

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Grado en Ingeniería Informática

**Asesor nutricional inteligente  
 basado en ingeniería de reglas**

Michael Laudrup

Dirigido por:

Curso: 2021-2022, Septiembre



**Título proyecto**

**Proyecto de Fin de Grado en Ingeniería Informática**

**de modalidad específica**

Realizado por: nombre alumno

Dirigido por: nombre tutor

Fecha de lectura y defensa: XX de Junio de 2022

Agradecimientos

Resumen

Este trabajo de fin de grado consiste en el desarrollo de una aplicación full stack con un sistema basado en reglas cuyo objeto es el de servir como asesor nutricional inteligente a las personas que quieren conseguir unos objetivos de forma física a través de una alimentación saludable, actuando principalmente como asesor nutricional. Además, dado que el dominio de negocio, es decir, la nutrición y dietética, no entran en el ámbito del grado en ingeniería informática, este trabajo de fin de grado se limitará al análisis de la parte más técnica del proyecto.

Como aspecto a reseñar este trabajo aborda la mayoría de las ramas de la informática, como, desarrollo de interfaces gráficas, usabilidad y accesibilidad, protocolos de comunicación, implementación de patrones arquitectónicos avanzados tanto en front como en back, ciberseguridad, inteligencia artificial, bases de datos, desarrollo por el lado del servidor basado en el patrón MVC , pruebas, testing, virtualización con docker y despliegue de aplicaciones, aderezándolo finalmente con un análisis de la situación actual del mundo profesional en el campo de desarrollo de aplicaciones full-stack.

Palabras clave

|  |  |
| --- | --- |
| Concepto | Defnición |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Índice general

[Índice general 7](#_Toc113817312)

[Índice de tablas 9](#_Toc113817313)

[Índice de figuras 10](#_Toc113817314)

[Capítulo 1. Introducción, objetivos y estructura 11](#_Toc113817315)

[1.1 Introducción 11](#_Toc113817316)

[1.2 Objetivos 11](#_Toc113817317)

[1.3 Estructura 11](#_Toc113817318)

[Capítulo 2. Estado del arte 12](#_Toc113817319)

[2.1 Introducción 12](#_Toc113817320)

[2.2 Marco teórico 12](#_Toc113817321)

[2.2.1. Series temporales 12](#_Toc113817322)

[2.2.2. Teoría de colas 12](#_Toc113817323)

[2.3 Marco tecnológico 12](#_Toc113817324)

[2.4 Soluciones existentes 12](#_Toc113817325)

[Capítulo 3. Análisis 13](#_Toc113817326)

[3.1 Modelo de dominio 13](#_Toc113817327)

[3.2 Análisis de requisitos 13](#_Toc113817328)

[Capítulo 4. Diseño 14](#_Toc113817329)

[4.1 Introducción 14](#_Toc113817330)

[4.2 Diseño de base de datos. 14](#_Toc113817331)

[4.2.1. Entidades de la aplicación. 14](#_Toc113817332)

[4.2.2. Relaciones entre entidades. 16](#_Toc113817333)

[4.3 Diseño de sistema basado en reglas 18](#_Toc113817334)

[4.4 Diseño arquitectónico del servidor. 18](#_Toc113817335)

[4.5 Diseño de sistema de autenticación y autorización (Ciberseguridad). 18](#_Toc113817336)

[4.6 Contratos de comunicación servidor-cliente 18](#_Toc113817337)

[4.7 Diseño arquitectónico del servidor. 18](#_Toc113817338)

[4.8 Conclusiones 18](#_Toc113817339)

[Capítulo 5. Implementación 19](#_Toc113817340)

[5.1 Introducción 19](#_Toc113817341)

[5.2 Implementación de la comunicación con otros sistemas. 19](#_Toc113817342)

[5.3 Implementación del algoritmo de previsión. 19](#_Toc113817343)

[5.4 Implementación del sistema de dimensionamientos 19](#_Toc113817344)

[5.5 Implementación gráfica de los resultados 19](#_Toc113817345)

[5.6 Conclusiones 19](#_Toc113817346)

[Capítulo 6. Pruebas 20](#_Toc113817347)

