Panel informativo de eventos

JULIO 2018

VNiVERSiDAD
Đ SALAMANCA

AUTOR ÁLVARO MATEOS LABRADOR
TUTORA (USAL) VIVIAN FÉLIX LÓPEZ BATISTA
TUTORES (HP SCDS) ALONSO TORRES FRANCISCO
FERNANDO GONZÁLEZ PÉREZ

Da. Vivian Félix López Batista, profesora del Departamento de Informática y Automática de la Universidad de Salamanca.

CERTIFICA:

Que el trabajo titulado "Panel informativo de eventos" ha sido realizado por **Álvaro Mateos Labrador**, con DNI 70913896C, y constituye la memoria del trabajo realizado para la superación de la asignatura "*Trabajo de fin de grado"* del Grado en Ingeniería Informática de la Universidad de Salamanca.

Y para que así conste a todos los efectos oportunos.

Vivian Félix López Batista

Salamanca, 4 de julio de 2018.

Tabla de contenido

Tabla de ilustraciones

Resumen

Este proyecto nace de un acuerdo entre la empresa HP SCDS y la Universidad de Salamanca, mediante el cual la empresa presenta varias propuestas de trabajo de fin de grado a los alumnos de la universidad. El proyecto ha sido desarrollado imitando el entorno de desarrollo real en la empresa HP.

La empresa manifiesta la necesidad de una aplicación para presentar información útil a los asistentes de los eventos y reuniones, en los momentos previos a su inicio. Esta información se mostraría proyectada en una pared o en una gran pantalla. La pantalla es dividida en varios subpaneles o módulos.

Los módulos con los que ha de contar la aplicación son los siguientes: un reproductor de videos (locales o de la web), un difusor de Twitter (seguimiento de cuentas o hashtag), un informador de tráfico local en tiempo real y previsión del tiempo en las próximas horas.

La aplicación será una página web embebida en una aplicación nativa. Esto se consigue con el framework Electron. Principalmente se usan los lenguajes HTML y TypeScript.

Para el desarrollo de la aplicación web se ha utilizado el framework Angular.

Para obtener datos de Twitter y de otros servicios web (módulos de tiempo y de tráfico local) se consultan sus API REST.

Para gestionar el proyecto se han utilizado las metodologías ágiles Scrum y Kanban.

Palabras clave: Aplicación web, Electron, Angular, TypeScript, HTML, API REST, Scrum.

Summary

This project arises from an agreement between the company HP SCDS and the University of Salamanca, in which the company makes several project proposals to the students of the university. The project is carried out imitating the work environment of the company.

The company needs an application to present useful information to the attendees of its events and meetings, before its beginning. This information will be projected or displayed in a large screen. The screen would be divided in different modules.

The following modules should be included in the application: a local or online video player, a Twitter monitor (accounts or hashtags), a real time traffic map, and hourly weather forecast.

The application will be a web page embeded in a native application. This will be achieved with the framework Electron. HTML and TypeScripts are the coding languages used in the project.

Angular framework has been used during the development of the web page.

Some REST APIs have been consulted in order to fetch data from Twitter and other web services (traffic and weather forecast modules).

Scrum and Kanban are the agile methodologies used to manage the project workflow.

Keywords: Web app, Electron, Angular, TypeScript, HTML, API REST, Scrum.

Introducción

El presente documento recoge la memoria del trabajo de fin de grado titulado *Panel informativo para eventos* del *Grado en Ingeniería Informática* de la USAL.

Este trabajo de fin de grado tiene una particularidad. Es uno de los ofertados por la empresa HP SCDS, ubicada en León. El 25 de octubre de 2017 tuvo lugar una reunión en la que la empresa expuso cinco propuestas de TFG para desarrollar en un ámbito especial.

Estos TFG se realizarían bajo la tutoría de uno o varios profesionales de la empresa HP, además del tutor correspondiente por parte de la Universidad de Salamanca. Esto supone una gran oportunidad para el alumno, pues se desarrollaría del proyecto de la misma manera que se haría en la empresa, en un entorno de trabajo real.

Tras mostrar interés en uno de los proyectos, el *Panel informativo para evento*, este es adjudicado y se establece contacto con los tutores de HP para conocer más información sobre el proyecto a realizar.

El proyecto ha sido desarrollado por **Álvaro Mateos Labrador**, bajo una triple tutoría: **Vivian Félix López Batista** como tutora por parte de la USAL, y **Alonso Torres Francisco** y **Fernando González Pérez** como tutores por parte de HP SCDS. Cabe destacar que uno de los tutores de HP para este proyecto, Alonso Torres Francisco, tuvo que sustituir a Sergio Martínez Prieto por motivos internos de la empresa.

Este marco especial resulta en la imposición de la metodología utilizada por HP para el desarrollo de sus proyectos: Scrum. Más adelante se explicará qué aspectos de esta metodología ágil se utilizaron durante el desarrollo. Desde la empresa también se han dado algunas directrices sobre qué tecnologías y herramientas usar, con el objetivo de imitar el funcionamiento interno de HP.

El proyecto consiste en el desarrollo de una aplicación web que sería utilizada por la empresa en eventos o reuniones para mostrar en grandes pantallas información que pueda resultar útil para los asistentes. La descripción dada inicialmente por HP es la siguiente.

> *Se trata de hacer una web configurable para presentar en grandes pantallas (o murales) localizadas en eventos o reuniones. El tamaño de la página web sería de sólo una pantalla y se podrán configurar distintos módulos que aparezcan en ella. Entre estos módulos al menos deberán existir los siguientes: un reproductor de videos (locales o de la web), un difusor de Twitter (seguimiento de cuentas o hashtag), un informador de tráfico local en tiempo real y previsión del tiempo en las próximas horas.*

Esta es la única especificación que se realiza por parte de la empresa, dando al alumno una gran libertad de desarrollo, y al mismo tiempo la responsabilidad de tomar unas decisiones acertadas para satisfacer la especificación de la mejor manera posible.

Tras la adjudicación se realiza una reunión con los tutores de HP en la que se explica la metodología que se va a utilizar, Scrum, y se recomienda la lectura del libro "*Scrum and XP from the trenches*", de Henrik Kniberg [1]. También se aconseja el uso del framework *Angular* y de Git como sistema de control de versiones. Estos elementos serían estudiados por el alumno durante una fase inicial de documentación.

Necesidades de la empresa

Este subapartado explica la necesidad de la empresa con respecto al proyecto.

La necesidad surge desde el ámbito nacional de la empresa HP, no solo desde *HP SCDS León*. En todas las sucursales del grupo HP se realizan numerosos eventos y reuniones. Estos eventos se sitúan en salas de junta o en salones de actos.

Ilustración 1 Salón de actos utilizado por HP en Madrid

En la Ilustración 1 se muestra uno de los espacios utilizados para estos eventos. Estos espacios suelen contar con un equipo para proyectar la pantalla de un ordenador.

El proyecto busca la construcción de una aplicación de escritorio (basada en tecnología web) para sacar partido a estos proyectores en el **momento previo al comienzo del evento**. La aplicación mostraría en la pantalla información útil para los asistentes que están esperando el inicio del evento. Es importante destacar que la aplicación no se utilizaría durante el desarrollo del evento, sino en los momentos previos al comienzo.

La aplicación ocuparía la totalidad de la pantalla, mostrando un panel dividido en varios módulos a modo de cuadrícula. El panel se podría configurar hasta con cuatro módulos distintos:

- Reproductor de vídeos: La empresa suele contar con vídeos promocionales sobre sus eventos. Este módulo permitiría la reproducción en bucle de dichos vídeos.
- Difusor de *twitter*: La empresa suele acuñar un *hashtag* para facilitar la discusión sobre sus eventos en la red social *Twitter*. De aquí surge la necesidad de disponer de un módulo que muestre en pantalla *twits* relacionados con algún *hashtag* o cuenta de *Twitter* para fomentar la discusión en redes sociales.
- Previsión del tiempo: algunos de los eventos que realiza la empresa se extienden durante uno o varios días completos. Puede resultar útil para los asistentes conocer la previsión del tiempo en las próximas horas, y por eso se requiere de este módulo.
- Mapa del tráfico en la zona: algunos eventos cuentan con participantes de distintas áreas del territorio nacional. Conocer la situación del tráfico en el área próxima al evento puede resultar útil para los asistentes, tanto a la hora de predecir retrasos en el inicio del evento, como a la hora de decidir una ruta una vez finalice el mismo.

Estructura de la memoria

En este apartado se enumeran las distintas secciones del documento, incluyendo una breve descripción de su contenido.

En la sección **Objetivos del trabajo,** se presentan los objetivos funcionales y no funcionales que debe satisfacer la aplicación a desarrollar en el proyecto. También se incluyen una serie de objetivos personales que tendrán gran influencia en el proyecto.

En la segunda sección, **Metodología**, se describen las metodologías ágiles que han sido usadas en el proyecto. Principalmente *Scrum* y *Kanban*. Al final de la sección se resume el desarrollo del proyecto desde un punto de vista puramente metodológico.

En la sección **Técnicas y herramientas** se enumeran y describen brevemente todas las técnicas y teorías que respaldan el proyecto, así como las herramientas que han facilitado el desarrollo.

El apartado **Aspectos relevantes del desarrollo** se centra en explicar cómo se consiguió la aplicación final sin entrar en la metodología utilizada, a la cual se le ha dedicado un apartado en exclusiva.

En la sección **Líneas de trabajo futuras** se mencionan algunas funcionalidades o características que podrían desarrollarse en el futuro.

En la última sección, **Conclusión**, se analiza si se han satisfecho los objetivos planteados en la primera sección.

Objetivos del trabajo

El objetivo principal es la construcción de una aplicación que sería utilizada por la empresa en eventos o reuniones para mostrar en grandes pantallas información que pueda resultar útil para los asistentes.

Objetivos funcionales

- Crear, modificar y eliminar presentaciones con distintos módulos informativos.
- Las presentaciones y su configuración deben ser almacenadas en disco para conseguir su persistencia.
- Visualizar *twits* de una determinada cuenta, *hashtag*, o resultado de una búsqueda.
- Reproducir un video o una lista de ellos, tanto locales como de Internet.
- Mostrar la previsión del tiempo en las próximas horas en un lugar determinado.
- Mostrar información del tráfico cercano a un lugar determinado.
- Dado el carácter internacional de la empresa, es necesario poder cambiar el idioma de la aplicación.
- La aplicación debe ser compatible con los principales sistemas operativos de escritorio: Windows, Mac OS y Linux.

Objetivos no funcionales

- La aplicación debe ser una *Single Page Application* (web) desarrollada con el framework Angular y respetando la arquitectura de la misma.
- Experiencia de usuario agradable, con una interfaz que pueda ser usada por cualquier usuario sin la necesidad de leer un manual de instrucciones.
- Conseguir una estética moderna y acorde con los principios de diseño de HP.
- El código de la aplicación debe estar sujeto al sistema de control de versiones *Git*.

