

MEMORIA DE PROYECTO DE DESARROLLO DE APLICACIONES WEB (DAW)

CURSO 2024 - 2025



Autor: Teresa Fernández Martínez

Fecha: 04 de diciembre de 2024

Curso académico: 2º Desarrollo de Aplicaciones

Web

ÍNDICE

1. INTRODUCCIÓN.....	1
2. ORIGEN Y CONTEXTUALIZACIÓN DEL PROYECTO.....	1
2.1 Descripción de la necesidad a la que da respuesta el producto/servicio.....	1
2.2 Planteamiento del problema.....	2
2.3 Costes de los Recursos Utilizados.....	3
2.4 Plan de financiación.....	4
2.5 Plan de recursos humanos.....	5
2.6 Plan de prevención de riesgos.....	5
2.6.1 Riesgos identificados:.....	6
2.6.2 Soluciones sugeridas:.....	6
3. ESPECIFICACIÓN DE REQUISITOS.....	7
3.1 Requisitos fundamentales.....	8
3.2 Requisitos tecnicos.....	8
3.3 Requisitos de interfaz externa.....	9
3.4 Obligación de diseño.....	9
4. OBJETIVOS ESPECÍFICOS.....	10
4.1 Diseño de la Interfaz Gráfica.....	10
4.1.1. Creación de las Páginas Iniciales:.....	10
4.1.2 Programación.....	11
4.2 Interfaz de comunicaciones.....	11
4.2.1 Comunicación cliente-servidor.....	12
4.2.2 Protocolo HTTPS.....	12
4.2.3 Comunicación asincrónica.....	12
4.2.5 Integración de Firebase.....	13
4.2.6 API de LeafLet.....	13
4.2.7 Seguridad de las comunicaciones externas.....	13
5. TAREAS.....	13
5.1 Tarea 1: ANÁLISIS.....	13
5.1.1 Subtarea 1.1: Necesidad en el mercado.....	14
5.2 Tarea 2: Documentación.....	14
5.2.1 Subtarea 2.1: FireBase firestore.....	14
5.2.2 Subtarea 2.2: Atom.....	14
5.3 Tarea 3: Creación del Servidor en Firebase Firestore.....	14
5.4 Tarea 4: Diseño de diagramas.....	15
5.4.1 Subtarea 1: Diagrama de clases.....	15
5.4.2 Subtarea 2: Diagrama de usos.....	17
6. DISEÑO DE LA PLATAFORMA.....	19
6.1 Estructura de la página.....	19
6.2 Código de programación.....	28
6.3 Datos Almacenados en Firebase Firestore.....	29
7. IMPLEMENTACIÓN.....	32
7.1 Tecnologías Utilizadas en el Desarrollo del Proyecto.....	32

7.2 Descripción del Proyecto.....	33
7.2.1 Capa de Presentación.....	33
8. ANÁLISIS DAFO.....	52
9. BIBLIOGRAFÍA.....	56

1. INTRODUCCIÓN

ResFav, que es la abreviatura de “Restaurante Favorito”, es una página web desarrollada para restablecer la comunicación entre los usuarios y los restaurantes situados cerca de su ubicación. La plataforma ResFav existe para garantizar que la experiencia sea accesible e individual. Por lo tanto, los usuarios están capacitados para buscar los restaurantes localmente, encontrar la información más significativa y compartir críticas, en caso de que se registren en la plataforma. Aunque originalmente el proyecto fue planteado como una aplicación móvil, el análisis del mercado y las ventajas del acceso a la plataforma sugirieron a los desarrolladores considerar ResFav como una página web. Como resultado, ResFav puede conectarse desde cualquier lugar del mundo.

2. ORIGEN Y CONTEXTUALIZACIÓN DEL PROYECTO

2.1 Descripción de la necesidad a la que da respuesta el producto/servicio

La idea de crear ResFav surge de percibir una deficiencia en el mercado: los servicios actuales carecen de sistemas individualizados que ayuden a los restaurantes a precisar su ubicación y a la vez proporcionen información completa en un solo lugar. Aunque existen ciertos recursos como mapas de navegación o interfaces de búsqueda, les faltan capacidades como la clasificación sofisticada basada en la ubicación, las evaluaciones de los clientes o la función para compartir opiniones posteriores.

Para satisfacer esta necesidad, proporcionaré un remedio conveniente y sin

complicaciones que ayude a las personas a descubrir restaurantes cercanos o dirigirse a un lugar en particular A través de la plataforma, los usuarios pueden:

- Consultar información detallada de cada restaurante, como su número de contacto, puntuaciones medias, y un mapa interactivo con la ubicación exacta.
- Filtrar restaurantes según localidad y zona de interés.
- Dejar valoraciones y comentarios, siempre que se hayan registrado previamente.

Esta página web permite no solo obtener información útil, sino también fomentar la comunicación y el intercambio de conocimientos entre los clientes, promoviendo elecciones inteligentes en la selección de restaurantes.

2.2 Planteamiento del problema

La gente hoy en día quiere una forma rápida de encontrar un lugar para comer y hablar con los dueños con claridad. Aunque las plataformas de mapas e instrucciones generales no están adaptadas a las demandas individuales de los usuarios, carecen de funcionalidades para búsquedas de ubicaciones exactas, opciones de filtro personalizadas o plataformas de reseñas.

Esta personalización insuficiente en las estrategias actuales restringe la capacidad de los consumidores para tomar decisiones rápidas y con conocimiento de causa, lo que genera irritación y una separación de los establecimientos gastronómicos del vecindario.

ResFav aborda este problema con un enfoque innovador, permitiendo:

- Una localización rápida y precisa de lugares para comer según un área determinada.

- Obtener detalles completos como calificaciones, opiniones y datos de contacto.
- La interacción activa de los usuarios registrados a través de opiniones y calificaciones.

Con estas características, ResFav no solo mejora el proceso de búsqueda, sino que también estimula la participación activa del usuario, estableciendo una red comunitaria de recomendaciones que beneficia tanto a los clientes como a los establecimientos.

2.3 Costes de los Recursos Utilizados

En esta sección se presentan los elementos y costos involucrados en la ejecución del proyecto ResFav:

RECURSOS	COSTES
HP Laptop	1000€
Atom (Editor de Código)	0€
XAMPP (Servidor Local)	0€
Firebase (Base de Datos)	0€
HTML5	0€
JavaScript	0€
CSS	0€
Firebase Authentication	0€

Google Maps API	0€
-----------------	----

Descripción de los recursos utilizados:

- **HP Laptop:** Ordenador portátil de la marca HP, utilizado para el desarrollo y pruebas del proyecto.
- **Atom:** Editor de código de texto de código abierto, utilizado para escribir y editar el código fuente del proyecto.
- **XAMPP:** Software que proporciona un entorno de servidor local, facilitando la creación y pruebas de la página web.
- **Firebase:** Plataforma de Google utilizada para gestionar la base de datos, la autenticación de usuarios y el almacenamiento de información relacionada con restaurantes y valoraciones.
- **HTML5, CSS, JavaScript:** Tecnologías de código abierto utilizadas para la creación y diseño de la página web, asegurando su funcionalidad e interactividad.
- **Firebase Authentication:** Servicio de Firebase para la autenticación de usuarios, permitiendo iniciar sesión a través de métodos como Google.
- **Google Maps API:** Integración con Google Maps para mostrar la ubicación precisa de los restaurantes seleccionados.

2.4 Plan de financiación

El desarrollo y apoyo de ResFav será financiado por una variedad de fuentes, incluidas donaciones internas, inversores interesados en la industria gastronómica y acuerdos estratégicos con restaurantes. La financiación cubrirá los costos asociados con el desarrollo de la plataforma, desde el diseño y la programación hasta la implementación y las pruebas.

El presupuesto también incluye el costo de la infraestructura técnica, como servidores y bases de datos, así como los recursos necesarios para garantizar el funcionamiento de la plataforma. Desarrollo sostenible del sitio web, incluyendo campañas de marketing digital y posicionamiento en buscadores.

Las contribuciones de los restaurantes pueden incluir ofertas promocionales en la plataforma, lo que les permitirá destacarse de otros competidores y ganar mayor visibilidad entre los usuarios. A cambio, reciben una mayor exposición y la oportunidad de interactuar directamente con los clientes a través de calificaciones y reseñas.

Este enfoque también tiene como objetivo construir relaciones a largo plazo con restaurantes y clientes, asegurando que todos los involucrados,La plataforma beneficia tanto a empresas como a consumidores.

2.5 Plan de recursos humanos

El desarrollo y gestión de **ResFav** requiere un equipo de trabajo multidisciplinario con experiencia en varias áreas clave. A continuación, se detallan los roles esenciales y sus responsabilidades para llevar a cabo el proyecto:

PUESTO	NOMBRE	DNI
Técnico Desarrollo Aplicaciones Web	Teresa Fernández Martínez	71789560-M

2.6 Plan de prevención de riesgos

Aunque desarrollar y mantener una plataforma web como ResFav no implica una actividad física extenuante, es importante considerar los riesgos inherentes

a esta actividad. Algunos de los riesgos y precauciones identificados se describen a continuación:

2.6.1 Riesgos identificados:

- **Fatiga visual o muscular:** La exposición prolongada a pantallas y la realización de tareas repetitivas puede causar fatiga visual y muscular que afecta la salud de las personas, la productividad y la felicidad.
- **Problemas ergonómicos:** un entorno de trabajo mal diseñado (como sentarse frente a una computadora) puede provocar dolores musculares o problemas de salud a largo plazo.
- **Riesgos de equipos electrónicos:** el uso de equipos electrónicos como computadoras y servidores conlleva el riesgo de accidentes debido al contacto con cables defectuosos o equipos con mal mantenimiento.
- **Carga mental:** las tareas cognitivamente complejas, los plazos ajustados o los entornos de trabajo estresantes pueden afectar la salud mental y la productividad de su equipo.
- **Factores organizacionales:** los cambios en la estructura del equipo o la falta de comunicación interna pueden generar incertidumbre o estrés. Esto afecta la eficiencia del desarrollo y la gestión de la plataforma.

2.6.2 Soluciones sugeridas:

- **Fatiga visual o muscular:** tomar descansos regulares para descansar la vista y estirarse. Asegúrese de que las estaciones de trabajo estén espaciadas adecuadamente y sean ergonómicamente sólidas. , ajustar sillas, mesas y mamparas para mejorar la postura.
- **Cuestiones ergonómicas:** proporciona información y recursos sobre prácticas laborales saludables, como ajustar la altura de la pantalla o mantener una postura cómoda al sentarse.

