Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

T E S I S

“Propuesta de una herramienta web educativa semántica”

Presenta: *Alex Uriel Acoltzi Ruiz*

Para obtener el grado de: Licenciatura en ingeniería en tecnologías de la información

Director: *Dr. Abraham Sánchez López*

Puebla, Pue, 2025

Índice General

Índice de figuras

Capítulo 1: Introducción

El uso de las tecnologías nos ha aportado un gran valor en distintos ámbitos de nuestra vida cotidiana, desde procesos básicos como una comunicación más sencilla y efectiva, hasta el ambiente laboral, en el cual tenemos distintas formas de inclusive gestionar los proyectos, mediciones de métricas y proyecciones, etc. Otro ámbito que se ve beneficiado es el ambiente del entretenimiento, ya que poseemos diversas formas de distracciones como videojuegos, películas, etc. Sin embargo, contrastado con el ámbito escolar que forma parte del desarrollo personal ha quedado rezagada, actualmente, esto quiere decir que, las tecnologías se han aprovechado en muchos ámbitos de la vida diaria, entonces, ¿Por qué el sistema educativo no ha implementado de manera plena y efectiva el uso de las tecnologías? Hoy en día contamos con muchas herramientas que nos pueden servir de guía para continuar con el aprendizaje, haciendo una interacción más dinámica. Aunque en los niveles medio superior y superior se han tomado algunas herramientas para la implementación en la enseñanza, no se ha explotado como en otros ámbitos, es decir, su utilización no ha sido 100% implementada para lo que realmente fueron diseñadas. Esto es debido a que existen distintos factores que nos sitúan en una brecha remarcada para poder hacer uso de estas.

1.1 Planteamiento del problema

A pesar de todas las herramientas y avances tecnológicos que existen y su adaptación, en distintos ámbitos, en el sector educativo ha tenido una implementación limitada, esto debido a distintos factores externos, por ejemplo la brecha digital que afecta el acceso a dispositivos y a la

conectividad que se pueda poseer por la infraestructura que pueden llegar a tener los centros educativos, en algunos casos puede existir una deficiencia en la capacidad de los docentes, los cuales no cuentan con el conocimiento ni la formación adecuada para poder incluir algunas tecnologías en la impartición de sus cursos e inclusive algunos que tienen una resistencia a este cambio y seguir por el camino del aprendizaje sin incluir algunas plataformas educativas como parte de su plan de trabajo y así utilizar los enfoques pedagógicos tradicionales.

Esto conlleva a plantearse la siguiente pregunta, ¿Cómo se podría romper esta brecha tecnológica para que las instituciones educativas puedan integrar eficazmente las tecnologías en el proceso de enseñanza para garantizar un aprendizaje más inclusivo y dinámico?

1.2 Objetivo

Proponer una plataforma educativa que funcione como base en un entorno educativo, capaz de evolucionar conforme se vaya requiriendo, que sirva tanto a docentes y alumnos para tener un ambiente más productivo para el aprendizaje.

1.3 Estado del arte

La época de la pandemia causada por COVID-19 ha marcado un período de avances significativos en el ámbito de la educación. En el año de 2019, la inversión global asociada al uso de tecnologías para el ámbito educativo alcanzó la cifra de 18.66 mil millones de dólares (Markets Insider, 2019). Si bien, esta ya es una cifra considerable, el paso de esta enfermedad acelero la implementación de estos recursos, puesto que, se estima que alrededor de 1200

millones de estudiantes de 186 países fueron afectados debido al cierre de las instituciones educativas en distintos niveles académicos, teniendo en cuenta esto, se estima que las inversiones en EdTech para el año 2025 alcance una cifra de 350 mil millones de dólares (WebForum), lo cual deja un incremento de un 1973%, esto expande la visión, sobre el uso de las tecnologías en un ámbito que por el momento no ha cambiado su estructura de trabajo desde hace mucho tiempo o por lo menos no lo hacía con la eficacia con la que crecían dichas tecnologías.

En la actualidad, contamos con una amplia gama de tecnologías que, de diversas maneras, han contribuido al desarrollo cognitivo de los estudiantes. Entre estas tecnologías, destacan herramientas prácticas para la impartición de clases en línea o de forma híbrida, tales como Teams, Classroom, Meet, y muchas otras.

Si bien, estas tecnologías ya tienen tiempo en el mercado, tuvieron una mayor exposición debido a las necesidades mencionadas previamente, en los siguientes textos veremos un poco más las funcionalidades que tienen y el impacto que tuvieron dichas herramientas en el periodo de pandemia.

Google Classroom se ha convertido en una herramienta muy útil y una de las más conocidas, debido a que es una herramienta gratuita, esta herramienta fue desarrollada por Google y fue lanzada el 12 de agosto de 2014, una de sus principales funciones es organizar los cursos, en los cuales tendremos una fuente de comunicación bidireccional entre profesores y alumnos mediante anuncios, hilos de conversaciones, retroalimentación de actividades, etc., además, se pueden adjuntar guías, documentos, links, archivos y otro tipo de contenido, lo cual facilita el acompañamiento de la información que no se centra en lo que imparte el profesor, es decir, que se pueden adjuntar complementos para los cursos/clases, generando un espacio más amplio para la comprensión de los alumnos. También, es posible asignar tareas, ejercicios,

exámenes, etc. Y estos, pueden ser configurados por el profesor, permitiendo controlar los tiempos de entrega, evaluar y tener un control de las calificaciones de cada alumno, del lado del alumno, esto genera tener un conocimiento general del conteo de sus entregables. Ahora bien, esta herramienta se limita solo a esto, si bien, es una herramienta excepcional, sirve como una herramienta extra para la organización de entregables y brinda la facilidad de generar los espacios para tener información adicional y/o propias del curso. (Google LpublkLC, s.f.)

Concretamente no se tienen estadísticas sobre el incremento en sus usuarios debido a la pandemia, pero, se debe contemplar que es una buena herramienta, que integrada con el Workspace de Google es un sistema muy útil, por ejemplo, integrarlo con el uso de Google meet para la impartición de clases virtuales genera un espacio completo para la impartición de cursos de forma virtual. Otro punto importante es, la facilidad de uso, así como, una interfaz usable e intuitiva, lo que logró posicionarla como una de las herramientas preferidas tanto por profesores como alumnos.

Por otro lado, tenemos a Microsoft Teams, esta es otra herramienta la cual fue creada por Microsoft y lanzada en el año 2017, aunque esta esta más enfocada para el uso organizacional, sin embargo, también puede ser utilizada en el ámbito educativo. Dentro de sus principales funcionalidades está, la posibilidad de generar espacios especializados, en los cuales se tiene un medio para compartir ideas, archivos, etc. Esto se asemeja a un foro, en el cual, se tienen varios hilos de conversaciones, adicionalmente este cuenta con la posibilidad de manejar la comunicación de forma más directa, es decir, cuenta con espacios de mensajería instantánea entre miembros del equipo y por último este cuenta con la posibilidad de tener videoconferencias con la posibilidad de grabarlas en caso de que sea algo necesario y manteniendo la información en los hilos previamente mencionados.

Como se mencionó, Microsoft Teams, se lanzó en el año 2017 y para el año 2018 contaba con un total de 8 millones de usuarios y para años posteriores siguió la tendencia de incremento. En el año 2019 contaba con un total de 20 millones de usuarios, el siguiente año se registró un total de 75 millones de usuarios, lo cual ya representa un incremento de 275% de usuarios y para el año 2021 se registró un incremento del 94% aproximadamente, ya que se registró que la aplicación era usada por 145 millones de usuarios (Curry, 2024).

Microsoft Teams annual users 2017 to 2022 (mm)

Year	Users (mm)
2017	2
2018	8
2019	20
2020	75
2021	145
2022	270
2023	300

Source: Company data

Figura 1.1 Total de usuarios de Microsoft Teams

Sin embargo, hay que contemplar que estos números son generales, es decir, contemplan la herramienta para el uso organizacional, es decir, no solo contemplan su uso en el ámbito escolar, sino, también en el profesional.

A raíz de la pandemia pudimos notar que las herramientas para la educación en línea son muy necesarias, y si bien, es cierto que tenemos muchas herramientas a nuestra disposición, también, esto puede terminar en ser una problemática. Cada docente tiene su estructura de trabajo, que se tuvo que adaptar de forma radical para la impartición de los cursos de forma

online y esto generó el uso de distintas herramientas que se adaptaron al marco de trabajo de cada docente, esto dio pie a notar las deficiencias entre dichas herramientas, desde la falta de capacidad para tomar el provecho al 100% de estas, hasta la falta de información tanto para los docentes como para los alumnos, esto perjudicó en el uso pleno de las aplicaciones que estaban a nuestra disposición, puesto que, no nos podíamos centrar en el uso de una sola herramienta y esto generó una deficiencia en el aprendizaje.

Para poder hacer uso de estas herramientas y romper la barrera del aprendizaje, es necesario eliminar esa brecha tecnológica, el primer paso, para la comunidad BUAP es dejar de usar tantas herramientas y centrarse en una sola, aprender al máximo ésta y así sacarle el mayor provecho, para ello surge la idea de este proyecto: una plataforma diseñada para ofrecer, materiales para el aprendizaje, facilitar asignaciones, actividades y calificaciones a la comunidad BUAP. Además de su función académica, esta plataforma se concibe como una red social, destinada a facilitar la comunicación entre estudiantes, docentes y la administración educativa.

Es fundamental destacar que esta plataforma está pensada en exclusividad para la comunidad universitaria, por ende, es necesario estar activo ante la universidad, contando con un correo institucional o una matrícula.

Tomando como partida las aplicaciones antes mencionadas y aplicando el conocimiento de la web semántica y ontologías que brinden una mayor transparencia a la información y la facilidad de relación para encontrar más información asociada a esta o que pueda llevar a la profundización de la misma, lo que genere una facilidad de investigación con el tema. (Noy & McGuinness, 2001)

1.4 Web Semántica

Para comenzar con este trabajo, primero debemos partir de uno de los fundamentos principales y en este caso hablamos de la web semántica, pero primero, hay que conocer un poco de historia y desglosaremos el termino para dar un mejor contexto al fundamento.

1.4.1 Historia de la Web

La Web fue creada en 1989 por Tim Berns-Lee mientras mantenía relación laboral con la CERN (Organización Europea para la investigación Nuclear) en Suiza, tuvo origen dada la necesidad de compartir documentos e información entre científicos de manera más eficiente, dando así comienzo a la idea de un sistema de **hipertexto.**

Antes de continuar, expliquemos que es el término de hipertexto**,** por lo tanto, navegar entre información y documentos mediante hipervínculos.

En octubre de 1990, El propio Tim redacto los 3 pilares tecnológicos para la implementación de lo que hoy conocemos como la fundación de la web.

- HTML (HyperText Markup Lenguage): el lenguaje con el que actualmente se elaboran las páginas web, donde se puede definir la estructura y el contenido.
- URI Uniform Resource Identifier o identificador Uniforme de Recursos: es una cadena con la cual se puede identificar las páginas y su contenido en la web.
- HTTP (HyperText Transfer Protocol): Protocolo utilizado para transferir documentos mediante la web.

En el año de 1991 se plasmaron estos pilares en la primera página web que fue lanzada en la red de CERN, sin embargo, como se menciona la idea en general se tomó como una idea privada, pero, en el año de 1993, la propia CERN decidió hacer que la web fuera de acceso público y gratuita, lo que permitió el inicio de una expansión global (World Wide Web Foundation, 2009).

1.4.2 La era de los navegadores y la popularización de la web

El desarrollo de los navegadores también fue una pieza fundamental para la historia de la web. En 1993 con la creación del navegador Mosaic, por parte de Marc Andreessen y su equipo de la universidad de Illinois, dio pie a que el público en general pudiera hacer uso de la web, puesto que, fue el primer navegador gráfico que permitía mostrar imágenes en conjunto con el texto, facilitando la navegación visual.

En 1994, el mismo Andreessen cofundo Netscape Communications y con esto el lanzamiento del navegador Netscape Navigator, posicionándose como el más usado en su momento(Adamson, 2023), sin embargo, en el año de 1995 Microsoft decidió realizar competencia lanzando su navegador Internet Explorer, dando así el comienzo con la “guerra de los navegadores”(Day in Tech History, 2019) donde al final de los 90’s el ganador fue Microsoft y dando un foco más importante a nivel global sobre la web, con todas las nuevas herramientas que podían surgir.

1.4.3 Evolución de la web

Hasta este punto hemos hablado de la historia, pero, ahora es necesario hablar un poco sobre la evolución de la web, la cual ha sufrido grandes cambios, lo que en un principio era un sistema de hipertexto tuvo la necesidad de evolucionar, dado el crecimiento de la red, su alcance y complejidad, es decir, con la popularización de los navegadores y el uso de la red de forma abierta, cualquiera tenía acceso a la información por lo tanto, lo que en un principio se manejaba como información estática, que para principios básicos era funcional, puesto que no había la necesidad de contar con una gama tan alta de relación en la información, tendría que adaptarse a la basta información que tiende a extenderse, por ello, ya no se podía limitar la información a un estado simple de hipertexto, pero, antes de adentrarnos más en los avances de esto, debemos analizar, en realidad, ¿qué es la web?. Ya analizamos la historia, pero debemos enfocarnos en conocer que es en términos centralizados que es la web. 'La idea de una red información fue una idea técnica que en su momento solo podían comprender técnicos calificados' (Allemang, D., & Hendler, J. (2011)), es cierto que, en sus inicios, esta cuestión no tenía fundamentos entendibles para todo el mundo, todo cambio con la adopción del uso de la WWW, donde el concepto de una web fue tomado como una red compartida por todo el mundo. El comienzo de la web parte desde la idea fundamental de una comunidad, en la que, todos pueden tener acceso a esta y con ello poder contribuir con sus ideas a un conjunto, dando pie, al uso compartido, por ende, cualquiera que forme parte de esta red de conocimiento, tiene acceso a las ideas compartidas por otros y/o la posibilidad de plasmar sus propias ideas, dándole un crecimiento constante, por lo tanto, la información no dependía de una sola persona o entidad, cualquiera podría y puede dar un gran avance a los resultados. Entonces podemos definir a la web como una entidad orgánica que crece a base de los conocimientos de una comunidad que contribuya con el trabajo, ideas, comentarios, etc. De un tópico en particular que sea de su interés, sentando las bases para tener una basta

información, en esencia, ya tenemos una fuente de información basta, por lo que, cualquiera que tenga acceso a esta puede encontrar información de distintos temas que sea preciso consultar.

Sin embargo, aquí comenzamos con una problemática, se mencionó que la web, era información basada en hipertexto centralizada, esto nos generaba ciertos problemas, por ejemplo, si conocemos una web enfocada a la astronomía, en la que tenemos información sobre el sistema solar, estrellas, etc. Tenemos una página enfocada al sistema solar y páginas que nos hablen un poco de cada planeta que lo conforma, Antes nuestro sistema estaba compuesto por una estrella y 9 planetas, sin embargo, Plutón pasó de ser un planeta a un planeta enano, si visitamos la página que habla de este planeta ya se encuentra actualizada, en ella ya se maneja la idea de que ya es un planeta enano, sin embargo, regresamos a la página principal en la que observamos el sistema solar, esta información no está actualizada ya que se sigue manejando a Plutón como un planeta, esto quiere decir que, el cambio que se hizo a la información particular sobre Plutón no sincronizó esta información con la que se encuentra sobre la página principal, es decir, no estaba homologada, lo que genera esa discrepancia y a esto se le puede dar el nombre de “Dump web”. Esto fue lo que originó el nacimiento de la idea de la Web semántica.

1.4.5 ¿Qué es la Web semántica?

1.4.5.1 La Evolución Hacia la Web Semántica

Desde la creación de la World Wide Web, la forma en que los usuarios interactúan con la información en línea ha cambiado drásticamente. Hoy en día, nos encontramos en una era en la que no solo accedemos a datos, sino que también buscamos comprenderlos de una manera más eficiente y relevante. En este contexto, surge la idea de la Web Semántica, un concepto que

pretende transformar nuestra interacción con la información en la red al añadir una capa de comprensión significativa que va más allá de simples palabras clave.

1.4.5.2 Comprendiendo la Semántica

Para abordar la Web Semántica, es esencial primero entender el significado de "semántica". La Real Academia Española define la semántica como la "disciplina que estudia el significado de las expresiones lingüísticas, sean palabras, sintagmas u oraciones". En otras palabras, la semántica se ocupa de interpretar el significado y el contexto detrás de las palabras y frases que utilizamos, considerando las variaciones en su uso dependiendo del entorno y la intención comunicativa.

Por ejemplo, tomemos la palabra "tarjeta". En un contexto financiero, podría referirse a una tarjeta de crédito o débito, mientras que, en un entorno tecnológico, podría implicar una tarjeta gráfica o una tarjeta madre. La semántica analiza estas diferencias para determinar el significado más apropiado en cada situación. Esta capacidad de discernir y asignar significado contextual es la base sobre la que se fundamenta la Web Semántica.

1.4.5.3 ¿Qué es la Web Semántica?

La Web Semántica es la evolución de la web hacia una plataforma que pueda comprender y dar contexto a la información presentada en línea. Es una idea propuesta por el creador de la Web, Tim Berners-Lee, que busca dotar a la red de una especie de inteligencia artificial capaz de interpretar el contenido de las páginas web no solo por palabras clave, sino también por su significado y contexto. En lugar de tratar la web como un vasto conjunto de documentos desconectados, la Web Semántica permite que los datos se relacionen de manera significativa, haciendo posible que las máquinas comprendan la información como lo haría un ser humano.

El objetivo principal es proporcionar a la web una estructura que permita a los sistemas procesar, compartir y reutilizar datos entre diferentes aplicaciones y comunidades de manera automática y precisa. Un ejemplo claro son los motores de búsqueda, que en una web semántica podrían entender mejor las intenciones del usuario al realizar una consulta, proporcionando resultados más precisos y contextuales.

1.4.5.4 Ejemplos Prácticos y Beneficios de la Web Semántica

Uno de los ejemplos más evidentes del potencial de la Web Semántica es la capacidad de los motores de búsqueda para interpretar consultas de manera más intuitiva. Actualmente, cuando realizamos una búsqueda en un motor como Google, los resultados se basan en términos similares o coincidencias exactas. Sin embargo, una web semántica permitiría que el sistema comprendiera no solo las palabras, sino también la intención detrás de ellas, generando resultados más relevantes y específicos.

Por ejemplo, si buscas "restaurantes italianos que acepten mascotas cerca de mí", un sistema semántico ideal podría comprender que estás interesado en restaurantes específicos (italianos), que tengan una característica particular (aceptar mascotas) y que estén dentro de un rango geográfico determinado. En una web tradicional, estos parámetros deben indicarse de manera explícita, mientras que, en una web semántica, la consulta se interpretaría de manera más natural y precisa.

1.4.5.5 Los Desafíos de Implementar la Web Semántica

Aunque la idea de la Web Semántica es prometedora, su implementación ha encontrado varios desafíos. Uno de los problemas principales es la falta de una estandarización completa en la forma en que se codifican y describen los datos en la web. Las tecnologías como RDF (Resource Description Framework), OWL (Web Ontology Language) y SPARQL (SPARQL

Protocol and RDF Query Language) son herramientas desarrolladas para facilitar la creación de una web más semántica, pero su adopción ha sido lenta y desigual. (Bechhofer et al., 2004)

Además, la Web Semántica requiere que los datos en línea estén organizados de una manera que permita a los sistemas interpretarlos adecuadamente. Esto implica un esfuerzo significativo para las organizaciones que deben estructurar sus datos de acuerdo con estos nuevos estándares. También hay preocupaciones sobre la privacidad, ya que una mayor interconexión y comprensión de los datos podrían llevar a un uso indebido de la información personal.

1.4.5.6 Avances y Estado Actual de la Web Semántica

A pesar de los desafíos, se han logrado avances significativos en la comprensión semántica de la información. Google, por ejemplo, ha implementado mejoras en su algoritmo para ofrecer resultados más contextuales y precisos. En 2013, la introducción del algoritmo Hummingbird permitió a Google comprender mejor la intención detrás de las búsquedas en lugar de simplemente analizar palabras clave. A esto le siguieron desarrollos como RankBrain y BERT, que utilizan inteligencia artificial y aprendizaje profundo para mejorar la precisión de los resultados de búsqueda.

Un estudio realizado por Backlinko en 2023 señala que, en promedio, los usuarios tardan aproximadamente 9 segundos en hacer clic en un resultado de búsqueda, lo que indica que todavía existe una brecha en la precisión de los motores de búsqueda en términos de intención del usuario quiere que, aunque ha habido avances en la comprensión semántica, la web aún no ha alcanzado el nivel de inteligencia y precisión que promete la Web Semántica.

1.4.5.7 La Promesa de la Web Semántica en el Futuro

En el futuro, se espera que la Web Semántica transforme aún más la forma en que interactuamos con la información en línea. Tecnologías emergentes como la inteligencia artificial

y el procesamiento del lenguaje natural (NLP) están convergiendo con la Web Semántica para crear plataformas que no solo respondan a consultas, sino que también anticipen las necesidades de los usuarios. Además, la Web 3.0, que es considerada la evolución natural de la Web Semántica, busca una mayor descentralización, privacidad y control sobre los datos, permitiendo que la información en la red sea más comprensible y útil.

El potencial es enorme: desde asistentes virtuales que puedan mantener conversaciones complejas hasta sistemas de recomendación que entiendan tus gustos y preferencias en un nivel profundo, la Web Semántica ofrece la promesa de una internet más inteligente, conectada y centrada en el usuario.

