UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Implementación de Aplicaciones Sotfware

Para el Área de Relaciones Internacionales del Campus d'Alcoi (2/2)

MEMORIA PRESENTADA POR:

Román Bas Bas

Tutor:

Pau Micó Tormos

GRADO DE *INGENIERÍA INFORMÁTICA*

Convocatoria de defensa: *07/2017*

Resumen

En el área de Relaciones Internacionales (RRII) surgen una serie de necesidades que se pueden cubrir mediante el desarrollo de aplicaciones software a medida. La siguiente propuesta de Trabajo Final de Grado (TFG) consiste en la racionalización del plan de desarrollo y explotación de todas estas aplicaciones. Los objetivos del plan de racionalización son:

- Desde el punto de vista del usuario
    - Facilitar la explotación de todas las aplicaciones centralizadas a través de un portal único.
    - Ofrecer una interfaz de usuario amigable e intuitivo que facilite la navegación por la aplicación y la explotación de sus contenidos.
- Desde el punto de vista del administrador
    - Resolver el problema del mantenimiento de las aplicaciones desarrollando un sistema de gestión de contenidos (Content Management System, CMS).
    - Incluir la posibilidad de incrementar los contenidos de la aplicación, manteniendo la coherencia del sistema.

Además, el plan incluye el desarrollo de algunas de las aplicaciones a medida, a saber:

- **Espacio compartido Erasmus:** donde los usuarios podrán compartir sus apuntes en otros idiomas, de esta manera se evita la pérdida de documentos de tipo traducciones y demás, así como ayudar a los nuevos alumnos Erasmus que puedan tener problemas con el idioma al principio.
- **Tándem Erasmus:** donde los usuarios podrán organizar reuniones para interactuar con otros usuarios y así aprender idiomas. De esta forma, se pretende conseguir una mayor integración de los nuevos alumnos Erasmus en la comunidad universitaria.
- **Sistema de tickets:** donde el usuario podrá crear incidencias sobre el incorrecto funcionamiento de ciertos apartados de la aplicación. Una vez reportada la incidencia, el técnico verificará si realmente existe un mal funcionamiento, y en caso de haberlo, se arreglará y se notificará, con la resolución, al usuario que creó el ticket.

Actualmente el Área de Sistemas de la Información y Comunicaciones de la UPV ofrece el servicio avanzado de web (consola PLESK) que incluye CMS, hosting web, FTP y base de datos, entre otros. Por ello, y como principal requerimiento técnico de este TFG se exige el desarrollo de todas las aplicaciones a partir de una instancia de la consola de PLESK, para facilitar el posterior mantenimiento de la aplicación por parte del ASIC del Campus d'Alcoi.

Dirección

Pau Micó

Índice

Índice de figuras

Índice de tablas

1. Introducción

Desde el **Área de Relaciones Internacionales**, surgen las necesidades propias de cualquier entidad de prestación de servicios. Actualmente dichas necesidades no están siendo cubiertas por las propias herramientas que aporta la **Universidad Politécnica de Valencia - Campus de Alcoy**, ya que muchas tareas son de carácter específico y se realizan con herramientas de carácter general que no logran cubrir la tarea requerida, o si lo logran, pero no de la mejor forma.

De este modo se manifiesta la necesidad de llevar a cabo un plan de estudio de las tareas que se realizan en el área, con el fin de analizar qué herramientas serían las más adecuadas para optimizar el trabajo y si es necesario el desarrollo de nuevo software para facilitar dichas tareas. De las tareas analizadas destacan:

- Publicación de noticias en MicroWebs.
- Generación de documentación y su posterior publicación en MicroWebs.
- Envío de correo electrónicos a diversas Universidades colaborativas con información sobre programas ERASMUS.
- Publicación de destinos ERASMUS para difusión entre alumnado.
- Creación de Tándem de Idiomas para el uso por parte de los alumnos.

1.1 Antecedentes

Como se ha mencionado en el punto anterior, únicamente se dispone de una serie de herramientas las cuales no logran solucionar los problemas que acontecen en el día a día. Es posible destacar entre ellas las siguientes:

- **Gestor de MicroWebs de la Universidad Politécnica de Valencia**. Es un gestor de Contenido *CMS* muy limitado, que impide la elaboración de páginas webs interactivas para los usuarios. En la *figura 1* se observa el aspecto que tiene dicho gestor, destacando las limitaciones en cuanto a publicación de páginas con cierto nivel de carga *CSS* o *Javascript*.

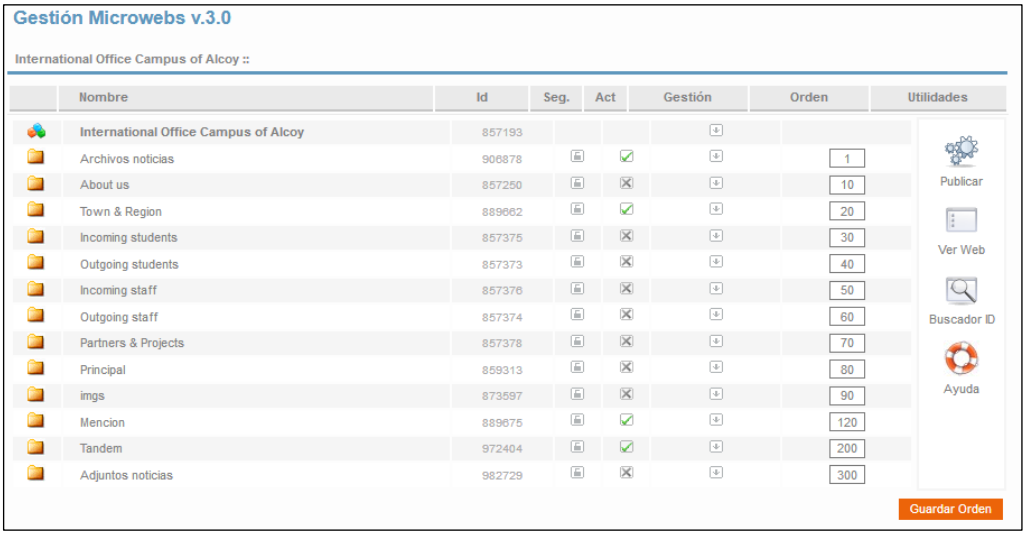


Figura 1- Panel de edición de MicroWebs

Otro de los problemas que tiene este gestor, es la necesidad de emplear *Java* para cualquier tipo de edición. Este problema se agrava debido a que cada vez menos navegadores permiten ejecutar contenido Java. (*Figura 2*).

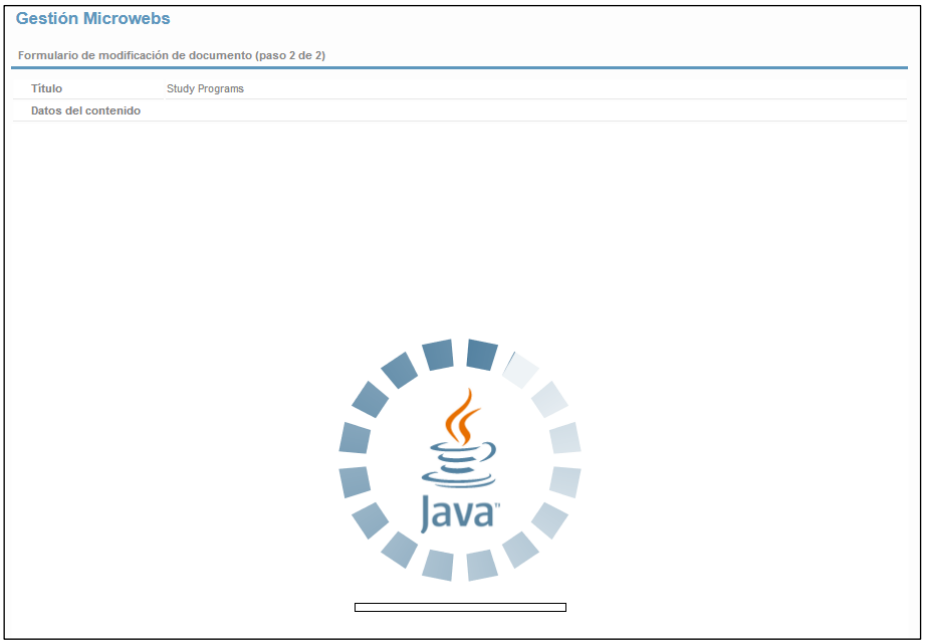


Figura 2 - Carga de contenido Java

Para editar cualquier noticia o edición de un contenido nuevo, existen dos modos de hacerlo, o bien editando el texto como si de un documento se tratara (*Figura 3*) o bien utilizando código *HTML* y *CSS*. El inconveniente de este último es que no se muestra ningún tipo de errores en el código, el gestor simplemente omite la parte que contiene un fallo y no la muestra.

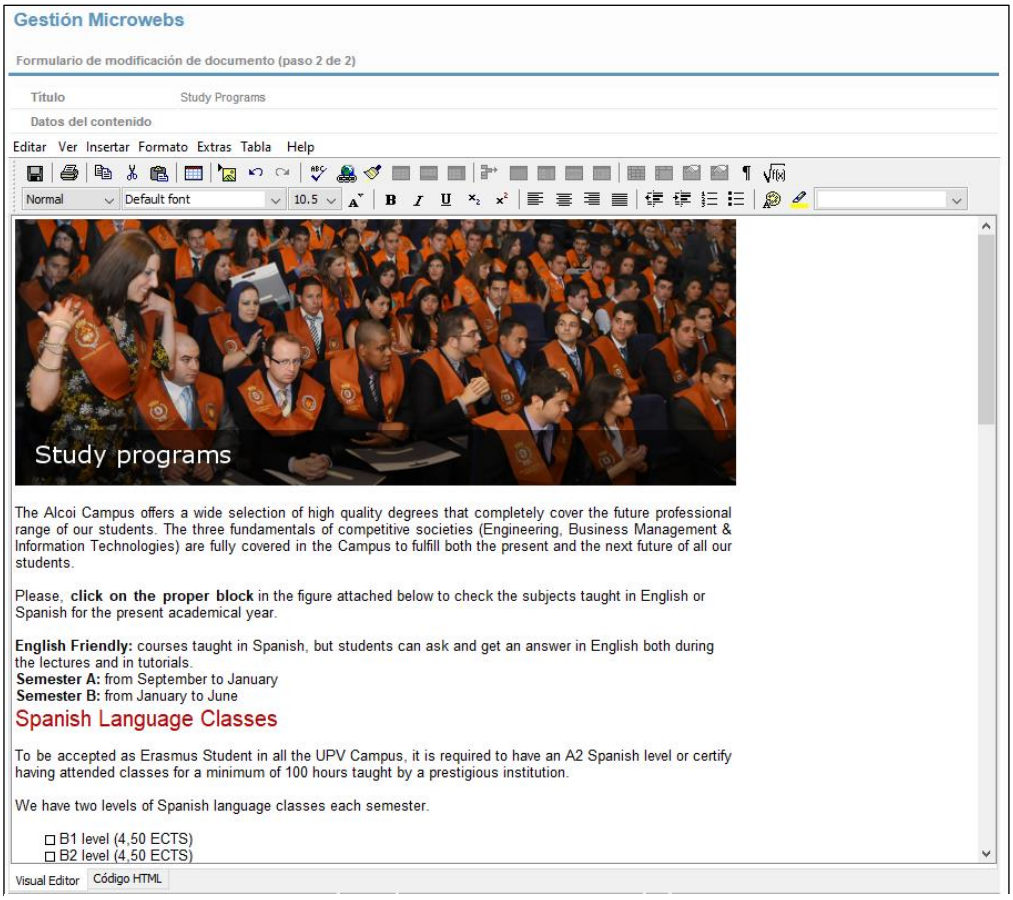


Figura 3 - Edición de un archivo

Pero a pesar de todas las limitaciones que tiene el gestor para la edición, existe otra y es a la hora de publicar los nuevos contenidos. Una vez creados los nuevos artículos, es necesario publicarlos para que los cambios se hagan visibles, el gestor de *MicroWebs* en lugar de publicar únicamente los últimos añadidos, publica todo de nuevo, tanto lo nuevo como lo viejo. De este modo, una vez realizando cualquier cambio en la web, por muy pequeño que sea, se necesitará volver a subir toda la web (*Figura 4*). Esto a la larga es un serio problema ya que cada vez hay más documentos y archivos lo que se traduce en mayor tiempo necesario para realizar una publicación.

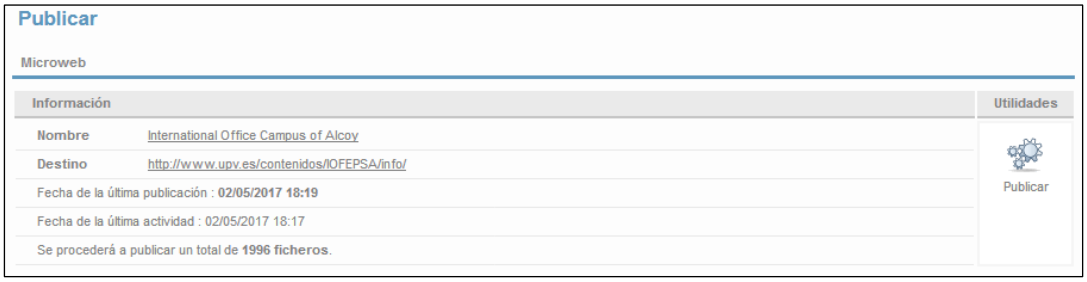


Figura 4 - Publicación del contenido

- **PowerPoints y PDFs** como herramienta para transmitir información a los alumnos. Actualmente, son los únicos medios junto con las MicroWebs para transmitir información a los alumnos. Este material debe rehacerse por cada modificación y posteriormente publicarse en la MicroWeb correspondiente. La *figura 5* muestra un ejemplo de un documento PowerPoint, en este caso de la publicación de convocatorias Erasmus. Dicho documento debe rehacerse por cada nueva publicación o eliminación de convenios con una universidad colaboradora.

