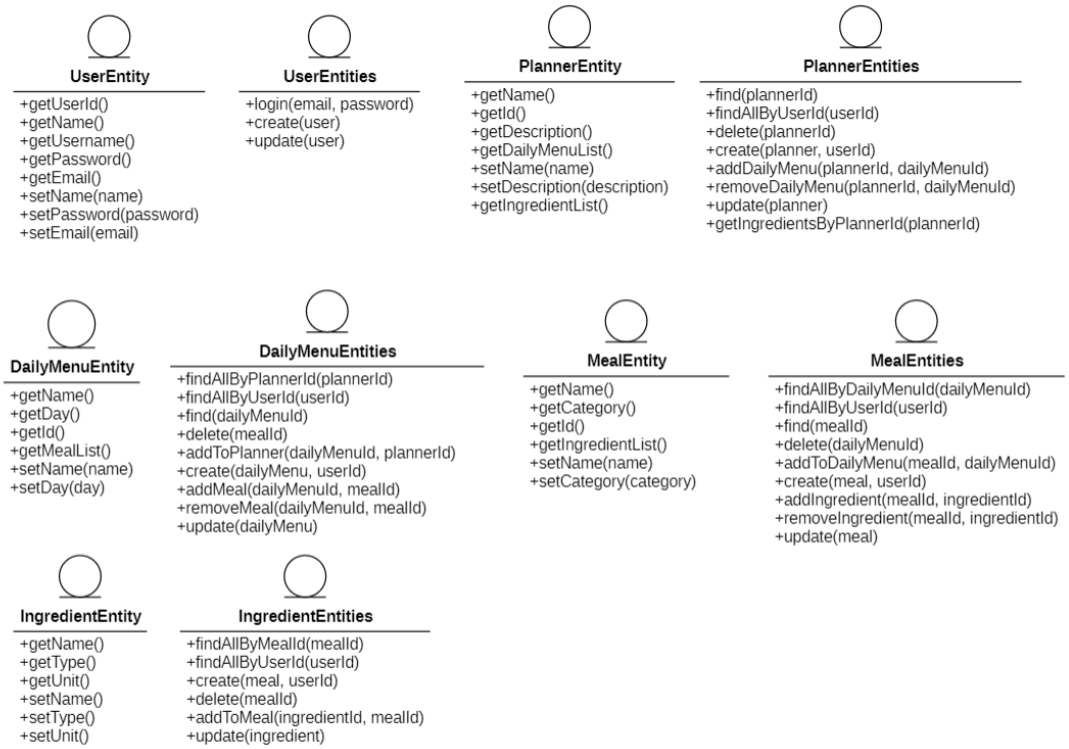
En este apartado se presenta el análisis de la arquitectura del sistema, que define la organización del software y los componentes estructurales que lo conforman.



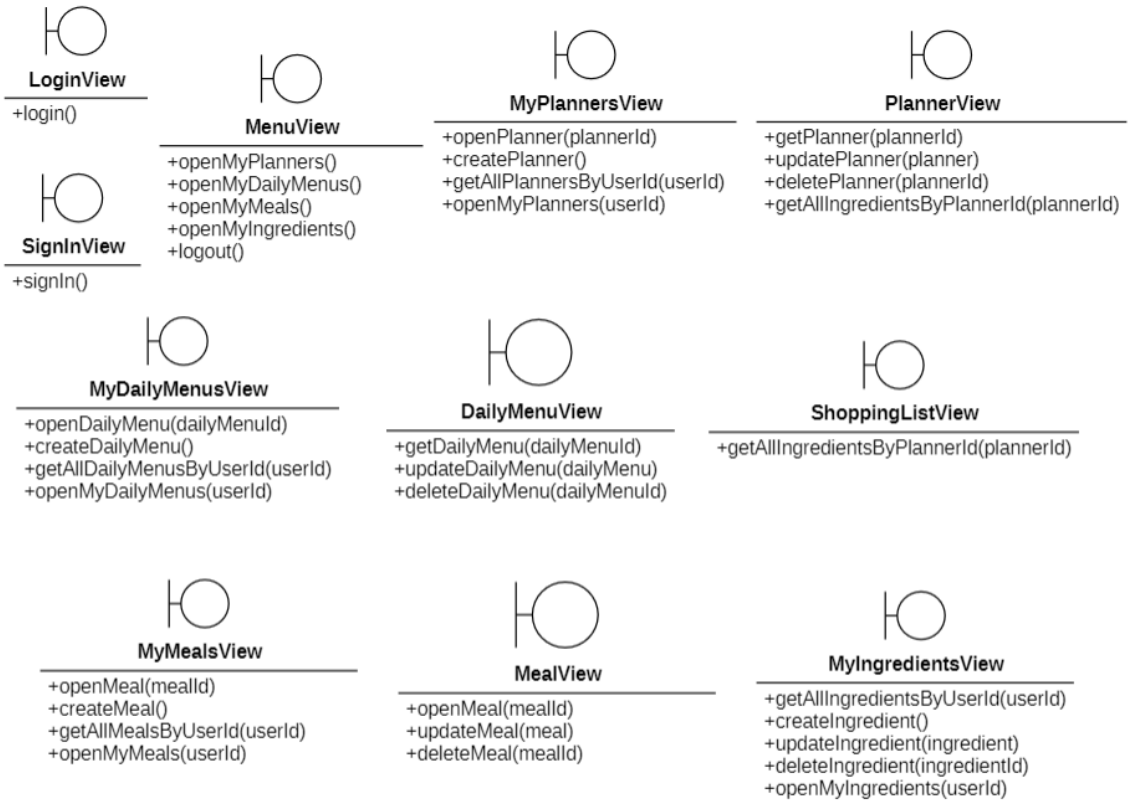
Controllers



Models



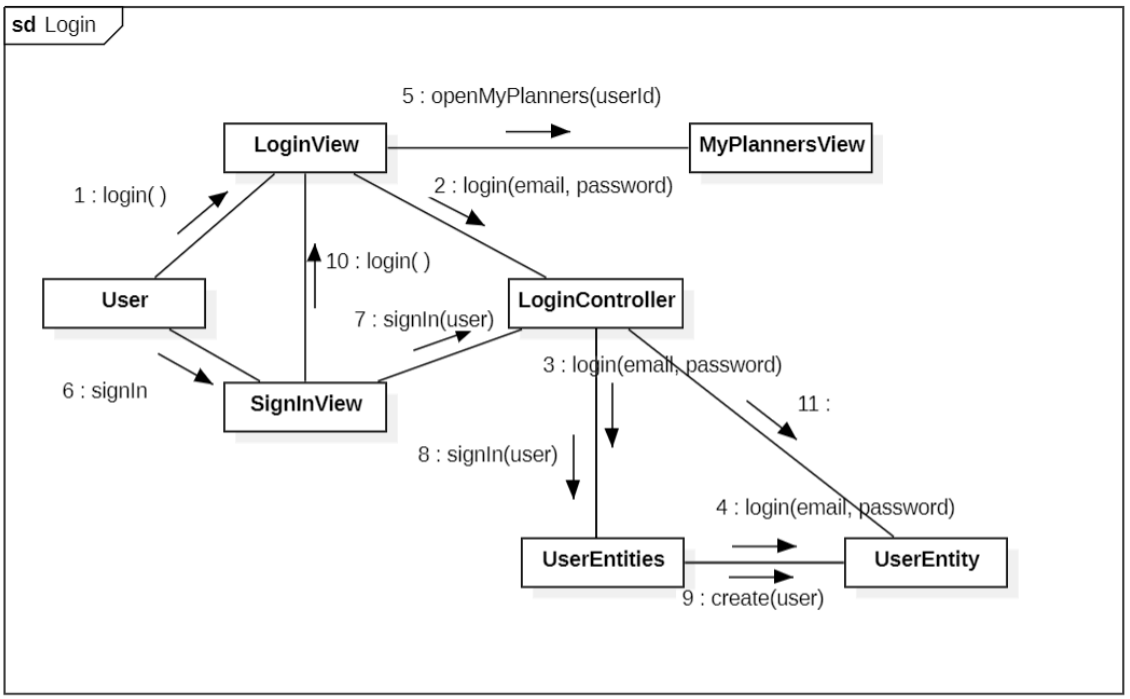
Views



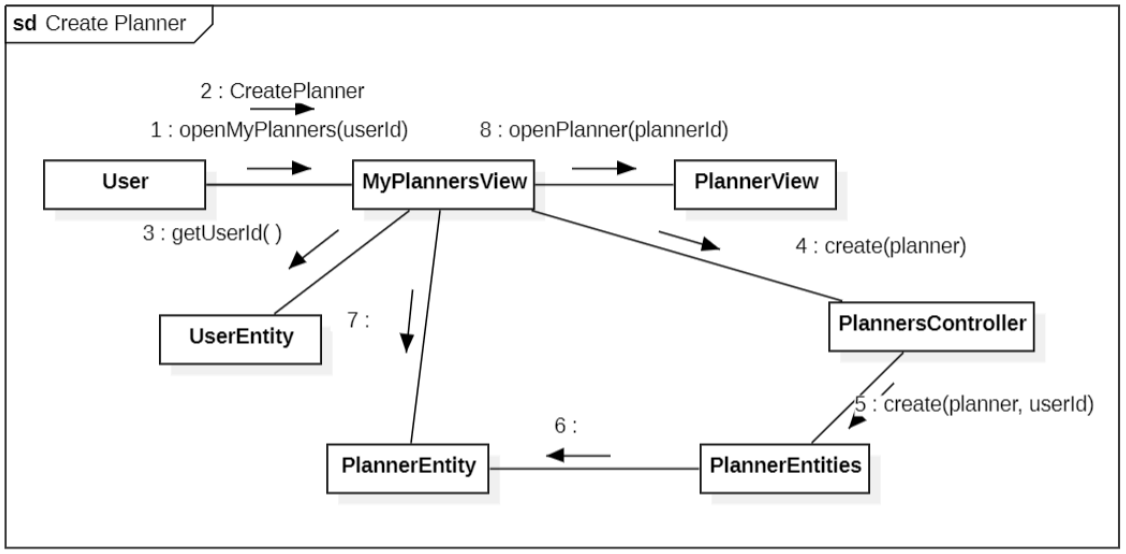
Análisis de Casos de Uso

El objetivo de este apartado es describir las interacciones entre los objetos de análisis mediante Diagramas de Colaboración, explicando cómo interactúan estos objetos entre sí.

