# TA Timetable:

# Aplicación multiplataforma para la gestión de la asignatura de Tendencias Actuales

Ruben Velez Requena

**Resumen**—Si se le preguntara a cualquier usuario base cuál ha sido uno de los avances tecnológicos más importantes de los últimos 5-10 años, la mayoría por no decir prácticamente todos, mencionarían las plataformas móviles. Durante estos últimos años estas tecnologías se han popularizado y han salido aplicaciones en muchísimos ámbitos de la sociedad, desde videojuegos hasta aplicaciones para pedir comida a domicilio. Aún y así en uno de los ámbitos que más ha costado que entrara, ha sido en el ámbito del e-Learning. Mediante este proyecto se intentará sustituir si es necesario una plataforma tan extendida como Moodle que actualmente se utiliza para una asignatura del Grado de Ingeniería Informática de la UAB, por una que sea multiplataforma y que pueda ser utilizada en cualquier tipo de dispositivo.

**Palabras clave**—Aplicación multiplataforma, e-Learning, Open-Source, IONIC, CORDOVA, HTML, Angular 2, Framework, Android, IOS, Web Responsive, MongoDB, NoSQL, Typescript, Webservice, PHP

**Abstract**—If we ask to the people, which was one of the most technological advancements of the last 5-10 years, almost all of them mention mobile platforms. During the last years these type of technology have been growing and it have been released a lot of applications in almost all kinds of. From videogames to applications to order food at home. Even though, one of the areas, where the mobile applications haven't entered, it is e-Learning. With that project it will try to substitute if it is necessary, the platform it was using on the past in one of the subjects of the Grade in Computer Science of the UAB, the Moodle for one it was multiplatform friendly. And can be watch it in any kind of device.

**Index Terms**— Multiplatform applications, e-Learning, Open-Source, IONIC, CORDOVA, HTML, Angular 2, Framework, Android, IOS, Responsive Web, MongoDB, NoSQL, Typescript, Webservice, PHP

## 1. Introducción

Tendencias Actuales es una asignatura del Grado en Ingeniería Informática de la "Universidad Autónoma de Barcelona". Esta asignatura trata de completar el curso yendo a exposiciones, actividades o ponencias sobre distintos temas relacionados con la informática y el Grado al que pertenece. Además de la posibilidad de hacer cursos relacionados con el Grado al que pertenece; como por ejemplo sobre algún lenguaje de programación. Después de ir a las ponencias o hacer los cursos hace falta hacer un documento respondiendo a las preguntas que la asignatura hace y resumir que se ha aprendido, o que ha podido extraer de cada uno de las actividades que se hizo.

Actualmente se utiliza Moodle como plataforma de unión entre el profesor y los alumnos, además del correo oficial de la universidad para mantener una comunicación activa entre el profesor y los alumnos. En este documento primero de todo estudiaremos si alguna plataforma existente se amolda a las necesidades y objetivos del proyecto, o si es necesario hacer una aplicación a medida des de cero.

## 2. Requisitos del Proyecto

Para saber si realmente alguna aplicación que ya exista en el mercado se amolda a las necesidades del proyecto, hace falta conocer cuáles son realmente los requisitos, restricciones y necesidades del proyecto.

Extraer correctamente esta información será vital, ya que dictaminará la naturaleza del proyecto de principio a fin.

## 2.1. Restricciones del Proyecto

Este proyecto no tiene realmente restricciones. Solamente es necesario poder llegar a todos los usuarios de la asignatura; alumnos y profesor. Esto significa que la tecnología a utilizar debe llegar a todos, utilizando cualquier plataforma. Y que ningún usuario (profesor y alumnos) se quede sin poderlo utilizarla, porque no tiene la tecnología correspondiente.

## 2.2. Requisitos del Proyecto

Tras la reunión con el cliente se extrajeron los requisitos del proyecto y lo que realmente el cliente buscaba con el proyecto. Para ordenar mejor y priorizar el proyecto dado el tiempo que se tenía, se categorizaron los requisitos para darles más peso a los más prioritarios y preparar el proyecto para futuros evolutivos, si fuera necesario. Ya fuera por el propio iniciador del proyecto o por otros actores futuros.

Tras la primera reunión estos fueron la lista de Requisitos:

1- La aplicación ha de ser accesible des de cualquier plataforma. Principalmente ha de poder ser visionada des de un dispositivo móvil y des de una web.
2- La aplicación ha de contener un calendario donde se sitúen las actividades hacer, y contengan todos los datos necesarios sobre dicha actividad (incluido Google Maps integrado)
3- La aplicación ha de contener dos tipos de usuarios, administrador y usuarios base.
4- La aplicación ha de contener una barra de progreso para llevar un control de las actividades y cursos hechos por cada estudiante en cada momento.
5- Los estudiantes han de poder apuntar-se y puntuar la actividad tras completarla.
6- La aplicación ha de centralizar al máximo todo lo que necesita la asignatura para su correcto funcionamiento

| Id Requisito | Prioridad 1 | Prioridad 2 | Prioridad 3 |
|---|---|---|---|
| 1 | Si | | |
| 2 | | Si | |
| 3 | | Si | |
| 4 | | Si | |
| 5 | | Si | |
| 6 | | | Si |

[Figura 1] Tabla Requisitos principales – Prioridad de los Requisitos

El Requisito de prioridad 1, sería la piedra angular del proyecto, obligatorio y por el que principalmente se haría el propio proyecto. Ya que actualmente, Moodle que era la plataforma que se utilizaba junto al email, no se podía acceder correctamente des de un dispositivo móvil.

Los Requisitos de Prioridad 2 son aquellos que para un buen funcionamiento del proyecto deberían ser completados y/o planteados, o donde la aplicación debía ser preparada para ser completados en el futuro.

El Requisito de Prioridad 3 sería un requisito donde como mínimo el proyecto debería ser preparado para poder cumplirlo en el futuro

Más adelante en siguientes reuniones se extrajeron más características deseadas por el cliente pero fueron estas las que se obtuvieron al inicio y las que marcaron el proyecto y la definición de él.

## 2.3. Comparativa de las soluciones del mercado

Existen otras alternativas a Moodle, como son Sensei de WooCommerce, LearnDash o Chamilo [3] [4] [5]. A continuación se mostrará una tabla comparativa con las ventajas y las desventajas de cada solución (en la siguiente página se puede ver la tabla comparativa).

Realmente en el mercado no se ajusta ninguna solución existente a los requisitos iniciales del proyecto. Por lo tanto se decidió por la solución propia.

