|  |  |  |
| --- | --- | --- |
|  | UNIVERSIDAD DE BURGOS  ESCUELA POLITÉCNICA SUPERIOR  Grado en Ingeniería en Informática |  |

MEMORIA

|  |  |
| --- | --- |
| **TFG del Grado en Ingeniería en Informática**  **Aplicación para el Acceso a Plataformas de ELearning desde Dispositivos Móviles** |  |

Presentado por Daniel Puente Gabarri en Universidad de Burgos — dd de mm de YYYY Tutor: María Belén Vaquerizo García y Bruno Baruque Zanón

|  |  |  |
| --- | --- | --- |
|  | UNIVERSIDAD DE BURGOS  ESCUELA POLITÉCNICA SUPERIOR  Grado en Ingeniería en Informática |  |

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone: Que el alumno D. Daniel Puente Gabarri, con DNI 71347273-P, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

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

En Burgos, 10 de febrero de 2017

|  |  |
| --- | --- |
| Vº. Bº. del Tutor | Vº. Bº. del co-tutor |
| D. nombre tutor | D. nombre co-tutor |

**Agradecimientos**

**Resumen**

**Descriptores**

**Abstract**

**Keywords**

# Índice general

[MEMORIA 1](#_Toc476646442)

[Índice general 7](#_Toc476646443)

[Índice de ilustraciones 10](#_Toc476646444)

[Índice de tablas 11](#_Toc476646445)

[1. Introducción 12](#_Toc476646446)

[2. Objetivos del proyecto 12](#_Toc476646447)

[2.1. Objetivos funcionales 12](#_Toc476646448)

[2.2. Objetivos de carácter técnico 13](#_Toc476646449)

[2.3. Objetivos personales 13](#_Toc476646450)

[3. Conceptos teóricos 14](#_Toc476646451)

[3.1. Web API 14](#_Toc476646452)

[3.1.1. API 14](#_Toc476646453)

[3.1.2. Web API 14](#_Toc476646454)

[3.1.2.1. SOAP 14](#_Toc476646455)

[3.1.2.2. XML-RPC 14](#_Toc476646456)

[3.1.2.3. JSON-RPC 15](#_Toc476646457)

[3.1.2.4. REST 15](#_Toc476646458)

[3.2. LTI 15](#_Toc476646459)

[3.3. Soap 15](#_Toc476646460)

[3.4. Rest 16](#_Toc476646461)

[4. Técnicas y herramientas 18](#_Toc476646462)

[4.1. Lenguajes 18](#_Toc476646463)

[4.1.1. PHP 18](#_Toc476646464)

[4.1.2. JSON 18](#_Toc476646465)

[4.1.3. Android 18](#_Toc476646466)

[4.2. Herramientas 18](#_Toc476646467)

[4.2.1. XAMPP 18](#_Toc476646468)

[4.2.2. GitHub 19](#_Toc476646469)

[4.2.3. TortoiseSVN 19](#_Toc476646470)

[4.2.4. Moodle 19](#_Toc476646471)

[4.2.5. PhpStorm 19](#_Toc476646472)

[4.2.6. phpMyAdmin 19](#_Toc476646473)

[4.2.7. PHPUnit 19](#_Toc476646474)

[4.2.8. SonarQube 19](#_Toc476646475)

[4.2.9. Advanced REST Client 19](#_Toc476646476)

[4.2.10. Modelio 19](#_Toc476646477)

[4.3. Técnicas 20](#_Toc476646478)

[4.3.1. LTI 20](#_Toc476646479)

[4.3.2. MVC 20](#_Toc476646480)

[4.3.3. SCRUM 20](#_Toc476646481)

[5. Trabajos relacionados 21](#_Toc476646482)

[6. Aspectos relevantes del desarrollo del proyecto 21](#_Toc476646483)

[7. Conclusiones y líneas de trabajo futuras 21](#_Toc476646484)

[ANEXOS 22](#_Toc476646485)

[I. Plan de proyecto software 23](#_Toc476646486)

[I.1. Introducción 23](#_Toc476646487)

[I.2. Planificación temporal del proyecto 23](#_Toc476646488)

[I.2.1. Sprint 1. Inicio del proyecto (7 Feb 2017 a 16 Feb 2017) 23](#_Toc476646489)

[I.2.2. Sprint 2. Estudio de refactorización del servidor (16 Feb 2017 a 2 Mar 2017) 24](#_Toc476646490)

[I.2.3. Sprint 3. Diseño del BackEnd (2 Mar 2017 a 15 Mar 2017) 26](#_Toc476646491)

[I.3. Estudio de viabilidad 26](#_Toc476646492)

[I.4. Estudio de viabilidad económica. 26](#_Toc476646493)

[I.5. Estudio de viabilidad legal. 26](#_Toc476646494)

[II. Especificación de requisitios 27](#_Toc476646495)

[II.1. Introducción 27](#_Toc476646496)

[II.2. Objetivos generales 27](#_Toc476646497)

[II.3. Catálogo de requisitos 27](#_Toc476646498)

[II.3.1. Requisitos funcionales 27](#_Toc476646499)

[I.1.1. Requisitos no funcionales 28](#_Toc476646500)

[I.1.2. Restricción de requisitos 28](#_Toc476646501)

[II.4. Especificación de requisitos 28](#_Toc476646502)

[I. Especificación de diseño 29](#_Toc476646503)

[I.1. Introducción 29](#_Toc476646504)

[I.2. Diseño de datos 29](#_Toc476646505)

[I.3. Diseño procedimental 29](#_Toc476646506)

[I.4. Diseño arquitectónico 29](#_Toc476646507)

[II. Manual del programador 29](#_Toc476646508)

[II.1. Introducción 29](#_Toc476646509)

[II.2. Estructura de directorios 29](#_Toc476646510)

[II.3. Manual del programador 29](#_Toc476646511)

[II.4. Compilación, instalación y ejecución del proyecto 29](#_Toc476646512)

[II.5. Pruebas del sistema 29](#_Toc476646513)

[III. Manual del usuario 29](#_Toc476646514)

[III.1. Introducción 29](#_Toc476646515)

[III.2. Requisitos de usuarios 29](#_Toc476646516)

[III.3. Instalación 29](#_Toc476646517)

[III.4. Manual del usuario 29](#_Toc476646518)

[Bibliografía 30](#_Toc476646519)

# Índice de ilustraciones

[Ilustración 1 - Estructura de un mensaje soap [3] 16](#_Toc476646525)

[Ilustración 2 - Gráfico Burndown Sprint 1. Inicio del proyecto 24](#_Toc476646526)

[Ilustración 3 - Gráfico burndown sprint 2. Estudio de refactorización del servidor 25](#_Toc476646527)

# Índice de tablas

[Tabla 1 - Coste Hardware 26](#_Toc476657551)

[Tabla 2 - Coste Software - 1 27](#_Toc476657552)

[Tabla 3 -Coste Software – 2 27](#_Toc476657553)

[Tabla 4 - Coste de aprendizaje 28](#_Toc476657554)

[Tabla 5 - Otros costes 29](#_Toc476657555)

[Tabla 6 - Costes totales 29](#_Toc476657556)

# Introducción

Este trabajo de final de grado nace de la base de un proyecto anterior <nombre\_del proyecto>, debido a la necesidad de ampliar la funcionalidad de dicho proyecto a una aplicación móvil que permita llevar a cabo las mismas funcionalidades y acceder a los mismos contenidos a disposición del alumno por medio del sistema E-Learning de la Universidad de Burgos.

