UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

barterAPP

Manage your business time.

Presentado por Adrián Aguado
en Universidad de Burgos — 22 de junio de 2017
Tutor: Luis R.Izquierdo

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. Luis R. Izquierdo, profesor del departamento de Ingeniería Civil área de organización de empresas

Expone:

Que el alumno D. Adrián Aguado, con DNI 78759314T, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado barterAPP

y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 22 de junio de 2017

Vº. Bº. del Tutor:

Luis R. Izquierdo

Resumen

Existen numerosas situaciones en la vida diaria en las que se requiere un reparto de turnos. En muchas ocasiones, el reparto de turnos planificado supone un problema que podría mejorarse fácilmente si se llevaran a cabo cambios bilaterales que a veces no se llegan a producir simplemente porque las partes implicadas desconocen que existe esa oportunidad de mejora.

La aplicación desarrollada presenta una solución frente a los problemas que existen hoy en día a la hora de realizar un cambio de turno en determinadas empresas o instituciones públicas.

BarterApp se presenta como un aplicación web y una interfaz híbrida móvil. Esta desarrollada con las últimas tecnologías web presentes en el mercado.

Descriptores

WebApp, Angular, Nodejs, Express, MongoDB, MEAN

Abstract

There are a lot of occasions in life which sometimes a shift deal is required. At many times, this planned shift allocation is a problem that could easily be improved if bilateral changes are made, sometimes do not come about simply because the parts involved in the shift are unaware that there is an opportunity for improvement.

The application developed presents a solution to the problems that exist today when making a change of shift in public institutions or some companies.

BarterApp is developed as a web application and hybrid mobile interface. It is developed with the latest web technologies present nowadays.

Keywords

WebApp, Angular, Nodejs, Express, MongoDB, MEAN,

Índice general

Índice de figuras

Índice de tablas

Introducción

Hoy en día existen multitud de situaciones en las que se requiere un reparto de turnos o guardias entre un grupo de personas (por ejemplo personal sanitario en servicios hospitalarios, servicios de emergencias extrahospitalarias, bomberos, protección civil y muchos ejemplos más). En muchas ocasiones, el reparto de turnos planificado podría mejorarse fácilmente si se llevaran a cabo cambios bilaterales que a veces no se llegan a producir porque las partes implicadas simplemente desconocen que existe esa oportunidad de mejora, o porque intuyen que los costes de encontrar esa mejora y llegar a hacerla posible son excesivamente elevados. Otras veces los cambios sí que pueden llegar a producirse pero tras un largo período de negociación.

Actualmente, tras un proceso de investigación inicial, he determinado que no existe una herramienta sencilla que presente una solución a este problema. Es más en servicios como los sanitarios el problema no está ni si quiera digitalizado, se suele realizar simple y llanamente en papel. Por una parte esta no es una mala técnica pero la implantación de una herramienta digital accesible por cualquiera puede facilitar las cosas enormemente a nivel de tiempo, espacio y entendimiento.

El mundo de las aplicaciones, tanto web como móvil, resulta extremadamente competitivo hoy en día y posicionarse en el mercado es realmente complejo. Por una parte está la tarea de adaptarse a las nuevas tecnologías que cambian constantemente, tanto web como móvil, pero también la aceptación que el posible cliente o usuario haga de tu futuro producto o servicio.

1.1. Motivacion

Explicar por qué he elegido este trabajo y no otro es también describir cuáles son mis objetivos personales de cara a un futuro cada vez más cercano. Me gusta el desarrollo web y me apasiona la posibilidad de mejorar el mundo a través de la tecnología, si bien no tengo experiencia dentro del desarrollo

web sí que me gustaría encaminarme hacia ello en un futuro no muy lejano donde resulta tener un lugar muy prometedor [16].

1.2. Estructura de la memoria

- **Introducción:** Aquí se explica la motivación que me ha llevado a escoger este trabajo y no otro, la descripción del problema propuesto y una breve introducción general a la solución que se ha pretendido dar. Así mismo también una estructura de toda la memoria
- **Objetivos del proyecto:** tanto a nivel académico, personal como los objetivos alcanzados con el proyecto. También se comentará lo que le espera en un futuro al proyecto, ya que me gustaría seguir con él.
- **Conceptos teóricos:** breve explicación de los conceptos teóricos que he debido adquirir previamente para la realización del proyecto, necesarios para comprender mejor cómo funciona el proyecto.
- **Técnicas y herramientas:** software principal, metodologías empleadas durante el proyecto, *frameworks* empleados y herramientas.
- **Aspectos relevantes del desarrollo:** detalles sobre el la realización del proyecto.
- **Trabajos relacionados:** otros aspectos del desarrollo del proyecto.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas tras la realización de este trabajo fin de grado y hacia dónde va en el futuro.

Junto a la memoria se proporcionan los siguientes anexos:

- **Plan del proyecto software:** estudio de viabilidad y planificación del proyecto.
- **Especificación de requisitos del software:** se describe la fase de análisis; los objetivos generales, el catálogo de requisitos del sistema y la especificación de requisitos necesarios para su correcto funcionamiento.
- **Especificación de diseño:** se describe la fase de diseño de la aplicación tanto *front-end* como *back-end* y el paso de la interfaz web a la móvil.
- **Manual del programador:** recoge los aspectos más relevantes relacionados con el código fuente y su correcto manejo por parte de un programador.
- **Manual de usuario:** guía de usuario para el uso de la aplicación.

Objetivos del proyecto

En este apartado se van a detallar los diferentes objetivos que se buscaban, a diferentes niveles, con la realización del proyecto.

2.1. Objetivos generales

Comenzaremos con los objetivos generales del proyecto.

- Desarrollar una *aplicación web* que permita la gestión de turnos de una manera sencilla.
- Desarrollar una aplicación móvil híbrida asociada a la anterior que permita acceder desde cualquier plataforma a esa aplicación para dar un servicio multiplataforma.
- Permitir tener una herramienta que resuelva un problema real.

2.2. Objetivos técnicos

A continuación explicaré los principales objetivos técnicos que se pretendían.

- Aprendizaje de la implementación de una aplicación web con *frameworks* web/móvil, como puede ser el caso de Angular o Ionic.
- Breve toma de contacto con el mundo móvil (*PhoneGapp, Cordova*).
- Aplicar la metología Scrum.

2.3. Objetivos personales

Los objetivos personales que he perseguido durante todo el desarrollo han sido los siguientes:

- El diseño e implementación de una aplicación web, realizada con *Node.js*y *Express* para la parte del servidor, y *Angular* y *Bootstrap* para la parte del cliente. En definitiva aprender las diferencias fundamentales entre *back* y *front*.
- Conocer y manejar bases de datos NoSQL, como *MongoDB*.
- Realizar un algoritmo desde cero para aplicarlo a un problema real.
- Averiguar y aprender el mayor número de cosas sobre las tecnologías web del mercado actual.
- Aprender a realizar un proyecto de gran envergadura que sea aplicable a un problema real, y sobre todo, a gestionarlo.

2.4. Objetivos alcanzados

Los objetivos que finalmente se han alcanzado han sido:

- Algoritmo funcionando con éxito.
- Aplicación web.
- o3.

Conceptos teóricos

La parte del proyecto más ardua ha sido, sin lugar a dudas, la de aprender a utilizar *frameworks* web, para lo cual me he nutrido de diversos cursos online, tanto gratuitos como de pago. A continuación, en este apartado voy a relatar todas las tecnologías empleadas y a justificar por que he empleado unas y no otras. Las técnicas y tecnologías como tal se verán en el siguiente apartado pero todas ellas presentan unos conceptos teóricos comunes que es necesario conocer.

3.1. Introducción

El desarrollo de aplicaciones informáticas evoluciona continuamente para adaptarse a las tecnologías de la información y las comunicaciones (TIC). El auge de Internet y de la web ha influido notablemente en el desarrollo de software durante los últimos años. Hoy en día la interfaz de los sistemas de información se implementa utilizando tecnologías web que ofrecen numerosas ventajas tales como el uso de una interfaz uniforme y la mejora del mantenimiento del sistema. Sin embargo, la existencia de numerosos estándares y los intereses de los fabricantes de tecnologías web dificultan el desarrollo de este tipo de aplicaciones.

