

# Práctica 3. Desarrollo de una aplicación multiplataforma

Noelia Escalera Mejías      Jesús Torres Sánchez

25 de mayo de 2020

## Índice

<b>1. Fase de análisis</b>	<b>2</b>
1.1. Descripción . . . . .	2
1.2. Requisitos funcionales . . . . .	2
1.3. Requisitos no funcionales . . . . .	2
1.4. Listado de las partes interesadas y de las preocupaciones de cada una . . . . .	2
1.5. Listado de criterios de calidad . . . . .	3
1.6. Diagrama de componentes . . . . .	3
<b>2. Fase de diseño - Diagrama de clases de diseño</b>	<b>4</b>
<b>3. Fase de desarrollo</b>	<b>5</b>
3.1. Problemas encontrados con tecnologías finalmente no aplicadas .	5
3.1.1. Angular JS y Firebase . . . . .	5
3.1.2. Firebase y App Engine . . . . .	5
3.1.3. Android JS . . . . .	6
3.2. Tecnologías finalmente usadas . . . . .	6
3.2.1. Node JS . . . . .	6
3.2.2. Express JS . . . . .	6
3.2.3. MongoDB . . . . .	7
3.2.4. EJS . . . . .	7
3.2.5. HTML . . . . .	7
3.2.6. Android Studio . . . . .	7
3.3. Funcionamiento de la aplicación . . . . .	7
3.4. Aplicación para móvil . . . . .	16

Todas las prácticas en <https://github.com/Arelaxe/practicassDS>

## 1. Fase de análisis

### 1.1. Descripción

Se ha decidido implementar una aplicación de votaciones conjuntas siguiendo un modelo cliente-servidor. Por un lado tendremos un servidor que se encarga de recoger los distintos datos de las votaciones y tendremos una página que muestra dichos resultados. Por otro tendremos otra página mediante la cuál los usuarios podrán crear o unirse a salas de votación y/o votar en distintas votaciones.

### 1.2. Requisitos funcionales

- **RF-1.** Alta de usuario.
- **RF-2.** Unirse a votación.
- **RF-3.** Crear una sala de votación.
- **RF-4.** Votar en una votación.
- **RF-5.** Consultar resultados.
- **RF-6.** Seleccionar tipo de votación.

### 1.3. Requisitos no funcionales

- **RNF-1.** La interfaz gráfica será sencilla con el objetivo de facilitar el uso para la mayoría de usuarios.
- **RNF-2.** Al establecer comunicaciones con un servidor web debemos tener en cuenta múltiples aspectos de seguridad tanto en la propia aplicación como en el servidor en sí.
- **RNF-3.** La aplicación tiene que tener facilidad para ser portable.
- **RNF-4.** El servidor debe estar disponible en todo momento para poder acceder a las distintas votaciones en tiempo real.
- **RNF-5.** Habrá una amplia variedad de votaciones para elegir.

### 1.4. Listado de las partes interesadas y de las preocupaciones de cada una

- **Arquitecto.** Sus principales preocupaciones son que el sistema pueda ser portable y multiplataforma.

- **Cliente (Adquiriente).** Que la aplicación sea atractiva para distintos usuarios.
- **Desarrollador.** Conseguir un código legible y fácilmente actualizable.
- **Servicio al cliente.** Contar con una manera fácil de comunicarse con el cliente.
- **Administrador del sistema.**
- **Técnico de pruebas.**
- **Usuarios.** Que la interfaz sea intuitiva y que el servidor esté disponible el mayor tiempo posible.

