Práctica 3. Desarrollo de una aplicación multiplataforma

Noelia Escalera Mejías Jesús Torres Sánchez

20 de mayo de 2020

Índice

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.

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.

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.

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.

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 velociad 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).

6. 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:

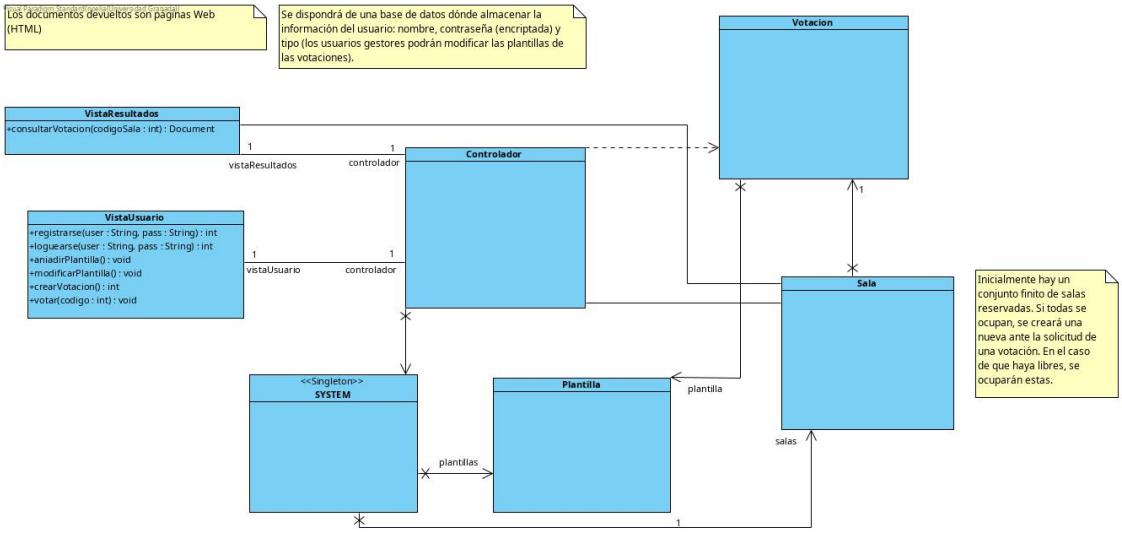
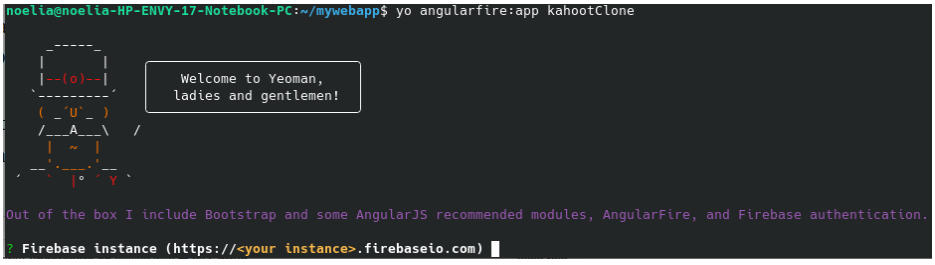


Figura 1: Diagrama de clases de diseño

7. Problemas encontrados con tecnologías finalmente no aplicadas

7.1. 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.



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!
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!
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 "wiredep" not found. Use --force to continue.

Aborted due to warnings.
```

7.2. 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] canno
t 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] ------------------------------------------------------------------------
[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] canno
t 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
```

7.3. 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
```

8. Tecnologías finalmente usadas

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

8.1. 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 vasado 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.

8.2. Express JS

Es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto solido 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/`.

8.3. 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.

8.4. EJS

EJS o Embebed 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/`.

8.5. HTML

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