Plataforma Web para la Gestión y Promoción de Proyectos de Extensión mediante un Sistema Centralizado

Elías Alberto Alvarado Raxón

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Plataforma Web para la Gestión y Promoción de Proyectos de Extensión mediante un Sistema Centralizado

Trabajo de graduación presentado por Elías Alberto Alvarado Raxón
para optar al grado académico de Licenciado en Ingeniería en Ciencias
de la Computación y Tecnologías de la Información

Guatemala,

2026

Vo.Bo.:

(f) _____
M. Ed. Dennis Aldana

Tribunal Examinador:

(f) _____
M. Ed. Dennis Aldana

(f) _____
PENDIENTE

(f) _____
PENDIENTE

Fecha de aprobación: Guatemala, _____ de _____ de PENDIENTE.

Dedicatoria

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vitae eleifend ipsum, ut mattis nunc. Pellentesque ac hendrerit lacus. Cras sollicitudin eget sem nec luctus. Vivamus aliquet lorem id elit venenatis pellentesque. Nam id orci iaculis, rutrum ipsum vel, porttitor magna. Etiam molestie vel elit sed suscipit. Proin dui risus, scelerisque porttitor cursus ac, tempor eget turpis. Aliquam ultricies congue ligula ac ornare. Duis id purus eu ex pharetra feugiat. Vivamus ac orci arcu. Nulla id diam quis erat rhoncus hendrerit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed vulputate, metus vel efficitur fringilla, orci ex ultricies augue, sit amet rhoncus ex purus ut massa. Nam pharetra ipsum consequat est blandit, sed commodo nunc scelerisque. Maecenas ut suscipit libero. Sed vel euismod tellus.

Proin elit tellus, finibus et metus et, vestibulum ullamcorper est. Nulla viverra nisl id libero sodales, a porttitor est congue. Maecenas semper, felis ut rhoncus cursus, leo magna convallis ligula, at vehicula neque quam at ipsum. Integer commodo mattis eros sit amet tristique. Cras eu maximus arcu. Morbi condimentum dignissim enim non hendrerit. Sed molestie erat sit amet porttitor sagittis. Maecenas porttitor tincidunt erat, ac lacinia lacus sodales faucibus. Integer nec laoreet massa. Proin a arcu lorem. Donec at tincidunt arcu, et sodales neque. Morbi rhoncus, ligula porta lobortis faucibus, magna diam aliquet felis, nec ultrices metus turpis et libero. Integer efficitur erat dolor, quis iaculis metus dignissim eu.

Índice

Lista de figuras

Lista de cuadros

Abreviaturas

ACID Atomicity, Consistency, Isolation, Durability (Atomicidad, Consistencia, Aislamiento y Durabilidad). VII

API Application Programming Interface (Interfaz de Programación de Aplicaciones). VII

AWS Amazon Web Services (Servicios Web de Amazon). VII

CSP Content Security Policy (Política de Seguridad de Contenidos). VII

CSR Client-Side Rendering (Representación del Lado del Cliente). VII

CSRF Cross-Site Request Forgery (Forjamiento de Solicitudes entre Sitios). VII

CSS Cascading Style Sheets (Hojas de Estilo en Cascada). VII

DOM Document Object Model (Modelo de Objetos del Documento). VII

GUI Graphical User Interface (Interfaz Gráfica de Usuario). VII

HTML HyperText Markup Language (Lenguaje de Marcado de Hipertexto). VII

ISR Incremental Static Regeneration (Regeneración Estática Incremental). VII

JSON JavaScript Object Notation (Notación de Objetos de JavaScript). VII

JSX JavaScript XML (Extensión de JavaScript para XML). VII

JWT JSON Web Token (Token Web JSON). VII

MVC Model-View-Controller (Modelo-Vista-Controlador). VII

NoSQL Not Only SQL (No Solo SQL). VII

pnpm Performant Node Package Manager (Gestor de paquetes Node de alto rendimiento). VII

PPR Partial Prerendering (Pre-renderizado Parcial). VII

REST Representational State Transfer (Transferencia de Estado Representacional). VII

SQL Structured Query Language (Lenguaje de Consulta Estructurado). VII

SSG Static Site Generation (Generación de Sitios Estáticos). VII

SSR Server-Side Rendering (Representación del Lado del Servidor). VII

UI User Interface (Interfaz de Usuario). VII

URL Uniform Resource Locator (Localizador Uniforme de Recursos). VII

UVG Universidad del Valle de Guatemala. VII

UX User Experience (Experiencia de Usuario). VII

XSS Cross-Site Scripting (Intercambio de Scripts entre Sitios). VII

Resumen

La modernización de los procesos institucionales a través de plataformas web se ha convertido en un componente esencial para las instituciones de educación superior. La centralización de información y digitalización de trámites permiten una mejor experiencia administrativa y académica. Estas herramientas contribuyen a la optimización de gestión de procesos, fortalece la comunicación entre los actores, y facilita a los estudiantes, directores y organizaciones externas interactuar de manera ordenada, transparente y trazable.

El presente proyecto tiene como objetivo desarrollar una plataforma web que centralice la administración, publicación, postulación y seguimiento de los proyectos de extensión de la Universidad del Valle de Guatemala. Ateniendo problemas actuales de gestión descentralizada: comunicación fragmentada, trámites manuales y baja trazabilidad; limitando la participación estudiantil y reduciendo la relevancia de estos proyectos.

La plataforma se estructurará en dos componentes principales: (1) un frontend orientado a directores, organizaciones externas y estudiantes para facilitar la interacción y gestión de los proyectos; (2) un backend implementado como una interfaz de programación de aplicaciones (API, por sus siglas en inglés) capaz de gestionar usuarios y proyectos. Finalmente, estos dos componentes se integrarán utilizando el API desarrollada para comunicar ambas partes.

Bajo este enfoque, se espera agilizar procesos administrativos vinculados a proyectos de extensión, mejorar la trazabilidad y transparencia de las actividades, y aumentar la accesibilidad de las oportunidades para los estudiantes.

Abstract

The modernization of institutional processes through web platforms has become an essential component for higher education institutions. The centralization of information and digitization of procedures allows for better administrative and academic experience. These tools contribute to the optimization of process management, strengthen communication between stakeholders, and make it easier for students, directors, and external organizations to interact in an orderly, transparent, and traceable manner.

The objective of this project is to develop a web platform that centralizes the administration, publication, application, and monitoring of extension projects at the Universidad del Valle de Guatemala. It addresses current problems of decentralized management: fragmented communication, manual procedures, and low traceability, which limit student participation and reduce the relevance of these projects.

The platform will be structured around two main components: (1) a frontend aimed at directors, external organizations, and students to facilitate interaction and project management; (2) a backend implemented as an application programming interface (API) capable of managing users and projects. Finally, these two components will be integrated using the API developed to communicate between both parties.

Under this approach, it is expected to streamline administrative processes related to outreach projects, improve the traceability and transparency of activities, and increase the accessibility of opportunities for students.

CAPÍTULO 1

Introducción

La extensión universitaria tiene un papel fundamental en el desarrollo estudiantil. Genera un vínculo entre las instituciones de educación superior y la sociedad, ya que permite que los estudiantes y organizaciones externas colaboren en actividades orientadas al desarrollo comunitario y formación integral [1]; sin embargo, muchas veces la gestión de estos procesos se realiza de manera descentralizada, comunicación dispersa y procedimientos manuales los cuales dificultan la organización, difusión y seguimiento de los proyectos disponibles [2]. Debido a estas limitaciones, la participación estudiantil se ve afectada, ya que se reduce la eficiencia administrativa y se genera una experiencia mal estructurada para todos los actores involucrados.

La modernización mediante plataformas digitales ha impulsado nuevas oportunidades y diversos estudios destacan que los sistemas centralizados mejoran la accesibilidad de la información, permiten un mejor control sobre los procesos incluidos y, sobre todo, facilitan la toma de decisiones [3]. En el entorno universitario, la digitalización de flujos de distintos procesos académicos y administrativos es esencial para aumentar la productividad [4]. Actualmente, la Universidad del Valle de Guatemala realiza de forma manual las distintas gestiones para motivar al estudiante a participar en proyectos de extensión universitaria; esto conlleva la difusión inicial de la propuesta del proyecto por parte del director de carrera a través de correo electrónico, aceptación de los estudiantes postulados y validación del trabajo realizado por el estudiante. Terminando en un proceso poco efectivo, trabajoso y, en muchas ocasiones, poco trazable de inicio a fin [5]. Afectado de gran manera a los estudiantes, ya que la mayoría de las ocasiones, el estudiante no llega a participar en muchos proyectos al ignorar por completo su existencia.

En base a esta necesidad, este proyecto propone desarrollar una herramienta web que unifique el ciclo completo de los proyectos de extensión: creación por parte de la organización externa, aprobación por parte del director de Carrera, publicación del proyecto de extensión universitaria, postulación de los estudiantes, seguimiento y validación final por parte del director de Carrera y la organización externa. Una plataforma web integrada no solo permitiría agilizar las gestiones internas, sino que también ofrecer una experiencia más

agradable y accesible para los estudiantes, quienes tendrán la información centralizada para poder participar en actividades formativas [6].

Este proyecto busca contribuir a la modernización institucional y fortalecer los distintos mecanismos de interacción entre la Universidad, su comunidad y organizaciones externas [7].

CAPÍTULO 2

Antecedentes

Antecendentes

CAPÍTULO 3

Justificación

Tradicionalmente, la modalidad de gestión de proyectos de extensión dentro de la Universidad del Valle de Guatemala (UVG) suele depender de procesos manuales, comunicación dispersa (correos electrónicos, documentos físicos, promociones en redes sociales) y canales descoordinados. Provocando demoras, errores, pérdida de información, dificultad para el seguimiento de actividades y baja motivación por parte de los estudiantes. En consecuencia, la participación estudiantil se ve limitada, por ende, el impacto comunitario de estos proyectos de extensión se ve reducido. La modernización de estos procesos con soluciones tecnológicas que centralicen la gestión de los proyectos de extensión se ha vuelto una necesidad.