En los principios de la informática las relaciones entre los usuarios y los programas de los que hacían uso era muy diferente a lo que tenemos ahora. Ahora se potencia el diseño basado en usuario, antes se potenciaba más que el programa estuviera implementado de manera idónea antes que la experiencia que el usuario tenía al hacer uso del mismo. Obviamente ahora también se impulsa esa manera de desarrollar código, lo que se conoce como *clean code* [5], pero en un mundo en el que la competitividad es tan alta prima sobre todo la experiencia del usuario final, que al fin y al cabo va a ser el que interactúa con tu producto ya sea web o móvil.

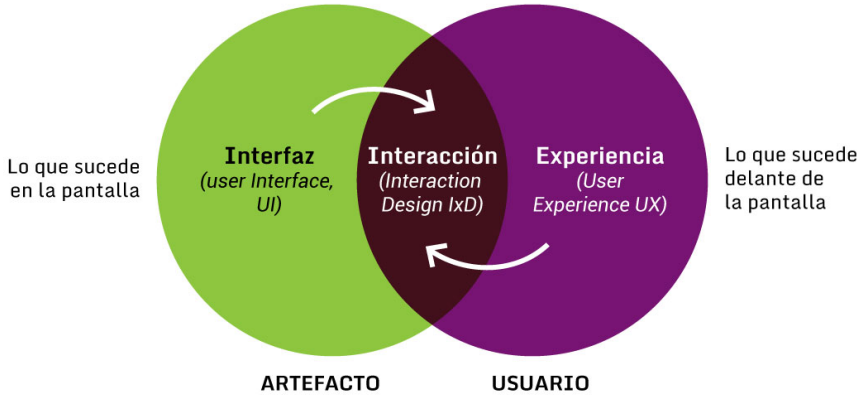

Figura 3.1: Esquema *UI, UX IxD*. Fuente dealfuel.com.

El mundo de las aplicaciones web es un mundo en constante evolución y actualización, donde aparecen nuevas tecnologías que ofrecen tanto mejoras visuales que mejoran la experiencia del usuario como de rendimiento. Este tipo de tecnologías abren un abanico inmenso de posibilidades, y hacen pensar a los analistas y programadores de aplicaciones web que puede haber otras soluciones que mejoren su aplicación, pero que no estaban disponibles cuando definieron la arquitectura y el diseño de sus aplicaciones. Es un mundo que avanza tan rápido que en muchas ocasiones es imposible estar al día de todos los *frameworks* o tecnologías que van surgiendo, por eso es mejor focalizar los esfuerzos en alguna en concreto e intentar aprenderla de la mejor manera posible.

3.2. Concepto general del proyecto

Hace algún tiempo para realizar una web existía una gran barrera a la hora de entender el concepto de cliente y servidor. Por un lado estaba la parte del cliente, la cual se realizaba en lenguajes puros, como *HTML* y *CSS* para las hojas de estilo. Por otro lado estaba el servidor lo cuál significa cambiar totalmente de lenguaje, lo que suponía un salto para un programador web que debía conocer ambos lados para crear webs seguras y robustas, la parte del cliente es la que interactúa con el usuario, la que nosotros vemos, y la parte del servidor es la parte que conecta con la base de datos, en caso de que sea necesario.

Todo cambió cuando se popularizó Javascript para la realización de webapps, la posibilidad de crear web con un mismo lenguaje en todas las partes resulta muy atractiva tanto para el programador como para la lógica del propio programa o web ya que se disminuye el número de errores o se consiguen

localizar de manera más sencilla al no tener que estar cambiando de lenguaje. En este punto es donde surge el stack MEAN:

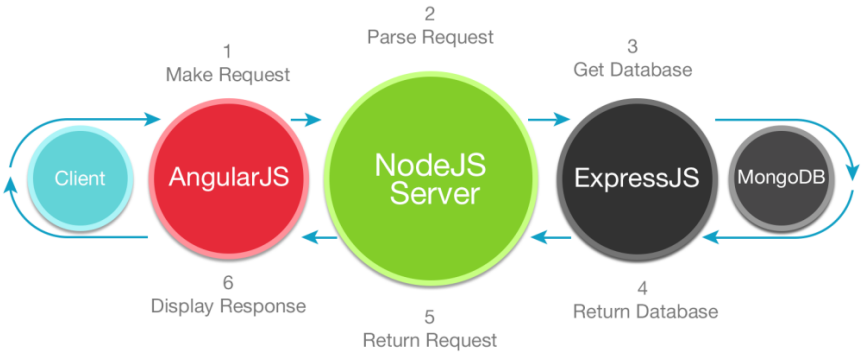

Figura 3.2: Esquema *MEAN*. Fuente: kambrica.com.

La presencia de *Javascript* para el desarrollo de software se está haciendo un hueco cada vez mayor en el mercado. Aquel humilde lenguaje que empezó en los años '90 como una vía sencilla de validar formularios, se ha convertido en parte fundamental del desarrollo de todo tipo de aplicaciones: web, móviles, bases de datos e, incluso, administración de sistemas. Esta proliferación ha llevado a *Javascript* a todas las capas de desarrollo, empezando por el lado cliente en sus inicios (el navegador), pero yendo también al servidor y a la capa de almacenamiento. En cualquiera de esos puntos podemos encontrar *Javascript* listo para ser utilizado.

3.3. Análisis del sistema: MVC

El *modelo-vista-controlador* es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción con el usuario.

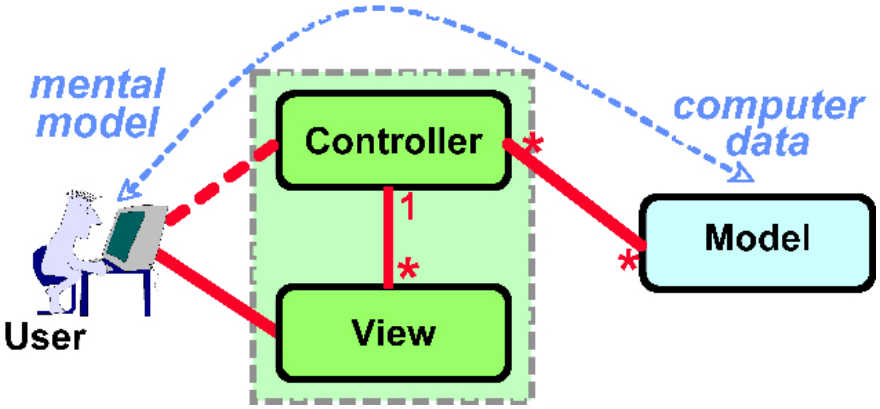

Figura 3.3: Dibujo *MVC*[15]. Fuente: artima.com.

3.4. Arquitectura

Como se ha comentado en secciones anteriores la aplicación tiene dos sistemas bien diferenciados: servidor y cliente.

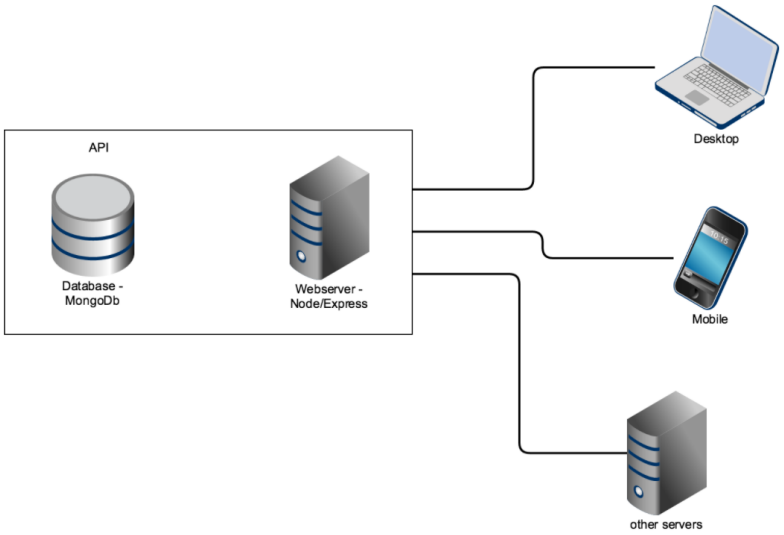

Figura 3.4: Arquitectura simplificada *MVC*[15]. Fuente: Elaboración propia.