Objetivos personales

- Desarrollar un proyecto para una empresa real y en un entorno real.
- Conseguir un puesto de trabajo la empresa con la que se desarrolla el proyecto, aprovechando la oportunidad que se brinda.
- Aprender sobre la metodología Scrum y otras metodologías ágiles, a penas mencionadas durante la carrera, pero con una importante presencia en el mundo empresarial. Desarrollar el proyecto haciendo Scrum supone un gran reto dado el poco conocimiento sobre el mismo antes de comenzar con el proyecto.
- Aprender sobre el entorno de desarrollo web y las tecnologías relacionadas con él: JavaScript, HTML, DOM, TypeScript, Node.js y NPM.
- Aprender sobre el framework Angular, utilizado por la empresa HP y por muchas otras en el mundo de la informática.
- Aprender a usar el sistema de control de versiones Git, ampliamente usado en la mayoría de las empresas y organizaciones informáticas.
- Aprender sobre otras teorías, técnicas y herramientas que sea necesario aplicar durante el desarrollo del proyecto.

- Conseguir una calidad máxima y conseguir que la aplicación tenga todas las funcionalidades requeridas. Supone un gran reto dado el desconocimiento de prácticamente todas las técnicas y herramientas con las que se va a desarrollar el proyecto. Para conseguir una buena calidad es necesario documentarse y comprender al máximo los elementos anteriormente descritos, para posteriormente implementar la aplicación, en un periodo de tiempo bastante limitado y que debe ser compaginado con la necesidad de aprobar todas las asignaturas de la carrera en cuatro años.

Metodología

Introducción

Desde HP se impuso el uso de la metodología Scrum para el desarrollo del proyecto. Se recomendó la lectura del libro "*Scrum and XP from the trenches*", de Henrik Kniberg [1]. Este libro explica los distintos artefactos de esta metodología desde un punto de vista práctico. Además, incluye información acerca de otras metodologías ágiles como *Extreme Programming* (XP) o Kanban. Adicionalmente se estudió "*La guía de Scrum*" escrita por Ken Schwaber y Jeff Sutherland [2], y se consultó el libro "*Kanban and Scrum: Making the most of both*" de Henrik Kniberg y Mattias Skarin [3].

Después de estas lecturas se conocen distintas metodologías para el desarrollo ágil. Resulta interesante clasificarlas en función de lo preceptivas que son. Una metodología es más preceptiva que otra si incluye más reglas a seguir. El adjetivo contrario sería adaptable, cuando una metodología ofrece más libertad a la hora de tomar decisiones [3]. Las metodologías conocidas son las siguientes, comenzando con la más adaptable: *Kanban*, *Scrum*, *Extreme Programming* (XP) y *Rational Unified Process* (RUP).

Según sus creadores, solo se hace Scrum si se siguen todas las reglas de Scrum. Si no, se está utilizando otra metodología e implementando artefactos adicionales de Scrum [2]. Dada la naturaleza unipersonal del equipo se hace imposible implementar todos los artefactos de Scrum, por lo que no se puede decir que el proyecto ha sido realizado haciendo Scrum. Sin embargo, Scrum es el marco de trabajo modelo que se ha tratado de imitar al máximo y por ello, disculpándonos de los creadores de Scrum, diremos excepcionalmente que se ha hecho Scrum.

La metodología de la que se puede decir que se ha hecho uso pleno es Kanban, al ser la menos restrictiva y fácil de implementar. Por tanto, el proyecto se ha desarrollado usando Kanban e implementando ciertos artefactos de Scrum.

En los siguientes subapartados se explicarán los artefactos o reglas más importantes de las metodologías utilizadas. También se darán ciertos detalles de cómo se han utilizado.

Kanban

Kanban es la metodología ágil más sencilla, ya que solo consta de tres reglas [4]:

- **Visualizar el flujo de trabajo**: en Kanban se suele utilizar una pizarra con tres columnas: una para las tareas por hacer, otra para las tareas en proceso, y otra para las tareas terminadas. Adicionalmente se pueden añadir otras columnas o modificar las existentes con el fin de adaptar Kanban a nuestras necesidades. Durante el desarrollo se utilizó una pizarra Kanban para cada *Sprint* (periodo de tiempo acotado en el que se realiza una iteración en la metodología Scrum). La primera columna almacenaba el *Sprint* Backlog. Cada elemento de la primera columna se divide en tareas que aparecían en la segunda columna ("por hacer"). La tercera columna se utilizaba para las tareas actualmente en proceso, y la cuarta para las tareas ya terminadas. En la quinta columna se colocaban las *User Stories* ya terminadas. Se puede ver un ejemplo del primer *sprint* en la Ilustración 2.

Ilustración 2 Tabla Kanban del Sprint 1

- **Limitar trabajo en proceso:** Solo puede haber como máximo un número de tareas determinadas en proceso. En este proyecto se ha limitado a una única tarea.
- **Medir la velocidad de desarrollo:** consiste en medir y evaluar la agilidad con la que se completan las tareas. En este proyecto se evaluaban en función de las *User Story* terminadas en cada *Sprint*, con lo cual esta medida está más ligada a Scrum que a Kanban y por tanto se explicará en el apartado correspondiente de Scrum.

Scrum

Scrum es un proceso iterativo e incremental, en el que en cada ciclo de desarrollo se entrega un producto terminado y listo para poner en producción.

Está compuesto por el equipo Scrum y por diversos artefactos, eventos y unas reglas asociadas [2] [1]. Los elementos de Scrum que se han utilizado durante el proyecto se detallan a continuación. Es importante destacar que solo se incluyen aquellos utilizados, omitiendo aquellos que no han sido utilizados porque no tenían cabida en el proyecto.

- **Product Backlog:** Consiste en una lista priorizada de requisitos que deben incorporarse al producto. Esta lista normalmente se compone de las llamadas User Story, aunque Scrum no especifica el formato del Product Backlog. El Product Backlog está en continua evolución, ya que se pueden añadir, eliminar y ordenar sus elementos en cualquier momento.
- Una **User Story** (US) describe una característica que se quiere ver en el sistema. Se escriben siguiendo el siguiente modelo, admitiéndose pequeñas variaciones:

  Como <rol> quiero <algo> para poder <beneficio>.

  Adicionalmente se estima una medida de tiempo para completar la historia. En el presente proyecto se ha utilizado la conocida medida de Story Points (SP). Los SP no especifican la unidad de tiempo, sino que se utiliza para comparar el tamaño de unas historias con otras. Esto significa que, si por ejemplo una US "A" mide 1 SP y otra US "B" mide 2 SP, B se tardará en completar aproximadamente el doble de tiempo que A. Cabe destacar que a petición de HP estos valores tienen que coincidir con un número de la serie de Fibonacci, a excepción del 2. Esto facilita la tarea de estimación, ya que establece unos incrementos significativos que hacen fácil comparar las distintas historias.
  En el proyecto el Product Backlog se elaboró en una tabla de cálculo (Excel) de Google, disponible en línea para que tanto el desarrollador como los tutores de HP (product owner) pudieran acceder y modificar la última versión de ésta.

	A	B	C
1	ID	User story	Story points
2	1	Como usuario quiero poder instalar la aplición desde un instalador para poder usarla en cualquier máquina	3
3	2	Como usuario quiero que la aplicación se muestre sin bordes para que tenga el mismo aspecto independientemente del SO	1
4	3	Como usuario quiero poder crear distintas presentaciones para distintos eventos	3
5	4	Como usuario quiero poder configurar cada presentación con distintos módulos	8
6	13	Como usuario quiero poder visualizar presentaciones en la ventana del panel de eventos	5
7	15	Como usuario quiero poder guardar la configuración en disco	1
8	7	Como usuario quiero poder añadir un módulo que muestre twits de una cuenta o hashtag	3
9	14	Como desarrollador quiero tener un manual para saber como generar ejecutables e instaladores de la app	1
10	9	Como usuario quiero poder añadir un módulo que reproduzca videos locales	3
11	5	Como usuario quiero poder cambiar el título de la presentación	1
12	12	Como usuario quiero poder añadir un módulo que muestre información del tiempo en la zona en las próximas horas	3
13	18	Como usuario quiero ver un mensaje informativo cuando un módulo esté en blanco o no pueda mostrar nada	1

Ilustración 3 Extracto del Product Backlog

En la Ilustración 3 se muestra un extracto del Product Backlog en un determinado momento (recordemos que el product backlog siempre está en evolución). Las US sombreadas ya han sido incluidas en algún *sprint*. Las que aparecen con fondo blanco serían el product backlog real, ya que las US ya realizadas deberían ser eliminadas. En el proyecto se han mantenido sombreadas para una mayor trazabilidad. El formato de la tabla (Excel *online*) permite que el product owner pueda priorizar o añadir nuevas US en cualquier momento. También es posible actualizar la estimación, tarea que también puede realizar el desarrollador.

El product backlog al completo se puede consultar en el ***Anexo I***.

- **Equipo Scrum**: compuesto por el *Product Owner,* el *Scrum Master* y el equipo de desarrollo (o desarrolladores).
    - **Product owner**: su tarea es gestionar el product backlog. Dicha gestión es la manera de dirigir al equipo de desarrollo. En el proyecto los tutores de HP han hecho las veces de Product Owner.

- **Development team:** equipo multidisciplinar de desarrollo. Debe ser capaz de transformar los elementos del product backlog en un producto entregable. Tiene que ser autoorganizado, multifuncional y horizontal. En el caso del proyecto está compuesto únicamente por el autor del mismo.
- **Scum master:** es el responsable de que se aplique Scrum correctamente en todas sus dimensiones. En el caso del proyecto esta figura no existe.

- ***Sprint*:** es un periodo de tiempo acotado (con duración máxima de un mes) durante el cual se crea un entregable. El *sprint* se planea en una reunión llamada *Sprint* Planning, en la que el equipo de desarrollo selecciona las US que pasarán a formar parte del ***Sprint* Backlog**. También se pone un objetivo para el *Sprint* y una fecha de fin.

  Cuando termina el *Sprint* se realiza una reunión llamada ***Sprint* Review**, en la que se realiza una demostración del entregable y se evalúa el *sprint*. Como inmediatamente después de un *sprint* comienza el siguiente estas reuniones se han hecho el mismo día.
  El objetivo es una frase concisa que describe lo que se pretende conseguir con el *Sprint*. Sirve para facilitar la toma de decisiones al equipo de desarrollo durante el *sprint*.

  Cada una de las otras hojas del libro Excel del Product Backlog está dedicada a un *sprint*. Contienen el extracto del product backlog que se va a realizar en el *sprint*. Esta lista sería lo que se conoce como *sprint* backlog. También se incluye el objetivo del *Sprint* y la fecha de inicio y fin del *Sprint*, así como un enlace a la tabla Kanban (plataforma Trello) que utiliza el equipo de desarrollo durante el *sprint*. También están incluidas en el ***Anexo I***.