[6.1 Introducción 20](#_Toc113817348)

[6.2 Pruebas unitarias 20](#_Toc113817349)

[6.3 Pruebas de integración 20](#_Toc113817350)

[6.4 Pruebas de validación 20](#_Toc113817351)

[6.5 Conclusiones 20](#_Toc113817352)

[Capítulo 7. Cronograma y costes del proyecto 21](#_Toc113817353)

[Capítulo 8. Conclusiones y trabajos futuros 22](#_Toc113817354)

[Bibliografía 23](#_Toc113817355)

[Glosario 24](#_Toc113817356)

[Anexos 25](#_Toc113817357)

[I. Manual de la aplicación 25](#_Toc113817358)

[II. Datos origen para simulación y pruebas 25](#_Toc113817359)

# Índice de tablas

[Tabla 1 Entidad usuario 15](#_Toc113809709)

# Índice de figuras

# Introducción, objetivos y estructura

## Introducción

El objetivo inicial de este trabajo de fin de grado era el desarrollo de una aplicación web full stack con algo de inteligencia artificial aplicada en el dominio de negocio de la dietética y la nutrición, aunque a grosso modo a nivel práctico y tecnológico se ha implementado este objetivo, es verdad que debido a que el desarrollo de este trabajo ha sido paralelo a mi desarrollo en el mundo profesional real, me he dado cuenta de que este trabajo de fin de grado puede cubrir una necesidad mucho más importante, la cual expongo en los siguiente párrafos.

Una de las carencias más importantes de nuestro sistema universitario es que esta caracterizado por la falta de practicidad y de conexión con el mundo laboral real, prueba de ello es que mucha gente de formación profesional este más preparada inicialmente para trabajar en proyectos software reales que personas que vienen de grado universitario, aunque también hay que mencionar que a medio-largo plazo la curva de aprendizaje del universitario suele ser mucho más corta y la perspectiva mucho más amplia.

Sin embargo, considero que, si se aderezara el grado de ingeniería informática con un poco más de práctica y conexión con el mundo profesional real, se podría llegar mucho más lejos, este trabajo de fin de grado es un granito de arena más en el intento de acercar más a los estudiantes o cualquier persona con inquietud en el mundo de la tecnología al mundo del desarrollo software real, con un ejemplo práctico de una aplicación full stack durante sus diferentes fases, acompañada de comentarios de como se procesan estas fases en el mundo laboral real.

Además, a nivel tecnológico este proyecto es muy rico dado que abarca todas las temáticas fundamentales del grado en ingeniería informática, como son:

* **Programación:** Desarrollo web con MEAN stack, es decir, MongoDB, Express, Angular y Nodejs.
* **Pruebas y testing:** Pruebas manuales de peticiones a back-end con POSTMAN
* **Redes:** Protocolos de comunicación entre lado cliente y lado servidor, con peticiones HTTP con verbos como GET, PATCH, POST, PUT, DELETE
* **Inteligencia artificial:** Implementación práctica de un sistema basado en reglas.
* **Ciberseguridad:** Implementación de sistema de autorización y autenticación con Json Web Token.
* **Usabilidad y accesibilidad:** Experiencia de usuario en aplicaciones web.
* **Virtualización y despliegue**

Con todo lo anteriormente expuesto, tras leer e investigar este trabajo de fin de grado se enriquece al lector con una visión mucho más práctica no solo de como se desarrolla una aplicación sino de muchas implementaciones prácticas de los anteriores temas expuestos, por lo que sirve de complemento perfecto para interiorizar conceptos.

## Objetivos

En este epígrafe se ha optado por dividir los objetivos en tres pilares fundamentales, siendo objetivos tecnológicos, del domino sobre el que se trabaja y de contribución a la comunidad científica, cabiendo señalar que a continuación se expone una jerarquía que va de objetivo más genéricos a objetivos más específicos, dado que un pilar fundamental para la consecución de cualquier objetivo es poder dividirlo en tareas prácticas pequeñas, o como diríamos en algoritmia utilizado estrategias de tipo “Divide y vencerás”.