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las tecnologías y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas y se pretende aquí realizar un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas.

No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas detalladamente, sino comentar los aspectos más destacados de cada opción.

He de decir que durante el desarrollo del proyecto tanto las herramientas como las técnicas a utilizar han ido mutando debido a que yo mismo iba descubriendo nuevas técnicas o nuevas herramientas que mejoraban mi manera de hacer el código o me facilitaban mi manera de trabajar. No ha sido así con la tecnología ya que esto hubiera supuesto más problemas que ventajas. El estar cambiando de herramienta puede que haya sido uno de los mayores errores del proyecto dado que siempre es mejor focalizarse en una sola antes de intentar abarcar demasiado.

4.1. Metodologías

En este apartado se describen las metodologías utilizadas para el desarrollo del sistema.

Scrum

Scrum es un sistema de desarrollo de software que está dentro de las metologías ágiles. Principalmente, se basa en la creación y asignación de tareas. Esta tarea entrará en un sistema iterativo que controlará los cambios de esta-

do de la tarea, hasta que esta resulta completada y se descarta del panel de scrum.

En definitiva Scrum propone seguir un proceso de desarrollo iterativo e incremental a través de una serie de iteraciones denominadas sprints y de revisiones [18].

MVC: Model-View-Controller

Ya nombrado anteriormente, Ees un patrón de diseño software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos. Es decir surge de la idea de separar el modelo por un lado, la vista por otro y por último el controlador.

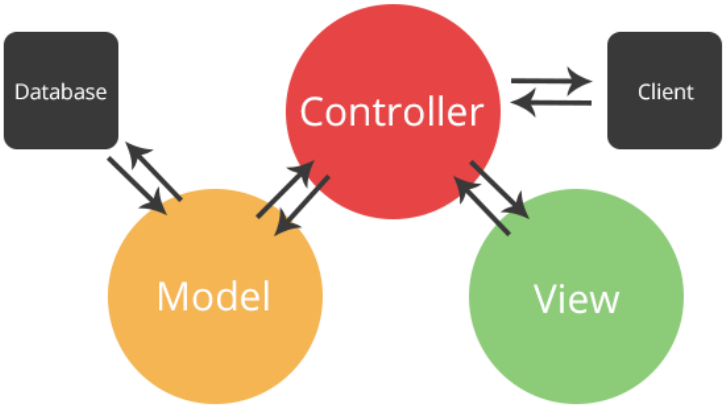

Figura 4.5: Esquema *MVC*. Elaboración propia.

Patrones de diseño: singleton

El patrón de diseño *singleton* [14] está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Es decir se crean objetos, declarados como tipo privados, una sola vez para se usados en muchos sitios diferentes.

4.2. Herramientas

En este apartado se describen las herramientas utilizadas para el desarrollo del sistema.

Atom

Atom es un editor de texto moderno de código abierto desarrollado por GitHub que se ha escogido en este proyecto para desarrollar la aplicación debido a que esta desarrollado utilizando tecnologías web, luego ha sido creado por y para la web. Es posible ampliar sus funcionalidades a través de plugins desarrollados con *Node.js* que puede ser instalados de una forma sencilla a través del gestor de paquetes internos con el que cuenta, así como diferentes tipos de temas. Esta posibilidad hace que se convierta en un editor de texto muy personal, ya que es el mismo desarrollador el que elige las características que desea tener sin afectar mínimamente al rendimiento.

Figura 4.6: Logo *Atom*. Fuente: `atom.io`.

- **Alternativas estudiadas**

  - Brackets
  - Eclipse

  Tal y como he nombrado en la introducción la necesidad o la curiosidad por mejorar mi trabajo hicieron que encontrara nuevas herramientas durante el desarrollo del proyecto, es el caso, por ejemplo, del entorno de desarrollo. Al comienzo empecé con Brackets como IDE, ya que era el entorno que utilizaba el profesor de los cursos que me recomendó mi tutor. Después y atraído por el uso a diario de Eclipse como herramienta principal de trabajo durante mi período de prácticas con la universidad decidí que podía ser una buena idea el emplearlo también para realizar la API, craso error dado que Eclipse no fue creado para el mundo del desarrollo web por lo que me vi con que cada vez me encontraba con menos soporte para lenguajes, sin ir más lejos eclipse no presenta, actualmente, soporte para *typescript* el lenguaje principal de desarrollo de *angular 2*. Por fin di con el IDE perfecto para desarrollar entornos web que es el descrito en el párrafo anterior.

Detalles: WebStorm

Otro *IDE* que recomiendo y que ofrece una grandes prestaciones es Webstorm. Según algunos artículos [11] es una de las mejores herramientas para

Javascript del mercado, esta enfocado totalmente a este lenguaje web y sigue la misma filosofía de paquetes que Atom.

Robomongo

Robomongo es una interfaz para *MongoDB* que nos permite conectarnos al servidor de base de datos de forma sencilla ya que nada más arrancarlo podemos crear una nueva conexión. Resulta también muy sencillo e intuitivo de utilizar.

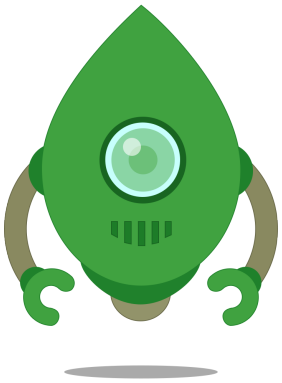

Figura 4.7: Logo *Robomongo*. Fuente: `robomongo.org`.

Firefox for developers

Para el navegador he elegido Firefox en su versión para desarrolladores: Firefox Developer Edition , el cuál considero que posee todas las herramientas esenciales para testear y probar una aplicación web.

Figura 4.8: Logo *FF Developers*. Fuente: `firefox.com/developer`.

4.3. Tecnologías

En este apartado se describen las tecnologías utilizadas para el desarrollo del sistema. Las siguientes líneas servirán al lector para conocer algunas de las tecnologías más modernas y poco conocidas que se han usado en el proyecto. Se omite el nombrar directamente lenguajes web como pueden ser *Javascript*, *HTML* o *CSS*. Existe mucha información accesible en la web sobre ellas y se hará más incapié en las tecnologías principales que usan propiamente estos lenguajes que no son tan comunes.

Front-end: Angular 2+

Angular es un framework cliente MVC *Javascript* de código abierto creado en sus inicios por Google permite crear *Single-Page Applications* (SPA) cuya principal característica es la de dar al usuario la impresión de que todo sucede en la misma página, sin hacer recargas de la misma. Es decir, trata de emular a las aplicaciones de escritorio, lo que se trata es de intercambiar las vista y no de recargar la página. En la actualidad cuenta con una amplia comunidad de desarrolladores que dan soporte al framework. Algunas características de Angular son:

- El sistema de databinding es muy completo y potente.
- Angular es una solución completa que incluye prácticamente todos los aspectos que puedes necesitar para crear una aplicación cliente en *Javascript*.
- La comunidad de desarrolladores crece cada día y la popularidad de Angular es un hecho.
- Está realizado para ser fácilmente testeable.

Figura 4.9: Logo *Angular*. Fuente: `angular.io`.

Detalles: Angular Arquitectura

Un app de Angular esta basada en componentes. En su día resultó una revolución en el desarrollo web ya que lo que se persigue es que cada parte de la aplicación posea un componente de manera que se consigue hacer aplicaciones web reutilizables de algún modo.

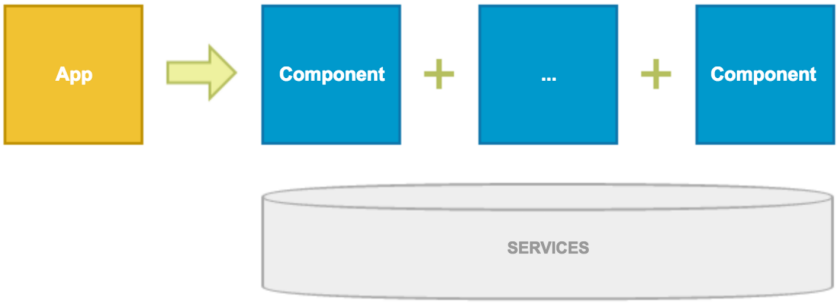

Figura 4.10: Esquema componentes. Fuente: `rldona.gitbooks.io`.

