UACM

Universidad Autónoma de la Ciudad de México

NADA HUMANO ME ES AJENO

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA DE SOFTWARE

Portal web LACECI usando tecnologías de Software Libre

T E S I S

PARA OBTENER EL TÍTULO DE

LICENCIADA EN INGENIERÍA DE SOFTWARE

P R E S E N T A

GUADALUPE LIZBETH MEXICANO FLORES

D I R E C T O R

M. EN C. ENRIQUE CRUZ MARTÍNEZ

Ciudad de México, marzo de 2026.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

A mis padres, Bernardo Mexicano López y Graciela Flores Aguilar,
por ser el cimiento de cada uno de mis sueños.

Al amor de mi vida, Daniel Flores Sánchez,
por caminar a mi lado y creer en mí en cada paso.

AGRADECIMIENTOS

A mi familia, mis hermanos y sobrinas, por su apoyo incondicional tanto en mi vida personal como en el transcurso de mi carrera; su presencia ha sido el motor para llegar a esta meta.

Al **M. en C. Enrique Cruz Martínez**, mi director de tesis, le agradezco profundamente su guía, la confianza depositada en este proyecto y el apoyo institucional que hizo posible la culminación de este trabajo.

Mi gratitud más especial y profunda es para el **Ing. José Pablo Hernández López**. Gracias, por ser un mentor excepcional y por compartir conmigo tu conocimiento sin reservas. Agradezco infinitamente tu paciencia, tu disposición para enseñarme desde la base y por haberme acompañado en cada desafío técnico de este desarrollo. Tu apoyo no solo fortaleció este software, sino que fue fundamental en mi crecimiento profesional y personal.

A la **Lic. Norma Yolanda Landeros Valverde**, le expreso mi más sincera gratitud por su invaluable apoyo, su tiempo y la dedicación profesional brindada.

Asimismo, agradezco a **Eduardo Méndez Santiago** por su colaboración y disposición durante este proceso, así como al **Laboratorio LACECI** , por brindarme un espacio de crecimiento profesional y por el apoyo técnico brindado en cada etapa.

A mis compañeros y amigos, con quienes compartí retos y alegrías en este camino académico.

Finalmente, un agradecimiento muy especial a quien me acompañó fielmente cada noche de desvelo, brindándome su compañía silenciosa y amor incondicional: **Jack**, gracias por estar siempre a mi lado.

A todos ustedes, mi más sincera gratitud.

Índice general

Glosario

API (Interfaz de Programación de Aplicaciones), Arquitectura de servicios web basada en operaciones HTTP. 1, 11, 13, 17–19, 27, 28

CERN La Organización Europea para la Investigación Nuclear,(CERN). 5

CSS En español Hojas de Estilo en Cascada,(Cascading Style Sheets), es un lenguaje utilizado para definir la presentación y el estilo de documentos HTML y XML, separando el contenido de su aspecto visual. Permite controlar colores, tipos de fuentes, tamaños, posicionamiento y animaciones, facilitando la creación de sitios web responsivos y fáciles de mantener. . 38

DNS (Domain Name System): Sistema de Nombres de Dominio.. 7

DOM (Modelo de Objetos del Documento): Estructura en forma de árbol que representa un documento HTML o XML, permitiendo a los lenguajes de programación acceder, modificar y manipular su contenido, estructura y estilos de manera dinámica.). 9, 11–14, 20

ENIAC Electronic Numerical Integrator and Computer. 4

Express.js Es el framework backend más popular para Node.js, y es una parte extensa del ecosistema JavaScript. 1, 9, 26, 30, 60

HTTP (HyperText Transfer Protocol): Protocolo de Transferencia de Hipertexto.. 7

HTTPS (HyperText Transfer Protocol Secure): Protocolo de Transferencia de Hipertexto Seguro.. 7

JWT (JSON Web Tokens), son un estándar abierto, que se define en especificación del token web JSON. Representan de forma segura (claims) Las notificaciones de token web JSON, se utilizan para autenticar y autorizar aplicaciones y API entre dos partes. 1

LACECI El laboratorio de Cómputo para la enseñanza de las Ciencias, (LACECI). 1, 2, 8, 17, 35–39, 41, 42, 48

NCP (Network Control Protocol):Protocolo de Control de Red utilizado en las primeras redes, como ARPANET, que permitía la comunicación entre computadoras antes del desarrollo de TCP/IP. Establecía las reglas básicas para el intercambio de datos en la red.. 5, 6

Node.js Es un proyecto de código abierto basado en el motor de Google Chrome JavaScript.Proporciona una plataforma para aplicaciones JavaScript del lado del servidor que se ejecutan sin navegadores. 1, 9, 24, 25, 40

PostgreSQL Base de datos de código abierto admite tanto tipos de datos no relacionales como relacionales. 1, 30, 31, 44

Página Web Es un documento digital accesible a través de internet que permite mostrar información, contenido multimedia y elementos interactivos mediante un navegador web.. 8, 11–15, 22, 37

QUERIES (Consultas): Instrucciones escritas en lenguaje SQL que se utilizan para interactuar con una base de datos, permitiendo realizar operaciones como la obtención, inserción, actualización y eliminación de datos de manera estructurada y eficiente.. 30

TCP/IP Conjunto de protocolos de comunicación que permite la transmisión de datos en redes, dividiendo los mensajes en paquetes que se envían por distintas rutas y se reensamblan en el destino siguiendo reglas estandarizadas.. 6

TLS (Transport Layer Security): Seguridad de la Capa de Transporte.. 7

URL (Uniform Resource Locator): Localizador Uniforme de Recursos.. 7

Capítulo 1

Introducción

La transformación digital ha revolucionado múltiples aspectos de nuestra vida, uno de ellos ofreciendo nuevas formas de gestionar y organizar información de manera eficiente con herramientas que les permitan optimizar sus procesos administrativos y de organización.

Esta tesis presenta el desarrollo de un portal web innovador que facilita la gestión integral de las actividades de un laboratorio académico, proporcionando una plataforma eficiente para la administración de proyectos de tesis, actividades de servicio social, eventos académicos y cursos.

La complejidad y cantidad de tareas que maneja este laboratorio requieren soluciones tecnológicas que permitan optimizar la administración de dichos procesos.

En muchas instituciones educativas, la gestión de tesis, servicios sociales y la organización de eventos académicos enfrenta desafíos relacionados con la fragmentación de la información y la falta de plataformas centralizadas. LACECI objeto de este estudio, no es la excepción, enfrentando dificultades para coordinar sus actividades y ofrecer una experiencia fluida tanto a estudiantes como a profesores.

El presente trabajo tiene como propósito el diseño y desarrollo de un portal web orientado a la gestión integral de las actividades del laboratorio académico LACECI. La propuesta se enfoca en la optimización de procesos administrativos y académicos, tales como la gestión de tesis, el seguimiento del servicio social estudiantil y la organización de eventos y cursos. Para su implementación, se emplea una arquitectura basada en tecnologías modernas de desarrollo web, destacando el uso de Node.js y Express.js para la construcción del backend, PostgreSQL como sistema de gestión de bases de datos, así como la integración de API para la comunicación entre servicios y mecanismos de autenticación seguros mediante JWT. En este sentido, la solución propuesta contribuye a la mejora de la eficiencia operativa del laboratorio, al tiempo que proporciona una plataforma robusta, escalable y de fácil acceso para estudiantes y profesores, garantizando la integridad y confidencialidad de la información gestionada.
Este proyecto no solo resuelve la necesidad de centralizar procesos, sino que también introduce una solución moderna y escalable que contribuye a la mejora continua de las operaciones del laboratorio.

Se construye una plataforma robusta y segura, que facilita la administración de los procesos clave del laboratorio. Con este proyecto se espera mejorar significativamente la eficiencia y organización de las actividades, resolviendo las principales necesidades identificadas.

1.1 Objetivo

Objetivo General: Desarrollar e implementar una plataforma web para el laboratorio LACECI que permita automatizar la gestión administrativa y optimizar la comunicación institucional, mediante el uso de tecnologías modernas de desarrollo web, con el fin de mejorar la eficiencia operativa y el servicio hacia la comunidad académica y externa.
Objetivos Específicos:

1. **Analizar la evolución de las tecnologías web** para fundamentar la selección de herramientas utilizadas en el desarrollo del portal.

2. **Diseñar una interfaz segura y funcional** que responda a las necesidades de administración del laboratorio LACECI.

3. **Evaluar el impacto de la herramienta** en la reducción de tiempos administrativos y mejora de la comunicación del laboratorio.

1.2 Motivación

El tema de este trabajo es sistematizar LACECI con todos sus componentes en servicio a los estudiantes e integrantes de la UACM.

Este trabajo se realiza como parte de una tesis y para apoyo de tener un mejor control, mayor alcance en el público de los servicios que se brindan y seguridad de los documentos realizados y publicados en este laboratorio.
Este trabajo surge a partir del crecimiento sostenido del laboratorio LACECI y del incremento en las actividades académicas y administrativas que se desarrollan en su interior, lo cual ha generado la necesidad de contar con una plataforma que permita una gestión más eficiente y organizada.

1.3 Organización de la tesis

La presente tesis está estructurada en cinco capítulos, los cuales abordan de manera progresiva el contexto, los fundamentos teóricos y el desarrollo del sistema propuesto.
En el Capítulo 1 se presenta el planteamiento general de la investigación, incluyendo la introducción al problema, los objetivos del proyecto y la motivación que sustenta el desarrollo de la propuesta.
El Capítulo 2 aborda la evolución del Internet, desde sus orígenes y el desarrollo de ARPANET, hasta la transición hacia el uso del protocolo TCP/IP y la consolidación de los conceptos actuales, proporcionando el contexto histórico y tecnológico necesario.
En el Capítulo 3 se desarrolla el marco teórico, donde se describen los conceptos fundamentales relacionados con la arquitectura de aplicaciones web, el lenguaje JavaScript como eje central, la arquitectura de microservicios y las tecnologías utilizadas en el desarrollo frontend y backend. Asimismo, se incluyen herramientas de gestión de datos e infraestructura.
El Capítulo 4 presenta la evolución del hipertexto y de las páginas web, destacando el paso de sitios estáticos a aplicaciones dinámicas e interactivas, así como los fundamentos tecnológicos, estándares web y tendencias actuales como el diseño responsivo.
En el Capítulo 5 se describe la metodología empleada para el desarrollo del sistema, basada en un modelo incremental. Se detallan las fases de análisis, diseño, desarrollo e implementación, así como los resultados obtenidos y los módulos que conforman la solución desarrollada.

Capítulo 2

Evolución del Internet

2.1 Origen del Internet

La historia se remonta a la década de 1940, periodo en el que la Segunda Guerra Mundial involucró a diversas naciones entre 1939 y 1945. Durante este difícil periodo, surgieron avances tecnológicos como las computadoras modernas. Las primeras de las que se tiene registro fueron la ENIAC, desarrollada en Estados Unidos, y la Z1 en Alemania durante los años 1945 y 1946.

Antes de la creación de Internet, la única forma de comunicarse digitalmente era por medio del telégrafo. El telégrafo se inventó en 1840, emitía señales eléctricas que viajaban por cables conectados entre un origen y un destino; utilizaba el alfabeto Morse para interpretar la información.

El alfabeto Morse es un sistema de representación de letras, números y símbolos mediante combinaciones de puntos y rayas. Fue desarrollado por Samuel Morse en la década de 1830 y se utilizó ampliamente en la transmisión de mensajes a través de telegrafía. Se pueden transmitir a través de una luz intermitente, un sonido o cualquier otro medio. Aunque hoy en día ha sido reemplazado por tecnologías más avanzadas, el alfabeto Morse sigue siendo un símbolo de la historia de las comunicaciones.

Mientras que los principales medios de comunicación eran digital y analógico, las computadoras eran grandes máquinas que realizaban cálculos y almacenaban información, por lo que su uso era simplemente científico o gubernamental. Entonces, ¿cómo llegamos a vivir en una llamada Sociedad de la Información donde la tecnología inunda todos los aspectos de nuestra vida?

Para esto tenemos que remontarnos a la historia del Internet. En la actualidad, vivir sin Internet es sencillamente impensable. Por eso, hablaremos de la línea del tiempo de Internet para descubrir sus orígenes y transitar por los momentos clave de su evolución y cómo es que llegamos hasta la era moderna (segunda mitad del siglo XX y primeras décadas del XXI).[2]

2.2 ARPANET

Es clave en la historia del Internet, pues fue responsable de la investigación y el desarrollo de nuevas tecnologías con propósitos defensivos y militares, entre esas, las redes de computadoras [7].

En 1958, los Estados Unidos fundaron la *Advanced Research Projects Agency* (ARPA) a través del Ministerio de Defensa. ARPA estaba formado por unos 200 científicos de alto

nivel y tenía un gran presupuesto. ARPA se centró en crear comunicaciones directas entre ordenadores para poder comunicar las diferentes bases de investigación.
La ARPANET (*Advanced Research Projects Agency Network*) o Red de Agencias de Proyectos de Investigación Avanzada, era una red de computadoras construida en 1969 como un medio resistente y persistente para enviar datos militares y conectar los principales grupos de investigación a través de los Estados Unidos. ARPANET usó NCP (*Network Control Protocol*) y subsecuentemente la primera versión del protocolo de Internet o la suite TCP/IP. El proyecto original finalizó a comienzos de 1990.
En 1972, ARPANET se presentó en la *First International Conference on Computers and Communication* en Washington DC. Los científicos demostraron que el sistema era operativo creando una red de 40 puntos conectados. Esto estimuló la creación de otras redes entre 1974 y 1982:

- **Telenet (1974):** Versión comercial de ARPANET.
- **Usenet (1979):** Sistema abierto centrado en el e-mail.
- **Bitnet (1981):** Unía universidades americanas usando sistemas IBM.
- **Eunet (1982):** Unía al Reino Unido, Escandinavia y Holanda.

En 1982, ARPANET adoptó el protocolo TCP/IP y en aquel momento nació formalmente Internet (*International Net*). Aunque fue desmantelada en 1990, su legado perdura: sentó las bases para la revolución digital que cambiaría el mundo contemporáneo.

2.3 De ARPANET a WWW

A principios de los 80 los ordenadores se desarrollaron de forma exponencial. Se temía que las redes se bloquearan debido al gran número de usuarios y el fenómeno del correo electrónico (*e-mail*).
La *World Wide Web* (WWW), nació en La Organización Europea para la Investigación Nuclear, (CERN) de la mano de Tim Berners-Lee. En 1982, se adoptó el protocolo TCP/IP y se creó la *International Net* [6] como un sistema de intercambio de datos para los 10,000 científicos de la institución. En 1991 se creó el primer servidor web, y en 1993 el CERN liberó el proyecto al dominio público.
Internet es una de las más grandes invenciones del siglo XX. Uno de sus principales pioneros fue Vinton Cerf, reconocido por su contribución al desarrollo de los protocolos de comunicación TCP/IP, los cuales establecieron las bases para la interconexión de redes a nivel global. Posteriormente, el trabajo de Tim Berners-Lee resultó imprescindible para el desarrollo de la World Wide Web, sentando las bases del entorno web moderno.

A partir de ello, la web ha evolucionado en sus lenguajes fundamentales, entre los que destacan:

- **HTML (HyperText Markup Language):** Lenguaje de marcado utilizado para estructurar el contenido de las páginas web, definiendo elementos como encabezados, párrafos, enlaces e imágenes.
- **CSS (Cascading Style Sheets):** Lenguaje encargado de la presentación y el diseño visual de los documentos HTML, permitiendo definir estilos, colores, tipografías y la disposición de los elementos.
- **JavaScript:** Lenguaje de programación que aporta interactividad y dinamismo a las páginas web, permitiendo la manipulación del contenido, la gestión de eventos y la comunicación con servidores.
- **HTML:** Código utilizado para estructurar y desplegar el contenido de las páginas.
- **HTTP:** Protocolo de comunicación que posibilita la circulación de información a través de la WWW usando hipervínculos.

2.4 WAN (Wide Area Network)

El gran paso se dio en 1965 cuando Lawrence G. Roberts y Thomas Merrill conectaron una computadora TX2 con una Q-32 mediante una línea telefónica conmutada. Este éxito marcó el hito de la primera red de área amplia (WAN) de la historia.

2.5 Nacimiento del NCP y el Correo Electrónico

En 1970, S. Crocker estableció el protocolo NCP, permitiendo que las aplicaciones de computadoras puedan ser conectadas. En 1972, Ray Tomlinson creó el software básico del correo electrónico, cambiando la naturaleza de la comunicación humana. Para 1974, más de 50 universidades ya estaban conectadas.

2.6 Transición a TCP/IP

Robert Kahn y Vinton Cerf desarrollaron el protocolo TCP/IP en 1974 para permitir la comunicación entre redes heterogéneas (radio, satélite), asegurando que los paquetes no se perdieran. En 1983, ARPANET sustituyó oficialmente el NCP por el TCP/IP.

2.7 Actualidad y Conceptos Clave

El lanzamiento de Google en 1997 supuso un nuevo punto de quiebre. Hoy, la web funciona mediante protocolos fundamentales:

1. **DNS:** Convierte nombres de dominio en direcciones IP.
2. **HTTP/ HTTPS:** Solicita el contenido del sitio.
3. **TLS:** Asegura la conexión mediante encriptación.
4. **URL:** (*Uniform Resource Locator*) Identifica y permite localizar recursos en la red.

La Web es solo una aplicación construida sobre los protocolos de Internet, pero es, por mucho, la más popular.

Capítulo 3

Marco Teórico

3.1 Arquitectura de Aplicaciones Web

El desarrollo del sistema LACECI se fundamenta en una arquitectura moderna que divide las responsabilidades en capas, permitiendo un desarrollo ágil y un mantenimiento eficiente.

Frontend y Backend

El desarrollo actual de portales web se basa en la separación de responsabilidades a través de dos capas fundamentales. El **frontend** es la parte con la que los usuarios interactúan directamente; utiliza lenguajes como HTML y CSS para estructurar y diseñar, mientras que JavaScript agrega funcionalidad y dinamismo, permitiendo animar imágenes y crear contenido que se actualiza frecuentemente sin necesidad de recargar la página.

Por otro lado, el **backend** es el encargado de interactuar con bases de datos, manipular archivos y generar respuestas del lado del servidor. Esta separación facilita que el sistema sea más rápido, ya que el procesamiento se distribuye eficientemente entre el cliente y el servidor.

3.2 El lenguaje JavaScript como eje central

Fue creado por Brendan Eich en 1995 y se convirtió en un estándar oficial del World Wide Web Consortium (W3C) en 1997. Surgió a partir del desarrollo del navegador Netscape, en el cual se identificó la necesidad de incorporar mecanismos que permitieran interactuar con las acciones del usuario dentro de documentos HTML, lo que dio origen a la integración de un lenguaje que proporcionara dinamismo a las páginas web [20].

Momento importante en la historia de JavaScript es la creación del elemento XML HttpRequest por parte de Microsoft en 1998. Este elemento viene a ser el padre de AJAX y nos permite cargar contenido o interactuar con el backend sin tener que volver a cargar la totalidad de la Página Web.

JavaScript es un lenguaje de programación versátil y de tipado dinámico ampliamente utilizado en el desarrollo web, tanto del lado del cliente como del servidor. Su principal función es dotar de interactividad y dinamismo a las aplicaciones, permitiendo la actualización frecuente de contenido, la manipulación de elementos, la animación de

interfaces y la gestión de eventos. En conjunto con HTML y CSS, que se encargan de la estructura y el diseño de las páginas web, JavaScript aporta la lógica y funcionalidad necesarias para construir aplicaciones web modernas, integrándose de manera eficiente con estas tecnologías y apoyándose en una amplia biblioteca estándar.

Algunos puntos de JavaScript

- JavaScript es un lenguaje de un solo subproceso que ejecuta una tarea a la vez.
- Es un lenguaje interpretado, lo que significa que ejecuta el código línea por línea.
- El tipo de datos de la variable se decide en tiempo de ejecución en JavaScript, por eso se llama tipado dinámico.

3.2.1 Características de JavaScript

- Scripting del lado del cliente: JavaScript se ejecuta en el navegador del usuario, por lo que tiene un tiempo de respuesta más rápido sin necesidad de comunicarse con el servidor [27].
- JavaScript se puede utilizar para una amplia gama de tareas, desde cálculos simples hasta aplicaciones complejas del lado del servidor.
- JavaScript puede responder a las acciones del usuario (clics, pulsaciones de teclas) en tiempo real.
- JavaScript puede manejar tareas como obtener datos de servidores sin congelar la interfaz de usuario.
- Existen numerosas bibliotecas y marcos creados en JavaScript, como React.js, Angular y Vue.js, que hacen que el desarrollo sea más rápido y eficiente.