# Objetivos del proyecto

A continuación, se va a llevar a cabo una citación y argumentación de los distintos objetivos a realizar en este proyecto con el objetivo de transmitir la finalidad del mismo.

## Objetivos funcionales

Partiendo del proyecto de partida identificamos los siguientes nuevos objetivos:

* + 1. Creación de una aplicación móvil que permita acceder y realizar las mismas actividades que permite la aplicación Web. Estas actividades varían en función del tipo de usuario que las desempeñe.
       1. Docentes:
          1. Llevarán a cabo las tareas de crear, editar, publicar o duplicar un cuestionario en Moodle.
          2. Estos cuestionarios permitirán a los docentes poder llevar a cabo una evaluación de los conocimientos adquiridos por su alumnado.
          3. Al duplicar los cuestionarios estos podrán reutilizarse por los docentes para los distintos grupos o asignaturas que dicho docente imparta.
          4. Además, gracias a esta aplicación el docente podrá llevar a cabo todas estas funcionalidades con total comodidad desde su Smartphone.
       2. Alumnos:
          1. Ofrecer la posibilidad de resolver los distintos cuestionarios a los que tenga que enfrentarse desde su Smartphone.
          2. Al finalizar el mismo, el sistema notificará al alumno de la calificación obtenido junto con una retroalimentación de las diferentes preguntas.
          3. Estas calificaciones podrán variar en función de distintas recompensas a la hora de enfrentarse al cuestionario.
          4. Estas recompensas o *comodines* a partir de ahora, permitirán al alumnado enfrentarse a la prueba de una manera más amigable al enmascarar la verdadera finalidad del cuestionario.

## Objetivos de carácter técnico

* + 1. La aplicación móvil deberá de poder ser lo suficientemente amigable para los distintos usuarios para facilitar su correcta utilización y finalidad.
    2. La aplicación móvil a desarrollar se llevará a cabo para Android.
    3. Para llevar a cabo la creación de la aplicación se utilizará el entorno de desarrollo Android Studio.
    4. <Versión: de la aplicación>
    5. Como conectaremos Android con la Web
    6. aPI APLICACIÓN, FUNCIONE EN UN DISPOSITIVOS. VERSION

## Objetivos personales

* + 1. Destacar que uno de los objetivos principales de este proyecto es adquirir nuevos conocimientos dentro del desarrollo *Android*, junto con otros conocimientos necesarios para poder llevar a cabo la correcta integración del proyecto de partida a el proyecto a desarrollar.
    2. Afrontarme a nuevos retos que pongan a prueba todos mis conocimientos adquiridos a lo largo del grado.
    3. Enfrentarme a un posible trabajo o proyecto que se asemeje a mi vida profesional.

# Conceptos teóricos

En esta sección se va proceder a la explicación de ciertos conceptos teóricos necesarios para la correcta compresión de este trabajo. Además, dichos conceptos han sido necesarios para llevar a cabo la toma decisiones sobre cómo resolver el trabajo junto con la realización del mismo.

## Web API

Antes de hablar de que es un *Web API* tendremos que explicar que es un *API,* ya que la funcionalidad de un *Web API* es similar a la de un *API,* pero orientada a la Web.

### API

Una posible definición podría ser la siguiente: “*Un API (siglas de ‘Application Programming Interface’) es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software”.* [1]

Es decir, permite la comunicación entre distintos componentes software. La mayor ventaja es que permite reutilizar métodos escrito en un determinado lenguaje o software, de esta manera evitamos la existencia de duplicidad de una misma funcionalidad en los diferentes componentes. Además, estas funcionalidades se encuentran testeadas y funcionan de forma adecuada en un determinado componente software.

### Web API

En este caso y relacionado con lo anteriormente explicado la lógica de un *Web API* es la misma que un *API* salvo que en este caso esta comunicación, es decir, el intercambio de información se realiza entre un servicio web y una aplicación mediante una *URL.* Para llevar a cabo esta comunicación se utilizan peticiones *HTTP o HTTPS* y toda esta información se encuentra encapsulada generalmente en *XML o JSON*.

Existen principalmente cuatro tipos de *Web API*.

### SOAP

Es un protocolo estándar de intercambio de información y datos en *XML* entre dos objetos cuyas siglas son las siguientes *Simple Object Access Protocol.* Posteriormente en esta misma sección se dedicará un apartado donde se explicará en mayor.

### XML-RPC

Es un protocolo que llama a un procedimiento remoto que utiliza *XML* para encapsular los datos y llamadas *HTTP* para llevar a cabo la comunicación.

### JSON-RPC

Es un protocolo cuya lógica es igual que el protocolo explicado anteriormente, salvo que en este caso utiliza el formato *JSON* para encapsular los datos.

### REST

En este caso es una arquitectura software para sistemas hipermedia en la *World Wide Web.* Además, esta arquitectura utiliza el protocolo *HTTP* para llevar a cabo la comunicación. No obstante, en esta misma sección se dedicará un apartado para llevar a cabo una explicación en mayor detalle.

[2]

## LTI

## Soap

Anteriormente se ha llevado a cabo una pequeña explicación sobre este protocolo de comunicación. Relacionado con esto definíamos *SOAP* como un protocolo que posibilita el intercambio de información, es decir, la comunicación mediante internet entre aplicaciones. Para llevar a cabo este intercambio de datos se utiliza el formato *XML*, gracias a este el intercambio de datos se puede realizar independientemente de la plataforma o lenguaje utilizado.

