



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Wheely Tasty

Un servicio integrado en Facebook para sugerir sitios donde comer

Trabajo de Fin de Grado
Grado en Ingeniería Informática

Autora: Carolina Gomez Gilabert

Tutor: Joan Josep Fons i Cors

2014/2015

Resumen

Wheely Tasty es una idea concebida en la oficina donde trabajo. Es una respuesta a la pregunta que se repite todos los días: ¿dónde comemos hoy? Invariablemente, esa pregunta genera discusiones, que tomamos demasiado tiempo en concluir.

El objetivo de la aplicación es solucionar ese problema de manera sencilla, eficaz y, cuando sea necesario, democrática. Para ello se diseñó un sistema que, basado en la ubicación del usuario, recoge los restaurantes cercanos a él disponibles en Facebook, a través de su Graph API.

Una vez el sistema tiene esa información, la dispone en una ruleta, que el usuario debe girar para elegir un sitio al azar.

El sistema también soporta decisiones en grupo, donde el giro de la ruleta por parte de cada miembro cuenta como un voto, sea de su ubicación actual o de un código postal elegido por él.

Finalmente, éste es un proyecto que, sobretodo, busca facilitar la vida de las personas en un proceso que, aunque parezca trivial, es relevante a la gran mayoría de nosotros.

Palabras clave: ruleta, Facebook, restaurantes, Graph API, comida

Abstract

Wheely Tasty is an idea born in the office I work at. It's the response to the question that arises every day: where do we have lunch today? Invariably, that question generates discussion, that we often take too long to resolve.

The main goal of the application is to solve that problem in a simple, effective and whenever necessary, democratic way. With that intent in mind, the system was designed to read the user's location and obtain information about the restaurants near him from Facebook, through their Graph API.

Once the system receives that information, it arranges it on a spinning wheel, that the user spins in order to choose a random restaurant.

The system also supports group decisions, where each member's spin counts as a vote, be it from the user's location at the time or a postcode entered by him.

Finally, this is a project that attempts to, above all, make peoples lives easier in a situation that may seem trivial, but is relevant to the vast majority of us.

Keywords: spinning wheel, Facebook, restaurants, Graph API, food

Índice de Contenidos

1. Introducción	7
1. Descripción	7
2. Antecedentes	7
3. Objetivos	7
4. Metodología	8
2. Contexto tecnológico	11
1. Aplicaciones similares	11
1. The Techdev Lunch Deciding Tool	11
2. What Should I Eat?	12
3. Wheel Decide	13
4. Wheel of Lunch	14
2. Herramientas necesarias	15
3. Caso de estudio: Wheely Tasty	17
1. Descripción	17
2. Especificación de requisitos	18
1. Requisitos funcionales	18
2. Requisitos de diseño	19
3. Requisitos tecnológicos	19
4. Análisis	20
1. Diagrama de clases	20
2. Interfaces de usuario	21
5. Diseño	25
1. Diagrama de clases	25
2. Diseño de la interfaz	27
6. Implementación	28
1. Base de Datos	28
2. Funcionalidad	30
3. Integración con Facebook	34
4. Interfaz de usuario	36
7. Conclusiones	38
1. Objetivos logrados	38
2. Objetivos no logrados	38
3. Ampliaciones	38
4. Comentario personal	39
8. Bibliografía	40
9. Anexos	41

Índice de Figuras

• 1.1 Detalle del diagrama de Gantt para la tarea 1	8
• 1.2 Detalle del diagrama de Gantt para la tarea 2	9
• 1.3 Detalle del diagrama de Gantt para la tarea 3	9
• 1.4 Detalle del diagrama de Gantt para la tarea 4	10
• 2.1 The Techdev Lunch Deciding Tool	11
• 2.2 What Should I Eat?	12
• 2.3 Wheel Decide	13
• 2.4 Wheel of Lunch	14
• 4.1 Diagrama de clases - análisis	20
• 4.2 Boceto de la pantalla principal	12
• 4.3 Boceto de la ventana de creación de grupo	22
• 4.4 Boceto de la ventana de votación en grupo	23
• 4.5 Boceto de la ventana de resultados de grupo	24
• 5.1 Diagrama de clases - diseño	25
• 6.1 Diagrama de la base de datos	28
• 6.2 Parte de código de la función loadRestaurants	30
• 6.3 Método formatData de la clase Wheel	31
• 6.4 Método codePostcode de la clase GoogleMaps	32
• 6.5 Método group_cleanup	32
• 6.6 Método registerVoteGroup de la clase Group	33
• 6.7 Petición de búsqueda a Graph API	34
• 6.8 Petición de información sobre una página	35
• 6.9 Pantalla principal de la aplicación	36
• 6.10 Detalle del menú	36
• 6.11 Ventana de búsqueda por código postal	37
.....	37
• 6.12 Método renderCreateGroupModal de la clase Modals	

1. Introducción

1.1 Descripción

Este proyecto aborda la creación de una aplicación para solucionar la problemática de la decisión de dónde ir a comer, sobretodo en grupos.

Lo hace obteniendo la ubicación del usuario, recogiendo sitios cercanos a él en Facebook, y poniéndolos en una ruleta para que el usuario la gire y elija uno al azar.

La aplicación también soporta votaciones en grupo, donde el giro de cada miembro cuenta como un voto.

1.2 Antecedentes

Las aplicaciones existentes en el mercado hasta el momento no cumplen los objetivos de este proyecto de manera completa.

Existen soluciones parciales, que permiten al usuario añadir sitios manualmente para elegir, o simplemente elegir entre tipos de culinaria, pero que requieren un mayor esfuerzo por parte del usuario.

Estas alternativas se detallan en el siguiente capítulo.

1.3 Objetivos

El objetivo fundamental de la aplicación es hacer la toma de decisión de dónde comer un proceso sencillo y divertido, que exija muy poco del usuario y sea altamente informativo.

A nivel de interfaz, se desea que sean sencillas, claras y con la mínima información necesaria, para hacer la experiencia del usuario lo más fácil posible.

También se desea implementar un sistema de votación en grupo, que permita a usuarios en diferentes localizaciones tomar una decisión en conjunto.

1.4 Metodología

Para el desarrollo de este proyecto, se ha optado por una metodología de desarrollo en cascada, dividiendo la carga de trabajo en diferentes tareas de desarrollo secuenciales y cada una dependiente de la anterior. El total de estas tareas suman unas 320 horas de trabajo. En los anexos se incluye el diagrama de Gantt completo, y las tareas se describen a seguir:

1. Estudio de Requisitos
 - Duración estimada: 10 días
 - Entrada: Usuarios de la aplicación
 - Salida: Especificación de requisitos
1. Descripción del caso de estudio
 - Duración estimada: 3 días
 - Entrada: Usuarios de la aplicación
 - Salida: Documento de descripción de la aplicación
2. Estudio de las herramientas existentes
 - Duración estimada: 4 días
 - Entrada: Documento de descripción de la aplicación
 - Salida: Listado de las alternativas existentes
3. Elicitación de requisitos
 - Duración estimada: 3 días
 - Entrada: Documento de descripción de la aplicación
 - Salida: Especificación de requisitos

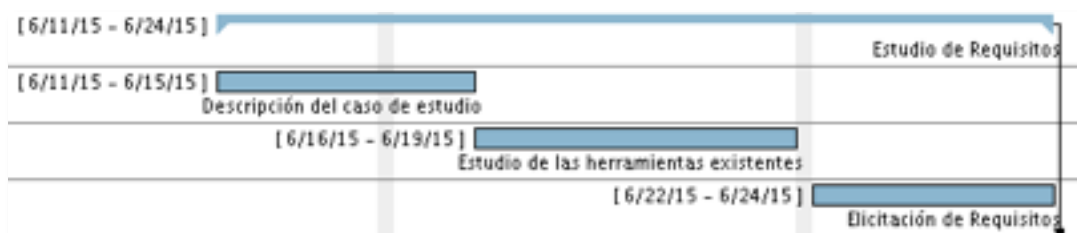


Figura 1.1 Detalle del diagrama de Gantt para la tarea 1

2. Análisis

- Duración estimada: 7 días
- Entrada: Especificación de requisitos
- Salida: Diagrama de clases de análisis y bocetos de la interfaz

1. Definición de clases del dominio

- Duración estimada: 2 días
- Entrada: Especificación de requisitos
- Salida: Diagrama de clases de análisis

2. Definición de las interfaces de usuario

- Duración estimada: 5 días
- Entrada: Especificación de requisitos
- Salida: Bocetos de las interfaces de usuario

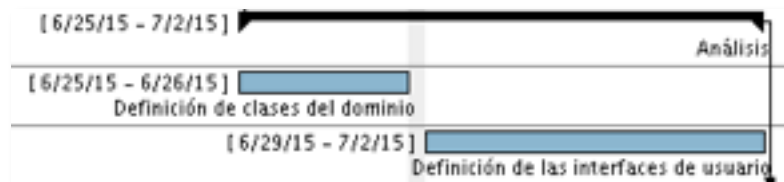


Figura 1.2 Detalle del diagrama de Gantt para la tarea 2

3. Diseño

- Duración estimada: 8 días
- Entrada: Diagrama de clases de análisis y bocetos de la interfaz
- Salida: Diagrama de clases de diseño y diseño de la interfaz

1. Definición de las clases externas

- Duración estimada: 4 días
- Entrada: Diagrama de clases de análisis
- Salida: Diagrama de clases de diseño

2. Diseño detallado de interfaces

- Duración estimada: 4 días
- Entrada: Bocetos de la interfaz de usuario
- Salida: Diseño de la interfaz de usuario