Cada componente se compone de los siguientes elementos que ayudan a visualizar, controlar, acceder o servir a cada componente.

- Vistas: es la parte visible por el usuario. Suelen estar parametrizadas, y todas las vistan tienen asociado un controlador.
- Controladores: Sirven los datos y funcionalidades a las vistas asociadas a él. Suelen ser estos los que utilizan los servicios para obtener los datos.
- Directivas: Ofrecen elementos nuevos o nuevos comportamientos en elementos ya existentes dentro de las propias vistas.
- Servicios: son los encargados de proveer los datos. Lo más normal es que estos datos provengan de una API externa.

Detalles: Typescript

Angular esta basado en Typescript es un lenguaje de código abierto que resulta ser un superset de Javascript. Es fuertemente tipado y orientado a objetos basado en clases.

Detalles: Angular CLI

Es un intérprete de línea de comandos de Angular que ayuda en el inicio y desarrollo de proyectos, ocupándose de la creación del esqueleto de la mayoría de los componentes de una aplicación Angular. Es interesante utilizarlo dado que ahorra mucho trabajo al desarrollador además de evitar despistes

innecesarios. Por poner un ejemplo, cada vez que un componente es creado en Angular es necesario modificar el componente principal manualmente para poder usarlo con Angular CLI tan solo es necesario introducir un comando para crear el nuevo componente y Angular añade todas las dependencias por ti. Se podría pensar que se trata de una heramienta de terceros pero no es así, la interfaz de comandos es proporcionada directamente por el equipo de Angular. Se darán consejos de como usarlo en los anexos.

Por otro lado el uso de Angular CLI permite realizar en un solo comando el preparado de la aplicación para lanzarla al servidor, lo cual resulta también muy interesante de cara a una parte más seria de la aplicación como puede ser por ejemplo lanzarla al mercado.

El aspecto de una aplicación '**hola mundo**' creada por Angular CLI contendría los siguientes elementos:

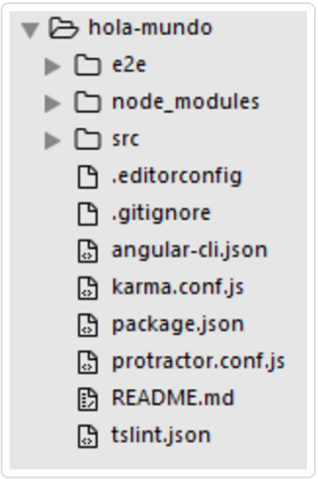

Figura 4.11: Vista *App hola Mundo*. Elaboración propia.

Más información en el Anexo: Manual del programador.

Detalles: Calendario Angular

El calendario elegido ha sido encontrado en github y realizado como un componente de Angular2+ es heredado de un calendario para AngularJS que ya tuvo mucho éxito y posteriormente realizado para versiones superiores. Es un calendario flexible y pensado para muchos usos por lo que no es completamente *responsive* tal y como se puede ver en la documentación [4] que está disponible y nos explica de manera detallada cómo incluirlo en nuestro proyecto Angular.

- **Alternativas estudiadas: otros calendarios**
  - Full Calendar (Javascript)
  - El componente de FullCalendar para Angular2
  - Day pilot
  - Calendario UI interesante pero no disponible en versiones de angular 2+.

- **Alternativas estudiadas**

  Algunas de las alternativas estudiadas a Angular 2+ han sido las siguientes:

  BackBone JS Es un framework web que permite construir aplicaciones usando Javascript siguiendo el patrón MVC (modelo-vista-controlador), es decir, de características muy similares a Angular

  Ember JS Se trata de un framework web también que representa un cambio de paradigma con respecto a otros frameworks ya que, por ejemplo, para funcionar correctamente nos exige que respetemos totalmente una convenciones de nomenclatura a la hora de nombrar a cada uno de los objetos de la App.

  React Propiedad de Facebook, ReactJS es una librería Javascript de código abierto que ofrece grandes beneficios en performance, modularidad y promueve un flujo muy claro de datos y eventos, facilitando la planeación y desarrollo de apps complejas.

Front-end: Boostrap

Boostrap es un framework CSS desarrollado inicialmente por Twitter que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web.

Figura 4.12: Logo *Boostrap*. Fuente: `getbootstrap.com`.

Ionic 2

Ionic2 es un *framework* gratuito y open source para desarrollar aplicaciones híbridas multiplataforma que utiliza HTML5, CSS (generado por SASS) y Cordova como base. Asegura aplicaciones rápidas, con un alto rendimiento y escalables.

Figura 4.13: Logo *Ionic*. Fuente: `ionicframework.com`.

Back-end: Node js

Nodejs es un entorno de programación pensado para realizar funciones de servidor. Permite la construcción de servidores de forma muy sencilla y rápida, además se puede aplicar para otros usos. La ejecución es asíncrona. Esto significa que las funciones no son ejecutadas secuencialmente (si existen dos llamadas consecutivas, no es necesario que acabe la primera llamada para ejecutar la segunda). Está pensado para ser un gestor de entradas y salidas. No ha sido diseñado para ejecutar gran cantidad de código, sino para realizar comunicaciones muy rápidas y abundantes, tanto con eventos locales como por comunicación por red.

Figura 4.14: Logo *Node JS*. Fuente: `nodejs.com`.

- **Alternativa estudiada: PHP**

  Un lenguaje ampliamente conocido, con una extensa documentación y multitud de ejemplos. No se integra tan bien como *Node s* con *MongoDB*.

Back-end: Express

Express js es un framework web flexible para *Nodejs* que proporciona un conjunto robusto de características para aplicaciones web y móviles, proporciona una capa delgada de características fundamentales de aplicaciones web.

Facilita la creación de API's gracias la gran variedad de métodos HTTP y middleware que proporciona.

Figura 4.15: Logo *Express*. Fuente: `expressjs.com`.

Base de datos: MongoDB

Para la parte de la base de datos he escogido MongoDB estamos hablando de un sistema de gestión de bases de datos no relacionales, o noSQL, es decir, un base de datos que no tiene tablas, se basa en colecciones de datos.

En las colecciones se almacenan contenidos que pueden tener diferentes campos. Este sistema almacena los datos en documentos de tipo JSON, cosa que puede favorecer la integración con las aplicaciones que trabajen con este tipo de formatos, como es el caso de una API REST.

Figura 4.16: Logo *MongoDB*. Fuente: `mongodb.com`.

Detalles: Mongoose

Se ha utilizado un ODM (*Object-document mapping*) para Mongo y Node.js. Nos permite interactuar con la base de datos mediante objetos de JavaScript, y facilita las operaciones CRUD (Create, Read, Update, Delete) y las validaciones de los usuarios.

Detalles: Mongoskin

Se ha utilizado un ODM (*Object-document mapping*) para Mongo y Node.js. Nos permite interactuar con la base de datos mediante objetos de JavaScript, y facilita las operaciones CRUD (Create, Read, Update, Delete) y las validaciones de los usuarios.

- **Alternativa estudiada: Cassandra**

  Apache Cassandra es una de las mejores opciones cuando la escalabilidad, la alta disponibilidad y la integridad de los datos son las prioridades de un proyecto.

4.4. Testing

Postman

Las pruebas básicas se realizaron con Postman. Esta herramienta nos permite construir y gestionar de una forma cómoda nuestras peticiones a servicios REST (Post, Get, etc). Su manejo es realmente intuitivo ya que simplemente tenemos que definir la petición que queremos realizar, es decir introducir la ruta, y pulsar el botón de enviar. Es realmente fácil de usar y está disponible como extensión del navegador Chrome, pero también como aplicación de escritorio.

Figura 4.17: Logo *Postman*. Fuente: `getpostman.com`.

