

IES Maestre de Calatrava

Paseo de la Universidad,1

13005-Ciudad Real

# F.P. INFORMÁTICA

**CURSO 2023-24**

**Marzo – Junio**

**Convocatoria: Ordinaria**

**Técnico Superior en**

**Desarrollo de Aplicaciones**

**Multiplataforma**

# ZEROGLUTEN

**Nombre Alumno : Jorge Herrera Martín**

**DNI : 05722163Q**

1. **INTRODUCCIÓN**

**MISMA PORTADA QUE EL ANTEPROYECTO**

**COMPROBACION DEL TEXTO, PUNTOS FINALES, QUE TODOS LOS – SEAN IGUALES.**

**IMÁGENES CON UN BUEN TAMAÑO, LEGIBLES, PUNTOS FINALES**

**PORTADA E INDICE DINÁMICO**

* 1. **Objetivos del proyecto**

El objetivo principal de este proyecto es facilitar a las personas con alguna intolerancia alimentaria el descubrimiento de nuevos productos y recetas aptos para su consumo, mejorando así su calidad de vida y ampliando sus opciones alimenticias de forma segura y accesible.

**1.2 Contexto**

La idea del proyecto surge a partir de la experiencia de familiares y personas cercanas que padecen diferentes intolerancias alimentarias. Estas personas deben tener un cuidado especial al realizar la compra, revisando cuidadosamente el etiquetado de cada producto para asegurarse de que no contiene ingredientes perjudiciales para su salud. Además, al momento de cocinar, deben tener en cuenta la contaminación cruzada. Incluso al salir a comer fuera, necesitan consultar detalladamente las cartas de los restaurantes para comprobar si existe algún plato apto para su consumo.

**1.3 Estado del arte**

En el mercado existen diversas aplicaciones que intentan ofrecer asesoramiento sobre productos alimenticios, aunque la mayoría de ellas están enfocadas únicamente en una sola intolerancia específica. A diferencia de estas, nuestra aplicación ofrece una solución más completa, ya que no solo proporciona información detallada sobre alimentos, sino que también incluye recetas adaptadas a diferentes tipos de intolerancias alimentarias, todo integrado en una misma plataforma.

Algunas aplicaciones ya existentes son:

* **Yuka:** Permite escanear productos y cosméticos para obtener información detallada sobre ellos, con un enfoque especial en los ingredientes y su impacto en la salud.
* **Gluten Free Scanner:** Especializada en el escaneo de códigos de barras para verificar si un producto contiene gluten.
* **Find Me Gluten Free:** Se centra en la búsqueda de establecimientos que ofrecen opciones sin gluten.
* **ShopWell:** Permite escanear productos y recibir recomendaciones sobre su calidad nutricional, abarcando diversas alergias e intolerancias.

**1.4 Metodología de desarrollo a utilizar**

Para desarrollar la aplicación he utilizado la metodología ágil de scrum, adaptándola a un trabajo en solitario, llevando reuniones semanales con la tutora de proyecto.

Se ha priorizado en un primer momento tanto el diseño, como la funcionalidad de la aplicación, desde las funcionalidades más importantes como obtener productos y recetas de la API hasta las funcionalidades más secundarias pero necesarias para que la aplicación esté completa (como los filtros). Gracias a la flexibilidad de la tecnología he podido ir adaptando los tiempos según las necesidades que he ido teniendo durante el proceso.

El desarrollo consiste en varias fases:

* Análisis de requisitos: En primer lugar, se ha realizado un análisis detallado del problema a resolver. Se ha detectado que existe un gran número de personas que sufren intolerancias alimentarias, y que en muchos casos no se trata de una sola intolerancia, sino de varias. Esto complica significativamente su acceso a una alimentación variada y segura.

Al estudiar los productos alimenticios de forma individual, se ha comprobado que estos usuarios no solo deben tener precaución con los ingredientes, sino que además necesitan adaptar sus recetas y formas de cocinar para ajustarse a sus necesidades nutricionales. Por ello, además de proporcionar una herramienta que permita consultar productos aptos según sus alergias o intolerancias, se plantea también ofrecer recetas nuevas e innovadoras que se adapten a dichas restricciones alimentarias, con el objetivo de facilitar y enriquecer su experiencia culinaria.

* Diseño de la base de datos: Se ha optado por un diseño de base de datos sencillo, pero lo suficientemente robusto como para permitir una amplia variedad de operaciones. El objetivo principal ha sido garantizar una estructura clara y funcional que facilite la implementación de las funcionalidades básicas desde el inicio del desarrollo.

Además, este enfoque permite escalar y mejorar la aplicación de forma progresiva, añadiendo nuevas funcionalidades en el futuro sin necesidad de rediseñar por completo la base de datos.

* Diseño de la interfaz de usuario: En la ventana de interfaz de usuario contamos con únicamente tres ventanas, una de ellas es la de registrarse la cual tiene dos bloques para que insertes el nombre de usuario y la contraseña, un botón de registrarse por si el usuario es nuevo y el botón de iniciar sesión.

Cuando el usuario hace clic en el botón de registrarse se abre la ventana de registro donde el usuario tendrá que rellenar un formulario con diferentes preguntas personales sobre el y su dieta y el botón de registrarse para que una vez rellenos todos los campos pueda crearse su usuario y perfil y pueda utilizar la app.

En cuanto a la ventana de la aplicación cuenta con diferentes botones como productos y recetas el cual carga productos y recetas según el filtro que selecciones, los filtros para los productos y recetas cambian dinámicamente según pulses un botón u otro. Por ultimo encuentras el botón buscar que te busca los productos o recetas según los filtros que tengas seleccionados y también encuentras el botón limpiar filtros para borrar los filtros que haya escrito el usuario.

