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 — 22 de enero de 2018
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, 22 de enero de 2018

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

Resumen

Teniendo en cuenta el avance acaecido en la tecnología, así como la necesidad de poder cursar cualquier titulación de manera remota, ya sea por comodidad o por falta de medios, se produce un aumento considerable de la necesidad de docencia online en dichas titulaciones o estudios. Es por ello que la Universidad de Burgos continúa reforzando sus medios añadiendo continuamente nuevas titulaciones con posibilidad de ser impartidas de manera no presencial.

Aunque resulte todo un reto adaptar ciertas asignaturas a docencia virtual, ya existen varios proyecto que intentan facilitar dicho fin a base de la creación de herramientas informáticas que permitan una simulación de la parte de dichas asignaturas que no puede ser presentada de manera virtual. En nuestro caso, este proyecto surge de la necesidad de posibilitar el trabajo con huesos de la asignatura de osteología humana del Grado en Historia y Patrimonio impartida por María Rebeca García, que supone la figura de cliente en nuestro proyecto. Como predecesor encontramos el proyecto de Alberto Vivar Arribas («3D Viewer» [**?**]) en el que se consiguió el objetivo principal y del cual nosotros continuaremos mejorando y ampliando.

Nuestro trabajo consistirá en mejorar la sensación del cliente con la aplicación ampliando funcionalidades ya existentes y creando nuevas herramientas que faciliten el trabajo a los docentes. A u vez, realizaremos el despliegue de nuestra aplicación en un servidor web de manera que ésta sea accesible desde cualquier punto de la Universidad de Burgos y, por tanto, pueda utilizarse impartiendo docencia.

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 de figuras

Índice de tablas

Introducción

Siendo una de la grandes dificultades la de impartir de manera *online* el material docente impartido en clases de prácticas, partiremos de un proyecto que busca enriquecer las herramientas disponibles para dicho fin en el Grado en Historia y Patrimonio.

Nuestro objetivo principal será seguir la estela de dicho proyecto y aumentar las funcionalidades de esta herramienta, así como pulir las partes actualmente funcionales. Un punto importante será el poder obtener acceso desde cualquier ordenador a nuestra herramienta. Para ello, trataremos de ejecutar la aplicación en un servidor accesible para todos los alumnos de la Universidad de Burgos que se encuentren cursando el Grado en Historia y Patrimonio. A su vez, como objetivo clave se tiene el proporcionar seguridad en los modelos 3D de los huesos debido a su unicidad y privacidad, por lo que se adoptarán las medidas necesarias para dicho fin.

Por otro lado, cabe mencionar la importancia de los docentes en nuestra aplicación, por lo que se les tratará de facilitar el uso de la misma enfocándonos en el ámbito de la corrección de ejercicios y comprobación de copias entre alumnos.

Objetivos del proyecto

Objetivo de Alberto. Dentro de nuestros objetivos estará la gestión de roles de usuario otorgando funcionalidades diferentes dependiendo el rol de dicho usuario dentro de la asignatura. También incluiremos en la aplicación una página en la que el profesor de la asignatura pueda corregir los ejercicios docentes de manera sencilla y clara visualmente. Para ello, se utilizará el visor de modelos con el fin comparar las soluciones de los ejercicios (resueltas por el profesor) con las soluciones propuestas por los alumnos a dichos ejercicios.

A su vez, nos centraremos principalmente en proporcionar seguridad a los modelos que estarán albergados en el servidor debido a su unicidad y dificultad de encontrar. Esto es muy importante ya que una vez que estos modelos estén subidos a nuestro servidor, podrán ser accesibles a ciertas amenazas como podría ser su eliminación.

2.1. Gestionar roles de usuario

Previo a explicar el resto de los objetivos deberemos establecer qué funcionalidades tiene cada usuario dependiendo de su rol, aunque en nuestra aplicación definiremos dos clases de usuarios (siendo los posibles usuarios los proporcionados por *Moodle*):

- **Profesor**: Este será el *superusuario* de la aplicación teniendo acceso a todas las funcionalidades posibles que se integren en el proyecto. Podrá acceder a visualizar modelos, realización de ejercicios, subida y eliminado de los modelos al servidor en el que estará aojada la aplicación, etc.

- **Cualquier otro usuario**: Este apartado engloba los usuarios de tipo Estudiante, Invitado, Creador del curso, Mánager y Profesor sin permiso de edición, es decir, cualquier usuario que no sea Profesor está en este grupo. Dicho grupo simplemente tiene la opción de visualizar los modelos que se encuentren en la aplicación, así como hacer uso de las herramientas incluidas en el mismo (anotaciones, medidas, importación y exportación de puntos, etc).