3.2.2 Naturaleza del lado del cliente y del lado del servidor de JavaScript

Del lado del cliente: implica controlar el navegador y su DOM, y gestionar eventos del usuario como clics e ingresos de formularios. Se utilizan comúnmente bibliotecas como AngularJS, ReactJS y VueJS.

Del lado del servidor: implica interactuar con bases de datos, manipular archivos y generar respuestas. Con Node.js y frameworks como Express.js, JavaScript también se usa ampliamente en el lado del servidor.

JavaScript se considera un lenguaje ligero debido a su bajo uso de CPU, sintaxis minimalista y facilidad de implementación. Sin tipos de datos explícitos y una sintaxis similar a C++ y Java, es fácil de aprender y se ejecuta de manera eficiente en los navegadores.

JavaScript se compila y se interpreta. El motor V8 mejora el rendimiento interpretando primero el código y luego compilando las funciones que se usan con frecuencia para aumentar la velocidad. Esto hace que JavaScript sea eficiente para las aplicaciones web modernas.

3.2.3 Compilación JS

La compilación Just-In-Time (JIT) es una técnica que utilizan los motores de JavaScript (como V8) para mejorar el rendimiento. Así es como funciona

- **Interpretación:** Inicialmente, el código es interpretado línea por el linea por el motor.
- **Detección de código activo:** el motor identifica el código que se ejecuta con frecuencia, como las funciones llamadas con frecuencia.
- **Compilación:** El código "caliente" se compila en código de máquina optimizado para una ejecución más rápida.
- **Ejecución:** El código de máquina compilado se ejecuta directamente, lo que mejora el rendimiento en comparación con la interpretación repetida.
- **La compilación JIT** equilibra la interpretación (para un inicio rápido) y la compilación (para una ejecución más rápida).

3.2.4 Frontend (Parte del cliente)

El frontend es una capa de software que se ejecuta en el cliente. Es la parte de una Página Web o aplicación que los usuarios ven y con la que interactúan, también es la parte del sitio web que interactúa con los usuarios, lo que podemos ver, oír y clicar.

Es la parte de la programación que implementa la capa visible, con la que el usuario interactúa. Esta capa se conoce con varios nombres: Interface gráfica, User interface (UI) o capa de presentación.

Se centra en presentar la información de manera atractiva y facilitar las interacciones del usuario. El diseño del frontend se adapta a las necesidades y expectativas de los usuarios, priorizando una experiencia de usuario positiva.

El código dice cómo se va a mostrar el contenido, el tamaño, el color, la posición, entre otras. En esta capa encontramos: menús de navegación, textos, imágenes, videos, botones, etcétera.

Esa capa pareciera estar realizando todo el trabajo, en realidad, se comunica constantemente con el backend, que se encarga del procesamiento de datos. Esta comunicación se realiza a través de interfaces de programación conocidas como API.

Siguiendo el principio de "Separation of concerns", en español "Separación de responsabilidades", el backend y el frontend se desarrollan como capas independientes pero complementarias.

Los dos objetivos principales de un desarrollador de frontend son mejorar el rendimiento y la capacidad de respuesta, lo que significa que el frontend se cargue rápidamente y funcione bien en todo tipo de dispositivos.

El desarrollo web frontend hace uso de HTML, CSS y JavaScript creando elementos visuales, como botones, imágenes, casillas de verificación, gráficos y mensajes de texto, de modo que un usuario pueda ver e interactuar con ellos.

En términos técnicos, una página o pantalla que el usuario ve con varios componentes de la interface de usuario se denomina modelo de objetos del documento DOM.
Los tres lenguajes principales que afectan a la forma en que los usuarios interactúan con el frontend:

- El lenguaje HTML define la estructura del frontend y los diferentes elementos del DOM.

- Las hojas de estilo en cascada **(CSS)** definen el estilo de una aplicación web, incluido el diseño, las fuentes, los colores y el estilo visual.

- JavaScript agrega una capa de funcionalidad dinámica mediante la manipulación del DOM.

JavaScript puede activar cambios en una página y mostrar información nueva. En resumen el frontend puede gestionar las interacciones (o solicitudes) fundamentales de los usuarios, como mostrar un calendario o comprobar si el usuario ha ingresado una dirección de correo electrónico válida. El frontend transmite las solicitudes más complejas al backend.

Figura 3.1: Ilustración de la separación de responsabilidades entre el frontend, orientado al usuario, y el backend, enfocado en la lógica y la gestión de la información.

DOM Modelo de Objeto de Documento

Las siglas DOM significan Document Object Model, o lo que es lo mismo, la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM (o simplemente DOM).

El DOMcontiene modelizados en objetos todos y cada uno de los elementos que existen dentro de una Página Web, a los que podemos acceder para leer o modificar sus propiedades, de modo que analizaremos o cambiaremos el estado de la página.

Es una interface de programación para los documentos HTML y XML. Facilita una representación estructurada del documento y define de qué manera pueden acceder los programas a fin de modificar tanto su estructura, estilo y contenido.
El DOM da una representación del documento como un grupo de nodos y objetos estructurados que tienen propiedades y métodos. Esencialmente, conecta las páginas web a scripts o lenguajes de programación.

Una Página Web es un documento que puede mostrarse en la ventana de un navegador o también como código fuente HTML. En los dos casos, es el mismo documento. El modelo de objeto de documento (DOM) proporciona otras formas de presentar, guardar y manipular este mismo documento. El DOM es una representación completamente orientada al objeto de la Página Web y puede ser modificado con un lenguaje de script como JavaScript.

El DOM fue diseñado para ser independiente de cualquier lenguaje de programación particular. Hace que la presentación estructural del documento este disponible desde una simple y consistente API.

¿Cómo se accede al DOM?
El DOM nos permite hacer cualquier cosa con sus elementos y contenidos, pero lo primero que tenemos que hacer es llegar al objeto correspondiente del DOM.
Todas las operaciones en el DOMcomienzan con el objeto document. Este es el principal "punto de entrada" al DOM. Desde ahí podremos acceder a cualquier nodo.El DOM es la estructura de nuestro documento HTML que forma un árbol de jerarquía de nodos.

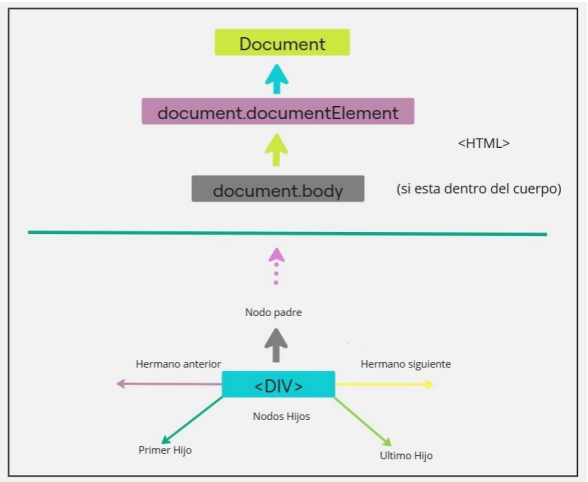


Figura 3.2: Enlaces que nos permiten viajar a través de los nodos del DOM.

Para poder realizar modificaciones, cambios y mejoras en este documento desde JavaScript, se emplea un objeto denominado document, el cual forma parte del Modelo de Objetos del Documento (DOM). Dentro de este modelo se identifican dos conceptos fundamentales: **Element** y textbfNode. Un Element representa una etiqueta específica del documento HTML, como párrafos, encabezados o divisiones, mientras que un Node constituye la unidad básica del DOM, abarcando no solo elementos, sino también otros tipos de nodos como texto, comentarios y atributos. Esta distinción permite una manipulación más precisa de la estructura del documento, ya que JavaScript puede interactuar con cualquier tipo de nodo presente en el árbol del DOM.

Dentro del objeto document se encuentran múltiples elementos que permiten estructurar y manipular el contenido de una Página Web. A continuación, se presentan algunas de sus características más relevantes.

Tipo de dato		Tipo específico	Etiqueta	Descripción
ELEMENT	HTMLElement	HTMLDivElement		Capa divisoria invisible(en bloque).
ELEMENT	HTMLElement	HTMLDivElement		Capa divisoria invisible(en línea).
ELEMENT	HTMLElement	HTMLDivElement	<img>	Imagen.
ELEMENT	HTMLElement	HTMLDivElement	<audio>	Contenedor de audio.

Figura 3.3: Descripción de algunos elementos que conforman el DOM, destacando su tipo, etiqueta y funcionalidad en la estructura del documento.

3.2.5 Backend (Lado del servidor)

El backend se refiere a la tecnología e interfaz que se ejecuta en la parte trasera de un sitio o aplicación web para producir una interfaz completamente orientada al servidor o para hacer posible la funcionalidad del frontend.

El backend es la parte invisible, pero esencial de un sitio, encargada de manejar la lógica y el procesamiento de datos necesarios para que todo funcione de manera correcta y segura. El backend se ocupa de:

- Almacenar y recuperar datos de una base de datos.
- Procesar formularios.

- Autenticar usuarios.
- Gestionar la seguridad del sitio.
- Acceder al servidor.

Es responsable de todo lo que sucede detrás de la interfaz visible. Además, debe garantizar la seguridad de los sitios web y optimizar al máximo los recursos para que las páginas sean ligeras.
Lo que se hace desde atrás, es decir en la parte del backend, es el proceso de administrar el almacenamiento de datos y acceder a ellos en una base de datos para mostrarlos en una Página Web, para que los usuarios puedan consumirlos desde cualquier dispositivo, es decir hacer uso de ellos, mostrarlos y editarlos.

El Backend es el "cerebro" detrás de la aplicación. Se encarga de procesar datos, interactuar con la base de datos, aplicar lógica y coordinar la comunicación entre el cliente (el navegador) y el servidor.

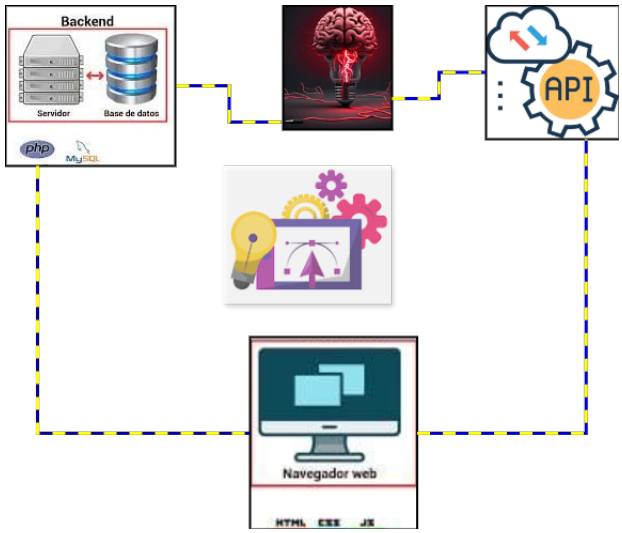


Figura 3.4: Representación del backend como la capa encargada de la lógica del sistema, la comunicación mediante APIs y la gestión de datos entre el servidor y el navegador web.

El Backend se refiere a la parte del sistema que opera "detrás de escena," gestionando la lógica, procesamiento y almacenamiento de datos de una aplicación o sitio web. Es el motor que permite que el software funcione, ya que se encarga de las operaciones complejas que el usuario final no ve directamente.
En una arquitectura de software, el Backend incluye :

- Servidor
- Base de datos
- API
- Lógica de negocio
- Seguridad y autenticación

En resumen, el Backend es la estructura invisible que permite que el software funcione correctamente, procesando las solicitudes del usuario y gestionando la información que permite a la aplicación ofrecer sus servicios.

Estos dos conceptos explican a grandes rasgos cómo funciona un sitio o aplicación web y son fundamentales para cualquier persona que trabaje en el mundo digital.

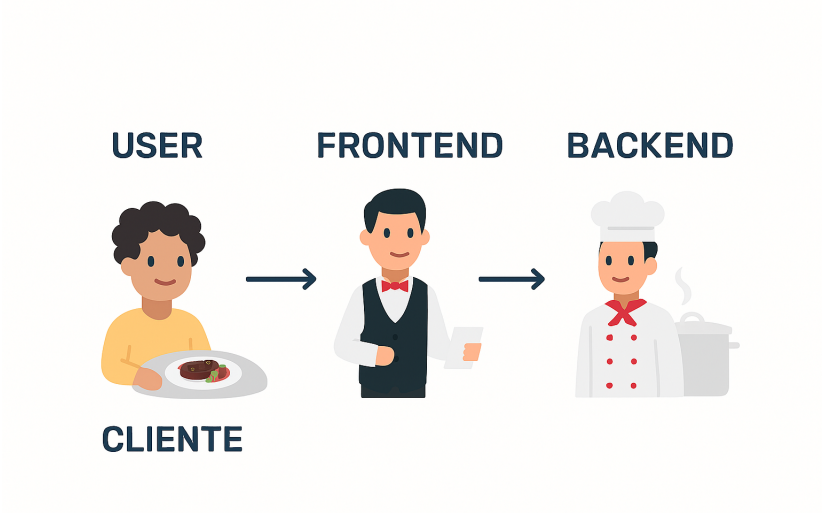


Figura 3.5: Front end y back end.
División de roles en el desarrollo web. Se muestra el flujo desde la interfaz del lado del cliente (frontend) hasta la estructura de procesamiento del lado del servidor (backend).

La separación entre frontend y backend no es solo una división técnica, sino una estrategia para garantizar la escalabilidad y el mantenimiento del sistema a largo plazo.

Como se ilustra en la analogía del restaurante (Figura 3.5), cada componente cumple un rol especializado: mientras el frontend se enfoca en la experiencia, estética y facilidad de uso para el cliente, el backend asegura que toda la "maquinaria" lógica y de datos funcione con precisión y seguridad.

En resumen, la arquitectura presentada en este capítulo (desde los endpoints de la API hasta el modelo de capas y el despliegue final) busca:

- **Robustez:** Un servidor capaz de procesar múltiples solicitudes sin errores.
- **Claridad:** Una interfaz intuitiva que guíe al usuario en sus tareas diarias.
- **Eficiencia:** Una comunicación fluida entre ambas partes que minimice los tiempos de espera y optimice los recursos de la institución.

Con esta estructura definida y el personal debidamente entrenado, el sistema LACECI queda listo para su fase operativa, asegurando que la tecnología sea una herramienta de apoyo real para la productividad académica y administrativa.

3.3 Arquitectura de Microservicios

Una arquitectura de microservicios divide una aplicación en una serie de servicios implementables de forma independiente que se comunican a través de una API. Este enfoque permite implementar y escalar cada servicio individual de forma independiente, así como la entrega rápida y frecuente de aplicaciones grandes y complejas. A diferencia de una aplicación monolítica, una arquitectura de microservicios permite a los equipos implementar nuevas funciones y hacer cambios más rápido, sin tener que volver a escribir una gran parte del código existente.

La arquitectura monolítica puede resultar práctica al principio de un proyecto para aliviar la sobrecarga cognitiva de la gestión de código, así como la implementación. Pero una vez que una aplicación monolítica se vuelve grande y compleja, resulta difícil escalarla, la implementación continua pasa a ser un desafío y las actualizaciones pueden resultar complicadas [1].
Si bien una aplicación monolítica se crea como una sola unidad indivisible, los microservicios dividen esa unidad en una colección de unidades independientes que contribuyen a un todo más amplio. Una aplicación se construye como una serie de servicios que se pueden implementar de forma independiente, están descentralizados y se desarrollan de forma autónoma.

[1] Una arquitectura monolítica es un modelo tradicional de un programa de software que se compila como una unidad unificada y que es autónoma e independiente de otras aplicaciones.

Los microservicios entran en la categoría de sistema distribuido. Un sistema distribuido es un conjunto de programas informáticos que utilizan recursos computacionales en varios nodos de cálculo distintos para lograr un objetivo compartido común. Estos sistemas consiguen mayor fiabilidad, rendimiento y facilidad de escalabilidad. Los nodos de un sistema distribuido ofrecen redundancia, de modo que, si un nodo falla, hay otros que pueden sustituirlos y reparar el error. Cada nodo se puede escalar de forma horizontal y vertical, lo que mejora el rendimiento. Si un sistema se somete a una carga extensiva, pueden añadirse nodos adicionales para ayudar a absorber dicha carga.

Los microservicios son los componentes básicos de una aplicación. Cada servicio suele contener una base de datos, una capa de acceso a los datos, lógica empresarial y una API. Como cada microservicio se ejecuta independientemente, su actualización, implementación y escalamiento es más sencillo, lo que ayuda a satisfacer la demanda de funciones específicas de una aplicación. Las API incluyen detalles de programación como el formato de los datos, las expectativas de intercambio de datos y los protocolos. La API, por su parte, conecta diferentes funciones o servicios dentro o fuera de una aplicación. El alcance de las API internas se limita a una sola aplicación.

La arquitectura de microservicios tiene como objetivo mejorar la eficiencia del desarrollo de software mediante la división de un bloque de código grande en varios servicios más pequeños. De esta manera, varios desarrolladores pueden trabajar en diferentes microservicios simultáneamente en función de las especificaciones acordadas.

Con los microservicios, las aplicaciones se dividen en sus elementos más pequeños e independientes entre sí, que funcionan en conjunto para completar las mismas tareas. Los microservicios ganaron popularidad en los últimos años gracias al avance de la tecnología en la nube, ya que son fundamentales para la optimización del desarrollo de aplicaciones basadas en este ecosistema.

Como el software se divide en módulos pequeños y bien definidos, los equipos pueden utilizar funciones para diferentes propósitos, reutilizando el código. De esta manera, una una aplicación puede evolucionar de manera autónoma, ya que los desarrolladores pueden crear nuevas funcionalidades sin necesidad de programar desde cero.

La arquitectura de microservicios se caracteriza por la descomposición de aplicaciones complejas en un conjunto de módulos ligeros, autónomos y débilmente acoplados, los cuales funcionan como unidades funcionales independientes. Este enfoque arquitectónico introduce implicaciones significativas en el ámbito tecnológico, al permitir que

cada microservicio sea aprovisionado, desplegado, monitoreado y actualizado de manera aislada, lo que contribuye a una mayor flexibilidad operativa y facilita las tareas de mantenimiento. En este contexto, el uso de tecnologías de virtualización a nivel de sistema operativo, como los contenedores, desempeña un papel fundamental, ya que posibilitan la creación de entornos reproducibles, la automatización de procesos y la reducción de la complejidad asociada a la configuración de infraestructuras. Asimismo, estos mecanismos favorecen la identificación y corrección eficiente de fallos en componentes específicos, lo que se traduce en una mejora sustancial en la confiabilidad del sistema. De igual forma, la adopción de microservicios se alinea con prácticas modernas de desarrollo de software, tales como la integración y entrega continua, promoviendo la agilidad en la incorporación de nuevas funcionalidades y el escalamiento eficiente de las aplicaciones.

Por otra parte, los microservicios pueden clasificarse, en términos generales, en sistemas con estado y sin estado, dependiendo de su capacidad para conservar información entre solicitudes. Para garantizar la interoperabilidad entre dichos servicios, resulta indispensable el uso de interfaces de programación de aplicaciones (API), las cuales permiten la exposición controlada de funcionalidades y facilitan la comunicación e intercambio de datos tanto entre servicios internos como con sistemas externos. Estas API pueden categorizarse en función de diversos criterios, tales como su audiencia, arquitectura o protocolo, incluyendo variantes como API privadas, públicas, de socios, así como implementaciones basadas en estándares como REST o SOAP. En conjunto, la integración de la arquitectura de microservicios con el uso de API constituye una estrategia adecuada para abordar la complejidad inherente a sistemas monolíticos, permitiendo mejorar la escalabilidad, la mantenibilidad y la eficiencia en el desarrollo de software a gran escala.

3.4 Tecnologías de Desarrollo Frontend

3.4.1 React.js

Es un marco de JavaScript que permite crear interfaces de usuario de manera rápida y eficiente al incluir archivos Java en el HTML. También se puede describir como una biblioteca de Java que se puede usar para crear una interfaz de usuario a través de diferentes componentes que ayudan a construir y definir la estructura de su aplicación.

Cada aplicación web de React.js se compone de componentes reutilizables que forman partes de la interfaz de usuario: podemos tener un componente independiente para nuestra barra de navegación, uno para el pie de página, otro para el contenido principal, etcétera.

Tener estos componentes reutilizables facilita el desarrollo porque no se tiene que repetir código recurrente. Solo tenemos que crear su lógica e importar el componente en cualquier parte del código donde sea necesario.

React.js también es una aplicación de una sola página. Por lo tanto, en lugar de enviar una solicitud al servidor cada vez que se debe renderizar una nueva página, el contenido de la página se carga directamente desde los componentes de React.js. Esto permite una renderización más rápida sin tener que recargar la página.