	User story	Story points
1	Como usuario quiero poder instalar la aplición desde un instalador para poder usarla en cualquier máquina	3
2	Como usuario quiero que la aplicación se muestre sin bordes para que tenga el mismo aspecto independientemente del SO	1
3	Como usuario quiero poder crear distintas presentaciones para distintos eventos	3
4	~~Como usuario quiero poder configurar cada presentación con distintos módulos (ELIMINADA POR FALTA DE TIEMPO)~~	8

Ilustración 4 Extracto de la hoja del Sprint 1

En la Ilustración 4 se puede ver la hoja correspondiente al *Sprint* 1, en la que se eliminó la última US. Esto se debe a la imprecisión de las estimaciones al inicio del proyecto. Hay muchas maneras de **estimar** en *Sprint*. En el proyecto se estimó utilizando una técnica conocida como "*Yesterday's weather*". Este

método se basa únicamente en estimar basándose en la intuición y la experiencia. Todos los métodos de estimación, incluyendo *Yesterday's weather*, dependen en gran medida de la experiencia y necesitan ser refinados con el tiempo.
Durante un proyecto realizado en la asignatura "Gestión de Proyectos" tuve la oportunidad de estimar la duración de un proyecto utilizando un método llamado *Puntos de caso de uso*. Este método de estimación era complejo y costoso. No obstante, el resultado de la estimación dependía de unos factores de peso que ajustaban los resultados. Estos factores se dividen en dos categorías y se puntúan del 1 al 5. La manera de estimar estos factores es bastante subjetiva y también depende de la experiencia. Por ejemplo, uno de estos factores, concretamente de la categoría "Factores ambientales", era "Experiencia en orientación a objetos", que no es evaluable de manera objetiva. Resultaba curioso ver que alterando ligeramente la puntuación de distintos factores la estimación final podía cambiar radicalmente.
En el libro "*Scrum and XP from the trenches"* se explican otros métodos de estimación más acordes con *Sprint*, de complejidad similar a UCP. Sin embargo, el propio autor afirma que el coste y complejidad de planificar con los mismos no es recompensado con una precisión mucho mayor a "*Yesterday's weather*", especialmente en proyectos jóvenes y equipos con poca experiencia estimando.
Por todo esto decidí estimar tanto los User Story Point, como las US que se podían incluir en un *Sprint* utilizando el simple método de *Yesterday's weather*. Esto me permitió dedicar más tiempo a, por ejemplo, documentarme sobre Angular y su arquitectura, que en mi opinión iba a traer más calidad y "disciplina" al proyecto que unas estimaciones complejas pero imprecisas.

Resumen de la metodología aplicada

El proceso de desarrollo se puede resumir de la siguiente manera:

En un primer momento se realizó por parte del alumno un primer boceto del Product Backlog, refinado posteriormente por los tutores de HP.

En la primera reunión se seleccionaron los primeros US y se planificó el primer *sprint*. Después el alumno creaba una tabla Kanban para gestionar el flujo de trabajo durante el *Sprint*. Las distintas US se descomponen en tareas, que se van realizando y trasladando a la columna de tareas terminadas.

Cuando ha finalizado un *sprint*, se contactaba con los tutores para concretar la fecha de una reunión en la que se evaluaba el *sprint* terminado y se planificaba el siguiente *sprint*.

Es importante destacar que hay muchas maneras de aplicar Scrum. Scrum define únicamente roles y algunas normas, pero la forma de implementarlas en un equipo concreto es responsabilidad de dicho equipo. Así, todo lo explicado anteriormente

conforma la implementación particular de *Sprint* que hemos realizado el desarrollador y los tutores del proyecto.

Técnicas y herramientas

En este apartado se mencionan las tecnologías, técnicas, herramientas y principios teóricos que se han utilizado en el proyecto. A pesar de la extensa tarea documentación que el alumno ha realizado sobre todos los puntos, se tratará únicamente de explicar brevemente en que consiste cada una de las técnicas o herramientas para que el lector pueda hacerse una idea de en que consisten las mismas.

Esto se debe a que el objetivo de la memoria es presentar como y con qué tecnologías se ha desarrollado el proyecto. Este documento no tiene como finalidad ser una fuente de aprendizaje sobre las técnicas y herramientas utilizadas, y por tanto se debe acudir a otras fuentes más aptas si se quiere profundizar el conocimiento de estas.

Aplicación web embebida

La aplicación consistirá una página web. Esta página web estará embebida en un framework llamado Electron, que se explicará posteriormente. Este framwork permite mostrar una página web en una ventana como si de una aplicación nativa se tratase. También permite comunicarse con el SO para conseguir funcionalidades propias de una aplicación de escritorio que no son accesibles para páginas web normales. Por tanto, la aplicación estará escrita en HTML, para definir los componentes visuales de la aplicación, Javascript para manejar y alterar el DOM que genera dicho código HTML, y CSS para dar forma y estilo a los distintos componentes de la aplicación.

TypeScript y JavaScript

JavaScript [5] es el lenguaje de programación que se utiliza para manipular y dar comportamiento a las páginas web. Es un lenguaje interpretado y orientado a objetos. Se rige por la especificación ECMAScript, la cual incorpora grandes novedades en sus últimas versiones.

TypeScript [6] [7] es un lenguaje de programación creado por Microsoft. Es un superconjunto de JavaScript, es decir, una extensión de este. La principal diferencia radica en que TypeScript es fuertemente tipado. El código TypeScript es compilado a JavaScript para poderse ejecutar en cualquier intérprete de JavaScript, como Node.js o un navegador web. Mejora la robustez, legibilidad, trazabilidad y calidad del código de grandes aplicaciones frente a la que tendría en JavaScript plano.

TypeScript es el principal lenguaje utilizado para desarrollar la aplicación.

Programación orientada a objetos

TypeScript [7] es un lenguaje de programación orientado a objetos, por lo que se ha hecho un extensivo uso de este paradigma en el proyecto. La POO establece el concepto de objetos, que encapsulan datos y un código asociado en una única entidad.

HTML

HTML [8] es un lenguaje estándar de marcado con el que se definen los distintos elementos que componen una página web, así como sus atributos. Los navegadores web leen las etiquetas HTML de un documento y las transforman en elementos visuales con

los que el usuario puede interactuar. Se ayudan hojas de estilos CSS, que contienen reglas para darle diferente aspecto a los elementos representados.

El framework Angular amplia los elementos básicos de HTML permitiendo crear nuevos componentes con funcionalidad propia.

Document Object Model (DOM)

Define una interfaz para acceder y modificar unos objetos que representan los elementos que definen lenguajes de marcado como HTML o XML [9]. En JavaScript, y por tanto TypeScript, se utiliza para darle un comportamiento dinámico a la página web.

NodeJS

Según la página oficial: "Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome." [10]

Se usa en el proyecto de manera indirecta ya que está incluido en *Electron*.

NPM

Es el gestor de paquetes de Node.js [10]. Cuenta con un gran número de librerías JavaScript y permite gestionar e instalar dependencias con gran facilidad.

Electron

Electron [11] es una librería que permite embebir páginas web en una aplicación ejecutable en cualquier sistema operativo. Básicamente es una especie de navegador que muestra una página web como si de una aplicación nativa tratase. Combina Chromium y Node.js en un único entorno de ejecución. Cuenta con varias API para comunicarse con el sistema operativo y gestionar las funcionalidades propias de aplicaciones de escritorio.

Adicionalmente se ha utilizado la librería *electron-builder* para generar ejecutables e instaladores para Windows, Linux y Mac OS.

El código relacionado con Electron se encuentra en el archivo "main.ts" en la carpeta raíz del proyecto. Este código crea una ventana y abre en ella la aplicación de Angular, la cual ocupa el resto de los archivos del proyecto.

Angular

Angular [12] es un framework que permite crear páginas web modulares. Cuenta con una excelente y amplia documentación.

Angular gira en torno a una arquitectura modular basada en NgModules. Un módulo angular (o NgModule) es un contenedor de código generalmente relacionado con una funcionalidad o característica concreta. Puede contener componentes, servicios, y otros elementos de Angular. Un módulo está compuesto por los componentes, servicios y directivas que declara. Puede importar funcionalidades de otros módulos y exportar algunos de sus componentes.

Uno de los principales conceptos es el de componente, que agrupa el código correspondiente a una vista y su funcionalidad. Un componente está formado por un

template, una hoja de estilos CSS o SASS y una clase TypeScript. El *template* es el código HTML que define la estructura y apariencia del componente. Además de HTML estándar, se pueden incluir componentes y directivas de Angular, que extienden el código HTML básico. Esta extensión permite además enlazar datos con la clase del componente (*data binding*). La clase TypeScript almacena el código relacionado con la vista. Normalmente se encarga de obtener datos de un *servicio* y reaccionar a eventos causados por la interacción del usuario. El código de un componente debe tener la única finalidad de servir datos a la vista y reaccionar a eventos. Deben ser simples y de tamaño reducido.

Toda aplicación Angular cuenta con un componente raíz que utiliza a su vez otros componentes Angular, pudiéndose elaborar una jerarquía de vistas como la que se puede ver en la Ilustración 5.

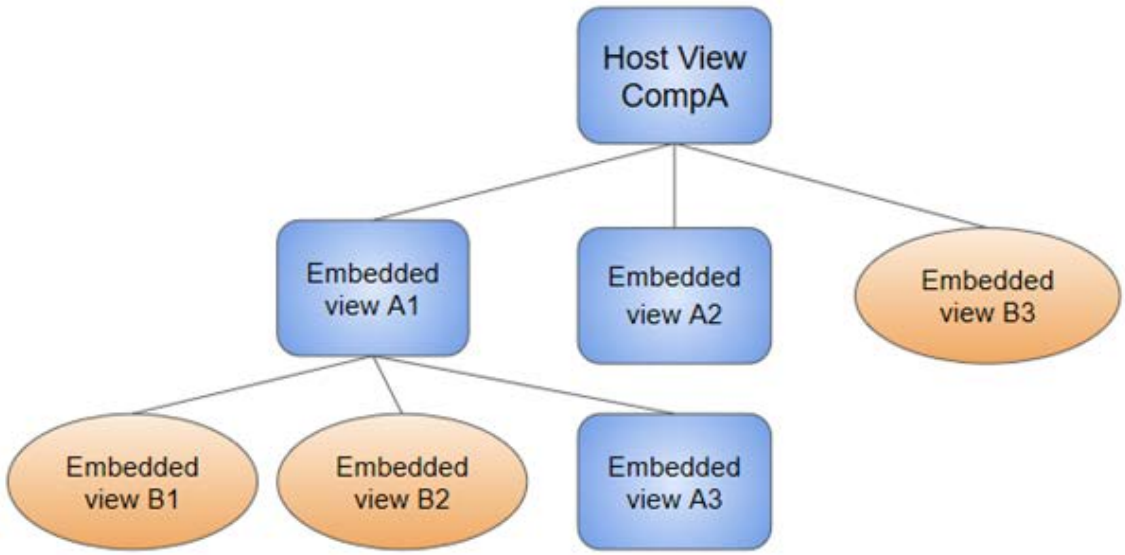

Ilustración 5 Jerarquía de vistas Angular (fuente: angular.io)

Otro de los principales conceptos es el de *proveedor de servicio*, o sencillamente *servicio*. Su principal cometido es separar código necesario para la aplicación del código de los componentes para conseguir una mayor modularidad. Normalmente hacen peticiones a APIs REST u otras fuentes para obtener datos del modelo (patrón Modelo-Vista-Controlador), aunque pueden contener cualquier otro tipo de código complejo que no deba estar en un componente. Pueden servirse de otros servicios.

Angular marca el desarrollo de todo el proyecto, ya se ha intentado que el código se adecúe lo máximo posible a la arquitectura y las guías de estilo propuestas en la página web del framework.

Feature Modules

"Módulos de características" en español. Se trata de una técnica recomendada por Angular para organizar el código de la aplicación. Un *feature module* es un *NgModule* agrupa componentes, servicios y otros elementos de Angular que están relacionados al proveer funcionalidad para una misma característica [13]. Mejora la organización lógica (clases) y física (archivos y directorios) del sistema.

