GESTIÓN DE RESTAURANTE

Bea Couchoud Ruiz

Memoria del Proyecto de DAW

IES Abastos
Curso 2019-2020
Grupo 7K
Valencia, 4 de junio de 2020
Tutor individual: Carlos Furones

Tabla de contenidos

1. Introducción

En el siguiente documento se pretende presentar la aplicación desarrollada, comentar el origen y la motivación que llevó a hacer realidad este proyecto, los objetivos del mismo, y detallar todas las características técnicas que permitan entender qué hace la aplicación y cómo lo hace.

1.1 Justificación

Actualmente, a causa de la COVID-19, muchos restaurantes se están viendo obligados a ofrecer sus servicios a través de internet mediante la venta online de sus productos. Detectada esta necesidad se ha buscado la creación de una herramienta fácil de usar que permita a cualquier restaurante aceptar y gestionar pedidos online y mantener una pequeña página web de manera sencilla y rápida.

Por tanto, este proyecto no busca innovar, si no crear una herramienta útil y funcional que permita tanto realizar como gestionar pedidos a domicilio de un determinado restaurante.

1.2 Objetivos

El objetivo principal del proyecto es crear una aplicación web en la cual, por un lado, los usuarios puedan obtener información sobre el restaurante y pedir comida a domicilio y, por otro lado, que el restaurante pueda gestionar rápidamente tanto estos pedidos como cualquier otro aspecto del restaurante (qué se muestra en la web a los usuarios, creación de menús nuevos de manera sencilla, etc).

Como objetivo secundario, se ha buscado realizar la aplicación utilizando tecnologías que no se han visto a lo largo del curso, aprovechando así la realización de este proyecto para obtener nuevos conocimientos.

1.3 Idea inicial

En un principio las necesidades que se buscan cubrir con esta aplicación web son, en función del rol del usuario:

Los posibles clientes (personas que entran en la web a obtener información) deben poder informarse de los menús disponibles, consultar la carta y conocer la ubicación del restaurante. Lo usuario registrados, además, deben poder realizar pedidos, consultar sus pedidos previos y administrar sus datos de usuario. Por otra parte, los empleados deben poder consultar rápidamente los pedidos, su estado y modificarlos (cancelar el pedido, actualizar su estado…). Por último, el administrador tiene que poder crear platos y menús nuevos, editar los ya existentes y gestionar los usuarios de la aplicación.

Todo esto debe poder controlarse mediante una interfaz gráfica intuitiva y sencilla de manejar.

2. Diseño

Este apartado está dedicado a la fase de diseño del proyecto. Basándose en las necesidades del sistema y la información de la etapa de análisis, se detallarán los componentes que forman el sistema, con el fin de llevar a cabo su desarrollo.

El objetivo de la etapa de diseño, es la definición de la arquitectura del sistema, tanto hardware como software, y la especificación detallada de sus distintos componentes, así como del entorno tecnológico necesario para dar soporte a su funcionamiento.

2.1 Análisis

Aquí se recoge toda la información relativa a la fase de análisis del sistema. Quedan definidos tanto las tecnologías y herramientas necesarias para llevar a cabo este proyecto como los requisitos de software necesarios para usar la aplicación y todo ello sirve de base para la siguiente etapa del proyecto.

2.1.1 Tecnologías y Herramientas

Las tecnologías se han elegido para realizar este proyecto son las siguientes:

Angular / TypeScript

Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript.

Angular se basa en clases tipo "Componentes", cuyas propiedades son las usadas para hacer el binding de los datos. En dichas clases tenemos propiedades (variables) y métodos (funciones a llamar).

Algunas de las ventajas que ofrece son:

- **Mejora de la productividad**: el planteamiento de arquitecturas estándar repercute directamente en la productividad del proyecto. Angular proporciona controladores, servicios y directivas para organizar el proyecto.

- **Generación de código**: Angular convierte tus plantillas en código altamente optimizado para las máquinas virtuales de JavaScript de hoy en día, ofreciéndote todas las ventajas del código escrito a mano con la productividad de un framework.

- **División del código**: Las aplicaciones de Angular se cargan rápidamente gracias al nuevo enrutador de componentes. Éste ofrece una división automática de códigos para que los usuarios sólo carguen el código necesario para procesar la vista que solicitan.

- **Reutilización de código**: El diseño de Angular adopta el estándar de los componentes web. Un componente en Angular es una porción de código que es posible reutilizar en otros proyectos de Angular sin apenas esfuerzo, en otras palabras, te permiten crear nuevas etiquetas HTML personalizadas, reutilizables y auto-contenidas, que luego puedes utilizar en otras páginas y aplicaciones web. Esto facilita el desarrollo de aplicaciones de manera mucho más ágil, pasando de un "costoso" MVC a un juego de puzles con nuestros componentes.