### Objetivos tecnológicos.

Los objetivos tecnológicos son aquellos objetivos más vinculados a la tecnología y el grado en ingeniería informática.

* Desarrollo de un back-end con nodejs (JavaScript por el lado del servidor), express (framework de nodejs).
* Estructuración y diseño de base de datos no relacional y definición de arquitectura de la información.
* Implementación de un sistema basado en reglas sencillo que permita hacer filtrado de datos en la información que se recupera de la base de datos no relacional.
* Aplicación de sistema de autenticación y autorización basado en Json web token que permita al usuario mantener sesiones seguras en la aplicación.
* Investigación y puesta en práctica de buenos patrones arquitectónicos tanto en front como en back, basándose en las necesidades del proyecto.
* Desarrollo de un front-end (lado del cliente) con Angular(framework de javascript) con una interfaz amigable, fluida y que permita al usuario la consecución de sus objetivos.
* Aplicación del patrón FLUX en el front-end.

### Objetivos del dominio sobre el que se trabaja.

### Objetivos de contribución a la comunidad científica.

## Estructura del documento

El capítulo 2 se ceñirá a analizar que es una aplicación web full stack, ventajas y desventajas y estado actual en el mercado laboral del desarrollo de aplicaciones web, así como una breve exposición de hacia dónde se está dirigiendo ahora mismo el desarrollo de aplicaciones web en el mundo laboral.

Los capítulos del proyecto del 3 al 7 representan cada uno los diferentes ciclos de desarrollo de una aplicación (analisis, diseño, implementación, pruebas) y uno adicional que habla de la integración del proyecto en entornos reales.

Tanto la fase de diseño como de implementación serán paralelas en estructura, pero diferentes en cuanto a nivel de especificidad

# Estado del arte

## Introducción.

Este apartado podría enfocarse desde dos perspectivas:

* La primera, análisis del estado de las aplicaciones de dietética y nutrición en el mercado, ver sus deficiencias y sus virtudes y como en este trabajo de fin de grado se potencian sus virtudes y se reducen sus deficiencias.
* La segunda, la cual, será la que tomaremos en este epígrafe, consiste en un análisis del estado actual, a nivel tecnológico, del desarrollo de aplicaciones web full stack, tanto a nivel tecnológico como en el mundo profesional real, dado que el objeto de este TFG es demostrar conocimientos en el ámbito tecnológico.

## ¿Qué es una aplicación web full stack?

Una aplicación web full stack, es una aplicación cuyo funcionamiento y desarrollo global esta extendido por un lado denominado front-end (navegadores webs) y por otro lado denominado back-end (lado del servidor).

Estos dos lados están estrechamente bien comunicados para que la aplicación funcione en su conjunto, por lo tanto, una aplicación web no es ni una aplicación nativa de escritorio o servidor, ni una aplicación nativa de dispositivos móviles (Android, IOS..etc.).

Estas aplicaciones normalmente desde el lado del cliente se ejecutan mediante un navegador web, lo cual hace que la aplicación se abstraiga del sistema operativo nativo o de la maquina que ejecuta la aplicación, dado que es el navegador el intermediario entre sistema operativo o maquina y la aplicación web.

## Ventajas de las aplicaciones web full stack

* El hecho de que la aplicación web se abstraiga del sistema operativo y del dispositivo hardware especifico hace que sea multiplataforma sin mucho esfuerzo.
* El lenguaje front y back no cambia entre dispositivos, no hay que diseñar con diferentes lenguajes y tecnologías una misma aplicación para IOS, otra para Android y otra para escritorio, todas funcionan.
* Su potencial para ser fácilmente configurable en cuanto a diseño para cualquier tipo de pantalla.
* Comunidad amplia: sin lugar a duda, una de las características determinantes del éxito de una aplicación es como de participativa sea en ella la comunidad científica de computadores, en este caso, javascript tiene una muy fuerte comunidad científica, prueba de ello es que hay muchísimos frameworks, librerías…etc. desarrollados por la comunidad.