1.4.6 Web 2.0: La Revolución de la Participación en la Red

El concepto de **Web 2.0** surgió a principios de la década de 2000, acuñado por Tim O'Reilly y Dale Dougherty en una conferencia en 2004. Aunque la expresión ya había aparecido antes, fue O'Reilly quien la popularizó, refiriéndose a la evolución que había sufrido la web desde su creación en los años 90. Esta evolución marcó una transición crucial de la **Web 1.0**, un entorno mayormente estático, hacia una web interactiva y colaborativa, donde los usuarios ya no solo eran consumidores de contenido, sino también creadores (O'Reilly, 2005).

De la Web Estática a la Web Dinámica

Antes del surgimiento de la Web 2.0, la red estaba basada en una estructura informativa estática. Las páginas web eran, en su mayoría, documentos de texto plano que ofrecían información sin muchas posibilidades de interacción. El contenido era creado por un número reducido de autores, generalmente expertos en el tema, y el papel del usuario se limitaba a

consumir esa información sin opción a contribuir. Este tipo de web fue posteriormente denominado "Web 1.0" (Wikipedia, 2024).

Con la llegada de la **Web 2.0**, este enfoque cambió radicalmente. La interacción pasó a ser el centro de la experiencia en línea. A través de plataformas que facilitaban la participación, los usuarios podían crear contenido, colaborar en proyectos comunes y comunicarse de formas innovadoras. Así, la información comenzó a generarse en tiempo real y se incrementó exponencialmente la cantidad de datos disponibles en la red. Esto llevó a la creación de comunidades en línea y a la formación de nuevas estructuras sociales en el mundo digital (Backlinko, 2023).

Herramientas y Plataformas de la Web 2.0

La revolución de la Web 2.0 no hubiera sido posible sin la aparición de herramientas que facilitaron la creación y difusión de contenido generado por usuarios (User-Generated Content o UGC). Entre las principales plataformas que marcaron esta etapa, encontramos:

1. Blogs

Los blogs fueron uno de los primeros formatos que permitieron a los usuarios comunes compartir sus opiniones, conocimientos y experiencias. A través de plataformas como **Blogger** (lanzada en 1999) o **WordPress** (2003), las personas podían crear sitios web sin necesidad de conocimientos técnicos avanzados. Esto democratizó el acceso a la publicación en línea y permitió la aparición de una gran variedad de contenidos, desde blogs personales hasta periodismo ciudadano y análisis especializado (O'Reilly, 2005).

2. Wikis

El concepto de "wiki" se popularizó con el lanzamiento de **Wikipedia** en 2001, un proyecto que ha crecido hasta convertirse en la enciclopedia en línea más grande del mundo. Las

wikis son plataformas colaborativas que permiten a cualquier usuario crear, editar y actualizar información. Este enfoque colectivo transformó la forma en la que accedemos al conocimiento, priorizando la actualización constante y el acceso libre a la información (Wikipedia, 2024).

3. Foros de Debate

Los foros, aunque anteriores a la Web 2.0, alcanzaron su auge durante esta era, volviéndose espacios esenciales para la discusión de temas específicos. Plataformas como **Reddit** y **Quora** permitieron que los usuarios debatieran y respondieran preguntas, generando comunidades alrededor de intereses compartidos y contribuyendo a la construcción colectiva del conocimiento.

4. Contenido Multimedia

La Web 2.0 vio nacer y crecer plataformas dedicadas al contenido multimedia, como **YouTube** (2005) para videos y **Flickr** para fotografías. Estas plataformas permitieron a los usuarios subir, compartir y comentar sobre imágenes y videos, haciendo posible la viralización de contenido y la creación de fenómenos culturales en línea (O'Reilly, 2005).

5. Redes Sociales

Las redes sociales, probablemente el fenómeno más destacado de la Web 2.0, transformaron completamente la interacción en línea. Plataformas como **Facebook** (2004) y **Twitter** (2006) permitieron a los usuarios conectarse, compartir sus vidas, noticias y opiniones en tiempo real. Este tipo de redes sociales también se convirtieron en una fuente principal de información para millones de personas, desplazando a los medios tradicionales en muchos aspectos (Backlinko, 2023).

La Tecnología Detrás de la Web 2.0

El avance hacia la Web 2.0 no solo implicó un cambio en la manera en la que los usuarios interactuaban con la información, sino también en las tecnologías utilizadas para hacerlo posible:

1. AJAX (Asynchronous JavaScript and XML)

AJAX fue una de las tecnologías clave que permitió la creación de aplicaciones web más dinámicas. Esta técnica de desarrollo permite actualizar partes de una página web sin tener que recargarla por completo, mejorando así la experiencia del usuario. Gracias a AJAX, plataformas como Google Maps o Gmail lograron ofrecer funcionalidades más fluidas e interactivas (O'Reilly, 2005).

2. APIs (Application Programming Interfaces)

El uso de APIs fue fundamental para la interoperabilidad entre plataformas, permitiendo que aplicaciones y sitios web se comunicaran entre sí. Esto facilitó la integración de diferentes servicios y permitió la creación de mashups, o combinaciones de contenidos y funciones de múltiples fuentes.

3. Diseño Centrado en el Usuario

El enfoque en el usuario fue otro aspecto central de la Web 2.0. El diseño web evolucionó para ser más intuitivo y atractivo, con interfaces que incentivaban la participación y facilitaban la navegación. Se priorizó la **usabilidad** y la accesibilidad, llevando al desarrollo de técnicas de diseño que se adaptaban a las necesidades del usuario (Backlinko, 2023).

Impacto y Críticas a la Web 2.0

La Web 2.0 tuvo un impacto profundo en la manera en la que nos comunicamos, trabajamos y entretenemos. Sin embargo, también ha enfrentado críticas:

1. **Desinformación y Noticias Falsas**: La facilidad para publicar contenido ha llevado a la propagación de información falsa. Las plataformas como Facebook y

Twitter han sido señaladas por permitir la difusión de noticias falsas, afectando la confianza en la información en línea.

2. **Monetización y Publicidad**: La necesidad de monetizar plataformas llevó a la implementación de publicidad dirigida, basada en datos de usuario. Esto generó preocupaciones sobre la **privacidad** y la recolección de datos personales.

3. **Fragmentación de la Información**: Aunque la Web 2.0 democratizó la publicación de contenidos, también condujo a la fragmentación de la información, dispersándola en múltiples fuentes y dificultando a veces la verificación de la calidad del contenido.

¿Qué sigue después de la Web 2.0?

El futuro de la web está en constante evolución. Después del auge de la Web 2.0, la atención se ha dirigido hacia la **Web 3.0** o Web Semántica, que busca una red más inteligente, capaz de entender el contexto de la información y relacionarla de manera autónoma. La evolución hacia tecnologías como la inteligencia artificial, el aprendizaje automático y la blockchain indica que la web seguirá transformándose, adaptándose a las necesidades cambiantes de sus usuarios (Backlinko, 2023).

1.4.7 Web Semántica y Web 2.0: La Nueva Web 3.0

Como se mencionó en los puntos anteriores, la evolución de la web ha pasado por varias fases, desde la **Web 1.0** estática y centrada en la información, hasta la **Web 2.0** dinámica e interactiva, que puso el poder en manos de los usuarios para crear y compartir contenido. El próximo gran paso es la **Web 3.0**, una fase que promete una web aún más avanzada, inteligente y

autónoma, capaz de contextualizar y procesar información de manera que imite la interpretación humana.

La Web Semántica: Contextualización y Comprensión Automática

La **Web Semántica**, un concepto propuesto por Tim Berners-Lee, busca dotar a la web de una capa de comprensión que permita a las máquinas entender el contexto de la información en la red. En lugar de ser una simple colección de páginas y enlaces, la Web Semántica está diseñada para que los datos en línea puedan ser interpretados automáticamente por algoritmos, facilitando la creación de una red de información más coherente y organizada. Esto se logra mediante el uso de **metadatos** y estructuras de datos que permiten a las máquinas entender la relación entre diferentes fragmentos de información (Berners-Lee, 2001).

El propósito de la Web Semántica es "eliminar la necesidad de la interpretación humana directa", permitiendo que las máquinas no solo recuperen datos, sino que los procesen en un contexto específico. Esto abre la puerta a nuevas aplicaciones, desde asistentes personales digitales más sofisticados hasta motores de búsqueda más precisos y eficientes. Sin embargo, el potencial de la Web Semántica todavía no ha sido completamente realizado, y muchos la consideran aún una "meta distante" debido a la complejidad técnica y a la cantidad de datos que deben ser categorizados adecuadamente (Hitzler, 2020).

La Interrelación entre la Web 2.0 y la Web Semántica

A pesar de que la Web Semántica y la **Web 2.0** son conceptos distintos, ambas están intrínsecamente relacionadas. La Web 2.0, con su énfasis en la interacción y la generación de contenido por parte de los usuarios, proporciona la materia prima necesaria para alimentar la Web Semántica. En otras palabras, "la Web 2.0 es un entorno fértil donde se generan datos

constantemente", lo que permite que las tecnologías semánticas analicen y comprendan mejor el contenido en línea (Hendler & Berners-Lee, 2010).

Para que la Web Semántica funcione, es crucial que exista una gran cantidad de datos estructurados que puedan ser interpretados por las máquinas. Aquí es donde la Web 2.0 entra en juego: al permitir que millones de usuarios contribuyan con contenido, etiqueten información y participen activamente en la web, se crea un entorno rico en datos. No obstante, esta dependencia en los usuarios también es una limitación, ya que la calidad de la información y su estructura depende del compromiso y la precisión de los creadores de contenido.

Web 3.0: Hacia una Web Autónoma y Basada en IA

El concepto de **Web 3.0** no tiene una definición única y consensuada, ya que abarca varias ideas emergentes que buscan llevar la web a un nuevo nivel. Sin embargo, una de las características más destacadas de la Web 3.0 es la integración de tecnologías de **inteligencia artificial** (IA) que permiten una web más autónoma e inteligente. Esto significa que "la Web 3.0 podría tener la capacidad de entender, aprender y tomar decisiones por sí misma", en lugar de depender exclusivamente de la aportación humana (Spivack, 2022).

En un mundo ideal de la Web 3.0, las máquinas no solo interpretarían datos según las reglas establecidas por los usuarios, sino que aprenderían de forma autónoma mediante algoritmos avanzados de IA. Esto permitiría que la web se adapte continuamente a las necesidades y preferencias individuales de los usuarios, proporcionando información más relevante y precisa. Por ejemplo, **Copilot**, un asistente de codificación desarrollado por OpenAI, utiliza la web como fuente de información para sugerir código en tiempo real, lo que demuestra un avance significativo en la capacidad de las máquinas para comprender contextos específicos.

Sin embargo, "a veces estos sistemas no pueden proporcionar respuestas totalmente concisas", lo que resalta las limitaciones actuales de la inteligencia artificial (OpenAI, 2023).

Retos y Oportunidades de la Web 3.0

Aunque la visión de una Web 3.0 plenamente funcional es atractiva, existen varios desafíos que deben superarse para que esta realidad se concrete. Algunos de los retos más destacados incluyen:

1. Complejidad Técnica

Implementar una web completamente semántica e inteligente requiere la creación y adopción de estándares globales que permitan estructurar los datos de manera uniforme. Sin una estructura de datos clara y consistente, la Web 3.0 corre el riesgo de fragmentarse, dificultando la interoperabilidad entre plataformas y servicios. Según "Hitzler (2020), aún estamos lejos de una adopción masiva de los estándares necesarios para la Web Semántica", lo que implica que este es un problema en curso que requiere cooperación global.

2. Dependencia del Aprendizaje Autónomo

La visión de una web autónoma basada en IA plantea preguntas sobre cómo se alimentará esta inteligencia artificial. Aunque se espera que las máquinas aprendan de forma independiente, aún requieren un punto de partida en forma de datos y conocimientos proporcionados por los usuarios o generados en la Web 2.0. Además, el aprendizaje autónomo plantea cuestiones sobre la precisión y la ética de las decisiones tomadas por las máquinas, ya que "los modelos de IA, como Copilot, pueden reproducir sesgos presentes en sus datos de entrenamiento" (Bender et al., 2021).

3. Privacidad y Seguridad

A medida que la web se vuelve más autónoma, surgen nuevas preocupaciones sobre la privacidad y la seguridad. La Web 3.0 promete personalización a un nivel sin precedentes, pero para lograr esto necesita recopilar y analizar grandes cantidades de datos personales. Esto plantea un dilema sobre la privacidad del usuario y la protección de la información sensible, algo que ha sido un problema persistente en la era de la Web 2.0. Según "un informe de la Electronic Frontier Foundation (EFF), las tecnologías emergentes deben priorizar la transparencia y la privacidad" para ganar la confianza del público (EFF, 2023).

La Web 3.0: ¿Una Realidad o una Utopía?

Aunque la idea de la Web 3.0 es prometedora, aún se debate si es una visión alcanzable o una utopía idealizada. La combinación de una Web Semántica completamente funcional con una IA autónoma que actúe de manera ética y precisa es un objetivo ambicioso. Sin embargo, es innegable que "el avance en tecnologías como la inteligencia artificial, el aprendizaje automático y la blockchain están acercando la realidad de una web más inteligente y eficiente" (Spivack, 2022).

La transición hacia la Web 3.0 no es un salto drástico, sino una evolución gradual basada en el progreso continuo de la Web 2.0 y la adopción de tecnologías semánticas. A medida que más plataformas adoptan estructuras de datos abiertas y accesibles, y se desarrollan sistemas de IA más avanzados, la idea de una Web 3.0 autónoma y comprensiva se vuelve cada vez más plausible.

1.5 Ontologías: Estructurando el Conocimiento

Ya que hemos establecido un punto, hacia adelante sobre una web inteligente, toca hablar de un punto esencial como son las ontologías y más preciso para el tema de esta tesis.

Pero primero, debemos entender este concepto tan importante.

1.5.1 ¿Qué son las Ontologías?

En términos generales, una ontología es una representación formal de un conjunto de conceptos dentro de un dominio y las relaciones entre ellos. En informática y en el contexto de la Web Semántica, las ontologías son utilizadas para estructurar y organizar información de una manera que pueda ser entendida por las máquinas. Según Gruber (1993), una ontología se define como "una especificación explícita de una conceptualización".

En el contexto de la Web Semántica, las ontologías permiten describir datos de forma estandarizada, lo que facilita su integración, interoperabilidad y reutilización. Esto se logra mediante la utilización de lenguajes como:

- **RDF (Resource Description Framework):** Para describir recursos y sus relaciones.
- **OWL (Web Ontology Language):** Para definir ontologías más complejas con semánticas enriquecidas.
- **SPARQL:** Para realizar consultas en datos descritos por ontologías.

1.5.2 Lenguajes y Herramientas para Ontologías: Profundización

La construcción y aplicación de ontologías son tareas fundamentales en la Web Semántica y la Web 3.0, ya que permiten estructurar el conocimiento y hacerlo accesible tanto para humanos como para máquinas. Para lograrlo, se emplean lenguajes específicos que permiten describir, modelar y gestionar las ontologías, así como herramientas que facilitan su creación,

mantenimiento y utilización en aplicaciones prácticas. A continuación, se presenta un análisis detallado de los lenguajes RDF, OWL, SHACL y de los razonadores semánticos.

RDF (Resource Description Framework): La Base de la Web Semántica

El RDF es el pilar fundamental de la Web Semántica, diseñado por el World Wide Web Consortium (W3C) para representar información estructurada en la web. Su propósito principal es describir recursos y sus relaciones mediante un modelo de datos basado en triples, que constan de un sujeto, un predicado y un objeto. Por ejemplo:

- Sujeto: "Juan Pérez"
- Predicado: "es autor de"
- Objeto: "El Quijote"

Este modelo es intuitivo y flexible, permitiendo representar casi cualquier tipo de información de manera sencilla.

Características clave de RDF:

1. **Universalidad:** Permite describir datos de cualquier dominio.
2. **Extensibilidad:** Los triples se pueden combinar fácilmente con otros conjuntos de datos RDF.
3. **Interoperabilidad:** Al ser un estándar abierto, permite que diferentes sistemas compartan y entiendan información.

Serializaciones de RDF:

RDF admite varias formas de serialización, que son maneras de expresar los datos:

- **RDF/XML:** Representación en XML, ideal para interoperabilidad.
- **Turtle:** Sintaxis más legible para humanos, popular en desarrollo.

- **JSON-LD:** Versión en JSON, ampliamente utilizada en aplicaciones web.

OWL (Web Ontology Language): Expresividad y Restricciones Avanzadas

OWL es una extensión de RDF que añade mayor expresividad al permitir la definición de conceptos, relaciones y restricciones más complejas. Esto lo convierte en una herramienta poderosa para modelar dominios ricos en detalles.

Diferencias entre RDF y OWL:

- RDF describe recursos y relaciones, pero no define lógica ni reglas.
- OWL introduce lógica descriptiva, lo que permite especificar propiedades como equivalencias, disyunciones y restricciones de cardinalidad.

Características clave de OWL:

1. **Restricciones de cardinalidad:** Definen cuántas veces una propiedad puede aplicarse a un sujeto. Ejemplo: "Un triángulo tiene exactamente 3 lados".
2. **Clases equivalentes:** Permiten identificar que dos clases son conceptualmente idénticas.
3. **Propiedades transitivas y simétricas:** Ejemplo: "Si A es mayor que B, y B es mayor que C, entonces A es mayor que C".

Niveles de OWL:

OWL se presenta en diferentes perfiles para adaptarse a necesidades específicas:

- **OWL Lite:** Simplificado para aplicaciones con menor complejidad.
- **OWL DL:** Compatible con la lógica descriptiva completa.
- **OWL Full:** Ofrece máxima flexibilidad, pero a costa de menor decidibilidad.

Casos de uso de OWL:

Un ejemplo práctico es la biomedicina, donde OWL permite modelar jerarquías de enfermedades y sus relaciones con síntomas y tratamientos.

SHACL (Shapes Constraint Language): Validación de Datos Basada en Ontologías

SHACL es un lenguaje estándar del W3C diseñado para definir restricciones y validar datos en función de las ontologías. Este lenguaje es esencial para garantizar la calidad y coherencia de los datos en sistemas basados en la Web Semántica.

Características principales de SHACL:

1. **Validación flexible:** Permite verificar si los datos cumplen con ciertas condiciones especificadas en la ontología.
2. **Compatibilidad con RDF:** Funciona sobre modelos RDF, validando triples de datos.
3. **Definición de formas (shapes):** Especifica las propiedades que un recurso debe tener y sus valores aceptables.

Ejemplo de SHACL:

En un sistema de gestión de transporte, se podría validar que cada vehículo tiene una "capacidad de pasajeros" que sea un número entero mayor a cero.

Herramientas Avanzadas: Protégé y Razonadores Semánticos

Además de los lenguajes, existen herramientas y tecnologías que facilitan la creación y aplicación de ontologías:

1. **Protégé:**

Protégé es un entorno de desarrollo gratuito y de código abierto ampliamente utilizado para construir ontologías. Ofrece:

- Una interfaz gráfica intuitiva para modelar clases, propiedades y relaciones.
- Compatibilidad con RDF, OWL y otros lenguajes semánticos.
- Extensibilidad mediante plugins que permiten incorporar nuevas funcionalidades.

2. **Razonadores Semánticos:**

Los razonadores son motores que procesan las ontologías y generan inferencias automáticas basadas en la lógica descriptiva. Algunos de los razonadores más populares son:

- **Pellet:** Especializado en OWL DL, conocido por su rendimiento y precisión.
- **Hermit:** Ofrece inferencias en tiempo real y soporte para OWL 2.
- **Fact++:** Ideal para grandes volúmenes de datos gracias a su diseño eficiente.

Aplicaciones de los Razonadores:

En diagnósticos médicos, los razonadores pueden analizar una base de datos de síntomas y relaciones para sugerir posibles enfermedades y tratamientos.

Integración de Lenguajes y Herramientas en Aplicaciones Reales

La combinación de lenguajes como RDF, OWL y SHACL con herramientas como Protégé y razonadores permite construir sistemas complejos y robustos. Por ejemplo, un sistema

de comercio electrónico puede utilizar una ontología para estructurar productos, relacionar categorías y validar descripciones, ofreciendo búsquedas más precisas y recomendaciones inteligentes.

Capítulo 2: Fundamentos de la Arquitectura SOA

La arquitectura SOA se basa en la construcción de **servicios independientes**, es decir, módulos autónomos que pueden ser invocados para realizar tareas específicas, como recuperar información de una base de datos, procesar pagos, autenticar usuarios o enviar notificaciones. Estos servicios interactúan a través de protocolos estandarizados, como **HTTP, SOAP, REST o gRPC**, permitiendo la comunicación entre sistemas heterogéneos.

Uno de los principios fundamentales de SOA es el **desacoplamiento**, lo que significa que cada servicio opera de manera autónoma sin depender directamente de otros. Esta independencia facilita la **reutilización de componentes**, ya que los servicios pueden ser utilizados en múltiples aplicaciones sin modificaciones significativas. Además, SOA promueve la **interoperabilidad**, permitiendo que sistemas escritos en distintos lenguajes de programación puedan comunicarse sin problemas.

Otro concepto clave en SOA es la **orquestación de servicios**, donde múltiples servicios se combinan para ejecutar procesos más complejos. Esta característica es ampliamente utilizada en entornos empresariales para optimizar flujos de trabajo y mejorar la eficiencia operativa.

2.1 Beneficios de la Arquitectura SOA

La adopción de SOA en el desarrollo de software conlleva numerosas ventajas, entre las que destacan:

1. **Escalabilidad y modularidad**: La posibilidad de añadir nuevos servicios sin afectar la estructura general del sistema permite que las aplicaciones crezcan de manera eficiente.
2. **Interoperabilidad entre plataformas**: SOA facilita la integración de sistemas desarrollados en diferentes lenguajes y tecnologías, favoreciendo la colaboración entre diversas aplicaciones.
3. **Mantenimiento simplificado**: Al estar compuestos por módulos independientes, los sistemas basados en SOA pueden actualizarse sin interrumpir el funcionamiento de otros componentes.
4. **Optimización de recursos**: La reutilización de servicios reduce costos de desarrollo y mejora la eficiencia en la gestión de sistemas.
5. **Automatización de procesos empresariales**: La orquestación de servicios permite diseñar flujos de trabajo automatizados, mejorando la productividad en entornos empresariales.