- **Riesgos asociados con los equipos electrónicos:** implementar inspecciones periódicas de los equipos y cables para garantizar que estén en buenas condiciones. Capacite a su equipo en seguridad eléctrica y garantice el uso de equipos de protección adecuados, como alargadores y cables de alta calidad.
- **Carga mental:** distribuya las tareas de manera justa, establezca plazos realistas y evite sobrecargar a los miembros del equipo. Proporciona un programa de apoyo para el manejo del estrés, como talleres o recursos de bienestar en el lugar de trabajo.
- **Factores organizativos:** mejorar las comunicaciones internas con reuniones periódicas y herramientas de gestión de proyectos para mantener a todos encaminados. Cree canales de retroalimentación donde los miembros del equipo puedan expresar inquietudes y sugerencias, promoviendo la colaboración y la transparencia.

Al abordar proactivamente estos riesgos y promover prácticas laborales seguras y saludables, ayudará al equipo de ResFav a mantener un ambiente de trabajo óptimo, lo que resultará en un desarrollo más eficiente y una mejora continua de la plataforma.

3. ESPECIFICACIÓN DE REQUISITOS

La especificación de requisitos proporciona la base para el diseño, desarrollo e implementación de ResFav. Esta sección detalla los requisitos funcionales, técnicos y de interfaz necesarios para que la Plataforma logre su propósito de conectar a los usuarios con restaurantes cercanos y brindar una experiencia perfecta para los usuarios y los restaurantes.

3.1 Requisitos fundamentales

Función ResFav. Los requisitos son lo que el sistema debe hacer para cumplir con las expectativas del usuario. Los requisitos funcionales clave incluyen:

- **Registro e inicio de sesión:** El sistema debe permitir a los usuarios registrarse y autenticarse utilizando varios métodos, como a través de una cuenta de Google o el registro manual utilizando una dirección de correo electrónico y una contraseña.
- **Buscar restaurantes:** los usuarios deberían poder buscar restaurantes por ubicación y poder filtrar por ubicación, tipo de comida o clasificación.
- **Calificaciones y reseñas:** los usuarios deberían poder dejar calificaciones y reseñas sobre los restaurantes visitados. Posibilidad de ver comentarios anteriores de otros usuarios.
- **Ver información del restaurante:** al seleccionar un restaurante, los usuarios deberían poder acceder a sus detalles, como número de teléfono, dirección, menú, calificaciones y un mapa interactivo de la ubicación.

3.2 Requisitos técnicos

Los requisitos técnicos se refieren a la arquitectura y la tecnología utilizadas en el desarrollo y operación del sitio web. Estos incluyen:

- **Base de datos:** el sistema utilizará Firebase como plataforma de base de datos para administrar la información de los restaurantes y los clientes. Firebase proporciona soluciones de almacenamiento de datos en tiempo real escalables y seguras.
- **Plataforma de desarrollo:** la aplicación utilizará HTML5, CSS y JavaScript para desarrollar la interfaz, y un backend, que será gestionado por

Firebase, para gestionar solicitudes y conexiones de bases de datos.

- **Seguridad:** la seguridad en la plataforma debe ser una prioridad y Firebase Authentication proporciona un sistema de autenticación sólido. Aparte de, se tomarán medidas de seguridad para proteger los datos de los usuarios.

3.3 Requisitos de interfaz externa

Esta sección detalla las interfaces necesarias para que ResFav interactúe con otros sistemas o servicios:

- **API de Google Maps:** ResFav utilizará Google Maps API para mostrar ubicaciones de restaurantes y permitir a los usuarios ver su dirección exacta en un mapa interactivo.
- **API de autenticación de Firebase:** se integrará con el sistema de autenticación de Firebase para permitir a los usuarios registrarse e iniciar sesión utilizando su cuenta de Google o su correo electrónico.
- **API de almacenamiento de información de Firestore por Firebase:** se integrará con el sistema de almacenamiento de información de los restaurantes.

3.4 Obligación de diseño

El diseño de ResFav debe cumplir con ciertas pautas para garantizar la experiencia del usuario. Eficiencia:

- **Responsividad:** el diseño debe ser totalmente responsive y adaptable a cualquier tipo de dispositivo, ya sea una computadora de escritorio, Tablet o teléfono.
- **Accesibilidad:** las plataformas deben ser accesibles para personas con

discapacidades, garantizando que las imágenes, los botones y el texto sean adecuados para usuarios con diferentes necesidades.

- **Interfaz de usuario:** la interfaz debe ser simple e intuitiva, con una navegación clara, que permita a los usuarios encontrar rápidamente los restaurantes y la información que desean.

4. OBJETIVOS ESPECÍFICOS

Los objetivos específicos de ResFav son fundamentales para el desarrollo exitoso de la plataforma web, incluido el diseño y la programación de funciones clave que brindarán una experiencia satisfactoria para los usuarios y los restaurantes. Estos objetivos incluyen:

4.1 Diseño de la Interfaz Gráfica

4.1.1. Creación de las Páginas Iniciales:

Al abrir la página web por primera vez, los usuarios encontrarán una página inicial con información sobre el proyecto ResFav, su misión, y cómo funciona la plataforma.

- **Página Principal:**

- Mostrar una lista de todos los restaurantes disponibles en la zona seleccionada, inicialmente centrada en Asturias.
- Los usuarios podrán registrarse o iniciar sesión para acceder a funciones personalizadas.
- Un menú de navegación permitirá acceder fácilmente a distintas pantallas y funcionalidades dentro de la plataforma.
- Funcionalidad de filtrado para que los usuarios puedan seleccionar la localidad y la ubicación de su interés, mejorando la experiencia de

búsqueda de restaurantes.

- **Página de Inicio de Sesión:**

- Los usuarios podrán iniciar sesión si ya están registrados en la plataforma.
- Se permitirá el inicio de sesión mediante un correo electrónico y contraseña que se almacena correctamente en el módulo de autenticación de Firebase.
- Opción de registro para nuevos usuarios.

- **Página de Registro:**

- Los usuarios podrán crear una cuenta introduciendo datos como correo electrónico y contraseña, y la localidad a la que pertenecen.

- **Página con la Información del Restaurante Seleccionado:**

- Al seleccionar un restaurante, los usuarios accederán a una página con detalles como su nombre, dirección, teléfono, valoraciones, la opción de poder dejar un comentario, puntuar con estrellas los restaurantes, y poder ver los comentarios de antiguos clientes.

4.1.2 Programación

- **Configurar el sistema de autenticación (Firebase Auth):**

- Integre el sistema de autenticación de Firebase para garantizar un registro e inicio de sesión seguros.

- **Crear una base de datos en Firebase Firestore:**

- Diseñe y cree una base de datos en Firebase Firestore para almacenar información de restaurantes y datos de clientes.
- **Desarrollo de plataforma web:** utiliza tecnologías web modernas (HTML5, CSS3, JavaScript) para implementar la plataforma, conectándose directamente a Firebase para administrar las funciones de registro e inicio de sesión.

- Pantalla de inicio (como lista de restaurantes, página de detalles del restaurante y función de calificación).

4.2 Interfaz de comunicaciones

Se necesita una interfaz de comunicación para garantizar una interacción fluida y eficiente entre los diversos componentes del sistema ResFav, así como para garantizar una experiencia de usuario segura, rápida y confiable.

4.2.1 Comunicación cliente-servidor

ResFav adopta una arquitectura cliente-servidor, donde:

- El cliente (navegador del usuario) envía una solicitud al servidor para realizar operaciones como buscar restaurantes, registrar nuevos usuarios o realizar comentarios.
- El servidor maneja estas solicitudes, consulta o actualiza la base de datos en Firebase y devuelve la información solicitada al cliente en tiempo real.

4.2.2 Protocolo HTTPS

Para garantizar la seguridad de los datos transmitidos, todas las comunicaciones del cliente y del servidor con Firebase se implementan mediante el protocolo HTTPS. Esto garantiza lo siguiente:

- Los datos se cifran durante la transmisión.
- Evita ataques como el acceso no autorizado o la manipulación de datos por parte de terceros.

4.2.3 Comunicación asincrónica

Para mejorar la experiencia del usuario y evitar recargar toda la página, ResFav implementa comunicación asincrónica utilizando los siguientes métodos:

- **Base de datos Firestore de Firebase:** actualizaciones en tiempo real de

nuevos comentarios o calificaciones y otros datos relacionados. Como el registro de nuevos usuarios registrados en la plataforma.

4.2.5 Integración de Firebase

ResFav utiliza los servicios de Firebase para:

- **Autenticación:** permite a los usuarios registrarse e iniciar sesión usando credenciales únicas.
- **Base de datos (Firestore):** almacena y recupera información como detalles de restaurantes, reseñas de clientes y registros de cuentas.

4.2.6 API de LeafLet

Para ayudarte a encontrar restaurantes, ResFav se integra con la API de LeafLet. Te permite:

- Mostrar la ubicación exacta de tu restaurante en un mapa interactivo.

4.2.7 Seguridad de las comunicaciones externas

La integración con servicios externos se ajusta a protocolos de seguridad estándar como OAuth, que se utiliza para autenticar y autorizar el acceso Firebase. Esto garantiza:

- Solo los usuarios autorizados pueden acceder a datos confidenciales.
- Utilice servicios externos de forma segura y eficiente.

5. TAREAS

En este apartado se describen las principales tareas realizadas durante el desarrollo de **ResFav**. Estas tareas se agrupan en actividades clave que han sido fundamentales para la correcta ejecución del proyecto.

5.1 Tarea 1: ANÁLISIS

Esta tarea inicial consistió en investigar la competencia existente en el

mercado y evaluar las necesidades que ResFav podía cubrir. Se llevó a cabo un análisis detallado de las aplicaciones ya disponibles, identificando oportunidades de mejora y diferenciación, especialmente enfocándose en la región de Asturias.

5.1.1 Subtarea 1.1: Necesidad en el mercado

- Identificar aplicaciones existentes y su falta de personalización para el contexto específico de Asturias.
- Evaluar la variedad de restaurantes en la región y las necesidades no cubiertas.

5.2 Tarea 2: Documentación

Esta tarea fue esencial para seleccionar las herramientas y tecnologías adecuadas para el desarrollo de ResFav. Se investigaron las funcionalidades de las plataformas utilizadas y se recopilaron manuales y guías para garantizar su correcto uso.

5.2.1 Subtarea 2.1: FireBase firestore

- Explorar la funcionalidad de Firebase Firestore como base de datos en tiempo real.
- Consultar manuales y guías para implementar Firestore de manera eficiente.