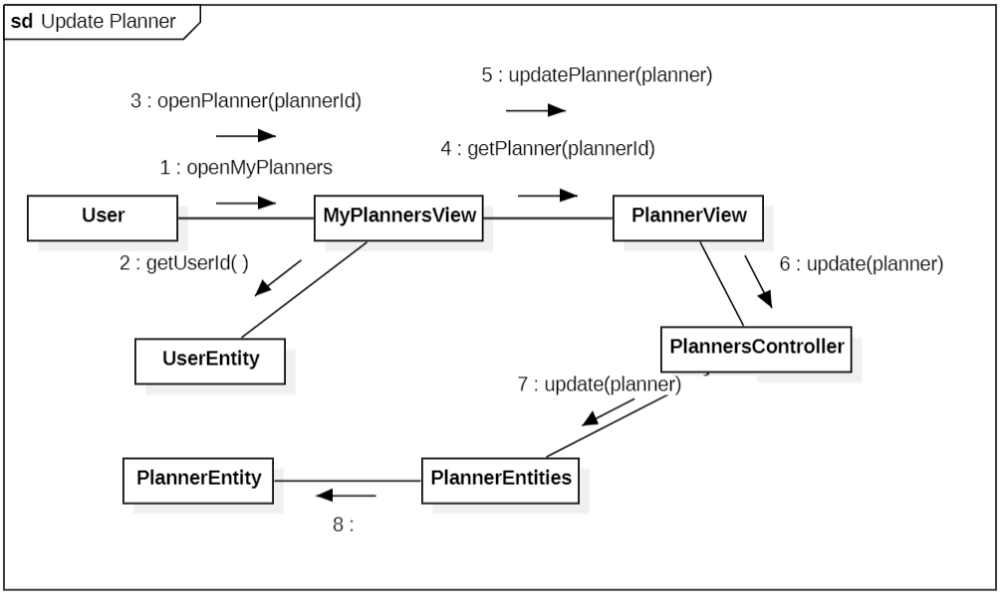
Login, Sign In



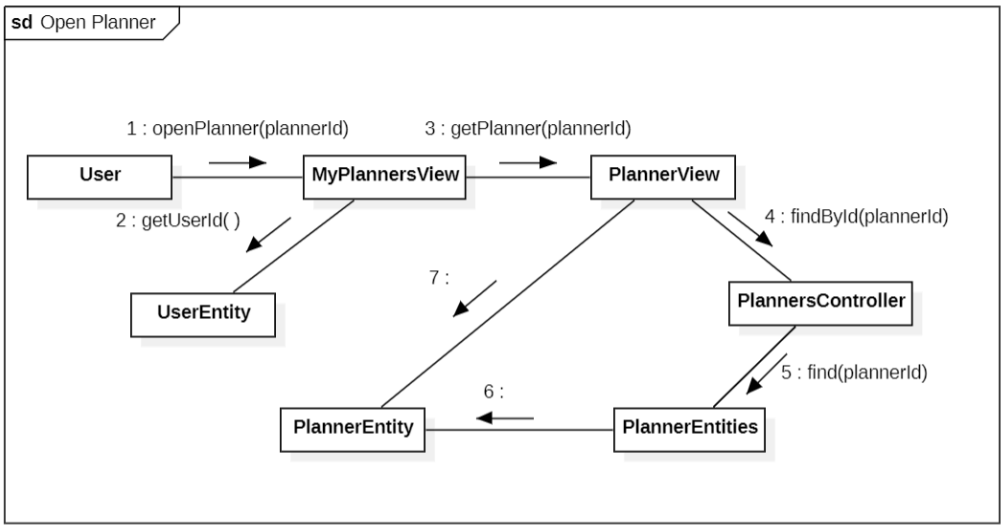
Create Planner



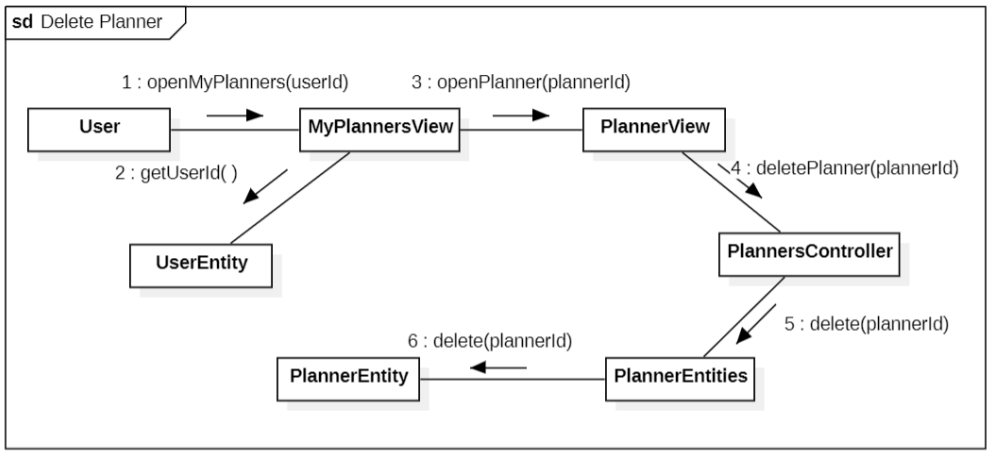
Update Planner



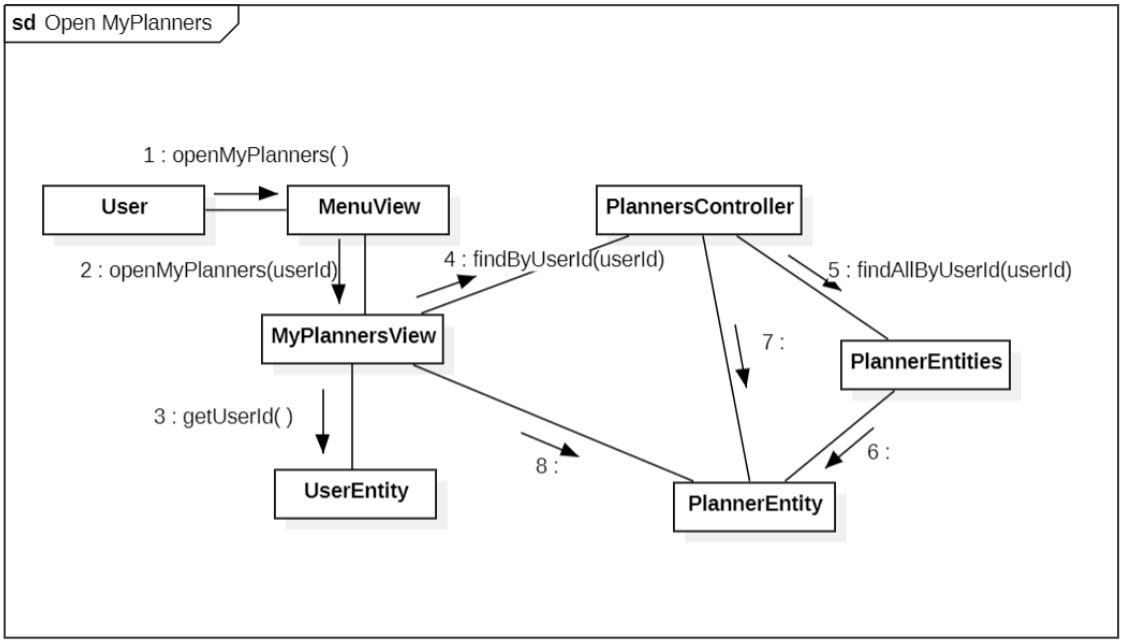
Open Planner



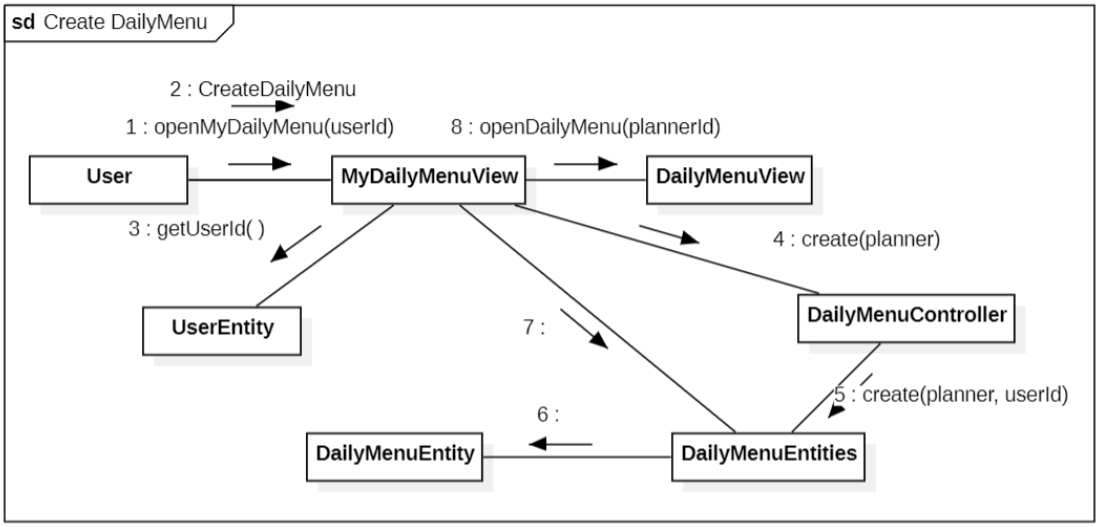
Delete Planner



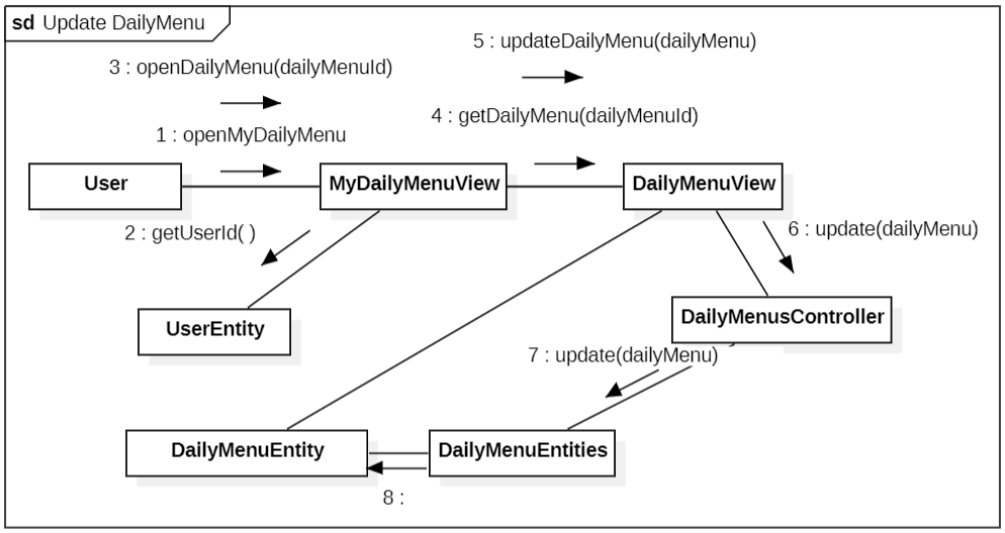
Open My Planners



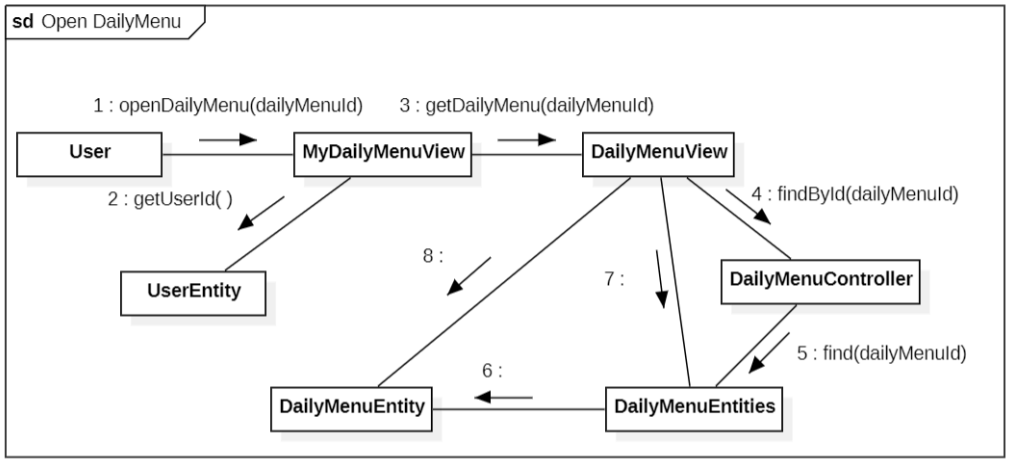
Create Daily Menu



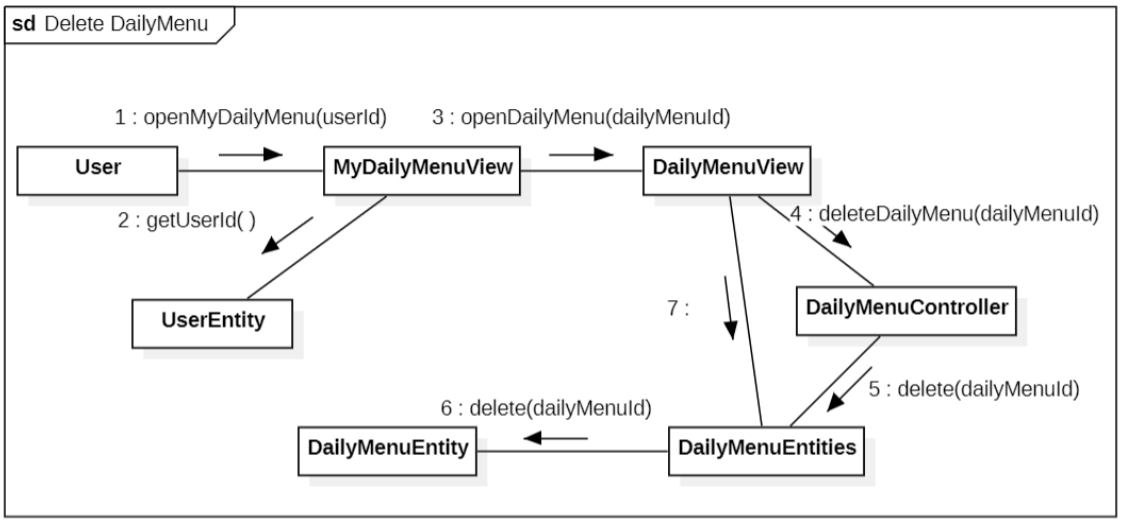
Update Daily Menu