Desafíos en la Implementación de SOA

A pesar de sus beneficios, la implementación de SOA conlleva desafíos que deben ser abordados para garantizar su éxito:

- **Complejidad arquitectónica**: Diseñar una arquitectura basada en servicios requiere una planificación cuidadosa para definir correctamente la interacción entre componentes.
- **Gestión de seguridad**: La exposición de servicios a diferentes clientes incrementa los riesgos de seguridad, lo que obliga a implementar mecanismos como **OAuth 2.0, JWT o certificados SSL**.
- **Sobrecarga en la comunicación**: La dependencia de protocolos como HTTP puede generar latencias si no se optimizan adecuadamente las interacciones entre servicios.
- **Gobernanza y monitoreo**: Es crucial establecer políticas para gestionar versiones, monitorear servicios y garantizar su disponibilidad en todo momento.

2.2 SOA y su Relación con la Web Semántica

La Web Semántica, propuesta por Tim Berners-Lee, busca mejorar la capacidad de las máquinas para comprender y procesar la información en la web mediante datos estructurados y relaciones semánticas. Esta visión está estrechamente relacionada con SOA, ya que ambos enfoques buscan la interoperabilidad y el acceso eficiente a la información.

En este contexto, los **Servicios Web Semánticos (SWS, Semantic Web Services)** representan la convergencia de SOA con la Web Semántica. Los SWS utilizan ontologías y lenguajes como **OWL-S (Ontology Web Language for Services)** para describir servicios de manera semántica, facilitando su descubrimiento, integración y composición automática.

Algunas formas en las que SOA y la Web Semántica se complementan incluyen:

1. **Descubrimiento automático de servicios**: Gracias a las ontologías, los servicios pueden describirse con metadatos semánticos, permitiendo que las aplicaciones los encuentren y utilicen de manera inteligente.
2. **Interoperabilidad mejorada**: Los datos estructurados permiten que diferentes sistemas comprendan la información de manera coherente, independientemente de la plataforma en la que operen.
3. **Razonamiento semántico**: Los razonadores semánticos pueden inferir relaciones entre datos y mejorar la toma de decisiones en entornos empresariales.
4. **Automatización en la composición de servicios**: La inteligencia semántica facilita la combinación dinámica de servicios para ejecutar procesos complejos de manera autónoma.

Un ejemplo práctico de esta integración se encuentra en los **chatbots y asistentes virtuales**, que combinan SOA con la Web Semántica para acceder a múltiples servicios y proporcionar respuestas basadas en conocimiento estructurado.

2.3 SOA y su Evolución hacia los Microservicios

En los últimos años, los **microservicios** han emergido como una evolución de SOA, enfocándose en la creación de servicios aún más pequeños y especializados. A diferencia de SOA, que suele emplear un **Bus de Servicios Empresarial (ESB)** para gestionar la comunicación entre servicios, los microservicios utilizan APIs REST y sistemas de mensajería asíncrona como **Kafka o RabbitMQ**.

Mientras que SOA busca la integración de sistemas empresariales a gran escala, los microservicios están diseñados para ofrecer mayor independencia en el despliegue y la escalabilidad, lo que los hace ideales para aplicaciones en la nube y entornos de desarrollo ágil.

Capítulo 3: Propuesta de la plataforma educativa

Después de un análisis y hablar un poco de las bases de interés en capítulos anteriores, nos abordaremos al planteamiento de un sistema que implemente las ideas plasmadas en esta tesis.

BUAPP

Análisis previo al desarrollo de BUAPP

El análisis previo al desarrollo de BUAPP contempla un estudio detallado de las herramientas utilizadas previamente por un grupo de estudiantes de la Facultad de Ciencias de la Computación. En este contexto, los profesores empleaban diversas plataformas para impartir cursos, gestionar actividades y facilitar la comunicación entre docentes y alumnos. Entre las plataformas analizadas se encuentran Microsoft Teams, Google Classroom, Google Meet y Blogger. A partir de este análisis, se extrajeron las siguientes conclusiones.

Google Classroom: Sencillez y estructura organizada

Uno de los aspectos más destacables de Google Classroom es su simplicidad y facilidad de uso. La creación de un curso es un proceso intuitivo y accesible para cualquier usuario con una cuenta de Google. No obstante, en el diseño de BUAPP no se adoptará este enfoque, ya que se considera fundamental establecer roles diferenciados para alumnos y profesores. En

consecuencia, solo los docentes tendrán la capacidad de crear cursos, asegurando una estructura organizativa más clara y alineada con los entornos educativos formales.

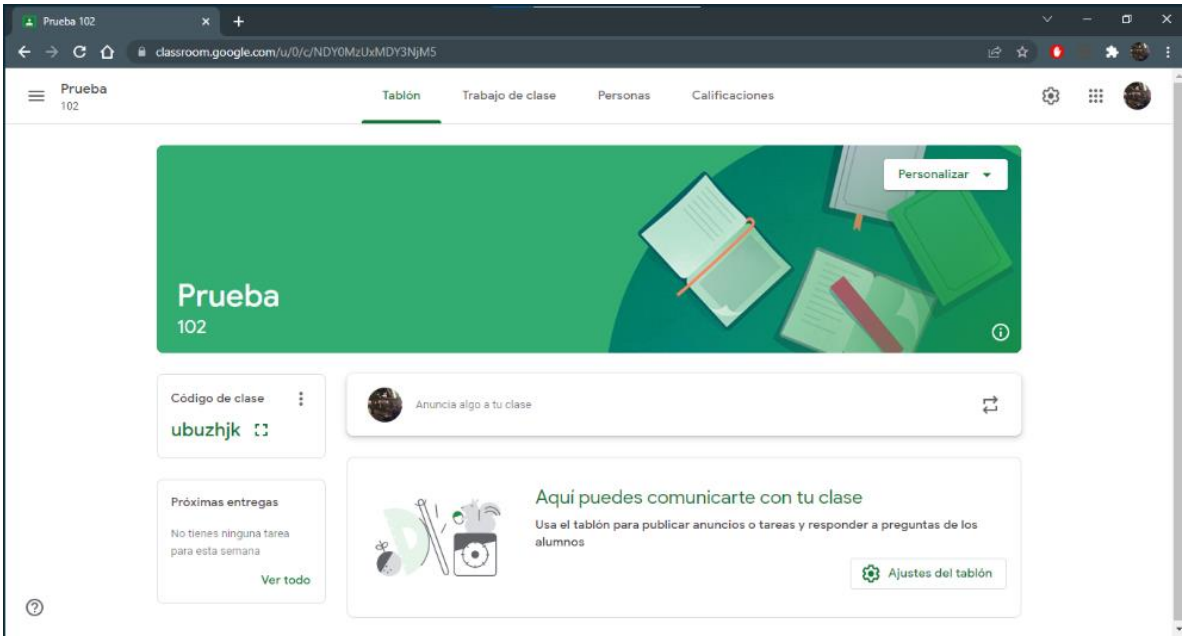


Figura 3.1 Ventana de un curso de prueba creado en Google Classroom

En cuanto a la gestión de anuncios y actividades, Classroom ofrece una interfaz eficiente que permite a los docentes publicar contenido en el tablero principal. Los anuncios pueden incluir documentos de repaso y materiales complementarios, facilitando su consulta por parte de los alumnos. Las actividades (exámenes, tareas, cuestionarios, etc.) pueden programarse con fechas de vencimiento, lo que garantiza su visibilidad hasta la fecha límite establecida. Además, los docentes pueden calificar y proporcionar retroalimentación sobre las entregas, mientras que los alumnos pueden visualizar sus calificaciones y comentarios de manera clara y organizada. Este modelo de gestión de contenido es altamente eficiente y facilita el seguimiento académico tanto para estudiantes como para profesores.

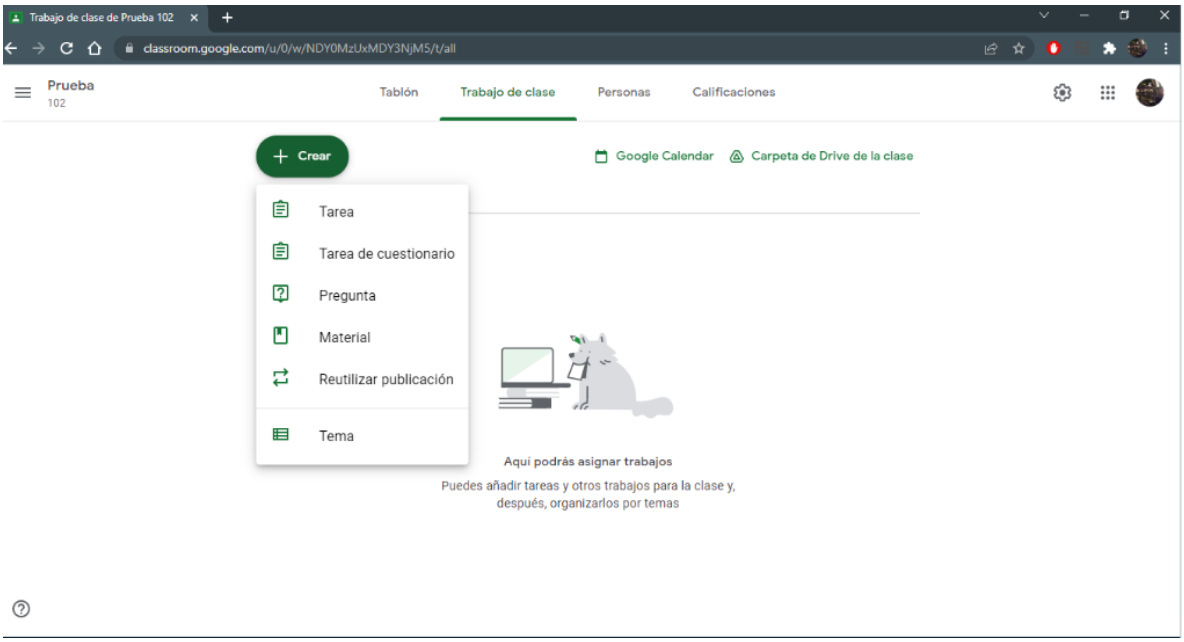


Figura 3.2 Ventana de creación de actividad en Google Classroom

Microsoft Teams: Colaboración y flexibilidad

Microsoft Teams, por su parte, está diseñado principalmente para entornos organizacionales y corporativos, lo que se refleja en su estructura y funcionalidades. A diferencia de Classroom, Teams no establece una diferenciación clara entre alumnos y profesores, permitiendo que cualquier usuario cree un canal de comunicación. Aunque esta flexibilidad puede ser útil en ciertos escenarios, también puede generar desorden y falta de estructura en entornos educativos formales.

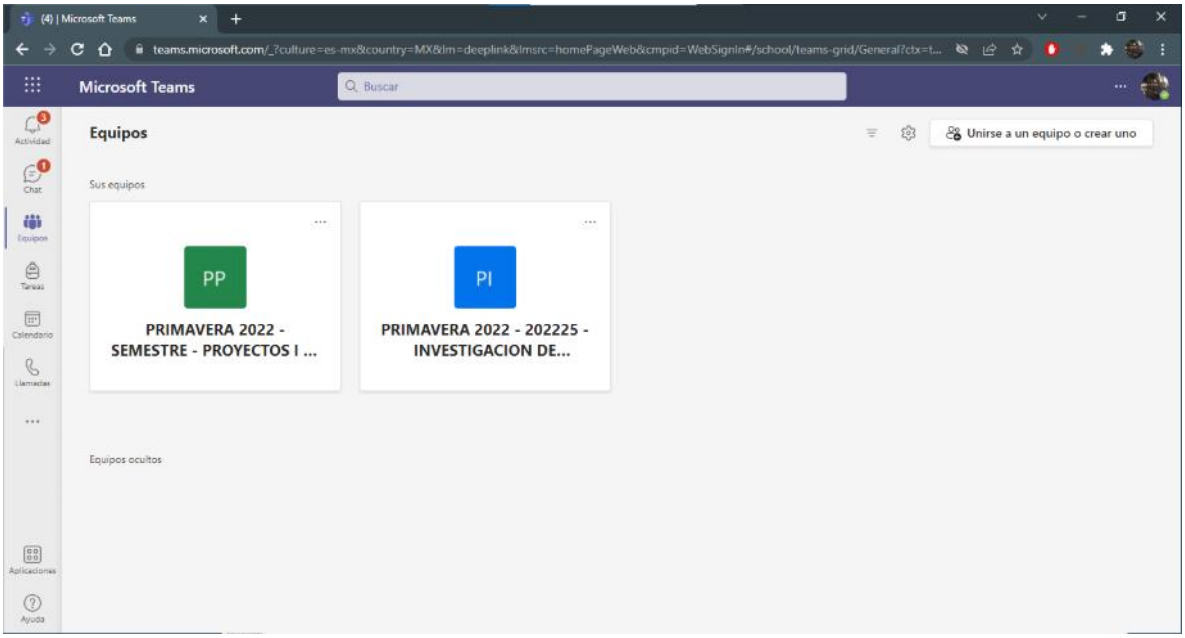


Figura 3.3 Ventana de cursos de Microsoft Teams

Una de las principales diferencias con Classroom radica en la gestión de tareas. Aunque Teams cuenta con un apartado para la asignación de tareas, este no posee la misma accesibilidad ni visibilidad que ofrece Classroom. En lugar de una interfaz específica para pendientes académicos, las actividades pueden perderse dentro del flujo de mensajes y publicaciones dentro de los canales, lo que dificulta su seguimiento por parte de los estudiantes.

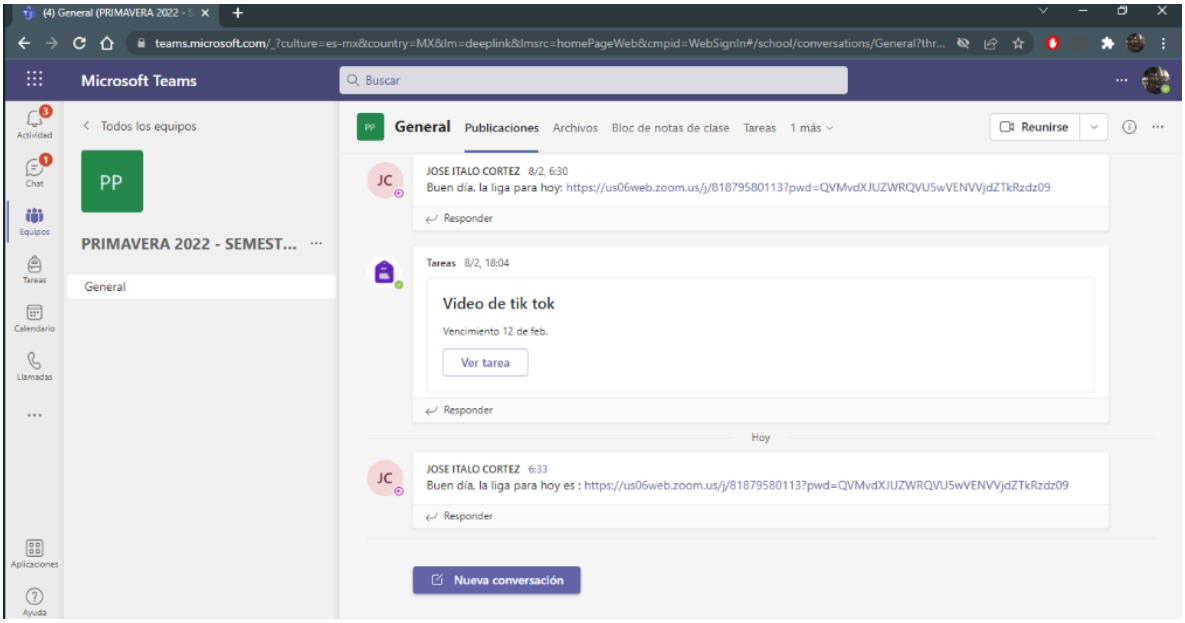


Figura 3.4 Ventana de historial del curso

Sin embargo, Teams sobresale en otros aspectos, como su capacidad para gestionar archivos de manera eficiente. Los usuarios pueden acceder a un repositorio de documentos dentro de cada canal, lo que facilita la localización de materiales sin necesidad de recorrer el historial completo de publicaciones. Además, Teams incorpora una funcionalidad de mensajería instantánea, permitiendo una comunicación fluida entre alumnos y docentes.

Uno de los mayores beneficios de Teams es su potente sistema de videoconferencias, que permite a los profesores impartir clases en vivo, con la posibilidad de grabar las sesiones. Esta función resulta especialmente valiosa para los estudiantes, ya que les permite revisar las clases grabadas y reforzar su aprendizaje de manera autónoma. Asimismo, las videoconferencias ofrecen un espacio interactivo donde los docentes pueden enriquecer sus explicaciones a través de herramientas colaborativas, mejorando la calidad de la enseñanza.

Conclusiones y aportes al desarrollo de BUAPP

A partir del análisis de estas plataformas, se han identificado aspectos clave que podrían incorporarse en el desarrollo de BUAPP:

- Estructura organizativa clara: A diferencia de Teams, BUAPP adoptará un sistema de roles diferenciados para alumnos y profesores, asegurando una gestión estructurada de los cursos.
- Interfaz intuitiva para tareas y actividades: Se buscará replicar la accesibilidad y organización de Classroom en cuanto a la programación y visualización de actividades académicas.
- Repositorio de archivos: Inspirado en Teams, BUAPP integrará un sistema eficiente para el almacenamiento y búsqueda de documentos.
- Mensajería instantánea: Se incluirá un sistema de comunicación directa entre alumnos y docentes para facilitar la resolución de dudas y el intercambio de información.
- Videoconferencias y clases grabadas: Se analizará la implementación de una herramienta similar a la de Teams para la realización de clases en vivo y la grabación de sesiones, brindando a los estudiantes una mayor flexibilidad en su aprendizaje.

En síntesis, BUAPP buscará combinar las mejores características de estas plataformas, adaptándolas a las necesidades específicas del entorno académico en el que se implementará. Este análisis previo resulta fundamental para el diseño de una plataforma robusta, eficiente y alineada con las exigencias actuales de la educación digital.

3.1 Primeros Pasos

Una vez definido el punto de partida y la estructura general del proyecto, es momento de abordar con mayor detalle la arquitectura del sistema, empezando por la estructura de base de datos y los componentes esenciales que permitirán el funcionamiento adecuado de una plataforma educativa en línea.

3.1.1 Base de datos

Con el objetivo de garantizar una estructura sólida, eficiente y estandarizada en la gestión de cursos virtuales, se propone el diseño de un esquema de base de datos relacional que permita almacenar, organizar y manipular adecuadamente la información de los diferentes actores del sistema: usuarios, profesores, alumnos, cursos y tareas, entre otros elementos clave. Esta estructura facilitará la administración de inscripciones, la carga y descarga de materiales, y el seguimiento de actividades académicas.

3.1.2 Tablas principales

Usuario

Esta tabla centraliza la información general de todos los usuarios registrados en la plataforma. Entre los datos almacenados se incluyen: nombre, apellidos, correo electrónico, contraseña cifrada y, opcionalmente, una imagen de perfil.

Profesor

Contiene la información específica de los usuarios con rol de profesor, incluyendo su matrícula profesional, especialidad y una relación con la tabla Usuario para heredar los datos generales.

Alumno

Registra los datos correspondientes a los estudiantes, incluyendo su matrícula, nivel académico y su referencia en la tabla Usuario.

Curso

Esta tabla almacena los cursos disponibles dentro del sistema. Cada curso está identificado por un código NRC único, y cuenta con atributos como nombre, descripción y una relación directa con el profesor responsable del mismo.

Relaciones y asignaciones

CursoAsignado

Establece la relación entre los alumnos y los cursos en los que están inscritos. Cada registro vincula una matrícula de alumno con el NRC de un curso específico, permitiendo rastrear las inscripciones activas.

Tarea

Registra las tareas o actividades asignadas dentro de cada curso. Incluye campos como identificador, título, descripción, fecha de entrega y el curso al que pertenece.

ArchivosCargados

Contiene el contenido y metadatos de los archivos subidos por los alumnos. Está vinculado con ArchivosCargados y permite un almacenamiento organizado de las entregas.

ArchivoCargado

Guarda el contenido de los archivos subidos por los alumnos. Está relacionado con la tabla de archivos cargados.

ArchivosGuías

Relaciona los archivos guía con las tareas, permitiendo que los alumnos accedan a material de referencia.

ArchivoGuía

Guarda los archivos de apoyo que los profesores suben como guía para las tareas.

