Trabajo de Fin de Grado

Grado en Ingeniería Informática

Voting Garden: Sistema de televotación mediante el uso de dispositivos móviles

Voting Garden: Remote voting system for mobile devices

Ángel David Martín Rodríguez

La Laguna, 3 de septiembre de 2017

D. **Elena Sánchez Nielsen**, con N.I.F. 42.848.599-J profesor Titular de Universidad adscrito al Departamento de Nombre del Departamento de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Voting Garden: Sistema de televotación mediante el uso de dispositivos móviles"

ha sido realizada bajo su dirección por D. Ángel David Martín Rodríguez

con N.I.F. 51.147.341-X.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de septiembre de 2017

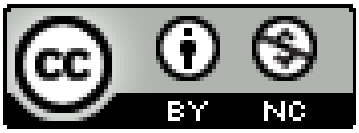
Agradecimientos

A mi tutora Elena, por tener tanta paciencia
con mi desastrosa organización.

A mi madre, por el sacrificio de financiarme los estudios, apoyarme
en toda la etapa universitaria, llenar de alegría
los momentos más difíciles a lo largo de la carrera y por
ser el mejor modelo de superación.

A Laura, Jose, las Beas, los Pablos y Victor, mis amigos, por
ser uno de los puntos de apoyo más importantes cuando los necesité
y por ayudarme a probar mi aplicación en un entorno real.

Licencia



Resumen

Las aplicaciones móviles representan, en su mayoría, soluciones globales a problemas diarios de los usuarios. En este trabajo se describe el desarrollo de Voting Garden, una aplicación web móvil para la creación de votaciones. El planteamiento inicial era su uso en un entorno específico, como es el Parlamento de Canarias, pero se desarrolló para adaptarse a cualquier ambiente.

Voting Garden nace con un objetivo principal: que cualquier persona pueda hacer uso de ella. Es por ello que la interfaz está adaptada para que tanto personas como organizaciones puedan beneficiarse de sus funcionalidades. Asimismo, el público al que va dirigido son aquellas personas que necesiten de un gestor y organizador de votaciones para cualquier tipo de decisión en grupo.

La versión funcional de la aplicación se lanzó para dispositivos Android, aunque las herramientas utilizadas permiten hacerlo también en dispositivos IOS y Windows Phone. Es necesaria una conexión permanente a Internet pues la base de datos se encuentra alojada en un servidor remoto. Además, se ha hecho uso de los servidores Iaas de la Universidad de La Laguna, lo que implica que para que la aplicación funcione correctamente el usuario debe estar conectado a la red de la universidad, ya sea por VPN o conectándose mediante usuario y contraseña a la red Wifi de la ULL.

Existen dos partes bien diferenciadas en Voting Garden, un cliente y un servidor. El cliente corresponde a la aplicación, y el servidor a las bases de datos y la comunicación mediante WebSockets entre usuarios. Ambos han sido desarrollados con el framework de JavaScript de código abierto AngularJS y con la herramienta Node.js. Para el cliente en especifico, se hizo uso del framework de desarrollo de aplicaciones web móviles Ionic.

Por último, se realizó un testeo en entorno real con nueve personas, tanto para buscar posibles fallos como para probar el desempeño de la aplicación en distintos terminales.

Palabras clave: Andorid, IOS, Whindows Phone, WebSocket, votaciones, aplicación web, framework, Node.js, Ionic, Iaas

Abstract

Mobile applications represent global solutions to daily users issues. This document describes the development of Voting Garden, a mobile web application for voting purposes. Its innitial approach was intended specifically for the parliament, yet it was developed to adapt to any background.

Voting Garden was created with a main goal: everyone should be able to use it. For this reason its interface is adapted to both people and organizations so they can benefit from its functionalies. Given this, the public intended to use it are those who seek for a voting manager in any type of group decision.

The functional version of this app was built on Android, although the tools used to perform it can be applied for IOS and Windows Phone. It is necessary a permanent connection to the internet, as the database used is stored on an online server. In addition, it has been used the Iaas server from the Universidad de La Laguna, which involves using a VPN connection to the university's network or conecting to it locally.

There are two well-defined parts in the development of Voting Garden, a client and a server. The client corresponds to the application and the server to the databases and communication between users through WebSockets. Both have been coded using the open source JavaScript framework AngularJs and NodeJs tool. Specifictally in what the client concerns, it has been used Ionic, a framework intended to develop mobile web applications.

Hence, nine people helped testing the app on a real environment, both to detect possible errors and test the performance of the application on different platforms.

Keywords: Andorid, IOS, Whindows Phone, WebSocket, vote, aplicación web, framework, Node.js, Ionic, Iaas

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

El origen del concepto aplicación móvil podría situarse en los años 90, con los primeros dispositivos que poseían aplicaciones como juegos, calendario o agenda. Estas eran muy básicas y los fabricantes no permitían la instalación de aplicaciones de terceros. Con el lanzamiento del primer iPhone en 2007 y la primera integración de una app store en un móvil, comienza a emerger el desarrollo de aplicaciones para los mismos, pasando a ser un elemento indispensable en los smartphones modernos.

Voting Garden se crea con el objetivo de brindar a los usuarios una herramienta para gestionar votaciones en grupo de forma remota. La aplicación permite tanto crear como administrar cualquier tipo de sufragio, ya sea para una organización o de forma individual.

Analizando las necesidades que pueden llegar a tener las personas a la hora de tomar decisiones, Voting Garden se presenta como una solución. Ha sido desarrollada teniendo en cuenta el esquema básico de las votaciones tradicionales, donde la urna de voto se encuentra representada como el servidor, los usuarios como el dispositivo móvil y las votaciones como listas en los mismos. Asimismo, proporciona un editor, permitiendo una alternativa con varias opciones, al clásico “si, no o abstención”.

1.1 Antecedentes y estado actual

Si analizamos las funcionalidades que se pretenden para esta aplicación, nos damos cuenta que el concepto a implementar no se encuentra ampliamente expandido en el mundo del desarrollo software, por lo que se buscaron herramientas con objetivos similares a Voting Garden.

Debido a que el mundo del desarrollo de aplicaciones está cada vez más extendido, se encontraron varios casos semejantes en la PlayStore de Android. En general eran aplicaciones obsoletas y con bajas puntuaciones, pero existen varias muy completas y entre ellas destacan: Votinga[1] desarrollada por Alejandro Santana Lima, Showt – Instant Global Voting[2] de Showt Limited y Anonymous Voting[3] de StarLin.

Figura 1.1: Logo Votinga

Votinga posee una interfaz simple y muy amistosa para el usuarios, incluyendo un chat para poder comunicarse. Su uso se basa en la creación de actividades, y dentro de cada una de ellas, los usuarios podrán crear votaciones representadas como eventos.

Figura 1.2: Logo Showt – Instant Global Voting

Showt, por otro lado, tiene una interfaz más compleja. La estructuración de sus votaciones no difiere mucho de la descrita en Votinga, pero además incluye funcionalidades para redes sociales como Facebook o Twitter.

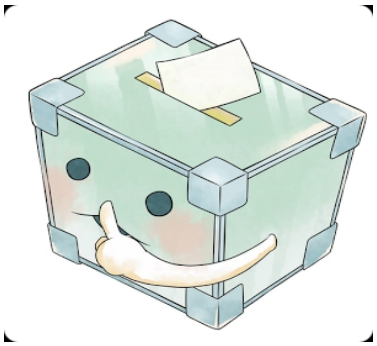


Figura 1.3: Logo Anonymous Voting

Por último tenemos Anonymous Voting, con una interfaz muy simple y funcionalidades más limitadas.

En general, este tipo de aplicación no está muy extendido, aunque posee un potencial incuestionable, pues son compatibles con herramientas chat como WhatsApp o Telegram.

1.2 Objetivos

Los objetivos de este trabajo de fin de grado podrían resumirse en los siguientes apartados:

- Analizar la finalidad de la aplicación: a quien va dirigida y como va a ser utilizada.
- Examinar las distintas aplicaciones encontradas y sus funcionalidades.
- Estudiar los distintos lenguajes y métodos que existen a la hora de desarrollar aplicaciones web móviles.
- Diseñar una aplicación móvil que permita a los usuarios organizarse en grupos para administrar sus votaciones.