La causa principal por la cual se decidió por la solución a medida fue que la gran mayoría de las soluciones de e-Learning no están preparadas para dispositivos móviles. Siendo la prioridad número uno del proyecto, o las que estaban más preparadas eran demasiado simples y poco configurables.

| Nombre Solución | Ventajas | Inconvenientes |
|---|---|---|
| moodle | - Plataforma más extendida. Por lo tanto tiene una muy amplia comunidad de usuarios y desarrolladores<br><br>-Plataforma general, que se puede hacer servir para prácticamente cualquier tipo de proyecto e-Learning | -Necesita una mayor configuración inicial, para muchas de las funcionalidades que ofrece<br><br>-Poco atractivo<br><br>-No tiene ninguna opción para configurarlo para móviles |
| WOO COMMERCE | -Solución construida mediante WordPress, fácil de desarrollar, modificar i configurar | -No se ajusta a lo que realmente se busca con el proyecto, y por lo tanto haría falta hacerlo prácticamente des de cero |
| Chamilo | -Tiene funcionalidades sociales, difíciles de encontrar en una plataforma de e-Learning<br><br>- Totalmente Open Source<br><br>-Configurable para mobiles | -Demasiado simple<br><br>-Solución no ajustada para educación avanzada (cualificada) |
| Solución própia | -Capacidad de hacer lo que el cliente realmente quiere y necesita. | -Alto precio inicial, al tener que desarrollarlo des de cero<br><br>-Necesidad de mucho más tiempo de desarrollo |

[Figura 2] Tabla ventajas e inconvenientes de las distintas soluciones del mercado

## 3. Estado del Arte

De este estudio se sacaron varias soluciones interesantes y se compararon con la que se utilizaba actualmente, además de una solución más a medida, creada des de cero.
Tras el estudio de mercado se extrajo que no existía ninguna solución existente que se amoldara a lo que el cliente necesitaba.

### 3.1. Solución

Finalmente la solución escogida fue una solución a medida, multiplataforma. Por lo tanto la tecnología escogida debía de ser ajustable tanto para móviles como web.

La tecnología escogida fue IONIC 2 [1], un Framework para la creación de aplicaciones móviles multiplataforma con la capacidad de construir una solución web en html5 y JavaScript (Angular2 JS).

#### 3.1.1. IONIC 2

[1] IONIC 2 es un Framework, con el que se pueden construir soluciones de apps móviles (Android y IOS) como aplicaciones webs.

IONIC 2 utiliza tanto Cordova [2] como Angular 2 [3], para poder construir dichas aplicaciones. La posibilidad de poder hacer aplicaciones tanto móviles como webs es porque utiliza HTML5 y JavaScript (Angular 2 JS) para poder hacerlo, siendo estas tecnologías, soluciones muy estandarizadas, que en la actualidad se pueden utilizar en cualquier plataforma.


[Figura 3] IONIC 2 Framework

#### 3.1.2. CORDOVA

[2] Framework creado por Apache, por debajo de IONIC 2 utilizado para compilar la aplicación de HTML5 y JavaScript. Transformando el código creado por los dos lenguajes anteriores a un lenguaje pseudo-móvil, que los Sistemas Operativos móviles (Android e IOS) entienden.


[Figura 4] Cordova Framework

#### 3.1.3. Angular 2

Framework de JavaScript creado por Google, donde orienta el JavaScript en un modelo basado en Componentes independientes de Vistas y Controladores de las vistas, además de Factorías/Servicios. Es en realidad un pseudo Modelo – Vista – Controlador, con un buen rendimiento y preparado para ser utilizado en plataformas webs y móviles.

[Figura 5] Angular 2 Framework

### 3.1.4. Otras tecnologías

A parte de las tecnologías expuestas, se ha utilizado Firebase, un conjunto de servicios que ofrece Google actualmente, de una manera más o menos sencilla para ayudar a los desarrolladores con servicios comunes que tienen prácticamente todas las aplicaciones móviles y webs en la actualidad. Como autentificación de usuarios, notificaciones o gestión de bases de datos.

[Figura 8] Firebase

También se utilizó código PHP para los Webservices que hizo falta hacer, durante el proceso de desarrollo. En futuros apartados se explicará claramente que son y porque se tuvieron que hacer, cuando en realidad no estaban planificados.

### 3.1.5. Idea general del proyecto

Con las tecnologías explicadas anteriormente se podrían construir tanto aplicaciones webs como móviles. El mayor problema del proyecto era las fechas reducidas que tenía y la imposibilidad de poder hacer por separado todas las aplicaciones (Web Responsive, Android e IOS). Había que priorizar que hacer en el proyecto, y que fuera lo más significativo posible.

## 4. Objetivos del Proyecto

A causa de lo ajustado del proyecto para la envergadura que podía llegar a tener. Hacía falta priorizar que hacer obligatoriamente, que preparar para futuros evolutivos si fuera necesario, y que se dejaría de hacer.

Para poder hacer esto, mediante los requisitos se extrajeron los objetivos del proyecto:

1. Utilizar la aplicación des de cualquier dispositivo (**Prioritario**)
2. Mediante un código poder construir una aplicación Web, otra Android y otra IOS. (**Prioritario**)
3. Mejorar el seguimiento del profesor durante todo el año lectivo. I saber el estado de los estudiantes de una manera lo más aproximada posible a cada instante de tiempo, con cada uno de los estudiantes.
4. Los estudiantes deben poder llevar un control de la asignatura de una manera sencilla.
5. Los estudiantes puedan saber cómo llegar a cada una de las actividades propuestas por la asignatura, de manera sencilla. (**Prioritario**)
6. Centralizar lo más posible toda la información y actividad relacionada con la asignatura

| Objetivos | Prioritario |
|---|---|
| Multiplataforma | Si |
| Código único | Si |
| Mejora del seguimiento por el profesor | |
| Control de los estudiantes sobre la asignatura | |
| Los estudiantes han de saber cómo llegar a cada una de las actividades | Si |
| Centralizar la actividad de la asignatura | |

[Figura 6] Tabla de objetivos

### 4.1. Un Código para gobernarlos a todos

El requisito número uno del proyecto era poder hacer una aplicación que fuera capaz de ejecutar-se en múltiples tipos de dispositivos. La opción número uno sería crear una aplicación web Responsive que pudiera ser visionada en los navegadores tanto de ordenadores como de móviles. El mayor problema de hacer solamente una aplicación web, es que muchas veces, no es muy amigable para un usuario utilizar durante un tiempo razonable una web en un dispositivo móvil. Por lo tanto el objetivo principal del proyecto pasaría a ser construir una solución web al ser la plataforma principal en la que se desarrolla IONIC 2, i que mediante pocos cambios en la codificación o solamente cambios visuales se pudiera generar también aplicaciones móviles.

