Grado en Ingeniería del Software

Curso 2014/2015

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE RESTAURANTES

Melanie Ramos Otero

Tutor: Francisco Javier Gil Rubio

Índice general

1

Introducción

Índice

1.1 Resumen

Actualmente existen una gran cantidad de restaurantes diferentes en todo el mundo, tanto por su tipo de comida, especial de cada país o ciudad, como por la forma en que ofrecen sus servicios o por la temática que presentan al público.

Cada día, millones de personas buscan el restaurante perfecto donde disfrutar de un desayuno, comida o cena, solos o en compañía de otras personas.

A veces no es nada fácil encontrar sitios nuevos a los que ir, alejándonos un poco de la rutina del día a día. O simplemente queremos viajar a otra ciudad o país y no sabemos dónde podemos ir a comer, o dónde encontrar la comida típica.

Por otra parte, a veces los propios restaurantes encuentran un poco difícil la tarea de darse a conocer o promocionar su comida.

De este planteamiento surge la idea de realizar una aplicación web en la que los restaurantes puedan crear una cuenta y personalizarla para que ésta sea fiel a la imagen del establecimiento, y no una página más entre miles de restaurantes.

Además, esta aplicación será el medio perfecto para que las personas puedan buscar ese lugar al que quieren ir a disfrutar de su comida, de una manera rápida y eficaz y todo desde una misma página web.

1.2 Abstract

Currently, there are a huge amount of different restaurants around the world, and they are different for their type of food, which is specific of each country or city, for the way they offer their services or for the thematic they present to the costumers.

Every day, millions of people search for the perfect restaurant where they can enjoy their breakfast, lunch or dinner, on their own or in company of others.

Sometimes it is not easy to find new places to go, getting away from the routine of the day-to-day. Or simply we want to travel to another country or city and we don't know where we can go out for a meal, or where we can find the typical food.

On the other hand, sometimes the restaurants find it hard to make themselves known or to promote their food.

From this proposal appears the idea of making a web application where the restaurants could create an account and customize it so it is faithful to the image of the establishment and it is not just one more web page among miles of restaurants.

In addition, this application will be the perfect way for people to search that place where they want to go to enjoy their meal, in a fast and efficient way and everything through the same web page.

1.3 Objetivos

Con este proyecto se pretende ahondar en el desarrollo web para aprender más sobre el protocolo http y obtener un conocimiento más avanzado sobre tecnologías como html, css, javascript y php.

Además de profundizar en las tecnologías mencionadas anteriormente, uno de los objetivos principales de este proyecto es aprender a manejar frameworks que ayudan al desarrollo de aplicaciones webs y permiten construir una estructura de directorios y un código más organizado y mantenible de cara al futuro.

Por último, mediante la realización de esta web, se quiere aprender a aplicar buenas prácticas utilizando sistemas de control de versiones y metodologías ágiles, de tal manera que se lleve una correcta organización y planificación del proyecto desde el principio.

2

Especificación de requisitos

Índice

2.1 Introducción

2.1.1 Propósito

Con esta especificación se pretende definir los requisitos y restricciones de un sistema digital para realizar reservas en restaurantes, así como las funciones de los actores que interactúan con el mismo. Este documento servirá como base para la aplicación web que se va a desarrollar para este Proyecto Final de Grado.

Esta especificación de requisitos está dirigida a la persona responsable de diseñar y desarrollar la aplicación web, que en este caso es la misma persona que escribe estos requisitos.

2.1.2 Ámbito del sistema

2.1.2.1 Identificación del sistema con un nombre

El nombre que se ha elegido para el sistema ha sido *Tu Mesa*, ya que será un sistema para que los usuarios reserven mesas en restaurantes.

2.1.2.2 Qué hace y qué no hace el sistema

- Este sistema es una herramienta digital, es decir, una página o aplicación web, que sirve de intermediaria entre restaurantes y clientes para gestionar reservas.

- En este sistema los clientes pueden consultar información sobre los restaurantes, dejar valoraciones y comentarios y realizar reservas.

- Los restaurantes pueden crear una cuenta e introducir datos referentes a su establecimiento.

- Este sistema no es una herramienta de gestión de mercaderías o presupuestos para restaurantes.

- Esta herramienta no permite crear cuentas de usuarios, sólo los restaurantes podrán crear cuentas como empresa para promocionar su establecimiento y llegar más rápidamente a sus clientes.

2.1.2.3 Beneficios, objetivos y metas

Con este sistema los restaurantes podrán crear páginas personalizadas de acuerdo a su marca y sello personal. De esta manera, cada página de un restaurante será diferente de las demás, creando una identidad propia para poder venderse mejor.

El objetivo de este sistema es diferenciar a cada restaurante por su estilo para que cada uno destaque por sus colores y logos personales. Además, se pretende ofrecer una herramienta para que los restaurantes puedan llegar a un público más amplio de una manera sencilla tanto para ellos como para los clientes.

2.1.3 Definiciones, acrónimos, y abreviaturas

Término	Significado
App	**App**lication.
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers.
HTML	**H**yper**T**ext **M**arkup **L**anguage.
CSS	**C**ascading **S**tyle **S**heets.
AJAX	**A**synchronous **J**avaScript **A**nd **X**ml.
PHP	**P**ersonal **H**ome **P**age.
XAMPP	**X** (para cualquier SO) **A**pache, **M**ySQL, **P**HP, **P**erl.
SO	**S**istema **O**perativo.
SQL	**S**tructured **Q**uery **L**anguage.

JSON	**J**ava**S**cript **O**bject **N**otation.
ORM	**O**bject-**R**elational **M**apping.
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol.
URL	**U**niform **R**esource **L**ocator.
UML	**U**nified **M**odeling **L**anguage.
POST	Uno de los métodos de consulta del protocolo HTTP. Normalmente se utiliza para actualizar datos sensibles o enviar formularios ya que los datos se envían ocultos en el mensaje de la consulta.
GET	Otro método de consulta HTTP. Se utiliza para realizar consultas con datos menos sensibles ya que éstos se envían junto con la URL y son visibles y de fácil acceso para los usuarios.
Framework	Conjunto de herramientas y prácticas estandarizadas que ayudan a resolver un problema o realizar una tarea concreta en un ámbito determinado.
Bcrypt	Función para encriptar contraseñas diseñada por Niels Provos y David Mazières, basada en el cifrador Blowfish.
Servidor	Aplicación que se encarga de atender las peticiones de los clientes y realizar operaciones con la base de datos para poder devolver los datos requeridos.
Clientes	Todas aquellas personas que utilicen el Sistema final, tanto restaurantes como personas que reservan.
IDE	**I**ntegrated **D**evelopment **E**nvironment

2.1.4 Referencias

Se han consultado las siguientes referencias para escribir esta especificación de requisitos:

- Estándar IEEE 830-1998 Recommended Practice for Software Requirements Specifications.
  <http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf>

- Especificación de Requisitos según el estándar de IEEE 830. IEEE Std. 830-1998.
  <https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf>

- Rumbaugh, J., Jacobson, I., Booch, G. (2000): El lenguaje unificado de modelado. Manual de referencia.
  <https://ingenieriasoftware2011.files.wordpress.com/2011/07/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf>

2.1.5 Visión general del documento

Este documento sigue la estructura de especificación de requisitos según el estándar IEEE 830-1998. Se divide en dos apartados:

2. Descripción general

En esta sección se describe el contexto que rodea a los requisitos que se explicarán en el siguiente apartado. Este apartado se divide en los siguientes: *perspectiva del producto*, *funciones del producto*, *características de los usuarios*, *restricciones* y *requisitos futuros*.

3. Requisitos específicos

En este apartado se describen con detalle todos los requisitos del sistema. Esta parte está compuesta por los apartados siguientes: *funciones* y *restricciones de diseño*.

2.2 Descripción general

2.2.1 Perspectiva del producto

Este sistema constituye una unidad independiente y no forma parte de un sistema mayor.

2.2.1.1 Interfaz de sistema

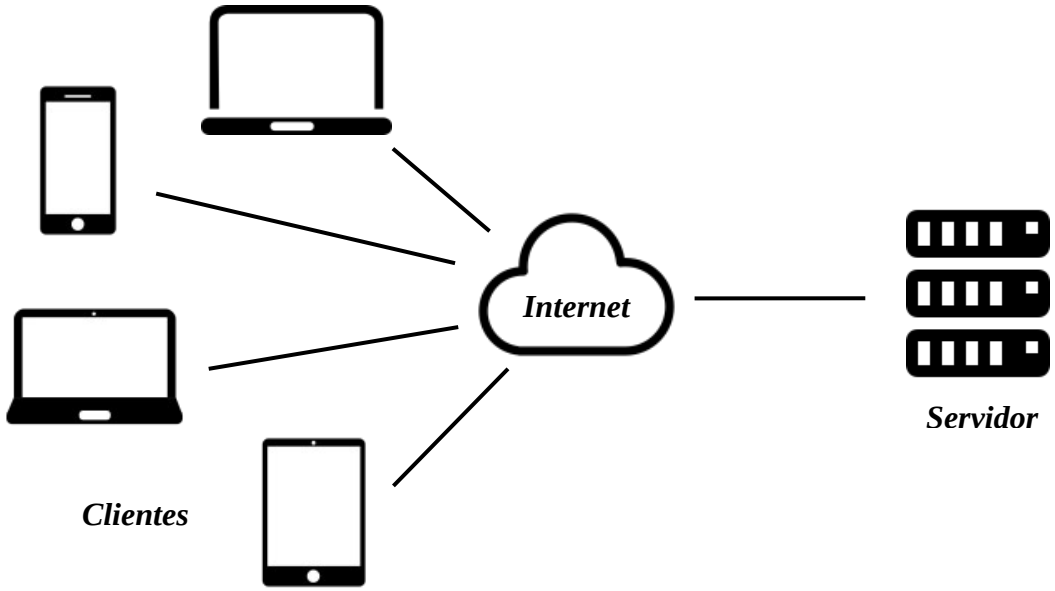
El sistema debe interactuar correctamente con los navegadores web más utilizados (Chrome, Mozilla Firefox, Internet Explorer, Opera, Safari) así como con navegadores de dispositivos móviles como smartphones y tablets.

2.2.1.2 Interfaz de usuario

- La aplicación se deberá poder utilizar en cualquier navegador.
- La aplicación se deberá poder adaptar a cualquier tamaño de pantalla tanto de monitores de ordenadores de sobremesa, portátiles, smartphones y tablets.

2.2.1.3 Interfaz hardware

El sistema consta de un servidor en el que se almacenarán todos los datos relativos a los restaurantes y éste dará servicio a varios clientes que podrán acceder desde un navegador web mediante diferentes dispositivos.



2.2.1.4 Interfaz software

El sistema hará uso de herramientas externas como los distintos navegadores web de cada cliente (Chrome, Mozilla Firefox, Internet Explorer, Opera, Safari) y un servicio online que proporciona herramientas para enviar correos electrónicos llamado Mailgun.

2.2.2 Funciones del producto

- Crear una cuenta de restaurante.
- Editar información sobre el restaurante.
- Consultar información referente a los restaurantes.
- Valorar y comentar en la página de un restaurante.
- Gestionar las valoraciones y comentarios del restaurante.
- Reservar en un restaurante.
- Gestionar las reservas del restaurante.

2.2.3 Características de los usuarios

Existen dos tipos de usuario en este sistema, el usuario restaurante y el usuario cliente.

2.2.3.1 Usuarios restaurantes

- Un usuario restaurante debe ser dueño de un restaurante y disponer de un dispositivo con conexión a internet y un navegador para poder acceder al sistema.
- Este usuario debe crear una cuenta para poder utilizar el sistema.
- No se requiere un nivel educativo específico, experiencia o conocimientos avanzados en la utilización de herramientas de este tipo para utilizar el sistema.

2.2.3.2 Usuarios cliente

- Un usuario cliente puede ser cualquier persona que disponga de un dispositivo con conexión a internet y un navegador web para poder acceder a este sistema.
- Un usuario restaurante también puede ser usuario cliente y reservar en otro restaurante aunque se encuentre en su cuenta de restaurante.
- No se requiere un nivel educativo específico, experiencia o conocimientos avanzados en la utilización de herramientas de este tipo para hacer uso de este sistema.

2.2.4 Restricciones

2.2.4.1 Limitaciones de hardware

El sistema debe poder verse en navegadores tanto de ordenadores (de sobremesa y portátiles) como de dispositivos móviles (smartphones y tablets).

2.2.4.2 Lenguajes de programación

- El sistema se debe desarrollar utilizando las siguientes tecnologías del lado del cliente: HTML5, CSS3, JavaScript, jQuery y AJAX.
- Se utilizará el framework Semantic UI para la interfaz de usuario.

- El sistema se debe desarrollar utilizando la siguientes tecnologías del lado del servidor: PHP y SQL.
- Se utilizarán objetos en formato JSON para el intercambio de datos entre el cliente y el servidor.
- Se utilizará el framework Laravel para desarrollar en la parte del servidor, por lo tanto la estructura del proyecto se deberá adaptar a la estructura de directorios de Laravel.
- Se utilizará el ORM Eloquent para la comunicación con la base de datos.

2.2.4.3 Protocolos de comunicación

El sistema debe adaptarse al protocolo de comunicación HTTP. Se utilizarán dos tipos de comunicación:

- Una mediante el método POST para realizar operaciones con el servidor que involucren datos críticos que no se puedan mostrar en la URL, como el correo o la contraseña de la cuenta de un restaurante o datos de la base de datos como el identificador de un restaurante.
- Otra comunicación mediante el método GET para el resto de casos.

2.2.4.4 Consideraciones acerca de la seguridad

- Se mantendrá un sistema de seguridad para las cuentas de restaurantes mediante un formulario de inicio de sesión en el que se deberá introducir un correo electrónico y una contraseña.

- Las contraseñas se guardarán en la base de datos cifradas mediante un hash en Bcrypt generado por una función del framework Laravel.

- Las páginas que formen parte de las cuentas de restaurantes no serán accesibles por usuarios no autorizados.

2.3 Requisitos específicos

2.3.1 Funciones

2.3.1.1 Funciones de los usuarios clientes

- Los clientes pueden consultar datos relacionados con los restaurantes que poseen una cuenta en este sistema. Dichos datos son:
  - Una descripción del restaurante.
  - Fotografías de las instalaciones o platos del restaurante.
  - Menú del restaurante.
  - Dirección y localización del restaurante en un mapa.
  - Datos de contacto como un teléfono y correo electrónico.
  - Número máximo de personas por reserva.
  - Horarios de apertura del restaurante.

- Los clientes también pueden dejar una valoración en estrellas (de una a cinco estrellas) y un comentario en la página principal de cada restaurante.

