Foto del alumno

(foto actual)

MÓDULO PROYECTO

Ciclo Superior Desarrollo de Aplicaciones Web

Departamento: Informática y Comunicaciones

IES “María Moliner”

Curso: 2023/2024

Grupo: S2I

Proyecto: Comida En Marcha

**Fernando Juan Estrada Gallardo**

Email: fern.j.e.g@gmail.com

Tutor individual: María José González García

Tutor colectivo: Enrique Carballo Albarrán

Fecha de presentación: (fecha de presentación)

Contenido

[Descripción general del proyecto 4](#_Toc167654289)

[Objetivos 4](#_Toc167654290)

[Cuestiones metodológicas 5](#_Toc167654291)

[Entorno de trabajo 5](#_Toc167654292)

[Descripción general del producto 7](#_Toc167654293)

[Límites del sistema 7](#_Toc167654294)

[Funcionalidades básicas 7](#_Toc167654295)

[Usuarios 7](#_Toc167654296)

[Hosting 7](#_Toc167654297)

[Planificación y presupuesto 10](#_Toc167654298)

[Planificación 10](#_Toc167654299)

[Metodología 11](#_Toc167654300)

[Bocetos 12](#_Toc167654301)

[Interfaces finales 14](#_Toc167654302)

[Admin 14](#_Toc167654303)

[Cliente 16](#_Toc167654304)

[Presupuesto 18](#_Toc167654305)

[Modelo de negocio 18](#_Toc167654306)

[Documentación técnica 19](#_Toc167654307)

[Base de datos 19](#_Toc167654308)

[Entidades 19](#_Toc167654309)

[Rutas API 21](#_Toc167654310)

[Estructura del código 22](#_Toc167654311)

[Librerías 22](#_Toc167654312)

[Códigos de interés 23](#_Toc167654313)

[Subida de imágenes 23](#_Toc167654314)

[Errores customizados para respuestas 25](#_Toc167654315)

[Enrutamiento con React Router 26](#_Toc167654316)

[Pruebas 27](#_Toc167654317)

[Manuales de usuario 28](#_Toc167654318)

[Instalación en local 28](#_Toc167654319)

[Instalación en la nube (versión recomendada) 28](#_Toc167654320)

[General 28](#_Toc167654321)

[Base de datos 28](#_Toc167654322)

[Frontends 31](#_Toc167654323)

[Servidor 33](#_Toc167654324)

[Servidor 36](#_Toc167654325)

[Uso 39](#_Toc167654326)

[Conclusiones, correcciones y ampliaciones 40](#_Toc167654327)

[Conclusiones 40](#_Toc167654328)

[Ampliaciones 40](#_Toc167654329)

[Correcciones 40](#_Toc167654330)

[Relación de ficheros en formato digital 41](#_Toc167654331)

[Bibliografía 42](#_Toc167654332)

[Glosario 43](#_Toc167654333)

# Descripción general del proyecto

## Objetivos

Se pretende la realización de una aplicación web que sirva para varios establecimientos y consiga reemplazar a parte del capital humano que se dedica a pedir nota. Para visualizarlo mejor un ejemplo parecido son las típicas pantallas de sitios que sirven comida rápida.

Como objetivos secundarios se pretende permitir ver todos los pedidos, configurar el sitio con datos del restaurante y otras customizaciones.

En el apartado de conclusiones y ampliaciones habrá más información sobre objetivos futuros.

## Cuestiones metodológicas

## Entorno de trabajo

La herramienta de control de versiones usadas es GitHub. El enlace al repositorio es el siguiente: <https://github.com/FerZeg/TFG>

Como herramienta principal del desarrollo de la aplicación se usará Visual Studio Code, este es el editor de texto más usado en la actualidad. Es propiedad de Microsoft pero es libre y multiplataforma.

Para manejar la base de datos uso MongoDBCompass, una interfaz super potente que permite tanto ejecutar comandos como modificar configuraciones del sistema.

