PROYECTO INTERMODULAR

SPRINT 5 DISEÑO MODELADO (FRONTEND)

Actividad 1 – Diagrama de Casos de Uso

- **Objetivo:** Identificar y documentar los **actores** y **casos de uso principales** del proyecto.
  Herramienta / Tipo de diagrama:
  - PlantUML usando el **diagrama de casos de uso** (PlantUML: *Use Case Diagram https://plantuml.com/es/use-case-diagram*).
- **Alcance mínimo:**
  - Identificar todos los **actores externos** (usuarios, sistemas externos, etc.).
  - Definir los **casos de uso principales** del sistema (no ejemplos genéricos: relacionados con vuestro proyecto real).
  - Incluir relaciones de **inclusión** y **extensión** cuando corresponda.

1. Objetivos de la Actividad

El objetivo de esta actividad es identificar y documentar los actores principales del sistema, así como los casos de uso más relevantes de la aplicación. También se debe representar gráficamente esta información mediante un diagrama de Casos de Uso con PlantUML.
El sistema modelado corresponde a una aplicación para calcular calorías, gestionar comidas, guardar recetas y compartirlas dentro de una pequeña comunidad de usuarios.

2. Actores identificados

- **Usuario:** persona que utilizará la aplicación para calcular sus calorías, gestionar comidas, guardar y compartir recetas ¿?.
- **API de alimentos:** fuente externa de datos nutricionales usada para obtener calorías y macros de alimentos.
- **Administrador:** encargado de mantener la app así como de moderar las recetas y a los usuarios.

3. Casos de uso principales

Autenticación

- Registrarse
- Iniciar sesión
- Cerrar sesión

Gestión del Perfil

- Consultar perfil
- Editar perfil

Cálculo de Calorías

- Buscar alimento
- Consultar información nutricional
- Añadir alimento a una comida
- Ver calorías totales de la comida
- Eliminar alimentos de una comida

Gestión de Recetas

- Crear receta
- Guardar receta
- Consultar mis recetas
- Compartir receta
- Marcar receta como favorita
- Ver recetas de otros usuarios

Interacción Social (mini red social)

- Comentar receta
- Dar “me gusta”
- Ver perfil de otros usuarios

Moderación

- Reportar receta
- Revisar reportes (si existe rol admin)

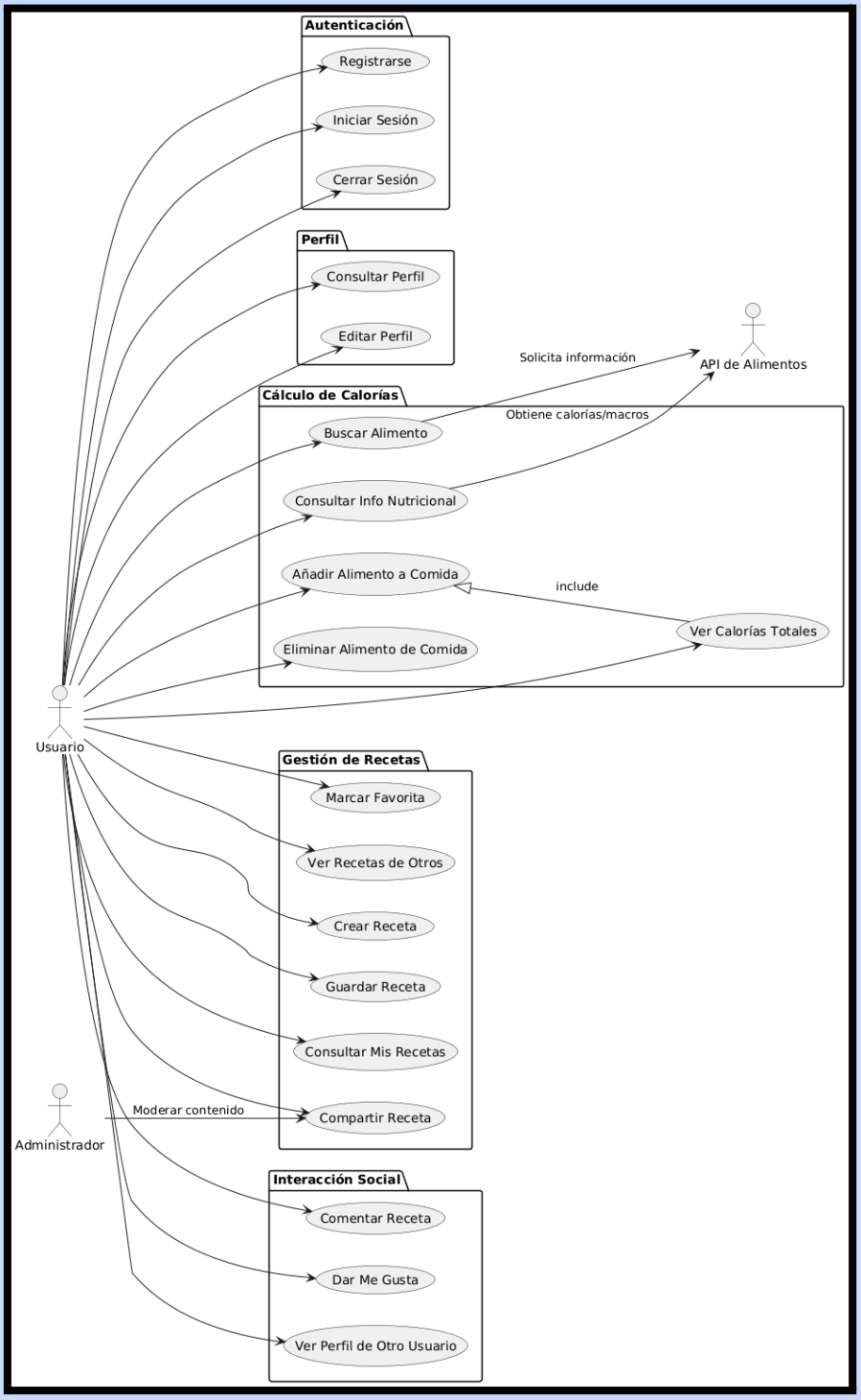
4. Diagrama de casos de uso (PlantUML)

Se adjunta código e imagen del diagrama.