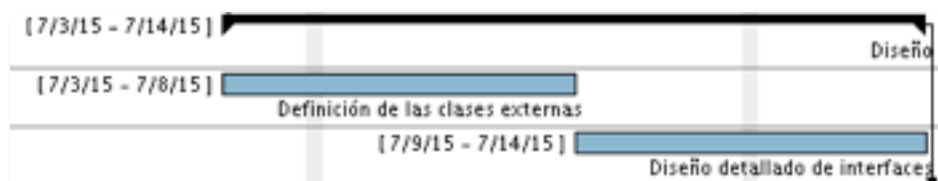


Figura 1.3 Detalle del diagrama de Gantt para la tarea 3

4. Implementación

- Duración estimada: 15 días
 - Entrada: Diagrama de clases de diseño y diseño de la interfaz
 - Salida: Código de la aplicación
1. Diseño de la base de datos
 - Duración estimada: 1 día
 - Entrada: Diagrama de clases de diseño
 - Salida: Diagrama de la base de datos
 2. Implementación de la base de datos
 - Duración estimada: 1 día
 - Entrada: Diagrama de la base de datos
 - Salida: Estructura MySQL de la base de datos
 3. Implementación de la ruleta
 - Duración estimada: 5 días
 - Entrada: Diagrama de clases de diseño
 - Salida: Código en JavaScript de la ruleta
 4. Implementación de la conexión con Facebook
 - Duración estimada: 2 días
 - Entrada: Diagrama de clases de diseño y código en JavaScript de la ruleta
 - Salida: Código de la clase de conexión entre la ruleta y Facebook
 5. Implementación de las interfaces de usuario
 - Duración estimada: 6 días
 - Entrada: Diseño de la interfaz de usuario
 - Salida: Código de las interfaces de usuario



Figura 1.4 Detalle del diagrama de Gantt para la tarea 4

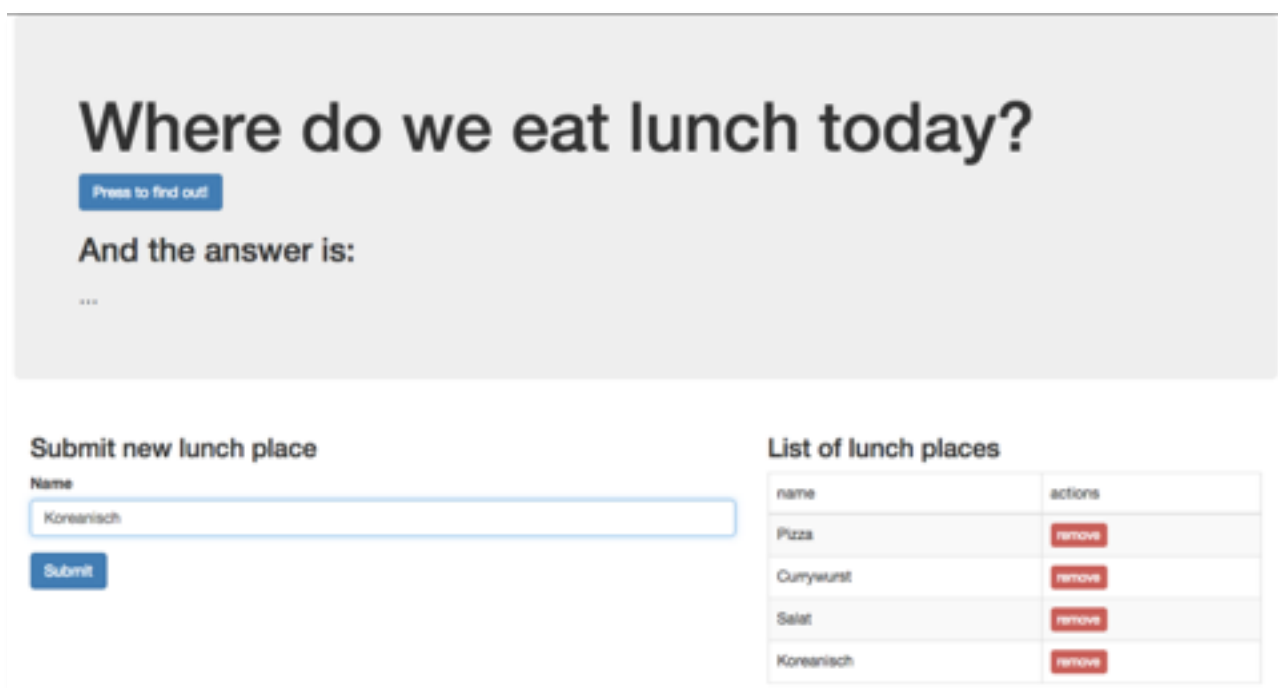
2. Contexto Tecnológico

2.1 Aplicaciones similares

Una de las principales motivaciones para este proyecto viene justamente de la falta de alternativas presentes. Las herramientas disponibles son demasiado limitadas o tienen una utilización difícil y tediosa, como puede ser el caso de las aplicaciones que permiten introducir opciones para luego elegir una al azar.

Haremos un breve repaso de las alternativas existentes, que en su mayoría siguen los rasgos descritos en el párrafo anterior:

2.1.1 The Techdev Lunch Deciding Tool



List of lunch places	
name	actions
Pizza	remove
Currywurst	remove
Salat	remove
Koreanisch	remove

Figura 2.1 The Techdev Lunch Deciding Tool

La aplicación desarrollada por el equipo alemán de Techdev tiene una interfaz sencilla, pero esconde un impresionante nivel de innovación al ser implementada en una Raspberry Pi utilizando tecnologías como OpenResty y knockout.js.

La principal desventaja de esta aplicación es la necesidad de introducir las opciones manualmente, lo que entorpece bastante lo que debería ser un proceso sencillo.

2.1.2 What Should I Eat?

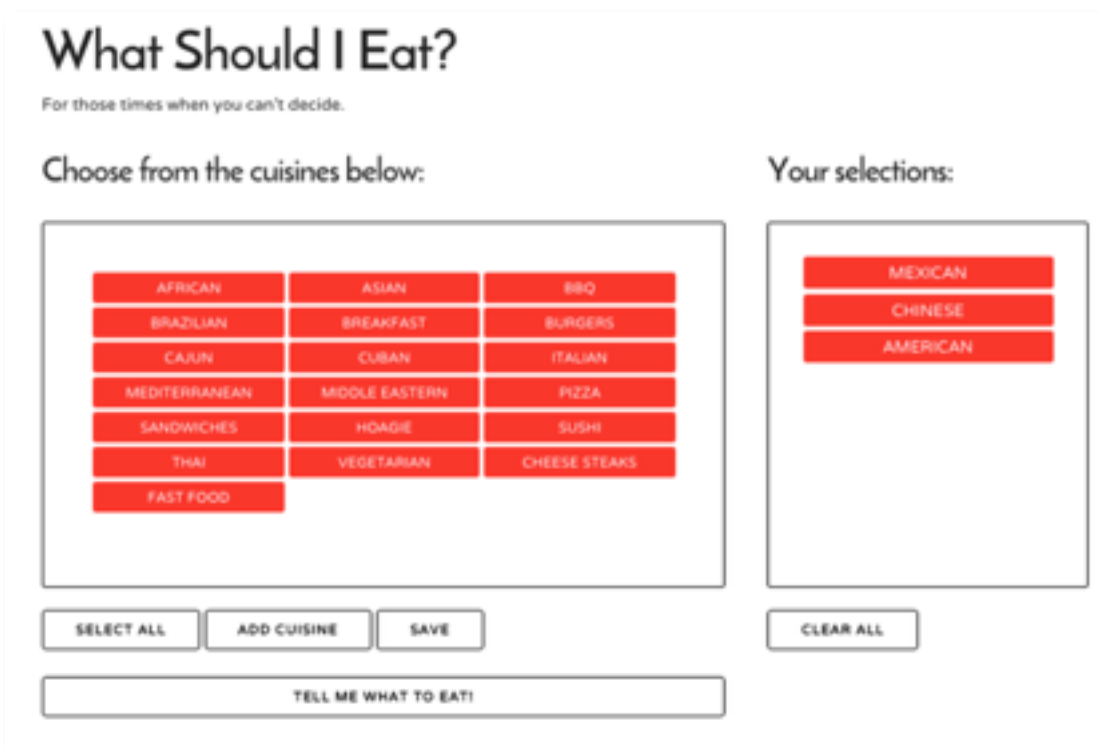


Figura 2.2 What Should I Eat?

What Should I Eat? es una aplicación con un objetivo bastante menos específico que las demás, ya que sirve para decidir qué tipo de comida comer, sobretodo cocinas internacionales.

Puede ser personalizada para utilizar sitios en concreto, ya que se pueden añadir opciones, pero una de las principales ventajas de esta aplicación es justamente el hecho de que tiene opciones predefinidas, lo que hace el proceso sencillo y eficaz.

2.1.3 Wheel Decide



Figura 2.3 Wheel Decide

Wheel Decide es una alternativa bastante similar a la descrita en el punto anterior, ya que permite decidir entre diferentes tipos de comida, y es completamente personalizable.

Lo que la distingue de las demás es el atractivo visual de la ruleta y el hecho de que los usuarios pueden personalizarla y hacer públicas sus versiones de la ruleta.

Eso hace que la aplicación cuente con un gran número de opciones predefinidas, pero aún así las opciones no se adaptan a cualquier usuario fácilmente, ya que rara vez habrá una ruleta con los restaurantes locales al usuario disponible.

2.1.4 Wheel of Lunch



Figura 2.4 Wheel of Lunch

Wheel of Lunch es definitivamente la alternativa que más se asemeja a este proyecto, ya que tiene un funcionamiento muy similar al propuesto.

El contenido de la ruleta se ajusta al código postal introducido por el usuario, y la información es obtenida a través de la API de Yelp, que también provee calificaciones de los establecimientos.

2.2 Herramientas necesarias

En este capítulo, listaremos los lenguajes de programación, librerías y herramientas necesarias para el desarrollo del proyecto.

• HTML5

Además de la funcionalidad básica de HTML, inherente al proyecto al ser una aplicación web, se destaca la utilización de la etiqueta canvas.

La etiqueta canvas aparece en la versión 5 de HTML y permite mostrar una imagen en la web y animarla o manipularla con JavaScript. La parte más importante de la interfaz de esta aplicación – la ruleta – está dibujada en una etiqueta canvas.

• JavaScript

JavaScript es un lenguaje de lado cliente interpretado, y que permite la implementación de clases a partir de prototipos. Inicialmente relegado a hacer pequeñas tareas en la presentación de páginas web, hoy en día es ampliamente utilizado en servidores, a través de node.js y también permite la transferencia de partes lógicas de la aplicación al lado cliente, lo que reduce el tiempo de carga de las páginas y mejora la experiencia del usuario. La mayor parte de la lógica de esta aplicación y su disposición en pantalla corre a cargo de JavaScript.