- **Alternativa estudiada: SoapUI**

  Mucho más completo que Postman, SoapUI tiene una versión gratuita de código abierto y un versión de pago con algunas funcionalidades que hacen que sea mucho más productiva. El motivo de elección del anterior es porque en los cursos realizados aprendí a manejar éste.

Karma

Jasmine

4.5. Documentación

En este apartado se describen algunas de las herramientas utilizadas para la parte de la documentación del proyecto.

LaTeX

LaTex es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades, es usado de forma especialmente intensa en la generación de artículos y libros científicos.

Detalles: texmaker

Como editor para realizar las modificaciones pertinentes he usado textmaker, es un editor de LaTeXpara el sistema operativo MAC.

Detalles: ShareLatex

ShareLatex es un servicio online que nos permite crear y compartir documentos en LaTeX. La característica por la que he empleado este editor es por que permite la edición conjunta de un proyecto desde el navegador y ofrece diferentes tipos de plantillas según el uso que le vayamos a dar. Por lo tanto para realizar tablas, pruebas de imágenes u otros ensayos antes de copiarlo al documento final las he realizado todas aquí ya que resulta demasiado costoso compilar todo el documento.

- **Alternativa estudiada: Open Office**

  El editor de textos propiedad de Apache fue la primera alternativa elegida, aunque más tarde decidí realizar la memoria en LaTeX. para aprender otra manera de realizar documentos de textos que quizá fuera interesante para el futuro.

Mendeley

Mendeley es un gestor de referencias bibliográficas. Es una aplicación web y de escritorio, propietaria y gratuita. Permite gestionar y compartir referencias, links, libros o documentos de investigación. Recibí un curso introductorio por parte de la encargada de gestión de la biblioteca centra de la Universidad de Burgos.

4.6. Otras herramientas

En este apartado se describen otras herramientas que se han empleado como son el alojamiento web, el alojamiento de la base de datos, los repositorios u otras tecnologías.

Heroku

Heroku es un servicio de almacenamiento en la nube que además tiene mecanismos y herramientas para que la puesta en producción de las aplicaciones web sea prácticamente automática.

- **Alternativas estudiadas: Plataformas en la nube**

  Plataformas en la nube como Amazon AWS u OpenShift son muy populares hoy en día y permiten prácticamente lo mismo que la aplicación elegida. Se ha elegido Heroku finalmente por la facilidad de la puesta en marcha de la aplicación en producción.

Mlab

Mlab Es una plataforma de base de datos como servicio (*DBaaS*), para alojar y gestionar bases de datos MongoDB. Es gratuita, hasta cierto punto, y muy intuitiva en el uso.

NPM

Es el gestor de paquetes por defecto para *Nodejs*.

Github

Para el control de versiones se ha utilizado Github.

Zenhub

Zenhub es una extensión de Chrome para github. Se utiliza para gestionar proyectos y funciona de manera nativa en la interfaz. Se basa en la metodología ágil yresulta verdaderamente útil a la hora de realizar y gestionar un proyecto, eso sí, hay que cumplir la tareas.

- **Alternativa estudiada: Trello**

  Trello es un gestor de tareas que permite el trabajo de forma colaborativa mediante tableros compuestos de columnas que representan distintos estados. Se basa en el método Kanban para gestión de proyectos, con

tarjetas que viajan por diferentes listas en función de su estado. Lo comencé utilizando pero al final cambié a Zenhub por su mejor integración con el código. Sin embargo me parece una herramienta muy interesante de la cual hago uso para temas personales.

StackOverFlow

Stack Over Flowst es una de las comunidades de desarrolladores más importantes del mundo [12] en la que se responden cuestiones de diferentes lenguajes. Sinceramente es una herramienta fundamental y existe una gran comunidad de usuarios de Angular que crece cada día.

Dillinger

Dillinger es una plataforma *cloud* basada en HTML5 que permite de manera online modificar código Markdown. Empleada para crear, modificar, añadir mejoras en los archivos **README.md** .

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger de manera breve los aspectos más interesantes del desarrollo del proyecto así como su justificación: decisiones que se han tomado, desarrollo, progreso general y problemas que surgieron durante toda la realización del proyecto.

5.1. Inicio del proyecto

Cuando llegó la hora de buscar proyecto, entre las diferentes opciones disponibles estaba *SWAPP*, una herramienta realizada años atrás en la universidad por un alumno que desarrolló una aplicación nativa en Android sobre el mismo tema.

Desde el primer momento me llamo la atención el proyecto porque resultaba un aplicación que en un futuro podía tener un uso real. Además se trataba de tecnologías web con las cuales poder realizar una aplicación híbrida. Me pareció una idea fascinante, sobre todo porque desconocía plataformas como *Cordova* o *Phohegapp*. Está claro que el principal peso del proyecto se lo lleva la aplicación web, dado que es la parte fundamental y sobre la que se va a sustentar todo lo demás. Por lo que resultaba fundamental escoger un buen *framework* web y estudiarlo de manera concienzuda.

En consecuencia surge el primero de los debates, qué tipo de *framework* usar. La multitud de opciones que existe en el mercado es realmente extensa.: una de las opciones a tener en cuenta para la elección de tu aplicación es comprobar que el *framework* sigue recibiendo soporte actualmente (Por ejemplo github tiene 'escaparates' donde agrupa los *frameworks*por tipos, algunos repositorios son públicos y es posible saber la frecuencia con la que se actualizan). Además una de las mejores cosas del software libre es que pue-

des interactuar con los propios creadores o desarrolladores de una tecnologías, abrir *issues* en github y en la mayoría de ocasiones recibes respuesta.

5.2. Formación

Desde el primer momento se priorizó la formación por delante de todo ya que el proyecto requiere conocimientos de dos tipos: por una parte conocimientos web, y por otro lado conocimientos que eran totalmente desconocidos para el alumno como manejo de la parte del servidor, bases de datos NosQL o desarrollo de aplicaciones híbridas.

Gracias a las nuevas tecnologías tan solo es necesario un ordenador y una buena conexión a internet para tener acceso a conocimiento infinito. Desde hace algunos años está en auge la formación online y las plataformas web que ofrecen cursos de diversos tipos. Existen numerosísimas plataformas [1] de este estilo, una de las más conocidas es Coursera, que reúne cursos online masivos, abiertos y gratuitos (MOOCS). El curso fue recomendado por mi tutor, desde hace algún tiempo la visibilidad de la parte gratuita se ha reducido pudiendo solo acceder a partes limitadas de los contenidos. Es posible realizar el curso completo pero no obtener *feedback* de otros usuarios ni tampoco subir a la plataforma los avances de código.

Por lo que en un primer momento se decidió realizar la siguiente especialización compuesta de seis cursos diferentes:

- *HTML, CSS y JavaScript*, una introducción al desarrollo web con las principales tecnologías. Qué es DOM y CSS.
- *Front-End Web UI Frameworks and Tools*, conocimiento de lo que es un framework web: Boostrap. También introducción a los preprocesadores Less, Sass.
- *Front-End JavaScript Frameworks: AngularJS*, introducción y desarrollo con AngularJS.
- *Multiplatform Mobile App Development with Web Technologies*, desarrollo UI/UX, el framework Ionic como complemento a AngularJS para crear aplicaciones hibridas con ayuda de Cordova.
- *Server-side Development with NodeJS*, lado del servidor con Node JS.
- *Full Stack Web Development Specialization Capstone.*

Los cursos pertenecen al curso *Full-stack web develompent* y son bastante completos en lo que a materia se refiere. Resulta fundamental saber algo de inglés porque, en caso de no saber, se hacen bastante pesados.