Programación reactiva y patrón de diseño *observer*

El patrón de diseño *observer* permite el paso de mensajes mediante objetos entre un emisor y un suscriptor [14] [15]. El patrón gira entorno a un objeto llamado *Observable.* Existe un emisor, que crea un objeto *observable* al que enviará valores o mensajes nuevos cuando precise. Un suscriptor se suscribe a cambios en el *observable*, siendo avisado cuando el emisor publica nuevos valores.

El uso de *observables* es propicio en el desarrollo web para reaccionar tanto a eventos de interacción del usuario, como eventos de respuesta de procedimientos asíncronos. El concepto de programación reactiva hace referencia a una programación asincrónica usando este patrón. Este patrón es recomendado por Angular en numerosas ocasiones en su documentación.

En el proyecto se ha utilizado la librería *RxJS* para facilitar el uso de este patrón. La librería facilita la creación de observables. Ofrece una gran cantidad de operaciones sobre observables, como por ejemplo encadenar, mapear, modificar y filtrar.

```
setUpAutocomplete() {
    const elem = this.cityInput.nativeElement as HTMLInputElement;
    this.filteredCities = fromEvent(elem, 'input').pipe(
      map((e: KeyboardEvent) => e.target['value']),
      filter(value => value.length > 2),
      debounceTime(300),
      distinctUntilChanged(),
      map(cityName => this.filter(cityName))
    );
  }
```

Ilustración 6 Creación de un observable

La Ilustración 6 muestra una función que se utiliza para generar el objeto observable, del cual se obtendrán los valores para una lista de autocompletado en un cuadro de búsqueda. Se puede observar la creación de un observable a partir de un evento con la función *fromEvent*. Con la función *pipe* permite alterar el observable con una gran cantidad de funciones predefinidas o creadas por el desarrollador. Así, se puede ver como se mapea el valor del mensaje con la función *map*, se filtran mensajes con *filter*, etc.

Patrón modelo-vista-controlador

El patrón modelo-vista-controlador establece una separación entre los datos, la representación visual de los mismos, y la lógica de negocio. Facilita la reutilización de código y la separación de conceptos [16]. Angular facilita la aplicación de este patrón, ya que establece la separación entre la vista (*template* y estilos de un componente) del controlador (clase TypeScript del componente). Deja al desarrollador la tarea de la creación del modelo, que en el caso de este proyecto es simple y reducido.

Inyección de dependencias (Dependency injection)

Angular hace un extensivo uso del patrón de inyección de dependencias. Según este patrón, un objeto llamado inyector se encarga de instanciar distintos objetos aportándole mediante el constructor las dependencias que necesiten [17]. Angular se encarga de realizar esta tarea automáticamente en la creación de componentes y servicios.

API REST

Una API REST es una interfaz de un servicio web. Permite realizar cuatro operaciones (GET, POST, PUT y DELETE) sobre los recursos del servicio web. Los recursos se transportan en un formato estándar, normalmente JSON. Los mensajes se transmiten en forma de peticiones HTTP sin estado [18]. La aplicación que ocupa a este proyecto hace peticiones a API REST para obtener los datos que muestra en algunos de los módulos. El acceso a estas API se ha codificado con *proveedores de servicio* de Angular.

Material Design

Según la página oficial: "Material Design es un lenguaje visual sintetiza los principios clásicos de buen diseño con la innovación de la tecnología y la ciencia" [19]. Cuenta con una a amplia documentación en la que se explican las directrices y reglas a seguir para conseguir un buen diseño. Además, cuenta con librerías con componentes que pueden ser directamente usados en cualquier aplicación.

El diseño de la aplicación sigue los principios de Material Design.

Prototipado en papel

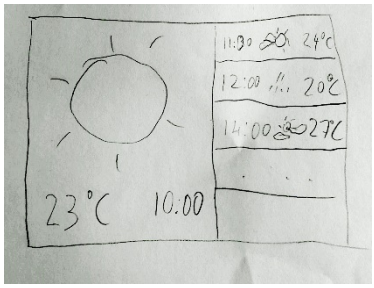

Ilustración 7 Prototipado en papel de la vista del módulo del tiempo

A la hora de diseñar las interfaces se ha utilizado la técnica conocida como prototipado en papel. Esto ha facilitado las decisiones de diseño a un coste mínimo. Dadas las numerosas disciplinas que implica el desarrollo de una aplicación, y el hecho de contar con solo una persona para el mismo, técnicas como el prototipado en papel, con una curva de aprendizaje mínima, hacen factible el desarrollo completo de la aplicación.

En la Ilustración 7 se muestra el diseño en papel, que entre otros tantos prototipos, se eligió finalmente para inspirar el diseño de la vista del módulo del tiempo.

Semver

Semver es el acrónimo de *semantic versioning*. Es un estándar para el número de las versiones de un sistema. Se describe en el RFC 2119. Según este estándar, una versión se compone de tres números X.Y.Z, que se incrementan solo en determinadas situaciones [20].

Está muy extendido entre los paquetes de NPM ya que facilita la gestión de dependencias. Se ha utilizado para dar nombre a las distintas versiones del sistema.

Git

Git es un sistema de control de versiones desarrollado inicialmente por Linus Torvalds. Permite llevar un registro histórico de la evolución de un conjunto de archivos. Git facilita la coordinación entre varias personas o equipos durante el desarrollo del software.

Uno de los objetivos del proyecto era aprender a usar esta herramienta, requerida en la mayoría de los puestos de trabajo del sector y crucial en los proyectos de código abierto. Para el desarrollo de la aplicación se creó un repositorio en la plataforma GitHub. El repositorio es privado por petición de HP.

Gitflow

Gitflow es un modelo que orienta en la creación y mantenimiento de las distintas ramas de desarrollo que puede haber en un repositorio de Git. Propuesto por Vincent Driessen, establece unas reglas para la creación de ramas y lanzamiento de versiones.

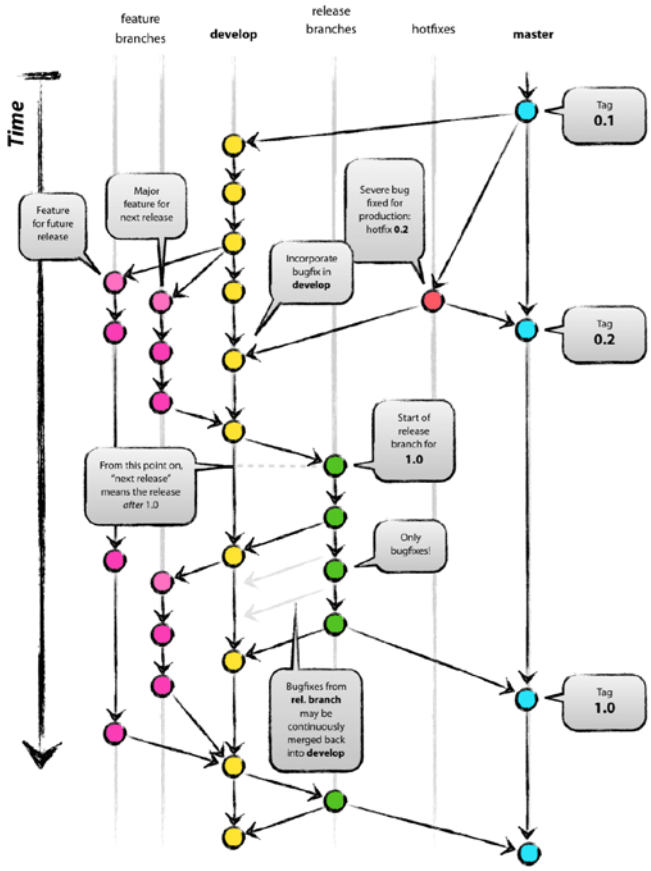

Ilustración 8 Diagrama de ramas gitflow (fuente: nvie.com)

En la Ilustración 8 es un diagrama que resume el flujo de trabajo de *gitflow*. Así, por ejemplo, se puede ver como en la rama master solo se vuelca código estable apto para producción, con normalmente un tag asociado.

Las ramas de característica, o *feature branches*, son creadas para el desarrollo de características concretas. Estas ramas surgen de la rama de desarrollo, y también se vuelcan a ella una vez concluido el desarrollo de la característica.

También establece las ramas *release* y *hotfix* para la corrección de errores.

Este modelo de desarrollo ha sido utilizado durante todo el proyecto. Al ser un repositorio privado con un solo desarrollador, no ha sumado un gran valor organizativo, pero ha servido para el aprendizaje de este popular modelo de trabajo.

Visual Studio Code

Visual Studio Code es un editor de texto personalizable con numerosas extensiones, hasta el punto de que podría considerarse un IDE. Se ha utilizado para la edición de código durante todo el proyecto.

Star UML

Star UML es un software gratuito para el diseño de diagramas UML.

Microsoft Visio

Programa para la creación de diagramas.

Trello

Se trata de una plataforma *online* para la gestión de proyectos. Principalmente se utilizan tarjetas virtuales que se agrupan en listas o columnas. Se ha utilizado para gestionar el flujo de trabajo durante los *sprint*.

Aspectos relevantes del desarrollo

El desarrollo del proyecto comienza la primera semana de febrero, tras la realización de los exámenes del primer semestre. El proyecto se divide en dos fases.

Una primera fase de **documentación** durante febrero, marzo y parte de abril. Se realiza una lectura e investigación acerca de la metodología de desarrollo a utilizar, las tecnologías más adecuadas para el proyecto, el estado del arte, y las técnicas y teorías que el alumno considera clave para el desarrollo del proyecto.

Una segunda fase de **diseño e implementación**, fuertemente solapadas dada la metodología utilizada. Esta fase se produce durante los meses de abril, mayo y junio de 2018.

A pesar de que cada uno de los seis *sprint* finalizó con un entregable de la aplicación, en este apartado solo se comenta el resultado final. El grueso de la aplicación es en realidad una página web. Gracias a la librería Electron, se dispone de un ejecutable basado en Chromium que abre la página web. Con la librería electron-builder es posible empaquetar el ejecutable y todos los archivos de la página web en un solo ejecutable, o generar instaladores. El resultado es siempre asemejable a una aplicación nativa, pero con la ventaja de contar con un único código fuente común.

Diseño visual de la aplicación

Todas las vistas se han diseñado de acuerdo con los principios de *Material Design*. Antes de codificar cualquier vista se realizaban sobre el papel uno varios bocetos con varias propuestas para la interfaz. Finalmente se elegía el diseño que más se adaptase a los principios de *Material Design*.

