

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

CHEF MANAGER



TRABAJO FIN DE GRADO

Tutor: Antonio Carmona Lara

Autor: David Chavarria Paccha

ÍNDICE

1. RESUMEN	3
2. ABSTRACT	4
3. JUSTIFICACIÓN DEL PROYECTO	5
4. ANÁLISIS DEL MERCADO	7
5. OBJETIVOS	8
5.1. RFTP	8
6. DISEÑO	14
6.1. DIAGRAMA DE CLASES	14
6.2. DIAGRAMA DE BASE DE DATOS	16
6.3. DIAGRAMA INTERFACES	17
7. TECNOLOGÍAS UTILIZADAS	25
7.1. VISUAL STUDIO	25
7.1.1. PAQUETES NUGGET	25
7.2. .NET MAUI	26
7.3. FIREBASE	27
7.3.1 REALTIME DATABASE	27
7.3.2 CLOUD STORAGE	28
7.4. OPEN WEATHER API	28
8. METODOLOGÍA	29
8.1. DIAGRAMA DE GANTT	31
8.2. GIT	32
9. CASOS DE USO	34
10. BIBLIOGRAFÍA	40

1. Resumen

Chef Manager es una aplicación para la correcta gestión de restaurantes cuyo propósito es simplificar y optimizar las operaciones diarias en la industria de la hostelería. La aplicación sustenta diferentes aspectos de la gestión de restaurantes, como la administración de proveedores, empleados, inventario y de los beneficios/pérdidas. El objetivo de Chef Manager es ofrecer a sus usuarios una herramienta eficiente para reducir las tareas administrativas, mejorar la comunicación y optimizar la gestión de su negocio. Entre sus principales características se encuentran:

1. *Administración de proveedores:* Permite a los usuarios gestionar sus relaciones con los proveedores, controlar el historial de compras, realizar el seguimiento de los productos y albergar la información de todos los proveedores de una manera organizada,
2. *Gestión de empleados:* Apartado en el cual se administra el personal. Se almacena toda la información de cada empleado como el cargo, la nómina, turnos/horarios y su rendimiento.
3. *Control del inventario:* Ayuda para que los usuarios realicen un seguimiento de las existencias, establecer niveles de reordenamiento y minimizar el desperdicio de alimentos.
4. *Caja:* En esta sección se administrará todo el dinero obtenido en la caja, tanto diario, como semanalmente.
5. *Calendario:* Página que muestra el horario mensual que tendrá el usuario. Días festivos, no lectivos, periodicidad de los proveedores etc.
6. *Notas:* Breve apartado para escribir cualquier anotación sobre la empresa que se quiera hacer.

En resumen, Chef Manager está dirigido a propietarios y gerentes de restaurantes que buscan mejorar su eficiencia y rentabilidad ya que es una herramienta esencial que combina una interfaz intuitiva y funciones avanzadas.

2. Abstract

Chef Manager is an application for the correct management of restaurants whose purpose is to simplify and optimize daily operations in the hospitality industry. The application supports different aspects of restaurant management, such as supplier, employee, inventory, and profit/loss management. The objective of Chef Manager is to offer its users an efficient tool to reduce administrative tasks, improve communication and optimize the management of their business. Among its main features are:

1. *Supplier Management:* Allows users to manage their relationships with suppliers, control purchasing history, track products and house information on all suppliers in an organized way,
2. *Employee management:* Section in which personnel is managed. All information about each employee is stored, such as position, payroll, shifts/schedules, and performance.
3. *Inventory Control:* Helps users track stock, set reorder levels, and minimize food waste.
4. *Cash:* In this section all the money obtained in the cash register will be managed, both daily and weekly.
5. *Calendar:* Page that shows the monthly schedule that the user will have. Holidays, non-school days, frequency of suppliers, etc.
6. *Notes:* Brief section to write any notes about the company that you want to make.

In short, Chef Manager is aimed at restaurant owners and managers looking to improve their efficiency and profitability as it is an essential tool that combines an intuitive interface and advanced features.

3. Justificación del proyecto

En la actualidad, la gestión eficiente de un restaurante es fundamental para su éxito y competitividad en un mercado cada vez más exigente y digitalizado. Añadiendo además que la hostelería está en su máxima expansión en España. El uso de una aplicación dedicada a la gestión de un restaurante ofrece numerosos beneficios que resuelven muchos problemas. Por experiencias personales, la mayoría de los hosteleros gestionan su negocio mediante cientos de cuadernos, cientos de bolígrafos que ante cualquier descuido (se te pierde o se te cae una taza de café etc.) puedes terminar perdiendo mucho tiempo o información.

Según datos de la *National Restaurant Association* ¹, el 73% de los restaurantes americanos consideran que la tecnología les ayuda a mejorar la eficiencia operativa. Por lo tanto, siendo España un país que alberga aproximadamente 800.000 restaurantes, una app de gestión que permita optimizar procesos como gestión de inventario, control de costos... puede conllevar una economía más ágil y rentable. Chef Manager satisface todas estas necesidades, es eficaz para mejorar la gestión de restaurantes ya que puede disminuir el tiempo empleado en labores administrativas, mejorar la comunicación con proveedores y empleados, y optimizar la gestión de inventarios y reservas mediante una interfaz sencilla y fácil de manejar a la par de útil.

La cifra de negocios del sector servicios por sector de actividad en marzo de 2024 en España

	Anual (%)	Media en lo que va de año (%)
Servicios de alojamiento	17,2	15,4
Servicios de información	13,6	10,3
Hostelería	12,1	10,8
Actividades de agencias de viajes, operadores turísticos, servicios de reservas y actividades relaci	11	10,9
Servicios de comidas y bebidas	10	9
Transporte por taxi	9,8	12,5
Actividades de seguridad e investigación	8,1	10,1
Programación, consultoría y otras actividades relacionadas con la informática	7,1	10,7

Fuente: INE, www.epdata.es

En resumen, el principal objetivo de este proyecto es poder brindar ayuda a muchos hosteleros con su negocio y poder tener más impacto en la industria. Contribuir al crecimiento y la prosperidad del sector hostelero, apoyando a los emprendedores y profesionales que trabajan incansablemente para ofrecer una experiencia gastronómica de calidad. Con esta aplicación, espero simplificar la gestión diaria de los restaurantes y permitir que los hosteleros se enfoquen en lo que más les apasiona: crear mejores platos y generar más cercanía con sus comensales.

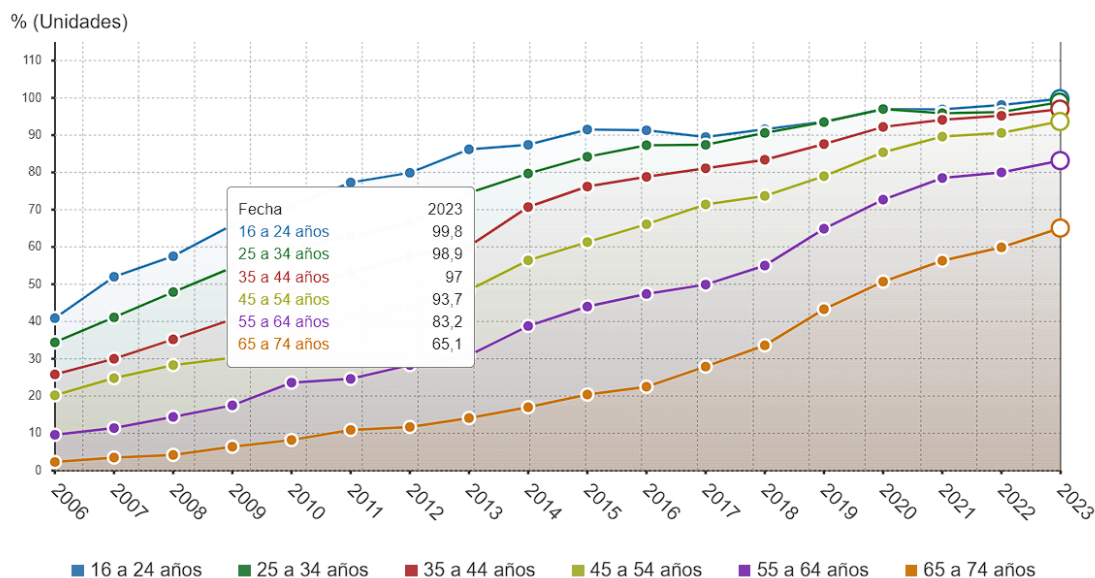
¹ National Restaurant Association es una asociación empresarial de la industria de restaurantes en los Estados Unidos que representa a más de 380.000 restaurantes. También opera la Fundación Educativa de la Asociación Nacional de Restaurantes. La asociación fue fundada en 1919 y tiene su sede en Washington, D.C.

4. Análisis de mercado

Actualmente, el mercado ofrece diversas soluciones de software para la gestión de restaurantes. Sin embargo, muchas de estas aplicaciones se centran en aspectos más generales y no ofrecen una solución concreta. Además, suelen ser complejos y no siempre fáciles de usar, lo que dificulta su adopción por parte del personal con distintos niveles de habilidades tecnológicas. Las aplicaciones existentes suelen ser eficaces para gestionar reservas, pedidos y pagos, pero muchas no integran eficazmente el control de inventario, la gestión del personal y de los proveedores, ni ofrecen una visión clara de la salud financiera del restaurante. El vacío que busca llenar nuestra aplicación es la integración de todas estas funciones en una interfaz intuitiva y accesible.

Además, el optar por una aplicación tanto móvil como web significa ganar una buena posición en el mercado debido al avance del alto uso de la tecnología por edad, por lo tanto al ser muy intuitiva será un punto positivo.

Evolución de personas que usan diariamente Internet en España por grupo de edad



Fuente: INE, www.epdata.es

5. Objetivos

Requisitos Funcionales

R001- Iniciar la aplicación.

R002- Registrar usuario en la aplicación

R003- Inmediatamente registrar restaurante en la aplicación.

R004- Establecer relación usuario-restaurante.

R005- Iniciar sesión en la aplicación.

R006- El usuario debe poder crear, editar, eliminar y ver sus propios empleados, proveedores, productos y notas, es decir, personalizar su aplicación a su gusto.

R007- El administrador de la app tiene que ser capaz de controlar todo lo ingresado en la base de datos desde la app

R008- Implementar módulo de búsqueda en cada vista

R009- Implementar API

Requisitos no funcionales:

R010- El software debe validar todos los campos para no crear conflictos con la base de datos ni con la interfaz.

R011- La interfaz tiene que ser intuitiva y fácil de entender.

R012- Validar que se suben solo imágenes a la aplicación.