## Desventajas de las aplicaciones web full stack.

* La aplicación web no puede exprimir al máximo las funcionalidades del sistema operativo, dado que depende del navegador, por ejemplo, si los nuevos dispositivos aportan un lector de huella dactilar para una identificación biométrica y, por ende, fase de autorización más robusta, hasta que el navegador no aporta mediante javascript librerías o funciones para utilizar esa funcionalidad, la aplicación web no puede explotar esta funcionalidad.
* Muchas veces, los navegadores guardan automáticamente en memoria caché virtual los archivos que reciben de servidor, con el fin de que cuando se vuelvan a solicitar esos archivos se vayan a buscar a memoria y no a servidor, por lo que, si, por ejemplo, el usuario tiene almacenado el software en un archivo .js y en lugar de pedirlo en servidor, recoge uno que esta almacenado en memoria, es probable, que el servidor.

## Tendencias actuales de las aplicaciones web full-stack.

# Análisis

## Modelo de dominio

El modelo de dominio sobre el que está enfocado la aplicación es el de nutrición y dietética, este dominio se caracteriza por ser bastante parametrizable. En este caso lo que se busca es que el usuario pueda hacer un seguimiento de su dieta y recibir algún tipo de asesoramiento por parte de la aplicación para poder conseguir sus objetivos nutricionales (subir de peso, bajar de peso, mantener peso, ganar masa muscular o definir).

Además, dado que el foco del TFG es la ingeniería de software y no el mundo de la nutrición el modelo de dominio es un subconjunto muy pequeño del amplio mundo de la nutrición y dietetica, quedando, a grandes rasgos, como sigue:

* Un alimento está compuesto de kilo calorías que definen **su aporte energético** y macronutrientes.
* Una persona tiene unas características fisiológicas (altura, peso, edad…etc.) que definen la cantidad de **aporte energético diario** que necesita durante un día.
* Además, un usuario puede tener cierto objetivo que incrementa o decrementa en un determinado porcentaje la cantidad de calorías diarias a consumir, es decir, el aporte energético diario necesario.
* Con todo esto, el objetivo de la aplicación es ofrecer los alimentos que debe consumir el usuario diariamente para satisfacer sus necesidades energéticas según su objetivo y proporcionar datos sobre su progreso.

A partir de esta base se pueden sacar muchas funcionalidades, que van extendiendo el modelo de dominio que se va abarcando, sin embargo, no es objeto de este trabajo de fin de grado desviarse del foco de este, que es el mundo tecnológico

## Análisis de requisitos

El primer requisito que se capta es la necesidad de crear, leer, modificar o eliminar alimentos, dado que la unidad básica fundamental de la aplicación son los alimentos. Además, que el usuario pueda agrupar alimentos por secciones en una especie de almacén personal y privado hace que este pueda organizarse mejor.

Ahora bien, muchos usuarios no querrán meter manualmente un alimento, si no que quieren añadir un alimento ya creado poniendo el nombre, para ello, es necesario hacer consultas filtradas, por lo que de todo esto se deduce que es necesario tanto un CRUD de secciones de alimentos como búsquedas filtradas sobre los alimentos para añadirlos a estas secciones.

Además, es necesario tener información del usuario como altura, peso, edad, actividad física diaria, genero, que tipo de alimentación lleva (vegana, vegetariana, omnívora) y cual es su objetivo. Con esta información se pueden calcular otros datos como la cantidad de kilo calorías diarias que necesita, cantidad de agua necesaria al día, índice de masa corporal, distribución de macronutrientes diarios…etc.

También hará falta que la aplicación recomiende alimentos, para ello, con los datos obtenidos del usuario debe ser capaz de discernir que alimentos son adecuados para el usuario. Otros requisitos son que la información se persista si el usuario sale de la aplicación, por lo que los datos se deben asociar a ese usuario y además hace falta algo de autenticación y autorización para que un usuario tenga protegidos sus datos sensibles. Podemos concluir que la captación de requisitos en la etapa inicial queda como sigue:

* CRUD de alimentos.
* CRUD de secciones de alimentos.
* CRUD de datos de usuario.
* Gestor de usuarios con sistema de autenticación y autorización.
* Generar recomendaciones basado en ingeniería de reglas.