CÓDIGO

```
@startuml
left to right direction
actor "Usuario" as U
actor "API de Alimentos" as API
actor "Administrador" as ADM

package "Autenticación" {
  usecase "Registrarse" as UC1
  usecase "Iniciar Sesión" as UC2
  usecase "Cerrar Sesión" as UC3
}

package "Perfil" {
  usecase "Consultar Perfil" as UC4
  usecase "Editar Perfil" as UC5
}

package "Cálculo de Calorías" {
  usecase "Buscar Alimento" as UC6
  usecase "Consultar Info Nutricional" as UC7
  usecase "Añadir Alimento a Comida" as UC8
  usecase "Ver Calorías Totales" as UC9
  usecase "Eliminar Alimento de Comida" as UC10
}

package "Gestión de Recetas" {
  usecase "Crear Receta" as UC11
  usecase "Guardar Receta" as UC12
  usecase "Consultar Mis Recetas" as UC13
  usecase "Compartir Receta" as UC14
  usecase "Marcar Favorita" as UC15
  usecase "Ver Recetas de Otros" as UC16
}

package "Interacción Social" {
  usecase "Comentar Receta" as UC17
  usecase "Dar Me Gusta" as UC18
  usecase "Ver Perfil de Otro Usuario" as UC19
}

U --> UC1
U --> UC2
U --> UC3
U --> UC4
U --> UC5
U --> UC6
U --> UC7
U --> UC8
U --> UC9
U --> UC10
U --> UC11
U --> UC12
U --> UC13
U --> UC14
U --> UC15
U --> UC16
U --> UC17
U --> UC18
U --> UC19

UC6 --> API : "Solicita información"
UC7 --> API : "Obtiene calorías/macros"

ADM --> UC14 : "Moderar contenido"

UC8 <|-- UC9 : include
@enduml
```

IMAGEN


Autenticación
Registrarse
Iniciar Sesión
Cerrar Sesión
Perfil
Consultar Perfil
Editar Perfil
API de Alimentos
Solicita información
Obtiene calorías/macros
Cálculo de Calorías
Buscar Alimento
Consultar Info Nutricional
Añadir Alimento a Comida
include
Ver Calorías Totales
Eliminar Alimento de Comida
Usuario
Gestión de Recetas
Marcar Favorita
Ver Recetas de Otros
Crear Receta
Guardar Receta
Consultar Mis Recetas
Compartir Receta
Moderar contenido
Administrador
Interacción Social
Comentar Receta
Dar Me Gusta
Ver Perfil de Otro Usuario

Actividad 2 – Diagrama de Actividad

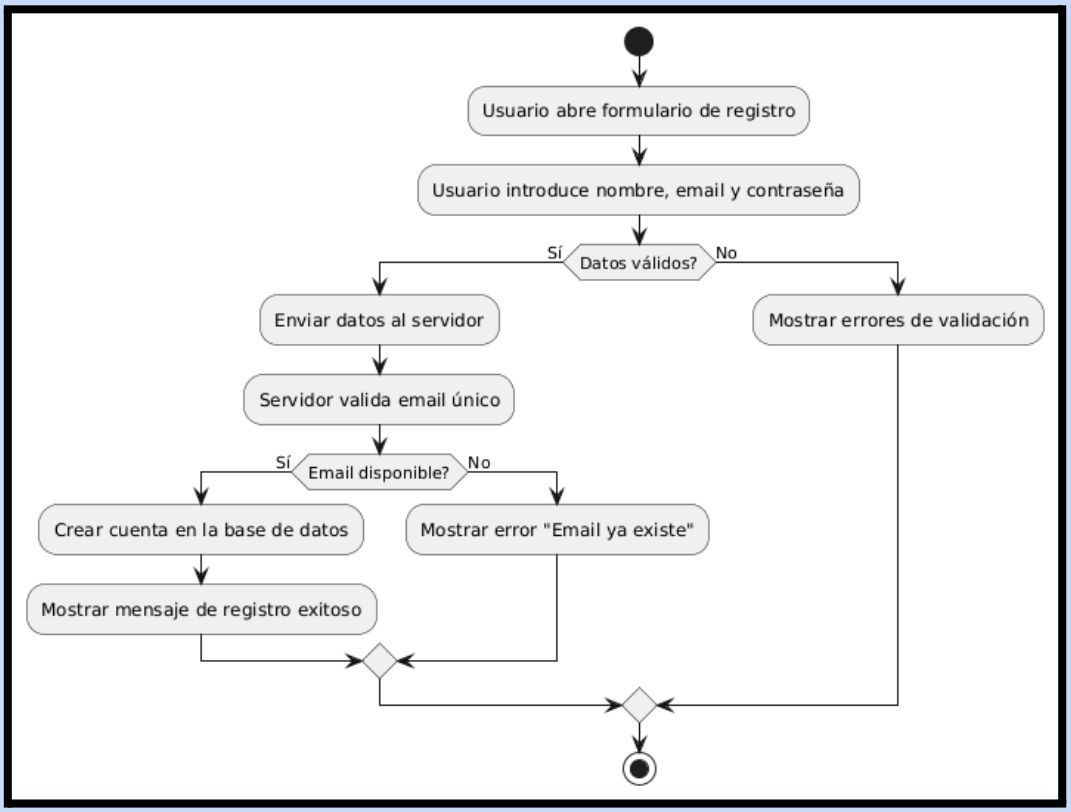
1. Registro de usuarios

Este diagrama representa el flujo que deberá seguir el usuario para registrarse en la aplicación, incluye la validación de datos y el manejo de errores.

