Sistema de gestión de objetos virtuales creativos pregrado de creación digital de la Universidad de Antioquia

Autor

Juan Pablo Romero Laverde

Universidad de Antioquia

Facultad de ingeniería,

Departamento de ingeniería de sistemas.

Medellín, Colombia

2021

Sistema de gestión de objetos virtuales creativos pregrado de creación digital de la Universidad de Antioquia

Juan Pablo Romero Laverde

Informe de práctica presentado como requisito parcial para optar al título de:

Ingeniero de Sistemas

Asesores:

Juan Diego Vélez Vargas

Ingeniero de sistemas

Carlos Mauricio Duque Restrepo

Ingeniero de sistemas

Universidad de Antioquia

Facultad de ingeniería,

Departamento de ingeniería de sistemas.

Medellín, Colombia

2021.

Tabla de contenido

Lista de figuras

Lista de tablas

Resumen

El marco del proyecto de desarrollo creación digital se encuentra acompañado por Ude@ educación virtual, en el cual surgió la necesidad de crear un sistema que permitiera la manipulación de la información de los objetos virtuales creativos (OVC) elaborados por los estudiantes en los diferentes cursos que se plantea el pregrado.

Para desarrollar este proyecto se realizó una distribución por módulos, uno de estos es para el backend que contempla la creación, edición, modificación y lectura de la información que será almacenada en el sistema, para el frontend se realizó un módulo para la vista del login que además fue conectado al sistema Mares de la universidad de Antioquia, otro modulo tiene las vistas relacionadas a los objetos virtuales creativos, donde el estudiante podrá interactuar con diversos formulario y tablas para realizar las operaciones elaboradas en el backend y por ultimo un módulo de rizoma donde los estudiantes podrán visualizar los objetos virtuales creativos mediante un grafo donde conseguirán consultar la información general de cada uno.

Finalmente se realizaron pruebas de carga y peticiones a la aplicación para comprobar cuales son los tiempos estimados, la desviación y los porcentajes de error, estos datos nos dirían que tanta capacidad de estudiantes soporta el sistema en varios de sus aspectos, puesto que, el pregrado tiene una estimación de 500 usuarios para los próximos 5 años, y según los resultados obtenidos en los test se garantiza la capacidad esperada con rendimientos por encima de la desviación, lo que indica que son óptimos y pueden soportar la carga máxima esperada.

Introducción

Actualmente uno de los proyectos que se vienen asesorando y desarrollando en Ude@ Educación Virtual es el pregrado de creación digital, en el cual se están desplegando una gran variedad de posibilidades tecnológicas con la intención de crear el primer pregrado totalmente virtual en la universidad de Antioquia.

Por esta razón, dentro del marco proyecto pregrado creación digital surge la necesidad de proporcionar un sistema que permita manipular la información de objetos virtuales creativos (ovc's), los cuales son desarrollados por los estudiantes como parte de la elaboración de recursos individuales y colectivos en los distintos cursos del pregrado. También, el sistema debe permitir a los estudiantes realizar diferentes operaciones como lo son: la creación, modificación, eliminación y visualización de objetos virtuales creativos, asimismo, el sistema debe permitir almacenar todos los recursos elaborados por cada estudiante.

Para la elaboración del sistema se definen cuatro módulos, uno para el backend y tres para el frontend con los cuales se pudiese cumplir con todos los requisitos presentados por los involucrados en el proyecto.

Para el backend se contempló el módulo OVC el cual contiene toda la lógica de las operaciones CRUD de la información respectiva para los objetos virtuales creativos.

Para el frontend se presentaron los módulos login, OVC y rizoma, los cuales son una serie de vistas con las que el usuario podrá interactuar, allí iniciara sesión con sus credenciales del portal Mares de la universidad de Antioquia y conseguirá enlazarse a las operaciones CRUD definidas en el backend mediante la interacción con una serie de formularios y tablas para el manejo de la información de los objetos virtuales creativos,

además los usuarios podrán ver estos en forma de grafo donde cada uno representa un nodo y se podrá consultar la información general de cada uno de esos nodos.

Por último, es importante recalcar el manejo del tiempo para el desarrollo del proyecto, para esto fue fundamental el uso de la metodología scrum, ya que esta permite tener una comunicación cercana con el cliente y así evaluar oportunamente cada una de las actividades y funcionalidades que se van desarrollando.

Objetivos

Objetivo general

Desarrollar un sistema que permita la gestión de objetos virtuales creativos elaborados en el pregrado de creación digital.

Objetivos específicos

- Realizar el levantamiento de los requisitos necesarios para el funcionamiento del sistema de gestión de objetos virtuales creativos elaborados en el pregrado de creación digital.
- Diseñar el o los modelos de datos necesarios para la información del sistema.
- Identificar el motor de base de datos más acertado para el manejo de la información recolectada en el sistema.
- Elaborar un backend que permita realizar operaciones CRUD en el sistema para el manejo de la información.
- Elaborar un frontend que permita al usuario la operatividad de la información y la visualización de todos los objetos virtuales creativos interconectados como nodos en un grafo desplegado en pantalla.
- Desplegar el sistema en un entorno web, adaptable a computadores, Tablet y celulares, para facilitar el acceso de los usuarios.
- Almacenar recursos como imágenes, videos, comprimidos, entre otros; en un servidor dispuesto por la universidad de Antioquia.
- El sistema debe operar de manera adecuada en condiciones de alta concurrencia debido al número de estudiantes que puede tener el pregrado en futuros semestres (500 en los próximos 5 años).

Marco Teórico

En el desarrollo de la aplicación de gestión de objetos virtuales creativos se utilizó el modelo de aplicación cliente-servidor, el cual distribuye tareas entre proveedores de recursos y servicios a los cuales se le llama servidores. Las aplicaciones clientes deben realizar peticiones a una o varios servidores que deben encontrarse en ejecución para atender a las demandas realizadas. Una de las disposiciones más frecuentes son los sistemas de múltiples capas es la descomposición del servidor en diversos programas que son ejecutados en diferentes computadores, ejemplo de ello en una aplicación web típica de tres capas que puede estar compuesta por una capa de presentación que es la funcionalidad relacionada con la interfaz de usuario, una capa de negocios que representa el procesamiento de reglas de negocio y una capa de datos que contienes la funcionalidad relacionada con el acceso a datos, cada una de estas capas puede estar dispuesta en tres computadores o servidores diferentes y pueden ser llamados entre sí sin la necesidad de tener las tres capaz alojadas en el mismo sitio. Una de las principales ventajas que cobijan este modelo es el mantenimiento, debido a que los cambios de la lógica del negocio no afecta directamente la lógica de presentación y gracias a esto no es necesario recompilar ni redistribuir código cliente, también el modelo permite un mayor rendimiento que otros, gracias a que logra un procesamiento distribuido; por otro lado cuenta con una escalabilidad que se logra debido a la separación de capas que simplifica el mantenimiento y procesamiento de la información, y por último se encuentra la seguridad donde la autentificación de la capa de datos se hace a nivel de la aplicación lo cual no compromete el sistema operativo o los diferentes servicios creados en el servidor.