En el mundo real, es muy difícil captar todos los requisitos funcionales de una aplicación en el momento inicial, es por ello, que se exponen a continuación los requisitos funcionales detectados en fases como la de diseño, implementación y testing. Cabiendo señalar que en el caso hipotético de que un cliente nos hubiera mandado a hacer esta aplicación, normalmente, durante el propio desarrollo se le ocurren nuevos requisitos, corrige anteriores requisitos especificados e incluso elimina algunos, por lo que conviene de primera mano tener una muy buena comunicación con el cliente.

## Casos de uso.

# Diseño

## Introducción

En este capítulo se bajará un escalón más en la capa de abstracción, enfocándose más en los detalles del desarrollo de la aplicación, para ello, se abordará una visión desde el lado del servidor hacia el lado del cliente, siguiendo el siguiente esquema:

* Diseño de base de datos: Modelado de la información.
* Diseño del servidor: Arquitectura REST
* Contratos entre servidor-cliente: compromiso de comunicación entre servidor-cliente
* Diseño del fron-end: Arquitectura front-end

## Diseño de base de datos.

Sin lugar a duda, uno de los aspectos más importantes en el desarrollo de cualquier aplicación sobre un determinado modelo de dominio es como se diseña la arquitectura de datos, de tal forma que el resultado de este diseño puede ser determinante en el rendimiento de la base de datos y en como de eficiente es el acoplamiento entre el controlador software que dirige el comportamiento lógico de la aplicación con la manera que está estructurada la información.

Subrayar que se resaltan en verde aquellas filas que tienen que ver con el desarrollo de la ingeniería basada en reglas.

### Entidades de la aplicación.

#### Usuario.

La entidad usuario se encarga de almacenar información de un usuario para su autenticación y autorización de acceso a determinados recursos.

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descripción** | **Tipo de dato** |
| Nombre | Nombre del usuario | Cadena de texto |
| Email | Email del usuario. Es un atributo que no admite duplicados | Cadena de texto |
| Password | Contraseña | Cadena de texto |
| PasswordConfirm | Contraseña repetida (Para doble validación) | Cadena de texto |
| rol | Rol del usuario (para control de permisos en la aplicación) | Cadena de texto |
| Último cambio del password | Última vez que cambio el password, para añadir una capa de seguridad en el control mediante Json Web Token. | Fecha |

Tabla Entidad usuario

#### Datos de usuario.

En este caso, por cada usuario tenemos unos datos fisiológicos que permiten al sistema inteligente ser más preciso con las recomendaciones que hace al usuario a nivel dietético y nutricional.

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descripción** | **Tipo de dato** |
| Objetivo | “Ganar peso”, “Perder peso”, “Aumentar músculo” …etc. | Cadena de texto |
| Edad | Edad del usuario | Número |
| Peso | Peso del usuario | Número |
| Altura | Altura del usuario | Número |
| Tipo de alimentación | Dieta omnívora, vegetariano, vegano…etc. | Cadena de texto |
| Alimentos prohibidos | Alimentos que el usuario ha especificado de manera explicita que no quiere saber nada de ellos. | Array de ids de alimentos |
| Actividad física | Sedentario, Moderada, intensa…etc. | Número |
| **Resultado de ingeniería de reglas** | Información que optimiza y precisa mejor el asesoramiento nutricional al usuario. | Objeto complejo de booleanos |
| Identificador usuario | Identificador de la entidad usuario asociado. |  |
| Foto | Url de la foto en el servidor |  |

Tabla Entidad Datos usuario

Con respecto al último campo, cuando los datos de un usuario se introducen en el Sistema basado en regla, esta nos devuelve información que filtra el tipo de recomendaciones que se hacen al usuario a nivel nutricional. Dado que la altura, la edad, peso…etc. no cambian frecuentemente en el tiempo es recomendable guarda resultado de la ingeniería de reglas en lugar de ejecutarla constantemente cada vez que se haga una consulta

#### Alimento.

Un alimento es una de las unidades fundamentales del dominio de negocio, dado que estamos en el ámbito de la nutrición.

