

SocialUGR

Marcos Castillo Trigueros
Jose Armando Albarado Mamani
Luis Escobar Reche

Desarrollo del Software
3 GII



1) Intro:

Este proyecto surgió de la idea de crear una red social para las personas de la UGR, en él intentamos establecer las bases mínimas de una red social utilizando tecnologías web que hoy en día se están utilizando bastante, NodeJS, además consideramos que para llevar a cabo la red social necesitamos una base de datos muy eficiente, que permite bastante flexibilidad y seguridad, es por ello que hemos optado por MongoDB.

2) Desarrollo separado:

2.1) Login:

El login fue tarea de Luis Escobar Reche. A continuación describe su propia experiencia.

El login se empezó como mis compañeros haciendo el estilo css y la plantilla html aproximada. Esto no fue ningún problema ya que tenemos bases de HTML y CSS gracias a la asignatura de SIBW de tercero.

Una vez terminado empecé a documentarme sobre Nodejs, y una vez entendidos los conceptos básicos empecé a mirar a fondo el tema de sesiones en Nodejs y como crear un login y un signup.

Tras mirar muchos foros que distaban en cuanto a cómo hacerlo, acabé en Youtube en un canal (Referencia1) que hace videos de programación y que tenía varios vídeos de Nodejs y sobre todo tenía un par de videos enseñando a hacer un login en Nodejs. En este vídeo se enseñaba a hacer un login local y también a hacerlo con Google o Facebook. Opté por hacerlo solo local.

El módulo que gestiona sesiones y hace todo este proceso medianamente automático se llama "passport", en un principio planteamos que el usuario pudiera logearse dando su username o su email. El problema es que passport solo admite username y password, por lo que teníamos que elegir una de las dos opciones, así que elegí solo username. También hacía falta una base de datos de mongoDB, por lo que usamos el paquete de "mongoose", que nos permitía hacer esquemas de los modelos de datos y consultar o introducir datos de forma sencilla. Una vez tenía mi esquema de usuarios y las sesiones surgieron algunos errores puntuales pero relativamente sencillos de arreglar. Pude comprobar que si que dejaba logearse, abría bien la sesión y la cerraba correctamente cuando configuré el logout. Quedaba el signup.

El registro fue tarea más difícil, ya que no era un método post convencional, tenía que guardar datos en mi BD. Tras probar varias veces conseguí entender cómo funcionan los formularios con POST, en el login fue sencillo implementarlo, simplemente buscaba un usuario y si lo encontraba abría sesión. El problema de nuestro registro era que editaba usuarios y perfiles de usuario, es decir, dos acciones en un mismo post. Registrar el usuario con su contraseña fue medianamente sencillo así que aproveché y metí un encriptador de contraseñas para almacenar en la BD la contraseña encriptada, este módulo se llama "bcryptjs". Tras mirar en muchos foros y no encontrar solución se me ocurrió una idea, ¿que pasaba si el formulario llamaba a una ruta POST y esta ruta POST llamaba a otra ruta POST? Tras probarlo comprobé que el redirect no servía, así que busqué como hacer que eso funcionara. La forma correcta era hacer `res.redirect(307, '/signup');` de esta forma el

redirect se hace con el método POST y funcionaba correctamente, el 307 por lo visto se usa para indicar que es una redirección temporal aunque no aporta ninguna función por lo que se podría poner cualquier número en teoría. Con esto en mente hice que el formulario llamara a crear perfil primero (ya que si había algún error en perfil no debía crear el usuario) y este hacia el redirect con POST para crear el nuevo perfil, haciendo las comprobaciones necesarias. Una vez creado el usuario se crea la sesión y entra al feed.

En resumen, la página principal tiene un apartado para registro y otro para identificarse. El registro es mediante un formulario que llama a la ruta `/perfilNuevo`, esta introduce en la base de datos haciendo comprobaciones los datos del nuevo usuario, y si termina sin errores llama con otro POST a `/signup`. Esta ruta crea al usuario si no existe, aunque tampoco se llega a llamar nunca a esta ruta si el usuario existe, tras crearlo lo mete en la base de datos y inicia la sesión, redirigiendo a `/feed`.

2.2) Perfil:

El perfil fue tarea de Marcos Castillo Trigueros. A continuación describe su propia experiencia.