**1.5 Estudio de viabilidad técnica y económica del proyecto**

En cuanto a la viabilidad técnica nos dividimos en requisitos para el desarrollador y requisitos para el usuario final:

* Requisitos para el desarrollador: Es el que va a diseñar y programar la aplicación este necesita un ordenador más potente que pueda soportar las diferentes aplicaciones que se van a utilizar en el desarrollo. Como requisito mínimo necesitaremos un ordenador con 8GB de RAM y un procesador i5. En cuanto al servidor necesitaremos que este soporte base de datos MySQL ya que es donde vamos a guardar nuestra información de la aplicación.
* Requisitos para el usuario final: Estamos hablando de una aplicación de escritorio donde solo necesitarás un ordenador con Windows 10 o superior sin este sea muy potente ya la aplicación no requiere de muchos servicios del sistema. Solo con descargar la aplicación es suficiente para que funcione, puesto que la base de datos viene incluida en ella. También es necesario una conexión a internet para poder hacer las peticiones de la API.

En cuanto al software utilizado contamos con diferentes aplicaciones como visual Studio 2022, MySql Workbrench, Git y GitHub las cuales no tienen ningún coste de licencias.

En cuanto a los recursos humanos en el equipo de trabajo estoy solo yo, con conocimientos de todas las tecnologías mencionadas anteriormente y capacitado para realizar la aplicación al completo sin ayuda. Como puntos de apoyo puedo buscar información en internet y en la IA.

En cuanto a los riesgos técnicos encontramos:

* La escalabilidad de la base de datos: Donde un diseño perfecto de esta es fundamental para que en un futuro se pueda ampliar la información de la aplicación sin necesidad de cambiarla.
* Complejidad en el filtrado de productos: Es muy importante tener las búsquedas bien perfiladas, para que el usuario a la hora de buscar, encuentre los productos que necesite.
* Actualización de la información de los productos: Los productos están en constante cambio en cuanto a los precios o incluso a los ingredientes que lo componen debido a cambios que pueda llevar a cabo la fabrica a la hora de elaborar los productos. Para ello debemos de estar seguros y corroborar de forma exhaustiva la información de la aplicación
* Curva de aprendizaje si se amplía las tecnologías de la aplicación: La curva de aprendizaje es elevada si se quiere ampliar la información y las tecnologías de la aplicación a más dispositivos y plataformas

En conclusión, el proyecto es técnicamente viable, ya que cuento con los conocimientos necesarios para utilizar de forma eficiente las herramientas técnicas y tecnológicas implicadas. Además, los riesgos se consideran controlables gracias a la planificación detallada que se llevará a cabo durante el desarrollo de la aplicación.

Hablando de la viabilidad económica para el desarrollo de la aplicación tenemos en cuenta que el ordenador para el desarrollo ya lo tenemos a la vez que las aplicaciones y licencias para programas, las cuales todas son gratuitas.

En un primer momento la aplicación es solo de escritorio la cual no necesitaríamos pagar ningún dominio web, aunque en un futuro podamos extender a otras tecnologías.

En cuanto al mantenimiento de la aplicación ya hay que tener en cuenta la base de datos que costaría unos 60€ al año, el mantenimiento y/o actualizaciones de la API el precio puede variar según lo realice yo o contratemos un equipo para ello, en un primer momento lo realizaría yo. En total anualmente nos costaría 60€ el hosting en la base de datos.

Como presupuesto de inicio contamos con 100€ ya que la aplicación es desarrollada por mi y con este presupuesto sirve para tapar las necesidades de host y alguna necesitad extra que surja.

En un principio la financiación de la aplicación es propia de mi mismo, ya que en un futuro si la aplicación llega a tener éxito se puede buscar promocionarla para ganar dinero e invertirlo en mejorar la funcionalidad de la aplicación.

Como modelo negocio creemos que el mejor es proporcionar la aplicación totalmente gratis, con todas las funcionalidades, esto nos permite llegar rápido a más clientes en un principio, aunque no se descarta en un futuro ampliarlo a un modelo fremium para aumentar el presupuesto de la aplicación y poder desarrollar nueva funcionalidad y aumentar el equipo de desarrollo. A su vez incluiremos algo de publicidad dentro de la aplicación para reducir los gastos obligatorios de hosting.

En conclusión, la parte económica también sale viable gracias al bajo coste inicial y la posibilidad de autofinanciación. Además, tiene potencial de ganancias económicas si se amplía a una estrategia fremium o publicidad dentro de la aplicación.

**1.6 Fases del proyecto**

El proyecto comenzó el día 1 abril, con dos meses de antelación a fecha de entrega.

Se hicieron estimaciones de que en desarrollar la aplicación se tardarían unas 7 semanas y dejar las últimas dos semanas para escribir la documentación sobre la aplicación. Y se ha llevado acabo con las siguientes fases:

* Fase de inicio: O también podemos llamarla fase de investigación donde hemos identificado el problema que queríamos resolver. Posteriormente hemos estudiado las necesidades del usuario, y aquellas soluciones que hay de nuestro problema y eso nos ha ayudado a definir los objetivos de nuestro proyecto. Para ayudarnos hemos realizado un diagrama de casos de uso, un diagrama de la base de datos y hemos ido creando bocetos de como queremos que quedara nuestra aplicación. Esta primera fase nos ha llevado 2 semanas hacerla.
* Fase de ejecución: Esta fase ha sido la más larga como estaba previsto con una duración de 5 semanas. En ese tiempo hemos desarrollado la aplicación tanto el diseño (parte de frontend) como la parte de código (backend). Durante este tiempo hemos tenido que hacer cambios en algunos diseños realizados en la fase de inicio debido a los inconvenientes que hemos tenido. Esto nos ha llevado 4 semanas. La semana restante la hemos utilizado para perfeccionar tanto el diseño como la parte del código y a perfeccionar aquellas funcionalidades que son fundamentales para nuestra aplicación.
* Fase de documentación: Última fase del proyecto, que nos lleva a explicar el por que y el como de nuestra aplicación. Explicamos el paso a paso que hemos seguido para diseñarla, creamos las presentaciones, los videos explicativos y el manual de uso. Para ello hemos utilizado 2 semanas.