2.2. Añadir restricciones de escritura para usuarios

Otro aspecto importante a tener en cuenta dentro de la aplicación es que el usuario puede almacenar nombres de anotaciones, medidas, ejercicios, etc, de manera autónoma sin ningún tipo de control. Esto puede suponer un problema ya que si el profesor guarda el nombre de un ejercicio como `.ejercicio_numero_1` podría interpretarse por parte del sistema operativo como que dicho nombre en realidad es una extensión. Dicho problema supondría la pérdida del fichero que alberga el ejercicio y por consiguiente la pérdida del mismo. A su vez, si un alumno o profesor exporta los nombres de anotaciones y medidas incluyendo algún carácter especial como puede ser un «@» nos causaría otro problema importante relacionado con la corrección de ejercicios por parte del profesor. Esto es debido a que las anotaciones de alumnos a la hora de ser exportadas se guardan con ciertos caracteres especiales (los cuales son los restringidos) para que al importarlos, el visor sea capaz de diferenciar entre importaciones de alumnos y profesores.

2.3. Posibilitar plataforma de ejercicios para el profesor

Con el fin de que el profesor pueda realizar pruebas a sus alumnos en la docencia online del grado, se quiere incluir en la aplicación un apartado que permita al profesor realizar y albergar ejercicios propuestos por el mismo. De esta manera, este profesor podrá presentar ejercicios a los alumnos y, tras recibir una solución a dichos ejercicios, poder cotejarlas con los ejercicios previamente resueltos en la aplicación. Dentro de la herramienta de ejercicios se podrá apreciar, en el visor de ejercicios, la cercanía de los puntos elegidos como referencia de los alumnos a los puntos de referencia del profesor así como nombres técnicos de ciertas partes del modelos examinado.

2.4. Despliegue de la aplicación en un servidor

Encontramos también el despliegue de nuestra aplicación en un servidor como objetivo del proyecto ya que queremos que nuestra aplicación pueda servir como herramienta docente para los alumnos de la asignatura de osteología humana del Grado en Historia y Patrimonio. Para ello deberemos configurar un servidor el cual nos será proporcionado por la Universidad de Burgos con el fin de albergar tanto la aplicación como los modelos en sí mismos.

Cabe mencionar la gran complejidad que supone desplegar una aplicación en un servidor real teniendo que realizar previamente un configuración del mismo. Esto es debido a que se debe realizar una configuración partiendo de cero de una máquina en concreto con el fin de que dicha máquina sea capaz de ejecutar nuestra aplicación en un dominio específico.

2.5. Dar seguridad a los modelos

Debido a que los modelos utilizados en el grado son únicos, caros y muy difíciles de encontrar, deberemos dotar a nuestra aplicación de la seguridad necesaria para poder almacenar estos modelos en nuestro servidor. Centraremos nuestro objetivo en dar una encriptación a los modelos que serán subidos al servidor con su posterior desencriptación a la hora de visualizarlos, de manera que estos no puedan ser obtenidos fácilmente desde un navegador web. De esta manera, conseguimos que los modelos queden inservibles en caso de que se intente obtener información de ellos.

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* [4], 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[1].

Se compone de un conjunto de operaciones *CRUD* (create, read, update, delete) con sus métodos *POST, GET, PUT, DELETE, PATCH* correspondientes, 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 el formato que se ve en la figura 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 [**?**].

[1] https://books.google.es/books?hl=es&lr=&id=eABpzyTcJNIC&oi=fnd&pg=PR3&dq=API+REST&ots=vzPw65ecGD&sig=BMINNHBRFQAzWYcRsrG-r2CXmYw#v=onepage&q=API%20REST&f=false

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 podremos distinguir entre dos grandes bloques:

- El lado del servidor
- El lado de la aplicación

Para procesar la petición *WSGI* [8], el lado 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 del lado de la aplicación es transmitida al *middleware*, que posteriormente reenvía dicha respuestaal lado del servidor y finalmente es reenviada al cliente.

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

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

3.3. Mod_wsgi

Técnicas y herramientas

4.1. Moodle

Previo a centrarnos en *Moodle* en particular, tendremos que saber que se trata de un sistema de gestión de aprendizaje (*Learning Management System*). Estos sistemas se utilizan para administrar y gestionar las actividades de formación no presencial, permitiendo un trabajo asíncrono entre los participantes. Dentro de sus competencias encontramos la gestión de usuarios, gestión de recursos o material docente, actividades de formación, seguimiento en el proceso de aprendizaje de los participantes, realizar evaluaciones, foros de consulta, etc. [6]. Una vez conocido esto, indaguemos en por qué *Moodle* forma parte de este apartado.