• CSS

CSS (Cascading Style Sheets) es una tecnología que permite describir en un documento la apariencia del contenido de una página web. El objetivo de esta tecnología es separar el contenido de una página de su aspecto, para hacer los cambios en una parte u otra más sencillos.

En esta aplicación, la mayor parte del CSS viene definido por Bootstrap, herramienta que describiremos más adelante.

• jQuery

jQuery es una librería JavaScript bastante extendida en el mercado. Se basa en hacer JavaScript más sencillo, lo que le hace actuar como una especie de recubrimiento para el mismo.

Una de sus mejores características es la facilidad de utilización, sobretodo con sus selectores y llamadas AJAX (Asynchronous JavaScript and XML), que hacen la integración con lenguajes del lado del servidor muy sencilla.

La utilización de jQuery en este proyecto es elevada, desde la creación de elementos dinámicos de la GUI (Graphic User Interface) hasta la comunicación con el lado servidor, a través de llamadas AJAX.

• **Facebook Graph API**

Graph API es, en pocas palabras, la puerta de acceso a toda la información en Facebook, o la gran mayoría.

Se puede acceder desde algunos SDK diferentes, como JavaScript o PHP, con diferentes peticiones, desde actualizaciones de perfil hasta información sobre páginas registradas.

Para hacer peticiones a Graph, se debe incluir un identificador de acceso, que se puede obtener creando una aplicación en el centro de desarrolladores de Facebook.

• **Google Maps JavaScript API**

Google geocoding provee una serie de servicios relacionados con localización, y es una parte del servicio Google Maps.

Es una parte fundamental de uno de los segmentos de la aplicación, ya que permite enviar un código postal y la API (Application Program Interface) responde con una localización representada en latitud y longitud.

• **Bootstrap**

Bootstrap es una herramienta que facilita enormemente la construcción de interfaces de usuario.

Sencilla de utilizar, ya que básicamente consiste en una hoja de estilos, es una herramienta ideal para construir una GUI atractiva rápidamente.

3. Caso de estudio: Wheely Tasty

3.1 Descripción y funcionamiento

Wheely Tasty es una aplicación que tiene como objetivo simplificar el proceso de toma de decisión en una situación cotidiana. Por ese motivo, la aplicación debe ser lo más sencilla posible.

La aplicación inicialmente pide autorización al usuario para acceder a su localización. Las opciones de localización se suelen guardar en el navegador, haciendo futuras visitas aún más fluidas.

A seguir, utiliza esa localización para obtener información sobre restaurantes cercanos al usuario, a través de la API de Facebook (Graph API). Una vez los datos son recibidos, los restaurantes se ordenan por proximidad, y se colocan en una ruleta – la ordenación se hace debido al hecho de que la ruleta acepta un máximo de 20 restaurantes, y se intentan incluir los restaurantes más cercanos al usuario.

Una vez la ruleta está dibujada, el usuario simplemente tiene que hacer clic sobre ella para hacerla girar. Cuando ésta se para, un panel se abre mostrando información sobre el restaurante, como: horario de apertura, dirección, teléfono, descripción, etc. La información que se dispone al usuario es recogida también de Facebook, con lo cual la disponibilidad de dicha información queda sujeta a la dedicación de los administradores de las páginas de los restaurantes en publicarla.

En el caso de que el usuario decida hacer su elección entre sitios cercanos a otro lugar, que no su ubicación, puede utilizar la opción de búsqueda por código postal, que hace que el sistema transforme el código postal en una ubicación y repita el proceso descrito en el caso anterior.

Finalmente, si un grupo de personas quiere decidir dónde comer, pueden utilizar la opción de crear un grupo para ello. Cuando un grupo se crea, el usuario recibe un identificador, que debe transmitir a los demás participantes del grupo. Cada uno de los miembros del grupo debe acceder a la opción de votar en un grupo, y decidir si quiere votar por un restaurante cercano a su localización o un código postal introducido por él. El siguiente giro de la ruleta se registrará como un voto del miembro en cuestión al restaurante seleccionado por la ruleta.

El resultado de la votación del grupo se puede visualizar accediendo a la opción ver resultados, en el menú de grupos.

3.2 Especificación de Requisitos

Los requisitos de la aplicación serán especificados a través del estándar IEEE 830. Dichos requisitos se detallan a continuación:

3.2.1 Requisitos funcionales

• **Búsqueda por ubicación**

El sistema muestra restaurantes cercanos a la ubicación del usuario para su elección.

- ❖ Entrada: El usuario acepta el acceso a su ubicación por parte del sistema.
- ❖ Proceso: El sistema recoge información de restaurantes cercanos al usuario.
- ❖ Salida: Ruleta con restaurantes cercanos al usuario para su elección.

• **Búsqueda por código postal**

El sistema muestra restaurantes cercanos al código postal introducido por el usuario para su elección.

- ❖ Entrada: El usuario pulsa sobre la opción de búsqueda por código postal e introduce el mismo.
- ❖ Proceso: El sistema recoge información de restaurantes cercanos al código postal introducido.
- ❖ Salida: Ruleta con restaurantes cercanos al código postal introducido para la elección por parte del usuario.

• **Creación de un grupo de votación**

El sistema crea un grupo para almacenar distintos votos por parte de sus miembros.

- ❖ Entrada: El usuario abre el menú de opciones de grupo y pulsa sobre la opción de crear un grupo.
- ❖ Proceso: El sistema registra un nuevo grupo de votación.
- ❖ Salida: Identificador del grupo.

• **Votación en un grupo**

El sistema registra un voto en el grupo en cuestión.

-
- ❖ Entrada: El usuario abre el menú de opciones de grupo y pulsa sobre la opción de votar en un grupo. Luego introduce el identificador y elige si quiere utilizar su ubicación o introducir un código postal.
 - ❖ Proceso: El sistema registra un voto del usuario al grupo en cuestión.
 - ❖ Salida: Resultados del grupo.

• Visualización de resultados de un grupo

El sistema muestra los resultados del grupo en cuestión.

- ❖ Entrada: El usuario abre el menú de opciones de grupo y pulsa sobre la opción de ver los resultados de un grupo. Luego introduce el identificador del grupo.
- ❖ Proceso: El sistema recoge los votos para el grupo en cuestión.
- ❖ Salida: Resultados del grupo.

3.1.2 Requisitos de diseño

Los requisitos de diseño, que se detallan a continuación, van de acuerdo con el rol facilitador de la aplicación.

- ❖ Interfaz sencilla, con colores neutrales, que muestre la información de manera clara.
- ❖ Facilidad de interacción, buscando la mínima intervención del usuario para obtener un resultado satisfactorio.

3.1.3 Requisitos tecnológicos

Los requisitos tecnológicos de esta aplicación son bastante sencillos, al tratarse de una aplicación basada en HTML5 y JavaScript, que puede ser ejecutada por cualquier navegador moderno.

4. Análisis

4.1 Diagrama de clases

En este capítulo, haremos un análisis del funcionamiento deseado de la aplicación, sin entrar en detalles de implementación.

Para tener una idea de la disposición de elementos en nuestro sistema, se adjunta un diagrama de clases:

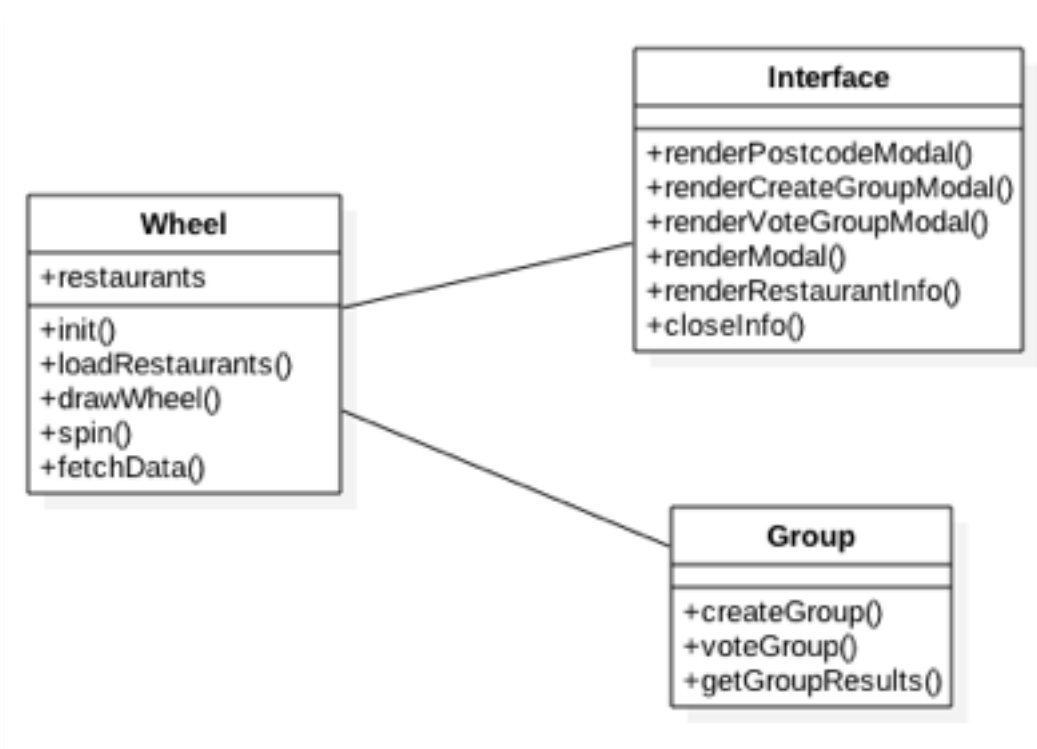


Figura 4.1 Diagrama de clases – análisis

A partir de los requisitos de la aplicación, se han delineado las clases que figuran en el diagrama anterior, y que se describen a seguir:

- **Wheel:** es la clase principal de la aplicación, maneja lo relacionado a la ruleta, desde la obtención de la ubicación del usuario a la recogida de información de Facebook.
- **Interface:** es la clase que se encarga de dibujar los diferentes elementos de la aplicación en la pantalla y de ocultarlos cuando necesario.
- **Group:** es la clase que se encarga de gestionar la creación de los grupos, las votaciones y la muestra de resultados.