- **Angular CLI**: Las herramientas de línea de comandos permiten empezar a desarrollar rápidamente, añadir componentes y realizar test, así como previsualizar de forma instantánea la aplicación.

Aunque Angular no te obliga a usar TypeScript, se ha decidido usarlo para la realización del proyecto por varias razones, una de las primeras es la consistencia en la documentación. Por ejemplo, ES6 (o sea, ECMAScript 2015) ofrece varias formas diferentes de declarar un objeto, lo cual puede confundir a muchos. Con TypeScript esto no pasa, y toda la sintaxis y la manera de hacer las cosas en el código es la misma, lo que añade coherencia a la información y a la forma de leer el código. Además, la búsqueda de errores de runtime en JavaScript puede ser una tarea imposible. TypeScript proporciona detección temprana de errores (en tiempo de compilación), y tipado fuerte de clases, métodos, así como de objetos y APIs JavaScript ya existentes.

Además, TypeScript es un superset de ECMAScript 6, es decir, incluye todas las funcionalidades de ES6, y además incorpora una capa por encima con funcionalidades extra.

Esto, entre otras cosas, significa que puedes mezclar código TypeScript con código ES6 estándar sin problema, y el compilador de TypeScript seguirá pasando el código a ES5 (recordemos que tanto ES6 como TypeScript se transpilan a ES5 para que los navegadores actuales puedan ejecutar el código).

Express / NodeJS (JavaScript)

Express es un framework web transigente, escrito en JavaScript y alojado dentro del entorno de ejecución NodeJS.

Node (o más correctamente: *Node.js*) es un entorno que trabaja en tiempo de ejecución, de código abierto, multiplataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript. El entorno omite las APIs de JavaScript específicas del explorador web y añade soporte para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros. Algunas de sus ventajas son:

- El código está escrito en "simple JavaScript", lo que significa que se pierde menos tiempo ocupándose de las "conmutaciones de contexto" entre lenguajes cuando estás escribiendo tanto el código del explorador web como del servidor.

- JavaScript es un lenguaje de programación relativamente nuevo y se beneficia de los avances en diseño de lenguajes cuando se compara con otros lenguajes de servidor web tradicionales (Python, PHP, etc.).

- El gestor de paquetes de *Node* (NPM del inglés: Node Packet Manager) proporciona acceso a cientos o miles de paquetes reutilizables. Tiene además la mejor en su clase resolución de dependencias y puede usarse para automatizar la mayor parte de la cadena de herramientas de compilación.

Express es el framework web más popular de *Node*, y es la librería subyacente para un gran número de otros frameworks web de Node populares. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Con miles de métodos de programa de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla.

A pesar de que *Express* es en sí mismo bastante minimalista, los desarrolladores han creado paquetes de middleware compatibles para abordar casi cualquier problema de desarrollo web. Hay librerías para trabajar con cookies, sesiones, inicios de sesión de usuario, parámetros URL, datos POST, cabeceras de seguridad y *muchos* más.

En contraposición, esta flexibilidad es una espada de doble filo. Hay paquetes de middleware para abordar casi cualquier problema o requerimiento, pero deducir cuáles son los paquetes adecuados a usar algunas veces puede ser un auténtico reto. Tampoco hay una "forma correcta" de estructurar una aplicación, y muchos ejemplos que puedes encontrar en la Internet no son óptimos, o sólo muestran una pequeña parte de lo que necesitas hacer para desarrollar una aplicación web.

PhpMyAdmin (XAMPP) / MySQL

El modelo relacional es el modelo de datos que emplean la mayoría de las bases de datos actuales. MySQL es un sistema gestor de bases de datos que emplea el modelo relacional y que utiliza SQL como lenguaje de consulta.

- MySQL es el sistema de gestión de bases de datos relacional más extendido en la actualidad al estar basada en código abierto, permitiendo su uso gratuito.
- Es muy fácil de usar. Podemos empezar a usar la base de datos MySQL sabiendo unos pocos comandos.
- Pocos requerimientos y eficiencia de memoria. Tiene una baja fuga de memoria y necesita pocos recursos de CPU o RAM.
- Es compatible con Linux y Windows.

Por otra parte, phpMyAdmin, un cliente de MySQL muy popular, es una herramienta gratuita, que permite de una manera muy completa acceder a todas las funciones de la base de datos MySQL, mediante una interfaz web muy intuitiva. Esta aplicación nos permitirá realizar las operaciones básicas en base de datos MySQL, como son: crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos,

ejecutar sentencias SQL, administrar claves de campos, administrar privilegios y exportar datos en varios formatos. La función de exportar datos se emplea muchas veces para realizar backups de la base de datos y poder restaurar esta copia de seguridad en el futuro a través de phpMyAdmin mediante la opción “importar”. Para utilizar phpMyAdmin instalaremos XAMPP.