5.1. RFTP

Desarrollo Requisitos

R001: Iniciar la aplicación

└─ R001F01: Lanzar el splash screen

└─ R001F01T01: Crear el logo

└─ R001F01T01P01: Visualizar correctamente el logo (buena calidad)

└─ R001F01T01P02: Encuadre correcto

└─ R001F01T02: Diseñar la interfaz de bienvenida

└─ R001F01T02P01: Implementar navegación a Login y Registro

└─ R001F01T02P02: Visualizar correctamente la imagen fondo y que encuadren los colores

R002: Registrar usuario en la aplicación

- └─ R002F01: Permite al usuario registrarse en el sistema
 - └─ R002F01T01: Diseñar la interfaz
 - └─ R002F01T01P01: Comprobar elementos funcionales
 - └─ R002F01T01P02: Comprobar correcta visualización
 - └─ R001F02T02: Validaciones en los campos
 - └─ R002F01T02P01: Comprobar la navegación entre clases dependiendo del usuario introducido
 - └─ R002F01T02P02: Comprobar las alertas establecidas por errores en el registro.

R003: Inmediatamente registrar restaurante en la aplicación.

- └─ R003F01: Permite al usuario registrar su restaurante en el sistema
 - └─ R003F01T01: Diseñar la interfaz
 - └─ R003F01T01P01: Comprobar elementos funcionales
 - └─ R003F01T01P02: Comprobar correcta visualización
 - └─ R003F02T02: Validaciones en los campos
 - └─ R003F01T02P01: Comprobar la navegación entre clases dependiendo del restaurante
 - └─ R003F01T02P02: Comprobar las alertas establecidas por errores en el registro.
- └─ R003F02T03: Eliminar usuario si se cierra la aplicación sin haber registrado un restaurante
 - └─ R003F01T03P01: Comprobar que el usuario se elimina en la base de datos para que no haya contradicciones en la bbdd (Ej: Usuario sin restaurante asignado)

R004: Establecer relación usuario-restaurante.

- └─ R004F01: Verificar si el usuario tiene bien asignado su restaurante
 - └─ R004F01T01: Comprobar en las bases de datos
 - └─ R004F01T01P01: Verificar UsuarioDatabase
 - └─ R004F01T01P02: Verificar RestauranteDatabase
 - └─ R004F01T01P03: Comprobar los ids

R005: Iniciar sesión en la aplicación

- └─ R005F01: Autenticación

- └─ R005F01T01: Validaciones

- └─ R005F01T01P01: Comprobar que funcionan las validaciones de los campos (Ej: que no estén vacíos).

- └─ R005F01T01P02: Probar que el usuario autenticado con sus credenciales pueda acceder a las funcionalidades permitidas según su id. (EJ: Que vaya a la Vista Principal o a la Vista Administrador).

- └─ R005F01T01P03: Comprobar los campos con la base de datos

- └─ R005F02: Creación de sesión

- └─ R005F01T01: Almacenar datos del usuario

- └─ R005F01T01P01: Comprobar que el Id del restaurante se corresponde correctamente con el restaurante id del usuario. Para el manejo de la aplicación

R006: El usuario debe poder crear, editar, eliminar y ver sus propios empleados, proveedores, productos y notas, es decir, personalizar su aplicación a su gusto.

- └─ R006F01: Implementar módulo CRUD

- └─ R006F01T01: Crear

- └─ R006F01T01P01: Verificar que se puedan crear nuevos registros en la base de datos correctamente.

- └─ R006F01T01P02: Validar que todos los campos obligatorios se completen al momento de crear un nuevo registro y con su tipo de dato.

- └─ R006F01T01P03: Comprobar que se genere un identificador único para cada nuevo registro creado.

- └─ R006F01T02: Leer

- └─ R006F01T02P01: Verificar que se puedan recuperar los registros existentes de la base de datos de forma correcta.

- └─ R006F01T02P02: Comprobar que se puedan filtrar los registros según criterios específicos.
- └─ R006F01T02P01: Validar que se puedan ordenar los registros de acuerdo con diferentes campos.
- └─ R006F01T03: Actualizar
 - └─ R006F01T03P01: Validar que, al actualizar las vistas, la transición sea suave y sin errores visuales o de funcionamiento
 - └─ R006F01T03P02: Comprobar que los cambios realizados en un registro se reflejen correctamente en la base de datos y en la interfaz.
 - └─ R006F01T03P03: Validar que se puedan actualizar múltiples campos a la vez si es necesario.
- └─ R006F01T04: Eliminar
 - └─ R006F01T04P01: Verificar que se puedan eliminar registros de la base de datos de forma segura.
 - └─ R006F01T04P02: Comprobar que, al eliminar un registro, se actualicen correctamente las relaciones con otros registros.
 - └─ R006F01T04P03: Validar que se implementen medidas de seguridad para prevenir eliminaciones accidentales.

R007: El/Los administradores de la app tiene que ser capaz de controlar todo lo ingresado en la base de datos desde la propia app

- └─ R007F01: Implementar el módulo CRUD igual que el R006
 - └─ R007F01T01: Diseño vista
 - └─ R001F01T01P01: Correcta funcionalidad de los botones de navegación
 - └─ R001F01T01P02: Correcta visualización de los datos dependiendo de donde clique el usuario

R008: Implementar módulo de búsqueda en cada vista

- └─ R008F01: Crear la barra de búsqueda
 - └─ R008F01T01: Atribuirle al modelo de datos de la vista
 - └─ R008F01T01P01: Comprobar si obtiene bien la información de la base de datos y hace el filtro

R009: Implementar API

- └─ R009F01: Dependiendo del lugar seleccionado del restaurante, poner datos
 - └─ R009F01T01: Conexión al api
 - └─ R009F01T01P01: Buscar el api
 - └─ R009F01T01P02: Obtener requisitos para dicha conexión
 - └─ R009F01T02: Crear modelo de datos
 - └─ R009F01T02P01: Atribuir datos a la interfaz
 - └─ R009F01T02P02: Comprobar que se muestran correctamente

R010: El software debe validar todos los campos para no crear conflictos con la base de datos ni con la interfaz.

- └─ R010F01: Datos en la interfaz
 - └─ R010F01T01: Establecer un máximo de caracteres por campo
 - └─ R010F01T01P01: Comprobar si daña la interfaz, descuadrando algún elemento

R011: La interfaz tiene que ser intuitiva y fácil de entender.

- └─ R011F01: Implementar alertas
 - └─ R011F01T01: Generar una por cada acción del usuario
 - └─ R011F01T01P01: Comprobar que la información de la alerta es correcta
 - └─ R011F01T01P02: Verificar que salte adecuadamente la alerta en el momento preciso
- └─ R011F02: Navegación entre clases
 - └─ R011F01T01: Validar que, al navegar entre vistas, la transición sea suave y sin errores visuales o de funcionamiento

└─ R011F01T01P01: Verificar que los elementos de navegación, como botones o enlaces, funcionen correctamente al dirigir al usuario a las vistas correspondientes

└─ R011F01T01P02: Validar que, al regresar a vistas anteriores, la aplicación mantenga el estado adecuado y no se pierda información importante

R012: Validar que se suben solo imágenes a la aplicación.