Un estudio reciente indica que las instituciones de educación superior deben de adaptar sus procesos, incluidos los de extensión, cuando operan bajo un contexto virtual, redefiniendo mecanismos de coordinación con el fin de asegurar inclusión, pertinencia social y calidad organizacional [8].

Además, la consolidación de un sistema centralizado permite una gobernanza institucional más eficiente: facilitando la toma de decisiones informadas, promoviendo transparencia y, sobre todo, optimiza la gestión operativa [9]. Una plataforma web ayuda a reducir dependencias de múltiples sistemas heterogéneos. Lo que contribuye a que se agilicen los procesos de gestión.

La transformación digital en educación superior demuestra que adoptar ecosistemas digitales bien diseñados, trae beneficios tanto para los administradores como para los estudiantes. Los procesos internos son más eficientes, los costos operativos se reducen, y se mejora la accesibilidad y usabilidad para los usuarios finales [10]. Una plataforma web como la propuesta facilitaría a directores un control centralizado, a organizaciones externas visibilidad institucional, y a estudiantes el acceso a oportunidades [7].

La relevancia de modernizar, unificar y profesionalizar la gestión de los proyectos de extensión en la UVG la convierte en una necesidad. Se lograría mejorar la eficiencia administrativa, aumentar la motivación y participación estudiantil, y promover la transparencia en todo el proceso para todos los actores. Diversos estudios, como [6], o artículos relevan-

tes como [9], [10] y [11] respaldan que portales web sirven como canales de comunicación entre todo el personal académico, siendo herramientas de apoyo en la gestión de procesos académicos y administrativos; además, la descentralización tiene un impacto negativo en la asignación de recursos, gobernanza de datos, y retención y éxito de los estudiantes.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar una plataforma web para centralizar la administración, publicación, participación y seguimiento de los proyectos de extensión.

4.2. Objetivos específicos

- Desarrollar el backend mediante una API estructurada que permita la gestión de los proyectos de extensión por parte de directores, estudiantes y organizaciones externas.
- Implementar el frontend como interfaz de usuario de la plataforma web para que directores, estudiantes y organizaciones externas realicen sus gestiones sobre los proyectos de extensión.
- Integrar el frontend y backend estableciendo comunicación bidireccional entre la interfaz de usuario y la API desarrollada, logrando así la plataforma web unificada.

CAPÍTULO 5

Alcance

Podemos usar Latex para escribir de forma ordenada una fórmula matemática.

CAPÍTULO 6

Marco teórico

6.1. Extensión Universitaria

La extensión universitaria es un componente fundamental de la misión en las universidades, pues busca llevar el conocimiento académico y científico de su comunidad académica hacia la sociedad en general. Esta se puede realizar en diversas formas, tales como: conferencias, cursos cortos, talleres, proyectos de servicio comunitario, consultorías, entre otros. Cada una de estas actividades comparten el conocimiento y recursos adquiridos en la Universidad con la sociedad, contrubuyendo así al desarrollo social, cultural, económico y científico [12].

Además, la extensión universitaria representa un compromiso de las universidades con la sociedad, apoyando el fácil acceso al conocimiento y fomentando la participación activa con el desarrollo integral de las comunidades. Esta actividad ha sido fundamental en distintas instituciones educativas, promoviendo una educación inclusiva y relevante para las distintas necesidades sociales [13].

Bajo este contexto, el Artículo 2 del Reglamento de Extensión de la Universidad del Valle se alinea con esta visión y compomiso, indicando lo siguiente:

> *."En la Universidad del Valle de Guatemala se define como Extensión al trabajo realizado en actividades orientadas al desarrollo social, cultural, económico y tecnológico de la comunidad."* [5]

6.1.1. Extensión Universitaria en el Desarrollo Social

El desarrollo social requiere que inicialmente exista un desarrollo personal de cada individuo que compone esa sociedad, pasando por un proceso de formación y adquisición o mejora de sus capacidades. La Universidad se considera como un agente clave en este proceso, un ente de transformación social. Puesto que su comunidad académica serán los futuros

profesionales, que en el desarrollo de su trabajo profesional, tendran la capacidad de influir directa o indirectamente en su entorno social [14, 15].

La Educación y la Universidad como institución educativa, tiene un papel fundamental en el desarrollo social, pues su labor de gestión del conocimiento y formación de profesionales, contribuye al desarrollo de capacidades, destrezas y habilidades, que permiten a los individuos participar activamente en la solución de problemáticas existentes en su entorno social [14]. Debido a esto, la extensión universitaria siendo pieza esencial, expresa un compromiso con la sociedad, siendo un sistema abierto y flexible que atienda necesidades de la comunidad [15].

6.2. Transformación Digital en la Educación Superior

6.2.1. Era Digital

La era digital se caracteriza por la integración de tecnologías digitales en todos los aspectoss de la vida. Donde predominan los servicios y la *«experiencia de usuario»* sobre la producción de bienes. Dichos servicios incorporan *«inteligencia»* en los productos y servicios digitales, rediseñando procesos y en varias ocasiones, se convierte la provisión de bienes en una prestación de servicios[16].

6.2.2. Transformación Digital

Se refiere al proceso donde una entidad permite la integración de tecnologías digitales, por las cuales se puede dar respuestas estratégicas con valor agregado gracias a la innovación tecnológica. En el contexto universitario, la Transformación Digital impacta de lleno, puesto que genera una redefinición del modelo de Universidad [17].

6.2.3. Evolución de la Gestión Universitaria

Las unversidades, al igual que varias entidades, han tenido que adaptarse a los cambios tecnológicos y sociales que la Era Digital ha traído. Esto ha llevado a una evolución en la gestión universitaria, favoreciendo naturalmente a la eficiencia, transparencia y accesibilidad en los procesos administrativos y académicos. En su mayoría, esta evolución ha sido impulsada por la propia necesidad de las unversidades de mantenerse relevantes, competitivas e innovadoras frente a las demás instituciones como de su comunidad académica.Contemplando una sistematización de procesos administrativos, académicos y técnicos. No solo haciendo referencia a la implementación de nuevos procesos, sino a la innovación y mejoramiento de los procesos actuales, aún cuando estos ya sean digitales [16, 18].

De acuerdo a Chinkes y Julien, la transformación digital en instituciones de educación superior es una necesidad la cuál debe de ser abordada con una visión crítica y bajo las particularidades de cada institución [16].

6.2.4. Aplicación Web

Una aplicación Web es un programa informático o software el cuál se ejectua en Internet, sin la necesidad de que exista una instalación local en el dispositivo del usuario, el uso del navegador web basta para poder acceder a esta. Puede comprenderse como un conjunto de aplicaciones autónomas modulares que se pueden consumir desde una máquina a través de una red, como Internet. Estas aplicaciones permiten comunicación e intercambio de datos entre diferentes sistemas y plataformas, facilitando la interoperabilidad máquina a máquina sobre una red [19]. Permitiendo acceso a la información de manera rápida y sencilla, así como realizar diversas interacciones que la aplicación Web permita [20].

Beneficios de una aplicación web en procesos universitarios

Los beneficios de implementar una aplicación web en procesos universitarios son múltiples, entre los cuales se pueden destacar:

- **Ahorro de costos:** La migración a una aplicación web puede reducir costos operativos al tener menor dependencia de recursos físicos.
- **Eficiencia operativa:** La agilización de procesos aumenta significativamente la eficiencia operativa, reduciendo tiempos de espera y entregando información en tiempo real.
- **Innovación:** Frente a la competencia, tener un sistema digitalizado permite a las universidades demostrar su dominio y adopción de tecnologías emergentes.

Según Urmeneta et al., la implementación de servicios web en instituciones educativas permite mejorar la eficiencia, accesibilidad y calidad de los servicios ofrecidos a estudiantes, docentes y personal administrativo. Abordando ineficiencias detectadas en los métodos manuales tradicionales, un sistema permite centralizar y agilizar el proceso de presentación, revisión y aprobación de resultados. Además, ayuda a conectar y comunicar correctamente a todos los actores. [7].

Arquitectura cliente-servidor

Esta arquitectura se basa en la división clara de responsabilidad. Por una parte se tiene el cliente, que es la parte de la aplicación con la que un usuario interactúa directamente, y por otra parte se tiene el servidor, que es la parte encargada de realizar todas las operaciones lógicas necesarias para hacer funcionar la aplicación. En esta arquitectura, el cliente envía solicitudes al servidor, el servidor entonces procesa la solicitud bajo lógica de negocio y procesamiento de información y finalmente devuelve una respuesta al cliente [7, 20].

6.3. *Frontend*

El término *Frontend*, hace referencia al clienta el cuál es la interfaz gráfica de usuario (GUI, por sus siglas en inglés), la cual es la parte visual con la que el usuario interactúa directamente. En pocas palabras, es la parte que permite la interacción del usuario con la aplicación. Bajo el contexto Web, la GUI vista por el usuario desde su navegador se le conoce como modelo de objetos del documento (DOM, por sus siglas en inglés) [21].

6.3.1. Arquitecturas

Para el *Frontend* la arquitectura se centra en el lugar donde se realiza el procesamiento y entrega del contenido. La arquitectura influirá no solo en la experiencia de usuario, sino también en el rendimiento, seguridad y escalabilidad [22].

Single Page Application (SPA)

Una SPA es cuando la aplicación se renderiza y corre completamente en el navegador del cliente. Permitiendo una experiencia de usuario fluida e interactiva, no requiere recargar para actualizar el contenido, ya que realiza llamadas al *Backend* de manera asíncrona [23].