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descripción** | **Tipo de dato** |
| Nombre | Nombre del alimento | Cadena de texto |
| Descripción | Descripción del alimento | Cadena de texto |
| Peso | Peso de una unidad | Número |
| Calorías | Aportación energética alimento | Número |
| Hidratos | Hidratos del alimento por 100gr | Número |
| Proteínas | Proteínas del alimento por 100gr | Número |
| Grasas | Grasas del alimento por 100gr | Número |
| Etiquetas | Etiquetas que definen el alimento: [‘Carne’, ‘Vegano’, ‘Sin gluten...etc.]. Estas etiquetas son fundamentales para la ingeniería basada en reglas | Array de Cadena de texto |
| Para que comida es recomendable | Desayuno, Almuerzo, cena | Array de Cadena de texto |
| Imagen fuente | url en la que se almacena la imagen | Cadena de texto |
| Fecha de creación | Fecha de creación del alimento | Fecha |

Tabla Entidad alimento

#### Secciones de alimentos.

El usuario podrá crear sus propias agrupaciones de alimentos de manera personalizada, esto le permitirá buscar alimentos propios de manera más eficiente en incluso cargar en ciertas comidas estas secciones directamente, siendo más prioritarias que las recomendaciones de alimentos del sistema inteligente.

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descripción** | **Tipo de dato** |
| Nombre | Nombre de la sección | Cadena de texto |
| Descripción | Descripción de la sección | Cadena de texto |
| Imagen | Imagen representativa de la sección |  |
| Alimentos | Alimentos que constituyen la sección | Array de identificadores de entidades de tipo alimento |

Tabla Entidad sección de alimentos

#### Registro diario de comidas.

Algo muy intrínseco de una aplicación de nutrición es el factor tiempo, dado que no todos los días comemos lo mismo, es de vital importancia que exista un registro de las comidas que se consumen a lo largo del tiempo

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descripción** | **Tipo de dato** |
| Fecha del registro | Día en el que se efectuó la comida | Fecha |
| Comidas y alimentos | Array de objetos complejos, los cuales, están compuestos de un nombre de comida (Desayuno, Almuerzo…etc) y un array de entidades de tipo alimento | Array de objetos complejo |
| Identificador usuario | Usuario asociado | Identificador |
|  |  |  |

Tabla Entidad registro diario comidas

### Relaciones entre entidades.

Una ves definidas las entidades del dominio se procede a establecer relaciones entre ellas, en este caso, se pasará de una visión pequeña a una visión más grande del conjunto de relaciones de toda la aplicación.

Además, cabe señalar que el diseño no se contempla la especificación de la base de datos elegida, que en este caso es una base de datos No relacional , se verá como influye este hecho en la fase de implementación.

#### Relación Usuario – Datos de Usuario.

En este caso se trata de una relación 1:1, es decir, por cada entidad de tipo “usuario” existirá una entidad de tipo “Datos de usuario” y lo mismo a la inversa, por cada entidad “Datos de usuario” existirá un único “Usuario” asociado. Aunque no es nada común que una relación 1:1 se desacople de una misma entidad transformándolo en dos entidades, en este caso es así, porque en el mundo del software es buena praxis separar la parte de autenticación de usuarios con la de los datos propios del usuario en el dominio de negocio. Es más, normalmente se suele separar en otra entidad los roles del usuario y las contraseñas, nunca yendo en la misma entidad usuario-contraseñas-rol, sin embargo, no se ha hecho en este caso así, dado que la aplicación está más enfocada en un sistema basado en reglas que en ciberseguridad.

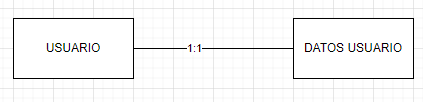


Ilustración Relación Usuario-Datos Usuario

#### Relación Usuario – Sección de alimentos.

En este caso un usuario puede tener varias secciones de alimentos y una sección solo puede tener un usuario, es decir, se trata de una relación 1 a muchos. Acoplándolo a nuestro sistema anterior queda como sigue:

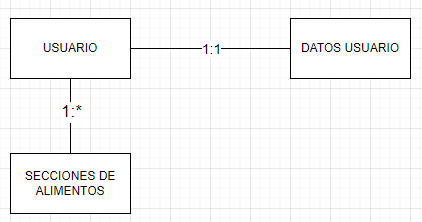


Ilustración Relación usuario - Sección alimentos

#### Relación Sección de alimentos – Alimentos.

Está relación es de muchos a muchos, dado que una sección de alimentos puede tener varios alimentos y un alimento puede estar en varias secciones de alimentos. Esto implica un doble entrelazamiento, tanto la sección de alimentos apunta a varias entidades de Alimentos como un Alimento apunta a varias secciones.