4.2 Interfaces de usuario

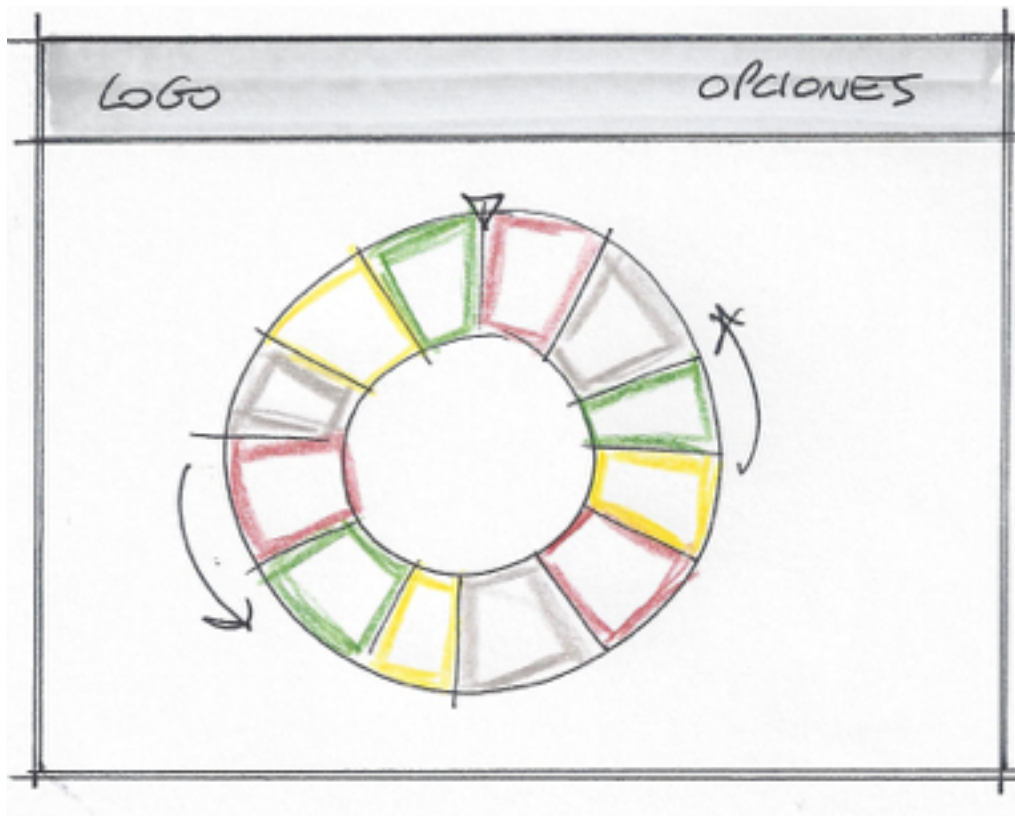
Para cumplir el objetivo principal de la simplicidad, se han esbozado interfaces sencillas, que permiten al usuario usar la aplicación sin cumplir ninguna especie de curva de aprendizaje.

Las principales se describen a continuación:

- Pantalla principal

Cuando el usuario abre la aplicación, si concede acceso a su ubicación, se cargará directamente la ruleta con restaurantes cercanos a él, permitiendo que el usuario obtenga un resultado con apenas un clic.

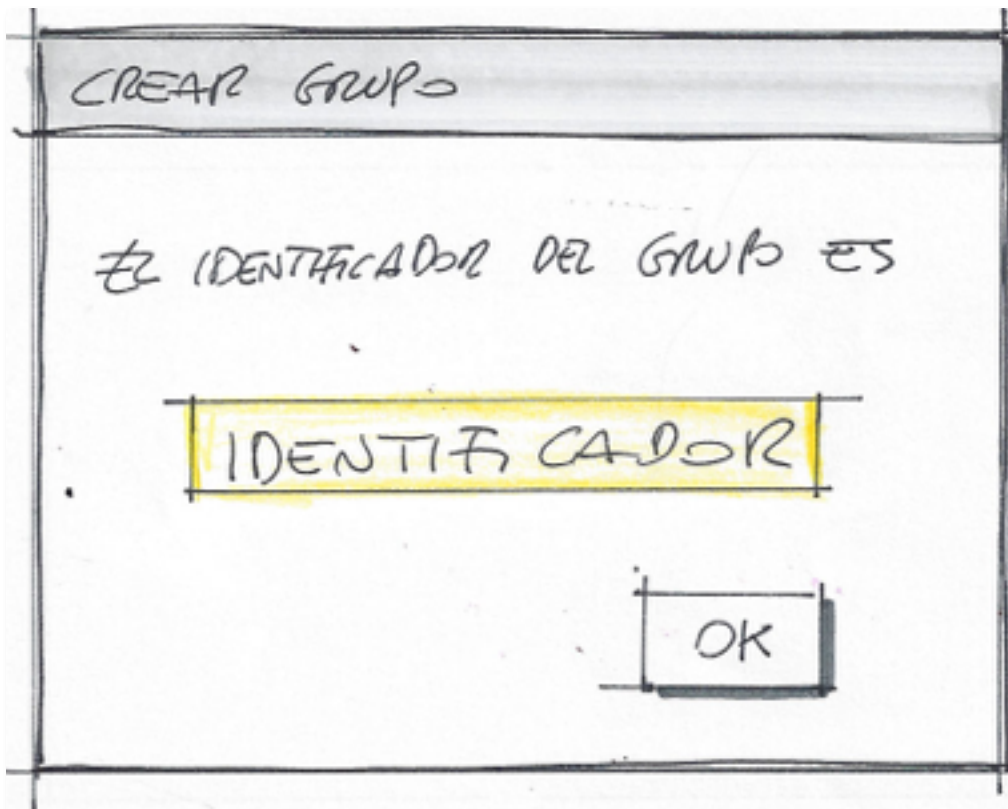
En caso de que desee acceder a las opciones, éstas se ubican al lado superior derecho de la pantalla, mayormente agrupadas en un menú para evitar la sobrecarga de la barra superior.



4.2 Boceto de la pantalla principal

- Ventana de creación de grupo

Para la creación de un grupo, el usuario simplemente debe pulsar sobre la opción pertinente, y una ventana se abrirá comunicándole el identificador del grupo. La ventana también indicará que el usuario debe comunicar ese identificador a los demás miembros del grupo para que éstos voten.



4.3 Boceto de la ventana de creación de grupo

- Ventana de votación en grupo

Para votar en un grupo, el usuario debe pulsar sobre la opción correspondiente, que abrirá una ventana con un cuadro de texto para la introducción del identificador del grupo, otro para el nombre del usuario y un grupo de controles radio para que el usuario elija si quiere votar por un restaurante cerca suyo o de un código postal introducido por él.

VOTAR EN GRUPO

ID. GRUPO

NOMBRE

UTILIZAR:

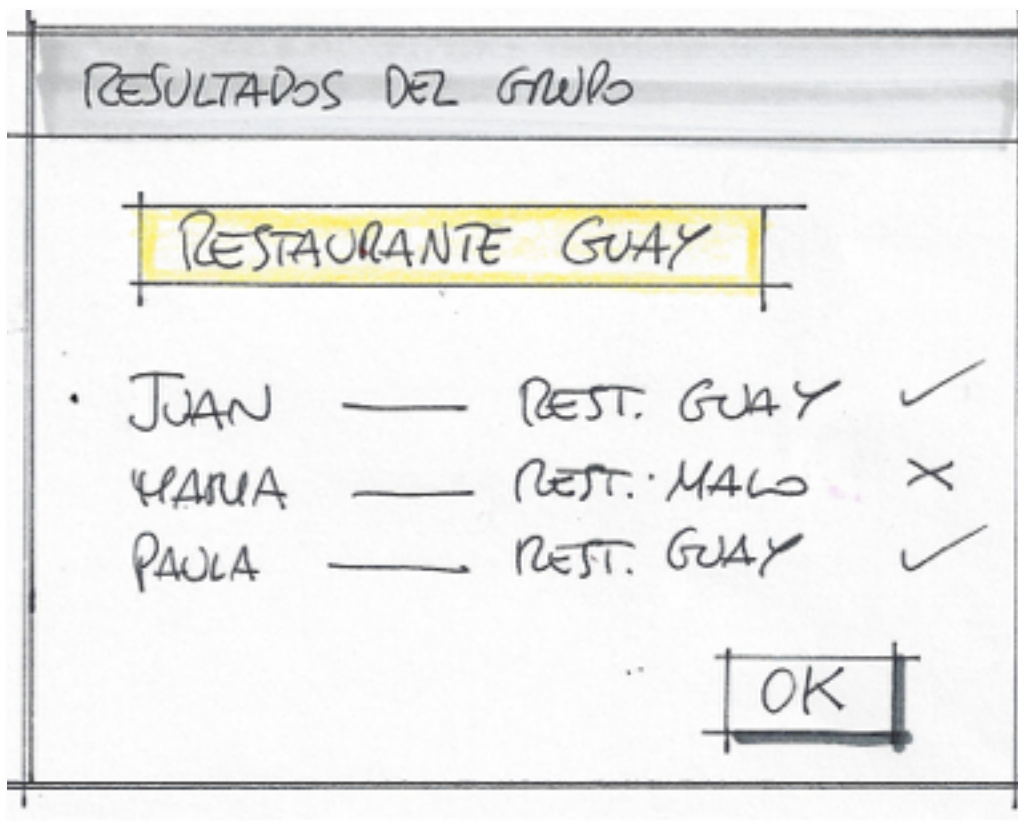
UBICACIÓN CÓDIGO POSTAL

VOTAR

4.4 Boceto de la ventana de votación en grupo

- Ventana de resultados de grupo

Para ver los resultados de un grupo, el usuario debe pulsar sobre la opción relevante y se abrirá una ventana con el restaurante elegido en destaque y una lista de los miembros y sus votos abajo. En caso de empate, esa ventana mostrará un botón de desempate que hará que el sistema elija uno de los sitios ganadores al azar.



4.5 Boceto de la ventana de resultados de grupo

5. Diseño

5.1 Diagrama de clases

En este capítulo, se desarrollarán las ideas extraídas de la fase de análisis, de modo que tendremos directrices más concretas de cómo implementar nuestro proyecto.