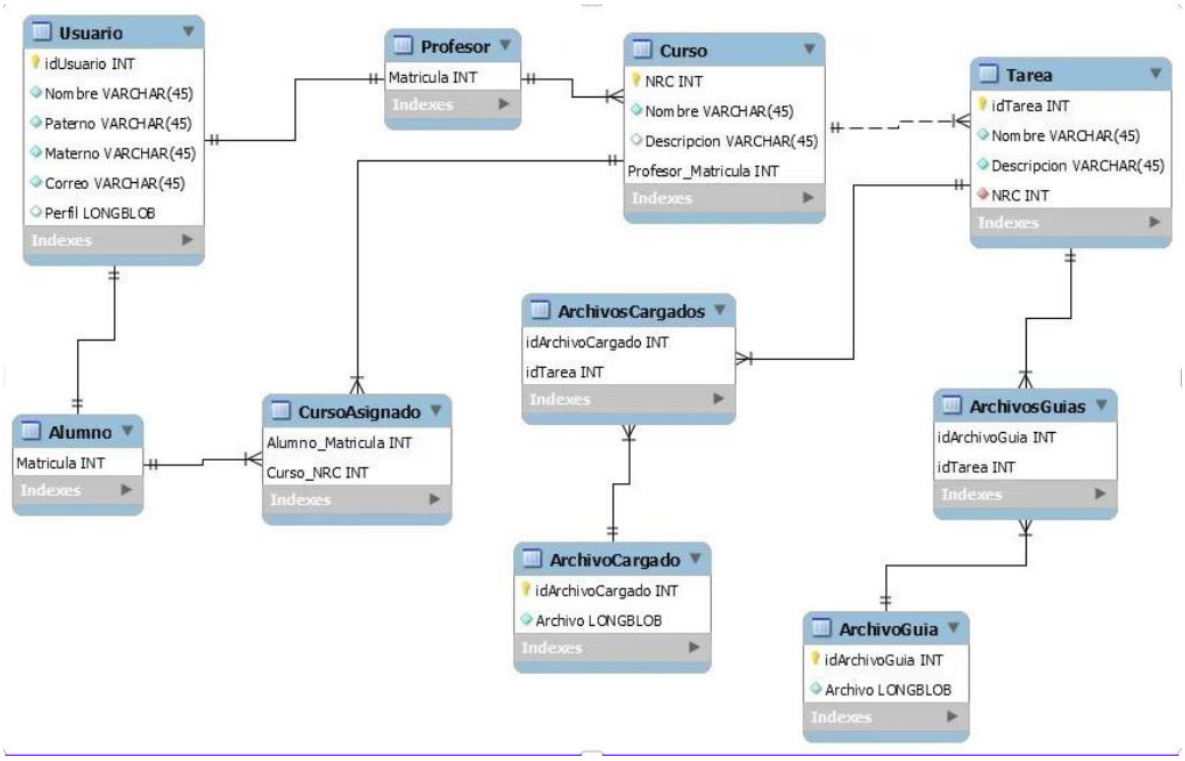


Figura 3.5 Diagrama de base de datos

3.1.3 Casos de Uso

Después del análisis estructural de la base de datos, se desarrollaron los diagramas de casos de uso básicos para explicar las funcionalidades principales del sistema y los diferentes tipos de usuarios involucrados.

Tipos de Usuarios:

- **Estudiante**: Puede autenticarse en el sistema, consultar sus cursos asignados, realizar actividades académicas y subir material en respuesta a tareas.
- **Profesor**: Tras autenticarse, tiene la capacidad de crear cursos, asignar actividades, subir materiales y consultar las entregas de los estudiantes.

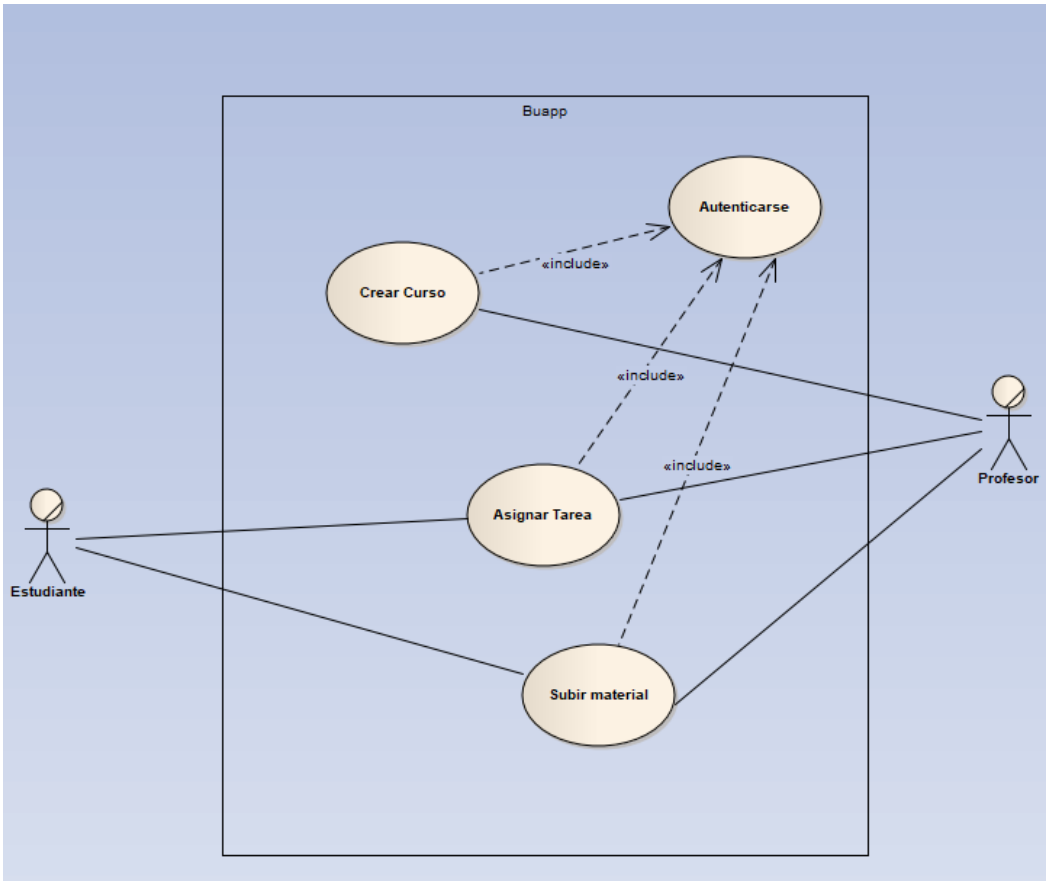


Figura 3.6 Diagrama de casos de uso

3.1.4 Planeación Diagramas de secuencia

Con base en los casos de uso definidos, se procedió al diseño de los diagramas de secuencia, que permiten visualizar de forma lógica y estructurada el flujo de información entre los distintos componentes del sistema. A continuación, se describen algunos de los procesos clave:

3.1.4.1 Autenticación

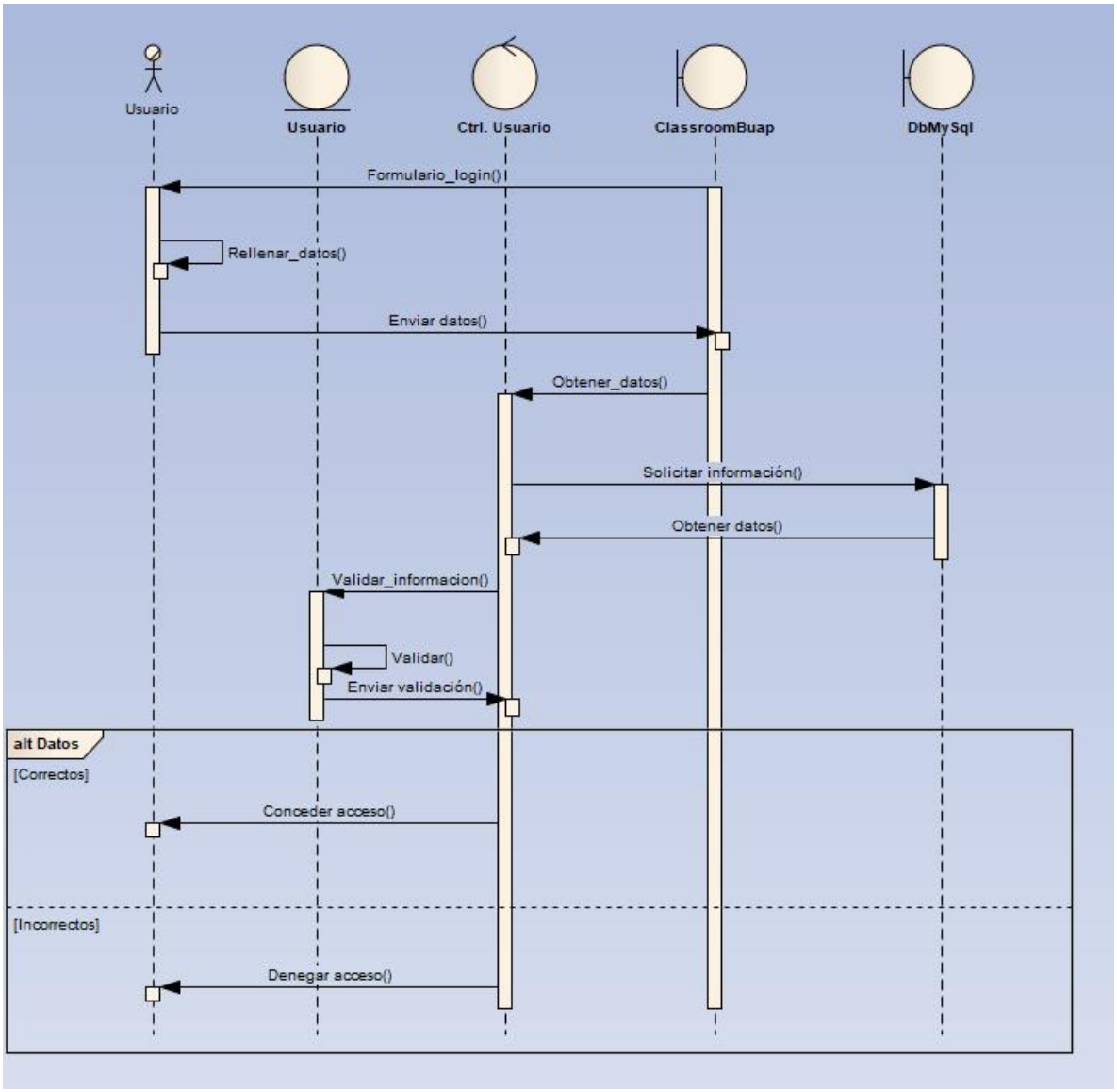


Figura 3.7 Diagrama de secuencia "Autenticación"

En el diagrama podemos observar como el usuario interactúa con la aplicación, en este caso, con la pantalla principal en el cual se va a mostrar el formulario del login, en el cual el usuario ingresa sus credenciales de acceso, se manda esta información al control de usuarios, este busca en la base con los datos correspondientes, obtenemos el usuario, se manda al control y validamos si la información es correcta, si es válido el usuario se concede el acceso, en caso contrario simplemente se muestra el mensaje de error en las credenciales

3.1.4.2 Creación de un curso

Como se ha mencionado, parte importante es poder estructurar los cursos, por ello, se plasmó el poder crear un curso en el sistema de la siguiente manera

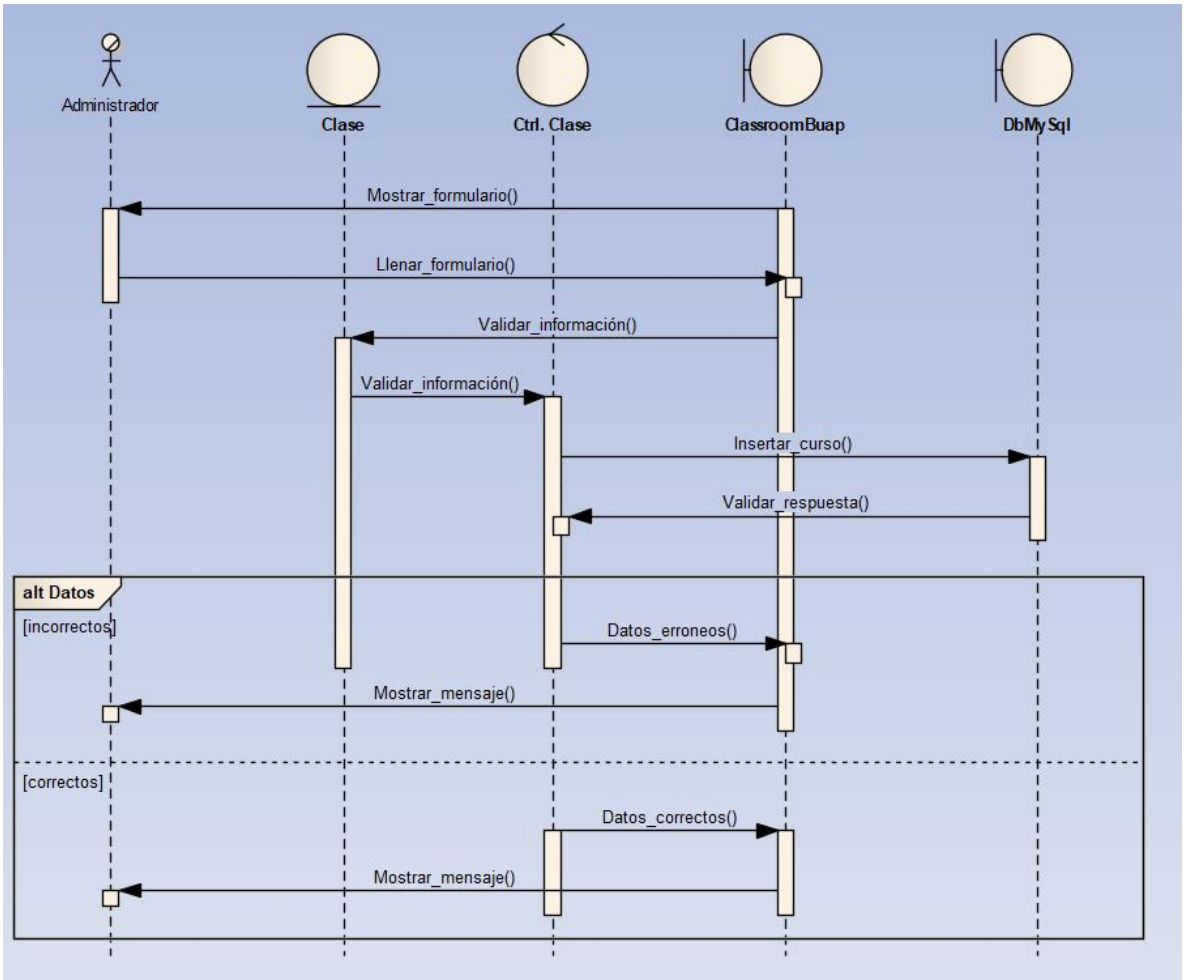


Figura 3.8 Diagrama de secuencia "Creación de un curso"

Nuestro flujo inicia con el sistema mostrando el formulario básico para la creación del curso al profesor, una vez que el profesor concluye con el llenado de dicho formulario, se crea la entidad para su validación, que toda la información ese correcta, se almacena la información en la base y se muestran dos posibles escenarios, uno donde la información sea incorrecta o faltante,

se le avisa al profesor para solicitarle la corrección, o el caso en el que todo este correcto, el curso finalmente será creado y estará listo para la inscripción de los alumnos a dicho curso.

3.1.4.3 Asignación de trabajos

Otra parte importante del sistema consiste en la posibilidad de que el profesor asigne las actividades y esta engloba ya sea tareas, proyectos, exámenes, etc. Ya que para el sistema el comportamiento sería el mismo, la idea subyacente del comportamiento sería un estándar para el manejo de cualquier actividad, por lo tanto, se presenta el siguiente diagrama.

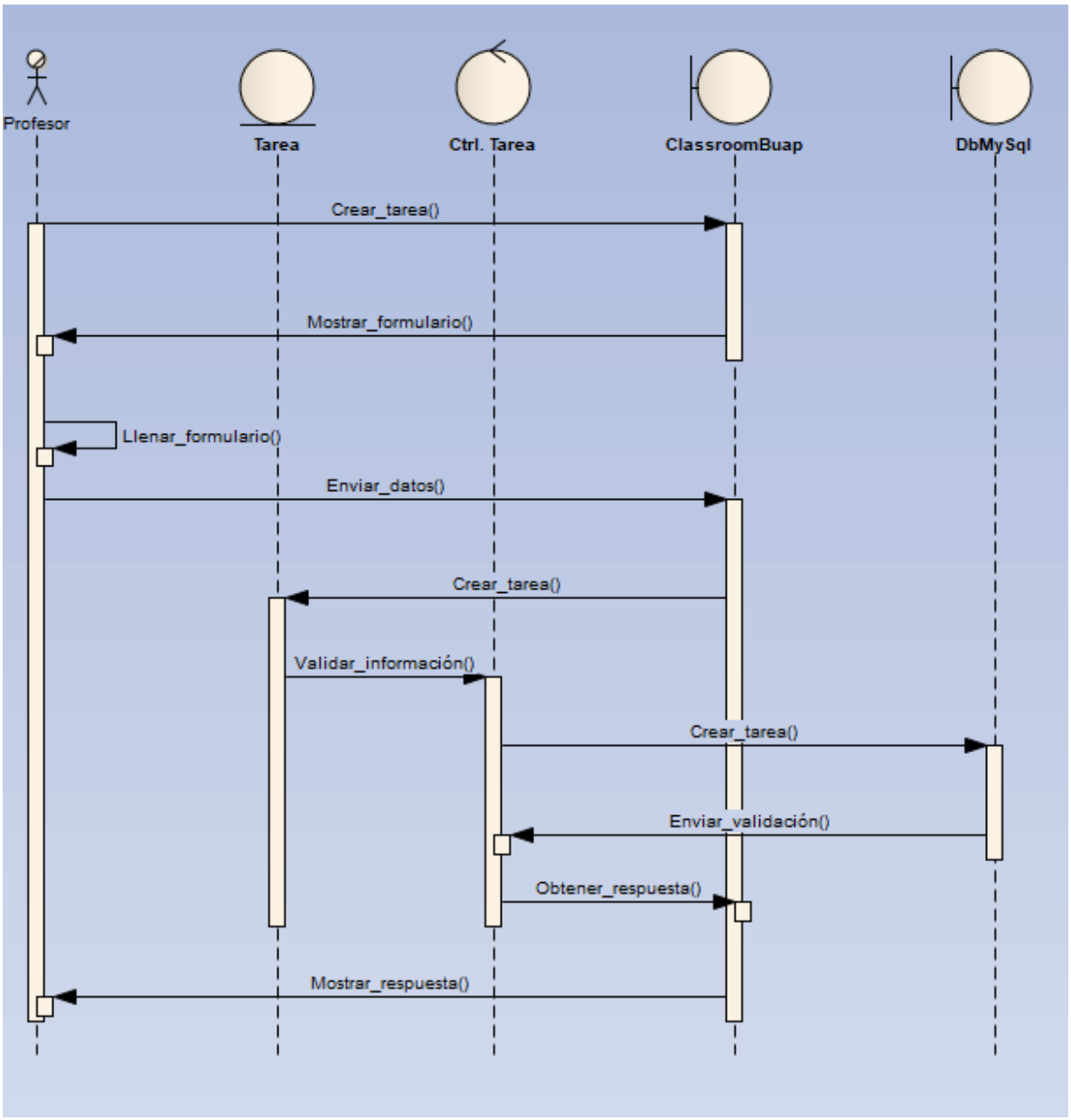


Figura 3.9 Diagrama de secuencia "Asignación de trabajos"

Como se comentó anteriormente, parte desde la instrucción del profesor de asignar alguna actividad, el sistema mostrará un formulario para los datos de la actividad, se enviarán los datos al sistema y posteriormente se hacen las validaciones e inserciones a los alumnos, por último, se le muestra el mensaje a profesor donde se confirma la creación de la actividad.

3.1.4.4 Carga de material

Una vez contemplado los puntos anteriores es necesario crear la funcionalidad para poder subir archivos, al igual que el anterior no hay distinción, puede ser la posibilidad de subir archivos, libros, actividades, etc. Se contempla el mismo proceso para poder agilizar el desarrollo de la aplicación, la única diferencia es saber el autor, es decir, si el material es de un profesor o un alumno, por lo tanto, se desarrolló el siguiente diagrama.

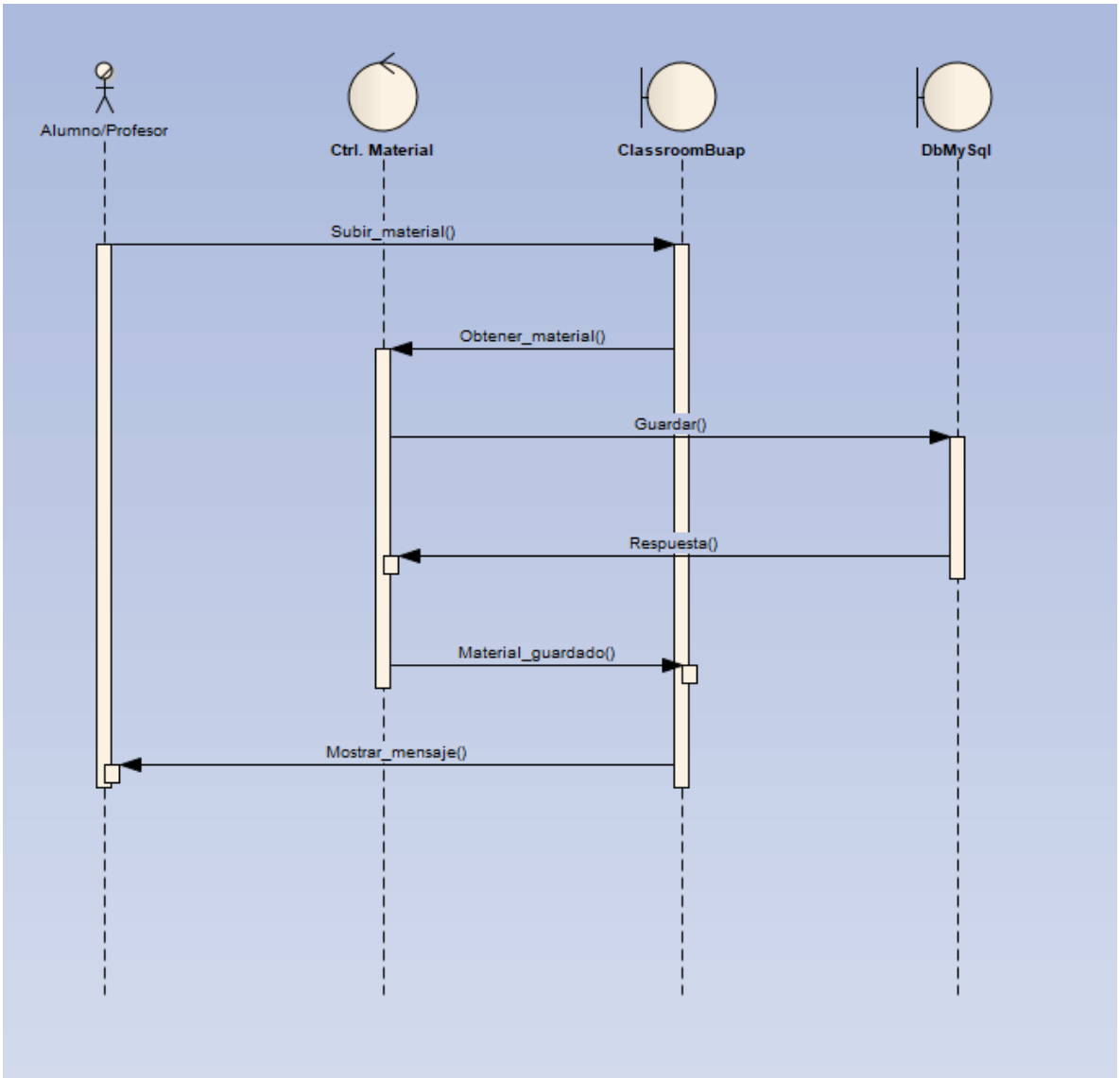


Figura 3.10 Diagrama de secuencia "carga de material"

En el cual es la simple interacción entre el alumno o profesor que hacen uso del sistema, por ende, solo se hacen el flujo entre el sistema y la base que almacena la información en el sistema.