- Por último, los clientes pueden reservar en cualquiera de los restaurantes proporcionando la siguiente información:
  - Día de la reserva.
  - Franja horaria de la reserva (desayuno, comida o cena)
  - Hora de la reserva.
  - Número de personas que asistirán.
  - Nombre y apellidos de contacto.
  - Correo electrónico de contacto.
  - Teléfono de contacto.
  - Aceptar la política de privacidad.

2.3.1.2 Funciones de los usuarios restaurantes

- Los restaurantes podrán crear una cuenta en el sistema y completar datos referentes a su establecimiento. Estos datos son:
    - Fotografías de las instalaciones o platos del restaurante.
    - Nombre del restaurante.
    - Descripción.
    - Número máximo de comensales en el restaurante.
    - Dirección: vía, calle, número, CP, Población, Ciudad, País.
    - Indicar cómo se puede llegar al restaurante (mediante transporte público, privado o andando)
    - Contacto: prefijo, teléfono, correo electrónico.
    - Horarios: desayunos, comidas, cenas y días que cierra el restaurante.
    - Agregar un menú del restaurante.

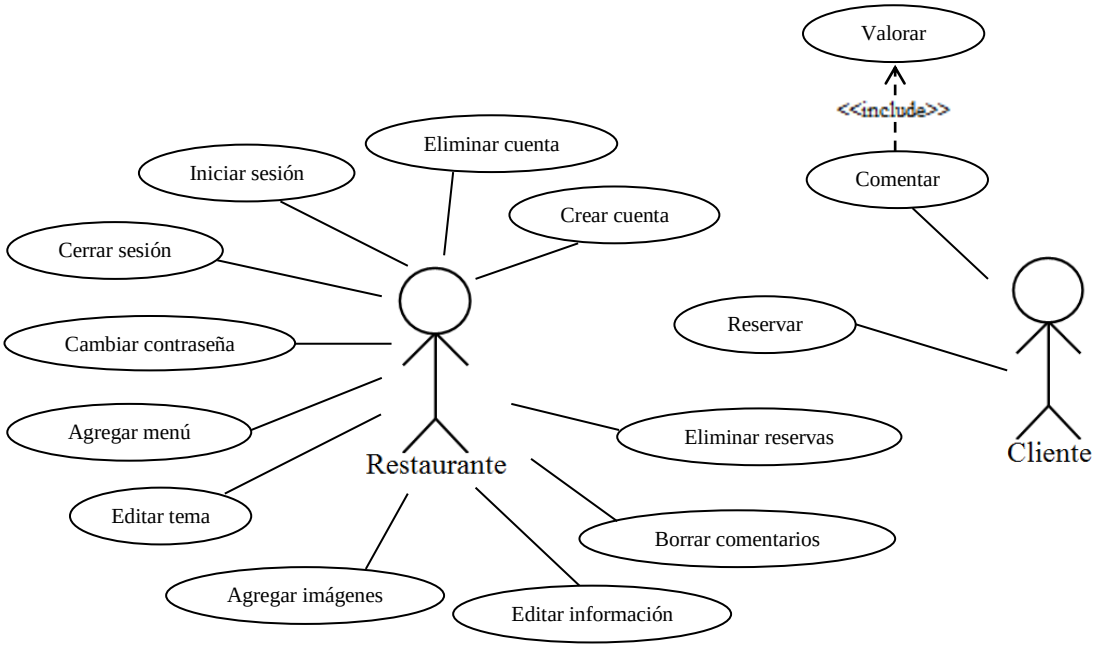
- Los restaurantes también podrán editar el tema de su cuenta para que su página tenga un aspecto personalizado. Éstos pueden editar lo siguiente:
    - Agregar imagen de encabezado.
    - Agregar un logo.
    - Cambiar el color de fondo del cuerpo de la página.
    - Cambiar el color de la letra del cuerpo de la página.

- Además podrán realizar las siguientes acciones relacionadas con su cuenta:
    - Iniciar y cerrar sesión en su cuenta.
    - Cambiar la contraseña de la cuenta.
    - Eliminar la cuenta.

- Podrán ver y borrar las valoraciones y comentarios de los clientes.

- Por último, cada restaurante podrá consultar los datos de todas las reservas que se hayan hecho en su establecimiento y eliminarlas.

2.3.1.3 Diagrama de casos de uso



2.3.1.4 Casos de uso

- **Casos de uso de Restaurantes**

1. Crear cuenta

Nombre	*Crear cuenta*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	-
Descripción	*Registro de restaurantes.*
Curso típico de eventos	
Restaurante	**Sistema**
	1. Mostrar una el formulario para crear una cuenta
2. Rellenar campos ***correo electrónico, nombre del restaurante***, ***contraseña*** y ***repetir contraseña***	
	3. Validar que los campos no están vacíos
	4. Validar que las dos contraseñas son iguales
	5. Validar que el usuario no está ya registrado
	6. Guardar al usuario en la base de datos
	7. Iniciar sesión automáticamente

Curso alternativo al paso 3	
Restaurante	**Sistema**
	3. Validar que los campos no están vacíos
	4. Mostrar mensaje de error indicando que se deben completar todos los campos
	5. Volver al paso 2 del curso típico de eventos

Curso alternativo al paso 4	
Restaurante	**Sistema**
	4. Validar que las dos contraseñas son iguales
	5. Mostrar mensaje de error indicando que las dos contraseñas introducidas deben ser iguales
	6. Volver al paso 2 del curso típico de eventos

Curso alternativo al paso 5	
Restaurante	**Sistema**
	5. Validar que el usuario no está ya registrado
	6. Mostrar mensaje de error indicando que ya existe un usuario registrado con el correo introducido
	7. Volver al paso 2 del curso típico de eventos

2. Iniciar sesión

Nombre	*Iniciar sesión*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	-
Descripción	*Inicio de sesión de restaurantes.*
Curso típico de eventos	
Restaurante	**Sistema**
	1. Mostrar el formulario para iniciar sesión
2. Rellenar campos ***correo electrónico*** y ***contraseña***	
	3. Validar que los campos no están vacíos
	4. Validar que el usuario y la contraseña existen y coinciden
	5. Mostrar página de inicio del restaurante correspondiente

Curso alternativo al paso 3	
Restaurante	**Sistema**
	3. Validar que los campos no están vacíos
	4. Mostrar mensaje de error indicando que se deben completar todos los campos
	5. Volver al paso 2 del curso típico de eventos

Curso alternativo al paso 4	
Restaurante	**Sistema**
	4. Validar que el usuario y la contraseña existen y coinciden
	5. Mostrar mensaje de error indicando que no se pudo iniciar sesión porque el usuario o la contraseña no son correctos
	6. Volver al paso 2 del curso típico de eventos

3. Cambiar contraseña

Nombre	*Cambiar contraseña*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Haber iniciado sesión*
Descripción	*Cambio de contraseña de una cuenta de Restaurante*

Curso típico de eventos	
Restaurante	**Sistema**
	1. Mostrar el formulario para cambiar la contraseña
2. Rellenar campos ***contraseña actual***, ***nueva contraseña*** y ***repetir nueva contraseña***	
	3. Validar que los campos no estén vacíos
	4. Validar que la contraseña actual introducida coincide con la guardada en la base de datos
	5. Validar que la contraseña nueva introducida no es la misma que la actual
	6. Mostrar un mensaje indicando que la contraseña se ha modificado correctamente

Curso alternativo al paso 3	
Cliente, Restaurante	**Sistema**
	3. Validar que los campos no estén vacíos
	4. Mostrar mensaje de error indicando que los campos no pueden estar vacíos
	5. Volver al paso 2 del curso típico de eventos

Curso alternativo al paso 4	
Restaurante	**Sistema**
	4. Validar que la contraseña actual introducida coincide con la guardada en la base de datos
	5. Mostrar mensaje de error indicando que la contraseña actual es incorrecta
	6. Volver al paso 2 del curso típico de eventos

Curso alternativo al paso 5	
Restaurante	**Sistema**
	5. Validar que la contraseña nueva introducida no es la misma que la actual
	6. Mostrar mensaje de error indicando que la contraseña nueva introducida no puede coincidir con la contraseña actual
	7. Volver al paso 2 del curso típico de eventos

4. Eliminar cuenta

Nombre	*Eliminar cuenta*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Haber iniciado sesión*
Descripción	*Eliminar cuenta de un Restaurante*
Curso típico de eventos	
Restaurante	**Sistema**
	1. Mostrar una ventana emergente con un aviso de que se va a borrar la cuenta
2. Pulsar en aceptar	
	3. Eliminar registro en la tabla clientes

5. Cerrar sesión

Nombre	*Cerrar sesión*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Haber iniciado sesión*
Descripción	*Cerrar sesión de una cuenta de Restaurante*
Curso típico de eventos	
Restaurante	**Sistema**
1. Pulsar en cerrar sesión	
	2. Eliminar la sesión

6. Editar tema

Nombre	*Editar Tema*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Tener una cuenta y haber iniciado sesión como Restaurante*
Descripción	*Editar el tema de la página de un restaurante*
Curso típico de eventos	
Restaurante	**Sistema**
	1. Mostrar una página con las opciones para modificar el tema
2. Cambiar ***color de fondo*** y ***color de letra***	
	3. Guardar colores en la tabla *tema*

7. Agregar imágenes

Nombre	*Agregar imágenes*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Haber iniciado sesión*
Descripción	*Agregar logo, encabezado e imágenes del restaurante*

Curso típico de eventos	
Restaurante	**Sistema**
	1. Mostrar una página para agregar las imágenes
2. Subir el ***logo*** del restaurante	
	3. Comprobar que el formato y el tamaño de la imagen son correctos
	4. Guardar imagen en la carpeta correspondiente del restaurante
5. Subir el encabezado de la página del restaurante	
	6. Comprobar que el formato y el tamaño de la imagen son correctos
	7. Guardar imagen en la carpeta correspondiente del restaurante
8. Subir imágenes del restaurante	
	9. Comprobar que el formato y el tamaño de las imágenes son correctos
	10. Guardar las imágenes en la carpeta correspondiente del restaurante

Curso alternativo al paso 3, 6 y 9	
Restaurante	**Sistema**
	3. Comprobar que el formato y el tamaño de la imagen son correctos
	4. Mostrar mensaje de error indicando que el formato o tamaño de la imagen no es válido
	5. Volver al paso 2, 5 u 8 del curso típico de eventos

8. Agregar menú

Nombre	*Agregar menú*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Haber iniciado sesión*
Descripción	*Agregar menú del restaurante*

Curso típico de eventos	
Restaurante	**Sistema**
	1. Mostrar una página para agregar el menú
2. Subir el ***menú*** del restaurante	
	3. Comprobar que el formato del menú es .pdf
	4. Guardar el menú en la carpeta correspondiente del restaurante

Curso alternativo al paso 3	
Restaurante	**Sistema**
	3. Comprobar que el formato del menú es .pdf
	4. Mostrar mensaje de error indicando que el formato del menú debe ser .pdf
	5. Volver al paso 2 del curso típico de eventos

9. Editar información

Nombre	*Editar Información*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Haber iniciado sesión*
Descripción	*Editar los datos de la página de un Restaurante*

Curso típico de eventos	
Restaurante	**Sistema**
	1. Mostrar una página para editar la información del restaurante
2. Editar la información obligatoria	
	3. Comprobar que ningún campo obligatorio está vacío y que el formato es correcto
	4. Guardar los datos del restaurante en la tabla de la base de datos

Curso alternativo al paso 3	
Restaurante	**Sistema**
	3. Comprobar que ningún campo obligatorio está vacío y que el formato es correcto
	4. Mostrar mensaje de error indicando los campos que están vacíos o que no tienen un formato correcto
	5. Volver al paso 2 del curso típico de eventos

10. Borrar comentarios

Nombre	*Borrar comentarios*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Haber iniciado sesión y tener algún comentario en la página*
Descripción	*Borrar comentarios en la página principal de un Restaurante*
Curso típico de eventos	
Restaurante	**Sistema**
1. Pulsar el botón borrar en un comentario	
	2. Mostrar mensaje de confirmación
3. Pulsar aceptar	
	4. Borrar comentario de la base de datos

11. Eliminar reservas

Nombre	*Eliminar reservas*
Actores	*Restaurante*
Tipo	*Primario*
Precondiciones	*Haber iniciado sesión y tener alguna reserva en el restaurante*
Descripción	*Eliminar reservas de un Restaurante*
Curso típico de eventos	
Restaurante	**Sistema**
1. Pulsar el botón borrar en una reserva	
	2. Mostrar mensaje de confirmación
3. Pulsar aceptar	
	4. Borrar asistentes de la base de datos para la reserva correspondiente
	5. Borrar reserva de la base de datos

- **Casos de uso de Clientes**

1. Comentar/Valorar

Nombre	*Comentar/Valorar*
Actores	*Cliente*
Tipo	*Primario*
Precondiciones	-
Descripción	*Valorar y escribir un comentario en la página principal de un Restaurante*
Curso típico de eventos	
Cliente	**Sistema**
1. ***Valorar*** al restaurante, escribir un ***nombre*** y un ***comentario***	
	2. Comprobar que se ha valorado al restaurante y que se ha introducido un nombre y un comentario
	3. Agregar valoración, nombre y comentario a la tabla Valoraciones

Curso alternativo al paso 2	
Cliente	**Sistema**
	2. Comprobar que se ha valorado al restaurante y que se ha introducido un nombre y un comentario
	3. Mostrar mensaje de error indicando que se deben completar todos los campos
	4. Volver al paso 1 del curso típico de eventos

2. Reservar

Nombre	*Reservar*
Actores	*Cliente*
Tipo	*Primario*
Precondiciones	-
Descripción	*Reserva de una mesa en un restaurante*

Curso típico de eventos	
Cliente	**Sistema**
1. Pulsar el botón ***Reservar***	
	2. Mostrar la página para reservar en el restaurante correspondiente
3. Rellenar campos ***día, tipo de comida***, ***hora, número de personas***, ***nombre***, ***apellidos***, ***correo electrónico***, ***teléfono*** y aceptar la ***política de privacidad***	
	4. Comprobar que ningún campo está vacío y que se ha aceptado la política de privacidad
	5. Comprobar que hay mesas disponibles para el día, hora y número de personas indicados por el cliente
	6. Actualizar la tabla Asistentes en el día de la reserva para el restaurante correspondiente
	7. Agregar la reserva en la tabla Reservas
	8. Enviar correo electrónico al cliente con los datos de su reserva

