Aplicación web: Classcity

Mario Fernandez Guerrero

m.fernandezgue@alumnos.urjc.es

18 de julio de 2018

Índice

Introducción

- Aplicación de gestión de clases particulares.
- Aplicaciones que me han servido de inspiración.
- Arquitectura de una aplicación web

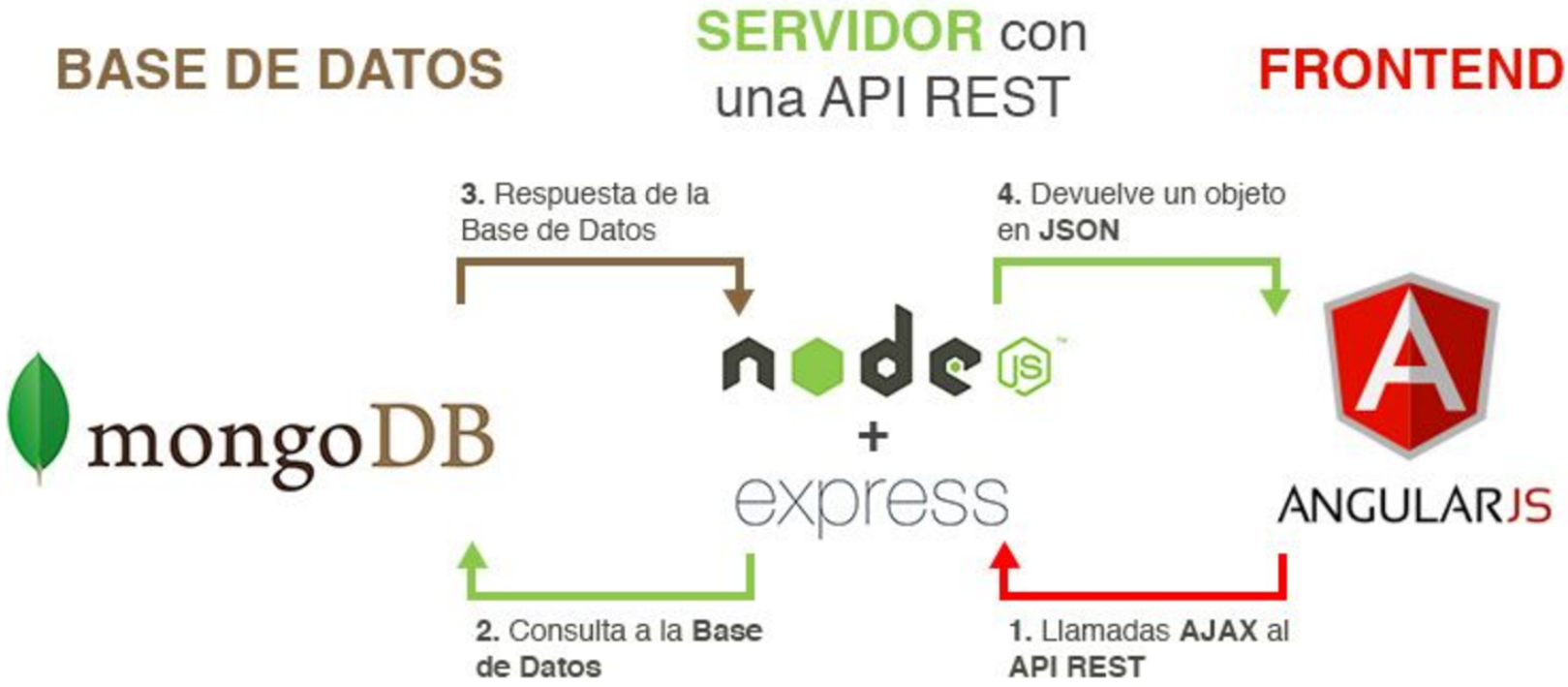
Objetivos

Objetivo general: Desarrollo y despliegue de una aplicación web que permita el contacto entre alumnos y profesores.