**2. Recursos materiales y personales para realizarlo**

Como recursos utilizados en el desarrollo de la aplicación encontramos

* Para crear los diagramas iniciales y bocetos de las pantallas se ha utilizado Draw.io ya que contiene una gran cantidad de paletas que nos permite diseñar diferentes esquemas dentro de una misma aplicación.
* API de Spoonacular: Fuente de datos donde hemos sacado la información de los productos y recetas.
* Visual Studio como IDE de programación donde se ha creado una aplicación de diseño WPF en la parte de diseño y se ha utilizado C# como lenguaje principal de programación. Dentro de la aplicación podemos encontrar diferentes imágenes para los logos y para hacer mas ameno el diseño de la base de datos, al igual que alguna imagen para utilizar en caso de que la API falle.
* El proyecto ha sido desarrollado completamente por Jorge Herrera Martín con un seguimiento tutorizado de Rosa María Zapata Calle y ha sido desarrollado en un equipo con un ryzen 7, con 32 gb de RAM.

**3. Fases y secuenciación de las actividades del proyecto**

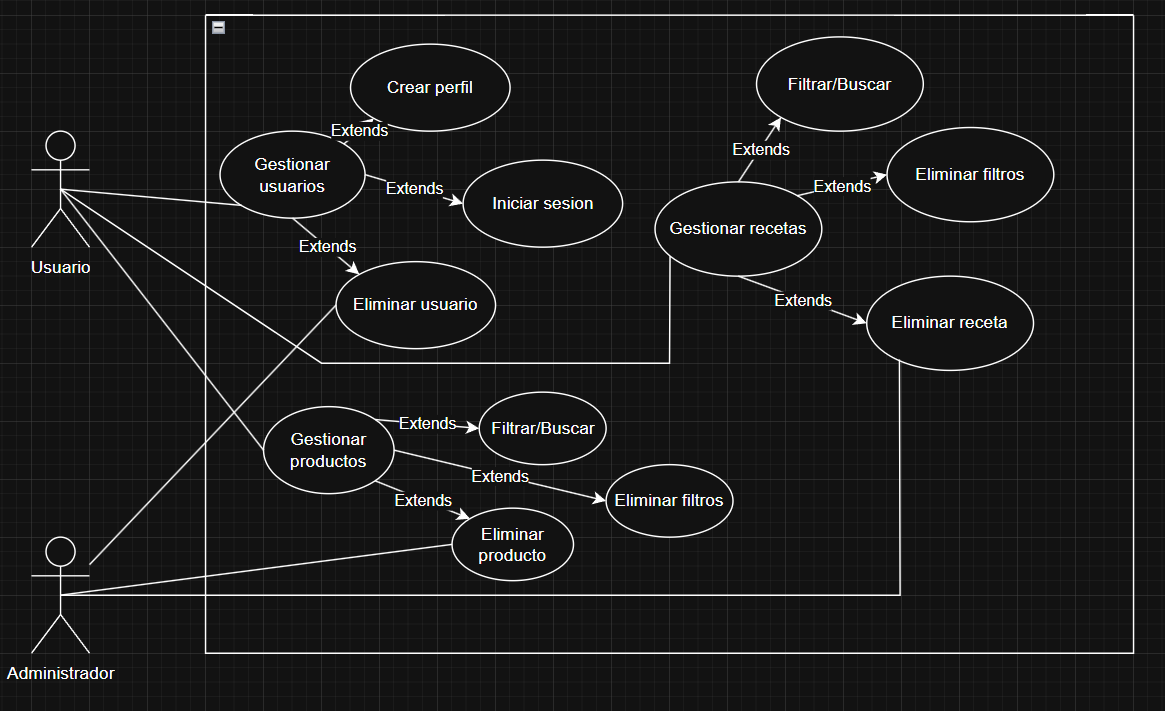
**3.1 Análisis de requisitos: Diagrama de casos de uso**

En cuanto al diagrama de casos de uso encontramos dos tipos de personas que manejarán la aplicación:

* El usuario: Se caracteriza por tener un control amplio sobre las operaciones disponibles dentro de la aplicación. En primer lugar, puede realizar búsquedas de productos y aplicar diversos filtros para ajustar los resultados a sus necesidades. Entre estos filtros se incluyen: nombre del producto (campo de texto libre), calorías máximas, proteínas mínimas y grasas máximas (todos ellos mediante campos seleccionables).

Asimismo, el usuario tiene la posibilidad de explorar recetas según sus preferencias. Al igual que en la búsqueda de productos, puede filtrar por nombre (campo de texto libre), calorías máximas, proteínas mínimas y grasas máximas (campos seleccionables), lo que facilita encontrar recetas que se adapten a sus restricciones alimentarias y objetivos nutricionales.

* El administrador: Además de contar con todas las funcionalidades accesibles para los usuarios, el administrador cuenta con permisos especiales que le permiten gestionar y moderar el contenido de la aplicación. Tiene la capacidad de eliminar usuarios si lo considera necesario, así como suprimir productos o recetas que presenten información incorrecta o que no cumplan con los estándares de calidad. De este modo, se asegura la fiabilidad y el correcto funcionamiento de la plataforma

****

Este no era el esquema que se desarrolló en la fase de inicio de la aplicación, sin embargo debido a los fallos que hemos ido teniendo a la hora de desarrollar la aplicación nos hemos visto obligados a cambiarlo, aún así no nos hemos olvidado de esa funcionalidad que entrará como mejoras en versiones posteriores de la aplicación.

**3.2 Diseño. Capa persistencia: Diagrama E/R de la base de datos/Relacional**

En cuanto a la base de datos podemos fijarnos en que la tabla principal de nuestra aplicación es la de usuario.