En la mayoría de los casos, la sintaxis utilizada para crear aplicaciones React.js se denomina JSX (JavaScript XML), que es una extensión de sintaxis de JavaScript. Esto nos permite combinar la lógica de JavaScript y la lógica de la interfaz de usuario de una manera única. Con JSX, eliminamos la necesidad de interactuar con el DOM mediante métodos como **document.getElementById**, **querySelectory** otros métodos de manipulación del DOM.

Estas son algunas de las características principales de React.js:

- JSX: Es una extensión de sintaxis de JavaScript que amplía las características de ES6 (ECMAScript 2015). Esto nos permite combinar la lógica y el marcado de JavaScript en un componente.
- DOM virtual: es una copia del objeto DOM que primero actualiza y vuelve a renderizar nuestras páginas cuando se realizan cambios; luego compara el estado actual con el DOM original para mantenerlo sincronizado con los cambios. Esto conduce a una renderización de páginas
- Componentes: Las aplicaciones React.js están compuestas por distintos componentes reutilizables que tienen su propia lógica e interfaz de usuario. Esto las hace eficientes para escalar aplicaciones y mantener un alto rendimiento porque no duplicas el código con tanta frecuencia como en otros marcos.

ReactJS puede mejorar la optimización en los motores de búsqueda (Search Engine Optimization, SEO) de las aplicaciones web al aumentar su rendimiento. SEO es una práctica para posicionar mejor en los motores de búsqueda. Permitiendo que los motores de búsqueda rastreen tu aplicación web, impulsando tu SEO.

3.4.2 Bootstrap y Diseño Responsivo

Bootstrap es una enorme colección de fragmentos de código reutilizables que pueden resultar muy útiles para los desarrolladores. Es un marco de desarrollo front-end gratuito y de código abierto para la creación de sitios web y aplicaciones web [19].

Fue creador por Mark Otto y Jacob Thornton en Twitter para mejorar la coherencia de las herramientas utilizadas en el sitio y reducir el mantenimiento. El software se conocía anteriormente como Twitter Blueprint y a veces se lo conoce como Twitter Bootstrap.
Bootstrap incluye los elementos básicos para el desarrollo web responsivo, por lo que los desarrolladores solo necesitan insertar el código en un sistema de cuadrícula predefinido. El framework combina CSS y JavaScript para estilizar los elementos de una página HTML. Permite mucho más que, simplemente, cambiar el color de los botones y los enlaces.

Esta es una herramienta que proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso y más.

Además de todas las características que ofrece el framework, su principal objetivo es permitir la construcción de sitios web responsivos para dispositivos móviles. Esto significa que las páginas están diseñadas para funcionar en escritorio, tabletas y teléfonos inteligentes, de una manera muy simple y organizada.

Hay un archivo principal llamado bootstrap.css, que contiene una definición para todos los estilos utilizados. Básicamente, la estructura del framework se compone de dos directorios:

- **css:** contiene los archivos necesarios para la estilización de los elementos y una alternativa al tema original.

- **js:** contiene la parte posterior del archivo bootstrap.js (original y minificado), responsable de la ejecución de aplicaciones de estilo que requieren manipulación interactiva.

Diseño responsive

Una de las características principales de Bootstrap es permitir que la adaptación de la página se realice según el tipo de dispositivo utilizado. Para garantizar la responsividad, el framework funciona con:

- La estilización del elemento.
- El uso del class container.

En la práctica, el elemento, funciona para crear una serie de notas, similar a una tabla, capaz de estructurar la página de forma adaptable. El elemento es flexible, ya que permite definir y cambiar el tamaño de la longitud fácilmente.

Bootstrap le ha asignado al elemento una característica de class container, que funciona para determinar las dimensiones apropiadas para los elementos insertados en ese espacio.

Este framework funciona con tres tipos de containers:

- Container: como un conjunto con una propiedad de ancho máximo, que determina qué tamaño de lienzo es ideal para crear el diseño de página.
- Container-fluid: considera la longitud total del lienzo del dispositivo para definir el diseño. Para esto, se considera la propiedad width 100 % en todos los límites de tamaño de lienzo.
- Container-breakpoint: considera ancho 100 % hasta alcanzar un cierto tamaño. La finalidad es crear contenedores responsivos, del ancho hasta que se alcanza un punto de ruptura específico (breakpoint). Después de ese breakpoint. el contenedor utiliza un ancho máximo predefinido para el tamaño de pantalla, lo que permite que el diseño se adapte a diferentes dispositivos.

Así bootstrap permite que una Página Web o aplicación detecte el tamaño y la orientación de la pantalla del visitante y adapte automáticamente la visualización en consecuencia.

Las principales características de Bootstrap se pueden describir de la siguiente manera:

- Bootstrap es un framework front-end gratuito, cuyo propósito es hacer que el desarrollo web sea más rápido y sencillo.

- También incluye plantillas de diseño basadas en HTML y CSS para formularios, tipografía, botones, navegación, tablas, modales, carruseles de imágenes y muchos otros componentes junto con otros complementos de JavaScript opcionales.
- Bootstrap también ofrece a los usuarios la posibilidad de crear fácilmente diseños responsivos.
- El CSS responsivo de Bootstrap también se ajusta fácilmente a teléfonos, tabletas y computadoras de escritorio.
- Es compatible con todos los navegadores modernos como Chrome, Firefox, Internet Explorer, Edge, Safari y Opera.

Bootstrap proporciona una variedad de componentes que se pueden incluir fácilmente en aplicaciones web, como:

- Menús desplegables
- Formularios
- Barras de navegación
- Botones
- Tablas
- Barras de progreso
- Miniaturas

Todos los componentes se verán con gran resolución sin importar el tamaño de la pantalla o el dispositivo que se use para verlos. Por lo tanto, proporciona una gran cantidad de funcionalidades listas para usar.

Cuadrícula adaptable

Bootstrap tiene su propio sistema de cuadrícula predefinido. Podemos empezar a llenar los contenedores con el contenido requerido.

Los usuarios pueden definir ciertos puntos de interrupción personalizados para cada columna, utilizando sus puntos de interrupción extra pequeños, pequeños, medianos, grandes y extragrandes. También tiene una opción predeterminada que se puede utilizar en la mayoría de los escenarios. Por lo tanto, la cuadrícula adaptable de este marco facilita la vida de los desarrolladores.

Personalizable
Los usuarios siempre pueden modificar el archivo CSS si no están satisfechos con la plantilla de diseño de Bootstrap. Además, se puede combinar con diseños existentes y, por lo tanto, pueden complementar las funciones de cada uno. Es muy útil cuando se requiere que la aplicación tenga un aspecto único, pero no se tiene suficiente tiempo para aprender o codificar CSS personalizado desde cero.

3.5 Tecnologías de Desarrollo Backend y Comunicación

3.5.1 Node.js y Express.js

Node.js es un entorno de ejecución de JavaScript multiplataforma y de código abierto. que ejecuta el motor JavaScript V8, el núcleo de Google Chrome, fuera del navegador, lo cual permite tener un gran rendimiento [1].

Por lo que, una aplicación Node.js se ejecuta en un único proceso, sin crear un nuevo hilo para cada solicitud. en lugar de bloquear el hilo y desperdiciar ciclos de CPU esperando, Node.js reanudará las operaciones cuando reciba la respuesta y proporciona un conjunto de primitivas para realizar una operación de E/S, como leer de la red, acceder a una base de datos o al sistema de archivos.

Esto permite que Node.js gestione miles de conexiones simultáneas con un solo servidor sin introducir la carga de gestionar la simultaneidad de subprocesos, lo que podría ser una fuente importante de errores.Gracias a primitivas de E/S asincrónas en su biblioteca estándar que evitan que el código JavaScript se bloquee de formar general, las bibliotecas en Node.js se escriben utilizando paradigmas no bloqueantes, lo que hace que el comportamiento bloqueante sea la excepción en lugar de la norma.

Node.js es un entorno de tiempo de ejecución de JavaScript, de ahí su terminación «.js». Este entorno de tiempo es open source, es decir, de código abierto, multiplataforma y que se ejecuta del lado del servidor. Creado en 2009 por los desarrolladores de JavaScript con el objetivo de ir un paso más allá con este lenguaje de programación.
Hasta la creación de Node.js, el lenguaje de programación JavaScript únicamente podía ejecutarse del lado del navegador o cliente,Como JavaScript únicamente se podía utilizar dentro del marco de las etiquetas `<script>` `<script>` , los desarrolladores tenían que usar diferentes lenguajes y herramientas tanto en el frontend como en el backend.

Node.js posee todo lo necesario para ejecutar código JavaScript del lado del servidor. Algo que facilita mucho el trabajo de los desarrolladores y el motivo por el que actualmente es una de las herramientas de trabajo más usadas en desarrollo web.

Uno de los puntos fuertes de Node.js es Event Loop, también conocido como bucle de eventos, un modelo que permite gestionar una gran cantidad de eventos de forma asíncrona, ya que se ejecutan de forma independiente y sin interferir unos en otros. Para conseguir esto, Node.js ha modificado la forma en la que se realizan las conexiones al servidor. En vez de generar un hilo para cada cliente, algo que resulta muy ineficaz debido al alto consumo de memoria RAM ante múltiples conexiones, utiliza un modelo en el que genera un evento para cada petición que se gestiona de manera independiente y sin bloqueos. Esta capacidad de dar respuesta a muchas más peticiones de forma concurrente hace que Node.js sea un entorno muy estable y con gran rendimiento, especialmente para proyectos de gran envergadura.

Figura 3.6: Interacción JavaScript, Node.js, Express.js Base de Datos
Integración de tecnologías en el lado del servidor. Se muestra el esquema de conectividad donde Node.js actúa como entorno de ejecución, Express.js como framework de aplicaciones y su relación directa con el almacenamiento de datos persistentes. Fuente: [28].

Express.js

Es un framework web minimalista y flexible que se ejecuta sobre Node.js, diseñado para facilitar el desarrollo de aplicaciones web y APIs del lado del servidor utilizando JavaScript. Su principal objetivo es simplificar la creación de servidores mediante una estructura clara para el manejo de rutas, solicitudes y respuestas HTTP, permitiendo definir endpoints de forma organizada y escalable.

Además, Express.js incorpora el uso de middleware, que son funciones intermedias capaces de procesar las peticiones antes de llegar a su destino final, lo que permite implementar funcionalidades como autenticación, validación de datos, manejo de errores y registro de actividad. Esta arquitectura modular favorece el desarrollo de aplicaciones mantenibles y extensibles.

Otra de sus ventajas es su amplia compatibilidad con diversas librerías y herramientas del ecosistema de JavaScript, como controladores de bases de datos, sistemas de autenticación y gestores de sesiones, lo que lo convierte en una opción ideal para la construcción de APIs REST. En conjunto, Express.js permite desarrollar sistemas backend eficientes, capaces de manejar múltiples solicitudes concurrentes y adaptarse a las necesidades de aplicaciones modernas.

Express.js gestiona de manera eficiente los endpoints definidos en el diseño del backend. Las razones principales para utilizar Express.js es que permite:

- **Robustez en el Enrutamiento:** Permite definir rutas claras y semánticas (ej. /api/usuarios, /api/tesis), facilitando la implementación de los métodos HTTP (GET, POST, PUT) de forma organizada.
- **Integración con Clean Architecture:** Su estructura minimalista no impone una organización rígida, lo que permite acoplarse perfectamente al modelo de capas. Esto asegura que los controladores de Express.js solo actúen como intermediarios, delegando la lógica a los casos de uso.
- **Gestión de Middleware:** Facilita la incorporación de capas de seguridad, validación de datos y manejo de errores antes de que las peticiones lleguen a la lógica de negocio, garantizando la integridad del sistema.
- **Escalabilidad y Rendimiento:** Al heredar el modelo asíncrono de Node.js, Express.js puede manejar múltiples solicitudes sin bloquear el servidor.
- **Conectividad con la Base de Datos:** Ofrece una integración fluida con diversos sistemas de gestión de bases de datos, permitiendo que la persistencia de la información sea rápida y segura.

3.5.2 Interfaz de Programación de Aplicaciones (API)

Una API, es un conjunto de protocolos que permiten que distintos componentes de software se comuniquen y transfieran datos. Los desarrolladores utilizan las API para salvar las brechas entre fragmentos pequeños y discretos de código con el fin de crear aplicaciones que sean potentes, resistentes, seguras y capaces de satisfacer las necesidades de los usuarios.

Se puede considerar una API como un acuerdo contractual entre un proveedor de servicios y un consumidor. El proveedor ofrece servicios como datos, funcionalidades o recursos, y el consumidor accede a estos servicios a través de la API.

Una API tiene varias responsabilidades para facilitar la comunicación entre un sistema de software y el mundo exterior. Define entradas, salidas, formatos de datos, protocolos y mecanismos de autenticación y autorización.

Las API funcionan compartiendo datos entre aplicaciones, sistemas y dispositivos. Esto sucede a través de un ciclo de solicitud y respuesta. La solicitud se envía a la API, que recupera los datos y los devuelve al usuario.

- **API CLIENTE**

  El cliente API es responsable de iniciar la conversación enviando la solicitud al servidor API. La solicitud se puede activar de muchas maneras. Por ejemplo, un usuario puede iniciar una solicitud API ingresando un término de búsqueda o haciendo clic en un botón. Las solicitudes API también pueden activarse por eventos externos, como una notificación de otra aplicación.

  Figura 3.7.Flujo de comunicación entre el cliente y el servidor en una API, mostrando el envío de solicitudes mediante métodos HTTP y la recepción de respuestas por parte del cliente

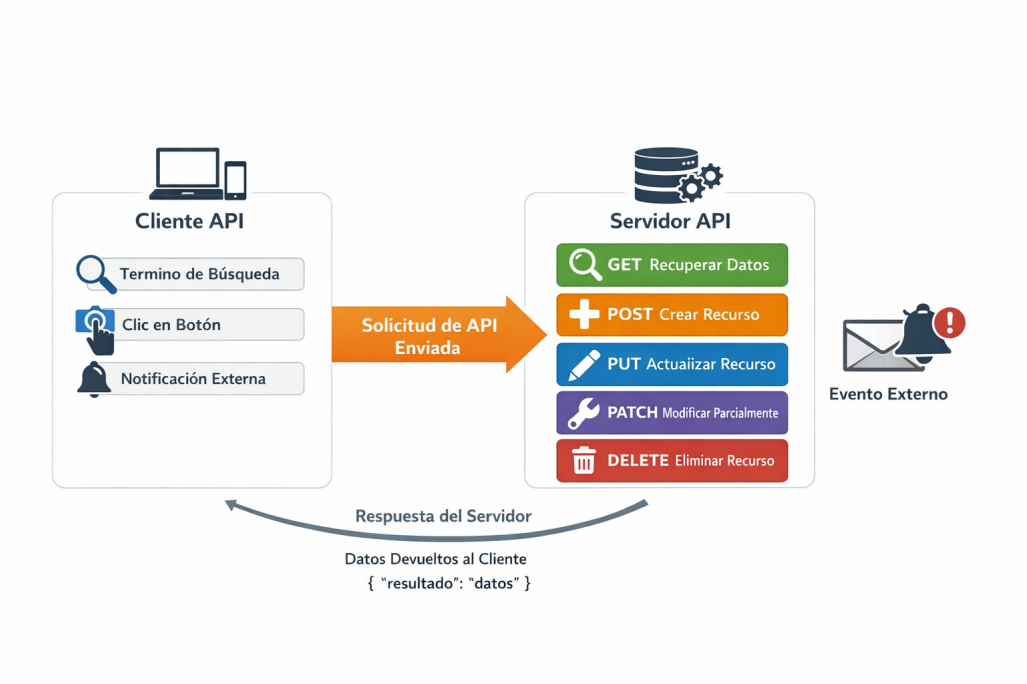


Figura 3.7: Envío de solicitudes.

- **SOLICITUD DE API**

  Una solicitud de API se verá y se comportará de manera diferente según el tipo de API, e incluirá los siguientes componentes:

- **Punto final:** Es una URL dedicada que brinda acceso a un recurso específico. Por ejemplo, el punto final /books en una aplicación de blogs incluiría la lógica para procesar todas las solicitudes relacionadas con los libros.

- **Método (HTTP Method):** Indica el tipo de operación que el cliente desea realizar sobre un recurso dentro de una API. Las API REST utilizan métodos HTTP estándar para ejecutar acciones específicas, entre los más comunes se encuentran: GET (recuperar información), POST (crear nuevos recursos), PUT (actualizar completamente un recurso), PATCH (actualizar parcialmente un recurso) y DELETE (eliminar un recurso). Estos métodos permiten estructurar de forma clara y consistente las operaciones sobre los datos.

- **Parámetros:** Los parámetros son las variables que se pasan a un punto final de API para proporcionar instrucciones específicas que la API debe procesar. Estos parámetros se pueden incluir en la solicitud de API como parte de la URL, en

la cadena de consulta o en el cuerpo de la solicitud. Por ejemplo, el punto final /books de una API de blogs podría aceptar un parámetro "topic", que utilizaría para acceder y devolver libros sobre un tema específico.

- **Encabezados de solicitud:** Son pares clave-valor que proporcionan detalles adicionales sobre la solicitud, como su tipo de contenido o credenciales de autenticación.

- **Cuerpo de la solicitud (Request Body):** Es la sección principal de una petición HTTP que contiene los datos necesarios para crear, actualizar o eliminar un recurso dentro de un sistema. En este espacio se envía la información relevante que el servidor requiere procesar, usualmente en un formato estructurado como JSON, permitiendo llevar a cabo las operaciones solicitadas de manera adecuada.

- **Servidor API**

  Es el componente encargado de gestionar la lógica de negocio del sistema, incluyendo la autenticación de usuarios, la validación de los datos de entrada y la recuperación o manipulación de la información almacenada en la base de datos. Además, procesa las solicitudes realizadas por el cliente (frontend) y envía las respuestas correspondientes, generalmente en formato JSON, asegurando una comunicación eficiente y estructurada entre la interfaz de usuario y el sistema backend.

Respuesta de la API

El servidor API envía una respuesta al cliente. La respuesta API incluye los siguientes componentes:

- **Código de estado:** Los códigos de estado HTTP son códigos de tres dígitos que indican el resultado de una solicitud de API. Algunos de los códigos de estado más comunes incluyen 200 OK, que indica que el servidor devolvió correctamente los datos solicitados, 201 Created, que indica que el servidor creó correctamente un nuevo recurso, y 404 Not Found, que indica que el servidor no pudo encontrar el recurso solicitado.

- **Encabezados de respuesta:** Los encabezados de respuesta HTTP son muy similares a los encabezados de solicitud, excepto que se utilizan para proporcionar

información adicional sobre la respuesta del servidor.

- **Cuerpo de la respuesta:** Incluye los datos o el contenido real que solicitó el cliente, o un mensaje de error si algo salió mal.

3.6 Gestión de Datos e Infraestructura

3.6.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, orientado a objetos, que utiliza el lenguaje SQL como estándar para la manipulación y consulta de información. Se caracteriza por su alta confiabilidad, extensibilidad y rendimiento, permitiendo el manejo eficiente de grandes volúmenes de datos. Además, incorpora mecanismos avanzados de seguridad, control de concurrencia y soporte para diversos tipos de datos, lo que lo convierte en una solución robusta y versátil para el desarrollo de aplicaciones modernas.
Algunas características técnicas:

- **Integridad y Robustez de Datos:** PostgreSQL es reconocido por su estricto cumplimiento de las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que garantiza que la información académica y administrativa se almacene de forma segura y sin corrupciones.
- **Escalabilidad para Entornos Académicos:** Su capacidad para manejar grandes volúmenes de datos y consultas complejas lo hace ideal para un sistema que crecerá con el tiempo a medida que se registren más alumnos, tesis y servicios sociales.
- **Compatibilidad con Node.js y Express.js:** Como se observa en el flujo de la Figura 3.6, PostgreSQL se integra de manera eficiente con entornos de desarrollo basados en Node.js y el framework Express.js, mediante el uso de librerías especializadas como pg y ORMs como Sequelize. Esta integración permite establecer una comunicación fluida entre el servidor y la base de datos, facilitando la ejecución de consultas asíncronas de alto rendimiento. Como resultado, se optimiza el manejo de múltiples solicitudes concurrentes y se reducen posibles cuellos de botella en el sistema.
- **Gestión Profesional con pgAdmin:** El uso de esta herramienta facilita el diseño del esquema, la escritura de QUERIES complejas y el monitoreo del rendimiento de la base de datos, permitiendo una administración visual y eficiente del Backend.

