Proyecto
Truequéalo

Grupo
The Red Chicken

"ARQUITECTURA TÉCNICA"

Hito: 2
Fecha entrega: 30-01-2017
Versión: 1

Componentes:
- Juan Francisco Bustos Correas
- Pablo Serna Martínez
- Yolanda Torregrosa Hernández
- Alejandro Torres Mateu
- Raquel Yuste Torregrosa

Contenido

1. Introducción

En este documento se describen las tecnologías utilizadas tanto en el front-end como en la parte del servidor: lenguajes de programación, marcado y diseño. También se describen los frameworks y plugins utilizados y su finalidad.

2. Lenguajes de programación

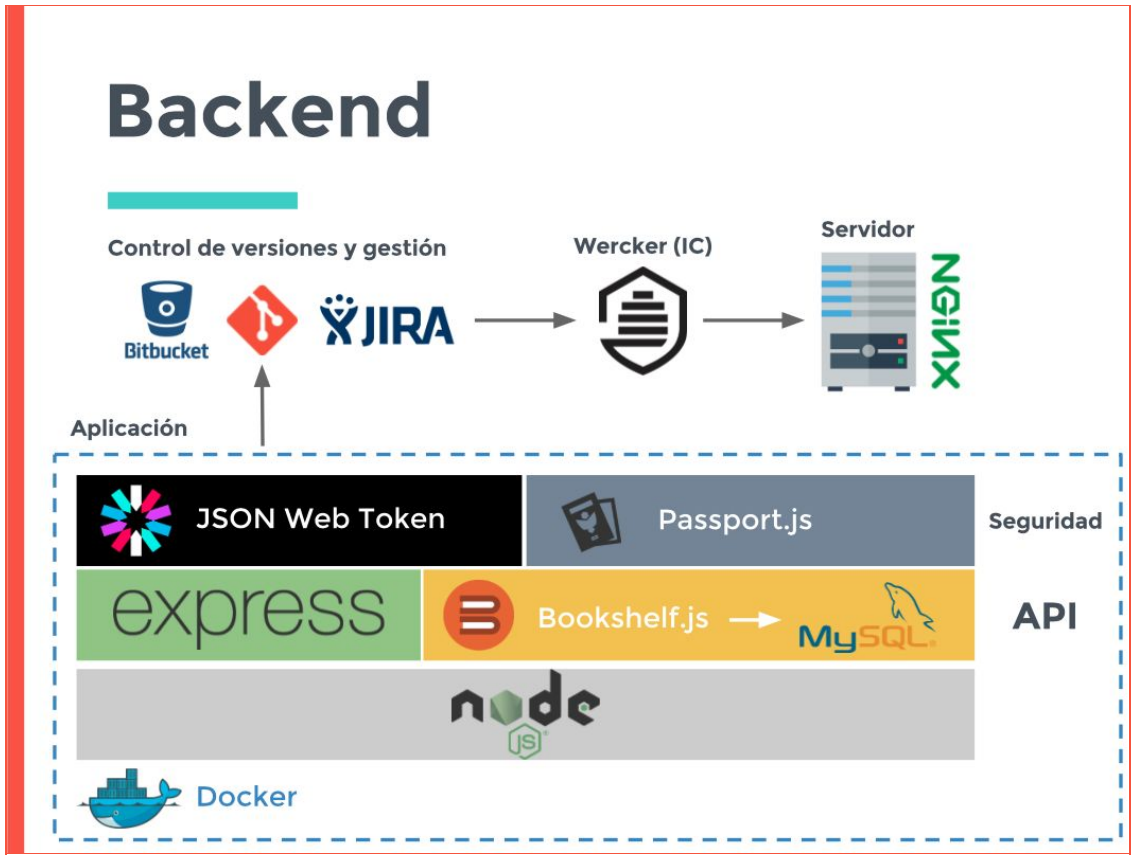
El lenguaje de programación utilizado tanto para la parte del cliente como la del servidor es **JavaScript.** Los principales motivos para escoger este lenguaje han sido las ventajas que proporciona utilizar el mismo lenguaje en el front y en el back y la buena gestión de librerías y paquetes que ofrece JavaScript gracias a **NPM**.