Por otra parte, el manejo del servidor (Backend) utilizó una arquitectura REST, esto consta de una serie de restricciones arquitectónicas que se aplican a los componentes para

crear un estilo organizacional del software, estas restricciones son resumidas por LANTHALER, Markus; GÜTL, Christian citadas por Puerta(2015) como “la interacción sin estado, interfaz uniforme, la identificación de los recursos, manipulación de los recursos a través de representaciones, los mensajes autodescriptivos, e hipermedia como el motor del estado de la aplicación”, este estilo arquitectónico se caracteriza por ser un sistema el cual no tiene estado, puesto que cada petición realizada a un servicio tiene que ser independiente a las demás.

De manera que los sistemas basados en la arquitectura REST son escalables, pues, uno de los fundamentos que tiene es la construcción del sistema basado en capas que son quienes dan la escalabilidad de las aplicaciones, además la forma de desacoplar los recursos que se manejan con la intención de conceder acceso a una gran variedad de formatos como lo son JSON, XML, entre otros.

Igualmente, es importante tener presente que para el manejo del formato de salida de datos es basado en Json (JavaScript Object Notation) o Notación de Objetos de JavaScript, el cual es un formato de intercambio de datos, fácil de leer y escribir. Hoy en día se utiliza para hacer intercambio de datos en aplicaciones y servicios web en aplicaciones cliente-servidor. Este formato de texto tiene convenciones conocidas para lenguajes como C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros, siendo este ideal para el intercambio de datos entre diferentes lenguajes.

Del mismo modo, el almacenamiento de la información se hizo mediante mongodDB, que es una un base de datos NoSql, esta funciona como mecanismo de provisión y recuperación de datos que implementan diferentes estructuras de almacenamiento, las más extendidas son clave-Valor, columnas, documentos y grafos. Entre las características más comunes entre las bases de datos NoSql resalta la flexibilidad

en el esquema de datos que maneja, puesto que son esquemas dinámicos, es decir, que la escritura de los datos se puede y debe adaptar a una estructura específica, por otro lado, la tolerancia a fallos y redundancia ya que la especificidad de la estructura de la base de datos dependerá del problema a resolver, de igual modo se podrán agrupar diferentes modelos de datos sin perder información. Y para lograr esto Araujo (2016) describe las cuatro categorías en las que las bases de datos NoSql cumplen con las característica mencionada:

Base de datos de Documentos: Este tipo de base de datos almacena la información como un documento, usando para habitualmente para ello una estructura simple como JSON, BSON o XML y donde se utiliza una clave única para cada registro. Este tipo de implementación permite realizar búsquedas por clave–valor, realizar consultas más avanzadas sobre el contenido del documento. Son las bases de datos NoSQL más versátiles.

Almacenamiento Clave-Valor: Son el modelo de base de datos NoSQL más popular, además de ser la más sencilla en cuanto a funcionalidad. En este tipo de sistema, cada elemento está identificado por una clave única, lo que permite la recuperación de la información de forma muy rápida, información que suele almacenarse como un objeto binario. Se caracterizan por ser muy eficientes tanto para las lecturas como para las escrituras.

Bases de datos de grafos: Usadas para aquellos datos cuyas relaciones se pueden representar adecuadamente mediante un grafo. Los datos se almacenan en estructuras grafo con nodos (entidades), propiedades (información entre entidades) y líneas (conexiones entre las entidades).

Base de datos Columnar (o Columna ancha): En vez de “tablas”, en las bases de datos de columna tenemos familias de columnas que, son los contenedores de las filas. A

diferencia de los RDBMS, no necesita conocer de antemano todas las columnas, cada fila no tiene por qué tener el mismo número de columnas. Este tipo de bases de datos se adecuan mejor a operaciones analíticas sobre grandes conjuntos de datos. (p. 1)

De igual forma, es importante tener en cuenta que para el almacenamiento de los archivos se utilizó un espacio en el servidor mediante la librería multer de JavaScript. Esta implementación se esperaba realizar con Google drive api, pero debido al costo de almacenamiento y el cambio de políticas de Google se optó por utilizar un almacenamiento físico propio de la universidad de Antioquia. Por el lado de la seguridad cabe destacar que Google drive limitaba la posibilidad de controlar el rol que tienen los usuario sobre los recursos almacenados, por ejemplo un usuario x podría modificar o incluso borrar el recurso de un usuario w, mientras que con la nueva implementación tenemos total control de quien puede modificar o eliminar un recurso que este almacenado en el servidor, así mismo el lugar donde se guardan los recursos es desconocida para el usuario debido a que esto se hace a nivel de la aplicación, mientras que una de las condiciones específica de Google drive api era compartir la carpeta raíz donde se encuentran las carpetas de trabajo de todos los usuarios.

Luego, se desarrolló un frontend basado en SPA (Single page application). Las SPA son aplicaciones o páginas web las cuales renderizan todo el contenido en una sola página, es decir, se cargan un solo archivo HTML y todo su contenido se despliega dentro de ese único archivo, generando una experiencia rápida y fluida. Las SPA ofrecen una experiencia de usuario similar a la de una aplicación de escritorio, pero desde un navegador, manejando datos, flujos y estructuras que dependen de una conectividad a Internet en menor medida que el resto de las aplicaciones web tradicionales, generalmente están construidas con

JavaScript, para lo cual se pueden utilizar frameworks frontend como Angular y VueJS o librerías como ReactJS.

Por otra parte, unos de los conceptos importante que se tuvieron presentes en este proyecto es la integración continua que es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. Esta práctica es una solución para los problemas que puede generar la integración del código nuevo a los equipos de desarrollo y de operaciones.

Es importante mencionar que los despliegues de la aplicación se realizaron de la siguiente manera. Se cuenta con un servidor de pruebas llamado test app, en el cual se despliegan las diferentes versiones que se muestran del proceso que se va realizando. Una vez la aplicación sea aprobada por diferentes entes universitarios, se procederá a realizar el montaje en el servidor central de la universidad de Antioquia y que está dispuesto para las aplicaciones de ude@ educación virtual.

Por último, pero no menos importante, se realizaron pruebas de carga, que permite identificar la cantidad de peticiones que el sistema puede soportar. Son importantes cuando los sistemas deben tolerar un gran volumen de usuarios o transacciones concurrentes. Por ejemplo, un sistema de transporte tiene una cantidad establecida de usuarios que puede movilizar. De manera similar se puede probar un programa para identificar si se cumple con la cantidad de peticiones que se establece desde el inicio del proyecto, asimismo, para realizar estas pruebas se utilizó la herramienta Jmeter la cual está diseñada para medir el rendimiento, carga y comportamiento funcional de un sistema web.

Metodología

La elaboración del proyecto se realizó bajo la metodología SCRUM, la cual es un proceso de gestión donde se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Esta metodología busca mejorar los resultados obtenidos en el desarrollo, aportando en aspectos como la flexibilidad en la adopción de cambios y nuevos requisitos durante el proyecto e igualmente la posibilidad de colaboración e interacción entre el equipo y el cliente.

Así mismo, la metodología SCRUM se compone de releases que son una serie de entregas frecuentes con el objetivo de mostrar al cliente diferentes avances que se van teniendo en la aplicación y para ello se pone en producción nuevas versiones del producto. Otro componente importante son los sprints que son periodos breves de tiempo donde el equipo trabaja en tareas específicas.

Este proyecto pactó la entrega de un producto mínimo viable con todos los requisitos solicitados por el cliente y que pudiese ser escalable en el tiempo. Para el desarrollo con la metodología SCRUM se realizar una distribución en sprint de uno cada dos semanas y para los releases se pacta una distribución en base a los módulos del proyecto y fueron distribuidos en 4 releases los cuales están descrito a continuación.