- **Soporte para Tipos de Datos Avanzados:** Además de los tipos de datos relacionales estándar,PostgreSQL permite el manejo de JSONB, lo cual ofrece flexibilidad en caso de que ciertos módulos del sistema requieran almacenar datos semiestructurados sin perder la potencia de una base SQL.

3.6.2 Contenedores y Docker

Contenedores

Los contenedores son componentes estandarizados y ejecutables que combinan el código fuente de la aplicación con las bibliotecas del sistema operativo (SO) y las dependencias necesarias para ejecutar ese código en cualquier entorno. [22]

Los contenedores son una forma de virtualización del sistema operativo. Un solo contenedor se puede usar para ejecutar cualquier cosa, desde un microservicio o un proceso de software a una aplicación de mayor tamaño. Dentro de un contenedor se encuentran todos los ejecutables, el código binario, las bibliotecas y los archivos de configuración necesarios. Sin embargo, en comparación con los métodos de virtualización de máquinas o servidores, los contenedores no contienen imágenes del sistema operativo. Esto los hace más ligeros y portátiles, con una sobrecarga significativamente menor. En implementaciones de aplicaciones de mayor tamaño, se pueden poner en marcha varios contenedores como uno o varios clústeres de contenedores.

Estos clústeres se pueden gestionar mediante un orquestador de contenedores, como Kubernetes[23].
En ocasiones, las personas confunden la tecnología de contenedores con Virtual Machine (VM) o con la tecnología de virtualización de servidores. Aunque existen algunas similitudes básicas, los contenedores son muy diferentes de las máquinas VM. El propósito de los contenedores es ejecutar varios procesos y aplicaciones por separado para que se pueda aprovechar mejor la infraestructura y, al mismo tiempo, conservar la seguridad que se obtendría con los sistemas individuales.

Cada contenedor comparte el mismo sistema operativo host o kernel del sistema y tiene un tamaño mucho menor, a menudo de sólo unos megabytes. Esto suele implicar que un contenedor puede tardar unos segundos en iniciarse (en comparación con los gigabytes y los minutos necesarios que requiere una VM “máquina Virtual” típica).

Docker

Docker es una plataforma de código abierto que permite a desarrolladores crear, desplegar, ejecutar, actualizar y administrar contenedores.

Docker es un sistema operativo para contenedores. Docker se instala en cada servidor y proporciona instrucciones sencillas que puede utilizar para crear, iniciar o detener contenedores.

La tecnología Docker utiliza el kernel de Linux y sus funciones, como los grupos de control y los espacios de nombre, para dividir los procesos y ejecutarlos de manera independiente.

El enfoque de Docker sobre la organización en contenedores se centra en la capacidad de separar una parte de la aplicación para actualizarla o repararla, sin necesidad de deshabilitar por completo. Además de aprovechar este modelo basado en los microservicios, puede intercambiar procesos entre varias aplicaciones casi de la misma forma en que funciona la arquitectura orientada a los servicios (SOA).

Un contenedor Docker es un paquete de software liviano, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, tiempo de ejecución, herramientas del sistema, bibliotecas del sistema y configuraciones. Docker es una plataforma de código abierto que permite crear, probar e implementar aplicaciones en contenedores [2].

[2]Las VM se ejecutan en un entorno de hipervisor en el que cada máquina virtual debe incluir su propio sistema operativo invitado dentro del mismo, junto con sus archivos bi narios, bibliotecas y archivos de aplicaciones correspondientes. Esto consume una gran cantidad de recursos y genera mucha sobrecarga, especialmente cuando se ejecutan varias VM en el mismo servidor físico, cada una con su propio sistema operativo invitado.

Capítulo 4

Evolución del hipertexto (HTML) y los Portales

4.1 Hipertexto

Hipertexto es un texto que está vinculado a otra locación en el mismo documento o a diferentes documentos en la misma computadora o en Internet. Es una herramienta con estructura no secuencial que permite crear, agregar, enlazar y compartir información de diversas fuentes por medio de enlaces asociativos.

Es un sistema para accesar información organizado de forma que pueda vincular lexías o fragmentos de datos que están interconectados de una manera no lineal por medio de lo que conocemos como hipervínculos (hyperlinks en inglés).

El concepto de hipertexto [10] fue creado por Vannevar Bush, un estadounidense que con la invención de Memex, un dispositivo que servía como base de datos y que posteriormente dió posibilidad de interactuar con los usuarios, permitió mecanizar y conectar la información con el fin de aumentar el desarrollo en aquella época [3].

Ted Nelson, en 1965, fue el primero en acuñar la palabra "hypertext", su propuesta Xanadu, un sistema que permite que un mismo documento aparezca en múltiples contextos sin tener que haber sido duplicado.

El hipertexto ha sido una de las principales ideas de la cibernética que ha llevado a las computadoras no a ser solo máquinas de cálculo o contabilidad, sino una herramienta para poder crear bases de datos textuales donde se puedan hacer lecturas no lineales y facilitar el análisis de información.
Para Robert Balzer en su obra Hypertext and Software Engineering, publicada en 1989, De Madrid, U. C. (s. f.). Hipertexto. `http://www.hipertexto.info/documentos/internet.htm` hipertexto es:
"una base de datos que tiene referencias cruzadas y permite al usuario (lector) saltar hacia otra parte de la base de datos, si este lo desea " .

[3]Hypertext Robert Balzer\

Tras el diseño de Berners-Lee de la WWW, se integra un consorcio internacional llamado World Wide Web Consortium (W3C) para desarrollar protocolos como el Lenguaje de marcas de hipertexto o HTML y Cascading Style Sheets o CSS por sus siglas en inglés. La más importante característica de HTML radica en que consiste solamente de texto. No contiene tablas, ni tipografías ni imágenes. Pero contienen las instrucciones para las tablas, las tipografías y las imágenes.

El diseño final no muestra el archivo HTML, este documento solo describe a través de marcas, el diseño de la página. HTML es un protocolo de diseño gráfico, con el cual se representan todas las páginas de internet.

La Figura (Figura **??**) muestra una línea de tiempo con algunos de los acontecimientos más importantes en el desarrollo del hipertexto y la World Wide Web, destacando innovaciones que permitieron la evolución de las páginas web y el acceso a la información en internet.

Cuadro 4.1: Cronología del Hipertexto

Año	**Innovación**
1945	**Memex**: Vannevar Bush presenta la idea de enlaces asociativos para acceder y conectar información.
1965	**Ted Nelson** acuña el término "hipertexto.ªl definir el concepto de enlaces navegables entre textos.
1972	**Tim Berners-Lee** desarrolla el **Aspen Movie Map**, primer sistema hipermedia que funciona como videodisco interactivo.
1987	**Apple** lanza **HyperCard**, primera herramienta mainstream de gestión de hipertextos para Macintosh.
1990	**Tim Berners-Lee** crea el **World Wide Web (WWW)** y el protocolo HTTP, sentando las bases de la Web moderna.
1990	**Marc Andreessen** lanza World Wide Web (WWW) y el protocolo HTTP, sentando las bases de la Web moderna.
1993	**Tim Berners-Lee** crea el World Wide Web (WWW) y el protocolo HTTP, sentando las bases de la Web moderna.
1993	**Marc Andreessen** lanza **Mosaic**, primer navegador gráfico que populariza el acceso a la Web para el público general.

4.2 Fundamentos Tecnológicos y Estándares Web

Para el desarrollo del sistema LACECI, bajo el modelo incremental, se seleccionó un stack tecnológico basado en JavaScript. En el extremo del cliente (frontend) se emplea la librería **React.js** para la gestión de componentes, mientras que en el servidor (Backend) se utiliza **Node.js** con el framework **Express**. Esta combinación permite una comunicación fluida mediante una arquitectura de **APIs** y **Microservicios**, garantizando que cada incremento del sistema sea escalable y modular. La persistencia de datos se gestiona a través de **PostgreSQL**, asegurando la integridad de la información académica.

1. **HTML5: Estructura Semántica y Accesibilidad**

   En este proyecto, HTML5 no se utiliza únicamente como un lenguaje de marcado, sino como la base de una arquitectura de información robusta. Su importancia radica en:

   - **Semántica Web:** Permite que la información sea interpretada correctamente por motores de búsqueda y tecnologías asistivas, mejorando la inclusión de usuarios con discapacidades.
   - **Integración Multimedia:** Facilita la incorporación de recursos técnicos del laboratorio sin depender de complementos externos.
   - **Compatibilidad:** Asegura que el portal sea funcional en cualquier navegador moderno.
   - **CSS3: Experiencia de Usuario (UX) y Diseño Responsivo**

   La implementación de CSS3 es crítica para transformar una herramienta funcional en una solución profesional. Su relevancia se divide en:

   - **Diseño Adaptativo (Responsive Design):** Garantiza que la interfaz se ajuste automáticamente a móviles, tablets o computadoras.
   - **Optimización de la Carga:** Utiliza capacidades nativas (Flexbox/Grid) para reducir el peso del portal y agilizar la gestión administrativa.
   - **Separación de Conceptos:** Facilita actualizaciones estéticas sin necesidad de reprogramar la lógica interna del sistema.

4.3 Evolución del Diseño Web: De las Primeras Páginas a la Interactividad

El diseño web ha recorrido un camino de transformación constante desde el surgimiento de las primeras páginas web en la década de los noventa. Inicialmente,se caracterizaban por ser documentos estáticos, basados únicamente en texto e hipervínculos simples, con capacidades gráficas limitadas que apenas permitían una interacción básica con el usuario.
Para llegar a los estándares actuales de modernidad, el diseño web atravesó etapas clave de evolución tecnológica que permitieron la transición de la información estática a la experiencia dinámica:

- **El surgimiento de la estética visual:** Con el avance de los navegadores, las páginas web comenzaron a integrar tablas y marcos para organizar el contenido, sentando las bases de lo que hoy conocemos como arquitectura de información.
- **La era de la interactividad (Flash):** A partir de 1996, el diseño experimentó un cambio radical con *Adobe Flash*. Esta plataforma permitió crear sitios animados y dinámicos sin depender de código complejo. Aunque revolucionó el sector, su uso disminuyó debido a limitaciones en seguridad y falta de compatibilidad con dispositivos móviles.
- **El auge de JavaScript y CSS:** Paralelamente, la introducción de JavaScript permitió adaptar el comportamiento de los sitios a las acciones del usuario. Con la llegada de HTML3 y las primeras hojas de estilo (CSS), el diseño web permitió separar por primera vez el contenido (estructura) de la presentación visual, optimizando el desarrollo.
- **Diseño Web Moderno:** Actualmente, el diseño se fundamenta en estándares abiertos (HTML5, CSS3 y JavaScript moderno), priorizando la velocidad de carga. Esta evolución es la que permite que el portal del laboratorio LACECI, sea una herramienta profesional y accesible desde cualquier navegador contemporáneo.

4.4 Páginas Web Dinámicas y la Tríada del Desarrollo Front-end

A partir de los avances mencionados en la evolución del diseño web, surgió la necesidad de superar las limitaciones de los sitios estáticos. La demanda de interactividad creció debido a que las instituciones, como el laboratorio LACECI, requerían sistemas capaces de gestionar flujos de información constantes, registros de usuarios y consultas en tiempo real. Esto dio lugar al concepto de **páginas web dinámicas**.

Una Página Web dinámica se distingue por su capacidad de generar contenido en respuesta a las acciones del usuario o a datos almacenados.

Para lograr este dinamismo, es fundamental la interacción coordinada de la llamada "tríada del desarrollo web " en el lado del cliente (**front-end**):

- **HTML (Estructura de Datos):** En una página dinámica, el HTML funciona como un contenedor flexible. Ya no es un documento fijo, sino una estructura que recibe y organiza la información que proviene de una base de datos, permitiendo que el portal muestre resultados de búsqueda o perfiles de usuario de forma personalizada.
- **CSS (Presentación Adaptable):** Su función es garantizar que, independientemente de qué tan variable sea el contenido dinámico, la interfaz mantenga su integridad visual. CSS permite que el diseño se adapte al volumen de datos mostrados, asegurando que la experiencia del usuario sea profesional y legible en todo momento.
- **JavaScript (Lógica y Comportamiento):** Es el componente que otorga la interactividad real. JavaScript permite que la página reaccione sin necesidad de recargar el navegador, gestionando desde la validación de formularios de contacto hasta la actualización instantánea de la información administrativa del laboratorio.

Este ecosistema visual se complementa con entornos de ejecución del lado del servidor, como **Node.js**. A diferencia de los lenguajes tradicionales como PHP, Node.js permite utilizar JavaScript en el servidor, actuando como un motor de alto rendimiento que conecta la interfaz con bases de datos para procesar solicitudes complejas. Gracias a esta arquitectura de **JavaScript unificado (Fullstack)**, LACECI trasciende de ser un simple sitio informativo para convertirse en una solución tecnológica funcional que facilita la administración automatizada.

4.5 La culminación del CSS y su arquitectura técnica

El CSS [12] fue desarrollado por el consorcio W3C en 1996 para separar la estructura de la presentación. Se denomina "Hojas de Estilo en Cascada", porque las reglas se aplican en un orden jerárquico de prioridad.

Para comprender su funcionamiento, es necesario definir sus componentes técnicos fundamentales:

- **Regla CSS:** Es la unidad básica de estilo, compuesta por un **selector** y un bloque de declaraciones (propiedades y valores) encerrados en llaves.
- **Selector y tipos:** Es el elemento que indica a qué parte del HTML se aplicará el estilo.
- **Pseudo-clases y Pseudo-elementos:** Permiten estilar estados especiales. Las **pseudo-clases** (como `:hover`) controlan el aspecto cuando el usuario interactúa con un elemento, mientras que los **pseudo-elementos** (como `::before`) permiten insertar o estilar partes específicas de un componente sin modificar el HTML.
- **Media Queries:** Es la tecnología que permite que el CSS sea condicional. Gracias a ellas, el portal puede detectar el tamaño de la pantalla y aplicar reglas distintas, lo que constituye la base del diseño responsivo.

El año 2005 marcó un hito, pues la madurez de estos estándares permitió mejorar la creatividad y reducir drásticamente los tiempos de carga al evitar el uso de imágenes pesadas para crear bordes o sombras, delegando esa tarea al código puro.

4.6 Diseño web centrado en mobile first

Hasta 2008, el diseño se centraba en ordenadores de escritorio. Sin embargo, la penetración masiva de dispositivos móviles obligó a una transición hacia la estrategia **Mobile First**. Bajo este enfoque, el diseño se crea primero para pantallas pequeñas y luego se expande hacia monitores de escritorio.
Esta filosofía garantiza que el 100 % de las funcionalidades de los portales, como el de LACECI sean operativas desde un teléfono inteligente. Considerando que más del 50 % de la población mundial utiliza estos dispositivos para navegar, el uso de configuraciones **responsive** no es solo una opción estética, sino una necesidad de accesibilidad que Google prioriza en su posicionamiento desde 2016.

4.7 Portales Web

En la actualidad, el ecosistema web ha evolucionado hacia la simplicidad con tendencias como el *Flat Design* y el uso de librerías como **React.js** o **Angular** para crear interfaces altamente interactivas.

Tecnologías como las Aplicaciones Web Progresivas (PWA) y WebAssembly permiten que las páginas web alcancen un rendimiento cercano al de las aplicaciones nativas.

Concepto y Arquitectura de un Portal Web

A diferencia de un sitio web convencional, el proyecto desarrollado para el LACECI se define técnicamente como un **Portal Web**. Un portal es un punto de acceso unificado que centraliza servicios y recursos de manera organizada, permitiendo la interacción personalizada según el perfil del usuario. Sus características clave implementadas en este trabajo son:

1. **Centralización de servicios:** El portal aglutina herramientas de consulta, gestión de inventarios y comunicación en una sola interfaz.

2. **Autenticación y Seguridad:** Implementa mecanismos de inicio de sesión para proteger la integridad de los datos administrativos del laboratorio.

3. **Gestión de Roles y Permisos:** Esta es la característica principal que permite la administración automatizada del laboratorio. Para el portal LACECI se han definido tres niveles de acceso:

    - **Administrador:** Posee control total sobre la plataforma. Sus facultades incluyen el registro de nuevos usuarios, edición de contenido general, carga de archivos, gestión de perfiles, así como la administración de eventos, directorio y fechas de registro.

    - **Profesor:** Cuenta con permisos para registrar alumnos y eventos específicos. Asimismo, puede integrarse al directorio del laboratorio y gestionar su información de perfil personal.

    - **Estudiante:** Los alumnos son registrados previamente por un superior (profesor o administrador). Tienen permisos para gestionar su perfil personal y subir archivos específicos, como documentos de tesis o servicio social, manteniendo acceso de lectura a todo el contenido del portal sin permisos de edición global.

4. **Integración de información:** Capacidad de conectar datos de diferentes fuentes (bases de datos de equipos, calendarios de uso, etc.) para presentarlos de forma coherente al usuario según su rol.

La gestión de estos roles y permisos se materializa a través de una API REST desarrollada en Node.js, la cual valida la identidad de cada usuario mediante tokens de seguridad, garantizando que cada perfil acceda únicamente a las funciones que le corresponden según su jerarquía en el laboratorio.

Capítulo 5

Metodología y Desarrollo del Sistema

5.1 Marco Metodológico: Modelo Incremental

Para el desarrollo del portal académico LACECI, se implementó el **Modelo Incremental**. Esta metodología se basa en la filosofía de “construir y entregar", donde el sistema se divide en varios módulos o componentes que se desarrollan y entregan de manera sucesiva.
Cada incremento añade una funcionalidad completa al sistema, permitiendo que el proyecto evolucione desde una estructura básica hasta el portal integral final.

5.1.1 Justificación de la Metodología

La elección de este modelo formal responde a las necesidades específicas del laboratorio y a las observaciones de los usuarios clave durante la fase de análisis:

- **Gestión de Riesgos:** Al ser un sistema que maneja diferentes niveles de acceso (Administrador, Profesor, Alumno), el modelo incremental permitió asegurar primero la robustez del módulo de autenticación antes de proceder con la gestión de archivos sensibles como las tesis.
- **Validación Continua:** Cada incremento fue sometido a pruebas de usuario. Esto permitió que los administradores del LACECI retroalimentaran el diseño de la interfaz (Mockups) antes de finalizar el desarrollo del Backend.
- **Adaptabilidad:** Debido a que los requerimientos de un laboratorio académico pueden variar (nuevos tipos de eventos o cambios en los formatos de servicio social), esta metodología permitió realizar ajustes sin comprometer la estructura global del software.

5.1.2 Estrategia de Incrementos para LACECI

El desarrollo se estructuró en tres etapas o incrementos principales, los cuales aseguraron una transición fluida de los procesos manuales del laboratorio hacia la automatización digital:

1. **Incremento 1 (Núcleo del Sistema):** Desarrollo de la arquitectura base, conexión con la base de datos PostgreSQL y el módulo de seguridad para el inicio de sesión por roles.

2. **Incremento 2 (Gestión Administrativa):** Implementación de las funcionalidades para el Administrador y Profesor, incluyendo el registro de usuarios, el directorio dinámico y el calendario de eventos.

3. **Incremento 3 (Servicios al Estudiante):** Integración del repositorio documental para la carga de archivos de tesis y servicio social, junto con la interfaz de consulta general para alumnos.

5.2 Fase 1: Análisis y Definición de Requerimientos

Esta etapa constituyó el pilar fundamental del proyecto, ya que permitió alinear las capacidades del portal con las carencias administrativas identificadas en LACECI. El análisis se realizó mediante un enfoque participativo, involucrando a los actores clave para obtener una visión completa de las necesidades del laboratorio.

5.2.1 Levantamiento de Información

Se realizaron entrevistas semiestructuradas y sesiones de observación directa con el personal administrativo y colaboradores de LACECI.

El guión de las entrevistas y el registro de las respuestas obtenidas se encuentran detallados en el **Anexo: Entrevistas**.

A través de este diagnóstico, se identificó que la principal debilidad era la fragmentación de la información y la dependencia de procesos manuales.

Como resultado de este proceso de consulta, se generó un listado formal de necesidades técnicas, el cual puede consultarse íntegramente en el **Anexo A.3: Requerimientos del Portal**.

5.2.2 Especificación de Requerimientos

Basado en la documentación del Anexo A.3, se categorizaron las necesidades en requerimientos técnicos que sirvieron de base para el desarrollo del software:

Requerimientos Funcionales (RF)

Extraídos del catálogo de requerimientos validado, se destacan:

- **RF1 - Gestión de Identidades:** El sistema debe permitir el registro y edición de perfiles diferenciados por roles, asegurando que solo alumnos logueados puedan subir archivos (véase requerimiento en Anexo A.3).

- **RF2 - Repositorio Documental:** El portal debe permitir la carga de archivos en formato PDF y LaTeX para trámites de tesis y servicio social, con capacidad de borrado exclusiva para el administrador.