Para planificar la aplicación se ha usado Notion, un software de gestión de proyectos y para tomar notas. Con esta aplicación se ha organizado tanto la temporalidad del proyecto como las diferentes tareas a realizar.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

# Descripción general del producto

## Límites del sistema

La aplicación es totalmente autónoma, es decir, no depende de otro proyecto web. Al ser una aplicación web puede ser accesible desde cualquier lado en cualquier momento.

## Funcionalidades básicas

La aplicación se define por las siguientes funcionalidades básicas:

* Gestión de tickets. (Ver, borrar)
* Gestión de usuarios y roles.
* Gestor de pedidos de comida.
* Gestión de datos internos.

## Usuarios

La aplicación para la posible futura ampliación se ha desarrollado de tal manera que pueda haber usuarios con más privilegios que otros. En el contexto de la aplicación hay 2 tipos de usuarios; los superadmin y los usuarios normales. Para la presentación de esta aplicación un usuario superadmin solo podría interactuar con la API de la plataforma, no se ha diseñado el frontend para soportarlo.

Dentro de los usuarios normales existen otros dos tipos de usuarios hasta la fecha. Los cocineros tienen acceso únicamente a las funcionalidades básicas del restaurante, como ver los pedidos o editar los platos. Por otro lado, los administradores tienen acceso completo al entorno de su restaurante, lo que les permite modificar los datos del establecimiento, gestionar los usuarios y mesas, y acceder a los tickets.

## Hosting

Hay 3 hostings utilizados, diferenciando la parte del front(interfaces de usuario), la del servidor y la de la base de datos.

* Para las diferentes interfaces se usa Cloudflare.
* Para el servidor se usará Northflank.
* Para la base de datos se usará Mongo Atlas.

Cloudflare y Northflank son de uso gratuito, siendo Cloudflare un *CDN (Content Delivery Network)* bastante popular y ampliamente usado por muchas aplicaciones, en contraposición a Northflank que es un servicio bastante nuevo y poco conocido.

El frontend al ser una **SPA (Single Page Application)** termina siendo una compilación de archivos estáticos que se pueden distribuir a través de la red mundial de Cloudflare. Esto reduce de forma considerable el tiempo de carga.

Northflank es una plataforma Cloud que trae muchas funcionalidades para facilitar el despliegue de la aplicación. Con un dockerfile sencillo defino el entorno de ejecución de mi servidor. (Referencia en el apartado de instalación).

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

Entre otras funcionalidades este servicio ofrece:

* Panel de control donde ver los logs de la aplicación
* Diferentes tipos de métricas
* Una consola de la instancia que se está ejecutando
* Sistema para exponer puertos diferentes según las necesidades
* Herramienta para configurar dominios customizados, certificados de SSL automáticos al conectar el domino.
* Ejecutar código cada cierto tiempo (Cron Jobs) o manualmente.
* Bases de datos.
* Usuarios y equipos.
* Notificaciones.

Todo esto con una capa gratuita bastante generosa para probar proyectos sin un tráfico excesivo.

Captura de pantalla de computadora

Descripción generada automáticamente

Ambas plataformas ofrecen un servicio de *CD (Continuos Deployment),* lo que facilita muchísimo el despliegue de la aplicación al ser automatizado al hacer un push a la rama Main del repositorio.