└─ R012F01: Subir imágenes con su correcto formato

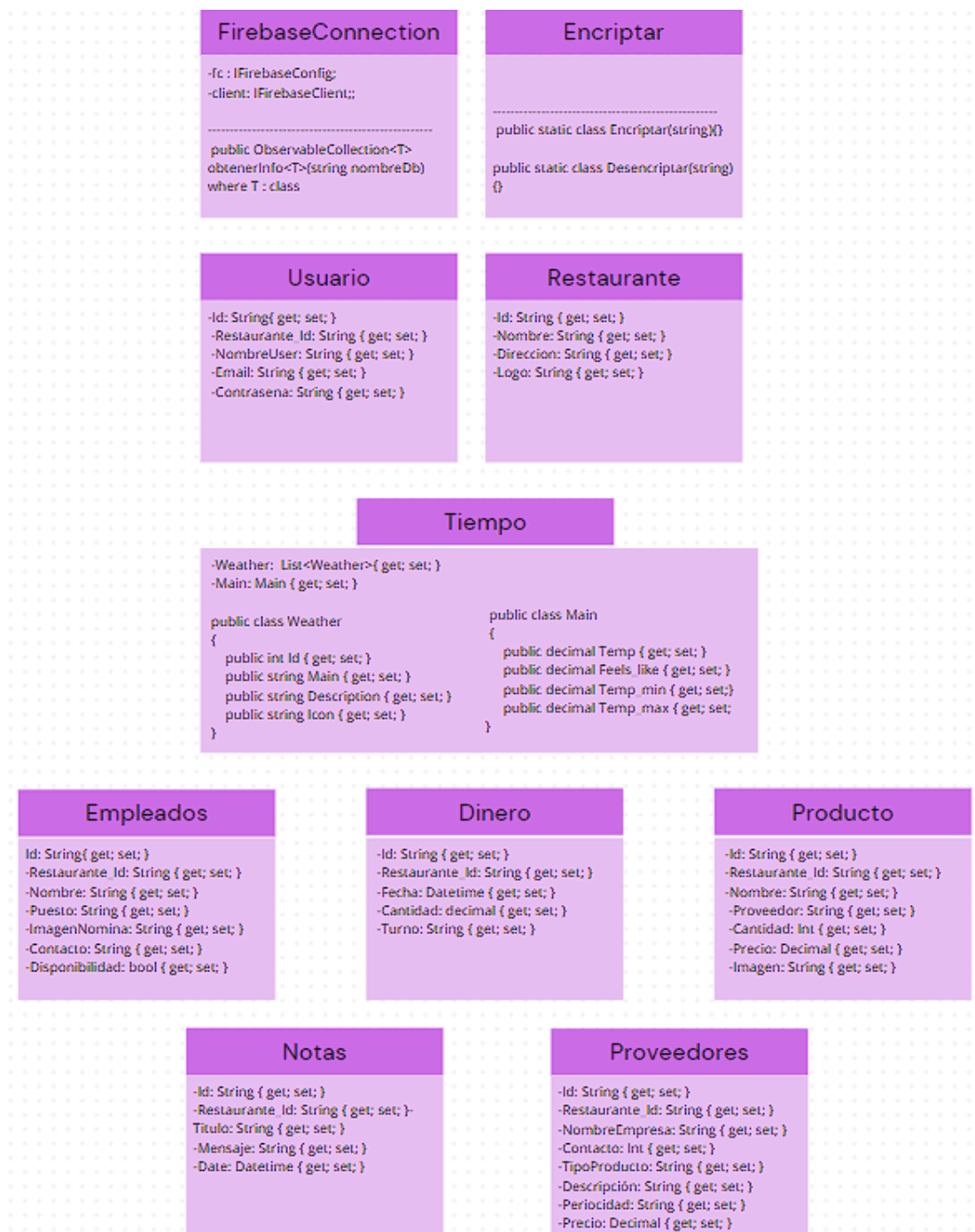
└─ R012F01T01: Establecer formatos

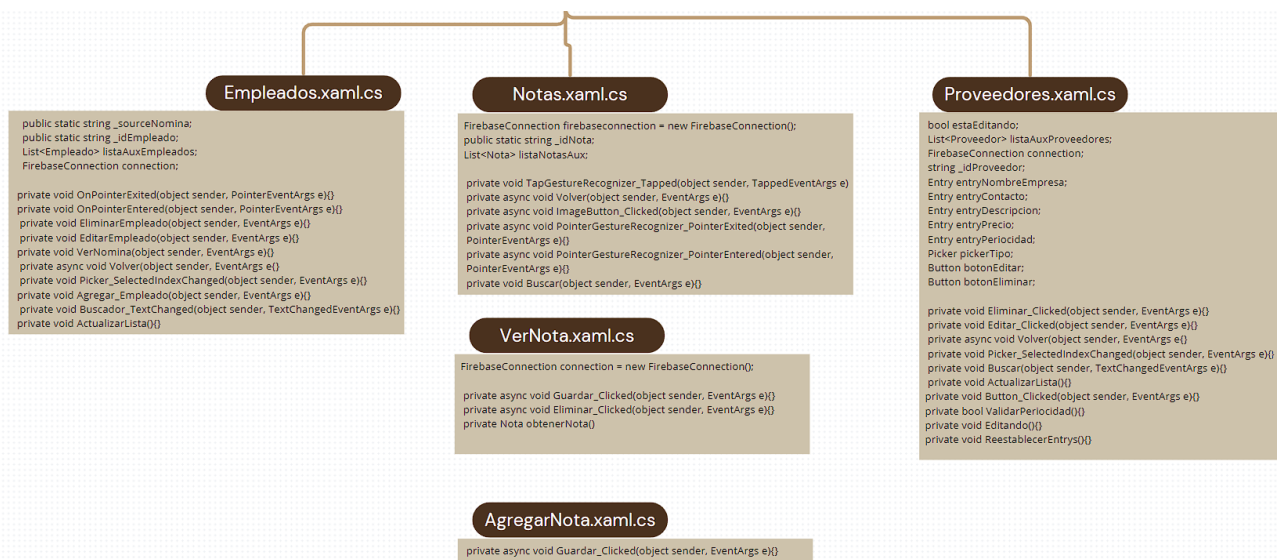
└─ R012F01T01P01: Verificar solo con las extensiones de imágenes

6. Diseño

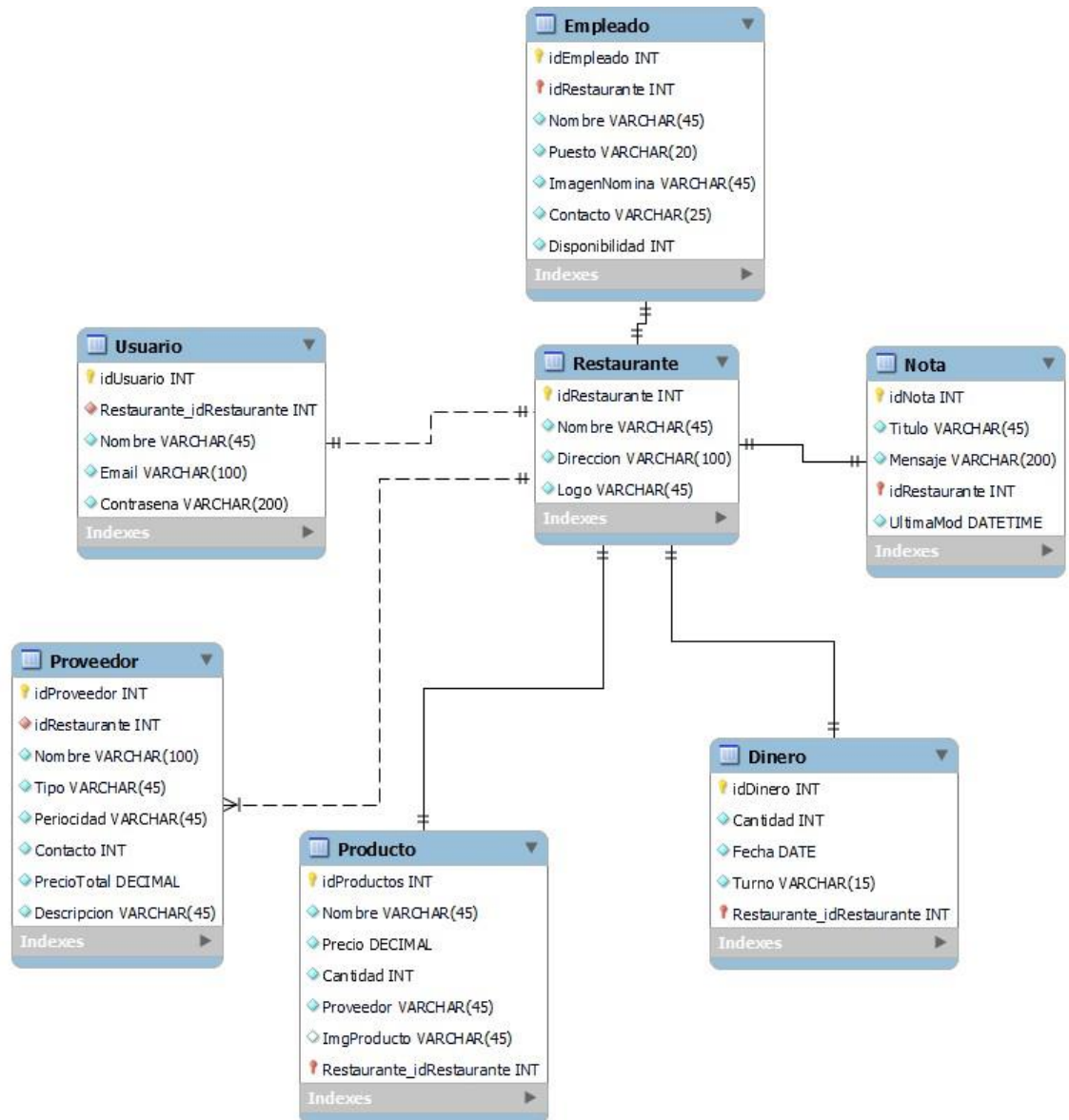
6.1. Diagrama de clases / Flujo de navegación

Al ser un MVVM (Modelo-Vista-VistaModelo) la aplicación se separa en tres capas: la vista, el modelo y el modelo de vista. Esta separación permite una mayor flexibilidad, escalabilidad y mantenibilidad en el desarrollo de aplicaciones. Por lo tanto el modelo de vista se encarga de manejar los datos y las operaciones de negocio unificando la vista (.xaml) y el modelo (.cs).