Progressive Web App (PWA)

Una PWA continúa con el principio de SPA, añadiendo funcionalidades que permiten simular comportamientos que solamente tiene una aplicación nativa de escritorio o móvil, haciendo posible ejecutar servicios adicionales aún cuando el ambiente donde se ejecuta la aplicación es el navegador [23].

6.3.2. Seguridad

La seguridad en el *Frontend* se basa en estrategias de validación de entradas, desactivación de configuraciones predeterminadas y la implementación de políticas de seguridad de contenido (CSP, por sus siglas en inglés) [21].

6.3.3. Renderizado Web

El Renderizado Web es el proceso mediante el cual el navegador interpreta y transforma el código (HTML, CSS, JavaScript) en una página visual e interactiva para el usuario [24]. Existen varias estrategias de renderizado web, entre las cuales se encuentran :

Static Site Generation (SSG)

SSG es una técnica de renderizado donde se preprocesan las páginas en el momento de la compilación, dando como resultado archivos HTML estáticos que permiten enviarlos de forma más rápida y eficiente. Ideal para páginas con contenido que cambie con muy poca frecuencia.

Beneficios:

- Excelente rendimiento SEO.
- Reducción de la carga del servidor.
- Carga de página rápida.
- Costes de infraestructura mínimos.

Consideraciones:

- Páginas con gran número de páginas, aumentan el tiempo de compilación.
- Una actualización de contenido requiere una compilación completa.

Incremental Static Regeneration (ISR)

ISR permite la actualización de páginas espefícicas sin necesidad de compilar todo de nuevo. Combinando los beneficios de SSG con una actualización más fluida y rápida. Ideal cuando SSG no se da a basto debido a su tiempo de compilación.

Beneficios:

- Carga de página rápida.
- Permite actualizaciones bajo demanda, sin la necesidad de compilar todo.
- Adaptación a gran número de páginas.

Consideraciones:

- Contoles de caché más complejos.

Server-Side Rendering (SSR)

SSR es una técnica de renderizado donde el servidor genera un HTML completo en cada petición y se lo envía al cliente tal cual, lo cuál permite contenido en tiempo real y personalizado. Ideal para aplicaciones con contenido dinámico y cambio constante.

Beneficios:

- Contenido actualizado.
- Mejor rendimiento SEO.

Consideraciones:

- Mayor carga en el servidor.
- Tiempos de carga inicial más lentos.

Client-Side rendering (CSR)

CSR es una técnica donde el servidor genera un HTML básico, dejando que el cliente sea el encargado de renderizar el contenido completo mediante JavaScript. Ideal para aplicaciones web interactivas y con alta dinámica en su contenido.

Beneficios:

- Interacciones en tiempo real con datos externos.
- Reducción de la carga del servidor.
- Experiencia de usuario fluida.

Consideraciones:

- Carga inicial más lenta.
- Se pierde rendimiento SEO.
- Gestión rigurosa del estado de la aplicación.

Partial Prerendering (PPR)

PPR es una técnica que se encuentra en fase experimental, pero busca combinar los beneficios de las demás estrategias. Para ello, prerenderiza desde el servidor cualquier parte estática de la página y luego transmite el contenido dinámico basándose en los límties de React Suspense. Al estar en fase experimental, su uso ideal aún no está definido, así como sus limitaciones.

Beneficios:

- Carga inicial rápida (como SSG).
- Contenido dinámico actualizado (como SSR/CSR).
- Mejor rendimiento SEO.
- Reducción de gastos generales de desarrollo.

[24]

6.4. *Backend*

El término *Backend*, hace referencia al servidor que es el encargado de administrar la funcionalidad general de la aplicación. Esto incluye la lógica de negocio y la gestión del almacenamiento persistente de datos. Además, el *Backend* es el responsable de manejar cualquier solicitud proveniente del *Frontend*, donde la procesa y devuelve una respuesta [21].

6.4.1. Arquitecturas

Model-View-Controller (MVC)

El patrón de diseño Modelo-Vista-Controlador (MVC, por sus siglas en inglés) es una arquitectura que separa la aplicación en capas, dividiendo las aplicaciones en tres componentes lógicos [25, 26]:

- **Modelo:** También conocido como la capa de almacenamiento persistente, esta componente tiene como función almacenar y gestionar los datos guardados en bases de datos (SQL, NoSQL), garantizando una correcta persistencia de datos.
- **Vista:** También conocido como la capa de presentación, es la parte de la aplicación con la que el usuario interactura directamente. Este componente incluye la interfaz de usuario y la experiencia de usuario (UI/UX), y su función principal es presentar información y reunir datos de entrada.
- **Controlador:** También conocido como la capa de aplicación, es la parte de la aplicación que maneja la lógica de negocio, procesamiento de datos, seguridad y comunicación con la base de datos.

Este patrón de diseño, generalmente utilizado en aplicaciones Web *client-server* o sistemas distribuidos, ofrece muchas ventajas, entre ellas la facilidad de mantenimiento, escalabilidad y reutilización. Ya que cada componente se puede gestionar de manera independiente [25].

Arquitecturas por capas

La arquitectura por capas se caracteriza por dividir la aplicación en capas lógicas que tienen funciones únicas. La organización se basa en jerarquía, estructura y separación de responsabilidades según los componentes técnicos del sistema. Cada capa se comunica únicamente con una capa adyacente. Esto permite que el flujo se propague de manera ordenada de una capa a otra de manera secuencial. Busca aislamiento y modularidad, garantizando que las modificaciones de una capa no influyan ni afecten a las demás directamente [27]. Las capas más comunes son:

- **Capa de presentación:** Es la responsable de la interacción directa con el usuario, incluyendo validaciones de entrada, presentación de resultados y comunicación con la capa de negocio.

- **Capa de negocio:** Es la responsable de reglas de negocio, lógica en procesamiento y orquestación de servicios.
- **Capa de persistencia:** Es la responsable de la gestión de datos, incluyendo comunicaciones con bases de datos, almacenamiento y recuperación de datos.
- **Capa de Base de datos:** Es la responsable de almacenar físicamente los datos.

6.4.2. Seguridad

La seguridad en el *Backend* se basa en la seguridad del almacenamiento de datos y el tránsito. Esto incluye encriptación de datos, controles de acceso, seguridad de sesión y protección contra amenzas comunes tales como: inyección SQL, *cross-site scripting* (XSS) y *cross-site request forgery* (CSRF) [21].

6.5. Tecnologías de Implementación

6.5.1. *Amazon Web Services* (AWS)

AWS es una plataforma de *Cloud Computing (Computación en la Nube)* más completa y adoptada a nivel mundial. Permite a empresas y desarrolladores acceder a una amplia gama de servicios de computación. Entre los cuales destacan [28]:

- Servicios de computación: *Amazon Elastic Compute Cloud* (EC2), AWS Lambda, Amazon Lightsail, entre otros. Que permiten ejecutar servidores virtuales, aplicaciones sin servidor, contenedores, etc.
- Almacenamiento: *Amazon Simple Storage Service* (S3), *Amazon Elastic Block Store* (EBS), *Amazon Glacier*, entre otros. Que permiten almacenar cualquier cantidad de datos, archivos u objetos de manera segura y escalable.
- Bases de datos: *Amazon Relational Database Service* (RDS), *Amazon DynamoDB*, Amazon Aurora, entre otros. Que permiten iniciar, gestionar y escalar bases de datos relacionales o no relacionales de manera sencilla y eficiente.

Amazon Simple Storage Service (S3)

S3 es un servicio de Amazon para almacenamiento de objetos basado en la nube. Porporciona disponibilidad, seguridad y alta escalabiliadd. Permite conexiones con otros servicios de AWS, así como aplicaciones de terceros, lo que facilita la integración de esta tecnología a proyectos como aplicaciones móviles, aplicaciones web, entre otros. Entre sus principales usos, se encuentra el almacenamiento estático de archivos, copias de seguridad, archivos multimedia, entre otros [29].

6.5.2. Docker

Docker es una plataforma de código abierto que permite crear, desplegar y ejecutar aplicaciones dentro de contenedores. Un Contenedor es una unidad de software ligera y portátil capaz de empaquetar código, librerías, dependencias y configuraciones esenciales para ejecutar una aplicación. Por lo que, actúa como una máquina virtual virtualizada. Altamente utilizado en el desarrollo de aplicaciones modernas, incluyendo aplicaciones web, gracias a su gran control y consistencia en el entorno de ejecución, permitiendo un mayor control sobre la infraestructura y despliegues [30].

Orquestación con *Docker Compose*

Docker Compose es una herramienta que facilita la configuración, interacción y ejecución de múltiples contenedores Docker. Permite desplegar aplicaciones complejas con múltiples servicios (como bases de datos, servidores web, etc.) de manera rápida y eficiente [30].

6.5.3. PostgreSQL

PostgreSQL es un sistema de base de datos relacional de código abierto que amplía las capacidades de SQL tradicional. Combinando numerosas funciones avanzadas que logran almacenar y escalar de forma segura las cargas de trabajo de datos más complejas. Gracias a su arquitectura, fiabilidad, integridad, extensibilidad y gran comunidad, se ha convertido en una de las bases de datos más utilizadas y sólidas. Cumpliendo con ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) y su alta gama en tipos de datos, funcionalidades internas y extensiones, lo hacen ideal para la mayoría de aplicaciones, incluyendo aplicaciones web [31].

6.5.4. Node.js

Node.js es un entorno de ejecución de código abierto para JavaScript que utiliza el motor V8 de Google Chrome, ideal para construir aplicaciones web escalables, alto rendimiento, y lo más importante, no bloqueante. Gracias a que utiliza un modelo basado en eventos de E/S (entrada/salida) lo que permite manejar múltiples conexiones simultáneamente sin bloquearse. Permitiendo así crear servidores web, APIs RESTful, aplicaciones en tiempo real (chat, juegos, etc.) [32].