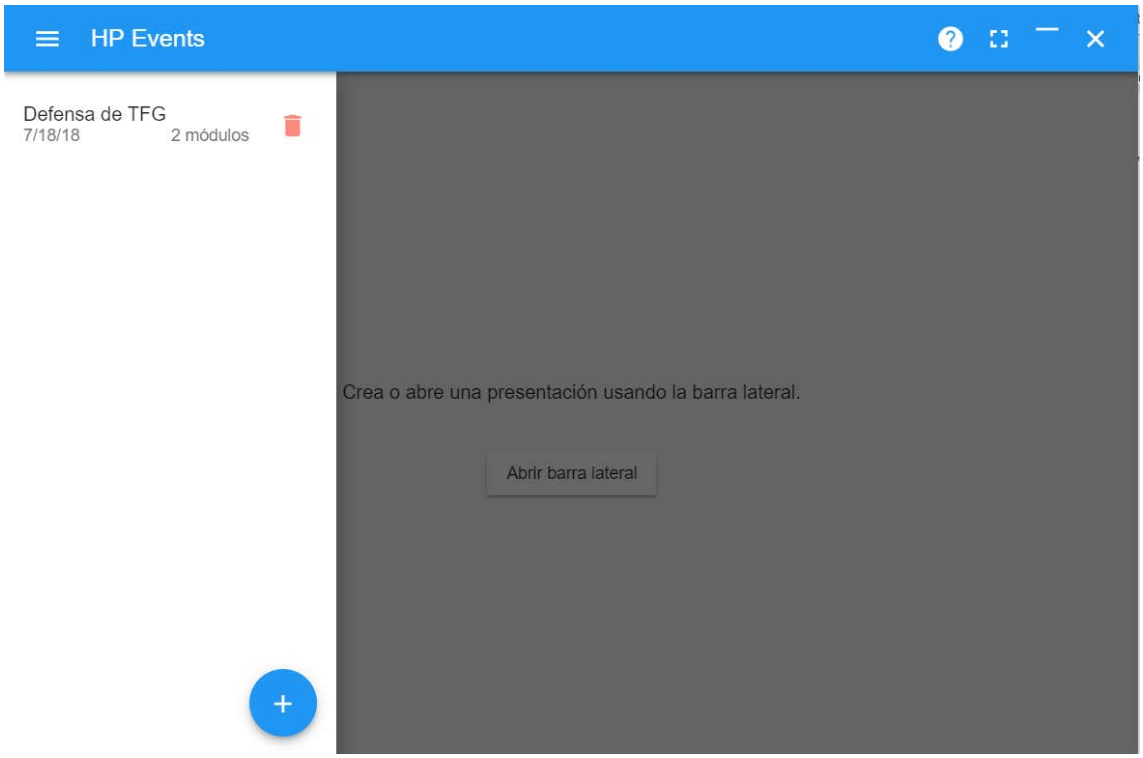

Ilustración 9 Pantalla inicial de la aplicación

La Ilustración 9 muestra la vista de la aplicación una vez abierta. El menú izquierdo muestra una lista de las presentaciones existentes, junto a un botón para crear una nueva. El espacio que esconde debajo está destinado a la visualización de las presentaciones una vez abiertas. En la barra superior se puede ver un botón para abrir y cerrar el menú con la lista de presentaciones y unos botones, que, de izquierda a derecha, son los siguientes:

- Acerca de: muestra un cuadro de diálogo con información acerca de la aplicación y la posibilidad de cambiar el idioma de la app. Se puede ver este cuadro de dialogo en la Ilustración 10.
- Pantalla completa: activa o desactiva el modo pantalla completa.
- Maximizar/minimizar.
- Cerrar.

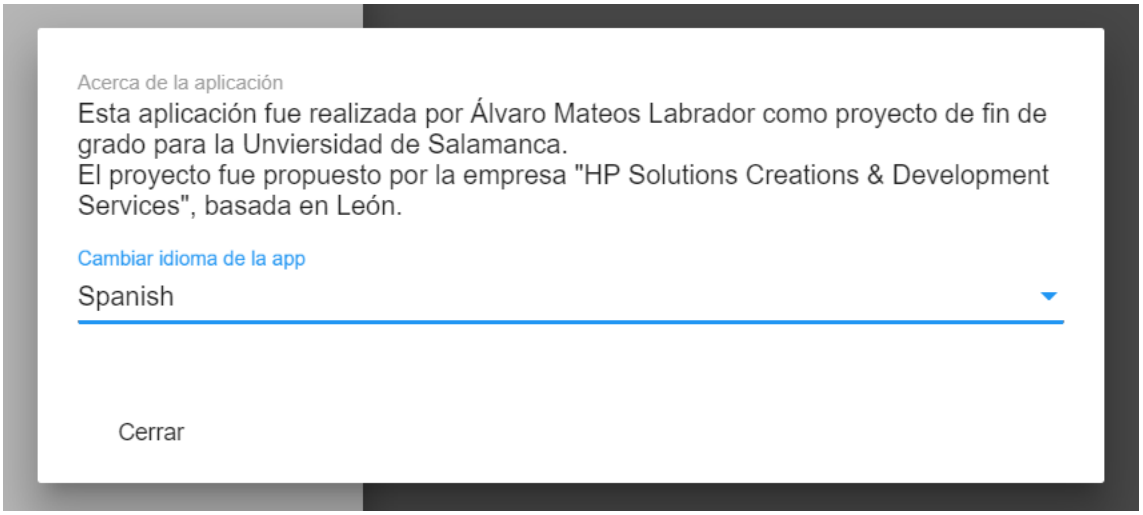

Ilustración 10 Cuadro de dialogo "Acerca de"

Al abrir una presentación, esta se mostrará en modo edición. Existen dos modos: edición y presentación. El primero está destinado a configurar los distintos módulos que pueden aparecer en el panel, y el segundo muestra los módulos informativos.

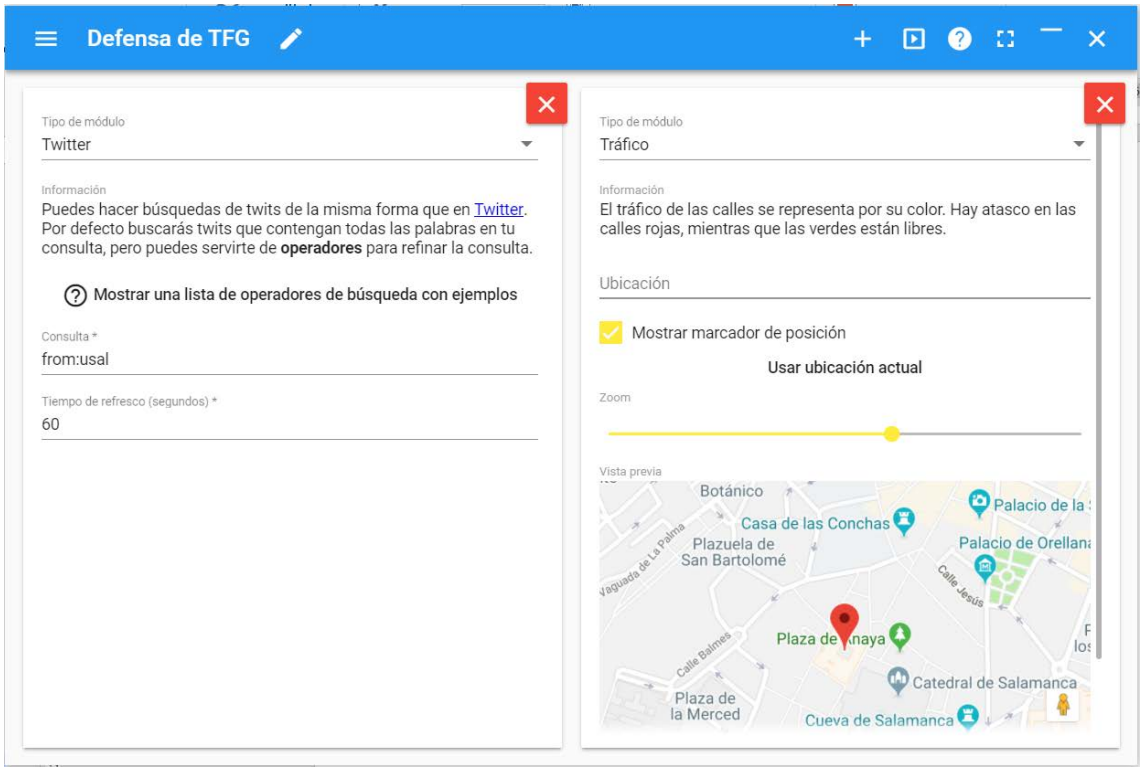

Ilustración 11 Modo edición

La Ilustración 11 muestra una presentación en modo edición, con un módulo configurado para mostrar twits de la cuenta "@usal", y otra que muestra el tráfico cercano a la universidad de Salamanca. También es posible configurar un reproductor de videos y un módulo que muestra la previsión del tiempo.

Han aparecido botones adicionales en la barra superior. Uno con forma de lápiz para cambiar el título de la presentación, otro con forma de aspa (+) para añadir módulos, y otro con forma de botón "*play*" para cambiar al modo presentación. Al hacer *click* en este último botón se cambiará al modo presentación, mostrado en la Ilustración 12.

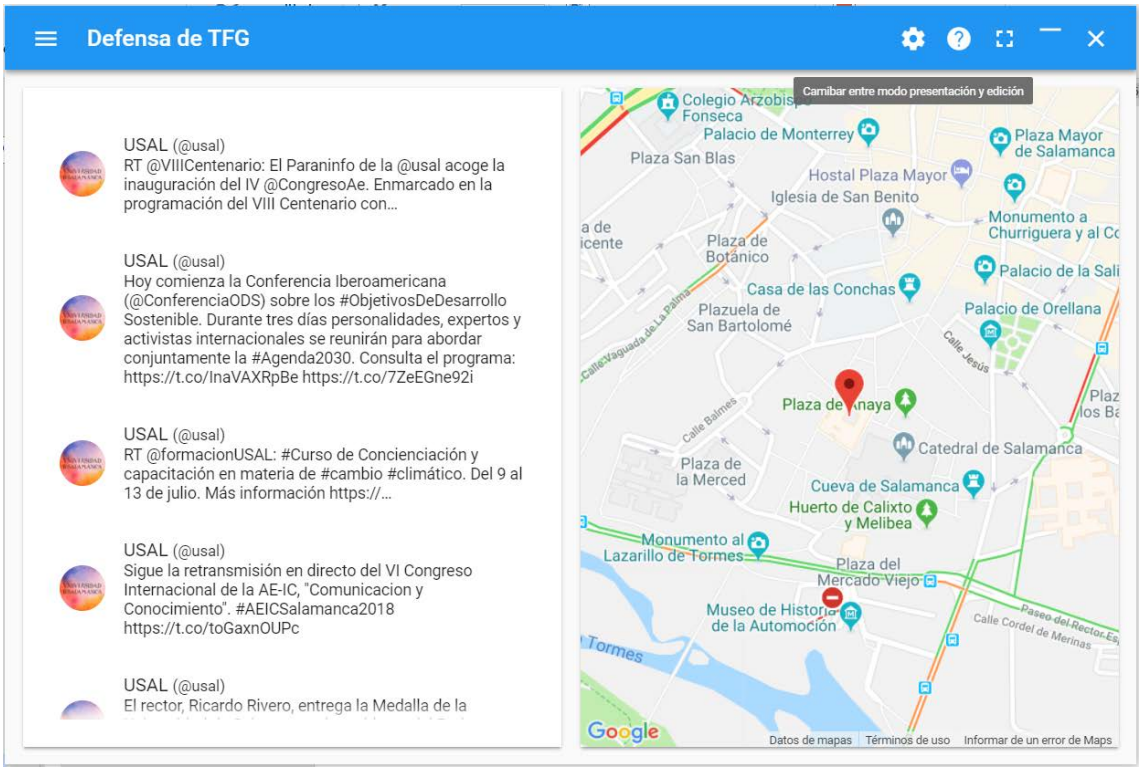

Ilustración 12 Modo presentación

A la izquierda se pueden ver distintos *twits* de acuerdo con la configuración, que se irán desplazando hacia abajo para dejar paso a nuevas entradas.