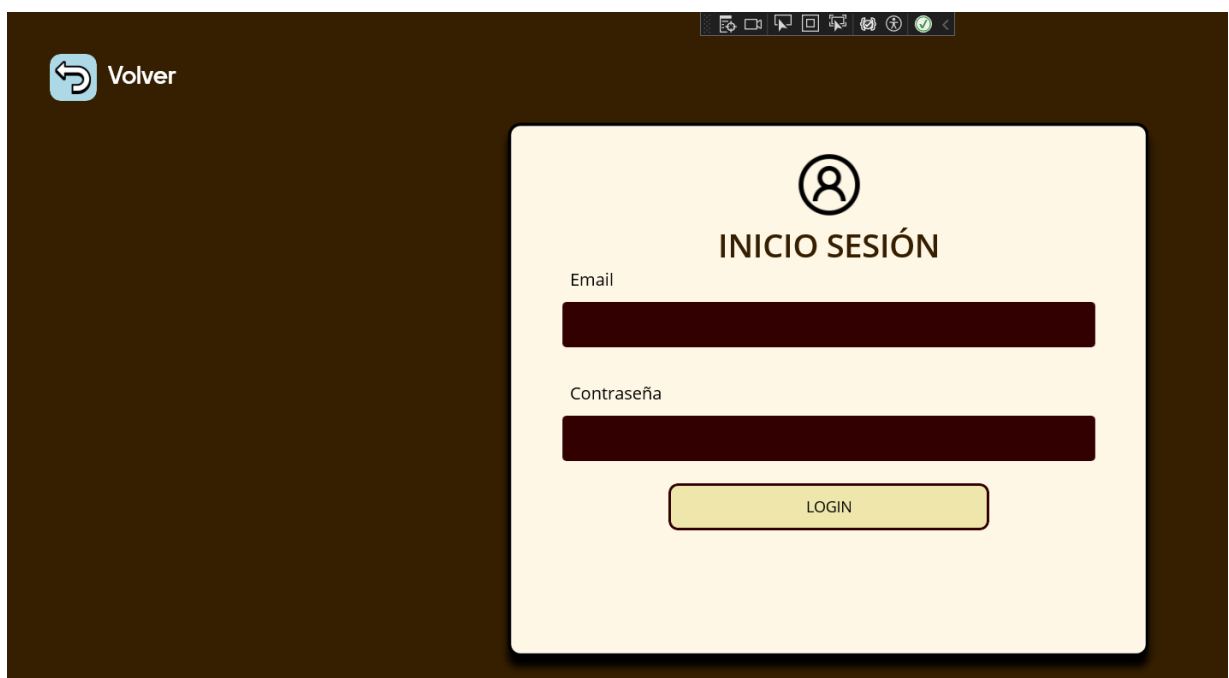




6.2. Diagrama de base de datos (No relacionable)



6.3. Diagrama interfaces





REGISTRO RESTAURANTE

Nombre

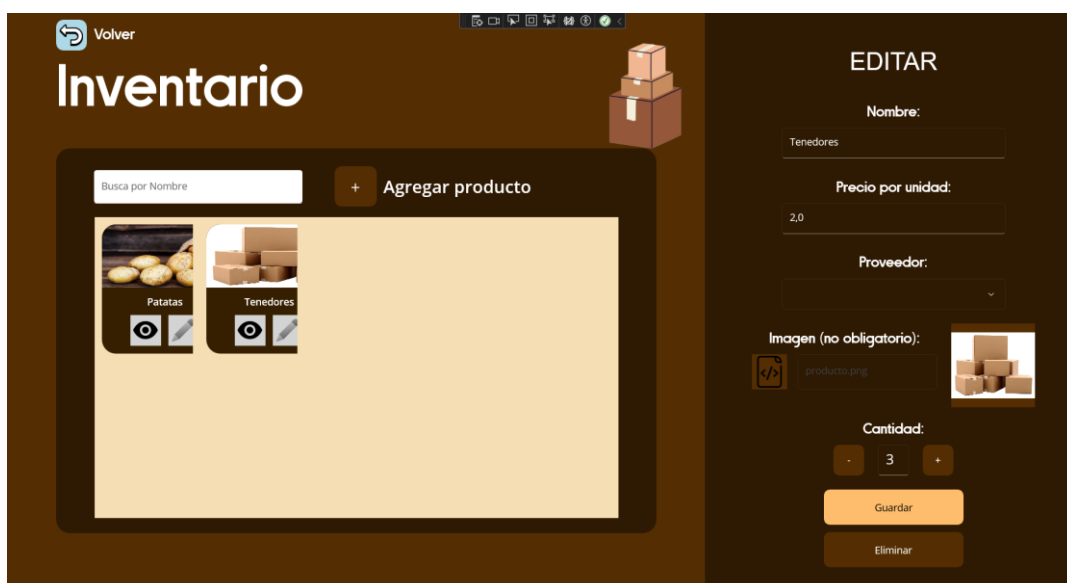
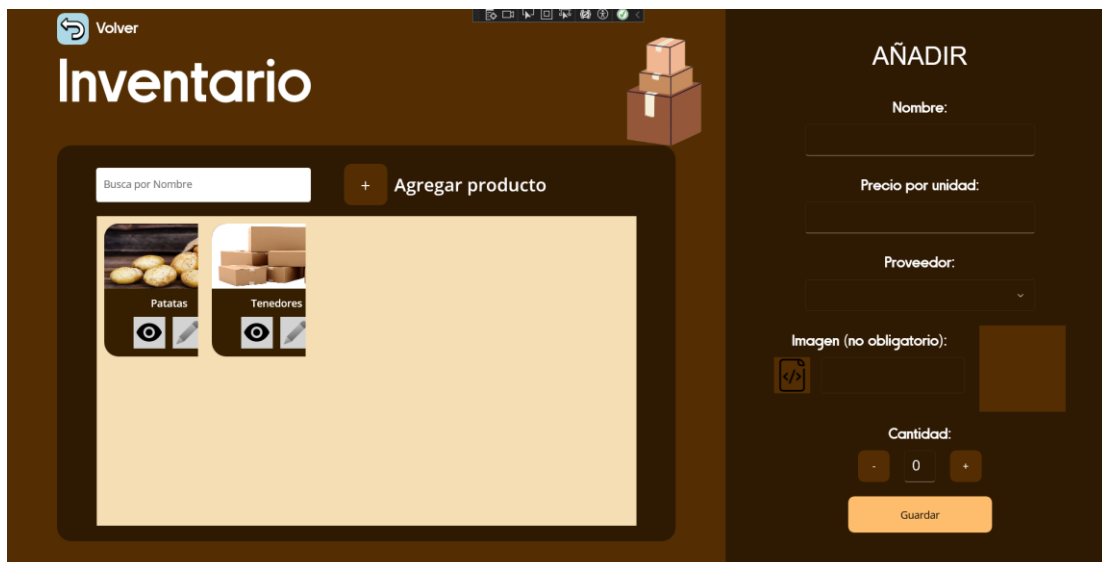
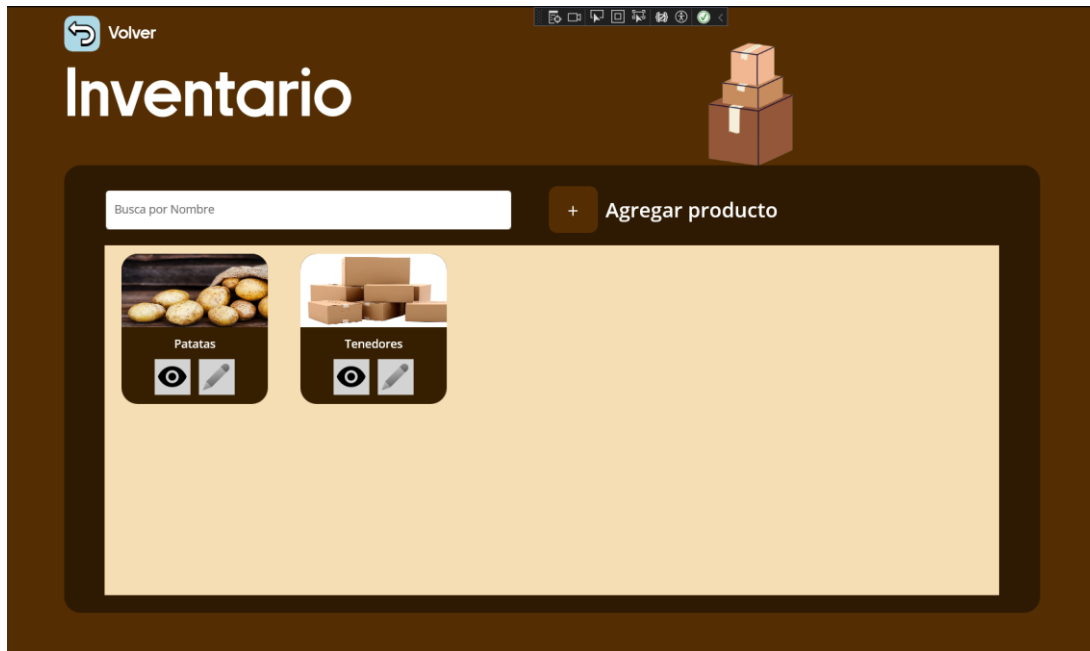
Dirección

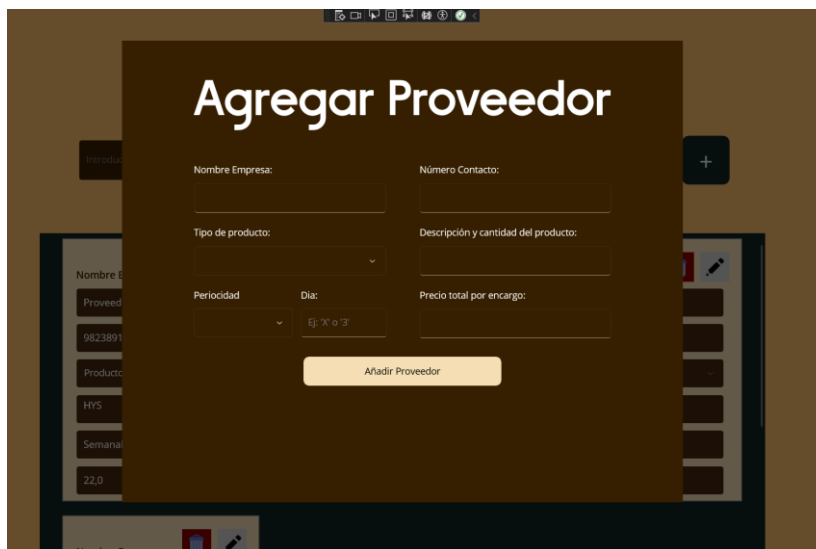
Logo



Registrar








[Volver](#)

Caja

☐ Mezclar filtros

0€ - 1000€ Día: 6/9/2024 Turno: ▼

No hay ningún registro



Agregar recuento:

Turno: ▼ 0€

Día: 6/9/2024 Cantidad:

[Agregar](#)

[Volver](#)

Caja

☐ Mezclar filtros

0€ - 1000€ Día: 5/31/2024 Turno: ▼

Fecha: 28/05/2024 0:00:00	Turno: Tarde	Cantidad: 423,0€	
Fecha: 27/05/2024 0:00:00	Turno: Mañana	Cantidad: 48,0€	
Fecha: 29/05/2024 0:00:00	Turno: Mañana	Cantidad: 88,0€	
Fecha: 30/05/2024 0:00:00	Turno: Tarde	Cantidad: 333,4€	

[Ver recuento semanal/mensual](#)



Agregar recuento:

Turno: ▼ 0€

Día: 5/31/2024 Cantidad:

[Agregar](#)

Recuento semanal/mensual


Fecha Inicio: 5/31/2024 Fecha Fin: 5/31/2024 [Ver](#)

Turno de mañana:

Turno de tarde:

Total:

[Volver](#)

 Volver

CALENDARIO

←

Mayo

→

LUN	MAR	MIÉ	JUE	VIE	SÁB	DOM
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

8 may 2024 ↑

PEpes

Tipo: Productos de limpieza e higiene	Precio por encargo: 323,0 €	Periodicidad: Mensual-8
Descripción: Cosas de limpiar	Contacto: 324353453	

 Volver

Empleados

Nombre Empleado

Filtrar por:

Número empleados: 2



Nombre: Antonio

Puesto: Encargado

Ver nómina: ☒

Dispon: ☐ Off

Contacto: 672797922

Modificar: 



Nombre: Pedro

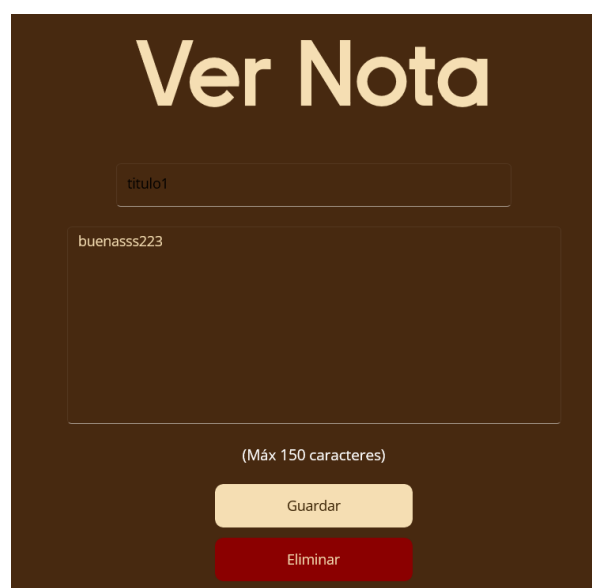
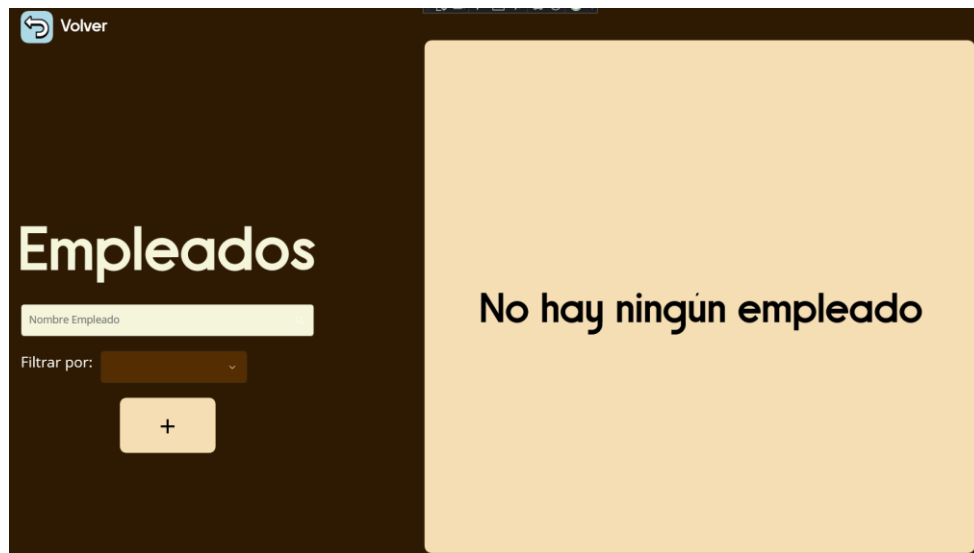
Puesto: Limpiador