Captura de pantalla de un celular

Descripción generada automáticamente

# Planificación y presupuesto

## Planificación

Interfaz de usuario gráfica, Aplicación

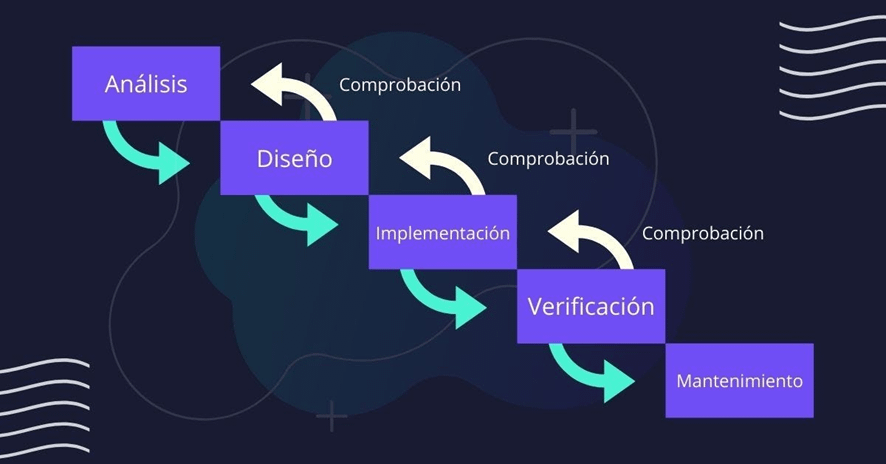
Descripción generada automáticamente

Esta es la planificación que he propuesto para la consecución de los objetivos de este trabajo. Es un resumen grosso modo de las diferentes fases troncales del desarrollo.

### Metodología

La metodología usada es la metodología en cascada retroactiva, es simplemente una modificación a la tan conocida metodología en cascada paso por paso pero con comprobaciones y pudiendo volver a una fase anterior en caso de ser necesario.

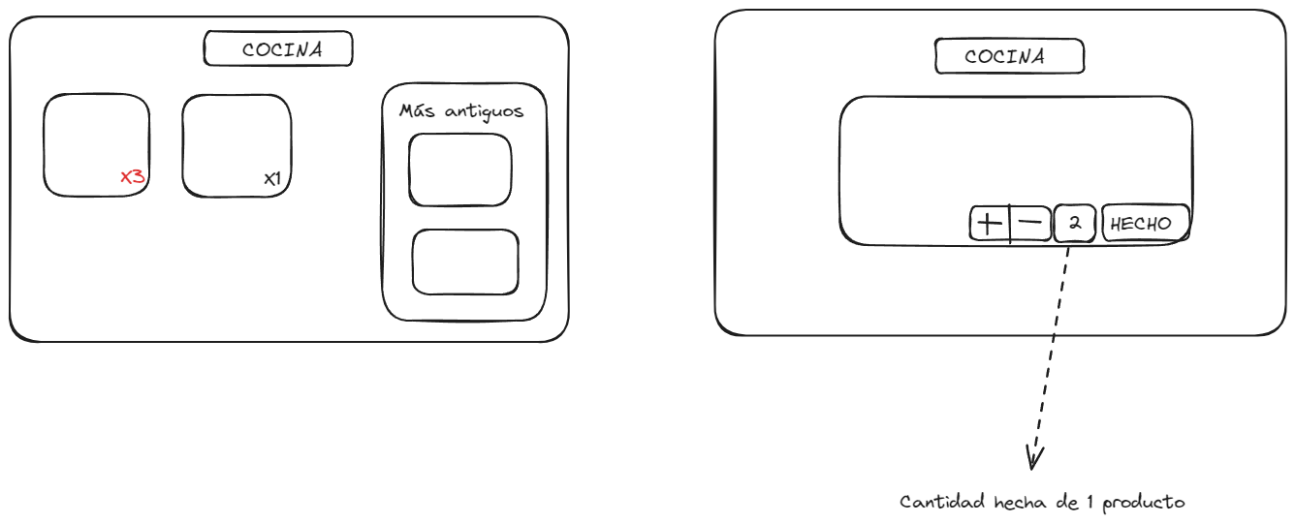
En realidad, ninguna metodología es usada al 100% ya que es inviable por todas las variables diferentes que ocurren en un desarrollo, pero esta es la que mejor se ajusta a la usada.