### 1.5. Listado de criterios de calidad

- **Gestión del producto.** Consideramos el sistema como un producto para evaluar la evolución en su conjunto.
- **Magnitud de cambio.** Si apreciamos esto, se evitarán grandes costos.
- **Dimensión del cambio.** A nivel funcional, de plataforma, de integración y de crecimiento de uso.
- **Probabilidad del cambio.** Hay que ser muy precisos en la apreciación de que los cambios sean realmente necesarios.
- **Temporización del cambio.** Hay que estimar el momento probable para realizar cambios.
- Elegir el momento del cambio equilibrando los costes de desarrollo (mayores a mayor flexibilidad del sistema a desarrollar) y los costes de mantenimiento (mayores a mayor velocidad en el desarrollo).
- Considerar los cambios impuestos por factores externos (fin de la vida útil, cambios en la población de usuarios o perfiles, etc).
- Preservar el conocimiento mediante alguna documentación para cuando cambie el equipo de mantenimiento.
- Probar los cambios (automatizar las pruebas, hacer una buena gestión de la configuración, abstraer procesos repetibles).

### 1.6. Diagrama de componentes

A partir de los requisitos mencionados anteriormente, podemos obtener el siguiente diagrama de cajas y líneas, que muestra las características del sistema a nivel contextual:

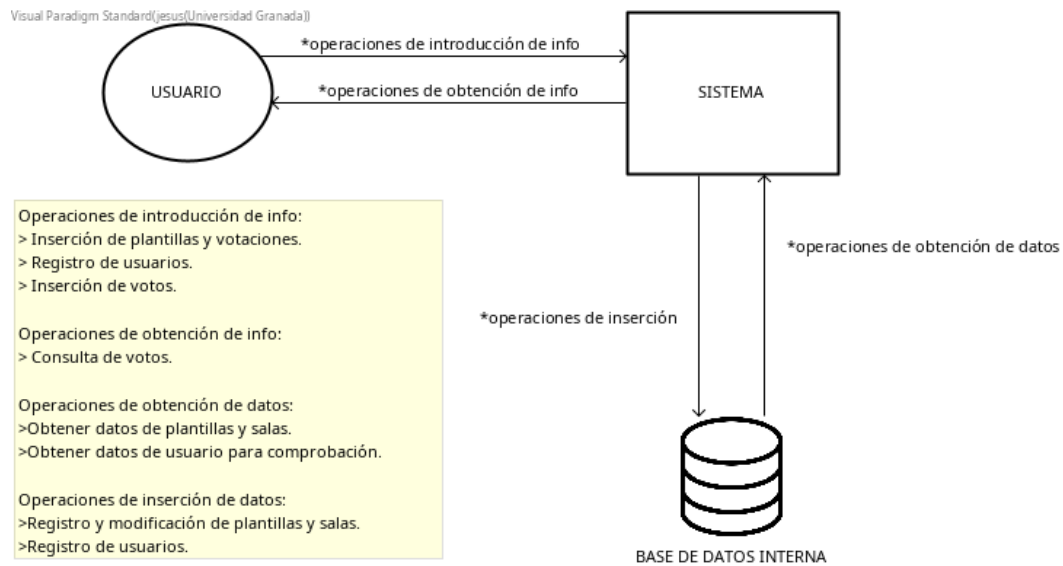


Figura 1: Diagrama contextual

## 2. Fase de diseño - Diagrama de clases de diseño

A continuación se adjunta un diagrama de clases de diseño, dónde se especifican los elementos más importantes del sistema así como las relaciones entre los mismos:

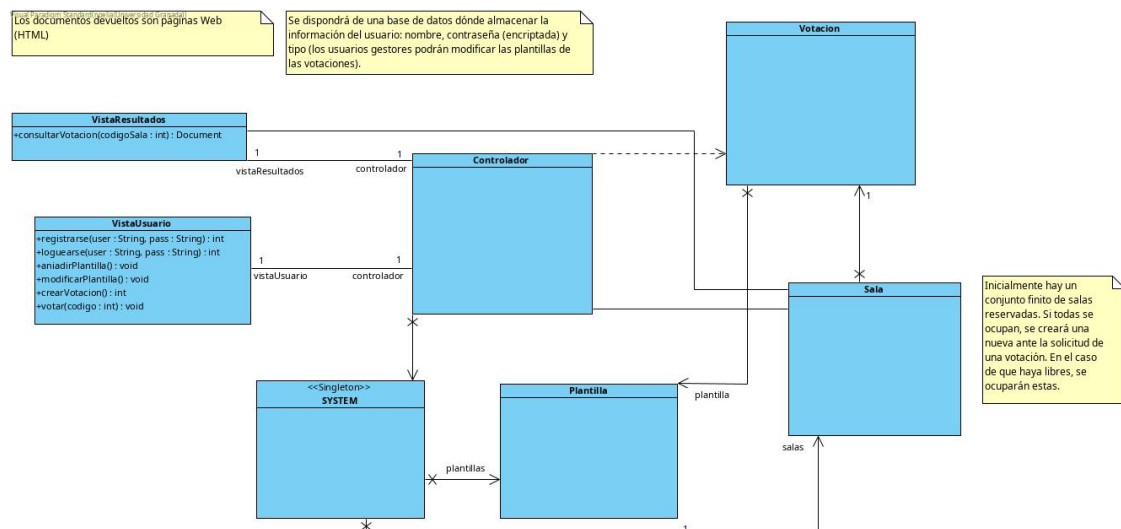


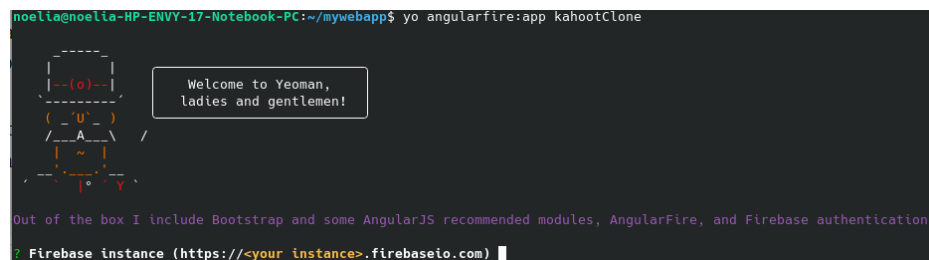
Figura 2: Diagrama de clases de diseño

### 3. Fase de desarrollo

#### 3.1. Problemas encontrados con tecnologías finalmente no aplicadas

##### 3.1.1. Angular JS y Firebase

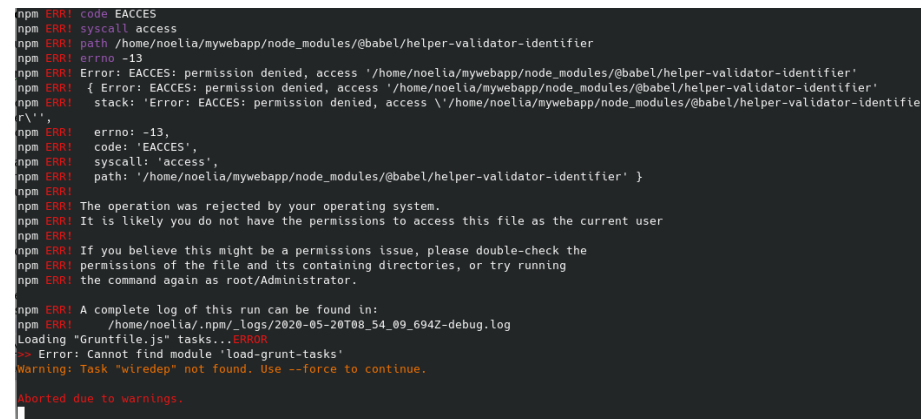
Nuestra primera idea fue desarrollar la aplicación para móvil. En primer lugar, intentamos utilizar Angular JS y Firebase, para ello, intentamos seguir el siguiente tutorial. No nos quedó claro cómo conectar Firebase a la aplicación, ya que no daba muchos detalles. Además, el enlace de la demo de la página está roto. Al llegar al paso de usar el generador angular-fire, te pide una instancia de Firebase, no dejaba muy claro qué había que poner.



```
noelia@noelia-HP-ENVY-17-Notebook-PC:~/mywebapp$ yo angularfire:app kahootClone
Welcome to Yeoman,
ladies and gentlemen!

Out of the box I include Bootstrap and some AngularJS recommended modules, AngularFire, and Firebase authentication.
? Firebase instance (https://<your instance>.firebaseio.com) [ ]
```