Ver nómina: ☒

Dispon: ☒ On

Contacto: joselete23@gmail.com

Modificar: 




Agregar Nota

(Máx 150 caracteres)

Abrir/Cerrar Menú

Cerrar sesión



Administrador
Admin

USUARIOS

RESTAURANTES

PRODUCTOS

PROVEEDORES

EMPLEADOS

DINERO

NOTAS







USUARIOS

Mostrar

registros

Añadir usuario

Buscar por NombreUser

Restaurante_ID	NombreUser	Email	Acciones
ed8a7e73-070c-4c3c-UsuarioDos a369-a97c08e16c9a		user2@gmail.com	 
EsAdmin	Admin	Admin@gmail.com	 
5dc4f715-fe7e-4e35- David Chavarria a9da-01b55dff01b3		david@gmail.com	 







USUARIOS

Mostrar

registros

Añadir usuario

Buscar por NombreUser

Restaurante_ID	NombreUser	Email	Acciones
ed8a7e73-070c-4c3c-UsuarioDos a369-a97c08e16c9a		user2@gmail.com	 
EsAdmin	Admin	Admin@gmail.com	 
5dc4f715-fe7e-4e35- David Chavarria a9da-01b55dff01b3		david@gmail.com	 

7. Tecnologías utilizadas

En este apartado se comentarán las diferentes herramientas y tecnologías utilizadas en el proyecto.

7.1. Visual Studio


Como entorno de desarrollo integrado (IDE) para crear y gestionar el código tanto como la parte visual como la integral del proyecto se utilizó Visual Studio. Es un entorno de desarrollo integrado para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Fortran, Visual Basic (en mi caso C#).

Incluye compiladores, herramientas de completado de código, diseñadores gráficos y muchas más funciones para mejorar el proceso de desarrollo de software, además de bibliotecas de código reutilizable (paquetes nugget)



7.1.1. Paquetes Nugget

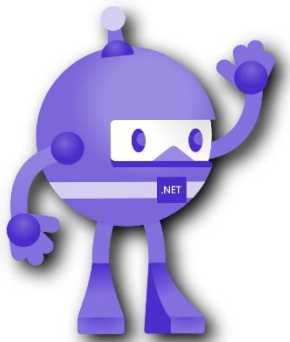
Los paquetes NuGet son bibliotecas de código reutilizable que se pueden agregar fácilmente a proyectos de Visual Studio para proporcionar funcionalidades adicionales. En mi caso los que utilicé son:

- CommunityToolkit.Maui  / CommunityToolkit.Mvvm  : Estos paquetes se utilizan para el desarrollo de aplicaciones con .NET MAUI. Incluye animaciones, comportamientos, convertidores, efectos y asistentes que simplifican tareas comunes para desarrolladores al compilar aplicaciones de iOS, Android, macOS y WinUI con .NET MAUI

- FirebaseAuthentication.net  / Firesharp  : Ambos paquetes están relacionados con la autenticación en Firebase para la correcta conexión del proyecto a la base de datos de RealTimeDatabase.
- FirebaseStorage.net  : Este paquete se utiliza para interactuar con el almacenamiento en la nube de Firebase. Permite subir, descargar y gestionar archivos en la nube de manera segura
- Newtonsoft.Json  : Este paquete es una biblioteca de serialización y deserialización de JSON (JavaScript Object Notation) que se utiliza para trabajar con datos en formato JSON. En mi caso lo utilicé para la conexión a la API.
- Plugin.Maui.Calendar  : Este paquete es un plugin para .NET MAUI que proporciona funcionalidades para trabajar con calendarios permite mostrar y manipular información mediante eventos.

7.2. .NetMaui

Es un marco multiplataforma para el desarrollo de aplicaciones móviles y de escritorio que ha sido utilizada para crear la interfaz de usuario y la lógica del proyecto. Además, proporciona una estructura flexible y con facilidad en el manejo de datos



7.3. Firebase

Se utilizó esta herramienta para el almacenamiento de los datos de la aplicación. Es una nube que proporciona una amplia gama de herramientas para manejar datos y la autenticación de usuarios, lo que facilita la gestión de la información y la seguridad de la aplicación.



7.3.1. RealTime Database

Es una base de datos NoSQL alojada en la nube que permite almacenar y sincronizar datos entre la aplicación y el usuario. Esta tecnología ofrece varias ventajas clave por ejemplo la sincronización en tiempo real, el acceso sin conexión y la seguridad basada en usuarios.



7.3.2. Firebase Cloud Storage

Es una herramienta propia de Firebase que permite almacenar y acceder a archivos como imágenes, audio, video y otros tipos de contenido multimedia. Utiliza Google Cloud Storage para proporcionar un servicio de almacenamiento de objetos escalable y confiable



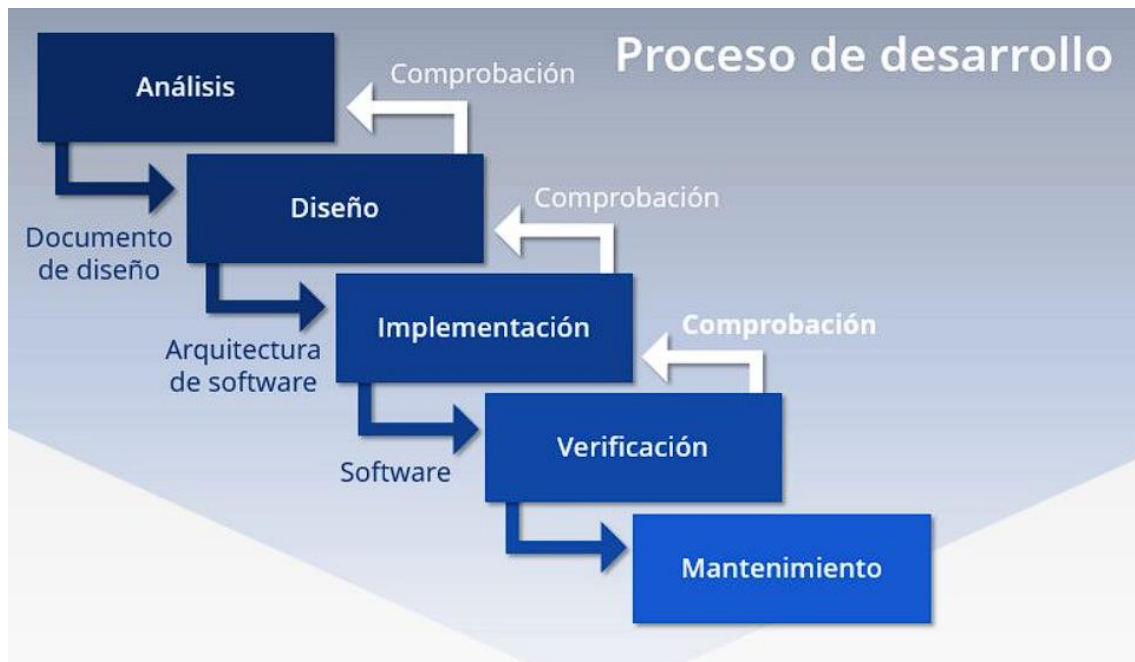
7.4 API Open Weather

Para la funcionalidad de implementar una API, se utilizó OpenWeather, que alberga una enorme cantidad de datos como el tiempo histórico en una ciudad, la radiación solar, cantidad de lluvia, etc. Pero para ChefManager decidí implementar el tiempo actual que hace en una determinada ciudad. Ej. (Madrid,es). La cual requiere un token de acceso único para llamar a la api.



8. Metodología

La metodología que utilicé para este proyecto es el de “Metodología Cascada” el cual toma un enfoque estructurado y secuencial para el desarrollo de aplicaciones que se basa en el concepto de fases discretas y lineales. Cada fase se completa antes de pasar a la siguiente, y el proceso no se vuelve a visitar a menos que se detecten errores o se requieran cambios:



Por ejemplo, dos ejemplos sobre dónde se implementaron las metodologías son:

	ANÁLISIS
CREACIÓN DISEÑO	<ul style="list-style-type: none">-Identificar el objetivo de la vista (por ejemplo, qué información se va a mostrar o qué acción se va a realizar en la vista)-Definir los campos necesarios (por ejemplo, qué campos se necesitan para mostrar o procesar la información en la vista)-Definir las reglas de negocio (por ejemplo, qué condiciones se deben cumplir para mostrar o procesar la información)
IMPLEMENTAR CRUD	<ul style="list-style-type: none">-Identificar las entidades principales del sistema (por ejemplo, Proveedor, Empleado, Notas, etc.).-Definir las operaciones CRUD necesarias para cada entidad.-Determinar las relaciones entre las entidades.-Identificar los requisitos funcionales y no funcionales del sistema.