En primer lugar se definió la estructura visual en grupo utilizando HTML y CSS, tarea relativamente fácil ya que en otras asignaturas se ha practicado.

A posteriori repartimos las funcionalidades y empezamos a trabajar por separado con nodeJS, JavaScript y EJS (view engine para JS), la primera toma de contacto con nodeJS la hicimos en otra asignatura, aunque muy por encima, por lo que instalar los módulos necesarios para el proyecto y recibir peticiones REST del navegador no supuso un gran reto. El principal problema que he encontrado en la elaboración de esta parte del proyecto eran casos no tenidos en cuenta, que se mostraban como errores cuando realmente era que no se habían considerado y tratado, como el seguir infinitamente a un usuario, modificar perfiles ajenos, etc...

Mencionar que el usar una base de datos no relacional es algo que me ha tenido un poco perdido a la hora de almacenar datos, consultarlos y modificarlos. La falta de costumbre al tratar con objetos JSON en lugar de clases definidas, aunque cuando se le cogió el punto fue de gran utilidad el hecho de no depender de una estructura fija para tratar con los modelos. Mongoose fue un módulo de gran ayuda en el aspecto de la base de datos, permitiéndonos conectarnos a la base de datos de forma sencilla, crear esquemas fácilmente y utilizar los modelos con métodos predefinidos de consulta y modificación.

He utilizado como herramienta de apoyo para el diseño Bootstrap 4 para algunos elementos como los formularios de modificación del perfil y los botones de follow/unfollow y una librería de iconos (FontAwesome) que proporciona elementos visuales predefinidos y adaptables mediante CSS.

El papel principal de este apartado es mantener la información básica social del usuario, imágenes personalizadas, sus posts, seguidores y perfiles a los que sigue.

En conclusión, la parte del perfil me ha permitido aprender nuevas herramientas con las que no estaba familiarizado y mejorar algunas que ya tenía un ligero conocimiento añadiendo elementos/librerías nuevas que facilitan la calidad del producto y ahorran tiempo.

2.3) Feed:

El Feed fue tarea de Jose Armando Albarado Mamani. A continuación describe su propia experiencia.

Para definir la parte de la vista he definido varios he creado el HTML y el css (Feed.html y Feed.css), en el Feed.css realizo toda la configuración de el estilo de la página principal, respecto a colores, fondos, etc. En el archivo HTML y con ayuda de EJS (Embedded JavaScript) para definir todas las estructuras.

En primer lugar se aprecia en el html dos botones con los que se puede crear un post de tipo textual, el cual contendrá únicamente texto, y por otro lado también se puede crear un post multimedia en este caso se debe añadir una imagen el cual aparecerá cuando se cree el post. Ambas peticiones se realizan empleando Rest.

Cuando se carga a página principal de feed, se cargan todos los posts de las personas a las que sigue el usuario que ha iniciado sesión, además se incluyen los posts del usuario propio, ésta lista de posts se ordenan en función de la fecha de publicación, por tanto en primer lugar siempre aparecerán los posts publicados recientemente. Para comunicar estos posts a la plantilla principal utilizo las plantillas de EJS. Para obtener información acerca de los posts de las personas a las que se sigue y de los tuyos propios se realiza una petición empleando esquemas de mongoose a la base de datos.

Cada uno de los posts está formado por una foto de perfil, información acerca del usuario que ha realizado el post, una imagen y una descripción si el post es de tipo imagen, en el caso de que sea un post tipo textual es una sección de texto. En la parte inferior de cada una de los posts se encuentra la sección de interacción, un botón de comentarios y un botón de likes, cada uno de ellos contiene un contador debajo de la imagen. Si se da click en el icono de comentarios se abre una sección para introducir nuevos comentarios, mientras que si se da click en el icono de like se dará me gusta al post. Tanto el envío del post como el envío del like se realiza empleando AJAX, por tanto se realiza en background y no es necesario recargar la página para que se almacenen los datos en la base de datos. Si se vuelve a dar click en el icono de comentario y se tiene abierto la sección de nuevo comentario se oculta. Cada vez que se realiza un comentario o se da like a un comentario los cambios se reflejan en la web, se ve el nuevo comentario añadido y se incrementa el contador, y en el caso del like el icono de la imagen cambia de color y se actualiza el contador de likes.