5.2.2 Subtarea 2.2: Atom

Verificar que Atom estuviera correctamente configurado con los paquetes necesarios para trabajar con HTML, CSS y JavaScript.

5.3 Tarea 3: Creación del Servidor en Firebase Firestore

Esta tarea se centró en la configuración de la base de datos para almacenar la información sobre restaurantes y usuarios. Incluyó la creación del proyecto en Firebase, la configuración de la base de datos y la conexión con el código fuente desarrollado en Atom.

5.4 Tarea 4: Diseño de diagramas

En esta tarea se desarrollaron los diagramas necesarios para estructurar el proyecto, como los diagramas de clases y de casos de uso.

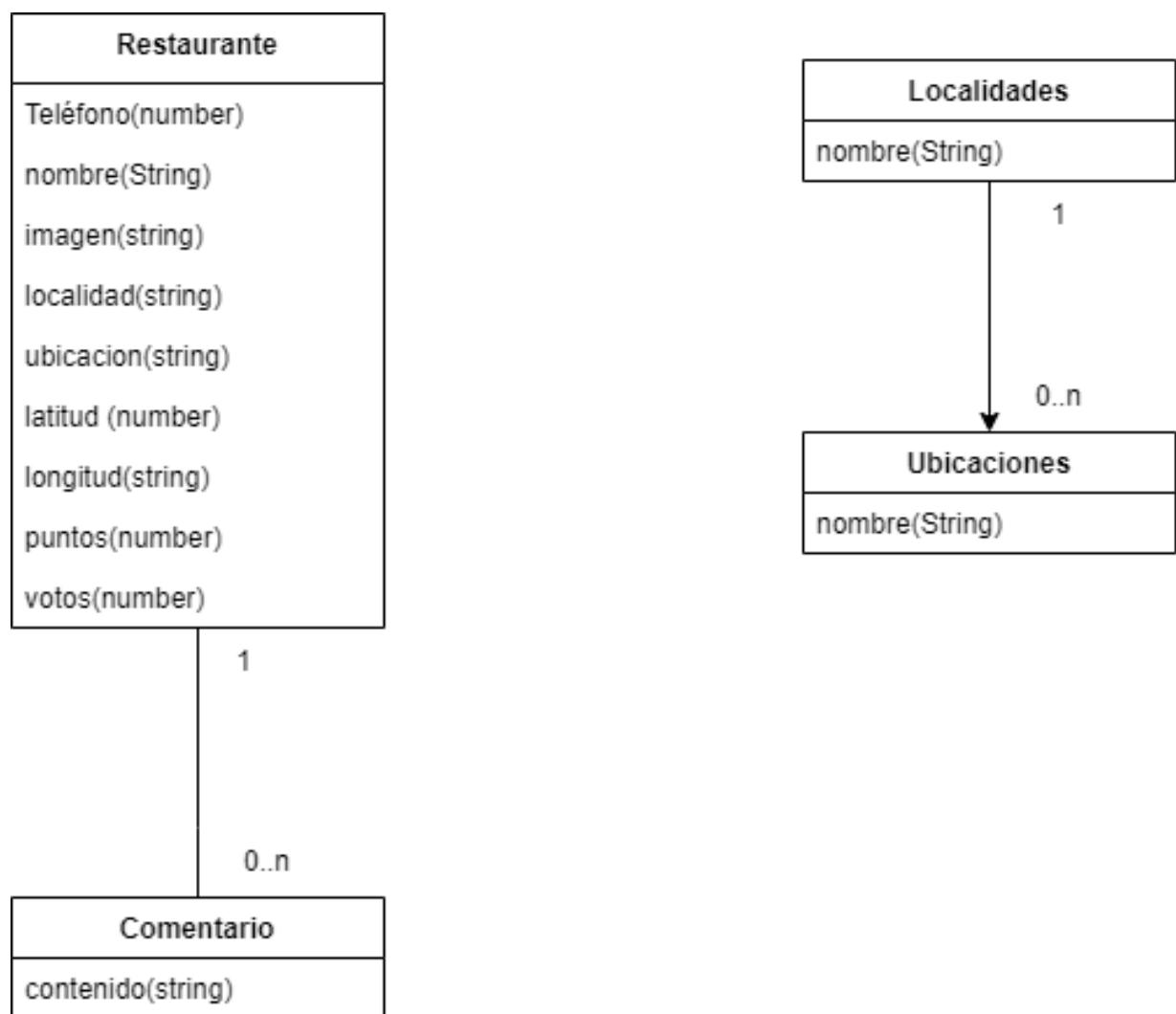
5.4.1 Subtarea 1: Diagrama de clases

La aplicación utiliza una base de datos no SQL, que organiza la información en documentos. La base de datos está formada por colecciones y estas a su vez contienen documentos.

Nuestra base de datos tiene dos colecciones, una para almacenar los restaurantes y otra para almacenar las localidades.

La colección que almacena documentos tipo restaurantes, es en la que almacenaremos la información sobre los restaurantes de nuestra aplicación. Cada restaurante a su vez contiene los comentarios que los usuarios dejaron sobre él.

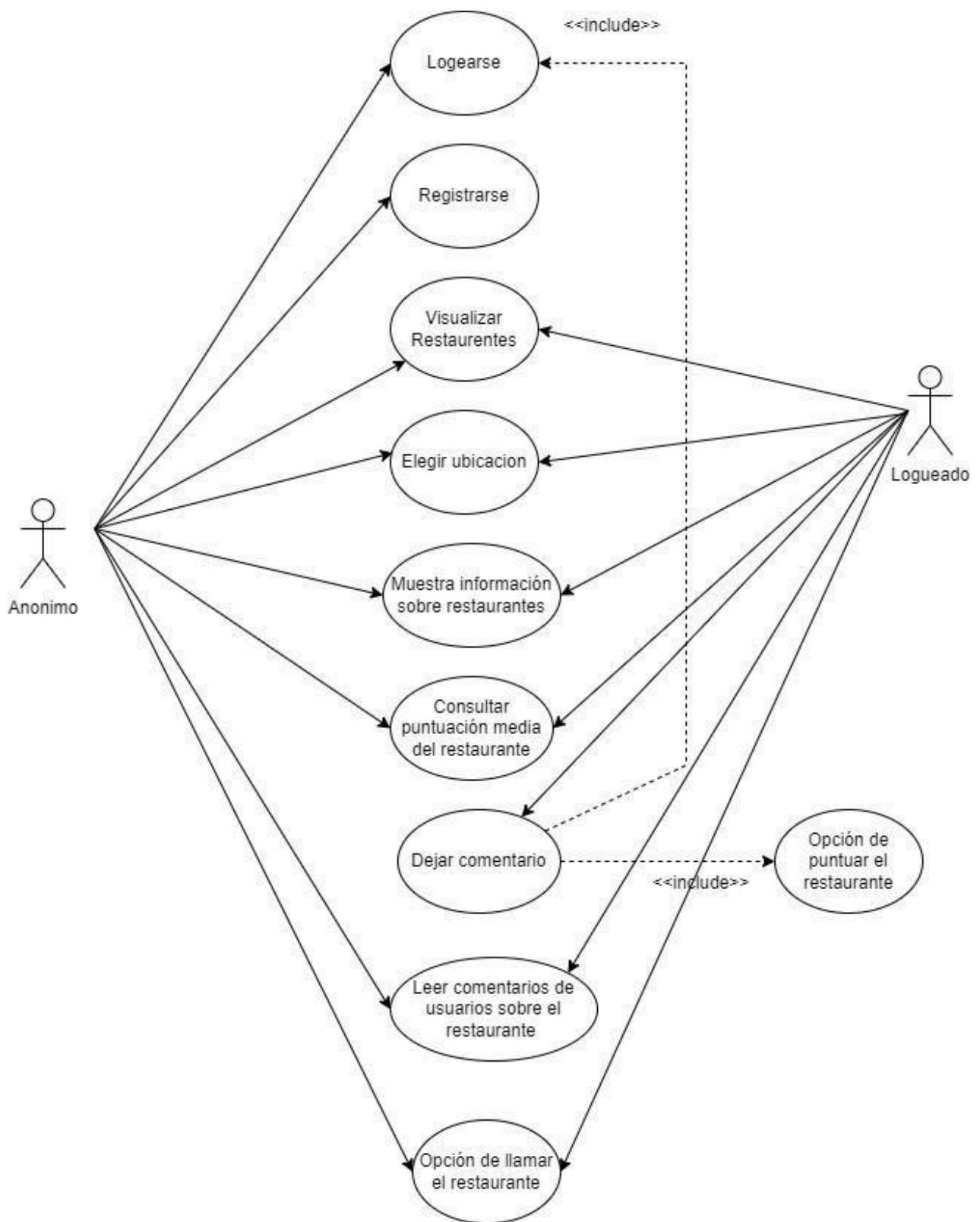
La otra colección que almacena documentos tipo localidad, es donde se almacenan los nombres de localidad en los que se encuentran los restaurantes de nuestra base de datos. Cada localidad incluye los documentos de ubicación que pertenecen a ese local.



5.4.2 Subtarea 2: Diagrama de usos

Este diagrama ilustra las acciones y privilegios disponibles para los diferentes tipos de usuarios en **ResFav**. Se distinguen dos tipos de usuarios: aquellos que están previamente registrados y los que acceden de forma anónima, sin haberse registrado ni iniciado sesión.

- **Usuarios Registrados/Logueados:** los usuarios que han completado el registro e inicio de sesión tienen acceso completo a las funcionalidades de la plataforma. En el diagrama, se puede ver que tienen la capacidad de dejar comentarios sobre los restaurantes y asignar puntuaciones, lo que les permite participar activamente en la comunidad de ResFav.
- **Usuarios Anónimos:** los usuarios que no han iniciado sesión ni se han registrado tienen un acceso limitado. Aunque pueden navegar por la plataforma y visualizar la información de los restaurantes, no podrán dejar comentarios ni puntuaciones. Si un usuario anónimo intenta comentar o puntuar, el sistema lo redirigirá automáticamente a la pantalla de inicio de sesión para completar el registro o iniciar sesión.



6. DISEÑO DE LA PLATAFORMA

Diseño de la plataforma ResFav, está diseñada para proporcionar una experiencia de usuario intuitiva, accesible y agradable. Esta sección detalla las páginas principales de la plataforma, los elementos clave de cada página y sus interrelaciones, así como el código que implementa estas funciones.