XAMPP es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl.

Una de las ventajas de usar XAMPP es que su instalación es de lo más sencilla, basta descargarlo, extraerlo y comenzar a usarlo. Las configuraciones son mínimas o inexistentes, lo cual nos ahorra bastante tiempo.

Bootstrap

Bootstrap es uno de los frameworks CSS más populares utilizado para crear sitios web y aplicaciones responsivas y fiables. Se trata de un paquete compuesto por una estructura de archivos y carpetas de código estandarizado (HTML, CSS, documentos JS, etc.) que se pueden utilizar para apoyar el desarrollo de sitios web, como base para comenzar a construir un sitio.

Bootstrap tiene una gran comunidad de usuarios (lo que facilita cualquier tipo de consulta), más plugins desarrollados y una buena documentación. Sin duda, este framework es la mejor elección a día de hoy para comenzar tu página web.

Visual Studio Code

Visual Studio Code es un potente editor de código fuente multiplataforma Windows, Mac, Linux con reconocimiento de sintaxis de código y coloreado de una multitud de lenguajes e integración con Git.

Visual Studio Code es un editor ligero no un IDE completo, es decir no es un sustituto del Visual Studio, sino que más bien es una herramienta básica para cubrir las necesidades de edición de código simple. Lo más importante de este editor es su gratuidad y que es multiplataforma.

Git

Git es el software de control de versiones más utilizado. Entre otras muchas cosas, Git te permite subir y actualizar el código de tu página web a la nube de GitHub. De esta forma siempre puedes disponer de él cuando lo necesites. Pero, además puedes:

- Conocer quién ha sido el responsable de una determinada modificación y cuándo la ha realizado.
- Realizar comparaciones entre versiones de una aplicación.
- Observar la evolución del proyecto con el paso del tiempo.

- Contar con una copia del código fuente para poder volver atrás ante cualquier imprevisto en la página web.
- Estar al tanto de los cambios en el código fuente.
- Tener una copia de seguridad del proyecto al completo.
- Disponer de un historial en el que se detallen las modificaciones realizadas en el código del sitio web.
- Git es software libre y open source

2.2 Planificación

En esta fase, el objetivo prioritario es identificar cada una de las necesidades que deben ser satisfechas al desarrollar el sistema final. Esto se hará efectivo mediante la obtención de requisitos que deben estar presentes en el sistema de información final, con la intención de dar solución a las necesidades identificadas con anterioridad. Al finalizar esta primera etapa en el desarrollo del producto se consiguen los requisitos, tanto funcionales como no funcionales, y su modelado mediante los casos de uso, desarrollados estos últimos en diagramas. Todo esto facilita la comprensión del sistema haciendo que pueda ser diseñado cumpliendo con todos los objetivos y cubriendo las necesidades de forma satisfactoria.

2.2.1 Requisitos

En esta sección la finalidad es detallar de una manera clara y concisa cada uno de los objetivos que presenta el proyecto, así como todas sus características.

Es de suma importancia para que el desarrollo del proyecto concluya con éxito que antes de empezarlo se tenga una completa y plena compresión de los requisitos del software.

Para organizar de la forma más clara posible todos los requisitos se utilizará esta tabla. El formato del identificador será RF o RNF (requisito funcional o requisito no funcional respectivamente), seguido de un número (un índice de dos dígitos). Estos identificadores podrán ser después usados para organizar tareas con mayor facilidad.

Por ejemplo: “RF - 01”.

IDENTIFICADOR
NOMBRE:
PRIORIDAD:
DESCRIPCIÓN:

2.2.1.1 Requisitos funcionales

RF - 01
NOMBRE: Creación y edición de platos
PRIORIDAD: Muy alta
DESCRIPCIÓN: El administrador debe poder crear y editar platos a través de un formulario sencillo. Los platos se crearán habilitados por defecto y una vez creados se podrán habilitar y deshabilitar.

RF - 02
NOMBRE: Creación y edición de menús
PRIORIDAD: Muy alta
DESCRIPCIÓN: El administrador debe poder crear y editar menús a través de un formulario sencillo. Los menús se crearán habilitados por defecto y una vez creados se podrán habilitar y deshabilitar.

RF - 03
NOMBRE: Creación y edición de usuarios
PRIORIDAD: Muy alta
DESCRIPCIÓN: Los usuarios deben poder registrarse en la página y editar sus datos una vez se ha creado su cuenta. Además, el administrador debe poder modificar el rol del usuario y habilitarlo o deshabilitarlo.

RF - 04
NOMBRE: Creación y edición de pedidos
PRIORIDAD: Muy alta
DESCRIPCIÓN: Los usuarios deben poder realizar pedidos fácilmente y cancelarlos en caso necesario. A su vez, los empleados deben poder ver los pedidos y su estado a simple vista, rechazarlos y modificar su estado, tanto para su propia información como para mantener al usuario al corriente del estado de su pedido.