CÓDIGO

```
@startuml
start
:Usuario abre formulario de registro;
:Usuario introduce nombre, email y contraseña;

if (Datos válidos?) then (Sí)
    :Enviar datos al servidor;
    :Servidor valida email único;

    if (Email disponible?) then (Sí)
        :Crear cuenta en la base de datos;
        :Mostrar mensaje de registro exitoso;
    else (No)
        :Mostrar error "Email ya existe";
    endif

else (No)
    :Mostrar errores de validación;
endif

stop
@enduml
```

IMAGEN


Usuario abre formulario de registro
Usuario introduce nombre, email y contraseña
Sí
Datos válidos?
No
Enviar datos al servidor
Mostrar errores de validación
Servidor valida email único
Sí
Email disponible?
No
Crear cuenta en la base de datos
Mostrar error "Email ya existe"
Mostrar mensaje de registro exitoso

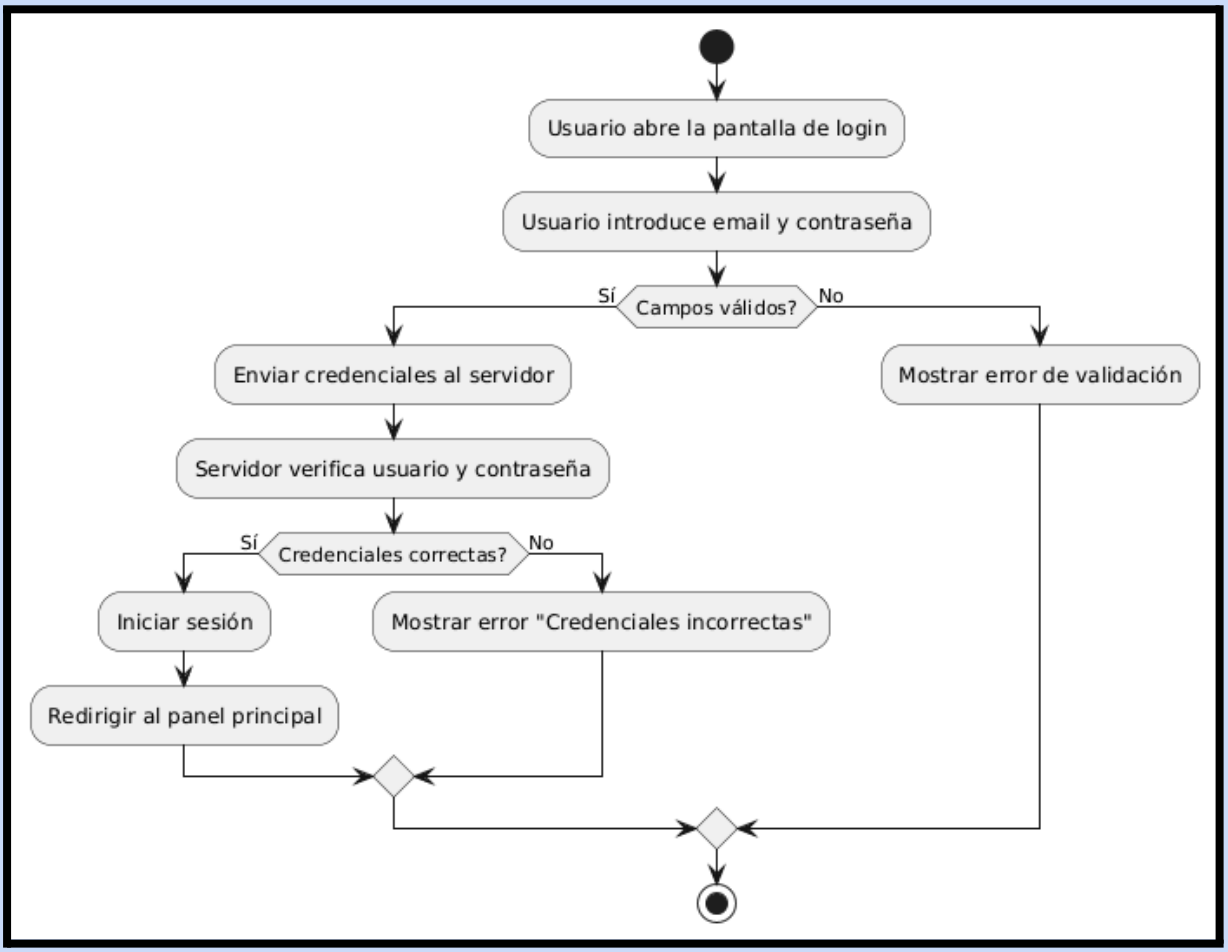
2. Iniciar sesión

Este diagrama representa el proceso de autenticación, incluyendo validación de credenciales y la consecuente respuesta del sistema.

CÓDIGO

```
@startuml
start
:Usuario abre la pantalla de login;
:Usuario introduce email y contraseña;

if (Campos válidos?) then (Sí)
    :Enviar credenciales al servidor;
    :Servidor verifica usuario y contraseña;

    if (Credenciales correctas?) then (Sí)
        :Iniciar sesión;
        :Redirigir al panel principal;
    else (No)
        :Mostrar error "Credenciales incorrectas";
    endif

else (No)
    :Mostrar error de validación;
endif

stop
@enduml
```