El primer release, fue el módulo CRUD backend OVC, este soportar las operaciones para la gestión del objeto virtual de aprendizaje como lo son edición, creación, eliminación y modificación. Para que esto fuera posible se utilizó el framework loopback es un marco Node.js y TypeScript de código abierto para conseguir un api altamente extensible que pudiese conectarse con base de datos mongoDB.

Para el siguiente el release, se contaban con dos ítems, siendo el primero un módulo creación de login, este tiene la capacidad de hacer un logeo mediante la comunicación con

el sistema Mares de la Universidad de Antioquia, logrado mediante el consumo de un api expuestas por el departamento de gestión informática de la Universidad. El segundo ítem es el Módulo OVC el cual contiene una interfaz clara y simple mediante una serie de formularios donde los usuarios pueden realizar operaciones como edición, creación, eliminación y modificación de los objetos virtuales creativos de cada una de las materias del pregrado.

En el tercer release, contiene el módulo Rizoma en el cual se creó un grafo con base a los objetos virtuales creativos, este fue elaborado mediante la librería visJS que es una biblioteca de visualización dinámica basada en navegador, esta biblioteca está diseñada para manejar grandes cantidades de datos dinámicos y permitir la manipulación e interacción con los datos con componentes DataSet, Timeline, Network, Graph2d y Graph3d. Adicional al grafo se creó una ventana que puede mostrar la información correspondiente del ovc en cada nodo del grafo.

El último release que se desarrolló, se pretendía crear un módulo backend consumo api Google Drive para realizar el almacenamiento del recurso creado en los objetos virtuales creativos y un módulo frontend para la captura de archivos en vista para almacenamiento Google Drive. Estos dos módulos fueron creados pero descartados por el cambio de políticas en los precios de almacenamiento en Google drive que se le hace a la Universidad de Antioquia, para suplir esta necesidad se realizó un anexo al backend ya desarrollado mediante el uso de la librería multer con la cual se capturan los archivo y se almacenan en un servidor físico, desde allí también se puede hacer descarga y consultas de los diferentes recursos almacenados.

Una vez finalizado el desarrollo del código se realizaron las pruebas respectivas a la carga del frontend desplegado en un entorno de prueba, se testeo el backend que despliega

las solicitudes masivas que se espera realicen los usuarios, los cuales son la carga de los datos en el objeto rizoma y relación.

Resultados y análisis

Como resultado de esta práctica se logró hacer un sistema de recopilación de información y almacenamiento de archivos para los objetos virtuales creativos del pregrado creación digital. Inicialmente se recopilaron todos los requisitos funcionales y no funcionales del sistema de acuerdo a la información recolectada de una serie de reuniones que se llevaron a cabo en el año 2020 cuando se estaban proyectando la necesidad del nuevo pregrado, respecto a los requerimientos se realiza un planteamiento de una arquitectura cliente-servidor con conexión a base de datos, partiendo de las tecnologías mongoDB, nodeJS y reactJS con la implementación de redux para el desarrollo del sistema.

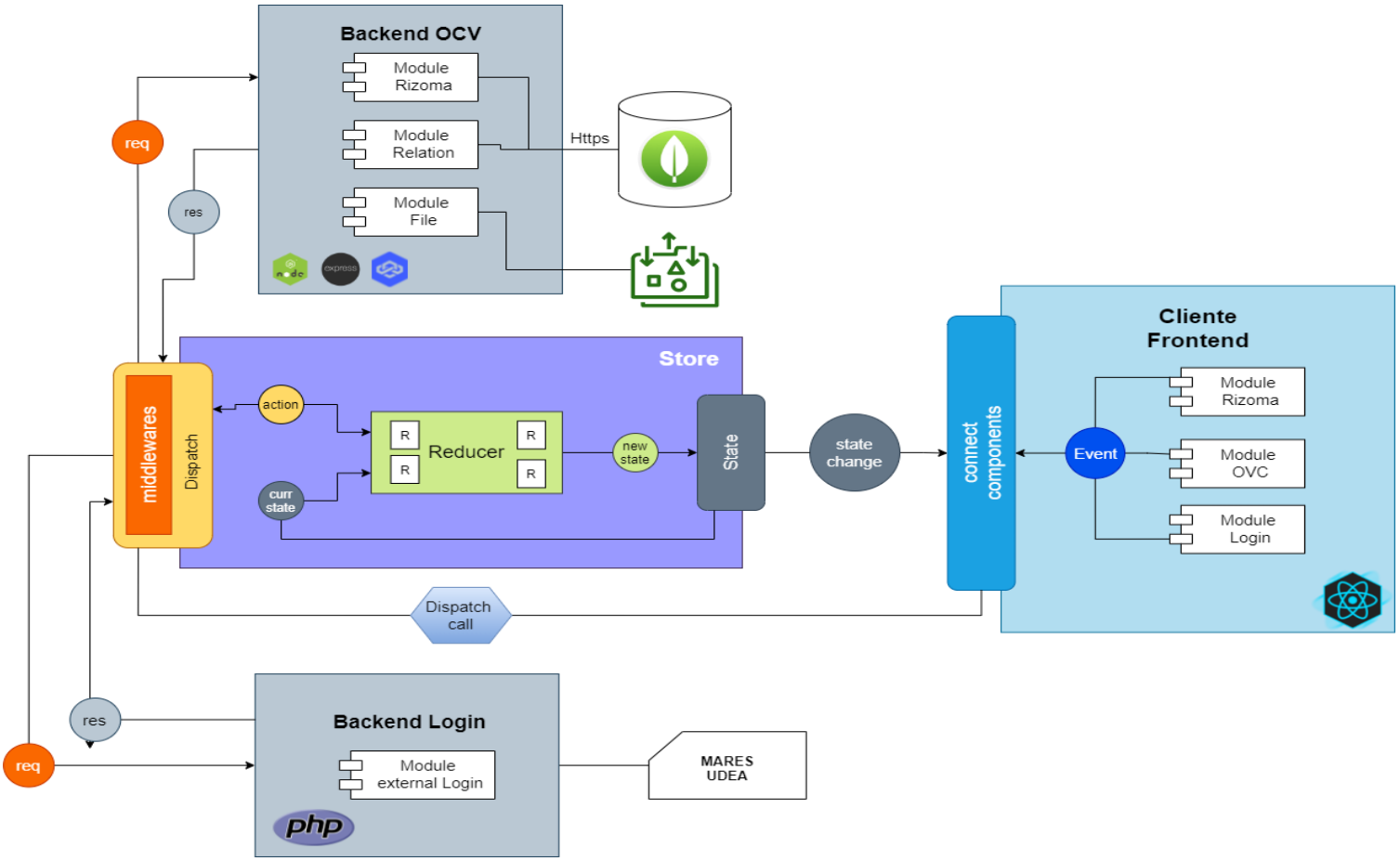

Figura 1. Arquitectura del sistema.

Se realiza un modelo de bases de datos no relacional para la conexión con un backend, este contiene el CRUD específico para los datos que serán almacenados del ovc y las diferentes relaciones o conexiones que son guardadas como referencias de cada ovc, adicionalmente se realizó un módulo para el almacenamiento de los recursos, este envía el archivo que requiere ser guardado a una carpeta en el fuente del backend y retorna los datos específicos del archivo para que sea almacenados en la base de datos como parte de la información general del ovc.