A seguir, se adjunta un diagrama de clases más profundizado que el de análisis, que ya tiene en cuenta funciones auxiliares que se necesitarán e integraciones con servicios externos.

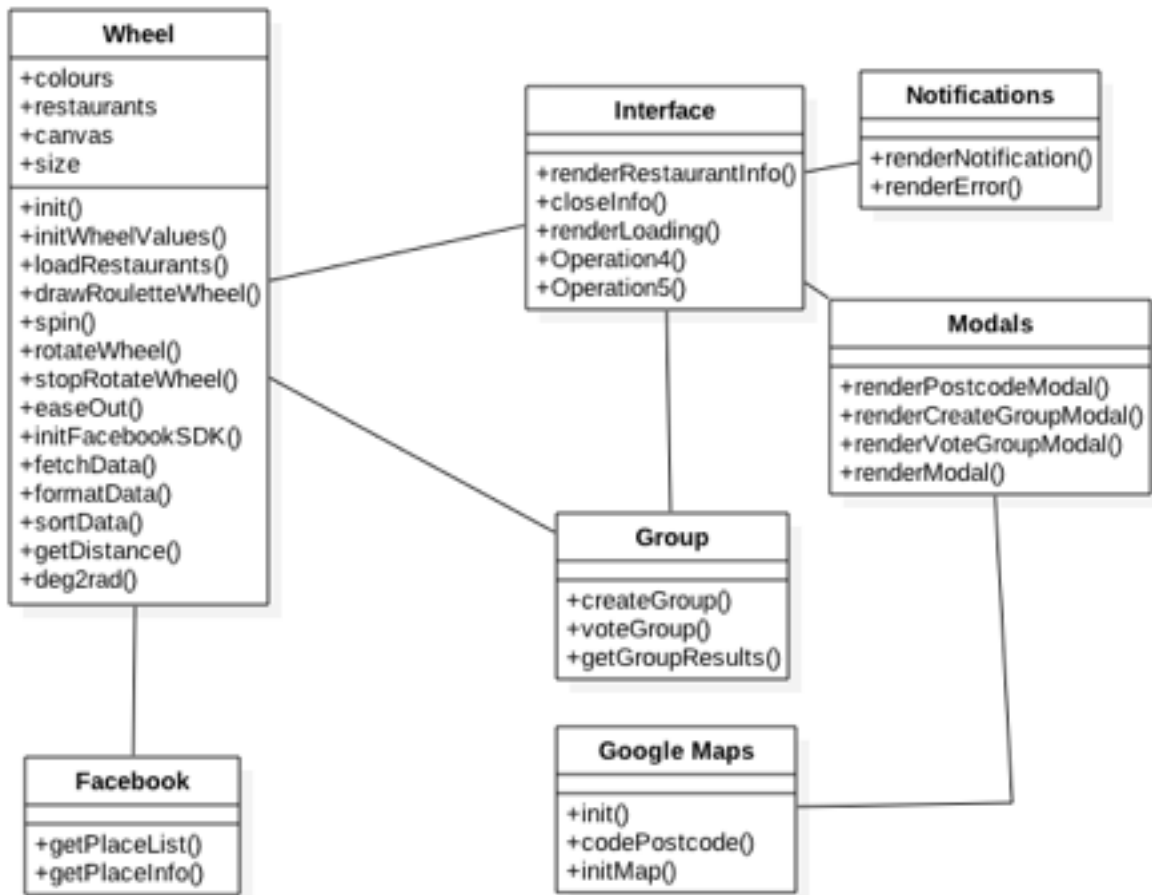


Figura 5.1 Diagrama de clases - diseño

Haremos un repaso de las clases resultantes del proceso de diseño, con sus atributos y métodos:

- **Wheel:** esta clase contiene la lógica de la ruleta, bien como la conexión con la API de Facebook. Además de lo expuesto durante el análisis, se necesitarán una serie de funciones auxiliares para tratar la información que se expone y hacer que la ruleta gire, éstas se detallarán en la fase de implementación.
- **Interface:** es la clase que se encarga de dibujar las interfaces de usuario y gestionar el envío de los datos introducidos por el usuario a las clases que gestionan la lógica de la aplicación.
- **Modals:** es una clase auxiliar a la clase Interface que se encarga de dibujar las ventanas modales, que son responsables por todas las interacciones con el usuario.
- **Notifications:** esta clase, también auxiliar a Interface, genera y muestra notificaciones al usuario, tanto de error como de confirmación.
- **Group:** se encarga de la lógica necesaria para gestionar las operaciones de grupo, ya que esta clase se conecta al lado servidor que accede a la base de datos con la información de los grupos.
- **Google Maps:** esta clase contiene las funciones que hacen uso de la API de Google Maps.
- **Facebook:** esta clase hace las conexiones necesarias con la API de Facebook para extraer la información necesaria a nuestra aplicación.

5.2 Diseño de la interfaz

Como ya se ha descrito en el apartado anterior, la clase Interface se encargará de dibujar las interfaces de usuario.

En este proyecto se ha optado por el paradigma SPA (Single Page Application) que resulta en interfaces fluidas y rápidas, que mejoran la experiencia del usuario. Para cumplir con ese paradigma, la aplicación se basará en una pantalla principal - donde se ubicará la ruleta - y las demás interacciones con el usuario se harán a través de ventanas modales.

Las ventanas modales las dibujará la clase auxiliar Modals y se pondrán por encima de la pantalla principal con jQuery, y se retirarán cuando necesario, evitando cargas de página para las operaciones del usuario. La clase Modals cuenta con un método auxiliar renderModal, que dibuja una ventana modal a partir de un objeto JavaScript de configuración. Los demás métodos que dibujan ventanas modales simplemente crean diferentes ficheros de configuración y a su vez llaman a renderModal.

Además, la clase auxiliar Notifications se encargará de mostrar notificaciones de confirmación y error al usuario.

6. Implementación

6.1 Base de datos

La base de datos para este proyecto es bastante pequeña, ya que los únicos datos que deben persistir son los de las votaciones de grupo.

Se adjunta un diagrama de la base de datos:

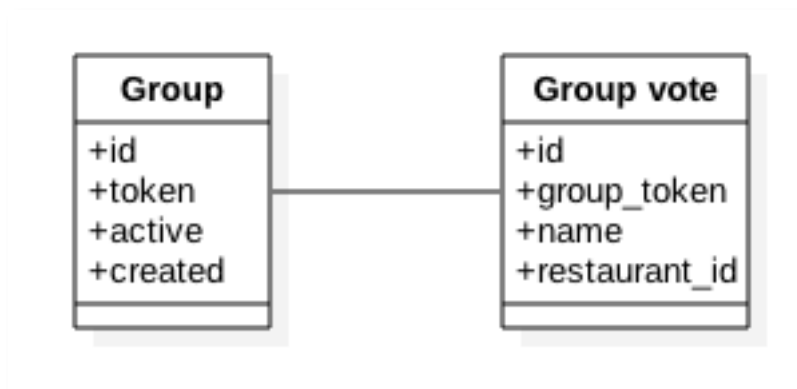


Figura 6.1 Diagrama de la base de datos

Conforme se observa en la figura anterior, la base de datos se compone de dos tablas, que se detallan a continuación:

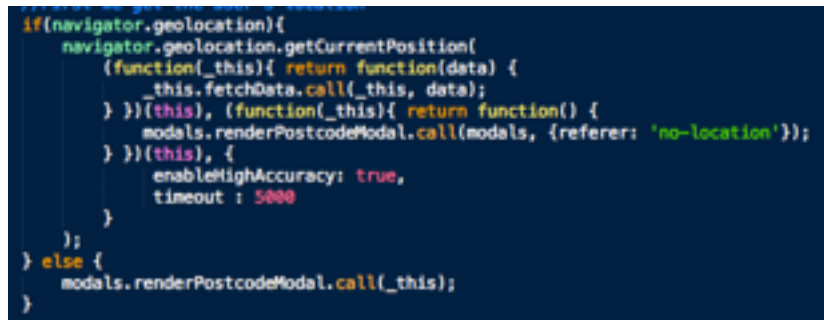
- Group: tabla que contiene un registro para cada grupo, con los siguientes atributos.
 - ❖ id: entero auto incremental que almacena el id interno del grupo.
 - ❖ token: campo varchar que almacena el identificador del grupo - cada identificador es una palabra única al azar.
 - ❖ active: campo de tipo booleano indica si el grupo está activo o no - ya que los identificadores siempre están en la base de datos, haya un grupo activo asignado a ellos o no.
 - ❖ created: almacena un dato de tipo datetime que indica cuando se creó el grupo. Ya que disponemos de un grupo de identificadores limitados, los grupos tienen una caducidad de 24 horas.
- Group vote: tabla que almacena los votos de los miembros de cada grupo, con los siguientes atributos.
 - ❖ id: entero auto incremental que almacena el id interno del voto.

-
- ❖ group_token: almacena el identificador del grupo al que corresponde el voto.
 - ❖ name: campo de tipo varchar que almacena el nombre del miembro al que corresponde el voto. Junto con el campo group token, forman una clave de unicidad en la tabla.
 - ❖ restaurant_id: almacena el identificador de la página de Facebook correspondiente al restaurante votado.

6.2 Funcionalidad

La funcionalidad del proyecto se explicará a través de las diferentes acciones posibles en la aplicación. Son las acciones que se listan a seguir:

- Elección de un restaurante a partir de ubicación: cuando la página se carga, la primera función que se ejecuta es la que requiere la ubicación actual del usuario. Eso se hace a través del método `getCurrentPosition`, del objeto `navigator.geolocation`, como se muestra en la figura siguiente.



```
if(navigator.geolocation){
  navigator.geolocation.getCurrentPosition(
    (function(_this){ return function(data) {
      _this.fetchData.call(_this, data);
    }})(this), (function(_this){ return function() {
      modals.renderPostcodeModal.call(modals, {referer: 'no-location'});
    }})(this), {
    enableHighAccuracy: true,
    timeout : 5000
  });
} else {
  modals.renderPostcodeModal.call(_this);
}
```

Figura 6.2 Parte de código de la función `loadRestaurants`.