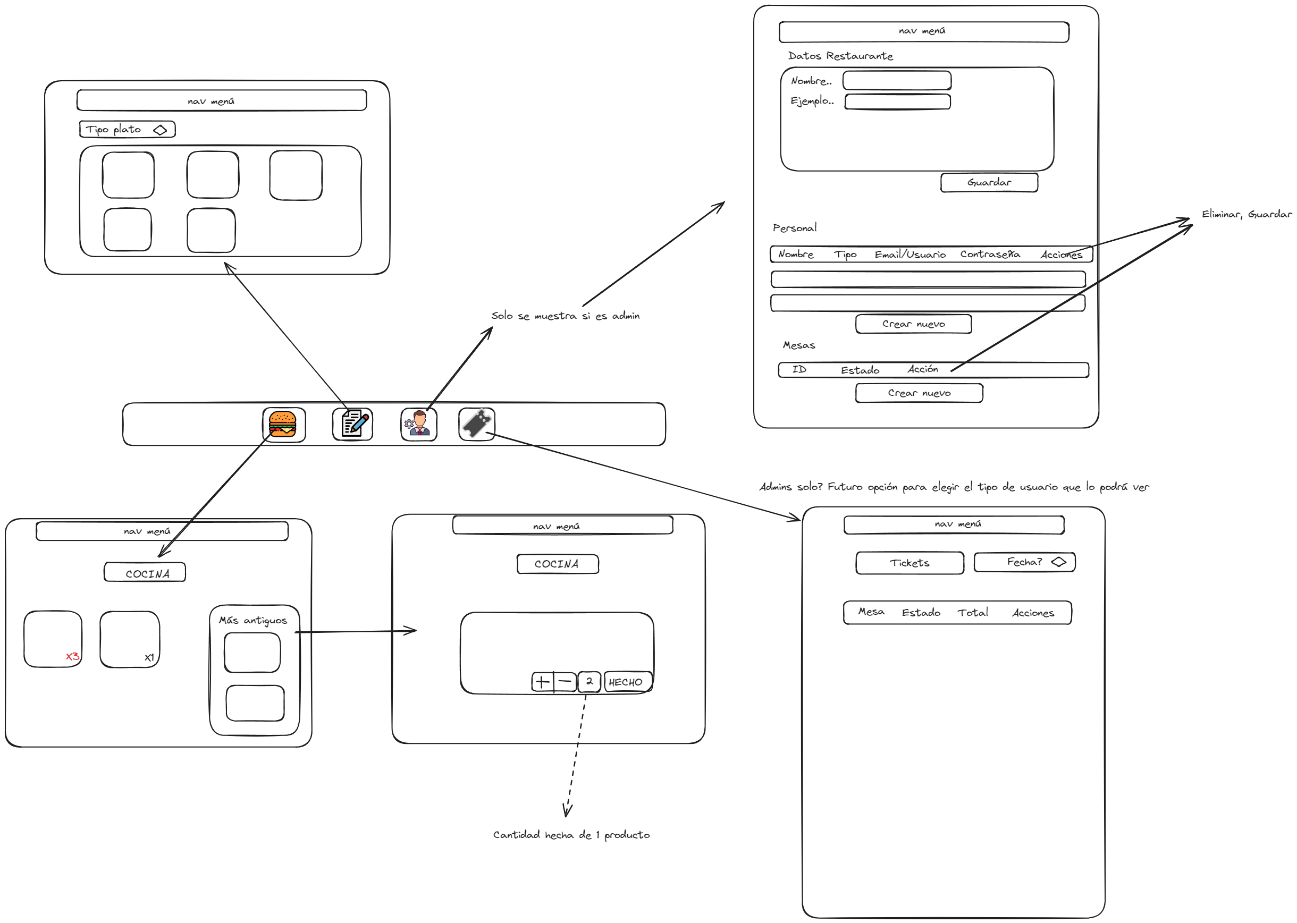


### Bocetos

Forma, Rectángulo

Descripción generada automáticamente





## Interfaces finales

### Admin

#### Página de error

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

#### Pantalla de Login

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

#### Editor de comidas

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

En la siguiente interfaz se pueden tanto visualizar con diferentes filtros (tipo, estado y búsqueda) como editar los productos existentes e incluso añadir nuevos.