El lenguaje de marcado utilizado para elaborar las vistas es **HTML5** y el lenguaje de diseño gráfico escogido para dotar a las vistas de diseño visual es **Sass**, un metalenguaje de script que es compilado y traducido a CSS.

2.1. Servidor

El servidor de nuestra aplicación se basa en **NodeJS**, un entorno JavaScript basado en eventos. Usamos **Express** como framework para el API, **Bookshelf** como conector con la base de datos **mysql** y **JSON Web Token** y **Passport** para controlar el acceso a la API (seguridad). El servidor se encuentra encapsulado en **Docker**, que lo encapsula generando un sistema de archivos completo que contiene todo lo necesario para ejecutar: código, tiempo de ejecución, herramientas del sistema, bibliotecas del sistema,… todo lo que tenemos instalado en el servidor. Esto garantiza que el software siempre ejecutará de la misma forma, independientemente del entorno de ejecución.

Además, utilizamos **GIT**, **JIRA** y **Bitbucket** para el control de versiones y gestión del proyecto, **Wercker** para la integración continua y **Nginx** como pasarela.



2.1.1. Frameworks/paquetes

bcryptjs v2.3.0

Librería que se utiliza para encriptar las contraseñas de los usuarios para almacenarlas cifradas en la BDD. El encriptado de las contraseñas es asimétrico (o de un sólo sentido), lo que hace que las contraseñas de los usuarios sean prácticamente indescifrables, proporcionándonos gran seguridad al respecto.

bluebird v3.4.1

Librería de promesas que se centra en características innovadoras y rendimiento.

body-parser v1.15.2

Nos permite obtener los datos del body de las peticiones entrantes al servidor (GET, POST, etc.) mediante la propiedad *req.body*.

bookshelf v0.10.2

Es un ORM (mapeo de objeto relacional) para *NodeJS* basado en el constructor de consultas SQL de *Nex*. Nos permite conectar con la base de dato transformando las entidades en objetos para poder trabajar con ellos en JavaScript.

bookshelf-cascade-delete v2.0.0

Plugin de *Bookshelf* que permite borrar en cascada.

composable-middleware v0.3.0

Permite utilizar una serie de funciones de middleware como si fueran una sola función middleware.

ejs v2.5.5

Permite introducir plantillas embebidas de *JavaScript*.

email-templates v2.5.4

Módulo de *NodeJS* que permite renderizar plantillas de correo electrónico junto con la librería *ejs*.

express v4.14.0

Express es un framework de aplicaciones web *NodeJS* sencillo y flexible que proporciona un conjunto robusto de características para crear APIs REST.

express-fileupload v0.0.5

Middleware sencillo de carga de archivos para *Express.*

express-jwt v5.1.0

Middleware que valida Json Web Tokens y establece req.user. Este módulo permite autenticar las solicitudes HTTP utilizando JWT.

fs-extra v1.0.0

Contiene métodos que no se incluyen en el paquete vanilla Node.js fs, como mkdir -p, cp -r, y rm -rf. De esta forma, ya no tenemos que incluirlos.

gm v1.23.0

Paquete que permite redimensionar y renombrar las imágenes subidas al servidor.

helmet v2.1.0

Ayuda a proteger la aplicación estableciendo varios encabezados HTTP.

jsonwebtoken v7.1.9

Implementación de JWT. Json Web Token es un conjunto de medios de seguridad para peticiones HTTP que permite representar de forma segura las peticiones transferidas entre dos partes (cliente y servidor).

knex v0.12.2

Constructor de consultas *SQL* flexible y portátil.

less v2.7.1

Preprocesador *CSS* que agrega características que permiten variables, mixins, funciones, etc. Se utiliza junto a *ejs* y *email-templates* para los correos electrónicos.

less v2.7.1

Preprocesador *CSS* que agrega características que permiten variables, mixins, funciones, etc. Se utiliza junto a *ejs* y *email-templates* para los correos electrónicos.