Para poder probar que esto es posible, sería necesario poder como mínimo, probar de construir la solución web Responsive y una aplicación móvil nativa. Se escogió Android, a causa de las restricciones existentes a la hora de desarrollar para IOS, como necesitar un ordenador con Sistema Operativo IOS para compilar la aplicación con XCode, siendo el compilador que utilizan los Sistemas Operativos IOS. A diferencia de los que utilizan los Sistemas Operativos Android, donde actualmente es Java, siendo accesible este último des de cualquier ordenador con Sistema Operativo Windows o IOS.

### 4.2. División del proyecto en sub-proyectos

Al poder mediante el mismo código principal levantar distintas aplicaciones para distintas plataformas, se dividió el proyecto en sub-proyectos:

[Figura 7] Tabla de sub-proyectos

| Sub-Proyectos | Proyecto crítico | Proyecto prioritario | Proyecto opcional | Orden prioridades |
|---|---|---|---|---|
| Aplicación web "Responsive" | Si | No | No | 1 |
| Aplicación Android | No | Si | No | 2 |
| Aplicación IOS | No | No | Si | 3 |

## 5. Metodología

Para realizar este proyecto ha sido necesario familiarizarse con todas las tecnologías que se han utilizado durante el proyecto. IONIC 2 [1], Angular 2 [2] y Cordova [3].

Para la elección de la metodología de trabajo se ha tenido en cuenta que se trata de un proyecto individual y que los requisitos recogidos inicialmente pueden estar sujetos a variaciones durante el desarrollo del proyecto. Por lo tanto, se decidió que el método de desarrollo debía alejarse de los ciclos de vida tradicionales tales como el modelo secuencial, utilizados para proyectos de gran envergadura con requisitos muy bien definidos. Para este proyecto se utilizó Incremental Build Model como metodología de trabajo.

### 5.1. Incremental Build Model

Metodología donde el diseño, la implementación y el test se hacen de manera incremental, añadiendo en cada uno de las entregas, algo más de funcionalidades, hasta que el producto se finaliza. El proyecto se define como finalizado cuando satisface todo los requisitos y necesidades, cuando el cliente está de acuerdo con el resultado final del proyecto o cuando se llega a la fecha pactada al inicio del proyecto entre el cliente y los productores como final del proyecto.

Los diferentes paquetes (entregas) incrementales serán los paquetes (entregas) definidos en la planificación inicial del proyecto. Donde normalmente serían una semana después de la entrega del informe (de seguimiento o final) correspondiente a cada entrega con el tutor.

## 6. Planificación

En este punto se explicará cómo se planificó el proyecto al inicio de él, y más adelante se podrá comparar, como afectó esta planificación inicial a los problemas y los cambios que surgieron durante el desarrollo.

Para poder completar el proyecto se dividió este en hitos o entregas. Estos paquetes se relacionaron con informes de seguimiento que se tenían que entregar para que fuera más sencillo el entendimiento de estas entregas. En la siguiente tabla se puede ver la granularidad y la división que se hizo de las funcionalidades que había que hacer del proyecto en tareas:

| Tarea | Resumen |
|---|---|
| **Entrega 1** | |
| Calendario Actividades | Calendario donde se sitúan les actividades de la asignatura |
| Actividades en detalle | Al clicar en una actividad en el calendario, se muestra la información relacionado con la actividad |
| Lista Cursos | A parte de las actividades normales, los estudiantes pueden hacer cursos online para completar algunas horas |
| Lista Actividades | Lista de todas las actividades que salen |
| Diseño tratamiento de los datos | Como utilizará la aplicación los datos, y como serán distribuidos |
| Gestor de usuarios | Configuración del servicio de Firebase |
| **Entrega 2** | |
| Crear, editar y borrar cursos | Formulario de tratamiento de cursos |
| Navegación lateral | Navegación lateral multiplataforma preparado para cualquier tipo de dispositivo |
| Lista estudiantes | El profesor ha de poder ver cómo van los estudiantes de la asignatura |
| Crear, editar y borrar actividades | Formulario de tratamiento de actividades |
| Corrección bugs | |

| **Entrega 3** | |
|---|---|
| Formulario proponer cursos | |
| Formulario proponer | |
| Barras de progreso | Para terminar el curso se ha de completar un número de horas, mediante estas barras se conoce el estado del curso |
| Proponer actividades y cursos | Los estudiantes tiene la capacidad de proponer al profesor tanto cursos como actividades |
| Google Maps | Añadir Google Maps a las actividades para saber dónde exactamente son dichas actividades |
| Corrección de bugs | |

[Figura 9] Tabla Tareas Planificadas

### 6.1. Diseño del tratamiento de Datos

Al ser un proyecto orientado a múltiples tipos de dispositivos, se buscó una tecnología con un rendimiento lo más alto posible. Por lo tanto el paradigma a hacer servir, sería NoSQL, organizado por documentos y gestionado mediante JSONs. Siendo los tipos de ficheros más eficientes, si nos referimos al rendimiento. [7]

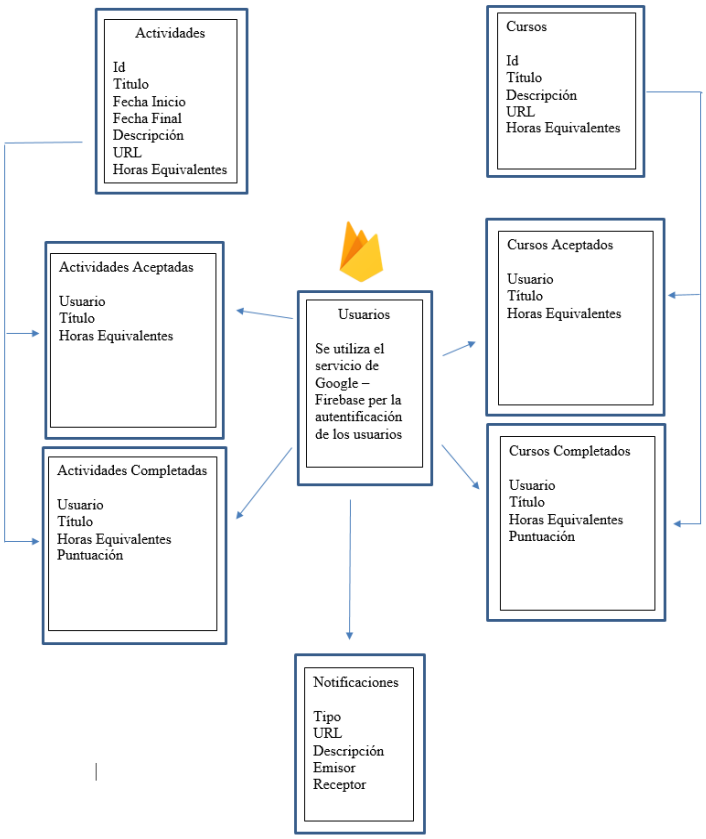
Al ser NoSQL, no contaría con una relación entre tablas convencional, ni existirían dichas tablas. Los datos estarían divididos en documentos, que serían los diferentes JSON que la aplicación utilizaría.