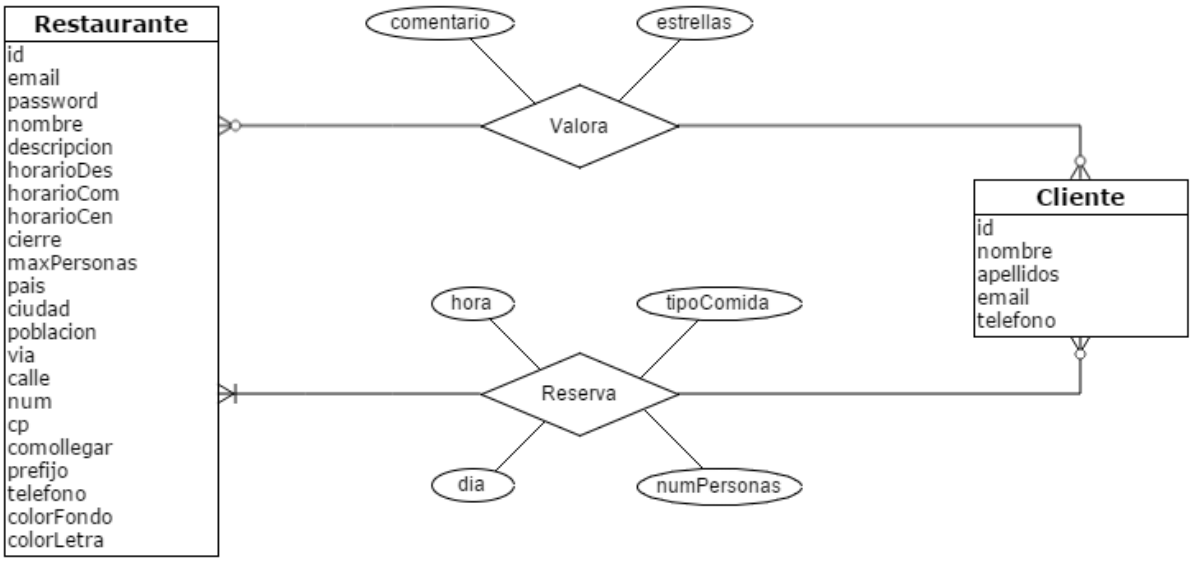
Curso alternativo al paso 4	
Cliente	**Sistema**
	4. Comprobar que ningún campo está vacío y que se ha aceptado la política de privacidad
	5. Mostrar mensaje de error indicando que se deben completar todos los campos
	6. Volver al paso 3 del curso típico de eventos

Curso alternativo al paso 5	
Cliente	**Sistema**
	5. Comprobar que hay mesas disponibles para el día, hora y número de personas indicados por el cliente
	6. Mostrar mensaje de error indicando no quedan mesas libres para la combinación indicada
	7. Volver al paso 3 del curso típico de eventos

2.3.2 Restricciones de diseño

- Un restaurante puede tener cero o más valoraciones pero una valoración pertenece a un sólo restaurante.
- Un restaurante puede tener cero o más reservas pero una reserva pertenece sólo a un restaurante.
- Un restaurante cuenta el número de asistentes por día. Así, un día podrá tener cero o más asistentes para desayunar, comer o cenar.

2.3.2.1 Diagrama de Entidad Relación



2.3.2.2 Modelo de datos

El modelo de datos representa las tablas que se crearán en la base de datos.

Éste diagrama parte del diagrama de Entidad-Relación, ya que cada entidad se convertirá en una tabla, y cada relación con cardinalidad N-N (de varios a varios ejemplares) también se convertirá en una tabla.

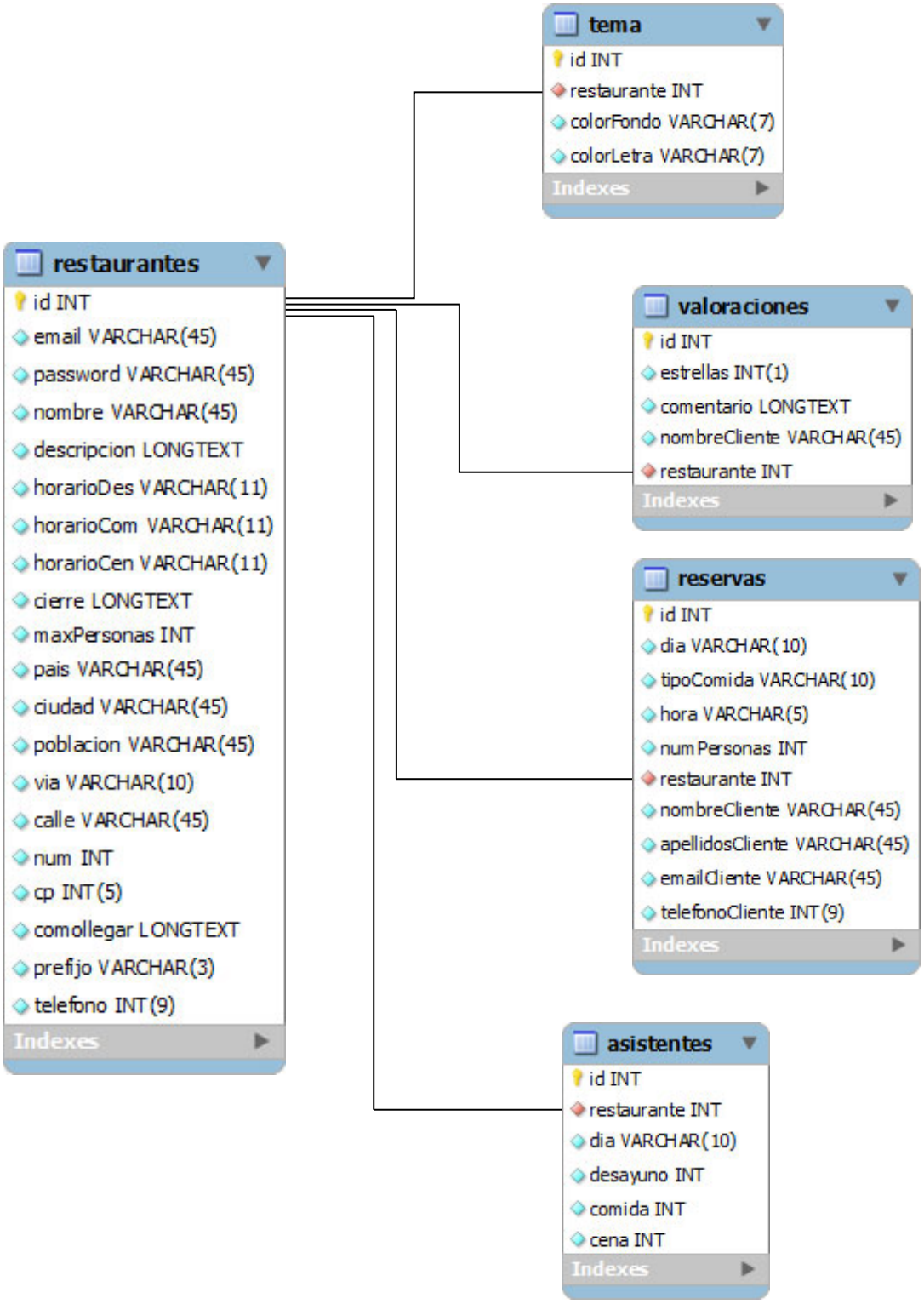
En cuanto a las columnas de las tablas, cada atributo de una entidad será una columna. Para las relaciones tenemos que cada atributo de la relación será una columna y además las claves primarias de cada entidad que relaciona también serán columnas de esta tabla.

Debido a que se ha decidido no crear cuentas de usuarios, la tabla usuarios no se ha incluido en la base de datos y sus atributos se encuentran en el resto de tablas relacionadas con los clientes.

De esta manera, las tablas resultantes del anterior diagrama de Entidad-Relación son: *Restaurantes*, *Valoraciones* y *Reservas*.

Aparte de estas tablas, se ha creado una tabla llamada *Tema*, en la cual se guarda el tema de la página de cada restaurante. Esta decisión se ha tomado debido a la comodidad de tener los datos de los restaurantes y los datos de las páginas por separado, ya que la tabla Restaurantes tenía demasiadas columnas.

Además, para llevar una gestión de reservas más óptima, se ha creado una tabla adicional llamada *Asistentes*. En esta tabla se actualizará el número de asistentes que pueden reservar en un restaurante, en un día determinado y en una franja horaria (desayuno, comida o cena) determinada. Así, si alguno de los turnos está lleno un día en concreto, no se podrá reservar y el cliente recibirá un mensaje para saber que no quedan mesas libres para la combinación de día, hora y número de personas que ha elegido.


tema
id INT
restaurante INT
colorFondo VARCHAR(7)
colorLetra VARCHAR(7)
Indexes
restaurantes
id INT
email VARCHAR(45)
password VARCHAR(45)
nombre VARCHAR(45)
descripcion LONGTEXT
horarioDes VARCHAR(11)
horarioCom VARCHAR(11)
horarioCen VARCHAR(11)
cierre LONGTEXT
maxPersonas INT
pais VARCHAR(45)
ciudad VARCHAR(45)
poblacion VARCHAR(45)
via VARCHAR(10)
calle VARCHAR(45)
num INT
cp INT(5)
comollegar LONGTEXT
prefijo VARCHAR(3)
telefono INT(9)
Indexes
valoraciones
id INT
estrellas INT(1)
comentario LONGTEXT
nombreCliente VARCHAR(45)
restaurante INT
Indexes
reservas
id INT
dia VARCHAR(10)
tipoComida VARCHAR(10)
hora VARCHAR(5)
numPersonas INT
restaurante INT
nombreCliente VARCHAR(45)
apellidosCliente VARCHAR(45)
emailCliente VARCHAR(45)
telefonoCliente INT(9)
Indexes
asistentes
id INT
restaurante INT
dia VARCHAR(10)
desayuno INT
comida INT
cena INT
Indexes

3

Herramientas

Índice

A continuación se pasan a explicar las distintas herramientas mencionadas en la especificación de requisitos, así como los entornos de programación utilizados y otras herramientas útiles.

3.1 Tecnologías del lado del cliente

3.1.1 HTML

HTML (HyperText Markup Language)[1] es un lenguaje de marcado que se utiliza para describir la estructura y la información de las páginas web. Es un estándar del W3C (World Wide Web Consortium), lo que significa que todos los navegadores web lo interpretarán de la misma manera y mostrarán el mismo contenido.

En este lenguaje todos los elementos están rodeados por "etiquetas" que marcan el inicio (<etiqueta>) y el final (</etiqueta>) de dicho elemento. La última versión es HTML5 y es la que se ha usado en este proyecto.

3.1.2 CSS

Aunque dentro de los documentos HTML se puede incluir información referente a la apariencia o estilo de las páginas web, esto no debería hacerse así. HTML describe sólo el contenido de la página, la información que se va a mostrar.

Las hojas de estilo CSS (Cascading Style Sheets u Hojas de Estilo en Cascada)[2] son las encargadas de describir el aspecto y formato de la información que se presenta en un documento escrito en lenguaje de marcado, como puede ser HTML, XML, SVG o XUL. La última versión es CSS3 y ésta incluye nuevas propiedades y características con respecto a versiones anteriores.

[1] http://www.w3.org/html/

[2] http://www.w3.org/TR/CSS/

En este proyecto, se han utilizado hojas de estilo CSS3 para darle formato a las páginas escritas en HTML. Esto se consigue haciendo referencia a la hoja de estilo que le va a dar formato a dicho documento. Para conseguir esta referencia se define, en el documento HTML, la siguiente etiqueta:

<link href="ruta/HojaDeEstilo.css" type="text/css">

- En el atributo "href" se especifica la ruta en la que se encuentra el documento .css que queremos cargar.

- En la etiqueta "type" se indica el tipo del documento que estamos importando, en este caso es de tipo texto escrito en CSS.

La manera en la que las hojas CSS hacen referencia a los elemento de los documentos HTML para cambiar su estilo es mediante identificadores o clases. Se debe definir un identificador (*id*) o una clase (*class*) en el elemento cuyo estilo queremos editar, y hacer referencia a éste desde el documento CSS de la siguiente manera respectivamente:

HTML:

*** ***

*** ***

CSS:

#id_del_elemento{…}

.clase_del_elemento{…}

La diferencia entre una clase y un identificador es que los identificadores son únicos, por lo que no se le debe asignar un mismo identificador a dos elementos diferentes. Esto sirve para fijar un punto de ancla en un elemento y así poder hacer referencia a él desde

otra parte de la página web o desde el código javascript. También se usan en el css en ciertos casos en los que sólo se quiera cambiar el estilo de un elemento aislado. Por el contrario, si queremos aplicar el mismo estilo a varios elementos diferentes, debemos definir una clase. Las clases no son únicas y se pueden asignar a más de un elemento a la vez, incluso un mismo elemento puede tener más de una clase. Éstas se suelen usar con más frecuencia en el css aunque también se pueden usar para hacer referencia a varios elementos en el código javascript.

3.1.3 JavaScript

JavaScript[3] es un lenguaje de programación orientado a objetos, interpretado (no necesita una previa compilación del programa para ejecutarse) y débilmente tipado. Se suele utilizar principalmente del lado del cliente, mejorando la interfaz web al añadirle funcionalidad dinámica.

Al igual que las hojas de estilo, podemos agregar scripts directamente en las páginas HTML, pero ésta no es la forma correcta de hacerlo. La forma de hacer referencia a estos documentos javascript es similar a la anterior con los documentos CSS:

<script src="ruta/Script.js"></script>

- En el atributo "src" se especifica la ruta en la que se encuentra el documento .js que queremos cargar.

La manera en la que Javascript interactúa con los elementos HTML es accediendo a ellos mediante su id, clase o nombre y realizando operaciones con éstos. La manera de acceder a un id, una clase (*class*) o un nombre (*name*) respectivamente es la siguiente:

var elemento1 = document.getElementById("idDelElemento");

[3] http://www.w3schools.com/js/

var elemento2 = document.getElementsByClassName("claseDelElemento");

var elemento3 = document.getElementsByName("nombreDelElemento");

Con Javascript también se puede acceder al árbol DOM de HTML. Recorriendo las etiquetas como si formasen un árbol, se puede acceder al padre (*parent*), hijos (*children*) o hermanos (*siblings*) a través de un elemento en concreto cuya referencia podemos obtener mediante uno de los métodos mencionados anteriormente.

var elemento = document.getElementById("id").childNodes[0].nodeValue;

También se puede utilizar este método para modificar el estilo CSS, por ejemplo:

document.getElementById('id').style.color = 'red'

3.1.4 JQuery

JQuery[4] es una librería de Javascript que facilita y hace más simple el desarrollo y la interacción con los elementos HTML, peticiones Ajax, animaciones o el manejo de eventos.

La manera en la que jQuery accede a los elementos HTML mediante identificadores, clases o nombres es la siguiente respectivamente:

$("#id_del_elemento").método();

$(".clase_del_elemento").método();

$("[name="nombre_del_elemento"]").método();

[4] https://jquery.com/

En este proyecto se han utilizado librerías de jQuery para insertar elementos específicos como el autocompletado del buscador de restaurantes (*JQuery Autocomplete*) o un selector de hora y fecha para las reservas (*JQuery DateTimePicker*)

3.1.5 Semantic UI

Semantic UI (*User Interface*)[5] es un framework que facilita la creación de interfaces de usuario de una manera simple a través de elementos HTML flexibles, es decir, que se adaptan a diferentes tamaños de pantalla y dispositivos (ordenadores, móviles o tablets).