1.3 Metodología

La metodología de desarrollo ha seguido la siguiente planificación establecida al comienzo del trabajo.

Tarea 0.- Coordinación.

La primera de las tareas se basa en la coordinación del proyecto, fijando reuniones con la tutora.

Tarea 1.- Estudio de herramientas y lenguajes a utilizar.

Una vez analizados los antecedentes de la materia que va ser tratada, se procederá a estudiar y elegir las herramientas que cumplan con los requisitos necesarios para el desarrollo de una aplicación móvil. Debido al conocimiento previo en tecnologías web, se da preferencia a los lenguajes HTML, CSS y JavaScript.

Tarea 2.- Preparación del entorno de desarrollo.

Tras estudiar las distintas alternativas para el desarrollo de aplicaciones web para dispositivos móviles, es necesario preparar e instalar las herramientas que van a ser usadas como NodeJs, el framework Ionic, HeidiSql como gestor gráfico de las bases de datos, Visual Studio Code para el desarrollo de TypeScripting con AngularJS y Postman para el testeo de las peticiones al servidor.

Tarea 3.- Implementación de la base de datos.

Planificaremos e implementaremos la base de datos con las tablas consideradas necesarias para nuestra aplicación.

Tarea 4.- Implementación del servidor y métodos de comunicación.

Una vez creada la base de datos, procederemos a desarrollar el servidor con las tecnologías de AngularJS y los métodos necesarios para la comunicación entre el mismo y los datos almacenados.

Tarea 5.- Implementación del cliente.

Desarrollaremos la aplicación web haciendo uso de Ionic. Para ello se crearán los distintos componentes que se explicados más adelante.

Tarea 6.- Pruebas en un entorno real.

Realizaremos pruebas de la herramienta en Android, así como con los usuarios, de forma que podamos obtener opiniones por parte de los mismos.

Tarea 7.- Difusión de los resultados.

Finalmente, se elaborará un acta y una presentación con los aspectos generales del trabajo, con el objetivo de explicar punto por punto el desarrollo de la aplicación.

1.4 Organización de la memoria

Este documento se divide en cuatro capítulos principales y dos secundarios:

- Capítulo 2: Descripción de las herramientas utilizadas.
- Capítulo 3: Descripción de la aplicación y sus funcionalidades.
- Capítulo 4: Descripción del desarrollo de la aplicación.
- Capítulo 5: Descripción de las pruebas.
- Capítulo 6: Conclusiones del trabajo.
- Capítulo 7: Presupuesto y factura final.

Capítulo 2
Herramientas y tecnologías

En el capítulo anterior se ha introducido la aplicación Voting Garden, una aplicación móvil administradora de votaciones de usuarios. Para llevarla a cabo han sido necesarias las siguientes herramientas y tecnologías.

2.1 Ionic

Ionic[4] es un SDK de código abierto para desarrollo de aplicaciones móviles multiplataforma. La versión original se basaba en AngularJS y Apache Cordova[5], pero recientemente, con Ionic 2 y Ionic 3(Beta), el framework se basa únicamente en Angular. Este SDK brinda una serie de herramientas y utilidades para crear aplicaciones móviles con tecnologías web como TypeScript[6], Sass y HTML5. Así pues, las apps pueden ser implementadas con las tecnologías anteriormente citadas y distribuidas en las tiendas nativas de aplicaciones como PlayStore o AppStore.

Figura 2.1: Logo Ionic

La herramienta permite a los desarrolladores construir aplicaciones con todas las funcionalidades que se pueden encontrar en otros SDK de desarrollo nativo en móviles. Los usuarios pueden crear las aplicaciones, personalizarlas para Android, IOS o Windows Phone y desplegarlas mediante Cordova. Además, incluyen componentes móviles, tipografías, paradigmas interactivos y un extensible tema(IU) de base.

Haciendo uso de Angular, Ionic proporciona componentes personalizados y personalizables, así como métodos para la interacción con y entre ellos.

Por último, junto con el SDK, Ionic también incluye servicios que los desarrolladores pueden utilizar para el testeo, análisis y despliegue de las aplicaciones implementadas.

2.2 MariaDB

MariaDB[7] es un sistema gestor de bases de datos relacionales gratuito de algunos desarrolladores de MySQL[8], con el cual, mantiene un alto grado de compatibilidad, asegurando equivalencias entre sistemas y coincidencias exactas con MySQL APIs y comandos. Así mismo, además de los motores de almacenamiento estándar como pueden ser MyISAM, BLACKHOLE o CSV, MariaDB inlcuye muchos otros como MyRocks, TokuDB o Aria.

Figura 2.2: Logo MariaDB

2.3 HeidiSQL

HeidiSQL[8] es una herramienta gratuita y de código abierto para la administración de bases de datos. Soporta MySQL y varias de sus vertientes como PostgreSQL y MariaDB.

Figura 2.3: LogoHeidiSQL

Algunas de sus principales características son: conectarse a múltiples servidores de forma simultanea en una misma ventana, crear tablas, eventos, vistas, rutinas y triggers de forma gráfica, permite exportar bases de datos enteras para posteriormente añadirlas a otro servidor, administración de usuarios y editor de texto para probar consultas en las bases de datos.

2.4 Socket.io

Socket.IO[14] es una librería de javascript para el desarrollo de aplicaciones web. Permite comunicación en tiempo real y bidireccional entre clientes web y servidores. Para un correcto uso de la misma, es necesario instalarla tanto en el cliente como en el servidor mediante npm.

Figura 2.4: Logo socket.io

2.5 ChartJS

ChartJs[15] es una librería javascript que permite crear un gran número de gráficas haciendo uso de tecnologías web. Es sencilla y muy facial de usar y ha sido utilizada en este trabajo para visualizar los resultados de las votaciones.

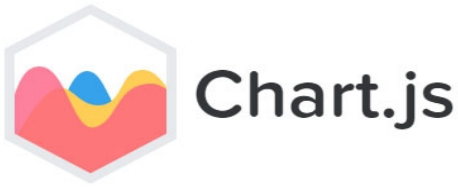


Figura 2.5: Logo ChartJS

2.6 Postman

La última herramienta que vamos a citar es Postman[10], a diferencia de las demás es una simple extensión de Chrome, sin embargo, es de mucha utilidad a la hora de trabajar con aplicaciones web, ya que permite hacer peticiones de cualquier tipo a una dirección dada. Por ello, ha sido imprescindible en el desarrollo de esta aplicación, teniendo por objetivo testear las rutas de las peticiones get, post o put al servidor.

Figura 2.6: Logo Postman

Capítulo 3
Aplicación y funcionalidades

Una vez estudiados los antecedentes y las herramientas utilizadas para el trabajo, procedemos a mostrar como navegar y usar Voting Garden. Así mismo, explicaremos algunos de los elementos HTML de Ionic.

3.1 Componentes de Ionic

Previo a la explicación de las distintas páginas que componen nuestra aplicación, procederemos a detallar como se relacionan los ficheros en este framework para la programación de aplicaciones web.

Ya que Ionic está basado en Angular, los proyectos creados en el mismo interaccionan por medio de componentes TypeScript. Asimismo, un ejemplo de componente podría ser la página con fichero mypage.ts cuyo código sería el siguiente:

```
@Component({
    selector: 'my-page',
    tempalteUrl: 'mypage.html'
})
export class MyPage {
    //Código de la página
}
```

Como podemos apreciar, la variable selector apunta al selector CSS de nombre *my-page*, el cual debemos editar para poder aplicar estilos a nuestra página. Al igual que con el selector, la variable *templateUrl* apunta al tempalte html que mostrará por pantalla el contenido del componente. Ambas variables deben tener un fichero del mismo nombre que la página donde aplicar y obtener los datos. En este caso serían mypage.html y mypage.scss, y deben estar localizadas en la misma carpeta que mypage.ts.