La parte positiva de utilizar NoSQL es que no hace falta integrar ninguna tecnología de Base de Datos relacionales, principalmente al ser engorrosas para tecnologías móviles, además de que cada documento (o JSON), tendría como mínimo el 75% de los datos para poder rellenar una pantalla. Al utilizar NoSQL no todo son puntos positivos, y uno de los grandes problemas de este tipo de tratamientos de datos es la repetición de la misma información en diferentes documentos (JSONs), generando más contenido y capacidad en una misma aplicación que tuviera una SQL relacional.

Al utilizar NoSQL no era necesario integrar una distribución de base de datos. Esto provocó también, que no tuviera la seguridad que llevan estas distribuciones, como la creación de las copias de seguridad, o guardar copias cada "x" tiempo por si ocurre algún problema con los datos. Provocando que durante el desarrollo, se tuviera que reiniciar los datos bastantes veces por ello.

A causa de las fechas y el tiempo que se tiene para completar el proyector una de las posibles mejoras que se propondrán más adelante, será añadir MongoDB como gestor, para crear copias de seguridad de los datos, y restablecer los datos por posibles inconsistencias en los datos.

### 6.2. Dibujo de la Estructura de los Datos




[Figura 10] Diseño Datos Aplicación

## 7. Desarrollo

En este apartado se explicará cómo fue el desarrollo del proyecto. Se explicaran como se hicieron cada una de las tareas en mayor o menor medida y que problemas surgieron durante el desarrollo.

### 7.1. El Código que los domina a todos

El punto principal del proyecto era crear un código que se pudiera utilizar para generar tanto una aplicación Web Responsive, como aplicaciones móviles en Sistemas Operativos Android e IOS. Por lo tanto lo primero que se hizo fue mediante las tecnologías

explicadas al inicio del documento [1] [2] [3], conseguir crear una aplicación Web Responsive y otra Android.

Para poder hacer esto se tuvo que configurar el proyecto y el entorno de desarrollo con las herramientas necesarias para generar dichas aplicaciones. En el caso de Android hubo que instalar los kits de desarrollo necesarios e instalar y configurar Android en el entorno de desarrollo.

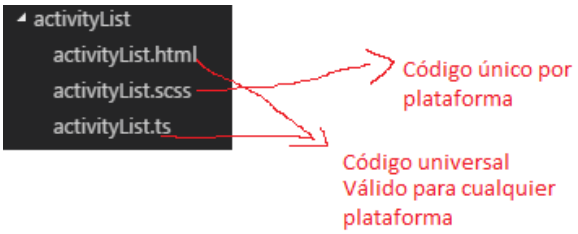
Finalmente se consiguió que mediante un simple comando se generara la aplicación Android y la aplicación Web Responsive

```
C:\Desarrollo\Cordova_ionic\myApp>ionic build browser
^C¿Desea terminar el trabajo por lotes (S/N)? S

C:\Desarrollo\Cordova_ionic\myApp>ionic build android
^C¿Desea terminar el trabajo por lotes (S/N)? SA
```

[Figura 11] Comando para generar las aplicaciones

Esto es posible ya que, IONIC 2 trabaja con un paradigma orientado a componentes. Donde cada componente está dividido en los ficheros que podemos ver en la siguiente imagen




El fichero html es la vista y el fichero ts es el "controlador". Cambiando el código del fichero scss configurándolo para el tamaño de pantalla en el que lo queremos mostrar estaríamos personalizando la aplicación. La extensión scss es una psuedo css utilizada por los Frameworks actuales que utilizan Typescript.

Sin tener nada más que hacer. La única diferencia entre las dos aplicaciones seria la CSS al tener distintos tamaños de pantalla. Pero el código funcional de la aplicación seria exactamente el mismo.

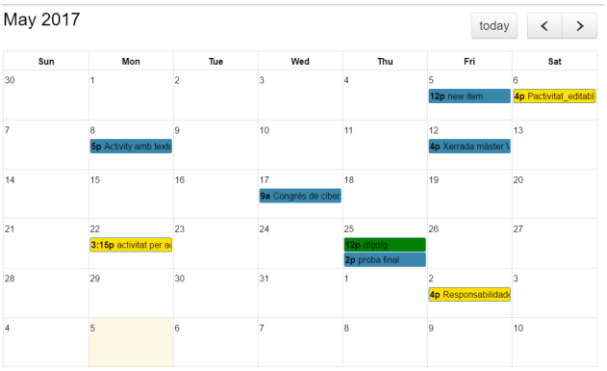
En los apéndices del documento se pueden ver muestras de la misma aplicación en cada una de las distintas plataformas creadas. Android y Web Responsive.

## 7.2. Características de la aplicación

En este punto se va a exponer las características de la aplicación más importantes y como se hizo para diseñarlo, construirlo e implementarlo. Y si surgieron problemas, como se resolvieron.

### 7.2.1. Actividades

Durante la primera entrega se construyeron el calendario de actividades y las actividades en detalle cuando clicabas a una actividad, además de la lista de las actividades. Uno de los requisitos que tenía el proyecto era un calendario visual para que fuera fácil de entender cuando se realizaran las actividades de la asignatura, y cuando se clicara en una de ellas se pudiera ver en detalle cada una de estas actividades.




[Figura 11] Calendario de Actividades. TA Timetable

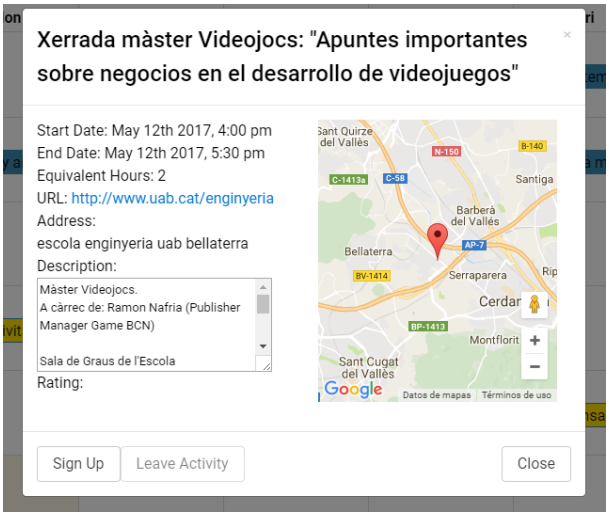
Tras uno de los primeros feedbacks por el cliente, este propuso que era difícil por parte de los usuarios, saber a cuáles actividades el propio usuario había completado, a cuáles se había apuntado pero no había completado y cuáles tenía la opción de hacer aún y no se había apuntado. Por lo tanto se decidió tener tres colores distintos dependiendo del estado de la actividad sobre el usuario que estuviera activo en la aplicación.