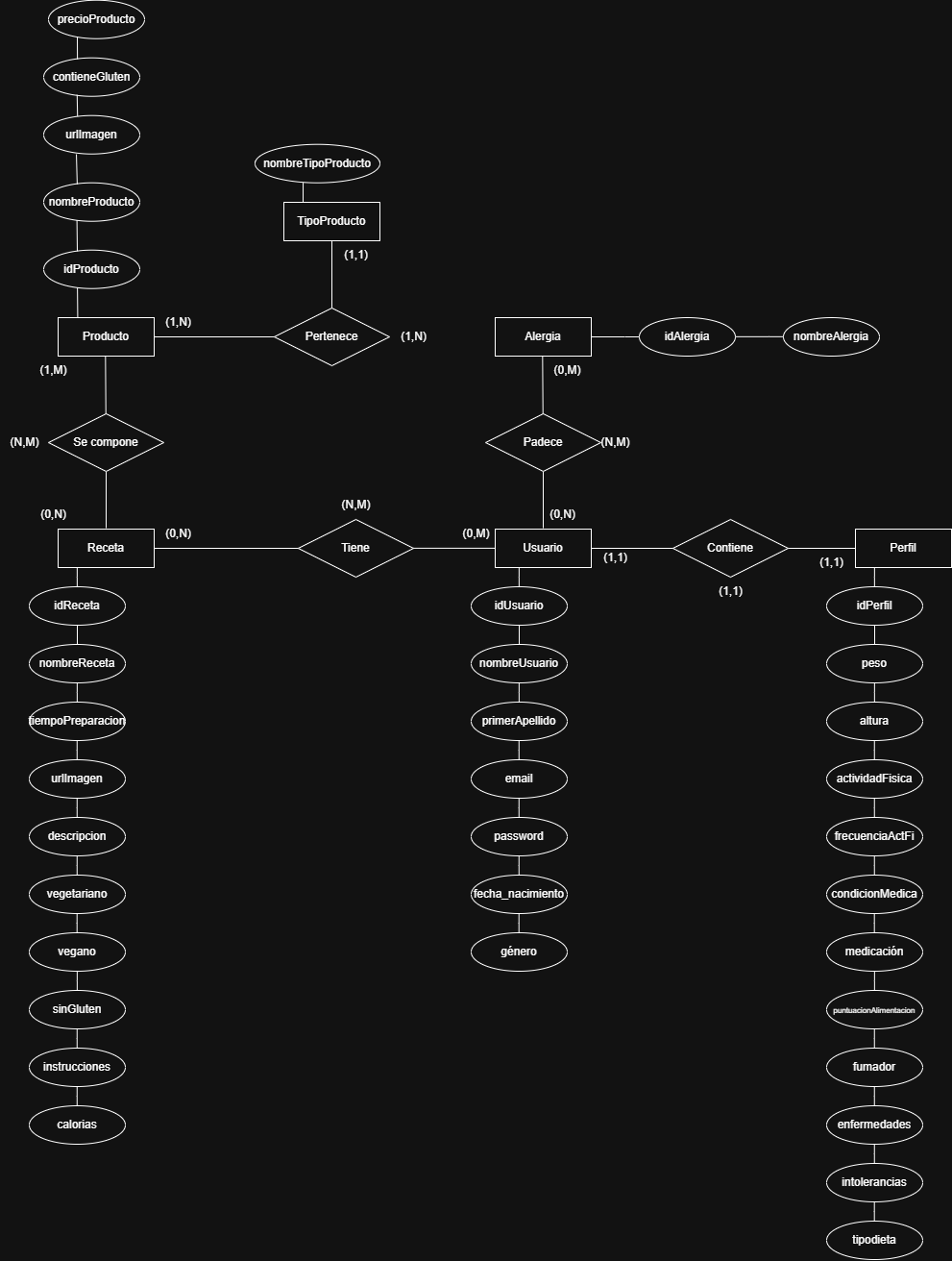
En un primer momento vemos que a cada usuario se le “asigna” un perfil, este perfil se genera cuando el usuario se registra, gracias a nuestro formulario de registro y se guarda toda la información de ese formulario entre la tabla usuario y la tabla perfil. La relación entre ellos es de 1,1 puesto que un usuario tiene un único perfil, y un perfil solo puede ser tenido por un usuario, esto nos hace ver que aun que dos usuarios tengan las mismas respuestas en el estudio el perfil es único de cada usuario ya que el idPerfil siempre va a ser el mismo que el idUsuario.

Después vemos como de usuario sale una segunda relación con la tabla alergia, lo hemos definido como una relación N,M porque nosotros no queremos cerrar al usuario a que ponga solo 1 alergia si este tiene más de una, por lo tanto se podría decir que un usuario padece ninguna o muchas alergias y una misma alergia pueden tenerla ninguno o muchos usuarios. Añadimos la opción de que no la tengan ningún usuario por que tal vez haya alguien que utilice nuestra aplicación de modo de aprendizaje o le dé un uso normal sin tener ninguna intolerancia.

La tercera y última tabla que está unida a usuario es receta, esto lo hemos hecho así porque al tener una relación N,M (un usuario puede tener 0 o muchas recetas favoritas y una receta puede ser la favorita de ninguno o muchos usuarios) sale una tabla intermedia que podemos llamar recetas\_favoritas donde se guardarán en un futuro las recetas que el usuario quiera tener marcadas como favoritas.

Otras tablas que hemos creado pero que no tienen relación con el usuario es la de producto, esta va unida receta puesto que dentro de una receta tenemos uno o varios productos. En cuanto a la cardinalidad la hemos hecho de N,M puesto que una receta se compone de mínimo 1 producto o muchos productos, sin embargo, un producto puede estar en ninguna o en muchas recetas. Al tener una cardinalidad N,M nos sale una tabla intermedia pero a diferencia con la tabla recetas\_favoritas esta no tiene ninguna utilizad hasta el momento