#### Editor del restaurante

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Esta interfaz es exclusiva para los administradores del establecimiento, se pueden editar diversos datos del propio negocio como las cuentas de usuario relacionadas con este y las mesas.

#### Pantalla de tickets

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En esta interfaz se pueden visualizar los diferentes tickets que han sido creados en la aplicación. De momento la única funcionalidad disponible es borrar el ticket.

### Cliente

#### Página para iniciar

Texto, Carta

Descripción generada automáticamente

#### Interfaz principal

Interfaz de usuario gráfica

Descripción generada automáticamente

#### Historial de pedido

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Presupuesto

El groso del coste del desarrollo de la aplicación es el capital humano ya que las herramientas utilizadas son todas software libre, así como los alojamientos están usando capas gratuitas de diferentes proveedores.

El único coste extra a parte del propio trabajo ha sido el dominio cocinaenmarcha.com que tiene un coste de 10€/año.

El coste por horas de un programador Junior oscila entre 15€ - 20€ la hora en España. Al ser el primer proyecto el código base se computará a 15€ la hora.

El coste total por lo tanto será, 140 horas \* 15 € + 10 € = 2110€.

## Modelo de negocio

Esta aplicación no está diseñada para venderse a un cliente particular, si no para ser un modelo de suscripción a contratar por diferentes establecimientos.

El coste sería de alrededor de 150€/mes por establecimiento, y debido a la necesidad de ampliar los recursos de servidor se tendría un coste variable de 10€ por establecimiento.

Se tendrán en cuenta proposiciones de cambios o si hubiese una necesidad de una funcionalidad personalizada se presupuestaría.

Los mínimos como objetivo serían tener 5 clientes al mes y como cifra de éxito serían alrededor de 10. Estos darían un ingreso de 700€ o 1400€ respectivamente siendo el único trabajo un mantenimiento pasivo y ocasionalmente trabajos presupuestados.

# Documentación técnica

## Base de datos

La base de datos que he elegido para almacenar la información es MongoDB, una base de datos no relacional muy flexible. He utilizado este sistema para aprender más sobre él. En este tipo de aplicaciones pequeñas es un poco indiferente qué tipo de sistema elegir, si relacional o no relacional, debido a que el coste computacional es muy bajo. En una aplicación más grande yo habría optado por un sistema mixto o por el uso de PostgreSQL, que tiene muchas más funcionalidades que MySQL u otros derivados.

## Entidades

He definido las diferentes entidades existentes de la aplicación mediante un archivo de typescript para visualizar los “tipos” claramente.

Las capturas realizadas han sido producidas por “JSON Crack”, una extensión del Visual Studio Code para la visualización de archivos JSON.

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Rutas API

**Restaurantes**

## Estructura del código

### Librerías

Librerías principales del proyecto en el lado del servidor:

* Express: Framework de desarrollo para facilitar la elaboración del servidor, consta de un sistema avanzado de routing y librerías para parsear las cookies u otros.
* Mongoose: Interfaz para la base de datos Mongo, permite crear Schemas (Estructuras de objetos) y muchas funciones añadidas.
* Bcrypt: Librería para encriptar las contraseñas.
* Jsonwebtokens: Librería para generar tokens con una carga que se pueden codificar, decodificar y verificar mediante una contraseña. Es muy útil su uso para el sistema de autentificación y persistencia de la sesión.
* Formidable: Módulo utilizado para manejar formularios y archivos enviados mediante peticiones HTTP. Es especialmente útil para procesar formularios multipart/form-data, que son comunes en la carga de archivos a través de la web.