6.5.5. JavaScript

JavaScript es un lenguaje de programación utilizado principalmente para el desarrollo de páginas web interactivas. Puede ser utilizado tanto en el *Frontend* como en el *Backend*, gracias a entornos de ejecución como *Node.js*. Es un lenguaje interpretado, orientado a objetos y basado en prototipos y multiparadigma; incluso tiene acceso al *DOM* de la página web, lo cuál permite manipular elementos HTML [33].

6.5.6. HTML

El lenguaje de marcas de hipertexto (HTML, por sus siglas en inglés) es la base de la mayoría de las páginas web. Define la estructura y contenido de una página web por medio de etiquetas y atributos que indican al navegador qué y cómo mostrar [33].

6.5.7. CSS

Las hojas de estilo en cascada (CSS, por sus siglas en inglés) es un lenguaje de reglas de estilo que ayudan a definir estilos al contenido HTML. Estas reglas indican al navegador cómo renderizar los elementos HTML [33].

6.5.8. TypeScript

TypeScript es un superconjunto de JavaScript, lo que significa que un código JavaScript válido también es un código TypeScript. TypeScript perfecciona JavaScript al agregar tipos en la sintaxis, lo que permite a herramientas de edición de código detectar errores en tiempo de desarrollo, también conocido como *Type-safe (Tipado Seguro)* [33].

6.5.9. React

React es una biblioteca JavaScript de código abierto creada por Facebook (Meta). Desde su lanzamiento en 2013, e ha convertido en una de las bibliotecas más populares para el *Frontend*, superando a otras tecnologías como Angular y Vue.js. Gracias a su enfoque basado en componentes, React permite construir aplicaciones Web altamente dinámicas y escalables. React crea un DOM virtual en cada componente (copia del DOM) para compararlo con el estado del DOM real y aplicar el cambio solamente al elemento actualizado, mejorando significativamente el rendimiento de la aplicación al no tener que renderizar toda la página nuevamente. Cada componente es una fusión de la estructura HTML y la lógica JavaScript, lo que se conoce como sintaxis JSX (JavaScript XML) [34].

Beneficios

- El DOM virtual ayuda a ahorrar recursos y tráfico, mejorando el rendimiento.
- La reutilización de componentes permite un desarrollo más limpio y estructurado.
- Su estructura basada en comopnentes ayuda a la hora de escalar o darle mantenimiento a la aplicación.
- Integración sencilla con otras bibliotecas.

Consideraciones

- La biblioteca en sí puede aumentar el tamaño total de la aplicación.

- Evoluciona constantemente, por lo que requiere aprendizaje continuo.
- Al no ser un *Framework*, requiere la integración de otras bibliotecas para funcionalidades adicionales.

6.5.10. TanStack

TanStack es una colección de bibliotecas de código abierto de alto rendimiento para construir aplicaciones web modernas. Gracias a su versatilidad, puede integrarse a un proyecto sin importar el *framework*, pero es comúnmente utilizado con React, Vue, Solid y Angular; ya que sus paquetes son especialmente para el *Frontend* [35].

TanStack Router

TanStack Router es una biblioteca de enrutamiento *type-safe* basado en el sistema de archivos. Compartiendo muchas características con React Router, pero con soporte SSR, admitiendo múltiples tipos de rutas diferentes. Cada ruta puede tener sus propios *loaders*, *actions* y *error boundaries* [35].

TanStack Query

TanStack Query, antes conocido como React Query, es una biblioteca de gestión de estado asíncrono *type-safe* para *fetching*, almacenamiento en caché, sincronización y actualización de datos del servidor. Se basa en tres conceptos principales [36]:

- ***Queries*:** Para *fetching* y lectura de datos.
- ***Mutations*:** Para crear, actualizar o eliminar datos.
- ***Query Client*:** Punto central de coordinación.

Además, ofrece características como TypeScript integrado, soporte para *pagination* e *infinite scrolling* (altamente optimizados) y utilidades *server-state* [36].

TanStack Form

TanStack Form es una biblioteca para la gestión de estado de formularios *type-safe* de alto rendimiento. Diseñada para ser flexible y adaptable a cualquier caso de uso, permite la creación de formularios complejos con validaciones personalizadas, así como el manejo de errores. Además, permite integrar fácilmente cualquier componente de interfaz de usuario propia o de terceros [37].

6.5.11. *Material UI* (MUI)

MUI es una biblioteca de componentes de interfaz de usuario de código abierto para React, diseñada bajo principios de *Material Design* de Google. Esta biblioteca proporciona una gran variedad de componentes preconstruidos personalizables y adaptables en cualquier aspecto gracias a su robusto sistema de temas. Compatible con TypeScript, hace un desarrollo más fluido y seguro [38].

6.5.12. Axios

Axios es un Cliente HTTP basado en promesas para el navegador y Node.js. Gracias a que es Isomórfico (Código), puede ejecutarse tanto en el *Frontend* como en el *Backend*. Una de sus grandes ventajas es que transforma de manera automática las respuestas del servidor a formato JSON, también que permite interceptar peticiones y/o respuestas para realizar acciones personalizadas, además soporta *timeouts*, cancelación de peticiones y manejo de errores [39].

6.5.13. Zustand

Zustand es una biblioteca ligera de gestión de estado para React. Funciona mediante *hooks* los cuales permiten crear estados globales sin la necesidad de componentes proveedores de contexto. Busca minimizar tanto el código como la complejidad, ofreciendo estados inmutables y actualizaciones predecibles con un rendimiento óptimo. Además, es compatible con TypeScript [40].

6.5.14. *JSON Web Token* (JWT)

JSON Web Token es un estándar abierto (RFC 7519) el cuál ayuda a definir de manera compacta y autónoma una forma de transmitir información de manera segura entre partes como un JavaScript Object Notation (JSON, por sus siglas en inglés). Siendo segura al contener una firma digital, lo que permite verificar la integridad de su contenido, esta firma puede ser creada utilizando un secreto o un par de claves pública/privada utilizando RSA o ECDSA. En general, un JWT se utiliza para la autenticación y autorización dentro de una aplicación. Cada solicitud que un usuario logueado realice, llevará su JWT, permitiendo así acceder a los recursos y acciones protegidas por la aplicación [41].

En su forma compacta, un JWT se compone de tres partes codificadas en Base64Url separadas por puntos (.), las cuales son:

- *Header*: Suele constar de dos partes, el tipo de token (JWT) y el algoritmo de firma utilizado, como HMAC SHA256 o RSA.
- *Payload*: Contiene las reclamaciones, son declaraciones sobre una entidad (el usuario) e información adicional. Esta información puede ser datos del usuario, permisos, entre otros.

- *Signature*: La firma se crea tomando el contenido codificado del *Header* y el *Payload* y un secreto, utilizando el algoritmo especificado en el *Header*. Esta firma garantiza que el JWT no ha sido alterado después de su emisión.

6.5.15. *Application Programming Interface* (API)

Una interfaz de programa de aplicación (API, por sus siglas en inglés) define las reglas a seguir para la comunicación entre sistemas de software, a este conjunto de reglas se le conoce como *Endpoint*. En pocas palabras, una API es un conjunto de definiciones que ayudan a estandarizar la comunicación con cierto sistema (servidor), permitiendo que distintas aplicaciones (cliente) puedan interactuar con dicho sistema sin la necesidad de conocer su implementación interna [42].

6.5.16. *Representational State Transfer* (REST)

La transferencia de estado representacional (REST, por sus siglas en inglés) es una arquitectura de software que define un conjunto de condiciones como intefaz uniforme, operaciones HTTP estándar, ausencia de estado y representaciones intercambiables de información. Una API que cumple con estas condiciones se le conoce como API RESTful [42].

Solicitud

Para que una API RESTful responda correctamente hacia una solicitud HTTP, la solicitud debe de contener los siguientes componentes principales [42]:

- ***Endpoint*:** Generalmente, es una URL (siglas en inglés para Localizador Uniforme de Recursos) que identifica un recurso único en el servidor.
- **Método:** Define la acción que se desea realizar sobre el recurso. Los métodos HTTP más comunes son:
  - **GET:** Recuperar datos de un recurso.
  - **POST:** Crear un nuevo recurso.
  - **PUT:** Actualizar un recurso existente.
  - **DELETE:** Eliminar un recurso.
- **Encabezados:** Son metadatos que proporcionan información como el formato de la solicitud, tipo de autenticación , entre otros.
- **Datos**: Para los métodos POST y PUT, y otros métodos, la solicitud puede incluir datos.
- **Parámetros:** Brindan al servidor más detalles sobre la solicitud como: filtros, cookies, etc.

Respuesta

Los principios REST indican que la respuesta del servidor contenga los siguientes componentes principales [42]:

- **Línea de estado:** Contiene un código de estado HTTP de tres dígitos que indica el resultado de la solicitud.
- **Cuerpo del mensaje:** De la mano con los encabezados de la respuesta, contiene la representación del recurso.
- **Encabezados:** Al igual que en la solicitud, estos son metadatos que brindan contexto sobre la respuesta, incluyendo información como el servidor, codificación, etc.

6.5.17. *Performant npm* (PNPM)

PNPM es un gestor de paquetes para Node.js que se enfoca en ahorro de espacio en disco y mejora la velocidad de instalación gracias a su enfoque de almacenamiento compartido mediante almacenamiento cnetralizado y enlaces simbólicos, el cuál permite que múltiples proyectos compartan dependencias comunes sin la necesidad de duplicar archivos [43].