loadash v4.15.0

Biblioteca de utilidades *JavaScript* que ofrece modularidad, rendimiento y extras.

morgan v1.7.0

Permite formatear los registros sacados por consola de las peticiones HTTP.

mysql v2.11.1

Controlador *NodeJS* para *mysql*. Escrito en *JavaScript*, no requiere compilación.

nodemailer v2.7.0

Librería que permite enviar correo electrónico desde nuestra aplicación basada en *NodeJS*.

nodemailer-mailgun-transport v1.2.2

Complemento de transporte que junto con *nodemailer* permite enviar correo electrónico usando *Mailgun*.

passport v0.3.2, passport-facebook v2.1.1, passport-google-oauth20 v1.0.0, passport-local v1.0.0

Middleware de autenticación para *NodeJS* flexible y modular. Cuenta con un conjunto completo de estrategias de apoyo a la autenticación mediante usuario y contraseña, Facebook, Google, etc.

promise-map-series v0.2.3

Para mapear arrays evitando la ejecución paralela mediante el uso de promesas.

uuid v3.0.1

Para generar automáticamente identificadores de las entidades de la BD.

winston v2.2.0

Biblioteca de logs asíncronos que permite almacenarlos en diferentes dispositivos.

2.1.2. Dependencias de desarrollo (devDependencies)

babel-preset-es2015 v6.9.0

Plugin de *Babel* que compila ES2015 a ES5.

eslint v3.8.1

Inspector de patrones basado para JavaScript que permite identificarlos y encontrarlos.

gulp v3.9.1

Conjunto de herramientas que permiten automatizar tareas que resultan costosas en el proceso de desarrollo de aplicaciones.

nodemon v1.9.2

Script de monitorización para el desarrollo de aplicaciones *NodeJS*. Observa si se han producido cambios para automáticamente reiniciar la aplicación.

require-dir v0.3.0

Ejecuta una secuencia de tareas *Gulp* en el orden especificado.