RF - 05
NOMBRE: Creación y edición de pedidos
PRIORIDAD: Muy alta
DESCRIPCIÓN: Los usuarios deben poder realizar pedidos fácilmente y cancelarlos en caso necesario. A su vez, los empleados deben poder ver los pedidos y su estado a simple vista, rechazarlos y modificar su estado, tanto para su propia información como para mantener al usuario al corriente del estado de su pedido.

RF - 06
NOMBRE: Inicio y cierre de sesión
PRIORIDAD: Muy alta
DESCRIPCIÓN: Los usuarios deben poder iniciar/cerrar sesión en todo momento.

RF - 07
NOMBRE: Contenidos de la web
PRIORIDAD: Media/Baja
DESCRIPCIÓN: El administrador puede modificar los textos que aparecen en la web que ven los clientes.

2.2.1.2 Requisitos no funcionales

RNF - 01
NOMBRE: Tiempo
PRIORIDAD: Alta
DESCRIPCIÓN: El tiempo de desarrollo debe ser de 25 horas.

RNF - 02
NOMBRE: Seguridad
PRIORIDAD: Alta
DESCRIPCIÓN: Control de sesiones, autenticación de nivel de usuario y encriptación de contraseñas.

RNF - 03
NOMBRE: Interfaz de usuario
PRIORIDAD: Media/Alta
DESCRIPCIÓN: La interfaz debe ser simple y fácil de manejar. Los colores se usarán de forma estratégica para permitir identificar acciones/estados de diferentes elementos de la página a simple vista. Es importante dar feedback al usuario del resultado de su interacción con la página.

RNF - 04
NOMBRE: Validación de formularios
PRIORIDAD: Media
DESCRIPCIÓN: Los formularios tienen que validarse en el HTML y en el controlador. Además deberán sanitizarse los datos en el servidor antes de realizar las consultas a base de datos. La base de datos no contendrá información extremadamente sensible y el riesgo de ataque al servidor es muy baja pero es preferible mantener esta medida de seguridad.

2.2.2 Gestión del proyecto

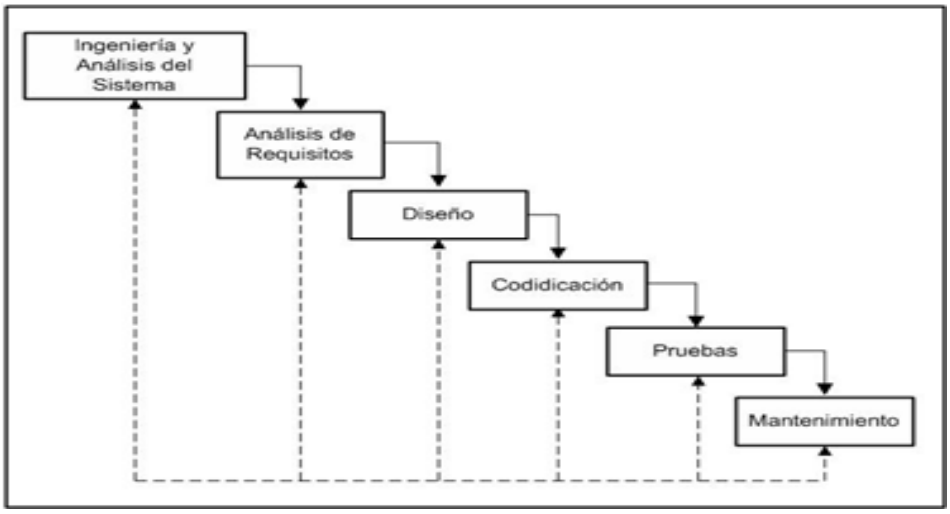
Para gestionar de manera correcta el proyecto se necesita elegir el ciclo de vida más apropiado para el proyecto en función de nuestras necesidades. Otra parte muy importante de la gestión del proyecto es la estimación de tiempo de las diferentes tareas, para lo cual se realiza un diagrama de Gantt.

2.2.2.1 Ciclo de vida del proyecto

El proceso para el desarrollo de software, también denominado ciclo de vida del desarrollo de software, es una estructura aplicada al desarrollo de un producto de software. Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque distinto para diferentes actividades que tienen lugar durante el proceso.

El paradigma elegido es el desarrollo en cascada con retroalimentación. Este modelo admite la posibilidad de hacer iteraciones, es decir, durante las modificaciones que se hacen en el mantenimiento se puede ver por ejemplo la necesidad de cambiar algo en el diseño, lo cual significa que se harán los cambios necesarios en la codificación y se tendrán que realizar de nuevo las pruebas, es decir, si se tiene que volver a una de las etapas anteriores al mantenimiento hay que recorrer de nuevo el resto de las etapas.