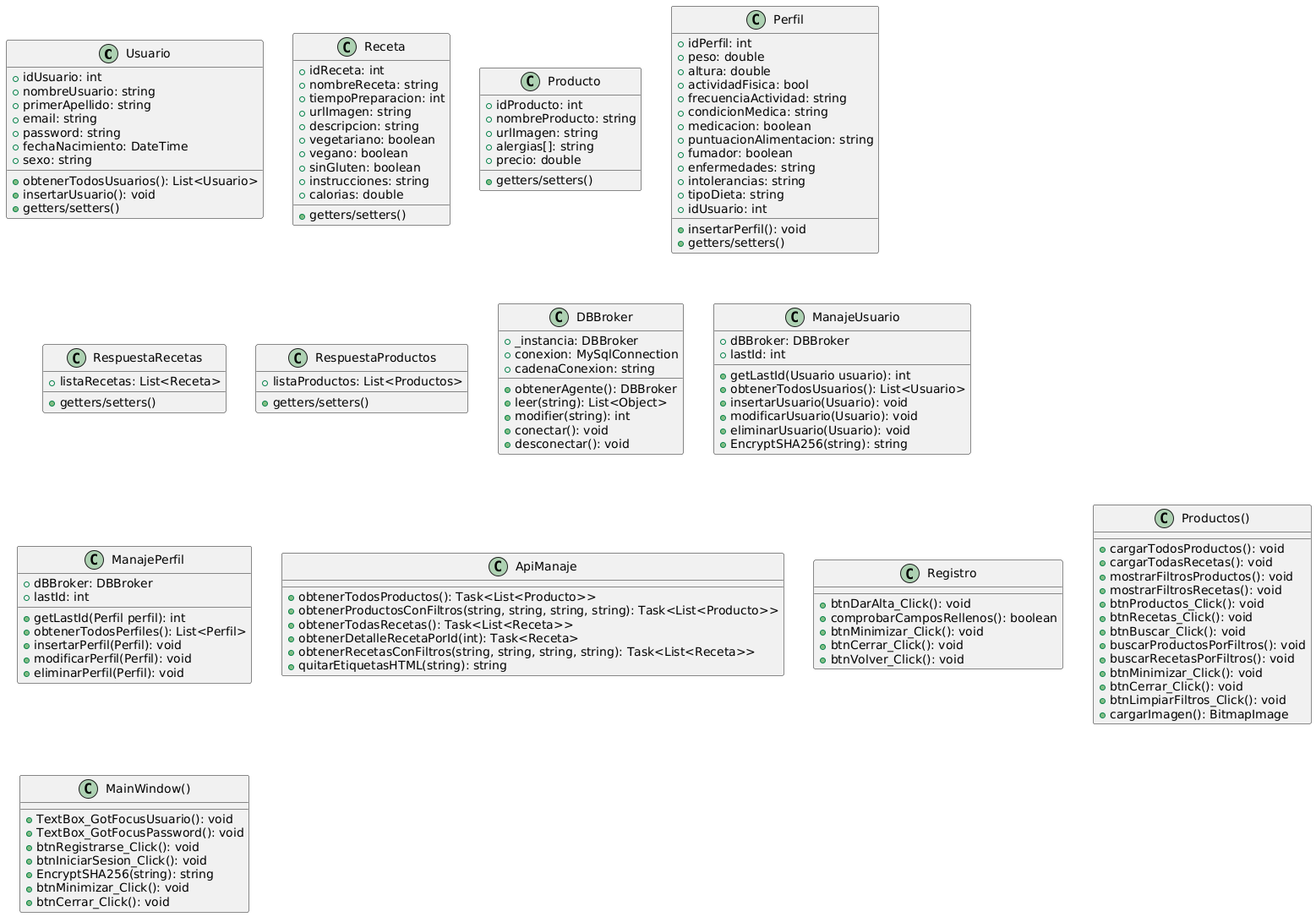
Y por último y como ayuda para la clasificación de los productos hemos pensado en crear la tabla tipoProducto. La funcionalidad de esta tabla consiste en agrupar los diferentes productos (de la tabla producto) para que estos sean más fácil de encontrar a la hora de filtrar por ellos.



* 1. **Diseño. Capa de negocio: Diagrama de clases (UML)**

Nuestro proyecto se compone de 13 trece clases. En la capa de Dominio nos encontramos las clases Perfil, Producto, Receta, RespuestaProducto, RespuestaReceta y Usuario. En la capa de persistencia encontramos las classes ApiManage, ManagePerfil, ManageUsuario, DBBroker. Y en la vista encontramos MainWindow, Productos y Registro a estas tres últimas hay que añadirle una clase más de cada en .xaml que es la extensión de las ventanas de interfaz gráfica. A continuación vamos a explicar las clases por partes.

* Clase usuario: En esta clase hacemos referencia a los atributos personales que tiene un usuario con atributos como idUsuario el cual es único para cada usuario, nombre, primerApellido, email, password, fechaNacimiento y sexo. Y como métodos encontramos obtenerTodosLosUsuarios() que busca todos los usuarios que haya en la base de datos y los devuelve en una lista, este método es necesario a la hora de que el usuario inicie sesión en la aplicación, para comprobar que inserta correctamente el nombre y la contraseña. InsertarUsuario() se ejecuta internamente cuando el usuario se registra, lo inserta en la base de datos
* Clase Receta: En esta clase vemos los atributos que tienen las recetas. Nos encontramos idReceta unico para cada receta, nombreReceta, tiempoPreparación del plato/receta, urlImagen del alimento o receta, descripción breve explicación de que se compone el plato, vegetariano, vegano, sinGluten, son 3 atributos que indican si es apto o no para personas que comen comidas con estas características, instrucciones de como se hace el plato y las calorías que contiene el plato
* Clase Producto: Hacemos referencia a los productos de la API con atributos como idProducto el cuál es único para cada producto, nombreProducto, urlImagen del producto, lista de las alergias que contiene el producto y el precio del producto
* Clase Perfil: En esta clase se guardan aquellos atributos que el usuario ha seleccionado a la hora de registrarse como el idPerfil (este atributo se crea automáticamente de forma autoincremental), peso, altura, actividadFisica, condicionMedica, medicación, puntuacionAlimentación, si es fumador o no, si tiene enfermedades y/o intolerancias, el tipoDieta que sigue. En cuanto a los métodos encontramos el insertarPerfil que inserta el perfil en la base de datos.
* Clase Registro: La cual solamente tiene la funcionalidad de dar de alta al usuario (btnDarAlta\_Click) y otros métodos para que el botón de minimizar y cerrar de la ventana haga sus funciones.
* Clase Productos: esta clase tiene toda la funcionalidad que puede utilizar el usuario con los productos (y recetas) como cargarTodosProductos y cargarTodasRecetas que son los métodos que se ejecutan nada más seleccionar el botón productos o recetas respectivamente, para que en un primer momento el usuario tenga una lista de productos aleatorios que le ayuden en su búsqueda. Métodos que hacen la ventana dinámica como mostrarFiltrosProductos o mostrarFiltroREcetas los cuales cambian los filtros dinámicamente según si el usuario esté viendo recetas o productos. buscarProductosPorFiltros o buscarRecetasPorFiltros que sirven para buscar productos o recetas según los filtros que haya seleccionado el usuario.
* Clase RespuestaProductos y RespuestaRecetas: Estas solamente tienen un atributo listaProductos y listaRecetas respectivamente y sirven para deserializar el JSON de la respuesta de la api a nuestra petición.
* Clase DBBroker: Esta clase es la que interactúa directamente con la base de datos con atributos como \_instancia, conexión y la cadenaConexion. Y algunos métodos como conectar/desconectar de la base de datos, el método modifier que es el que realiza alguna operación de insertar, modificar y eliminar.
* Clase ManajeUsuario: Esta clase lleva todas las acciones del usuario. Lo más destacado son los métodos insertar, modificar y eliminar usuario que estos hacen peticiones a la clase DBBroker para realizar la acción correspondiente en la base de datos. Tenemos que tener en cuenta también el método getLastId para obtener un id nuevo sin que se repita de la base de datos.
* Clase ManajePerfil: Lleva las acciones del perfil, igual que en la clase anterior los más destado es el insertar, modificar y eliminar perfil de la base de datos, sin olvidarnos del getLastId para asignarle un id nuevo al perfil.
* Clase ApiManaje: Lleva el manejo de las acciones con la API. Aquí nos encontramos métodos como obtenerTodosProdcutos o obtenerTodasRecetas las cuales obtienen una lista de productos/recetas sin ningún filtro y los métodos obtenerProductosConFiltros y obtenerRecetasConFiltros los cuales obtienen recetas según los filtros que el usuario desee. Hay otro métodos como obtenerDetalleReceta es una búsqueda secundaria de cada receta para obtener más datos y proporcionar al usuario una mayor información.
* Clase MainWindow(): Lo más a destacar de esta clase son los métodos btnRegistrarse\_Click que sirve para que te lleve a la pantalla de registro y btnIniciarSesicion\_Click que una vez tengas escrito el correo y la contraseña comprueba si lo has escrito bien o no