Librerías principales del proyecto en el lado del cliente:

* React: Framework de javascript creado por Facebook (actualmente META) que facilita mucho la creación de software mediante la encapsulación de código, tanto el lenguaje de marcado como las funcionalidades, para una mayor reutilización de este. Es actualmente el framework de javascript más usado en la actualidad.
* Vite: Herramienta de compilación de código ampliamente usada en la actualidad por su rapidez en tiempo de compilación o actuando como servidor de desarrollo para aplicar los cambios en el código en tiempo real cuando guardas un archivo.
* Sonner: Librería que se encarga de las notificaciones mediante cuadros de diferentes colores.
* React-router-dom: Librería hecha para react pero que no es parte de ella directamente, la uso para manejar las rutas de la aplicación y crear una SPA (Single Page Application).

## Códigos de interés

### Subida de imágenes

Con este código se maneja el acceso al contenedor que contiene los archivos estáticos. Usa el mismo api que AWS y son casi compatibles al 100% por lo que hay bastante documentación.

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

Este trozo de código ha sido un verdadero reto para realizar. Es el controlador que maneja la edición de los platos, pero en específico la parte para subir archivos a “R2 de Cloudflare”.

Texto

Descripción generada automáticamente

Usando la librería formidable, recojo los datos que llegan al servidor y se procesan para poder crear una tubería entre mi servidor y Cloudflare. Esto se llama comúnmente “stream de datos”, mi servidor no está guardando los archivos directamente, si no que redirige el flujo de los datos hacía el contenedor de Cloudflare.

Captura de pantalla de computadora

Descripción generada automáticamente

También verifica que la imagen sea de un formato predeterminado y un tamaño máximo. Después de subir la imagen se actualiza el plato en cuestión en la base de datos con la información recibida junto con la dirección de archivo generada por Cloudflare. En este caso se usan diferentes rutas para la versión de desarrollo y para la de producción.

### Errores customizados para respuestas

Es una practica muy buena crear errores propios para poder estandarizar las respuestas del servidor y poder manejarlas correctamente.

Texto

Descripción generada automáticamente

Pantalla de computadora

Descripción generada automáticamente con confianza media

Este es un “**middleware**” que captura los errores por defecto en express. Toda ruta que lance un error pasará por aquí automáticamente y responderá al cliente.

### Enrutamiento con React Router

Texto

Descripción generada automáticamente

Usando una jerarquía con formato JSON y con los diferentes componentes creados en la aplicación se crea un enrutamiento en la parte del cliente. Por cómo funciona react, todas las peticiones van dirigidas al index.html que carga el javascript. La libería se encarga de cargar las páginas o componentes necesarios para cada ruta, por lo que /edit y /admin serán totalmente diferentes.

Cuando hay zonas que se repiten o se necesitan implementar funcionalidades comunes, véase autentificación o elementos como la barra de navegación se usan layouts.

## Pruebas

# Manuales de usuario

## Instalación en local

1. Descargar los siguientes softwares para poder empezar:
   * Node.js (versión 20 o superior, 22 recomendada): <https://nodejs.org/en>
   * MongoDB: <https://www.mongodb.com/try/download/community>
   * Git: <https://git-scm.com/>
2. Descargar una copia del proyecto con el comando “git clone <https://github.com/FerZeg/TFG.git>”
3. Abrir 3 terminales, como recomendación abrir 3 pestañas diferentes en el Visual Studio Code ya que la parte del servidor y los 2 frontends están separados.Las carpetas son las siguientes en las que debes situarte:
   1. \TFG\Código\Backend
   2. \TFG\Código\Frontends\cocina
   3. \TFG\Código\Frontends\cliente
4. Instalar las dependencias con el manejador de paquetes que viene con Node.js por defecto NPM (Node Package Manager), “npm install”. Se pueden usar otros más eficientes como pnpm que debes instalar antes usando este mismo. Esto debe hacerse en cada una de las carpetas al ser proyectos diferentes.

## Instalación en la nube (versión recomendada)

### General

Para instalarlo en la nube se recomiendan los siguientes servicios que han sido usados en el desarrollo de esta; Cloudflare, Northflank y MongoDB Atlas.