Este framework proporciona elementos como iconos, botones, tablas, listas, formularios, etc. Además ofrece una variedad de estilos a través de sus clases. De esta manera, por ejemplo, podemos cambiar el color de un elemento simplemente añadiendo la clase *ui* y uno de los colores que ofrece Semantic UI, como por ejemplo *teal*.

También cuenta con una cuadrícula o *grid* que permite situar los elementos en la pantalla de una forma ordenada y simétrica. Cuenta con 16 columnas de ancho, es decir, podemos especificar el número de columnas que ocupará cada elemento en una línea en la pantalla.

Pero esto es sólo la parte visual, para que estos elementos tengan funcionalidad, se utilizan métodos jQuery. Éstos son algunos ejemplos de código:

HTML:

```
<div class="ui grid">
    <div class="four wide column">
        <div class="ui pointing label">Esto es un label</div>
    </div>
    <div class="four wide column">
        <div class="ui input">
```

[5] http://semantic-ui.com/

```
                    <input type="text" placeholder="Buscar...">
                </div>
            </div>
            <div class="four wide column">
                <div class="ui inverted purple button">Buscar</div>
            </div>
            <div class="four wide column">
                <div class="ui checkbox">
                    <input type="checkbox" name="ejemplo">
                    <label>Esto es un checkbox</label>
                </div>
            </div>
        </div>
```

Jquery:

```
            $('.ciudades.dropdown')
                .dropdown('setting', 'transition', 'vertical flip')
                .dropdown('set selected', 'Madrid')
                .popup('setting', 'content', 'Selecciona tu ciudad') ;
```

Otra característica muy interesante que tiene este framework es su sistema de validación de formularios. De esta forma, a través de jQuery, podemos especificar cualquiera de las restricciones que ofrece Semantic UI para los campos del formulario, como validación del formato de un correo electrónico o el número máximo o mínimo permitido en un campo. Además se pueden crear restricciones personalizadas y aplicarlas a cualquier elemento del formulario. También podemos definir errores para cada restricción que, con muy poco código, se mostrarán en el formulario cuando las restricciones no se cumplan.

```
            $('.ui.form')
```

```
.form({
        nombre: {
                identifier : 'nombre',
                rules: [ {
                        type : 'empty',
                        prompt : 'Este campo es obligatorio'
                } ]
        },
        correo: {
        identifier : ' correo ',
        rules: [ {
                type : 'email',
                prompt : 'Introduce un correo electrónico válido'
        } ]
},
…
}) ;
```

3.1.6 AJAX

AJAX (Asynchronous Javascript And XML)[6] es una tecnología web que se utiliza para actualizar datos en una página web sin necesidad de recargar la página entera. De esta forma, se pueden hacer peticiones de cualquier tipo (GET, POST,…) al servidor, realizar operaciones y devolver los resultados de manera asíncrona, es decir, en segundo plano y de forma transparente para el usuario. Esto permite que las páginas web sean más rápidas e interactivas.

En este proyecto se ha usado AJAX mediante jQuery para comprobar errores en todos los formularios.

[6] http://www.w3schools.com/ajax/

El formato de una petición AJAX en jQuery es el siguiente:

```
$.ajax({
    url: "ruta/fichero.php",
    type: "POST",
    data: {nombre1: datos1, nombre2: datos2, …},
    dataType: "json",
    success: function (result) {
       …
    },
    error: function () {
       …
    }
});
```

- ***url:*** es la ruta al fichero PHP que se encargará de realizar las operaciones necesarias en la parte del servidor

- ***type:*** es el tipo de petición HTTP. Las más comunes son POST y GET pero también existen otras como PUT y DELETE. Si no se indica nada en el atributo *type*, la petición por defecto es de tipo GET.

- ***data:*** son los datos que se pasan al servidor. Puede ser tanto un objeto como una cadena de datos. En este proyecto se han utilizado siempre objetos JSON.

- ***dataType:*** es el tipo de datos que se espera que devuelva el servidor. Los tipos pueden ser: xml, html, script, json, jsonp y text.

- ***success:*** es la función que se llama si la petición AJAX se ha realizado con éxito. El parámetro *result* es el resultado de la consulta.

- ***error:*** es la función que se llama si la petición AJAX no se ha podido realizar debido a un error.

3.2 Tecnologías del lado del servidor

3.2.1 PHP

PHP (Hypertext Preprocessor)[7] es un lenguaje de programación web del lado del servidor. Éste, al igual que las hojas de estilo CSS y los scripts JavaScript, se puede incrustar en las páginas HTML, pero una buena práctica es crear ficheros .php aparte y acceder a ellos mediante peticiones HTTP.

Aun así, a veces es inevitable insertar algo de código PHP dentro de alguna página HTML y para hacer esto, se debe encerrar dicho código entre las siguientes etiquetas:

```
<?php
    código PHP
?>
```

Una variante de estas etiquetas la podemos encontrar si simplemente queremos utilizar una variable php, en este caso, las etiquetas serán las siguientes:

```
<?= $variablePHP ?>
```

3.2.2 Laravel

Laravel[8] es un framework de PHP de código abierto. Éste facilita el desarrollo en este lenguaje de programación mediante una serie de herramientas que incorpora, como *Artisan*, *Composer* o el ORM *Eloquent*, entre otras.

[7] https://php.net/

[8] http://laravel.com/

Además, su estructura de directorios se organiza siguiendo la arquitectura MVC, ya que divide su estructura en modelos, vistas y controladores.

Aunque la última versión de este framework es la 5, en este proyecto se ha utilizado la versión 4 ya que era la última versión estable en el momento en que se empezó a desarrollar esta página web. Debido a los cambios tan grandes que se han introducido en la versión 5, no se ha podido migrar el código y se ha decidido continuar con la versión 4.

La decisión de utilizar Laravel sobre otros frameworks de PHP más extendidos como *Symfony* ha sido principalmente por su simplicidad y rapidez. Aparte de utilizar librerías de *Symfony*, Laravel dispone de una amplia documentación que resulta más sencilla que la de otros frameworks y una estructura de directorios a la que no cuesta mucho adaptarse.

En el apartado *Arquitectura del sistema* de esta memoria se muestra y se explica la estructura de directorios del proyecto realizado con Laravel.

3.2.2.1 Composer

Composer[9] es una de las herramientas de gestión de dependencias más utilizada, junto con Bower o npm. Laravel utiliza esta herramienta para manejar sus dependencias y permite instalar un nuevo proyecto listo para empezar a trabajar de una manera tan sencilla como es escribiendo un simple comando:

composer create-project laravel/laravel {directory} 4.2 --prefer-dist

[9] https://getcomposer.org/

3.2.2.2 Artisan CLI

Artisan CLI (Command-Line Console)[10] es otra de las herramientas que ofrece Laravel. Es una interfaz, dirigida por la consola de Symfony, que proporciona una serie de comandos muy útiles a la hora de desarrollar.

Además de los comandos de Artisan, también se pueden crear otros nuevos propios, lo cual facilita aún más el desarrollo con Laravel. Éstos se crean por defecto en la carpeta *app/commands* y la manera de crearlos es la siguiente:

php artisan command:make AssignUsers --command=users:assign

3.2.2.3 ORM Eloquent

El ORM (Object-Relational Mapping) Eloquent[11] es una herramienta que se incluye con Laravel y se utiliza para trabajar con bases de datos. Su funcionamiento es el siguiente:

Cada tabla se corresponde con un modelo de Laravel (*Model*) que hace de intermediario y se encarga de interactuar con la auténtica tabla que tenemos en nuestra base de datos.

Después de haber configurado la conexión con una base de datos en el fichero *database.php* que se encuentra en la carpeta *config*, se crea un modelo por cada tabla. Eloquent asume que el nombre de cada tabla será el plural del nombre del modelo. Así, por ejemplo, si tenemos un modelo *Usuario*, el nombre de la tabla asociada a éste será *usuarios*.

Otra cosa que asume Eloquent es que cada base de datos contiene una clave primaria con el nombre *id* y además ésta se usa en la mayoría de los métodos de este ORM.

[10] http://laravel.com/docs/4.2/artisan

[11] http://laravel.com/docs/4.2/eloquent

Para crear un modelo se utiliza Artisan, con el siguiente comando:

php artisan make:model Usuario

Aunque se pueden seguir las convenciones que Eloquen tiene implantadas, aun así se pueden cambiar el nombre de la tabla asociada y de la clave primaria por el nombre que queramos.

```
class Usuario extends Model {

    protected $table = “clientes”;

}
```

Además, Laravel puede automatizar la creación y actualización de marcas temporales o *timestamps*. Si queremos añadir esta característica sólo tenemos que añadir las columnas *created_at* y/o *updated_at* en nuestra tabla y activar la marca temporal en el modelo de la siguiente manera:

```
class Usuario extends Model {

    public $timestamps = true;

}
```

Tan sólo con esto podemos empezar a comunicarnos con la base de datos y realizar todo tipo de operaciones. Éstas son algunas de ellas:

```
$usuario = new Usuario();

$usuario ->email = 'ejemplo@gmail.com';

$usuario ->save();
```

3.2.2.4 Query Builder

Query Builder[12] es otra herramienta de Laravel, que permite construir y ejecutar consultas SQL. Ofrece una sintaxis simple y fácil de leer, y con ella se puede crear casi cualquier operación para comunicarnos con la base de datos.

Este mecanismo se ha utilizado bastante en este proyecto junto con el ORM Eloquent para realizar las operaciones con la base de datos.

A continuación se describen algunos ejemplos:

```
$usuario = DB::table('usuarios')->where('nombre', 'Juan')->first();
```

```
$usuarios = DB::table('usuarios')
        ->orderBy('nombre', 'desc')
        ->groupBy('count')
        ->having('count', '>', 100)
        ->get();
foreach ($usuarios as $usuario) {
    …
}
```

[12] http://laravel.com/docs/4.2/queries

3.2.3 SQL

SQL (Structured Query Language)[13] es un lenguaje estandarizado para comunicarse con bases de datos. El sistema de bases de datos utilizado en este proyecto es MySQL.

Con este lenguaje se pueden realizar consultas a la base de datos y obtener, modificar y eliminar datos o tablas. La sintaxis de este lenguaje es bastante simple y se entiende a simple vista, por ejemplo, para obtener todos los campos de una tabla llamada usuarios se haría de la siguiente manera:

SELECT * FROM usuarios

3.2.4 JSON

JSON (JavaScript Object Notation)[14] es una estructura o formato para almacenar e intercambiar datos. Es una alternativa a XML mucho más simple. Se utiliza para intercambiar datos entre el cliente y el servidor.

La estructura de un objeto JSON es la siguiente:

```
{"empleados":[
    {"nombre":"Juan", "apellido":"Pérez"},
    {" nombre ":"Ana", " apellido ":"Sánchez"},
    {" nombre ":"Pedro", " apellido ":"González"}
]}
```

- Los objetos JSON son arrays asociativos del tipo ***clave:valor*** que pueden estar formados a su vez por más arrays asociativos anidados.

[13] http://www.w3schools.com/sql/

[14] http://www.w3schools.com/json/

En este proyecto se han utilizado los objetos JSON para pasar información de todos los formularios de la parte del cliente al servidor y viceversa.

3.2.4 XAMPP

XAMPP (Apache, Mysql, Php, Perl)[15] es el servidor que se ha utilizado para trabajar localmente con la página web y la base de datos. Es una distribución de Apache que proporciona una base de datos MySQL, y soporte para trabajar con PHP y Perl.

3.3 Entornos de programación

3.3.1 Sublime Text

Sublime Text[16] el entorno de programación que se ha utilizado para programar la página web. Se ha decidido utilizar Sublime Text por ser el editor de texto más completo y simple de utilizar. Junto con *Brackets*, es uno de los más extendidos y populares, que cuenta con numerosos *plugins* que permiten una mejor personalización del entorno y una mejor experiencia de programación.

3.3.2 MySQL Workbench

MySQL Workbench es una herramienta para administrar bases de datos MySQL. En este proyecto se ha utilizado para modelar las tablas de la base de datos y sus relaciones.

3.3.3 Selenium IDE

Selenium IDE[17] es una extensión del navegador Mozila Firefox. Es un entorno para realizar pruebas unitarias en una página web. Se pueden grabar acciones y luego

[15] https://www.apachefriends.org/es/index.html

[16] http://www.sublimetext.com/

[17] http://www.seleniumhq.org/projects/ide/

reproducirlas y comprobar si se han completado correctamente o si ha habido algún problema.

Cuenta con una serie de comandos por cada acción realizada, como “click” al hacer click en un botón o “type” al escribir algo en un campo de texto. Además de grabar las pruebas, éstas se pueden editar a mano, escribiendo los comandos y datos correspondientes. Cada prueba genera un fichero .html con las acciones realizadas, que se puede guardar y ejecutar en cualquier momento.

3.4 Otras herramientas

3.4.1 Balsamiq Mockups

Balsamiq Mockups[18] es una herramienta gráfica para crear bocetos de las interfaces de usuario de páginas web o aplicaciones móviles. Se ha utilizado antes de empezar a desarrollar para planificar la estructura de la página web y tener una idea de cuál sería el resultado final de la interfaz de usuario.

Debido a que estos bocetos han sido previos a la fase de programación, difieren un poco del resultado final. En el apartado *Interfaz Web* de esta memoria se muestran estos borradores y se explican detalladamente los cambios realizados.

3.4.2 Gliffy

Gliffy[19] es una herramienta online para crear diagramas de todo tipo. En este proyecto se ha utilizado Gliffy para crear el diagrama UML de Entidad-Relación que aparece en el apartado *Diagramas de estructura* de esta memoria.

[18] https://balsamiq.com/products/mockups/

[19] https://www.gliffy.com/

3.4.3 Trello

Trello[20] es una aplicación web que permite crear tablones con listas compartidas para cualquier fin. En este proyecto se ha utilizado Trello para realizar un seguimiento de las tareas pendientes, las que se estaban realizando y las tareas acabadas. De esta forma ha resultado más fácil aplicar una metodología ágil (Scrum), de la que se hablará en el apartado *Metodología* de esta memoria.

3.4.4 Git

Git[21] es el sistema de control de versiones utilizado en este proyecto. Es muy importante el uso de un sistema como éste, ya que no sólo es útil cuando se trabaja en equipo, sino también al realizar proyectos individuales, debido al control que se tiene de los cambios realizados y la facilidad con la que se pueden recuperar versiones anteriores del código.

3.4.5 Bitbucket

Bitbucket[22] es una web en la que se pueden almacenar proyectos que utilizan un sistema de control de versiones como Mercurial o Git. No sólo se almacena el proyecto, sino que se lleva un control de todas las versiones del código, pudiendo acceder a cada una de ellas y ver qué es lo que se ha modificado y los comentarios realizados en cada cambio.