Después de cada etapa se realiza una revisión para comprobar si se puede pasar a la siguiente.



Los motivos principales para su elección son que la planificación es muy sencilla, la calidad del producto resultante es alta, permite retroceder en cualquier momento si fuera necesario y es el más sencillo de utilizar cuando el desarrollador tiene poca experiencia.

Una vez elegido el modelo de ciclo de vida del proyecto se procedió a planificar cada una de sus fases. El objetivo es la realización de una estimación del tiempo que va a ser utilizado para la realización completa del proyecto.

2.2.2.2 Diagrama de Gantt

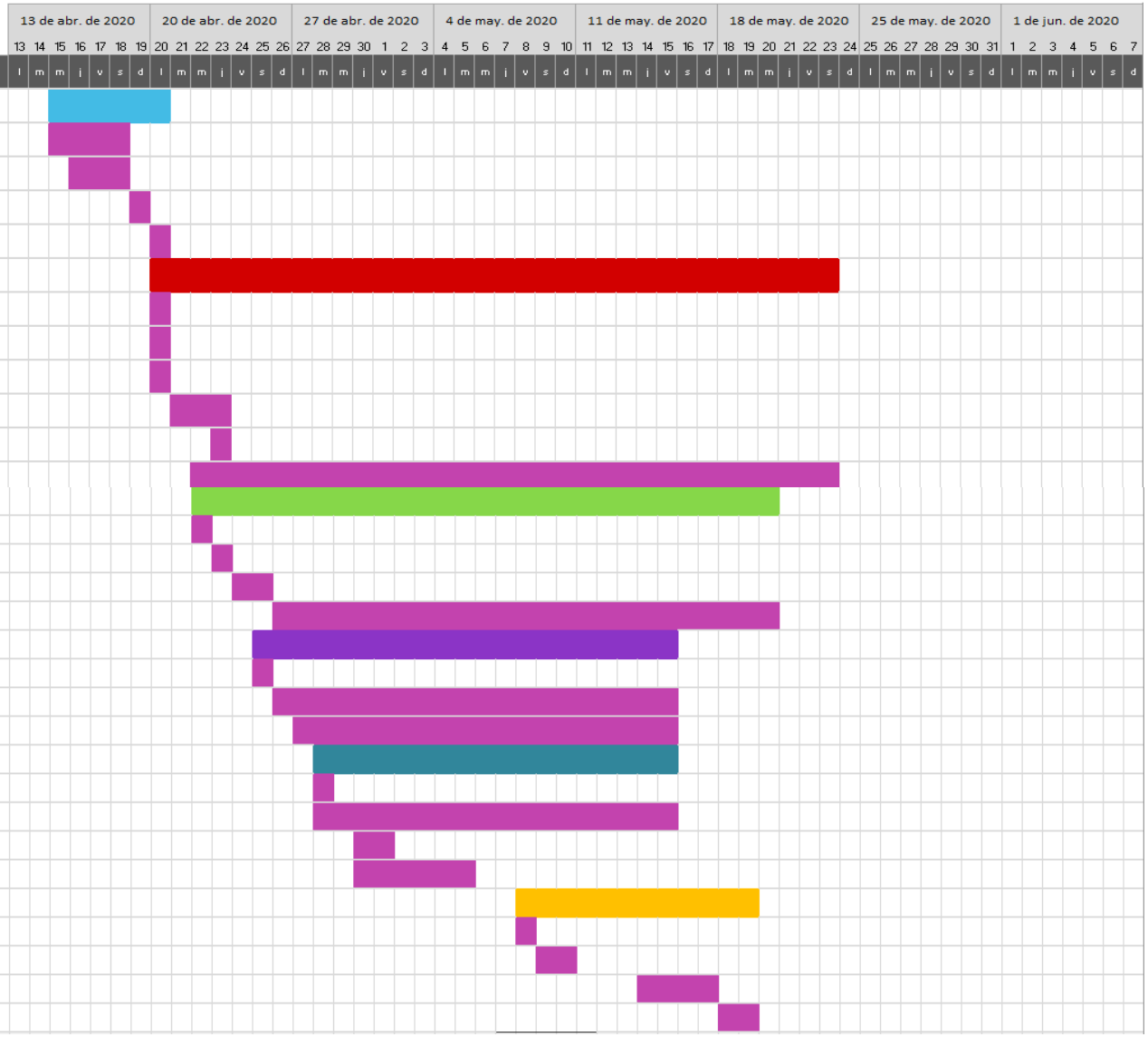
En el diagrama de Gantt se puede observar la duración de cada uno de los plazos de realización de tareas, separados por las fases anteriormente descritas. Se puede observar que algunas de las tareas se solapan puesto que pueden ser desarrolladas de forma simultánea, siendo independientes unas con otras, mientras que otras necesitan que las anteriores hayan sido finalizadas para poder iniciarse. La estimación realizada plantea una duración de, aproximadamente, dos meses.