6.1 Estructura de la página

El diseño de ResFav se basa en una estructura simple y clara, que permite a los usuarios encontrar fácilmente el contenido que desean y acceder a todas las funciones de la plataforma. Las páginas principales que componen el sitio web se describen a continuación:

- **Página de Registro/ Inicio de sesión:** esta es la primera página que verá el usuario al acceder por primera vez en la plataforma. Aquí, el usuario podrá registrarse, o iniciar sesión con el usuario y contraseña, que anteriormente haya creado.

Ordenador:

The image shows a screenshot of the ResFav platform. At the top, there is a photograph of a modern restaurant interior with wooden walls, plants, and a sign that says "INICIAR SESIÓN". Below the photo are two side-by-side forms. The left form is for "INICIAR SESIÓN" (Login) and includes fields for "Correo electrónico" (Email) and "Contraseña" (Password), followed by a "INICIAR SESIÓN" button. The right form is for "REGISTRARSE" (Register) and includes fields for "Correo electrónico" (Email), "Contraseña" (Password), and "Localidad a la que pertenes" (Locality you belong to), followed by a "REGISTRARSE" button. Both forms feature the ResFav logo at the top.

Tablet



INICIAR SESIÓN

Correo electrónico

Contraseña

INICIAR SESIÓN

REGISTRARSE

Correo electrónico

Contraseña

Localidad a la que pertenesces ▾

REGISTRARSE

Móvil



INICIAR SESIÓN

Correo electrónico

Contraseña

INICIAR SESIÓN

RF

- **Página de Inicio:** en esta página, verá una imagen de restaurante, en la que tienen un botón para que cargue directamente la página de todos los restaurantes. En esta página, al igual que en todas, podrás ver un mapa con todos los puntos donde se encuentran los restaurantes que están en la base de datos.

Ordenador



Tablet

Gracias por visitar nuestro sitio. Estamos comprometidos a brindarle la mejor experiencia gastronómica.

Inicio
Restaurantes
Iniciar Sesión

© 2024 ResFav. Todos los derechos reservados.

Móvil



- **Página de Restaurantes:** muestra una lista de todos los restaurantes disponibles. Tiene un filtrado, donde podrá filtrar por ubicación o puntuación. No obstante, cuando un usuario se registra podrá seleccionar a qué Localidad pertenece, con lo que, en la página de restaurantes, se mostrarán los primeros, los restaurantes de su misma localidad. Los usuarios pueden ver una breve descripción del restaurante, su puntuación promedio y un enlace para más detalles.

Ordenador



INICIO RESTAURANTES HISTORIA



Filtrar por Localidad: Filtrar por Puntuación:



MESÓN GATO NEGRO

Dirección: Posada de Llanera (Llanera)

Teléfono: 985 77 03 72

Tablet



Filtrar por Localidad: Filtrar por Puntuación:



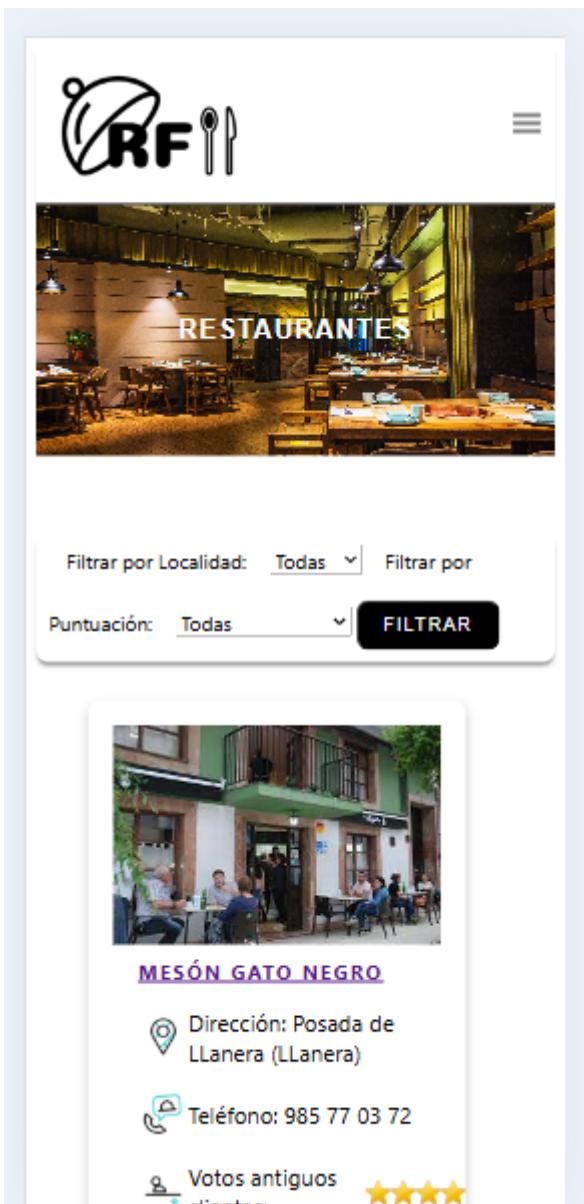
MESÓN GATO NEGRO

Dirección: Posada de Llanera (Llanera)

Teléfono: 985 77 03 72

Votos antiguos clientes:

Móvil



- **Página del restaurante:** cada restaurante tiene su propia página que muestra toda la información necesaria, como nombre, dirección y número de teléfono, ubicaciones en el mapa interactivo. Los usuarios registrados pueden dejar reseñas y valoraciones sobre sus experiencias, al igual que puede visualizar comentarios de antiguos clientes. Si el usuario no está registrado, será redirigido automáticamente a la página de inicio de sesión.

Ordenador USUARIO NO REGISTRADO



Filtrar por Localidad: Todas Filtrar por Puntuación: Todas FILTRAR



MESÓN GATO NEGRO

📍 Dirección: Posada de Llanera (Llanera)

☎️ Teléfono: 985 77 03 72

⭐️ Votos antiguos clientes: ★★★★☆

Tablet USUARIO NO REGISTRADO

Filtrar por Localidad: Todas Filtrar por Puntuación: Todas FILTRAR

MESÓN GATO NEGRO

📍 Dirección: Posada de Llanera (Llanera)

☎️ Teléfono: 985 77 03 72

⭐️ Votos antiguos clientes: ★★★★☆

Móvil USUARIO NO REGISTRADO

The screenshot shows the RRF app's search results page. At the top is the RRF logo. Below it is a large image of a restaurant interior with tables and chairs. Underneath the image are two filter buttons: "Filtrar por Localidad: Todas" and "Filtrar por Puntuación: Todas". A "FILTRAR" button is located below these filters. Below the filters is a smaller image of the restaurant's exterior. The restaurant's name, "MESÓN GATO NEGRO", is displayed in bold capital letters. Below the name are two contact details: "Dirección: Posada de Llanera (Llanera)" with a location pin icon, and "Teléfono: 985 77 03 72" with a phone icon.

Ordenador USUARIO REGISTRADO

The screenshot shows the RRF website's search results page. At the top is a banner image of a restaurant interior. Below it are two filter buttons: "Filtrar por Localidad: Todas" and "Filtrar por Puntuación: Todas". A "FILTRAR" button is located below these filters. Below the filters is a smaller image of the restaurant's exterior. The restaurant's name, "MESÓN GATO NEGRO", is displayed in bold capital letters. Below the name are three contact details: "Dirección: Posada de Llanera (Llanera)" with a location pin icon, "Teléfono: 985 77 03 72" with a phone icon, and "Votos antiguos clientes: ★★★★☆" with a star rating icon. At the bottom of the page is a button that says "¿HAS VISITADO ESTE RESTAURANTE? DEJA TU COMENTARIO".



Tablet USUARIO REGISTRADO

Filtrar por Localidad: Todas Filtrar por Puntuación: Todas FILTRAR



MESÓN GATO NEGRO

📍 Dirección: Posada de Llanera (Llanera)

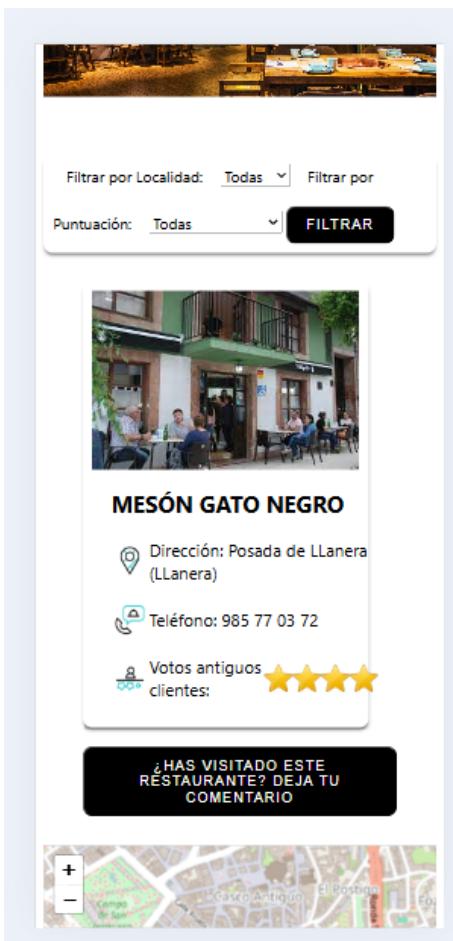
📞 Teléfono: 985 77 03 72

⭐ Votos antiguos clientes: ★★★★

¿HAS VISITADO ESTE RESTAURANTE? DEJA TU COMENTARIO

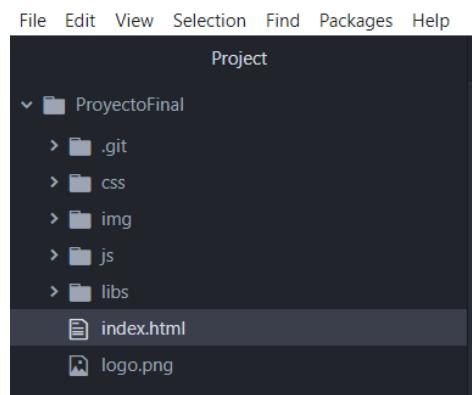


Móvil USUARIO REGISTRADO



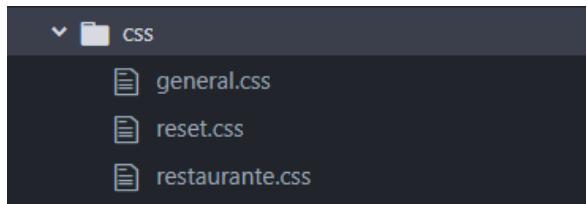
6.2 Código de programación

En este apartado se presentan los fragmentos más relevantes del código de **ResFav**, con explicaciones sobre su funcionalidad e implementación. La estructura del código ha sido diseñada para garantizar una experiencia fluida para los usuarios y facilitar el

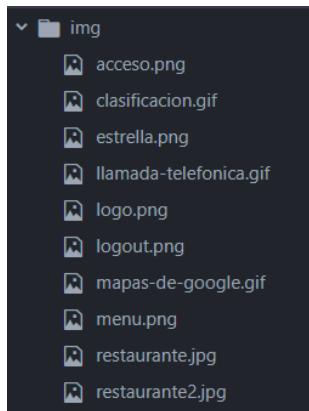


mantenimiento del sistema.