Además proporciona una serie de diagramas donde se puede ver fácilmente la trayectoria de versiones y quién ha realizado cada *commit*.

[20] https://trello.com/

[21] http://git-scm.com/

[22] https://bitbucket.org/

3.4.6 SourceTree

SourceTree[23] es una aplicación de escritorio creada por la empresa Atlassian, que sirve como cliente de la web *Bitbucket*.

SourceTree proporciona una interfaz con la que se puede trabajar con Git de una manera más visual e intuitiva, sin necesidad de recurrir a la línea de comandos. Enlazando un proyecto de *Bitbucket* con esta aplicación, los cambios que realicemos con SourceTree se actualizarán en la página web automáticamente.

[23] https://www.sourcetreeapp.com/

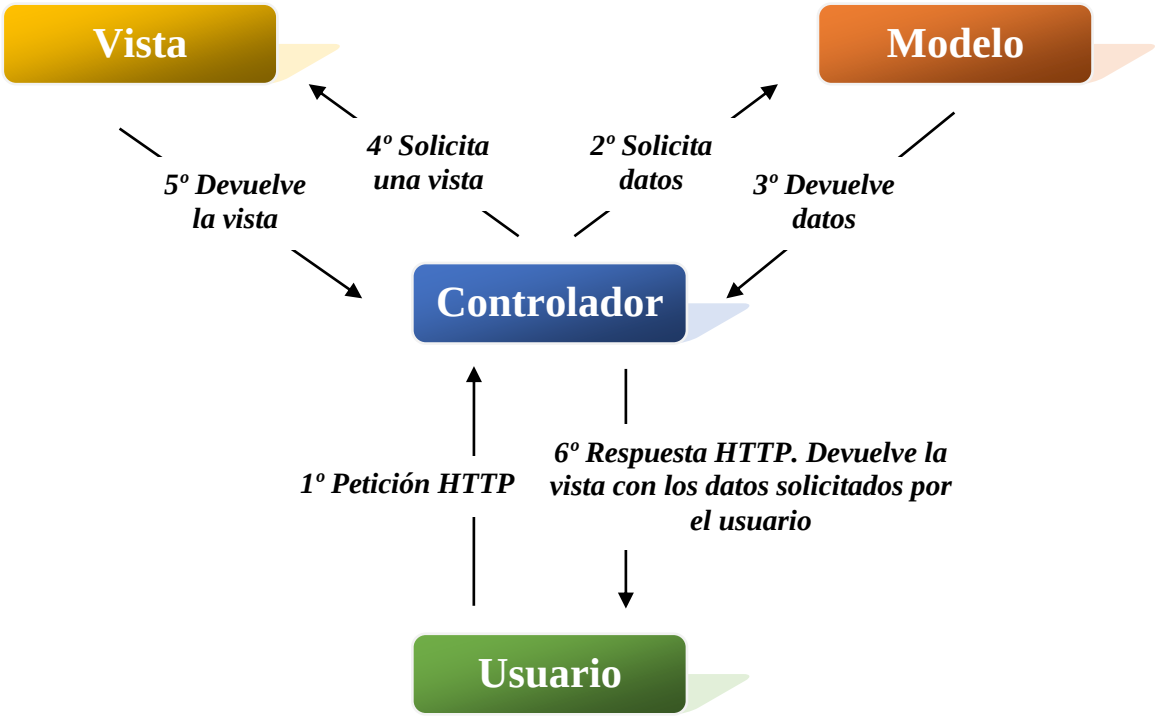
4

Arquitectura del sistema

Índice

4.1 Arquitectura MVC

En este proyecto se ha utilizado una arquitectura MVC (Modelo Vista Controlador)[24]. Esta arquitectura organiza el código en tres partes importantes: modelos, vistas y controladores.



El framework de PHP utilizado, Laravel, ofrece una estructura de directorios que sigue el modelo MVC, por lo que ha sido más fácil adaptarse a éste.

[24] http://librosweb.es/libro/jobeet_1_4/capitulo_4/la_arquitectura_mvc.html

4.1.1 Modelo

Los modelos son los encargados de comunicarse y realizar todas las operaciones con la base de datos. Estos reciben unas órdenes y devuelven los resultados en forma de datos. Esta parte se realiza del lado del servidor y no es visible para los usuarios, por lo que se puede trabajar con datos sensibles como contraseñas o datos privados de usuarios, realizar operaciones o consultas SQL entre otras cosas.
Para este proyecto se ha utilizado el lenguaje PHP para programar los modelos

4.1.2 Vista

Las vistas muestran la información al usuario. Son las interfaces, en este caso, interfaces web. Reciben los datos que los modelos han obtenido como resultado de las operaciones realizadas y los muestran.
Todo el código referente al contenido y estilo de las webs son vistas, es decir, todos lo que está escrito en lenguaje HTML o CSS.

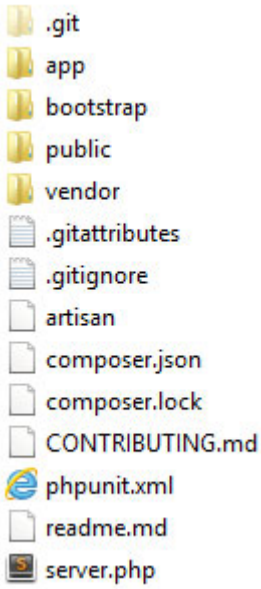
4.1.3 Controlador

Los controladores son los mediadores entre las vistas y los modelos.
Su función es la de recibir órdenes por parte se las vistas y comunicárselas a los modelos. Estos, tras realizar las operaciones oportunas, devuelven los resultados a los controladores, quienes se encargaran de comunicárselo a las vistas para que así puedan mostrar los datos a los usuarios.

A continuación se muestra la estructura de directorios del proyecto.

Laravel incorpora en su estructura muchos ficheros necesarios para su funcionamiento que no son relevantes en este proyecto, por lo que sólo se explicarán aquellos que han sido modificados o creados desde cero para esta aplicación web.

Raíz del proyecto

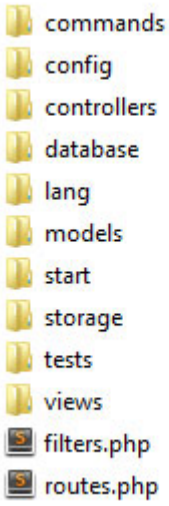


Esta es la raíz del proyecto.

- En la carpeta *app* es donde se encuentra todo el código php. Vistas, controladores, modelos, rutas y configuración necesaria para conectar con la base de datos y enviar correos electrónicos.

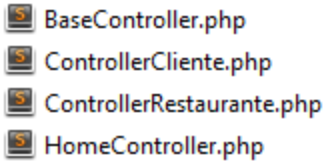
- En la carpeta *public* se encuentran los recursos. Ficheros javascript, hojas de estilo css, imágenes y paquetes utilizados, así como el índex de la web.

Carpeta *app*

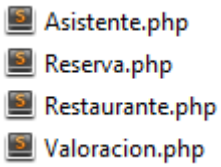


- En el fichero *routes.php* se encuentras todas las rutas de la aplicación. Éste sería el fichero que tiene la función de controlador en la arquitectura MVC, ya que es el intermediario entre las vistas y los modelos.

- En la carpeta *controllers* se encuentran los controladores de Laravel, estos realizan la función de los modelos de la arquitectura MVC ya que son los

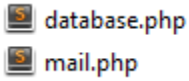


encargados de recibir información de las vistas, realizar las operaciones con la base de datos y devolver una respuesta a las vistas, todo ello a través del modelo.

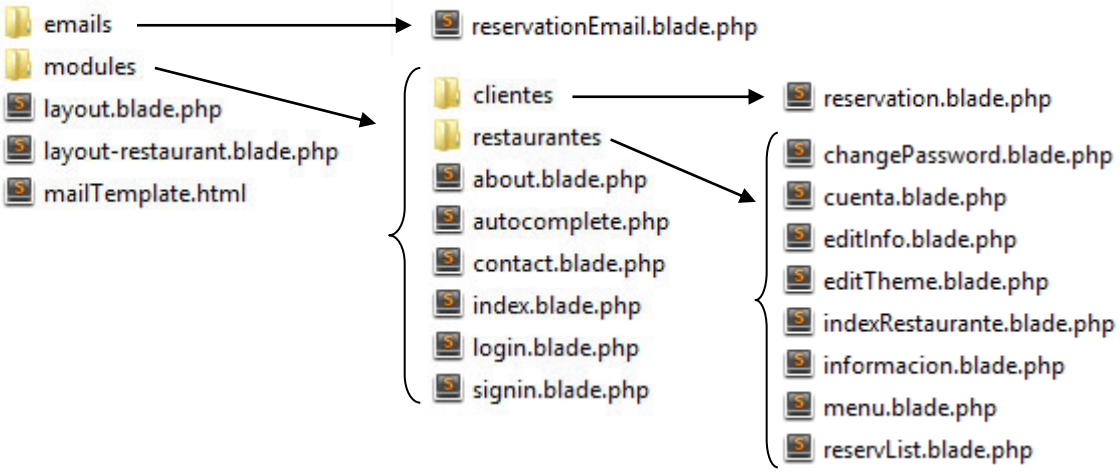


- En la carpeta *models* tenemos una serie de clases que no son exactamente los modelos de la arquitectura MVC, aunque Laravel los llame modelos. Estas clases sirven para comunicarse con la base de datos utilizando el ORM Eloquent. Cada clase se corresponde con una tabla de la base de datos, siento el nombre de la clase el mismo que el de la tabla pero en singular. Así, por ejemplo, si tenemos la tabla "restaurantes", la clase correspondiente será "Restaurante".

- La conexión con la base de datos se ha configurado en el fichero *database.php* que se encuentra dentro de la carpeta *config*. En éste se pueden configurar distintas plataformas como SQLite, MySQL, PostgreSQL o SQL Server. En este caso se ha utilizado MySQL. La configuración para el envío de correos electrónicos se realiza en el fichero *mail.php* que se encuentra en la misma carpeta. Los tipos soportados son SMTP, mail, sendmail, mailgun, mandrill y log. En este proyecto se ha utilizado SMTP con un servidor de mailgun.org.

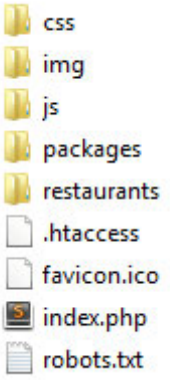


- Las vistas se encuentran en la carpeta *views*. Esta carpeta se ha separado en diferentes directorios para diferenciar las vistas relacionadas con los clientes que reservan de las de los restaurantes. Además, en esta carpeta también se encuentra el modelo de correo electrónico que se envía a los clientes cuando reservan una mesa o cuando hacen una consulta en la página de contacto.

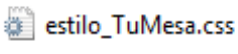


Carpeta *public*

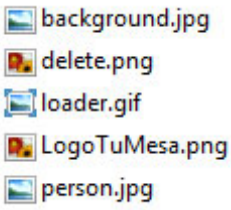
En esta carpeta se encuentra el fichero *index.php* y una serie de recursos.



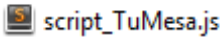
- La hoja de estilos css.



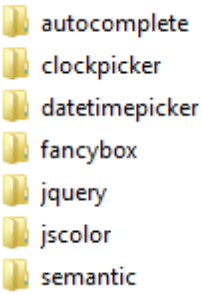
- Imágenes de la página web.



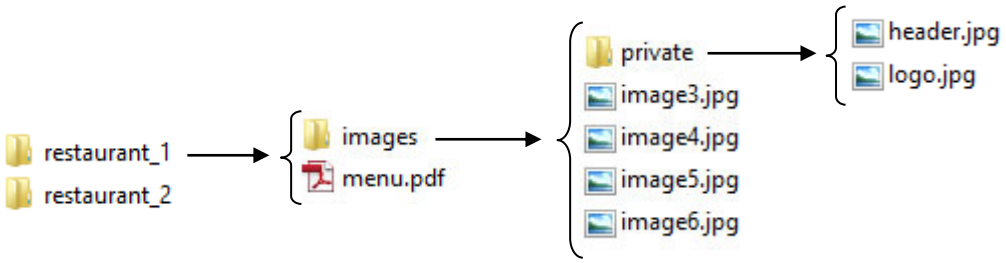
- El fichero javascript.



- Paquetes utilizados como *jquery*, *jquery.clockpicker*, *jquery.ui.autocomplete*, *jquery.fancybox*, *jscolor* y *semantic ui*.



- Además podemos encontrar la carpeta *restaurants* que contiene todos los recursos locales de los restaurantes. Cada restaurante, al crearse una cuenta, puede guardar imágenes y un fichero pdf con el menú. Todos esos recursos se encuentran en esta carpeta.



5

Metodología

Índice

5.1 Scrum

La metodología utilizada para realizar este proyecto ha sido Scrum[25]. Ésta es una metodología ágil, es decir, se trabaja de forma incremental, dividiendo el trabajo en tareas pequeñas, de manera que se pueda estimar un tiempo de finalización razonable para cada una. De esta manera se planea mucho mejor el trabajo y las tareas que se van realizando, así como las tareas pendientes y el trabajo que ya se ha finalizado.

Ésta metodología permite llevar un control de la magnitud del proyecto y del tiempo que llevará acabar el mismo, así como cada una de las tareas que nos hayamos propuesto. El hecho de poder conseguir pequeñas metas en un período de tiempo corto fomenta la motivación y da la sensación de estar consiguiendo los objetivos propuestos, por lo que mejora mucho la productividad.

Aunque se pueden aplicar a proyectos realizados por una sola persona, como es este caso, las metodologías ágiles mejoran muchísimo el trabajo en grupo, ya que ayudan a mantener una organización y una constante revisión de las tareas que se van realizando, lo cual fomenta la comunicación y el apoyo entre los integrantes del equipo.

Sin más dilaciones, pasemos a explicar el funcionamiento de esta metodología.

En Scrum se definen una serie de roles entre los miembros de un equipo. Estos roles son *Scrum Master*, *Product Owner* y *Team*.

5.1.1 Scrum Master

Éste es el encargado de asegurarse de que la metodología se está aplicando correctamente. No es ni un jefe ni un líder, ya que todos los miembros del equipo se

[25] https://www.scrum.org/

Scrum en menos de 10 minutos: https://www.youtube.com/watch?v=XU0llRltyFM

organizan por sí mismos, pero es el que se encarga de mantener el orden y de que nadie se desvíe de Scrum.

5.1.2 Product Owner

El Product Owner es el rol que representa a los *stakeholders* o clientes. Será el encargado de definir los requisitos y comunicarse con el equipo para decidir cuáles son más importantes o urgentes y cuáles serán los objetivos al final de cada fase o *sprint* de Scrum.