- **RF3 - Gestión de Contenido Multimedia:** Capacidad de cargar avisos, fechas y vídeos (formatos AVI, MP4) para la sección de conferencias y seminarios.

- **RF4 - Directorio Institucional:** El home debe incluir un directorio de profesores con liga a su ficha curricular institucional.

Requerimientos No Funcionales (RNF)

- **Identidad Institucional:** El diseño debe integrar obligatoriamente el logo del laboratorio y los colores oficiales de la universidad.

- **Accesibilidad Visual:** La tipografía y el tamaño de letra deben garantizar la legibilidad a una distancia mínima de 30 cm.

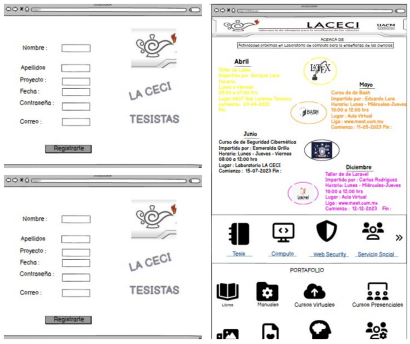
- **Seguridad:** Restricción de acceso a carpetas de archivos según el estado de logueo del usuario.

5.3 Fase 2: Diseño del Sistema y Arquitectura

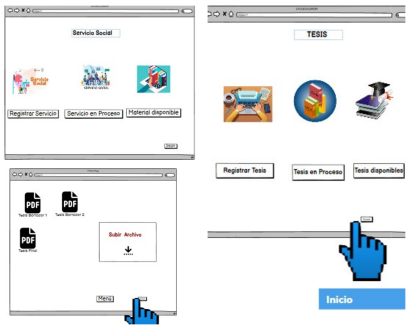
En esta fase se transformaron los requerimientos del Anexo A.3 en planos técnicos. El diseño se dividió en tres dimensiones: la interfaz de usuario, la estructura de datos y la arquitectura del servidor.

5.3.1 Diseño de la Interfaz (UX/UI)

Antes de la codificación, se diseñaron prototipos de baja fidelidad utilizando la herramienta *Balsamiq Mockups*, en el anexo mockups se encuentran todos los prototipos realizados . El objetivo fue validar la disposición de los elementos y la navegación por roles antes de implementar el código final.
Como se observa en la Figura 5.1, se prioriza una estructura limpia para el *Home* y una sección de gestión documental intuitiva para los módulos de tesis y servicio social.



(a) Mockups de páginas Principales



(b) Mockups de tesis y servicio social

Figura 5.1: Mockups portal LACECI. En esta imagen se presenta el diseño del módulo de autenticación y la interfaz principal del portal.

5.3.2 Diseño de la Base de Datos

Para soportar la persistencia de la información, se diseñó un modelo relacional en PostgreSQLL. Este diseño asegura que cada archivo cargado esté vinculado a un autor, una fecha y un tipo de trámite específico, cumpliendo con los requerimientos de integridad definidos anteriormente.

*El diagrama Entidad-Relación detallado se presenta en el **Anexo A: Diagramas**, donde se especifican las tablas de Usuarios, Profesores, Tesis y Eventos.*

5.3.3 Diseño de Backend y API REST

El sistema se basa en una arquitectura de servicios. Se definieron *endpoints* específicos para que el frontend pueda comunicarse con la base de datos de manera segura. En la tabla 5.1 se presentan los principales endpoints implementados en la API REST del portal LACECI. Cada endpoint permite realizar operaciones de consulta, registro y actualización de información relacionada con usuarios, alumnos, profesores, tesis, servicio social y eventos dentro del sistema.

se detallan los métodos HTTP (GET, POST, PUT) utilizados para la gestión de usuarios y documentos académicos.

Método	Endpoint	Descripción
GET	/api/usuarios	Obtiene todos los usuarios registrados
GET	/api/alumnos	Obtiene todos los alumnos
GET	/api/profesores	Obtiene todos los profesores
GET	/api/tesis	Obtiene todas las tesis
GET	/api/serviciosocial	Obtiene todos los servicios sociales
GET	/api/eventos	Obtiene los datos de los eventos
GET	/api/usuarios/{id}	Obtiene los datos de un usuario específico
GET	/api/alumnos/{id}	Obtiene los datos de un alumno específico
GET	/api/profesores/{id}	Obtiene los datos de un profesor específico
GET	/api/tesis/{id}	Obtiene los datos de una tesis específica
POST	/api/usuarios	Registra un nuevo usuario
POST	/api/alumnos	Registra un nuevo alumno
POST	/api/profesores	Registra un nuevo profesor
POST	/api/tesis	Registra una nueva tesis
POST	/api/serviciosocial	Registra un nuevo servicio social
POST	/api/eventos	Registra un nuevo evento
PUT	/api/usuarios/{id}	Actualiza la información de un usuario
PUT	/api/alumnos/{id}	Actualiza la información de un alumno
PUT	/api/profesores/{id}	Actualiza la información de un profesor
PUT	/api/tesis/{id}	Actualiza la información de una tesis
PUT	/api/serviciosocial/{id}	Actualiza la información de un servicio social
PUT	/api/eventos/{id}	Actualiza la información de un evento

Cuadro 5.1: Endpoints principales de la API REST implementada en el portal LACECI.

5.3.4 Arquitectura de Software (Clean Architecture)

Se adoptó un modelo de arquitectura en capas (véase Figura 5.2). Este enfoque permite separar la lógica de negocio de los detalles tecnológicos (como el motor de base de datos o el framework web).

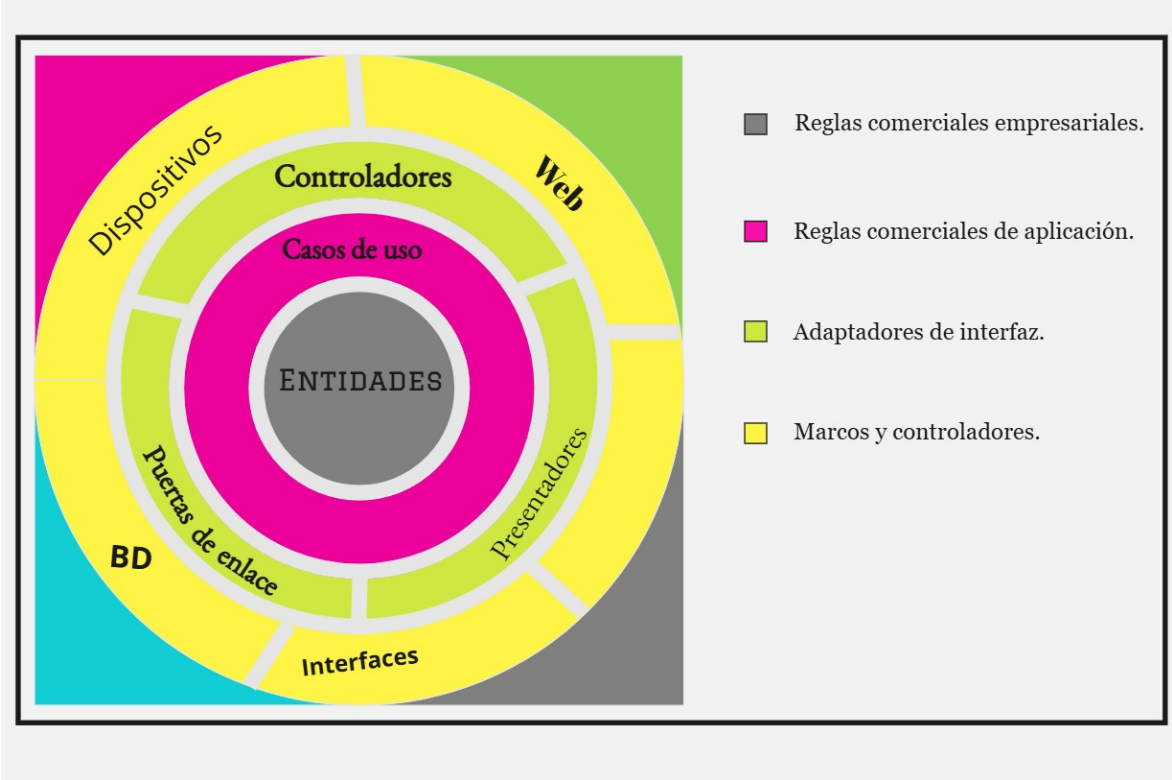


Figura 5.2: Modelo de arquitectura en capas adoptado.

Esta separación garantiza que el portal sea escalable y fácil de mantener, permitiendo actualizaciones en la interfaz sin afectar el núcleo del sistema.

5.4 Fase 3: Desarrollo y Configuración

Para el desarrollo del portal web LACECI se definió un conjunto de tecnologías y herramientas que conforman el stack tecnológico del sistema. Estas tecnologías fueron seleccionadas considerando criterios como escalabilidad, mantenimiento, rendimiento y compatibilidad entre los distintos componentes del sistema. La arquitectura implementada sigue un enfoque moderno de desarrollo web, integrando tecnologías tanto del lado del cliente como del servidor, así como herramientas de infraestructura que permiten una implementación modular y eficiente. En la Tabla 5.2 se presenta el conjunto

de tecnologías utilizadas durante la fase de desarrollo y configuración del sistema, las cuales permitieron construir una aplicación web robusta, dinámica y adaptable a las necesidades del portal.

Componente	**Tecnología / Herramienta**
Lenguaje de Programación	JavaScript / ES6+ (Fullstack)
Framework Frontend	React.js
Framework Backend	Express.js sobre Node.js
Arquitectura	Microservicios y API REST
Framework de Estilos	Bootstrap 5 / CSS3
Gestor de Base de Datos	PostgreSQL
Infraestructura	Docker y Contenedores
Editor de Código	Visual Studio Code

Cuadro 5.2: Stack tecnológico final utilizado para el desarrollo del Portal LACECI.

La programación se realizó de forma modular, permitiendo validar cada funcionalidad de manera independiente.

Los principales incrementos fueron:

- **Módulo de autenticación:** Desarrollo del login seguro que identifica si el usuario es Administrador, Profesor o Alumno.
- **Gestor de archivos:** Programación de las funciones de subida para archivos **.pdf y .tex**, asegurando que solo el administrador tenga permisos de eliminación (según requerimiento en Anexo A.3).

En esta fase se llevó a cabo la codificación de los módulos siguiendo el diseño de incrementos.
La programación se realizó de forma modular, aprovechando las capacidades asíncronas de **Node.js** y la creación de interfaces basadas en componentes de **React.js**. Se utilizó un entorno de desarrollo entorno de desarrollo basado en **Docker**, lo que facilitó la integración de los **microservicios** y aseguró que la conexión con **PostgreSQL** fuera estable y escalable desde las primeras fases del desarrollo.

5.4.1 Implementación de la Lógica de Negocio

El desarrollo se centró en asegurar que las reglas definidas en el análisis de requerimientos se ejecutarán correctamente, tales como la restricción de borrado de archivos exclusivamente para el administrador y la organización alfabética de los documentos.

5.4.2 Integración de la Base de Datos

Se realizó el mapeo del diagrama relacional (ver Anexo: Diagramas) a las tablas físicas en el motor de base de datos, asegurando la integridad referencial y el correcto almacenamiento de metadatos (título, fecha y autor) para cada archivo subido.

5.4.3 Control de Calidad y Pruebas

El primer incremento se enfocó en el control de acceso. La Figura 5.3 muestra la implementación final del módulo de autenticación.

- **Módulo de autenticación:** Desarrollo del login seguro que identifica si el usuario es Administrador, Profesor o Alumno.
- **Gestor de archivos:** Programación de las funciones de subida para archivos **.pdf y .tex**, asegurando que solo el administrador tenga permisos de eliminación.
- **Componente de Interfaz Dinámica (Carousel):** Implementación de una galería de imágenes tipo *carousel* en el Home para mostrar fotos de eventos del laboratorio, permitiendo al administrador actualizar el contenido visual.
- **Gestor de Eventos y Avisos:** Creación de tablas dinámicas para la publicación de actividades académicas. Estas tablas incluyen campos obligatorios según el análisis: nombre del evento, horario, lugar, fecha e imagen publicitaria.

La lógica para la gestión de archivos se implementó en el backend mediante el uso de librerías de Node.js especializadas en el manejo de streams y almacenamiento seguro, permitiendo que los documentos de tesis y servicio social se vinculen directamente a los registros de la base de datos PostgreSQL.

5.5 Fase 4: Resultados e Implementación Final

En esta sección se presentan los resultados obtenidos tras el ciclo de desarrollo incremental. El portal académico LACECI se encuentra funcional y desplegado, cumpliendo con los estándares visuales y técnicos requeridos.

5.5.1 Interfaz de Acceso y Seguridad

El acceso al sistema se realiza a través de un módulo de autenticación centralizado. Como se observa en la Figura 5.3, la interfaz permite el ingreso mediante credenciales validadas, ofreciendo además la opción de registro para nuevos usuarios.

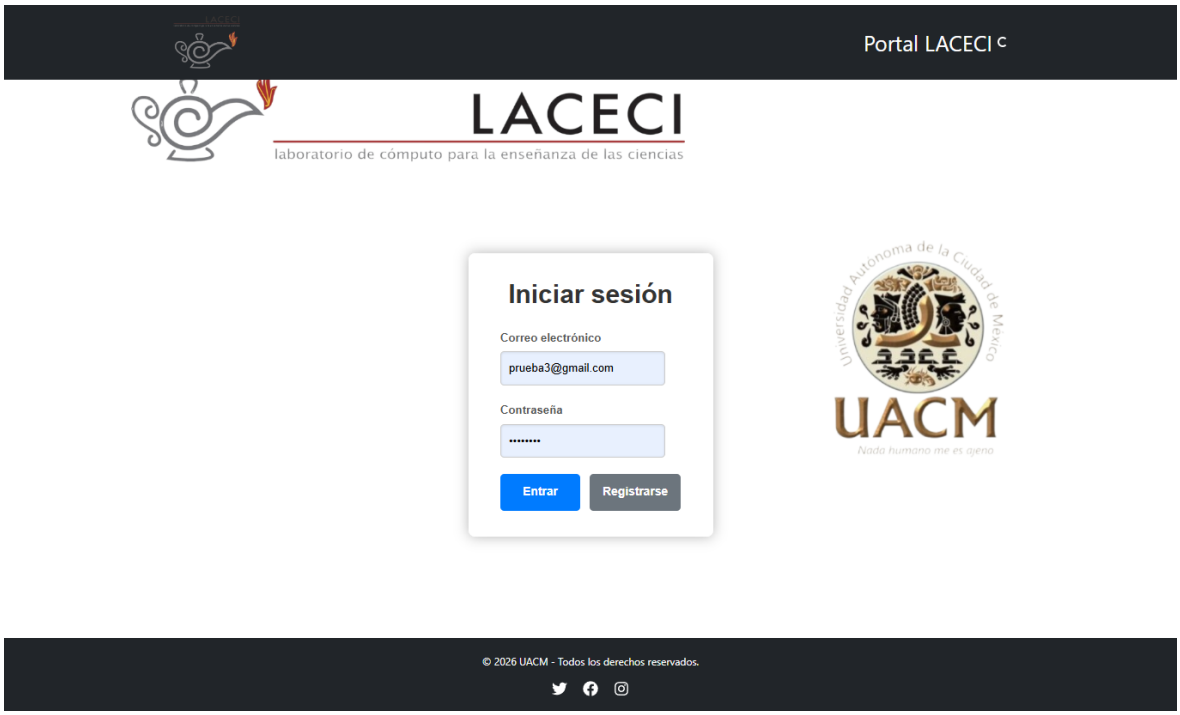


Figura 5.3: Interfaz de inicio de sesión del portal LACECI.

5.5.2 Panel Principal (Home)

El Home (Figura 5.4) funciona como el tablero de anuncios principal. Presenta una sección de "Tesis Recientes", un carrusel informativo de los eventos próximos en LACECI y el listado de los últimos trabajos cargados con opción de descarga directa, facilitando el acceso rápido a la producción académica del laboratorio.

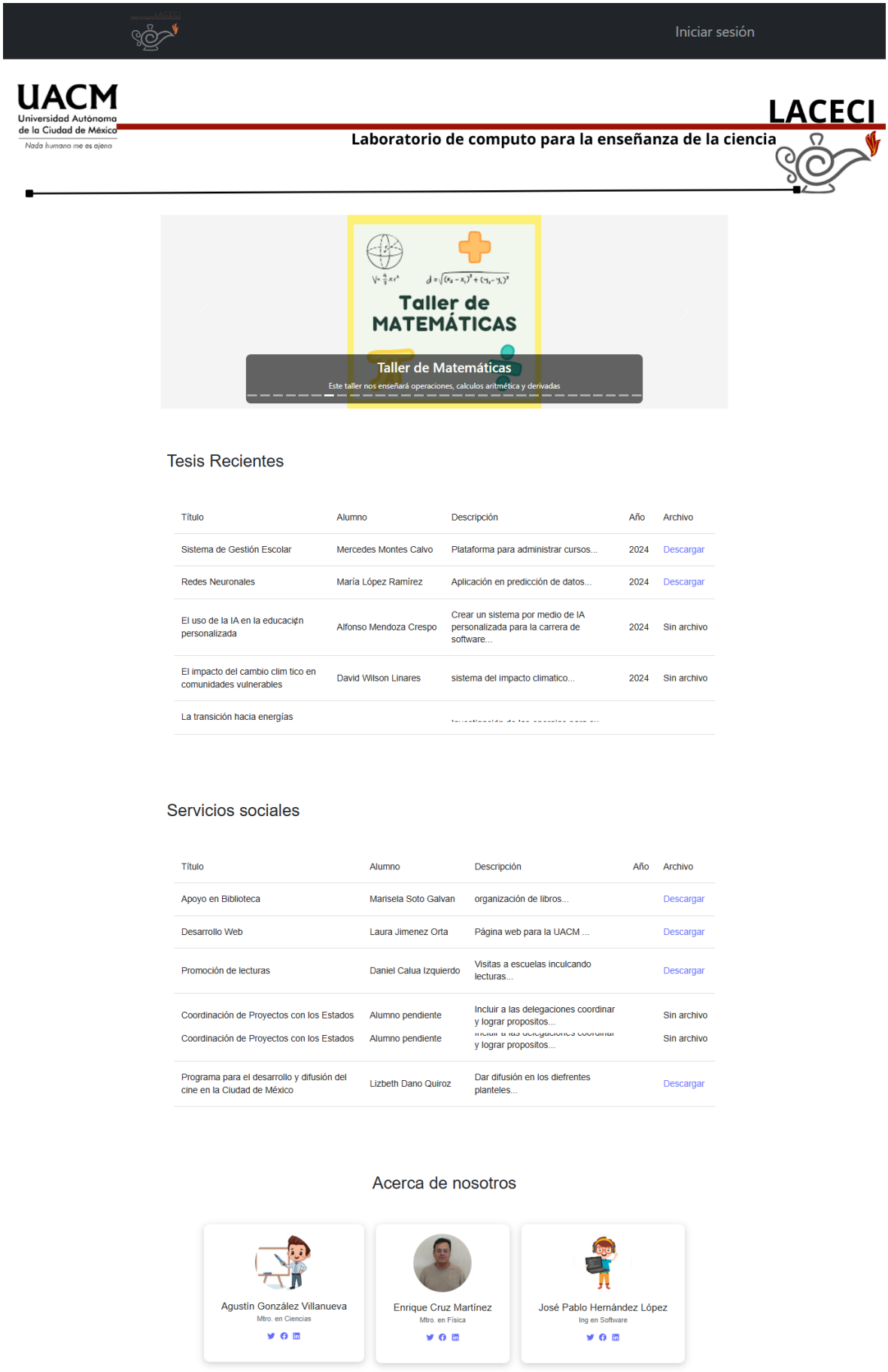


Figura 5.4: Vista principal con gestión de tesis y servicios sociales recientes.

5.5.3 Directorio de Profesores y Responsables

Para fomentar la comunicación académica, se implementó un directorio (Figura 5.5) con búsqueda filtrada por nombre o colegio. Complementariamente, la vista de responsables (Figura 5.6) permite visualizar la asignación de directores, co-directores y lectores para cada trabajo de tesis o servicio social.