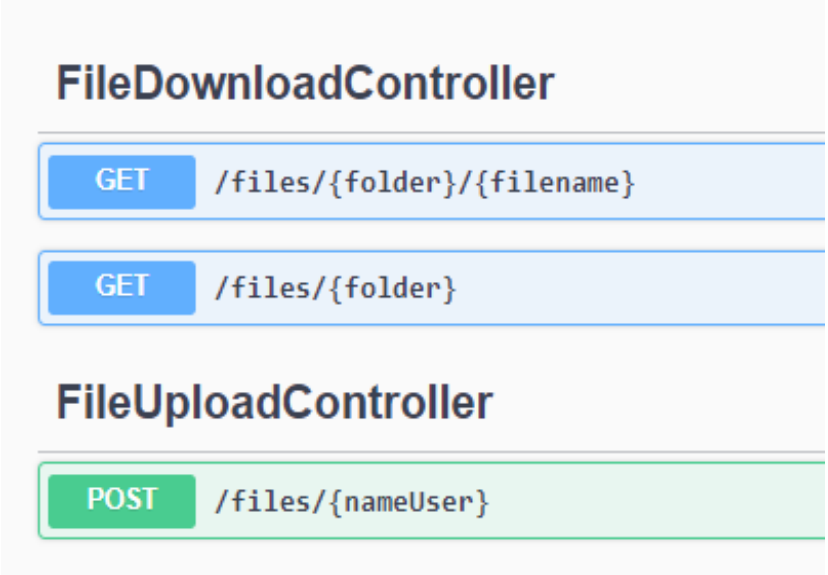

Figura 2. Módulo de almacenamiento de archivos desplegado.

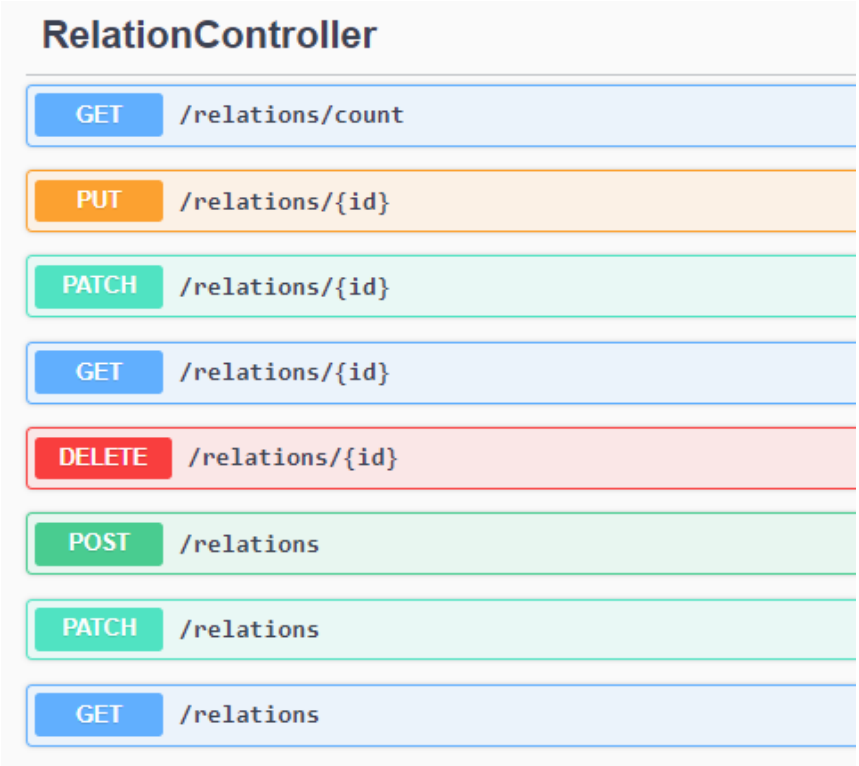

Figura 3. Módulo de relaciones backend desplegado.

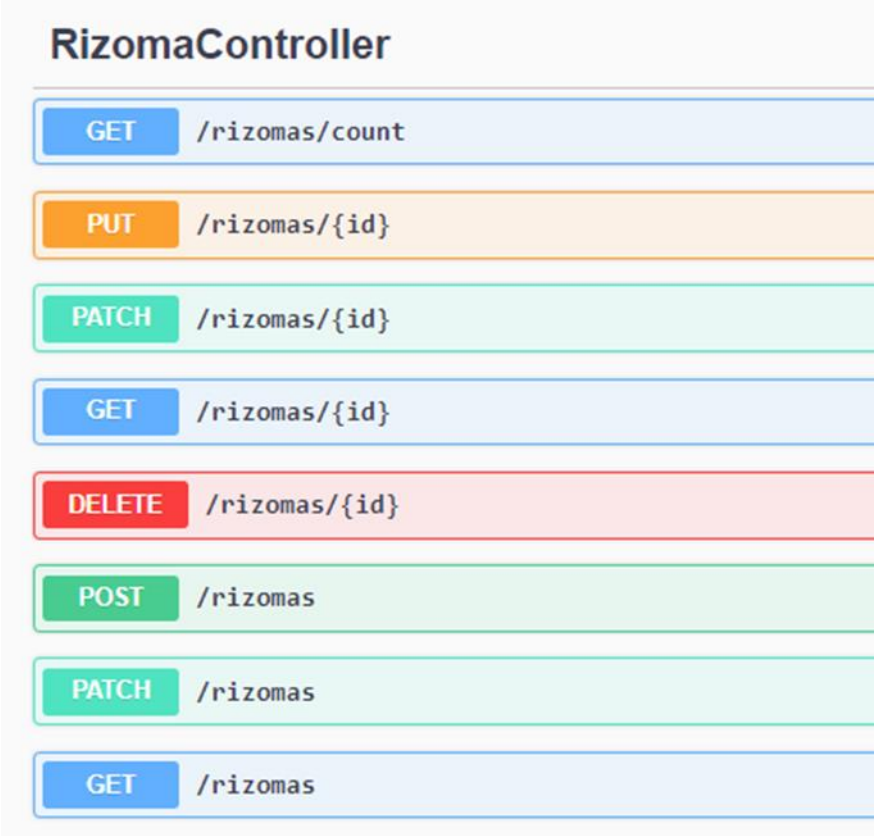

Figura 4. Módulo de rizoma backend desplegado.

Más adelante se desarrolló un frontend basado en SPA (single page application) con las tecnologías reactJS y redux. En la vista se realizó inicialmente una página de entrada donde cualquier persona que pertenezca o no al pregrado podrá conocer algunos datos relevantes del sistema, (ver figura 5).