Capítulo 4: Implementación de la plataforma educativa

En este capítulo, hablaremos un poco de los desafíos que nos encontramos al aterrizar todo el análisis previo, si bien, las bases ya fueron cubiertas, es necesario, plasmar todo de la manera más eficiente posible.

Cubriendo las necesidades básicas para la implementación del proyecto se elaboró el proyecto, culminando de esta manera lo planteado y analizado en capítulos anteriores.

4.1 Herramientas de desarrollo

Como se mencionó anteriormente, ya tenemos las bases para comenzar a plasmar el análisis en una aplicación web, pero, no sin antes poder hablar de las herramientas utilizadas.

Se tomó la decisión de trabajar con los enfoques estándar de dividir la aplicación en dos partes, el Front-end y Back-end.

4.1.1 Angular como Herramienta Estratégica para el Desarrollo Front-End

En el dinámico entorno del desarrollo web moderno, la elección de una herramienta adecuada para la construcción del front-end de una aplicación no solo determina la eficiencia del desarrollo, sino también la escalabilidad, mantenibilidad y calidad de la experiencia del usuario. Entre las diversas opciones disponibles, **Angular** se presenta como una solución robusta, integral y ampliamente adoptada por empresas y desarrolladores de todo el mundo. Diseñado y mantenido por Google, Angular ha evolucionado hasta convertirse en un framework completo

que facilita la creación de aplicaciones web modernas, modulares y de alto rendimiento (Google, 2024).

Una de las principales razones para adoptar Angular radica en su **arquitectura basada en componentes**. Este enfoque promueve una organización clara y reutilizable del código, permitiendo a los desarrolladores dividir la interfaz en bloques funcionales independientes. Esta estructura facilita tanto el mantenimiento como la escalabilidad de los proyectos, ya que cada componente puede desarrollarse, probarse y modificarse de forma aislada (Freeman, 2022).

Además, Angular ofrece un **enfoque integral** que cubre múltiples aspectos del desarrollo front-end. Mientras que otras bibliotecas como React se centran únicamente en la interfaz, Angular proporciona herramientas y soluciones para enrutamiento, gestión de formularios, comunicación HTTP, validaciones, animaciones, pruebas, y más, todo dentro del mismo ecosistema (Gackenheimer, 2017). Esta cohesión reduce la necesidad de integrar librerías externas, minimizando los conflictos y la curva de aprendizaje al trabajar con un único marco conceptual.

Otro aspecto destacado es su uso de **TypeScript**, un superconjunto tipado de JavaScript que aporta mayor seguridad y claridad al desarrollo. TypeScript permite detectar errores en tiempo de compilación, facilita el autocompletado en los editores modernos y mejora la experiencia del desarrollador mediante una codificación más estructurada (Microsoft, 2024). Esto es especialmente beneficioso en proyectos a gran escala o en equipos de desarrollo numerosos, donde la consistencia y la legibilidad del código son fundamentales (Flanagan, 2020).

Angular también pone énfasis en las buenas prácticas de desarrollo a través de su **CLI (Command Line Interface)**, una poderosa herramienta que automatiza tareas comunes como la

generación de componentes, servicios, módulos y pruebas. Esto no solo acelera el desarrollo, sino que asegura que el código siga convenciones estandarizadas y organizadas desde el principio (Freeman, 2022).

Desde el punto de vista de rendimiento, Angular implementa técnicas como el **renderizado anticipado (ahead-of-time compilation)** y la **detección eficiente de cambios**, lo que resulta en aplicaciones más rápidas y con mejor experiencia para el usuario final (Google, 2024). Combinado con el soporte para aplicaciones progresivas (PWA), Angular permite construir aplicaciones web con capacidades cercanas a las de una aplicación nativa.

Por último, la **amplia comunidad y respaldo corporativo** de Angular garantiza una evolución constante del framework, acceso a una gran cantidad de recursos educativos y soporte técnico sólido. Esta comunidad activa también fomenta el desarrollo de bibliotecas y extensiones que complementan sus funcionalidades (Freeman, 2022).

En conclusión, Angular se posiciona como una opción sólida para el desarrollo front-end por su enfoque integral, arquitectura modular, soporte a herramientas modernas como TypeScript y su compromiso con las buenas prácticas de desarrollo. Para equipos que buscan construir aplicaciones escalables, mantenibles y con alto estándar de calidad, Angular representa una inversión tecnológica estratégica y confiable.

4.1.2 Beneficios de usar Spring Boot en el desarrollo de backend

En el contexto del desarrollo de aplicaciones modernas, no solo es fundamental una interfaz atractiva y eficiente, sino también una arquitectura de backend sólida, escalable y segura. **Spring Boot**, un proyecto de la plataforma Spring, se ha consolidado como una de las herramientas más utilizadas en el desarrollo backend para aplicaciones empresariales gracias a su simplicidad, flexibilidad y capacidad de integración.

Spring Boot permite a los desarrolladores construir aplicaciones Java basadas en microservicios de forma rápida, minimizando la configuración manual y reduciendo la curva de entrada. A diferencia de configuraciones tradicionales de Spring, Spring Boot elimina la necesidad de definir beans explícitos, archivos XML o servidores externos, gracias a su enfoque de 'convención sobre configuración' y su servidor embebido (Freeman, 2022).

Además, Spring Boot integra de forma nativa funcionalidades como seguridad (Spring Security), acceso a bases de datos (Spring Data JPA), monitoreo (Actuator), pruebas unitarias, y servicios RESTful. Esto convierte a Spring Boot en un ecosistema completo que facilita la creación de aplicaciones robustas y productivas desde las primeras fases del desarrollo (Walls, 2022).

Gracias a estas características, Spring Boot permite a los desarrolladores concentrarse en la lógica del negocio sin preocuparse excesivamente por la infraestructura, facilitando tanto la escalabilidad como el mantenimiento del código.

4.1.3 Beneficios de integrar Spring Boot con Angular

Un aspecto crucial en el desarrollo moderno es la **integración entre el front-end y el backend**. Angular y Spring Boot forman una combinación poderosa para el desarrollo full stack. Angular puede consumir servicios REST expuestos por Spring Boot, facilitando una arquitectura desacoplada donde el front-end se comunica con el backend a través de peticiones HTTP (Google, 2024). Esta separación de responsabilidades permite que equipos trabajen de manera paralela y especializada.

Uno de los principales beneficios de integrar Angular con Spring Boot es la **escalabilidad**. Mientras Angular se encarga de una experiencia de usuario fluida y dinámica en el navegador, Spring Boot maneja operaciones complejas del servidor como procesamiento de datos, autenticación, y persistencia en bases de datos. Además, ambos frameworks soportan herramientas modernas como JWT (JSON Web Tokens), lo que permite establecer mecanismos de autenticación y autorización seguros entre cliente y servidor.

Otro punto a favor es la **facilidad de despliegue**. Al utilizar Spring Boot con herramientas como Maven o Gradle, se puede empaquetar toda la lógica del servidor como un archivo ejecutable. Simultáneamente, el código de Angular puede compilarse y ser servido directamente desde el backend como archivos estáticos, o bien, desplegarse por separado en entornos de producción como contenedores Docker o servidores en la nube.

En conjunto, Angular y Spring Boot proporcionan una arquitectura moderna, modular y altamente mantenible que permite desarrollar aplicaciones web completas, rápidas, y listas para escalar en entornos empresariales exigentes.

4.2 Desarrollo del Back-end con Spring Boot: Creación de Entidades, Repositorios y Controladores

Una vez seleccionadas las herramientas tecnológicas adecuadas para el desarrollo de la aplicación, como **Spring Boot** para el backend y **JPA (Java Persistence API)** para la interacción con la base de datos, se procedió con la creación de las entidades principales del sistema. Estas entidades representan directamente las tablas de la base de datos, por lo que su correcta definición es fundamental para garantizar la integridad de los datos y la eficiencia en las operaciones.

El primer paso consistió en realizar las configuraciones necesarias para establecer la conexión entre la aplicación y el sistema gestor de base de datos (como MySQL o PostgreSQL). Esto se logró mediante la definición del archivo application.properties o application.yml, donde se especifican parámetros como la URL de conexión, el nombre de usuario, la contraseña, y el driver correspondiente.

src > main > resources > application.properties

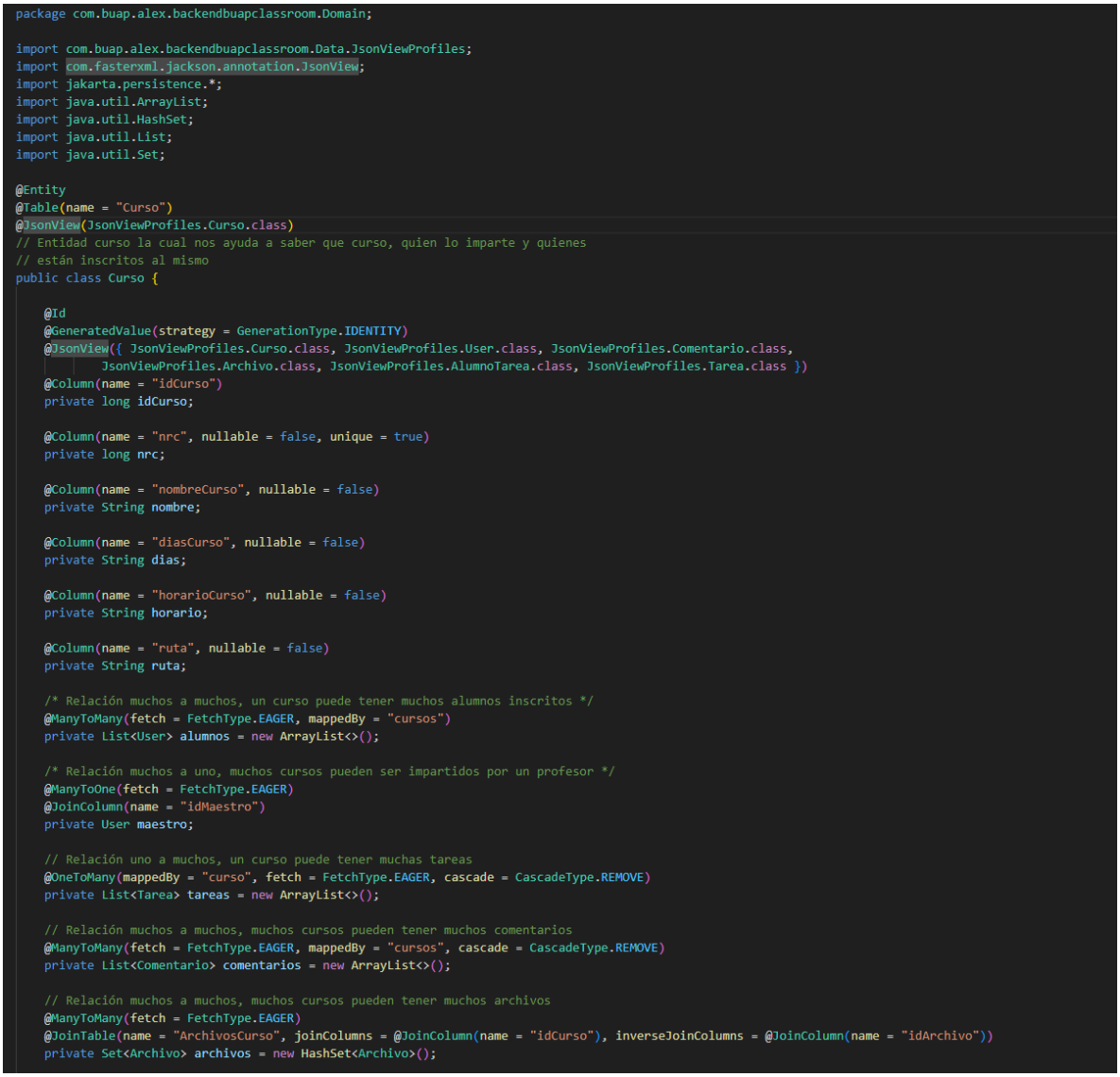
```
#logging
logging.level.root = info
logging.file.name = debugAppInfo.log

#DataBase
spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/datatest
spring.datasource.username =
spring.datasource.password =

#update if it's necessary
spring.jpa.hibernate.ddl-auto = update

#
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect

#Debug SQL
spring.jpa.show-sql = true

server.maxHttpRequestHeaderSize = 30720
```

Figura 4.1 Configuración Inicial del proyecto Spring Boot

Una vez comprobada exitosamente la conexión, se procedió al mapeo de las entidades de la base de datos a clases Java.

Gracias a Spring Boot y su integración con JPA, se facilita enormemente el trabajo con las relaciones entre entidades, como relaciones @OneToMany, @ManyToOne o @ManyToMany, sin necesidad de escribir consultas SQL personalizadas. JPA se encarga de traducir estas relaciones en consultas eficientes a nivel de base de datos, lo cual optimiza el

rendimiento y reduce significativamente el código repetitivo. Por ejemplo, la entidad Curso puede mapearse de la siguiente manera:



```java
package com.buap.alex.backendbuapclassroom.Domain;

import com.buap.alex.backendbuapclassroom.Data.JsonViewProfiles;
import com.fasterxml.jackson.annotation.JsonView;
import jakarta.persistence.*;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
@Table(name = "Curso")
@JsonView(JsonViewProfiles.Curso.class)
// Entidad curso la cual nos ayuda a saber que curso, quien lo imparte y quienes
// están inscritos al mismo
public class Curso {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @JsonView({ JsonViewProfiles.Curso.class, JsonViewProfiles.User.class, JsonViewProfiles.Comentario.class,
            JsonViewProfiles.Archivo.class, JsonViewProfiles.AlumnoTarea.class, JsonViewProfiles.Tarea.class })
    @Column(name = "idCurso")
    private long idCurso;

    @Column(name = "nrc", nullable = false, unique = true)
    private long nrc;

    @Column(name = "nombreCurso", nullable = false)
    private String nombre;

    @Column(name = "diasCurso", nullable = false)
    private String dias;

    @Column(name = "horarioCurso", nullable = false)
    private String horario;

    @Column(name = "ruta", nullable = false)
    private String ruta;

    /* Relación muchos a muchos, un curso puede tener muchos alumnos inscritos */
    @ManyToMany(fetch = FetchType.EAGER, mappedBy = "cursos")
    private List<User> alumnos = new ArrayList<>();

    /* Relación muchos a uno, muchos cursos pueden ser impartidos por un profesor */
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "idMaestro")
    private User maestro;

    // Relación uno a muchos, un curso puede tener muchas tareas
    @OneToMany(mappedBy = "curso", fetch = FetchType.EAGER, cascade = CascadeType.REMOVE)
    private List<Tarea> tareas = new ArrayList<>();

    // Relación muchos a muchos, muchos cursos pueden tener muchos comentarios
    @ManyToMany(fetch = FetchType.EAGER, mappedBy = "cursos", cascade = CascadeType.REMOVE)
    private List<Comentario> comentarios = new ArrayList<>();

    // Relación muchos a muchos, muchos cursos pueden tener muchos archivos
    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "ArchivosCurso", joinColumns = @JoinColumn(name = "idCurso"), inverseJoinColumns = @JoinColumn(name = "idArchivo"))
    private Set<Archivo> archivos = new HashSet<Archivo>();
```

Figura 4.2 Entidad desarrollada en Spring Boot para la plataforma educativa "Curso"

La anotación @Entity indica que esta clase es una entidad que será gestionada por JPA, y la anotación @Id marca el campo que actuará como identificador único de cada registro. El uso de @GeneratedValue permite que este valor se genere automáticamente. Además, si un curso está relacionado con un profesor, se puede representar dicha relación mediante @ManyToOne.

Una vez creadas las entidades, se procede con una parte esencial del backend: la implementación de los **repositorios**. Estos son interfaces que extienden de JpaRepository o CrudRepository, y permiten realizar operaciones como guardar, buscar, eliminar y actualizar registros sin necesidad de definir manualmente las consultas SQL. Por ejemplo, el repositorio para la entidad Curso podría definirse así:

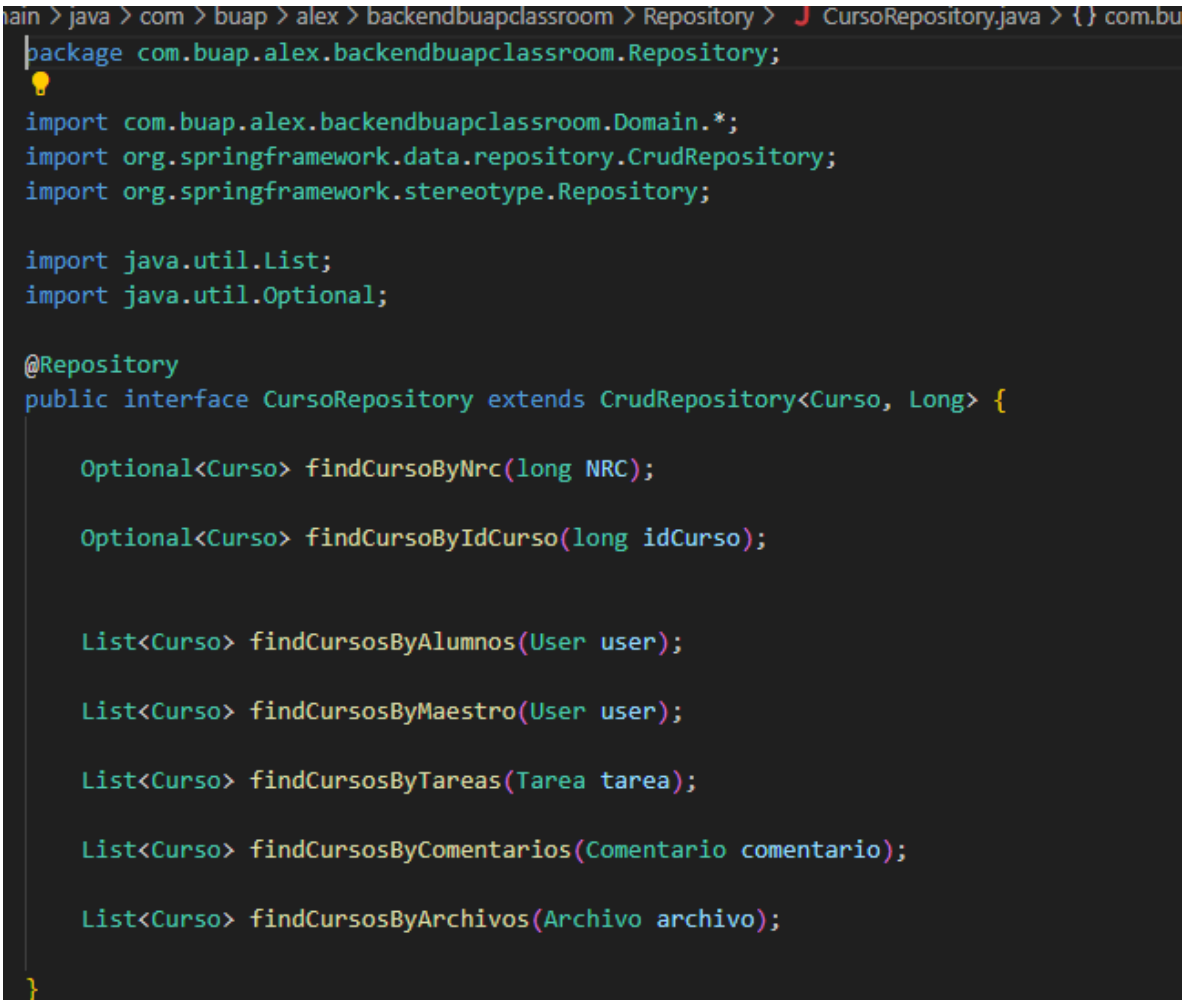


Figura 4.3 Repositorio desarrollado en Spring Boot ara la plataforma educativa "CursoRespository"

Aquí, la anotación @Repository marca esta interfaz como componente de acceso a datos, y al extender JpaRepository, Spring Boot provee automáticamente métodos como findAll(), findById(), save(), entre otros.