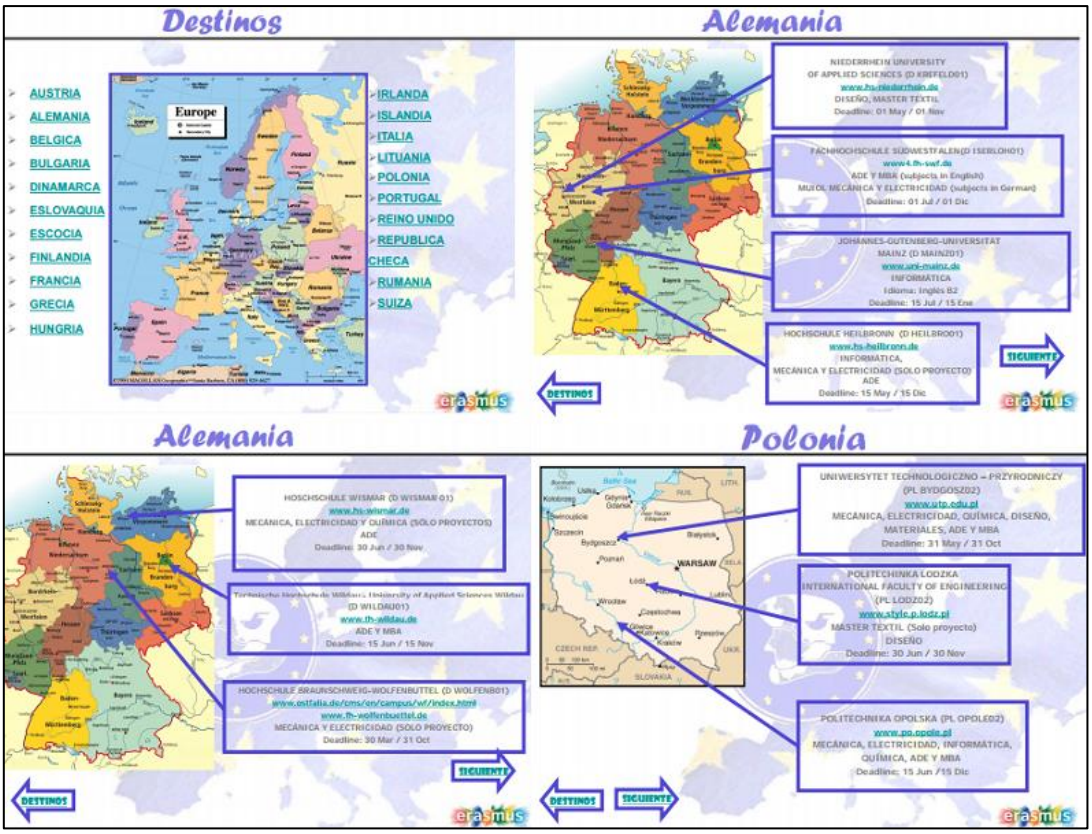


Figura 5 – Ejemplo PowerPoint de convocatorias Erasmus

- **Outlook** como gestor de correos electrónicos. No permite tener una lista de contactos por Universidad, aunque sí el envío masivo de correos electrónicos. Este proceso es tedioso, ya que actualmente no se cuenta con una base de datos centralizada de todos los contactos con las universidades colaboradoras. En la *figura 6* se observa cómo se redacta actualmente un correo.

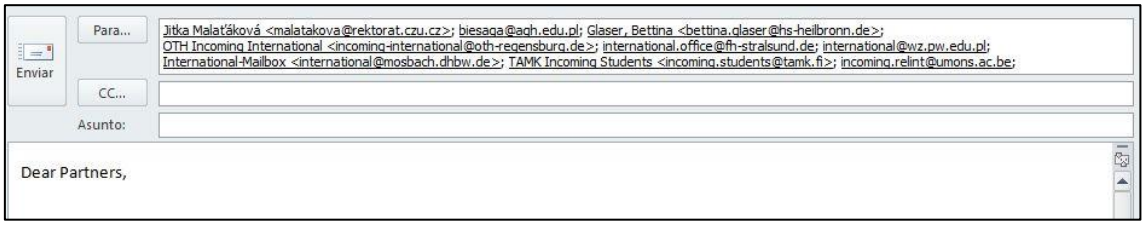


Figura 6 – Ejemplo de envío de correo

1.2 Objetivos

Por los motivos anteriormente citados, se persigue el desarrollo de un conjunto de aplicaciones que estando centralizadas en un único portal y bajo un único cuadro de mandos (CMS), faciliten las tareas diarias a los trabajadores y, por otro lado, permitan a los alumnos obtener información de forma más rápida, veraz y sencilla. Entre estas nuevas herramientas destacar:

- **Espacio compartido Erasmus:** donde los usuarios podrán compartir sus apuntes en otros idiomas, de esta manera se evita la pérdida de documentos de tipo traducciones y demás, así como ayudar a los nuevos alumnos Erasmus que puedan tener problemas con el idioma al principio.
- **Tándem Erasmus:** donde los usuarios podrán organizar quedadas para interactuar con otros usuarios y así aprender idiomas. De esta forma, se pretende conseguir una mayor integración de los nuevos alumnos Erasmus en la comunidad universitaria.
- **Sistema de tickets:** donde el usuario podrá crear incidencias sobre el incorrecto funcionamiento de ciertos apartados de la aplicación. Una vez reportada la incidencia, el técnico verificará si realmente existe un malfuncionamiento, y en caso de haberlo, se arreglará y se notificará, con la resolución, al usuario que creó el ticket.

1.3 Requerimientos

Los requerimientos para la nueva herramienta a desarrollar se dividen en dos, puesto que se va a tener dos entornos: **desarrollo** y **producción**. Por ello a continuación se detallan los requerimientos necesarios para cada entorno:

1.3.1 Entorno de desarrollo

Para el entorno de **desarrollo** se ha utilizado la implementación ***WAMP*** (*Windows Apache Mysql Php*) para simulación de entornos de producción web. Se puede encontrar la última versión de esta herramienta en el siguiente enlace: http://www.wampserver.com/en/

En cuanto al Software necesario para el desarrollo del código, se ha empleado ***PHPStorm*** de ***IntelliJ*** el cual se puede descargar desde: https://www.jetbrains.com/

En cuantos a los requerimientos de la plataforma Hardware donde desarrollar el proyecto, destacar que debe cumplir como mínimo las siguientes características:

- 1 GB RAM mínimo, 2 GB RAM recomendado
- 300 MB espacio en disco + 1 GB de caché
- 1024x768 resolución de pantalla mínima

Nota**: Estos requerimientos son específicos para el Software ***PHPStorm, ya que será quién más recursos consuma en la máquina de desarrollo.

1.3.2 Entorno de producción

Requisitos Hardware mínimos:

- 1 core 2,4 Ghz
- 2 GB Ram
- HDD 10 GB

Requisitos Software:

- Debian 9
- Base de datos MySQL
- PHP

1.4 Población objetivo

El proyecto está dirigido principalmente al departamento Internacional de la Universidad Politécnica de Valencia, Campus de Alcoy, así como a los alumnos de Erasmus y alumnos que deseen utilizar la aplicación web.

2. Anteproyecto

2.1 Planificación

Todo proyecto debe partir de una buena planificación para evitar sobre costes tanto en tiempo, material y recursos humanos, por ello en la *figura 7* se expone el diagrama de ***Gantt*** donde se recogen las principales actividades que se llevarán a cabo durante el transcurso del proyecto. Destacar que únicamente se muestra las columnas de datos, las columnas de tiempo se han omitido por ser éstas demasiado extensas. En el ***Anexo 8.2*** se añade completo.

Id.	*Nombre de tarea*	*Comienzo*	*Fin*	*Duración*	*% completado*
1	Análisis de requerimientos	03/04/2017	03/04/2017	1d	100%
2	Definición de objetivos	04/04/2017	04/04/2017	1d	100%
3	Planteamiento de funcionalidad y modelos	05/04/2017	06/04/2017	1d 4h	100%
4	Diseño de Base de Datos	06/04/2017	07/04/2017	1d 4h	100%
5	División de módulos y reparto tareas	10/04/2017	11/04/2017	1d 4h	100%
6	Maquetación interfaz gráfica	11/04/2017	14/04/2017	3d 4h	100%
7	Preparación entorno de desarrollo	17/04/2017	18/04/2017	2d	100%
8	**Desarrollo del proyecto**	**19/04/2017**	**19/05/2017**	**23d**	**96,41%**
9	Módulos Aplicación	19/04/2017	19/05/2017	23d	95%
10	Test y pruebas	24/04/2017	25/04/2017	2d	100%
11	Test y pruebas	04/05/2017	05/05/2017	2d	100%
12	Test y pruebas	15/05/2017	19/05/2017	5d	100%
13	Documentación	22/05/2017	26/05/2017	5d	100%
14	Puesta en producción	29/05/2017	30/05/2017	2d	95%
15	Test finales	31/05/2017	01/06/2017	2d	95%

Figura 7 - Diagrama de Gantt

2.2 Evaluación de los riesgos

Antes de comenzar a elaborar un nuevo proyecto, existen ciertos riesgos que han de ser evaluados y que se deben de tener en cuenta. Así pues, es posible distinguir tres tipos de riesgos dependiendo a lo que estos afecten:

- **Riegos del proyecto:** un nuevo proyecto siempre supone un gasto, por este motivo es necesario evaluar la necesidad del proyecto para comprobar si será funcional y utilizado por los usuarios objetivo.
- **Riesgos técnicos:** fallos del servidor, pérdidas de datos, etc. son algunos de los riesgos que hay que evitar para el correcto funcionamiento del servicio.
- **Riesgos del servicio:** Dentro de esta categoría es posible distinguir algunas amenazas como:
    - Pérdida de interés por parte de los usuarios ante el servicio.
    - Personal insuficientemente cualificado para el mantenimiento del servicio.

2.3 Estado del arte

Cómo la herramienta a desarrollar cumple con los requisitos para ser enmarcada dentro de un **CMS** (*Content Management System*) se ha estudiado el estado del arte actual para dichos sistemas.

- **WordPress**
- **Joombla**
- **Drupal**
- **Blogger**

Los sistemas nombrados anteriormente se podrían utilizar para la creación y edición de nuevo contenido, de una manera bastante ordenada y clara. Pero a pesar de sus ventajas, tiene ciertas limitaciones y estas son a la hora de personalizar el propio contenido y tener el control sobre todas las acciones que se realizan en la propia aplicación. De este modo es necesario otro tipo de software, por lo tanto, se estudia la posibilidad de utilización de *Frameworks* como herramienta de trabajo para la generación de aplicaciones de una forma más rápida y siguiendo un standard.

Richard Helm, Ralph Johnson y John Vlissides [1] definen *Framework* cómo:

*"Un **framework** es un conjunto de clases cooperantes que constituyen un diseño reutilizable para una clase específica de software… El framework determina la arquitectura de nuestra aplicación. Definirá la estructura general, su partición en clases y objetos, las responsabilidades clave, cómo colaboran las clases y objetos y el hilo de control. Un framework predefine estos parámetros de diseño de manera que el diseñador o el programador de la aplicación puedan centrarse en las particularidades de dicha aplicación."*

Después de descartar los CMS más comerciales, y una vez presentado el concepto de *framework*, se exponen las alternativas estudiadas cuyo nivel de personalización y edición de código es superior a los anteriores, lo que permite realizar aplicaciones más específicas caracterizadas por ser **DRY**, (*Don't Repeat Yourself*). Dentro de este sector y dependiendo del lenguaje de programación en el que están basados, cabe destacar:

- **Symfony (php)**
- **Laravel (php)**
- **Django (python)**

Estas herramientas permiten al programador, tener el control absoluto sobre toda la aplicación web, permitiendo de este modo realizar controles de errores personalizados y hacer que el sistema responda de manera adecuada a cualquier tipo de evento.

2.4 Estudio de propuestas

De las propuestas anteriores se ha decidido analizar tres de ellas, cada una representativa dentro de su campo de trabajo.

2.4.1 WordPress

Es un potente **CMS**, con una gran cantidad de Plugins o complementos que añaden nuevas funcionalidades. Se pueden editar nuevos Plugins para necesidades concretas pero su dificultad o curva de aprendizaje hace que sea inviable para proyectos como en el que pretende implementar.

Web Oficial: *"WordPress started in 2003 with a single bit of code to enhance the typography of everyday writing and with fewer users than you can count on your fingers and toes. Since then it has grown to be the largest self-hosted blogging tool in the world, used on millions of sites and seen by tens of millions of people every day.*

WordPress is completely customizable and can be used for almost anything. There is also a service called WordPress.com which lets you get started with a new and free WordPress-based blog in seconds, but varies in several ways and is less flexible than the WordPress you download and install yourself."

2.4.2 Django

Wikipedia: *"Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–vista–controlador. La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos."*

Web Oficial: *"Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. It is characterized by:*

- ***Ridiculously fast**. Django was designed to help developers take applications from concept to completion as quickly as possible.*
- ***Reassuringly secure.** Django takes security seriously and helps developers avoid many common security mistakes.*
- ***Exceedingly scalable**. Some of the busiest sites on the Web leverage Django's ability to quickly and flexibly scale."*

Dispone de diversos plugins para extender su funcionalidad y además dispone de una amplia documentación actualizada que se puede consultar en su web: https://www.djangoproject.com/

2.4.3 Symfony

Wikipedia: *"Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web."*

Web Oficial: "Symfony is a set of PHP Components, a Web Application framework, a Philosophy, and a Community — all working together in harmony.

- ***Symfony Framework.** The leading PHP framework to create websites and web applications. Built on top of the Symfony Components.*
- ***Symfony Components**. A set of decoupled and reusable components on which the best PHP applications are built, such as Drupal, phpBB, and eZ Publish.*
- ***Symfony Community.** A huge community of Symfony fans committed to take PHP to the next level.*
- ***Symfony Philosophy.** Embracing and promoting professionalism, best practices, standardization and interoperability of applications."*

Además, cuenta con una gran cantidad de plugins, como ocurre con *Django* lo que permite extender muchas de sus funcionalidades sin tener que ser programadas desde cero. Nuevamente dispone de una web propia donde se documentan todas las nuevas versiones: https://symfony.com/

2.4.4 Elección final

Finalmente, después de las valoraciones anterior, se ha optado por seleccionar *Symfony* como *framework* de desarrollo mediante el cual realizar el proyecto que acontece.

Para justificar dicha elección a continuación se presenta una gráfica con la curva de aprendizaje de cada una de la propuesta estudiadas:

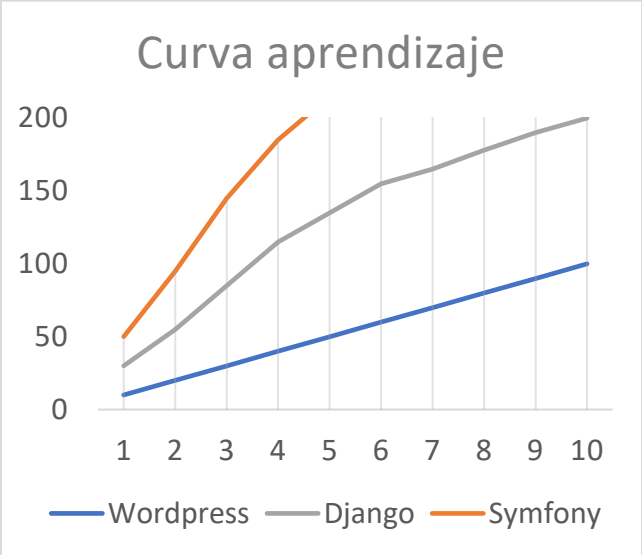


Figura 8 - Curva de aprendizaje

2.5 Elección del sistema gestor de bases de datos

Al realizar un proyecto de una aplicación web, es necesario evaluar el tipo de datos que se va a guardar y la manera en que se trabajará con dichos datos. Este es un paso muy importante ya que, en función del sistema gestor de base de datos elegido, la base de datos será más o menos escalable, tendrá unas funcionalidades u otras y el tiempo de respuesta será mayor o menor. Para ello es necesario evaluar los pros y los contras de los sistemas más utilizados:

- **MySQL:** Sistema gestor de base de datos open-source.
    - Ventajas:
        - Permite crear y configurar usuarios, asignando permisos a cada uno de ellos.
        - Facilidad de exportación e importación de datos.
        - Alta velocidad al realizar operaciones.
    - Desventajas:
        - Muchas de sus utilidades no están documentadas.
- **PostgreSQL:** Sistema de gestión de base de datos relacional bajo licencia BSD.
    - Ventajas
        - Permite realizar diferentes tareas sobre una misma tabla sin que haya bloqueos.
        - Maneja mucha información sin afectar el tiempo de respuesta.
        - Escalable y confiable.
    - Desventajas:
        - Elevado consumo de recursos del sistema.
        - Ciertas sintaxis no son nada intuitivas.

- **SQL Server:** Sistema de gestión de base de datos desarrollado por Microsoft.
    - Ventajas:
        - Soporta procedimientos almacenados.
        - Permite trabajar en modo cliente-servidor, donde los datos se alojan en un servidor y los clientes solo acceden a la información.
        - Escalabilidad, estabilidad y seguridad.
    - Desventajas:
        - Licencias de pago.