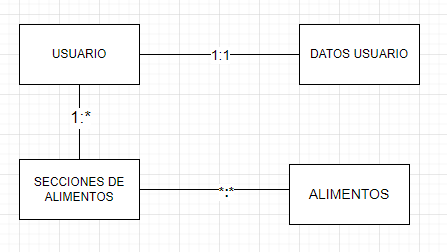


Ilustración Relación sección de alimentos – Alimentos

#### Relación Usuario – Registro histórico diario de comidas.

En este caso se trata de una relación de 1 a muchos, un usuario puede tener varios registros históricos de comida diaria pero un registro solo puede tener un usuario asociado.

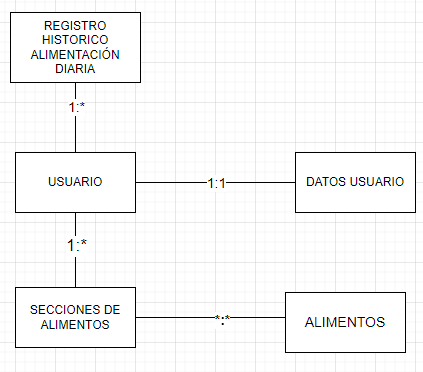


Ilustración Relación usuario - Registro diario alimentación

## Diseño de sistema basado en reglas.

## Diseño arquitectónico del servidor.

## Diseño de sistema de autenticación y autorización (Ciberseguridad).

### Introducción

En esta aplicación se deci

## Contratos de comunicación servidor-cliente

## Diseño arquitectónico del servidor.

## Conclusiones.

# Implementación

## Introducción

En este apartado se baja un escalón más hacia un nivel menos de abstracción, siendo más específicos con la tecnología y la manera en la que se va implementar la aplicación, empezando por la definición arquitectónica global y continuando con una estructura similar a la de la fase de diseño con la diferencia de que en este caso tiene un nivel más de especificidad y de concreciones adaptados a la tecnología a utilizar.

## Visión global arquitectónica del proyecto.

## Implementación de la comunicación con otros sistemas.

## Implementación del algoritmo de previsión.

## Implementación del sistema de dimensionamientos

## Implementación gráfica de los resultados

## Conclusiones

# Pruebas

## Introducción

Este apartado será para definir y comprobar las pruebas realizadas, las primeras para ver que cada bloque iba funcionando correctamente y la última es donde haremos una interpretación de los resultados. Como tenemos datos de los últimos 3 meses, haré una prueba completa para ver que errores nos hubiera dado el sistema en la previsión.

Con el sistema de dimensionamiento, compararé los dimensionamientos, indicados, los que realmente se han tenido y obtendremos la fiabilidad de los datos.

## Pruebas unitarias

## Pruebas de integración

## Pruebas de validación

## Conclusiones

# Cronograma y costes del proyecto

Incluiré el cronograma real, horas de investigación, análisis y elaboración de la aplicación y del trabajo fin de grado.

De esta forma puedo aplicar un coste real a la aplicación y a su implantación separándola de lo que supone hacer un TFG, en el que incluye memoria, defensa, preparación, etc...

# Conclusiones y trabajos futuros

Usaré este apartado para indicar que el sistema está en producción, que está dando una tasa de error en la desviación baja y que ya se han solicitado varías mejoras al sistema, como incluir la previsión de ventas en base a tasas de conversión históricas y un módulo para la revisión de costes asociados a distribuir el tráfico entre plataformas de otra forma.

# Bibliografía

# Glosario

# Anexos

## Manual de la aplicación

## Datos origen para simulación y pruebas