Hoy en día, estos mensajes de tipo *SOAP* se envían generalmente mediante el protocolo *HTTP*, aunque también se pueden utilizar otros como *SMTP, TCP o JMS* entre otros. Otro aspecto importante es que es un pilar fundamental para los *Web Services*, ya que estos utilizan también el formato *XML* para describir los servicios. Además, permite el intercambio de datos o llamadas a procedimientos remotos mediante *RPC.*

Para poder llevar a cabo la correcta lectura de un mensaje *SOAP* el documento *XML* debe de tener una determinada estructura. Dicha estructura debe estar formada por las siguientes partes:

* **Envelope:** esta parte de la estructura del mensaje es obligatoria, ya que identifica el mensaje.
* **Header:** esta parte de la estructura permite enviar información adicional sobre cómo debe de procesarse el mensaje. Es por esto por lo que esta parte no es obligatoria, debido a que únicamente aporta cierta información relacionada con la lectura del mismo.
* **Body:** esta parte de la estructura es la encargada de almacenar toda la información del mensaje.
* **Fault:** esta parte de la estructura al igual que el *Header* es opcional, ya que aporte información relacionada con ciertos errores producidos durante el procesado del mensaje.

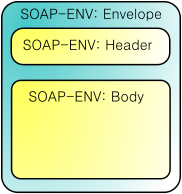


Ilustración 1 - Estructura de un mensaje soap [3]

[3] [4]

## Rest

Como ya hemos explicado anteriormente,es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.[5]cuyas siglas son las siguientes **RE**presentational **S**tate **T**ransfer o en castellano Transferencia de Estado Representacional.

En la actualidad, el termino *REST* se utiliza para aquellas interfaces que utilicen *HTTP* para el intercambio de información entre sistemas pudiendo utilizar cualquier formato para encapsular los datos, aunque generalmente los más utilizados son *XML y JSON.* Además, actualmente los sistemas que siguen las pautas o principios *REST* también se les suelen denominar *RESTful.*

Para que un sistema se considere *RESTful* debe de cumplir con las siguientes pautas o principios.

* **Protocolo cliente/servidor sin estado:** de manera que cada petición *HTTP* debe contener toda aquella información necesaria para poder ejecutarse, es decir, de esta forma evitamos que tanto el cliente como el servidor tengan que almacenar información sobre el estado previo de la misma para poder llevarla a cabo.
* **Operaciones**: un sistema *REST* debe poder llevar cabo las siguientes operaciones: ***POST*** (crear), ***GET*** (leer o consultar), ***PUT*** (editar) y ***DELETE*** (borrar). Estas operaciones como podemos observar se asemejan en gran parte a las operaciones *CRUD* en bases de datos.
* **Sistema de capas**: el sistema deberá de utilizar una arquitectura jerárquica entre los distintos componentes que la formen, de esta manera garantizamos que cada una de estas capas se encargue de llevar a cabo una única funcionalidad.
* **Manipulación de recursos**: para realizar la manipulación de los objetos se lleva a cabo mediante la *URI.* Dicha *URI* se utiliza como el identificador único para cada recurso, un recurso es un elemento de información. De esta manera se simplifica el acceso a la información para su posterior manipulación.
* **Uso de hipermedios**: para las transiciones entre los distintos estados de la aplicación y para la información de la misma.

[6]

# Técnicas y herramientas

En esta sección se llevará a cabo una mención y breve explicación sobre el conjunto técnicas y herramientas utilizadas durante el desarrollo del proyecto.

## Lenguajes

### PHP

*PHP* es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en *HTML* [7]cuyas siglas es un acrónimo recursivo de ***P****HP* ***H****ypertext* ***P****reprocesssor*. Este lenguaje se utiliza principalmente en el lado del servidor y está enfocado en el desarrollo web para el contenido dinámico [8].

Este lenguaje se ha decidido utilizar frente a otras alternativas en el lado del servidor debido principalmente a que el proyecto de partida se basa en este lenguaje para el acceso a la base de datos. Además, cabe destacar que desde mi punto de vista me parece una gran decisión al ser uno de los lenguajes más utilizados en el lado del servidor, existe una gran documentación al respecto y la curva de aprendizaje es menos costosa al asemejarse a otros lenguajes orientados a objetos.

### JSON

*JSON* es un formato ligero de intercambio de datos cuyas siglas es un acrónimo de ***J****ava****S****cript* ***O****bject* ***N****otation,* es decir, Notación de Objetos de JavaScript [9]. *JSON* es un formato de texto independientemente del lenguaje y se le considera una gran alternativa frente al formato de intercambio de datos *XML* al ser mucho más sencillo de parsear por un analizador sintáctico [10].

Este lenguaje se ha decidido utilizar para la devolución de los diferentes datos a la aplicación Android. De esta forma, la comunicación de la aplicación se podrá realizar de una forma más sencilla con la base de datos. Además, las propias librerías de *Android* contienen ciertas clases para poder realizar de forma fácil un correcto tratamiento de los datos.

### Android

## Herramientas

### XAMPP

*XAMPP* es un paquete de instalación independiente de la plataforma y de software libre. Dicho paquete contiene el sistema de gestión de bases de datos *MySQL¸* el servidor web *Apache* y los interpretes para lenguajes de script: *PHP* y *Perl* [11]. Su nombre proviene del acrónimo ***X*** indicando que es compatible para cualquier sistema operativo y el resto de siglas *AMPP* hace referencia a cada uno de los elementos que contiene el paquete, es decir, ***A****pache,* ***M****ySQL,* ***P****HP,* ***P****erl* respectivamente.

Esta herramienta se ha decidido utilizar principalmente al ser utilizada en el proyecto de partida. Además, cabe destacar que me parece una gran decisión ya que el propio paquete contiene todo lo necesario para poder crear un servidor web.

### GitHub

*GitHub* es una plataforma de desarrollo colaborativo para alojar proyectos utilizando un sistema de control de versiones *Git* [12].

Se ha decidido optar por esta herramienta al haberla utilizado en ciertas ocasiones, al ser una de las más utilizadas para el desarrollo colaborativo y al existir un plugin *ZenHub* para la gestión desde el propio repositorio de las distintas tareas a realizar en un panel de trabajo*.* Además, cabe destacar que el repositorio permite dos tipos de formato para el repositorio: público y privado. En nuestro caso se encuentra público

**Enlace**: <https://github.com/danielpuente-dpg/GII14.K.QUICKTEST>

### TortoiseSVN

*TortoiseSVN* es un software libre utilizado para llevar a cabo el control de versiones del proyecto sobre *GitHub* [13]. Implementa una extensión del shell de Windows, por lo que el control de versiones no tiene por qué realizarse mediante línea de comandos, sino que se puede realizar mediante la interfaz gráfica que proporciona esta herramienta la cual es muy fácil de utilizar.

### Moodle

### PhpStorm

### phpMyAdmin

### PHPUnit

### SonarQube

### Advanced REST Client

### Modelio

*Modelio* es una herramienta *Open Source*, desarrollada por *Modeliosoft*. Esta herramienta soporta los estándares *UML2 y BPMN* [14].