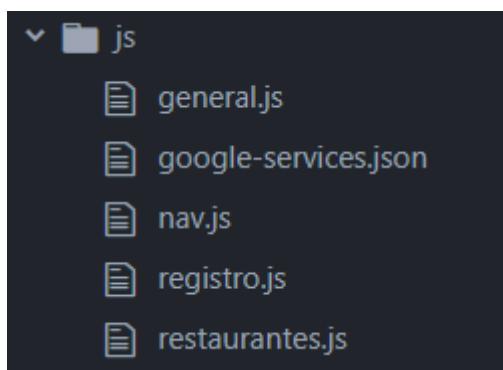
Estructura de todas las carpetas con la lógica de la Aplicación Web en Atom.



Dentro de la carpeta CSS, contiene todo el contenido CSS con que diseñamos el aspecto gráfico de nuestra página web.



En la carpeta de **Img**, se almacenan todas las imágenes y gif, que se están usando actualmente en la página web.



En la carpeta JS, disponemos de todo el código de programación que hemos usado. El archivo **nav.js**, contiene la configuración del menú. El **general.js**, toda la configuración del código de la aplicación web. Google-service.json y

registro.js, se nos ha implementado al conectar con FireBase. Restaurantes.js , donde se configura para que se muestren correctamente todos los restaurantes.

6.3 Datos Almacenados en Firebase Firestore

Firebase Firestore es la base de datos utilizada en **ResFav** para almacenar y gestionar la información de los restaurantes, los usuarios y las localidades. Su diseño flexible y escalable permite organizar los datos en **colecciones** y

documentos, adaptándose perfectamente a las necesidades del sistema.

La primera colección en **Firebase Firestore** es la de **Localidades**. En esta colección se almacenan los nombres de las localidades disponibles en la plataforma. Cada documento de la colección representa una localidad específica y contiene una subcolección denominada **Ubicaciones**, en la que se registran las ubicaciones específicas asociadas a esa localidad. Esto permite organizar y relacionar de manera eficiente la información geográfica con los restaurantes registrados en cada zona.

The screenshot shows the Firebase Firestore interface. At the top, there's a navigation bar with icons for home, collections, and a specific document ID: EU0WFBNsChdGaMqrHX4y. On the right, there's a link to 'Más funciones en Google Cloud' and a dropdown menu. Below the navigation, there's a header with a 'default' icon, the collection name 'localidades', and a document ID 'EU0WFBNsChdGaMqrHX4y'. There are also buttons for 'Iniciar colección' and 'Agregar documento'. The main area displays two documents under 'localidades': 'restaurantes' and 'usuarios'. The 'restaurantes' document has a subcollection 'ubicaciones' containing three items: 'Oviedo', 'La Corredoria', and 'Monte Cerrao'. Each item is numbered 0, 1, and 2 respectively. The 'usuarios' document has a single field 'nombre' with the value 'Oviedo'.

La segunda colección en **Firebase Firestore** es la de **Restaurantes**. En esta colección se almacena toda la información relevante sobre cada restaurante, incluyendo:

- Nombre del restaurante.
- Dirección y número de contacto.
- Referencia a la localidad donde está ubicado.
- Imágenes relacionadas con el restaurante, como fotografías de la fachada o del menú.
- Valoración promedio calculada a partir de los comentarios de los usuarios.
- Una subcolección de **Comentarios**, donde se registran las opiniones y

puntuaciones realizadas por los usuarios.

Esta estructura permite gestionar y mostrar de manera eficiente todos los detalles necesarios para que los usuarios puedan tomar decisiones informadas al seleccionar un restaurante.

The screenshot shows the Firebase Firestore interface. At the top, there's a navigation bar with icons for home, collections, and a specific document path: 'restaurantes > 2afpDTolbisK9dzmUGjC'. To the right of the path is a 'Más funciones en Google Cloud' dropdown. Below the navigation is a table structure. The left column shows a tree view of collections: '(default)', 'localidades', 'restaurantes' (which is expanded to show documents like '2afpDTolbisK9dzmUGjC', 'awKDF0xD122ZCXT0KOW7', etc.), and 'usuarios'. The middle column shows the details of the selected document '2afpDTolbisK9dzmUGjC', which includes fields like 'comentarios', 'telefono', 'imagen', 'latitud', 'localidad', 'longitud', 'nombre', 'puntos', 'ubicacion', and 'votos'. The right column contains buttons for 'Iniciar colección' and 'Agregar campo'.

La última colección en **Firebase Firestore** es la de **Usuarios**. En esta colección se almacena la información de los usuarios registrados en la plataforma, incluyendo:

- Nombre del usuario.
- Correo electrónico utilizado para el registro.
- Fecha de registro en la plataforma.
- Localidad asociada al usuario, que permite personalizar la experiencia al mostrar primero los restaurantes pertenecientes a esa localidad.

Esta estructura no solo mejora la personalización de la plataforma, sino que también facilita la implementación de funciones como la priorización de restaurantes locales en los resultados de búsqueda, creando una experiencia

más relevante para cada usuario.

A screenshot of the Google Cloud Firestore interface. The left sidebar shows collections: 'localidades', 'restaurantes', and 'usuarios'. The 'usuarios' collection is selected. Under it, a document named 'wU2BHsVvh1jxQKp1Ki4' is expanded, showing its fields: 'email' with the value 'prueba10@yopmail.com' and 'localidad' with the value 'oviedo'. The top right corner has a link to 'Más funciones en Google Cloud'.

7. IMPLEMENTACIÓN

7.1 Tecnologías Utilizadas en el Desarrollo del Proyecto

El desarrollo de **ResFav** ha sido posible gracias al uso de diversas tecnologías modernas, seleccionadas estratégicamente para cubrir las necesidades tanto del frontend como del backend de la plataforma:

- **Frontend:**
 - **HTML5:** Para la estructura de las páginas web.
 - **CSS3:** Para el diseño y estilo visual de las páginas.
 - **JavaScript:** Para agregar interactividad y dinamismo a la interfaz de usuario.
 - **Leaflet:** Para integrar mapas interactivos en la plataforma.
- **Backend y Gestión de Datos:**
 - **Firebase Firestore:** Para el almacenamiento y gestión en tiempo real de

los datos de usuarios, restaurantes y localidades.

- **Firebase Authentication:** Para el registro e inicio de sesión seguro.

- **Desarrollo:**

- **Atom:** Editor de código utilizado para escribir y organizar el proyecto.
- **XAMPP:** Para pruebas locales de desarrollo en servidor.

7.2 Descripción del Proyecto

Este apartado detalla las diferentes capas de la arquitectura de la aplicación y presenta el código correspondiente a las principales funcionalidades.

7.2.1 Capa de Presentación

La capa de presentación corresponde al **frontend**, es decir, la parte visible para el usuario. A continuación, se muestran los códigos de las páginas principales.

index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ResFav</title>
    <script src="https://www.gstatic.com/firebasejs/10.12.0/firebase-app-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/10.12.0.firebaseio-firestore-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/10.12.0.firebaseio-storage-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/10.12.0.firebaseio-auth-compat.js"></script>
    <script src="libs/rater.js-master/index.js"></script>
    <link rel="stylesheet" href="css/general.css" >!-- Si tienes un archivo de estilos CSS -->
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
        integrity="sha256-p4NvAqJ8HIIhWzRCf9tD/mIzyoh55obTRR9bMY="
        crossorigin="" />
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"
        integrity="sha256-20nQchB9co0qIjJZRGuk2/Z9VM+kNiyxN1lvTlZBo="
        crossorigin="" ></script>
</head>
<body>
    <script src="js/general.js"></script>
    <script src="js/restaurantes.js"></script>
    <script src="js/nav.js"></script>
    <header>
        <!-- Contenedor para el logo y el menú -->
        <div class="header-container">
            <!-- Logo -->
            <a href="index.html"></a>
            <!-- Menú -->
            <nav class="menu_principal oculto-movil">
                <ul>
                    <li><a href="#" onclick="irAIInicio()">Inicio</a></li>
                    <li><a href="#" onclick="irARestaurantes()">Restaurantes</a></li>
                    <li><a href="#" onclick="irAHistoria()">Historia</a></li>
                    <li id="IniciarSesion"><a href="#" class="icono-iniciar-sesion" onclick="irAIniciarSesion()"></a></li>
                    <li id="cerrarSesion"><a href="#" class="icono-iniciar-sesion" onclick="logout()"></a></li>
                </ul>
            </nav>
            <a href="index.html" onclick="mostrarMenu(event)" class="icono-menu icono-iniciar-sesion">
                </a>
            </a>
        </div>
    </header>
    <!-- Página de inicio -->
    <div class="pag" id="pag-inicio">
        <div class="imagen_portada">
            <div class="contenido_portada">
```

```

        <h1>Bienvenido a nuestra página de restaurantes</h1>
        <a onclick="irARestaurantes()" class="boton_portada">Conoce los restaurantes</a>
    </div>
</div>
</div>

<!--Pagina de inicio -->

<div class="pag" id="pag-restaurantes">
    <div class="imagen_portadas_pag">
        <div class="contenido_portada">
            <h1>Restaurantes</h1>
        </div>
    </div>
    <!-- Mostrar lista de restaurantes-->
    <div class="contenedor-filtros">

        <form id="filtro-form">
            <label for="localidad">Filtrar por Localidad:</label>
            <select id="localidad" name="localidad">
                <option value="">Todas</option>
                <option value="Llanera">Llanera</option>
                <option value="Oviedo">Oviedo</option>
                <option value="Siero">Siero</option>
                <option value="Gijon">Gijon</option>
            </select>
            <label for="puntuacion">Filtrar por Puntuación:</label>
            <select id="puntuacion" name="puntuacion">
                <option value="">Todas</option>
                <option value="5">5 estrellas</option>
                <option value="4">4 estrellas o más</option>
                <option value="3">3 estrellas o más</option>
                <option value="2">2 estrellas o más</option>
                <option value="1">1 estrella o más</option>
            </select>
            <button type="button" onclick="filtrarRestaurantes()>Filtrar</button>
        </form>
    </div>