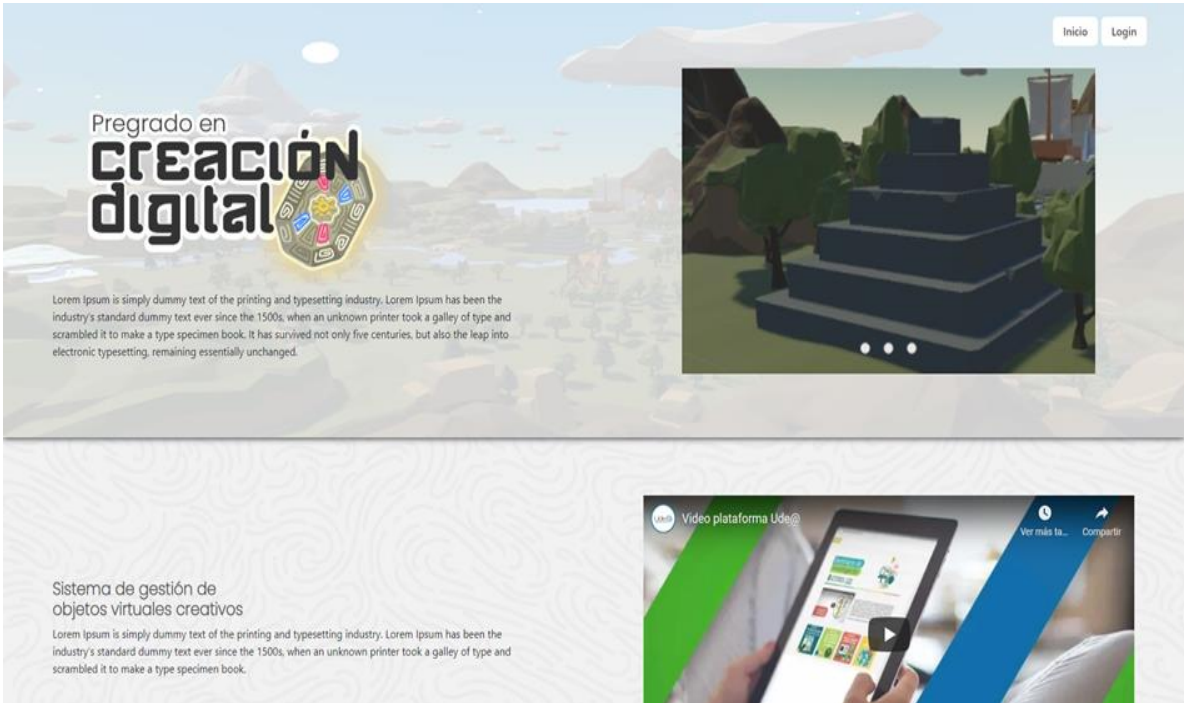

Figura 5. Aplicativo página inicial.

Para interactuar con las diferentes funcionalidades del sistema es necesario realizar un inicio de sesión, para ello basta con colocar el usuario y contraseña que tiene un estudiante o profesor con el sistema Mares de la Universidad de Antioquia, el servicio para poder lograr esto fue dado por el departamento de gestión informática quien expone un API que es usada por múltiples sistemas de la universidad, (ver figura 6).

Figura 6. Aplicativo login.

Una vez se ingresa al sistema se cuenta con un menú estilo sidebar donde se puede navegar y en cada ruta se tienen múltiples acciones. Igualmente se podrán encontrar algunas tablas con los datos respectivos a los ovc; la primera tabla contiene todos los ovc's que ya han sido publicados por los estudiantes y que tienen un estado activo, la segunda los cuenta con los ovc's elaborados por el usuario que está logueado en el momento, allí podrá inhabilitar o editar un ovc e incluso se presenta la opción de agregar uno nuevo con sus respectivas referencias, las cuáles serán las conexiones que hay entre un ovc y otro (Ver figura 11), la tercera tabla solo se muestra a los profesores, allí el docente tiende la opción de escoger el curso que dicta y visualizar los diferentes ovc's que realicen sus estudiantes, y por último en cada una de estas tablas, se puede desplegar una ventana con toda la información referente al ovc seleccionado (ver figura 12)

Figura 7. Aplicativo página inicial con menú sidebar.

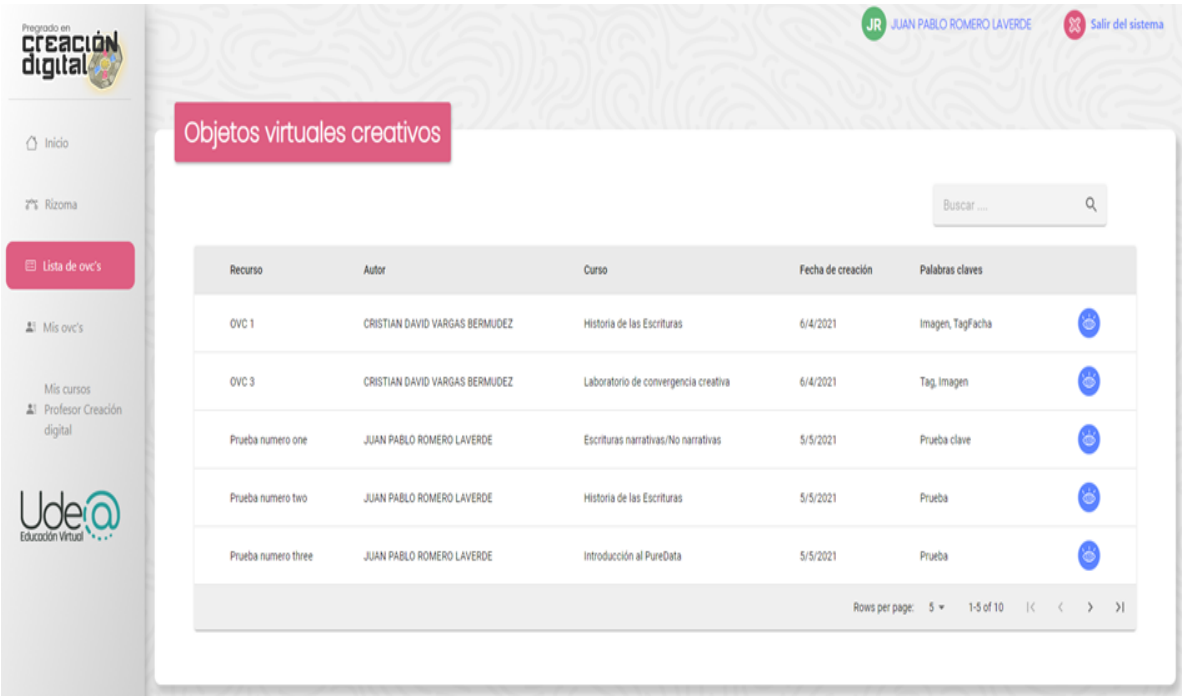

Figura 8. Aplicativo tabla ovc's generales.