Subobjetivos:

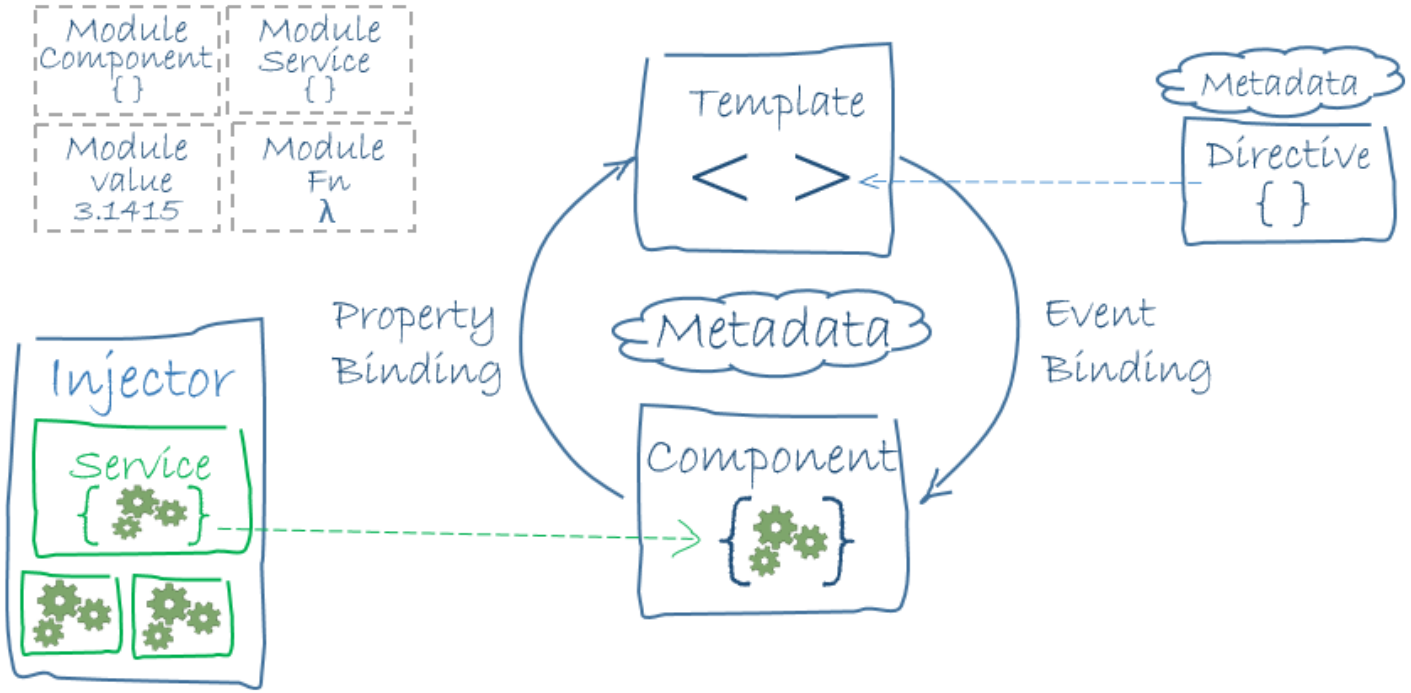
1. Desarrollo del lado cliente de la aplicación.
2. Desarrollo del lado servidor de la aplicación.
3. Implementación de la base de datos.
4. Despliegue en la red.

Infraestructura



Cliente

Angular framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.



Angular

Podemos identificar los 8 bloques principales de una aplicación web con Angular:

1. **Módulos**
2. **Componentes**
3. **Plantilla**
4. **Metadatos**
5. **Data Binding**
6. **Directiva**
7. **Servicio**
8. **Inyección de dependencias**

Servidor

Node: entorno en tiempo de ejecución multiplataforma, basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos

Express: entorno de Node, esta diseñado para construir aplicaciones web y APIs.

1. **Start Server**
2. **Rutas**
3. **Controladores**
4. **Gestión del chat**

Base de datos

MongoDB sistema de base de datos NoSQL orientado a documentos BSON.