Cada uno de los comentarios de los posts están ordenados por fecha, de modo que si se introduce un nuevo comentario se verá añadido al final de la sección de comentarios. Además cada uno de ellos tiene un campo que indica el tiempo que ha transcurrido desde su inserción.

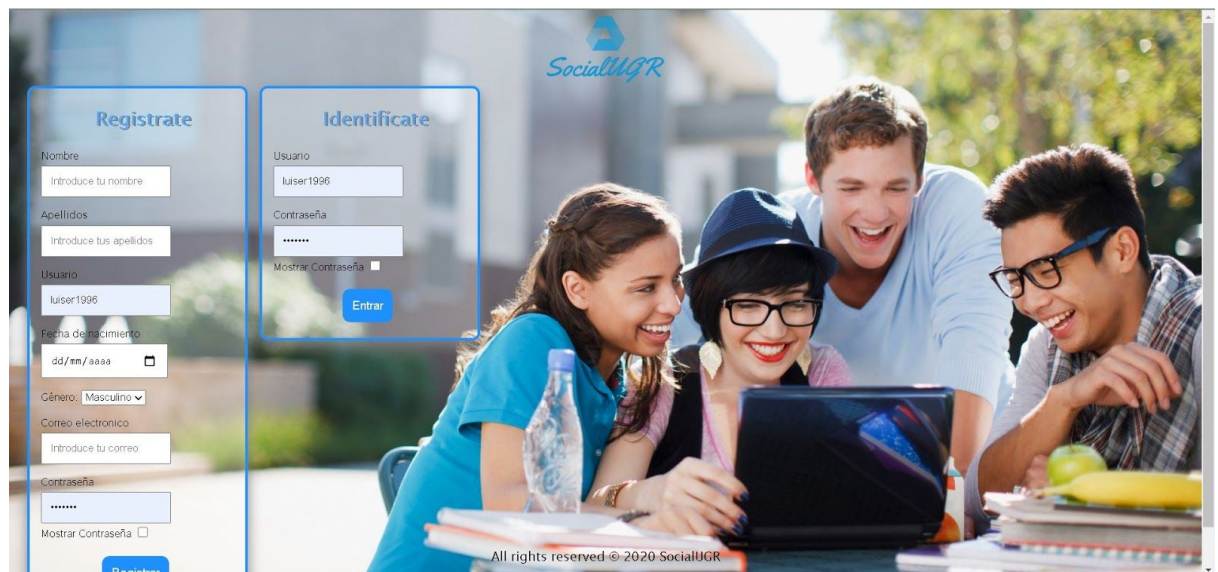
Conclusión: Me ha llevado bastante trabajo llevar a cabo y que funcione bien esta parte, pues he empleado AJAX y cada uno de los post es una copia de un único bloque n

veces, es decir, si hay 20 posts se realizan 20 copias de un bloque, por tanto, requiere bastante trabajo poder realizar esto y que se hagan todas las peticiones de AJAX de forma correcta, pues todos comparten una misma estructura. Para solucionar este problema he hecho uso en todo momento del `idPost` para poder diferenciar un post de otra copia, aprovechando este ID si he podido llevar a cabo todo perfectamente y funcionando correctamente. He aprendido muchísimo acerca de NodeJS, MongoDB, AJAX y JQuery, por tanto, creo que a pesar de que he dedicado muchísimas horas he conseguido mi objetivo y además creo que ha quedado algo bastante bueno.