6.5.18. Postman

Postman es una plataforma de colaboración para el desarrollo, pruebas, documentación y monitoreo de APIs. Es de las más utilizadas gracias a su interfaz intuitiva y sus múltiples funcionalidades como el envío de solicitudes HTTP/HTTPS con sus respectivos encabezados, parámetros, cuerpos y autenticación, así como el manejo de las respuestas. Soporta solicitudes RESTful, SOAP, GraphQL, entre otros. Además, permite la creación de colecciones para organizar las solicitudes y la configuración de entornos para poder gestionar variables de entorno y saltar de entornos de desarrollo a producción de manera sencilla [44].

CAPÍTULO 7

Metodología

En este capítulo se describe la metodología empleada para el desarrollo de la plataforma web, desde la fase de análisis de requerimientos hasta la unificación de todas las partes del proyecto.

La metodología aplicada se fundamentó en un enfoque estructurado de cinco fases secuenciales e interrelacionadas, que permitieron una gestión eficiente del proyecto y una integración fluida de los distintos productos finales de cada fase.

7.1. Fase 1: Recolección de Requerimientos

La primera fase se centró en la extracción de requerimientos técnicos y funcionales a partir del análisis de los diseños de Interfaz de Usuario previamente elaborador por Andrés Rodríguez en [45].

Esta fase tiene como objetivo principal identificar y documentar las reglas de negocio, flujos de información, roles, restricciones y validaciones necesarias para que la plataforma opere de manera correcta.

7.1.1. Definición de módulos funcionales

Con el fin de identificar acciones, validaciones y flujos de navegación se realizó un análisis detallado de cada una de las páginas, componentes y formularios presentes en los diseños de interfaz. Este análisis permitió extraer los módulos funcionales que permitirán un soporte alineado y coherente con el fin de la plataforma. Los módulos funcionales identificados fueron:

- **Módulo de Autenticación:** Este módulo se encarga del registro, inicio de sesión, recuperación de contraseña, *logout* y gestión de sesiones de los usuarios. Además, la asignación automática de roles a los usuarios.

- **Módulo de Usuarios:** Este módulo se encarga de la actualización de información, cambio de contraseña, aprobación de cuentas para usuarios con roles específicos, vista de perfiles.
- **Módulo de Proyectos:** Este módulo se encarga de la creación, edición, eliminación y visualización de proyectos, así como la gestión de sus estados y la aprobación de proyectos por parte de usuarios con roles específicos.
- **Módulo de Postulaciones:** Este módulo se encarga de la gestión de postulaciones a proyectos, incluyendo la postulación, revisión y aprobación de postulaciones por parte de usuarios con roles específicos. Además, de permitir la visualización de historial de postulaciones.
- **Módulo de Horas:** Este módulo se encarga de la gestión de horas dedicadas a proyectos, incluyendo el registro, revisión y aprobación de horas por parte de usuarios con roles específicos. Además, de permitir la visualización de historial de horas.
- **Módulo de Archivos:** Este módulo se encarga de la gestión de archivos relacionados con usuarios, proyectos y horas, incluyendo la creación y eliminación de archivos, así como la visualización de archivos relacionados a cada entidad.

7.1.2. Análisis de formularios y mapeo de datos

Alineado con los módulos funcionales, cada formulario presente en los diseños de interfaz fue desglosado para identificar los campos de entrada, sus tipos de datos y validaciones necesarias. Permitiendo establecer un mapeo claro entre los datos requeridos por la plataforma para cada acción posible.

7.1.3. Definición de roles y permisos

A partir de los diseños de interfaz fue sencillo identificar los distintos roles de usuario y sus respectivas acciones permitidas. De esta manera, se definieron los roles y permisos por usuario, resultando en la siguiente clasificación:

- **Todos los usuarios:**
  - Login y registro.
  - Verificación de correo electrónico.
  - Restablecimiento de contraseña.
  - Visualización de proyectos.
  - Visualización de perfiles de usuario.
  - Visualización de archivos relacionados a cada entidad.
  - Visualización de las motivaciones.
  - Actualización de información personal y cambio de contraseña.
- **Estudiantes:**

- Postulación a proyectos.
- Participación en proyectos.
- Abandonar proyectos.
- Registro de horas trabajadas con evidencias.
- Visualización de horas aprobadas en proyectos.
- Visualización de historial de postulaciones y horas.

- **Organizaciones externas:**
  - Creación, edición y eliminación de proyectos con archivos (recursos) ligados.
  - Iniciar, finalizar y cancelar proyectos propios.
  - Visualización de postulaciones a proyectos propios.
  - Aprobación de postulaciones de proyectos propios.
  - Visualización de participantes de proyectos propios.
  - Eliminar participantes de proyectos propios.
  - Aprobación de horas trabajadas por estudiantes en proyectos propios.
- **Directores:**
  - Lo mismo que organizaciones externas.
  - Aprobación de proyectos enviados por organizaciones externas.
  - Aprobación de cuentas de organizaciones externas.

7.1.4. Modelación de la Base de Datos

Con el fin de garantizar integridad, seguridad y escalabilidad de los datos, se optó por una base de datos relacional. La Figura 1 muestra el diagrama entidad-relación de la base de datos.

Estandarización de estados lógicos

Con el fin de estandarizar estados lógicos, evitando inconsistencias, se definieron tipos enumerados (ENUMs) para los roles de los usuarios y estados de proyectos, postulaciones y horas.

Entidades

- **Campuses:** Almacena los distintos campus universitarios disponibles.
- **Faculties:** Almacena la información de las facultades académicas.
- **Careers:** Almacena las carreras académicas disponibles.

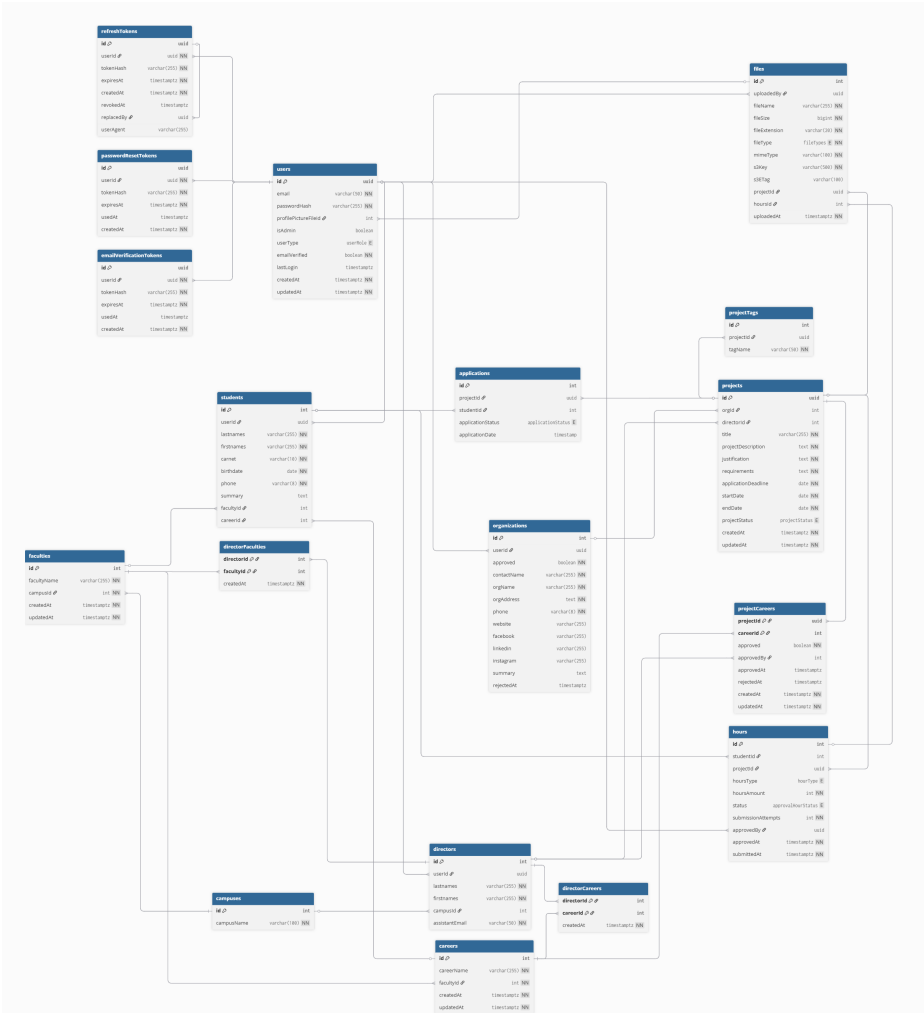

Figura 1: Diagrama Entidad-Relación de la Base de Datos

- **Users:** Almacena la información de los usuarios registrados, principalmente incluyendo información de autenticación y rol.
- **Students:** Almacena la información académica y personal de los estudiantes.
- **Directors:** Almacena la información académica de los directores académicos.
- **DirectorFaculties:** Tabla intermedia que permite vincular directores y facultades.
- **DirectorCareers:** Tabla intermedia que permite vincular directores y carreras.
- **Organizations:** Almacena la información de las organizaciones externas.
- **RefreshTokens:** Almacena los tokens de autenticación persistente utilizados para renovar sesiones de usuario.
- **PasswordResetTokens:** Almacena tokens temporales de un solo uso para el proceso de recuperación de contraseña.
- **EmailVerificationTokens:** Almacena los tokens utilizados para verificar la dirección de correo electrónico de los usuarios.
- **Projects:** Almacena los proyectos disponibles en la plataforma. Incluyendo información como título, descripción, fechas, estado, requisitos, entre otros.
- **ProjectCareers:** Tabla intermedia que permite vincular proyectos con carreras.
- **ProjectTags:** Tabla intermedia que permite vincular etiquetas con proyectos.
- **Applications:** Almacena las postulaciones de los estudiantes a proyectos específicos, incluyendo el estado de la aplicación.
- **Hours:** Almacena las horas realizadas por los estudiantes en los proyectos, incluyendo el tipo de hora, cantidad y estado de aprobación.
- **Files:** Almacena los archivos asociados al sistema, como fotografías de perfil, recursos de proyectos o evidencias de horas.