3.2 LogIn y SignIn

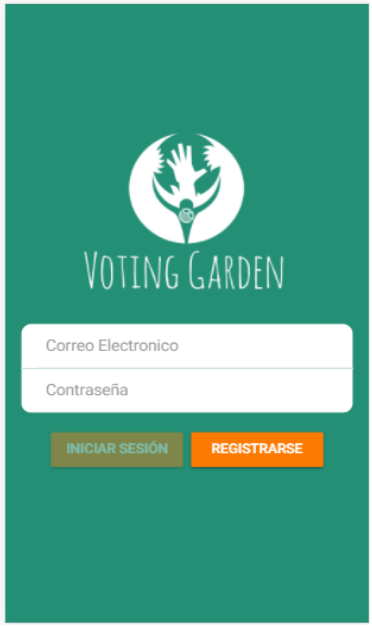


Figura 3.1: Log In

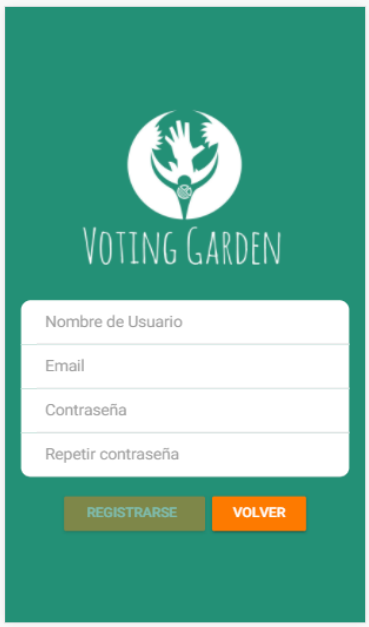


Figura 3.2: Sign In

Al iniciar la aplicación se nos muestra la página de Log In, representada en la figura 3.2. La interfaz se construye a partir de varios componentes de Ionic explicados a continuación:

- ion-content: representa el área de contenido de toda la página. Debe ser único en la misma vista y su modo de uso es el siguiente:

  <ion-content>

  Contenido de la página de Ionic

  </ion-content>

- form: es la misma etiqueta tradicional de HTML.

- ion-list: elemento de la interfaz ampliamente usado para listar grupos de elementos hijos. Tanto la lista como sus hijos pueden ser cualquier tipo de etiqueta HTML.

  <ion-list>

  elementos HTML

  </ion-list>

- ion-item: son las fila de las tablas de ionic, dentro de estos componetes se encuentran los input fields para introducir los datos de usuario.

  <ion-item>

  elemento html

  </ion-item>

- ion-input: representa los campos de entrada y es utilizado al igual que los input de HTML.

  <ion-input tyoe=”text”> </ion-input>

Figura 3.3: Sign In con errores

Al pulsar el botón de registro, nos mostrará la página de registro, cuya única diferencia con la de inicio de sesión son el número de campos de entrada.

Ambas páginas responden con errores si los datos introducidos no se ajustan a los requerimientos, como que el campo del correo sea un correo electrónico válido, que el nombre sea una cadena alfanumérica, que la contraseña sea muy corta y contenga caracteres especiales o que el segundo campo de la contraseña no sea igual al primero. En la figura 3.3 se muestra la página de registro con todos los errores posibles.

Para mostrar los errores, se hacemos uso de la directiva de Angular *ngif y el condicionante viene dado por el controlador del formulario en la clase de la página:

<ion-item *ngIf="form.get('correo').errors && form.get('correo').dirty">

3.3 Usuarios Independientes

3.3.1 Ver votaciones de usuario

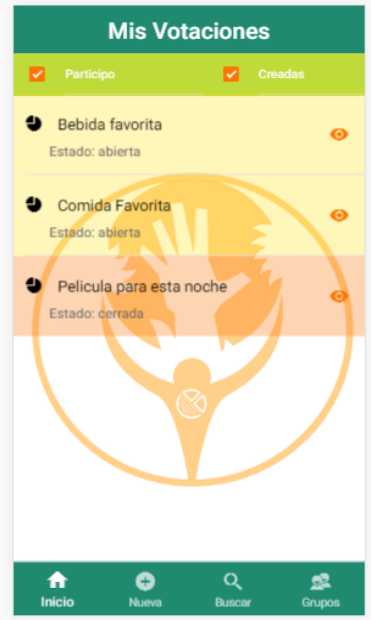


Figura 3.4: Crear una nueva votacion

Una vez nos hayamos registrado e iniciado sesión, la primera página que se muestra son las votaciones que hemos creado (figura 3.4), en las que participamos y que no pertenecen a ningún grupo. Del mismo modo, las votaciones que ya estén cerradas aparecerán en un color distinto del que tienen las votaciones que aún no han acabado. Para cada sufragio tenemos la oportunidad de ver los detalles del mismo pulsando en el icono del ojo situado a su derecha.

En la interfaz también se aprecian dos checkbox utilizados para filtrar las votaciones por las que el usuario ha creado y en las que está participando.

<ion-checkbox [(ngModel)]="variable" (ionChange)="método()" ></ion-checkbox>

3.3.2 Crear nuevas votaciones y Buscar Votaciones

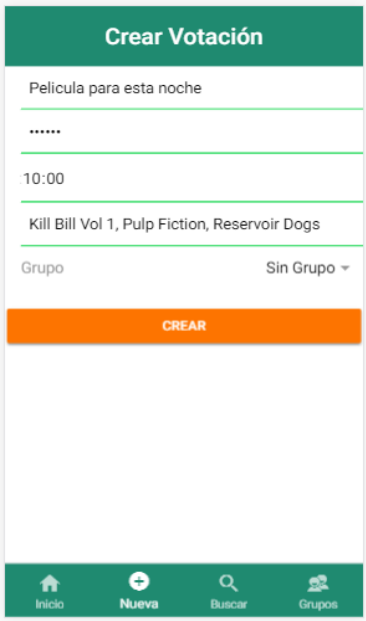


Figura 3.5: Visualización de las votaciones

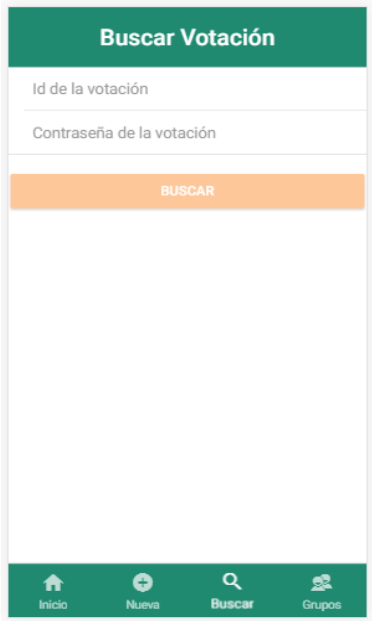


Figura 3.6: Buscar Votación

La figura 3.5 representa la página donde podremos crear las votaciones, teniendo un campo de texto para el nombre, la contraseña, el tiempo que permanecerá abierta la votación y las opciones. Además, podremos seleccionar el grupo al que pertenecerá la votación.

El nuevo elemento HTML que encontramos en esta página es ion-select, utilizado para elegir un grupo. Esta es su estructura:

```
<ion-select formControlName="controlador">
    <ion-option [value]="3">Opcion 1</ion-option>
    <ion-option [value]="2">Opcion 2</ion-option>
</ion-select>
```

Por último, tenemos la página donde los podemos buscar cualquier votación sin grupo. En ésta página debemos ingresar el id de la votación en la que se quiere participar y la contraseña, ambos suministrados ya sea por el creador o por otro usuario que ya haya participado.

3.4 Grupos

3.4.1 Mis Grupos

Figura 3.7: Mis Grupos

La página Mis Grupos (figura 3,7) nos permite administrar los grupos a los que pertenecemos. En la interfaz encontramos como elemento central una lista con nuestro grupos. Situada en la esquina inferior izquierda, se encuentra un contenedor con dos campos de entrada de texto cuya finalidad es la de unirnos a nuevas organizaciones. Por último, situada a la derecha de la anterior citada, hayamos otro contenedor con el para la creación de nuevas entidades.

3.4.2 Gestión de grupo

Por último tenemos la página de gestión de los grupos, en las figura 3.11 y 3.12. Podemos observar dos listas distintas: la primera representa a los usuarios de la organización y la segunda las votaciones asociadas. El *id* y el nombre de la votación aparecen en la cabecera de la página, así como un botón que difiere dependiendo del modo.