5.1.3 Team

Éste es el equipo encargado de llevar a cabo las tareas definidas en cada *sprint*. Es el que tiene que comunicarse con el cliente para acordar las qué requisitos se van a realizar y el tiempo que se va a tardar. Lo ideal es que sea un grupo de entre 3 y 9 personas, entre las cuales se encuentren las necesarias para realizar todas las tareas propuestas.

La dinámica de esta metodología es la siguiente:

El proyecto se divide en diferentes fases llamadas *sprints*. En cada uno de estos *sprints*, cuya duración debe ser corta y es recomendable que sea entre una y cuatro semanas, se realizan siempre los mismos pasos.

- Primero se realiza una reunión llamada *Sprint Planning meeting*, entre el equipo y el *Product Owner*, para especificar los requisitos que se van a llevar a cabo en dicho *sprint* (*Sprint Backlog*). Estos requisitos se eligen de entre todos los especificados para el proyecto que se encuentran en un documento llamado *Product Backlog*.

- Una vez definidos los requisitos para un *sprint* y el tiempo que se tardará en acabarlos, se reparten las tareas entre los miembros del equipo. Cada persona

escogerá la tarea que mejor se adapte a sus capacidades y procederá a realizarla durante el período de tiempo establecido.

- Al principio de cada día del *sprint*, se realizan pequeñas reuniones entre los miembros del equipo, llamadas *Daily Scrum* o *Stand-up meeting*. En éstas, cada integrante contará qué ha realizado el día anterior, cuáles han sido los problemas encontrados y qué es lo que pretende hacer ese mismo día. Esto sirve tanto para aclarar las ideas de la persona que lo está contando como para que los demás miembros del equipo estén al tanto de las dificultades o inconvenientes que sus compañeros están teniendo y así poder ayudar cuando sea posible. Estas reuniones deben ser cortas ya que se realizan diariamente, por lo que nunca deben superar los 15 minutos.

- Una vez acabado un *sprint*, se realiza una reunión llamada *Sprint Review meeting*, para presentar el trabajo finalizado.

- Por último, cada vez que se acaba este ciclo, se realiza una reunión llamada *Sprint Retrospective* en la que se hace una valoración del *sprint* finalizado para corregir posibles errores a la hora de aplicar esta metodología y así seguir mejorando.

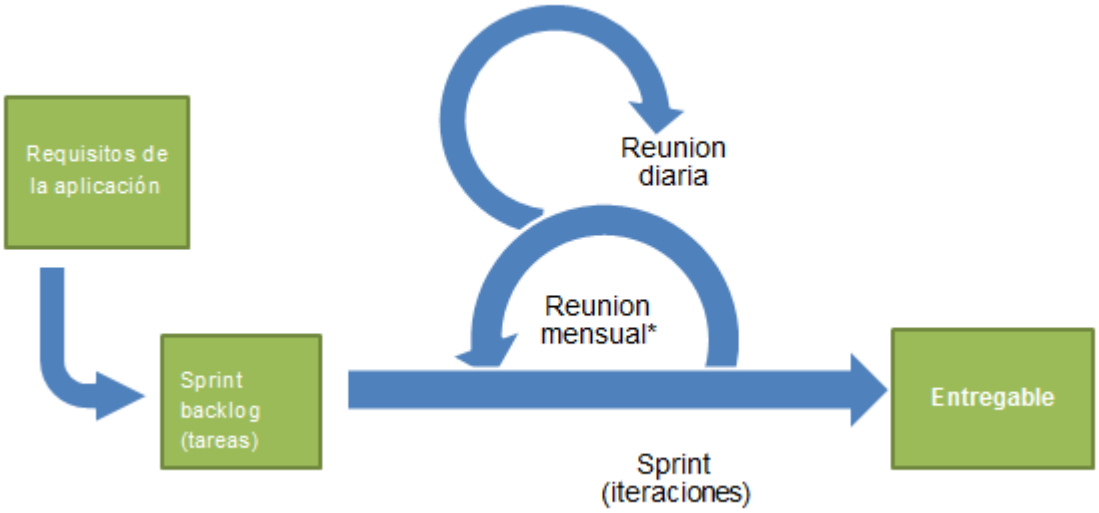


Imagen por Maxie Ayala[26]

[26] https://commons.wikimedia.org/w/index.php?title=User:Maxie_Ayala&action=edit&redlink=1

6

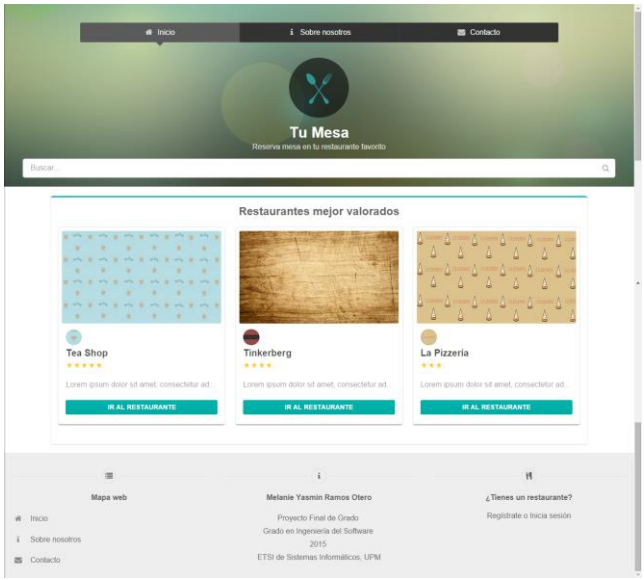
Interfaz web

A lo largo de la realización de este proyecto, la interfaz web ha ido variando con respecto a la idea original debido a ciertas decisiones que se han tomado que han llevado a cambiar la estructura y el diseño de la web. Estas decisiones se explican en el apartado *Anexo: decisiones sobre la estructura y el propósito de este proyecto* de esta memoria.

Para el diseño de la interfaz, se ha utilizado la aplicación Barsamiq Mockups para realizar bocetos de las diferentes pantallas de la página web. Ya que estos borradores se realizaron antes de empezar a desarrollar, no se corresponden con el resultado al que se ha llegado tras finalizar el proyecto.

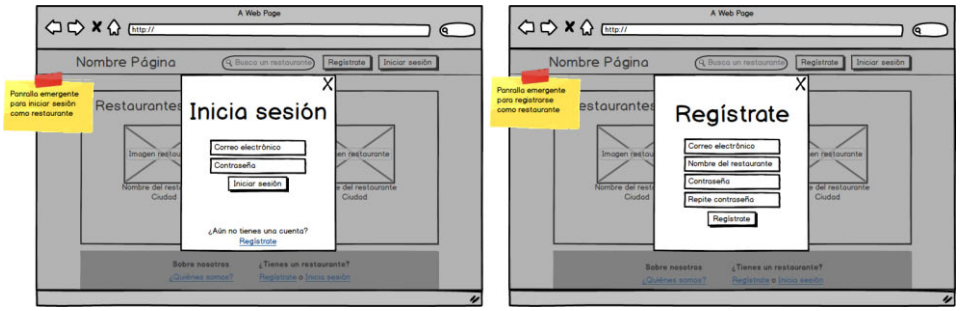
A continuación se muestra una comparación de los primeros borradores con el resultado final:

Borrador de la pantalla de inicio

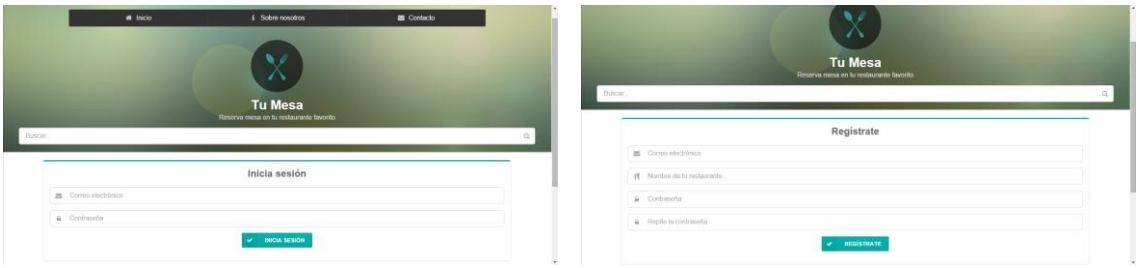


Captura de la pantalla de inicio

Pantallas relacionadas con los actores Restaurantes:

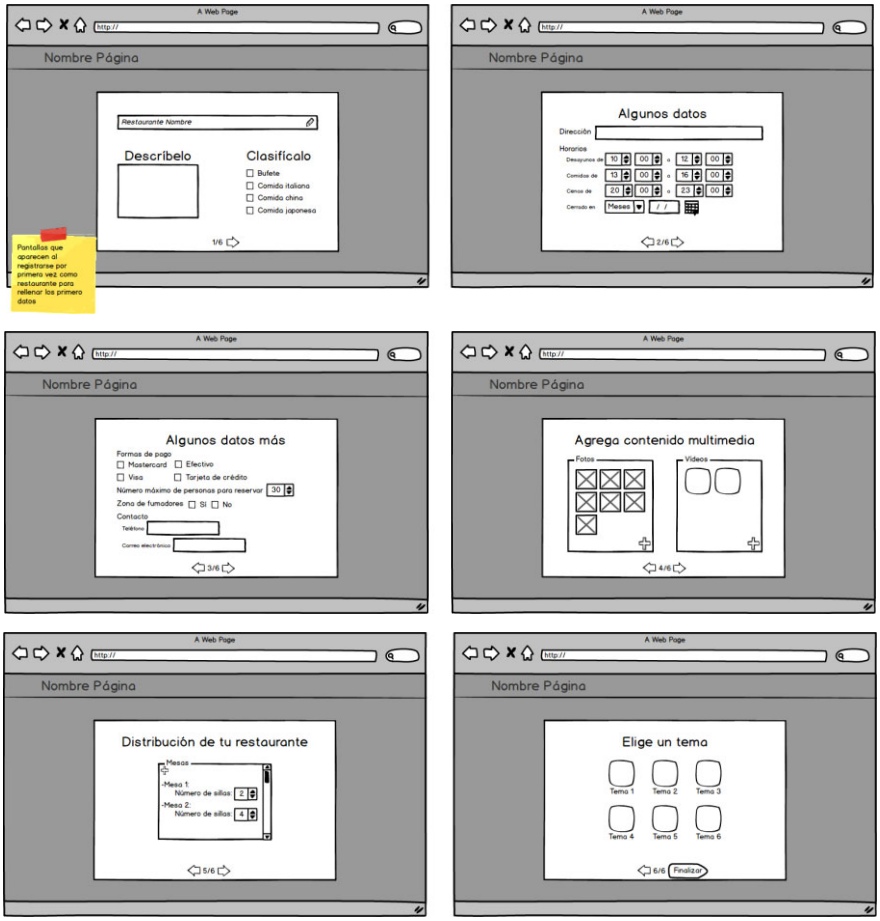


Borradores de las pantallas de inicio de sesión y registro de restaurantes



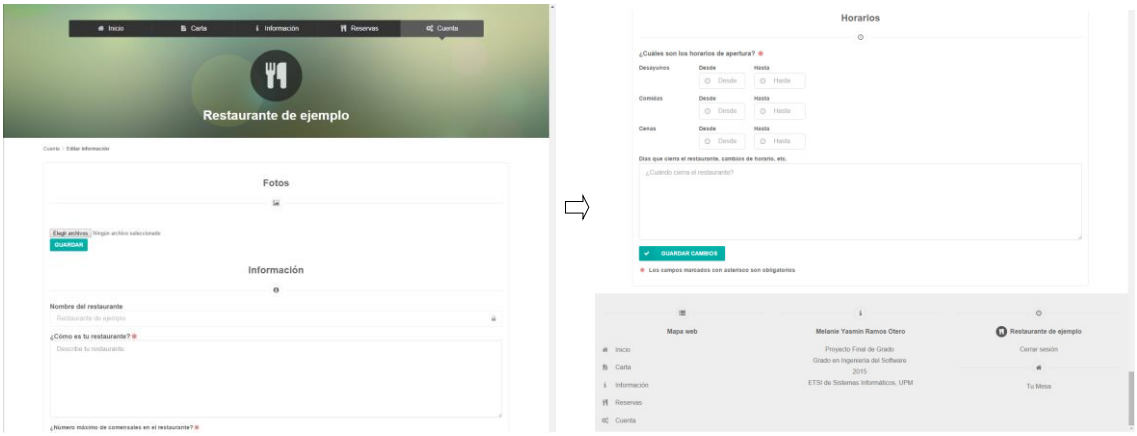
Capturas de las pantallas de inicio de sesión y registro de restaurantes

Estas dos pantallas también se habían diseñado para clientes con la misma estructura, pero como se ha comentado anteriormente, se decidió no hacer cuentas de clientes ya que de esta forma la web sería mucho más rápida y cómoda para reservar.