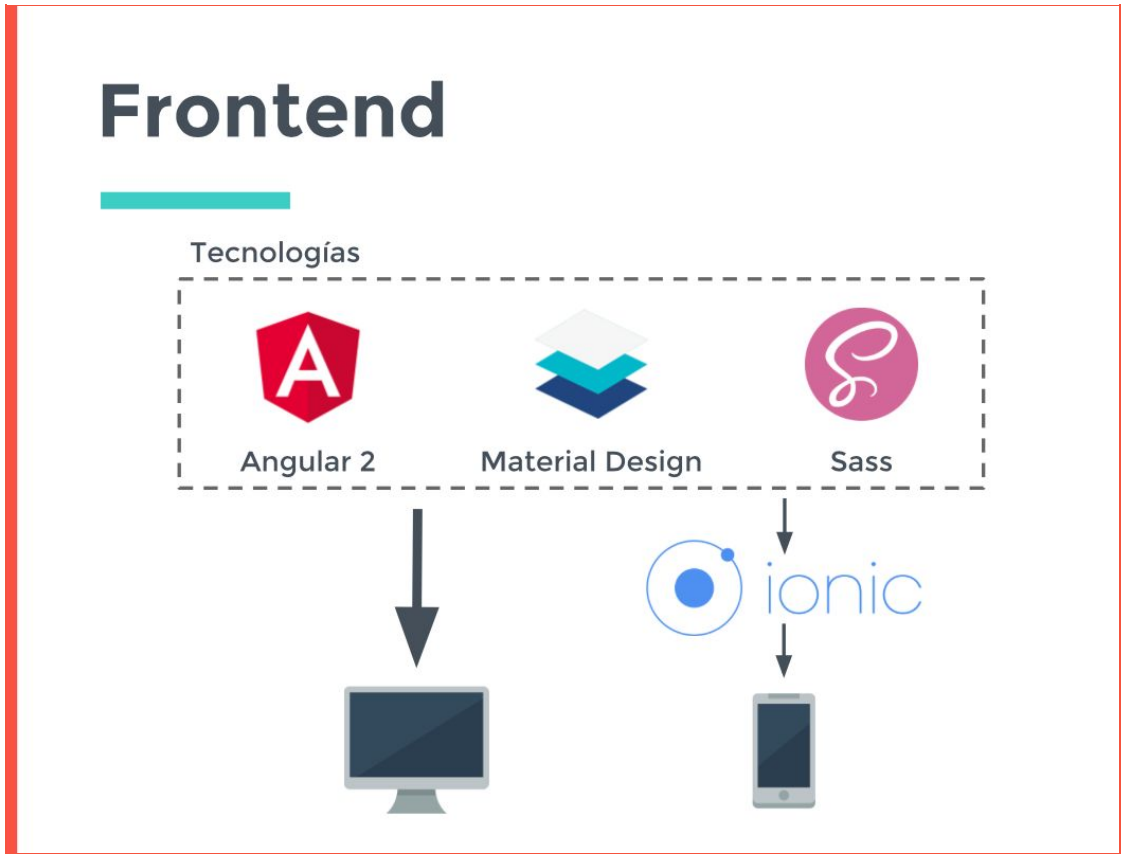
2.2. Cliente

Para el desarrollo del cliente web, hemos partido del cliente oficial de Angular 2, **AngularCLI**, que está construido sobre **Webpack**. AngularCLI facilita la creación de una aplicación de Angular 2 desde cero siguiendo las buenas prácticas recomendadas a la hora de desarrollar utilizando esta tecnología. Además, incorpora **LiveReload** para reiniciar la aplicación automáticamente cuando se produce un cambio en el código fuente.

En Angular 2, el comportamiento de los componentes se implementa en **TypeScript**, un superconjunto de JavaScript que incorpora tipado estático y objetos basados en clases. Cada uno de los componentes definidos en la aplicación tiene asociado una vista y una hoja de estilos. La vista está implementada en **HTML5** y el lenguaje escogido para la hoja de estilos es **Sass** (AngularCLI permite incorporar Sass como lenguaje de estilo de forma sencilla). Como framework de estilo hemos decidido utilizar **Angular Material 2**, a pesar de estar aún en versión beta.

Cuando iniciamos la aplicación, AngularCLI se encarga de compilar todos los ficheros TypeScript en JavaScript, los .scss en .css y hacer *'build'*, es decir, construir la aplicación para que el navegador sea capaz de interpretarla.

Por otra parte, para el desarrollo del cliente móvil utilizaremos **Ionic2**, concretamente **IonicCLI**, que también está basado en Angular2. Esto nos permitirá reutilizar gran cantidad de código de los componentes del font-end web.



2.2.1. Frameworks/paquetes

@angular/common v2.2.1

Incorpora directivas y servicios comunes de Angular2

@angular/compiler v2.2.1

Librería para compilar Angular2

@angular/core v2.2.1

El framework del 'core' de Angular2

@angular/flex-layout v2.0.0-beta.1

Librería que permite utilizar las directivas flex y layout de manera similar a como se utilizan en Angular.

@angular/forms v2.2.1

Directivas y servicios para la creación de formularios.

@angular/http v2.2.1

Incorpora el servicio para hacer peticiones HTTP.

@angular/material v2.0.0-alpha.10

Framework de estilo Angular Material 2.

@angular/platform-browser v2.2.1

Librería que permite utilizar Angular2 en un navegador web.

@angular/platform-browser-dynamic v2.2.1

Librería que permite utilizar Angular 2 en un navegador web con compilación JIT (compilación en tiempo de ejecución).

@angular/router v3.2.1

Librería que incorpora todas las funcionalidades de enrutamiento.

lodash v4.17.4

Librería Javascript que permite manipular estructuras JSON de forma sencilla. Simplifica de forma considerable las operaciones más frecuentes que se realizan sobre JSON, como ordenaciones, filtros, búsquedas, etc.