Se ha decidido utilizar esta herramienta para el modelado de los distintos diagramas del proyecto al ser recomendada anteriormente en una asignatura, al ser *Open Source* y principalmente al ser fácil de usar.

## Técnicas

### LTI

### MVC

### SCRUM

*SCRUM* es un modelo de referencia que define un conjunto de práctica y roles, que pueden utilizarse como punto de partida para llevar a cabo la definición de como se elaborará el proceso de desarrollo del proyecto [15]. Cabe destacar que este modelo se encuentra dentro de los marcos de las metodologías agiles y que es de los más utilizados actualmente.

Los roles principales son:

* **Scrum Master:** se encarga de gestionar los cambios y procura facilitar la aplicación de esta metodología.
* **Product Owner:** en este grupo se encontrarán el personal interno o externo que representa al cliente. Este grupo se encargará de que el trabajo se realice de forma acorde con las necesidades y peticiones del cliente.
* **Team:** representa al equipo de desarrollo encargados de ejecutar el desarrollo y de entregar el producto deseado.

[15]

Para realizar el desarrollo del producto se deben de definir un *Product Backlog,* el cual contendrá todas las historias de usuario a realizar. Dichas historias de usuario se asignarán a los distintos *Sprints* o iteraciones que se realicen a lo largo del desarrollo del producto hasta finalizar con todas las historias de usuario. Se ha definido que la duración de los distintos *Sprints* sea de 2 semanas al ser lo más recomendable.

Además, cabe destacar que se ha decidido utilizar esta metodóloga de desarrollo principalmente al ser la metodología más recomendada a utilizar a lo largo de la carrera, tanto mis tutores como yo nos encontramos cómodos usando dicha metodología, se puede llevar a cabo ciertas entregas por cada sprint distribuyendo el trabajo de manera más adecuada y principalmente, garantizamos una mayor productividad y calidad del producto a entregar.

# Trabajos relacionados

# Aspectos relevantes del desarrollo del proyecto

# Conclusiones y líneas de trabajo futuras

Patron singleton, en las dos DB

Usar PDO para las sentencias preparadas y para accerder a la DB

Uso de un Batch para tratar las peticiones que realicen cambios sobre la DB post,put,delete, asi ahorramos batería

Mejorar el método createCuestionario

ANEXOS

|  |  |  |
| --- | --- | --- |
|  | UNIVERSIDAD DE BURGOS  ESCUELA POLITÉCNICA SUPERIOR  Grado en Ingeniería en Informática |  |
|  |  |  |

|  |  |
| --- | --- |
| **Anexos**  **Aplicación para el Acceso a Plataformas de ELearning desde Dispositivos Móviles** |  |

Presentado por Daniel Puente Gabarri en Universidad de Burgos — dd de mm de YYYY Tutor: María Belén Vaquerizo García y Bruno Baruque Zanón

# Plan de proyecto software

## Introducción

A continuación, en esta sección se encuentra toda aquella información relacionada con la gestión del proyecto. Esta información no solamente va a estar relacionada con el desarrollo del producto pedido, sino que englobará otros aspectos relevantes como **la definición de los riesgos o costes** del plan proyecto.

Como se ha comentado anteriormente en la sección 4.3.3, la metodología de trabajo utilizada para este proyecto es *SCRUM*. Para llevar cabo al desarrollo del proyecto se ha divido el desarrollo del mismo en *Sprints o Iteraciones* de dos semanas. Cabe destacar que algún *Sprint* varia en esta duración establecida debido a la disponibilidad del equipo para poder llevar a cabo la reunión al finalizar cada iteración.

## Planificación temporal del proyecto

Como se comentaba, se ha decidido dividir el desarrollo del proyecto en *Sprints* de dos semanas de duración. Gracias a esta duración se podrá compaginar de una manera más eficiente el desarrollo del proyecto de final de grado junto con el resto de asignaturas de este cuatrimestre. En este cuatrimestre el número de asignaturas cursadas son tres junto con el trabajo de fin de grado. Además, se ha estipulado que el número de horas de dedicación por iteración sea de alrededor de cuarenta, este es el principal motivo por el cual se ha decidido que la duración de cada iteración sea de quince días para poder compaginar el desarrollo del proyecto junto con el resto de asignaturas.

### Sprint 1. Inicio del proyecto (****7 Feb 2017**** a ****16 Feb 2017****)

Este día tuvo lugar la primera reunión para la explicación sobre el producto a desarrollar. Una vez explicado el mismo, se decidió aceptar este proyecto y se comenzó con la definición de los principales pilares a superar para construir el producto. Una vez definidas estas bases se decidió realizar en este *Sprint* las siguientes tareas.

* **Cargar código fuente:** este primer *commit* fue realizado por los tutores para incluir el proyecto de partida.
* **Objetivos del proyecto:** esta tarea ha sido realizada por el alumno sobre la documentación a entregar. En dicho apartado del documento se definirán las bases principales del proyecto junto con sus objetivos más destacados.
* **Instalar proyecto de partida:** en tarea fue realizada por el alumno, para ello se ha realizado una lectura de la documentación del proyecto de partida y su posterior instalación para conocer al máximo la lógica a implementar.
* **Generar el Product Backlog:**  esta tarea fue realizada por el alumno en la memoria. En esta sección se llevó a cabo la definición de los principales requisitos funcionales las cuales derivarán en las historias de usuario a realizar en los diferentes *Sprint.*
* **Aprendizaje PHP:** esta tarea fue la más costosa de esta iteración ya que se desconocía cualquier conocimiento acorde a este lenguaje. Para ello se realizado una lectura de la documentación proporcionada en la propia *web de PHP* [16] junto con otras fuentes online como *tutoriales sobre PHP desde cero.*

A continuación, en la siguiente *Ilustración* se incluye un gráfico *Burndown* resumiendo el *Sprint*, cabe destacar que en esta ocasión solamente se aprecia que todas las tareas fueron finalizadas el mismo día, aunque esto no fue así. Este error nace fruto de un desconocimiento previo de la herramienta, al no incluir en la configuración de la creación de dicha gráfico las tareas que se encuentren dentro del tablero en la sección *Done*. Posteriormente, una vez localizado y subsanado este error en las siguientes iteraciones este fallo no se produce.



Ilustración 2 - Gráfico Burndown Sprint 1. Inicio del proyecto

### Sprint 2. Estudio de refactorización del servidor (****16 Feb 2017**** a ****2 Mar 2017****)

Al comienzo de esta iteración se realizó la revisión de las tareas a realizar en el *Sprint* anterior y una vez revisadas se dio por finalizado el *Sprint 1. Inicio del proyecto (****7 Feb 2017*** *a* ***16 Feb 2017****).*

En esta iteración se decidió realizar las siguientes tareas:

* **Estudio de inclusión de un Framework:** esta tarea fue realizada por el alumno y se realizado un estudio de diferentes alternativas como: *CakePHP* [17], *Laravel* [18], *Symphony* [19], *Slim* [20]. No obstante, la decisión ha sido no utilizar ningún Framework al tratarse de un proyecto pequeño en dimensiones y al añadir mayor complejidad a la resolución del mismo.
* **Listar los métodos a implementar en el API:** esta tarea ha sido realizada por el alumno y en ella se listan las principales lógicas a implementar sobre la base de datos.
* **Estudio de diferencias entre REST y SOAP:** esta tarea también fue realizada por el alumno, para ello se realizó una correcta lectura y posterior documentación en el apartado *Soap* y *Rest*. Fruto de este estudio se ha decidido utilizar *REST* como arquitectura para facilitar el uso y tratamiento de las peticiones desde el programa.
* **Documentación: Aspectos Teóricos / Técnicas y Herramientas:** esta tarea consiste en llevar a cabo una documentación en la memoria sobre estos aspectos en la sección *Conceptos teóricos* y *Técnicas y herramientas*.

En la siguiente *Ilustración* podemos observar el gráfico Burndown correspondiente a esta iteración.



Ilustración 3 - Gráfico burndown sprint 2. Estudio de refactorización del servidor

### Sprint 3. Diseño del BackEnd (2 Mar 2017 a 15 Mar 2017)

## Estudio de viabilidad

Uno de los objetivos principales de cualquier proyecto por no clasificarlo como el más importante es conocer si este proyecto es viable. Para ello se debe realizar un estudio sobre la viabilidad del mismo. Es por esto que se deberá realizar este estudio sobre dos enfoques: ***económico y legal***. Y gracias a estos sabremos si el proyecto es rentable y viable.

### Estudio de viabilidad económica.

Relacionado con lo anteriormente expuesto en esta sección se va realizar un estudio sobre la viabilidad del proyecto en el enfoque ***económico.*** Para ello se va realizar un análisis sobre los diferentes costes que entran en juego y son necesarios para el desarrollo del mismo junto con el mantenimiento una vez desarrollado.

#### Coste Hardware

Para el desarrollo del proyecto y posterior corrección de ciertos problemas que puedan surgir será necesario:

* ***Un ordenador potente en prestaciones***, ya que el *IDE* de *Android Studio* es bastante pesado en tareas principalmente de emulación.
* ***Un dispositivo móvil***, es decir un Smartphone con *S.O* *Android.*

En este caso se utilizarán las herramientas propias del alumno encargado de realizar el proyecto, que son las siguientes:

* ***Asus GL552VW-DM142T:*** coste actual 1199€ sin *S.O.*
* ***OnePlus 2:*** coste actual 319€

Ambos componentes son bastante potentes y por lo tanto no ocasionarán ninguna problemática de que el hardware se quede anticuado en prestaciones en posteriores años. Para redondear los posteriores cálculos estimaremos que el tiempo de vida de ambos componentes será de 5 años, es decir 60 meses. Aunque seguramente su tiempo de vida podría alargarse aún más. Además, al realizarse el proyecto en un cuatrimestre la duración de la utilización de dichos componentes será de 4 meses.

|  |  |
| --- | --- |
| Tiempo de amortización | 60 meses |
| Coste de los componentes | 1199,00 € + 319,00 € = 1518,00 € |
| Coste de amortización/mes | 1518,00 € / 60 meses = 25,3 €/mes |
| Coste final | 25,3 €/mes \* 4 meses = 101,2 € |

Tabla 1 - Coste Hardware

#### Coste Software

Para la realización del proyecto y su posterior mantenimiento será necesario lo siguiente:

|  |  |  |
| --- | --- | --- |
| Software | Licencia | Coste |
| Microsoft Windows 10 | Home | 135,00 € |
| Android Studio | Licencia Apache 2.0 [21] | 0,00 € |
| XAMPP | GNU [11] | 0,00 € |
| GitHub |  |  |
| TortoiseSVN | GNU General Public License [13] | 0,00 € |
| Moodle | GNU GPL [22] | 0,00 € |
| PhpStorm | Estudiante | 0,00 € |
| SonarQube | LGPL | 0,00 € |
| Advanced REST Client |  |  |
| Modelio | GPL, Licencia Apache 2.0, EULA | 0,00 € |
| Microsoft Office | Hogar | 99,00 € |
|  |  | 234,00 € |

Tabla 2 - Coste Software - 1

En este caso la licencia de *Microsoft Office* es para un único año de duración, pero la licencia del *S.O* no tiene caducidad por lo que solamente tendremos en cuenta una duración anual.

|  |  |
| --- | --- |
| Tiempo de amortización | 12 meses |
| Coste de los componentes | 135,00 € + 99,00 € = 234,00 € |
| Coste de amortización/mes | 234,00 € / 12 meses = 19,5 €/mes |
| Coste final | 19,5 €/mes \* 4 meses = 78,00€ |

Tabla 3 -Coste Software – 2

#### Coste de instalación

Este coste será nulo y no se tendrá en cuenta ya que la instalación de los distintos componentes será realizada por el propio alumno.

#### Coste de aprendizaje

Este proyecto será desarrollado por el alumno por lo que este coste de realización será un sueldo mensual a dicho alumno por llevar a cabo el diseño, implementación y testeo del mismo. Como se comentaba anteriormente se ha estipulado que el número de horas dedicadas al desarrollo del mismo será de 40 horas cada dos semanas.

|  |  |
| --- | --- |
| Duración | 4 meses \* 4 semanas = 16 semanas |
| Salario/hora | 10,00 €/hora |
| Nº horas/semana | 20 horas/semana |
| Coste final | 10,00 €/hora \* 20 horas/semana \* 16 semanas =3.200,00 € |

Tabla 4 - Coste de aprendizaje

#### Coste de soporte

Este coste sería realizado por un hipotético técnico que se ocuparía de resolver cualquier incidencia en la utilización del producto desarrollado. Al ser una aplicación que, si resulta viable, será publicada en el *Play Store* no existirá ningún técnico encargado de subsanar directamente los errores a los usuarios, sino que serán los propios operarios de dicha página los que se remitirán cualquier incidencia.

Es por esto que coste de soporte será nulo y por tanto no se tendrá en cuenta.

#### Coste de mantenimiento

En este caso será el propio alumno el encargado de realizar el mantenimiento de la aplicación por lo que esta tarea se encuentra remunerada dentro de su salario. No obstante, cabe destacar que la distribución desde esta App no incluye costes de almacenamiento de la información resultantes, es decir, las propias instituciones que compren el producto tendrán que realizar un mantenimiento propio de sus servidores.

#### Otros costes

Además, también hay que tener en cuenta otros costes como los *derivados del material mobiliario, de oficina, documentación, electricidad, Internet*.

|  |  |  |
| --- | --- | --- |
| Tipo de coste | Coste | Total |
| Infraestructura | 10,00 €/mes | 10,00 €/mes \* 4 meses = 40,00 € |
| Documentación | 36,10 € [23] | 36,10 € |
| Electricidad | 35 €/mes | 35 €/mes \* 4 = 140 € |
| Internet | 37, 90 €/mes | 37, 90 €/mes \* 4 = 151,60 € |
|  |  | 367,70 € |

Tabla 5 - Otros costes

#### Total

Finalmente, los costes globales del proyecto serán:

|  |  |
| --- | --- |
| Tipo de coste | Total |
| Coste Hardware | 101,2 € |
| Coste Software | 78,00 € |
| Coste de instalación | 0,00 € |
| Coste de aprendizaje | 3.200,00 € |
| Coste de soporte | 0,00 € |
| Coste de mantenimiento | 0,00 € |
| Otros costes | 367,70 € |
|  | 3.746,90 € |

Tabla 6 - Costes totales

#### Análisis coste-beneficio

Como ya se ha explicado, este proyecto nace de la base fundamental de otro proyecto. Este proyecto fue desarrollado con una base y un fin muy bien definido y fijado, la herramienta *QuickTest* es compatible con cualquier plataforma que cumpla el estándar *LTI*. Fruto de esta idea se desarrolló este proyecto como un complemento de resolución de cuestionarios para *Moodle.* Es por esto que este análisis de coste-beneficio se va realizar para *Moodle.*

*Moodle* cuenta actualmente con más de *100 millones de usuarios* en todo el mundo, lo que la convierte en la plataforma digital más utilizada en todo mundo por la comunidad educativa [24] [25]. Además, se encuentra registrada en *234 países*, en la que se encuentra España en el segundo puesto con *7.152 inscripciones*.

Como el principal objetivo de esta aplicación es la utilización de la misma para un entorno universitario vamos a reducir estos cálculos de inscripciones al número de universidades en España. Actualmente, el número de universidades tanto públicas como privadas es de 82.

Al ser el proyecto compatible con *Moodle* y al ser este, la plataforma digital más influyente en este sector, cabe destacar que no deberíamos de tener ningún inconveniente en vender licencias de este softwareen el *PlayStore*. Sería una suposición muy idealizada que todas estas comprarán nuestro producto por lo que realizaremos los cálculos estimando que solamente *40 universidades* deciden adquirir la licencia.

Ante este estudio preliminar, se decide vender cada licencia por 100,00 €. Este precio solamente incluirá el producto junto con la documentación necesarios para su correcta utilización, es decir, no se incluye un soporte de almacenamiento de la información generada por su utilización. Además, esta distribución de licencia se hará con carácter *comercial* [26].

Finalmente, cabe destacar que valoraremos este análisis como factible ya que únicamente distribuyendo estas 40 licencias recuperaríamos la inversión realizada.

### Estudio de viabilidad legal.

Al utilizar herramientas que se encuentran bajo licencias de software libre la distribución de esta herramienta no ocasionará ninguna problemática legal.

No obstante, la herramienta fruto de este proyecto se distribuirá con la licencia *Android License Verification Library* [27] propia de *PlayStore* y utilizada por cualquier aplicación que resida en la misma para evitar posibles pirateos.

Como ya se comentaba en la sección anterior este producto se distribuirá como *licencia comercial*, ya que todos los activos utilizados para el desarrollo del mismo poseen licencia GNU.

# Especificación de requisitios

## Introducción

A continuación, en este apartado se va a llevar a cabo a la numeración y correspondiente explicación de los requisitos funcionales junto con los diagramas de casos de uso generados a partir de estos requisitos. También, se van a citar aquellos requisitos no funcionales que cumple la aplicación.

## Objetivos generales

## Catálogo de requisitos

### Requisitos funcionales

https://es.wikipedia.org/wiki/Requisito\_funcional

#### Rol profesor

|  |  |
| --- | --- |
| Nº RF | Descripción |
| RF-P1 | El profesor deberá registrarse en QuickTest utilizando como nombre de usuario, el mismo correo que en Moodle. |
| RF-P2 | Podrá iniciar sesión una vez logeado. |
| RF-P3 | Podrá crear un nuevo cuestionario. |
| RF-P4 | Podrá editar un cuestionario. |
| RF-P5 | Podrá eliminar un cuestionario. |
| RF-P6 | Podrá duplicar un cuestionario. |
| RF-P7 | Podrá configurar los comodines |
| RF-P8 | Podrá editar las preguntas de un cuestionario. |
| RF-P9 | Podrá configurar la puntuación de cada preguntar. |
| RF-P10 | Podrá elegir la respuesta correcta de cada pregunta. |
| RF-P11 | Podrá cerrar sesión. |
| RF-P12 | Podrá cerrar sesión y olvidar. |

#### Rol alumno

|  |  |
| --- | --- |
| Nº RF | Descripción |
| RF-A1 | El alumno deberá registrarse en QuickTest utilizando como nombre de usuario, el mismo correo que en Moodle. |
| RF-A2 | El alumno podrá iniciar sesión una vez logeado. |
| RF-A3 | Podrá responder al cuestionario. |
| RF-A4 | Podrá decidir si usar los comodines. |
| RF-A5 | Por cada pregunta del cuestionario podrá guardar su respuesta. |
| RF-A6 | Por cada pregunta del cuestionario podrá anular la respuesta guardada. |
| RF-A7 | Podrá finalizar el cuestionario y acabar. |
| RF-A8 | Podrá guardar el estado de todas las preguntas y finalizar y acabar. |
| RF-A9 | Podrá cerrar sesión. |
| RF-A10 | Podrá cerrar sesión y olvidar. |
| RF-A11 | Podrá revisar sus respuestas. |
| RF-A12 | Podrá ver su calificación. |
| RF-A13 | Si el cuestionario está cerrado podrá revisarlo y ver su calificación. |

### Requisitos no funcionales

https://es.wikipedia.org/wiki/Requisito\_no\_funcional

### Restricción de requisitos

## Especificación de requisitos

Aquí D.Casos de uso para los RF

# Especificación de diseño

## Introducción

A continuación, se incluye una explicación/descripción del diseño de la aplicación Android. Para ello se dividirán los diagramas en dos enfoques: *el APIREST* utilizado para comunicarse con el proyecto de partida y la propia *app Android* que se comunicará con dicho API para la obtención y volcado de los datos.

## Diseño de datos

En este apartado se incluirán los diagramas de clases y paquetes de ambos enfoques.

### Diagrama de clases

En estos diagramas se ha decidido utilizar un criterio de color para diferenciar aquellas clases que forman parte del *APIREST,* en color azul y en amarillo aquellos que forman parte del proyecto de partida.

#### APIREST

Este paquete será el encargado de comunicarse con el *Controlador* del proyecto de partida. A continuación, se mostrarán de manera fraccionada cada una de las clases que lo forman y con qué clases del controlador del proyecto de partida se comunican.

##### Clase SolucionCuestionario

Esta clase será la encargada de realizar el control de acceso por parte de los profesores a la web *QuickTest*. Esta web provee a los profesores de una clave privada dentro de la aplicación para poder utilizarla en la sincronización de sus cuestionarios con *Moodle.* Esta clase permite registrar a un nuevo profesor o comprobar si puede iniciar sesión. Ambas lógicas se controlan mediante una petición *post*. Para llevar a cabo este cometido se comunican con el controlador llamando al método *registrarNuevoUsuario* o *login* respectivamente.

* P**ost:** petición que permitirá logearse o registrarse en *QuickTest*.

Ilustración 4 - REST - Registrarse en QuickTest

http://localhost/\_QuickTest\_TFG/app/apiREST/controlAccesoProfesor/registro

Ilustración 5 - REST - Iniciar sesión en QuickTest

http://localhost/\_QuickTest\_TFG/app/apiREST/controlAccesoProfesor/login

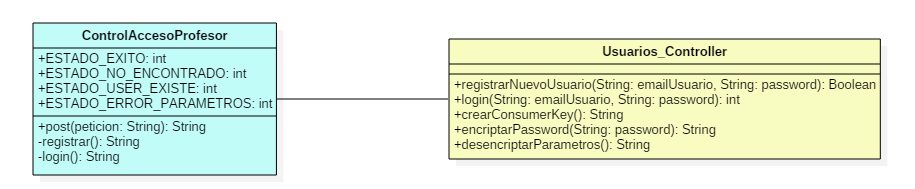


Ilustración 6 - Clase ControlAccesoProfesor

##### Clase GestionCuestionario

La lógica de esta clase será toda aquella que se encuentre relacionada con la gestión sobre los Cuestionarios: *insertar, obtener, duplicar o borrar*. Cabe destacar, que en el controlador del proyecto de partida se ha incluido un método para poder duplicar un cuestionario existente, de esta manera evitamos comunicarnos con el modelo.

Para ello se comunica con el controlador. Todas estas acciones se realizan mediante diferentes peticiones:

* **Get:** esta petición será la encargada de *mostrar* aquellos cuestionarios dada un identificador de asignatura.

Ilustración 7 - REST - Obtener cuestionario

http://localhost/\_QuickTest\_TFG/app/apiREST/gestionCuestionario/id

* **Post:** esta petición maneja dos lógicas: *duplicar* o *insertar*/*editar* un cuestionario. Para duplicar un cuestionario basta con indicar el identificador de cuestionario y para insertar o editar será necesario incluir en esta petición el resto de información propia de un cuestionario. Cabe destacar que el método insertar manejará la lógica de insertar o editar un cuestionario en función de si un campo se encuentra a verdadero o falso. Esta lógica venia dado por el proyecto de partida.

Ilustración 8 - REST - Duplicar cuestionario

http://localhost/\_QuickTest\_TFG/app/apiREST/gestionCuestionario/duplicar/id

Ilustración 9 - REST - Insertar cuestionario

http://localhost/\_QuickTest\_TFG/app/apiREST/gestionCuestionario/insertar

* **Delete:** esta petición se encarga de eliminar un determinado cuestionario dado un identificador.

Ilustración 10 - REST - Eliminar cuestionario

http://localhost/\_QuickTest\_TFG/app/apiREST/gestionCuestionario/id

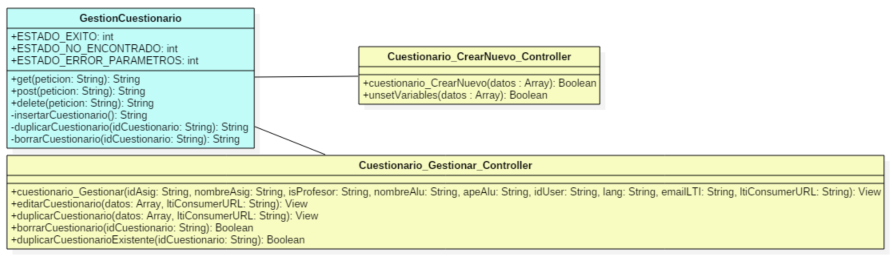


Ilustración 11 - Clase GestionCuestionario

##### Clase SoluciónCuestionario

Esta clase se encarga de la lógica a realizar durante la resolución de un cuestionario: *iniciar, finalizar o mostrar resultado.* Para ello se comunica con el controlador. Destacar que al tener que basarse en el proyecto de partida existen ciertas funcionalidades que no se encuentran en el controlador. Es por esto que será necesario comunicarse también con el modelo.

Todas estas acciones se realizan mediante peticiones *post* y son las siguiente:

* **Post:** esta petición será la encargada de manejar toda la lógica de esta clase, es decir, iniciar un cuestionario, finalizar un cuestionario y mostrar la calificación.