    <div id="lista-restaurantes"></div>
</div>

<!--Pagina de iniciar sesion-->

<div class="pag" id="pag-iniciarsesion">
    <div class="imagen_portadas_pag">
        <div class="contenido_portada">
            <h1> Iniciar Sesión</h1>

```

```

</div>
<div class="contenedor-login">

<div class="container-iniciar">
    
    <form class="form-signin" action="codigo/php/login.php">
        <h1 class="piginiciar">Iniciar sesión</h1>
        <div class="contenedoremail"><input type="email" id="inputEmail" class="form-control mb-3" placeholder="Correo electrónico" required autofocus></div>
        <input type="password" id="inputPassword" class="form-control mb-3" placeholder="Contraseña" required>
        <div class="row">
            <div class="col">
                <button type="button" onclick="login()">Iniciar sesión</button>
            </div>
        </div>
    </form>
</div>
<div class="container-registrarse">
    
    <form class="form-signin" action="codigo/php/login.php">
        <h1 class="piginiciar">Registrarse</h1>
        <div class="contenedoremail"><input type="email" id="inputEmailRegistrarse" class="form-control mb-3" placeholder="Correo electrónico" required autofocus></div>
        <div class="contenedordcontraseña"><input type="password" id="inputPasswordRegistrarse" class="form-control mb-3" placeholder="Contraseña" required></div>
        <select id="inputLocalidadRegistrarse" class="form-control mb-3" placeholder="Localidad a la que pertenes" required>
            <option value="opcion">Localidad a la que perteneces</option>
            <option value="oviedo">Oviedo</option>
            <option value="Llanera">Llanera</option>
            <option value="Oviedo">Oviedo</option>
            <option value="Siero">Siero</option>
            <option value="Gijon">Gijón</option>
        </select>
        <div class="row">
            <div class="col">
                <button type="button" onclick="registrarse()">Registrarse</button>
            </div>
        </div>
    </form>
</div>
</div>
<!---insetar mapa con todas las ubicaciones-->
<div id="map"></div>
<footer>
    <div class="footer-contenedor">
        <div class="footer-columna">
            
        </div>
        <div class="row">
            <div class="col-12">
                <div class="row">
                    <div class="col-6">
                        <p>Gracias por visitar nuestro sitio. Estamos comprometidos a brindarle la mejor experiencia gastronómica.</p>
                    </div>
                    <div class="col-6">
                        <ul class="footer-enlaces">
                            <li><a href="index.html">Inicio</a></li>
                            <li><a href="restaurantes.html">Restaurantes</a></li>
                            <li><a href="iniciar_sesion.html">Iniciar Sesión</a></li>
                        </ul>
                    </div>
                </div>
                <p>© 2024 ResFav. Todos los derechos reservados.</p>
            </div>
        </div>
    </div>
</footer>
<div id="alert">
    <div id="alert-contenido">
        <h1 id="alert-titulo">Título del mensaje</h1>
        <p id="alert-texto">Texto del mensaje</p>
        <div class="botones">
            <button type="button" onclick="ocultarAlert()">Cerrar</button>
        </div>
    </div>
</div>
<script src="js/general.js"></script> <!-- Archivo de script JavaScript -->
</body>
</html>

```

La página **index.html** es una página estática que utiliza **HTML5** para la estructura básica y **CSS3** para los estilos. Además, incluye un diseño responsive que asegura una visualización adecuada en diferentes dispositivos.

La página sirve como el punto de entrada a la plataforma **ResFav**, presentando enlaces a las secciones clave de la web, como la lista de restaurantes, el inicio de sesión y el registro de nuevos usuarios. Incluye un encabezado con el título de la aplicación y un menú de navegación intuitivo.

En el contenido principal, se encuentra un mensaje de bienvenida, junto con un enlace destacado para explorar restaurantes disponibles. El pie de página contiene un mensaje de derechos reservados y asegura una experiencia coherente en todas las páginas de la plataforma.

El diseño está optimizado para carga rápida y navegación sencilla, mejorando la experiencia de los usuarios desde el primer contacto con la plataforma.

general.js

El archivo **general.js** gestiona diversas funcionalidades en la aplicación ResFav. Su propósito es facilitar la interacción con la base de datos en Firebase, manejar la autenticación de usuarios y actualizar dinámicamente la información mostrada en la interfaz de usuario. A continuación, se describe el funcionamiento de las principales funciones:

```

1 const config = {
2   apiKey: "AIzaSyCTbIaLKWlQXgT5zGSg4Iobq72005Q2e_A",
3   authDomain: "proyectofinal-1a97e.firebaseio.com",
4   projectId: "proyectofinal-1a97e",
5   storageBucket: "proyectofinal-1a97e.appspot.com",
6   messagingSenderId: "1086701222701",
7   appId: "1:1086701222701:web:b87ef0f5c40f5717cd71a4"
8 };
9 // Configurar Firebase con las credenciales del archivo JSON
10 firebase.initializeApp(config);
11
12 // Referencia a la base de datos
13 var database = firebase.firestore();
14 var storage = firebase.storage();
15 var auth = firebase.auth();
16
17 var provider = new firebase.auth.GoogleAuthProvider();
18
19 var restaurantes = [];
20 var comentarios = [];
21 // Recuperar los restaurantes de la base de datos
22
23 async function inicializar() {
24   // await cargarLogin();
25   await cargarRestaurantes();
26   //mostrarTodos();
27
28
29   const user = auth.currentUser;
30   if (user == null) {
31     irAIniciarSesion();
32   }
33   else {
34     irAInicio();
35   }
36   //await cargarComentarios();
37
38   const map = L.map('map').setView([43.361394558091256, -5.850188564940662], 13);
39
40   const tiles = L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
41     maxZoom: 19,
42     attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
43   }).addTo(map);
44
45   mostrarPuntosRestaurantes(map);
46
47   actualizarBotonesLogin();
48 }

```

```

49
50  /*cargar los comentarios de los clientes*/
51  async function cargarComentarios(idRestaurante) {
52    const snapshot = await database
53      .collection('restaurantes')
54      .doc(idRestaurante)
55      .collection('comentarios').get();
56
57    comentarios = [];
58    snapshot.forEach(async function(doc) {
59      const comentario = doc.data()
60      comentario._id = doc.id
61      comentarios.push(comentario);
62      console.log(comentario);
63    });
64  }
65
66 /*mostrar solamente un restaurante*/
67 async function mostrarUno(id, event) {
68   if (event && event.preventDefault)
69     event.preventDefault();
70
71   cargarComentarios(id);
72
73   document.getElementById('lista-restaurantes').innerHTML = '';
74   restaurante = restaurantes.find(r => r._id == id);
75   // Aquí puedes hacer algo con cada restaurante, como agregarlo a tu página HTML
76
77   var estrellas = restaurante.puntos / restaurante.votos;
78   var imagenesEstrellas = '';
79   var contEstrella = 0;
80   while (contEstrella < estrellas) {
81     imagenesEstrellas += '';
82     contEstrella++;
83   }
84
85   var nombre = restaurante.nombre;
86   var direccion = restaurante.ubicacion + ' (' + restaurante.localidad + ')';
87   var telefono = restaurante.Telefono;
88   var puntuacion = imagenesEstrellas;
89   var ubicacion = restaurante.latitud + '/' + restaurante.longitud ;
90   // var comentarios = restaurante.comentarios;
91
92   var refImagen = await storage.refFromURL(restaurante.imagen).getDownloadURL();
93
94   // Agregar el restaurante a tu página HTML
95   var restauranteElemento = document.createElement('div');
96
97   var html = `
```

```
97  var html = "
98    <div class="contenedor-restaurante">
99      
100     <div id="restaurante-info">
101       <h2>${nombre}</h2>
102       <p>
103         
104         Dirección: ${direccion}
105       </p>
106       <p>
107         
108         Teléfono: ${telefono}
109       </p>
110       <p>
111         
112         Votos antiguos clientes: ${puntuacion}
113       </p>
114     </div>
115   </div>
116 `;
117
118 restauranteElemento.innerHTML = html;
119 }
120 const comentarios = await mostrarcomentarios(id);
121 comentarios.style.display = 'none';
122 //botón para opinar clientes
123 const user = auth.currentUser;
124 if (user !== null) {
125
126   const botón = document.createElement('button');
127   botón.textContent = '¿Has visitado este restaurante? Deja tu comentario';
128
129   //popup para q dejen comentario
130   botón.addEventListener('click', function() {
131     comentarios.style.display = 'block';
132     mostrarAlert('¿Has visitado este restaurante?', 'Dejanos tu opinión para que el resto de participantes de la aplicación puedan saber tus opiniones');
133   });
134
135   // Agregar el botón al contenedor del restaurante
136   restauranteElemento.appendChild(botón);
137   restauranteElemento.appendChild(comentarios);
138 }
139
140 document.getElementById('lista-restaurantes').appendChild(restauranteElemento);
141 }
142
143
144 async function mostrarcomentarios(idRestaurante) {
```

```
144  async function mostrarcomentarios(idRestaurante) {
145    var resultado = document.createElement('div');
146
147    /*escribir comentario*/
148    const raterDivCont = document.createElement('div'); //estrellitas para opinar del restaurante
149    const raterDiv = document.createElement('div'); //estrellitas para opinar del restaurante
150    const rater = raterJs({
151      element: raterDiv,
152      starSize: 40,
153      rateCallback:function rateCallback(rating, done) {
154        this.setRating(rating);
155        done();
156      }
157    });
158
159    raterDivCont.appendChild(raterDiv);
160    resultado.appendChild(raterDivCont);
161
162
163    const textocomentario = document.createElement('textarea');
164    textocomentario.setAttribute('id', 'textocomentario');
165    textocomentario.placeholder = 'Escribe tu comentario aquí...';
166    resultado.appendChild(textocomentario);
167
168    const botoncomentario = document.createElement('button');
169    botoncomentario.textContent = 'Deja un comentario sobre este restaurante';
170
171    const mensajeConfirmacion = document.createElement('p');
172    mensajeConfirmacion.textContent = 'Tu comentario se guardó correctamente';
173    mensajeConfirmacion.style.display = 'none'; // Inicialmente oculto
174    resultado.appendChild(mensajeConfirmacion);
175
176    botoncomentario.addEventListener('click', function() {
177      aniadirComentario(idRestaurante, rater.getRating()); // Asume que esta función está definida en otro lugar
178
179      // Mostrar mensaje de confirmación
180      mensajeConfirmacion.style.display = 'block';
181
182      // Ocultar mensaje después de unos segundos
183      setTimeout(function() {
184        mensajeConfirmacion.style.display = 'none';
185      }, 3000); // Ocultar mensaje después de 3 segundos
186
187      // Ocultar textarea y botón de comentario
188      textocomentario.style.display = 'none';
189      raterDiv.style.display = 'none';
190      botoncomentario.style.display = 'none';
191    });
192
```