@types/lodash v4.14.48

Definiciones TypeScript para *Lo-Dash*.

core-js v2.4.1

Librería de JavaScript que proporciona herramientas para crear código JS orientado a objetos y eventos.

moment v2.17.1

Librería que permite parsear, validar, manipular y mostrar fechas en JavaScript.

rxjs v5.0.0-beta.12

Librería que nos permite utilizar todas las extensiones y características de ES6.

ts-helpers v1.1.1

Librería que da soporte a la compilación en TypeScript.

zone.js v0.6.23

Framework que implementa 'zonas' en JavaScript. Una zona es un mecanismo que se encarga de interceptar y hacer un seguimiento del trabajo asincrónico.

2.2.2. Dependencias de desarrollo (devDependencies)

@angular/compiler-cli v2.2.1

Compilador de AngularCLI para Node.js.

@types/jasmine v2.5.38

Paquete que se encarga de definir tipados para Jasmine. Jasmine es un marco de desarrollo orientado a la conducta para probar código JavaScript.

@types/node v6.0.42

Definiciones TypeScript para Node.js.

@types/hammerjs v2.0.32

Paquete que se encarga de definir tipados para Hammer.js. Hammer.js ayuda a añadir soporte para acciones táctiles y elimina el retraso de 300 ms en los clicks. Admite los gestos multi-touch más comunes, y es completamente extensible para agregar gestos personalizados.

angular-cli v1.0.0-beta.21

Paquete que contiene AngularCLI, el cliente oficial de Angular2.

jasmine-core v2.5.2

Paquete oficial de los principales archivos de Jasmine para su uso en proyectos basados en Node.js.

jasmine-spec-reporter v2.5.0

Muestra por consola los resultados de las pruebas realizadas con Jasmine.

karma v1.2.0

Permite lanzar tests para JavaScript.

karma-chrome-launcher v2.0.0

Plugin de *Karma* para lanzar tests en *Chrome*.

karma-cli v1.0.1

Interfaz de línea de comando de *Karma*.

karma-jasmine v1.0.2

Plugin de *Karma* que permite adaptarse al marco de pruebas *Jasmine*.

karma-remap-istambul v0.2.1

Librería para generar informes de las pruebas de Karma.

protractor v4.0.9

Marco de prueba de extremo a extremo para aplicaciones AngularJS. Es decir, un programa Node.js construido sobre WebDriverJS que ejecuta pruebas contra la aplicación ejecutada en un navegador real, interactuando con él como lo haría un usuario.

ts-node v1.2.1

Entorno de ejecución TypeScript y REPL (bucle Lectura-Evaluación-Impresión) para Node.js. Permite ejecutar archivos TypeScript con Node.js mediante AngularCLI.

tslint v3.13.0

Lint de análisis estático extensible para el lenguaje TypeScript. Lint es una herramienta de programación utilizada para detectar código sospechoso, confuso o incompatible.

codelyzer v1.0.0-beta.3

Conjunto de reglas lint para el análisis de código estático de proyectos Angular2 con TypeScript.

typescript v2.0.3

Paquete que permite trabajar con Angular2 usando TypeScript. TypeScript es un lenguaje para aplicaciones JavaScript escalables que añade tipos, clases y módulos opcionales a JavaScript. Soporta herramientas para aplicaciones JavaScript de gran

escala para cualquier navegador, host o sistema operativo. TypeScript compila JavaScript de forma legible y basado en estándares.

webdriver-manager v10.2.5

Servidor Selenium y administrador de controladores de navegador para pruebas de extremo a extremo.

3. Arquitectura de la aplicación

La arquitectura de nuestra aplicación es de tipo **client-side**, lo que implica que el servidor únicamente envía datos hacia el cliente (el cliente realiza peticiones HTTP a la API Rest) y es el cliente el que se encarga de procesar estos datos y utilizarlos para componer la vista.

Patrón arquitectónico

Angular 2 utiliza un patrón arquitectónico basado en componentes. Las directivas y los servicios sirven de apoyo a estos componentes, y también están definidas con una arquitectura similar. Los componentes base de la aplicación contienen dependencias, una vista y una declaración de clase que puede ser considerada como el controlador. Por tanto, podemos considerar que cada uno de los componentes de la aplicación es un conjunto individual de arquitectura MVC.