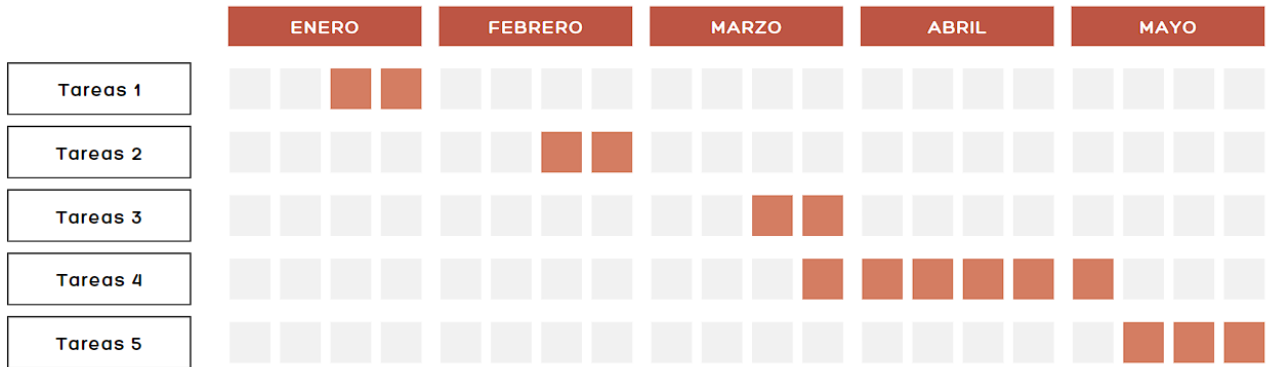
	DISEÑO
CREACIÓN DISEÑO	<ul style="list-style-type: none">-Crear un nuevo proyecto de vista.-Definir la estructura de la vista: Organiza los elementos de la vista en una estructura lógica y visualmente atractiva.-Utilizar controles y componentes según sea necesario para mostrar la información y permitir la interacción con el usuario.
IMPLEMENTAR CRUD	<ul style="list-style-type: none">-Diseñar el modelo de datos para cada entidad, incluyendo sus propiedades y relaciones.-Definir la estructura de la aplicación (capas de presentación, lógica de negocio y acceso a datos).-Diseñar las interfaces de usuario para las operaciones CRUD.-Diseñar los servicios y métodos necesarios para implementar las operaciones CRUD.

	IMPLEMENTACIÓN
CREACIÓN DISEÑO	<ul style="list-style-type: none">-Integrar la lógica y presentación y utilizar el controlador en la vista e implementar el MVVM Utiliza el controlador en la vista para mostrar y procesar la información.
IMPLEMENTAR CRUD	<ul style="list-style-type: none">-Crear las clases de entidad para cada modelo de datos.Implementar la lógica para las operaciones CRUD.Implementar las interfaces de usuario para las operaciones CRUD.

	VERIFICACIÓN
CREACIÓN DISEÑO	<ul style="list-style-type: none">Realizar pruebas unitarias para asegurarse que el diseño funcione correctamente.Realizar pruebas de integración para asegurarse de que la vista y la funcionalidad funcionan correctamente juntas.
IMPLEMENTAR CRUD	<ul style="list-style-type: none">Realizar pruebas unitarias para cada método CRUD implementado.

En mantenimiento, simplemente comprobar que a medida que se van agregando funcionalidades en otros sectores, sigue funcionando correctamente.

8.1. Diagrama de Gantt



Tareas1 (1 y 2 semana: 22-31 Enero)

- Establecer el concepto de gestión de restaurante
- Buscar API e implementarla en el código
- Crear logo y establecer colores de las vistas
- Creación Vista Bienvenida

Tareas2 (3 y 4 semana: 20-30 Febrero):

- Creación de Vista Registro Usuario, Registro Restaurante
- Creación Vista Administrador

Tareas3 (5 y 6 semana: 21-28 Marzo):

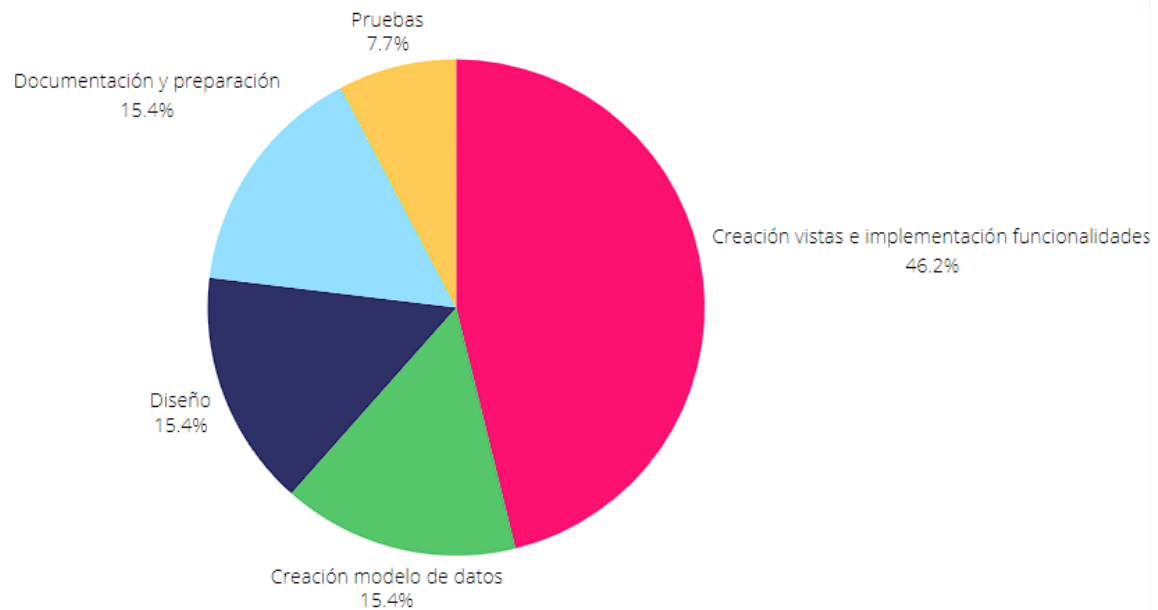
- Reestructuración Vista Administrador, Mejora navegación en el registro
- Conexión a Firebase

Tareas4 (6,7,8,9,10,11 semana: 30 Marzo -17 Mayo):

- Creación de Vista Principal, Empleados, Proveedores, Notas
- Implementar CRUD a cada Vista

Tareas5 (12,13,14 semana: 18-31 Mayo):

- Finalizar memoria
- Mejorar interfaz



Total: 52h

CREACIÓN INTERFAZ E IMPLEMENTACIÓN CRUDS EN TODAS VISTAS (25H)

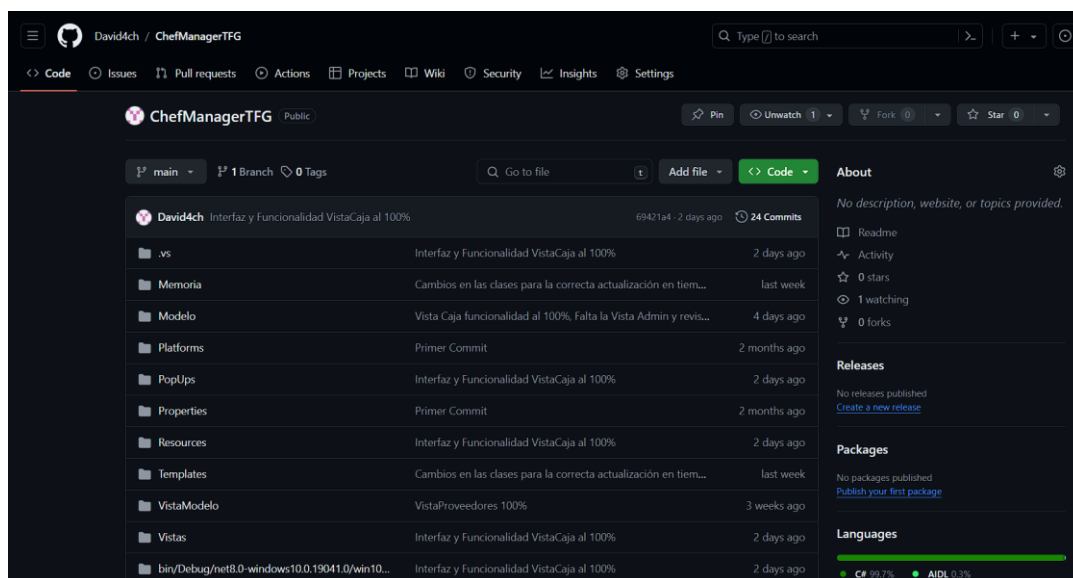
PRUEBAS DE CODIGO Y RESOLUCION DE PROBLEMAS (3,4H)

DOCUMENTACION / PREPARACION PROYECTO PRE INCIO (7,2H)

CREACIÓN MODELO DE DATOS (TANTO BBDD COMO EN CODIGO) (8,2H)

TRATAR EL DISEÑO DE LAS VISTAS CON SU ADECUADA PALETA DE COLORES (8,2 H)

8.2. Git



obj	Interfaz y Funcionalidad VistaCaja al 100%	2 days ago
App.xaml	Reestructuracion2 login/register. Crud implementado, falta f...	2 weeks ago
App.xaml.cs	Primer Commit	2 months ago
AppShell.xaml	Cambios en las clases para la correcta actualización en tiem...	last week
AppShell.xaml.cs	Cambios en las clases para la correcta actualización en tiem...	last week
ChefManager.csproj	Interfaz y Funcionalidad VistaCaja al 100%	2 days ago
ChefManager.csproj.user	Interfaz y Funcionalidad VistaCaja al 100%	2 days ago
ChefManager.sln	Primer Commit	2 months ago
MauiProgram.cs	Vista Inventario al 80% falta botón editar y mejorar interfaz ...	2 weeks ago
README.md	Update README.md	2 weeks ago

 README

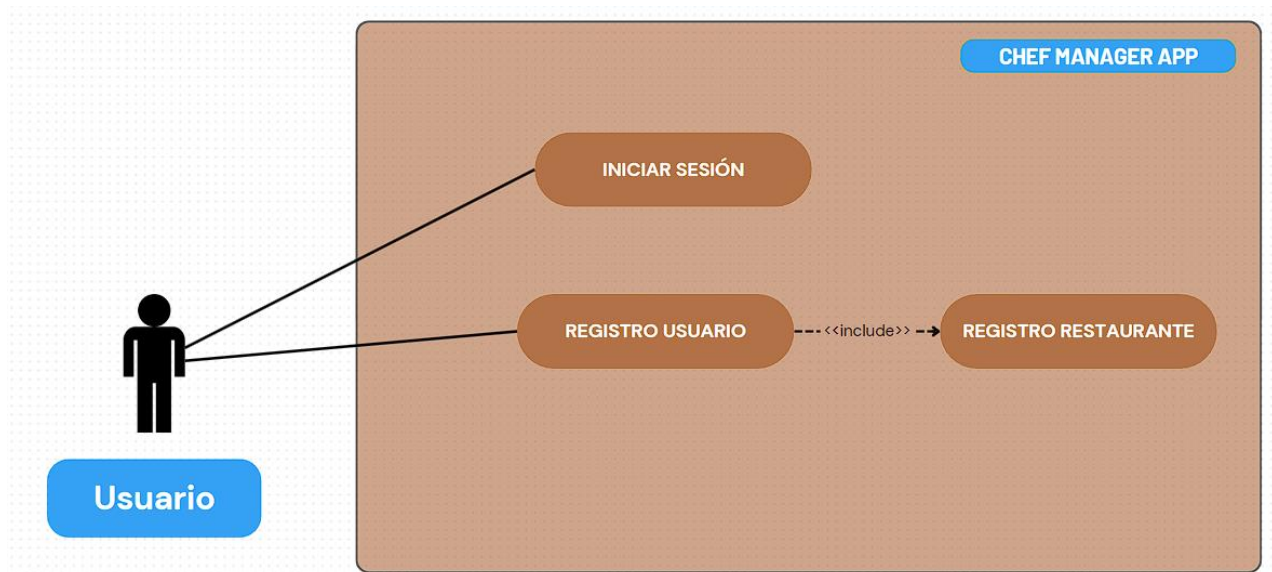
TFG: Aplicación Móvil y Web para la gestión de un bar o restaurante

Este repositorio contiene el Trabajo de Fin de Grado (TFG) desarrollado por David Chavarria Paccha, para IMF CAPITOL.