IMAGEN



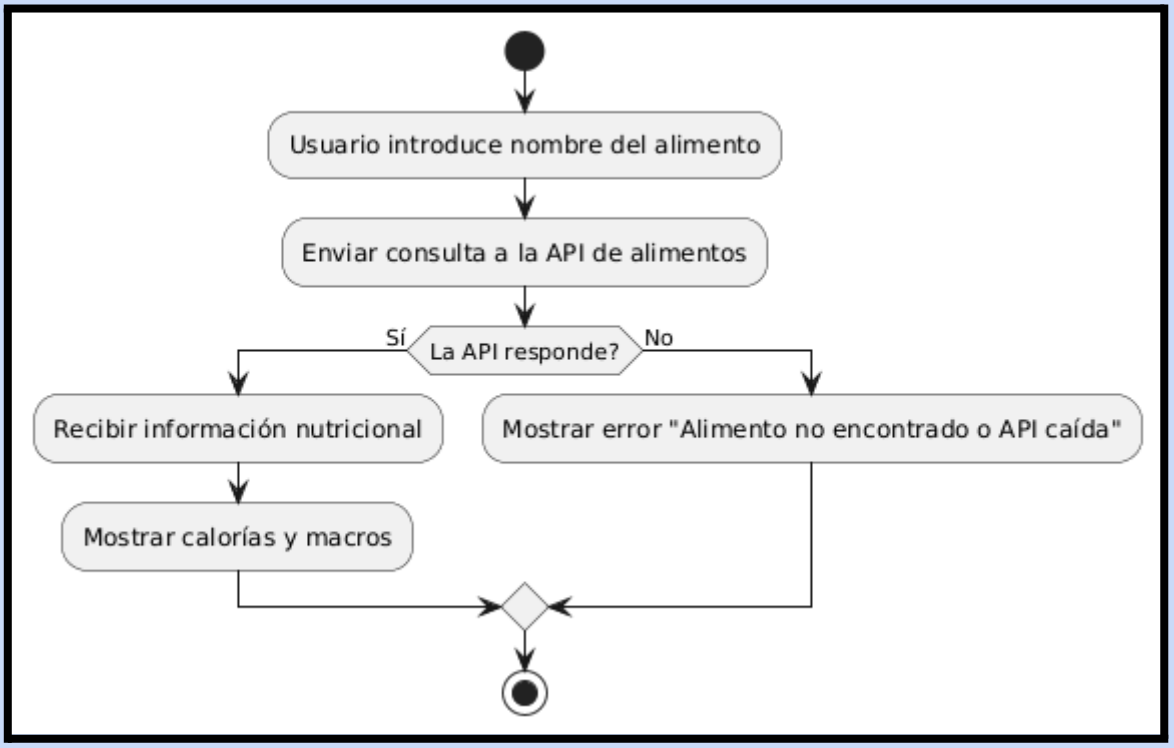
3. Buscar alimento y consultar información nutricional

El usuario puede buscar un alimento y la app obtiene los datos nutricionales mediante una API externa.

CÓDIGO

```
@startuml
start
:Usuario introduce nombre del alimento;
:Enviar consulta a la API de alimentos;

if (La API responde?) then (Sí)
    :Recibir información nutricional;
    :Mostrar calorías y macros;
else (No)
    :Mostrar error "Alimento no encontrado o API caída";
endif

stop
@enduml
```

IMAGEN



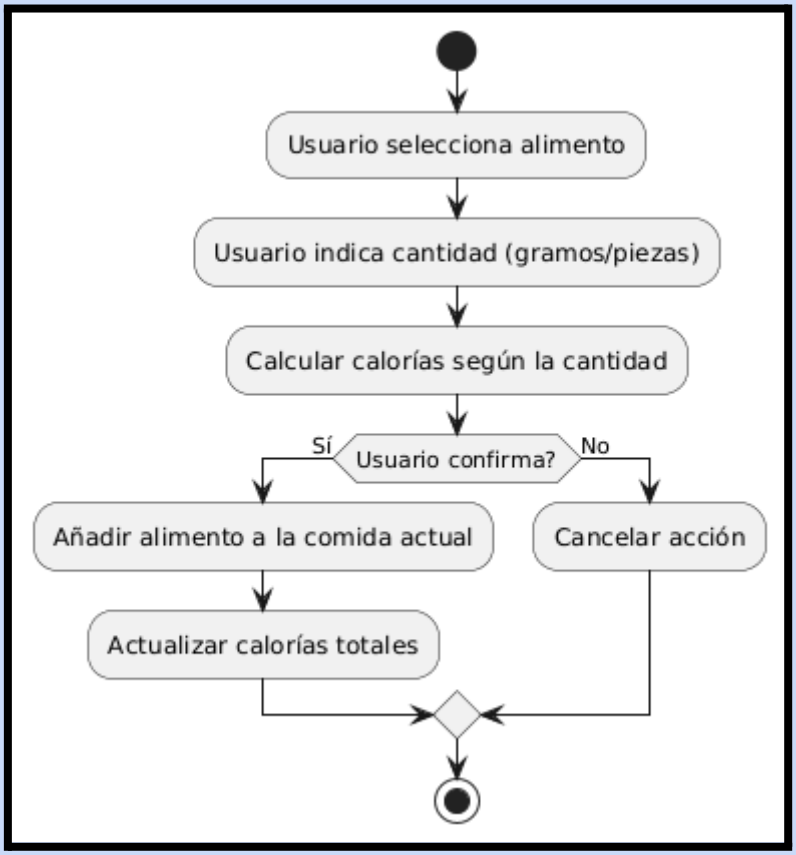
4. Añadir alimento

El usuario selecciona un alimento y lo añade a una comida registrando sus calorías.