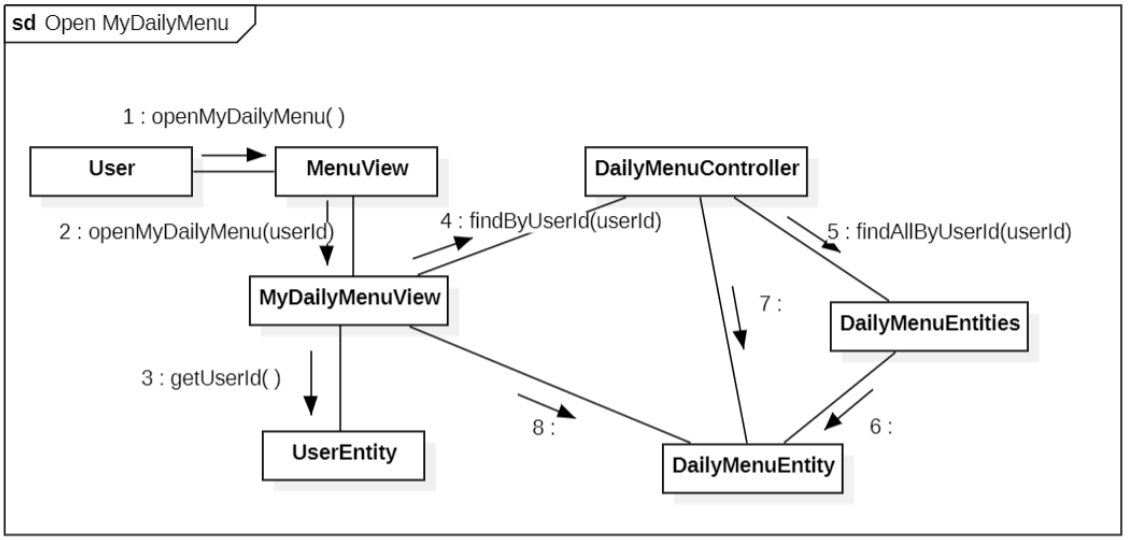
Open Daily Menu



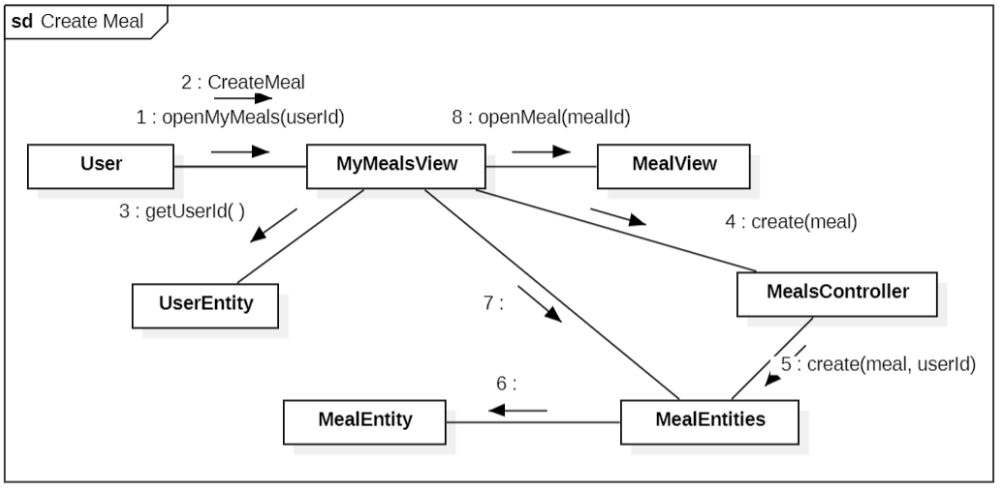
Delete Daily Menu



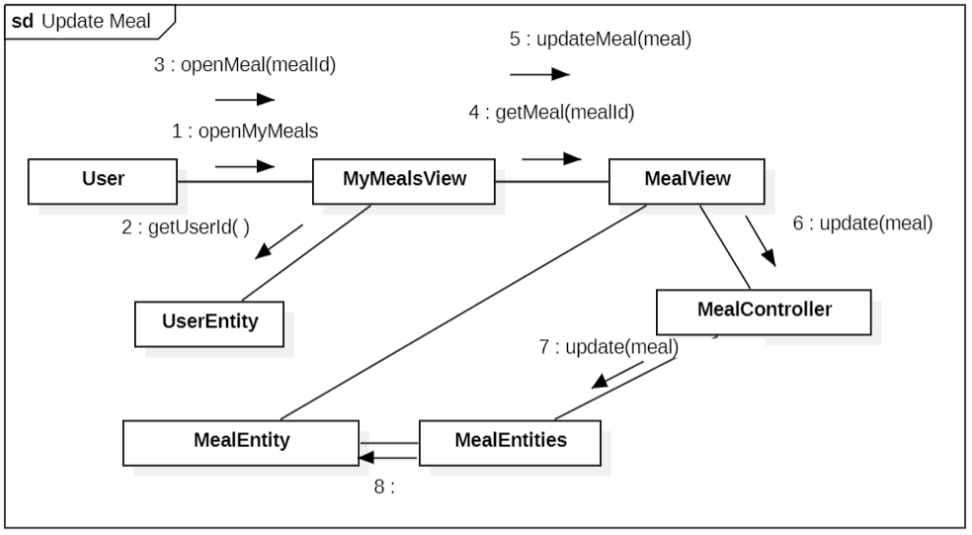
Open My Daily Menus



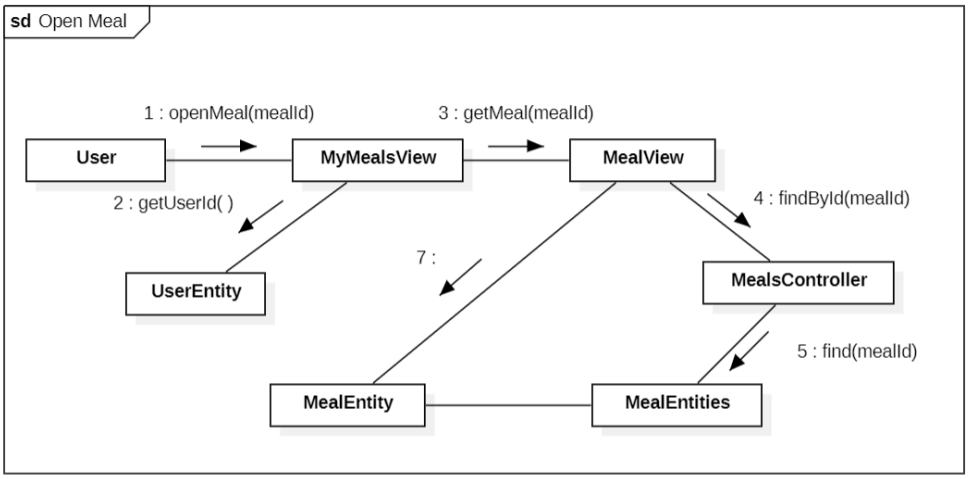
Create Meal



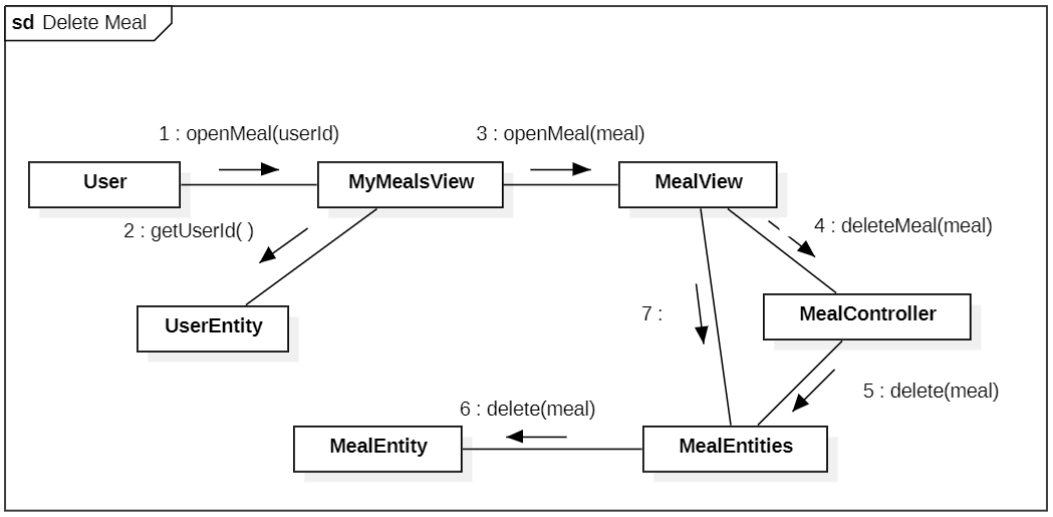
Update Meal



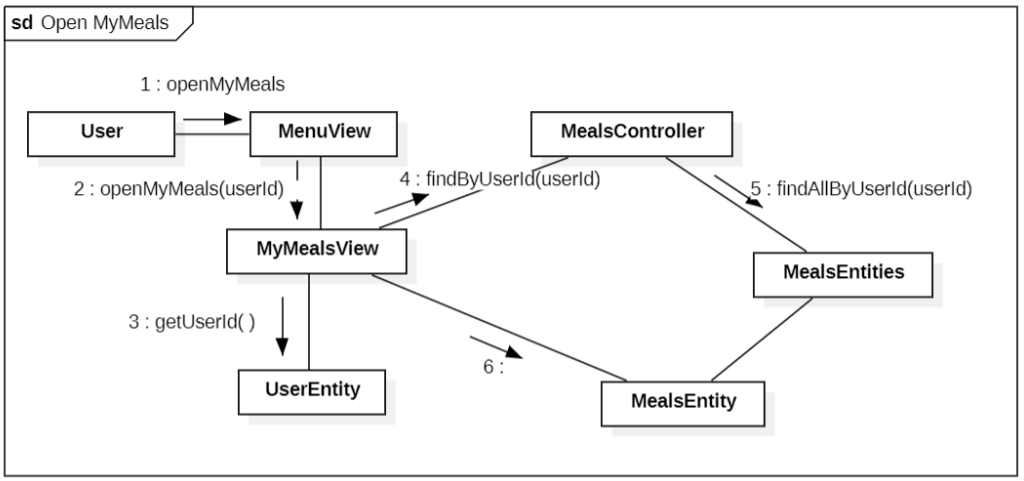
Open Meal



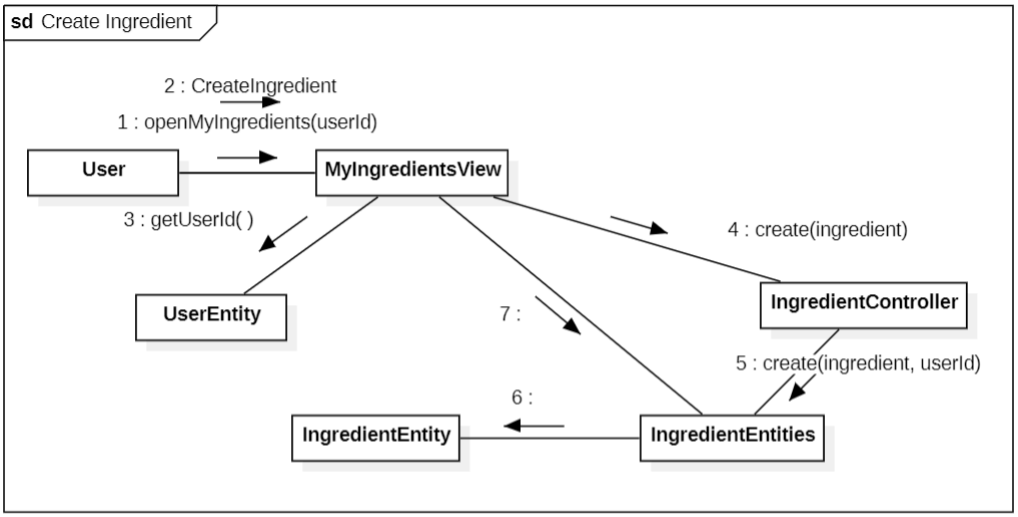
Delete Meal



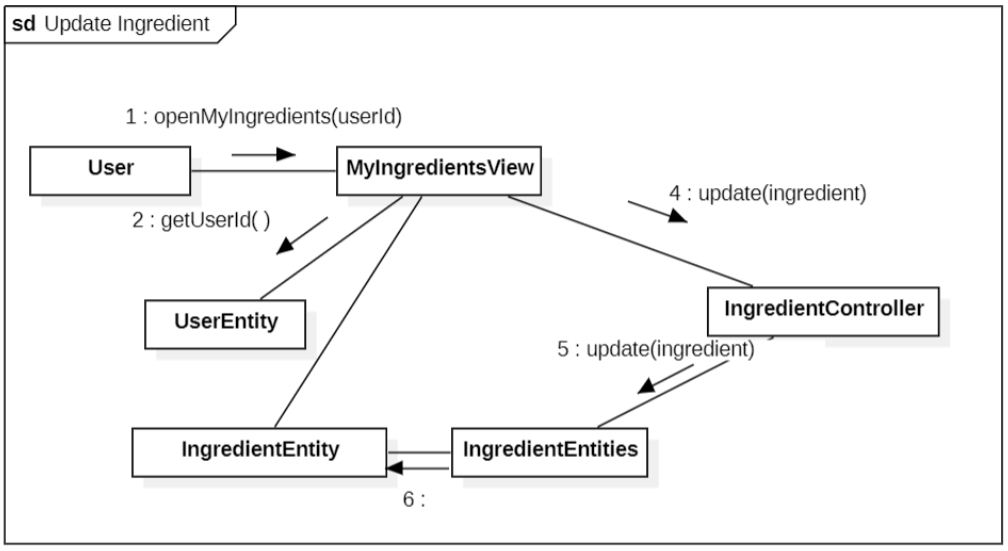
Open My Meals



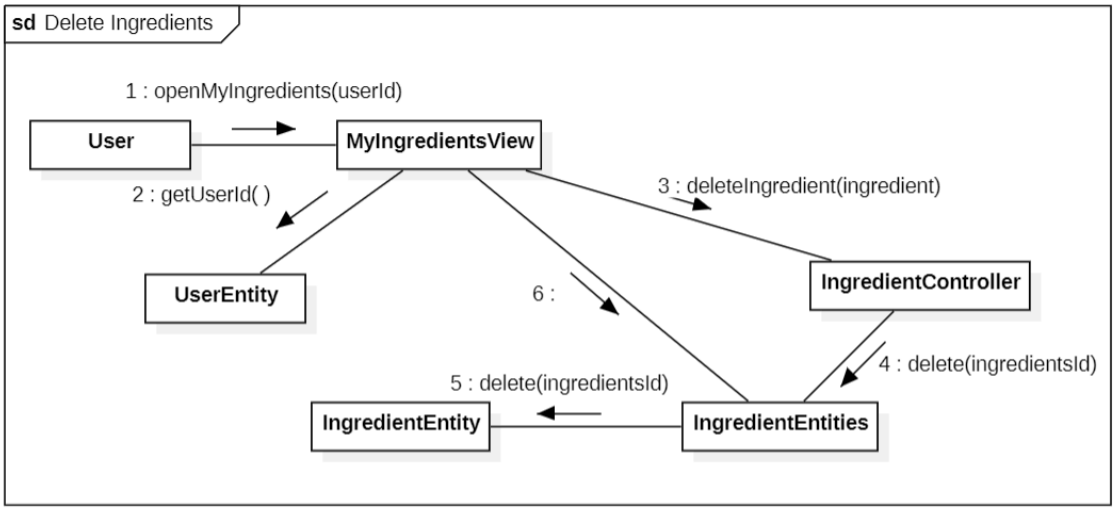