Descripción

La aplicación desarrollada como parte de este TFG tiene como objetivo ayudar a tener un mejor orden de lo que sucede en tu local, tanto los ingresos y pérdidas como la gestión de empleados y proveedores. Se ha diseñado y construido utilizando .NETMaui, Firebase y una API de CloudWeather

9. Casos de uso



Iniciar sesión y registrarse

Condiciones: Usuario pueda iniciar sesión y registrarse correctamente

Tablas:

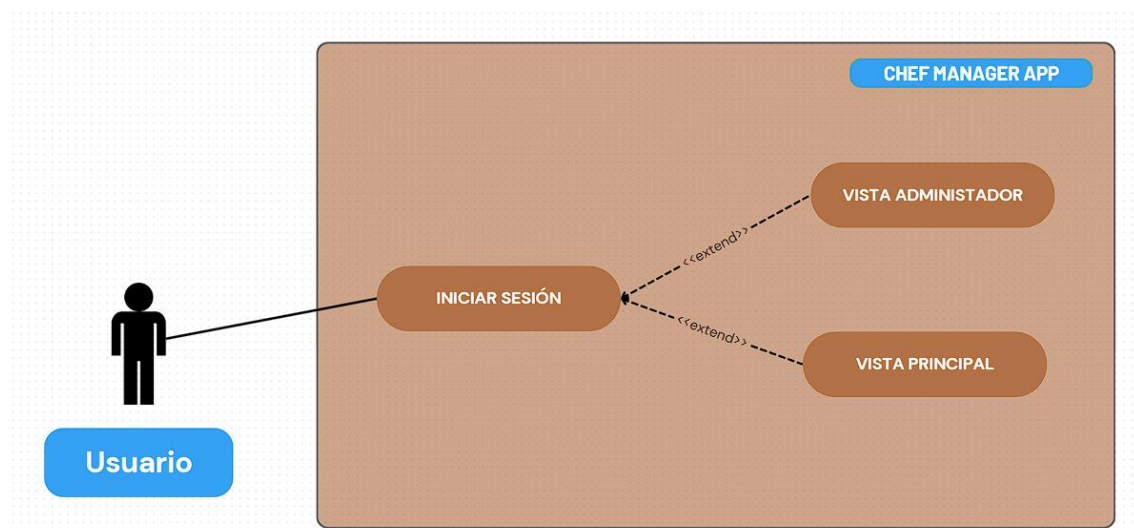
- Usuario
- Restaurante

Clases:

- Usuario
- Encriptacion
- Restaurante
- FirebaseConnection

Interfaces:

- VistaLogin.xaml
- RegistroUser.xaml
- RegistroRestaurante.xaml



Ir a VistaAdministrador y VistaPrincipal

Condiciones: Ir a determinada interfaz dependiendo del user introducido

Tablas:

-Usuario

Clases:

-Usuario

-FirebaseConnection

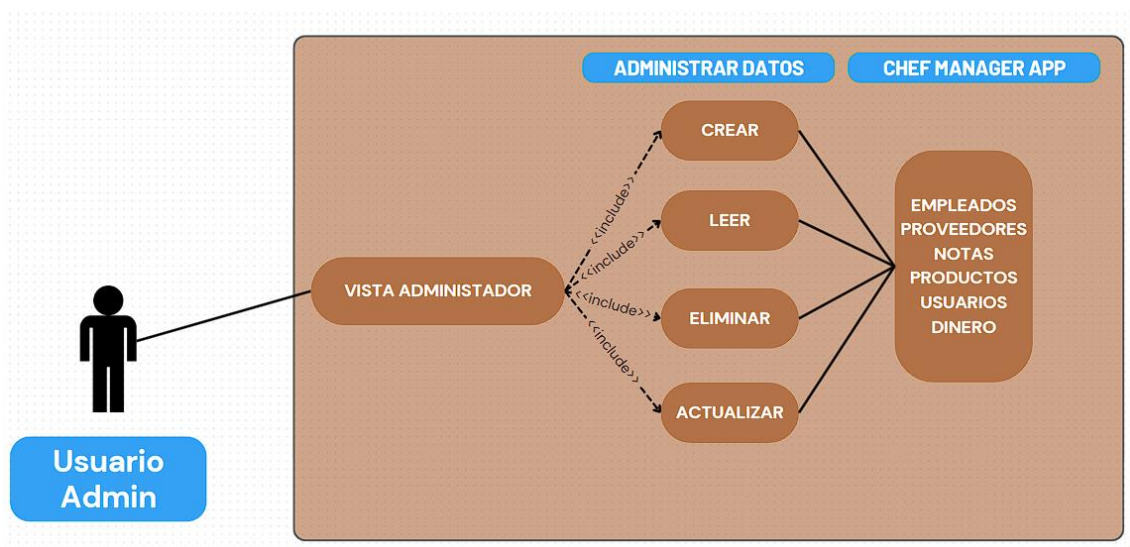
-Encriptacion

Interfaces:

-VistaLogin.xaml

-VistaAdmin.xaml

-VistaPrinc.xaml



Administrar App

Condiciones: Que el administrador pueda borrar, editar, leer y crear elements

Tablas:

-Usuario

-Restaurante

-Empleado

-Dinero

-Nota

-Proveedores

Clases:

-Usuario

-Restaurante

-Empleado

-Dinero

-Nota

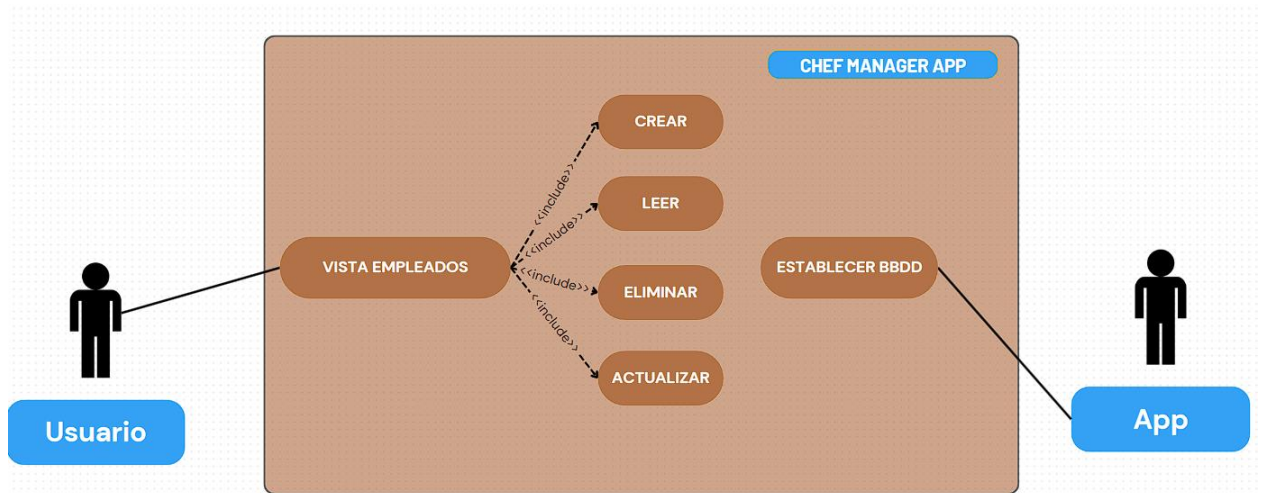
-Proveedores

-FirebaseConnection

-Encriptacion

Interfaces:

-VistaAdmin.xaml



Administrar Empleados

Condiciones: CRUD en interfaz empleados

Tablas:

-Empleado

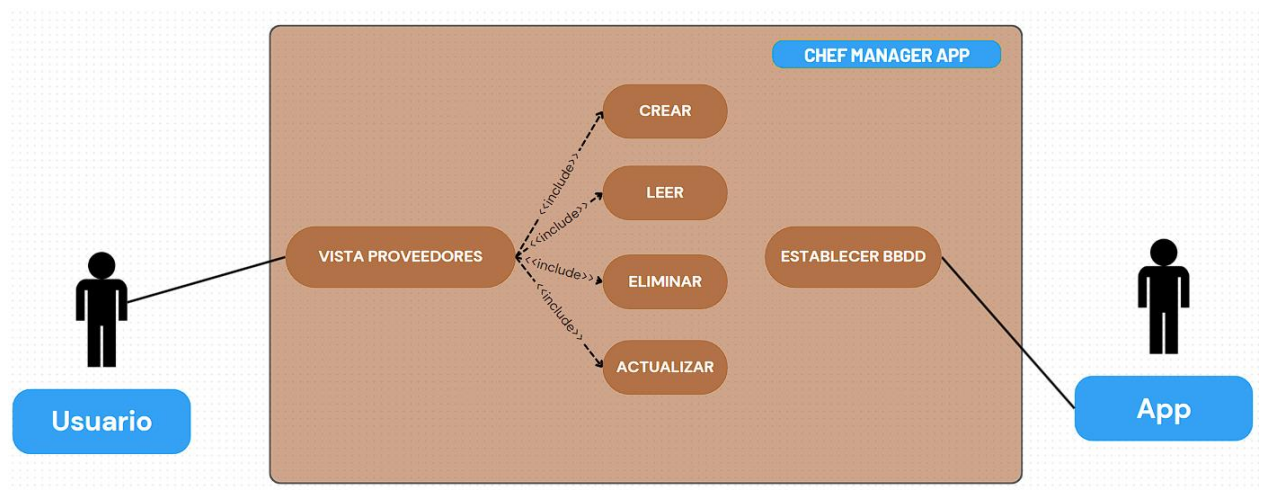
Clases:

-FirebaseConnection

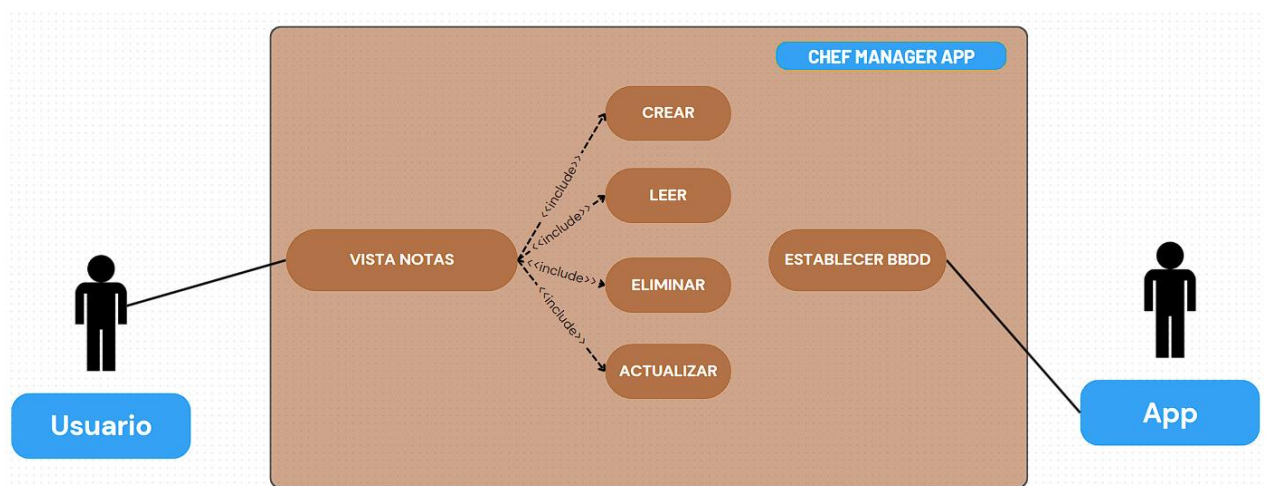
-Empleado

Interfaces:

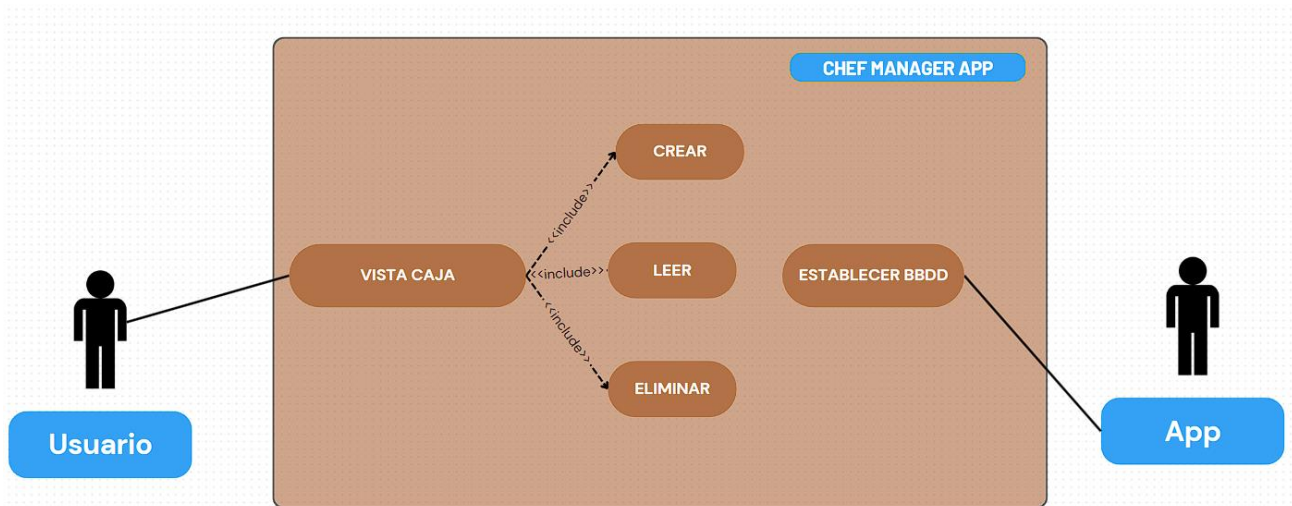
-Empleados.xaml



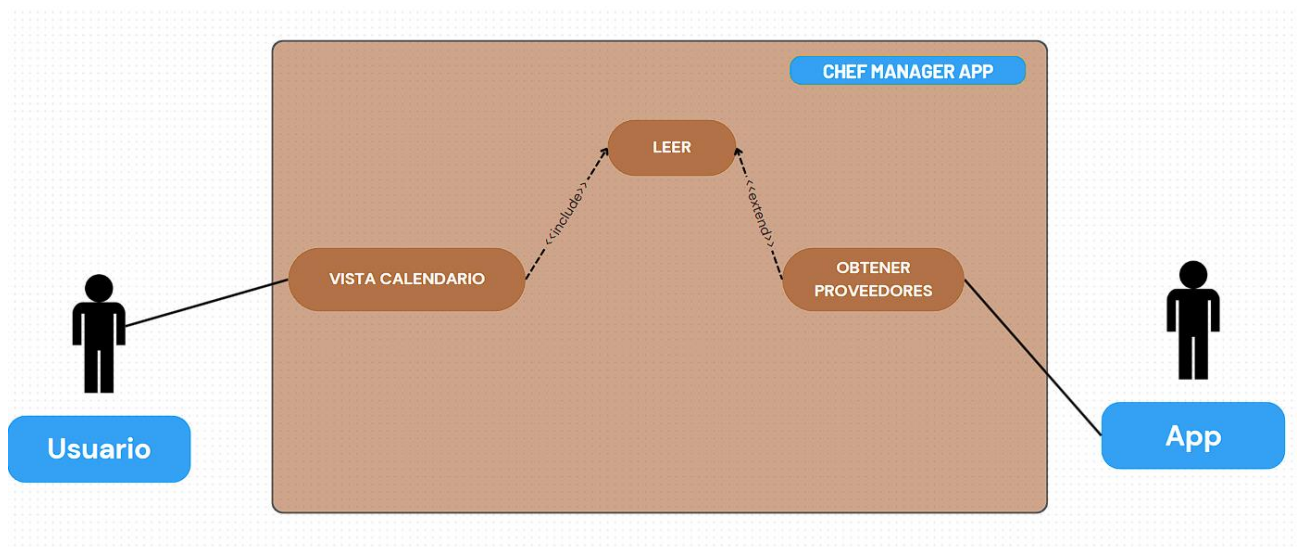
Administrar Proveedores	
Condiciones: CRUD en interfaz proveedores	
Tablas: -Proveedor	Clases: -FirebaseConnection -Proveedor
Interfaces: -Proveedores.xaml	



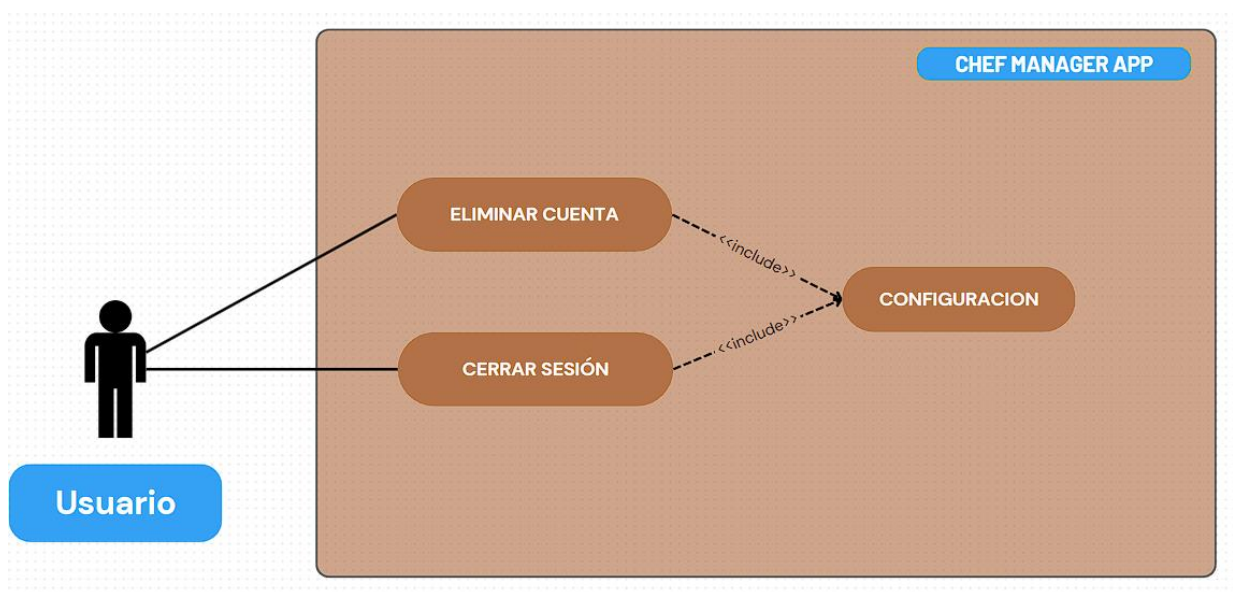
Administrar Notas	
Condiciones: CRUD en interfaz notas	
Tablas: -Nota	Clases: -FirebaseConnection -Nota
Interfaces: -Notas.xaml -VerNota.xaml -EditarNota.xaml	



Administrar Caja	
Condiciones: CRUD en interfaz caja	
Tablas: -Dinero	Clases: -FirebaseConnection -Dinero
Interfaces: -Caja.xaml	



Administrar Proveedores mediante interfaz Calendario	
Condiciones: Consultar Proveedores	
Tablas: -Proveedor	Clases: -FirebaseConnection -Proveedor
Interfaces: -Calendario.xaml	



Administrar Cuenta	
Condiciones: Eliminar y cerrar sesión de la cuenta vigente	
Tablas: -Usuario	Clases: -FirebaseConnection -Usuario
Interfaces: -VistaPrinc.xaml	

10. Bibliografía

Información acerca de la subida de imágenes al proyecto con Firebase:

<https://firebase.google.com/docs/storage/web/upload-files?hl=es-419>

Como implementar el CRUD en C#:

<https://github.com/yurkinh/Plugin.Maui.Calendar>

[Microsoft Documentation]:

<https://learn.microsoft.com/es-es/dotnet/maui/?view=net-maui-8.0>

Noticia1:

<https://www.eleconomista.es/retail-consumo/noticias/12590895/12/23/espana-suma-mas-restaurantes-que-nunca-abren-3500-en-el-ultimo-ano.html>

Noticia2:

<https://www.eleconomista.es/retail-consumo/noticias/12610792/01/24/la-hosteleria-creara-mas-de-50000-empleos-y-batira-otro-record-en-2024.html>

Información sobre el paquete nugget de Plugin.Maui.Calendar:

<https://github.com/yurkinh/Plugin.Maui.Calendar>