Figura 3.8: Gestión de Grupo
Vista de Usuario

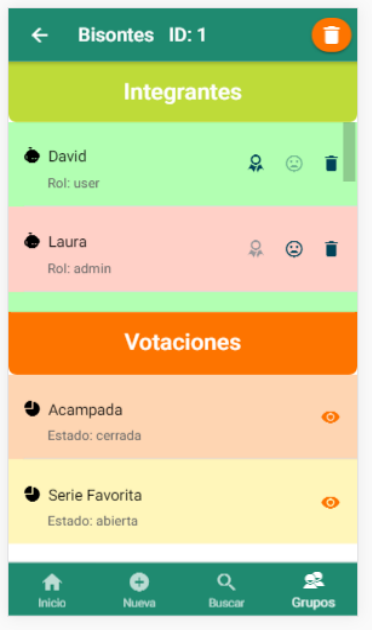


Figura 3.9: Gestión de Grupo
Vista de Creador

Las diferencias entre las dos figuras vienen dadas en función de si somos el creador de ese grupo o no. Por lo tanto, si tenemos dicho rol, nos será posible borrar el grupo, pulsando sobre el icono de la esquina superior derecha, ascender o descender a los usuarios de su rol y borrar a los mismos presionando los iconos de la entrada en la lista para dicho integrante. Por el contrario, si entras en la página de un grupo en el que no eres el creador solo podrás ver otros usuarios, salir del grupo o participar en las votaciones.

3.5 Votación

Las siguientes figuras corresponden con las situaciones que se dan en la visualización de la página de detalles de las votaciones.

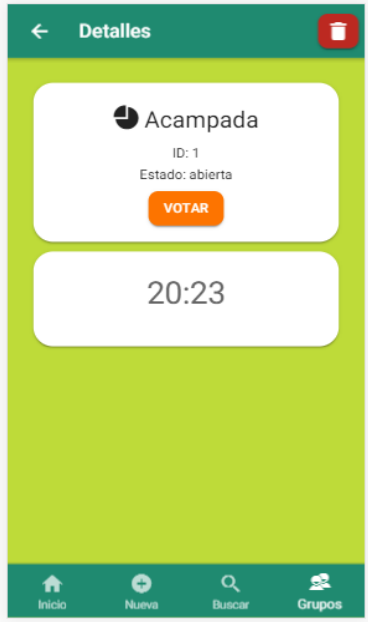


Figura 3.11: Detalles Votación sin finalizar

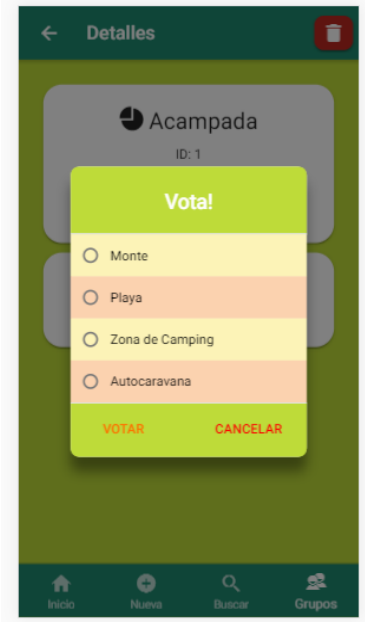


Figura 3.10: Detalles opciones de la votación

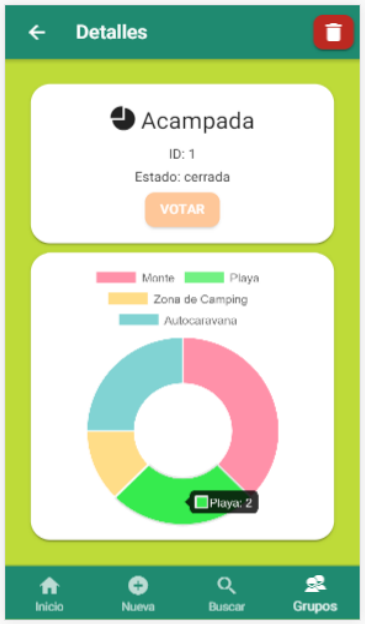


Figura 3.12: Detalles Votación Finalizada

En la figura 3.9, se puede apreciar la página de detalles de la votación mientras la misma se encuentra abierta. Podemos votar haciendo uso del botón, donde nos saldrá una alerta con las opciones para que elija una (figura 3.8), en caso de ya haber votado, la aplicación nos avisara del cambio en nuestro voto. También apreciamos un contador de tiempo, el cual al llegar a 00:00 cierra la votación y muestra los resultados como en la figura 3.10 (explicaremos todas estas funcionalidades más adelante). Por último, destacar el botón rojo con una basura en la esquina superior izquierda para borrar la votación, el cual en el caso de los grupos está activo para creadores y administradores de grupo y en las votaciones sin grupo, para el creador de la votación.

En esta página encontramos un elemento nuevo de Ionic, las ionic-card. Su uso es el siguiente:

<ion-card >

<ion-card-content>

<ion-card-title> Titulo de la carta de Ionic</ion-card-title>

Contenido

</ion-card-content>

</ion-card>

En este caso, el elemento lo hemos usado para mostrar la información referente a la votación, así como la cuenta atrás y los resultados.

Capítulo 4
Desarrollo de la aplicación

En este capítulo se describirán algunos de los aspectos técnicos más importantes en el desarrollo del trabajo.

4.1 Instalación de las herramientas

La aplicación Voting Garden ha sido desarrollada bajo el sistema operativo Windows 10, es por ello que la mayoría de herramientas utilizadas poseen un proceso de instalación sencillo mediante un ejecutable .exe. En consecuencia, en los siguientes sub-apartados se describirá la instalación de aquellas requeridas por Node.js.

4.1.1 Instalación de Ionic

En el proceso de desarrollo de nuestra aplicación encontramos dos posibilidades a la hora de crear un proyecto con Ionic.

La primera metodología, para la cual no nos detendremos mucho, consistía en la instalación del entorno de desarrollo Visual Studio 2017 y por medio de su gestor de proyectos añadiríamos Apache Cordoba y luego el framework de Ionic. La principal problemática encontrada en este procedimiento fue el elevado costo de recursos para el equipo que conlleva Visual Studio 2017, haciendo casi imposible el desarrollo si se ejecutan otros programas como HeidiSQL necesarios para el proyecto.

Tras descartar la primera opción procedimos a instalar Ionic por medio de la consola de comandos de NodeJS. Esta otra opción era mucho más viable puesto que configuraríamos todo lo necesario con un solo comando:

```
$ npm install -g ionic cordova
```

Una vez finalizada la instalación de Cordova e Ionic, ya estaba todo preparado para iniciar el proyecto por medio del comando:

$ ionic start VotingGarden

Con la ejecución del comando anterior, se crearía una carpeta con el nombre del proyecto VotingGarden. El último paso consistía en iniciar la emulación:

$ ionic serve -lab

De esa forma comenzaba a compilarse el proyecto y con la opción -lab emularía la aplicación en el buscador por defecto del equipo.

4.1.2 Instalación de Socket.io

Para la puesta en marcha de Socket.io era necesario instalar GitHub en nuestro equipo, para así poder descargar el repositorio de la herramienta y probar su correcto funcionamiento.

Una vez configurado GitHub fro Desktop, el siguiente paso consistía en ejecutar en la consola de NodeJS el comando para instalar Socket.io:

$ npm install socket.io

Para finalizar y con el objetivo de probar el funcionamiento de la nueva herramienta instalada, se procedio a descargar el ejemplo de prueba del repositorio de Socket.io para luego ejecutarlo:

$ git clone git://github.com/LearnBoost/Socket.IO-node.git socket.io-node --recursive && cd socket.io-node/example/

$ node server.js

4.2 Desarrollo del servidor

4.2.1 Puesta en marcha por medio de Express

Cuando iniciamos el proyecto, se habían planificado una serie de tareas. La primera de ellas era el desarrollo de un servidor utilizado como puente entre los distintos clientes móviles y el mismo. En consecuencia, y debido a la naturaleza web de la aplicación, la implementación del servidor fue realizada por medio de Express[11]. El primer paso consistía en desarrollar el servicio y ejecutarlo par comprobar que todas las dependencias y frameworks estaban correctamente instalados.

Para la instalación de Express, utilizaríamos nuevamente NodeJS, ejecutando el comando:

```
$ npm install express
```

Una vez instalado Express, debíamos crear el servicio y ejecutarlo para comprobar la correcta configuración de la herramienta. Para ello se dispuso el puerto 8000 para la escucha en modo local del servidor.