A la derecha se ve un mapa sobre la Universidad de Salamanca y sus alrededores. Las calles en color naranja y rojo tienen más tráfico que las verdes. Por desgracia la API de tráfico de Google no cuenta con información sobre el tráfico en todas las calles, a pesar de ser la más completa.

Las interfaces anteriores se rigen por los principios de *Material Design*. Se ha conseguido darle un aspecto moderno y consistente, idéntico independientemente del sistema operativo.

Modelo de datos del sistema

La aplicación es sencilla, por lo que el modelo de datos también lo es. El modelo de datos actúa como interfaz entre la aplicación y el almacenamiento secundario, en el que se persisten los datos. En el Anexo II se puede consultar el diagrama de clases del modelo, también mostrado en la Ilustración 13.

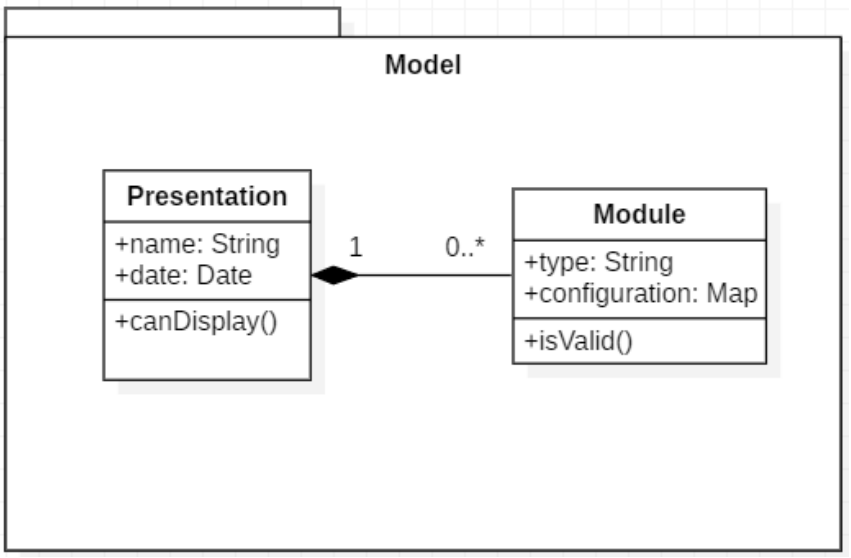

Ilustración 13 Diagrama de clases del modelo

La clase Presentation representaría un panel informativo destinado a un evento. Cuenta con un nombre y una fecha. Tendría varios módulos asociados. Un módulo puede ser de varios tipos (Twitter, video, etc) y tiene un mapa de configuración, en el que se almacenan distintos parámetros de configuración de cada módulo.

Arquitectura del sistema

La arquitectura del sistema está inspirada por la propuesta de Angular. Se trata de una arquitectura modular, en la que en cada módulo se agrupan ciertos elementos relacionados con una característica concreta. Los elementos dentro de estos módulos son componentes o servicios. Cada componente es una pequeña implementación del patrón Modelo Vista Controlador (MVC), tal como se ha explicado en el apartado "Angular", dentro de "Técnicas y herramientas".

La aplicación está centrada en la visualización de información. Así, los módulos estarán caracterizados por el tipo de información que representan. Existen 5 módulos, además del módulo app-raíz que engloba a los demás. Son los siguientes:

- Presentations: Contiene toda lógica y componentes asociado a la gestión y visualización de presentaciones. Usa el resto de los componentes para mostrar los submódulos (o paneles) que contiene.
- Twitter: Componentes y servicios para mostrar información de Twitter.
- Traffic: Componentes para mostrar información del tráfico en una zona.
- Weather: Componentes y servicios para mostrar la previsión del tiempo en una ubicación.
- Video-player: Componentes para la reproducción de vídeos.

Además de estos módulos, se ha hecho uso de muchos otros. En Angular las librerías externas de código también se agrupan en módulos, de los cuales se importan los componentes y servicios que se necesite. Las dependencias con módulos externos no se muestran en los diagramas ya que, dada la elevada reutilización de código, resulta muy complicado incluir todos los módulos externos utilizados en los diagramas y además dificultarían enormemente la lectura de los mismos.

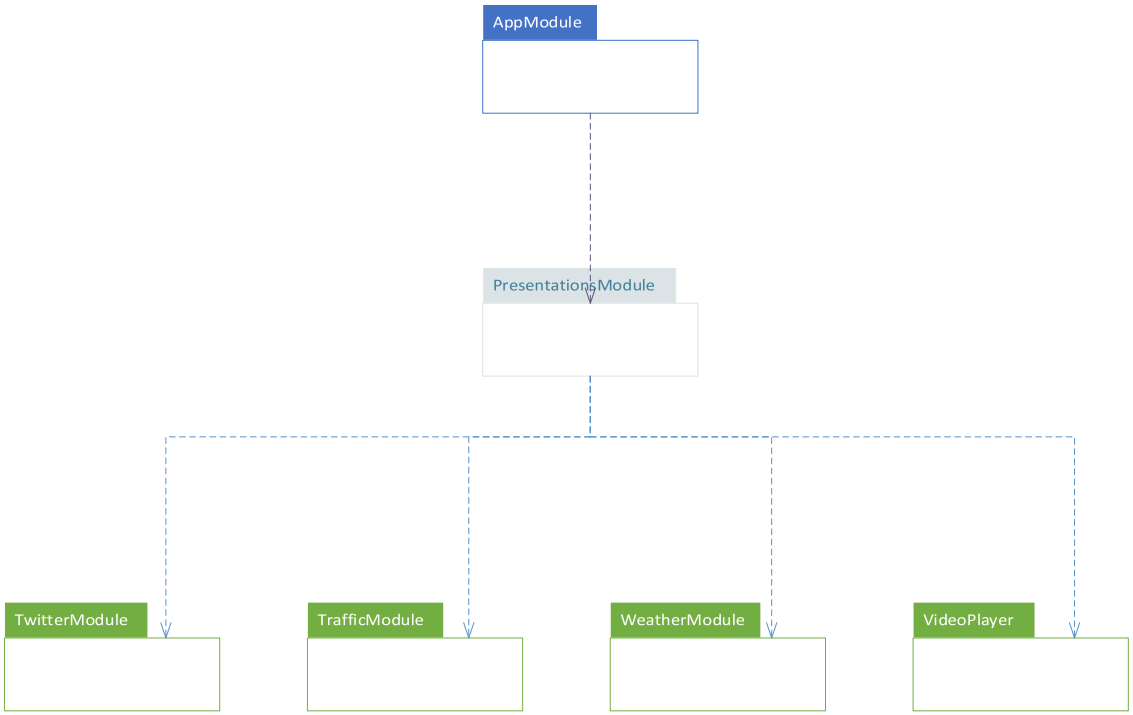

Ilustración 14 Diagrama de arquitectura

La Ilustración 14, presente en el ***Anexo III*** expresa la arquitectura modular del sistema. Las dependencias entre los módulos vienen determinadas únicamente por las dependencias entre los componentes (vistas). Así, se puede establecer una jerarquía de vistas que muestra estas dependencias. Se mostrará más adelante, pero antes se detallará el contenido de cada uno de estos módulos.

AppModule

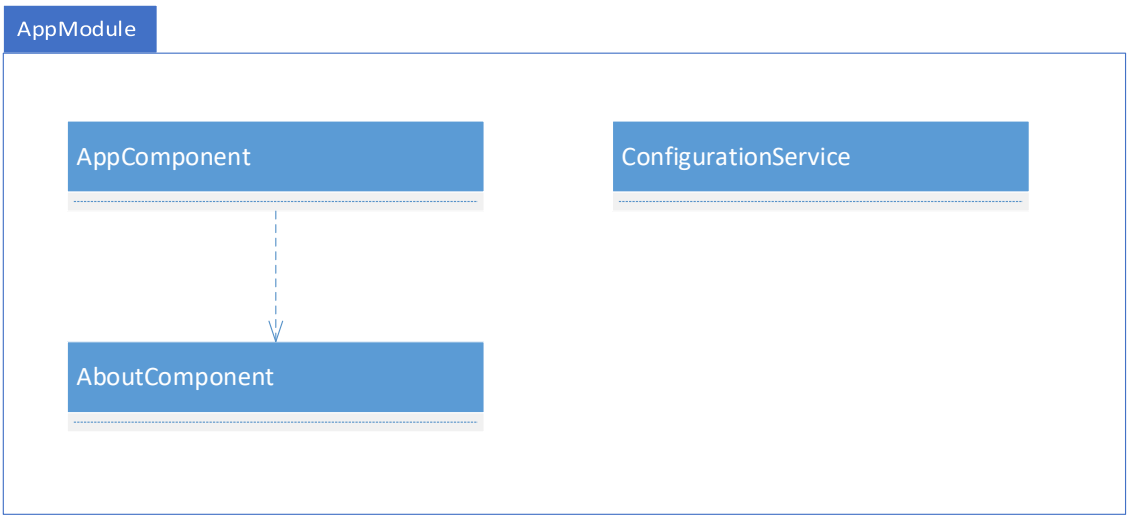

Ilustración 15 Módulo App (Anexo III)

La Ilustración 15 muestra módulo App, o AppModule, cuenta únicamente con dos componentes. AppComponent es el componente raíz y es el inyectado por angular en el *index* de la página web. Utiliza componentes y servicios de AppModule y de PresentationsModule.

PresentationsModule

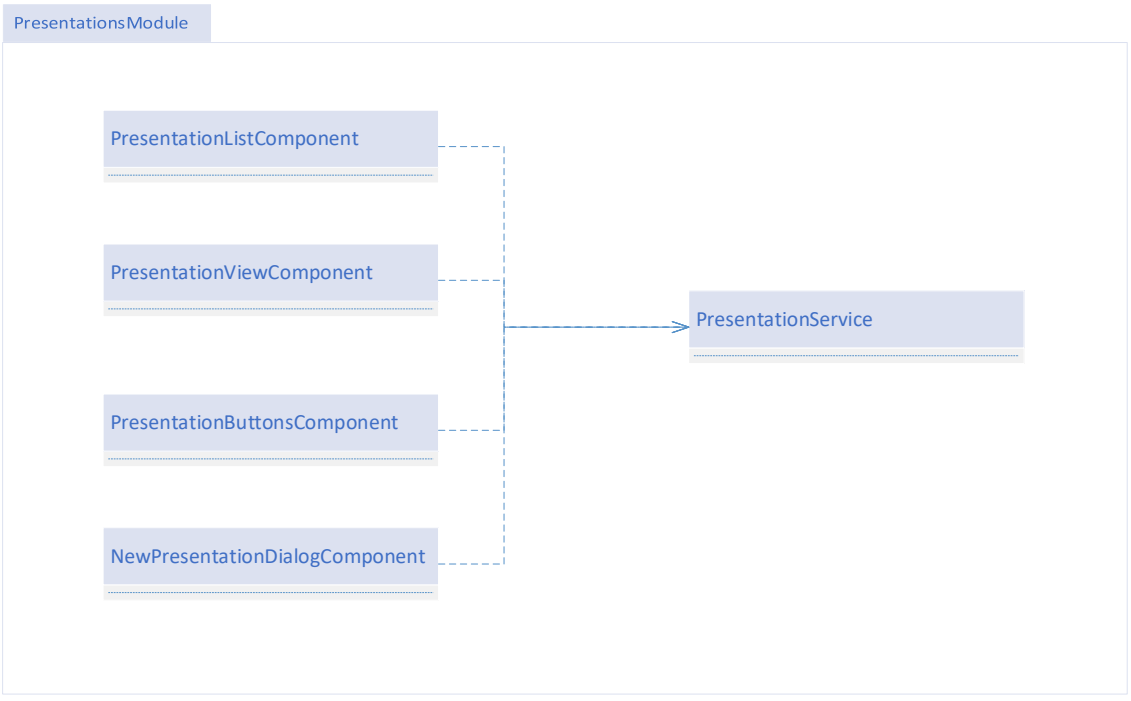

Ilustración 16 Módulo Presentations (Anexo III)

La Ilustración 16 muestra el contenido del módulo *Presentations*.