1. **Modelos**
2. **Moongoose**

Funcionalidades destacables en la aplicación

1. Login del alumno y del profesor con credenciales cifradas.
2. Registro del alumno y del profesor con credenciales cifradas.
3. Buscar al profesor por distancia, curso y asignatura.
4. Ir al perfil del profesor.
5. Actualizar imagen de perfil del profesor.
6. Enviar solicitud de contacto a un profesor.
7. Chat.

Login del alumno y del profesor con credenciales cifradas.

1. Petición AJAX con método POST.
2. Servidor procesa el body de la petición.
   - Si el usuario esta registrado retorna un 201 con token y con fecha de expiración 5 minutos.
   - Si el usuario no esta registrado retorna un 401 'A user with that username already exists'.
3. El cliente procesa la respuesta del servidor, guardando en cache el token durante 5 minutos.

Registro del alumno y del profesor con credenciales cifradas.

1. Petición AJAX con método POST.
2. Servidor procesa el body de la petición.
   - Si el usuario consigue registrarse retorna un 201 con token cuya fecha de expiración es de 5 minutos.
   - Si el usuario ya existe retorna un 400 con el siguiente mensaje: 'A user with that username already exists'.
   - Si el usuario no introduce algún campo obligatorio el servidor responde con un 400.
3. El cliente procesa la respuesta del servidor, guardando en cache el token.

Buscar al profesor por distancia, curso y asignatura.

1. Usuario completa parametros de busqueda y le da click a buscar.
2. Petición AJAX con método POST.
3. Servidor procesa el body y busca en la base de datos aquellos profesores que cumplan las características que se solicitan en el body.
4. El cliente procesa la respuesta del servidor y los pinta en el mapa.

Ir al perfil del profesor.

1. Petición AJAX con método GET;
2. Servidor busca en la base de datos el profesor con su id.
3. El cliente procesa la respuesta del servidor y pinta los datos del profesor en la ficha técnica.

Actualizar imagen de perfil del profesor.

1. Utilización de la librería 'FileUploader' en la parte cliente para poder enviar ficheros al servidor.
2. Utilización de la dependecia 'multer' para la recepción de ficheros en el servidor.
3. Guardamos en la base de datos del profesor el 'path' de la nueva imagen.

Enviar solicitud de contacto a un profesor.

1. Cuando el alumno hace click en el botón 'Enviar notificación'
2. Se envia petición AJAX con metodo POST.
3. Actualización en la base de datos del profesor en el campo 'notification'.
4. El profesor cuando accede a su perfil verá una petición de contacto del alumno y podrá aceptar o no la solicitud.

Chat

1. Cuando un profesor acepta una solicitud de contacto de un alumno, le da derecho a poder chatear con el.
2. El profesor tiene que estar conectado para que los alumnos puedan hablar con el.
3. Los alumnos se conectan a la sala donde el profesor esta escuchando.
4. Todos los mensajes del chat son enviados por websocket al servidor.
5. El servidor es el encargado de reeviar ese mensaje a todos los miembros que están en la sala.

Despliegue en la nube: AWS

1. Crear una instancia en AWS
2. Conectarnos por ssh a nuestra maquina virtual.
3. Correr el lado cliente en modo producción y copiar carpeta dist en la carpeta httdocs
4. Conseguir un dominio, en nuestro caso www.classcity.es
5. Correr lado servidor
6. Configurar servidor apache para redireccionar peticiones.

Conclusiones

- **Conclusión:** El objetivo general de desarrollar una aplicación web y desplegarla en la red se ha cubierto satisfactoriamente.
- **Trabajos Futuros:**
  1. WebRTC para videoconferencia.
  2. Valoración de los Profesores.
  3. Añadir calendarios.
  4. Añadir un método de pago.

Enlaces

- Mediawiki: `https://jderobot.org/Graylinx-tfg`
- Repositorio:
  `https://github.com/RoboticsURJC-students/2016-tfg-Mario-Fernande`