Después además realicé algún que otro curso en español, en esta ocasión la plataforma elegida fue *Udemy*. El enfoque de esta web es totalmente diferente a la anterior, mientras que la que hago referencia en el párrafo anterior en su mayoría se ofrecen grades cursos de universidades y profesores de prestigio de todo el mundo, Udemy se centra más en personas 'normales' . Es decir, *freelancers* que tienen conocimientos de algún campo en concreto y realizan un curso de una materia específica y lo venden a los usuarios. Existen tanto cursos gratuitos, de una duración menor, como cursos de diferentes precios. En mi caso he realizado algunos gratuitos y otros de pago, a saber:

- *Curso de Nodejs y Angular 2*: Aprende a desarrollar servicios RESTful (APIs) con NodeJS y MongoDB y a crear webs SPA con Angular2 y 4. [7]
- *Introducción a Angular 4 - Instalación y componentes*: Aprende las bases de Angular4 desde cero y paso a paso. [8]
- *Introducción teórica a los frameworks de desarrollo Web*: Aprende los conceptos teóricos necesarios para empezar a trabajar con frameworks de desarrollo MVC. [9]

La plataforma anterior también es muy didáctica, completa y accesible; además posee una característica que a mi parecer es fundamental: tienes contacto directo con el creador del curso que te resuelve dudas de código

5.3. Frameworks cliente

AngularJs vs Angular esta fue una de las dudas con las que más tiempo perdí al inicio del desarrollo y después de la fase de formación con la especialización de Coursera no sabía exactamente cuál de las dos escoger. Por una parte estaba la formación recibida en los seis cursos iniciales que todos fueron en AngularJS y por otra la inquietud del futuro ya que los frameworks evolucionan siempre a mejor y tarde o temprano los versiones iniciales se acaban quedando obsoletas.

Lo primero que hay que decir es que Angular no es la siguiente versión de AngularJS, sino que es un nuevo framework, escrito desde cero y con conceptos y formas de trabajar completamente distintos. Angular utiliza un sistema de inyección de dependencias jerárquico impulsando su rendimiento de forma increíble. Según algunos datos oficiales, Angular puede llegar a ser **5 veces más rápido** que la primera versión. Vamos a conocer algunas de las características generales de porque finalmente llegué a escoger este nuevo framework.

1. Angular está orientado a móviles y tiene mejor rendimiento.

2. Angular ofrece más opciones a la hora de elegir lenguajes.

3. Los controladores desaparecen haciendo ahora uso de componentes web, resulta más intuitivo y un estándar de futuro que todavía no encuentra soportado por todos los navegadores web pero en el futuro lo estará por lo que el rendimiento será todavía mayor.

4. Angular usa directamente las propiedades de los elementos y los eventos estándar del DOM

5. Su futuro es prometedor.

6. Una cosa curiosa: en varios años AngularJS tuvo unos 8500 *commits* en github, en tan solo unos meses Angular lleva más de 7845 *commits*.

Vamos a ver por ejemplo cómo han evolucionado los controladores (Así se conocían en AngularJS) a los componentes (Así se conocen en Angular) de una versión a otra.

En angularJS:

```
<body ng-controller="PeliculasController as peli">
  <h3>{{peli.data.titulo}}</h3>
  <h3 ng-bind="peli.data.titulo"></h3>
</body>
<script type="text/javascript">
(function (){
  angular
    .module('app')
    .controller('PeliculasController', PeliculasController);

  function PeliculasController() {
    var peli = this;
    peli.data= { id: 45, titulo: 'Una historia real' };
  }
})();
</script>
```

En Angular:

```
import { Component } from 'angular2/core';

@Component({
  selector: 'pelicula',
  template: '<h2>{{pelicula.titulo}}</h2>'
})
export class PeliculaComponent {
  public pelicula = { id: 45, titulo: 'Una historia real' };
}
```

Si podéis observar al inicio de la sección comencé haciendo referencia a *Angular2* y *AngularJS* pero ha medida que he avanzado simplemente he omitido el número. Podría parecer que lo he olvidado pero para nada es así, después del nacimiento de esa nueva versión que resultó ser un *framework* en sí mismo que aprender desde cero, desde el equipo de desarrollo de Angular lo que se intentó es describir al *framework* sin ningún tipo de numeración, simplemente existe un *framework* y se van a aplicar modificaciones sobre él para mejorarlo, nada más. Para entender mejor esta ruptura AngularJS y Angular no son compatibles pero sí lo son las versiones posteriores. De hecho hace dos meses, en Abril, lanzaron *Angular 4.0* que posee cambios menores respecto de su versión anterior pero son totalmente compatibles.

Figura 5.18: Características *Angular*. Fuente: `wikipedia.org/wiki/Angular`.

Respecto al futuro de Angular en la última presentación se dijo que podemos esperar una nueva versión 'mayor' cada seis meses, Angular 5 estará lista en Septiembre del 2017, la versión 6 en Marzo del 2018 y así sucesivamente. Está claro que todo proyecto de software que quiera avanzar a buena velocidad, en algún momento introducirá cambios que por desgracia no serán

compatibles con versiones anteriores, forzándote a tocar tu código para actualizarte y quizás sea esa una de las necesidades de autoformación ya que todo está en constante cambio.

Por supuesto cabe decir aquí que una no es mejor ni peor que la otra, AngularJS se convirtió en un framework extremadamente popular desde su lanzamiento [13] y a día de hoy se siguen realizando aplicaciones web con este *framework*. La comunidad de desarolladores que están detrás de su mantenimiento es enorme y es posible consultar dudas sin ningún tipo de problema. La apuesta por Angular es sin dudarlo una apuesta de futuro.

5.4. Tipo de base de datos

Otro de las disyuntivas sin lugar a dudas fue la elección de la base de datos. Mi conocimiento sobre bases de datos se limitaba a lo visto durante el grado en la universidad y lo aprendido durante mi estancia de prácticas, por lo que era suficiente para saber manejarme con una. Eso sí siempre había empleado **bases de datos SQL** o relacionales. Nada más comenzar a introducirme en el mundo de desarrollo web comencé a saber más sobre las bases de datos NoSQL. La diferencia conceptual entre SQL y NoSQL radica en que resuelven escenarios completamente diferentes y excluyentes el uno del otro; ya que para lo que resulta ideal SQL, no lo es NoSQL y viceversa. Aportada la idea principal vamos resumir las diferencias entre ambas de manera breve.

- SQL permite combinar de forma eficiente diferentes tablas para extraer información relacionada, mientras que NoSQL no lo permite o muy limitadamente.
- NoSQL permite distribuir grandes cantidades de información; mientras que SQL facilita distribuir bases de datos relacionales.
- SQL permite gestionar los datos junto con las relaciones existentes entre ellos; en NoSQL no existe este tipo de utilidades.
- NoSQL permite un escalado horizontal sin problemas – por su capacidad de distribución; mientras que escalar SQL resulta más complicado.

Entre las recomendaciones que he podido recopilar de algunos blogs [2] [10] estos son algunos de los entornos que **se recomiendan para usar** bases de datos NoSQL:

- Redes sociales: casi obligatorio.
- Desarrollo Web: debido a la poca uniformidad de la información que se encuentra en Internet; aun cuando también puede emplearse SQL.

- Desarrollo Móvil: debido a la tendencia – en crecimiento- de Bring Your Own Device.
- BigData: debido a la administración de grandísimas cantidades de información y su evidente heterogeneidad.
- Cloud (XaaS): *Everything as a service*. NoSQL puede adaptarse casi a cualquier necesidad del cliente, y sus particularidades varias.

Como ya he comentado anteriormente las elecciones son personales por lo que unas tecnologías no son ni mejores ni peores que las demás simplemente he tenido que decidir entre las alternativas de las que disponía en el mercado. En el caso de las bases de datos hay que tener claro qué y cómo se quiere almacenar y qué tipo de tecnologías se van a usar en, por ejemplo, la parte del cliente, para elegir que tipo de base de datos emplear, todas tienen sus ventajas e inconvenientes. Si bien es cierto que quizás MongoDB como base de datos haya sido ideal para este tipo de proyecto.

5.5. Metodologías

En cuanto a las metodologías, ya nombradas en apartados anteriores, se eligió una metología *Scrum* para la parte más técnica y la metodología *Lean start-up* para la parte más de empresa, de ésta última veremos más información en los anexos.

Scrum es un tipo de metología ágil que se aplica a grandes proyectos para cumplir las necesidades específicas de una manera estructurada y sistematizada. Entre las ventajas se encuentran la productividad, calidad y que se realiza un seguimiento regular de los avances del proyecto, logrando que los integrantes estén unidos, comunicados y que el cliente vaya viendo los avances. Lógicamente esta metodología no es posible aplicar al cien por cien en el proyecto ya que muchas de las características que posee están destinadas al trabajo en grupo, pero se ha intentado seguir una línea lo más semejante posible.