Borradores de las pantallas de edición de la información de los restaurantes

…

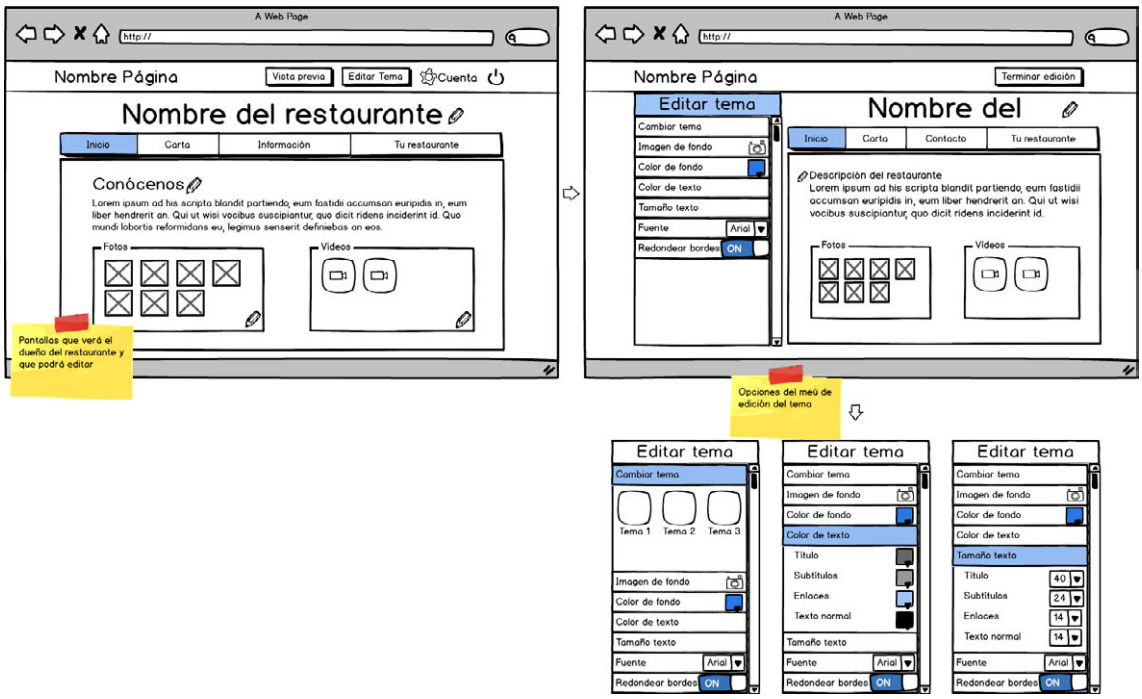


…

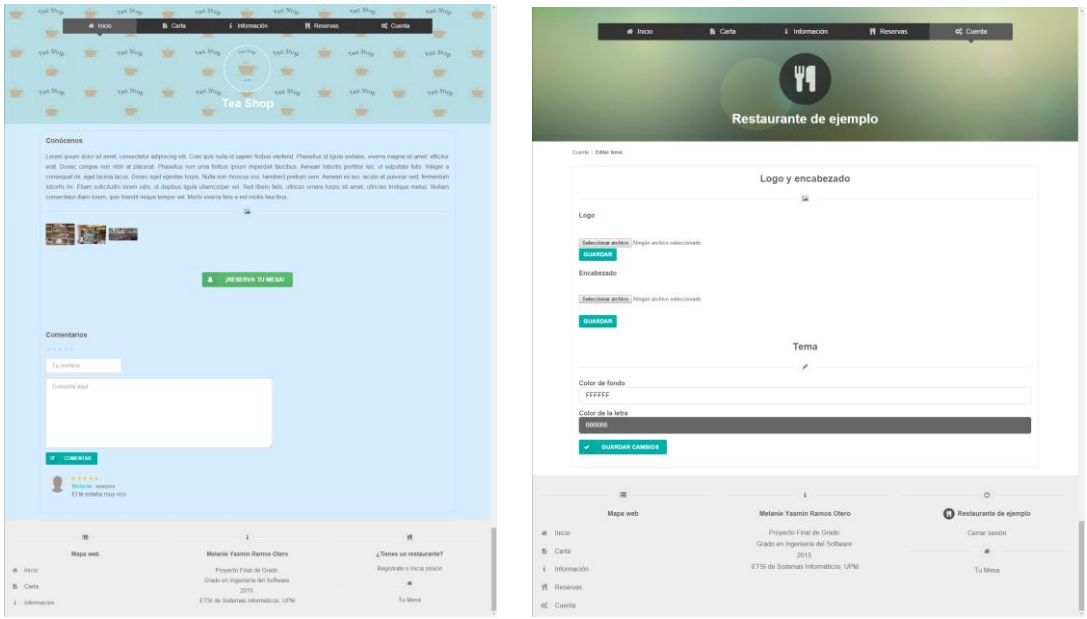
Capturas de las pantallas de edición de la información de los restaurantes

Se había pensado mostrar un formulario al inicio de la primera sesión, pero se decidió no incluirlo para que la creación de las cuentas fuese más fácil y dinámica. Sin

embargo, este formulario no se ha eliminado, sino que se puede acceder a él desde las preferencias de la cuenta.

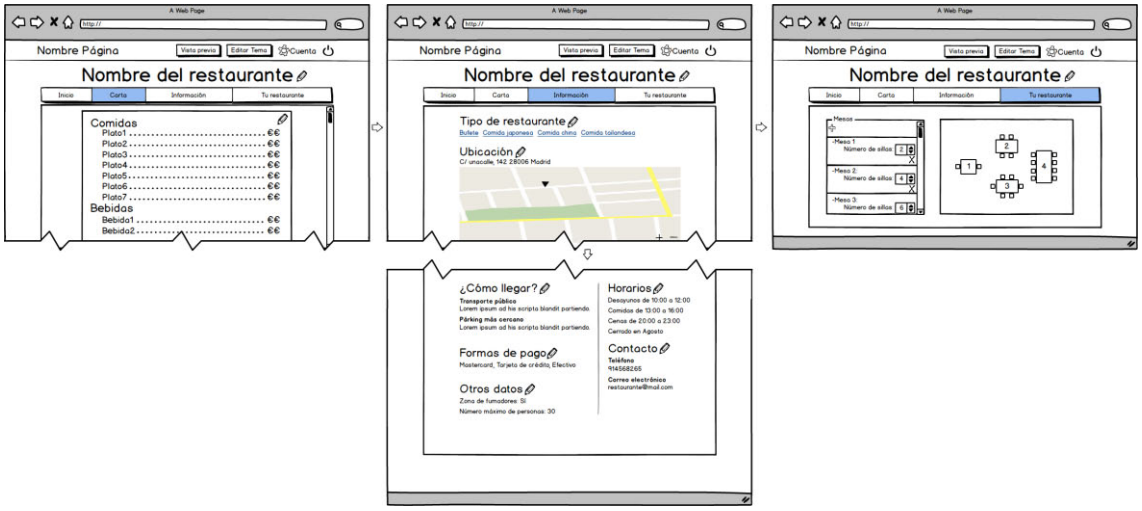


Borradores de las pantallas de inicio y editar tema de los restaurantes



Capturas de las pantallas de inicio y editar tema de los restaurantes

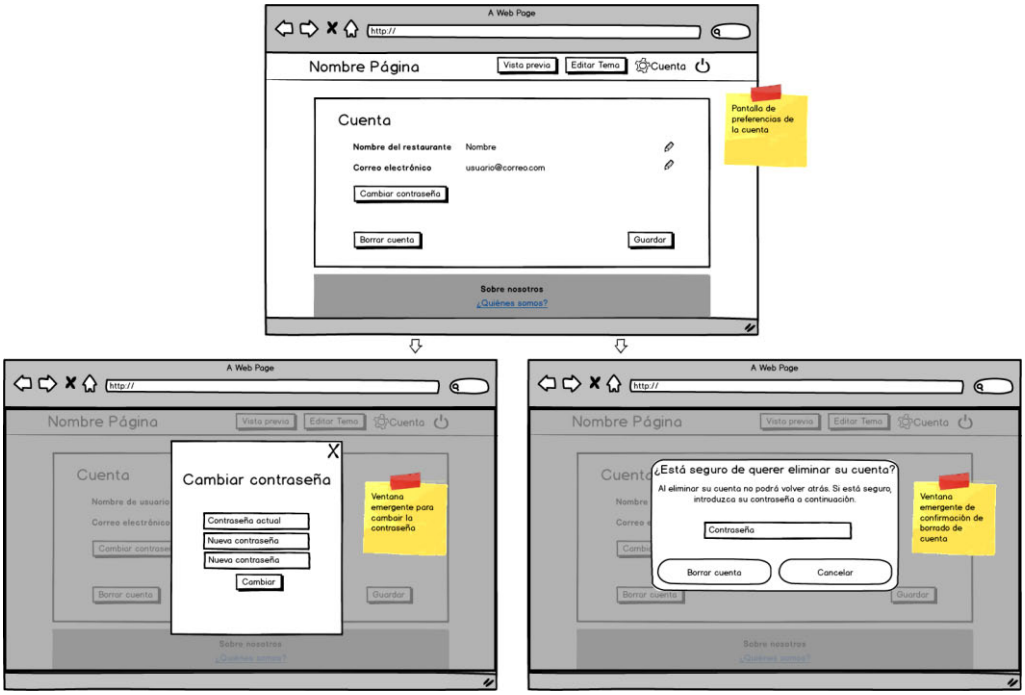
En este caso se había pensado una edición del tema de la página de cada restaurante más compleja de la que al final se implementó. Esta idea se descartó tras medir la carga de trabajo que llevaría realizar el proyecto, que era mucho mayor al tiempo del que se disponía.



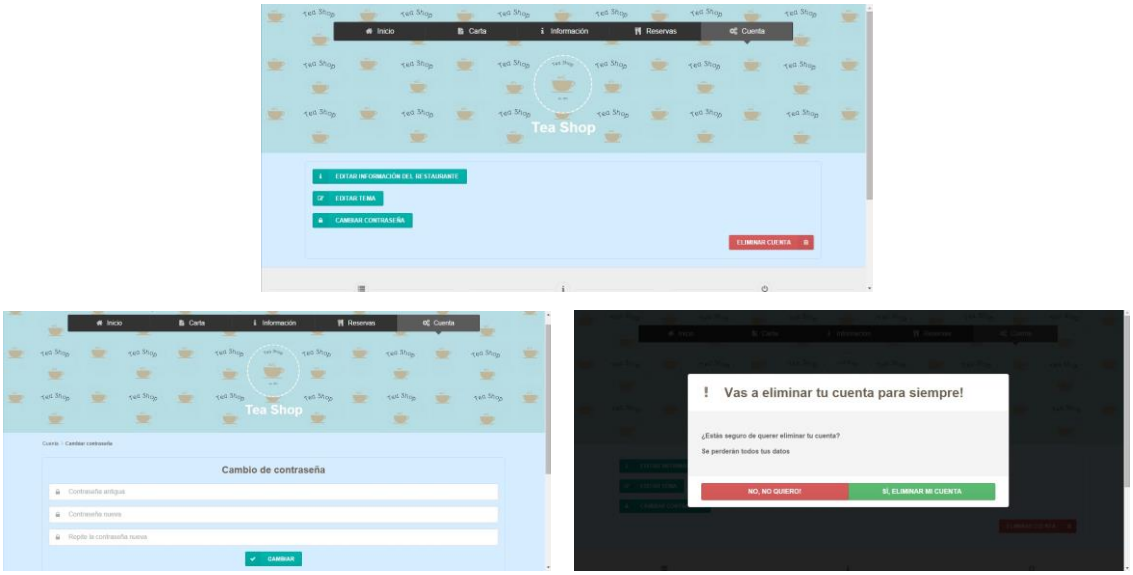
Borradores de las pantallas de Menú e Información de los restaurantes

Capturas de las pantallas de Menú e Información de los restaurantes

Tanto la pantalla de *Carta* como la de *Información* se han mantenido casi idénticas, con alguna excepción en su estructura. Sin embargo, la pantalla de *Tu restaurante*, al igual que antes, se acabó descartando tras analizar el alcance del proyecto y ver que no se podía realizar esta parte por falta de tiempo y recursos.



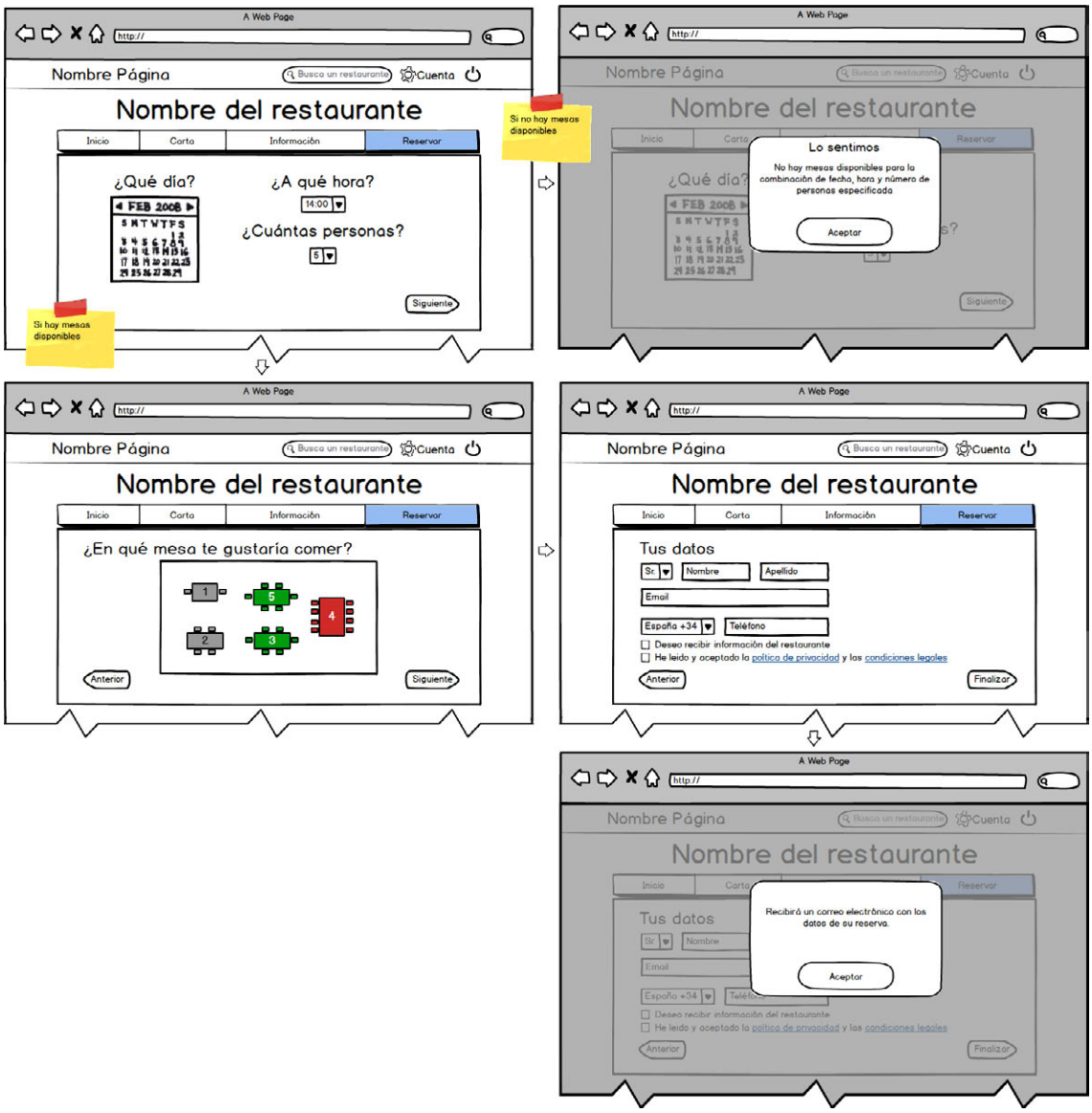
Borradores de las pantallas de los ajustes de cuenta de los restaurantes



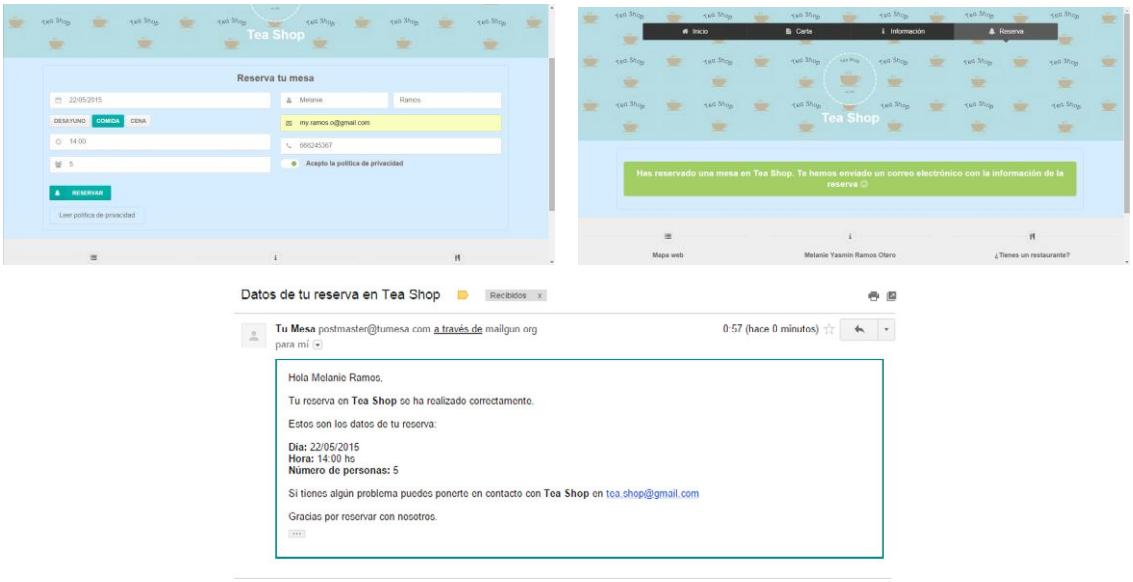
Capturas de las pantallas de los ajustes de cuenta de los restaurantes

Estas pantallas también se han mantenido y su diseño sólo varía en algunos elementos mínimos.