Una vez conseguimos averiguar qué nos pedía, ejecutamos la orden y nos dio el siguiente error:



```
npm ERR! code EACCES
npm ERR! syscall access
npm ERR! path /home/noelia/mywebapp/node_modules/@babel/helper-validator-identifier
npm ERR! errno -13
npm ERR! Error: EACCES: permission denied, access '/home/noelia/mywebapp/node_modules/@babel/helper-validator-identifier'
npm ERR! { Error: EACCES: permission denied, access '/home/noelia/mywebapp/node_modules/@babel/helper-validator-identifier'
npm ERR!   stack: 'Error: EACCES: permission denied, access \'/home/noelia/mywebapp/node_modules/@babel/helper-validator-identifie
r\'',
npm ERR!   errno: -13,
npm ERR!   code: 'EACCES',
npm ERR!   syscall: 'access',
npm ERR!   path: '/home/noelia/mywebapp/node_modules/@babel/helper-validator-identifier' }
npm ERR! The operation was rejected by your operating system.
npm ERR! It is likely you do not have the permissions to access this file as the current user
npm ERR! If you believe this might be a permissions issue, please double-check the
npm ERR! permissions of the file and its containing directories, or try running
npm ERR! the command again as root/Administrator.

npm ERR! A complete log of this run can be found in:
npm ERR! /home/noelia/.npm/_logs/2020-05-20T08:54:09_694Z-debug.log
Loading "Gruntfile.js" tasks...ERROR
Error: Cannot find module 'load-grunt-tasks'
Warning: Task "wired" not found. Use --force to continue.

Aborted due to warnings.
```

##### 3.1.2. Firebase y App Engine

Nos pareció interesante la tecnología Firebase e e investigamos sobre cómo conectarnos desde una aplicación. Encontramos el siguiente tutorial. Tras varias horas de trabajo, conseguimos llegar hasta casi el final del tutorial, hasta que llegó un momento en que para seguir se nos pedía la tarjeta de crédito por parte

de Firebase. Se supone que no te hacían ningún tipo de cargo, pero no nos gustó la idea. Además nos apareció el siguiente error al intentar ejecutar el servicio en el servidor local con `$mvn clean package appengine:run`

```
[INFO] -----
[ERROR] /home/noelia/firebase-appengine-backend/src/main/java/com/google/cloud/solutions/flexenv/common/Message.java:[23,17] cannot
access java.util.Map
bad class file: /modules/java.base/java/util/Map.class
class file has wrong version 55.0, should be 53.0
Please remove or make sure it appears in the correct subdirectory of the classpath.
[INFO] 1 error
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 2.676 s
[INFO] Finished at: 2020-05-20T11:12:22+02:00
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.7.0:compile (default-compile) on project backend:
Compilation failure
[ERROR] /home/noelia/firebase-appengine-backend/src/main/java/com/google/cloud/solutions/flexenv/common/Message.java:[23,17] cannot
access java.util.Map
[ERROR] bad class file: /modules/java.base/java/util/Map.class
[ERROR] class file has wrong version 55.0, should be 53.0
[ERROR] Please remove or make sure it appears in the correct subdirectory of the classpath.
[ERROR]
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

### 3.1.3. Android JS

También intentamos usar Android JS, para ello seguimos este tutorial. Tras seguirlo hasta el final, daba este error al empaquetar:

```
noelia@noelia-HP-ENVY-17-Notebook-PC:~/myapp$ androidjs b -f
Invalid project type: undefined
```

## 3.2. Tecnologías finalmente usadas

Finalmente decidimos hacer la aplicación para web, hemos usado las siguientes tecnologías:

### 3.2.1. Node JS

Es un entorno de ejecución multiplataforma, de código abierto para la capa del servidor, basado en JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Para utilizarlo nos hemos servido de lo aprendido en la asignatura Desarrollo de Sistemas Distribuidos, así como en distinta documentación de internet.

### 3.2.2. Express JS

Es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Nos ha servido de gran utilidad para manejar las sesiones de los usuarios, así como para manejar las redirecciones, entre otras cosas. Nos han

sido de gran utilidad los tutoriales que proporciona su página web: <https://expressjs.com/es/>.

### **3.2.3. MongoDB**

Se trata de un sistema de base de datos NoSQL, orientado a documentos y de código abierto, que es muy fácil de usar junto con Node. Lo hemos usado para guardar las cuentas de los usuarios, las encuestas, etc. Para usarlo nos hemos servido de lo aprendido en Desarrollo de Sistemas Distribuidos, así como de bibliografía de internet cuando nos surgían dudas.

### **3.2.4. EJS**

EJS o Embed Javascript Templating, es un motor de plantillas, que permite generar HTML con JavaScript. Nos ha sido muy útil la ayuda de la página oficial: <https://ejs.co/>.

### **3.2.5. HTML**

Por supuesto, hemos tenido que usar HTML como base para las distintas páginas de la aplicación.

### **3.2.6. Android Studio**

Para completar, se decidió hacer una pequeña aplicación para Android. para ello se usó Android Studio, el entorno de desarrollo integrado oficial para Android. Nos servimos de distintos tutoriales de la web para aprender a usarlo.

## **3.3. Funcionamiento de la aplicación**

Partiendo de los requisitos y criterios de calidad mencionados anteriormente, se ha decidido crear una página web usando las tecnologías previamente comentadas.

Al acceder a la página web, lo primero que nos encontramos es la opción de log-in, ya que es un requisito imprescindible estar identificado previamente para poder acceder al sistema. Adicionalmente, en el caso de que no estemos registrados, nos proporcionará una alternativa para poder registrarnos.