CÓDIGO

```
@startuml
start
:Usuario selecciona alimento;
:Usuario indica cantidad (gramos/piezas);
:Calcular calorías según la cantidad;

if (Usuario confirma?) then (Sí)
    :Añadir alimento a la comida actual;
    :Actualizar calorías totales;
else (No)
    :Cancelar acción;
endif

stop
@enduml
```

IMAGEN


Usuario selecciona alimento
Usuario indica cantidad (gramos/piezas)
Calcular calorías según la cantidad
Usuario confirma?
Sí
No
Añadir alimento a la comida actual
Cancelar acción
Actualizar calorías totales

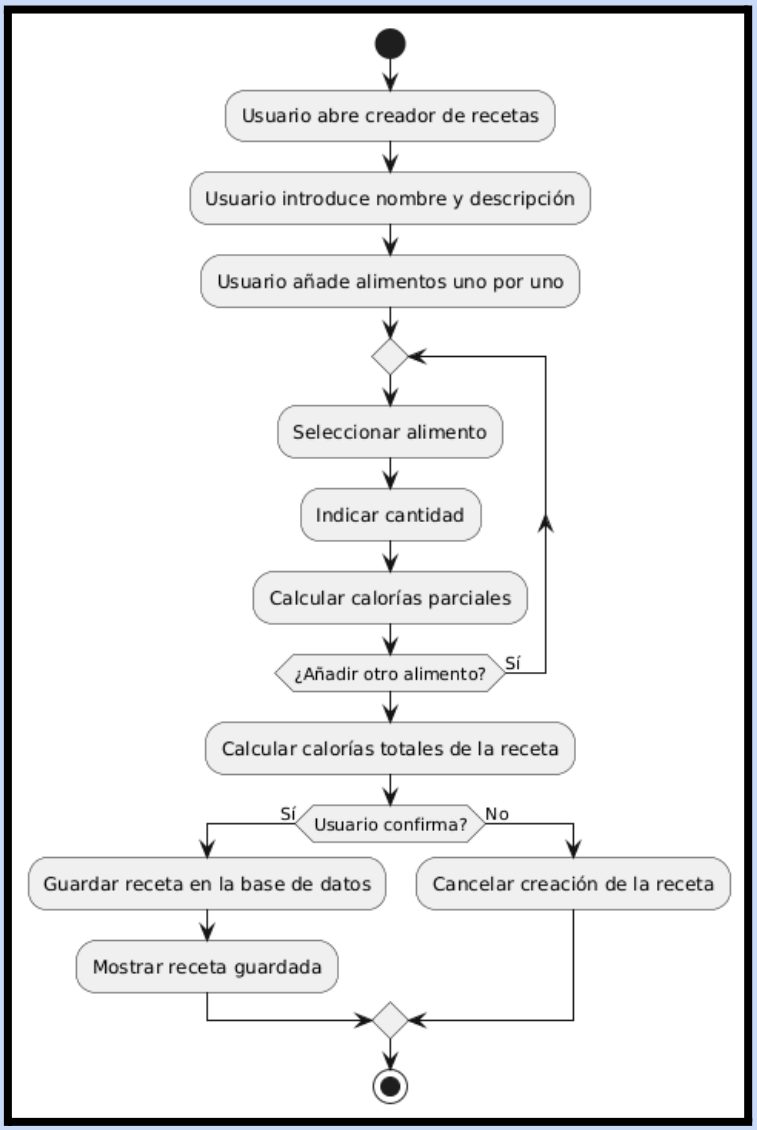
5. Crear y guardar receta de comida

El usuario puede crear recetas propias mediante un proceso compuesto por varios alimentos y guardar la receta en la base de datos del sistema.

CÓDIGO

```
@startuml
start
:Usuario abre creador de recetas;
:Usuario introduce nombre y descripción;
:Usuario añade alimentos uno por uno;

repeat
  :Seleccionar alimento;
  :Indicar cantidad;
  :Calcular calorías parciales;
repeat while (¿Añadir otro alimento?) is (Sí)

:Calcular calorías totales de la receta;

if (Usuario confirma?) then (Sí)
    :Guardar receta en la base de datos;
    :Mostrar receta guardada;
else (No)
    :Cancelar creación de la receta;
endif

stop
@enduml
```

IMAGEN


Usuario abre creador de recetas
Usuario introduce nombre y descripción
Usuario añade alimentos uno por uno
Seleccionar alimento
Indicar cantidad
Calcular calorías parciales
¿Añadir otro alimento?
Sí
Calcular calorías totales de la receta
Sí
Usuario confirma?
No
Guardar receta en la base de datos
Cancelar creación de la receta
Mostrar receta guardada