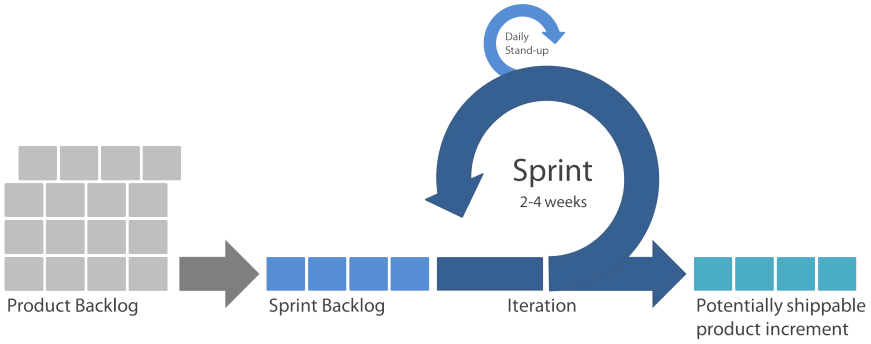

Figura 5.19: Esquema Scrum. Fuente: linkedin/pulse/posts/costa-ruitiago.

De una manera más o menos frecuente se siguió una estrategia de desarrollo incremental a través de iteracciones (*sprints)* englobadas en diferentes *milestones*. Al finalizar cada sprint se evaluaba lo realizado y, si era necesario, se realizaba una reunión con el tutor para esclarecer los pasos dados. Las tareas se estimaban en papel, ya que he utilizado un cuaderno en el que anotaba lo que iba aprendiendo y lo que iba realizando, además de los *boards* de Zenhub.

5.6. Desarrollo del algoritmo

El algoritmo que he usado para realizar el cambio de turno entre los usuarios es bastante sencillo y no muy complejo de entender.

Lo primero que hay que comprender es cómo funciona internamente la base de datos dado que va a ser la parte fundamental para que se produzca el intercambio deseado. *MongoDB* como ya sabemos se basa en colecciones de datos, por lo tanto para mi proyecto yo poseo una base de datos que consta de las siguiente colecciones:

- Users (Usuarios): es simplemente una colección para almacenar usuarios con los parámetros deseados, creada a partir de los cursos que he realizado.
- Event (Eventos): es la colección que proporciona el calendario escogido, es decir, se utiliza para almacenar cada uno de los eventos, en este caso turnos, que se añaden al calendario.
- Interchange (Intercambio): es una colección extra que se ha añadido para intercambiar entre los usuarios el turno

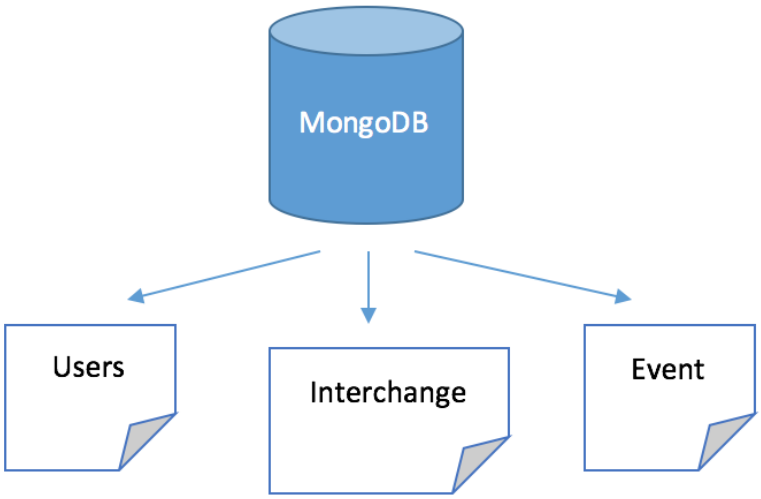

Figura 5.20: Esquema interno base de datos. Fuente: Elaboración propia.

Una vez entendido esto voy a explicar de una manera didáctica como se consigue el intercambio de turnos. Cada usuario posee su propio calendario, cada uno puede añadir sus propios turnos al calendario pero no ver los de los demás, los turnos son de cuatro tipos, se detallarán a continuación 5.6, y los cambios solo se producen entre tipos de turnos específicos.

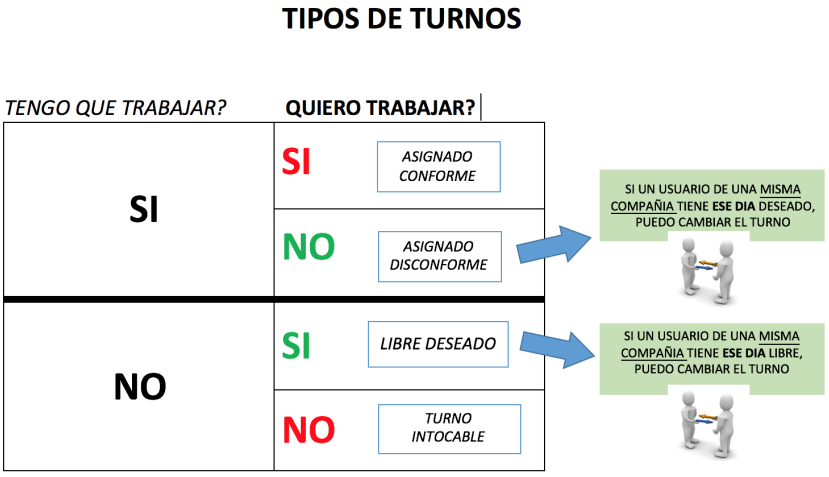

Figura 5.21: Esquema tipos de turnos. Elaboración propia.

Cada turno tiene asignada una prioridad que por defecto será **normal**. Existen varios tipo de prioridades que se asignarán dependiendo en que fase se encuentra ese turno, es decir, cuando un usuario solicita un turno cambiará de prioridad por que debe esperar hasta que el otro usuario acepte o rechace la petición. Por lo tanto se trata de una lógica que funciona a base de peticiones-respuesta. Cuando varios usuarios, de una misma empresa, puedan intercambiar sus turnos de acuerdo a la lógica de la imagen 5.6 entonces entran en juego las prioridades ya nombradas.

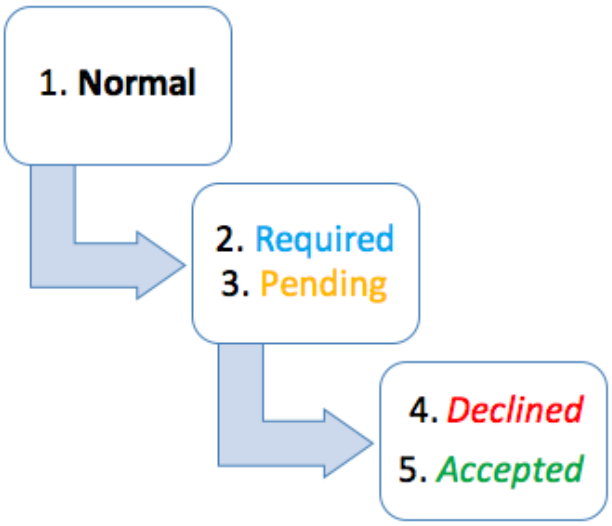

Figura 5.22: Esquema de las diferentes prioridades. Fuente: Elaboración propia.

Todos los turnos, mal llamados *eventos* dentro del código interno del calendario, poseen prioridad **normal**, cuando un usuario pregunta a otro por el cambio de turno, automáticamente el turno del usuario emisor pasa **required**, o requerido, y el turno del usuario destinatario a **pending**, o pendiente. Por lo que una vez se llega hasta aquí es el usuario destinatario el que tiene el poder de decisión, deberá aceptar o rechazar la petición. En función de lo que este usuario decida ese turno pasará a ser **accepted** o **declined**. Si el usuario que recibe la petición lo acepta se producirá el cambio de turno, si lo rechaza no se producirá el cambio y el turno pasará a estado normal.