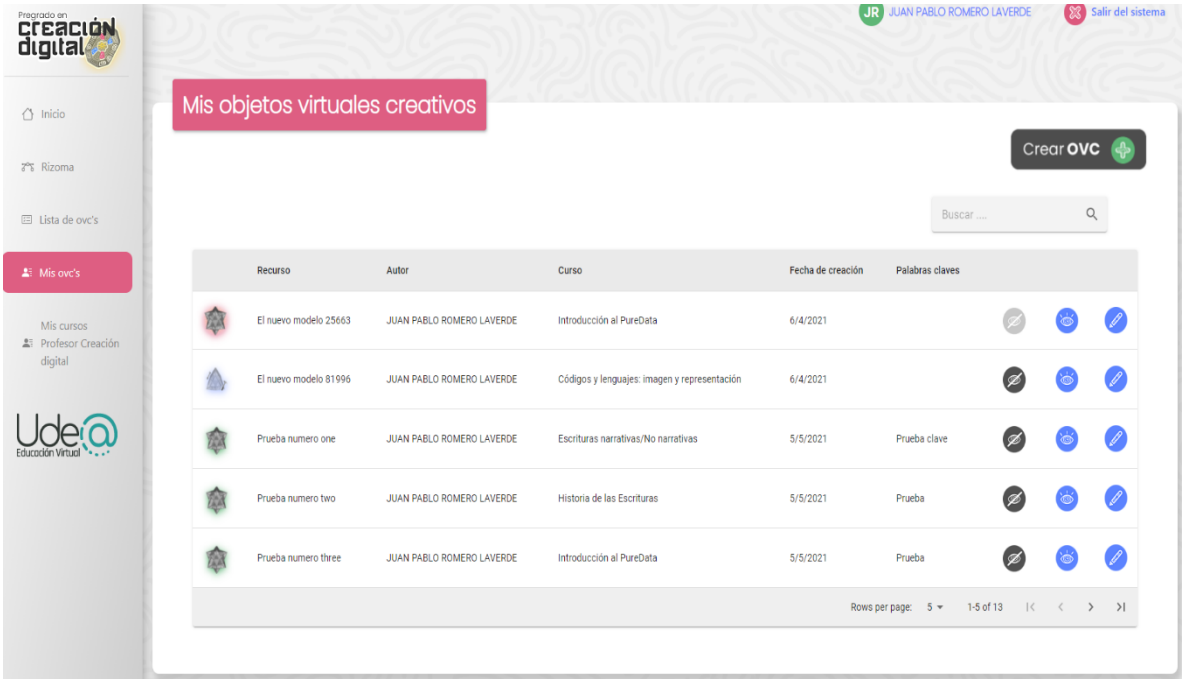

Figura 9. Aplicativo tabla ovc's estudiantes especifico.

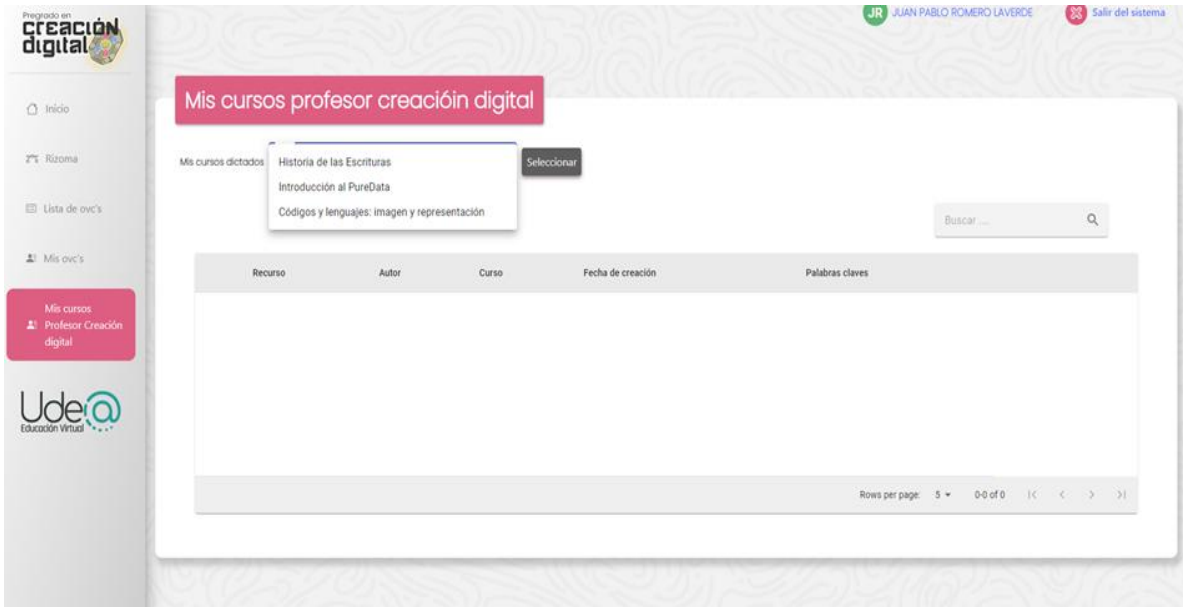

Figura 10. Aplicativo tabla ovc's profesor.

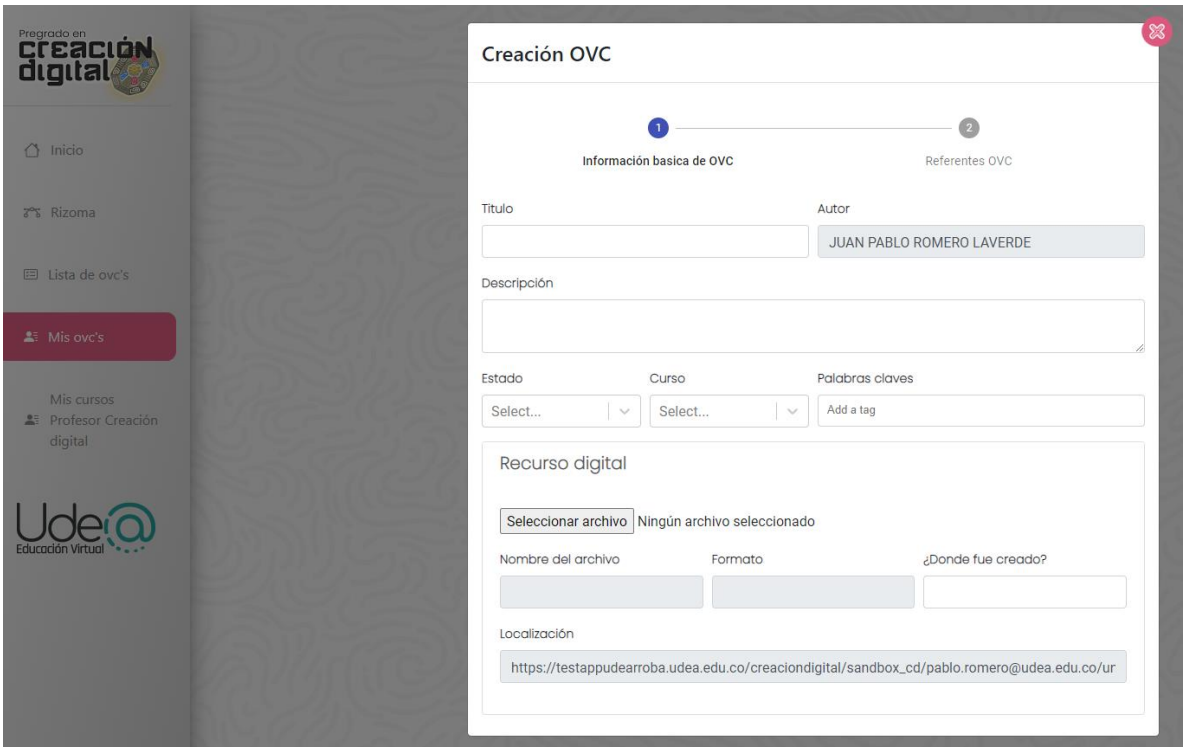

Figura 11. Aplicativo ventana registro de ovc.