-Azul: Actividades Disponibles

-Amarillo: Actividades en las que el usuario se había apuntado pero no había completado

-Verde: Actividades en las que el usuario se había apuntado y las había completado

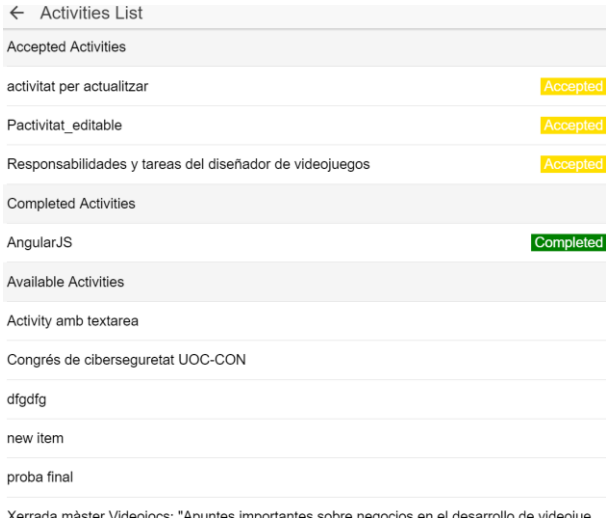
El tener como objetivo dispositivos de diversos tamaños se convirtió este calendario en un verdadero reto. Teniendo problemas visuales en dispositivos de 5 o menos de 5 pulgadas




[Figura 12] Actividad al detalle. TA Timetable

Cuando se clica en una actividad del calendario se muestra un recuadro como la imagen anterior, donde se muestra toda la información relevante. En este recuadro los estudiantes son capaces de apuntar-se en la actividad. Si se habían apuntado con anterioridad, podrán puntuar la actividad (siempre y cuando la fecha de la actividad haya pasado ya), o dejarla si finalmente no lo quieren hacer o no la han hecho al final. Durante la última entrega se añadió Google Maps a las actividades al ser uno de los requisitos iniciales al empezar el proyecto. Así cualquier estudiante sería capaz de ver fácilmente des de cualquier lugar donde se tiene que dirigir si quiere hacer la actividad.

Además del calendario, también se demandó la inclusión de una lista de todas las actividades, y clicando en una de ellas, se pudiera ver al detalle, la propia actividad y apuntarse a ella, si el estudiante quisiera.




[Figura 13] Lista Actividades. TA Timetable

Más adelante el cliente pidió que la lista estuviera ordenada entre:

- Actividades completadas
- Actividades aceptadas
- Actividades disponibles

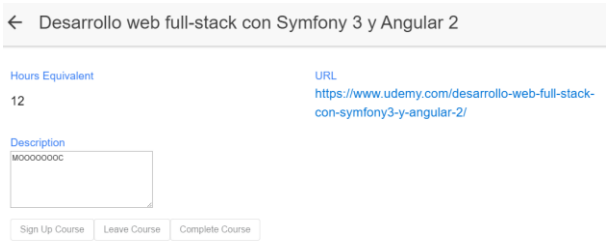
Y que cada una de estos tres subgrupos estuviera ordenado alfabéticamente.

Si fuera el profesor quien clicara en la actividad en vez de poder apuntarse, puntuarla o desapuntarse. El profesor podría eliminar y editar la actividad.

También se creó un formulario para que el profesor pudiera crear las actividades.

### 7.2.2. Cursos

A parte de las actividades, la asignatura da la opción de hacer una serie de cursos para completar la asignatura, pero no se puede completar todas las horas solamente con cursos. Eso hacía que fuera necesario mantener un control por separado de cursos y actividades. A diferencia de las actividades los cursos no se hacían un día a una hora concreta por lo tanto no era necesario añadirlas al calendario. Se hizo otra lista solamente con los cursos. Siguiendo las mismas directrices que la lista de actividades (tanto en ordenación como en estructura).

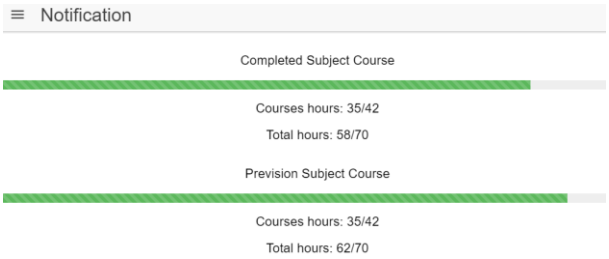



[Figura 14] Curso al detalle. TA Timetable

Al igual que las actividades se hizo un formulario para crear nuevos cursos.

### 7.2.3. Progreso de la asignatura

Para llevar un progreso de la asignatura y completar el número de horas necesarias para aprobar "Tendencias Actuales" se añadió a la aplicación unas barras de progreso

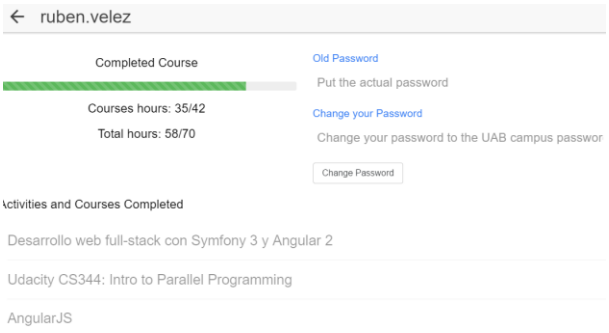



[Figura 15] Barras de progreso. TA Timetable

La primera indica todas las horas que ya se han completado, diferenciando entre cursos y total (actividades y cursos). Ya que no se puede completar la asignatura solamente haciendo cursos. La segunda muestra la previsión, si se completaran todas las actividades y cursos a los que el estudiante se apuntó con anterioridad.

### 7.2.4. Usuarios

La parte de los usuarios es totalmente distinta para el profesor y los estudiantes. Para los alumnos, cuando clican le sale su perfil, pudiendo visionar los cursos y actividades que han completado hasta ahora, pudiendo cambiar la contraseña de su usuario y pudiendo también conocer su avance en la asignatura.