Como primer paso casi indispensable, hacer una copia (**fork**) en git para poder aprovechar al máximo el “Continuos Deployment” y todas las facilidades que se desarrollan en torno al repositorio.

### Base de datos

MongoDB Atlas ofrece una capa gratuita, bastante conveniente para aplicaciones pequeñas o de pruebas.

1. Crear cuenta en <https://www.mongodb.com/cloud/atlas/register>. Puedes usar Google como vía rápida y fácil para el registro.
2. Directamente nos redirigirá a la página de creación de Clusters. Seleccionaremos la capa gratuita y daremos un nombre a nuestra BD.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

1. Esperar unos segundos a la creación de la BD.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. Una vez creada, te pedirá generar unas credenciales para poder administrar el sitio.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Una vez generadas las credenciales, importante apuntarlas, dar a conectar aplicación. Realmente el modo de conexión es muy parecida en las diferentes opciones pero usaremos la opción de driver de NodeJS para conseguir la cadena de conexión.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Remplazar <password> por tu contraseña y añadirlo a las variables de entorno.

Puedes crear varias instancias de MongoDB para usarlo también en local y no tener que instalar la base de datos en tu ordenador. Esta opción es más fácil ya que no tendrás que iniciarla cada vez que quieras usarla y además es mucho más conveniente ya que se necesita que la BD esté en modo réplica para el uso de transacciones. La instancia local no está en modo réplica por defecto por lo que es un poco más complicado configurarla.

### Frontends

1. Crear una cuenta en cloudflare <https://www.cloudflare.com/es-es/>
2. En la sección de workers y pages accionar el botón llamado “Create application” Interfaz de usuario gráfica, Texto, Aplicación

   Descripción generada automáticamente
3. En los apartados de creación habrá dos pestañas, Workers y Pages; seleccionar Pages.
4. Conectar con el repositorio de git donde almacenaste el fork.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Se pueden subir los archivos estáticos del frontend manualmente, pero es recomendable hacerlo de esta forma para que automáticamente actualice los cambios cuando haces un push a la rama principal.

1. Configurar el despliegue.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Esta sería la configuración para el panel de administrador de la aplicación. Vite construye la aplicación en la carpeta /dist y la ruta donde esta esta parte es la siguiente /Código/Frontends/cocina.

1. Repetir los mismos pasos con el cliente. La ruta es /Código/Frontends/cocina
2. Configurar el dominio en ambas partes desde el apartado de Custom domains.

Interfaz de usuario gráfica, Texto, Aplicación

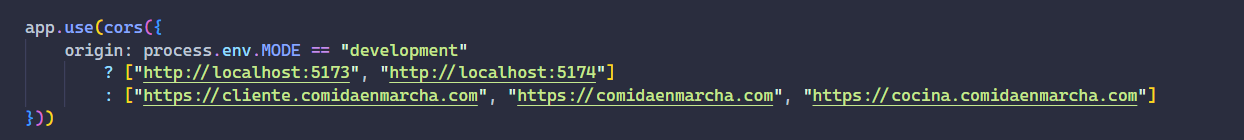
Descripción generada automáticamente

En el panel de control de la página podremos ver los diferentes entornos, en este caso, producción en main y desarrollo en development

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Es importante configurar la directiva **CORS** en el servidor dependiendo del dominio que hayas configurado



Configura los orígenes que tendrán permitidos el acceso. Esto es una restricción de seguridad de los navegadores para acceder a contenido externo a un origen.

### Servidor

1. Aprovechando que estamos en Cloudflare, se recomienda a continuación la creación de un bucket en la nube para el almacenamiento de archivos mediante R2 de cloudflare.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Conectar un dominio (recomendable un subdominio tipo media.comidanemarcha.com) para poder distribuir los recursos públicamente.