Este proceso se llevará a cabo gestionando las distintas peticiones en el servidor por medio de Express, y comprobando las credenciales a partir de la información almacenada en la base de datos.

Una vez nos hemos identificado o registrado, podremos acceder al menú principal. Este menú nos proporciona una serie de operaciones:

- Una operación de acceso a la sala. Para poder realizar esta operación debemos proporcionar un código de sala válido. Una sala es válida si tiene

asociada una votación con una fecha de finalización posterior a la fecha actual y una fecha de inicio anterior a la actual. Es decir, la votación de la sala actual debe estar en curso para que podamos acceder a la misma.

- Una operación de votación en una sala. Para poder votar en la sala, además de cumplir los requisitos de acceso mencionados anteriormente, también debemos comprobar que el usuario presenta permisos para participar en esta votación.
- Una operación para añadir un usuario a la lista de miembros de una votación. El usuario podrá realizar esta operación sobre todas las votaciones creadas por el mismo.
- Una operación para añadir una plantilla, especificando: el nombre de la plantilla, las distintas preguntas y las alternativas para cada pregunta.
- Una operación para añadir una votación, especificando: el nombre de la votación, la plantilla a usar, la fecha de inicio de la votación y la fecha de finalización. En el caso de que las fechas proporcionadas no sean válidas, generará los avisos correspondientes.
- Adicionalmente, se permitirá que el usuario se desloguee, destruyendo la sesión correspondiente.

Para poder desarrollar este proceso correctamente, requerimos disponer de información almacenada, sobre todo en cuanto a los usuarios, las votaciones asociadas a las salas y las plantillas disponibles. Para ello, haremos uso de una base de datos MongoDB, cargando cada vez que arranca el sistema la información correspondiente y actualizándola cada vez que se modifique. Por ejemplo, en el caso de las votaciones asociadas a las salas, cada vez que se cree una nueva votación, se comprobarán todas las salas asignadas de forma secuencial: si una sala tiene asociada una votación que ya ha expirado, podemos asignarle dicha sala a la votación en cuestión.