Create Ingredient



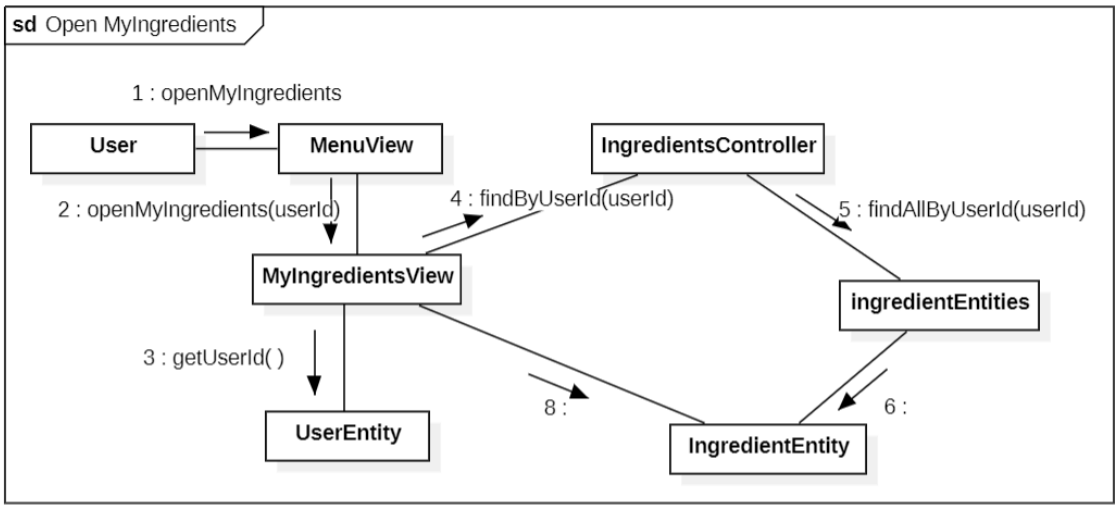
Update Ingredient



Delete Ingredient



Open My Ingredients

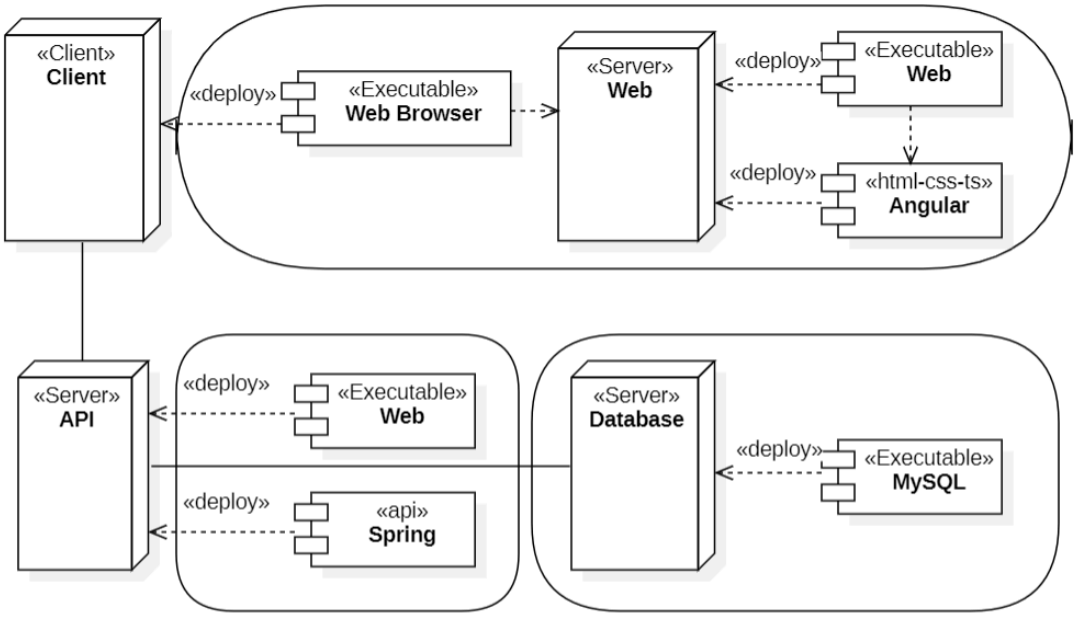


CAPÍTULO 4

Diseño

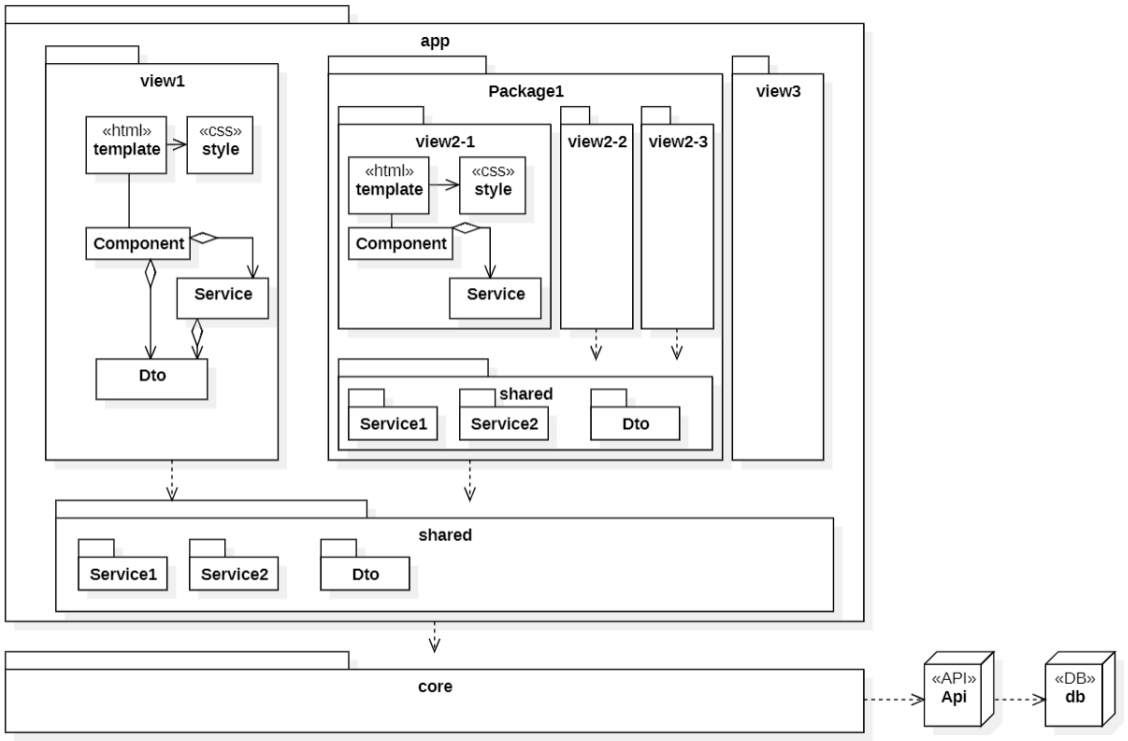
Diseño de la arquitectura

- Capa de presentación: en esta capa se encuentra el servidor web desarrollado usando Angular. Desde el navegador web ejecutado en el cliente se realizan peticiones HTTP a la API que estará montada en otro servidor dedicado.