- **MongoDB**: Es un modelo no relacional que utiliza bases de datos NoSQL.
    - Ventajas:
        - En lugar de trabajar con registros, la información se guarda en documentos de tipo BSON, esto facilita el intercambio de datos.
        - Sistema escalable que permite a los esquemas cambiar rápidamente según la necesidad.
    - Desventajas:
        - No soportan transacciones por lo que no se garantiza la integridad y durabilidad de los datos.

Finalmente se ha decidido utilizar *MySQL* debido a sus licencias gratuitas y además de que se adapta perfectamente a las funciones que se van a realizar en la aplicación web.

2.6 Justificación

Después del estudio de las diversas soluciones existentes en el mercado, y analizados sus pros y contras los cuales se resumen en la siguiente tabla (*Tabla 1*), se ha decidido por la utilización de *Symfony* como *framework* de desarrollo para todo el conjunto de nuevas aplicaciones del **Área de Relaciones Internacionales**.

	Wordpress	Symfony	Django
Fácil implementación en servidores	SI	SI	NO
Nivel de personalización	MEDIO	ALTO	ALTO
OpenSource	SI	SI	SI
Curva de aprendizaje	ALTA	MEDIA	MEDIA

Tabla 1 - Comparativa entre diversos Frameworks propuestos

2.6.1 Estimación de recursos

Una vez finalizado el proyecto, se prevé que el mantenimiento de este y los futuros desarrollos o mejoras que puedan surgir se realicen a cargo del departamento de ASIC. Es por esto que se realizará el despliegue del sistema en un servidor llamado PLESK (en los puntos posteriores se explicará con mayor detalle en que consiste, así como las funciones que ofrece y como se realizará dicho despliegue). Ya que se realiza el estudio, es interesante comprobar a qué precios es posible realizar la contratación de otro servidor de producción.

2.6.2 Impacto económico

Para la actual configuración seleccionada para la implementación del nuevo sistema de desarrollo de aplicaciones para el **Área de Relaciones Internacionales** se contempla el siguiente impacto económico (*tabla 2*):

Producto	Cantidad/Duración	Precio	Total
Licencia **IDE PHPStorm**	1 año	199,00 €/year	199,00 €
*Servidor de Producción	12 meses	3.69 €/month	44,28 €
Licencia **MySQL**	1	0 €	0 €
Licencia **Symfony**	1	0 €	0 €
TOTAL			**243.28 €**

Tabla 2 - Impacto económico de la propuesta seleccionada

*Precio del servidor en la empresa OVH en el plan de alojamiento: SSD a fecha 08/05/2015

**Importante: Destacar que debido a que la empresa *JetBrains* ofrece una cuenta gratuita de estudiantes, el precio de la licencia por el *IDE* quedará descontado en la versión de desarrollo. Una vez la aplicación pase a producción deberá adquirirse una licencia apropiada para la explotación de aplicaciones desarrolladas bajo este *IDE*.

**Importante: Debido a que el departamento de ASIC de la Universidad dispone de acceso gratuito a los servidores de PLESK, no supondrá ningún coste adicional al departamento de Relaciones Internacionales.

3. Implementación

3.1 Entorno de desarrollo

Como se ha mencionado en los puntos anteriores, se ha optado por *PHP* como lenguaje de programación y como *Symfony* como *framework* de trabajo. Estas decisiones desembocan en la decisión de seleccionar un *IDE* (*Integrated Development Environment*) a emplear. Esta no es una decisión trivial, ya que una mala elección de un entorno de desarrollo puede lastrar todo un proyecto y con una decisión anticipada y acertada se puede lograr una integración de todo el equipo de desarrollo bajo un mismo *IDE* evitando errores por ejemplo de compatibilidad. Por estos motivos, se ha optado por emplear *PhpStorm* de *JetBrains*, ya que es uno de los entornos más potentes en cuanto a soporte y mantenimiento.

En cuanto a la simulación del entorno de producción, se ha elegido una instalación de sistema *WAMP* (*Windows - Apache - MySQL - PHP*) que se adapta perfectamente a las características que tendrá el servidor de producción.

*Nota: Además con la instalación de *WAMP* se cubre la necesidad de instalar PHP de forma independiente, ya que este requisito será necesario para la creación de proyectos *Symfony* como se verá en el punto 3.1.2 Instalación y configuración de *PhpStorm*.

3.1.1 Instalación de WAMP

La instalación de *WAMP* se realiza con la descarga del ejecutable desde su página web, donde se deberá elegir la opción que mejor se adapte a nuestro equipo (x64 o x86): http://www.wampserver.com/en/

Una vez descargado, se procederá a su instalación, siguiendo los pasos que el asistente nos indica.

3.1.2 Instalación y configuración de PhpStorm

A continuación, se procede a detallar cómo se ha realizado la instalación del *IDE* de *PhpStorm* paso a paso en sistemas *Windows 10* y una breve introducción a la creación de un nuevo proyecto *Symfony*.

Nota: Como se ha comentado anteriormente, este software requiere una licencia de pago, pero los estudiantes pueden obtener una de forma gratuita. Para la descarga del instalador, se debe acceder al siguiente enlace y elegir la distribución según el sistema operativo, en este caso Windows: https://www.jetbrains.com/phpstorm/download/

3.1.2.1 Instalación

Lo primero es elegir la ubicación de la instalación. Una vez elegida, es necesario seleccionar que tipo de extensiones serán asociadas al *IDE*.

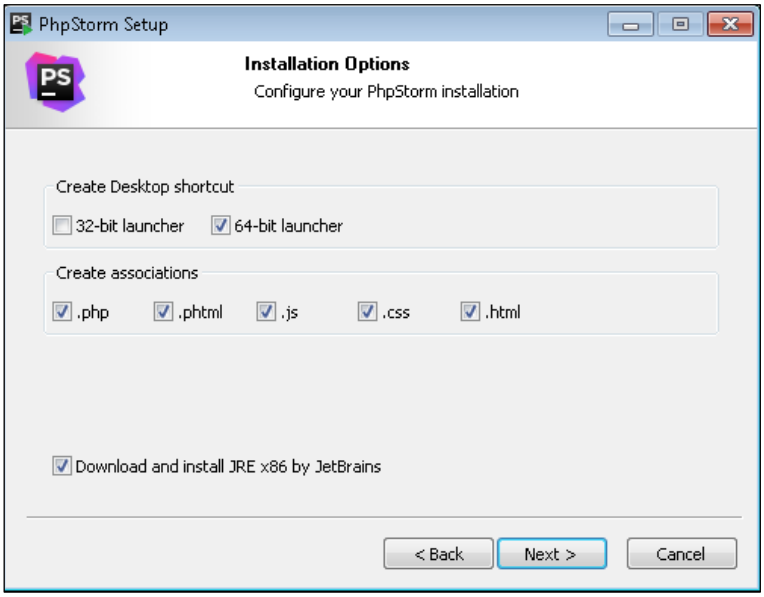


Figura 9 - Instalación de PhpStorm

A continuación, comenzará la instalación del software. Una vez finalizada, al ejecutar el programa, será necesario aceptar los acuerdos de uso. Y seguidamente será necesario introducir los datos de la cuenta del usuario de *PhpStorm*. Es posible obtener una cuenta desde el siguiente enlace:

https://account.jetbrains.com/login

Una vez se haya realizado la autenticación de la cuenta del usuario correctamente, el software permite realizar una personalización del aspecto visual, esto permite al desarrollador sentirse más cómodo con el *IDE*.

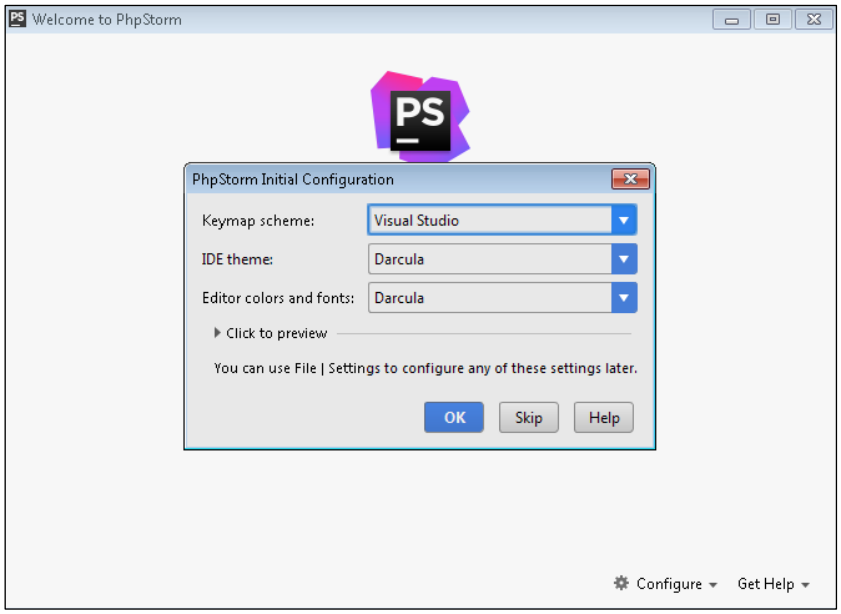


Figura 10 - Personalización de PhpStorm

3.1.2.2 Creación del proyecto

Un requisito para poder crear un proyecto de *Symfony* es tener instalado el plugin necesario. Para ello desde la pantalla inicial de *PhpStorm*, es necesario acceder al menú de configuración, subapartado de Complementos. Y ahí buscar *Symfony Plugin*. Si es la primera vez que se instala algún complemento, es posible que aparezca algún mensaje en el que se advierte de que el complemento de *Symfony* requiere de la instalación de otros complementos para su correcto funcionamiento, en cuyo caso será necesario instalarlos.

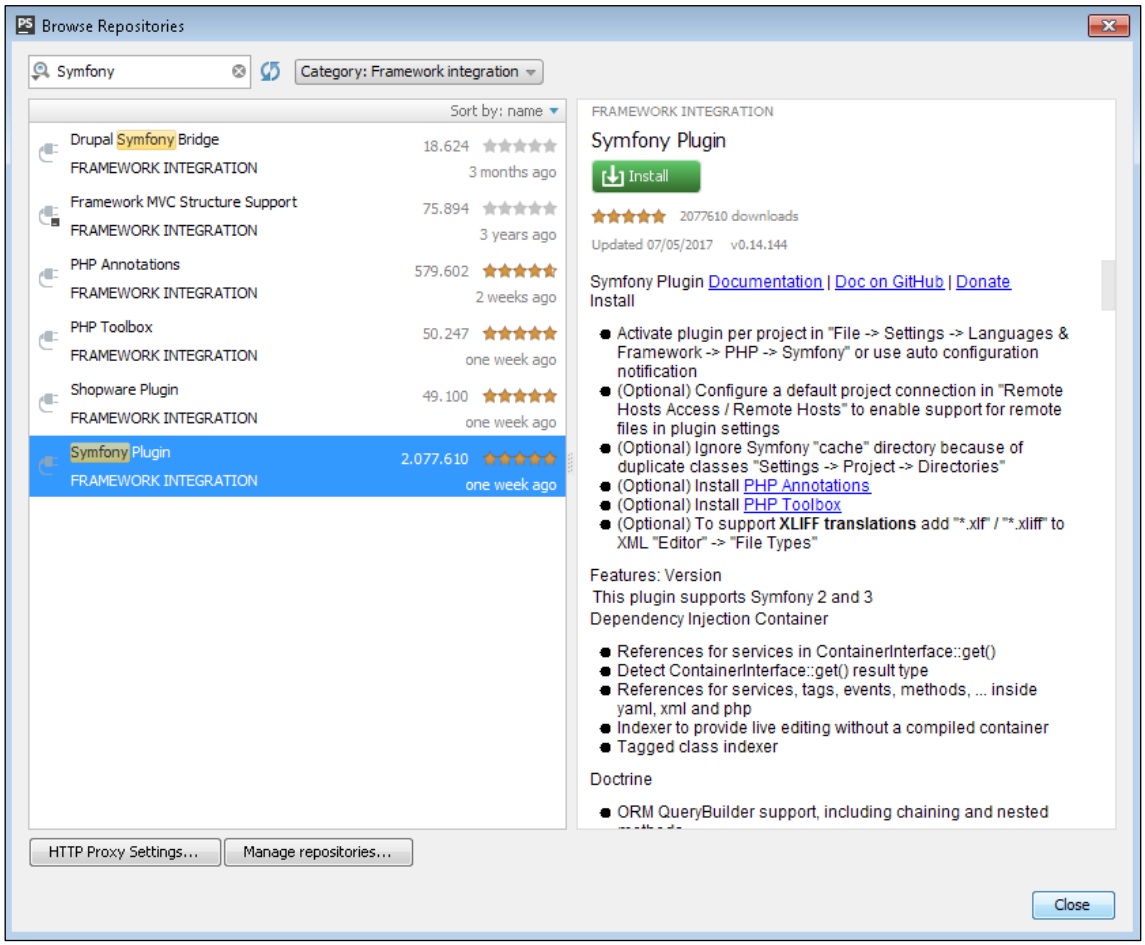


Figura 11 - Instalación de Plugins necesarios

Finalizados los pasos anteriores y previamente a la creación de un proyecto nuevo, será necesario reiniciar *PhpStorm*.

Seguidamente se procede a la creación de un nuevo proyecto. Para ello se creará un proyecto de tipo *Composer*, donde será necesario seleccionar un nombre y el paquete *Symfony/framework-standard-edition*. Esto proporcionará al proyecto la estructura que debería de tener y creará los archivos de configuración por defecto.

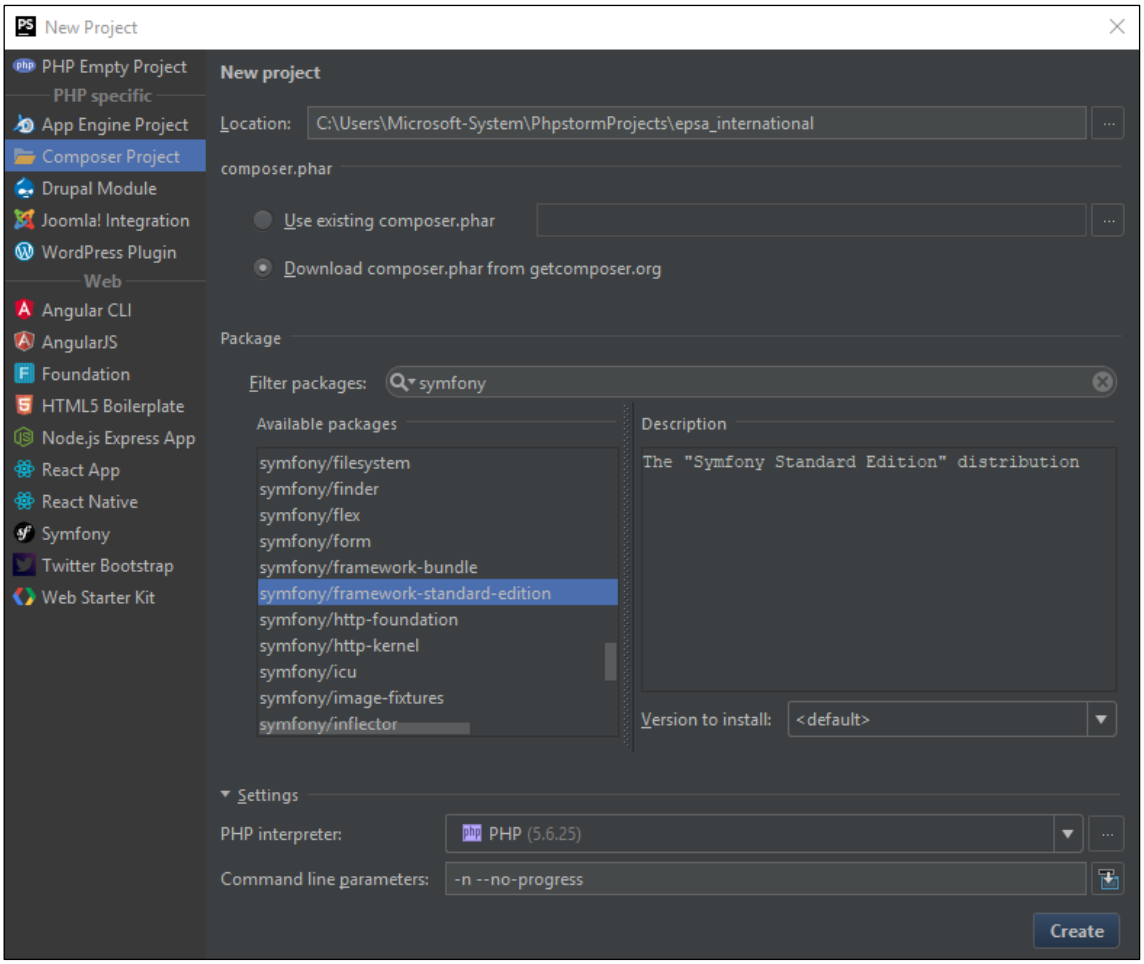


Figura 12 - Creación de un nuevo proyecto

Una vez creado el proyecto, el siguiente paso es habilitar el complemento *Symfony*, instalado anteriormente, para que esté habilitado únicamente para el proyecto que se está desarrollando. Esto permitirá utilizar una consola de comandos, para ejecutar ciertas funciones de *Symfony* y de este modo agilizar el desarrollo y creación de los diferentes apartados.