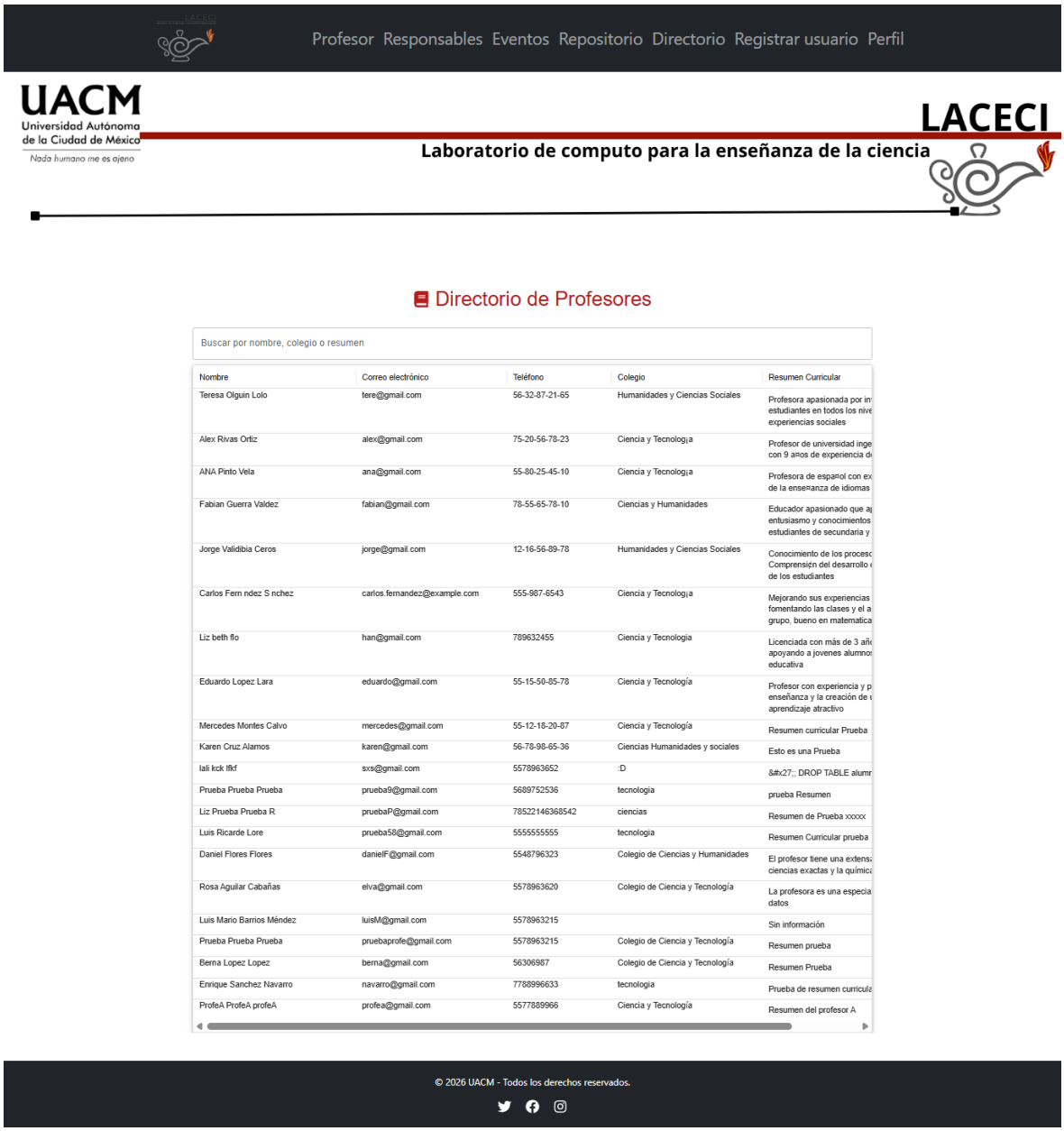


Figura 5.5: Directorio dinámico con búsqueda de perfiles académicos.

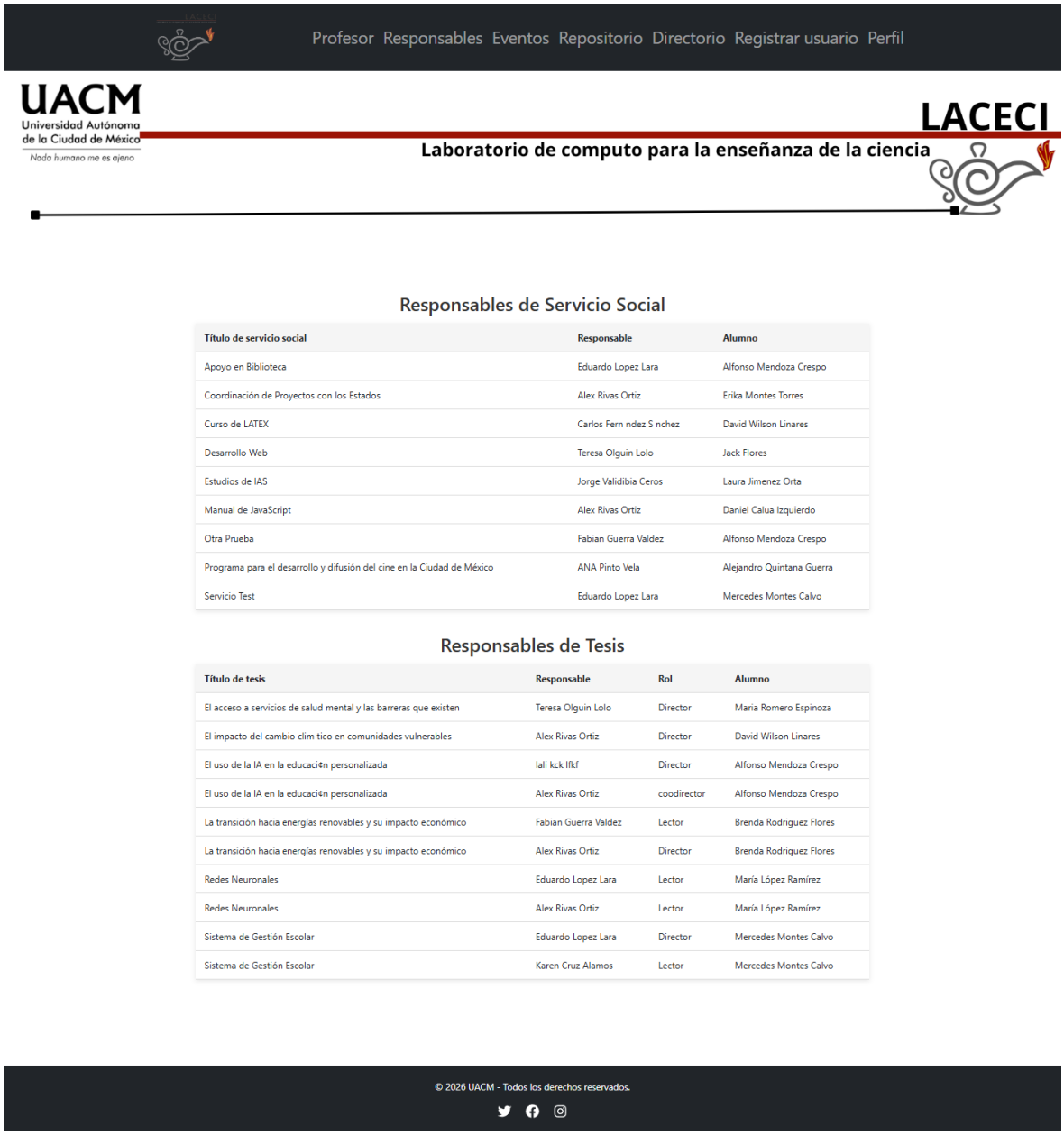
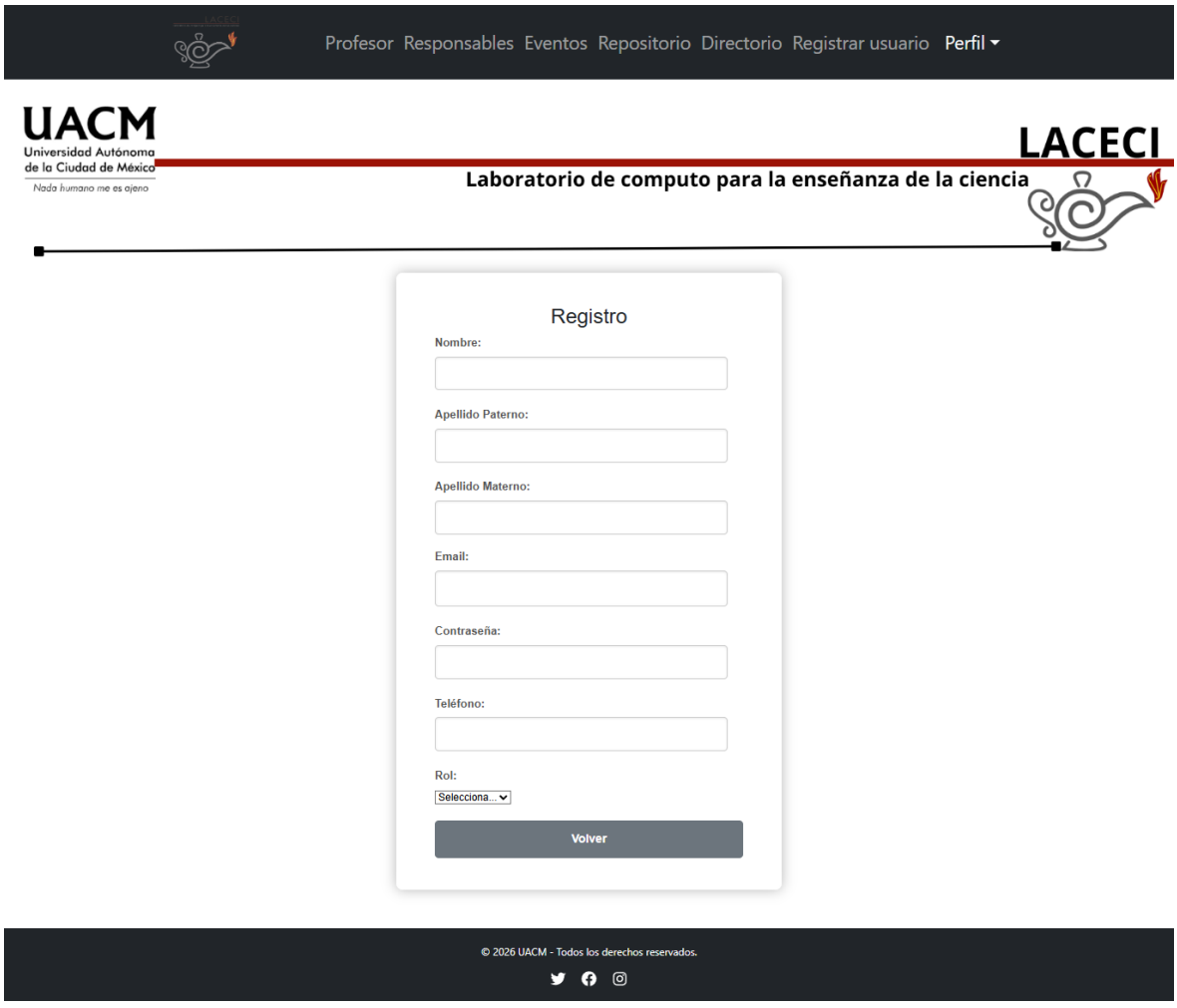


Figura 5.6: Módulo de seguimiento: Busqueda de responsables por proyecto académico.

5.5.4 Módulo de Registro Adaptativo

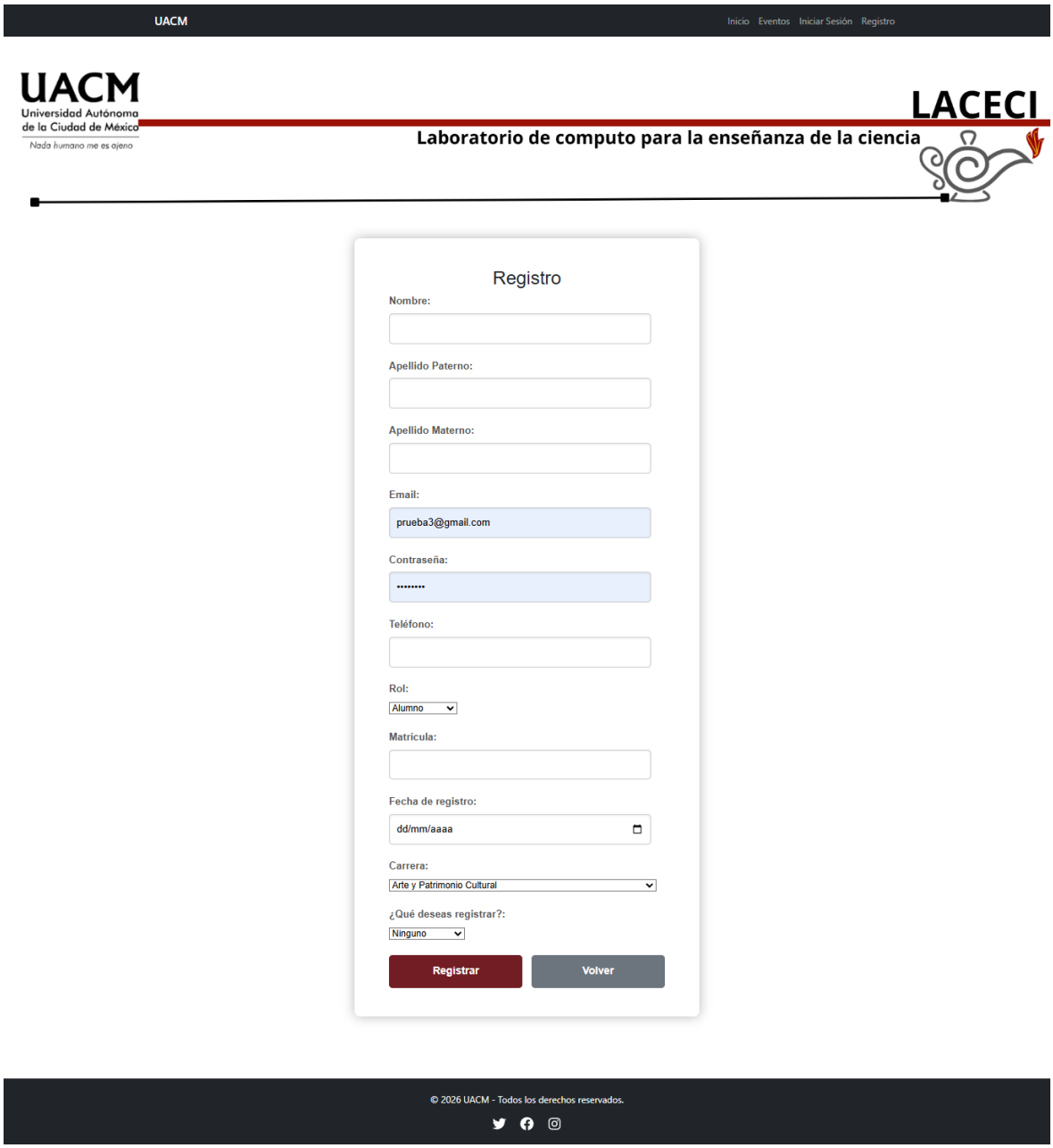
El portal utiliza lógica de React para adaptar el formulario de registro. En la Figura 5.7, se observa la evolución del formulario general de datos que todos los tipos de usuario deben llenar.



(a) Perfil General.

Figura 5.7: Evolución del registro dinámico (Parte 1).

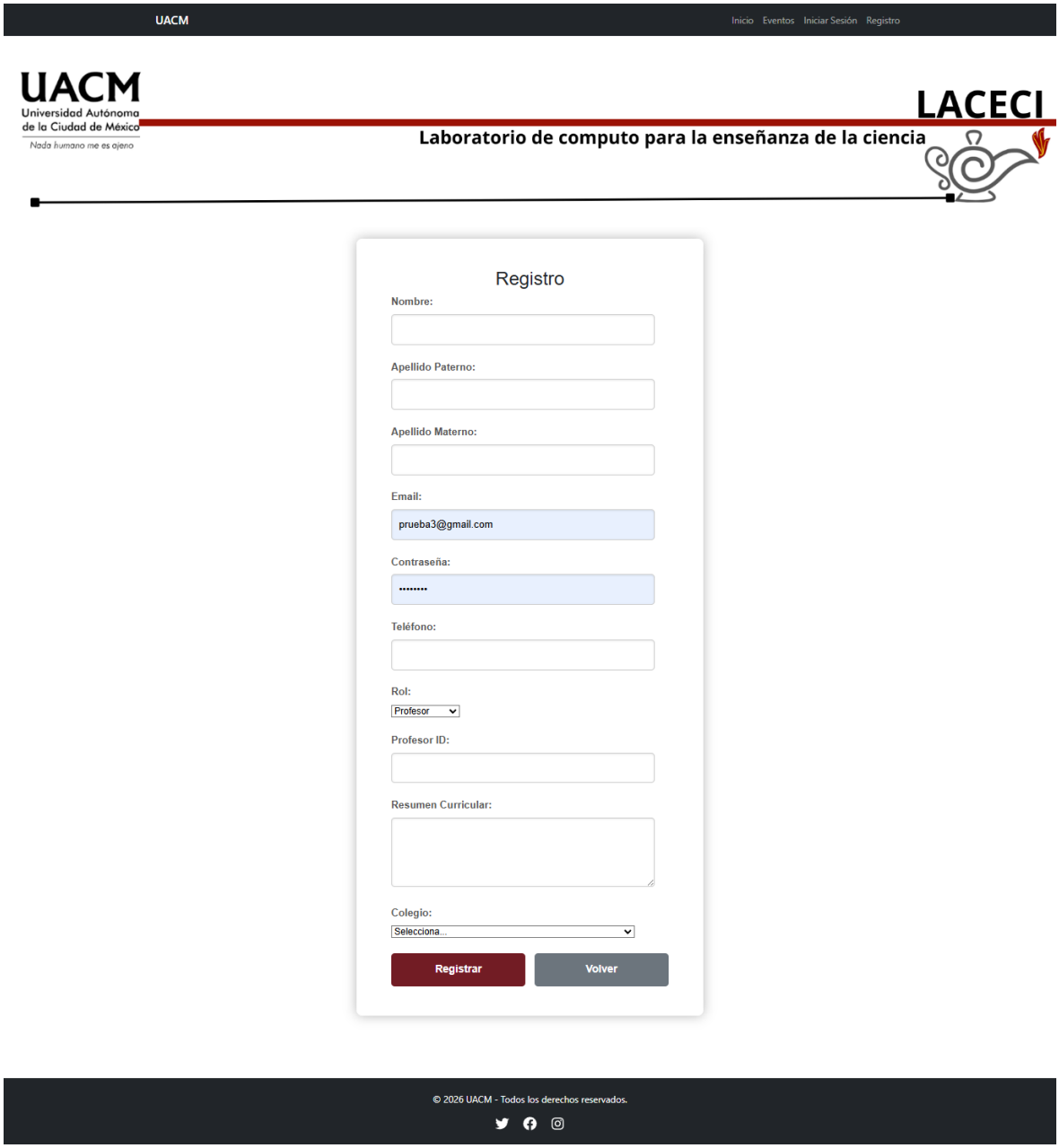
En la figura 5.8, se observa el formulario con los datos para un registro de alumno.



(b) Alumno.

Figura 5.8: Evolución del registro dinámico (Parte 2).

En la figura 5.9, se observa el formulario con los datos para el registro de un profesor.



(c) Registro Profesor.

Figura 5.9: Evolución del registro dinámico (Parte 3).