- Capa de negocio: esta capa tendrá el servidor de la API desarrollada con Spring.
- Capa de datos: se usará un servidor de datos con MySQL



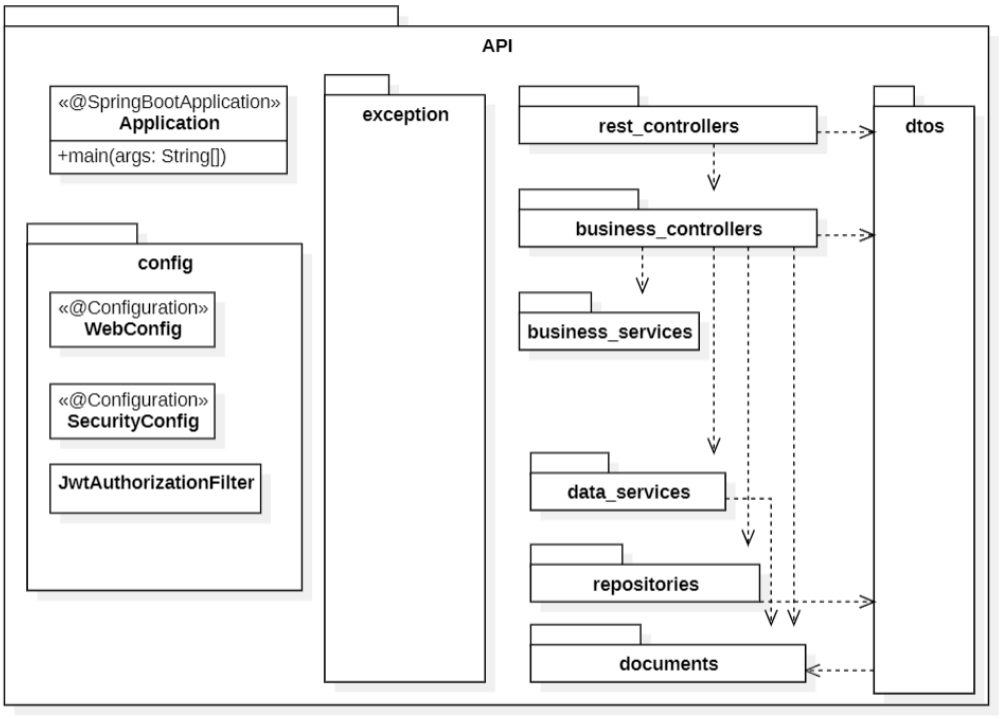
Capa de presentación.

A continuación se detalla la organización de paquetes de esta capa. Según la arquitectura recomendada por Angular se organiza de acuerdo a pantallas.



Capa de negocio

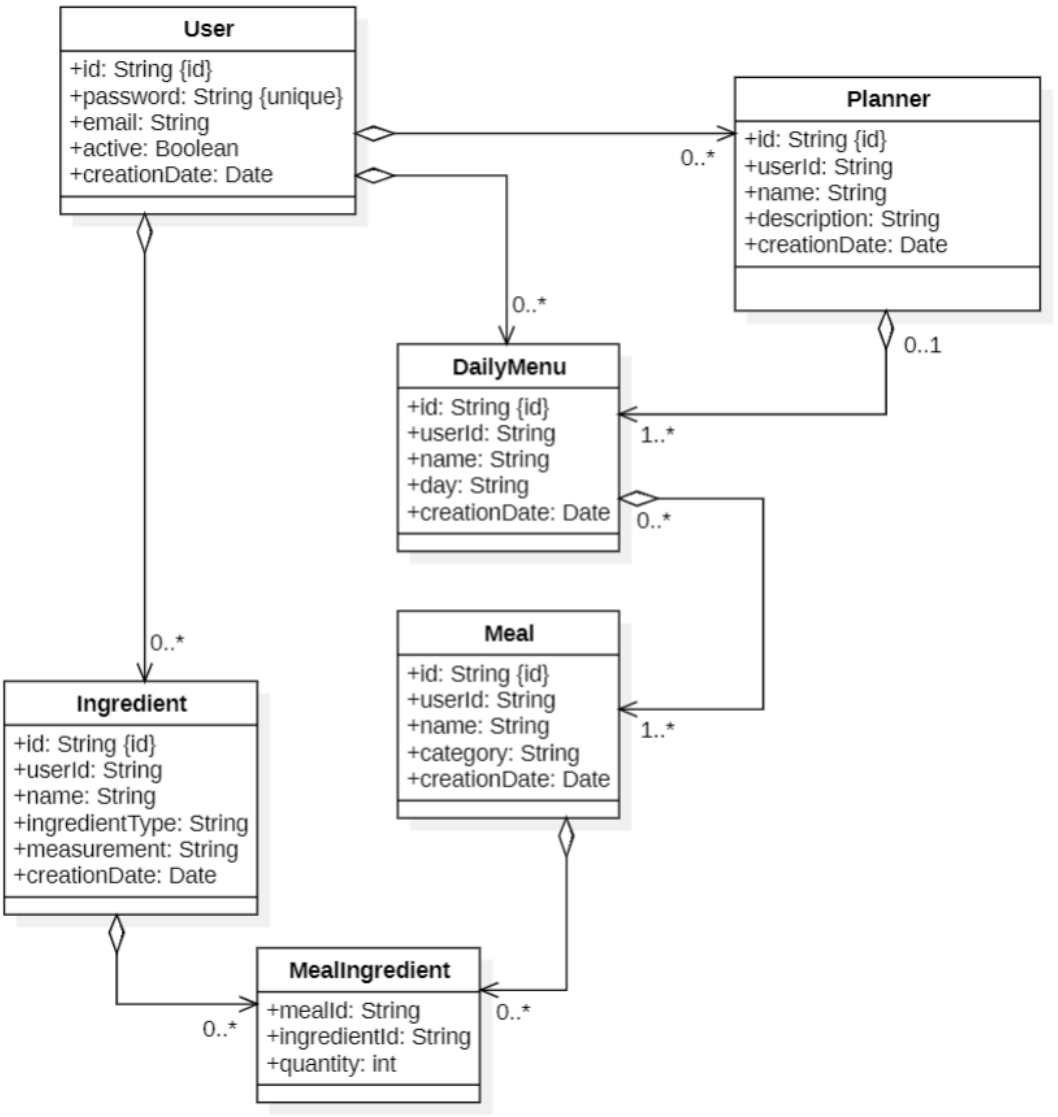
Para la capa de negocios se seguirá la arquitectura por capas aprendida en el Máster.