1. Crear API token de R2 en el menú principal. Esta será la llave de acceso para poder subir archivos desde el servidor

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Puedes configurarlo para que el token tenga acceso a todos los buckets o solo a algunos en específico. Es importante otorgarle el permiso de escribir objetos para subir las imágenes. Estas credenciales, importante guardarlas, serán usadas en el próximo apartado para instalar el servidor.

### Servidor

1. Crea una cuenta en <https://app.northflank.com/signup>, es tan fácil como usar tu cuenta de Google o Github directamente.
2. Elegir el plan developer, es totalmente gratis aunque tiene recursos limitados. La única pega de este servicio es que para evitar gente que abuse del plan gratuito te pedirá una tarjeta de crédito, aunque sea totalmente gratis.

Captura de pantalla de un celular

Descripción generada automáticamente

1. Tras esta pantalla nos redirigirá a crear un proyecto directamente, y si no es así, deberás navegar hasta los proyectos y crear uno nuevo.
2. Tras crear un proyecto debes añadir un servicio. Configura el repositorio conectando tu cuenta de github y selecciona en “Build options” la pestaña de “Dockerfile”. Tras hacer esto se visualizará el archivo que ya está creado en el proyecto.

Captura de pantalla de computadora

Descripción generada automáticamente

1. Tendrás que configurar las variables de entorno para tu entorno de producción

Pantalla de computadora con fondo negro

Descripción generada automáticamente con confianza media

1. En el servicio tienes que configurar el dominio personalizado también

Captura de pantalla de un celular

Descripción generada automáticamente

## Uso

# Conclusiones, correcciones y ampliaciones

## Conclusiones

El objetivo principal de este trabajo es presentar una aplicación para la consecución del grado superior de desarrollo de aplicaciones, sin embargo, he intentado igualar en importancia, el máximo aprendizaje posible.

## Ampliaciones

Se podrían introducir traducciones mediante la librería i18n para poder cambiar el idioma.

Poder editar los tickets a fondo, pedidos / precio.

Códigos de descuento al de pedir. (Sistema de fidelización).

Informes y reportes del estado del negocio.

* Suma de tickets, agrupación por mes/año/día o incluso horas.
* Reportes estadísticos por producto como el más o menos comprado.

Añadir descripciones a los productos.

Sistema de paginación en lista de registros (tickets).

## Correcciones

Validar mejor los datos, con alguna librería dedicada a ello.

Dar libertad para elegir las categorías de los diferentes artículos.

Mejorar la sintaxis del código unificándolo al inglés todo. (Ahora hay una pequeña mezcla en determinadas secciones)

## Relación de ficheros en formato digital

# Bibliografía

Modelar los datos con MongoDB:

* <https://www.youtube.com/watch?v=YsaOcUDUJKY>
* <https://www.youtube.com/watch?v=Hidk36H6hBY>

Información sobre metodología incremental:

* <https://www.reddit.com/user/BackNativos22/comments/taiqv2/describir_el_modelo_y_sus_principales/>

Algunos iconos (libre uso con atribución):

* <https://www.svgrepo.com/>

Librerías de NodeJS:

* <https://www.npmjs.com/>

# Glosario

* **CDN (Content Delivery Network):** Red de servidores interconectados que contienen copias locales de contenidos para distribuirlos rápidamente desde el nodo más cercano al usuario.
* **CD (Continuos Deployment):** Es una estrategia por la cual se publican automáticamente los cambios hechos a producción (producto final).
* **SPA (Single Page Application):** Aplicación web que interactúa con el usuario repintando la aplicación. Esto consigue crear una aplicación que no cambia de página (no se refresca). Se ahorran peticiones en el servidor y se gana velocidad de interacción al utilizarlas.
* **Fork:** Copia completa del repositorio de la que tienes control completo.
* **CORS:** mecanismo de seguridad que permite que los recursos de una página web sean solicitados desde otro dominio diferente al que originó la página.
* **Middleware:** Agente intermedio que se coloca entre diferentes componentes de una aplicación.