# Aspectos relevantes del desarrollo

Cuando se afronta el desarrollo de un proyecto y dado que este va a realizarse durante varios meses, es importante analizar todas las dudas y problemas que pudieran surgir, de modo que se encuentren soluciones que se adapten en la mayor medida posible a las necesidades del proyecto.

En este apartado se van a comentar los problemas y dudas que han surgido durante el desarrollo así como las soluciones tomadas, explicando en cada caso los motivos que nos han llevado a la decisión final.

## Elección de la forma de validar el consumer key

El objetivo principal de las primeras semanas fue intentar conseguir una conexión entre Moodle y el servicio web. Para ello se empleó una librería lti, la cual se descargó y se realizó una primera investigación de la misma.

Dicha librería lo que nos iba a permitir era llevar a cabo una comparación de los consumer\_key y secret, que se establecían en la tarea creada en Moodle, con los valores consumer\_key y secret correctos que se encontraban almacenados en la base de datos. Si dichos valores coincidían, entonces la conexión entre Moodle y el servicio web podría realizarse correctamente, en caso contrario no.

Sin embargo empezaron a surgir problemas en el tema de realizar la comparación del consumer\_key, principalmente debido a que la librería utilizaba métodos de acceso a la base de datos que ya estaban obsoletos y otros trozos de código que no se conseguían entender correctamente.

Por ello, este proceso se realizará de otra forma. Para ver el proceso completo: ENLACE AL APARTADO CORRESPONDIENTE DEL DOCUMENTO ANEXOS.

## Elección del framework CakePHP

Las primeras semanas fueron empleadas para investigar acerca de Moodle, el protocolo OAuth, y sobre todo de la librería LTI que iba a permitir establecer una conexión entre Moodle y la propia aplicación o servicio web.

Tras conseguir realizar esta conexión, es cuando empezó realmente la programación del código durante aproximadamente 2-3 semanas. Durante estas primeras semanas no se habló de utilizar ningún patrón para realizar la aplicación, sin embargo posteriormente en una reunión el tutor propuso que usara el patrón MVC (Modelo-Vista-Controlador) y como tal también me indicó que podría usar algún framework que me facilitara la implementación de dicho patrón y es donde se propuso usar el framework CakePHP.

Debido a esta propuesta, me dediqué durante varios días a investigar, realizar tutoriales y pequeñas aplicaciones que se basaban en el framework.

Finalmente tras haber adquirido un pequeño conocimiento inicial acerca del framework y tras observar que me podría facilitar en numerosos aspectos la implementación del patrón MVC, decidí adaptar todo el código que tenía ya realizado para que utilizara CakePHP.

## Elección de Maven

Dado que la aplicación consistía principalmente en un proceso de autocorrección de las prácticas subidas por los alumnos, a las cuales iban a aplicarse los test subidos por el profesor, era necesaria una herramienta que permitiría gestionar esta implementación. Por ello como opciones posibles había dos: Ant y Maven.

La decisión de elegir Maven frente a Ant fue principalmente porque disponía de más conocimiento sobre Maven, gracias a que en la asignatura “Sistemas distribuidos” impartida en la carrera, se realizaron varias aplicaciones que utilizaban dicha herramienta de software.

También otro factor que me ayudó a decantarme por Maven fue que este es más actual que Ant.

## Elección de Bootstrap

Las primeras semanas se dejó un poco de lado el diseño de la interfaz, centrándonos principalmente en la creación de los controladores, tablas de la base de datos…

En una reunión, el tutor puso como tarea comenzar a realizar pequeños diseños de interfaz. Y es por tanto cuando se planteó la idea de usar Bootstrap, de la cual ya me sonaba de haberlo leído en algunos artículos de internet pero sin embargo que nunca la había utilizado sobre una aplicación.

Tras leer información en la web oficial y realizar pequeños ejemplos, tomé la decisión de utilizar dicha librería ya que me pareció que era relativamente sencilla de utilizar y que me podría ayudar a realizar el diseño de la interfaz de la aplicación.

## Elección de los plugins para los reportes

Una duda a resolver, fue la de qué reportes se le iban a mostrar a los alumnos cada vez que realizaban un intento de subida de práctica.

Por ello los plugins que se seleccionaron para generar estos reportes fueron los siguientes:

* **JAVANCSS**: Ofrece información sobre métricas (complejidad ciclomática, número de líneas…).
* **JDEPEND**: Ofrece información sobre métricas a nivel de paquetes (abstractness, inestability, afferent couplings…).
* **PMD**: Ofrece información acerca de violaciones de código cometidas (variables sin usar, métodos privados sin usar…).
* **FINDBUGS**: Al igual que el PMD ofrece información acerca de violaciones de código, pero detecta violaciones que el PMD no detecta (por ejemplo, bucles infinitos).

## Elección de la forma de añadir los plugins

Dado que los plugins hay que añadirles dentro del fichero pom.xml del arquetipo Maven, había que elegir la forma con la que se iba a realizar este proceso.

Por ello la decisión tomada fue la de crear un método “editarPomArquetipoMaven” en el controlador “FicherosXmlController”, de forma que cuando el profesor suba su test por primera vez y se cree el arquetipo Maven, se llamara a dicho método el cual se va a encargar de:

1. Abrir el fichero pom.xml con una función propia de php.
2. Crear las etiquetas correspondientes para cada uno de los plugins.
3. Añadir dichas etiquetas al pom.xml y guardarlo.

Por lo tanto si en un futuro se quisiera añadir un nuevo plugin, sería necesario editar el método mencionado.

## Elección de la herramienta para los plagios

Dado que se le quería añadir una opción al profesor para que pudiera comprobar si había plagios entre las prácticas subidas por los alumnos, era necesaria una herramienta para ello.

Por ello la opción que se comenzó a utilizar al principio fue la del plugin CPD de Maven. Sin embargo dado que los reportes que generaba no eran muy claros y además tampoco te mostraba indicadores de los porcentajes de plagios, se optó por buscar otra herramienta.

Por ello se investigó acerca de MOSS y JPLAG, que finalmente se acabó utilizando JPLAG ya que era fácil de implementar y además los reportes eran claros y sencillos.