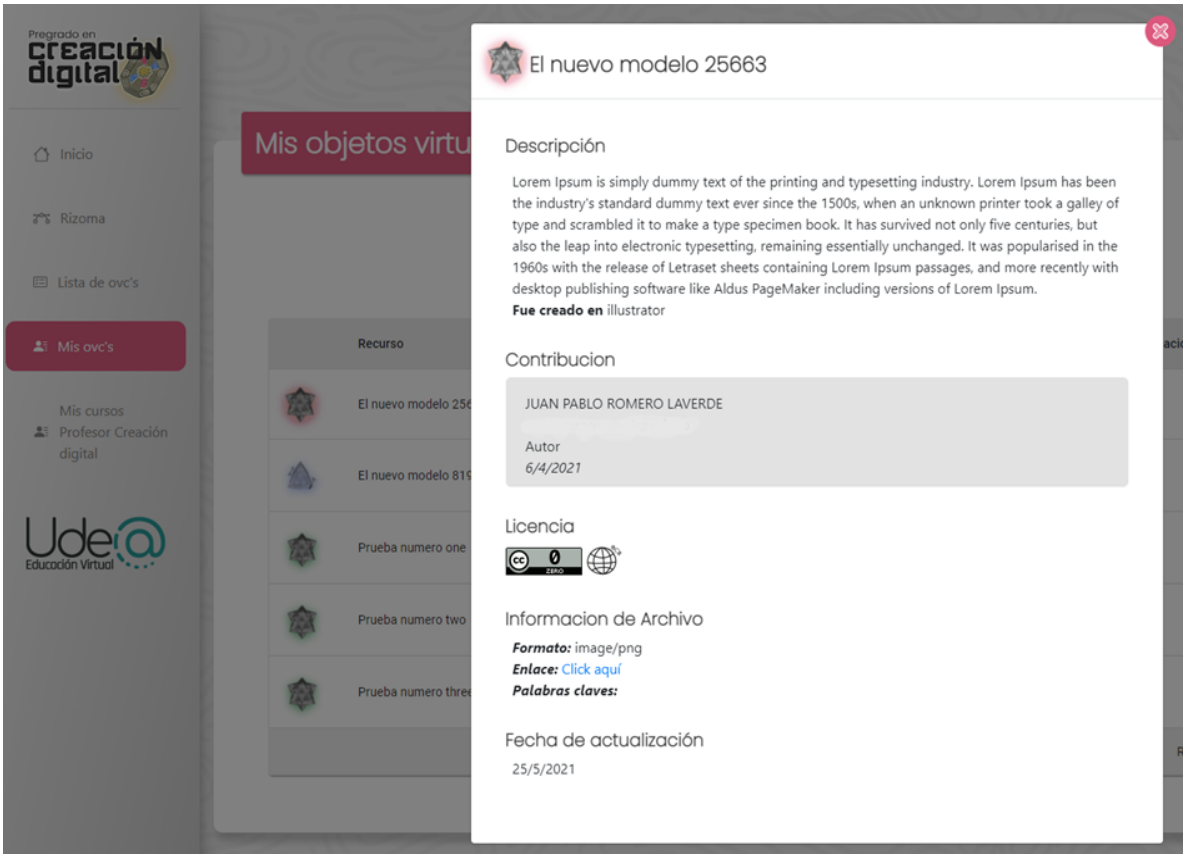

Figura 12. Aplicativo ventana información de ovc.

Por otro lado, en el sistema se tiene la posibilidad de visualizar un grafo donde cada uno de los nodos representa un ovc de un curso y las conexiones respectivas son las referencias que tiene un ovc sobre otro, además de esto se cuenta con filtros relacionados a valores específicos de los datos del ovc, como lo pueden ser autor, correo, título o curso (ver figura 13). De igual forma al presionar o dar clic sobre alguno de los nodos se despliega las ventanas que permiten al estudiante ver la información detallada del ovc (ver figura 12).

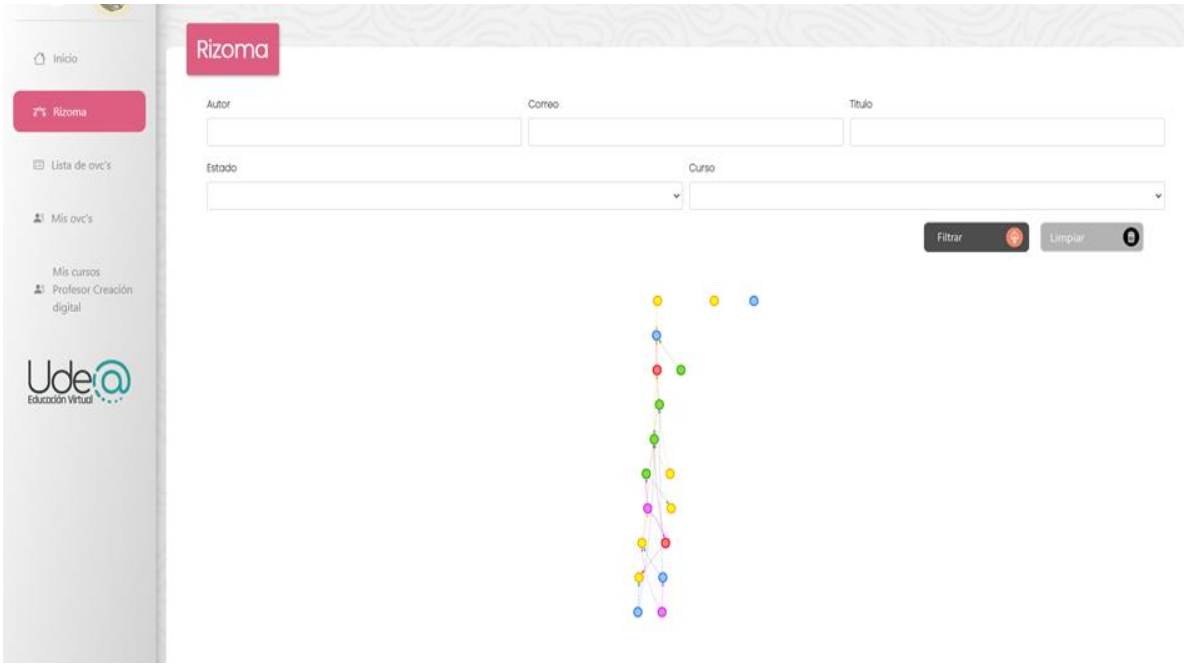

Figura 13. aplicativo página grafo.

Posteriormente se dispuso la realización de diferentes pruebas de carga para comprobar la cantidad de estudiantes que el sistema soporta en el momento, para estas pruebas se tomaron 500 hilos como la máxima carga esperada en 5 años, 100 hilos para la carga máxima esperada en 1 año y 50 hilos como la carga máxima esperada para un semestre. A estas pruebas se presentaron los siguientes resultados:

	Número de hilos	Segundos	Desviación	Rendimiento	Media	Mediana	Error
Carga a frontend	500	1	1287	3.721,161/minuto	5409 ms	5356 ms	0.00%
	100	1	16	4.866,18/minuto	222 ms	221 ms	0.00%
	50	1	654	1007.39/minuto	1242 ms	1225 ms	0.00%
Carga a backend con petición a Rizoma	500	1	1682	3.638,128//minuto	3519 ms	3337 ms	0.00%
	100	1	587	1.721,17/minuto	1493 ms	1445 ms	0.00%
	50	1	189	1.518,219/minuto	520 ms	483 ms	0.00%
Carga a backend con petición a Relation	500	1	924	7.037,298/minuto	1997 ms	1681 ms	0.00%
	100	1	128	3342,618/minuto	658 ms	685 ms	0.00%
	50	1	42	2279,635/minuto	374 ms	357 ms	0.00%

Tabla 1. Resultados pruebas de carga.