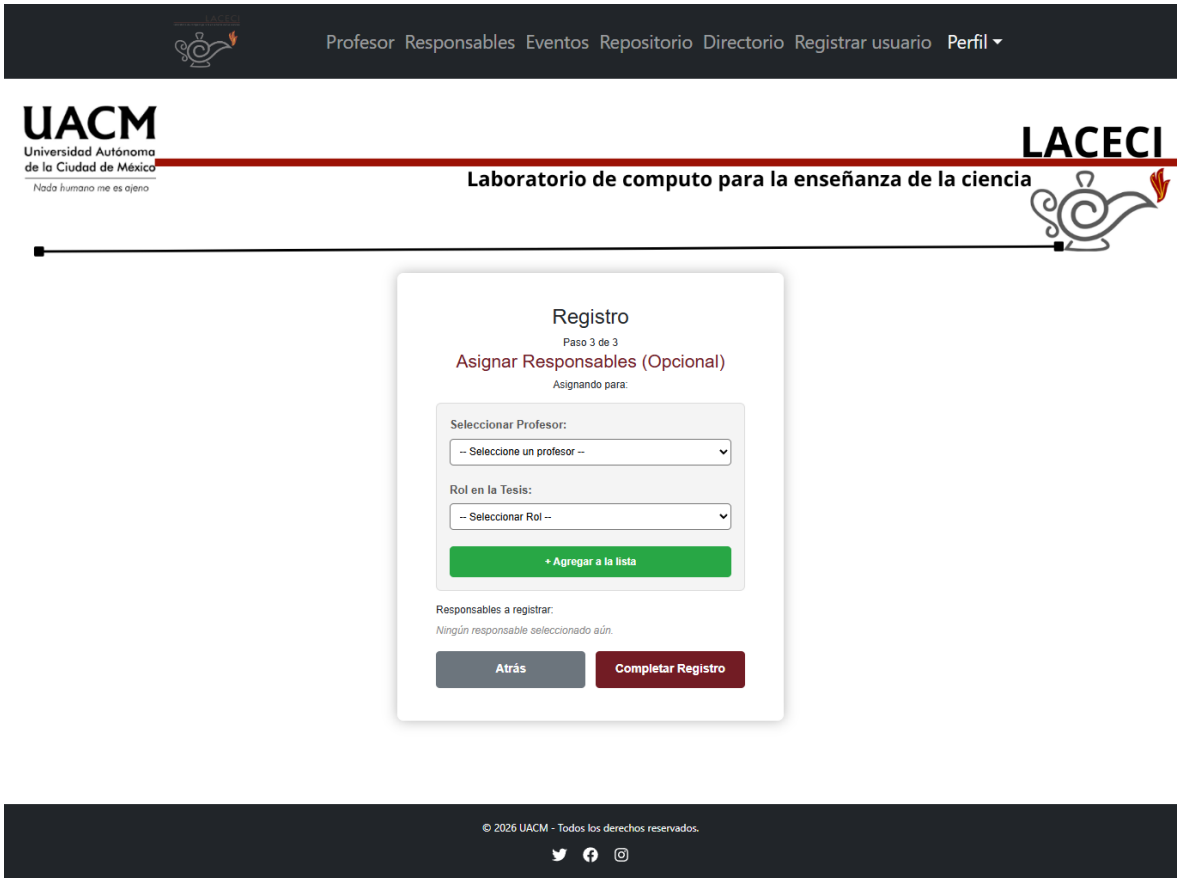


Figura 5.10: Registro de responsables de una tesis.

Figura 5.11: Repositorio de Tesis

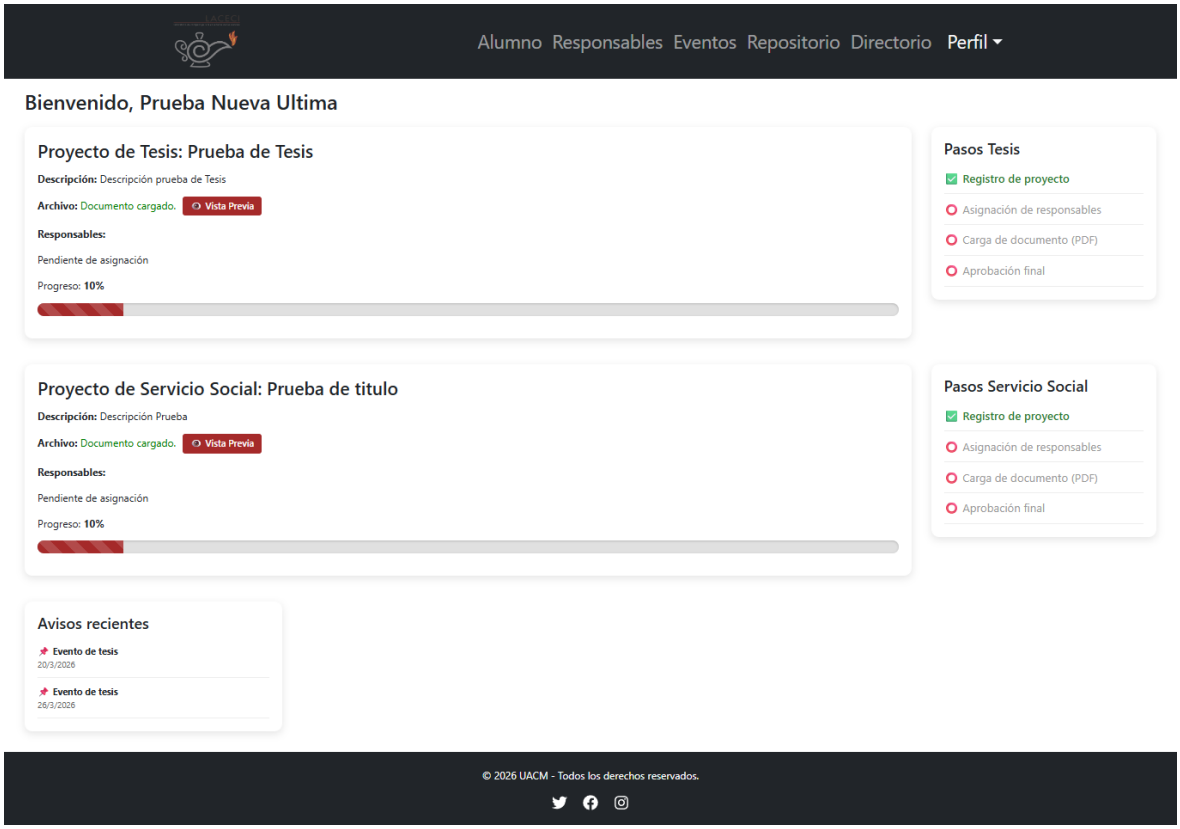


Figura 5.12: Dashboard de Alumno

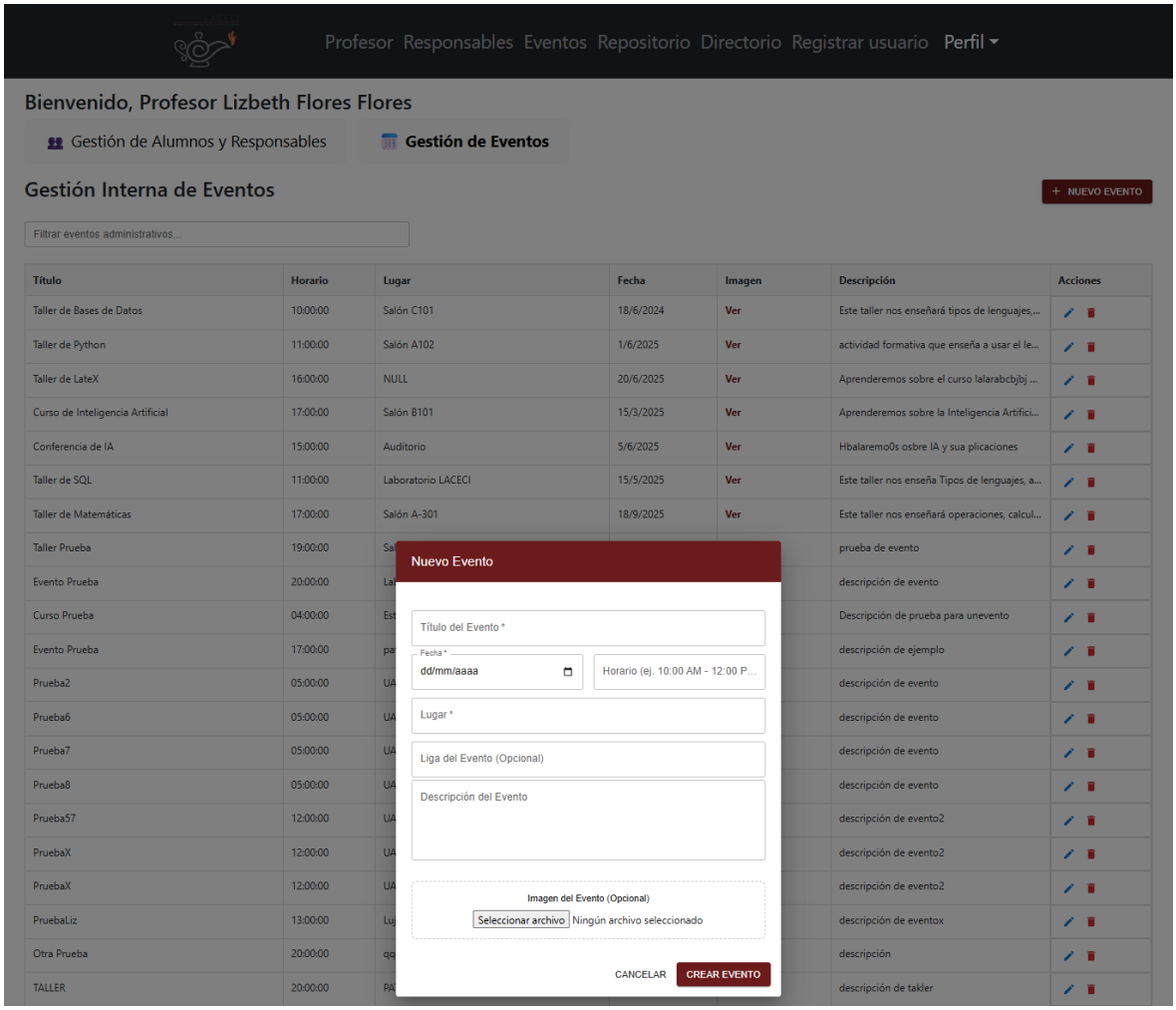


Figura 5.13: Dashboard del profesor para crear eventos

Como parte de la validación final del sistema, se definieron diversos indicadores técnicos y de usabilidad para medir la eficiencia del Portal Académico LACECI **??** detalla estas métricas de evaluación, las cuales confirman que el portal cumple satisfactoriamente con los requerimientos establecidos, destacando tiempos de respuesta óptimos, integridad total en el manejo de datos y una alta aceptación por parte de los usuarios piloto."

Indicador (Métrica)	**Descripción / Objetivo**	**Resultado Obtenido**
Tiempo de carga inicial	Velocidad de respuesta al acceder al Home.	1.5 - 2.1 segundos
Integridad de datos	Validación de llaves foráneas en el registro de tesis.	100 % exitoso
Seguridad de acceso	Bloqueo de rutas no autorizadas por rol (RBAC).	Validado
Usabilidad (Feedback)	Facilidad de uso percibida por usuarios piloto.	Alta (8.5/10)
Compatibilidad	Funcionamiento en Chrome, Firefox y Safari Mobile.	100 % funcional

Cuadro 5.3: Métricas de evaluación del Portal Académico LACECI.

Para validar el sistema, se ejecutó una batería de pruebas de carga. Los resultados se resumen en la siguiente tabla:

Prueba	**Escenario**	**Resultado**
Carga de Página	Conexión Local	< 1.2 segundos
Carga de Archivo (10MB)	10 usuarios concurrentes	Exitoso
Autenticación	Intento fallido / exitoso	Validado 100 %

Cuadro 5.4: Métricas de rendimiento y validación funcional.

Capítulo 6

Conclusiones

En este trabajo se desarrolló un portal web académico empleando un stack tecnológico de software libre, lo que permitió un control total sobre la arquitectura, flexibilidad en el desarrollo y la eliminación de costos por licenciamiento. El sistema se diseñó bajo una estructura desacoplada, garantizando que cada componente pueda actualizarse u optimizarse de forma independiente.

A continuación, se resumen los pilares tecnológicos implementados:

Frontend (Capa de Usuario)

- **React.js:** Como biblioteca principal para la creación de interfaces dinámicas basadas en componentes reutilizables.
- **Bootstrap:** Para garantizar un diseño responsivo y profesional que se adapte a cualquier dispositivo móvil o de escritorio.
- **JavaScript (ES6+):** Como lenguaje base para la manipulación del DOM y la interactividad en tiempo real.

Backend e Infraestructura (Capa de Lógica y Datos)

- **Node.js y Express:** Como entorno de ejecución y framework para la construcción de una API RESTful eficiente y escalable.
- **PostgreSQL:** Para la persistencia de datos bajo un modelo relacional robusto que garantiza la integridad de la información académica.
- **Seguridad:** Implementación de **JSON Web Tokens Express.js** para la autenticación segura y el control de acceso basado en roles (RBAC).
- **Docker:** Empleado para la contenedorización de la aplicación, facilitando un despliegue consistente en cualquier entorno.

Impacto de la Arquitectura de Microservicios

La decisión de implementar una arquitectura de microservicios fue fundamental para cumplir con el objetivo de modularización del sistema. Esta estructura permite que múltiples desarrolladores integren nuevas funcionalidades o modifiquen módulos existentes sin comprometer la estabilidad global del portal. Esta independencia tecnológica asegura que el LACECI cuente con una plataforma preparada para el crecimiento futuro y la evolución académica.

Finalmente, la aplicación del **Modelo Incremental** permitió una transición exitosa desde los procesos manuales hacia una automatización digital efectiva, validando cada fase con los usuarios finales y garantizando que el producto final responda fielmente a las necesidades del laboratorio.

Apéndices

Apéndice A

Documentación

A.1 Técnicas usadas

Técnicas que se usaron para levantar los Requisitos

Técnica Estudio Comparativo

Debilidades

El laboratorio tiene una mala organización con respecto a :

Las actividades de cada usuario.

Llevar un orden en los procesos.

La seguridad de la información.

El registro de los formatos .

No cuentan con formatos bien planteados.

Fortalezas

Todos los usuarios tienen el conocimiento de los procesos.

El laboratorio en su mayoría funciona correctamente,se busca facilitar las tareas.

Se crean archivos muy buenos y que se les da uso

Se cuenta con mucho material

Se crean archivos de mucha utilidad

Ya se cuenta con los datos definidos que usaremos

Técnica MoSCoW

M-Must Los requisitos indispensables para la solución del sistema en general es contar con una base de datos.

S-Should Requisitos que deberían incluir la solución; requisitos importantes pero no obligatoriamente necesarios.

Se deberá poder hacer comparaciones con los procesos.

C-Could Requisitos que podría incluir la solución la guinda del pastel

Se contará con categorías .

W-won't Requisitos que no van a hacer (por el momento)pero que en un futuro podrían incorporarse .

Usar un espejo para el llenado de registro de tablas .

Realizar la base de datos del sistema para sus necesidades.

A.2 Entrevistas

Universidad Autónoma de la Ciudad de México UACM

Nada humano me es ajeno

Proyecto: *Portal Web para Laboratorio de Cómputo para la Enseñanza de la Ciencia/UACM-LACECI (LACECI)*

Doc: 1.0 **exp:** G. Lizbeth Mexicano Flores

Fecha: 18 /03/2024

Entrevistado: Enrique Cruz Martínez

Cargo : responsable del laboratorio LACECI

Técnicas que se usarán para levantar los Requisitos

Debilidades

Carencia de un sitio que almacene, gestiones y actualice toda la información de los proyectos de investigación y desarrollo

Fortalezas

Generación de diversos tipos de documentos como libros y notas en PDF, LaTeX, videos, cursos en aulas virtuales y demás contenidos por un grupo muy activo de profesores y estudiantes.

Glosario de datos

Defina si hay términos clave o datos relevantes con los que no estamos familiarizadas y debamos saber :

En realidad no, pero si palabras clave

Lenguajes de programación, manuales técnicos, visualización, datos, clusters, tarjetas SBC, tesis, investigación, prototipos, virtualizar, servidores, portal, procesos paralelos.

Entrevista:

¿ Qué actividades realiza en el laboratorio?

Coordinar algunos proyectos de investigación, dirigir tesis y servicios sociales, impartir cursos y talleres de programación. Responsable de inventario de los equipos y materiales del laboratorio.

(todo está permitido,reglas de negocio o en este caso de laboratorio)

¿Las copias de respaldo deben almacenarse en un lugar diferente?

Si, de preferencia en un disco externo para tener un respaldo físico en otro dispositivo.

¿Tiene algo que agregar ?
Cree que pudieran existir algunas cuestiones relevantes que no fuesen abordadas en la entrevista. solo falta presentar la parte de diseño y organización del Portal académico

Solo ver la propuesta de cómo se organizará el portal académico para su fácil consulta
¿Qué debemos considerar?

La siguiente revisión, funcionalidad del Portal a nivel usuario, independientemente de la vista o diseño final.

!!Gracias¡¡

Universidad Autónoma de la Ciudad de México UACM
Nada humano me es ajeno

Proyecto: *Portal Web para Laboratorio de Cómputo para la Enseñanza de la Ciencia/UACM-LACECI (LACECI)*

Doc: 1.0 **exp:** G. Lizbeth Mexicano Flores

Fecha: 18 /03/2024

Entrevistado: **Agustín González Villanueva**

Cargo : **corresponsable**

Técnicas que se usarán para levantar los Requisitos

Debilidades

Falta de servidores y plataformas

Fortalezas

Red de cómputo con linux

Glosario de datos

Defina si hay términos clave o datos relevantes con los que no estamos familiarizadas y debamos saber :

Entrevista:

¿ Qué actividades realiza en el laboratorio?

Educación e investigación

¿Cuál es la actividad principal dentro del laboratorio ?

Creación de proyectos y acercamiento a la educación científica y computacional

¿ Cuáles son las principales problemáticas dentro del laboratorio ?
Ausencia de conectividad y servicios de contenidos

¿Cómo se resuelven actualmente?
Por el apoyo de los estudiantes

¿Quién o quiénes utilizarán el portal web?

Quienes participen en los proyectos

¿Cuántos y qué roles hay dentro del laboratorio ?

Dos profesores y estudiantes de servicio social y tesistas

¿Quién tendrá acceso al sistema?

Un grupo específico con habilidades para ello (interno) y usuarios externos en general

¿Los distintos colaboradores podrán tener sus propios usuarios para la manipulación de estos datos?

Dependerá de cada proyecto

¿Cuántos tipos de usuarios se espera haya dentro del sistema?

Dos administradores generales y un pequeño grupo de ayudantes de proyecto

¿Cuál es el objetivo que se desea alcanzar ?

Una o más plataformas de servicos educativos y de cómputo científico

¿Cómo beneficia este proyecto a su laboratorio?

Lo acerca con la comunidad universitaria y le ofrece herramientas especiales

¿ Qué entregables producen ?

Por el momento, archivos en postscript (pdf) de los reportes de servicio social y tesis realizadas y en curso, despues los avances de proyectos

¿Cómo se llevan a cabo los procesos para los entregables?

Por periodo en cada convocatoria

¿Cuentan con algún o algunos formatos ?
R= No

si (se anexa) **no**

¿De donde se obtiene esa información?

¿Con qué frecuencia lo hacen ?
R= Se hace a como lo soliciten los estudiantes

¿Los datos obtenidos son públicos o privados?
R= Todo lo que se hace en el laboratorio es público.

¿Existe algún responsable de obtener la información ?
R= No.

¿Para qué se utilizan estos datos ?

¿Dónde se almacena esta información actualmente ?

¿Qué restricciones hay se deben tener en cuenta?

¿ Qué se podrá hacer?
(todo está permitido,reglas de negocio o en este caso de laboratorio)

¿Las copias de respaldo deben almacenarse en un lugar diferente?

¿Tiene algo que agregar ?
Cree que pudieran existir algunas cuestiones relevantes que no fuesen abordadas en la entrevista.

¿Qué debemos considerar?

!!Gracias¡¡

¿Cuál es la actividad principal dentro del laboratorio ?
Administracion de laboratorio

¿ Cuáles son las principales problemáticas dentro del laboratorio ?