4.2.2 Configuración y tablas de la base de datos

Una vez lanzada la aplicación y comprobado su correcto funcionamiento, se procedió a implementar la conexión con la base de datos. Para la misma fue necesario instalar MariaDB, aunque también es posible desplegar el servidor con MySQL. La conexión es sencilla y fácil de desarrollar, pues tan solo hay que suministrar los datos del motor de gestión de bases de datos para conectarnos a las mismas. En la figura 4.1 se proporciona el modelo entidad relación de la base de datos.

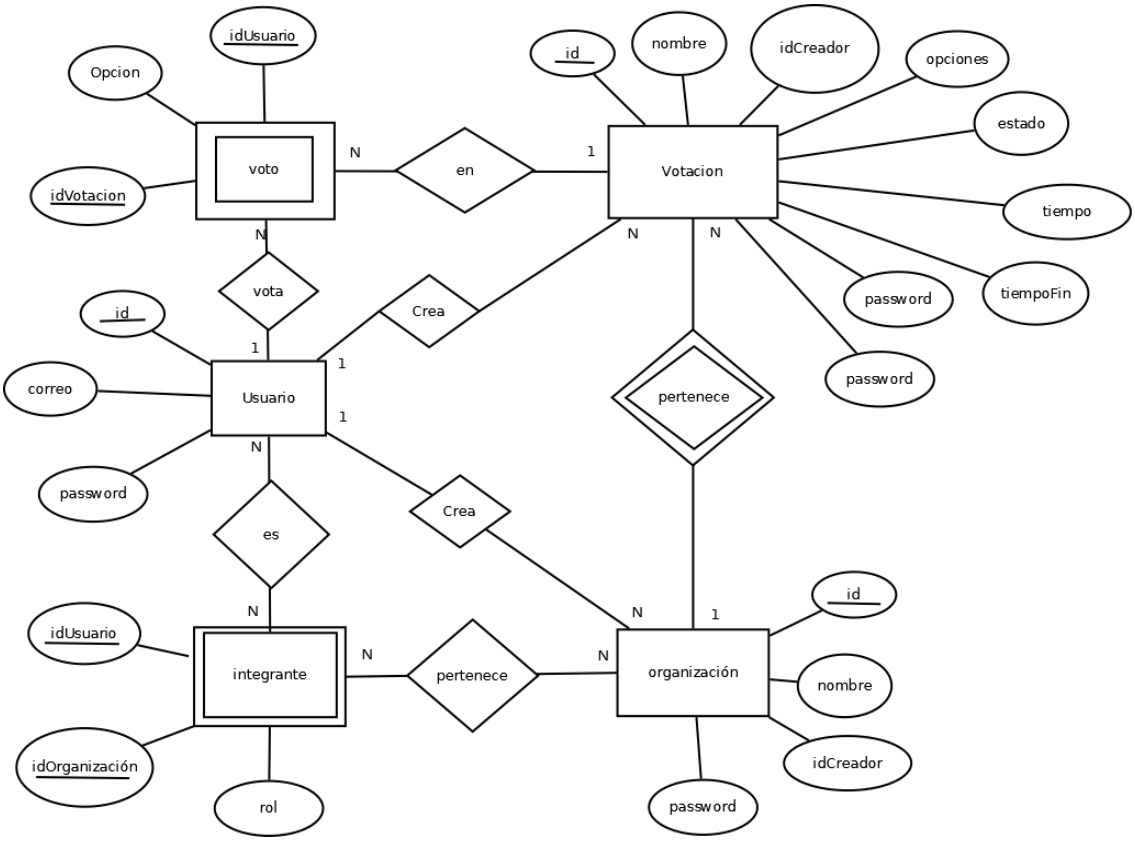


Figura 4.1: Modelo E/R base de datos de Voting Garden

El siguiente paso consistía en programar las consultas que iban a ser necesarias para el cliente. Un ejemplo de consulta es:

```
this.obtenerUsuarioPorId = function (id, respuesta) {
    conexion.obtener(function(er, cn) {
    cn.query('select * from usuarios where id = ?', id, function(error,
    resultado) {
        cn.release();
        if(error) {
            respuesta.send(error);
        } else {
            respuesta.send(resultado);
        } }) }) }
```

Una vez implementadas las consultas, se debíamos especificar las rutas de acceso a cada una de ellas, para poder hacer las peticiones get, post, put y delete al servidor:

```
app.get('/usuarios/', function(solicitud, respuesta) {
    bd.obtenerUsuarios(respuesta);
})
```

Por último, para testear las peticiones utlizamos de la extensión de Google Chrome Postman (figura 4.2).

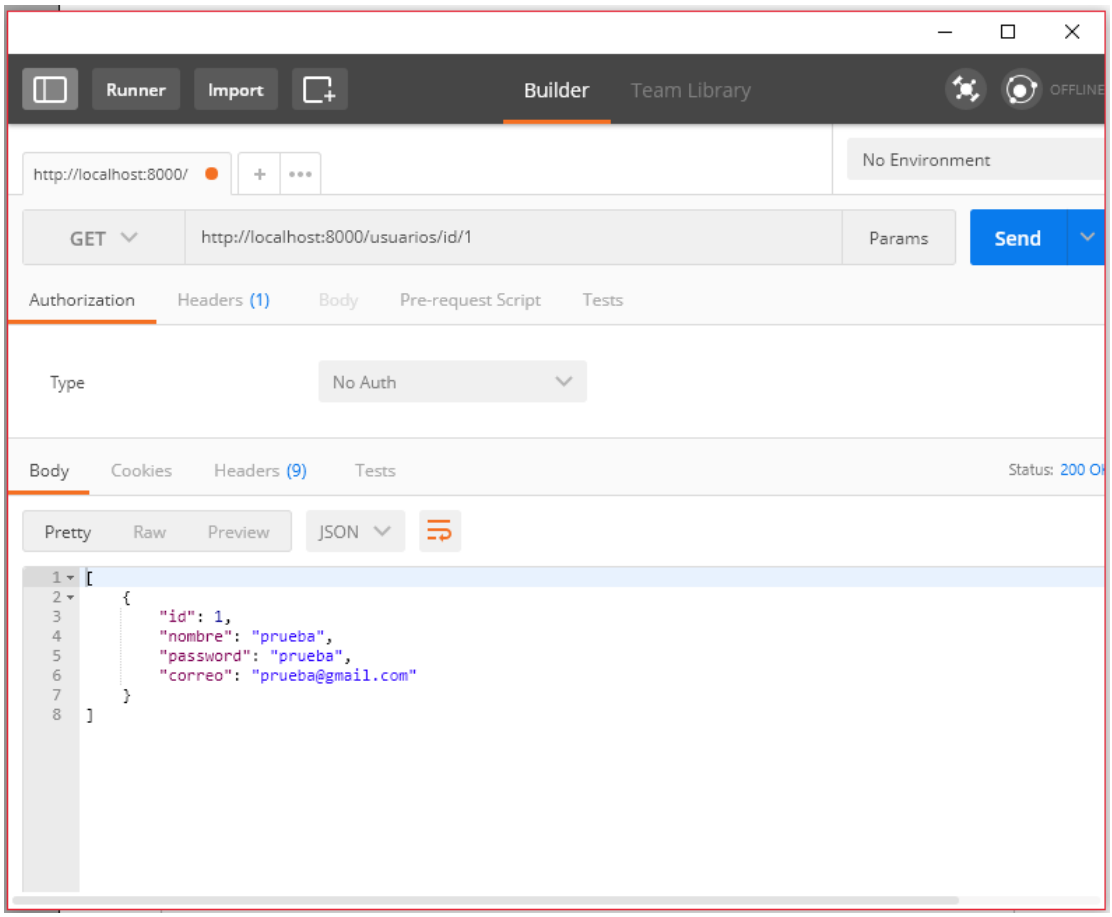


Figura 4.2: Respuesta de la petición del ejemplo en Postman

4.3 Desarrollo de una versión inicial de la aplicación

El siguiente paso consiste en el desarrollo de una primera versión de la aplicación para testear el apartado de la IU del framework Ionic. En consecuencia, se procedimos a implementar las primeras páginas de iniciar sesión y registro sin ningún tipo de funcionalidad y con el único objetivo de familiarizarse con las tecnologías y herramientas.