Ese objeto es nativo al navegador, y se encarga de pedir la autorización del usuario para obtener su ubicación y devolverla a una función de retrollamada (callback). Si esa función se obtiene sin inconvenientes, se ejecuta la función `fetchData`, de la clase `Wheel`, que se encarga de conectarse a Facebook y recoger la información de los restaurantes cercanos - la conexión con Facebook se detallará en el siguiente apartado. En caso de error al obtener la localización, sea por motivos técnicos o porque el usuario decide no compartir su ubicación, se muestra una ventana para que el usuario introduzca su código postal para la búsqueda. Ese proceso se detalla en el siguiente punto.

Una vez la clase de conexión con Facebook devuelve los datos, estos se añaden al vector que almacena los restaurantes, y a cada uno se les añade una propiedad `distancia`, que almacena la distancia de cada uno a la ubicación del usuario. Esa distancia se calcula a través de la fórmula del Haversine, que permite conocer la distancia entre dos puntos de un globo sabiendo su longitud y latitud.

La figura que se adjunta a continuación muestra el filtrado de la información recibida desde Facebook y su inclusión en el vector de restaurantes.

```
Wheel.prototype.formatData = function (data) {
  var result = data.data;
  var restaurants = [];
  for (var i = 0; i < result.length; i++) {
    if (result[i].category.toLowerCase().indexOf('restaurant') != -1) {
      result[i].distance = this.getDistance(result[i].location.latitude, result[i].location.longitude);
      restaurants.push(result[i]);
    } else {
      for (var j = 0; j < result[i].category_list.length; j++) {
        if (result[i].category_list[j].name.toLowerCase().indexOf('restaurant') != -1) {
          result[i].distance = this.getDistance(result[i].location.latitude, result[i].location.longitude);
          restaurants.push(result[i]);
          break;
        }
      }
    }
  }
  //If we have less than 20 restaurants and data.paging.next is defined, call it and keep filling the array
  this.restaurants = this.restaurants.concat(restaurants);
  if ((this.restaurants.length < 20 && data.paging != undefined && data.paging.next != undefined) {
    this.fetchData({url: data.paging.next});
  } else {
    this.sortData();
    this.initWheelValues();
  }
}
```

Figura 6.3 Método formatData de la clase Wheel

Cuando los datos se encuentran en el vector, el vector se ordena en función de la propiedad distancia explicada en el párrafo anterior, para que cuando se haga el corte - la ruleta no admite más de 20 restaurantes - tengamos las opciones más cercanas al usuario.

A continuación, se colocan los restaurantes en la ruleta, a través de varios cálculos trigonométricos y las funciones stroke y fill, que permiten dibujar sobre el canvas.

Una vez ese proceso se termina, el sistema espera a que el usuario haga clic sobre la ruleta para hacerla girar. El giro se hace a través de timeouts y redibujar el canvas en diferentes ángulos. Para que el lugar elegido sea aleatorio, el tiempo de giro es un número aleatorio de milisegundos acotado a un tiempo de espera máximo.

Cuando la ruleta para, el nombre del restaurante elegido se muestra en el centro de la misma y se abre un panel en el lado izquierdo de la pantalla, mostrando información sobre el restaurante - la que esté disponible en su página.

- Elección de un restaurante a partir de ubicación: sea por la imposibilidad de determinar la ubicación del usuario o por su propia iniciativa, la búsqueda por código postal se hace desde una ventana modal, que muestra un cuadro de texto para que el usuario introduzca su código postal.

Una vez el usuario haya introducido su código postal, el sistema accede al método codePostcode, de la clase Google Maps, para obtener una ubicación estimada a partir del código postal del usuario y la API de Google Maps. La figura adjunta muestra el método en cuestión:

```

GoogleMaps.prototype.codePostcode = function (postCode) {
  geocoder.geocode( { 'address': postCode}, function(results, status) {
    if (status == google.maps.GeocoderStatus.OK) {
      if (results[0].geometry) {
        wheel.fetchData(
          {
            coords: {
              latitude: results[0].geometry.location.G,
              longitude: results[0].geometry.location.K
            },
            clear: true
          }
        );
      }
    } else {
      return false;
    }
  });
}

```

Figura 6.4 Método *codePostcode* de la clase *GoogleMaps*

Una vez la API de Google Maps envía la localización correspondiente, la retrollamada llama al método de *fetchData* de la clase *Wheel*, y el proceso sigue igual que el del apartado anterior.

- Creación de un grupo: cuando el usuario pulsa sobre la opción de crear un grupo, la clase *Group* hace una llamada AJAX al lado servidor para obtener un identificador de grupo. El sistema a su vez, a cada vez que requiere un grupo nuevo, hace un barrido de la tabla *group* y desactiva los grupos que tienen más de 24 horas. Después de ese proceso, devuelve un identificador de grupo y marca el grupo en cuestión como activo en la BD.

La figura a seguir muestra el proceso de eliminación de los grupos caducados:

```

function group_cleanup() {
  global $dbh;

  $cutting_date = date_create('now');
  $cutting_date = date_sub($cutting_date, date_interval_create_from_date_string('1 day'));
  $stmt = $dbh->prepare("SELECT * FROM 'group'");
  $stmt->execute();
  $groups = $stmt->fetchAll(PDO::FETCH_OBJ);
  $ids_to_clear = array();
  for ($i = 0; $i < count($groups); $i++) {
    $group = $groups[$i];
    if ($group->active == 1 && $group->created != NULL) {
      if (strtotime($group->created) < $cutting_date->getTimestamp()) {
        $ids_to_clear[] = $i;
      }
    }
  }
  //first we deactivate the groups
  $stmt = $dbh->prepare("UPDATE 'group' SET 'active' = NULL, 'created' = NULL WHERE 'id' IN (" . implode(', ', $ids_to_clear) . ");");
  $stmt->execute();

  //then we delete the votes attached to them
  $stmt = $dbh->prepare("DELETE FROM 'group_vote' WHERE 'id' IN (" . implode(', ', $ids_to_clear) . ");");
  $stmt->execute();
}

```

Figura 6.5 Método *group_cleanup*

Cuando el lado servidor devuelve el identificador de grupo, el lado cliente se encarga de generar una ventana modal que muestra esa información al usuario, bien como instrucciones de cómo proceder a la votación.

- Votación en un grupo: para la votación en un grupo el sistema muestra al usuario una ventana modal para que éste introduzca el identificador de su grupo, su nombre, y si desea votar por un sitio cercano a su ubicación o a un código postal introducido por él. Una vez envíe esa información, la ruleta se rellenará con los restaurantes pertinente a su elección, y hará un giro, cuyo resultado se enviará a través de una llamada AJAX al lado servidor, para que éste registre el voto en la BD.

La figura adjunta a continuación muestra la estructura normal de una llamada AJAX:

```
Group.prototype.registerVoteGroup = () {
  $.ajax({
    url: 'ajax/registerVote.php',
    dataType: 'json',
    method: 'POST',
    success: function(data) {
      if (data.success !== undefined) {
        notifications.renderNotification('The vote has been registered successfully.');
```

Figura 6.6 Método registerVoteGroup de la clase Group

- Visualización de resultados del grupo: cuando el usuario pulsa sobre la opción de ver resultados del grupo, el sistema muestra una ventana modal que pide al usuario que introduzca el identificador de su grupo. Una vez el usuario lo haga y pulse el botón ver resultados, el sistema hará un petición al lado servidor y mostrará esos datos en la misma ventana modal.

En caso de que haya un empate, el sistema mostrará un botón de desempate, que hará que el sistema elija una de las opciones ganadoras al azar.

6.3 Integración con Facebook

La integración con Facebook es el aspecto fundamental de esta aplicación, ya que de ella deriva toda la información que se procesa.

Inicialmente, uno de los objetivos de este proyecto era mantener toda la lógica de la aplicación en el lado cliente, utilizando el SDK JavaScript de Facebook. Eso ha sido imposible, por el hecho de que en las llamadas a la Graph API, se debe incluir un identificador de acceso, que es secreto y particular a cada app de Facebook - para acceder a la Graph API una web debe registrar su aplicación en Facebook. Por ese motivo, se ha optado finalmente por utilizar el SDK PHP de Facebook.

Graph API acepta peticiones de diferentes tipos de información, en nuestro caso haremos dos tipos de peticiones:

- Petición de búsqueda: se puede especificar una cadena en concreto - en nuestro caso no lo hacemos - además de una serie de propiedades a la búsqueda. En nuestro proyecto hacemos uso sobretodo de la capacidad de indicar un punto central a la búsqueda y una distancia máxima desde ese punto. La figura a seguir muestra un ejemplo del formato de petición de búsqueda a Graph API:

```
$url = "https://graph.facebook.com/v2.3/search?q=";  
$url .= "&type=place";  
$url .= "&center=$lat,$lon";  
$url .= "&distance=50000";  
$url .= "&limit=5000";  
$url .= "&access_token=" . FACEBOOK_ACCESS_TOKEN;
```

Figura 6.7 Petición de búsqueda a Graph API

-
- Petición de información sobre una página: Graph API también permite peticiones de información sobre una página. Para ello, basta especificar el identificador de la página y la lista de campos que se quiere recibir. A seguir se puede observar el formato de una petición de información:

```
$url = 'https://graph.facebook.com/v2.3/.$id.?'  
$url .= 'fields='  
$url .= 'description,'  
$url .= 'about,'  
$url .= 'website,'  
$url .= 'picture.width(1024).height(1024),'  
$url .= 'location,'  
$url .= 'phone,'  
$url .= 'hours,'  
$url .= 'name,'  
$url .= '&access_token='.FACEBOOK_ACCESS_TOKEN;
```

Figura 6.8 Petición de información sobre una página

Graph API devuelve objetos con la información resultante de las peticiones, que a su vez se envían en formato JSON al lado cliente de la aplicación.

6.4 Interfaz de usuario

La interfaz de usuario está implementada casi en su totalidad con jQuery usando elementos de Bootstrap. A seguir detallamos las principales partes de la interfaz:

- Pantalla principal:



Figura 6.9 Pantalla principal de la aplicación

En esta pantalla, los elementos están descritos en el fichero HTML de la página, ya que son elementos que están presentes en todo momento en la aplicación.