Los componentes del módulo *Presentations* utiliza algunos componentes de los módulos destinados a cada subpanel (Twitter, Traffic, Weather y VideoPlayer). Todos estos módulos presentan una estructura similar, exponiendo un componente para mostrar la información configurada por otro componente que hace de formulario de configuración.

TwitterModule

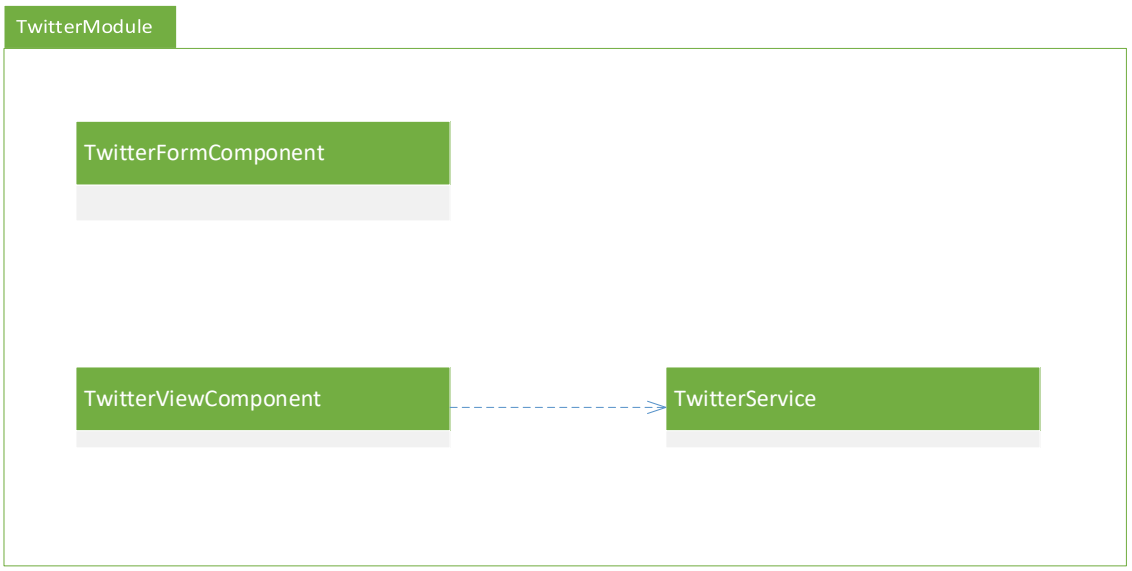

Ilustración 17 Módulo de Twitter (Anexo III)

En este módulo, presentado en la Ilustración 17, el componente formulario (TwitterFormComponent) modifica un objeto Module (ver Modelo de datos del sistema) actualizando la configuración según la entrada del usuario. En el resto de módulos de subpanel el funcionamiento es similar.

El componente TwitterView obtiene del servicio (TwitterService) unos *twits* que posteriormente muestra. El servicio se comunica directamente con la API de Twitter para obtener la información.

TrafficModule

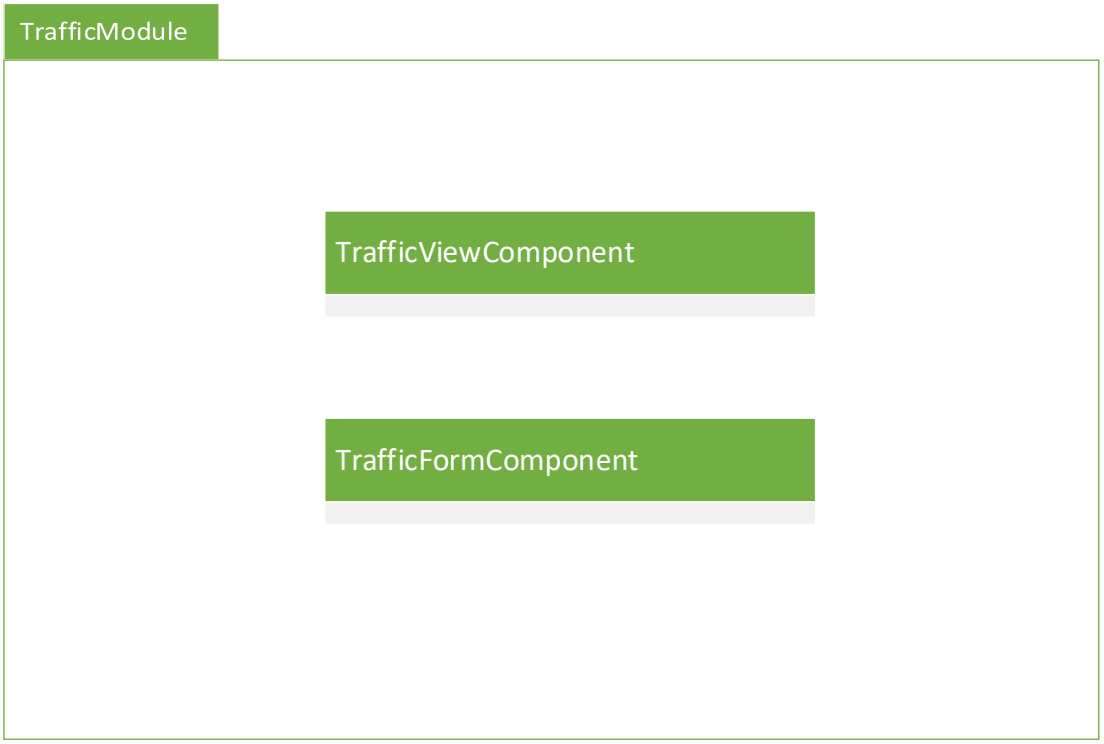

Ilustración 18 Módulo de Tráfico (Anexo III)

El módulo de tráfico (visto en la Ilustración 18) solo consta de dos componentes, gracias al uso de componentes externos para dibujar el mapa. La librería usada es "angular-google-maps".

WeatherModule

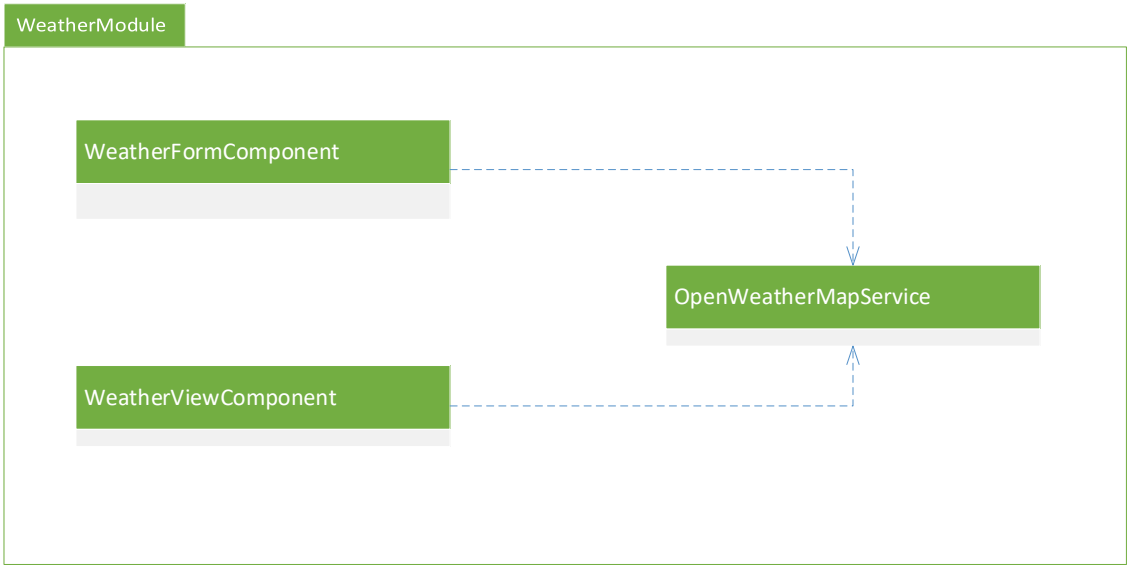

Ilustración 19 Módulo de previsión del tiempo (Anexo III)

La Ilustración 19 muestra el módulo de previsión del tiempo. Cuenta con una estructura análoga a los anteriores. El servicio *OpenWeatherMapService* se sirve de la API de http://openweathermap.org para obtener la previsión del tiempo.

VideoPlayerModule

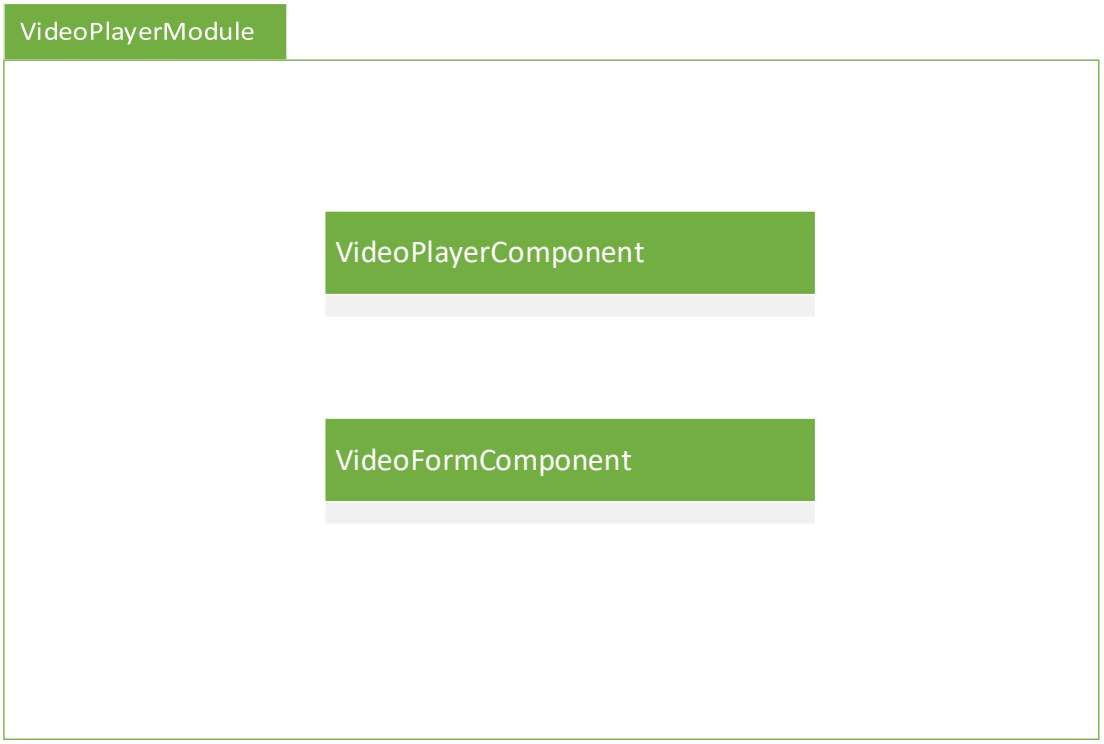

Ilustración 20 Módulo del reproductor de vídeos (Anexo III)

La Ilustración 20 muestra el módulo del reproductor de vídeos. De nuevo solo cuenta con dos componentes. Para reproducir los videos se ayuda de componentes de la librería "videogular", la cual actúa de interfaz con el elemento HTML5Video para reproducir los vídeos.