* 1. **Diseño. Capa de presentación. Interfaces gráficas**

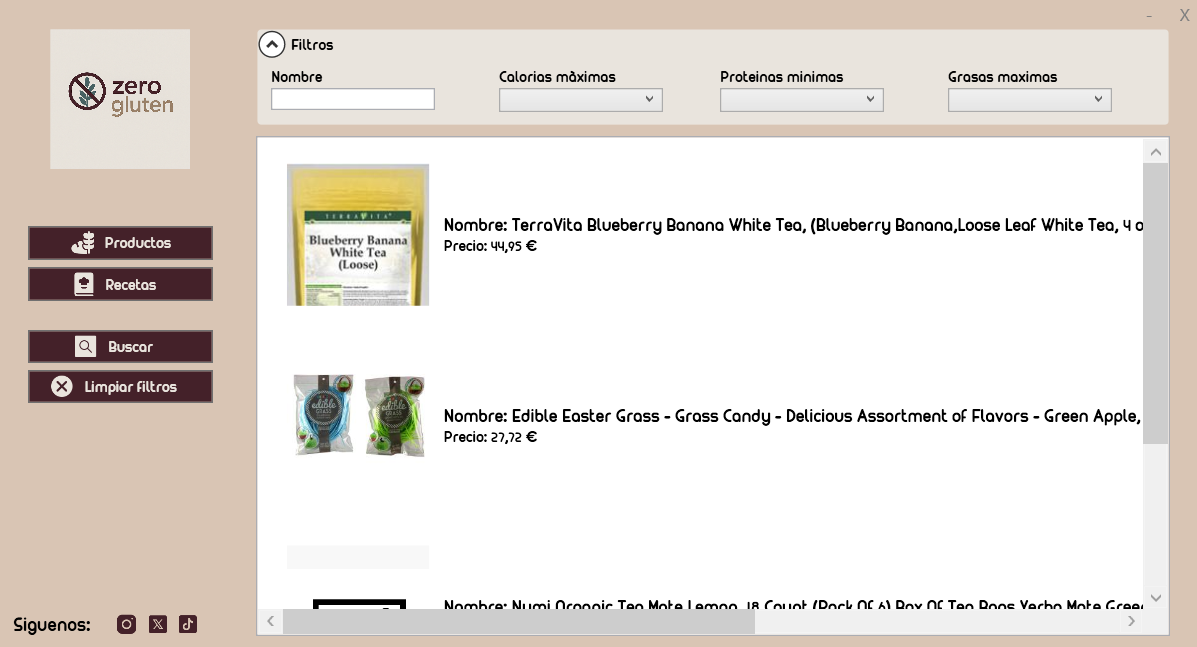
En cuanto al diseño nos hemos centrado en que sea lo más intuitiva y familiar posible, para facilitar la experiencia del usuario incluso para las personas que no estén acostumbradas a utilizar aplicaciones a menudo.

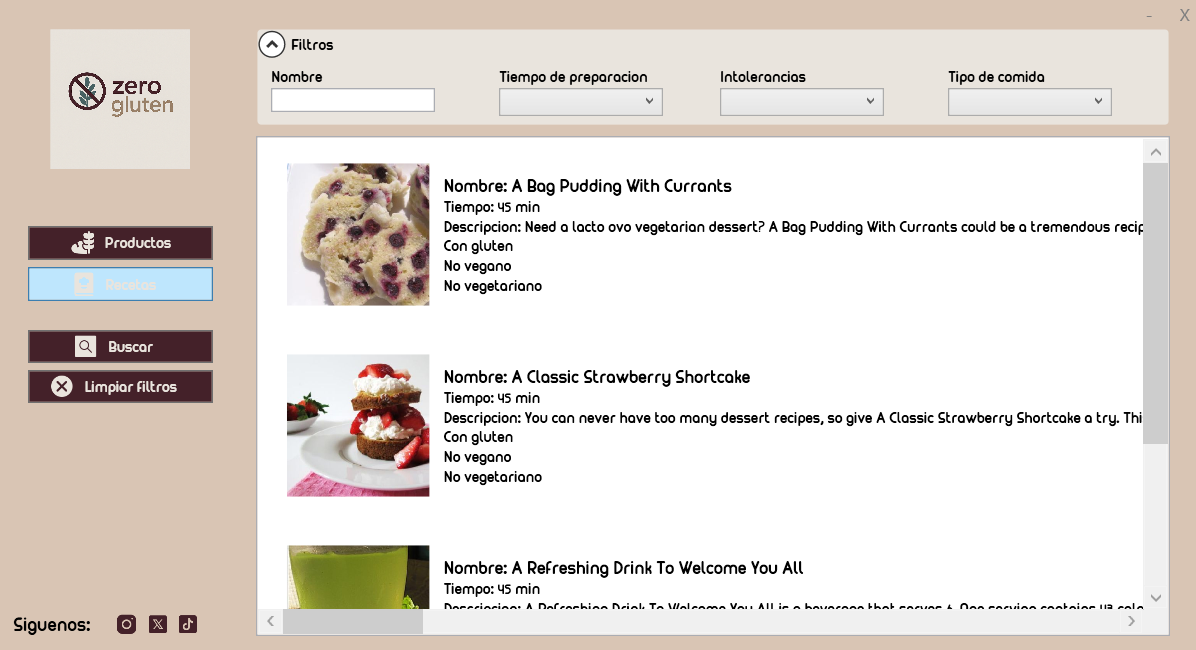
Nuestra aplicación cuenta con 3 ventanas