Asimismo es necesario, debido a como se comporta la base de datos internamente, evitar que se produzcan tareas simultáneas en el calendario. *Angular 2+* proporciona un mecanismo interno para realizar tareas simultáneas pero

he considerado oportuno el separar cada proceso en un mecanismo diferente de manera que el usuario tiene claro en cada momento: los usuarios a los que puede enviar una petición (de la misma manera el usuario que la recibe tiene constancia), si esa petición de cambio ha sido acepta o rechazada y por último confirmar ese cambio o rechazo. De esta manera estamos garantizando al cien por cien que el usuario ha sido informado sobre si su petición ha sido aceptada o rechazada y evitando posibles engaños por parte de los usuarios.

Por otro lado hay que notar que la colección *Interchange* empleada para realizar y almacenar el cambio de turno, puede resultar muy útil en el futuro para implementar, de alguna manera una forma de que el usuario pueda ver una lista de las peticiones de cambio de turno rechazadas o aceptadas en el pasado. En un primer momento se pensó en borrar la información de cada intercambio pero si se piensa un poco más se puede llegar a la conclusión de que estos datos pueden resultan realmente interesantes para, por ejemplo, un administrador del sitio.

5.7. Desarrollo web

El desarrollo de la aplicación web se ha realizado de manera progresiva aunque durante varias fases, la primera en

5.8. Desarrollo móvil

El desarrollo móvil ha sido la última parte

5.9. Documentación

La documentación se ha realizado de manera progresiva siempre que se podía se añadían partes de la misma.

5.10. Reconocimientos

Reconocimientos durante el desarrollo:

- YUZZ: BarterAPP fue elegida para participar en el programa YUZZ 2017, patrocinado por el Banco Santander-Universidades, se trata de un programa de formación que se extiende de enero a junio en el que se aprenden materias como marketing, financiación, puesta en marcha de un proyecto o análisis de mercados entre muchas otras cosas.
- ANGULAR CAMP: Quizás no sea un reconocimiento en sí mismo del proyecto pero teniendo en cuenta de que hace seis meses no tenía ni idea

de Angular, he sido seleccionado, tras una entrevista previa, para formar parte del Angular Camp que se desarrolla en la ciudad de Barcelona durante los días 6 y 7 de julio.

Trabajos relacionados

En esta sección se pretende realizar un acercamiento a los posibles competidores que la empresa podría tener en un potencial mercado, en caso de que llegara a ponerse a la venta. y también a los trabajos en relación ha ésta que hayan existido. En el plan de empresa realizado, del cual se dan más detalles en los anexos, se ha realizado un estudio de la competencia más exhaustivo.

6.1. Trabajos previos

Indispensable hacer una mención a un trabajo fin de grado previo realizado en este mismo grado y universidad. Aunque no tiene nada que ver el desarrollo, la idea y el tutor sí fueron el mismo luego era necesario nombrarlo.

El trabajo tuvo el nombre de ***SWAPP*- Aplicación para gestionar el cambio de turnos y guardias** realizado por el compañero Diegro Prado Nebreda. Desafortunadamente no se encuentra ya en github el proyecto para poder verlo.

6.2. Competidores

Nombre	*Tipo*
Saturnos Pro	Aplicación Android para smartphones
CuadraTurnos Free	Aplicación Android e IOS para smartphones
Shyft	Aplicación Android e IOS para smartphones

Tabla 6.1: Listado de posibles competidores

En la tabla anterior se describen tres aplicaciones. Se ha considerado incluir estas tres por diferentes razones que se expondrán a continuación.

Saturnos pro La primera de ellas fue la que más nombro la gente durante las encuestas realizadas para el plan de empresa ya nombrado. Si hay una aplicación usada en este campo en España sin duda es ésta, si bien es cierto que tiene algunas limitaciones como por ejemplo que el único medio para usarla sea un dispositivo móvil con Android. La principal desventaja quizás sea el precio pero también era la más conocida por lo que eso significa que sea usa aunque no masivamente.

Cuadraturnos Free Fue otra de las aplicaciones nombradas durante la encuesta aunque con mucha menor insistencia, tan solo tres o cuatro personas la conocían. Está disponible en las principales plataformas móviles y resulta fácil de usar pero la interfaz resulta ser un poco antigua .

Shyft Sin duda es la aplicación más reciente, sólo esta disponible en Estados Unidos pero es el modelo a fijarme como herramienta. La he probado y usado y quizás no resulta tan intuitiva como parece ser.

Nota: Se incorporará un anexo extra con el trabajo realizado en YUZZ sobre la competencia.

6.3. Otros trabajos

Aunque no directamente relaciones si he encontrado algún trabajo o proyecto que intentaban desarrollar algoritmos para suplir problemas de asignación de turnos. Me a parecido interesante y curioso incluirlos en esta sección.

Trabajo1 Diseño de un modelo de asignación de turnos para la operación de sistemas de transporte masivo tipo BRT.[6]

Trabajo2 Algoritmo GRASP [3] para resolver el problema de asignación de horarios en empresas de demanda variable [17]

Conclusiones y Líneas de trabajo futuras

En está última sección se exponen las conclusiones derivadas del trabajo. De la misma manera abordaremos las línea de trabajo futura que puede llevar la aplicación y que podrían ayudar a mejorarla y a posicionarla en el mercado si realmente se decide lanzarla.

7.1. Conclusiones proyecto

Resumen tecnologías empleadas

7.2. Conclusiones personales

7.3. Líneas de trabajo

Una vez finalizado el trabajo es importante

Tecnologías	Front-End	Back-End	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
Angular2/4	X			
Bower	X			
Gulp	X			
PHP		X		
Karma + Jasmine	X			
Ionic 2		X		
Latex		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 7.2: Herramientas y tecnologías utilizadas en cada parte del proyecto

Bibliografía

[1] Silvina Mariel Castro, Claudio Ariel Clarenc, Carmen López de Lenz, María Eugenia Moreno, and Norma Beatriz Tosco. Investigación colaborativa sobre lms, 2013. (Citado en página 24.)

[2] Alida Vergara Jurado Facilcloud.com. Sql vs nosql ¿cuál debo usar?., 2016. (Citado en página 28.)

[3] Fernando Berzal Galiano. Grasp: Greedy randomized adaptive search procedure, 2011. (Citado en página 36.)

[4] Matt Lewis. Angular 2+ calendar documentation, 2017. (Citado en página 15.)

[5] Robert Cecil Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. PRENTICE-HALL INTERNATIONAL EDITION, 2008. (Citado en página 5.)

[6] Diego F. Quintero-Moncada and Carlos L. Quintero-Araújo. Diseño de un modelo de asignación de turnos para la operación de sistemas de transporte masivo tipo brt, 2013. (Citado en página 36.)

[7] Victor Robles-Udemy. Curso de nodejs y angular - crea webapps con el mean stack., 2017. (Citado en página 25.)

[8] Victor Robles-Udemy. Introducción a angular 4 - instalación y componentes., 2017. (Citado en página 25.)

[9] Victor Robles-Udemy. Introducción teórica a los frameworks de desarrollo., 2017. (Citado en página 25.)

[10] Craig Buckler sitepoint.com. Sql vs nosql ¿cuál debo usar?., 2015. (Citado en página 28.)

[11] Slant.co. Javascript ides or editors, 2015. (Citado en página 11.)

[12] Cristina Sánchez. El genio que convirtió su blog para informáticos en un gigante de internet, 2017. (Citado en página 22.)

[13] W3Techs. Usage statistics and market share of angularjs for websites., 2017. (Citado en página 28.)

[14] Wikipedia. Singleton pattern — Wikipedia, The Free Encyclopedia, 2016. (Citado en página 10.)

[15] Wikipedia. Model view controller, 2017. (Citado en páginas v, v y 8.)

[16] Yeeply. El desarrollo web, en alza como nunca antes., 2016. (Citado en página 2.)

[17] David Álvarez Martínez, Eliana Mirledy Toro Ocampo, and Ramón Alfonso Gallego Rendón. Multiskilled workforce scheduling for business with variable demand using the grasp algorithm, 2010. (Citado en página 36.)

[18] Proyectos ágiles. ¿qué es scrum?, 2012. (Citado en página 10.)