Aunque nuestra aplicación esté orientada a impartir una docencia online desde la *API* de UBUVirtual, cabe mencionar *Moodle* como una herramienta. Esto es debido a que ha sido instalada de manera local con el fin de no tener que depender un super usuario que nos proporcione los recursos necesarios para las pruebas pertienentes (asignaturas, cursos, etc.) que necesitamos en cada caso.

Moodle es 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 [2].

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. TEXStudio

Como se menciona en *Wikipedia*: "TEX studio es un editor de LATEX de código abierto y multiplataforma con una interfaz similar a TEX maker". Esta herramienta es un IDE de LATEX que proporciona soporte de escritura incluyendo la corrección ortográfica, plegado de código y resaltado de sintaxis [7].

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* 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` [5].

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[2].

4.7. Comparativa servidores para desplegar Flask

Dado que la Universidad de Burgos nos ha dotado con un servidor (ver sección 4.6) tenemos que elegir la manera de desplegar nuestra *API Flask*, para 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:

[2] https://arquimedes.ubu.es/

Tabla 4.1: Tabla comparativa servidores. ¿Por qué utilizar cada servidor?

Apache	**uWSGI**	**Stand-Alone (Gunicorn)**	**Stand-Alone (Twisted Web)**
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.

Inicialmente escogimos *Gunicorn* como método de despliegue de nuestra aplicación debido a su sencillez y compatibilidad con *Apache* (compatible con extensiones y utilidades). A medida que avanzamos, nos dimos cuenta de que *Gunicorn* no cumplía nuestras expectativas, así como una falta de información de configuración en nuestro caso. Por ello, finalmente nos decantamos por *Apache* (mod_wsgi) para el despliegue de nuestra aplicación.

4.8. «Librería» Pyntcloud

La palabra librería se encuentra entrecomillada ya que no es una biblioteca en sí misma, sino una serie de recursos que en nuestro caso han sido utilizados para la lectura de os archivos *PLY*, tanto formato *ASCII* como *Binario* (ver sección **3.2 Formato PLY** [**?**][3])[4].

De esta manera, los modelos se leerán tanto en formato *ASCII* como en *Binario* almacenando los vértices y las caras por separado en dos *DataFrames* diferentes. A su vez, diferencia entre los modelos con color a los modelos en escala de grises utilizando estructuras como las que se aprecia en las figuras 4.2 y 4.3:

	x	y	z	red	green	blue
0	-34.811508	-117.870407	63.747543	61	43	55
1	-35.032043	-117.150902	64.284180	43	30	37
2	-34.425644	-117.567711	64.177994	64	48	59
3	-34.103275	-118.130844	63.940544	55	43	55
4	-34.569893	-116.868202	64.410629	58	39	43

Figura 4.2: DataFrame que contiene los vértices y colores.

[3]Consultar apartado Conceptos Teóricos del proyecto predecesor al nuestro.
[4]https://github.com/daavoo/pyntcloud/blob/master/pyntcloud/io/ply.py

	x	y	z
0	-329.420929	600.480591	122.643959
1	-332.396393	603.327759	118.564522
2	-335.312347	608.740234	114.125450
3	-340.071198	611.782471	117.577751
4	-341.737579	613.653564	124.656273

Figura 4.3: DataFrame que contiene únicamente los vértices.

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 mencionó.

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. Pero posteriormente, 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 3D privados, decidimos que la mejor manera de hacerlo era mediante la administración de dichos modelos como recursos en la *API* de UBUVirtual. De esta manera, estos recursos serían invisibles para el alumno y a los que solo el profesor tuviera acceso para modificar.

Pero finalmente, decidimos que los modelos se albergaría en el servidor «Arquímedes» (ver sección 4.6) de manera que solo se pudiera acceder a ellos a través de la aplicación.

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* [9] así como *MySql* [3].

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* (ver sección 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 Anexo del *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. Esto se debe a la unicidad y privacidad de estos modelos 3D ya que son significativamente caros de conseguir y no deberían ser públicos para usuarios ajenos.

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 Anexo del *Manual del Programador*).

5.5. Configuración del servidor «Arquímedes»

Debido a que conseguimos un servidor con el que desplegar nuestra aplicación para poder darla una mayor funcionalidad, hemos tenido que configurar este servidor acorde a lo mencionado en la sección 4.7. Dicha configuración queda reflejada en el anexo del *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] Wikipedia. Moodle – wikipedia, la enciclopedia libre.

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

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

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

[6] Wikipedia. Sistema de gestión de aprendizaje – wikipedia, la enciclopedia libre.

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

[8] Wikipedia. Web server gateway interface – wikipedia, la enciclopedia libre. https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface.

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