Actividad 3 – Diagramas de Secuencia

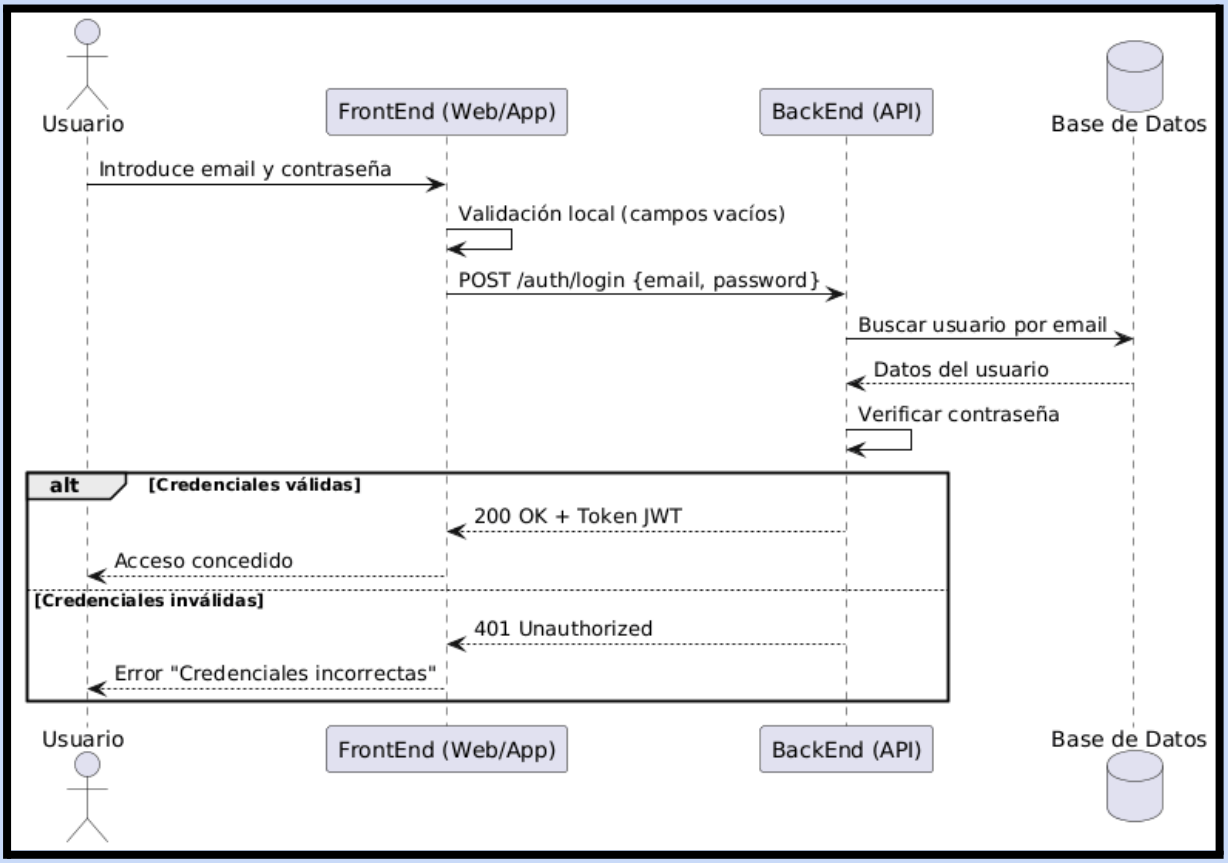
1. Inicio de sesión (LOGIN de usuario)

El usuario interactúa con el Frontend, éste consultará al Backend y a su vez, el Backend validará las credenciales en la base de datos.

CÓDIGO

```
@startuml
actor Usuario
participant "FrontEnd (Web/App)" as FE
participant "BackEnd (API)" as BE
database "Base de Datos" as DB

Usuario -> FE: Introduce email y contraseña
FE -> FE: Validación local (campos vacíos)

FE -> BE: POST /auth/login {email, password}
BE -> DB: Buscar usuario por email

DB --> BE: Datos del usuario

BE -> BE: Verificar contraseña

alt Credenciales válidas
    BE --> FE: 200 OK + Token JWT
    FE --> Usuario: Acceso concedido
else Credenciales inválidas
    BE --> FE: 401 Unauthorized
    FE --> Usuario: Error "Credenciales incorrectas"
end

@enduml
```

IMAGEN



Actividad 4 – Diagrama de Componentes

Aquí se define la arquitectura lógica con los diagramas de Frontend más uno global.

1. Diagrama de componentes (Frontend)

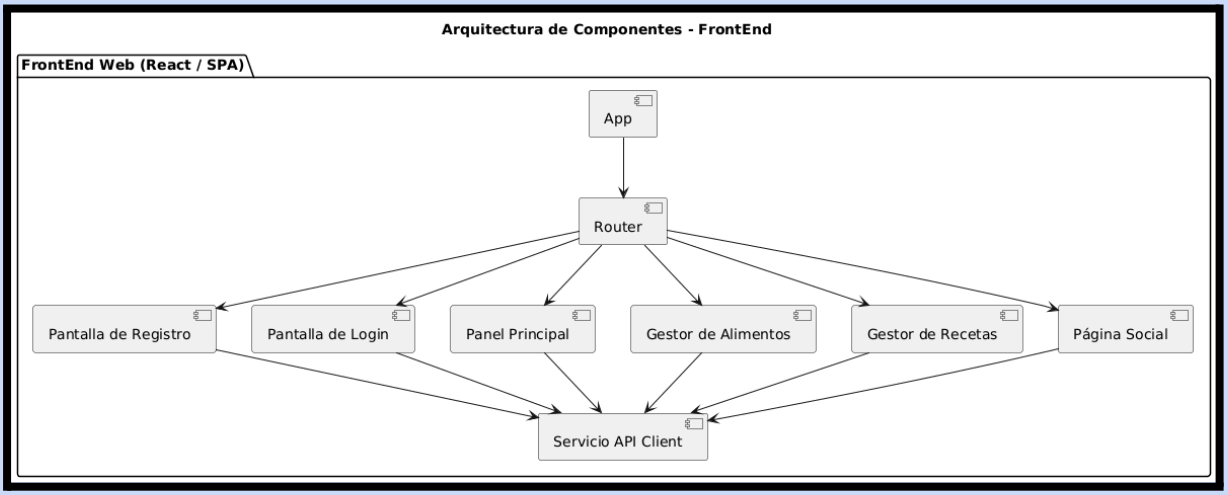
Este diagrama muestra los módulos principales para la app.