GESTIÓN DE RESTAURANTE

Proyecto Final DAW, IES Abastos
Bea Couchoud Ruiz

mi, 15/4/2020
1

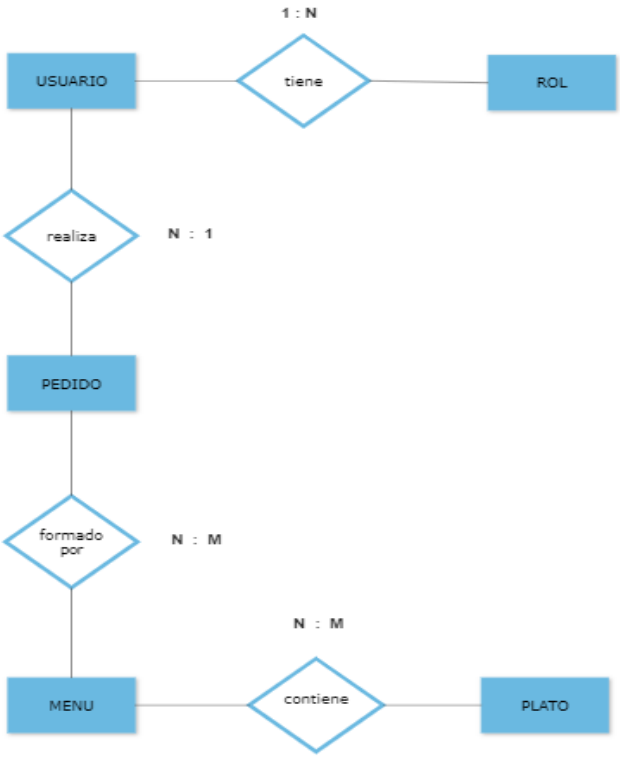
TAREA	HORAS	INICIO	FIN
IDEA	12h		
Planteamiento y desarrollo de la idea	6h	15-4-20	18-4-20
Elección de herramientas a utiilizar	4h	16-4-20	18-4-20
Creación del proyecto	1,5h	19-4-20	19-4-20
Creación del repositorio en GitHub	0,5h	20-4-20	20-4-20
PLANIFICACIÓN	14,5h		
Planteamiento del Modelo E-R	2h	20-4-20	20-4-20
Normalización de los datos	1h	20-4-20	20-4-20
Creación de la BD	1,5h	20-4-20	20-4-20
Definición de requisitos	4,5h	21-4-20	23-4-20
Casos de uso	1,5h	23-4-20	23-4-20
Diagrama de Gantt	4h	22-4-20	23-5-20
DISEÑO DE LA INTERFAZ	16,5h		
Realización de mockups	6h	22-4-20	22-4-20
Elección de colores y estilos	2,5h	23-4-20	23-4-20
Creacion de componentes html	4h	24-4-20	25-4-20
Estilos CSS	4h	26-4-20	20-5-20
CLIENTE	14h		
Creación de rutas a páginas principal	0,5h	25-4-20	25-4-20
Creación de componentes	9h	26-4-20	15-5-20
Creación de servicios	4,5h	27-4-20	15-5-20
SERVIDOR	10h		
Creación de la API y conexión a BD	0,5h	28-4-20	28-4-20
Creación de rutas y métodos	7h	28-4-20	15-5-20
Instalación de middlewares	0,5h	30-4-20	1-5-20
Control de sesiones y autenticación	2h	30-4-20	5-5-20
MEMORIA	22h		
Introducción	2h	8-5-20	8-5-20
Diseño	10h	9-5-20	10-5-20
Desarrollo	8h	14-5-20	17-5-20
Conclusiones	2h	18-5-20	19-5-20



