UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Herramienta de análisis semántico de la descripción de resultados de aprendizaje en los planes de estudio universitario

Proyecto Final de Carrera

Ingeniería Informática

Autor: Pablo Fuentes Rodríguez

Director: José Luis Poza Luján
Director: José Alberto Conejero Casares

28 de juliol de 2016

Resumen

El diseño de los planes de estudio, especialmente universitarios, se basa en las competencias que se espera que los alumnos adquieran. La adquisición de las competencias requiere de una serie de actividades formativas (antes sólo eran las clases teóricas; ahora hay prácticas, trabajos, etc.). Por medio de los actos de evaluación (antes sólo se hacía por exámenes), se comprueba el nivel de desarrollo que se ha logrado de cada competencia. Para poder ayudar al profesor acerca de qué tipo de competencia se relaciona con cada actividad formativa y con cada acto de evaluación, se propone este proyecto.

El proyecto consiste en la programación de algoritmos de análisis semántico que permitan al profesor determinar, de forma automática, las competencias cubiertas por su asignatura y, a los responsables de títulos, comprobar la coherencia de los planes de estudio basados en competencias.

Palabras clave: Competencias, CMS, gestión, software, educación

Resum

El disseny dels plans d'estudi, especialment universitaris, es basa en les competències que s'espera que els alumnes adquirisquen. L'adquisició de les competències requereix d'una sèrie d'activitats formatives (abans sols eren les classes teòriques; ara hi ha pràctiques, treballs, etc.). Mitjançant els actes d'avaluació (abans sols es feia per exàmens), es comprova el nivell de desenvolupament que s'ha adquirit de cada competència. Per a poder ajudar al professor sobre quin tipus de competència es relaciona amb cada activitat formativa i amb cada acte d'avaluació, es proposa aquest projecte.

El projecte consisteix en la programació d'algorismes d'anàlisi semàntica que permetran al professor determinar, de forma automàtica, les competències cobertes per la seua assignatura i, als responsables de títols, comprovar la coherència dels plans d'estudi basats en competències.

Palabras clave: Competències, CMS, gestió, software, educació

Abstract

The design of a syllabus, especially for university degrees, is based on the expected achievement of competences by the students. Their acquisitions require a series of training activities (before, only theoretical classes, and now, also practices, projects, etc. are considered). The level of achievement in the development of each competence can be measured by performing evaluation acts. Before, these acts were only exams, but now a broad range of acts is considered.

This project is intending for helping teachers to relate competences with training activities and evaluation acts. It involves programming of semantic analysis algorithms that allow teachers to determine, automatically, which competencies are covered by their subjects, and to allow the degrees managers to check the consistency of the curriculum of the degree based on competencies.

Palabras clave: Competencies, CMS, management, software, education

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1 Motivación

En los planes de estudio diseñados por competencias se espera que los alumnos las adquieran y que estarán cualificados para desempeñar un trabajo relacionado con su titulación.

Es importante disponer de herramientas que guíen en la definición de planes de estudios basados en competencias. Estas herramientas pueden asesorar, comparando las competencias a adquirir con las necesidades del entorno socioeconómico.

La herramienta desarrollada en este proyecto está pensada para ayudar a los profesores a formar mejor o de forma más coherente a las futuras generaciones de estudiantes. Además de orientar la docencia a la consecución de competencias.

1.2 Objetivos

El objetivo de este proyecto es desarrollar un sistema que permita a los docentes determinar de forma automática las competencias cubiertas por su asignatura en base a las actividades formativas que el profesor haya definido. Este sistema permitirá al profesor asegurarse de que sus actividades formativas y la forma de evaluarlas corresponde con las competencias que se han establecido para desarrollar en su asignatura en función de la taxonomía de Bloom.

Para desarrollar este sistema se implementará un servidor con la base de datos y el análisis de los datos. Y un cliente que permitirá al profesor interactuar con el servidor y mostrar los datos de forma elegante y ordenada.

1.3 Descripcion del documento

El presente documento esta estructurado en 6 capítulos que detallarán el desarrollo del proyecto. A continuación se especifica el contenido de cada capítulo.

En el capitulo 2 se han buscado sistemas similares y analizado puntos fuertes y características deseables a tener también este proyecto. En el capítulo 3

siguiendo el estándar IEEE 830 [EE84] se determinan los requisitos y la funcionalidad del proyecto, con esto se consigue una completa descripción de lo que hace y lo que no hace el sistema.

En el capítulo 4 antes de implementar el sistema se hará un diseño del sistema, basado en la especificación de requisitos del capítulo anterior. Se mostrará un esquema que engloba el sistema completo y al utilizar una arquitectura de tres capas, se adjuntan los esquemas de las tres capas mediante *mockups*, diagramas entidad relación y de secuencia UML.

En el capítulo 5 basándose en el diseño realizado en el capítulo anterior. Se detallarán la implementación y el desarrollo del proyecto, haciendo uso de capturas de pantalla para validar que se ha cumplido con el diseño especificado. Y se mostrarán la partes de código más interesante y con más peso en la aplicación.

Para acabar el documento en el capítulo 6 se extraerán las conclusiones indicando las dificultades a la hora de llevar a cabo el proyecto. La aportaciones personales a la hora del desarrollo y las posibles acciones por parte de futuros proyectandos.

Capítulo 2

Estudio del entorno

2.1 Introducción

La Real Academia de la Lengua Española define como competencia la «Pericia, aptitud o idoneidad para hacer algo o intervenir en un asunto determinado», dicho en otras palabras la capacidad, habilidad o aptitud de una persona para realizar una tarea o tratar un tema.

Un sistema de gestión es una herramienta software que permite a una entidad la organización de sus recursos, así como avanzar y mejorar en su proyecto.

Un sistema de gestión de competencias o ComMS es una herramienta imprescindible en una empresa o institución, ya que permite gestionar el activo más importante: las personas (recursos humanos). Estos sistemas incluyen, entre otros, la posibilidad de gestionar las competencias, análisis de habilidades o perfiles.

Gracias a esto, una institución puede, una vez definido sus objetivos y sus necesidades, encontrar a la persona que mejor cumple el perfil para alcanzar dichos objetivos. También puede establecer medidas para corregir las diferencias de competencias, entre los perfiles que dispone, y las competencias que necesita para llegar a las metas que se ha propuesto.

Hasta la actualidad, en la educación se ha evaluado a los estudiantes a través de pruebas orales y escritas mediante las cuales se comprueba fundamentalmente el nivel adquirido por el alumno sobre los conocimientos durante el curso académico.

Cada vez más es tendencia un sistema educativo basado en competencias. Como bien indica Elena Cano [Can08] nos encontramos inmersos en la sociedad del conocimiento. Estamos rodeados por información que se crea y se queda obsoleta rápidamente. El ser humano ha de ser capaz de encontrar, seleccionar y procesar la información pertinente a cada situación. Esto, además de competencias transversales como el liderazgo, el trabajo en equipo, el pensamiento critico y la comunicación efectiva, es precisamente lo que plantean los diseños de sistemas educativos basados en competencias.

2.2 Sistemas de gestión de competencias

2.2.1 Nasa CMS

Es un Software desarrollado por la NASA en el 2006 para corregir un problema de balanceo de cargas de trabajo de la Agencia y esfuerzos dedicados por los empleados. Con este software, basado en Web, se busca obtener las competencias de cada uno de los empleados. Los empleados pueden crear y modificar su propio perfil y, de esta manera, la Agencia puede hacer una mejor distribución de los trabajadores entre los proyectos.

El programa pretende facilitar el trabajo al departamento de recursos humanos y que sean capaces de detectar, de la forma más rápida, la necesidad de contratar a más personal.

La NASA tiene su propio diccionario de competencias y las relaciona con una serie de categorías de habilidades. Por ejemplo, la competencia 'Desarrollo de colaboración y negocios' se relaciona con habilidades como la capacidad de detectar la necesidad de colaboraciones u oportunidades de negocios, incluyendo financiación de proyectos y programas

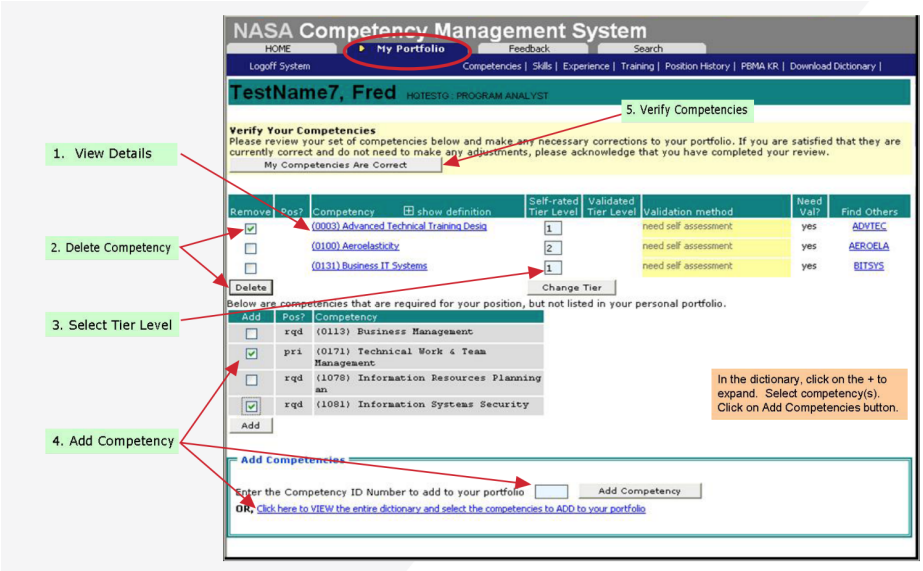

Figura 2.1: Captura de pantalla del software NASA CMS

En la figura 2.1 podemos ver una serie de acciones marcadas con un número, con todas ellas un trabajador puede gestionar sus competencias.

2.2.2 CVPlus Visual

Es un software propietario de la empresa Swiftpro Ltd. En su página web lo venden como una herramienta para la contratación en la que puedes mejorar tus relaciones con el cliente, manejar mejor tu plantilla, optimizar la búsqueda

de nuevos empleados y reducir costes de administración. Es utilizada por más de 1000 empresas en 30 países.

Según CVPlus Visual, los beneficios de este productos son:

1. **Ganar nuevos clientes**
   Proporciona las herramientas que se necesitan para, de forma proactiva, administrar clientes potenciales y convertirlos en clientes valiosos en un futuro.

2. **Construir relaciones duraderas**
   Ayuda a proyectar la imagen correcta cuando te relacionas con clientes o trabajadores para asegurarte los negocios abiertos y adquirir nuevas y valiosas referencias.

3. **Gestión eficaz**
   Con su supervisión del rendimiento, asegura a los directores que toda la información está bajo control.

4. **Alcanzar objetivos**
   Optimiza el tiempo de los encargados de la contratación, asegurando que los candidatos cumplen las expectativas.

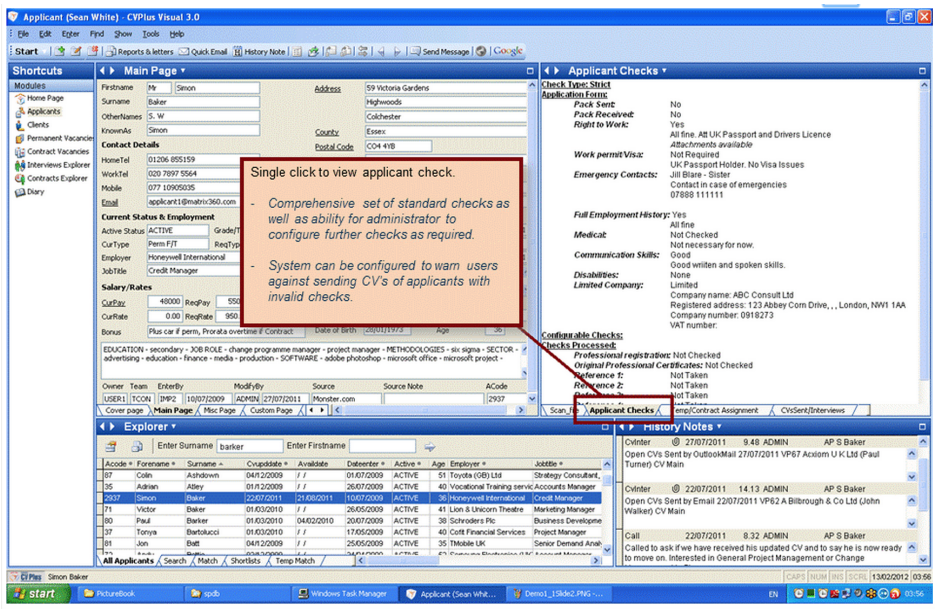

Figura 2.2: Captura de pantalla del software CVPlus Visual

La figura 2.2 es una captura de pantalla que se puede encontrar en la página web del CVPlus Visual. Muestra la vista para gestionar las fichas de los candidatos, editar sus datos y ver si cumple los requisitos que tiene la empresa establecidos.

2.2.3 GestPeople

Es un software desarrollado por una empresa madrileña Animasoft Consulting S.L. Así mismo, es una herramienta de R.R.H.H. que permite gestionar tanto a los empleados como a los clientes mediante competencias y objetivos.

Está dividido en ocho módulos. La mayoría de estos módulos están relacionados con la gestión de personal.

1. Empresa
2. Estructura organizativa
3. Perfiles profesionales
4. Objetivos
5. Evaluación del desempeño
6. Empleados
7. Seguridad
8. Red

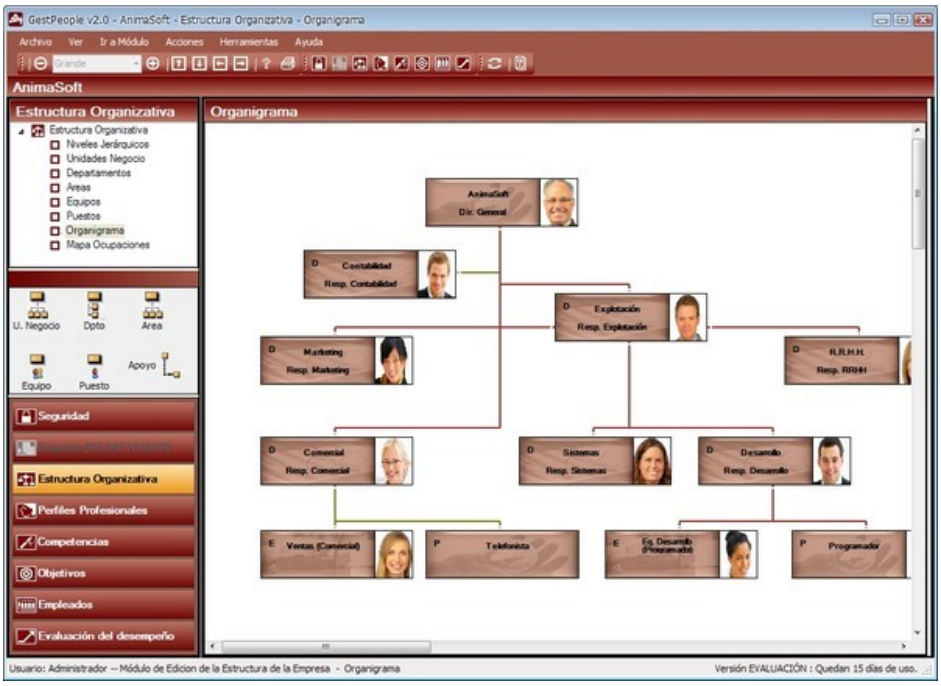

Figura 2.3: Captura de pantalla del software GestPeople

La figura 2.3 nos muestra una captura del organigrama empresarial. Desde este apartado, el responsable de recursos humanos puede dar de alta a todos los empleados de la empresa, asignarles un puesto de trabajo, un rol y establecer una jerarquía entre los trabajadores.

2.2.4 IonCUDOS

Se trata de un software que permite la creación y el diseño de un plan de estudios basado en los requisitos de las empresas o agentes externos a la universidad. De esta forma se pretende que los estudiantes, al acabar sus estudios, tengan las competencias demandadas y, por consecuencia, más posibilidades de incorporarse al mundo laboral.

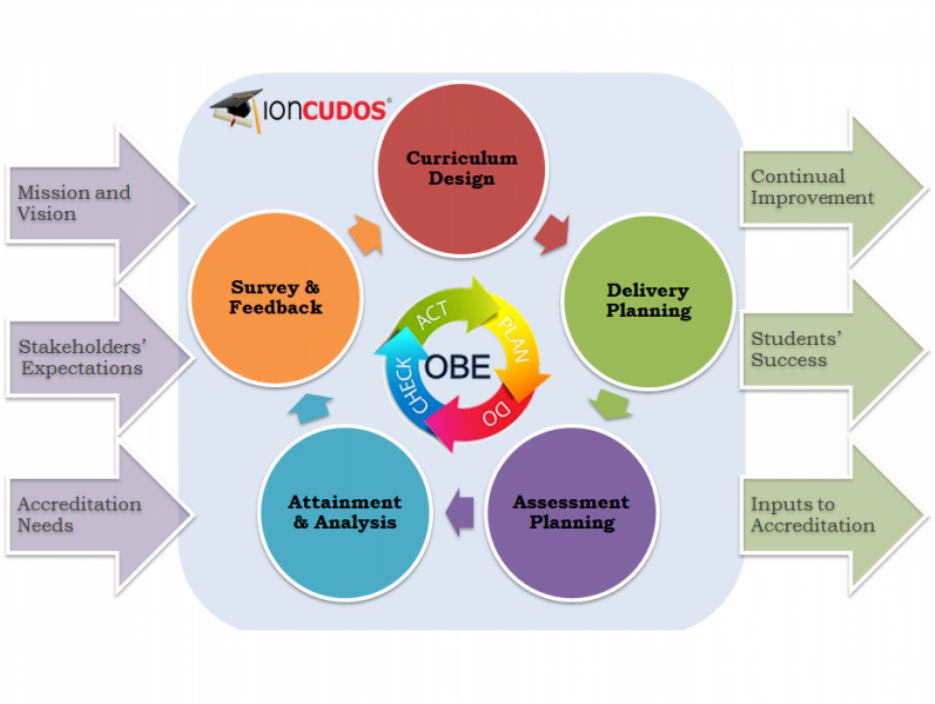

Figura 2.4: Workflow del software IonCUDOS

La figura superior 2.4 muestra el flujo de trabajo de IonCUDOS. Tal y como exponen en su página web, el plan de estudios se elabora en varias iteraciones. La elaboración está liderada por las expectativas, requisitos y comentarios de las empresas y agentes externos que trabajan conjuntamente con la Universidad.

2.3 Análisis

De los sistemas similares, expuestos anteriormente, se han extraído las características más interesantes. Junto a las deseadas para este proyecto, se han creado unas tablas comparativas en base a las cuales se llegará a la conclusión sobre las características que el sistema que debe cumplir.

2.3.1 Análisis cuantitativo

La evaluación de estas características es totalmente objetiva y se basan en la información que proporciona el proveedor en su página web.

1. **C1 - Interfaz web**
   Poder acceder al software con cualquier dispositivo, sistema operativo y navegador.
2. **C2 - Orientado a la educación**
   La gestión de competencias debe estar destinada al ámbito de la educación.
3. **C3 - Uso de taxonomías**
   Utilizar taxonomías para la clasificación de las competencias.
4. **C4 - Actualizado**
   Continua en uso y los desarrolladores siguen manteniendo el proyecto e incluyendo mejoras en él.
5. **C5 - Gratuito**
   El uso de la aplicación no tiene ningún coste para el usuario.
6. **C6 - Open Source**
   El código fuente está disponible para la comunidad, y permiten las aportaciones por parte de terceros.

Característica	Nasa CMS	CVPlus Visual	GestPeople	IonCUDOS
C1	✓	✗	✗	✓
C2	✗	✗	✗	✓
C3	✗	✗	✗	✓
C4	✗	✓	✗	✓
C5	✗	✗	✗	✗
C6	✗	✗	✗	✗

Cuadro 2.1: Análisis cuantitativo

2.3.2 Análisis cualitativo

1. **D1 - Diseño amigable**
   Interfaz gráfica elaborada y actual, siguiendo las últimas tendencias de diseño.
2. **D2 - Moderno**
   Desarrollado utilizando los últimos lenguajes de programación y/o frameworks del mercado.

Característica	Nasa CMS	CVPlus Visual	GestPeople	IonCUDOS
D1	✗	✗	✗	✗
D2	✗	✗	✗	✗

Cuadro 2.2: Análisis cualitativo

2.4 Síntesis

En base al resultado del análisis se han extraído una serie de características deseables para nuestro sistema. Se detallan en la siguiente tabla

Característica	Descripción
CA01 - Basado en web	El acceso a la aplicación debe ser mediante un navegador web compatible con todos los sistemas operativos.
CA02 - Arq. 3 capas	La aplicación tiene que estar diseñada utilizando una arquitectura por capas separando totalmente la persistencia, la lógica y el diseño.
CA03 - RESTful	Con la idea de posibles ampliaciones como aplicación móvil o escritorio. La forma de separar la lógica del diseño es mediante un servidor RESTful.
CA04 - Uso de taxonomías	Utilizar la taxonomía de Bloom para la clasificación de los resultados de aprendizaje.

Cuadro 2.3: Características del sistema

2.5 Tecnología a utilizar

En primer lugar, para desarrollar la parte de la vista (*fronted*) del proyecto, se ha elegido AngularJS [Angb] que es un framework javascript con el que poder crear complejas aplicaciones web del lado del cliente. Utiliza el patrón MVC y cuenta con potentes librerías que permiten, de forma muy fácil, la conexión con un servidor RESTful para obtener los datos que mostrar al cliente.

A continuación, para la base de datos, se ha elegido PostgreSQL [Pos] que es un motor de base de datos SQL muy utilizado en los proyectos Open Source. Lleva más de 15 años en el mercado, funciona en los sistemas operativos más utilizados y los lenguajes de programación más comunes tienen conectores para poder trabajar con PostgreSQL.

Finalmente, para la lógica se ha elegido Django [Dja] que es un framework de desarrollo web escrito en Python. Django utiliza una técnica de programación llamada ORM que convierte los objetos del lenguaje de programación en objetos de la base de datos, y viceversa. Gracias a esta técnica es posible olvidar la gestión de la base de datos ya que Django se encarga de ello. También dispone de la librería *Django REST framework* [fra] que es una gran herramienta para crear un servicio RESTful.

2.6 Conclusiones

En base al resultado de análisis de sistemas similares, la conclusión principal a la que se ha llegado es que no hay muchas herramientas. Las que hay son antiguas, sin soporte y no cumplen las características deseadas para este proyecto.

El único programa que tiene un gran parecido y es utilizado para trabajar en el sector de la educación es IonCUDOS. Los otros programas están pensados para el sector empresarial y lo único que comparten es la gestión de las competencias.

Con estos datos se demuestra que hay un gran sector por cubrir y que, aparentemente, no se está trabajando en ello (excepto IonCUDOS). Por esta causa, es una buena idea seguir hacia delante con el proyecto y dedicarle esfuerzos.

Capítulo 3

Especificación de requisitos

3.1 Introducción

El sistema permitirá al equipo docente consultar la taxonomía de Bloom, así como, visualizar los resultados de aprendizaje de las asignaturas. Además, cabe la posibilidad de añadir nuevos resultados de aprendizaje.

El objetivo del sistema es analizar los resultados de aprendizaje y clasificarlos en una categoría de la taxonomía de Bloom (en el nivel cognitivo). Así, a partir del análisis, el profesor podrá saber si sus resultados de aprendizaje y sus actos de evaluación cubrirán las competencias asignadas a su asignatura.

3.1.1 Propósito

Con la especificación de requisitos se pretende determinar completamente la descripción del sistema que se va a desarrollar. En la primera parte se hablará del ámbito, personal involucrado, definiciones, acrónimos y abreviaturas; en la segunda parte, se hará una descripción general del producto dejando para la parte final la especificación de requisitos.

3.1.2 Ámbito

Este proyecto es un componente de otro proyecto piloto interno de la ETSINF más grande llamado *Evalua2*.

3.1.3 Personal involucrado

En este apartado se va a mostrar a las personas involucradas en el proyecto, indicando:

- **Nombre**: Nombre de la persona involucrada en el proyecto.
- **Rol**: Papel que desesarrolla esta persona en el proyecto.
- **Categoría profesional**: Cargo que ocupa la persona
- **Responsabilidades**: De que tareas se encarga

- **Información de contacto**: Dirección de correo para poder contactar con esa persona

Nombre	Pablo Fuentes Rodríguez
Rol	Desarrollador
Categoría profesional	Estudiante
Responsabilidades	Desarrollar el sistema
Información de contacto	pabfuero@ei.upv.es

Cuadro 3.1: Miembro Pablo Fuentes

Nombre	José Luis Poza Luján
Rol	Director
Categoría profesional	Profesor
Responsabilidades	Dirigir el proyecto
Información de contacto	jopolu@disca.upv.es

Cuadro 3.2: Miembro José Luis Poza

Nombre	José Alberto Conejero Casares
Rol	Director
Categoría profesional	Profesor
Responsabilidades	Dirigir el proyecto
Información de contacto	aconejero@upv.es

Cuadro 3.3: Miembro Jose Alberto Conejero

3.1.4 Definiciones, acrónimos y abreviaturas

- **UPV**: Universidad Politécnica de Valencia.
- **ETSINF**: Escuela Técnica Superior de Ingeniería Informática de la UPV.
- **IEEE**: Institute of Electrical & Electronics Engineers.
- **Taxonomía**: Ciencia que trata de los principios, métodos y fines de la clasificación. Se aplica para la ordenación jerarquizada y sistemática.
- **Resultado de aprendizaje**: Declaración de lo que el estudiante se espera que conozca, comprenda y sea capaz de hacer al finalizar un período de aprendizaje.
- **Acto de evaluación**: Tarea que realiza el estudiante para ser evaluado.
- **Sistema**: Producto software, resultado de este proyecto final de carrera.
- **REST**: Es un estilo de arquitectura software para servidores web que utiliza peticiones HTTP para obtener o indicar la ejecución de operaciones sobre los datos del servidor.
- **RESTful**: Sistema que sigue los principios REST

- **Framework**: Es una estructura conceptual y tecnológica con módulos o artefactos ya definidos que sirven de base para la organización y el desarrollo de un desarrollo software.
- **JSON**: Es un formato de text ligero para el intercambio de datos.
- **HTML**: Lenguaje de marcada que se utiliza para la elaboración de páginas web.
- **CSS**: Lenguaje usado para definir y crear el estilo y la presentación de un documento HTML.
- **Javascript**: Lenguaje de programación interpretado, utilizado en la parte del navegador web, que permite mejoras de funcionalidad en las páginas web.
- **AngularJS**: Framework Javascript que se utilizará en este proyecto.
- **Python**: Lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca el código legible.
- **Django**: Framework Python que se utilizará en este proyecto.
- **IDE**: Entorno de desarrollo dinámico, aplicación informática que proporciona servicios integrales para facilitarle al programador el proceso del desarrollo software

3.2 Descripción general

3.2.1 Perspectiva del producto

El sistema se trata de una interfaz web que se encarga de mostrar los datos al usuario de forma elegante y ordenada. Los datos se obtienen mediante peticiones a un servidor RESTful que contiene la lógica y la base de datos. A continuación se van a detallar los requisitos necesarios que se han de cumplir para que el sistema sea considerado un producto válido.

3.2.2 Funcionalidad del producto

Como se viene indicando durante todo el documento, el proyecto está enfocado al personal docente de la UPV. Así que solo se dispondrá de un tipo de usuario en el sistema (profesor) y, a continuación, mediante diagramas de casos de uso, se indicarán las funciones que el profesor podrá realizar:

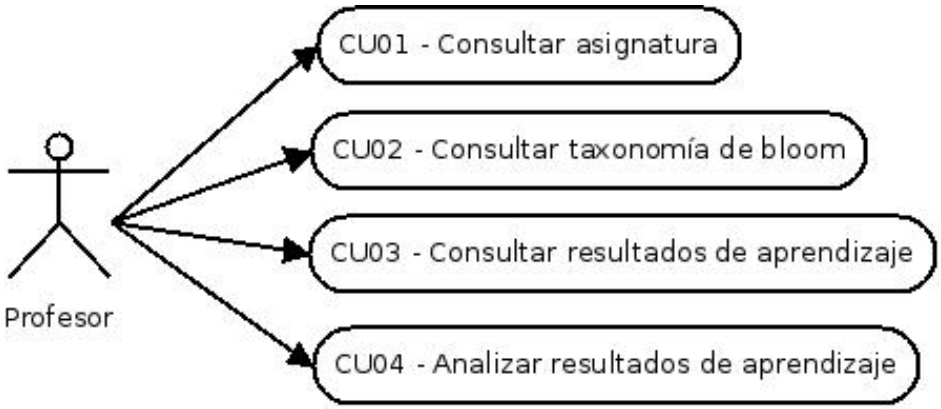

Figura 3.1: Casos de uso

Caso de uso	Descripción
CU01	El profesor podrá acceder y buscar todas las asignaturas de la carrera agrupadas por cursos. La vista hará una llamada al servidor el cual hará todos los cálculos necesarios y devolverá los datos para que puedan ser pintados.
CU02	El profesor podrá acceder y buscar todos los verbos de la taxonomía agrupados por las categorías a las que pertenecen.
CU03	El profesor podrá ver el listado de resultados de aprendizaje de cada asignatura, filtrarlo y consultar el resultado de aprendizaje que desee.
CU04	El profesor podrá subir un fichero en formato JSON con resultados de aprendizaje para ser analizado y clasificado por el servidor y poder ver el resultado por pantalla.

Cuadro 3.4: Casos de uso

3.2.3 Características del usuario

El sistema solo cuenta con un tipo de usuario, el usuario profesor quien podrá realizar toda la funcionalidad que ha sido expresada en el apartado anterior 3.2.2

3.2.4 Restricciones

Para poder utilizar el sistema no existen restricciones pero sí hay tres requisitos a cumplir. Disponer de un ordenador con sistema operativo Windows, Linux o Mac, el ordenador debe tener un navegador web moderno y debe estar conectado a Internet.

3.3 Requisitos específicos

El sistema que se va a desarrollar se trata de un prototipo, puesto que dispone de pocos requisitos y todos son altos o esenciales. En el último capítulo se

detallarán las posibles ampliaciones de este prototipo.

3.3.1 Requisitos no funcionales

Número	RE01			
Nombre	Accesibilidad web			
Tipo	Usuario	✓Sistema	Funcional	Restricción
Descripción	El sistema ha de adaptarse a todas las pantallas de ordenador permitiendo que la experiencia del usuario sea satisfactoria.			
Prioridad	✓Alta/Esencial		Media/Deseado	Baja/Opcional

Cuadro 3.5: Requisito no funcional RE01

Número	RE02			
Nombre	Desarrollo de la vista			
Tipo	Usuario	✓Sistema	Funcional	Restricción
Descripción	La vista del sistema tiene que estar desarrollada con HTML, CSS y Javascript. El framework Javascript a utilizar debe ser AngularJS			
Prioridad	✓Alta/Esencial		Media/Deseado	Baja/Opcional

Cuadro 3.6: Requisito no funcional RE02

Número	RE03			
Nombre	Desarrollo de la lógica			
Tipo	Usuario	✓Sistema	Funcional	Restricción
Descripción	La lógica del sistema tiene que estar desarrollada en el lenguaje de programación Python. Concretamente con el framework Django.			
Prioridad	✓Alta/Esencial		Media/Deseado	Baja/Opcional

Cuadro 3.7: Requisito no funcional RE03

Número	RE04			
Nombre	Desarrollo de la persistencia			
Tipo	Usuario	✓Sistema	Funcional	Restricción
Descripción	Para la capa de persistencia el sistema tiene que utilizar el sistema de gestión de base de datos relacionales PostgreSQL.			
Prioridad	✓Alta/Esencial		Media/Deseado	Baja/Opcional

Cuadro 3.8: Requisito no funcional RE04

3.3.2 Requisitos funcionales

Número	RE05			
Nombre	Listado de asignaturas			
Tipo	Usuario	Sistema	✓Funcional	Restricción
Descripción	El usuario podrá consultar todas las asignaturas de la carrera agrupadas por cursos, además se dispondrá de un buscador con el que poder filtrar las asignaturas.			
Prioridad	✓Alta/Esencial		Media/Deseado	Baja/Opcional

Cuadro 3.9: Requisito no funcional RE05

Número	RE06			
Nombre	Taxonomía de Bloom			
Tipo	Usuario	Sistema	✓Funcional	Restricción
Descripción	El usuario podrá consultar todos verbos de la taxonomía de Bloom agrupados por categoría, además se dispondrá de un buscador con el que poder filtrar los verbos.			
Prioridad	✓Alta/Esencial		Media/Deseado	Baja/Opcional

Cuadro 3.10: Requisito funcional RE06

Número	RE07			
Nombre	Analizar fichero resultados de aprendizaje			
Tipo	Usuario	Sistema	✓Funcional	✓Restricción
Descripción	El usuario, desde la interfaz, podrá subir un fichero en formato JSON al servidor. El servidor lo analizará y devolverá el resultado a la vista para que pueda ser mostrado de forma elegante y ordenada al usuario.			
Prioridad	✓Alta/Esencial		Media/Deseado	Baja/Opcional

Cuadro 3.11: Requisito funcional RE07

3.3.3 Otros requisitos

Número	RE08			
Nombre	Copias de seguridad			
Tipo	Usuario	✓Sistema	Funcional	Restricción
Descripción	Al estar todos los datos almacenados en la base de datos, es altamente recomendable crear un sistema de copias de seguridad diarios. Para evitar que si hubiera algún fallo no se pierdan datos.			
Prioridad	Alta/Esencial		Media/Deseado	✓Baja/Opcional

Cuadro 3.12: Otros resquisitos RE08

3.3.4 Conclusiones

Para realizar esta especificación de requisitos se ha seguido el estándar IEEE 830 [EE84]. Los requisitos se han establecido a partir de la tecnología a utilizar definida en el apartado 2.5 y de los casos de uso indicados en el apartado 3.4. A su vez, estos casos de uso han sido definidos siguiendo las características del apartado 2.3 del capítulo anterior.

Capítulo 4

Diseño del sistema

4.1 Introducción

Después de haber realizado la especificación de requisitos y usar de referencia los requisitos funcionales indicados en el apartado 3.3.2, se va a definir formal y conceptualmente los diferentes elementos necesarios para llevar a cabo el sistema.

4.2 Especificación formal

El diseño de la aplicación se basa en la arquitectura cliente-servidor de programación por capas.

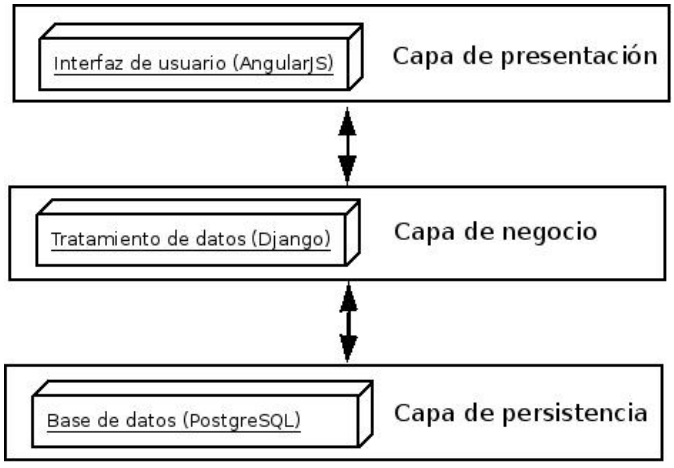

Figura 4.1: Arquitectura programación por capas

En esta sección se detallarán las tres capas con las que cuenta el sistema.

4.2.1 Capa presentación

La capa de presentación es la que permite la interacción entre el usuario y el sistema. A partir de esta interfaz gráfica el usuario podrá navegar entre las diferentes vistas de las que se compone el sistema para poder acceder a toda la información del sistema.

A continuación se muestra, mediante un diagrama de navegabilidad, una propuesta de disposición de la interfaz gráfica diseñada.

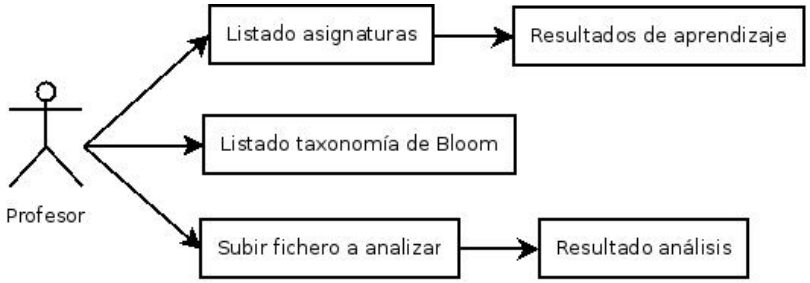

Figura 4.2: Diagrama de navegabilidad

Para el diseño de la interfaz se va a utilizar la librería Angular Material [Anga], sus componentes visuales siguen las especificaciones de Google Material Design [Des].

Figura 4.3: Propuesta disposición interfaz

Como se puede observar en la figura superior 4.3 el diseño de la aplicación es sencillo e intuitivo. La vista esta formada por tres bloques claramente diferenciados.

Una columna lateral con los elementos del menú para poder navegar, una fila superior con la barra de navegación para que el profesor sepa siempre en que apartado se encuentra y en el espacio del centro se muestra toda la información.

4.2.2 Capa negocio

En este apartado, con el objetivo que se pueda ver gráficamente el flujo de trabajo del sistema, se va a mostrar un diagrama de secuencia por cada caso de uso especificado en el apartado 3.2.2

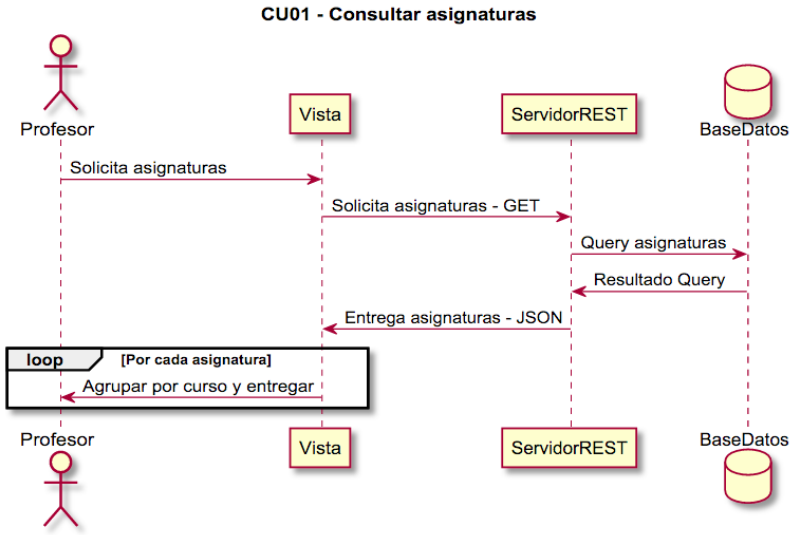

Figura 4.4: Diagrama secuencia CU01

El diagrama de la figura 4.4 muestra como el profesor podrá acceder a las asignaturas. Al acceder al menú de las asignaturas, se mostrarán todas las asignaturas de la carrera agrupadas por cursos.

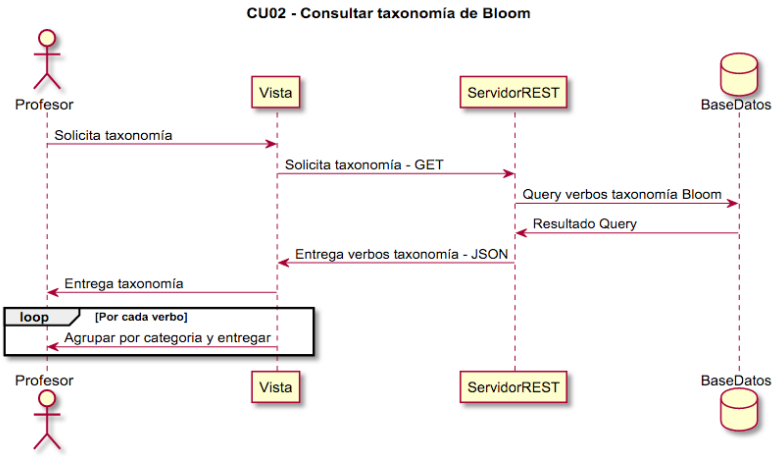

Figura 4.5: Diagrama secuencia CU02

El diagrama de la figura 4.5 muestra como el profesor podrá acceder a la taxonomía de Bloom. Al acceder al menú Taxonomía de Bloom, se mostrarán todos los verbos de la taxonomía agrupados por categoría.

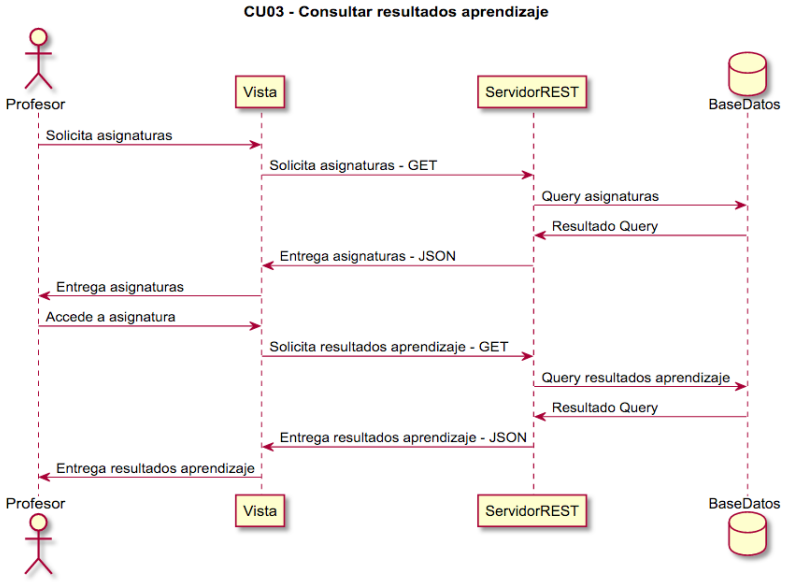

Figura 4.6: Diagrama secuencia CU03

El diagrama de la figura 4.6 muestra como el profesor podrá acceder a los resultados de aprendizaje. Primero deberá elegir que asignatura de entre todas las de la carrera quiere consultar y, en los detalles de esta asignatura, se encontrará el listado de resultados de aprendizaje vinculados a ella.

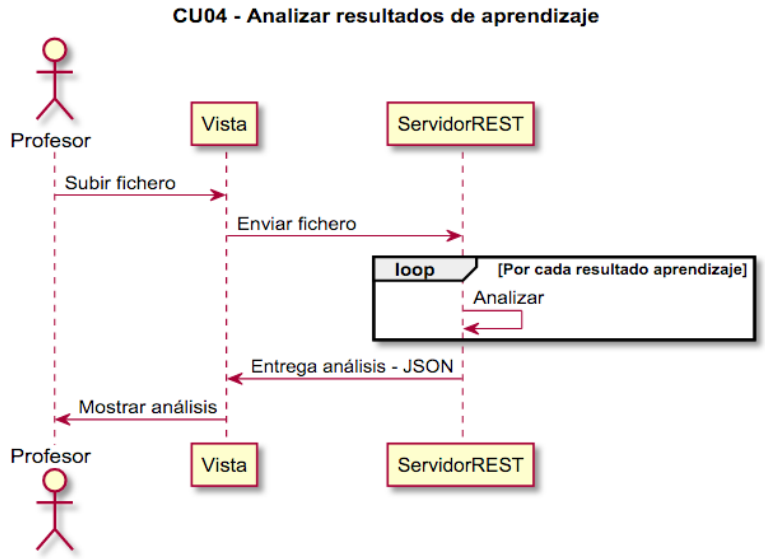

Figura 4.7: Diagrama secuencia CU04

El diagrama de la figura 4.7 muestra como el profesor podrá analizar sus propios resultados de aprendizaje. Al subir un fichero en formato JSON, la lógica lo analizará y posteriormente se mostrarán los resultados por pantalla.

4.2.3 Capa persistencia

Para el sistema se ha creado una base de datos que contendrá toda los datos necesarios para el correcto funcionamiento del sistema. El siguiente diagrama entidad relación muestra su estructura.

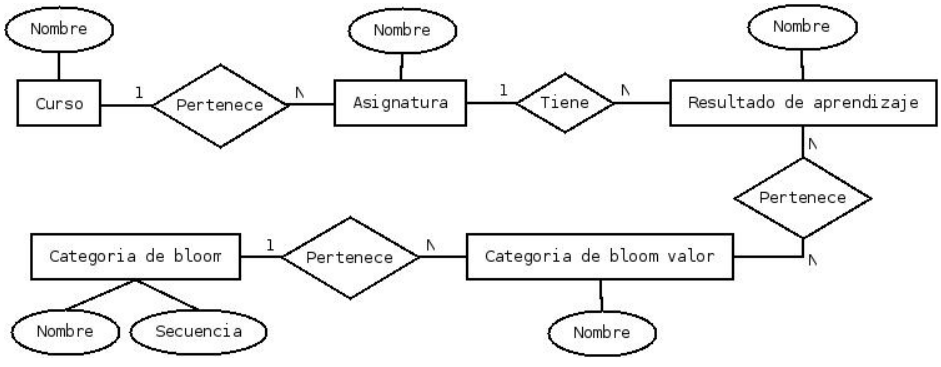

Figura 4.8: Diagrama entidad relación base de datos

Como se puede observar en la figura superior 4.8, la estructura de la base de datos es muy sencilla, pero es lo necesario para este sistema.

4.3 Conclusiones

En este capítulo se ha hablado del diseño de la aplicación, especificando un esquema general del sistema. Posteriormente se ha mostrado el diseño de la capa visual y el diagrama de secuencia de las capas de lógica y persistencia. Con todo el diseño del sistema hecho y utilizándolo como guía se procederá a la implantación en el capítulo siguiente.

Capítulo 5

Implementación e implantación

5.1 Introducción

Una vez diseñado el sistema en el capítulo anterior, en este se procederá a especificar su implantación. Se explicará la tecnología utilizada en el sistema, que ya ha sido introducida al lector en el apartado 2.5 entrando más en detalle y completando el capítulo de una manera más técnica.

También se explicarán las herramientas utilizadas para el desarrollo del sistema. Mostrando capturas reales del producto final y partes de código más significativas.

5.2 Implementación

Para el desarrollo se ha utilizado el IDE de desarrollo Pycharm [Pyc] que es una herramienta muy utilizada por los desarrolladores Python. Tiene dos versiones la Profesional (de pago) y la Community (gratuita). Como es evidente la gratuita tiene menos funcionalidad que la de pago, pero para el desarrollo de este proyecto la versión Community ha sido más que suficiente.

5.2.1 Capa presentación

Esta capa está desarrollada con HTML, CSS y AngularJS y es una interfaz que, sin conexión a la capa lógica, no funciona ya que no tiene datos. Todos los datos utilizados para poder mostrar la vista correctamente se obtienen mediante llamadas GET y POST a la REST API que hay implementada en la capa lógica. Gracias a la potencia de AngularJS esto se puede a hacer en muy pocas lineas con el uso de Factories.

Las Factories son contenedores de código que se pueden utilizar entre los controladores y tienen la característica de ser fragmentos de código que solo se instancian una vez, por lo que no pierden su estado.

El siguiente fragmento de código es el que se encarga de la conexión entre la vista y la lógica

```
angular.module('evalua2')
.factory('CursoFactory', function($resource){
    return $resource('http://localhost:8000/api/v1/curso/:id');
})
.factory('AsignaturaFactory', function($resource){
    return $resource('http://localhost:8000/api/v1/asignatura/:id');
})
.factory('CategoriaBloomFactory', function($resource){
    return $resource('http://localhost:8000/api/v1/categoria-bloom/:id');
})
.factory('CategoriaBloomValorFactory', function($resource){
    return $resource('http://localhost:8000/api/v1/categoria-bloom-valor/:id');
})
.factory('AnalizarFactory', function ($resource) {
    return $resource('http://localhost:8000/api/v1/analizar/', {}, {
        save: {method: 'POST',isArray: true}
    });
});
```

A continuación, se muestran unas capturas de pantalla de cada uno de los apartados del cliente web para verificar que se cumple el diseño planteado en el apartado 4.2.1

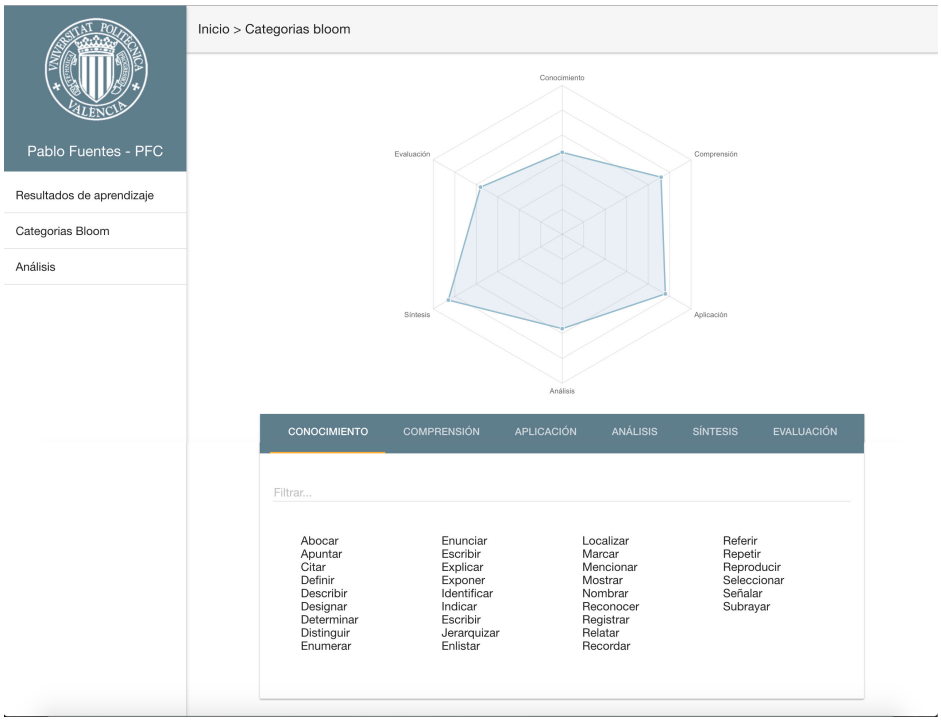

Figura 5.1: Captura vista taxonomía de Bloom

En la figura 5.1 se muestra un gráfico de radar que expone de forma muy visual qué categoría tiene mayor número de verbos asociados. Este gráfico esta desarrollado utilizando la librería Angular Chart [Chaa] que, a su vez utiliza para sus directivas la librería ChartJS [Chab] que es una libreria Javascript open source que permite crear gráficos utilizando el componente canvas de HTML.

El siguiente fragmento de código HTML muestra lo rápido que es generar una gráfica utilizando esta librería

```
<canvas id="categorias-chart" class="chart chart-radar"
        chart-data="data" chart-labels="labels" chart-series="series">
</canvas>
```

Las variables data, labels y series son listas que están definidas en el controlador de la vista, se inicializan a listas vacías y son rellenadas cuando la lógica devuelve los valores

```
$scope.labels = [];
$scope.data = [[]];
$scope.values = [];
CategoriaBloomFactory.query(function(categorias){
        angular.forEach(categorias, function(categoria){
                $scope.labels.push(categoria.name);
                $scope.data[0].push(categoria.values.length);
                $scope.values.push(categoria.values);
        });
});
```

Figura 5.2: Captura vista asignaturas

La figura superior 5.2 muestra el listado de asignaturas agrupadas por curso. Al pulsar sobre una de ellas se accede al detalle de la asignatura mostrando todos su resultados de aprendizaje. Para obtener el listado de asignaturas, la vista hace una petición GET a la url `http://localhost:8000/api/v1/curso/`

```
// Directiva vista listado asignaturas por curso
$scope.cursos = [];
$scope.asignaturas = [];
CursoFactory.query(function(cursos){
        angular.forEach(cursos, function(curso){
                $scope.cursos.push(curso.name);
                $scope.asignaturas.push(curso.values);
        });
});
```

Figura 5.3: Captura vista resultados aprendizaje

En la figura superior 5.3 se ve una tabla con cada uno de los resultados de aprendizaje y el número de ocurrencias que contiene la frase en cada una de las categorías de la taxonomía de Bloom.

Como se puede ver en la siguiente figura 5.4 al pasar el ratón sobre una de las palabras que pertenece a la taxonomía aparece un cartel indicando a que categoría pertenece.

Figura 5.4: Captura vista resultados aprendizaje en detalle

Por ultimo, queda por mostrar la vista de análisis, en la que aparece un botón grande y al hacer click sobre el botón se puede elegir el fichero que se ha de analizar.

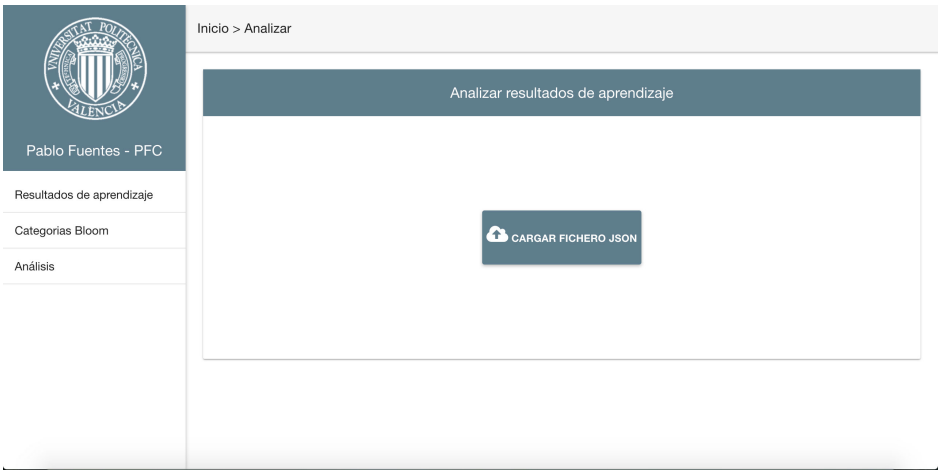

Figura 5.5: Captura vista analizar

Una vez analizado el fichero por el servidor, se muestra un listado como el de la figura 5.3 y, al pasar el ratón sobre las palabras, se muestra el mismo detalle que en la vista 5.4

5.2.2 Capa lógica

Esta capa es la encargada de servir los datos y realizar los cálculos. Todo esto ha sido desarrollado con Django.

Los proyectos Django se componen de apps, que son paquetes de código. La filosofía de Django es que un app solo realice una funcionalidad y un conjunto de apps (funcionalidades) generen un proyecto. Las apps pueden ser desarrollos

propios o de terceros. A continuación se detallan las apps utilizadas en este proyecto.

core

Esta aplicación es la base del proyecto y sobre la que construir posibles ampliaciones. Contiene los modelos básicos con los que trabajar

- Curso
- Asignatura
- CategoriaBloom
- CategoriaBloomValor
- ResultadoAprendizaje

En esta app solo están definidos los modelos y sus campos, nada más.

restapi

Esta aplicación es la base de la comunicación entre la lógica y la vista. Aquí están definidos los Serializers que son modelos que se encargan de convertir los objetos ORM a objetos JSON y viceversa. Hay un modelo serializador por cada modelo definido en la app core 5.2.2

- CursoSerializer
- AsignaturaSerializer
- CategoriaBloomSerializer
- CategoriaBloomValorSerializer
- ResultadoAprendizajeSerializer

Después están definidas las URLs de la API en la que el el servidor escucha peticiones. Django, al recibir una petición en una URL, ejecuta una función asociada a esa URL. A continuación, un ejemplo de la vista asociada al listado completo de cursos.

```
# Peticion GET a http://127.0.0.1:8000/api/v1/curso/
@api_view(['GET'])
def curso_list(request):
    if request.method == 'GET':
        cursos = Curso.objects.all()
        serializer = CursoSerializer(cursos, many=True)
        data = []
        for c in serializer.data:
            serializer_vals = AsignaturaSerializer(Asignatura
                                        .objects
```

```
                                        .filter(curso_id=c['id']),
                                        many=True)
            c['values'] = sorted(serializer_vals.data,
                                 key=lambda k: k['name'])
            data.append(c)
        return Response(data)
```

Este código se ejecuta al recibir una petición GET en la URL `http://127.0.0.1:8000/api/v1/curso/`. Por cada curso lo serializa, busca sus asignaturas y también las serializa y, después, las añade al objeto curso para más tarde devolver una lista como resultado

```
@api_view(['GET'])
def asignatura_list(request):
        if request.method == 'GET':
        asignaturas = Asignatura.objects.all()
        serializer = AsignaturaSerializer(asignaturas, many=True)
        return Response(serializer.data)
```

El código superior muestra como devolver todas las asignaturas que hay en la base de datos. Como se puede observar el procedimiento es muy fácil y rápido para el desarrollador en objetos que no requieren procesamiento. Django y rest-framework 5.2.2 se encargan de todo.

```
def analyze_ra(ra_name):
    """
    Comparamos cada palabra del resultado de aprendizaje con todos los verbos
    de las categorías de bloom.
    """
    res = {
        'phrase': [],
        'categs': {c.name: 0 for c in CategoriaBloom.objects.all()}
    }

    # Quitamos acentos y pasamos a minusculas
    unaccent_categs = {unicodedata
                .normalize('NFKD', c)
                .encode('ASCII', 'ignore').lower(): c
                for c in res['categs']}

    # Mientras no haya ningún token que pertenezca a alguna categoria de bloom
```

```
    # reconstruimos la frase en esta variable
    phrase = ''
    for token in ra_name.split(' '):
        unaccent_token = unicodedata
                .normalize('NFKD', token)
                .encode('ASCII','ignore').lower()
        cat_bloom = CategoriaBloomValor.objects.filter(name__iexact=token)
        if cat_bloom or unaccent_token in unaccent_categs.keys():
            # Guardamos la parte de la frase reconstruida hasta ahora
            if phrase:
                res['phrase'].append([phrase])
                phrase = ''
            tokens = [token]
            # Si el token actual coincide con algun verbo de bloom
            # añadimos a la lista las categorias de los verbos
            # y añadimos un ocurrencia más al contador de categorias.
            if cat_bloom:
                for c in cat_bloom:
                    res['categs'][c.categ_id.name] += 1
                    tokens.append([c.categ_id.name])
            else:
                # Si el token es una de las categorias y no es un verbo
                tokens.append(token)
                res['categs'][unaccent_categs[unaccent_token]] += 1

            res['phrase'].append(tokens)
        else:
            phrase = '%s %s' % (phrase, token) if phrase else token
    if phrase:
        res['phrase'].append([phrase])

    return res
```

Esta función es la encargada del análisis de los resultados de aprendizaje. Recibe por parámetro una frase y compara cada palabra de la frase con todos los verbos de la taxonomía de Bloom.

Si hace 'matching' entonces suma +1 en el contador por categorías y marca en la frase a que categoría pertenece este verbo.

El resultado es una lista en la que sus elementos son también listas de uno o más elementos. El primer elemento de esta sublista es la palabra y el resto de elementos de las sublistas (si los hay), son las categorías de la taxonomía de Bloom a las que pertenece esa palabra.

restframework

Esta es una aplicación de terceros [fra]. Es una herramienta que proporciona todo lo necesario para empezar a desarrollar una REST API.

- Página web desde la que hacer consultas a la API. Sirve de gran ayuda a la hora de empezar a programar.
- Varias políticas de autenticación entre las incluidas OAuth1a y OAuth2.
- Serialización de fuentes de datos ORM y no-ORM.
- Extensa documentación, con muchos ejemplos.
- Usada y reconocida por muchas empresas como Mozilla, Red Hat y Heroku.

corsheaders

Es una aplicación de terceros [Hea] que complementa a la app rest-framework 5.2.2 añade headers CORS (Cross-Origin Resource Sharing) a las respuestas HTTP.

Con corsheaders se puede añadir una lista blanca de dominios o expresiones regulares a rutas a las que si permitir CORS y así saltarnos la política del mismo origen que utilizan los navegadores, si no las peticiones AngularJS a la API darían error y no se podría acceder a los datos.

Para este proyecto se ha configurado una expresión regular que acepte CORS a todos los dominios si la ruta contine /api/v1/

5.2.3 Capa persistencia

Para la base de datos, la única gestión por parte del desarrollador que se ha de hacer es tener PostgreSQL instalado y crear la base de datos tal y como se explica en el apartado 5.3.1

Esto es gracias al object-relational mapper (ORM) de Django, que se encarga de crear las tablas y campos de la base de datos, así como también se encarga de leer y guardar los datos en la base de datos sin que el desarrollador tenga que preocuparse.

Este nivel de abstracción facilita el trabajo del desarrollador puesto que Django permite trabajar con PostgreSQL, MySQL y SQL lite y el desarrollador no necesita saber trabajar con todos los motores de base de datos puesto que Django ya lo hace por él.

5.3 Implantación

El proyecto se encuentra en un repositorio de acceso público `https://github.com/fuentes010/evalua2`, que pertenece a Pablo Fuentes, el proyectando que desarrolla este proyecto.

5.3.1 Instalación

Para poder utilizar el sistema hay que seguir unos sencillos pasos. En primer lugar, instalar git en la máquina Linux o Mac que vaya a ejecutar el proyecto y, posteriormente, clonar el repositorio en el directorio que se desee.

```
~ git clone https://github.com/fuentes010/evalua2.git
```

Tener instalado PostgreSQL. Se puede hacer mediante el gestor de paquetes en el caso de los sistemas operativos Linux

```
# Archlinux
~ sudo pacman -S postgresql

# Fedora
~ sudo yum install postgresql-server postgresql-contrib

# Debian, Ubuntu
~ sudo apt-get install postgresql postgresql-contrib
```

Para sistema operativo Mac la forma más cómoda es utilizar la aplicación Postgres.app que se puede descargar en `http://postgresapp.com/`

Y crear una base de datos nueva para el proyecto.

```
~ createdb pfc
```

Como último paso para tener la base de datos correctamente, hay que ir hasta el fichero de configuración del proyecto que se encuentra en la ruta backend/evalua2/settings.py y establecer en la variable DATABASES (línea 101) la conexión a la base de datos.

```
DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.postgresql',
            'NAME': 'NOBRE DE LA BASE DE DATOS',
            'USER': 'TU USUARIO CON PERMISOS DE ACCESO A POSTGRESQL',
            'PASSWORD': 'TU CONTRASENA',
            'HOST': '127.0.0.1',
            'PORT': '5432',
            }
}
```

También es recomendable instalar un software llamado virtualenvwrapper [wra]. No es obligatorio pero es altamente recomendable. Este software permite tener múltiples entornos virtuales de Python y, en cada uno de estos entornos, tener diferentes librerías y diferentes versiones de estas. Esto a la hora de programar es de gran ayuda, ya que nunca tendremos conflictos entre proyectos causados por incompatibilidades de versiones ni entre librerías. Cada entorno utiliza las suyas.

Para instalarlo podemos usar el gestor de paquetes de nuestro sistema operativo o pip, que es un gestor de paquetes software escritos en Python

```
# Archlinux
~ sudo pacman -S python-virtualenvwrapper

# Fedora
~ sudo dnf install python-virtualenvwrapper

# Debian, Ubuntu
~ sudo apt-get install virtualenvwrapper

# Linux, OS X
~ sudo pip install virtualenvwrapper
```

Estos son los comandos para poder trabajar con entornos virtuales

```
#  Crear un entorno virtual llamado pfc
~ mkvirtualenv pfc

# Borrar un entorno virtual llamado pfc
~ rmvirtualenv pfc

# Activar un entorno virtual llamado pfc
~ workon pfc

# Desactivar el entorno en el que se esta trabajando
~ deactivate
```

Una vez creado y activado el entorno virtual se procederá a instalar las librerías Python necesarias para que el proyecto funcione. Para ello hay que ir hasta la raíz del proyecto clonado e instalarlas mediante el gestor de paquetes pip

```
(evalua2) ~ pip install -r requirements.txt
```

Todas las librerías necesarias se encuentran en el fichero requirements.txt

Con todo esto, la máquina ya estará lista y tendrá todos los requisitos necesarios para poder ejecutar perfectamente el proyecto.

5.3.2 Ejecución

Una vez instalado el proyecto hay que tener dos sesiones de la terminal abiertas, una para ejecutar el proceso del frontend y otra para ejecutar el proceso del backend .

Backend

En la primera de las terminales hay que ir hasta la raíz del proyecto clonado, dentro hay dos carpetas frontend y backend. Entrar en la de backend y ejecutar

```
(evalua2) ~ python manage.py migrate
```

Este comando se encarga de hacer todas las migraciones del proyecto, crea todos los objetos y campos de la base de datos que falten.

```
(evalua2) ~ python manage.py runserver
```

Este comando ejecuta el servidor web de Django que escucha peticiones en el puerto 8000. Y con todo esto ya estará el backend funcionando.

Frontend

En la segunda terminal hay que ir hasta la raíz del proyecto clonado; dentro hay dos carpetas frontend y backend. Entrar en la de frontend y ejecutar

```
~ python server.py
```

Este comando ejecuta un servidor web ligero de Python que escucha en el puerto 3000. Será el encargado de servir todo el contenido estático del proyecto.

Para un entorno de desarrollo con este servidor hay más que suficiente y funciona genial. Pero para un entorno de producción, la parte del frontend debería estar detrás de un Apache o un Nginx y no necesitaríamos para nada este servidor web.

Después de esto el proyecto estará funcionando perfectamente en el servidor local y será accesible mediante un navegador web en la dirección `http://127.0.0.1:3000`

5.4 Conclusiones

Como el lector ya sabe, hay una gran cantidad de frameworks, lenguajes de programación y tecnologías que se podían haber utilizado para el desarrollo de este proyecto. Las tecnologías que se utilizan en este proyecto han sido elegidas, en el caso de Django, porque en mi día a día trabajo con él y, en el caso de Python es, como a mi me gusta decir, 'mi lenguaje materno'. Así mismo, la elección de AngularJS sobre otros frameworks como BackboneJS, Ember o ReactJS, ha sido por voluntad propia y por aprender una tecnología nueva. No había trabajado nunca con AngularJS pero, sin duda, lo volveré a utilizar. Me ha parecido muy cómodo y con una curva de aprendizaje muy rápida; además, hay mucha documentación en Internet. Después de todo se trata de que el desarrollador trabaje lo mas cómodo posible.

Capítulo 6

Conclusiones

6.1 Dificultades encontradas

A continuación, se describen las dificultades más importantes que se han encontrado a la hora de desarrollar este proyecto y las soluciones aportadas.

Como al inicio de cualquier proyecto profesional se tiene que hacer un estudio de mercado y dentro de este estudio se encuentra el analizar sistemas similares. Encontrar sistemas similares a este proyecto ha sido muy costoso y el único sistema que se ha encontrado y puede considerarse similar es IonCUDOS 2.4.

La mayoría de enlaces a páginas con softwares susceptibles de ser comparados eran antiguos o bien las páginas estaban inaccesibles.

Otro gran problema a la hora de comparar sistemas ha sido no poder probarlos. Ningún sistema nombrado en el capítulo 2 ha podido ser probado ya que, o no había una versión de prueba para descargar, o había que contactar y tratar con el departamento comercial para pedirles una versión de evaluación.

Así que, para poder llevar a cabo el capítulo ha habido que conformarse con lo poco que se ha encontrado y estaba accesible y, además, elaborar el análisis a partir de la documentación que las empresas proporcionan en su página web.

En cuanto al desarrollo, la principal dificultad ha sido aprender a utilizar AngularJS que, hasta el momento de desarrollar el proyecto, no había trabajado nunca con este framework Javascript. Una vez empezado a trabajar con él me ha sorprendido gratamente y pienso utilizarlo en próximos trabajos.

También la elección del framework web Python fue un problema ya que al principio empecé a trabajar con Flask pero a mitad de desarrollo vi que no era tan potente y era mucho más costoso avanzar que con Django. Por esta razón decidí cambiar y empezar de cero con Django.

6.2 Aportaciones

6.2.1 Tecnológicas

Las aportaciones al sistema son el diseño de la aplicación web, el algoritmo de análisis de resultados de aprendizaje y el diseño de la arquitectura completa. También son aportaciones el estudio de sistemas similares y la especificación que puede servir a próximos estudiantes que decidan seguir este proyecto.

6.2.2 Académicas

A consideración del autor, este sistema es la base de una buena herramienta que puede ayudar en un futuro a los profesores a la hora de plantear sus asignaturas, ya que podrán saber si sus resultados de aprendizaje y sus actos de evaluación cumplen las competencias que se han pensado para la asignatura.

De este mismo modo, los estudiantes sabrán que el día que se incorporen a la vida laboral, estarán cualificados para realizar el trabajo que necesitan de las competencias que cubrían las asignaturas cursadas en la universidad.

6.3 Ampliaciones

Considero que este proyecto es muy práctico y apasionante y, además, puede ser ampliado en muchos aspectos. Este proyecto puede ser la base de una buena herramienta de ayuda al personal docente.

Se podría crear un apartado para los gestores de títulos. Implementar una capa de seguridad ya que el proyecto actual no tiene en consideración usuarios, contraseñas, grupos, etc... Un profesor puede acceder a todas las asignaturas y a todos los resultados de aprendizaje y, sería conveniente, que solo pudiera ver las asignaturas que imparte.

En el algoritmo de análisis hay que tener en cuenta las conjugaciones verbales de la taxonomía de Bloom, ya que ahora solo se tiene en cuenta el verbo en infinitivo.

Bibliografía

[EE84] Institute of Electrical y Electronics Engineers. "IEEE Guide for Software Requirements Specifications". En: *IEEE Std 830-1984* (feb. de 1984), págs. 1-26. DOI: 10.1109/IEEESTD.1984.119205.

[Can08] Elena Cano. "La evaluacion por competencias en la educacion superior". En: *Profesorado: revista de curriculum y formacion del profesorado* 12.3 (2008), pág. 11.

[Anga] Material Angular. *Documentación Web Framework Material Angular*. URL: https://material.angularjs.org/latest/.

[Angb] AngularJS. *Documentación Web Framework AngularJS*. URL: https://angularjs.org/.

[Chaa] Angular Chart. *Librería AngularJS para crear gráficas*. URL: https://jtblin.github.io/angular-chart.js/.

[Chab] ChartJS. *Librería Javascript para crear gráficas*. URL: http://www.chartjs.org.

[Des] Material Design. *Especificaciones Google Materia Design*. URL: https://material.google.com/.

[Dja] Django. *Documentación Python Framework Django*. URL: https://www.djangoproject.com/.

[fra] Django REST framework. *Documentación Django REST framework*. URL: http://www.django-rest-framework.org/.

[Hea] Django Cors Headers. *Documentación django-cors-headers*. URL: https://github.com/ottoyiu/django-cors-headers.

[Pos] PostgreSQL. *Documentación bases de datos PostgreSQL*. URL: https://www.postgresql.org/.

[Pyc] Python IDE Pycharm. *Página web de Pycharm*. URL: https://www.jetbrains.com/pycharm/.

[wra] Virtual enviroment wrapper. *Documentación Virtual enviroment wrapper*. URL: http://virtualenvwrapper.readthedocs.io/en/latest/command_ref.html.