CÓDIGO

```
@startuml
title Arquitectura de Componentes - FrontEnd

package "FrontEnd Web (React / SPA)" {

  [App] --> [Router]

  component "Pantalla de Registro" as Registro
  component "Pantalla de Login" as Login
  component "Panel Principal" as Dashboard
  component "Gestor de Alimentos" as FoodManager
  component "Gestor de Recetas" as RecipeManager
  component "Página Social" as SocialPage
```

```
  component "Servicio API Client" as APIClient

  [Router] --> Registro
  [Router] --> Login
  [Router] --> Dashboard
  [Router] --> FoodManager
  [Router] --> RecipeManager
  [Router] --> SocialPage

  Registro --> APIClient
  Login --> APIClient
  Dashboard --> APIClient
  FoodManager --> APIClient
  RecipeManager --> APIClient
  SocialPage --> APIClient
}

@enduml
```

IMAGEN



1. Diagrama global Frontend + Backend

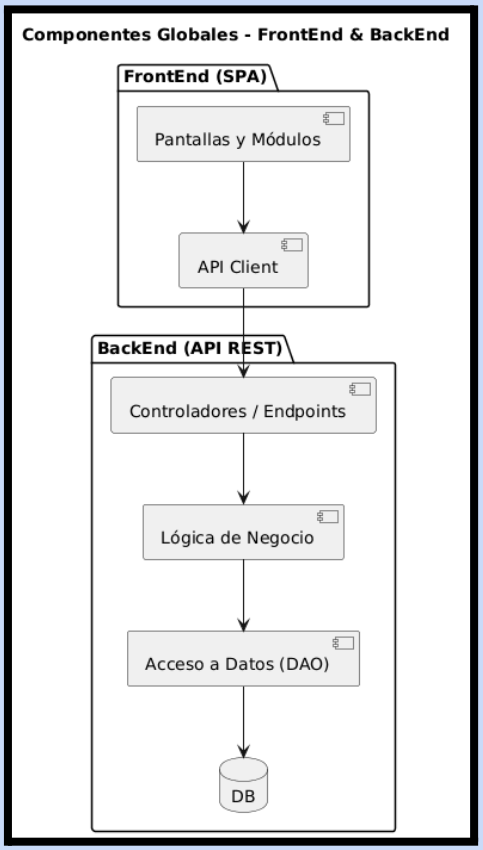
Unión de ambos diagramas para ver la estructura final.

CÓDIGO

```
@startuml
title Componentes Globales - FrontEnd & BackEnd

package "FrontEnd (SPA)" {
  component "Pantallas y Módulos" as FE_Modulos
  component "API Client" as FE_API
}
```

```
package "BackEnd (API REST)" {
  component "Controladores / Endpoints" as BE_API
  component "Lógica de Negocio" as BE_Service
  component "Acceso a Datos (DAO)" as BE_DAO
  database "DB" as DB
}

FE_Modulos --> FE_API
FE_API --> BE_API
BE_API --> BE_Service
BE_Service --> BE_DAO
BE_DAO --> DB

@enduml
```

IMAGEN



Actividad 5 – Modelo de Datos de Intercambio en JSON

Se debe definir los formatos de datos JSON que intercambia el sistema.

1. JSON de petición y respuesta (Crear comida del día)

Esto modela lo que envía el FrontEnd y BackEnd cuando el usuario añade alimentos a una comida, y lo que responde el BackEnd.

CÓDIGO DE PETICIÓN

```
@startjson
{
  "userId": "string",
  "mealType": "string",  // breakfast | lunch | dinner | snack
  "items": [
    {
      "foodId": "string",
      "quantity": "number",
      "unit": "string"   // g, ml, unidad...
    }
  ],
  "date": "string"  // ISO 8601
}
@endjson
```

CÓDIGO DE RESPUESTA

```
@startjson
{
  "mealId": "string",
  "totalCalories": "number",
  "mealType": "string",
  "date": "string",
  "items": [
    {
      "foodId": "string",
      "name": "string",
      "quantity": "number",
      "unit": "string",
      "calories": "number"
    }
  ]
}
@endjson
```

IMAGEN DE PETICIÓN

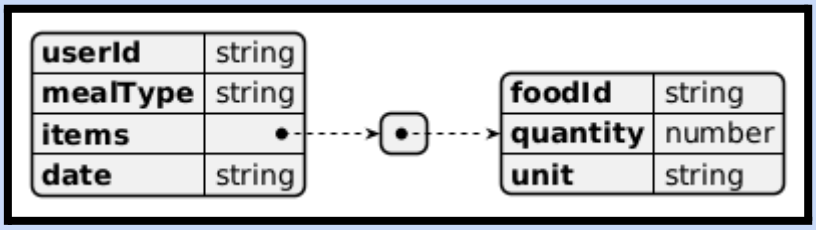
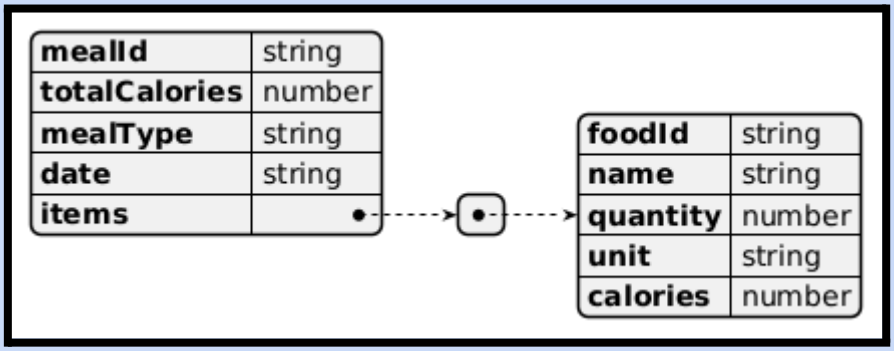


IMAGEN RESPUESTA



Actividad 6 – Diagrama ER (Entidad–Relación)