Capa de datos

La capa de datos presenta las distintas entidades que se crearán para almacenar los datos de esta aplicación así como también sus atributos y el tipo de cada uno de ellos.

Además se muestra la relación entre estas entidades. Para el diseño de esta aplicación se tendrá en cuenta que las relaciones de muchos a muchos se expresan con tablas intermedias que almacenan el identificador único de cada tabla.

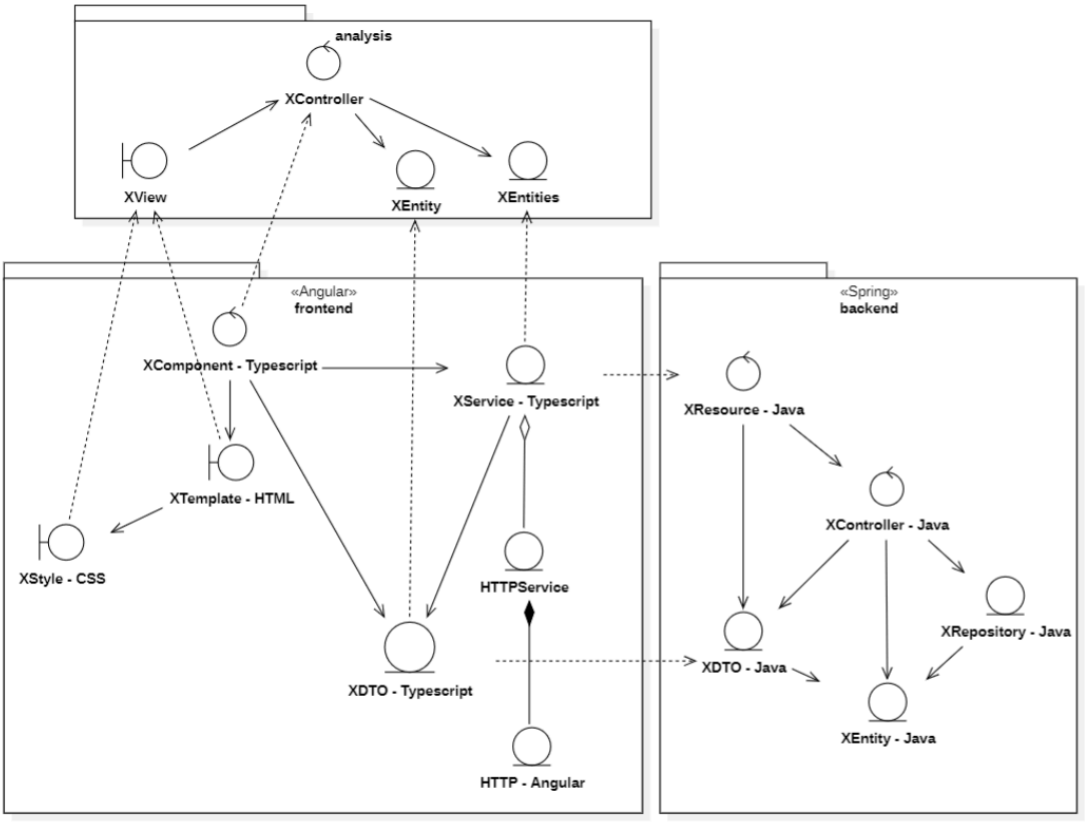


Diseño de Casos de Uso

El siguiente gráfico muestra la interacción entre cada una de las capas y cómo se reflejan en cada caso de uso en general.

Para cada caso de uso se seguirá la siguiente estructura, cada vista(XView) será representada por por una plantilla HTML y una hoja de estilos en Angular, estas vistas pertenecerán a un componente que refleja cada uno de los controladores desarrollados en el apartado de análisis. Los controladores se comunicarán por medio de servicios con la capa de negocio usando DTOs para el envío y lectura de datos, así cada servicio representa el conjunto de entidades (XEntities).

Por el lado de la capa de negocio, se contará con recursos que atiendan a cada uno de los servicios implementados en la presentación. Estos recursos se comunicarán con la capa de datos (MySQL) para realizar las consultas solicitadas y devolverán los datos a la capa de presentación.



CAPÍTULO 5

Implementación

En este capítulo se detallarán las herramientas elegidas para el desarrollo de la aplicación:

Ecosistema de desarrollo

- Sistema de control de versiones: Git
- Pruebas unitarias y de integración: JUnit
- Versión de Java para la capa de negocios: Java 11
- Versión de Spring para la capa de negocios : Spring 2.6
- Versión de Angular para la capa de presentación:
- Motor de Base de datos: MySQL 8
- Herramienta de despliegue:
- Herramienta de análisis de calidad de software: Sonar Cloud

Calidad interna del software

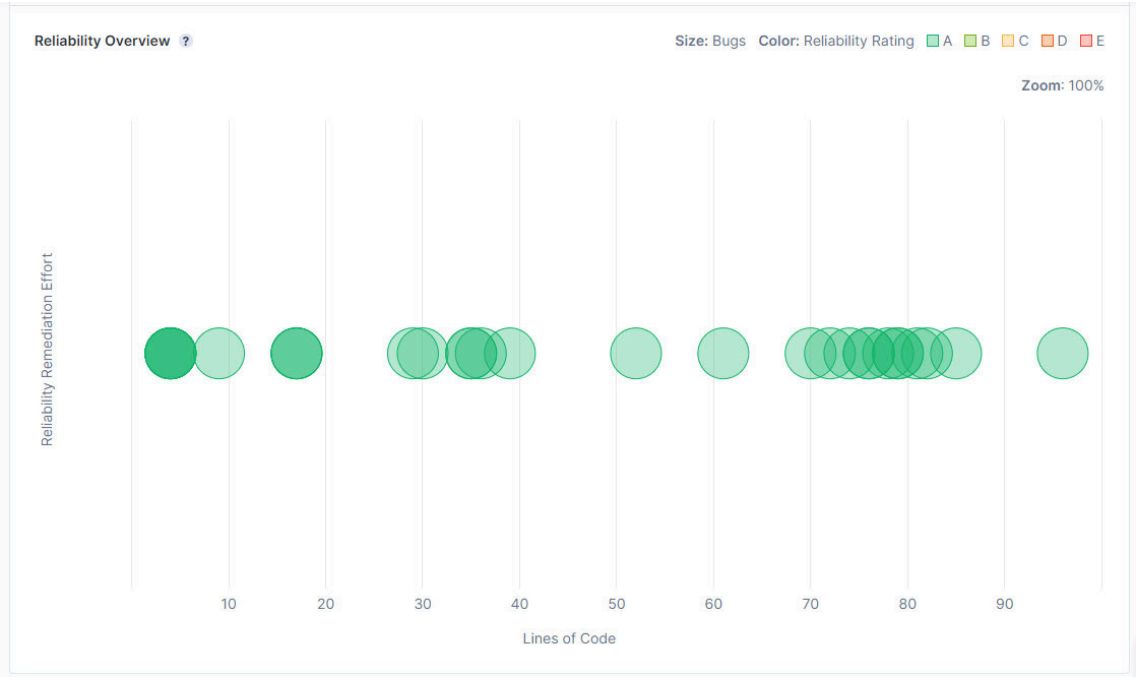
Se han seguido los estándares de diseño detallados a continuación:

- Evitar comentarios.
- Elegir nombres descriptivos.
- Evitar el código muerto, fragmentos de código no usados.
- Principio YAGNI, implementación de características solo si son necesarias para los requerimientos actuales y no cuando se prevé que se necesiten.
- Principio KISS, la simplicidad debe ser el objetivo clave del diseño, evitándose cualquier complejidad innecesaria.
- Alta Cohesión, las funciones deben hacer solo una cosa.
- Bajo Acoplamiento, los métodos y clases deben ser lo más independientes posibles unas a otras.
- Tamaño pequeño, métodos y clases no deben exceder:
    - 10-15 líneas para un método.
    - 200-500 líneas para una clase
    - 80-120 caracteres por línea de código.
- Evitar el código sucio por clases perezosas.
- Evitar el código sucio por atributos temporales.
- Evitar el código sucio por una librería incompleta.
- Evitar el código sucio por obsesión por tipos primitivos.
- Evitar el código duplicado.

Para medir lo anteriormente mencionado se ha utilizado la herramienta Sonar Cloud, a continuación se muestran distintas gráficas las cuales miden la fiabilidad, mantenibilidad y duplicidad de código.