Una vez comenzado el proyecto en blanco, haciendo uso de los precisos y sencillos docs[13] de Ionic, se comenzó a diseñar la interfaz gráfica de la aplicación. En las figuras 4.1 y 4.2 se puede apreciar la primera versión de la aplicación:

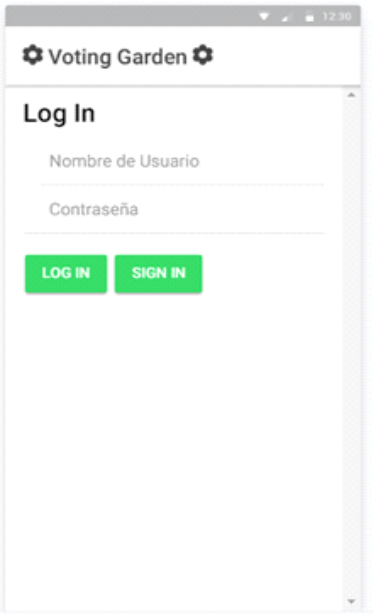


Figura 4.3: Versión inicial del Log In

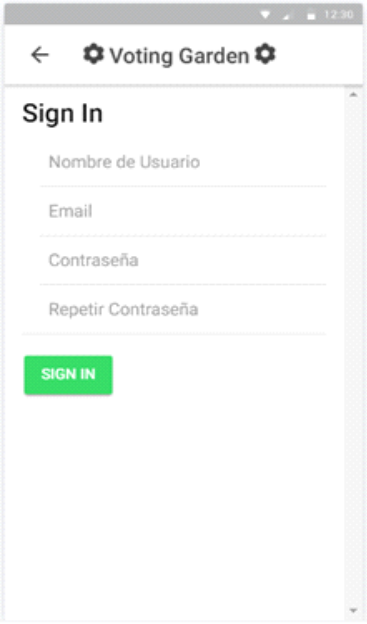


Figura 4.4: Versión inicial del Sign In

El siguiente paso consistía en implementar de forma muy básica las páginas de Mis Votaciones, Buscar Votación y Crear Votación, las tres mostradas en las figuras 4.3, 4.4 y 4.5.

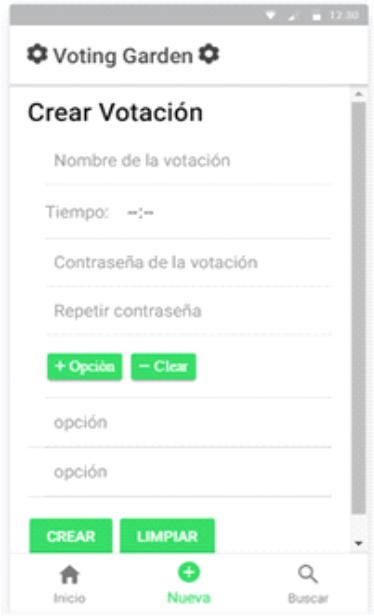


Figura 4.5: Versión inicial del la página Crear Votacion

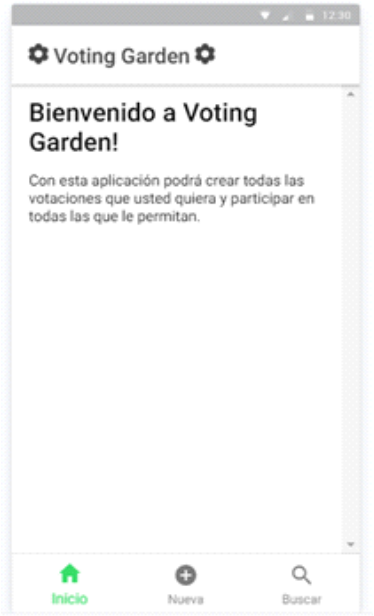


Figura 4.6: Versión inicial de la página de inicio

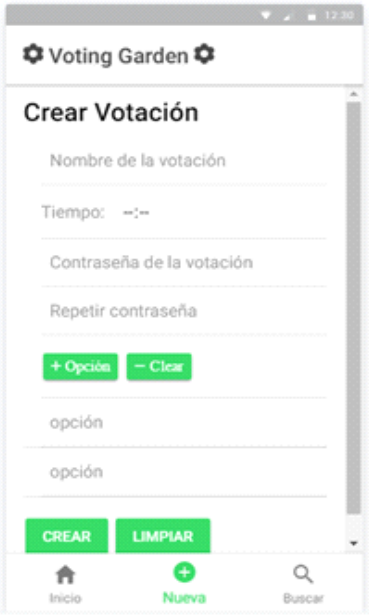


Figura 4.7: Versión inicial del la página Crear Votacion

4.4 Implementación de la comunicación entre el cliente y el servidor

Tras haber trabajado con los elementos HTML de Ionic, la siguiente tarea consistía en estudiar el TypeScripting, el lenguaje JavaScript utilizado por el framework. El primer paso fue el entendimiento de la estructura de ficheros seguida por Ionic, donde tenemos distintas páginas, las cuales están asociadas a un fichero html que mostraba el contenido, un fichero typescrip, para implementar cualquier tipo de funcionalidad y un fichero scss para darle estilos a la página.

Una vez entendidos los conceptos del desarrollo de aplicaciones web con Ionic, implementamos los servicios de comunicación entre la aplicación y el servidor. Para ello era necesario crear un servicio para cada tabla y una clase que representase los datos de las tablas. Así pues, la metodología consistía en el desarrollo de una función por cada ruta implementada en el servidor. Por ejemplo, para la ruta mostrada en el sub-apartado 4.2.2 el método a implementar en el servicio equivaldría a:

```
getUsuarioPorId(id: number): Observable<Array<Usuario>> {
    let url = `${this.url}/{id}`;
    return this.http.get(url)
                .map(r => r.json())
                .catch(this.handleError);
}
```

donde la url sería http://<ip del servidor>:puerto/usuarios.

Es importante destacar en este apartado la implementación del **tiempo de sincronización**, el cual definimos como la etapa en la que los usuarios tienen la posibilidad de votar y cambiar su voto. Se encuentra representado en la aplicación como el temporizador de la página de votación, y es calculado cuando creamos la votación.

Con el fin de evitar una sobrecarga constante a la base de datos actualizando el temporizador con cada cliente conectado, descartamos la posibilidad de almacenar el tiempo restante en la tabla.

Por consiguiente, la alternativa que proponemos es calcular en el momento de crear la votación la fecha en la que debe acabar. Esto se hace en función del tiempo introducido, añadiéndolo al momento de la creación. Así mismo, cuando entramos en la votación, tiempo restante mostrado se calcula como la diferencia entre la fecha actual y la fecha de finalización y, por último, a ese resultado se le va restando uno cada segundo para conseguir el efecto de temporizador.

Durante todo el periodo en el que el temporizador está activo, podemos votar y cambiar el voto tantas veces como queramos. Este proceso se implementa de la siguiente manera:

```
votar(data)  {
    this.voto = new Voto(+localStorage.getItem('currentUserId'), data,
        this.votacion.id);
    if(this.vacio(this.aux)) {
        this.votoService.postVoto(this.voto)
                .subscribe(
                    rs => (this.aux = rs),
                    er => console.log(er),
                    () => console.log(this.aux)
                )
    } else {
        this.votoService.cambiarVoto(this.voto)
                .subscribe(
                    rs => (this.aux = rs) && (this.votoCeado()),
                    er => console.log(er),
                    () => console.log(this.aux)
                )
        }
    }
```

Lo primero que hace el método es crear un objeto de tipo voto con la opción seleccionada, nuestro id y el id de la votación. Se comprueba si la variable que almacena el voto está vacía, en caso afirmativo se hace una petición POST al servidor cuya respuesta se almacena en la variable *this.aux*, para que en la próximo comprobación, la petición sea un PUT actualizando el voto.

4.5 Mejora de la interfaz gráfica e implementación de WebSockets

Con la aplicación funcional, haciendo uso del servidor, el siguiente paso consistía en crear una interfaz gráfica más amigable y acorde con el nombre de la aplicación. Lo primero fue buscar una paleta de colores que no desentonase estéticamente, para ello se utilizó la herramienta web Color Wheel[16] de Adobe. Con una paleta de colores definida, el siguiente paso era crear una interfaz amigable para el usuario, la cual fue desarrollada haciendo uso de sass y en específico de Flexbox (un buen tutorial de como utilizar Flexbox se encuentra en[17]), creando contenedores y asignándoles zonas y comportamientos específicos. Gracias a ello, se logró el resultado visto en las figuras del capítulo anterior.