En las siguientes capturas de pantalla se muestra el ejemplo de una votación sencilla, permitiéndonos observar de forma práctica el funcionamiento del sistema:



En primer lugar nos registramos:

**- VOTING SYSTEM -**

**Regístrate**

Volver al menú de [log in](#).

Username:

Password:

Figura 3:

O en el caso de que estuviésemos previamente registrados, iniciamos sesión:

**- VOTING SYSTEM -**

**Login**

¿No está registrado? Puede registrarse pulsando [aquí](#).

Username:

Password:

Figura 4:

Una vez nos identificamos, entramos a la página principal:

**- VOTING SYSTEM -**

**Página principal**

Sesión iniciada correctamente

¿Quiere desloguearse? Pulse [aquí](#).

Seleccione el código de la sala:

VotarAcceder a sala

Crear votacion  
Crear plantilla  
Consultar mis votaciones

Figura 5:

Creamos una nueva plantilla para usar en las votaciones:

**- VOTING SYSTEM -**

**Nueva plantilla**

Indique el nombre de la plantilla:

Indique el número de preguntas de la plantilla:

Indique el número de alternativas de la plantilla:

Establecer datos

Pregunta 1:  

Esta es la pregunta 1

Alternativa 1:

Alternativa 2:

Pregunta 2:  

Esta es la pregunta 2

Alternativa 1:

Alternativa 2:

Establecer votación

Figura 6:

Creamos una nueva votación a partir de la plantilla creada anteriormente:

**- VOTING SYSTEM -**

**Nueva votación**

Nombre

Seleccione una plantilla:

Hora de inicio

Hora de finalización

Volver al [menú principal](#)

Figura 7:

Una vez tengamos la votación creada, se genera un mensaje con el código de la sala creada:

**- VOTING SYSTEM -**

**Página principal**

El código de la sala es 1

¿Quiere desloguearse? Pulse [aquí](#).

Seleccione el código de la sala:

Figura 8:

Añadimos un usuario nuevo a la votación creada: de esta forma ahora podrán votar tanto *ejemplo* como *ejemplo2*.

**- VOTING SYSTEM -**

**Mis votaciones**

Votacion Nueva

Figura 9:

**- VOTING SYSTEM -**

**Página principal**

El usuario se ha registrado en la votación correctamente

¿Quiere desloguearse? Pulse [aquí](#).

Seleccione el código de la sala:

Figura 10:

Votamos como usuario *ejemplo*. Es importante que una vez el usuario haya votado, no podrá volver a votar (se notificará el mensaje de error correspondiente). Además, también se muestra simultáneamente el resultado de las votaciones en la sala.

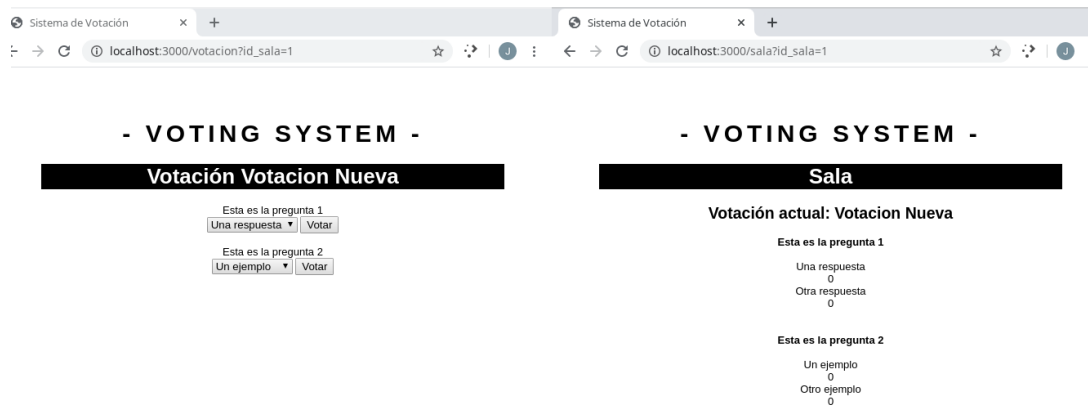


Figura 11:

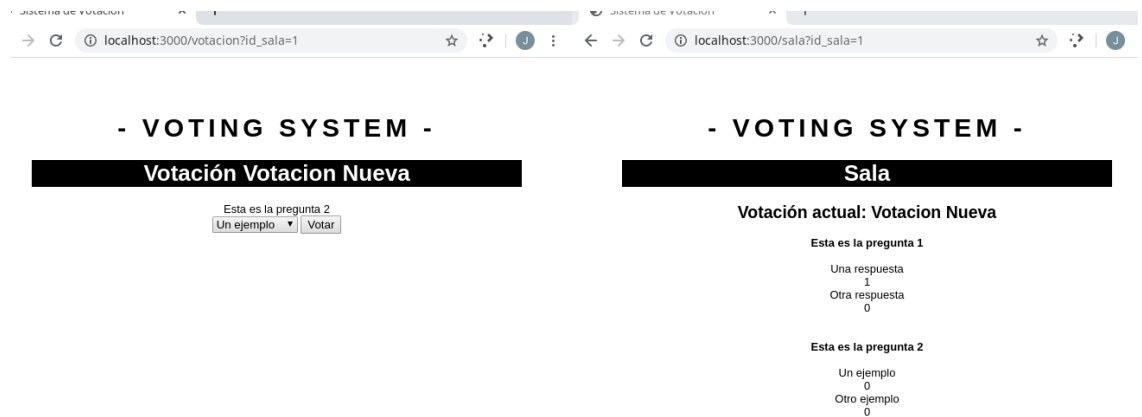


Figura 12:

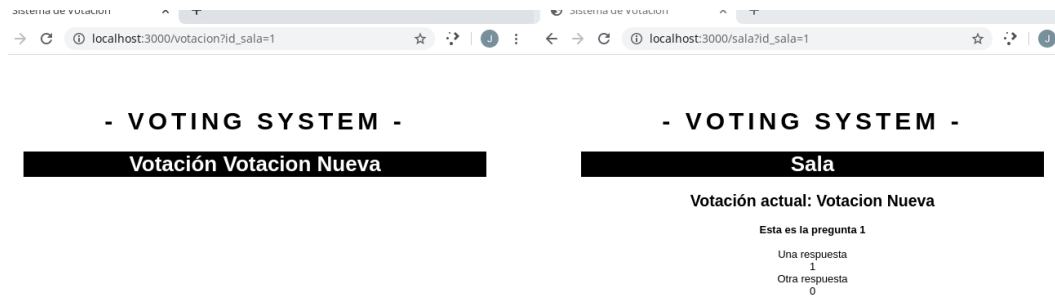


Figura 13:

A continuación nos deslogueamos, iniciamos sesión como *ejemplo2*, y procedemos a votar:

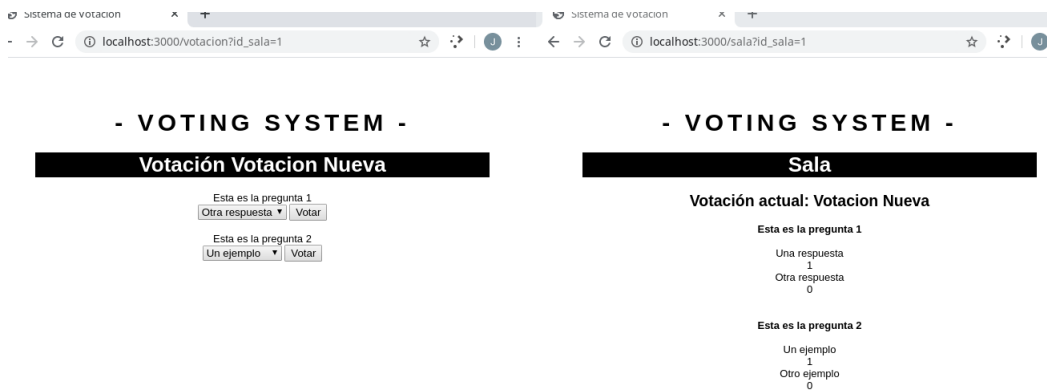


Figura 14:

## - VOTING SYSTEM -

### Sala

#### Votación actual: Votacion Nueva

Esta es la pregunta 1

Una respuesta

1

Otra respuesta

1

Esta es la pregunta 2

Un ejemplo

2

Otro ejemplo

0

Figura 15:

Finalmente, si iniciamos sesión como otro usuario distinto a *ejemplo1* o *ejemplo2*, se notificará el error correspondiente:

## - VOTING SYSTEM -

### Página principal

El usuario no puede votar en esta votación.

¿Quiere desloguearse? Pulse [aquí](#).

Seleccione el código de la sala:

Votar

Acceder a sala

Crear votacion

Crear plantilla

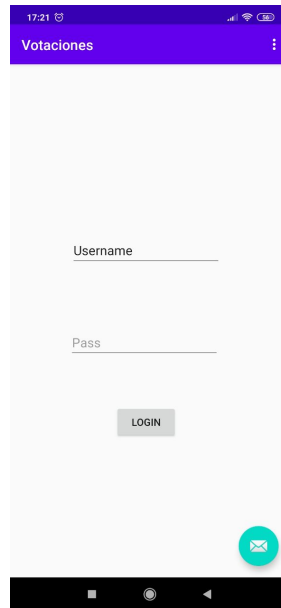
Consultar mis votaciones

Figura 16:

### 3.4. Aplicación para móvil

Para completar un poco la práctica, decidimos hacer una aplicación básica para Android con Android Studio. Se trata de una versión bastante recortada de la aplicación web, ya que ha sido nuestro primer desarrollo para Android. Vamos a ver un ejemplo de uso de la aplicación:

Nada más entrar en la aplicación aparecerá esta pantalla de login:



Tenemos los siguientes usuarios definidos en el código:

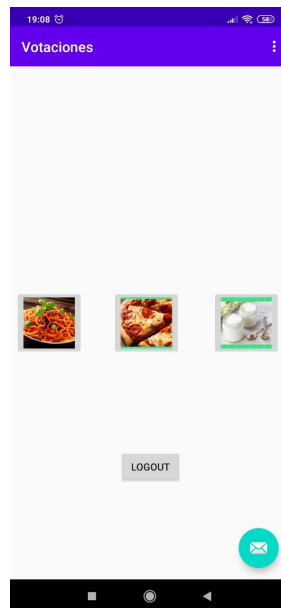
```
public static void addUsuarios(){  
    usuarios.add(new Usuario ( nombre: "noe", pass: "pass"));  
    usuarios.add(new Usuario ( nombre: "alguien", pass: "pass"));  
    usuarios.add(new Usuario ( nombre: "usu", pass: "pass"));  
}
```

Si nos equivocamos con el login nos mostrará un mensaje de error:





Si introducimos un usuario correcto accederemos a la pantalla de elegir una votación:



Tenemos tres votaciones distintas, una vez que votamos, no podremos volver a votar con el mismo usuario, tendremos que votar con otro usuario distinto (para ilustrar mejor esto, mirar el vídeo demo incluido):