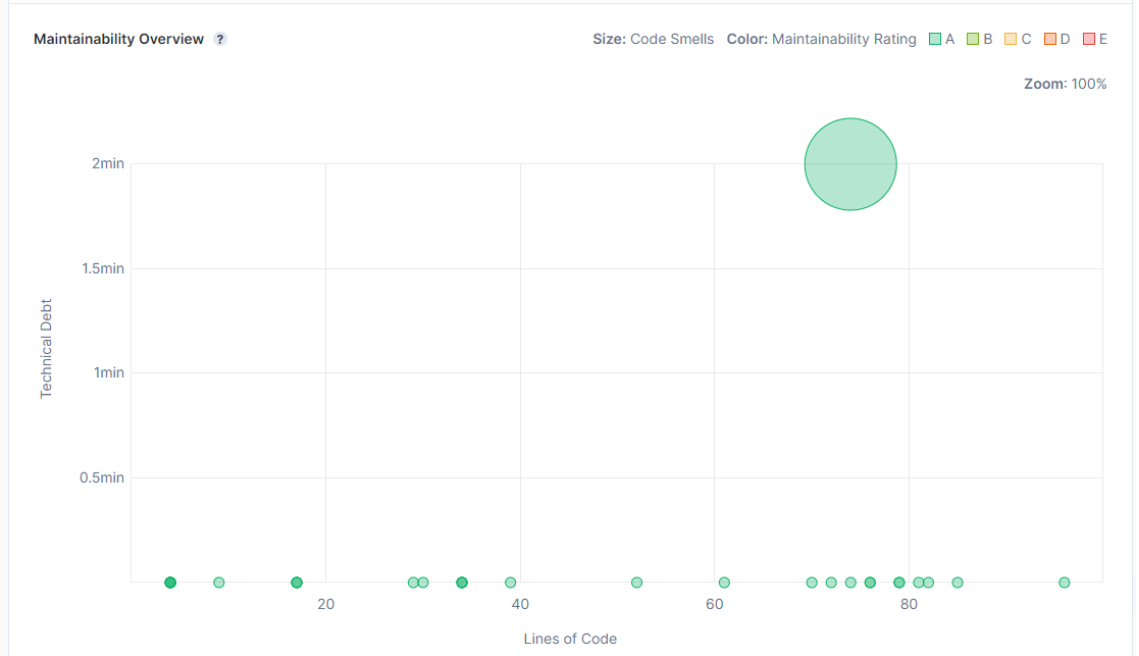
Fiabilidad

La gráfica muestra los problemas del código donde el comportamiento podría ser diferente al esperado. Como se puede observar la valoración de cada parte del código está entre A y B, por lo que se puede deducir que el código es altamente fiable.



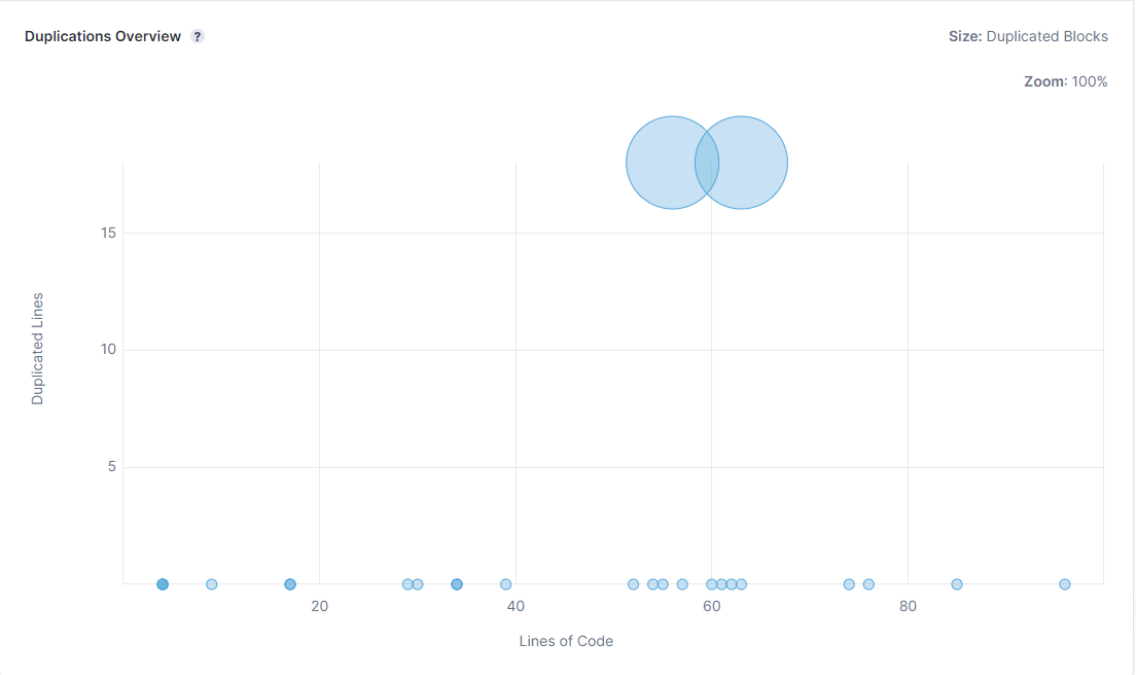
Mantenibilidad

La gráfica muestra el código que sería difícil de modificar, la posición de cada burbuja expresa el tiempo que demoraría realizar algún cambio. Como se puede observar la mayoría de puntos se muestran en la línea más baja, lo que significa que el código es altamente mantenible.



Duplicidad

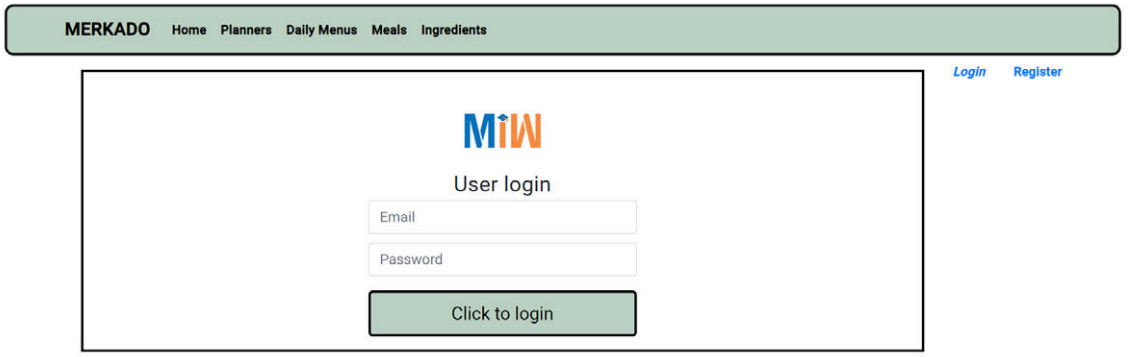
La siguiente gráfica muestra la duplicidad del código, donde la posición de cada burbuja expresa la cantidad de código duplicado y el tamaño de las burbujas el volumen de bloques duplicados. Como se puede observar, la mayor cantidad de código presenta muy poca duplicidad.



Calidad externa del software

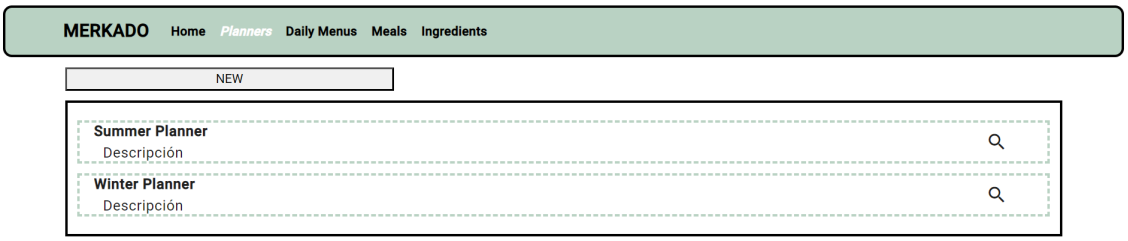
Se han cumplido los estándares de calidad externa del software, como la funcionalidad y la usabilidad. A continuación, se presentan diversas pantallas de la aplicación como muestra de esta calidad.