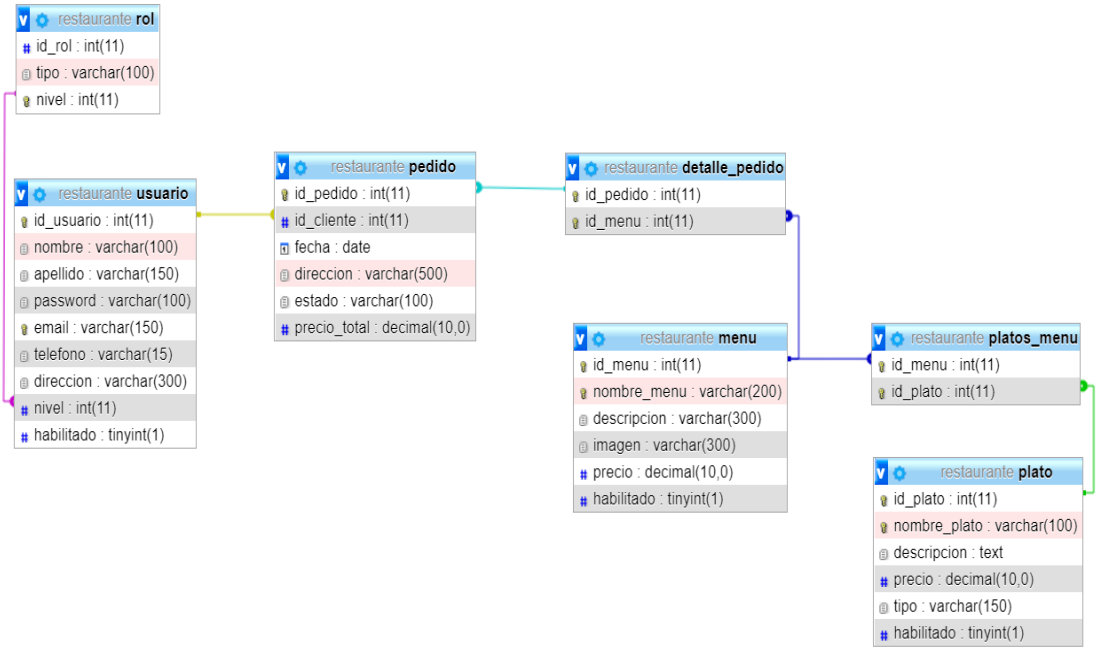
2.3 Modelo Entidad-Relación y Base de Datos

Mediante este diagrama entidad-relación se puede representar todas las entidades que presentan relevancia del sistema, junto con sus propiedades e interrelaciones.

Un primer diagrama ER muy primitivo quedaría así:



Tras añadir los atributos correspondientes y normalizar la base de datos para evitar la redundancia de información y facilitar su gestión posterior, el resultado final sería el que se muestra a continuación:

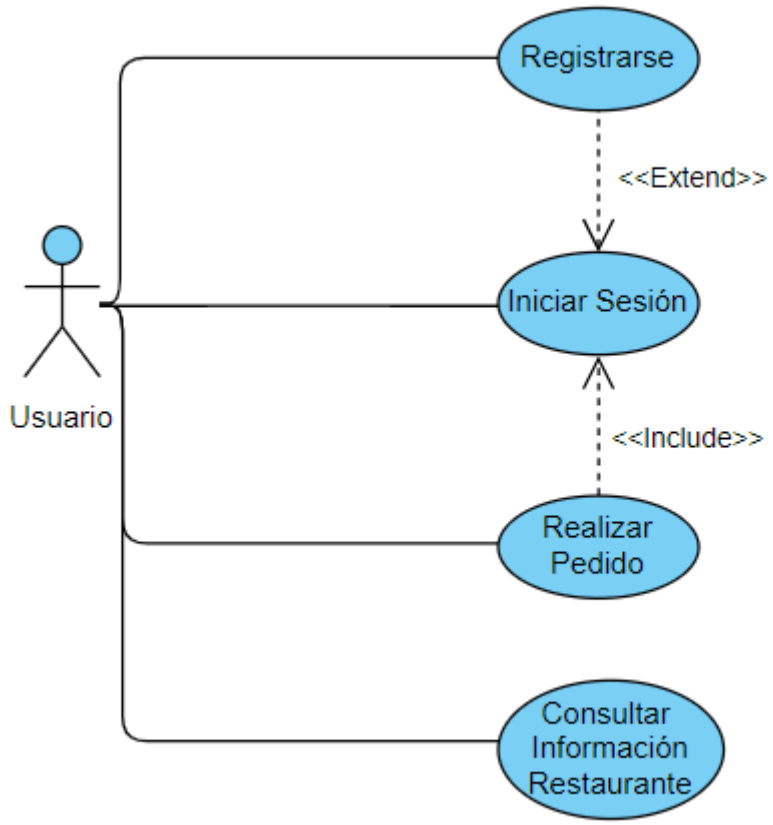


2.4 Casos de Uso

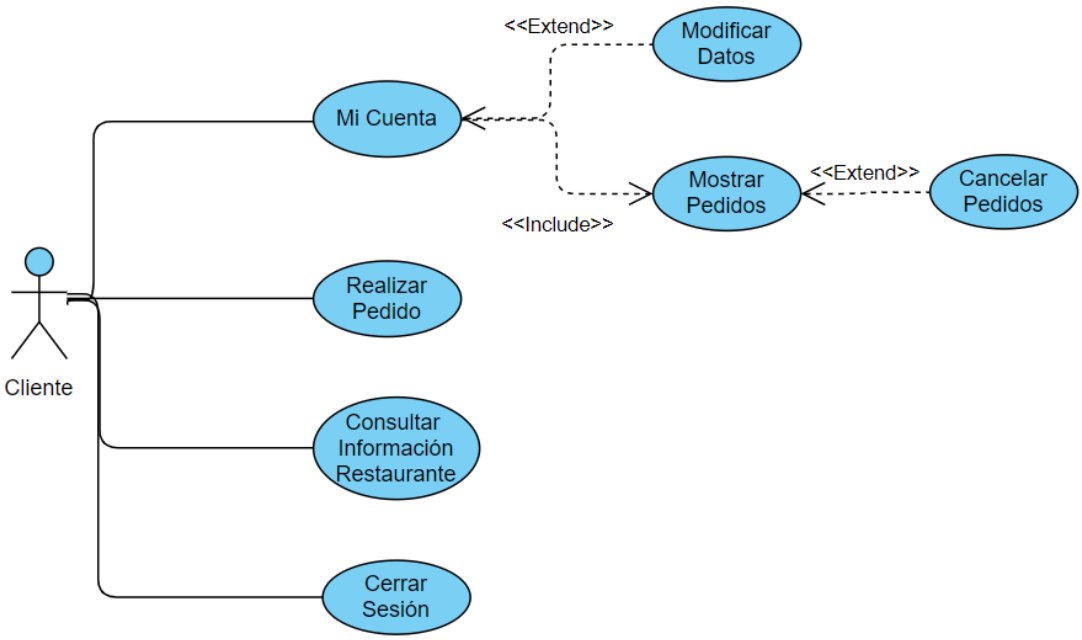
A continuación, se detallarán los casos de uso, describiendo todas las posibles interacciones entre el usuario del sistema y la aplicación con el fin de alcanzar un objetivo. De esta manera se proporcionan diversos escenarios que indican la sucesión de respuestas surgidas ante las acciones iniciadas por los usuarios sobre el sistema. Esta especificación de casos de uso facilita la comprensión del sistema y expresa la intención con la que el usuario interactuará con la aplicación, profundizando así en dichas necesidades, y que, complementándose con los requisitos anteriormente descritos, establece una firme base a la hora de comenzar con la fase de diseño.

Se ha creado un caso de uso para cada rol de usuario de la aplicación (usuario no registrado, cliente, empleado y administrador).

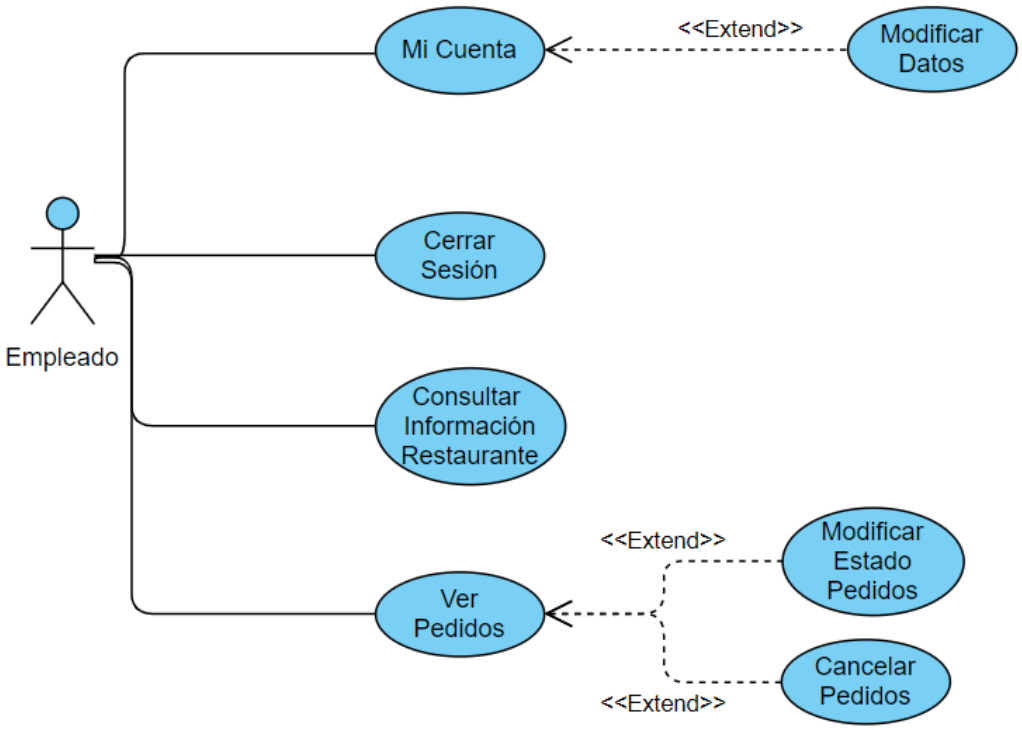
Caso de uso desde la perspectiva de un usuario no registrado



Caso de uso desde la perspectiva de un cliente o usuario registrado



Caso de uso desde la perspectiva de un empleado



Caso de uso desde la perspectiva de un administrador