Pantallas relacionadas con los actores Clientes:



Borradores de las pantallas para reservar en un restaurante



Capturas de las pantallas para reservar en un restaurante

Finalmente estas son las pantallas para reservar una mesa en un restaurante. Esta estructura también ha cambiado, ya que al eliminar la pestaña *Tu restaurante*, en la que se podía crear la estructura de mesas, también desaparece la elección de mesa de la reserva.

Otro cambio es el número de pantallas. En la versión final, la reserva se hace en una sola pantalla, donde se puede elegir tanto el día como la hora y rellenar los datos personales.

Al finalizar la reserva se envía un correo electrónico al cliente con los datos de la misma.

7

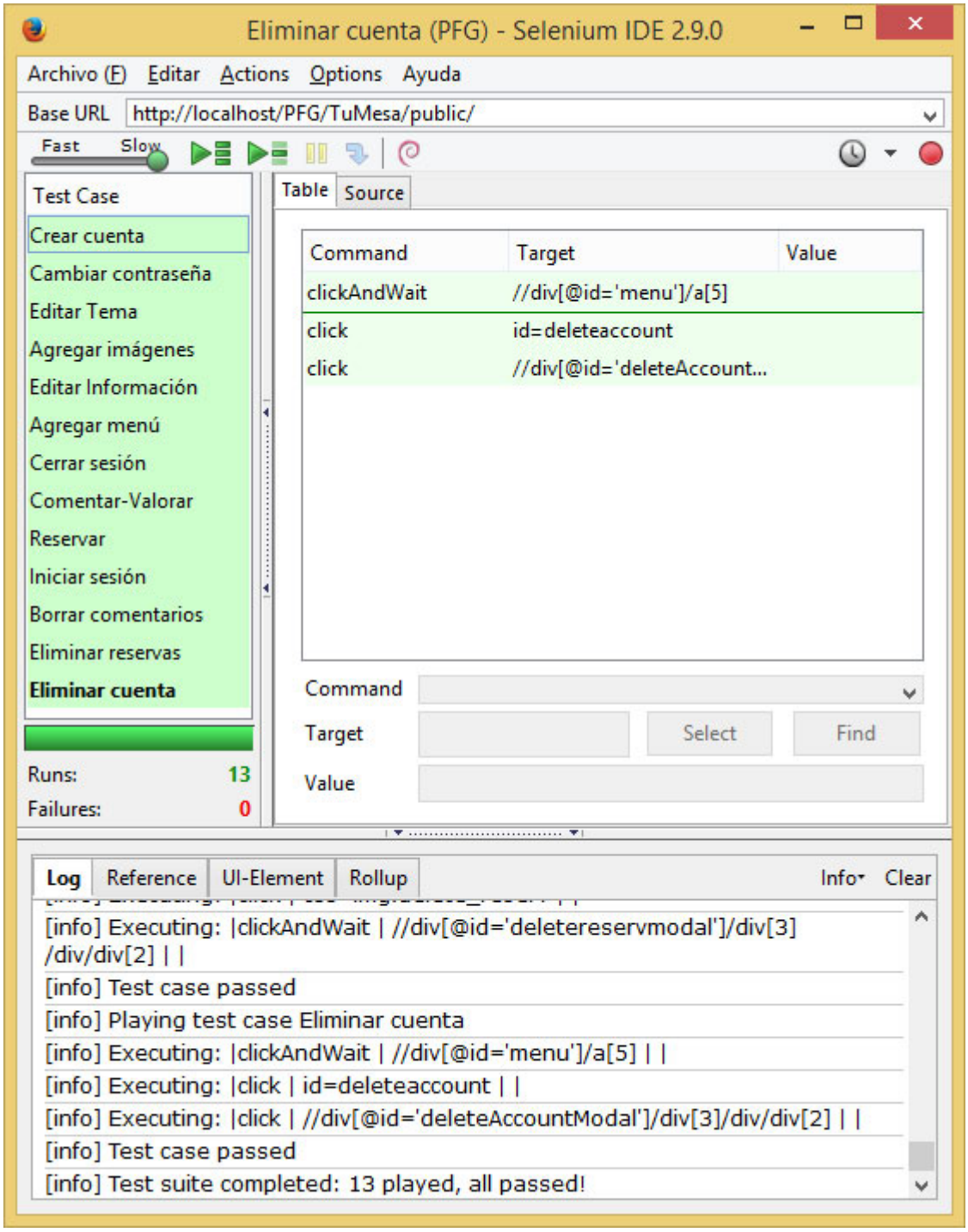
Pruebas unitarias

Durante la realización del proyecto se han realizado pruebas unitarias con el framework PHPUnit[27] y la herramienta Selenium IDE para automatizar los tests y mostrarlos de una manera más simple en la presentación.

Se ha realizado 13 pruebas, una prueba por cada caso de uso, repasando todas las funcionalidades de la página web y comprobando su correcta ejecución y funcionamiento.

La manera de trabajar con el IDE de Selenium es simple. Primero se crea un caso de prueba, luego se pulsa el botón grabar y se hace lo que se quiere probar en la propia página web. Esto guarda paso a paso lo que se ha hecho y se puede probar cuantas veces queramos, incluso cambiando los datos de entrada. De esta manera es mucho más sencillo probar la interfaz y el correcto funcionamiento sin necesidad de escribir casos de prueba a mano. Este IDE guarda un fichero con extensión .html por cada prueba, con los pasos que se han seguido.

[27] https://phpunit.de/



Selenium IDE tras pasar las 13 pruebas

8

Conclusiones

Tras la realización de este proyecto se han cumplido todos los objetivos propuestos. He mejorado mucho mis conocimientos sobre tecnologías web, he ampliado lo que ya sabía sobre HTML, CSS, JavaScript y PHP y he aprendido muchas otras cosas nuevas.

He aprendido a utilizar frameworks y librerías que me han ayudado mucho a desarrollar y organizar el proyecto. He incorporado buenas prácticas a la hora de estructurar el código y utilizar herramientas para realizar un trabajo más eficiente y mantenible.

Además he aprendido que dedicar un tiempo a la planificación previa, a trabajar siguiendo una arquitectura concreta como es la arquitectura MVC y aplicando una metodología como Scrum y ceñirse a ella, es una parte tan importante como la propia programación.

He descubierto que las tecnologías y herramientas que existen en este campo son muchísimas y que siguen creciendo cada día, por lo que hay que estar siempre actualizado. Pero esto no me supone un problema ya que esta experiencia ha fomentado mi interés por seguir aprendiendo y mejorando cada día en el ámbito del desarrollo y el diseño web.

9

Anexo: Ampliaciones

9.1 Cuenta de clientes

Una de las mejoras para este proyecto es la creación de cuentas para los clientes. Esto no sería obligatorio, ya que un cliente podría reservar en un restaurante sin la necesidad de crearse una cuenta. Sin embargo, al tener una cuenta, se podrían guardar restaurantes favoritos o realizar estadísticas para recomendar restaurantes a los clientes.

9.2 Aplicaciones móviles

Otra mejora sería la creación de aplicaciones para móviles con diferentes sistemas operativos (Android, iOS y Windows Phone). Las aplicaciones se adaptan muchísimo mejor a los dispositivos móviles que las páginas web y proporcionan una mejor experiencia de usuario.

Además, un cliente podría acceder a su cuenta desde la aplicación móvil o desde la web y todos sus datos estarían sincronizados para que pueda ver sus restaurantes favoritos desde cualquier dispositivo.

9.3 Programa de gestión

Por último, otra de las mejoras sería la de agregar un apartado en la cuenta de restaurantes con las herramientas necesarias para gestionar la mercadería que hay que comprar para la cocina o la contabilidad del restaurante, entre otras cosas.

10

Anexo: Decisiones sobre la estructura y el propósito de este proyecto

10.1 Gestión de restaurantes

Originalmente la aplicación web estaba destinada a la gestión de restaurantes, como el nombre de este proyecto indica, pero a medida que se han ido haciendo cambios, agregando alguna funcionalidad y descartando otra, el propósito de esta web ha variado un poco hasta llegar a ser una aplicación para reservar en restaurantes.

10.2 Cuenta de clientes

Con respecto a decisiones sobre las funciones de los actores, originalmente, ambos usuarios estarían obligados a crearse una cuenta para poder usar la aplicación web. Finalmente se decidió eliminar la cuenta de clientes por simplicidad y comodidad para las personas que quieran reservar. Si una persona quiere reservar sólo un día en un restaurante, seguramente no quiera perder tiempo en crearse una cuenta.

11

Referencias bibliográficas

11.1 Tecnologías web

11.1.1 Tecnologías del lado del cliente

W3C. *HTML*. <http://www.w3.org/html/>

Mozilla Developer Network. *HTML element reference*. <https://developer.mozilla.org/en-US/docs/Web/HTML/Element>

W3Schools. *HTML Tutorial*. <http://www.w3schools.com/html/>

W3C. CSS. <http://www.w3.org/TR/CSS/>

W3Schools. *CSS Tutorial*. <http://www.w3schools.com/css/>

W3Schools. *JavaScript Tutorial*. <http://www.w3schools.com/js/>

JQuery. <https://jquery.com/>

JQuery. *JQuery Date and Time picker*. <https://plugins.jquery.com/datetimepicker/>

DateTimePicker. *JQuery plugin select date and time*. <http://xdsoft.net/jqplugins/datetimepicker/>

JQuery User Interface. *Autocomplete*. <https://jqueryui.com/autocomplete/>

JSColor. *JavaScript / HTML Color Picker*. <http://jscolor.com/>

Semantic UI. <http://semantic-ui.com/>

W3Schools. *AJAX Tutorial*. <http://www.w3schools.com/ajax/>

JQuery.ajax(). <http://api.jquery.com/jquery.ajax/>

11.1.2 Tecnologías del lado del servidor

PHP. <https://php.net/>

PHP. *¿Qué es PHP?* <http://php.net/manual/es/intro-whatis.php/>

Laravel. Documentación en inglés <http://laravel.com/>
Laravel. Documentación en español <http://laraveles.com/docs/4.2/>

Laravel. *Query Builder*. Documentación en inglés.
<http://laravel.com/docs/4.2/queries>
Laravel. *Generador de consultas*. Documentación en español.
<http://laraveles.com/docs/4.2/queries>

Laravel. *Eloquent ORM*. Documentación en inglés.
<http://laravel.com/docs/4.2/eloquent>
Laravel. *Eloquent ORM*. Documentación en español.
<http://laraveles.com/docs/4.2/eloquent#introduction>

Laravel. *Artisan CLI*. Documentación en inglés. <http://laravel.com/docs/4.2/artisan>
Laravel. *Artisan CLI*. Documentación en español.
<http://laraveles.com/docs/4.2/artisan>

Laravel wiki. *Composer*. <http://wiki.laravel.io/Laravel_4#Composer>

Composer. <https://getcomposer.org/>

XAMPP. <https://www.apachefriends.org/es/index.html>

PhpMyAdmin. <http://www.phpmyadmin.net/home_page/index.php>

W3Schools. *SQL Tutorial*. <http://www.w3schools.com/sql/>

MySQL. <https://www.mysql.com/>

W3Schools. *JSON Tutorial*. <http://www.w3schools.com/json/>

MailGun. *Email Servic*e. <https://mailgun.com/>

11.2 Entornos de programación

Sublime Text. <http://www.sublimetext.com/>

MySQL. *MySQL Workbench*. <http://www.mysql.com/products/workbench/>

SeleniumHQ. *Selenium IDE*. <http://www.seleniumhq.org/projects/ide/>

SeleniumHQ. <http://docs.seleniumhq.org/>

PHPUnit. <https://phpunit.de/>

PHPUnit. *Chapter 13. PHPUnit and Selenium*. <https://phpunit.de/manual/current/en/selenium.html>

11.3 Otras herramientas

Balsamiq Mockups. <https://balsamiq.com/products/mockups/>

Gliffy. <https://www.gliffy.com/>

Trello. <https://trello.com/>

Git. *Control de versiones*. <http://git-scm.com/>

Bitbucket. *Repositorio de git*. <https://bitbucket.org>

SourceTree. *Cliente de Bitbucket*. <http://www.sourcetreeapp.com/>

Lorem Ipsum. <http://es.lipsum.com/>

Hipster Logo Generator. <http://www.hipsterlogogenerator.com/>

HTML Color Codes.<http://html-color-codes.info/codigos-de-colores-hexadecimales/>

Pixlr. *Editor*. <http://apps.pixlr.com/editor/>

FancyBox. <http://fancyapps.com/fancybox/>

Google Developers. *Google Maps API*. <https://developers.google.com/maps/?hl=es>

Google Developers. *Google Maps Geocoding API*. <https://developers.google.com/maps/documentation/geocoding/?hl=es>

Página oficial de Scrum. <https://www.scrum.org/>

Guía de Scrum. <http://www.scrumguides.org/>

Scrum en menos de 10 minutos. <https://www.youtube.com/watch?v=XU0llRltyFM>

Librosweb. *La arquitectura MVC*.
<http://librosweb.es/libro/jobeet_1_4/capitulo_4/la_arquitectura_mvc.html>

Internet Alchemy. *What are the benefits of MVC?*
<http://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/>

Rumbaugh, J., Jacobson, I., Booch, G. (2000): *El lenguaje unificado de modelado. Manual de referencia.* <https://ingenieriasoftware2011.files.wordpress.com/2011/07/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf>