UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Visor 3D v2.0

Presentado por Jose Manuel Moral Garrido
en Universidad de Burgos — 7 de noviembre de 2017
Tutor: José Francisco Díez Pastor, Álvar Arnaiz González

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Álvar Arnaiz González y D. José Francisco Díez Pastor profesores del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Jose Manuel Moral Garrido, con DNI 71301434-P, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Visor 3D v2.0.

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

En Burgos, 7 de noviembre de 2017

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. Álvar Arnaiz González	D. José Francisco Díez Pastor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Visor 3D, *STL*, *PLY*, osteología, *e-learning*, docencia virtual

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

3D Viewer, *STL*, *PLY*, osteology, *e-learning*

Índice general

Índice general III

Índice de figuras V

Índice de tablas VI

Introducción 1

Objetivos del proyecto 3

Conceptos teóricos 5
- 3.1. API Rest . . . 5
- 3.2. Web Server Gateway Interface (WSGI) . . . 5
- 3.3. Listas de items . . . 6
- 3.4. Tablas . . . 6

Técnicas y herramientas 9
- 4.1. Moodle . . . 9
- 4.2. Sublime Text . . . 10
- 4.3. Tex Studio . . . 10
- 4.4. JSONMate . . . 10
- 4.5. PuTTY . . . 10
- 4.6. Servidor arquimedes . . . 11
- 4.7. Comparativa servidores para desplegar Flask . . . 11

Aspectos relevantes del desarrollo del proyecto 13
- 5.1. Tratamiento de los roles de los usuarios . . . 13
- 5.2. Obtención de los Modelos . . . 13

5.3. Instalación y configuración de Moodle como API Rest 14
5.4. Encriptado y desencriptado de los modelos 14

Trabajos relacionados **15**

Conclusiones y Líneas de trabajo futuras **17**

Bibliografía **19**

Índice de figuras

3.1. Url de ejemplo de llamada a la API Rest correspondiente 5

Índice de tablas

3.1. Herramientas y tecnologías utilizadas en cada parte del proyecto 7
4.2. Tabla comparativa servidores 12

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos

3.1. API Rest

Para comprender como funciona nuestra *API* primero debemos comprender en que consiste una *API Rest*. Una *API Rest* es una arquitectura de desarrollo web basada en el protocolo *HTTP* [6], el cual es un protocolo de acceso sin estado en el que cada mensaje intercambiado tiene la información necesaria para su comprensión sin necesidad de mantener el estado de las comunicaciones.

Se compone de un conjunto de operaciones *CRUD* (create, read, update, delete) con sus métodos *POST, GET, PUT, DELETE, PATCH* correspondientemente, de los cuales se obtendrá información en un determinado formato (en nuestro caso *JSON*)

Una manera rápida de comprender como se realizan las peticiones a la *API* de UBUVirtual es comprobar la llamada con este formato: 3.1

```
"https://your.site.com/moodle/webservice/rest/server.php?wstoken=...&wsfunction=...&moodlewsrestformat=json"
```

Figura 3.1: Url de ejemplo de llamada a la API Rest correspondiente

[2]

3.2. Web Server Gateway Interface (WSGI)

Se trata de una especificación para una interfaz simple y universal entre servidores web y aplicaciones (o *frameworks*) para aplicaciones programadas en Python. De esta manera se tienen dos partes:

- La parte del servidor
- La parte de la aplicación

Para procesar la petición *WSGI* [9], la parte del servidor recibe una petición del cliente y la pasa al *middleware*. Después de ser procesada, esta petición pasa a la parte de la aplicación. La respuesta proporcionada por la parte de la aplicación es transmitida al *middleware*, seguidamente a la parte del servidor y consecutivamente al cliente.

Esta capara de *middleware* puede tener las siguientes funcionalidades:

- Relanzar la petición a múltiples objetos de tipo aplicación basados en *URL*.
- Permitiendo a múltiples aplicaciones ejecutarse de manera concurrente
- Mantener un equilibrio de carga

3.3. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.4. Tablas

Igualmente se pueden usar los comandos específicos de LaTeXo bien usar alguno de los comandos de la plantilla.

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

Técnicas y herramientas

4.1. Moodle

Aunque nuestra aplicación esté orientada a ser ejecutada con fin de poder dar una docencia online desde la *API* de UBUVirtual, cabe mencionar *Moodle* como herramienta ya que ha sido instalada localmente con el fin de poder realizar pruebas sin tener que depender un super usuario que nos proporcionase las asignaturas, cursos, etc, que necesitamos en cada caso.

Se trata de una herramienta de uso educativo virtual, teniendo las capacidades de gestionar cursos, asignaturas y demás funcionalidades que proporciona ayuda a los profesores a crear comunidades de aprendizaje en línea. Se trata de una herramienta escalable, personalizable, económica y segura a la par que flexible. Dentro del gran número de funcionalidades podemos destacar las siguientes:

- Facilidad de uso
- Gestión de perfiles de usuario
- Facilidad de acceso
- Administración sencilla
- Realización de exámenes en línea
- Gestión de tareas
- Implementación de aulas virtuales

Moodle ofrece ciertas características de administración, dentro de las cuales entran los roles de usuario, que tienen un papel importante en nuestra *API*(sección 5.1). Los privilegios de cada uno de los roles son diferentes y cada uno tiene una funcionalidad diferente y es por esto que necesitábamos una instalación local, para poder probar cada uno de los roles. [4]

4.2. Sublime Text

Para la edición del los diferentes *scripts* utilizaremos este editor de texto ya que, además de ser gratuito (aunque un tanto cargante con solicitar la compra de la versión de pago), es intuitivo y nos proporciona una interfaz cómoda para trabajar,además de una función de auto completar altamente útil

4.3. Tex Studio

TeXstudio es un editor de *LaTeX* de código abierto y multiplataforma con una interfaz similar a *Texmaker*. Esta herramienta es un IDE de *LaTeX* que proporciona soporte de escritura incluyendo la corrección ortográfica, plegado de código y resaltado de sintaxis [8]

4.4. JSONMate

Hemos utilizado esta herramienta, la cual en realidad es una página web que nos ayuda a interpretar la información obtenida en formato *JSON* y simplificarnos su vista [1].

4.5. PuTTY

PuTTY es un cliente *SSH, Telnet, rlogin, y TCP raw* con licencia libre disponible para *Windows* y en varias plataformas de *Unix* (Versión *Mac OS*). Hemos utilizado esta herramienta para acceder al servidor arquimedes (sección 4.6) desde nuestra máquina con *Windows*, ya que desde *Linux* realizamos las llamadas mediante el comando ***ssh*** [7]

4.6. Servidor arquimedes

Se trata de un servidor proporcionado por la Universidad de Burgos para poder desplegar nuestra aplicación. Este servidor contiene una máquina con un sistema operativo *Ubuntu* 16.04.

4.7. Comparativa servidores para desplegar Flask

Dado que la Universidad de Burgos nos ha dotado con un servidor (4.6) tenemos que elegir la manera de desplegar nuestra *API Flask*, con lo cual hemos realizado una comparativa con diversas herramientas para desplegarla con el fin de encontrar la mejor manera de hacerlo.

A continuación, mostraremos una tabla con las diferentes herramientas seleccionadas con el fin de elegir la que mejor se amolde a nuestro caso:

Tabla 4.2: Tabla comparativa servidores

	Apache	**uWSGI**	**Stand-Alone (Gunicorn)**	**Stand-Alone (Twisted Web)**
¿Por qué utilizarlo?	En el caso de tener experiencia utilizando Apache, así como que se tenga una dependencia del mismo, esto significará **estabilidad** en el entorno de producción de la aplicación, teniendo gran variedad de módulos estable sy completos. A su vez, es un software muy probado y fiable, teneindo gran variedad de información en la web.	Soporta aplicaciones Python por completo corriendo en WSGI, pudiendo sus componentes realizar muchas más funciones que correr la aplicación, con la correspondiente bajón en el uso de la memoria. Como **desventaja** hemos considerado que, como está actualmente en desarrollo, podría conllevar a un fallo que aún no se halla contemplado, teniendo a su vez una convención de nombres confusa.	**Gunicorn**: Si se desea extender de *Apache* utilizando Python(siempre y cuando sea necesario) y programarlo para alguna tarea en concreto. Además, tiene la ventaja de ser sencillo de ejecutar si no se necesita extender de *Apache*	**Twisted Web**: Si se desea extender de *Apache* utilizando Python(siempre y cuando sea necesario) siendo simple, estable y maduro. A su vez, puede soportar clientes virtuales.

Aspectos relevantes del desarrollo del proyecto

5.1. Tratamiento de los roles de los usuarios

Durante el progreso en la aplicación de partida, nos dimos cuenta de que el tratamiento de los roles podía también ser realizado mediante la cotejación del mismo contra la *API* de UBUVirtual en lugar de tener los roles almacenados en la base de datos junto con los usuarios autorizados. Por otro lado, cabe mencionar que en la aplicación de partida no se estableció ningún tratamiento de roles de usuario, aunque se mencionara.

Inicialmente se decidió llevar a cabo el objetivo inicial de tratamiento de roles, es decir, mediante la definición de los mismos en la base de datos con su posterior consulta a la hora de definir el usuario, aunque después nos dimos cuenta de que la *API* de UBUVirtual podría proporcionarnos estos roles, los cuales vienen dados a cada usuario en la asignatura correspondiente.

5.2. Obtención de los Modelos

La idea inicial de la aplicación era que los modelos proporcionados para su posterior visualización se administraran de manera local, es decir, en una carpeta con todos los modelos. Llegados al punto de pensar cómo podíamos proporcionar al usuario los modelos óseos privados, decidimos que la mejor manera de hacerlo era mediante la administración de dichos modelos como recursos en la *API* de UBUVirtual, siendo estos recursos invisibles para el alumno y a los que solo el profesor tenga acceso para modificar.

5.3. Instalación y configuración de Moodle como API Rest

Para poder realizar las pruebas pertinentes en cada parte del proyecto, hemos decidido que lo ideal es tener instalado *Moodle* de manera local para realizar las llamadas, así como la subida de recursos, asignación de roles, etc, sin tener que depender de un tutor (el cual puede crear una asignatura ficticia y realizar las pruebas ahí). Por ello, hemos realizado la instalación de *Moodle* con un paquete instalador (sección 4.1) para *Windows* en el cual viene incluido *XAMPP* [10] así como *MySql* [5].

Una vez instalado *Moodle* y creado un curso y un alumno para poder realizar las pruebas, nos hemos encontrado con el problema de que la *API* no era una *API Rest* 3.1 ya que a la hora de realizar las peticiones necesarias para la obtención de información del usuario se nos denegaba el acceso. Para poder solucionar este problema hemos tenido que configurar nuestra *API* cambiando los parámetros correspondientes (véase *Manual del Programador*).

5.4. Encriptado y desencriptado de los modelos

Con la finalidad de obtener seguridad en nuestra *API* en lo relacionado a los modelos decidimos realizar una operación de encriptado y desencriptado de los mismos para que alguien ajeno a la *API* no sea capaz de obtener los modelos o modificarlos.

Para realizar el encriptado realizamos la lectura del modelo y modificamos ciertos valores con el fin de que a la hora de cargar dicho modelo, el cargador de modelos sea capaz de desencriptar el modelo en cuestión y así visualizarlo. Sin embargo nos hemos encontrado con ciertos problemas a la hora de encriptar y desencriptar los modelos (véase *Manual del Programador*).

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Json mate.

[2] Fersacom. Como configurar api rest-full en moodle.

[3] Moodle. Web service api functions — moodle, Unknown.

[4] Wikipedia. Moodle – wikipedia, la enciclopedia libre.

[5] Wikipedia. Mysql – wikipedia, la enciclopedia libre.

[6] Wikipedia. Protocolo de transferencia de hipertexto – wikipedia, la enciclopedia libre.

[7] Wikipedia. Putty – wikipedia, la enciclopedia libre.

[8] Wikipedia. Texstudio – wikipedia, la enciclopedia libre.

[9] Wikipedia. Web server gateway interface – wikipedia, la enciclopedia libre.

[10] Wikipedia. Xampp – wikipedia, la enciclopedia libre.