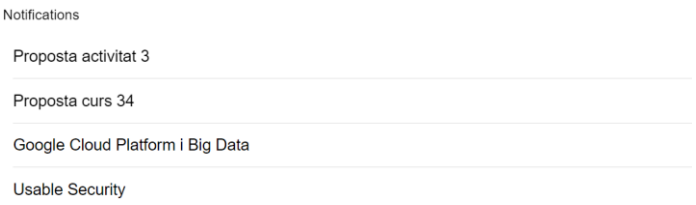



[Figura 16] Perfil usuario. TA Timetable

Pero si entráramos con el profesor, visualizaríamos la lista de todos los usuarios de la asignatura ordenados alfabéticamente y pudiendo ver en cada uno de ellos cómo va el estudiante, que ha completado, cuanto le queda etc…

### 7.2.5. Proponer actividades y cursos

Los estudiantes a petición del cliente son capaces de proponer tanto cursos como actividades al profesor, mediante un formulario. Cuando son propuesto le llegan al profesor en una lista. Clicando en uno de los ítems, el profesor podrá ver URL hacia la actividad o curso propuesto, y una descripción/comentario que el estudiante pudo dar opcionalmente.




[Figura 17] Lista de propuestas. TA Timetable

### 7.2.6. Webservices

Al iniciar la planificación y que tareas se tenían que hacer en cada una de las entregas. No se tuvo en cuenta la necesidad de tener que crear Webservices, para que cada vez que el profesor o un estudiante modificaban alguna información de la aplicación, llegara esa información a todos los demás usuarios.

Esto provocó un desvío muy importante dentro del organigrama de la aplicación. Y características que inicialmente estaban pensadas hacer como dos vistas totalmente distintas una para móviles y otra para web o el cierre y el histórico de cada año, tuvieron que ser congelados o simplemente preparados para futuras mejoras de la aplicación.

En total hay 10 Webservices en código PHP. Fueron necesarios de hacer, porque el código Angular 2 (JavaScript) no podía modificar los documentos del servidor. Estos documentos debían ser modificados por un código servidor que a la vez estuviera en el mismo servidor.

Por lo tanto se tuvieron que crear 7 Webservices en PHP que modificaran los documentos (JSON de información). Estos eran:

- Webservice modificar cursos
- Webservice modificar actividades
- Webservice modificar actividades completadas
- Webservice modificar actividades aceptadas
- Webservice modificar cursos aceptados
- Webservice modificar cursos completados
- Webservice notificaciones

Además fue necesario para utilizar el calendario escogido crear otros tres Webservices ligados al calendario:

- Get-events.php: Este Webservice era el principal y obtenía las actividades creadas. Este código llamaba a los otros dos servicios web ligados al calendario.
- Get-Timezones.php: Código simple, usado únicamente para obtener la franja horaria.

- Utils.php: Código utilizado para parsear el JSON y obtener la información útil que se la pasara a código principal (get-events.php).

### 7.3. Posibles mejores y evolutivos

Una aplicación de este estilo tiene muchas posibilidades y muchas posibles mejoras, aún y así se han escogido algunas, las que surgieron como posibles mejoras del proyecto durante la creación de este o las tareas que no se pudieron implementar.

Los principales mejoras que se podrían hacer a la aplicación serian esas características que no dieron tiempo de hacer al ser menos prioritarias. La creación de una vista móvil original y agradable para los dispositivos de dicho tamaño, principalmente para los dispositivos con un tamaño de pantalla menor o igual a siete pulgadas. Y la creación del cierre de año automático con un histórico de cada uno de los años, para que el profesor pueda ver la evolución de la asignatura. Serían las dos principales mejoras que se le podría hacer a la aplicación si hablamos sobre características que tuviera la app.

Además de estas características que estaban inicialmente pensadas añadir. También sería importante la inclusión de MongoDB para mantener una seguridad en los datos por si ocurriera algún problema, como se comentó en el apartado 6 de este documento.

Como última mejora sería bueno que los estudiantes pudieran ver cómo han puntuado las actividades y los cursos años anteriores (ligado al histórico), como referencia para saber si esa actividad se ajusta a lo que uno realmente buscaba.

## 8. Conclusiones

El objetivo principal de este proyecto era crear una aplicación multiplataforma que se ajustará a lo que demandaba el cliente, y si no era posible hacer todos los puntos que el cliente pedía, montar las bases para que en el futuro se pudieran hacer los evolutivos necesarios para que se ajustara a lo que realmente es necesario para la asignatura.

Aunque inicialmente se tenían las pautas sobre que tareas hacer muy fijadas y que tareas se tenían que hacer en cada momento, durante el proyecto se vio que el orden escogido no se ajustaba del todo al orden que se tendría que haber fijado inicialmente. Aún y así, como se pueden ver en los documentos de seguimiento se consiguió reordenar las tareas sin graves cambios en los fechas previstas inicialmente. Lo que realmente modifico las fechas fue la necesidad de crear los Webservices que originariamente no estaban previstos hacer, teniendo que recortar algunos de los objetivos menos prioritarios que había.

Realmente se puede estar contento con el resultado final de la aplicación y como se ha dejado preparada para poder poner las características que uno desee en el futuro.

## 9. Agradecimientos

Antes de concluir el artículo me gustaría agradecer a todo aquel que ha sido un apoyo durante la realización de este proyecto. Y en especial a Marc Tallo Sendra por darme los consejos adecuados sobre como conducir el proyecto y a Jordi Pons Aroztégui por haber representado el papel de cliente en el proyecto y haberle dedicado tanto tiempo, a encontrar posibles problemas tras cada iteración del proyecto.

Nunca olvidar a mi familia y pareja que siempre fueron un apoyo para mí.