* + *La ventana de inicio de sesión*: Contiene un campo para que escribas tu nombre y tu contraseña para que puedas acceder a la aplicación, un botón de inicio de sesión y un botón que te lleva a la ventana de registro si es que es la primera vez que ingresas en la aplicación. 
  + *La ventana de registro*: En esta pantalla el usuario debe rellenar un formulario que incluye información básica personal y datos relacionados con su estado físico, necesarios para realizar un estudio del **Índice de Masa Corporal (IMC).** Esta información permitirá a la aplicación ofrecer recomendaciones más personalizadas.
  + *La ventana de funcionalidad principal*: Finalmente cuando te registras o Inicias sesión en la aplicación encontramos la ventana principal que solo es una, pero cambia dinámicamente según estés mostrando productos o recetas.

Lo más destacado son los botones de productos y recetas cuya funcionalidad es cambiar los filtros que puedes utilizar según que quieras obtener. En cuanto a los filtros de productos encontramos diferentes como el nombre del producto, calorías máximas, proteínas mínimas y grasas máximas. En cuanto a los filtros de las recetas encontramos el nombre, igual que en los productos, el tiempo de preparación del plato, intolerancias, y el tipo de comida que desees preparar (plato, postre, bebida…).

También puedes encontrar otros botones como el de buscar que consiste en buscar los productos o recetas según los filtros que hayas seleccionado y el botón de limpiar filtros que consiste en limpiar todos los filtros para que puedas reiniciar tus búsquedas si es que lo necesitas

****

****

Para llevar un seguimiento de las tareas que nos permite organizar mejor nuestro día a día sin dejar funcionalidades a medias hemos utilizado Notion. Para el control de versiones de la aplicación se ha utilizado GitHub con GitHub Desktop

* 1. **Pruebas de Software**

Durante el proceso de desarrollo de la aplicación, se llevaron a cabo distintas pruebas con el objetivo de garantizar el correcto funcionamiento de las funcionalidades implementadas y la fiabilidad de los datos proporcionados. A continuación, se detallan las principales pruebas realizadas:

* **Prueba de funcionamiento de la API:** Se comprobó que la API utilizada responde correctamente a las peticiones realizadas desde la aplicación. Se verificó las conexiones con la base de datos y que se pudiera acceder a los datos de forma eficaz y sin errores. Fue la prueba que nos llevó más tiempo hacer ya que la base de nuestra aplicación se basa en la utilización de la API y queremos que las peticiones, sobre todo una vez el usuario decida filtrar productos o recetas, estén totalmente pulidas.
* **Carga de productos y recetas:** Se realizaron pruebas insertando una gran variedad de productos y recetas en la base de datos, asegurando que la carga de esta información se visualizara correctamente en la interfaz de usuario. Además, se confirmó que los datos se almacenaban y recuperaban de forma adecuada.
* **Pruebas de filtros:** Uno de los puntos clave de la aplicación es la posibilidad de filtrar productos y recetas según distintos criterios (nombre, calorías, proteínas, grasas, etc.). Se llevaron a cabo distintas pruebas para asegurar que estos filtros funcionaban correctamente, devolviendo los resultados esperados según los parámetros establecidos por el usuario.

**4. Implementación e integración**

**4.1 Tecnologías**

En cuanto a las tecnologías implementadas hemos considerado que estas eran las mejores debido a la facilidad que nos proporcionan combinar el frontend con el backend. Las tecnologías utilizadas son:

* Visual Studio 2022 como IDE principal para el desarrollo de nuestra aplicación en C#. En este lenguaje se ha programado la lógica de la aplicación.
* WPF como desarrollo de la interfaz de nuestro usuario.
* MySQL como gestor de base de datos.
* MySQL Workbench como aplicación para diseñar y gestionar de forma interna la base de datos.
* Librerías de C# para encriptar las contraseñas del usuario.
* API Spoonacular: Como base de nuestra aplicación que nos ayuda a obtener los datos e información de los productos y recetas.
* GitHub Desktop y GitHub para el control de versiones.

**4.2 Herramientas utilizadas**

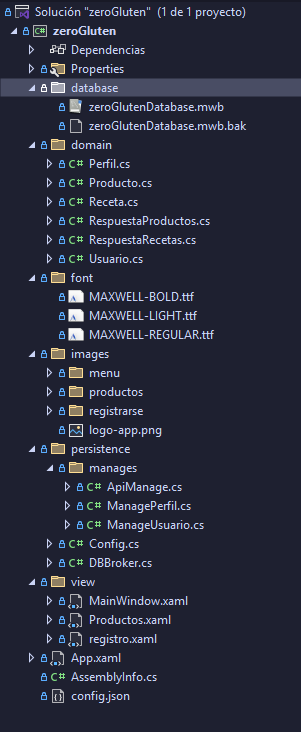
En cuanto a las herramientas utilizadas hemos necesitado tanto aplicaciones como páginas web que nos han ayudado ha realizar la aplicación de forma rápida y sencilla.