En estructuras más avanzadas, se recomienda incluir una **capa de servicios** (@Service), la cual se encarga de contener la lógica de negocio de la aplicación. Sin embargo, en este caso particular se optó por una arquitectura más directa, implementando directamente la capa de **controlador**.

El controlador, anotado con @RestController, se encarga de exponer las APIs que serán consumidas por el front-end o por otros servicios. A través de anotaciones como @GetMapping, @PostMapping, @PutMapping y @DeleteMapping, se define la manera en que los distintos métodos del controlador responden a las solicitudes HTTP. Un ejemplo básico del controlador de cursos sería:

```java
package com.buap.alex.backendbuapclassroom.Controller;

import com.buap.alex.backendbuapclassroom.Data.CursoMaestro;...

@RestController
@RequestMapping("/Cursos")
public class CursoController {

    final
    CursoRepository cursoRepository;
    final
    UserRepository userRepository;

    public CursoController(CursoRepository cursoRepository, UserRepository userRepository) {
        this.cursoRepository = cursoRepository;
        this.userRepository = userRepository;
    }


    //Servicio para crear un curso
    @PostMapping("/create")
    public ResponseEntity<?> crearCurso(@RequestBody CursoMaestro cursoMaestro) {
        Curso curso = cursoMaestro.getCurso();
        User user = verifyUser(cursoMaestro.getId());
        if (user.getTipo() != 0) {
            throw new ResourceNotFoundException(message:"El usuario no es un maestro");
        }
        String ruta = createDirectory(curso.getNombre(), curso.getNrc());
        curso.setRuta(ruta);
        curso.setMaestro(user);
        cursoRepository.save(curso);
        return new ResponseEntity<>(HttpStatus.CREATED);
    }


    //Servicio para obtener datos de un curso requerido
    @GetMapping("/getData")
    public ResponseEntity<?> listarCursos(@RequestParam long nrc) throws JsonProcessingException {
        Curso curso = verifyCursoByNrc(nrc);
        @SuppressWarnings("unchecked")
        DataCursos dataCursos = new DataCursos(curso.getAlumnos(), curso.getComentarios(),
                (List<Archivo>) curso.getArchivos(), curso.getTareas());

        String datosCurso = new ObjectMapper().writerWithView(serializationView:JsonViewProfiles.Curso.class)
                .writeValueAsString(dataCursos);
        return new ResponseEntity<>(datosCurso, HttpStatus.OK);
    }


    //Servicio para actualizar los datos de un curso
    @PutMapping("/update")
    public ResponseEntity<Curso> updateCurso(@RequestBody ModificarCurso modificarCurso) {
        Curso cursoData = modificarCurso.getCurso();
        Curso curso = verifyCursoById(modificarCurso.getIdCurso());

        curso.setDias(cursoData.getDias());
        curso.setHorario(cursoData.getHorario());

        cursoRepository.save(curso);
        return new ResponseEntity<>(HttpStatus.OK);
    }
```

Figura 4.4 Controlador desarrollado en Spring Boot para la plataforma educativa "CursoController"

En este fragmento, @RestController indica que la clase responderá a solicitudes web y devolverá datos en formato JSON por defecto. @RequestMapping("/Cursos") define la ruta base para las solicitudes, y el método listarCursos() expone un endpoint tipo GET que devuelve todos los cursos registrados en la base de datos.

Con esto se completa la implementación básica del backend utilizando Spring Boot. Esta arquitectura permite crear servicios robustos, escalables y fáciles de mantener, donde la separación entre entidades, repositorios y controladores proporciona una estructura limpia y clara.

Gracias a Spring Boot, gran parte de la complejidad del backend se abstrae, permitiendo a los desarrolladores concentrarse en la lógica del negocio y agilizando significativamente el desarrollo de servicios web listos para integrarse con un front-end moderno como Angular.

4.3 Desarrollo del Front-end con Angular

Una vez terminada la arquitectura para el back-end, se comenzó con el desarrollo del front-end.

Con el planteamiento de la interfaz gráfica, basada en la interfaz de Google Classroom, puesto que es la más eficaz y usable para el usuario, es intuitiva y fácil de usar.
La herramienta seleccionada para poder desarrollar el front end, es angular debido a que se presenta como una solución robusta, integral y ampliamente adoptada por empresas y desarrolladores de todo el mundo. Diseñado y mantenido por Google, Angular ha evolucionado hasta convertirse en un framework completo que facilita la creación de aplicaciones web modernas, modulares y de alto rendimiento.

Una de las principales razones para adoptar Angular radica en su arquitectura basada en componentes. Este enfoque promueve una organización clara y reutilizable del código, permitiendo a los desarrolladores dividir la interfaz en bloques funcionales independientes. Esta

estructura facilita tanto el mantenimiento como la escalabilidad de los proyectos, ya que cada componente puede desarrollarse, probarse y modificarse de forma aislada.

Además, Angular ofrece un enfoque integral que cubre múltiples aspectos del desarrollo front-end. Mientras que otras bibliotecas como React se centran únicamente en la interfaz, Angular proporciona herramientas y soluciones para enrutamiento, gestión de formularios, comunicación HTTP, validaciones, animaciones, pruebas, y más, todo dentro del mismo ecosistema. Esta cohesión reduce la necesidad de integrar librerías externas, minimizando los conflictos y la curva de aprendizaje al trabajar con un único marco conceptual.

Otro aspecto destacado es su uso de TypeScript, un superconjunto tipado de JavaScript que aporta mayor seguridad y claridad al desarrollo. TypeScript permite detectar errores en tiempo de compilación, facilita el autocompletado en los editores modernos y mejora la experiencia del desarrollador mediante una codificación más estructurada. Esto es especialmente beneficioso en proyectos a gran escala o en equipos de desarrollo numerosos, donde la consistencia y la legibilidad del código son fundamentales.

Angular también pone énfasis en las buenas prácticas de desarrollo a través de su CLI (Command Line Interface), una poderosa herramienta que automatiza tareas comunes como la generación de componentes, servicios, módulos y pruebas. Esto no solo acelera el desarrollo, sino que asegura que el código siga convenciones estandarizadas y organizadas desde el principio.

Desde el punto de vista de rendimiento, Angular implementa técnicas como el renderizado anticipado (ahead-of-time compilation) y la detección eficiente de cambios, lo que resulta en aplicaciones más rápidas y con mejor experiencia para el usuario final. Combinado

con el soporte para aplicaciones progresivas (PWA), Angular permite construir aplicaciones web con capacidades cercanas a las de una aplicación nativa.

Por último, la amplia comunidad y respaldo corporativo de Angular garantiza una evolución constante del framework, acceso a una gran cantidad de recursos educativos y soporte técnico sólido. Esta comunidad activa también fomenta el desarrollo de bibliotecas y extensiones que complementan sus funcionalidades.

En conclusión, Angular se posiciona como una opción sólida para el desarrollo front-end por su enfoque integral, arquitectura modular, soporte a herramientas modernas como TypeScript, y su compromiso con las buenas prácticas de desarrollo. Para equipos que buscan construir aplicaciones escalables, mantenibles y con alto estándar de calidad, Angular representa una inversión tecnológica estratégica y confiable.

4.3.1 Desarrollo de interfaces generales (Profesor/Alumno)

Una vez seleccionada la herramienta tecnológica para el desarrollo de la plataforma, se procede al diseño y construcción de las interfaces gráficas. Este proceso no solo implica la creación de pantallas funcionales, sino también la aplicación de lineamientos visuales que garanticen uniformidad y coherencia en toda la aplicación.

Para ello, se hace uso del **manual de identidad gráfica institucional**, documento que establece los parámetros a seguir en cuanto a colores, tipografías y elementos visuales. De esta manera, se asegura que la plataforma no solo sea funcional, sino que también refuerce la identidad institucional, transmitiendo confianza y profesionalismo a sus usuarios.

En las siguientes secciones se presentan con mayor detalle los componentes más relevantes que conforman la aplicación web.

Inicio de sesión

El inicio de sesión constituye la primera interacción que tienen los usuarios con la plataforma. Dado que esta aplicación está dirigida tanto a **profesores** como a **alumnos**, resulta fundamental garantizar un acceso seguro y una experiencia clara.

La interfaz de inicio de sesión se diseñó de manera minimalista, priorizando la facilidad de uso. Incluye campos básicos como correo institucional y contraseña. Con esto se busca que la autenticación sea sencilla, sin comprometer la seguridad.

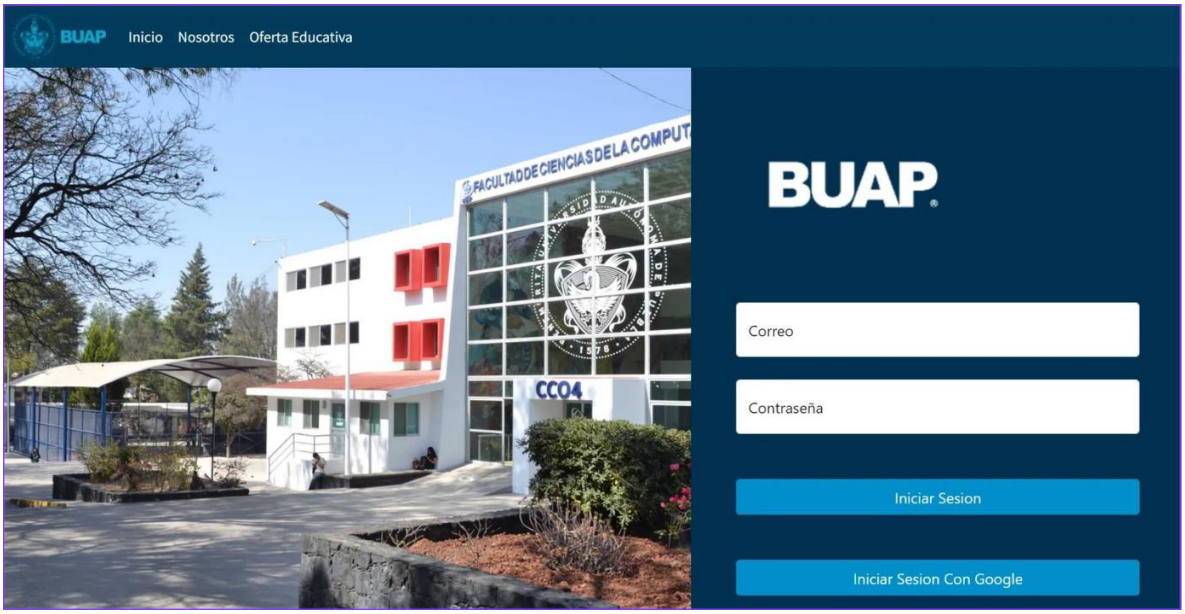


Figura 4.5 Interfaz de inicio de sesión desarrollada para la plataforma educativa

Barra de navegación

La **barra de navegación** es uno de los elementos centrales de la plataforma, ya que permite al usuario desplazarse entre las distintas secciones de manera intuitiva.

En esta se incorpora:

- **Logo institucional** de la universidad, reforzando la identidad gráfica.
- **Menú principal** con tres secciones clave:
    - **Inicio**: Dirige al dashboard o página principal personalizada según el rol del usuario.
    - **Nosotros**: Contiene información general sobre la institución o la propia plataforma.
    - **Oferta Educativa**: Permite acceder al catálogo de carreras, diplomados y cursos disponibles.
- En la parte derecha de la barra se incluye la **foto de perfil**, el **correo electrónico del usuario**, y un **menú desplegable** con opciones como:
    - Configuración
    - Perfil
    - Cerrar Sesión

La inspiración de este diseño proviene de plataformas educativas modernas como **Google Classroom**, donde se busca mantener un estilo limpio y minimalista, dando prioridad al contenido y evitando la saturación visual.

Figura 4.6 Barra de navegación desarrollada para la plataforma educativa

Figura 4.7 Menú desplegable desarrollada para la plataforma educativa

Ventana Principal

Una vez realizado el **inicio de sesión**, el sistema despliega la **ventana principal**, que funciona como punto de partida para la navegación del usuario dentro de la plataforma.

En esta sección se presentan los **cursos asignados**, adaptando la información mostrada según el rol del usuario (profesor o alumno). Aunque la estructura general es la misma para ambos, el contenido de las tarjetas varía de acuerdo con las necesidades de cada perfil:

- **Profesor**: Visualiza un conjunto de tarjetas donde se listan los cursos que imparte. Cada tarjeta incluye el **nombre del curso** y el **salón asignado**, lo cual facilita la identificación rápida de cada grupo.

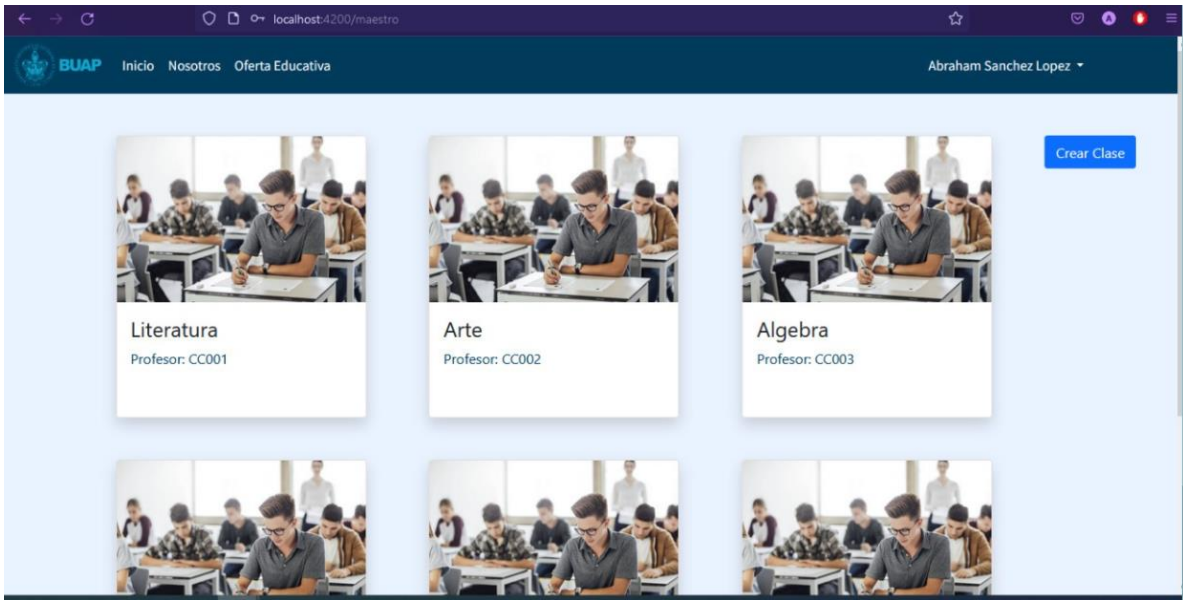


Figura 4.8 Interfaz cursos del docente desarrollada para la plataforma educativa

- **Alumno**: Observa tarjetas similares, pero en su caso, además del **nombre del curso**, se muestra el **nombre del profesor encargado**, permitiendo al estudiante reconocer fácilmente a quién corresponde la materia.

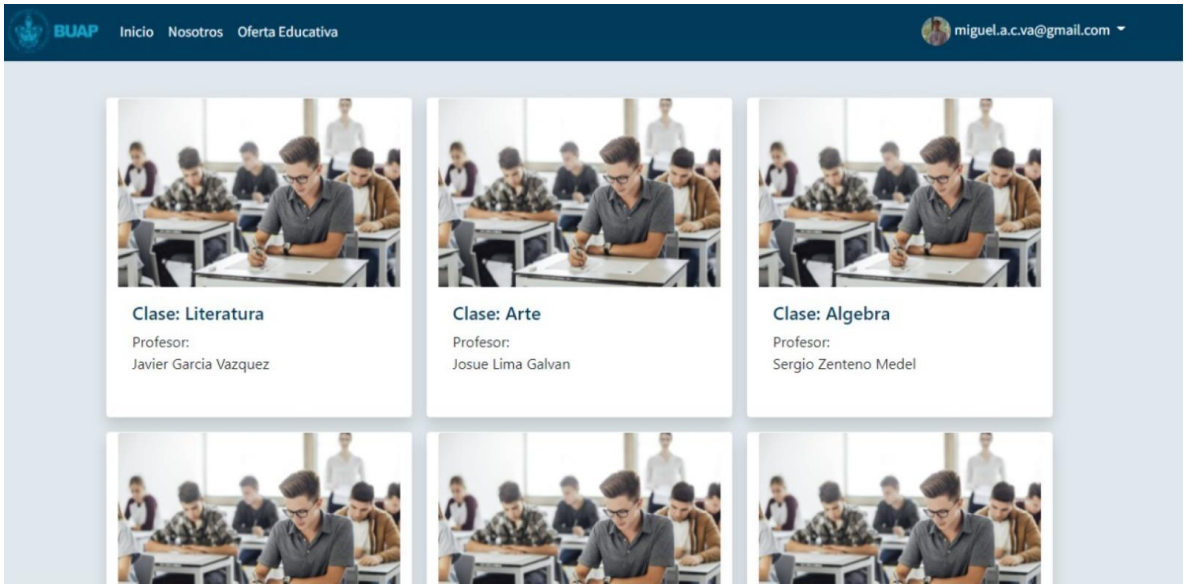


Figura 4.9 Interfaz cursos del alumno desarrollada para la plataforma educativa

Este diseño mantiene una presentación clara, organizada y uniforme, al mismo tiempo que personaliza la información de acuerdo con el rol de usuario, ofreciendo una experiencia más funcional e intuitiva.

Creación de Curso

Dentro de la plataforma, cada **profesor** cuenta con la posibilidad de crear sus propios cursos, los cuales posteriormente estarán disponibles para que los alumnos puedan inscribirse.

Para llevar a cabo este proceso, el sistema presenta un **formulario de registro de curso** en el que el docente debe proporcionar la siguiente información:

- **Nombre del curso**: Identificación principal del curso.
- **Sección a la que pertenece**: Permite distinguir entre diferentes grupos o divisiones académicas.
- **Materia por impartir**: Especifica el área o asignatura correspondiente.
- **Datos adicionales**: Información complementaria como número de salón, enlace a la clase en línea u otros detalles relevantes.
- **Matrícula del profesor**: Código o número único que identifica al docente responsable del curso.

Una vez completado este formulario y confirmada la creación, el sistema registra el curso en la plataforma, quedando disponible dentro del catálogo general. A partir de ese momento, los **alumnos podrán visualizarlo e inscribirse**, facilitando así la gestión académica y el inicio de actividades dentro del aula virtual o presencial.

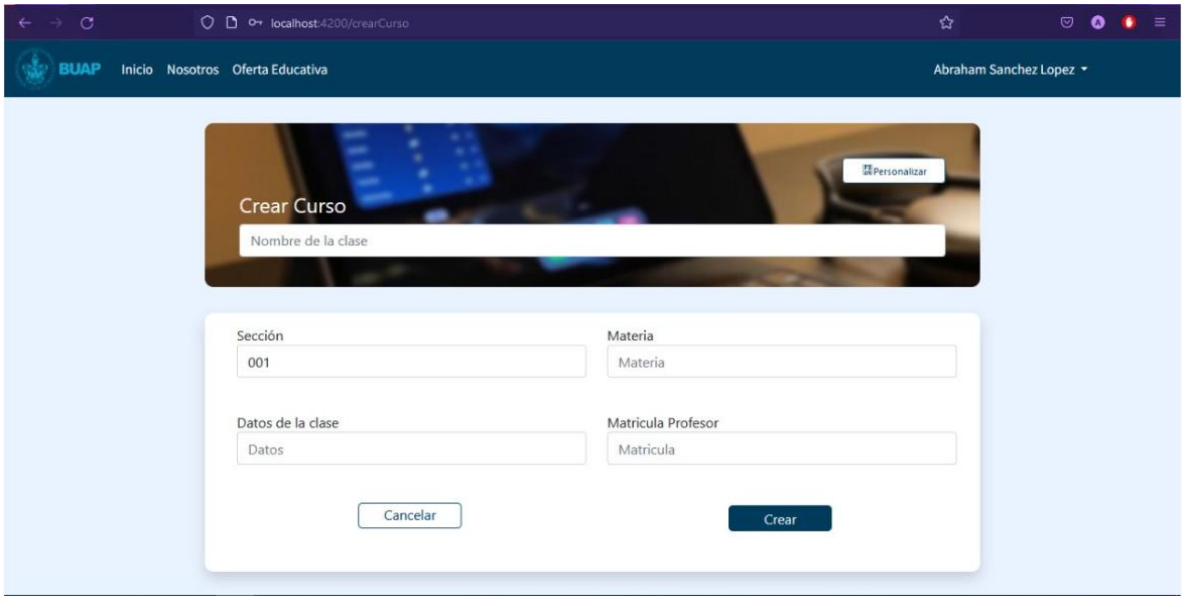


Figura 4.10 Interfaz creación de curso desarrollada ara la plataforma educativa

Editar Perfil

Un aspecto importante en cualquier sistema es la **personalización** del perfil. Por esta razón, se habilita una sección que permite a los usuarios modificar ciertos datos básicos, tales como su **nombre de identificación** y su **fotografía de perfil**.

Si bien las opciones son sencillas, esta funcionalidad contribuye a que cada usuario se sienta identificado dentro del sistema y pueda diferenciarse de otros miembros de la comunidad.