Jerarquía de vistas

Dado que Angular está centrada en el concepto de componente, y este agrupa el código relacionado con una vista, una forma más apropiada de expresar la arquitectura de la aplicación es con una jerarquía de vistas.

Visualmente, un componente se encarga de una porción de pantalla. El componente raíz sería el encargado de toda la pantalla. Sin embargo, normalmente es posible dividir una interfaz en pedazos con una función concreta. Estos pedazos de vista son extraídos a nuevos componentes, incluyendo su controlador. El componente raíz delega en diversos subcomponentes la responsabilidad sobre determinados pedazos de pantalla.

En la Ilustración 21 se muestra la jerarquía de vistas del sistema, incluida en el ***Anexo IV***.

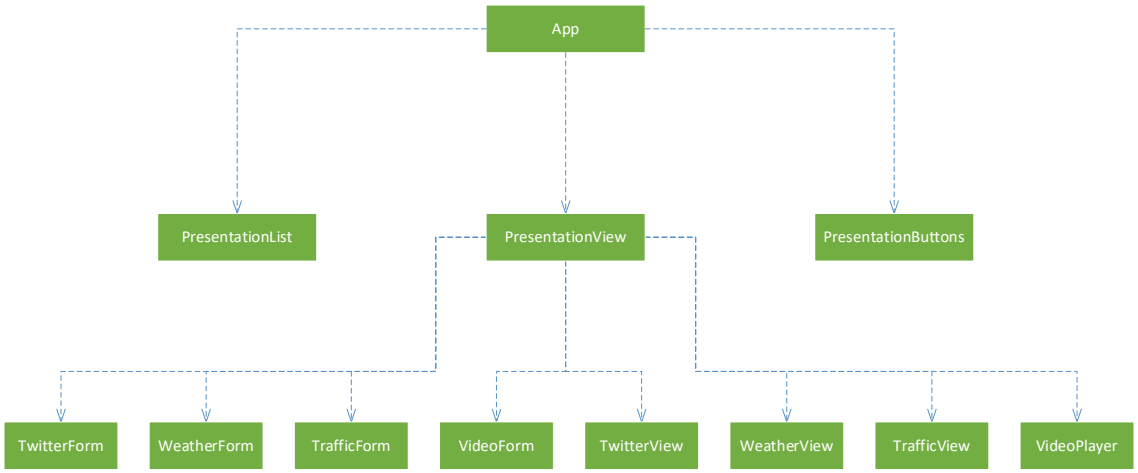

Ilustración 21 Jerarquía de vistas

Cabe destacar que, aunque las vistas de la jerarquía se corresponden con componentes de la aplicación, no aparecen en ella todos los componentes que se utilizan.

Por ejemplo, el componente PresentationList muestra la lista de presentaciones y el botón que se ve en el panel izquierdo de la aplicación (ver Ilustración 9 Pantalla inicial de la aplicación"). AppComponent se ayuda de otros componentes que complementan a PresentationList para conseguir las animaciones y efectos visuales que suceden.

Líneas de trabajo futuras

Aunque la aplicación cuenta con todas las funcionalidades inicialmente requeridas, existen algunas características que podrían ser incluidas en el futuro.

Dado que el producto ha sido elaborado con *Scrum*, estas funcionalidades tendrían que añadirse al *Product Backlog* en forma de nuevas *User Stories*.

De hecho, existe una *story* que no puedo ser implementada por falta de tiempo:

> Como usuario quiero poder visualizar las presentaciones en columnas

Una vez implementada se podrían visualizar los módulos en cuadrícula, tal como se hace actualmente, y por columnas.

En las reuniones con la empresa se consideró también la posibilidad de reproducir también vídeos de *YouTube* con el módulo de reproducción de vídeos.

Por parte del alumno existía el deseo de introducir un *tour* o guía interactiva al estilo *WalkMe*. Con él, la aplicación detectaría si se ejecuta por primera vez y ofrecería la posibilidad de tomar un *tour*. En estos *tours*, por medio de diálogos y sugerencias, la aplicación facilitaría al usuario el aprendizaje de todas sus funcionalidades y características. No pudo ser implementada por falta de tiempo, elaborándose en su lugar un pequeño manual de usuario.

Por último, también se ha considerado que en un futuro sería oportuno que la aplicación se comunicara con un *backend* para obtener datos de las distintas API con las que se comunica la aplicación. Actualmente la aplicación se comunica directamente con las API, por lo que las claves para autenticarse en dichas API están incluidas en el código fuente. La principal desventaja de contar con un *backend* para realizar estas consultas es su mantenimiento y disponibilidad: un problema en el servidor dejaría sin servicio a todos los usuarios de la aplicación.

Conclusión

En este apartado se analizará el grado de éxito con el que se han concluido los *Objetivos del trabajo*, clasificados en tres grupos.

Objetivos funcionales

Todos los *Objetivos funcionales* han sido satisfechos con éxito. La empresa HP ha quedado satisfecha y ha aplaudido las decisiones tomadas para conseguir implementar todas las funcionalidades. La consecución de estos objetivos se debe principalmente a su transformación en *User Stories*. Al estar vinculadas con los objetivos funcionales del sistema, se otorgó mayor prioridad a estas *stories* sobre otras de funcionalidades secundarias.

Objetivos no funcionales

No resulta tan sencillo determinar si los Objetivos no funcionales se han satisfecho por completo. Por ello, se analizarán uno a uno:

La aplicación debe ser una Single Page Application (web) desarrollada con el framework Angular y respetando la arquitectura de la misma.

Se puede concluir que la aplicación es en efecto una SPA desarrollada con Angular. Gracias a una extensa fase de documentación también se ha respetado en gran medida la arquitectura modular de Angular.

Experiencia de usuario agradable, con una interfaz que pueda ser usada por cualquier usuario sin la necesidad de leer un manual de instrucciones.

Resulta imposible definir de manera totalmente objetiva si se ha cumplido este objetivo. La empresa determinó que el objetivo ha sido cumplido, por lo tanto se dará por satisfecho.

Conseguir una estética moderna y acorde con los principios de diseño de HP.

Este objetivo se ha cumplido al aplicar los principios de diseño de *Material Design*, combinado con las reglas de la imagen de marca de HP.

El código de la aplicación debe estar sujeto al sistema de control de versiones Git.

Se ha usado Git durante todo el proyecto, siguiendo el método de trabajo *Gitflow*. Por tanto, este objetivo también se da por satisfecho.

Objetivos personales

La mayoría de estos objetivos se pueden resumir en aprender nuevas técnicas y habilidades. Dado que el alumno ha dedicado aproximadamente más de la mitad del tiempo a documentarse y aprender sobre las distintas teorías, técnicas y metodología propuestas, los objetivos personales se han cubierto en gran medida.

Entre los objetivos personales estaba desarrollar el proyecto en un entorno empresarial real, pudiendo llegar a conseguir una oferta de empleo firme. La empresa ha quedado

muy satisfecha con el desarrollo del proyecto, llegando a realizar una oferta de trabajo al alumno.

El único objetivo personal con el que el alumno ha quedado descontento se relaciona con la calidad. A pesar de considerar que la aplicación resultante logra unos niveles de calidad aceptables, no ha habido tiempo suficiente para aprender a trabajar con la técnica de desarrollo conocida como "*Test Driven Development*". Este método de trabajo consiste en centrar el desarrollo en las pruebas, ya sean unitarias, de integración, o de aceptación. Esta técnica es muy difícil de aplicar y dominar, especialmente para desarrolladores con poca experiencia. Dado el corto plazo de tiempo disponible para el desarrollo del proyecto, no ha sido posible aprender a usar esta técnica.

No obstante, se desconocía la existencia, que ha sido descubierta por serendipia durante la fase de documentación. Se puede considerar como positivo su hallazgo, y, aunque no haya sido posible dominar la técnica en el espacio temporal del proyecto, se aprenderá sobre ella en el futuro.

Glosario y siglas

- XP: Extreme Programming, metodología ágil de desarrollo.
- RUP: Rational Unified Process, modelo de proceso de desarrollo software.
- US: User Story.
- SP: Story Point.
- UCP: Use Case Points.
- HTML: HyperText Markup Language
- DOM: Document Object Model.
- CSS: Cascading Stylesheets.
- NPM: Node Package Manager.
- API: Application Programming Interface.

Bibliografía

[1] H. Kniberg, Scrum and XP from the trenches, C4Media, 2015.

[2] K. Schwaber y J. Sutherland, The Scrum Guide, ScrumInc, 2016.

[3] H. Kniberg y M. Skarin, Kanban and Scrum: making the most of both, C4Media, 2010.

[4] R. Nasiadek, «EL Passion,» [En línea]. Available: https://blog.elpassion.com/the-3-most-important-kanban-rules-b4b74314f69d. [Último acceso: 10 04 2018].

[5] Mozilla. [En línea]. Available: https://developer.mozilla.org.

[6] «Wikipedia: TypeScript,» [En línea]. Available: https://es.wikipedia.org/wiki/TypeScript.

[7] M. TypeScript. [En línea]. Available: https://www.typescriptlang.org/.

[8] Wikipedia, «Wikipedia: HTML,» [En línea]. Available: https://es.wikipedia.org/wiki/HTML.

[9] Wikipedia, «Wikipedia: DOM,» [En línea]. Available: https://es.wikipedia.org/wiki/Document_Object_Model.

[10] «NodeJS,» [En línea]. Available: https://nodejs.org.

[11] «Electronjs,» [En línea]. Available: https://electronjs.org/docs/tutorial/about.

[12] Angular, «Angular,» [En línea]. Available: https://angular.io/docs.

[13] «Angular: Feature modules,» [En línea]. Available: https://angular.io/guide/feature-modules.

[14] «Wikipedia: Observer,» [En línea]. Available: https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o).

[15] «Angular: Observables,» [En línea]. Available: https://angular.io/guide/observables.

[16] «Wikipedia: MVC,» [En línea]. Available: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador.

[17] «Angular: Dependendy Injection,» [En línea]. Available: https://angular.io/guide/dependency-injection-pattern.

[18] «Wikipedia: REST,» [En línea]. Available: https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional.

[19] «Material Design,» [En línea]. Available:
https://material.io/design/introduction/#principles.

[20] «Semver,» [En línea]. Available: https://semver.org/.