* Visual Studio 2022: Ha sido el entorno de desarrollo principal de nuestra aplicación aquí se ha desarrollado tanto el backend como el fronted.
* MySQL Workbrench: En esta aplicación se ha desarrollado toda la lógica de la base de datos.
* Draw.io: he desarrollado varios esquemas necesarios como el diagrama de casos de uso y el diagrama entidad relación.
* Planttext.com: Me ha ayudado a desarrollar el diagrama de clases
* Postman: Me ha ayudado con las pruebas de las peticiones de mi programa a la API
* Canva: Para realizar todos los esquemas e imágenes de forma muy visual y atractiva para el usuario.

**4.3 Detalles de la implementación**

Nuestra aplicación se compone de un único proyecto el cual contiene 6 paquetes donde se dividen las clases de las diferentes lógicas de la aplicación.

* Paquete Database: Este paquete contiene dos archivos donde está guardado el modelo de la base de datos.
* Paquete Domain: Representa las clases principales de la aplicación. Estas clases destacan por tener los atributos principales de los objetos que vamos a utilizar. Destacan la clase Usuario y Perfil que guarda la información del usuario. Las clases Recetas y Productos que contiene los atributos necesarios para guardar la máxima información posible de los alimentos y recetas. Y las clases RespuestaProductos y RespuestaRecetas que nos sirven de apoyo para deserializar los JSON de las respuestas de la API.
* Paquete Font: Aquí se guara la fuente personalizada de la interfaz y sus diferentes estilos
* Paquete images: En este paquete guardamos todas las imágenes necesarias de nuestra aplicación ya sean logos o imágenes. Este paquete a su vez se divide en otros los cuales nos ayudan a tener las imágenes de forma ordenada según vayan en una pantalla u otra. La imagen más destacada es la del logo de nuestra aplicación que se ve visible en las pantallas de inicio de sesión y registro
* Paquete persistence: Este paquete que contiene una carpeta más en su interior llamada manages la funcionalidad consiste en manejar/utilizar clases creadas en el paquete domain. Estas clases tiene la gran parte de la funcionalidad de la aplicación. Destacamos las clases de ApiManage que es la que se encarga de realizar las peticiones a la API, el ManagePerfil y el ManageUsuario las cuales hacen todas las funciones necesarias en cuanto al usuario y perfil como insertarlos en la base de datos, eliminarlos etc. También mencionar que la clase Config.cs sirve para leer la configuración de un archivo de configuración.json y la clase DBBroker es la que tiene toda la interacción con la base de datos.
* Paquete view: Contiene todas las clases dedicadas a la interfaz gráfica (.xaml), la clase MainWindow es la ventana principal de la aplicación o (inicio de sesión), Productos.xaml se muestran los productos y filtros y Registro.xaml es la pantalla donde se encuentra el formulario que el usuario tiene que rellenar para registrarse.

****

**5. Evaluación del proyecto**

En este proyecto se han cumplido todos los requisitos que se establecieron inicialmente, haciendo uso de los conocimientos adquiridos a lo largo de estos dos últimos años. La aplicación final está alineada con nuestro nivel de experiencia y se ha desarrollado utilizando tecnologías que han sido explicadas, trabajadas y probadas en clase. ZeroGluten es una herramienta funcional, pensada para ser completamente operativa y útil desde su entrega, aunque también deja margen para futuras mejoras tanto a nivel técnico como funcional. Este proyecto ha supuesto una excelente oportunidad para aplicar de forma práctica todo lo aprendido, y refuerza nuestra preparación para afrontar nuevos retos dentro del ámbito profesional del desarrollo de software.

**6. Conclusión**

**6.1 Dificultades encontradas y las soluciones adoptadas**

Como mayor dificultad a destacar la encontramos en el aspecto gráfico y en cuanto al manejo de las API. En cuanto al aspecto gráfico ha sido complicado encontrar unos colores que se ajusten al contenido de la aplicación y a su vez el propio diseño de las ventanas ha costado hacerlo simple y que a la vez recoja toda la información necesaria para el usuario, como solución nos hemos apoyado ventanas de otros proyectos hechos en clase. En el aspecto del manejo de la API hemos tenido problemas a la hora de entender bien como debíamos leer la información que esta nos proporciona devuelta de una petición nuestra, sin embargo, leyendo la documentación hemos podido solucionar cualquier problema al respecto.

**6.2 Desviaciones temporales o técnicos y soluciones adoptadas**

En cuanto a las desviaciones no hay ninguna que descar puesto que el proyecto se empezó con tiempo de sobra previendo que algunas acciones de la aplicación se podrían alargar más de lo esperado. La distribución del tiempo ha sido la correcta en su mayoría dando tiempo a realizar la acción, comprobarla y corregir aquellas cosas que se necesitan. En general el proyecto ha ido a tiempo según todas las fases estipuladas al principio.

**6.3 Propuestas de mejora del proyecto**

En cuanto a propuestas de mejora ya hay varias sobre la mesa.

* ChatBot: Se ha planteado la idea de implementar una IA para que los usuarios puedan expresar sus necesidades y que esta devuelva productos y/o recetas acordes a lo que el usuario necesite.
* Ampliación en los filtros: En un futuro es posible ampliar la lista de filtros para que los usuarios puedan elegir con mayor exactitud lo que necesiten.
* Mejora del código fuente: E internamente podemos hablar de que el código tiene una gran capacidad de optimización que haga la aplicación más ligera para los usuarios.

**7. Bibliografía**

En cuanto al desarrollo de la aplicación para resolver las dificultades en cuanto al código hemos utilizado las siguientes páginas:

* W3schools. Guardada el 24 de mayo de 2025: <https://www.w3schools.com/>
* Página oficial de C# Microsoft. Guardada el 24 de mayo de 2025: <https://learn.microsoft.com/es-es/dotnet/csharp/>

En cuanto a la API utilizada:

* Spoonacular. Guardada el 24 de mayo de 2025: <https://spoonacular.com/food-api>

**8. Anexos**

**8.1 Documentación metodología de desarrollo de proyecto**

**8.2 Manual de usuario**

**8.3 Manual de instalación y configuración de la herramienta**

1. Instalación de MySQL Worbrench

2. Ejecutar el siguiente script en la base de datos

3. Instalar la aplicación

4. Ejecutar

**8.4 Otros…**