```
194     resultado.appendChild(botoncomentario);
195
196
197
198     comentarios.forEach(async (comentarios) => {
199         console.log(comentarios);
200         // Aquí puedes hacer algo con cada restaurante, como agregarlo a tu página HTML
201         var correoelectronico = comentarios.nombre;
202         var contenido = comentarios.contenido;
203
204         // Agregar el restaurante a tu página HTML
205         var comentarioElemento = document.createElement('div');
206         comentarioElemento.innerHTML = `h2 id="h2comentarios">Comentario de antiguo cliente:</h2><p><b>Correo cliente: </b> ${correoelectronico} </p><p><b>Comentario: </b> ${contenido} </p>`;
207         resultado.appendChild(comentarioElemento);
208     });
209
210
211
212 /*boton cerrar */
213 const boton = document.createElement('button');
214 boton.textContent = 'Cerrar comentarios';
215
216 // popup para q dejen comentario
217 boton.addEventListener('click', function() {
218     resultado.style.display = 'none';
219 });
220
221 });
222 resultado.appendChild(boton);
223 return resultado;
224 }
225
226 /*opcion añadir comentario*/
227 async function añadirComentario(idRestaurante, puntuacion) {
228     console.log(puntuacion);
229
230     const user = auth.currentUser;
231     if (user === null) {
232         alert('No puedes comentar sin registrarte');
233     }
234     else {
235
236         const texto = document.getElementById('textocomentario').value;
237         const email = user.email;
238
239         const snapshot = await database
240             .collection('restaurantes')
241             .doc(idRestaurante)
```

```
242     .collection('comentarios').add({
243         contenido: texto,
244         nombre: email
245     });
246 }
247
248 const snapshot2 = await database
249     .collection('restaurantes')
250     .doc(idRestaurante)
251     .update({
252         votos: firebase.firestore.FieldValue.increment(1),
253         puntos: firebase.firestore.FieldValue.increment(puntuacion)
254     });
255
256
257 }
258
259
260 //poder registrar y cerrar sesion
261 function actualizarBotonesLogin() {
262     const user = auth.currentUser;
263     if (user === null) {
264         document.getElementById('iniciarSesion').style.visibility = 'visible';
265         document.getElementById('cerrarSesion').style.visibility = 'hidden';
266     }
267     else {
268         document.getElementById('iniciarSesion').style.visibility = 'hidden';
269         document.getElementById('cerrarSesion').style.visibility = 'visible';
270     }
271 }
272
273 async function login() {
274
275     const email = document.getElementById('inputEmail').value;
276     const password = document.getElementById('inputPassword').value;
277
278     try {
279         const userCredentials = await auth.signInWithEmailAndPassword(email, password);
280         console.log(userCredentials);
281         mostrarAlert('Login completado', 'Te has logueado correctamente en la aplicación');
282     }
283     catch (e) {
284         mostrarAlert('Error', 'El usuario o contraseña no son correctos');
285     }
286     actualizarBotonesLogin();
287 }
288
289 async function logout() {

```

```

288
289     async function logout() {
290         await auth.signOut();
291         mostrarAlert('Logout completado', 'Te has desconectado correctamente de la aplicación');
292         actualizarBotonesLogin();
293     }
294
295     async function registrarse() {
296
297         const email = document.getElementById('inputEmailRegistrarse').value;
298         const password = document.getElementById('inputPasswordRegistrarse').value;
299         const location = document.getElementById('inputLocalidadRegistrarse').value;
300         try {
301             const userCredentials = await auth.createUserWithEmailAndPassword(email, password);
302             console.log(userCredentials);
303
304             const snapshot = await database
305                 .collection('usuarios').add({
306                     email,
307                     localidad: location
308                 });
309             mostrarAlert('Registro completado', 'Te has registrado correctamente de la aplicación');
310         }
311         catch (e) {
312             mostrarAlert('Error', 'Correo o clave no son correctas');
313         }
314     }
315 }
316
317
318 //mostrar popup como ventana emergente cuando algo no esta correcto
319 function mostrarAlert(titulo, mensaje) {
320     document.getElementById('alert').style.display = 'flex';
321     document.getElementById('alert-titulo').innerText = titulo;
322     document.getElementById('alert-texto').innerText = mensaje;
323 }
324
325 function ocultarAlert() {
326     document.getElementById('alert').style.display = 'none';
327 }
328
329 window.onload = inicializar;
330

```

1. Inicialización y Firebase

- **Configuración de Firebase:**

Se inicializa Firebase con las credenciales del proyecto y se configuran las referencias a la base de datos (firestore), almacenamiento de imágenes (storage), y la autenticación de usuarios (auth).

- **Función inicializar():**

Se cargan los restaurantes desde la base de datos, se verifica si el usuario está

autenticado (para redirigirlo al inicio de sesión si es necesario), y se configura un mapa interactivo con **Leaflet** para mostrar la ubicación de los restaurantes.

2. Funcionalidades de Restaurantes y Comentarios

- **Mostrar Restaurantes:**

- La función **mostrarUno()** muestra la información de un restaurante seleccionado, incluyendo su nombre, dirección, puntuación y comentarios.
- Se carga la imagen del restaurante desde **Firebase Storage** y se agrega dinámicamente a la interfaz.

- **Mostrar Comentarios:**

- **mostrarcomentarios()** recupera y muestra los comentarios de los usuarios sobre un restaurante en particular.
- Permite a los usuarios registrados agregar nuevos comentarios y puntuaciones utilizando **rater.js** para la funcionalidad de calificación con estrellas.

- **Añadir comentarios:**

- **aniadirComentario()** guarda el comentario y la puntuación del usuario en Firestore y actualiza la puntuación y los votos del restaurante.

3. Gestión de Autenticación de Usuarios

- **Función actualizarBotonesLogin():**

Actualiza la visibilidad de los botones de "Iniciar sesión" y "Cerrar sesión" según si el usuario está autenticado o no.

- Login y Registro:

- **login()** permite a los usuarios ingresar utilizando su correo electrónico y contraseña.
- **logout()** cierra la sesión del usuario.
- **registrarse()** permite el registro de nuevos usuarios, incluyendo la

asociación con su localidad.

4. Ventanas Emergentes y Notificaciones

- **Mostrar Alertas:**

- La función **mostrarAlert()** se utiliza para mostrar mensajes emergentes (popup) con notificaciones de error o éxito, como en el caso del registro o inicio de sesión.

- **Ocultar Alertas:**

- **ocultarAlert()** cierra la ventana emergente cuando el usuario interactúa con ella.

5. Funciones Adicionales

- **Mapa Interactivo:**

Utiliza **Leaflet** para cargar el mapa de la localidad con la ubicación de los restaurantes.

- **Autenticación con Google:**

Se configura el proveedor de autenticación de Google (**GoogleAuthProvider**) para permitir a los usuarios iniciar sesión con sus cuentas de Google.

nav.js

El archivo **nav.js** se encarga de configurar el comportamiento del menú de navegación, destacando el enlace correspondiente según la sección seleccionada por el usuario.

```

1 //NAVEGACION DEL MENU
2 function ocultarTodas() {
3     var pags = document.querySelectorAll('.pag');
4     pags.forEach((pag, i) => {
5         pag.style.display = 'none';
6     });
7 }
8
9 function irAPagina(pagina) {
10    ocultarTodas();
11    var pagInicio = document.querySelector('#pag-' + pagina);
12    pagInicio.style.display = 'block';
13    document.querySelector('.menu_principal').classList.add('oculto-movil');
14 }
15
16 function irAInicio() {
17    irAPagina('inicio');
18 }
19
20
21 async function irARestaurantes() {
22    irAPagina('restaurantes');
23    await mostrarRestaurantes();
24 }
25
26 async function irARestaurante(id) {
27    irAPagina('restaurantes');
28    await mostrarUno(id);
29 }
30
31 function irAHistoria() {
32    irAPagina('historia');
33 }
34
35 function irAContacto() {
36    irAPagina('contacto');
37 }
38
39 function irAIniciarSesion() {
40    irAPagina('iniciarsesion');
41 }
42
43 function mostrarMenu(ev) {
44    ev.preventDefault();
45    document.querySelector('.menu_principal').classList.remove('oculto-movil');
46 }
47

```

1. Funciones de Navegación de Páginas

- **ocultarTodas()**

Esta función oculta todas las páginas de la aplicación. Se utiliza al cambiar de sección para asegurarse de que solo se muestre la página correspondiente.

- **irAPagina(pagina)**

Esta función permite cambiar a una página específica. Se ocultan todas las páginas y se

muestra solo la página seleccionada. El parámetro pagina corresponde al nombre de la página que se desea mostrar (como "inicio", "restaurantes", etc.).

- **irALInicio()**

Redirige al usuario a la página de inicio, utilizando la función irAPagina().

- **irARestaurantes()**

Muestra la página de restaurantes y carga los datos de los restaurantes mediante la función mostrarRestaurantes().

- **irARestaurante(id)**

Muestra la página de detalles de un restaurante específico. Carga los datos del restaurante utilizando la función mostrarUno(id).

- **irAHistoria()**

Muestra la página de historia de la plataforma.

- **irAContacto()**

Muestra la página de contacto.

- **irAlIniciarSesion()**

Muestra la página de inicio de sesión.

2. Funciones de Menú

- **mostrarMenu(ev)**

Muestra el menú principal cuando el usuario interactúa con el ícono de menú. Se elimina la clase oculto-movil para que el menú sea visible en dispositivos móviles.

registro.js

El archivo **restaurantes.js** es responsable de gestionar la visualización de los

restaurantes en la página, además de aplicar filtros por localidad y puntuación.

A continuación, se muestra el código para cargar los restaurantes y aplicar los filtros.