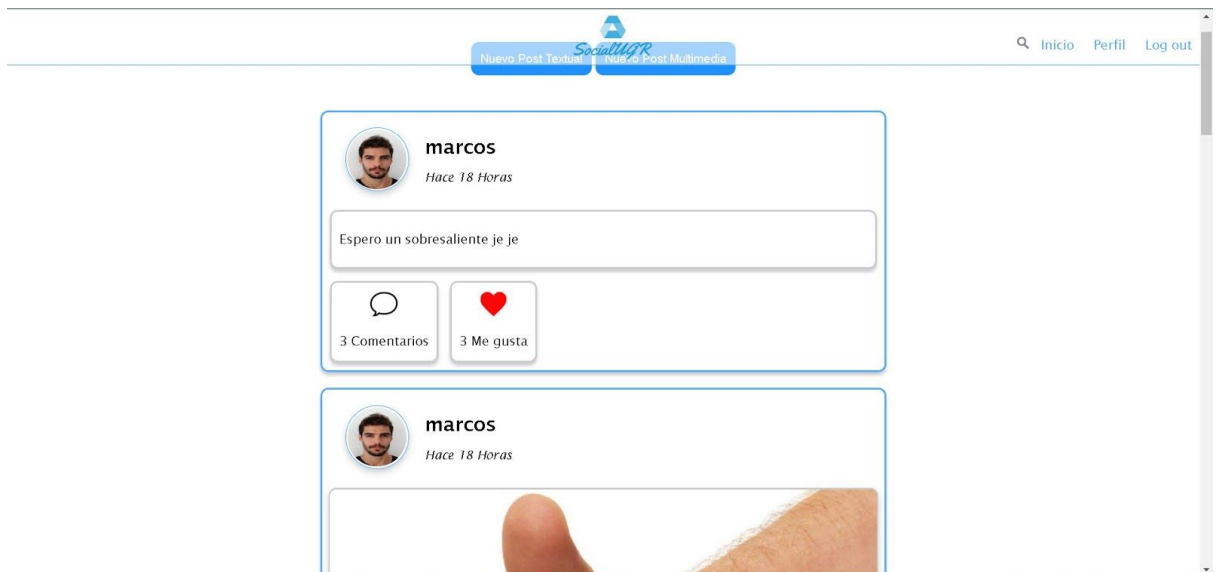
3) Desarrollo conjunto

Unimos todas las partes en una única rama (`develop`) y resolvimos los conflictos generados al haber desarrollado el proyecto por partes separadas. Una vez solucionados todos los errores encontrados obtuvimos el proyecto final que adjuntamos, del cual se han sacado las siguientes imágenes:

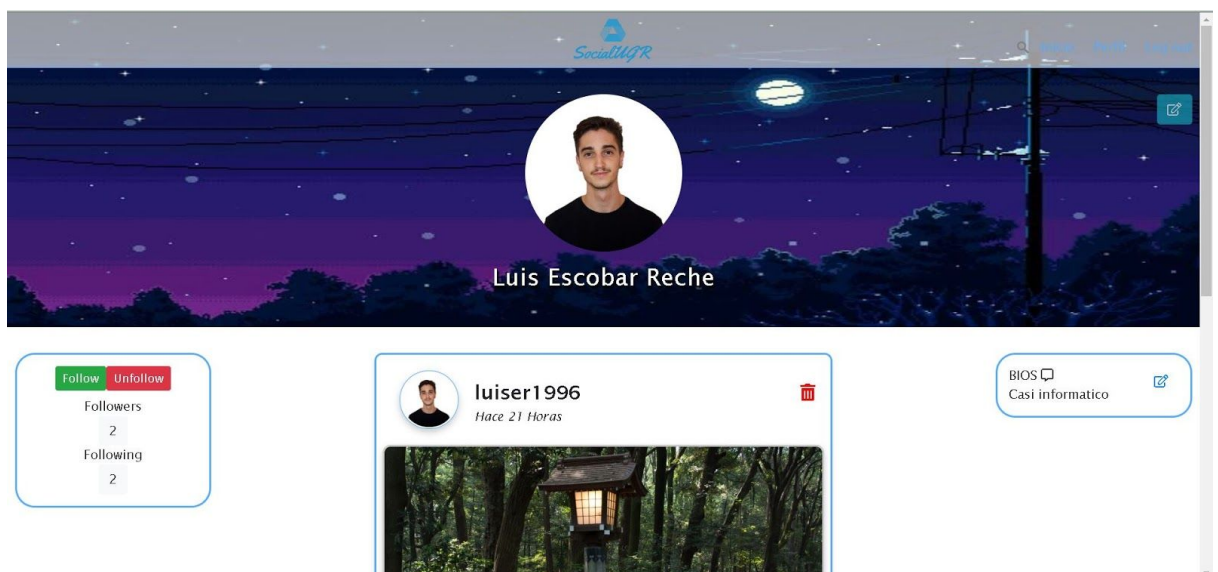
Login:



Feed:



Perfil:

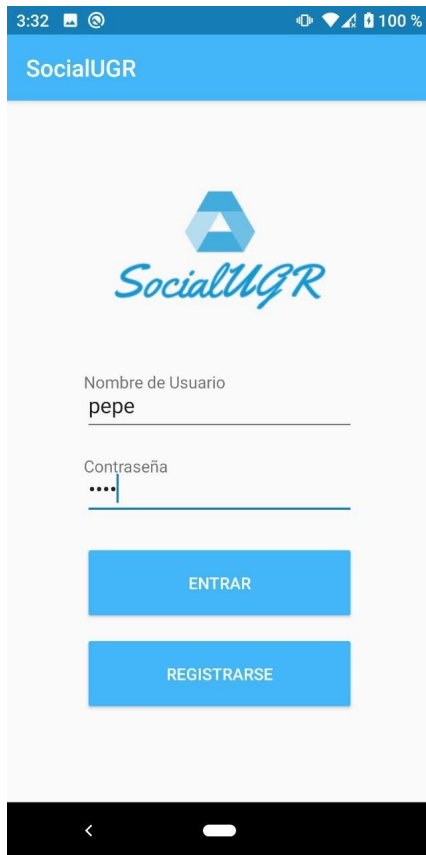


4) Aplicación de Android:

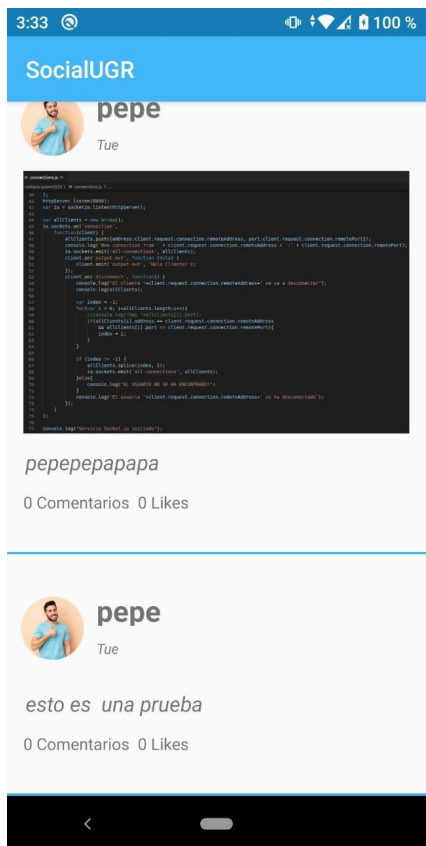
Nuestro enfoque fue más orientado a la aplicación web que a android, por lo que dedicamos más tiempo a la web. Nuestra aplicación Android consiste en algo sencillo que comunica con el servidor web. Tiene el login y el signup, que hacen su función, un feed de posts y el perfil con la información del usuario.

Así se ve nuestra app:

Login:



Feed:



Registro:

13:07 [icons] 100 %

SocialUGR

Nombre Genero ▾

Introduce tu nombre

Apellidos

Introduce tus apellidos

Usuario

Introduce tus username

Email

Introduce tu email

Fecha de Nacimiento

Selecciona tu fecha de nacimiento

Contraseña

Introduce tu contraseña



REGISTRARSE

< [home] [back]

Perfil:

13:55 [icons] 100 %

SocialUGR

Nombre
PerfilAndroid

Apellidos
V1

Nombre de Usuario
perfilandroid

Fecha de Nacimiento
Sat Jan 07 01:00:00 GMT+01:00 1995

Email
perfilandroid@gmail.com

Bios
La vida son dos diaaas! Aprovecha tu

< [home] [back]

5) Herramientas utilizadas:

Visual Studio Code

GitKraken

Git

Bootstrap 4

NodeJS

Express

MongoDB

Mongoose

Passport

Bcryptjs

Nodemon

Multer

AJAX

JQuery

EJS

Connect-flash

Path

Morgan

Fs

Android Studio

Picasso

Retrica

6) Referencias:

- Canal de Youtube:
<https://www.youtube.com/channel/UCX9NJ471o7Wie1DQe94RVlg>
- Video de Youtube:
<https://www.youtube.com/watch?v=-bl0diefasA>
- <https://nodejs.org/es/docs/>
- <https://expressjs.com/es/4x/api.html>
- <https://getbootstrap.com/docs/4.4/getting-started/introduction/>
- <https://docs.mongodb.com/manual/>
- <https://mongoosejs.com/docs/>
- GitHub del proyecto:
<https://github.com/marcoscastillo92/SocialUGR>