Relaciones

- Un campus puede tener múltiples facultades, pero cada facultad pertenece a un único campus.
- Una facultad puede tener varias carreras, pero cada carrera pertenece a una sola facultad.
- Un usuario puede estar asociado a un único estudiante, director u organización.
- Un director puede tener varias facultades.
- Un director puede tener varias carreras.
- Una organización puede crear múltiples proyectos, pero cada proyecto pertenece a una única organización (cuando aplica).

- Un director puede crear múltiples proyectos, pero cada proyecto pertenece a un único director (cuando aplica).
- Un proyecto puede estar disponible para varias carreras y una carrera puede participar en múltiples proyectos.
- Un proyecto puede recibir múltiples postulaciones, pero cada postulación corresponde a un único proyecto.
- Un estudiante puede postularse a múltiples proyectos, pero cada postulación pertenece a un único estudiante.
- Un estudiante puede registrar múltiples transacciones de horas en distintos proyectos.
- Un proyecto puede tener múltiples registros de horas asociados.
- Un usuario puede subir múltiples archivos.
- Un proyecto puede tener múltiples archivos asociados como recursos.
- Un registro de horas puede tener múltiples archivos asociados como evidencias.
- Un usuario puede tener múltiples tokens de autenticación, verificación o recuperación asociados.

7.1.5. Definición de *Endpoints*

A partir de los módulos funcionales y el análisis de formularios se estructuró la arquitectura *RESTful* de la API. Definiendo contratos de comunicación claros, incluyendo rutas, verbos HTTP, parámetros de entrada, formatos de respuesta y códigos de estado.

Autenticación

Ruta base: `/auth`. Este conjunto de *endpoints* se encarga de la autenticación, recuperación de contraseña, verificación de correo electrónico, gestión de sesiones y tokens de autenticación.

- `POST /auth/register`: Registro de usuarios.
- `POST /auth/login`: Inicio de sesión.
- `POST /auth/refresh`: Renovación de tokens de autenticación.
- `POST /auth/verify-email`: Verificación de correo electrónico.
- `POST /auth/forgot-password`: Solicitud de restablecimiento de contraseña.
- `POST /auth/reset-password`: Confirmación de restablecimiento de contraseña con token.
- `POST /auth/logout`: Cierre de una sesión específica.

- `POST /auth/logout-all`: Cierre de todas las sesiones activas.
- `GET /auth/sessions`: Listado de las sesiones activas.

Usuarios

Ruta base: `/users`. Este conjunto de *endpoints* gestiona la información de los usuarios registrados, incluyendo consulta de perfiles, actualización de datos, cambio de contraseña y eliminación de cuentas.

- `GET /users/`: Listar estudiantes.
- `GET /users/me`: Obtener perfil propio.
- `GET /users/:id/profile`: Obtener perfil público de un usuario.
- `GET /users/:userId`: Obtener usuario por ID (administrador).
- `PUT /users/:id`: Actualizar información de usuario.
- `PUT /users/:id/change-password`: Cambiar contraseña.
- `DELETE /users/:id`: Eliminar usuario.

Proyectos

Ruta base: `/projects`. Este conjunto de *endpoints* gestiona la creación, consulta, actualización, aprobación y eliminación de proyectos.

- `GET /projects`: Listar proyectos con filtros.
- `GET /projects/my-projects`: Listar proyectos propios.
- `GET /projects/pending-approvals`: Listar proyectos pendientes de aprobación o rechazo.
- `GET /projects/:projectId`: Obtener proyecto por ID.
- `POST /projects`: Crear proyecto.
- `PUT /projects/:projectId`: Actualizar proyecto.
- `POST /projects/:projectId/approve`: Aprobar o rechazar proyecto.
- `PATCH /projects/:projectId/status`: Cambiar estado del proyecto.
- `DELETE /projects/:projectId`: Eliminar proyecto.

Aplicaciones

Ruta base: `/applications`. Este conjunto de *endpoints* gestiona las postulaciones de estudiantes a proyectos y la administración de participantes.

- `POST /applications`: Aplicar a un proyecto.
- `GET /applications/completed`: Obtener historial de proyectos finalizados.
- `GET /applications/project/:projectId`: Obtener aplicaciones de un proyecto.
- `GET /applications/project/:projectId/participants`: Obtener participantes aceptados de un proyecto.
- `PATCH /applications/:applicationId/status`: Aprobar o rechazar aplicación.
- `PATCH /applications/project/:projectId/leave`: Abandonar proyecto.
- `PATCH /applications/project/:projectId/participant/:studentId`: Eliminar participante de un proyecto.

Horas

Ruta base: `/hours`. Este conjunto de *endpoints* gestiona el registro, consulta, aprobación y reenvío de horas asociadas a proyectos.

- `POST /hours`: Registrar horas con evidencias.
- `GET /hours/total`: Obtener total de horas aprobadas.
- `GET /hours/:hourId`: Obtener detalles de un registro de horas.
- `GET /hours/project/:projectId/pending`: Obtener horas pendientes de aprobación en un proyecto.
- `PATCH /hours/:hourId/status`: Aprobar o rechazar horas.
- `POST /hours/:hourId/evidences`: Agregar evidencias a registro.
- `DELETE /hours/:hourId/evidences/:fileId`: Eliminar evidencia específica.
- `PATCH /hours/:hourId/resubmit`: Reenviar horas rechazadas.

Archivos

Ruta base: `/files`. Este conjunto de *endpoints* gestiona la descarga o previsualización de los archivos de la plataforma. Esto incluye fotos de perfil, recursos de proyectos y evidencias de horas.

- `GET /files/:id/download`: Descargar archivo.
- `GET /files/:id/preview`: Previsualizar archivo.

7.2. Fase 2: Desarrollo del *Backend*

La segunda fase se centró en la implementación del *Backend* de la plataforma, capaz de soportar y gestionar todas las funcionalidades definidas en la fase anterior.

El objetivo de esta fase es desarrollar una API estructurada capaz de gestionar proyectos de extensión por parte de los usuarios. Esto incluye la gestión de usuarios, proyectos, postulaciones, horas y archivos.

7.2.1. Selección de tecnologías

Entorno de ejecución y *Framework*

Se seleccionó **Node.js** como entorno de ejecución, utilizando el *framework* **Express**. Este entorno permite que la API opere de manera asíncrona y facilitando el manejo de solicitudes simultáneas sin afectar el rendimiento. Express, por su parte, proporciona una estructura flexible y minimalista, lo cuál agiliza el desarrollo y facilita aspectos de mantenibilidad y escalabilidad.

Compilador

Para aumentar la robustez del código y tener un desarrollo más eficiente y seguro, se seleccionó **TypeScript** para introducir tipado estático en el proyecto. Con el fin de evitar errores durante el desarrollo y mejorar la legibilidad del código.

Persistencia de datos

Se optó por una base de datos relacional utilizando **PostgreSQL** como motor de base de datos para el almacenamiento de los datos de la plataforma. Esto gracias a su robustez, escalabilidad, soporte a tipos de datos avanzados (como UUID, gracias a la extensión pgcrypto).

Además, para el almacenamiento de archivos se optó por **Amazon S3**. Esto gracias a su alta disponibilidad, escalabilidad y seguridad. Esto permite cargar, eliminar o consultar archivos. Además, que AWS ofrece SDKs que facilitan la integración de la tecnología a proyectos, en este caso, el SDK (v3) de AWS para JavaScript en Node.js.

Librerías

Con el fin de garantizar seguridad en los datos en reposo, se integró **bcryptjs** para un *Hashing* seguro de contraseñas. También se integraron liberías como **jsonwebtoken** para la gestión de tokens de autenticación, **Joi** para validaciones de datos de entrada, **Nodemailer** para el envío de correos electrónicos, **Multer** para procesamiento de archivos, **Winston** para *Log* del sistema, entre otras liberías.

7.2.2. Estructura del proyecto

Para mantener una organización y separación de responsabilidades clara, el proyecto se estructuró en capas de la siguiente manera:

- **Routers:** Esta capa se encarga de definir y exponer los *endpoints* con el fin de interceptar las solicitudes entrantes y dirigirlas al controlador correspondiente.
- **Middlewares:** Esta capa (intermedia) se encarga de procesar las solicitudes entrantes y permite realizar validaciones como autenticación, autorización, validación de datos, manejo de errores, entre otros.
- **Controllers:** Esta capa se encarga de recibir las solicitudes enviadas por los routers, procesar la información de entrada e invocar los servicios correspondientes y devolver una respuesta al cliente.
- **Services:** Esta capa se encarga de toda la lógica de negocio, orquestando operaciones, reglas de negocio y permitiendo reutilización de servicios.
- **Repositories:** Esta capa se encarga de la interacción directa con la base de datos, realizando operaciones CRUD de las entidades y consultas directas.

Esta estructura fue aplicada a cada uno de los módulos definidos en la fase anterior.

Archivos adicionales

Además de las capas mencionadas, se crearon directorios y archivos adicionales para configuración de la plataforma, así como archivos de conexión (a la base de datos, a AWS S3 y servicios de correo electrónico), entre otros.

- **Types:** Este directorio contiene las definiciones de tipos utilizados en el proyecto.
- **Validators:** Este directorio contiene esquemas de validación de datos de entrada utilizando Joi.
- **DB:** Este directorio contiene el *Script* para la creación completa de la base de datos.
- **Config:** Este directorio contiene archivos de configuración global.
- **Connections:** Archivos de conexión a la base de datos, AWS S3 y servicios de correo electrónico.