El HTML está dividido en secciones, haciendo uso de las etiquetas `section` y `nav`, introducidas en HTML5 y que facilitan la organización del documento.

La figura a seguir muestra en detalle el menú de la esquina superior derecha:

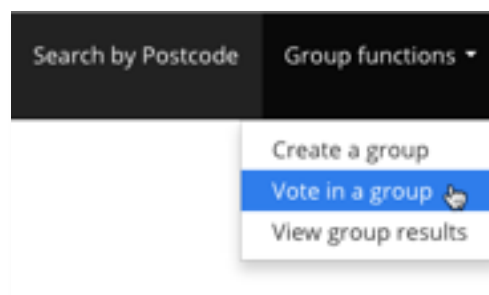


Figura 6.10 Detalle del menú

- Ventana de búsqueda por código postal:

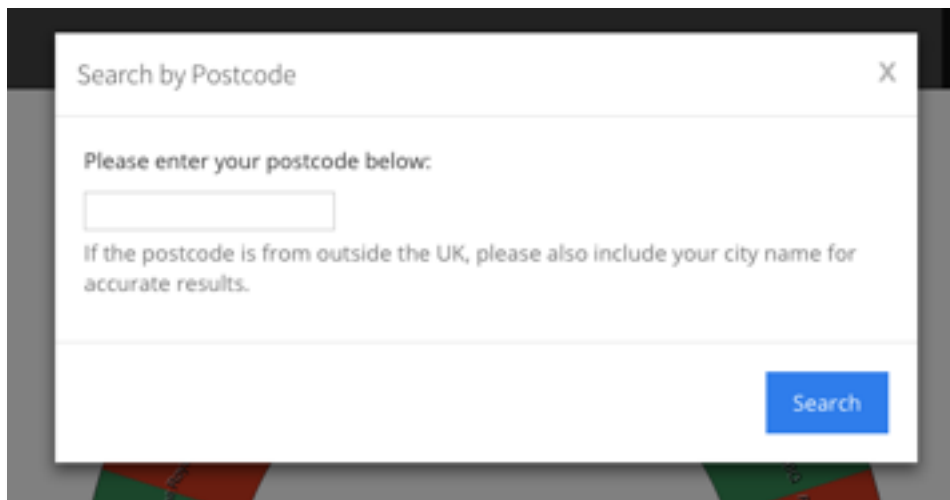


Figura 6.11 Ventana de búsqueda por código postal

La ventana de búsqueda por código postal, como todas las demás ventanas modales, se dibuja en la pantalla a través de jQuery, utilizando clases definidas en Bootstrap. Un ejemplo del código que genera una ventana modal sigue:

```
Modals.prototype.renderCreateGroupModal = function (_opt) {
  var opt = {
    group: false
  };

  opt = $.extend(opt, _opt);

  var modalClose = function() {
    $('.overlay').remove();
    $('.modal').remove();
  };

  var modalOpt = {
    title: 'Create a voting group',
    close: modalClose,
    buttonText: 'OK',
    actions: modalClose
  };

  var modalBody = $('<div>', {class: 'modal-body'});
  modalBody.append($('<p>', {text: 'The token for your group is:'}));
  modalBody.append($('<div>', {class: 'well', text: opt.group}));
  modalBody.append($('<span>', {class: 'help-block', text: 'To vote in this group, please select the Vote in a Group option from the menu and enter this token.'}));
  modalBody.append($('<span>', {class: 'help-block', text: 'Groups expire in 24 hours.'}));
  modalOpt.body = modalBody;

  this.renderModal(modalOpt);
}
```

Figura 6.12 Método renderCreateGroupModal de la clase Modals

Las demás ventanas modales se dibujan de la misma forma y se pueden ver en los anexos.

7. Conclusiones

7.1 Objetivos logrados

El objetivo fundamental de este proyecto era crear una herramienta que facilitase un proceso cotidiano que muchas veces hasta puede convertirse en una discusión o un problema - dentro de un grupo de personas sobretodo.

Ese objetivo fue logrado, ya que la herramienta en algunos casos resume la intervención del usuario a tan solo un clic, y mantiene una interfaz sencilla y grado de usabilidad elevado.

7.2 Objetivos no logrados

Desafortunadamente, algunos objetivos no fueron logrados, siendo ellos:

- Dotación de prioridades a diferentes restaurantes: una de las ideas iniciales era atribuir secciones de diferentes tamaños a los restaurantes en función de parámetros configurables como rango de precios, distancia, etc. La información recibida por parte de Facebook no fue lo suficientemente consistente como para soportar esa característica, ya que los dueños de los restaurantes se encargan de controlar esa información.
- Concentración de la lógica de la aplicación en el lado cliente: como ya mencionamos anteriormente, la necesidad de incluir un identificador de acceso en las peticiones a Facebook hace necesario el uso de un lenguaje en el lado servidor.

7.3 Ampliaciones

Como ampliaciones, recomendaría principalmente cambiar la API de utilización, posiblemente por alguna curada por una empresa, y no por parte de los propios usuarios, ya que así se puede obtener más información y de manera más consistente.

Una buena opción sería utilizar información de Trip Advisor, ya que se podría contar con datos como rango de precio, calificaciones, etc.

Con más información sobre los restaurantes, se puede añadir más funcionalidad a la aplicación, como adaptación del tamaño de las secciones - uno de nuestros objetivos iniciales - o búsqueda por tipo de restaurante (comida oriental, mexicana, etc.).

7.4 Comentario personal

Desde un primer momento, la elección de este proyecto viene de vivir la situación que la aplicación busca solucionar repetidamente en la oficina donde trabajo.

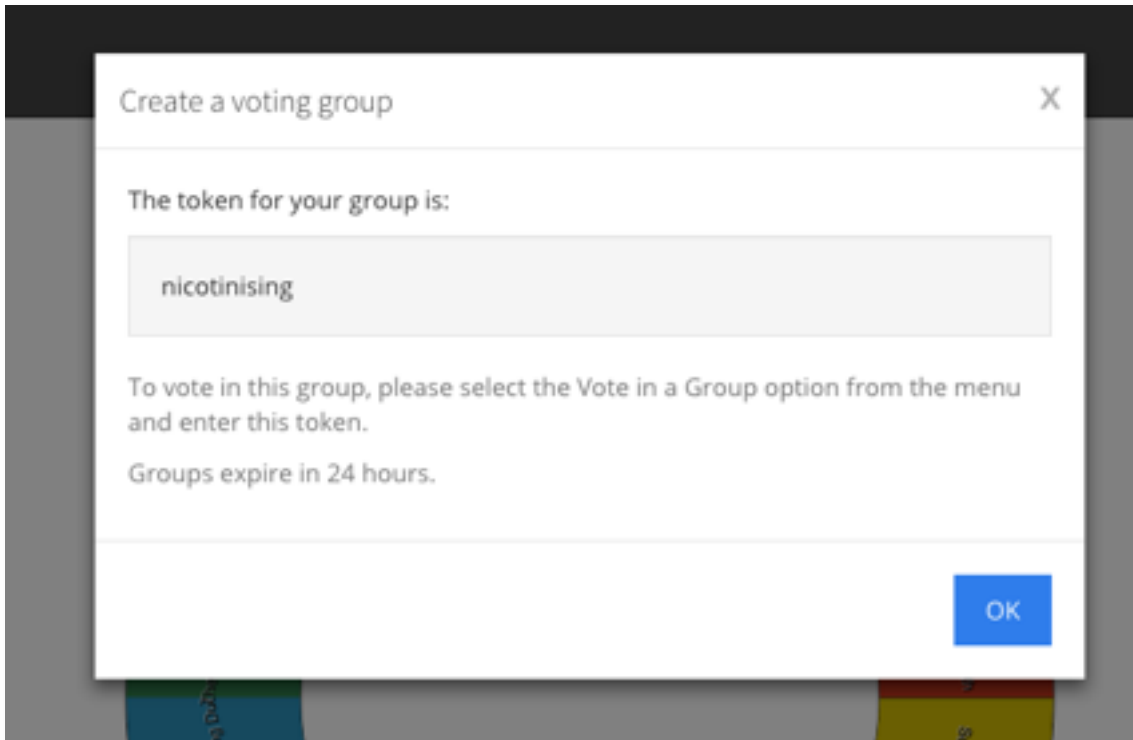
Me ha servido para aprender muchas cosas, desde como organizar una interfaz que funcione bien en una SPA hasta como extraer información de la API de Facebook de manera eficiente (y sin documentación - la sección de búsqueda de la API de Facebook no tiene referencia en sus docs).

Espero haber empezado el camino hacia una buena solución a ese problema, ya que definitivamente haré mi aplicación pública y la seguiré mejorando.