Por último surgió la problemática de en que momento se debían actualizar las vistas de usuario cuando creaba nuevas votaciones o estas se actualizaban. Teniendo esto en cuenta, se instalaron las librerías de socket.io tanto en el cliente como en el servidor, para poder crear y manejar eventos en momentos específicos. Un ejemplo del servicio socket.service.ts es:

```
export class SocketService {

    private url = 'http://localhost:8000/';

    private socket;


    sendEvent(event) {
        console.log("SENDING")
        this.socket.emit('event-firing', event);
    }
```

```
    getEvent() {
        let event;
        let observable = new Observable(observer => {
        this.socket = io(this.url);
        this.socket.on('event', (data) => {
            //Mandamos el evento
            observer.next(data);
            });
            return () => {
                console.log("disconnected from socket");
                this.socket.disconnect();
            }
        })
        return observable;
    }
}
```

Con el fin de ilustrar mejor el funcionamiento de los sockets en la aplicación, explicaremos momentos específicos en el código donde se lanzan eventos a través de los sockets y a su vez son capturados. En nuestra implementación existen dos tipos de eventos “reloadHome” y “reloadGroups”, el primero avisa que hay que recargar la página de Mis Votaciones y la segundo las páginas relacionadas con los grupos.

Un ejemplo de recepción del primer evento es el siguiente:

```
ngOnInit() {
    this.connection = this.socket.getEvent().subscribe(event => {
        if(event == "reloadHome") {
            this.obtenerVotaciones();
            event = "";
        }
    })
}
```

Como podemos observar, implementamos el método para la obtención de eventos al iniciar la vista de la página y asi escuchar continuamente en caso del lanzamiento de un evento.

El método que envía los eventos al socket suele ejecutarse con peticiones de tipo PUT, POST o DELETE ya que editan las tablas y por lo tanto las vistas de usuario se deben actualizar. El método para enviar eventos es el siguiente:

```
this.socket.sendEvent("reloadHome");
```

4.6 Implementación y gestión de grupos

El último paso antes de lanzar nuestra aplicación fue el desarrollo de la página Mis Grupos, para ello tuvimos que crear nuevas páginas , dos tablas más en la base de datos y los servicios correspondientes. En éste apartado se destacan los roles de los usuarios junto con sus permisos:

- Creador: Creación y borrado de votaciones. Participación en votaciones. Administración de usuarios (cambio de roles y borrado de integrantes).
- Administrador: Creación y borrado de votaciones. Participación en votaciones.
- Usuario: participación en votaciones.

4.7 Lanzamiento de la aplicación

Antes de hacer la build para dispositivos móviles alojamos el servidor en los servidores Iaas de la universidad. Para ello, mediante sftp subimos los archivos de Express, instalamos node y su cliente npm, montamos la base de datos con MariaDB y por último ejecutamos el servicio para poder acceder a la aplicación de forma remota.

Para poder publicar la aplicación, Ionic dispone de un tutorial[18] de bastante ayuda.

El primer paso consiste en ejecutar el siguiente comando:

$ ionic cordova build --release android

con esto se generará un .apk con tu aplicación sin firmar, por lo que no se podrá instalar en ningún dispositivo.

La siguiente tarea será crear una klave privada para poder firmar la aplicación. Para ello se ejecuta el comando:

$ keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000

Es importante saber que la herramienta keytool se encuentra en la carpeta del jdk de java.

Por último, firmamos la aplicación:

$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore HelloWorld-release-unsigned.apk alias_name

En el caso de la herramienta jarsinger, debemos dirigirnos al sdk de android en la carpeta raíz de nuestro ordenador.

Capítulo 5
Pruebas

5.1 Pruebas en un entorno real

Para testear el correcto funcionamiento en distintos dispositivos, seleccionamos un conjunto de 9 usuarios para que probasen la aplicación en sus dispositivos. Los resultados fueron satisfactorios puesto que la aplicación era completamente funcional en los dispositivos móviles Android. Una vez acabado el periodo de pruebas de 48 horas, se realizó una pequeña encuesta a los usuarios sobre la aplicación, cuyos resultados fueron los siguientes:

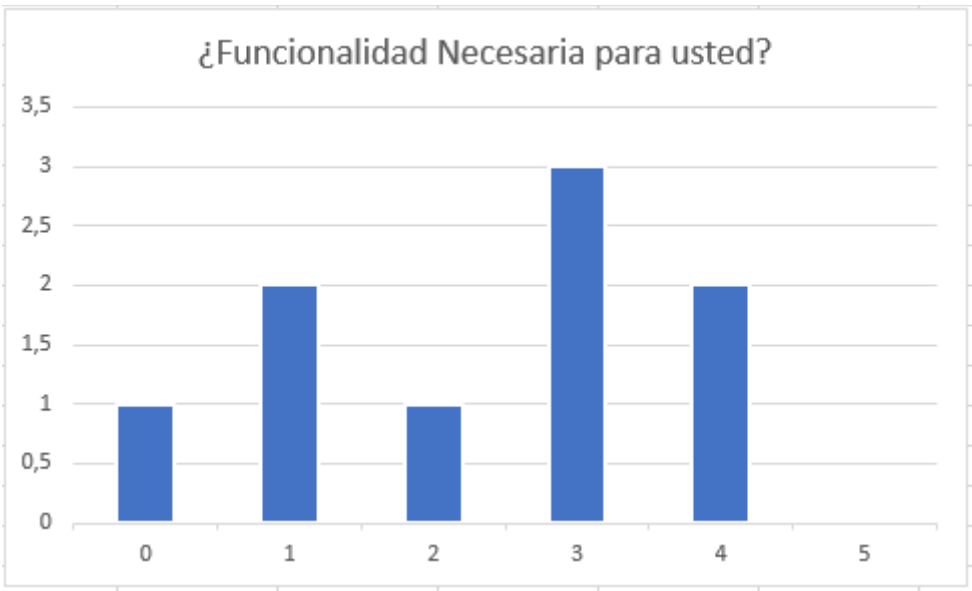


Figura 5.1: Resultado de pregunta 1

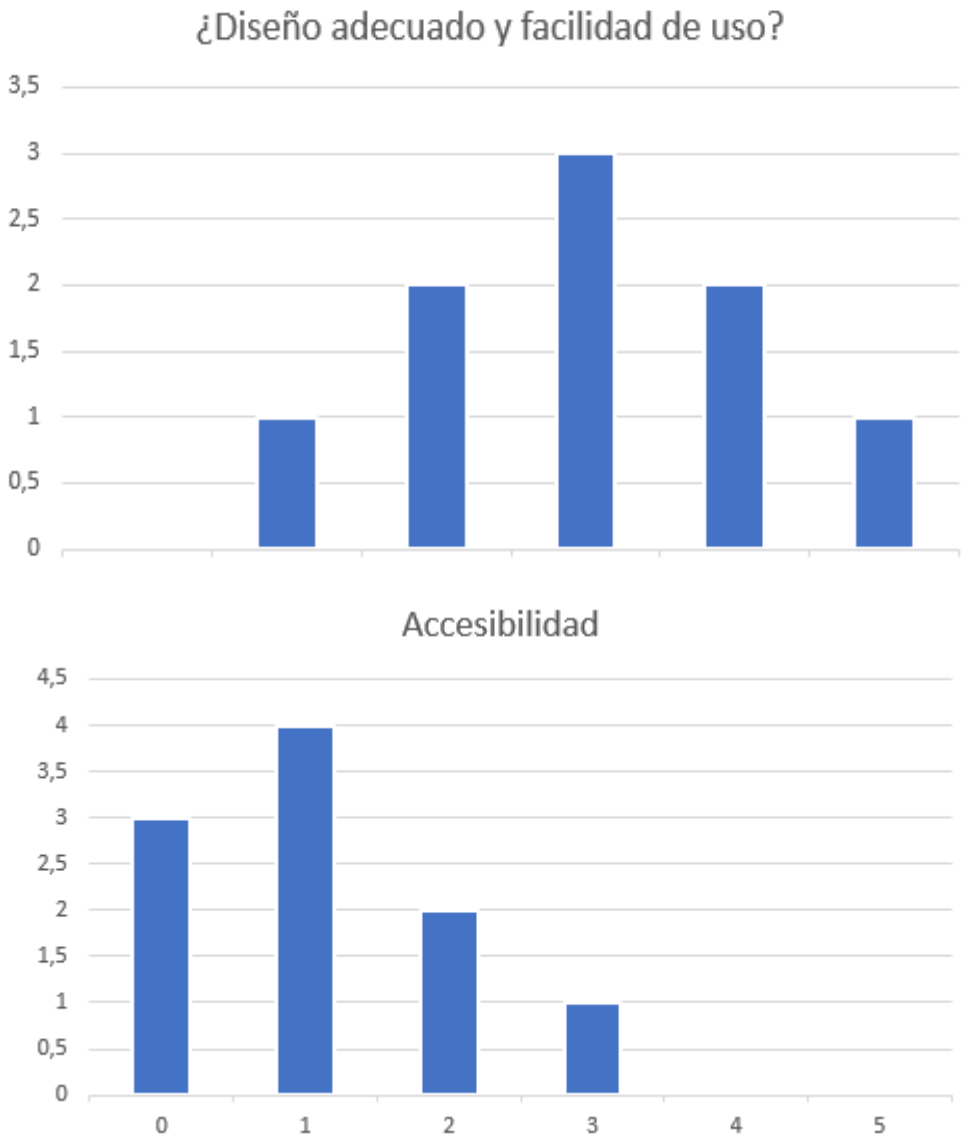


Figura 5.3: Resultado de pregunta 3

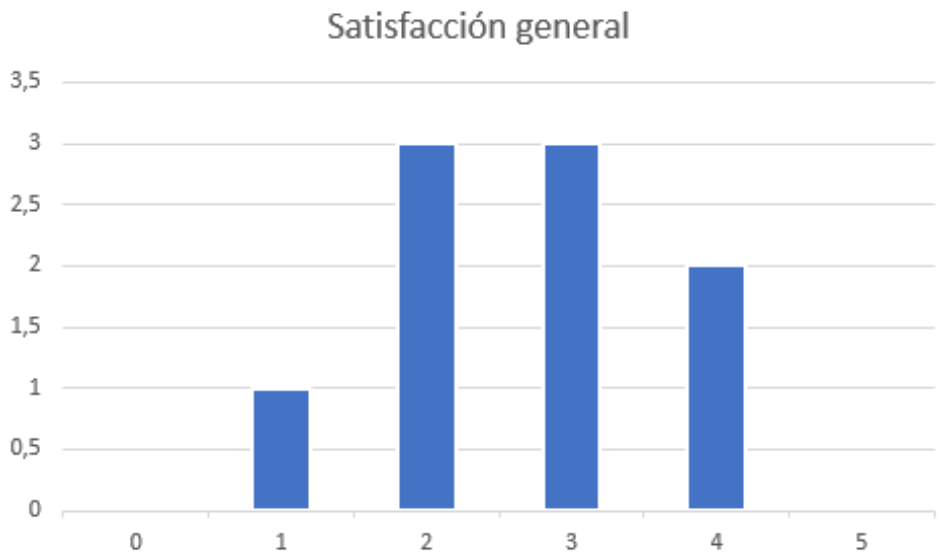


Figura 5.4: Resultado de pregunta 4

En los resultados destacamos la satisfacción general, pues a pesar de ser una aplicación con algunas restricciones, los usuarios demostraron una gran aceptación hacia Voting Garden.

Por el contrario, uno de los grandes problemas de la aplicación es su accesibilidad puesto que para la mayoría de usuarios es deficiente. Es por ello, que se tendrá en cuenta en el segundo apartado del siguiente capítulo.

En definitiva, aunque se han obtenido resultados en su mayoría positivos, la muestra de la encuesta ha sido pequeña, por lo que probarlo con mayor cantidad de usuarios podría esclarecer un poco mejor que lineas seguir en un futuro.

5.2 Fallos encontrados

Los fallos detectados por los usuarios mientras hacían uso de la aplicación son los siguientes:

- En la página de gestión de grupos al introducir texto en entrar o buscar grupo, la interfaz gráfica se descuadra mucho y se ve una franja negra.
- Los checkbox para filtrar en la página de Mis Votaciones a veces no

funcionan.

Capítulo 6
Conclusiones y líneas futuras

6.1 Conclusiones

El mundo del desarrollo web está cada vez más extendido y con el tiempo se vuelve más fácil desarrollar aplicaciones que agraden a los usuarios tanto a nivel estético como a nivel funcional. Es por ello que las principales tiendas de aplicaciones se encuentren llenas de todo tipo de herramientas y la que se ha desarrollado en éste trabajo no está exenta de ello.

Hoy en día utilizamos aplicaciones para todo tipo de problemas, ya sea para gestionar nuestras tareas como para movernos por las ciudades. Es innegable que la informática se expande exponencialmente y con ella todo lo relacionado con la tecnología.

Voting Garden no es sino una más de tantas otras aplicaciones, pero me ha enseñado a desarrollar aplicaciones web para móviles de forma sencilla y eficiente, además de demostrar que empezando con tan solo conocimientos de programación web, puedes llegar a crear una aplicación cliente-servidor completamente funcional.

6.2 Líneas futuras

Con Voting Garden ya en algunos dispositivos de varios conocidos y recibiendo las pequeñas criticas positivas a mi trabajo, existen alguhnos planes futuros en la continuación de la aplicación:

- Mejorar relación de aspecto en dispositivos con pantallas grandes.
- Evitar sobrecargas en las bases de datos planificando mejor.
- Controlar mejor los cambios inesperados de la interfaz gráfica.
- Ahondar más en la herramienta sockets.io pues ofrece muchas más funcionalidades de las que se aplican en Voting Garden.
- Mejorar la accesibilidad.

Capítulo 7
Summary and Conclusions

We live in a world where web development is increasing widely, and over time it becomes easier to develop apps that satisfy users both with functionality and beauty. That is why the main app stores are full of all kinds of tools and what has been developed in this work is not exempt from it.

Nowadays we use mobile applications for every single problem we have, no matter what, the solution is in our phone. It is undeniable that computing expands exponentially and with everything technology related.

Hence, Voting Garden is no more than other applications, yet, it has taught me how to develop mobile applications simply and efficiently, as well as showing me that creating functional and quality software starting with little knolwedge is possible.

Capítulo 8
Presupuesto

8.1 Presupuesto

Tareas	Horas	Presupuesto
Revisión Bibliográfica	20h	7€/h
Estudio antecedentes	20h	10€/h
Desarrollo de la aplicación	200h	20€/h
Pruebas de la aplicación	50h	20€/h
Pruebas en entorno real	7h	15€/h
TOTAL:	**297h**	**5.445 €**

Tabla 8.1: modelo de presupuesto

En la tabla 8,1 se aprecia que el presupuesto total asciende a 5.445€ por 297h de trabajo.

Bibligrafía

[1] Votinga https://play.google.com/store/apps/details?id=com.softotalss.votinga&hl=es

[2] Showt – Instant Global Voting https://play.google.com/store/apps/details?id=com.showt.Showt

[3] Anonymous Voting https://play.google.com/store/apps/details?id=com.onappstore.lin.voting&hl=es_419

[4] Ionic https://ionicframework.com/

[5] Apache Cordova https://cordova.apache.org/

[6] TypeScript https://www.typescriptlang.org/

[7] MariaDB https://mariadb.org/

[8] MySQL https://www.mysql.com/

[9] HeidiSQL https://www.heidisql.com/

[10] Postman https://www.getpostman.com/

[11] Express http://expressjs.com/es/

[12] Ionic Tutorial https://ionicframework.com/docs/intro/tutorial/

[13] Ionic Docs https://ionicframework.com/docs/

[14] Socket.IO https://socket.io/

[15] ChartJS http://www.chartjs.org/

[16] Color Wheel https://color.adobe.com/es/create/color-wheel/

[17] Tutorial Flex CSS https://css-tricks.com/snippets/css/a-guide-to-flexbox/

[18] Tutorial Publicar aplicaciones Ionic http://ionicframework.com/docs/v1/guide/publishing.html