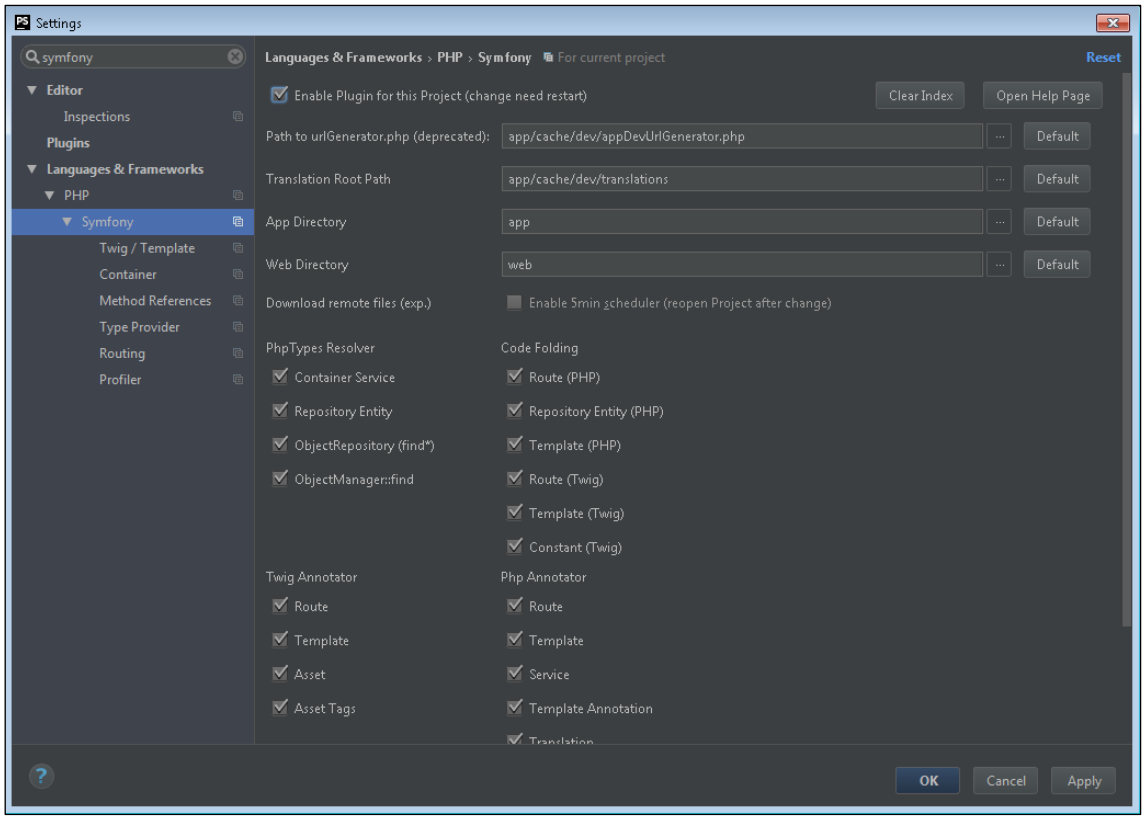


Figura 13 - Habilitación de Symfony

Esta será la estructura del proyecto, una vez creado.

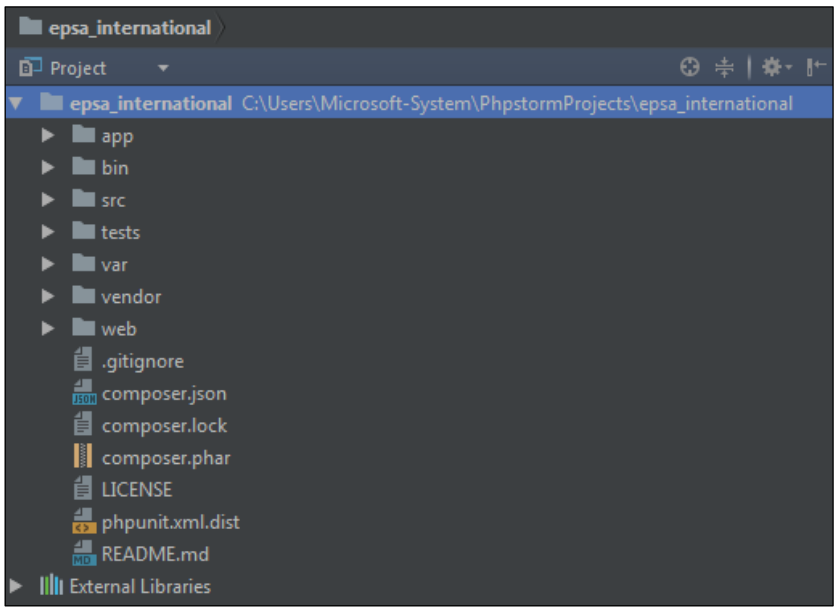


Figura 14 - Estructura de un proyecto Symfony

3.1.2.3 Contenido del proyecto

Una vez creado el proyecto, es necesario profundizar, en cierta medida, y entender qué es lo que contiene cada directorio del proyecto.

Así pues, el contenido del directorio app contiene principalmente archivos de configuración, ya sean de seguridad, de la conexión a la base de datos o del propio intérprete. Además, en este directorio se debe de colocar la plantilla base, que tendrán las vistas de la aplicación web.

Todos los archivos de configuración tienen la extensión "*.yml*", que proviene de *YAML*, que es un formato de serialización de datos de fácil lectura. El objetivo de *YAML* es representar cualquier tipo de datos y a su vez hacer que no se pierda legibilidad de dichos datos por parte del usuario.

El directorio *bin* contiene lo necesario para realizar una comprobación de si el equipo cumple los requisitos necesarios para la ejecución del software, además de un fichero que permite la integración de consolas de comandos.

El contenido del directorio *src* es, prácticamente, el más importante para el desarrollador. Dicho directorio contendrá unos subdirectorios llamados *Bundles* que de algún modo representan las diferentes partes de la aplicación web. Por ejemplo, si se creara un *Bundle* llamado *UniversidadesBundle*, indicaría que contiene todo lo relacionado con la parte funcional de las Universidad de la aplicación web. Lo más interesante de todo esto es la manera en la que dentro de cada *Bundle*, se separa el Modelo, la Vista y el Controlador. Esto se verá más en detalle en el siguiente punto.

El directorio *tests* contiene únicamente un archivo para realizar pruebas de funcionamiento, una vez creado el proyecto.

El directorio *var* es dónde se guardan toda la información de la sesión, así como los logs y la caché.

El directorio *vendor* contiene archivos de los diferentes plugins que se utilizan en el proyecto, así pues, si se instala un plugin nuevo, quedaría reflejado en dicho directorio.

Por último, el directorio web se utiliza para guardar en él todos los archivos de JavaScript, CSS o incluso imágenes que aparecen en la aplicación web.

3.1.2.4 Implementación y uso de la consola de Symfony

En *PhpStorm* es posible integrar una consola de comandos de *Symfony*, de este modo es posible agilizar ciertos procesos como por ejemplo crear de golpe los *Bundles* que sean necesarios o incluso crear la base de datos con las tablas y los campos que estén indicados en cada *Bundle*. Para ello desde las opciones del proyecto en el apartado de "*Command Line Tool Support*" es posible habilitar dicha consola.

Una vez habilitada, esto ayudará en gran medida a agilizar el desarrollo del proyecto. Por ejemplo, es posible crear un *Bundle* y que este tenga la estructura necesaria con la siguiente instrucción:

```
s generate:bundle --namespace=EPSA/UniversidadesBundle --bundle-
name=UniversidadesBundle  --dir=src/ --format=yml --no-interaction
```

Figura 15 - Ejemplo 1 de comando Symfony

O por ejemplo para crear la base de datos, una vez definidos los parámetros de las entidades:

```
s doctrine:schema:create
```

Figura 16 - Ejemplo 2 de comando Symfony

Una vez creado un *Bundle*, este quedaría de la siguiente manera:

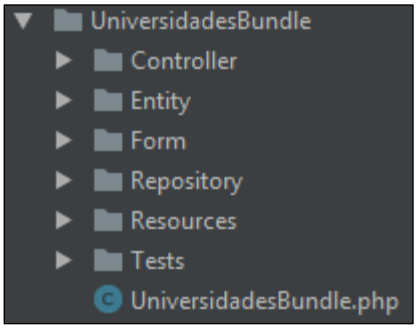


Figura 17 - Estructura de un Bundle

3.2 Entorno de producción

3.2.1 ¿Qué es Plesk?

Plesk es un panel de control que permite gestionar un sitio web de manera ordenada y sencilla y sin tener grandes conocimientos de informática. Ofrece servicios como:

- Administración de la base de datos.
- Administración de las cuentas FTP.
- Gestión de subdominios.
- Consultar estadísticas de visita.
- Controlar el consumo de recursos.
- Configuración de cuentas de correo del dominio.

3.2.2 ¿Por qué Plesk?

Plesk ofrece una administración muy sencilla sin importar el sistema operativo utilizado. Pero uno de sus aspectos más interesantes son las múltiples opciones de seguridad que ofrece. Estas han sido adaptadas para proteger el sitio web contra posibles ataques malintencionados. Algunas de las medidas de seguridad más destacadas son:

- **Fail2Ban**. Cuando se realiza un ataque de fuerza bruta para acceder al sistema, la aplicación *Fail2Ban* bloquea esos intentos de conexión, prohibiendo el acceso al sistema un determinado tiempo. Desde el panel de administración es posible establecer el número de intentos de inicio de sesión fallidos desde una dirección IP determinada. Esta medida de seguridad disuadirá en gran medida el tráfico malintencionado.
- **ModSecurity.** Actúa como *Firewall* para impedir ciertos ataques al sitio web. Lo que hace es verificar todas las peticiones *HTTP* que le llegan al servidor web, comprobar si son correctas, y en el caso de que fueran correctas, dicha petición se transferirá al contenido solicitado.

Otro aspecto a destacar es la posibilidad de realizar copias de seguridad. De este modo se podrán hacer cambios en la web sin temor a perder datos o alguna configuración previa. Las últimas versiones de *Plesk* permiten realizar copias de seguridad incrementales. Debido a que con estas copias solo se guardan los datos modificados de la última copia, la recuperación de datos será mucho más rápida.

Y como último aspecto destacable, es la posibilidad de disfrutar de los servicios de *Plesk* de manera gratuita con una cuenta que ofrece la Universidad para los alumnos. Es posible obtener más información y crear la cuenta en el siguiente enlace:

http://www.upv.es/entidades/ASIC/catalogo/433436normalc.html

3.2.3 Configuración del servicio Plesk

Base de datos

El primer paso es crear la base de datos. Para ello es necesario elegir un nombre de base de datos y elegir a qué dominio va a pertenecer. También es muy importante crear un usuario para dicha base de datos. Si se tuvieran más bases de datos, sería conveniente crear tantos usuarios administradores, como bases de datos se tuvieran para que un usuario no pudiera acceder al contenido que no le procede.

Una vez hecho esto, aparecerá un nuevo panel de control:

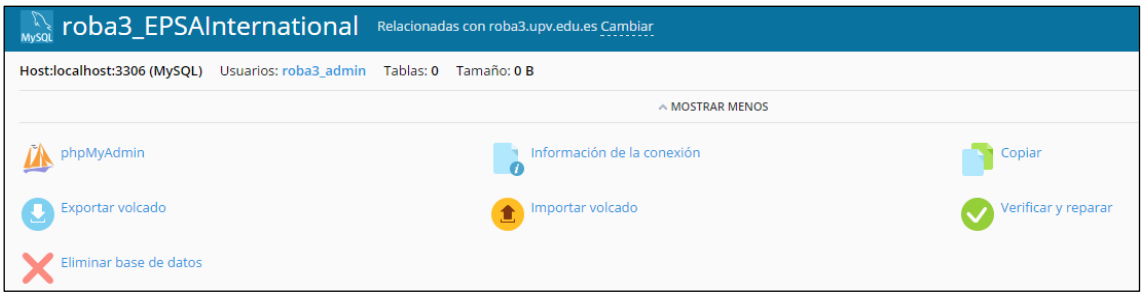


Figura 18 - Panel de control de Plesk

Este panel permite realizar múltiples tareas. En el caso de que no se tuviera la base creada, se podría utilizar la opción de *phpMyAdmin* para crear todas las tablas, y si ya se tuviera un archivo SQL, se podría importar directamente con la opción de Importar volcado.

Código fuente de la aplicación web

El siguiente paso es añadir todos los ficheros de configuración, así como el código fuente de cada parte de la página web. Para ello *Plesk* dispone de un directorio que representa un servidor web, en el cual hay que importar todo lo necesario.

El directorio nombrado anteriormente corresponde a un servidor web, por lo que todos los archivos, tanto de código *PHP, HTML, CSS y JavaScript*; así como las imágenes utilizadas en la web, deben colocarse ahí.

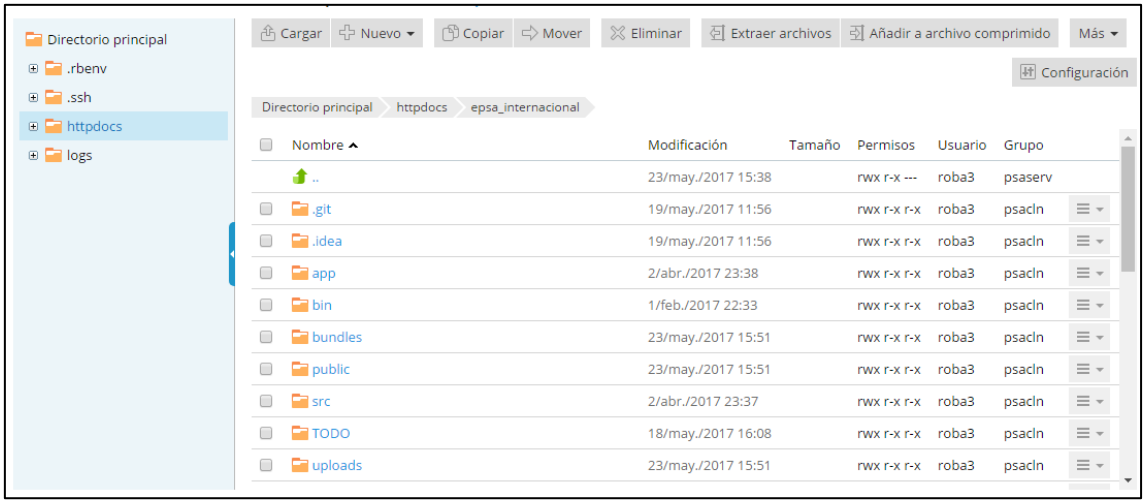


Figura 19 - Contenido del directorio de la página web

3.3 Implementación práctica

Una vez definidos los entornos de desarrollo, es momento de comenzar la fase de evaluación y diseño del producto a desarrollar. En esta fase se incluye desde los diagramas de base de datos, clases hasta los primeros diseños de la apariencia de la web final.

Nuevamente se presenta un punto crítico en el desarrollo de cualquier proyecto, ya que una mala decisión tomada en cualquiera de estos puntos podría ocasionar una reacción en cadena, una decisión errónea en una etapa X podría afectar a las siguientes etapas X+1, lo que podría causar un replanteamiento total de las fases del proyecto, y del proyecto en última instancia.

Antes de pasar a los puntos más específicos de la implementación es necesario matizar que diseño de programación ha sido empleado para dar forma al proyecto y al mismo tiempo dar una visión general de cómo se ha implementado el proyecto en la descomposición de los módulos que lo conforman.

3.3.1 Elección del patrón de diseño

Gracias a los patrones de diseño, se logra que la reutilización de buenos diseños y arquitecturas sea más fácil. Gracias a ellos, se consigue elegir las alternativas de diseño que hacen que un sistema llegue a ser reutilizable, y evitar aquellas que lo dificulten. Un buen patrón de diseño puede mejorar la documentación y el mantenimiento de los sistemas a desarrollar o existentes al proporcionar una especificación explícita de las interacciones entre clases, objetos.

Según Christopher Alexander [1]:

> *"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces".*

Ahora bien, en este caso, la elección del patrón de diseño viene impuesta por la elección del Framework, ya que, en muchas ocasiones, un Framework limita el abanico de posibles opciones en cuanto a patrones de diseño. *Symfony*, ofrece el patrón de diseño *MVC* (*Model View Controller*), un patrón muy generalizado en casi todos los lenguajes, por su sencillez y por la separación en capas de los componentes que conforman la aplicación. Richard Helm, Ralph Johnson y John Vlissides [2] definen el patrón *MVC* (Figura 3.2.1) como:

> *"MVC consiste en tres tipos de objetos. El Modelo es el objeto de aplicación, la Vista es su presentación en pantalla y el Controlador define el modo en que la interfaz reacciona a la entrada del usuario… MVC desacopla las vistas de los modelos estableciendo entre ellos un protocolo de suscripción/notificación. Una vista debe asegurarse de que su apariencia refleja es estado del modelo. Cada vez que cambian los datos del modelo, éste se encarga de avisar a las vistas que depende de él."*

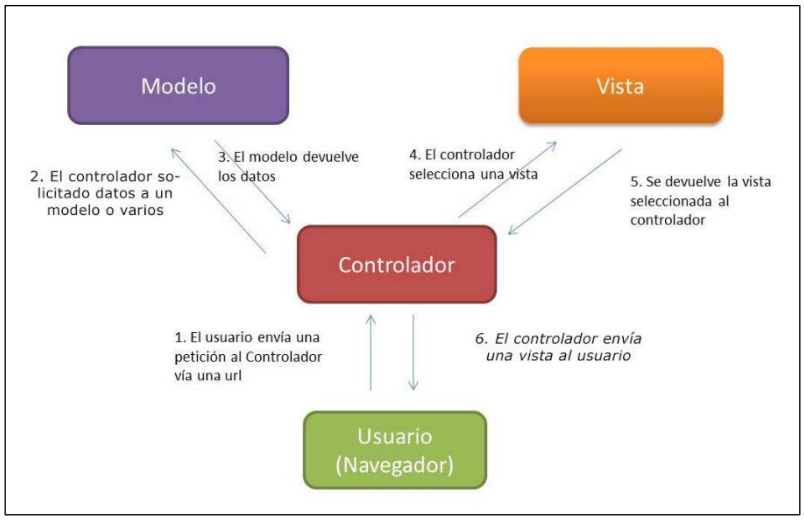


Figura 20 - Patrón Modelo Vista Controlador

3.3.2 Diseño del producto

Es momento de detallar la estructura final del producto a desarrollar, para ello se propone el siguiente esquema, que a grandes rasgos manifiesta como se ha separado la aplicación en múltiples aplicaciones para lograr una independencia y desacople entre las mismas, lo que facilitará el futuro mantenimiento y/o desarrollo.

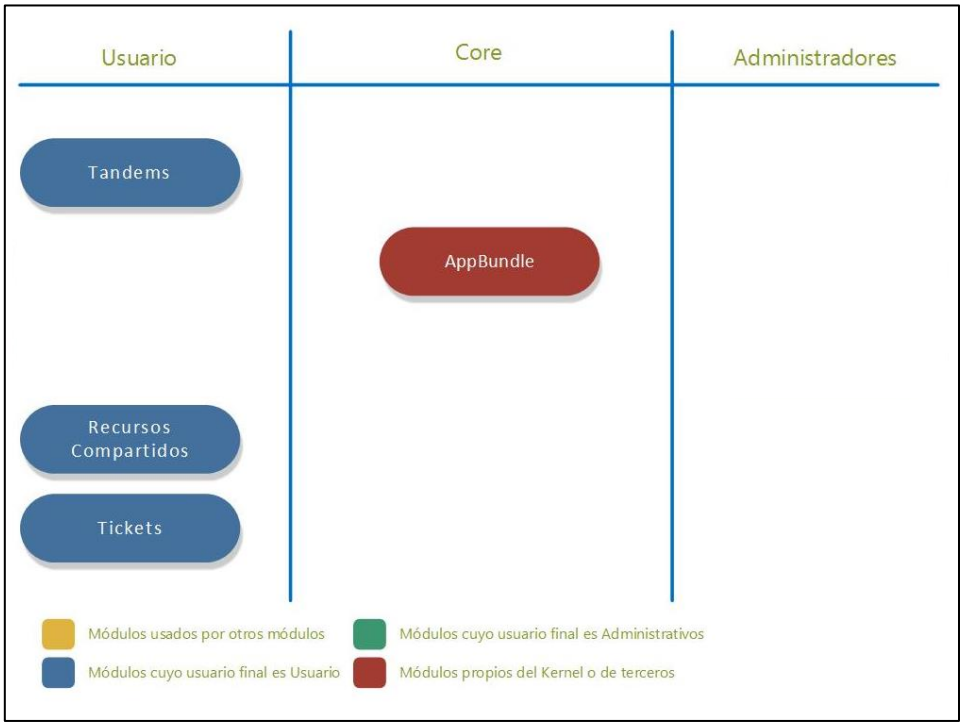


Figura 21 - Módulos implicados

3.3.3 Análisis de los usuarios

En la aplicación web es posible distinguir cuatro tipos de usuarios. Estos son:

- Usuario **anónimo** - Usuario que se conecta por defecto a la aplicación.
- Usuario **registrado** - Usuario registrado en el sistema.
- **Administrador** - Usuario registrado en el sistema con permisos de administración.
- **Super Administrador** - Usuario, con mismos permisos que el Administrador y algunos privilegios adicionales.

Cada uno de estos usuarios tiene sus permisos, los cuales se explican en los siguientes diagramas:

- **Usuario no registrado o usuario anónimo.** Es capaz de navegar superficialmente por la web, es decir, visitar solamente ciertos apartados, sin poder añadir nuevos registros. Este usuario podrá buscar experiencias de Erasmus o destinos de Erasmus, para poder utilizar el resto de funcionalidades, es necesario realizar el registro.

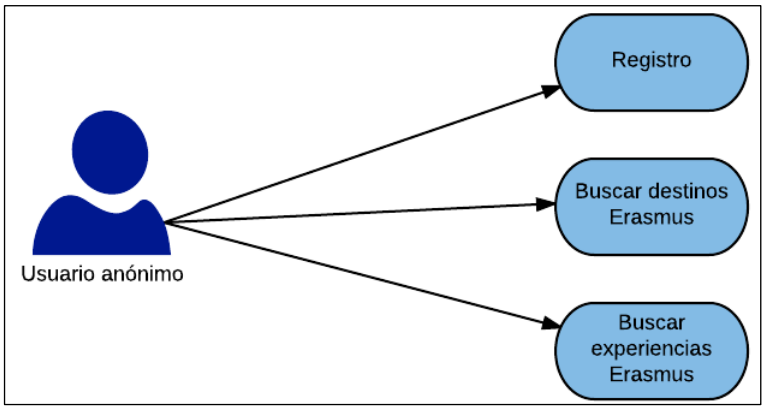


Figura 22 - Esquema usuario anónimo

- **Usuario registrado.** Es capaz de interactuar con la web, pudiendo editar sus datos, reportar incidencias, crear o participar en los tándems creados por otros usuarios, publicar experiencias de Erasmus, además de publicar y acceder a recursos compartidos.

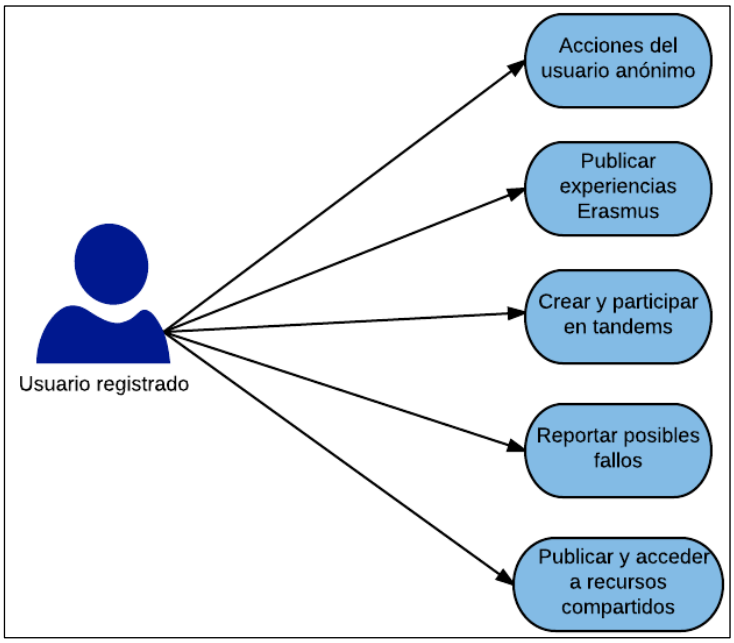


Figura 23 - Esquema usuario registrado

- **Usuario Administrador.** Su función es mantener el control en la web, ya sea arreglando posibles incidencias, gestionando las publicaciones de los usuarios o incluso penalizar a usuarios molestos.

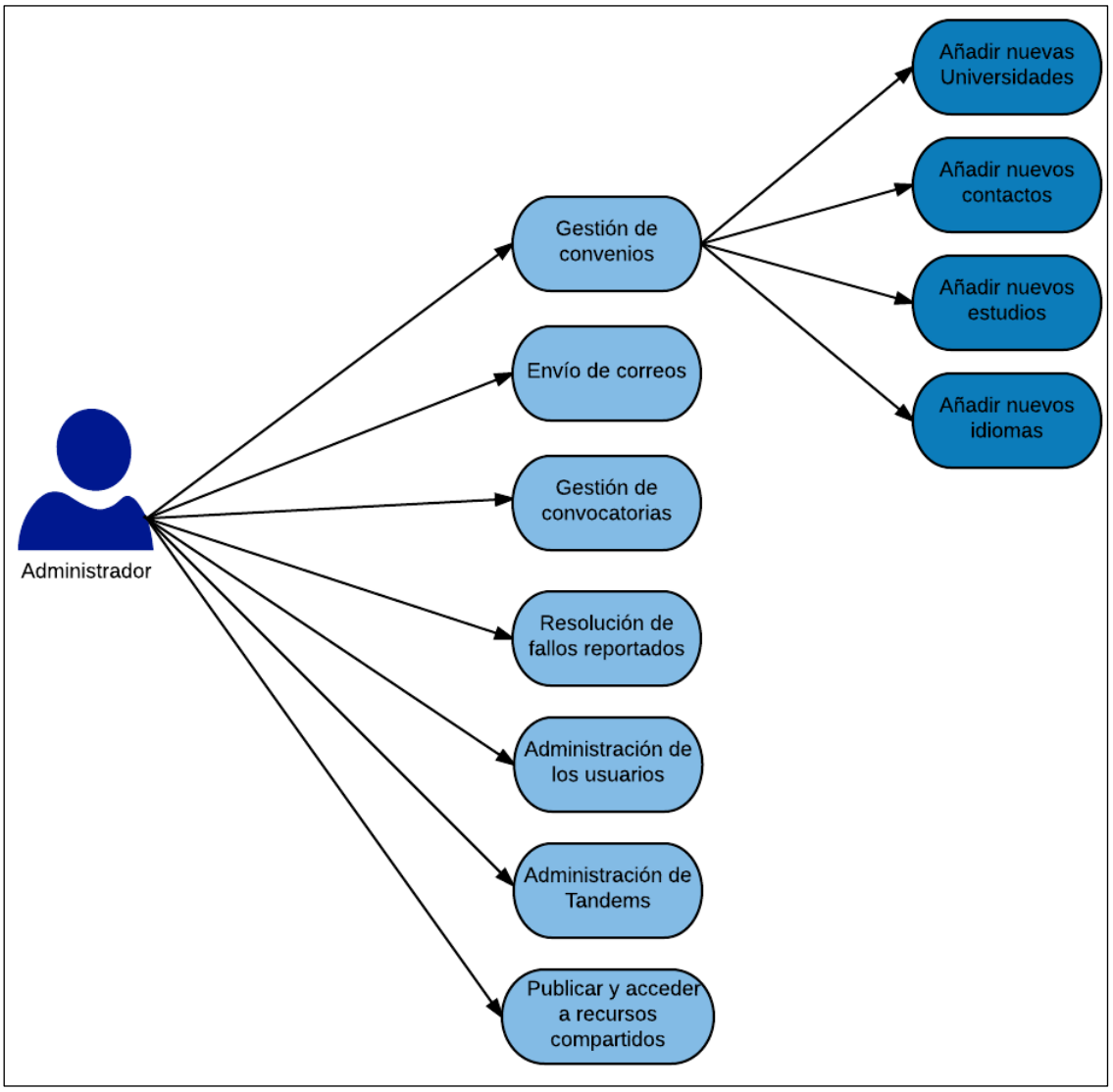


Figura 24 - Esquema usuario Administrador

- **Usuario Super Administrador.** Este es el usuario de mayor rango. Podrá realizar todas las acciones que realiza el Administrador, pero además tiene una función muy importante que es la de asignar el rol de Administrador a otros usuarios.

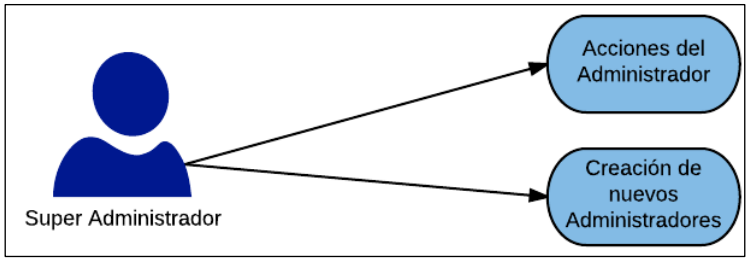


Figura 25 - Esquema usuario Super Administrador

3.3.4 Modelo de Datos

A continuación, se expone el modelo de datos que ha sido diseñado para dar solución a las aplicaciones planteadas. En él se define el conjunto de hechos relevantes y con algún significado implícito que pueden ser registrados en el modelo. Además, dichos modelos representan situaciones o cambios de situaciones del mundo real, como pueden ser: los tándems, experiencias erasmus, etc.

Del mismo modo se definen las relaciones entre los modelos intentando lograr la mayor independencia entre los modelos para así alcanzar una mayor escalabilidad de la aplicación. ¿Cómo se logra esto? Intentado realizar un diseño de los modelos lo más aislado de las aplicaciones, cuya integración con otras aplicaciones o módulos sea primordial en el diseño.

Por ejemplo, se ha diseñado un modelo llamado "*Usuario*" el cual se gestiona desde dentro del módulo "*Usuarios*" pero que puede ser empleado por cualquier otro módulo, sin la necesidad de estar definido en el módulo en el que vaya a usarse. De esta forma, el modelo "*Usuario*" es empleado en el módulo "*Tándems*" o "*Experiencias*", en ambos módulos se emplea, pero únicamente se ha definido una única vez (*Figura 25*). De esta forma se logra por ejemplo que si se realiza un cambio en el modelo "*Usuario*" no tener que realizar el mismo en todos los modelos.

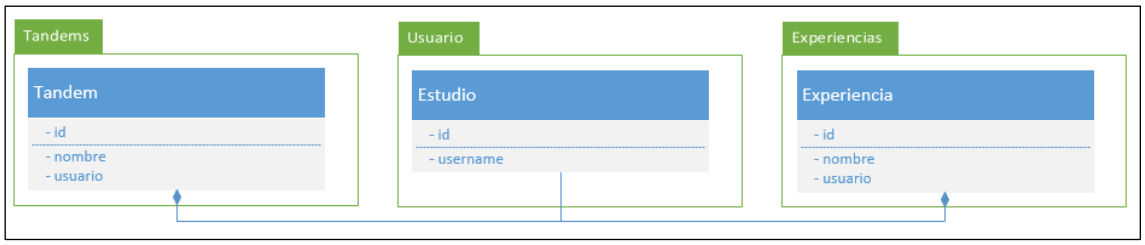


Figura 26 – Modelos independientes del módulo

Por otro lado, está el caso contrario, en el que cada modelo, se implementa de forma explícita dentro de cada módulo, con lo que no se evita la duplicidad de código y evita un buen mantenimiento y escalabilidad de la aplicación. Este caso se observa en la *figura 26*.

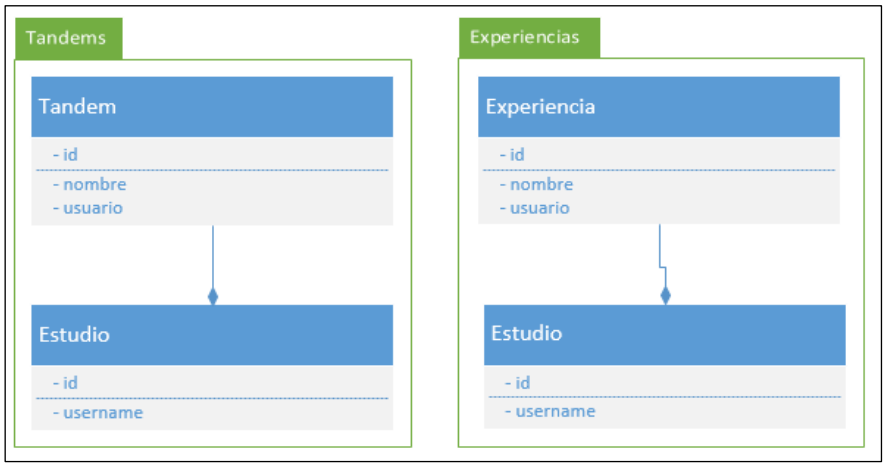


Figura 27 – Modelos dentro del módulo

3.3.5 Modelo de Clases

Una vez definido el producto y el modelo de datos, es momento de definir la representación de los mismos en código. Para ello se hace uso del diagrama de clases. Este tipo de diagramas consiste en representar la estructura del sistema y el comportamiento de los diferentes objetos que este contiene, así como la relación con otros objetos. La principal diferencia entre un diagrama de clases y un diagrama de entidad relación (DER) es que el DER es el primero que se define y en él se representan las entidades relevantes del sistema. Dichos diagramas se utilizan para diseñar la base de datos o algún tipo de sistema de información.

Una vez definido el diagrama de entidad relación, se procede a representar el diagrama de clases. Los elementos básicos de estos diagramas son:

- **Clases**. Las clases describen un conjunto de objetos con sus propiedades, así como su comportamiento. Dichos objetos son instancias de esa clase. Dentro de una clase se definen dos elementos:
    - Atributos: un atributo de una clase representa el tipo de dato asociado al objeto.
    - Métodos: es una función propia del objeto de la clase, lo que caracterizará dicho objeto.

  También es posible representar clases abstractas. Estas clases no existen como tal, sino que sirven de manera conceptual para diseñar ciertos aspectos. Básicamente se utilizan como depósitos de métodos y atributos para ser heredados por otras subclases.
- **Relaciones**. Las relaciones describen la manera en cómo se relacionan las clases entre sí, estas pueden ser de tipo:
    - Asociación: es el tipo de relación más general y representa una conexión entre objetos. Puede ser tanto unidireccional como bidireccional.
    - Agregación: es un tipo de relación que representa la jerarquía entre objetos y las partes que lo componen.
    - Composición: es una relación que indica que las partes asociadas a un objeto solo pueden existir asociadas a dicho objeto.
    - Herencia: es una relación entre una clase padre y sus subclases. Este tipo de relaciones hacen que la clase padre comparta sus métodos y atributos entre sus subclases.
    - Dependencia: es un tipo de relación que indica que hay dependencia entre dos clases, por lo que una clase requiere de otra para poder realizar sus funciones.
- **Interfaces**. Define un conjunto de operaciones de una clase que son visibles por otras clases.

*Nótese cómo se han añadido dentro de las clases atributos a clases relacionadas. Esto no está permitido dentro de la propia definición del diagrama de clases UML, pero se considera necesario está modificación del patrón de diseño de clases UML para tener una mejor visión de la estructura.

TandemBundle

Módulo utilizado para la creación y gestión de los Tándems, donde un usuario crea un Tándem y el resto puede participar. Se tiene un array de *participantes*, al cual es posible añadir o quitar usuarios utilizando los métodos "*addParticipante()*" y "*removeParticipante()*".

Figura 28 - Diagrama de clases Tándem

ApuntesBundle

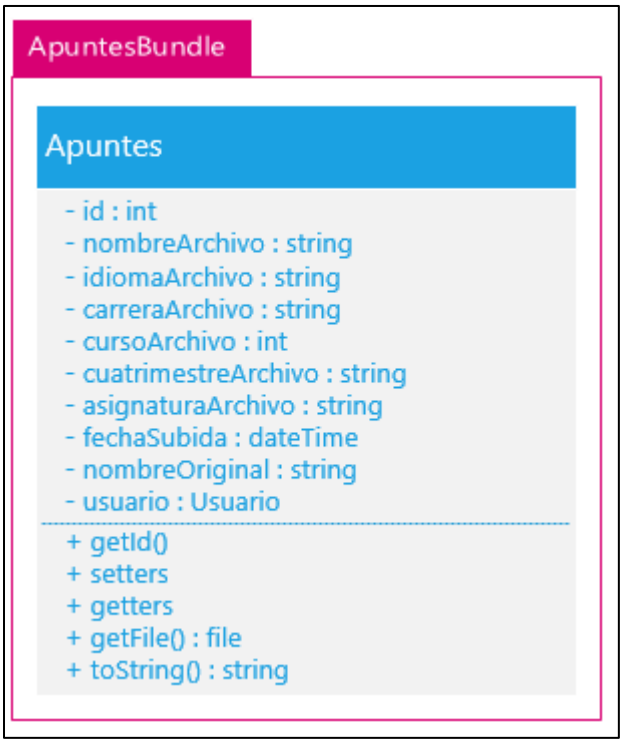


Figura 29 - Diagrama de clases Apuntes

Módulo encargado de gestionar la subida de archivos que se realiza al servidor. Por un lado, se guarda el nombre original del archivo y, por otro lado, se genera un nuevo nombre aleatorio, de este modo se pretende que no existan problemas en subir archivos que se llamen igual.

TicketsBundle

Módulo que se utiliza para gestionar los tickets que crea un usuario para indicar algún tipo de incidencia en algún apartado concreto de la aplicación. Además, se controla que usuario Administrador ha resuelto la incidencia y la fecha de la resolución.

Figura 30 - Diagrama de clases Tickets

3.3.6 Diseño de Base de Datos. Entidad Relación

Una vez razonado el diseño de Datos y Clases a implementar, se procede a la integración de dichos modelos en la base de datos. Para ello se extrapolan lo mismo al diseño Entidad Relación, dando como resultado el siguiente esquema.

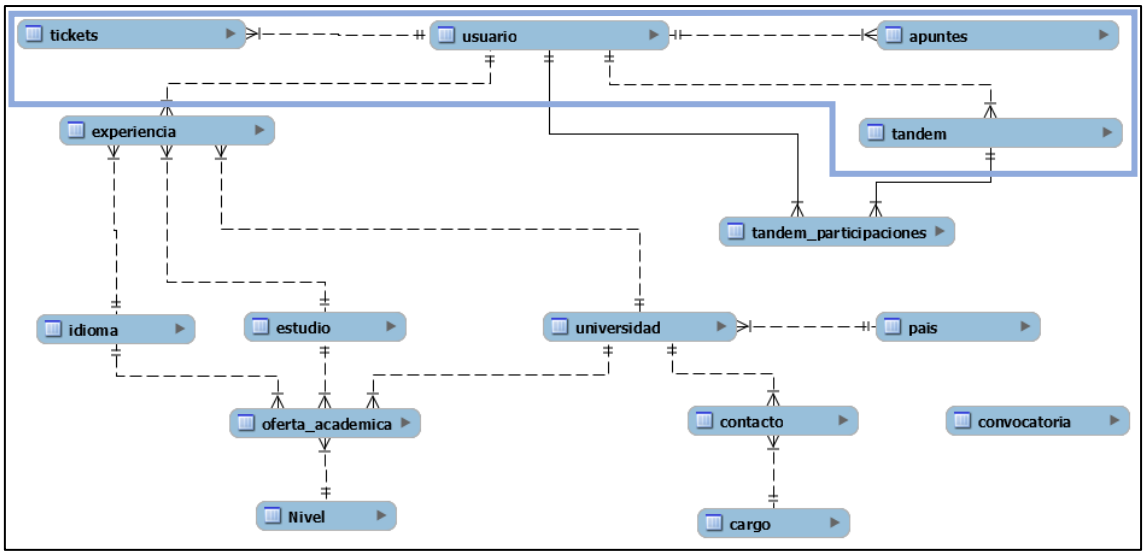


Figura 31 - Diagrama de Bases de Datos

Nótese como se han omitido el detalle de los campos que componen cada tabla, este detalle se desglosa en el Anexo Diagrama de Base de Datos.

* Las tablas englobadas dentro del recuadro azul, son las implementadas en este proyecto, siendo las otras implementadas por otro alumno en otro proyecto. Recordar que este proyecto está compuesto por dos proyectos finales de carrera.

3.3.7 Diseño de Interfaces y Usabilidad

El diseño de interfaces, y su grado de usabilidad, implica realizar estudios sobre la interacción entre los usuarios y las aplicaciones de las que van a hacer uso. Algunos autores definen la interacción persona-ordenador (o aplicación) como:

"La disciplina dedicada al diseño, la evaluación y la implementación de sistemas informáticos interactivos para el uso humano; y al estudio de los fenómenos relacionados más significativos". B. Hefley. "Curricula for Human-Computer Interaction". Ed. 1992

"El estudio de cómo las personas diseñan, implementan y usan sistemas informáticos interactivos; y de cómo los ordenadores influyen en las personas, las organizaciones y la sociedad". B.A. Myers; J.Hollan, I. Cruz. "Stategic DIrecions in Human Computer Interaction". Ed 1996.

Con estas definiciones se extrapola los elementos clave del diseño de interfaces persona computador:

- **Tecnología**: el diseño de interfaces persona computador es una disciplina dedicada al estudio de toda tecnología (ordenadores o aplicaciones) que ofrezca la capacidad de interacción y uso con usuarios.
- **Personas**: se necesita comprender el funcionamiento de las personas en sus tareas, capacidad y limitaciones; cómo resuelven sus problemas y su grado de aprendizaje en la toma de decisiones.
- **Diseño**: es la capacidad de idear soluciones a problemas de interacción. En el diseño de un producto se delimitan los posibles modos de uso, y será este diseño quien condiciona la interacción.

Dentro de este punto destaca el concepto de ***Usabilidad***, que es definido por la norma *ISO 9241-11:1998 (International Organization for Standarization)* como:

"Grado de eficacia, eficiencia y satisfacción con el que usuarios específicos pueden lograr objetivos específicos, en contextos de uso específicos".

Destacar que el término usabilidad no es un atributo universal. Un producto o solución usable, no implica que lo sea para todas las personas, sino sólo para aquellas para las que ha sido destinado.

La usabilidad relaciona la facilidad de uso con objetivos y contextos de uso específicos. Los productos están diseñados para satisfacer usos concretos y es para estos propósitos para los que deben de resultar fáciles de usar. Con esta premisa, se demuestra que las actuales aplicaciones empleadas en el Área de Relaciones Internacionales no son usables para el ámbito en el que son usadas.

Por lo tanto, la ***Usabilidad*** es una cualidad objetiva, ya que puede ser medida y evaluada por los atributos que la conforman: **eficacia, eficiencia** y **satisfacción de uso**.

A continuación, se presentan los bocetos iniciales de las interfaces, seguidos de los resultados finales del diseño.

El primer boceto correspondería a cualquier tipo de formulario para la creación de contenidos. Como por ejemplo un Tándem.

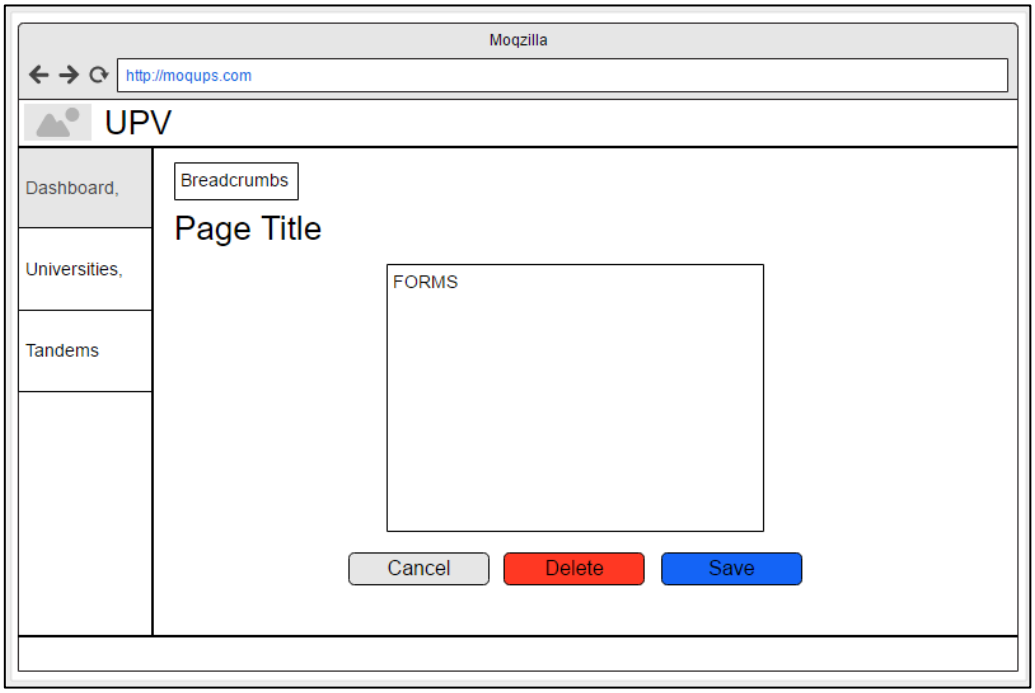


Figura 32 – Boceto de formulario de creación de contenido

En el siguiente boceto se puede observar la manera en la que se mostraría la información al usuario. Esto se haría mediante tablas, para que fuera más facil filtrar resultados y encontrar lo que más se desee.

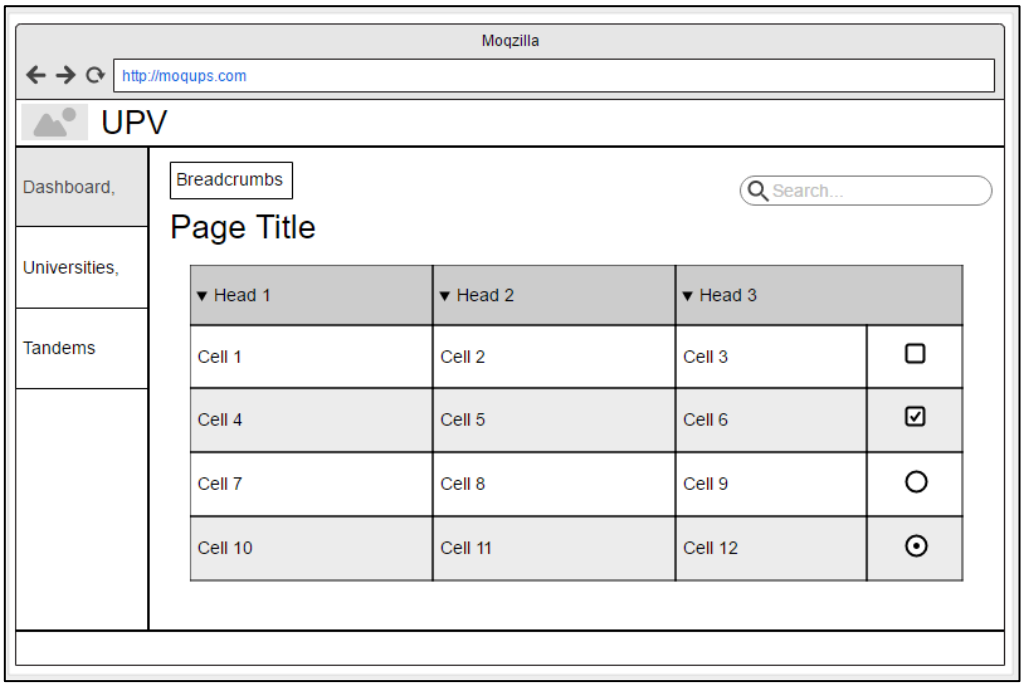


Figura 33 - Boceto de tabla de contenidos

3.3.8 Diagramas de flujo y colaboración

Para presentar un diagrama de funcionamiento es necesario, en primer lugar, definir qué es y cómo se puede implementar. O.C. Basttistutti [1], define como diagrama de flujo:

> "*Un diagrama de flujo representa la esquematización gráfica de un algoritmo. En realidad, muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. Su correcta construcción es sumamente importante porque a partir del mismo se escribe un programa en algún lenguaje de programación. Si el diagrama de flujo está completo y correcto, el paso del mismo a un lenguaje de programación es relativamente simple y directo.*"

Además, para modelar un diagrama de flujo es necesario seguir una serie de convenciones que establezcan unas reglas básicas, para así lograr que cualquier diagrama pueda ser comprendido por cualquier desarrollador ajeno al proyecto. De esta forma O.C. Basttistutti [3], recomienda utilizar los estándares de "International Organization for Standarization" (ISO) y la "American National Standards Institute" (ANSI). En el Anexo 8.3 se adjunta una tabla con los elementos para la representación de las etapas y procesos.

En este caso, no se está implementado un algoritmo al uso, pero si un proceso que a grandes rasgos puede ser considerado un algoritmo. Así, se logra que la definición anterior se adapte perfectamente al propósito.

Ahora bien, ¿qué beneficios aporta un diagrama de flujo?, como se ha citado anteriormente, un diagrama de flujo es una representación gráfica de procesos, donde cada paso o etapa está representado con un símbolo con una breve descripción. Cada etapa está unida a otras mediante flechas que indica la dirección de avance del proceso. Gracias a los diagramas de flujo es posible:

- Obtener una visión global o específica de un proceso o conjunto de procesos, mostrando la relación secuencial y causal entre ellas, facilitando la comprensión por parte de los usuarios o futuros desarrolladores.
- Proporcionar un modelo de comunicación, con un lenguaje más eficaz y conciso.
- Ayuda a referenciar mecanismos de control y medición de procesos, objetivos.
- Facilita el estudio y aplicación de las acciones.

Dentro de la aplicación se diferencian distintos flujos por cada usuario dependiendo del módulo que se desee utilizar:

- Usuario:
    - Registro/Recuperación password*
    - Tándems
    - Publicación Apuntes
    - Envío de tickets de incidencias
- Administrador
    - Registro/Recuperación password*
    - Aprobación de Tándems
    - Resolución de incidencias

*Nota: como el proceso de registro, inicio y recuperación de password es el mismo para ambos tipos de usuario, únicamente se definirá un diagrama de flujo que será aplicable a ambos usuarios.

Como la aplicación dispone de muchos módulos, y cada uno con diversas funcionalidades, se ha decidido mostrar únicamente los flujos de datos más importantes y significativos.

El primer diagrama expuesto es el que corresponde al inicio de sesión de un usuario. Donde lo primero que se comprobará es si el usuario está registrado. Aquí aparecen tres posibles situaciones:

- No registrado. Si el usuario no está registrado, se le proporcionará un formulario a rellenar con dator válidos.
- Registrado. Si el usuario está registrado, se le proporcionará un formulario para acceder a la aplicación.
- No recuerda password. Se enviará al usuario un correo con un enlace para establecer una contraseña nueva, tras lo cual podrá volver a intentar iniciar la sesión.

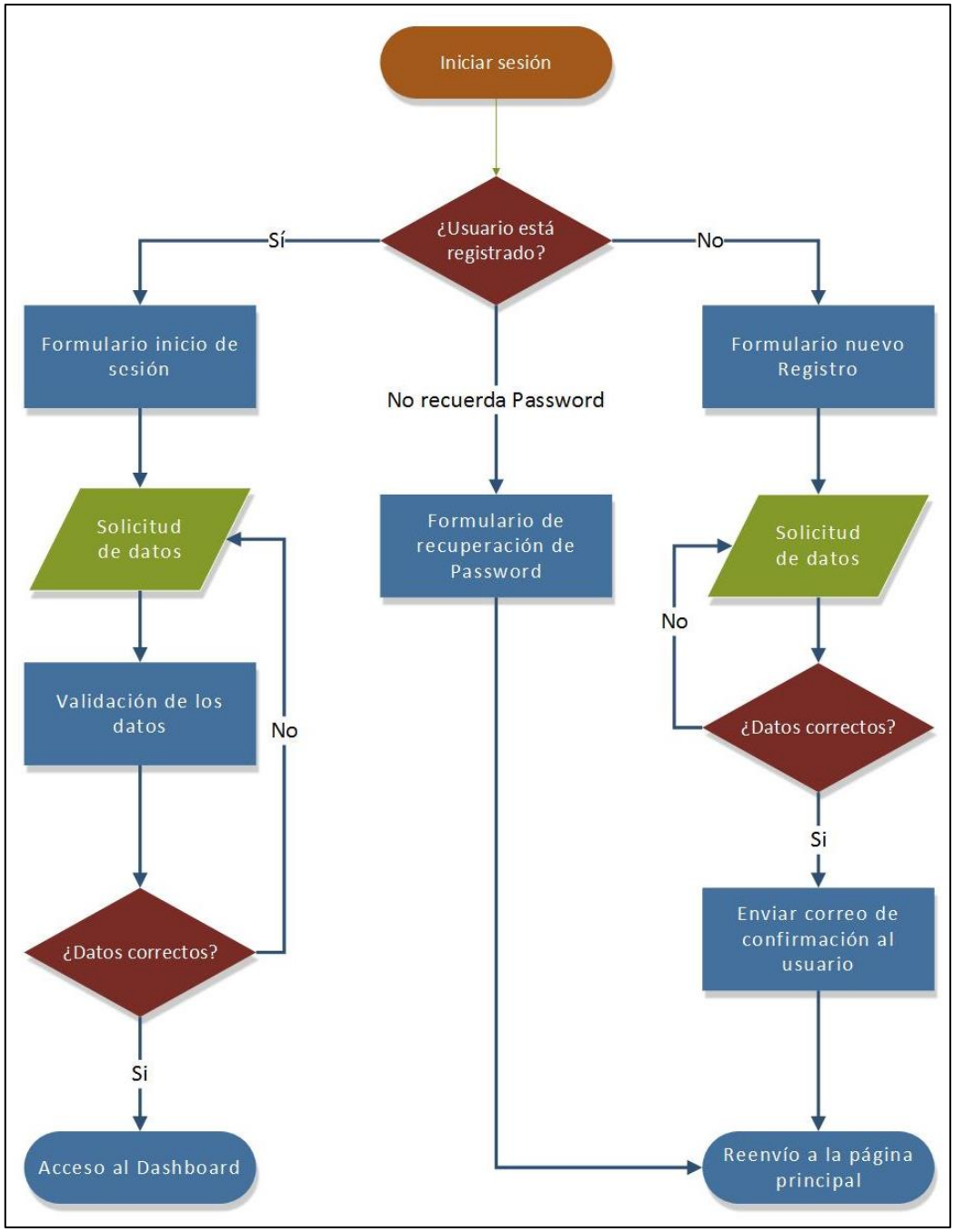


Figura 34 - Diagrama de flujos login/registro

El siguiente diagrama muestra el proceso de creación de un tándem. Este mismo proceso se repetiría para otros módulos que necesiten una validación y aprobación por parte de un administrador.

Lo primero que se hace es comprobar si el usuario que intenta crear el tándem es un usuario registrado, en caso de no serlo, se le redirige al menú principal. Si el usuario está registrado se le proporciona un formulario el cual debe de ser completado con datos válidos. Una vez validados los datos se crea el tándem.

Finalmente, para que un tándem sea visible, debe de ser aprobado por un administrador, de este modo se pretende evitar contenido inapropiado.

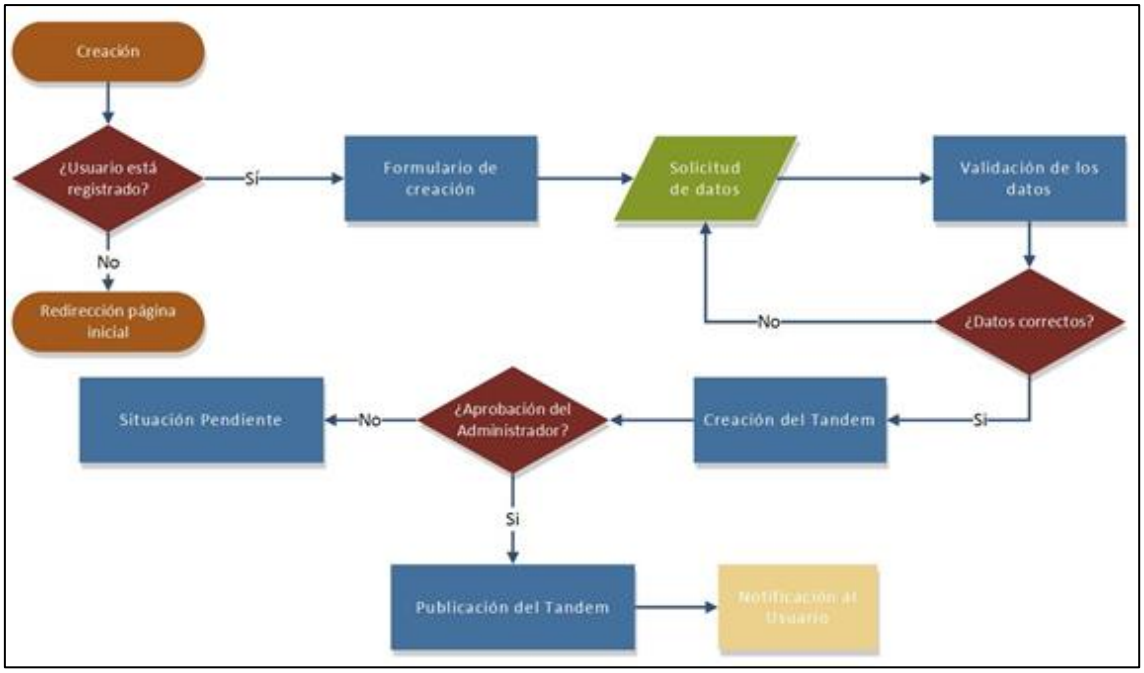


Figura 35 - Diagrama de lujos de creación de un tándem

A continuación, se procede a explicar mediante diagrama de secuencia, el proceso de aprobación un Tándem. El proceso de aprobación, siempre se realiza por parte del Administrador.

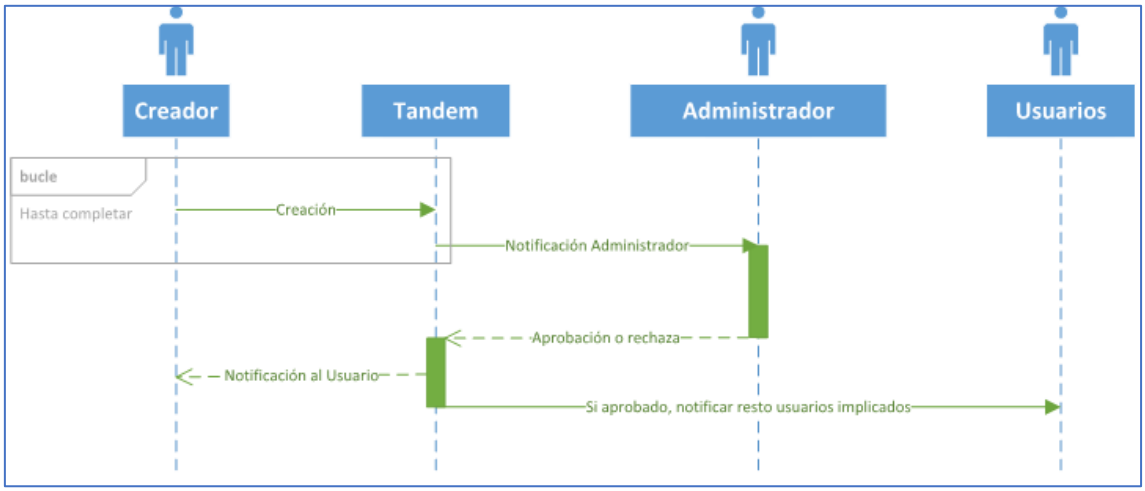


Figura 36 - Diagrama de secuencias de la aprobación de un tándem

3.4. Pruebas

En todo sistema informático es necesario realizar pruebas de funcionamiento. Es por esto que se han realizado pruebas de diferente índole. Estas han sido:

- Acceder a una ubicación disponible solo para usuarios administradores (/admin/usuarios), siendo un usuario corriente.
- Subir un archivo ejecutable (.exe) en lugar de un pdf.
- Dejar campos obligatorios sin rellenar en cualquier formulario.
- Intentar borrar contenido creado por otro usuario forzando la realización de una acción desde la barra de navegación (/ID_contenido/delete).