De acuerdo con los resultados obtenidos en la carga del sistema, se presentan datos en el rendimiento (Número de solicitudes procesadas por minuto) superiores a la desviación (Dispersión de los datos con respecto a la media), en consecuencia, se muestra que la aplicación desarrollada puede soportar la carga máxima de acuerdo con la cantidad de estudiantes máximo esperado en el pregrado para el primer semestre, primer año y los primeros 5 años.

Finalmente, se realiza una reunión con los involucrados en el proyecto y se entrega la información respectiva de la aplicación como lo es el código fuente y enlaces del sistema desplegado. Los manuales técnicos y de usuario se entregan para que sean revisados por guionistas y diseñadores, con la intención de ser utilizados una vez finalice el proceso de revisión por parte de los entes que darán el aval de inicio del nuevo pregrado creación digital.

Conclusiones

El proceso de ingeniería de software ha tomado un lugar muy importante hoy en día en las prioridades de las organizaciones y empresas que tiene por lo menos un departamento de tecnología, en el presente proyecto se logra crear un documento de análisis de requisito donde se pudo estimar el tiempo y gastos durante todo el proyecto, sino se hubiese realizado, en el mejor de los casos, el proyecto acabaría costando mucho más que si le hubiésemos prestado atención a alguna fase primordial desde el principio. Según Herrera, citado por Arias Chaves (2005) estos requerimientos están fundamentado en beneficios de la ingeniería de requisitos como:

- *Permite gestionar las necesidades del proyecto en forma estructurada:* Cada actividad de la IR consiste en una serie de pasos organizados y bien definidos.
- *Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados:* La IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- *Disminuye los costos y retrasos del proyecto:* es sabido que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro; especialmente aquellas decisiones tomadas durante la IR, ya que es una de las etapas de mayor importancia en el ciclo de desarrollo de software y de las primeras en llevarse a cabo.
- *Mejora la calidad del software:* La calidad en el software tiene que ver con cumplir un conjunto de requerimientos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.).

- *Mejora la comunicación entre equipos:* La especificación de requerimientos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre, el proyecto no será exitoso.
- *Evita rechazos de usuarios finales:* La ingeniería de requerimientos obliga al cliente a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema, por lo que se le involucra durante todo el desarrollo del proyecto.

(p. 4)

Por tal motivo, la importancia que dio la base de datos MongoDB fue significativa al momentos desarrollo del sistema, pues, proporciona una mayor fiabilidad de manipulación flexible y almacenamiento masivo de la información, igualmente de los tiempos de respuesta que genera, lo cual, se puede ver en el estudio realizado por Soldi (2017) donde concluye que los resultados de los estudios basados MongoDB tienen estructuras distribuidas de bases de datos que permiten mejorar los tiempos de respuesta sobre las estructuras centralizadas SQL.

Por otro lado, el backend elaborado con el framework loopback facilitó en gran medida el montaje de este, además de una importante reducción de tiempo al momento de codificar gracias a la reutilización de código y la promoción de buenas prácticas de desarrollo mediante patrones como el SOLID, entre otro. De igual manera es un backend escalable que puede expandirse a diferentes módulos que quieran ser montados en un futuro, eso permite que sea una herramienta enriquecedora, la cual queda muy bien para el sistema, ya que, al ser un pregrado nuevo, las ideas que pueden llegar en un futuro puede materializarse en este backend para el manejo de altos o pequeños flujos de datos.

Asimismo, para el lado del cliente utilizar el framework React en compañía de Bootstrap y material ui de react nos permitió realizar una aplicación responsive y adaptativa, con la cual tenemos una gran flexibilidad al momento de escoger el dispositivo con el cual se quiera acceder a la aplicación y esto hace que los usuarios puedan tener una buena experiencia de manejo y navegación sin encontrar errores que frustren la manipulación del sistema.

En consecuencia, al proceso de desarrollo elaborado, se logró realizar una aplicación que cuenta con múltiples vistas, información precisa de los diferentes recursos elaborados por los estudiantes y acciones claras para la creación, edición, eliminación y visualización de los diferentes ovc's que satisfacen los diferentes requisitos planteados por los involucrados en el proyecto y permite tener un espacio virtual en la universidad para que los estudiantes del pregrado creación digital tengan sus elaboraciones guardadas de forma segura.

De igual modo, se garantizan mediante pruebas de carga que puedan acceder a la aplicación 500 usuarios por minuto sin que esta sufra fallas y que el sistema pueda realizar peticiones para el manejo de la información con la carga máxima, esto indica que la buena experiencia no solo será visual, sino que se tendrá un sistema de respuesta rápido.

Finalmente, la realización de los manuales usuario y técnico se le da a diferentes involucrados con información muy clara y concisa para resolver muchas dudas que se pueden generar acerca del sistema. Todo esto es debido a la importancia que se tiene conocer cuales son los datos de entrada, salida, resultados y definir cuáles son las diferentes funciones que una persona puede encontrar y cómo debe ser la interacción para tener una buena experiencia.

Referencias

Arias, M. (2005). La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *InterSedes: Revista de las Sedes Regionales, 6*(10), 1-13. Recuperado de https://www.redalyc.org/pdf/666/66612870011.pdf

Araujo,A. (2016). ¿Qué es una Base de Datos NoSQL?. Oracle Blogs. Recuperado de https://blogs.oracle.com/spain/qu-es-una-base-de-datos-nosql

Puerta, J. M. (2015). *Desarrollo de una API para la descripción y gestión de Servicios Web RES*T (Trabajo de grado). Universitat Jaume I, España.

Soldi, P. J. (2017). *Estudio de rendimiento en MongoDB sobre arquitecturas centralizadas y distribuidas* (Trabajo de grado). Universidad Nacional de la Plata, Argentina.

Bibliografía

Abraira, V. (2002). Desviación estándar y error estándar. SEMERGEN, 28(11), 621,623. Recuperado de https://www.elsevier.es/es-revista-medicina-familia-semergen-40-pdf-S1138359302741385

Ciencia de la Computación e IA All rights reserved. (2013). Introducción a los Servicios Web RESTful. Recuperado de http://www.jtech.ua.es/j2ee/restringido/cw/sesion11-apuntes.pdf

https://blogs.oracle.com/spain/qu-es-una-base-de-datos-nosql

Marini, E. (2012). El Modelo Cliente/Servidor. Recuperado de https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf

Peng, D., Cao, L. Y Xu, W. (2011). Using JSON for Data Exchanging in Web Service Applications. *Journal of Computational Information Systems*, *7*(16), 5883-5890. Recuperado de https://www.researchgate.net/profile/Dunlu-Peng/publication/265874991_Using_JSON_for_Data_Exchanging_in_Web_Service_Applications/links/5523cd1f0cf2c815e07325ea/Using-JSON-for-Data-Exchanging-in-Web-Service-Applications.pdf

Rodríguez, A., Rodríguez, D. y Díaz, E. (2016). Selección de Base de Datos No SQL para almacenamiento de Históricos en Sistemas de Supervisión. *Revista Cubana de Ciencias Informáticas*, *10*(3), 159-170. Recuperado de http://scielo.sld.cu/pdf/rcci/v10n3/rcci12316.pdf