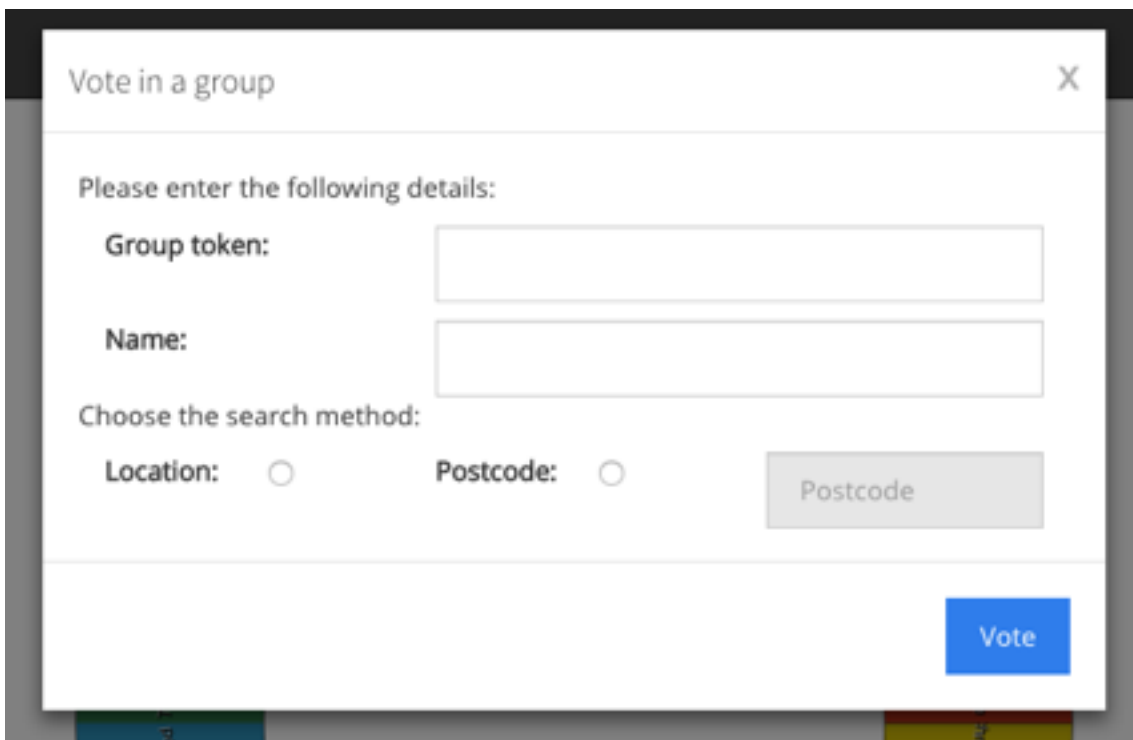
8. Bibliografía

- "The Techdev Lunch Decider"
 - URL: <http://blog.techdev.de/the-techdev-lunch-deciding-tool/>
 - Fecha de Visualización: 29/07/2015
- "What should I eat?"
 - URL: <http://andrewbowe.com/projects/lunch>
 - Fecha de Visualización: 29/07/2015
- "Wheel decide"
 - URL: <http://wheeldecide.com>
 - Fecha de Visualización: 29/07/2015
- "Wheel of Lunch"
 - URL: <http://wheelof.com/lunch>
 - Fecha de Visualización: 29/07/2015
- "About Javascript - MDN"
 - URL: http://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
 - Fecha de Visualización: 18/08/2015
- "How to calculate the distance between two latitude and longitude points"
 - URL: <http://stackoverflow.com/questions/27928/how-do-i-calculate-distance-between-two-latitude-longitude-points>
 - Fecha de Visualización: 15/07/2015

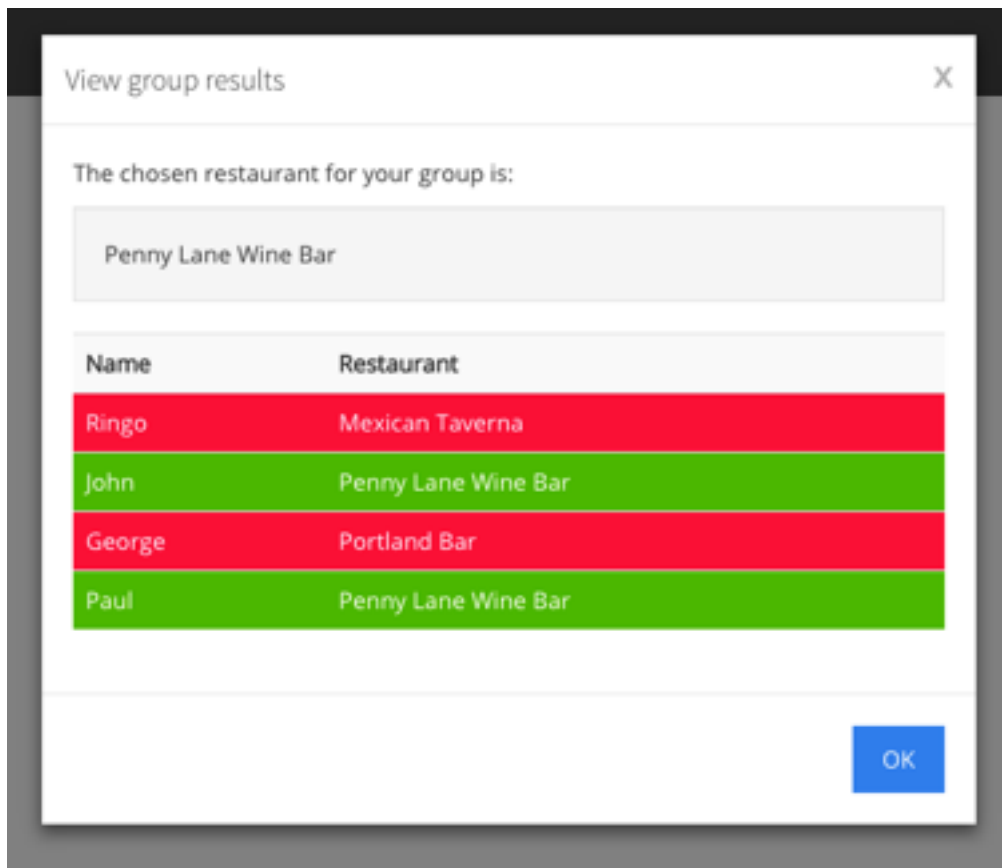
9. Anexos



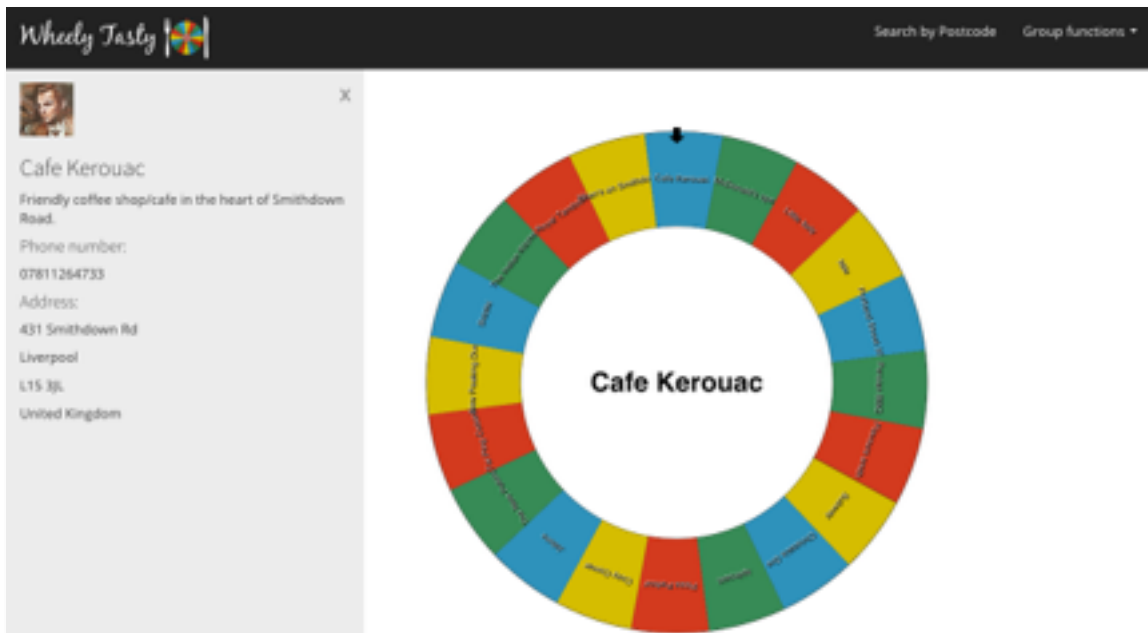
Ventana de creación de grupos



Ventana de votación en grupo



Ventana de resultados de grupo



Pantalla principal con la barra de información sobre el restaurante

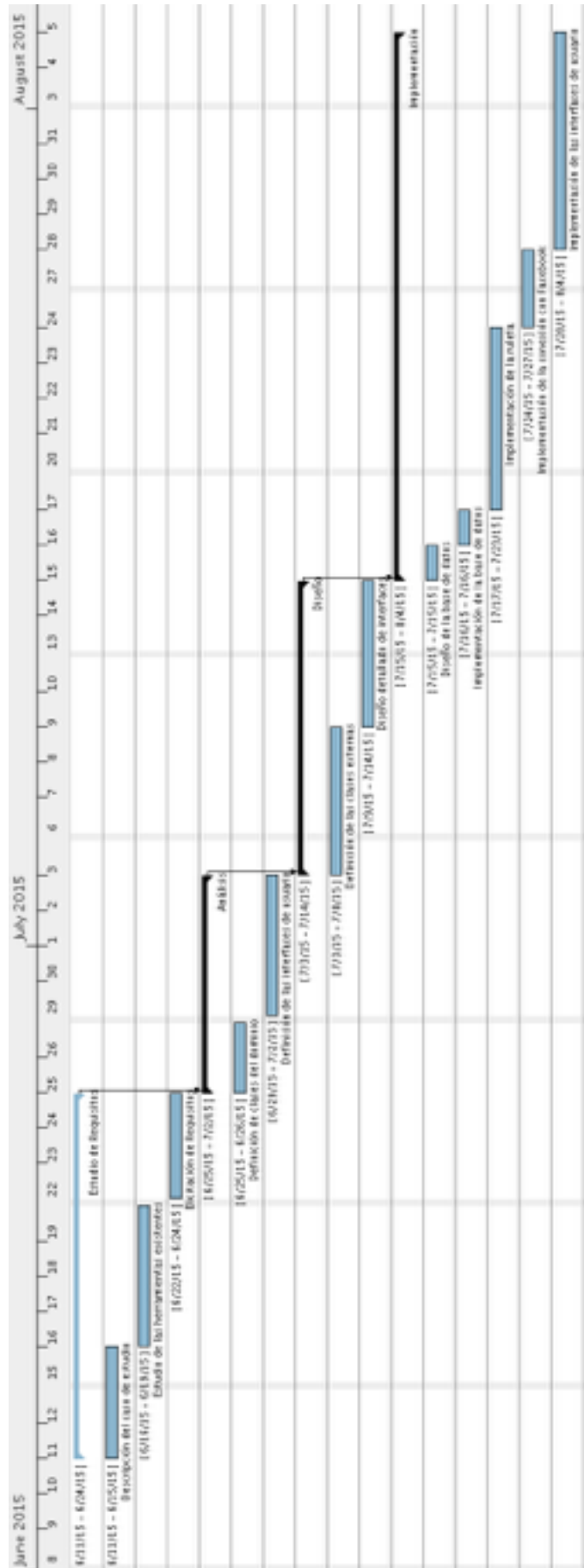


Diagrama de Gantt