```
var restaurantes = [];
async function cargarRestaurantes() {
    const snapshot = await database.collection('restaurantes').get();
    restaurantes = [];
    snapshot.forEach(async function(doc) {
        const restaurante = doc.data()
        restaurante._id = doc.id
        restaurantes.push(restaurante);
    });
}

async function mostrarRestaurante(restaurante) {
    // Aquí puedes hacer algo con cada restaurante, como agregarlo a tu página HTML
    var nombre = restaurante.nombre;
    var direccion = restaurante.ubicacion + (' + restaurante.localidad + ');
    var telefono = restaurante.Telefono;
    var estrellas = restaurante.puntos / restaurante.votos;
    var imagenesEstrellas = '';
    var contEstrella = 0;
    while (contEstrella < estrellas) {
        imagenesEstrellas += '';
        contEstrella++;
    }
    var refImagen = await storage.refFromURL(restaurante.imagen).getDownloadURL();
    console.log(refImagen);
    // Agregar el restaurante a tu página HTML
    var restauranteElemento = document.createElement('div');
    var html = `
        <div class="contenedor-restaurante diseñolistarestaurante" >
            
            <div id="restaurante-info">
                <h2 id="titulo_Restaurante">
                    <a href="#" onclick="mostrarUno('${restaurante._id}', event);">
                        ${nombre}
                    </a>
                </h2>
                <p>
                    
                    Dirección: ${direccion}
                </p>
                <p>
                    
                    Teléfono: ${telefono}
                </p>
        </div>
    `;
    restauranteElemento.innerHTML = html;
    document.body.appendChild(restauranteElemento);
}
```

```
44         Teléfono: ${telefono}
45     </p>
46     <p>
47         
48         Votos antiguos clientes: ${imagenesEstrellas}
49     </p>
50     <button onclick="mostrarUno('${restaurante._id}', event);">Conocer restaurante </button>
51     </div>
52   </div>
53 `;
54 restauranteElemento.innerHTML = html;
55 document.getElementById('lista-restaurantes').appendChild(restauranteElemento);
56 }
57 var pagina = 0;
58 /*mostrar todos los restaurantes*/
59 async function mostrarRestaurantes() {
60     document.getElementById('lista-restaurantes').innerHTML = '';
61     const user = auth.currentUser;
62     let localidad = false;
63     if (user) {
64         const snapshot = await database.collection('usuarios')
65             .where('email', '==', user.email)
66             .get();
67         snapshot.forEach(doc => {
68             localidad = doc.data().localidad.toLowerCase();
69         });
70     }
71     if (localidad) {
72         restaurantes = [...restaurantes];
73         restaurantes.sort((r1, r2) => {
74             if (r1.localidad.toLowerCase() == localidad && r2.localidad.toLowerCase() != localidad) return -1;
75             if (r1.localidad.toLowerCase() != localidad && r2.localidad.toLowerCase() == localidad) return 1;
76             return r1.nombre.localeCompare(r2.nombre);
77         });
78     }
79     for (var i = 0; i < restaurantes.length; i++) {
80         if (i > pagina*5 && i < (pagina+1)*5) {
81             await mostrarRestaurante(restaurantes[i]);
82         }
83     }
84     const paginas = document.createElement('div');
85     paginas.innerHTML = '<a href="#" id="boton-anterior" onclick="paginaAnterior()">Anterior</a> <a href="#" id="boton-siguiente" onclick="paginaSiguiente()">Siguiente</a>';
86     document.getElementById('lista-restaurantes').appendChild(paginas);
87 }
88 
```

```

55     paginas.innerHTML = `<a href="#" id="boton-anterior" onclick="paginaAnterior()">Anterior</a> <a href="#"
56     document.getElementById('lista-restaurantes').appendChild(paginas);
57 }
58
59 function paginaAnterior() {
60     if (pagina == 0) return;
61     pagina--;
62     mostrarRestaurantes();
63 }
64
65 function paginaSiguiente() {
66     pagina++;
67     mostrarRestaurantes();
68 }
69
70 async function filtrarRestaurantes() {
71
72     const puntuacion = document.getElementById('puntuacion').value;
73     const localidad = document.getElementById('localidad').value;
74
75     document.getElementById('lista-restaurantes').innerHTML = '';
76     restaurantes
77         .filter(restaurante => {
78             const estrellas = restaurante.puntos / restaurante.votos;
79             if (estrellas < puntuacion) return false;
80             if (localidad != '' && localidad != restaurante.localidad) return false;
81             return true;
82         })
83         .forEach(mostrarRestaurante);
84 }
85
86 function mostrarPuntosRestaurantes(map) {
87     restaurantes.forEach((r, i) => {
88         L.marker([r.latitud, r.longitud])
89             .on('click', async () => {
90                 irARestaurante(r._id);
91             })
92             .addTo(map);
93     });
94 }
95
96 }
97
98 
```

1. Función cargarRestaurantes()

Esta función recupera todos los restaurantes almacenados en Firestore y los guarda en el array restaurantes. A continuación, el código recorre todos los documentos de la colección de restaurantes y los agrega al array.

- Proceso:
 - Hace una consulta a Firestore para obtener todos los restaurantes.
 - Los restaurantes se almacenan en el array restaurantes.

2. Función mostrarRestaurante(restaurante)

Muestra la información detallada de un restaurante en la interfaz. Para cada restaurante, se crea un elemento HTML que incluye su nombre, dirección, teléfono, puntuación y una imagen del restaurante, además de un enlace para ver más detalles.

- Elementos visualizados:
 - Nombre, dirección, teléfono.
 - Puntuación representada con estrellas.
 - Imagen del restaurante.
 - Un botón para ver más detalles sobre el restaurante.

3. Función mostrarRestaurantes()

Muestra todos los restaurantes almacenados en el array restaurantes con paginación. Primero, la función obtiene la localidad del usuario si está autenticado y, luego, muestra los restaurantes de esa localidad primero, manteniendo el resto ordenado alfabéticamente.

- Página:
 - Los restaurantes se muestran en bloques de 5 por página.
 - Permite navegar entre páginas con los botones "Anterior" y "Siguiente".

4. Función filtrarRestaurantes()

Permite filtrar los restaurantes por puntuación mínima y localidad. La función toma los valores seleccionados por el usuario en los filtros y actualiza la lista de restaurantes mostrados.

- Filtros:
 - Puntuación mínima.

- Localidad específica.

5. Función mostrarPuntosRestaurantes(map)

Muestra los restaurantes en un mapa interactivo utilizando la librería Leaflet. Los restaurantes se marcan en el mapa con un marcador y, al hacer clic en un marcador, se redirige al usuario a la página de detalles del restaurante.

- Mapa interactivo:

- Los restaurantes se muestran como puntos en un mapa.
- Al hacer clic en un punto, se carga la página con los detalles del restaurante correspondiente.

6. Paginación

- paginaAnterior() y paginaSiguiente()

Controlan la navegación entre las páginas de restaurantes. Al hacer clic en "Anterior" o "Siguiente", se carga la página correspondiente.

8. ANÁLISIS DAFO

Fortalezas (Internas)

- **Facilidad de uso:** ResFav es una plataforma intuitiva que permite a los usuarios encontrar y consultar restaurantes de manera sencilla según su ubicación, lo que mejora su experiencia.
- **Interacción con la comunidad:** Los usuarios pueden dejar comentarios y calificaciones sobre los restaurantes, lo que no solo fomenta la participación, sino que también ayuda a crear una comunidad activa en torno a la plataforma.
- **Integración con Google Maps:** Gracias a los mapas interactivos, los usuarios

pueden navegar fácilmente y localizar restaurantes, lo que es un gran plus para la experiencia de búsqueda.

- **Tecnología moderna:** Utilizamos Firebase para gestionar nuestra base de datos y la autenticación de usuarios, lo que asegura que la plataforma sea segura, confiable y capaz de crecer con nosotros.
- **Diseño adaptable:** ResFav está optimizada para funcionar bien en diferentes dispositivos, lo que la hace accesible y fácil de usar, ya sea en un móvil o en una computadora.

Debilidades (Internas)

- **Dependencia de servicios externos:** Nuestra plataforma depende de Firebase y Google Maps. Si alguno de estos servicios experimenta problemas, podría afectar la funcionalidad de ResFav.
- **Opciones limitadas de búsqueda:** Actualmente, los filtros para buscar restaurantes son bastante básicos, limitándose a la localidad y la puntuación. Ampliar estas opciones podría enriquecer la experiencia del usuario.
- **Contenido generado por usuarios:** Si no incentivamos a nuestros usuarios a dejar comentarios y valoraciones, podríamos quedarnos sin contenido relevante, lo que podría afectar la confianza en la plataforma.
- **Escalabilidad de la base de datos:** Aunque Firebase es escalable, un crecimiento rápido en el número de usuarios y restaurantes podría generar problemas si no se gestiona adecuadamente.

Oportunidades (Externas)

- **Crecimiento del mercado gastronómico:** La demanda de aplicaciones que

facilten la búsqueda de restaurantes está en aumento, lo que representa una gran oportunidad para ResFav.

- **Colaboraciones con restaurantes:** Establecer alianzas con restaurantes para ofrecer promociones exclusivas o descuentos podría motivar a los usuarios a interactuar más con nuestra plataforma.
- **Expansión geográfica:** Aunque actualmente estamos centrados en Asturias, hay una gran oportunidad de expandirnos a otras regiones o incluso a nivel nacional, lo que podría aumentar nuestra base de usuarios.
- **Integración con redes sociales:** Permitir que los usuarios compartan sus experiencias y recomendaciones en redes sociales podría aumentar la visibilidad de ResFav y atraer a más personas.
- **Personalización mejorada:** Implementar recomendaciones personalizadas basadas en el historial de búsqueda de los usuarios podría enriquecer aún más su experiencia en la plataforma.

Amenazas (Externas)

- **Competencia creciente:** Hay muchas aplicaciones y plataformas similares, como TripAdvisor, Yelp y Google Reviews, que ya están bien establecidas en el mercado y ofrecen funciones parecidas.
- **Riesgos por dependencia de terceros:** La integración con servicios como Google Maps y Firebase puede ser un riesgo si sus políticas cambian o si enfrentan problemas técnicos.
- **Desconfianza en las opiniones:** Los usuarios pueden dudar de la autenticidad de los comentarios y calificaciones si no hay una moderación adecuada o si perciben que no son genuinos.
- **Cambios en regulaciones de datos:** La gestión de datos personales está sujeta

a regulaciones como el GDPR. Cualquier cambio en estas leyes podría afectar cómo manejamos la privacidad y la seguridad de los datos de nuestros usuarios.

9. BIBLIOGRAFÍA

- <https://firebase.google.com/docs/auth?authuser=0&hl=es>
- <https://firebase.google.com/docs/>
- <https://developer.mozilla.org/en-US/docs/Web/CSS>
- <https://app.diagrams.net/>
- <https://leafletjs.com/reference.html>