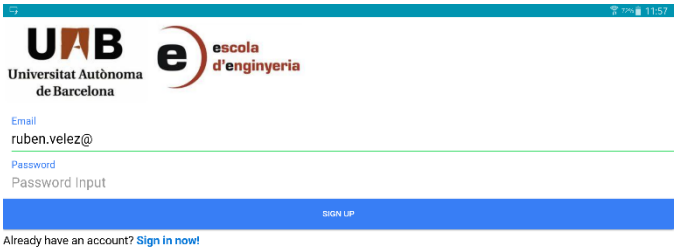
## 10. Bibliografía

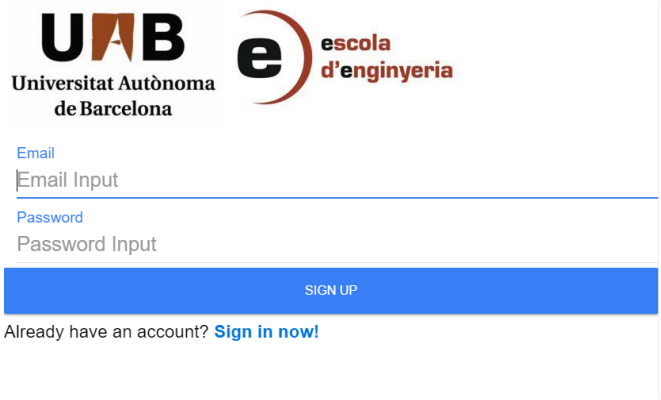
- [1] IONIC 2 Framework. (2017). 2017, de IONIC. Sitio web: http://ionicframework.com/docs/
- [2] APACHE CORDOVA. (2017). -. 2017, de APACHE. Sitio web: https://cordova.apache.org/
- [3] Google. (2017). Angular 2 Framework. 2017, de Google Sitio web: https://angular.io/
- [4] Mario G. Almonte. (2016). Las 5 Mejores Plataformas (LMS) De Elearning. 2017, de aprendizajeenred.es Sitio web: http://aprendizajeenred.es/5-mejores-plataformas-lms-elearning/
- [5] Jordi Martí. (2013). No sólo se puede vivir sin Moodle, es que se vive mejor sin él. 2017, de xarxatic.com Sitio web: http://www.xarxatic.com/no-solo-se-puede-vivir-sin-moodle-es-que-se-vive-mejor-sin-el/
- [6] Woo Commerce. (2017). -. 2017, de Woo Sitio web: https://woocommerce.com/products/sensei/
- [7] Craig Buckler. (2015). SQL vs NoSQL: The Differences. 2017, de sitepoint.com Sitio web: https://www.sitepoint.com/sql-vs-nosql-differences/

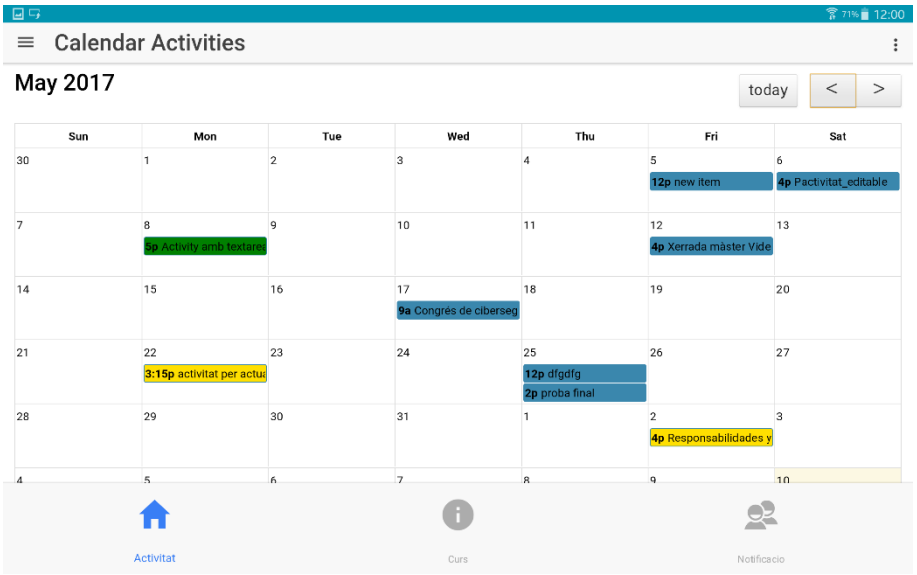
## 11. Índice de Figuras

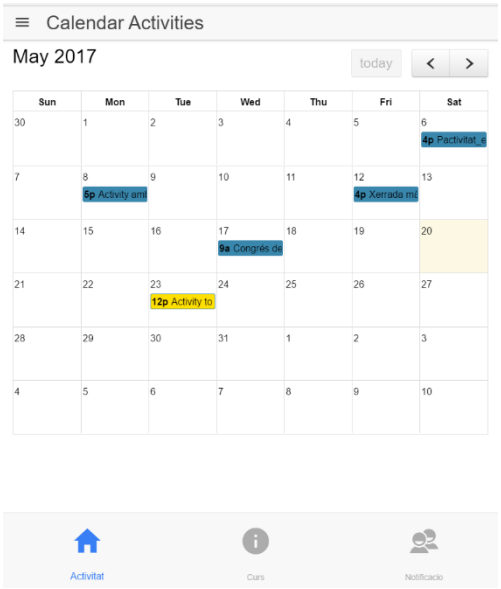
# 12. Apéndice

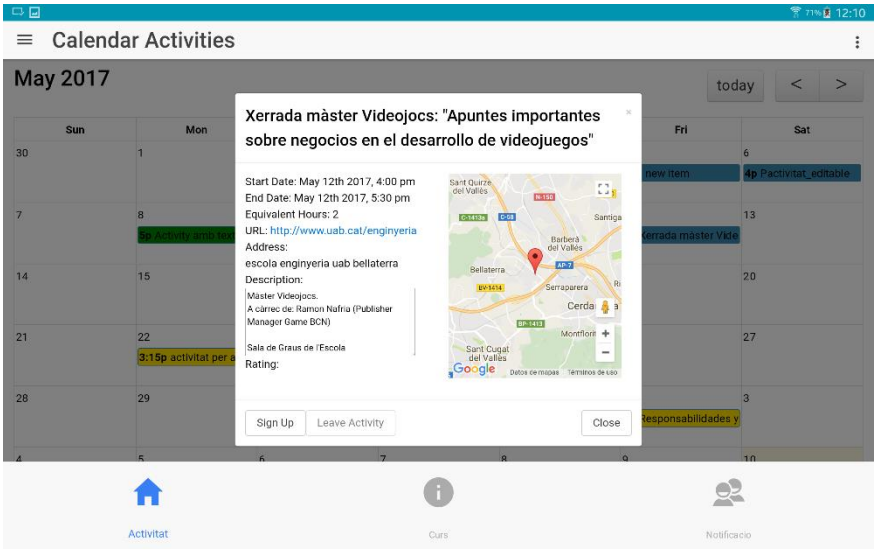
## 12.1.Comparativa Android (Samsung Galaxy Tab S) - Web