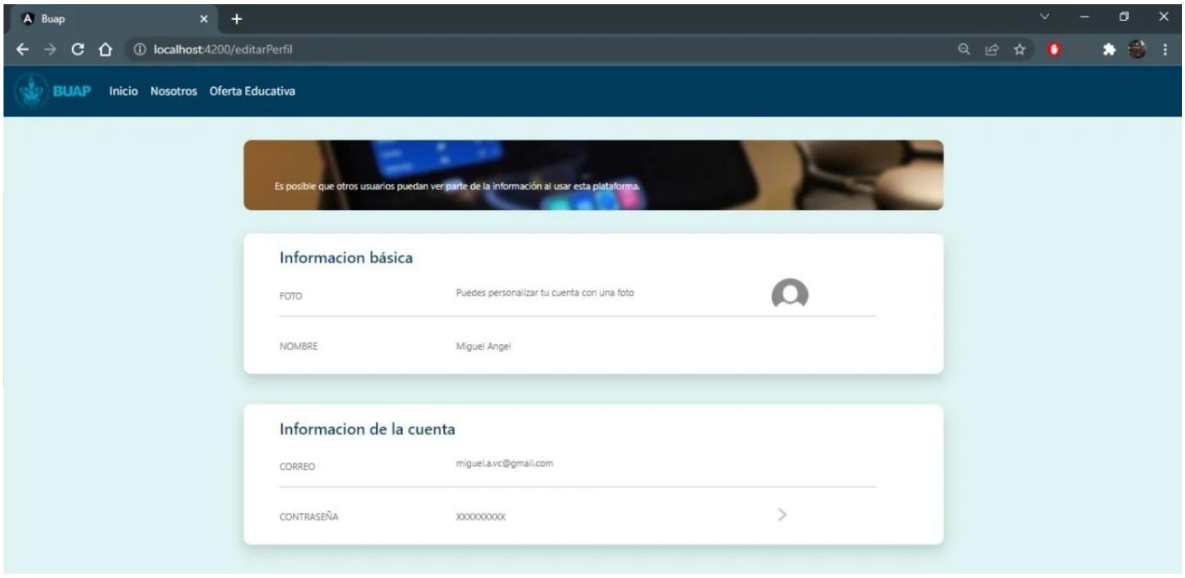


Figura 4.11 Interfaz editar perfil desarrollada para la plataforma educativa

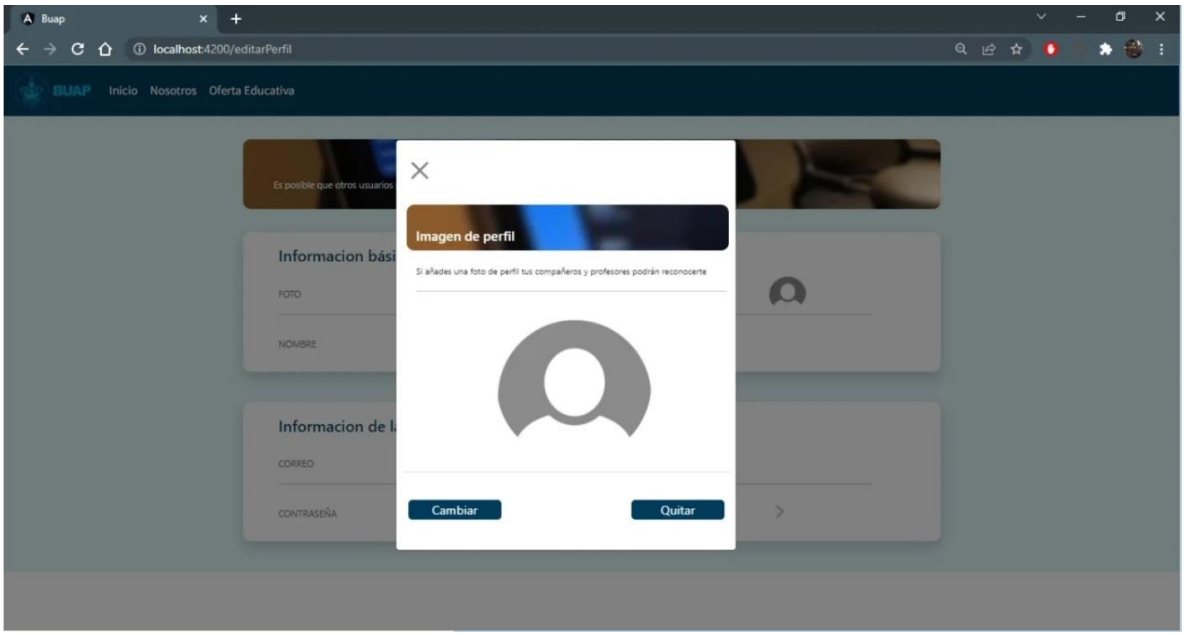


Figura 4.12 Modulo cambiar imagen de perfil desarrollada parala plataforma educativa

Sección de curso

La sección de curso constituye el **núcleo funcional** de la plataforma. Gracias al uso de **Angular como framework de desarrollo front-end**, es posible trabajar con componentes reutilizables que se adaptan dinámicamente según el rol del usuario (profesor o alumno).

Esto significa que no es necesario diseñar interfaces completamente distintas para cada perfil, sino que una misma interfaz puede mostrar opciones personalizadas según corresponda.

- **Profesor**: Tiene la capacidad de crear actividades (tareas, ensayos, exámenes y proyectos), gestionar entregas, así como compartir material adicional en forma de archivos o enlaces.

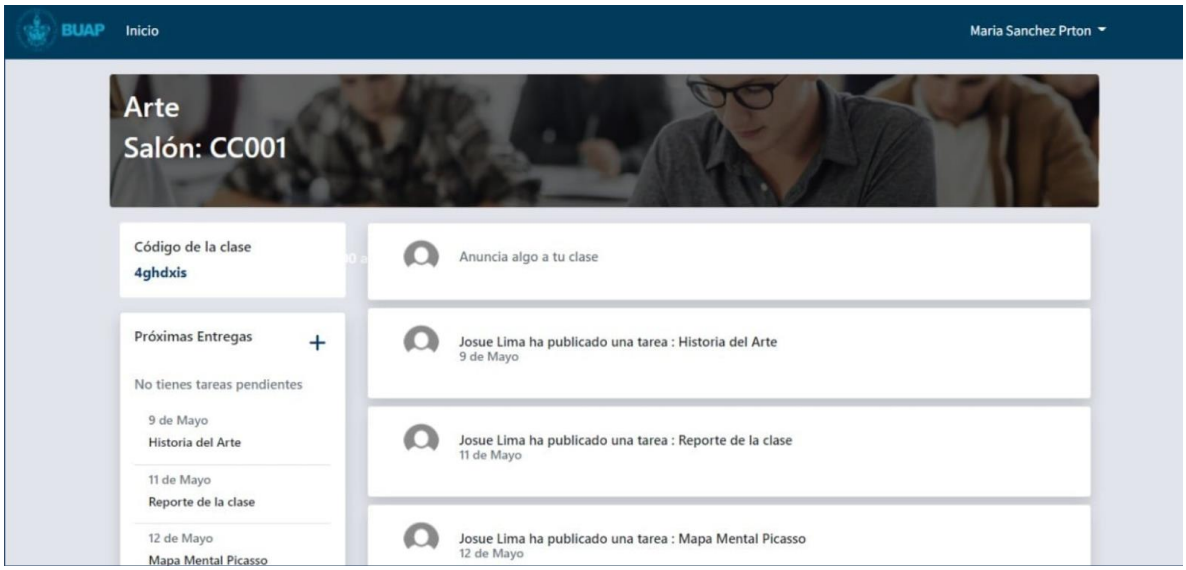


Figura 4.13 Interfaz sección del curso del docente, desarrollada para la plataforma educativa

- **Alumno**: Puede visualizar las actividades asignadas, entregar sus evidencias y acceder al material complementario proporcionado por el profesor.

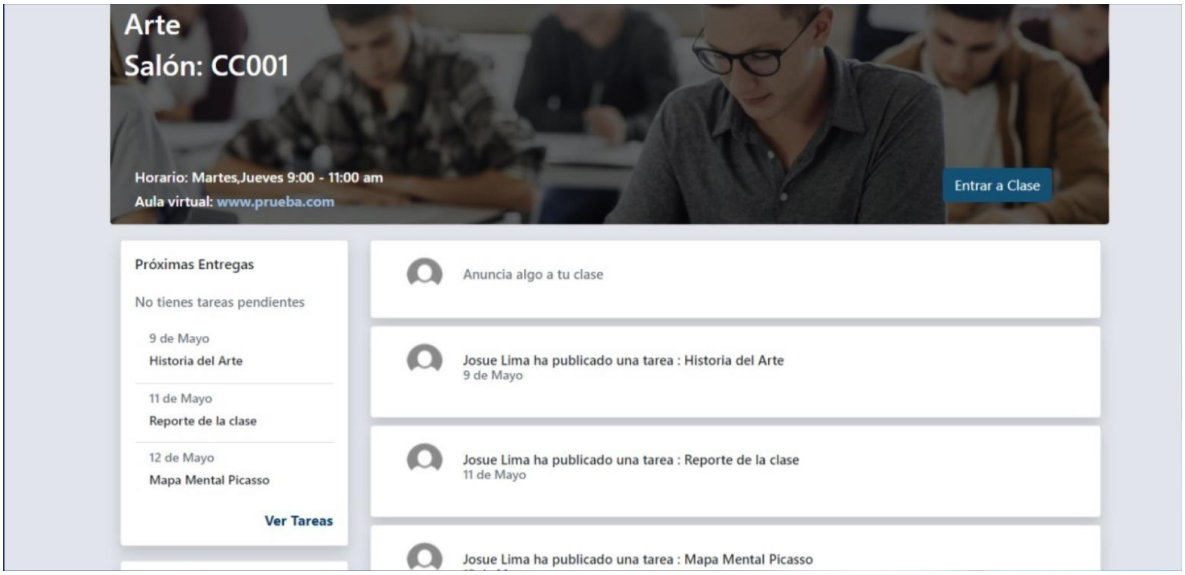


Figura 4.14 Interfaz sección del curso del alumno, desarrollada para la plataforma educativa

Además, esta sección cuenta con una **barra de búsqueda avanzada** que permite localizar recursos, actividades y documentos dentro del curso. Esta función se apoya en la **web semántica**, lo cual facilita establecer relaciones entre temas, archivos y conceptos relevantes para el aprendizaje.

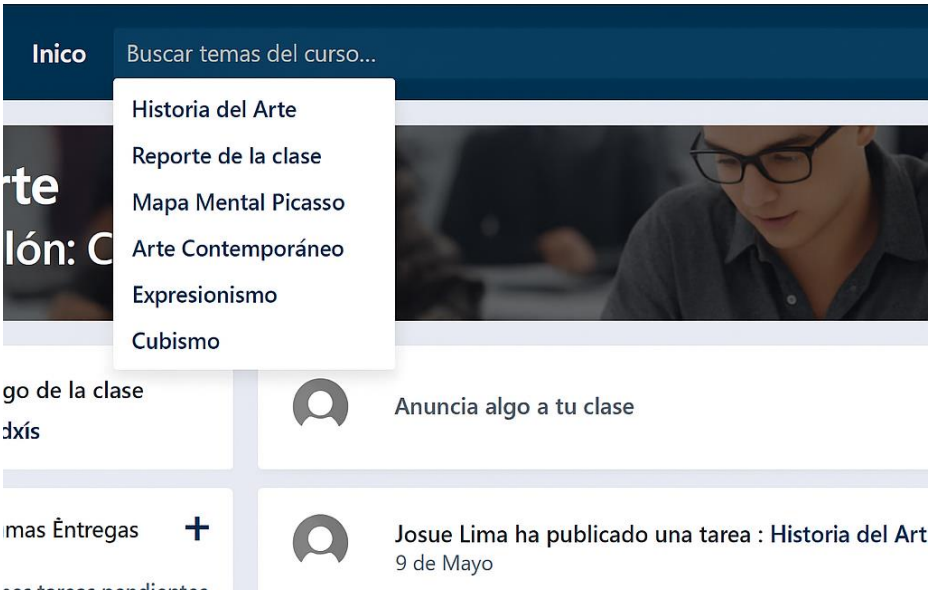


Figura 4.15 Barra de búsqueda implementando las ontologías, desarrollada para la plataforma educativa

Creación edición y eliminación de una actividad

Creación de Actividades

Dentro de la **sección de curso**, los profesores cuentan con la posibilidad de generar nuevas actividades dirigidas a sus alumnos. Para llevar a cabo esta acción, se habilita un **formulario estructurado** que contiene los siguientes campos principales:

- **Título de la actividad**
- **Instrucciones detalladas o especificaciones**
- **Fecha límite de entrega**, seleccionada a través de un calendario interactivo
- **Carga de archivos adjuntos** (guías, documentos de referencia o material complementario)

El objetivo de este diseño es garantizar que el proceso de creación sea **intuitivo, claro y eficiente**, reduciendo la posibilidad de errores y asegurando que los alumnos reciban información completa y precisa sobre las tareas asignadas.

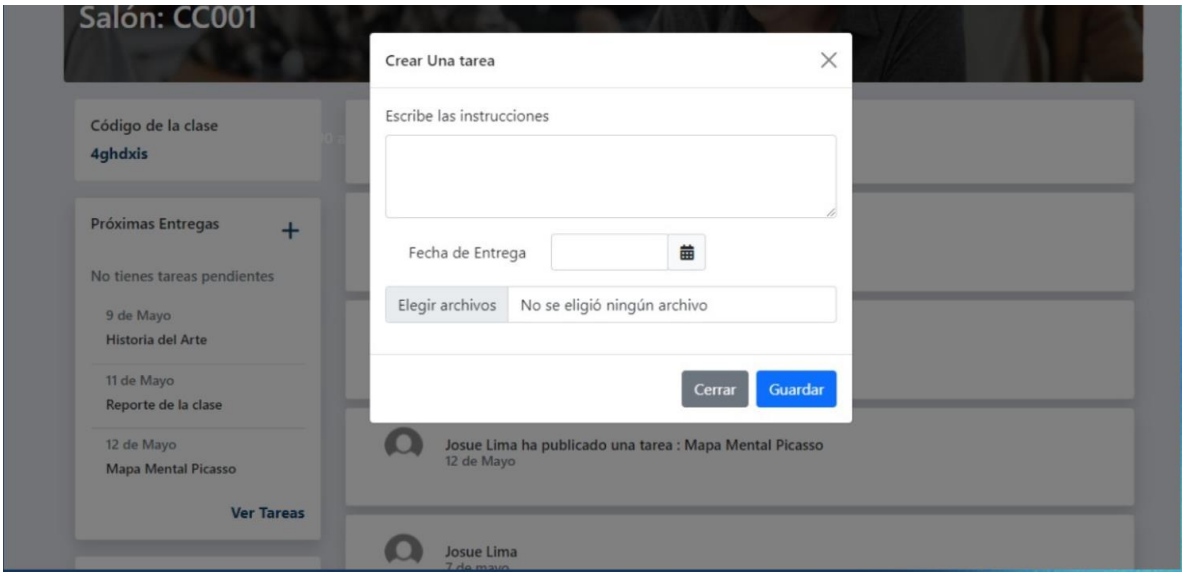


Figura 4.16 Modulo creación de actividad desarrollada para la plataforma educativa

Visualización de actividad

Una vez que el docente ha creado la actividad, el sistema genera una nueva página dedicada exclusivamente a ella. En este espacio se concentran los datos relevantes, tanto para el profesor como para los alumnos inscritos en el curso. Los elementos principales que se muestran son los siguientes:

- **Información del curso**
  - Nombre del curso
  - Aula virtual o presencial asociada
  - Horario correspondiente
- **Título de la actividad**

- **Fecha de creación**
- **Descripción de la actividad**
- **Fecha límite de entrega**
- **Botón “Editar Tarea”**
- **Botón “Eliminar”**
- **Espacio de comentarios**, en el cual tanto alumnos como el profesor pueden interactuar.

Este apartado no solo organiza la información de forma clara, sino que también sirve como punto de referencia para realizar ajustes posteriores o atender dudas de los alumnos.

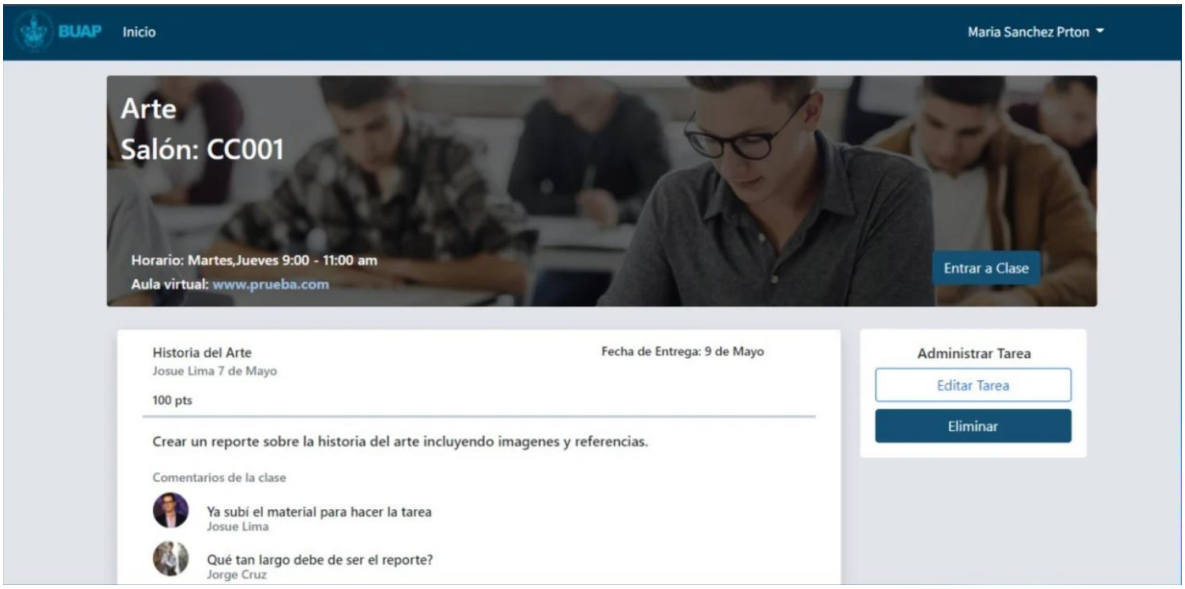


Figura 4.17 Interfaz para visualizar la información de la actividad desarrollada para la plataforma educativa

Edición de actividad

En ocasiones, es posible que el docente necesite realizar modificaciones en una actividad previamente creada, ya sea para **ampliar la información**, **ajustar la fecha de entrega** o **añadir material complementario**. Para estos casos se habilita el botón **“Editar Tarea”**.

Al hacer clic en dicha opción, se despliega un **formulario similar al de creación**, aunque con opciones específicas para edición:

- Modificación de las **instrucciones detalladas o especificaciones**
- Ajuste de la **fecha límite de entrega** mediante el calendario
- Carga o sustitución de **archivos adjuntos** relacionados con la actividad

De esta manera, el sistema ofrece flexibilidad al docente, evitando la necesidad de eliminar y volver a crear una actividad desde cero.

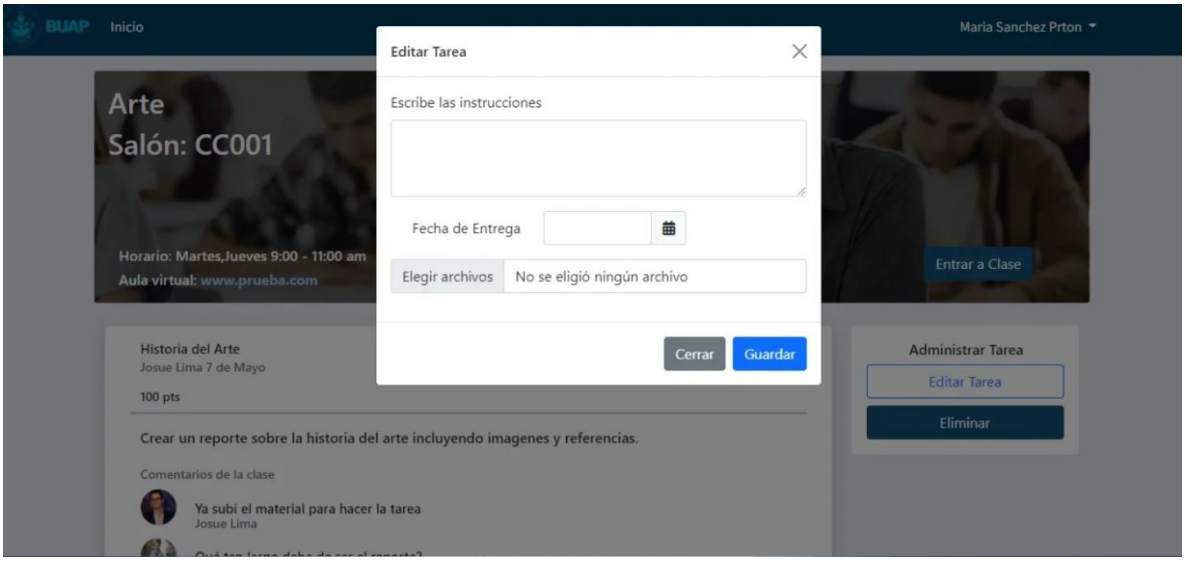


Figura 4.18 Modulo para editar una actividad desarrollada para la plataforma educativa

Eliminar Actividad

En caso de que una actividad ya no sea necesaria o deba retirarse por cualquier motivo, el profesor puede hacer uso del botón **"Eliminar"**. Esta acción borra el registro correspondiente de manera inmediata, garantizando que los alumnos ya no tengan acceso a dicha tarea.

Comentarios en las actividades

Un aspecto fundamental en el aprendizaje es la **comunicación constante** entre alumnos y profesores. Por ello, cada actividad cuenta con un **espacio de comentarios**, donde ambas partes pueden interactuar libremente.