CAPÍTULO 8

Resultados

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vitae eleifend ipsum, ut mattis nunc. Pellentesque ac hendrerit lacus. Cras sollicitudin eget sem nec luctus. Vivamus aliquet lorem id elit venenatis pellentesque. Nam id orci iaculis, rutrum ipsum vel, porttitor magna. Etiam molestie vel elit sed suscipit. Proin dui risus, scelerisque porttitor cursus ac, tempor eget turpis. Aliquam ultricies congue ligula ac ornare. Duis id purus eu ex pharetra feugiat. Vivamus ac orci arcu. Nulla id diam quis erat rhoncus hendrerit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed vulputate, metus vel efficitur fringilla, orci ex ultricies augue, sit amet rhoncus ex purus ut massa. Nam pharetra ipsum consequat est blandit, sed commodo nunc scelerisque. Maecenas ut suscipit libero. Sed vel euismod tellus.

Proin elit tellus, finibus et metus et, vestibulum ullamcorper est. Nulla viverra nisl id libero sodales, a porttitor est congue. Maecenas semper, felis ut rhoncus cursus, leo magna convallis ligula, at vehicula neque quam at ipsum. Integer commodo mattis eros sit amet tristique. Cras eu maximus arcu. Morbi condimentum dignissim enim non hendrerit. Sed molestie erat sit amet porttitor sagittis. Maecenas porttitor tincidunt erat, ac lacinia lacus sodales faucibus. Integer nec laoreet massa. Proin a arcu lorem. Donec at tincidunt arcu, et sodales neque. Morbi rhoncus, ligula porta lobortis faucibus, magna diam aliquet felis, nec ultrices metus turpis et libero. Integer efficitur erat dolor, quis iaculis metus dignissim eu.

CAPÍTULO 9

Discusión

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vitae eleifend ipsum, ut mattis nunc. Pellentesque ac hendrerit lacus. Cras sollicitudin eget sem nec luctus. Vivamus aliquet lorem id elit venenatis pellentesque. Nam id orci iaculis, rutrum ipsum vel, porttitor magna. Etiam molestie vel elit sed suscipit. Proin dui risus, scelerisque porttitor cursus ac, tempor eget turpis. Aliquam ultricies congue ligula ac ornare. Duis id purus eu ex pharetra feugiat. Vivamus ac orci arcu. Nulla id diam quis erat rhoncus hendrerit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed vulputate, metus vel efficitur fringilla, orci ex ultricies augue, sit amet rhoncus ex purus ut massa. Nam pharetra ipsum consequat est blandit, sed commodo nunc scelerisque. Maecenas ut suscipit libero. Sed vel euismod tellus.

Proin elit tellus, finibus et metus et, vestibulum ullamcorper est. Nulla viverra nisl id libero sodales, a porttitor est congue. Maecenas semper, felis ut rhoncus cursus, leo magna convallis ligula, at vehicula neque quam at ipsum. Integer commodo mattis eros sit amet tristique. Cras eu maximus arcu. Morbi condimentum dignissim enim non hendrerit. Sed molestie erat sit amet porttitor sagittis. Maecenas porttitor tincidunt erat, ac lacinia lacus sodales faucibus. Integer nec laoreet massa. Proin a arcu lorem. Donec at tincidunt arcu, et sodales neque. Morbi rhoncus, ligula porta lobortis faucibus, magna diam aliquet felis, nec ultrices metus turpis et libero. Integer efficitur erat dolor, quis iaculis metus dignissim eu.

CAPÍTULO 10

Conclusiones

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vitae eleifend ipsum, ut mattis nunc. Pellentesque ac hendrerit lacus. Cras sollicitudin eget sem nec luctus. Vivamus aliquet lorem id elit venenatis pellentesque. Nam id orci iaculis, rutrum ipsum vel, porttitor magna. Etiam molestie vel elit sed suscipit. Proin dui risus, scelerisque porttitor cursus ac, tempor eget turpis. Aliquam ultricies congue ligula ac ornare. Duis id purus eu ex pharetra feugiat. Vivamus ac orci arcu. Nulla id diam quis erat rhoncus hendrerit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed vulputate, metus vel efficitur fringilla, orci ex ultricies augue, sit amet rhoncus ex purus ut massa. Nam pharetra ipsum consequat est blandit, sed commodo nunc scelerisque. Maecenas ut suscipit libero. Sed vel euismod tellus.

Proin elit tellus, finibus et metus et, vestibulum ullamcorper est. Nulla viverra nisl id libero sodales, a porttitor est congue. Maecenas semper, felis ut rhoncus cursus, leo magna convallis ligula, at vehicula neque quam at ipsum. Integer commodo mattis eros sit amet tristique. Cras eu maximus arcu. Morbi condimentum dignissim enim non hendrerit. Sed molestie erat sit amet porttitor sagittis. Maecenas porttitor tincidunt erat, ac lacinia lacus sodales faucibus. Integer nec laoreet massa. Proin a arcu lorem. Donec at tincidunt arcu, et sodales neque. Morbi rhoncus, ligula porta lobortis faucibus, magna diam aliquet felis, nec ultrices metus turpis et libero. Integer efficitur erat dolor, quis iaculis metus dignissim eu.

CAPÍTULO 11

Recomendaciones

- Cras vitae eleifend ipsum, ut mattis nunc. Pellentesque ac hendrerit lacus. Cras sollicitudin eget sem nec luctus. Vivamus aliquet lorem id elit venenatis pellentesque. Nam id orci iaculis, rutrum ipsum vel, porttitor magna. Etiam molestie vel elit sed suscipit. Proin dui risus, scelerisque porttitor cursus ac, tempor eget turpis.
- Aliquam ultricies congue ligula ac ornare. Duis id purus eu ex pharetra feugiat. Vivamus ac orci arcu. Nulla id diam quis erat rhoncus hendrerit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed vulputate, metus vel efficitur fringilla, orci ex ultricies augue, sit amet rhoncus ex purus ut massa. Nam pharetra ipsum consequat est blandit, sed commodo nunc scelerisque. Maecenas ut suscipit libero. Sed vel euismod tellus.
- Proin elit tellus, finibus et metus et, vestibulum ullamcorper est. Nulla viverra nisl id libero sodales, a porttitor est congue.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit.

CAPÍTULO 12

Bibliografía

[1] T. Marwala, "Higher Education's Evolving Role in Sustainable Development," *United Nations University*, 2025. dirección: https://unu.edu/article/higher-educations-evolving-role-sustainable-development

[2] J. Salmi, *The challenge of establishing world-class universities (English)*, Washington, DC, 2009. dirección: http://documents.worldbank.org/curated/en/909281468339904574

[3] A. M. McCarthy, D. Maor, A. McConney y C. Cavanaugh, "Digital transformation in education: Critical components for leaders of system change," *Social Sciences & Humanities Open*, vol. 8, n.º 1, pág. 100 479, 2023, ISSN: 2590-2911. DOI: 10.1016/j.ssaho.2023.100479 dirección: https://www.sciencedirect.com/science/article/pii/S2590291123000840

[4] N. Thakur, *Designing the Digital Campus: A Framework for University Modernisation*, 2025. dirección: https://edutech.global/digital-campus-in-higher-education/

[5] Universidad del Valle de Guateamala (UVG), *Reglamento de Extensión*, Consultado el 26 de enero de 2026, Guatemala, UVG, 2013. dirección: https://res.cloudinary.com/webuvg/image/upload/v1651613422/WEB/Nosotros/reglamentos/2022/reglamento-extension.pdf

[6] C. Pinho, M. Franco y L. Mendes, "Web portals as tools to support information management in higher education institutions: A systematic literature review," *International Journal of Information Management*, vol. 41, págs. 80-92, 2018, ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2018.04.002 dirección: https://www.sciencedirect.com/science/article/pii/S0268401217303572

[7] J. A. A. Urmeneta, R. D. Clemente, C. F. D. Polinio, G. H. Guino, H. I. C. Sablawon y L. D. Alberca, "Design and Development of a Web-Based Extension Services Management Information System for State Universities and Colleges," *International Journal of Research and Innovation in Social Science*, págs. 3695-3706, 2025. DOI: 10.47772/IJRISS.2025.909000307 dirección: https://doi.org/10.47772/ijriss.2025.909000307

[8] L. García García, "Gestión de la función extension de las universidades ante la virtualidad," *Revista de Formación Gerencial*, n.º 1, págs. 10-27, 2020, ISSN: 1690-074X. dirección: https://dialnet.unirioja.es/servlet/articulo?codigo=9318542

[9] N. Barajas, "Data Centralization: Enhancing Decision-Making for Transformational Change," *EDUCASE review*, 2024. dirección: https://er.educause.edu/articles/sponsored/2024/11/data-centralization-enhancing-decision-making-for-transformational-change

[10] PagerDuty, *Digital Transformation Happening in Higher Education*, https://www.pagerduty.com/resources/digital-operations/learn/digital-transformation-education/, 2025.

[11] R. Harper, *Digital Transformation in Higher Education*, https://er.educause.edu/articles/sponsored/2021/9/digital-transformation-in-highereducation, 2021.

[12] R. Bastías y A. Rodríguez, "Extensión universitaria: función orientada al desarrollo regional," *Calidad en la Educación*, n.º 20, págs. 169-182, 2004.