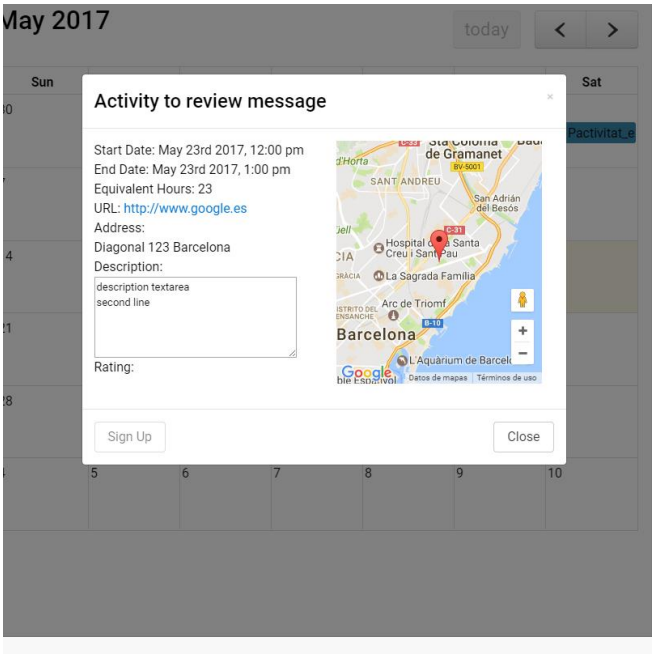


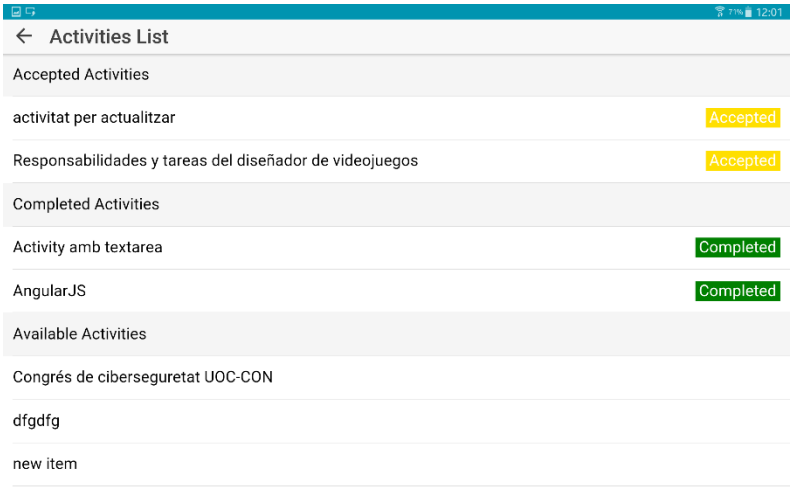




Calendar Activities
May 2017
today
Xerrada màster Videojocs: "Apuntes importantes sobre negocios en el desarrollo de videojuegos"
Start Date: May 12th 2017, 4:00 pm
End Date: May 12th 2017, 5:30 pm
Equivalent Hours: 2
URL: http://www.uab.cat/enginyeria
Address:
escola enginyeria uab bellaterra
Description:
Master Videojocs.
A càrrec de: Ramon Nafria (Publisher Manager Game BCN)
Sala de Graus de l'Escola
Rating:
Sign Up
Leave Activity
Close
Activitat
Curs
Notificacio


May 2017
today
Activity to review message
Start Date: May 23rd 2017, 12:00 pm
End Date: May 23rd 2017, 1:00 pm
Equivalent Hours: 23
URL: http://www.google.es
Address:
Diagonal 123 Barcelona
Description:
description textarea
second line
Rating:
Sign Up
Close


Activities List
Accepted Activities
activitat per actualitzar
Accepted
Responsabilidades y tareas del diseñador de videojuegos
Accepted
Completed Activities
Activity amb textarea
Completed
AngularJS
Completed
Available Activities
Congrés de ciberseguretat UOC-CON
dfgdfg
new item


Activities List
Accepted Activities
activitat per actualitzar
Accepted
Responsabilidades y tareas del diseñador de videojuegos
Accepted
Completed Activities
Activity amb textarea
Completed
AngularJS
Completed
Available Activities
Congrés de ciberseguretat UOC-CON
dfgdfg
new item
Pactivitat_editable
proba final
Xerrada màster Videojocs: "Apuntes importantes sobre negocios en el desarrollo de videojuegos"

## 12.2.Código PHP Ejemplo

[Ejemplo Webservices para modificar un fichero]

```
// Require our Event class and datetime utilities
require dirname(__FILE__) . '/utils.php';

// Short-circuit if the client did not give us a date range.
if (!isset($_GET['start']) || !isset($_GET['end'])) {
	die("Please provide a date range.");
}

// Parse the start/end parameters.
// These are assumed to be ISO8601 strings with no time nor timezone, like "2013-12-29".
// Since no timezone will be present, they will parsed as UTC.
$range_start = parseDateTime($_GET['start']);
$range_end = parseDateTime($_GET['end']);

// Parse the timezone parameter if it is present.
$timezone = null;
if (isset($_GET['timezone'])) {
	$timezone = new DateTimeZone($_GET['timezone']);
}

// Read and parse our events JSON file into an array of event data arrays.
$json = file_get_contents(dirname(__FILE__) . '/../json/events.json');
$input_arrays = json_decode($json, true);

// Accumulate an output array of event data arrays.
$output_arrays = array();
foreach ($input_arrays as $array) {

	// Convert the input array into a useful Event object
	$event = new Event($array, $timezone);

	// If the event is in-bounds, add it to the output
	if ($event->isWithinDayRange($range_start, $range_end)) {
		$output_arrays[] = $event->toArray();
	}
}

// Send JSON to the client.
echo json_encode($output_arrays);
```

[Ejemplo de Webservice del calendario]

```
<?php
 if (isset($_SERVER['HTTP_ORIGIN'])) {
        header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
        header('Access-Control-Allow-Credentials: true');
        header('Access-Control-Max-Age: 86400');    // cache for 1 day
    }

    // Access-Control headers are received during OPTIONS requests
    if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {

        if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_METHOD']))
            header("Access-Control-Allow-Methods: GET, POST, OPTIONS");

        if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']))
            header("Access-Control-Allow-Headers:        {$_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']}");

        exit(0);
    }


    $postdata = file_get_contents("php://input");
       if (isset($postdata)) {
              if ($postdata != "") {
                     echo "Server returns: " . $postdata;

                     $relativePath = "/ta/json/events.json";
                     $savePath = $_SERVER["DOCUMENT_ROOT"].$relativePath;
                     file_put_contents($savePath,$postdata);
              }
              else {
                     echo "Empty parameter!";
              }
       }else {
              echo "Not called properly with username parameter!";
       }

 /*$handle = fopen($savePath, "ab+");
if (!is_resource($handle)) { // Test if PHP could open the file
    echo "Could not open {$savePath} for writting.";
}
echo($handle);
fwrite($handle,"sss");
fclose($handle);*/

?>
```