A todas estas pruebas y algunas otras, el sistema ha respondido correctamente. A pesar de todo, en cualquier aplicación web pueden surgir errores al experimentarse ciertos comportamientos inesperados por parte del usuario. Es precisamente para este tipo de cosas para las que se ha creado un sistema de tickets, dónde un usuario podrá informar de cualquier tipo de incidencia.

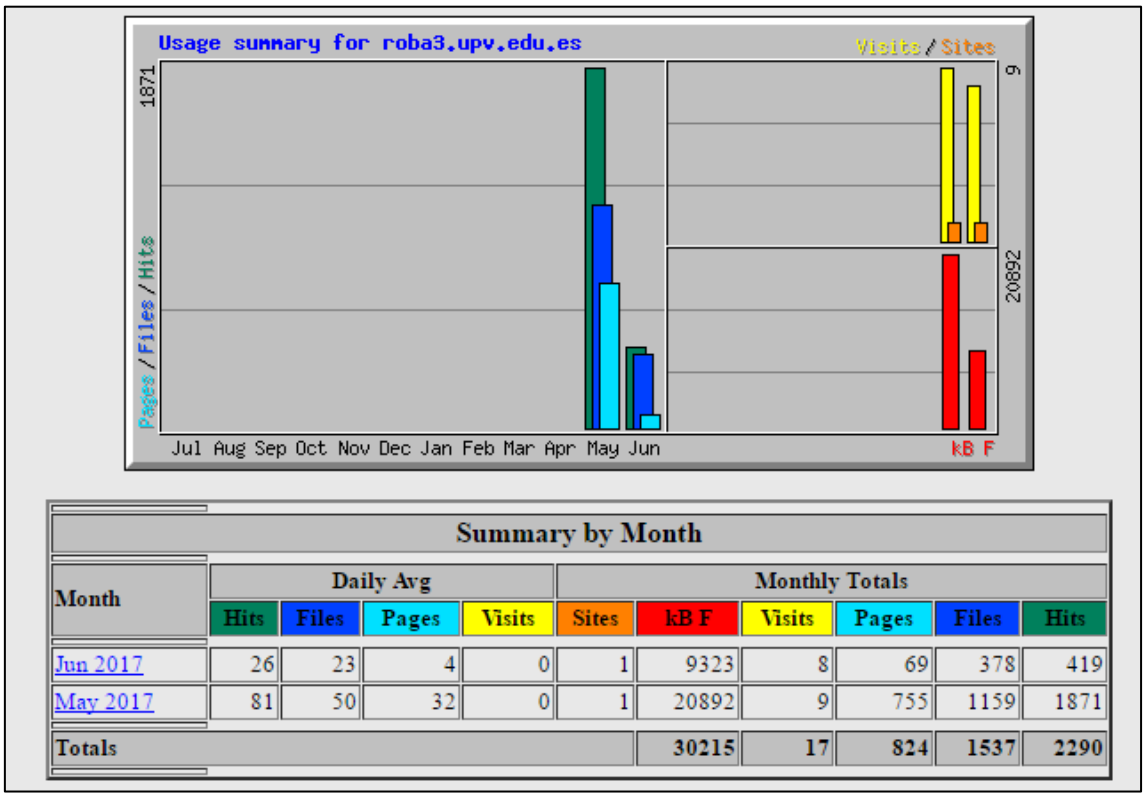
4. Resultados

4.1 Manual de usuario

Toda nueva aplicación requiere cierto tiempo de aprendizaje por parte del usuario, para lograr dominar el sistema y conseguir lo que el propio usuario quiere hacer. Es por esto que junto a la presente memoria, se adjuntará un manual de usuario que se podrá consultar, en el cual se explicará con el mayor detalle posible todas las funcionalidades que se ofrecen. De este modo se pretende que un alumno recién registrado, pueda llevar a cabo sus tareas de una manera rápida y sencilla.

4.2 Estadísticas de uso

El panel de ***Plesk*** permite visualizar estadísticas de uso de la página web que tiene alojada. Debido a que la aplicación web es nueva y no ha sido lanzada oficialmente, las estadísticas que se proporcionan no dicen mucho. Pero esto cambiaría con el paso del tiempo.



Summary by Month

Month	Daily Avg				Monthly Totals					
	Hits	Files	Pages	Visits	Sites	kB F	Visits	Pages	Files	Hits
Jun 2017	26	23	4	0	1	9323	8	69	378	419
May 2017	81	50	32	0	1	20892	9	755	1159	1871
Totals						**30215**	**17**	**824**	**1537**	**2290**

Figura 37 - Estadísticas de uso

5. Conclusiones

5.1 Conclusiones personales

Normalmente cuando una persona realiza cualquier tipo de proyecto en solitario, no aparece ningún conflicto ni choque de ideas, ya que el desarrollador planteará todo y hará todo como le parezca más correcto, aunque muchas veces no lo sea. El problema surge cuando en un mismo proyecto trabajan dos personas. A simple vista podría parecer que no hay ningún inconveniente ya que cada desarrollador trabaja en su parte de manera independiente, pero la realidad es muy distinta.

Solamente durante el planteamiento del proyecto pueden surgir múltiples conflictos, ya que cada uno de los miembros querrá llevar el desarrollo, como lo haría realizando el proyecto en solitario.

En situaciones como estas, es cuando uno aprende a trabajar en equipo y a tomar decisiones como un equipo. Y en estas mismas situaciones, uno descubre las cosas que ha estado haciendo mal, gracias a la opinión de otro compañero.

Es por esto, por lo que el proyecto ha sido una experiencia enriquecedora, tanto en el ámbito profesional por el aprendizaje de una nueva tecnología para el desarrollo de una aplicación, como en el ámbito personal, debido a que plantear un proyecto con otra persona supone pensar y actuar como un equipo.

5.2 Futuras líneas de desarrollo o posibles mejoras

Una de las mejoras que está prevista para el futuro es un buzón de sugerencias. Ya que esto permitiría abrir una nueva línea de posibilidades a la hora de añadir nuevas funcionalidades a la aplicación web.

Otra de las posibles mejoras es de cara el apartado de Tándems. Consistiría en la posibilidad de que un usuario personalice su perfil con datos sobre los idiomas que conoce y su nivel, así como los idiomas que quiere aprender. De este modo, se podría realizar búsquedas cruzadas y realizar emparejamientos de manera eficiente sin necesidad de ir buscando tándem por tándem.

El principal motivo por el que estos futuros desarrollos no han sido incluidos en el proyecto final, es por el hecho de que el proyecto en sí ha sido bastante grande ya que ha abarcado múltiples funcionalidades y ha requerido una gran dedicación para llevarlas a cabo. Por lo que se ha decidido priorizar para sacar adelante otros aspectos más importantes.

6. Bibliografía

1. Christopher Alexander, “Patrones de Diseño”, Ed. I, 2002
2. Richard Helm, Ralph Johnson, John Vlissides, “Patrones de Diseño”, Ed. 1, 2002
3. O. C. Battistutti, “Metodología de la programación”, 3ª edición, 2005
4. R. Elmasri y S. Navathe. “Fundamentos de los Sistemas de Bases de Datos”, 3ª edición. Addison-Wesley, 2002.
5. Silberschatz, H. F. Korth y S. Sudarshan. “Fundamentos de Bases de Datos”, 4ª edición. McGraw Hill, 2002.
6. Carlos Casado Martínez; Muriel Garreta Domingo; Yusef Hassan Montero; Loïc Martínez Normand; Enric Mor Pera. “Interacción persona ordenador”. Edición 2011. OpenLibra
7. Todo sobre symfony: https://symfony.com/
8. Documentación de los servidores Plesk: https://docs.plesk.com/en-US/onyx/

7. Acrónimos

- TFG: Trabajo Final de Grado
- RRII: Relaciones Internacionales
- CMS: Content Management System.
- DRY: Don't Repeat Yourself.
- ER: Entidad Relación.
- ASIC: Área de Sistemas de la Información y las Comunicaciones

8. Anexos

8.1 Código fuente

Debido al alto volumen que ocupa el código desarrolla se ha optado por crear un repositorio web donde centralizar todo el código desarrollado y que servirá como respaldo para futuras líneas de desarrollo.

https://bitbucket.org/Kondoriano/epsa_international-v2

8.2 Diagrama de Gantt

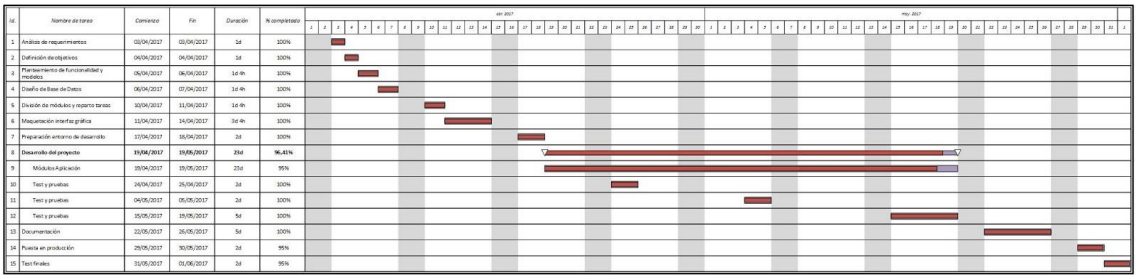


Figura 38 - Diagrama de Gantt

8.3 Diagrama de Base de Datos

Gracias al diseño modular del proyecto, permite una representación del diagrama de base de datos por módulos. De esta forma se presentan las siguientes agrupaciones:

8.3.1 Apuntes, tickets y tándems

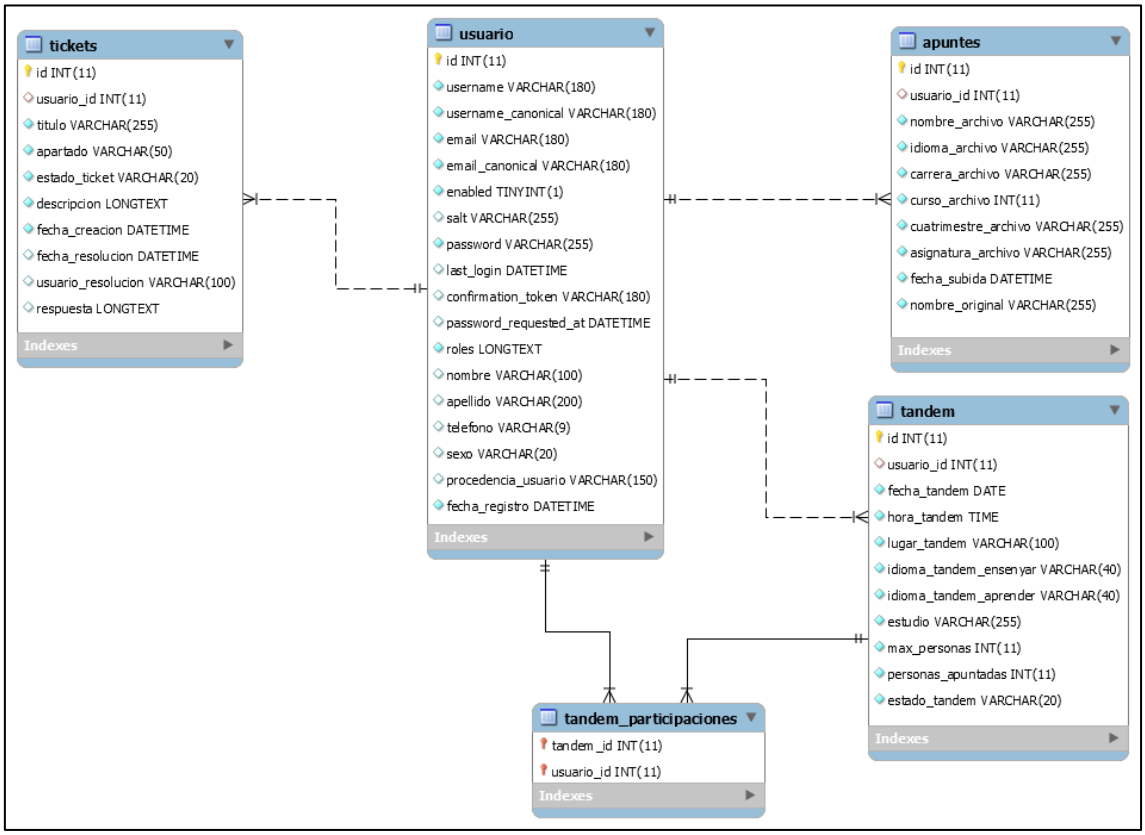


Figura 39 - Diagrama ER de apuntes, tickets y tándems

8.3.4 Tablas detalladas de la base de datos

Apuntes			
Nombre de campo	Tipo de atributo	Nulo	Adicional(NOMBRE TEMPORAL)
id	Integer	No	Clave primaria, autoincrementable
nombreArchivo	String (255)	No	Único
idiomaArchivo	String (255)	No	
carreraArchivo	String (255)	No	
cursoArchivo	Integer	No	
cuatrimestreArchivo	String (255)	No	
asignaturaArchivo	String (255)	No	
fechaSubida	Datetime	No	
nombreOriginal	String (255)	No	
usuario	Integer	No	Clave ajena de Usuario (usuario_id)

Tabla 3 - Tabla de Apuntes de la base de datos

Tandem			
Nombre de campo	Tipo de atributo	Nulo	Adicional(NOMBRE TEMPORAL)
id	Integer	No	Clave primaria, autoincrementable
fechaTandem	Date		
horaTandem	Time		
lugarTandem	String (100)		
idiomaTandemEnsenyar	String (40)		
idiomaTandemAprender	String (40)		
estudio	String (255)		
maxPersonas	Integer		
personasApuntadas	Integer		Default = 1
estadoTandem	String (20)		Default = ‘Pending’
usuario	Integer		Clave ajena de Usuario (usuario_id)
participantes	Integer		Clave ajena de Usuario (usuario_id)

Tabla 4 - Tabla de Tandem de la base de datos

Tickets			
Nombre de campo	Tipo de atributo	Nulo	Adicional(NOMBRE TEMPORAL)
id	Integer	No	Clave primaria, autoincrementable
id	Integer	No	Clave primaria, autoincrementable
titulo	String (255)	No	
apartado	String (50)	No	
estadoTicket	String (20)	No	Default = 'Pending'
descripcion	Text	No	
respuesta	Text	Si	
fechaCreacion	Datetime	No	
fechaResolucion	Datetime	Si	Default = null
usuarioResolucion	String (100)	Si	Default = null
usuario	Integer	No	Clave ajena de Usuario (usuario_id)

Tabla 5 - Tabla de Tickets de la base de datos