[13] M. Colotta, S. Dabreinche y A. Presa, *Políticas universitarias para el siglo XXI*, Teseo, ed. Editorial Teseo, 2019, ISBN: 9789877232011. DOI: 10.55778/ts877232011 dirección: http://dx.doi.org/10.55778/ts877232011

[14] C. D. Loor-Rodríguez, L. J. Guerrero-Reyes y N. E. Delgado-Vera, "Universidad y sociedad, como eje de promoción del desarrollo social," *Prohominum*, vol. 4, n.º 2, págs. 147-159, jul. de 2022. DOI: 10.47606/ACVEN/PH0120 dirección: https://acvenisproh.com/revistas/index.php/prohominum/article/view/336

[15] J. Cedeño Ferrín y E. F. Machado Ramírez, "Papel de la Extensión Universitaria en la transformación local y el desarrollo social," es, *Humanidades Médicas*, vol. 12, págs. 371-390, dic. de 2012, ISSN: 1727-8120. dirección: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1727-81202012000300002&nrm=iso

[16] E. Chinkes y D. Julien, "Las instituciones de educación superior y su rol en la era digital. La transformación digital de la universidad: ¿transformadas o transformadoras?" *Ciencia y Educación*, vol. 3, n.º 1, págs. 21-33, jun. de 2019. DOI: 10.22206/cyed.2019.v3i1.pp21-33 dirección: https://revistas.intec.edu.do/index.php/cienedu/article/view/1449

[17] A. E. D. Giusti, "Transformación Digital en Educación Superior. Posibilidades y Desafíos," *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, págs. 9-15, 2023, ISSN: 1850-9959. DOI: 10.24215/18509959.35.e1 dirección: https://doi.org/10.24215/18509959.35.e1

[18] D. A. Cueva Gaibor, "Transformación digital en la universidad actual," es, *Conrado*, vol. 16, págs. 483-489, dic. de 2020, ISSN: 1990-8644. dirección: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1990-86442020000600483&nrm=iso

[19] IBM Corporation, *Servicios Web*, IBM Corporation, 2025. dirección: https://www.ibm.com/docs/es/was/9.0.5?topic=services-web

[20] M. R. V. Pardo, J. A. H. Tapia, A. S. G. Moreno y L. F. V. Sánchez, "Comparación de tendencias tecnológicas en aplicaciones web," *Glosas de Innovación Aplicadas a la Pyme*, vol. 7, n.º 3, págs. 28-49, 2018, ISSN: 2254-4143. DOI: 10.17993/3ctecno.2018.v7n3e27.28-49/ dirección: https://doi.org/10.17993/3ctecno.2018.v7n3e27.28-49/

[21] AWS, *¿Cuál es la diferencia entre el front end y back end en el desarrollo de aplicaciones?* https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/, 2025.

[22] I. Romero, "Arquitectura de aplicaciones Web," *Técnicas de Programación para Internet*, 2021. dirección: https://www.academia.edu/download/51686916/Arquitectura_de_aplicaciones_Web.pdf

[23] Universidad El Bosque, *¿Qué es la Arquitectura de Aplicaciones Web?* https://www.unbosque.edu.co/educacion-continua/blog-educacion-continua/arquitectura-de-aplicaciones-web, 2024.

[24] Vercel, *How to choose the best rendering strategy for your app*, https://vercel.com/blog/how-to-choose-the-best-rendering-strategy-for-your-app, 2024.

[25] M. E. Rana y O. S. Saleh, "Chapter 15 - High assurance software architecture and design," en *System Assurances*, ép. Emerging Methodologies and Applications in Modelling, P. Johri, A. Anand, J. Vain, J. Singh y M. Quasim, eds., Academic Press, 2022, págs. 271-285, ISBN: 978-0-323-90240-3. DOI: https://doi.org/10.1016/B978-0-323-90240-3.00015-1 dirección: https://www.sciencedirect.com/science/article/pii/B9780323902403000151

[26] D. Gala Yalupalin, *La programación Front-End y Back-End*, 2021. dirección: https://repositorio.une.edu.pe/handle/20.500.14039/6052

[27] M. Richards, B. Lange y N. Ford, *Fundamentals of software architecture : an engineering approach / Mark Richards and Neal Ford.* eng, Rego Park, 2021.

[28] AWS, *¿Qué es AWS?* https://aws.amazon.com/es/what-is-aws/, 2026.

[29] AWS, *Amazon S3*, https://aws.amazon.com/es/s3/, 2025.

[30] AWS, *¿Qué es Docker?* https://aws.amazon.com/es/docker/, 2025.

[31] The PostgreSQL Global Development Group, *PostgreSQL Documentation*, https://www.postgresql.org/docs/, 2026.

[32] M. Cantelon, M. Harter, T. Holowaychuk y N. Rajlich, *Node. js in Action*. Manning Greenwich, 2014, ISBN: 9781617290572.

[33] AWS, *¿Qué es JavaScript (JS)?* https://aws.amazon.com/es/what-is/javascript/, 2025.

[34] J. A. Saavedra, *Qué es React y para qué sirve*, https://ebac.mx/blog/que-es-react, 2023.

[35] M. Rauhala, "Comparison of full-stack capabilities of modern React frameworks," Bachelor's thesis, Tampere University of Applied Sciences, Tampere, 2025. dirección: https://www.theseus.fi/handle/10024/886359

[36] T. Garg, "React Query: Revolutionizing Data Fetching and Modernizing User Interfaces," 2025. DOI: 10.36227/techrxiv.175000818.80756952/v1 dirección: https://doi.org/10.36227/techrxiv.175000818.80756952/v1

[37] TanStack Form, *TanStack Form*, https://tanstack.com/form/latest, Consultado el 04 de febrero de 2026.

[38] MUI, *Material UI (MUI)*, https://mui.com/material-ui/getting-started/, 2025.

[39] Axios, *¿Qué es Axios?* https://axios-http.com/es/docs/intro, 2025.

[40] Zustand, *Zustand*, https://zustand.docs.pmnd.rs/learn/getting-started/introduction, 2025.

[41] Auth0, *Introduction to JSON Web Tokens*, https://www.jwt.io/introduction, 2024.

[42] AWS, *¿Qué es una API RESTful?* https://aws.amazon.com/what-is/restful-api/, 2025.

[43] PNPM, *PNPM Motivation*, https://pnpm.io/es/motivation, 2025.

[44] Postman, Inc., *Postman*, https://www.postman.com/product/, 2025.

[45] A. S. R. Ovalle, "Diseño de experiencia de usuario e interfaz de un sistema para llevar el control y promover las actividades de extensión.," Universidad del Valle de Guatemala, inf. téc., 2023. dirección: https://repositorio.uvg.edu.gt/handle/123456789/5036

CAPÍTULO 13

Anexos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vitae eleifend ipsum, ut mattis nunc. Pellentesque ac hendrerit lacus. Cras sollicitudin eget sem nec luctus. Vivamus aliquet lorem id elit venenatis pellentesque. Nam id orci iaculis, rutrum ipsum vel, porttitor magna. Etiam molestie vel elit sed suscipit. Proin dui risus, scelerisque porttitor cursus ac, tempor eget turpis. Aliquam ultricies congue ligula ac ornare. Duis id purus eu ex pharetra feugiat. Vivamus ac orci arcu. Nulla id diam quis erat rhoncus hendrerit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed vulputate, metus vel efficitur fringilla, orci ex ultricies augue, sit amet rhoncus ex purus ut massa. Nam pharetra ipsum consequat est blandit, sed commodo nunc scelerisque. Maecenas ut suscipit libero. Sed vel euismod tellus.

Proin elit tellus, finibus et metus et, vestibulum ullamcorper est. Nulla viverra nisl id libero sodales, a porttitor est congue. Maecenas semper, felis ut rhoncus cursus, leo magna convallis ligula, at vehicula neque quam at ipsum. Integer commodo mattis eros sit amet tristique. Cras eu maximus arcu. Morbi condimentum dignissim enim non hendrerit. Sed molestie erat sit amet porttitor sagittis. Maecenas porttitor tincidunt erat, ac lacinia lacus sodales faucibus. Integer nec laoreet massa. Proin a arcu lorem. Donec at tincidunt arcu, et sodales neque. Morbi rhoncus, ligula porta lobortis faucibus, magna diam aliquet felis, nec ultrices metus turpis et libero. Integer efficitur erat dolor, quis iaculis metus dignissim eu.

Glosario

Backend Capa de la aplicación encargada de la lógica de negocio, procesamiento, seguridad y comunicación con la base de datos.. vii, 14

Cloud Computing (Computación en la Nube) Modelo de computación que permite el acceso a recursos informáticos (servidores, almacenamiento, bases de datos, redes, software) a través de Internet, ofreciendo escalabilidad, flexibilidad y eficiencia.. vii, 15

Contenedor Unidad de software ligera y portátil que empaqueta código y dependencias para ejecutarse de manera uniforme en cualquier entorno.. vii, 16

Endpoint Punto de acceso de comunicación (generalmente una URL) mediante el cual un cliente interactúa con una API.. vii, 20

Frontend Capa de la aplicación que interactúa directamente con el usuario, abarcando la interfaz gráfica y la experiencia de navegación.. vii, 11

fórmula Una expresión matemática. vii, 7

Hashing Proceso de transformar datos de entrada (como una contraseña) en una cadena de caracteres de longitud fija, utilizando un algoritmo específico, con el fin de proteger la información.. vii, 30

Isomórfico (Código) Código que tiene la capacidad de ejecutarse tanto en el entorno del cliente (navegador) como en el entorno del servidor (ej. Node.js).. vii, 19

latex Es un lenguaje de marcado adecuado especialmente para la creación de documentos científicos. vii, 7

Log Registro de eventos o actividades que ocurren dentro de un sistema, utilizado para monitoreo, depuración y análisis.. vii, 30

Renderizado Web Proceso informático mediante el cual un motor (como un navegador web) interpreta código (HTML, CSS, JS) para mostrar una interfaz gráfica interactiva al usuario.. vii, 11

Script Archivo de texto que contiene código ejecutable, generalmente utilizado para automatizar tareas o ejecutar funciones específicas dentro de un programa o sistema.. VII, 31

Type-safe (Tipado Seguro) Característica de un lenguaje de programación (como TypeScript) que previene errores al obligar a definir y respetar los tipos de datos en tiempo de desarrollo.. VII, 17