- **Alumnos**: Pueden plantear dudas, solicitar aclaraciones o compartir sugerencias relacionadas con la actividad.
- **Profesor**: Tiene la posibilidad de responder inquietudes, brindar retroalimentación y orientar a los estudiantes durante el proceso de resolución de la tarea.

Todos los comentarios son **visibles para los usuarios inscritos en el curso**, lo que favorece un entorno de **colaboración y aprendizaje colectivo**. Esta herramienta se convierte, por lo tanto, en un recurso esencial para enriquecer la experiencia educativa y fortalecer la comunicación bidireccional dentro de la plataforma.

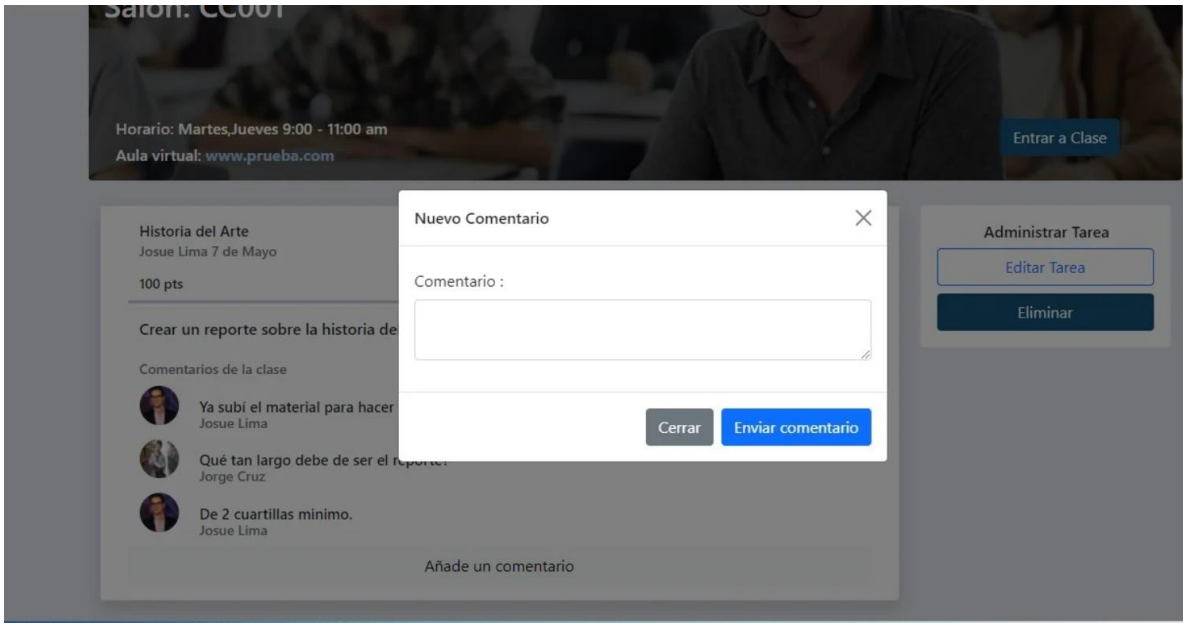


Figura 4.19 Modulo para agregar un comentario en la actividad desarrollada para la plataforma educativa

Entregar actividad

Un alumno tiene acceso a la misma ventana como a la que se menciona en el apartado Visualización de actividad.

dado las propiedades de Angular que nos permite reutilizar componentes, Sin embargo, si tiene un cambio y esta es en el apartado de los botones, mientras el profesor cuenta con los botones de **“Editar tarea”** y **“Eliminar”**, el alumno tiene la opción de **“Añadir trabajo”** y **“Entregar”.**

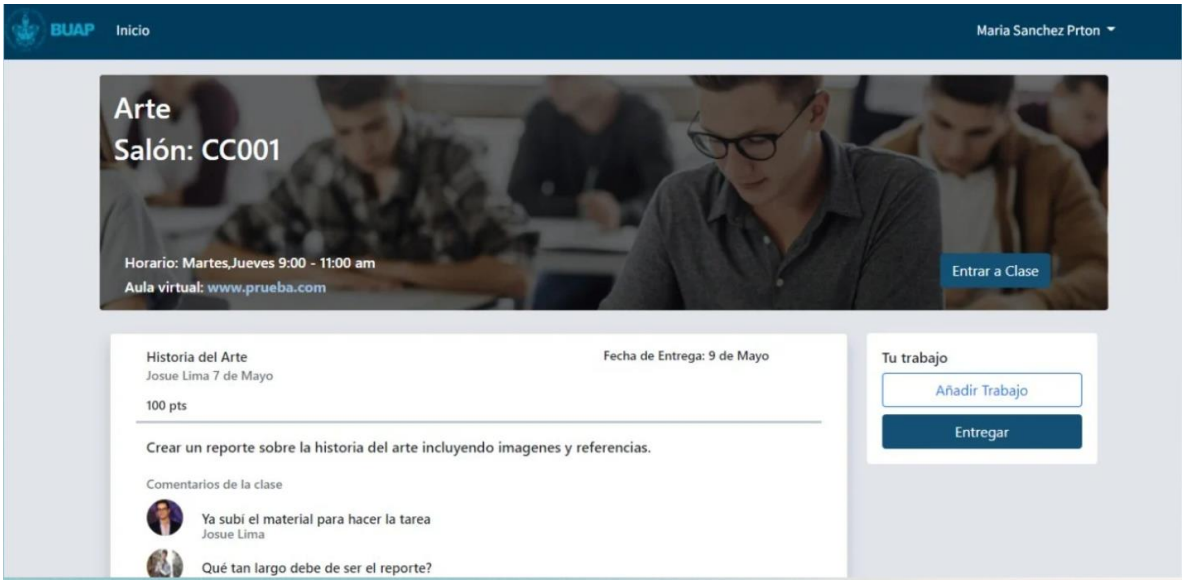


Figura 4.20 Interfaz de la actividad para el alumno desarrollada para la plataforma educativa

Añadir trabajo

En este apartado el alumno al hacer clic se muestra un simple form en el cual podrá adjuntar el material de evidencia relacionado a la finalización de la actividad.

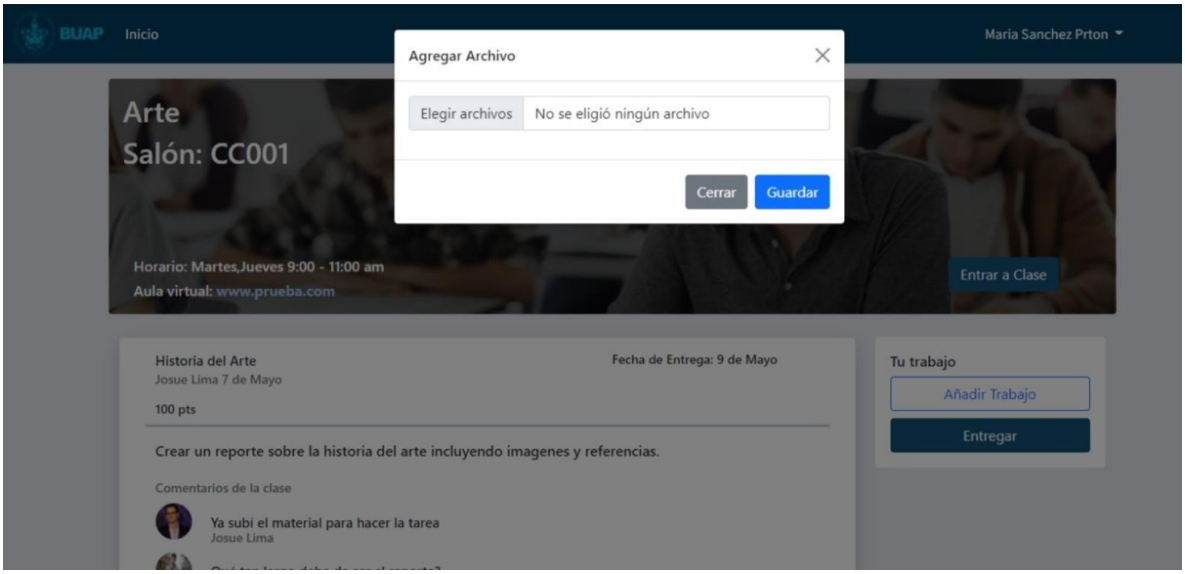


Figura 4.21 Modulo para subir la entrega de evidencia para la actividad desarrollada para la plataforma educativa

Entregar

Esta función corresponde a cambiar el estatus de la actividad del alumno a finalizada, esperando la revisión del profesor.

Información detallada de actividades

Dentro del sistema se habilito un apartado tanto para profesores como alumnos, en el cual se muestra el detalle de las actividades dentro del curso, retomando la idea de utilizar los componentes, en ambos casos la pantalla cuenta con algunos elementos en común, por ejemplo:

- Título de la actividad
- Estado de la actividad (Cerrada o En curso)
- Fecha de entrega

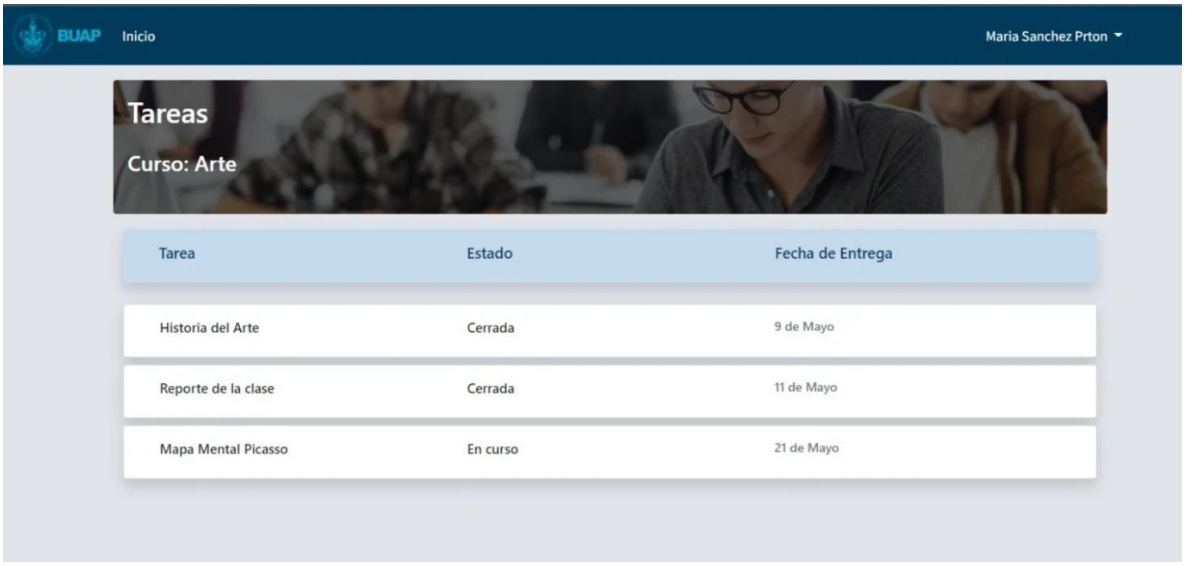


Figura 4.22 Interfaz para visualizar las actividades del curso por parte del docente, desarrollada para la plataforma educativa

En el caso particular del alumno este apartado cuenta con la calificación recibida por parte del docente.

Figura 4.23 Interfaz para visualizar las actividades del curso por parte del alumno desarrollada para la plataforma educativa

Capítulo 5: Implementación de Ontologías con Protégé

5.1 Introducción

En este capítulo se presenta la implementación de una ontología base para BUAPP utilizando la herramienta Protégé. El propósito de esta ontología es brindar soporte a las búsquedas relacionales que se realizan desde el menú de navegación (navbar), como por ejemplo Historia del Arte, Mapa Mental Picasso o Cubismo. Para ello se propone un modelo semántico que permite establecer relaciones entre las entidades Curso, Actividad y Usuario, favoreciendo así la organización y recuperación de la información.

5.2 Estructura de la ontología

La ontología desarrollada contempla como clases principales: Usuario, Profesor, Alumno, Curso, Actividad y Material. Dentro de esta estructura, las clases Profesor y Alumno se definen como subclases de Usuario. Asimismo, se establecen propiedades de objeto que permiten vincular las clases: imparteCurso (Profesor → Curso), asignadoA (Curso → Alumno), tieneActividad (Curso → Actividad) y subeMaterial (Usuario → Material). De manera complementaria, se incluyen propiedades de datos para describir atributos específicos como nombre, correo, título de la actividad y fecha de entrega.

5.3 Flujo de trabajo en Protégé

El proceso de construcción de la ontología en Protégé puede resumirse en los siguientes pasos:

1. Definir las clases principales de acuerdo con el dominio del problema.
2. Configurar las propiedades de objeto y de datos que permiten establecer relaciones y atributos.
3. Crear individuos de ejemplo (por ejemplo: Curso_HistoriaDelArte, Actividad_MapaMentalPicasso, Alumno_JosueLima).
4. Verificar la coherencia del grafo mediante el uso de un razonador.
5. Ejecutar consultas SPARQL sobre la ontología para alimentar el buscador del curso.

5.4 Consultas SPARQL basadas en el navbar

Para ilustrar la funcionalidad del modelo, se presentan algunos ejemplos de consultas SPARQL que permiten obtener información directamente de la ontología. Estos ejemplos se relacionan con los temas mostrados en el menú de navegación y se han incorporado en el documento en formato de imagen.

```
PREFIX ex: <http://www.ejemplo.org/ontologia#>
SELECT ?actividad
WHERE {
  ?curso ex:titulo "Historia del Arte" .
  ?curso ex:tieneActividad ?actividad .
}
```

Figura 5.1 Consulta SPARQL para obtener actividades de un curso

```
PREFIX ex: <http://www.ejemplo.org/ontologia#>
SELECT ?actividad
WHERE {
  ?alumno ex:nombre "Josue Lima" .
  ?curso ex:asignadoA ?alumno .
  ?curso ex:tieneActividad ?actividad .
}
```

Figura 5.2 Consulta SPARQL para obtener actividades asignadas a un alumno

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ex: <http://www.ejemplo.org/ontologia#>
SELECT ?actividad ?fecha
WHERE {
  ?actividad ex:fechaEntrega ?fecha .
  FILTER (?fecha >= "2024-05-10"^^xsd:date)
}
```

Figura 5.3 Consulta SPARQL para filtrar actividades por fecha de entrega

5.5 Integración con BUAPP

La ontología desarrollada puede integrarse con BUAPP mediante un proceso en el que el backend traduce los términos escritos en el buscador a consultas SPARQL sobre el grafo RDF. Los resultados obtenidos se devuelven al front-end implementado en Angular, con el fin de poblar el menú de sugerencias del navbar y las tarjetas de actividades. Este enfoque no solo

habilita búsquedas textuales, sino también búsquedas semánticas basadas en las relaciones entre entidades, lo que incrementa la precisión de la información recuperada.

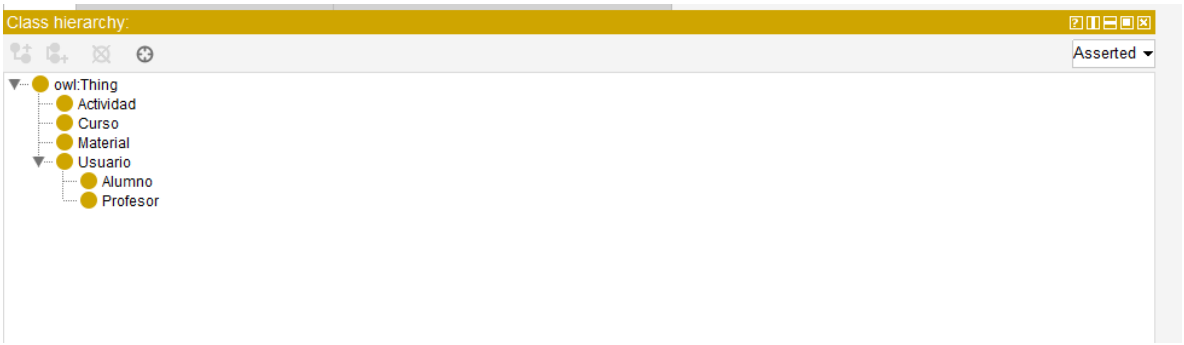


Figura 5.4 Vista de la jerarquía de clases en Protégé

Capítulo 6: Mejoras Futuras

6.1 Introducción

En este capítulo se presentan posibles líneas de mejora y evolución tanto para la ontología desarrollada como para la aplicación BUAPP en su conjunto. El objetivo es plantear escenarios que permitan ampliar la funcionalidad, optimizar el rendimiento y ofrecer una mejor experiencia de uso a estudiantes y profesores.

6.2 Optimización de la ontología

La ontología puede ampliarse con nuevas clases y propiedades, tales como Evaluación, Recurso Multimedia, Foro de Discusión y Notificación. Asimismo, es posible incorporar

relaciones semánticas más complejas que permitan inferir conocimiento, por ejemplo, detectar alumnos en riesgo académico por retrasos en entregas. La integración de razonadores avanzados permitirá automatizar sugerencias y alertas.

6.3 Integración tecnológica

La plataforma podrá vincularse con bases de datos externas y servicios web que enriquezcan el ecosistema de información. Entre las mejoras se contempla la compatibilidad con APIs educativas de Google Classroom, Microsoft Teams y otros LMS, así como la integración de servicios de inteligencia artificial para la recomendación personalizada de materiales de estudio.

6.4 Escalabilidad y rendimiento

Será necesario optimizar el motor de consultas para manejar grandes volúmenes de información. El uso de triplestores avanzados como Apache Jena o GraphDB permitirá gestionar grafos RDF a gran escala. Asimismo, la migración hacia infraestructuras en la nube garantizará mayor disponibilidad, resiliencia y escalabilidad.

6.4.1 Experiencia del usuario

El sistema podrá ofrecer un buscador semántico enriquecido con autocompletado y filtrado avanzado, además de módulos de accesibilidad para personas con discapacidad. También se planea diseñar interfaces adaptativas que se ajusten a diferentes dispositivos y resoluciones.

6.5 Funcionalidades futuras de la aplicación

Entre las funcionalidades que se consideran para versiones posteriores de BUAPP se incluyen: un panel de análisis académico con estadísticas de desempeño, un sistema de notificaciones inteligentes para alertas y recordatorios, espacios colaborativos como foros y chats, elementos de gamificación como insignias y recompensas, y soporte offline con sincronización automática.

6.6 Conclusión

La aplicación de estas mejoras permitirá que BUAPP evolucione hacia una plataforma educativa más robusta, inteligente y centrada en el usuario. De esta manera se garantizará una experiencia académica más personalizada y enriquecedora, consolidando su papel como herramienta innovadora en el ámbito de la educación digital.

Referencias

- Allemang, D., & Hendler, J. (2011). Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL (2nd ed.). Morgan Kaufmann.
- Backlinko. (2023). Search Engine User Behavior Study. Backlinko. https://backlinko.com
- Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2004). OWL Web Ontology Language Reference. W3C Recommendation. https://www.w3.org/TR/owl-ref/
- Berners-Lee, T. (2001). The Semantic Web. Scientific American, 284(5), 34–43. https://doi.org/10.1038/scientificamerican0501-34
- Berners-Lee, T., & Fischetti, M. (1999). Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. Harper San Francisco.
- Curry, D. (2024). Microsoft Teams Revenue and Usage Statistics. Business of Apps. https://www.businessofapps.com/
- Electronic Frontier Foundation (EFF). (2023). Privacy in the Age of AI and Web 3.0. EFF. https://www.eff.org
- Flanagan, D. (2020). JavaScript: The Definitive Guide (7th ed.). O'Reilly Media.
- Freeman, A. (2022). Pro Angular 13. Apress.
- Gackenheimer, C. (2017). Introduction to TypeScript. Apress.
- Google. (2014). Google Classroom Help. Google. https://support.google.com/edu/classroom

- Google. (2024). Google Meet Help. Google. https://support.google.com/meet
- Google. (2024). Angular Documentation. Google Developers. https://angular.io
- Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199–220. https://doi.org/10.1006/knac.1993.1008
- Hendler, J., & Berners-Lee, T. (2010). From the Semantic Web to social machines: A research challenge for AI on the World Wide Web. Artificial Intelligence, 174(2), 156–161. https://doi.org/10.1016/j.artint.2009.11.010
- Hitzler, P. (2020). A review of the semantic web field. Communications of the ACM, 64(2), 76–83. https://doi.org/10.1145/3397512
- Markets Insider. (2019). Global EdTech Investment Reaches $18.66 Billion. Markets Insider.
- Microsoft. (2017). Microsoft Teams Documentation. Microsoft. https://learn.microsoft.com/microsoftteams
- Microsoft. (2024). TypeScript Documentation. https://www.typescriptlang.org
- Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory.
- O'Reilly, T. (2005). What is Web 2.0? O'Reilly Media. https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html
- OpenAI. (2023). Introducing GitHub Copilot. OpenAI. https://openai.com

- Spivack, N. (2022). Web 3.0: The Future of the Internet. Medium.
- Walls, C. (2022). Spring Boot in Action. Manning.
- Wikipedia. (2024). Web 1.0. Wikipedia. https://en.wikipedia.org
- World Wide Web Foundation. (2009, October 18). History of the web. https://webfoundation.org/about/vision/history-of-the-web/
- Google. (1999). Blogger. https://www.blogger.com
- Adamson, B. (2023, 3 de agosto). The history of Netscape Navigator: The once popular web browser everyone forgot about. SlashGear. https://www.slashgear.com/1353587/history-netscape-navigator-web-browser-explained/
- Day in Tech History. (2019, 22 de abril). April 22, 1993: Mosaic browser. https://dayintechhistory.com/dith/april-22-1993-mosaic-browser-2/