Login

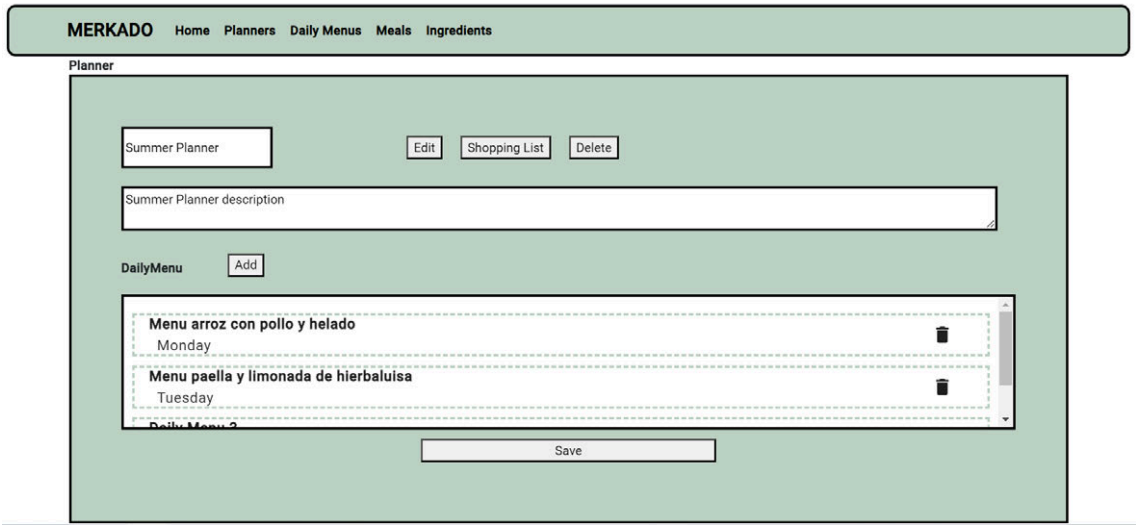


Register

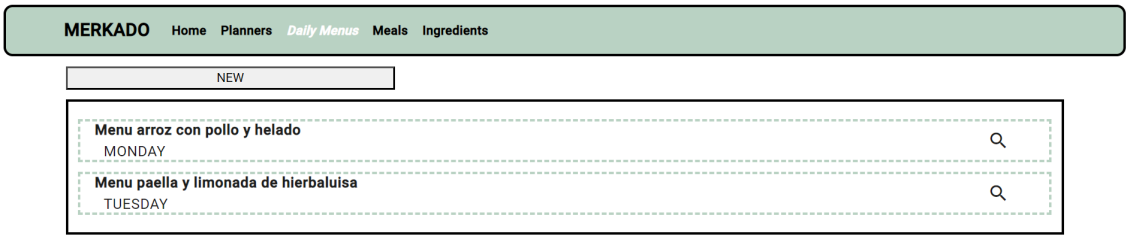
My Planners



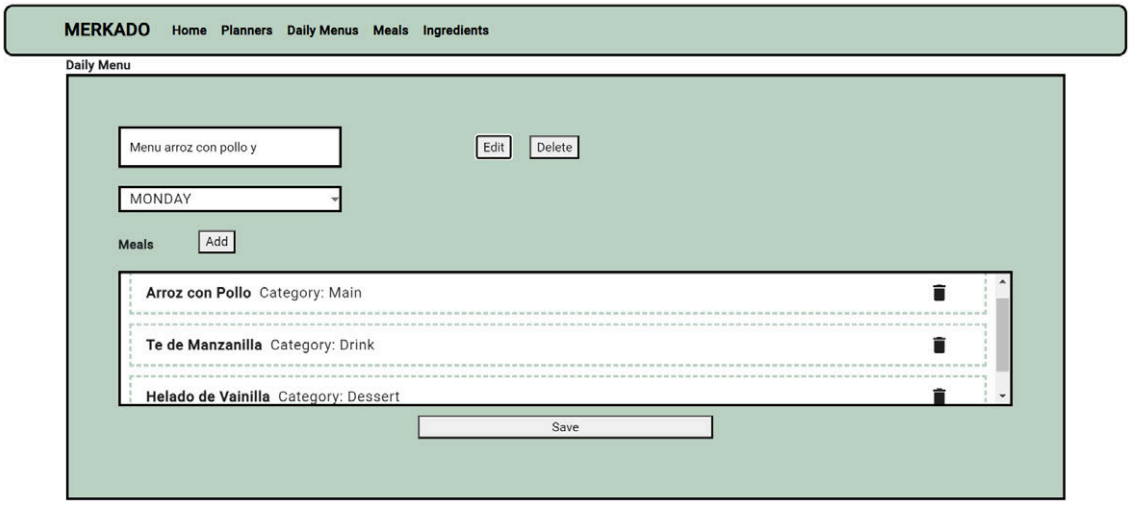
Open Planner



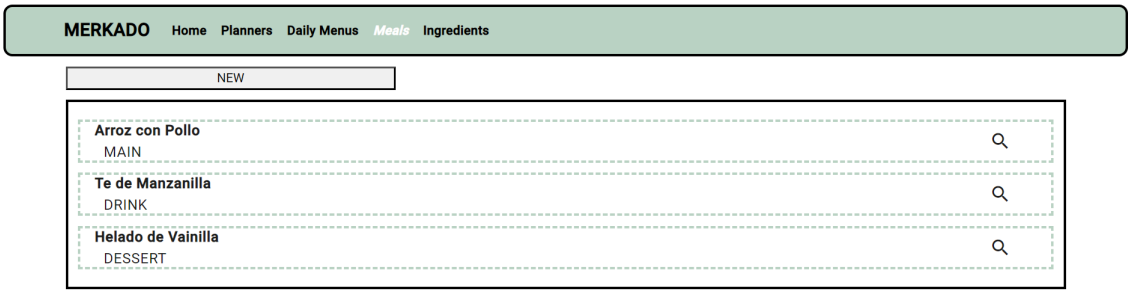
My Daily Menus



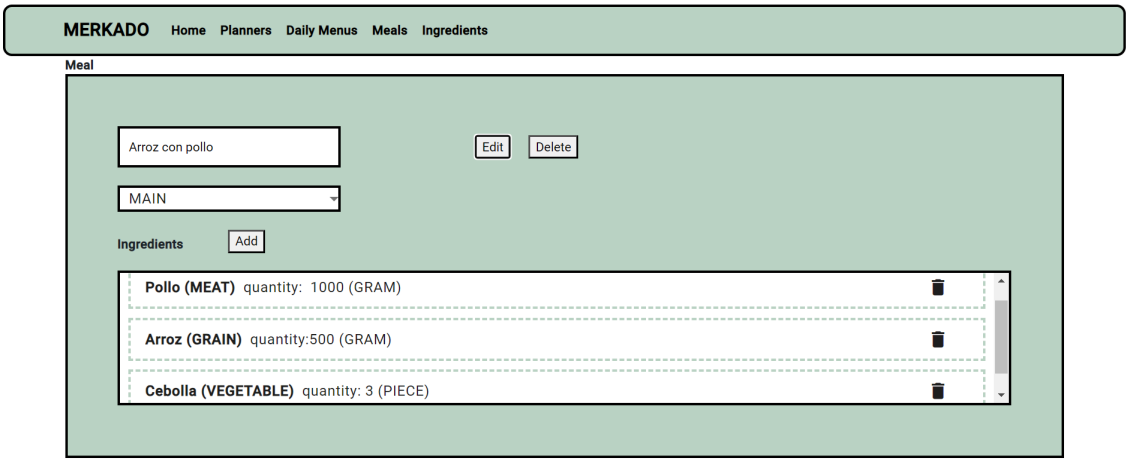
Open Daily Menu



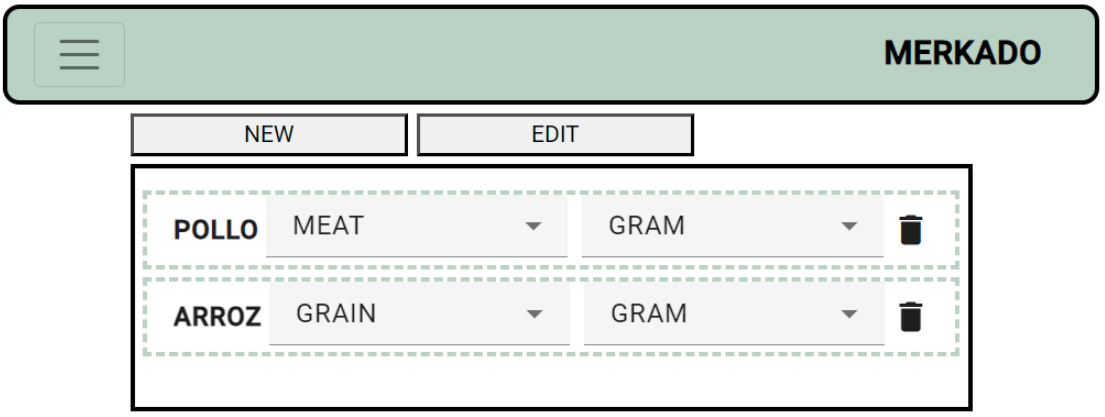
My Meals



Open Meal



My Ingredients



Pruebas

El objetivo de este capítulo es comprobar el resultado de la implementación.

A tomar en cuenta:

- No se han realizado pruebas en la capa de presentación debido a la sencillez de la lógica de la interfaz de usuario.
- Para la capa de negocios se han realizado pruebas a todos los endpoints, ya que esta parte es crítica para la implementación de la solución.
- La realización de pruebas se han realizado usando JUnit.

A continuación se muestran los tests para cada uno de los endpoints a utilizar obtenida de la funcionalidad de exportación de reportes de pruebas de IntelliJ.

java in tfm-merkado-api: 25 total, 25 passed

Collapse

MealResourceIT	
testFindAllByUserId()	passed
getMealById()	passed
testCreate()	passed
testDelete()	passed
testUpdate()	passed
testGetIngredientList()	passed
IngredientResourceIT	
testFindAllByUserId()	passed
getIngredientById()	passed
testCreate()	passed
testDelete()	passed
testUpdate()	passed

java in tfm-merkado-api: 25 total, 25 passed

Collapse

UserResourceIT	
testLogin()	passed
testRegister()	passed
DailyMenuResourceIT	
testFindAllByUserId()	passed
getDailyMenuById()	passed
testGetMealList()	passed
testCreate()	passed
testDelete()	passed
testUpdate()	passed
PlannerResourceIT	
testGetDailyMenuList()	passed
testFindAllByUserId()	passed
testGetPlannerById()	passed
testCreate()	passed
testDelete()	passed
testUpdate()	passed

Como se puede observar, estos resultados corresponden a las consultas que interactúan con el API, proporcionando una validación del comportamiento y la integridad de las operaciones expuestas por el servicio. Estas pruebas aseguran que cada llamada al API responde correctamente a las solicitudes y devuelve los datos esperados de manera consistente. Además, garantizan que la integración entre las capas de la aplicación, desde la capa de lógica de negocio hasta la persistencia de datos, funciona sin problemas.

De esta manera se puede verificar no solo la funcionalidad visible del API, sino que también asegura que la comunicación entre el cliente y el servidor se realiza de manera confiable y eficiente, promoviendo así una experiencia consistente y satisfactoria para los usuarios finales.

Conclusiones y Posibles Ampliaciones

Como se puede observar con lo antes mencionado, se ha cumplido con el objetivo principal de este proyecto, el cual es unificar y consolidar el conocimiento adquirido a lo largo de todo del Máster de Ingeniería Web.

Se ha logrado desarrollar una aplicación funcional, Merkado, que recoge y utiliza para su desarrollo los distintos cursos dictados a lo largo de este Máster.

Se afianzaron los conocimientos de Ingeniería Web Visión General al momento de poder plasmar los requisitos a cumplir que tendría la aplicación, además de los distintos patrones y antipatrones al momento del desarrollo de software.

El conocimiento adquirido en Arquitectura y Patrones para Aplicaciones Web y Backend con Tecnologías de Código Abierto sirvió como base para el correcto desarrollo de la API usando Spring.

Se consolidaron también los conocimientos adquiridos en el curso Front-end para Navegadores Web al momento del desarrollo de la capa de presentación del proyecto usando Angular, empleando la arquitectura aprendida y las distintas herramientas recomendadas como Git y SonarCloud para mantener un versionamiento y calidad adecuada del código.

Finalmente el proyecto ha sido muy enriquecedor al poder servir como práctica de lo aprendido en Metodologías de Desarrollo Web, específicamente el desarrollo de un proyecto usando la metodología RUP (Rational Unified Process) y sus distintas fases y disciplinas.

Como posibles ampliaciones al proyectos se pueden identificar las siguientes mejoras:

- Agregar la funcionalidad de búsqueda de comidas, insumos, menús de otros usuarios.
- Agregar la valuación de distintas comidas para incentivar la interacción entre distintos usuarios.
- Integrar la aplicación con distintas APIs de supermercados para poder calcular el costo total de cada comida.
- Desarrollar una aplicación móvil y de esta manera no limitar la aplicación solo al uso de un ordenador o navegador web.

Bibliografía

1. Spring, https://spring.io/
2. Angular, https://angular.dev/
3. Angular Material, https://material.angular.io/
4. MySql, https://www.mysql.com/
5. StarUML, https://staruml.io/
6. Balsamiq, https://balsamiq.com/
7. Transparencias del Máster en Ingeniería Web presentado en el Moodle de la universidad.
8. Melany Michelle Simbaña Sangucho, Trabajo Fin de Máster, Aplicación Web de Gestión de Donaciones, 2019.

Anexo I

1. Proyecto de StarUml (mdj) con los artefactos utilizados y desarrollados siguiendo RUP.
2. Proyecto de Balsamiq (bmpr) con los prototipos de la aplicación.