Ilustración 12 - REST - Mostrar resultados

http://localhost/\_QuickTest\_TFG/app/apiREST/solucionCuestionario/mostrar

http://localhost/\_QuickTest\_TFG/app/apiREST/solucionCuestionario/finalizar

Ilustración 13 - REST - Iniciar cuestionario

http://localhost/\_QuickTest\_TFG/app/apiREST/solucionCuestionario/resolver

Ilustración 14 - REST - Finalizar cuestionario

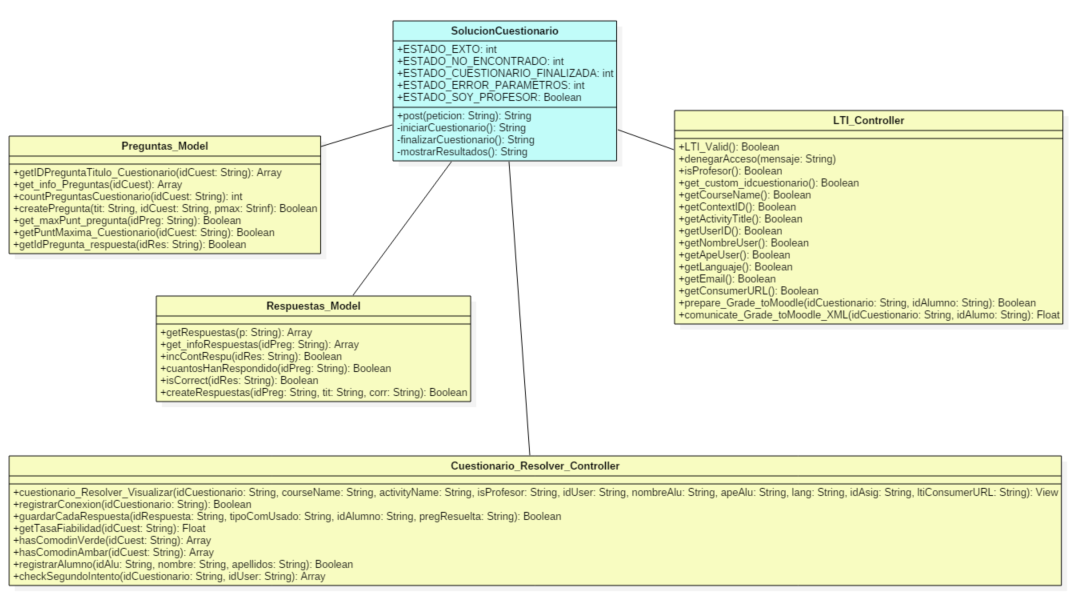


Ilustración 15 - Clase SolucionCuestionario

#### Aplicación Android

### Diagrama de paquetes

Para estos diagramas se ha decido utilizar el mismo criterio de color empleando anteriormente en el diagrama de clases.

#### APIREST

Este diagrama muestra un desglose de todos los componentes que entran en contacto en el API, como ya se ha comentado en color amarillo corresponde a la lógica dada en el proyecto de partida y en azul el APIREST desarrollado para interactuar con el controlador.

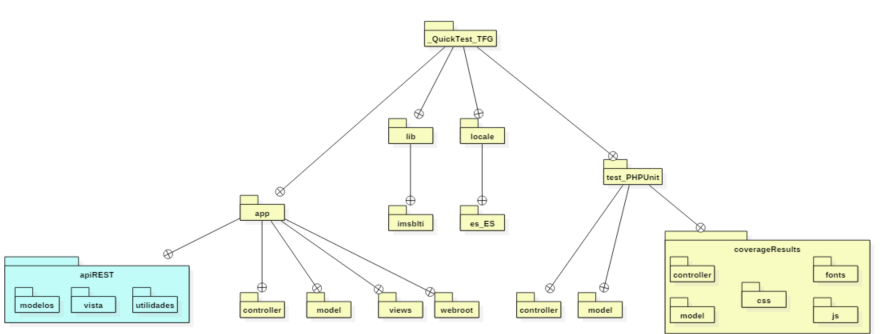


Ilustración 16 - Diagrama de paquetes - REST

## Diseño procedimental

### Diagrama de secuencias

## Diseño arquitectónico

### Diagrama de despliegue

# Manual del programador

## Introducción

## Estructura de directorios

## Manual del programador

## Compilación, instalación y ejecución del proyecto

## Pruebas del sistema

# Manual del usuario

## Introducción

## Requisitos de usuarios

## Instalación

## Manual del usuario

# Bibliografía

|  |  |
| --- | --- |
| [1] | M. Merino, «Ticbeat - API,» 19 2 2017. [En línea]. Available: http://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/. |
| [2] | «bbvaopen4u - API,» [En línea]. Available: https://bbvaopen4u.com/es/actualidad/que-es-una-api-y-que-puede-hacer-por-mi-negocio. |
| [3] | «Wikipedia - SOAP,» [En línea]. Available: https://es.wikipedia.org/wiki/Simple\_Object\_Access\_Protocol. |
| [4] | O. F. Brea, «DesarroloWeb - SOAP,» [En línea]. Available: https://desarrolloweb.com/articulos/1853.php. |
| [5] | «Wikipedia - REST,» 19 2 2017. [En línea]. Available: https://es.wikipedia.org/wiki/Transferencia\_de\_Estado\_Representacional. |
| [6] | «bbvaopen4u - REST,» [En línea]. Available: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos. |
| [7] | PHP, «PHP,» [En línea]. Available: http://php.net/manual/es/intro-whatis.php. |
| [8] | «Wikipedia-PHP,» [En línea]. Available: https://es.wikipedia.org/wiki/PHP. |
| [9] | «JSON,» [En línea]. Available: http://www.json.org/json-es.html. |
| [10] | «Wikipedia - JSON,» [En línea]. Available: https://es.wikipedia.org/wiki/JSON. |
| [11] | «Wikipedia - XAMPP,» [En línea]. Available: https://es.wikipedia.org/wiki/XAMPP. |
| [12] | «Wikipedia - GitHub,» [En línea]. Available: https://es.wikipedia.org/wiki/GitHub. |
| [13] | «Wikipedia - TortoiseSVN,» [En línea]. Available: https://es.wikipedia.org/wiki/TortoiseSVN. |
| [14] | «Wikipedia - Modelio,» [En línea]. Available: https://en.wikipedia.org/wiki/Modelio. |
| [15] | «Wikipedia - SCRUM,» [En línea]. Available: https://es.wikipedia.org/wiki/Scrum\_(desarrollo\_de\_software). |
| [16] | «PHP - Documentación,» [En línea]. Available: http://uk1.php.net/manual/es/langref.php. |
| [17] | «CakePHP,» [En línea]. Available: https://cakephp.org/. |
| [18] | «Laravel,» [En línea]. Available: https://laravel.com/. |
| [19] | «Symphony,» [En línea]. Available: https://symphony.com/. |
| [20] | «Slim,» [En línea]. Available: https://www.slimframework.com/. |
| [21] | «Wikipedia - Android Studio,» [En línea]. Available: https://es.wikipedia.org/wiki/Android\_Studio. |
| [22] | «Wikipedia - Moodle,» [En línea]. Available: https://es.wikipedia.org/wiki/Moodle. |
| [23] | «Libro de consulta,» [En línea]. Available: https://www.amazon.es/Android-Desarrollo-Aplicaciones-Smartphones-Tabletas/dp/2746092298. |
| [24] | «Lavanguardia,» 1 4 2016. [En línea]. Available: http://www.lavanguardia.com/vida/20160401/40813172673/martin-dougiamas-moodle-universitat-vict-doctor-honoris-causa-agora-barcelona-tencologia-software-libre.html. |
| [25] | «Moodle - Estadisticias,» [En línea]. Available: https://moodle.net/stats/. |
| [26] | «Licencia Comercial,» [En línea]. Available: http://es.videojuegos.wikia.com/wiki/Licencia\_comercial. |
| [27] | «Android License Verification Library,» [En línea]. Available: https://developer.android.com/google/play/licensing/index.html. |