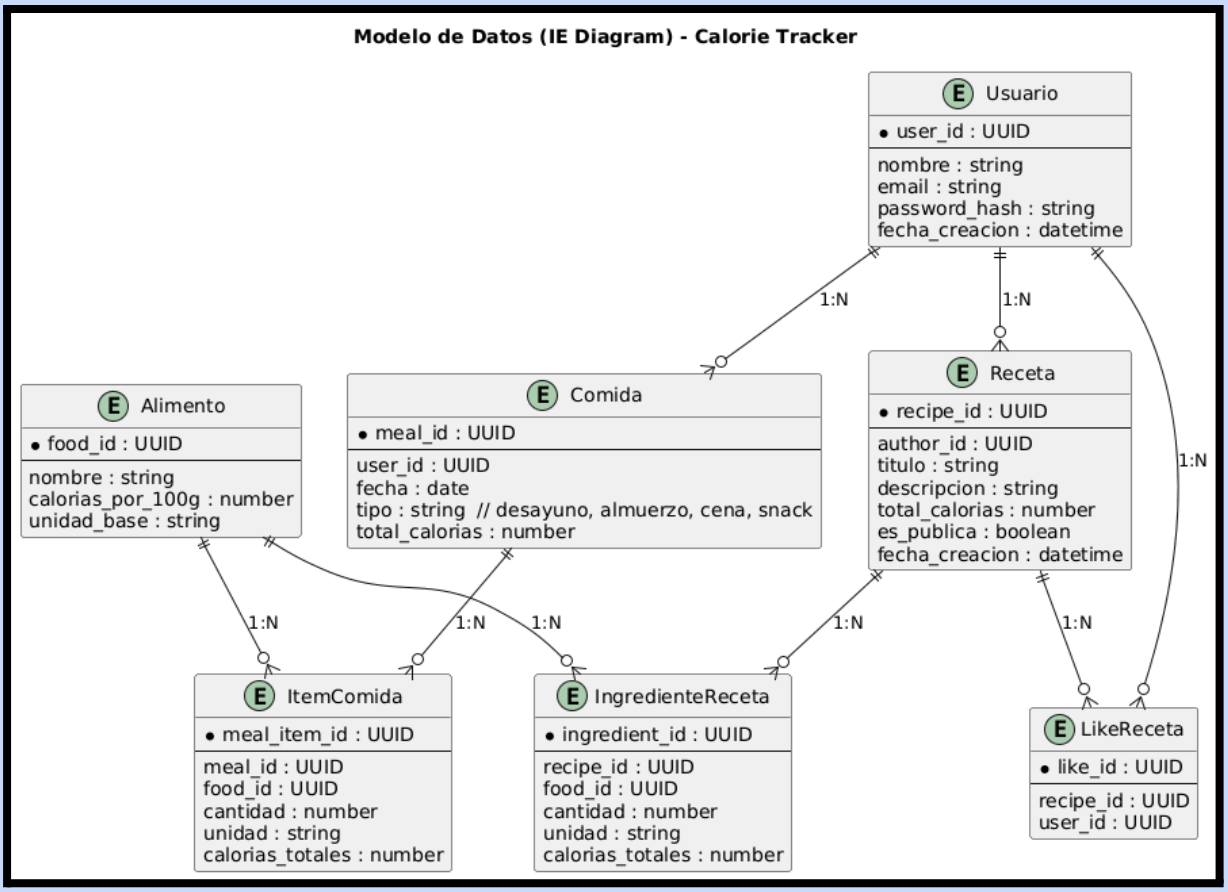
Se debe definir el modelo de datos relacional para la aplicación, no sería el modelo final para el proyecto pero me servirá para ir sentando las bases para cuando se empiece a hacer.

CÓDIGO

```
@startuml
title Modelo de Datos (IE Diagram) - Calorie Tracker

entity "Usuario" as user {
  * user_id : UUID
  --
  nombre : string
  email : string
  password_hash : string
  fecha_creacion : datetime
}

entity "Alimento" as food {
  * food_id : UUID
  --
  nombre : string
  calorias_por_100g : number
  unidad_base : string
}

entity "Comida" as meal {
  * meal_id : UUID
  --
  user_id : UUID
  fecha : date
  tipo : string  // desayuno, almuerzo, cena, snack
  total_calorias : number
}
```

```
entity "ItemComida" as meal_item {
  * meal_item_id : UUID
  --
  meal_id : UUID
  food_id : UUID
  cantidad : number
  unidad : string
  calorias_totales : number
}

entity "Receta" as recipe {
  * recipe_id : UUID
  --
  author_id : UUID
  titulo : string
  descripcion : string
  total_calorias : number
  es_publica : boolean
  fecha_creacion : datetime
}

entity "IngredienteReceta" as recipe_ing {
  * ingredient_id : UUID
  --
  recipe_id : UUID
  food_id : UUID
  cantidad : number
  unidad : string
  calorias_totales : number
}

entity "LikeReceta" as recipe_like {
  * like_id : UUID
  --
  recipe_id : UUID
  user_id : UUID
}

' Relaciones
user ||--o{ meal : "1:N"
meal ||--o{ meal_item : "1:N"
food ||--o{ meal_item : "1:N"

user ||--o{ recipe : "1:N"
recipe ||--o{ recipe_ing : "1:N"
food ||--o{ recipe_ing : "1:N"

user ||--o{ recipe_like : "1:N"
recipe ||--o{ recipe_like : "1:N"

@enduml
```

IMAGEN


Modelo de Datos (IE Diagram) - Calorie Tracker
Usuario
• user_id : UUID
nombre : string
email : string
password_hash : string
fecha_creacion : datetime
Alimento
• food_id : UUID
nombre : string
calorias_por_100g : number
unidad_base : string
Comida
• meal_id : UUID
user_id : UUID
fecha : date
tipo : string // desayuno, almuerzo, cena, snack
total_calorias : number
Receta
• recipe_id : UUID
author_id : UUID
titulo : string
descripcion : string
total_calorias : number
es_publica : boolean
fecha_creacion : datetime
ItemComida
• meal_item_id : UUID
meal_id : UUID
food_id : UUID
cantidad : number
unidad : string
calorias_totales : number
IngredienteReceta
• ingredient_id : UUID
recipe_id : UUID
food_id : UUID
cantidad : number
unidad : string
calorias_totales : number
LikeReceta
• like_id : UUID
recipe_id : UUID
user_id : UUID
1:N
1:N
1:N
1:N
1:N
1:N
1:N
1:N
1:N