No existe la infraestructura necesaria por parte de la universidad para que el laboratorio funcione de forma optima

¿Cómo se resuelven actualmente?
Con apoyo de profesores encargados y busqueda de implementación de opensource para bajar los costos

¿Quién o quiénes utilizarán el portal web?
Publico en general

¿Cuántos y qué roles hay dentro del laboratorio ?
Profesores
Tesistas
Alumnos

¿Quién tendrá acceso al sistema?
Profesores y tesistas

¿Los distintos colaboradores podrán tener sus propios usuarios para la manipulación de estos datos?
De preferencia sí

¿Cuántos tipos de usuarios se espera haya dentro del sistema?
3

¿Cuál es el objetivo que se desea alcanzar ?
Todo el cual abarque el proyecto que se este encomendando

¿Cómo beneficia este proyecto a su laboratorio?
Desconozco ya que no se cual es el fin general ni particular del proyecto

A.3 Requerimientos

Requerimientos Portal LACECI	Terminado V1	Comentarios
Home		
El home debe contener nombre del laboratorio	☑	
El home debe tener el logo del laboratorio	☑	
El home debe tener los colores de la universidad	☑	
Se requiere tipografía y color los proporcionados como diseño de LACECI y La letra debe ser de un tamaño que pueda leerse a 30 cm de distancia.	☑	
Vista principal con información importante de eventos próximos (avisos,fechas y publicidad de actividades del laboratorio)	☑	
Los eventos de la vista principal deben contar con esta información: nombre, horario, lugar, fecha, imagen	☑	
Debe tener una galería de fotos, las imágenes pueden ser fijas o un degradado de imágenes, la galería debe componerse de fotos de los eventos que se han realizado en el laboratorio, . estas fotos las deberán poder cargar los administradores.	☑	
Debe tener un contador de visitas.	☐	
Debe tener una sección llamada acerca de nosotros, aquí debe haber un directorio de profesores, un resumen curricular de los profesores y una liga a la ficha dentro del directorio institucional	☑	
Debe tener un menú principal este debe contener en sus opciones: registro de usuarios, tesis, servicio social y portafolio de archivos para la comunidad	☑	

Requerimientos Portal LACECI	Terminado V1	Comentarios
Registro de usuarios		
El registro de usuarios alumnos debe tener :nombre completo, correo electrónico,matrícula y deben poder crear una contraseña.	☑	matrícula como llave primaria por si el usuario quiere poner un correo diferente al institucional
Los usuarios profesores deben registrar : número de trabajador, nombre completo, correo electrónico, deben poder crear una contraseña	☑	#trabajador como llave primaria por si el usuario quiere poner un correo diferente al institucional
Los usuarios deben registrar el título de su proyecto en cualquier caso de registro: tesis o servicio social.	☑	Esto va en tesis y servicio
Los usuarios deben registrar la fecha de inicio en cualquier caso de registro: tesis o servicio social.	☑	Esto va en tesis y servicio
El registro debe tener un switch que cambie el formulario de registro según sea el caso:Alumno o profesor.	☑	
Debe contar con un botón de registro	☑	
Se debe confirmar el registro correcto de una persona, con un mensaje en pantalla	☑	
Se debe contar con protección en los datos personales,se usará el protocolo HTTPS	☑	Checar si HTTP O HTTPS con costo con la Uni
Esta área debe tener un botón de regreso al Home	☑	configurar en el logo LACECI
Esta área debe tener el menú principal en la parte superior	☑	En el header

Requerimientos Portal LACECI	Terminado V1	Comentarios
Login		
Los usuarios deben poder loguearse al sistema con usuario y contraseña.	☑	
Los usuarios deben poder recuperar su contraseña	☐	
Se debe contar con seguridad en las contraseñas. se deberá incluir una combinación de mayúsculas y minúsculas, números y símbolos sin relación con tu información personal.	☑	
Se debe tener un botón de iniciar sesión	☑	
Esta área debe tener un botón de regreso al Home	☑	configurar en el logo de la Ceci
Esta área debe contener el menú principal en la parte superior	☑	En el header
Repositorio de archivos y material para la comunidad		
Debe tener una sección de libros,revistas,manuales e investigaciones (tesis,paper,y documentos realizados en LACECI) son realizados en el laboratorio por tesistas, alumnos que realizan su servicio social o por profesores y colaboradores de LACECI. Deberán poder ser cargados por los administradores.	☑	
Debe tener una sección de cursos virtuales,contará con ligas de cursos que están siendo impartidos y ligas de videos de cursos que ya fueron impartidos, estos videos se deben poder subir en los formatos especificados y podrán subirlos los administradores	☑	EN ESPERA

Requerimientos Portal LACECI	Terminado V1	Comentarios
Los archivos aceptados son en los siguientes formatos pdf, txt, tex.	☑	
Debe tener una sección de talleres, la cual está compuesta por presentaciones archivos en ascii texto plano pdf estos archivos los deben poder subir los administradores	☑	EN ESPERA
Debe tener una sección de conferencias y seminarios, en esta sección habra links o videos y pdfs,los deberán poder cargar los administradores	☐	EN ESPERA
En esta subsección los archivos estarán acomodados por abecedario	☐	
Los archivos de esta sección deben poder descargarse	☑	
El o los administradores podrán subir estos archivos de cada sección	☑	
Todas estas secciones deberán contener archivos con titulo, fecha y autor del archivo	☑	
En las secciones donde se sube vídeos debe permitirse:AVI, MP4 o MOV	☐	
Cualquier usuario del portal podrá hacer uso de los archivos	☑	
Se requiere contador de descargas	☐	

Requerimientos Portal LACECI	Terminado V1	Comentarios
Tesis		
El usuario deberá haberse logueado correctamente	☑	
Debe haber un área para subir archivos de tipo pdf y látex , estos archivos deben poder ser cargados por los alumnos logueados que son los que están realizando su tesis	☑	
Los archivos subidos podrán ser borrados, unicamente por el administrador	☑	
Servicio Social		
El usuario deberá haberse logueado correctamente	☑	
En esta área los usuarios deben poder subir archivos de tipo pdf y latex , estos archivos deben poder ser cargados por los alumnos logueados que son los que están realizando su servicio social.	☑	
Los archivos subidos no podrán borrarse ,únicamente por los administradores	☑	

Apéndice B

Diagramas

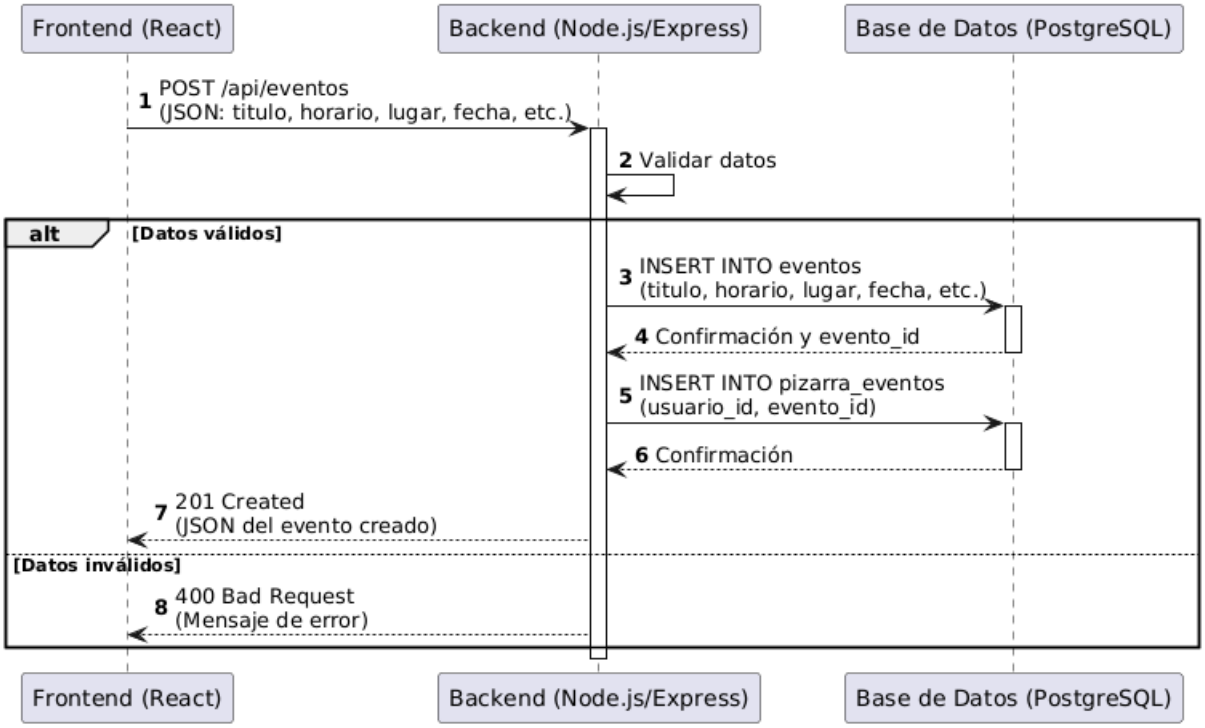


Figura B.1: Diagrama de secuencia eventos

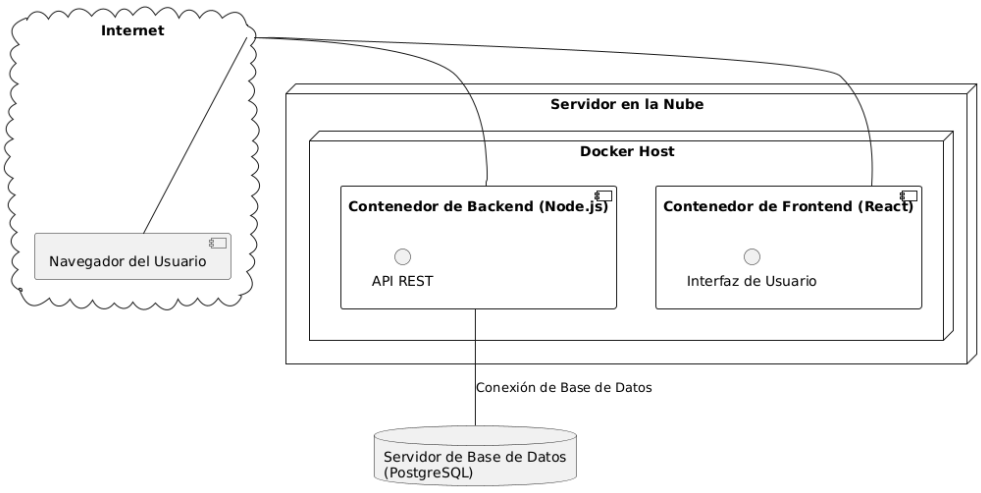


Figura B.2: Diagrama de despliegue del sistema.

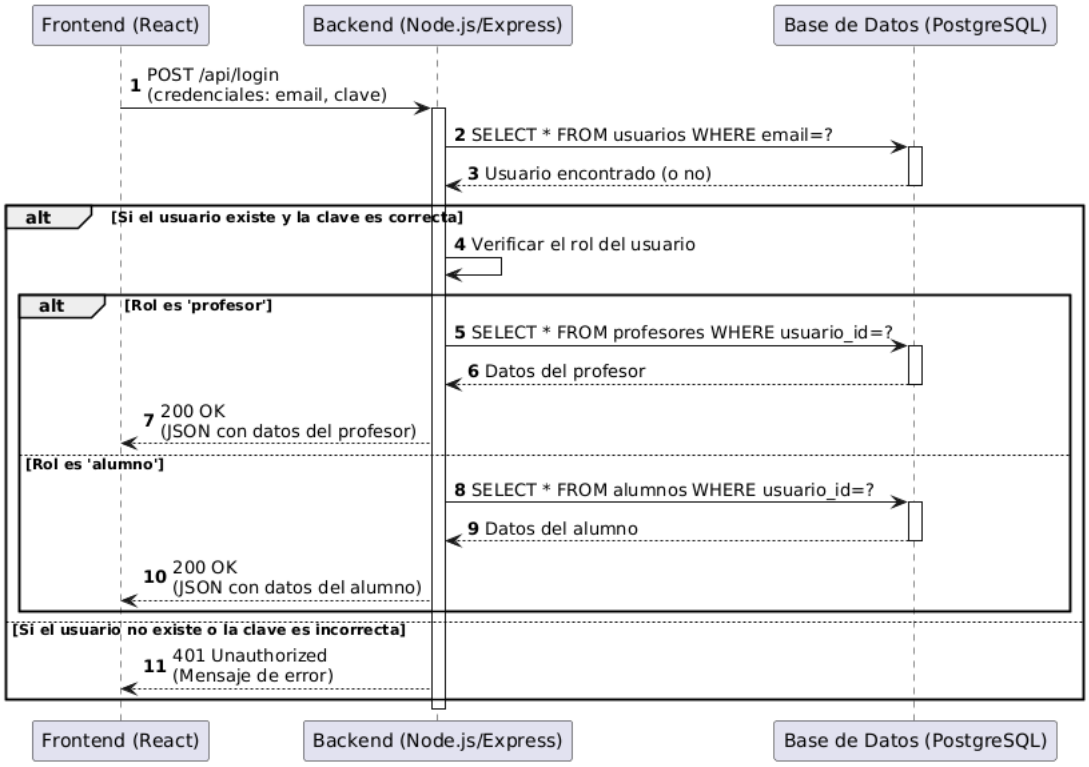


Figura B.3: Diagrama de secuencia login

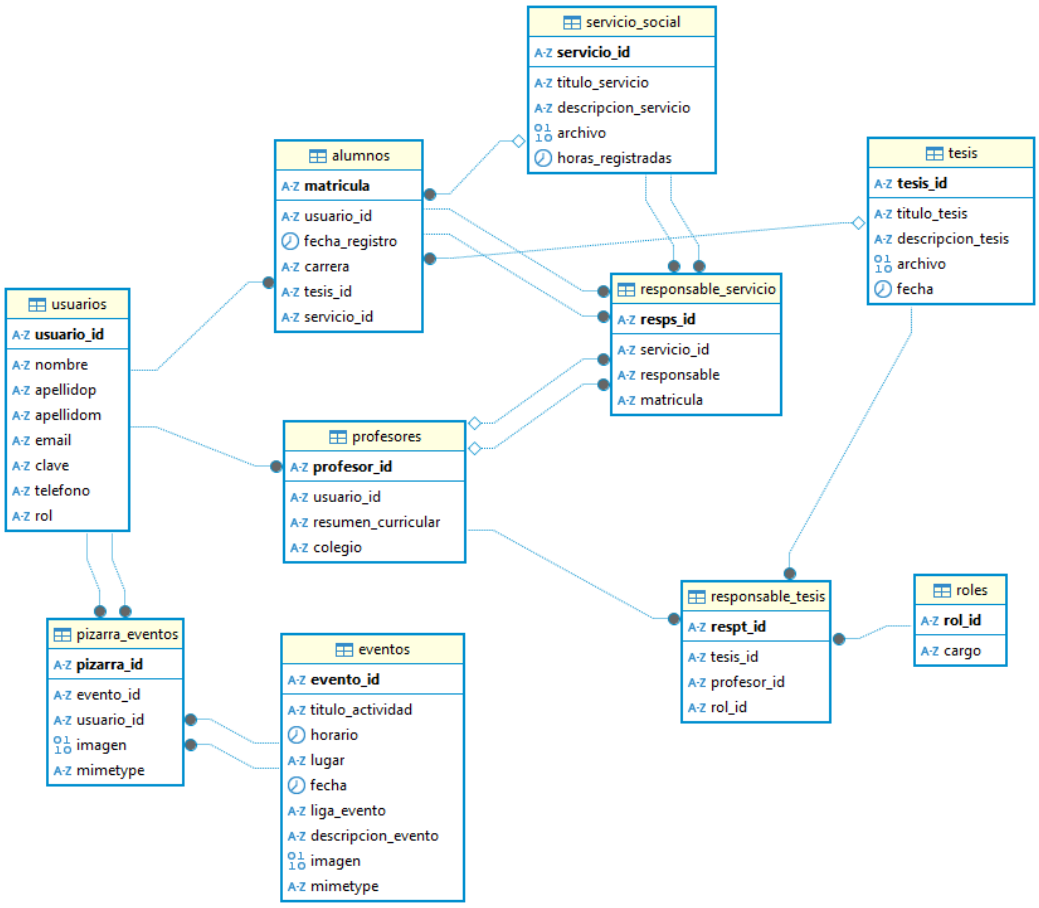


Figura B.4: Diagrama de Entidad-Relación (Modelo Conceptual)

Presenta el modelo entidad-relación del sistema, donde se definen las entidades principales (Usuarios, Alumnos, Profesores, Tesis, Servicio Social y Eventos) y sus asociaciones lógicas. El diagrama detalla la cardinalidad de las relaciones, como la vinculación de alumnos con sus respectivos responsables de tesis y servicios, permitiendo visualizar la integridad referencial necesaria para la base de datos de la institución.

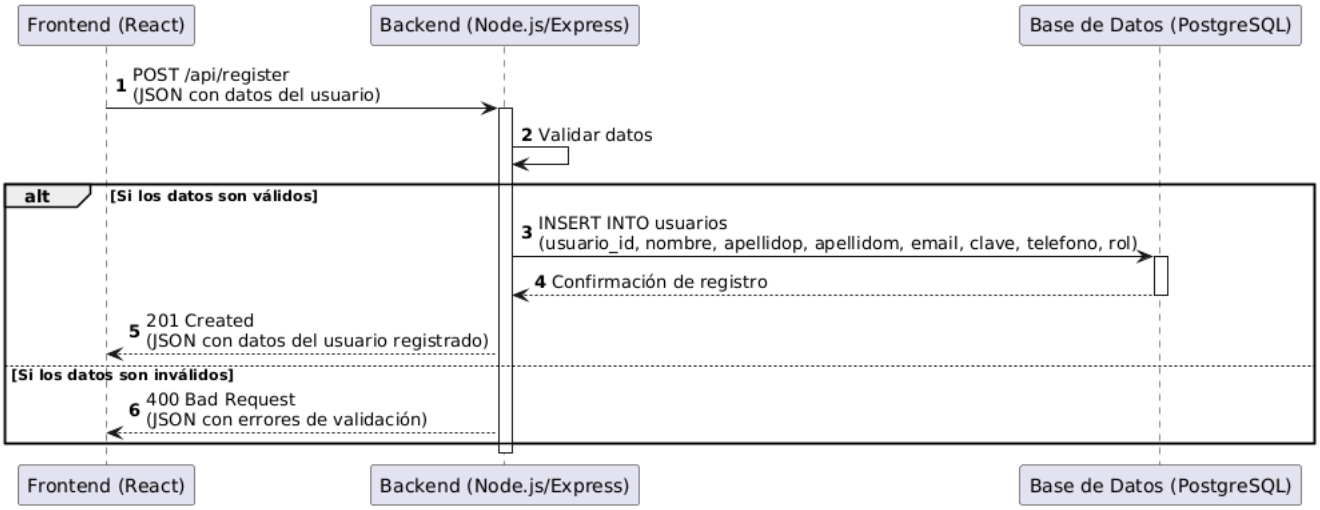


Figura B.5: Diagrama de secuencia registro

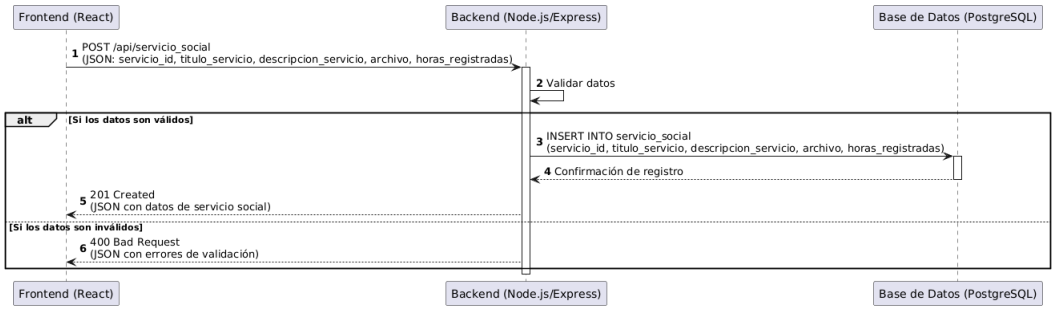


Figura B.6: Diagrama de secuencia Servicio Social

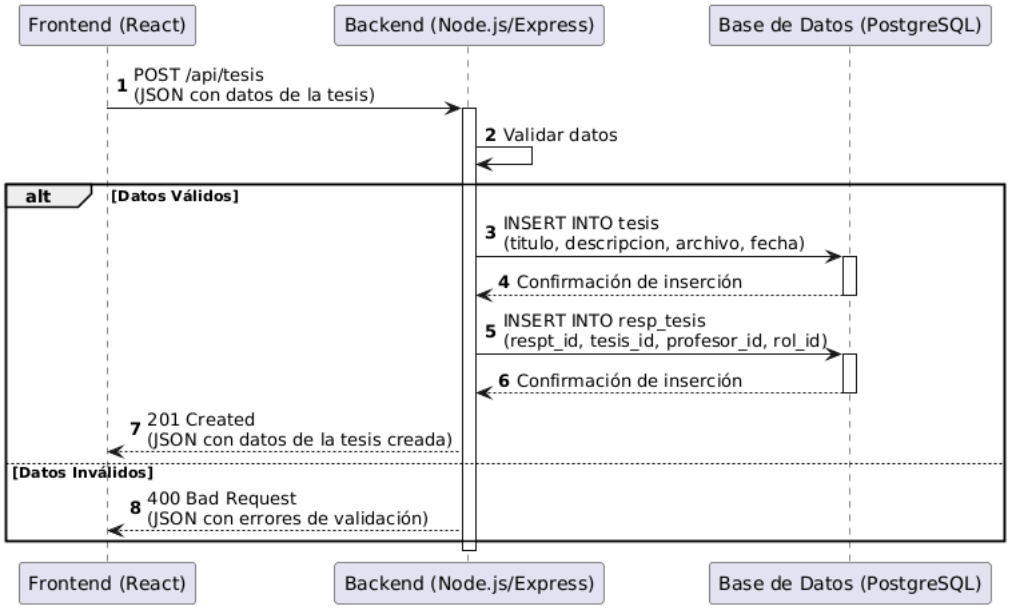


Figura B.7: Diagrama de secuencia tesis

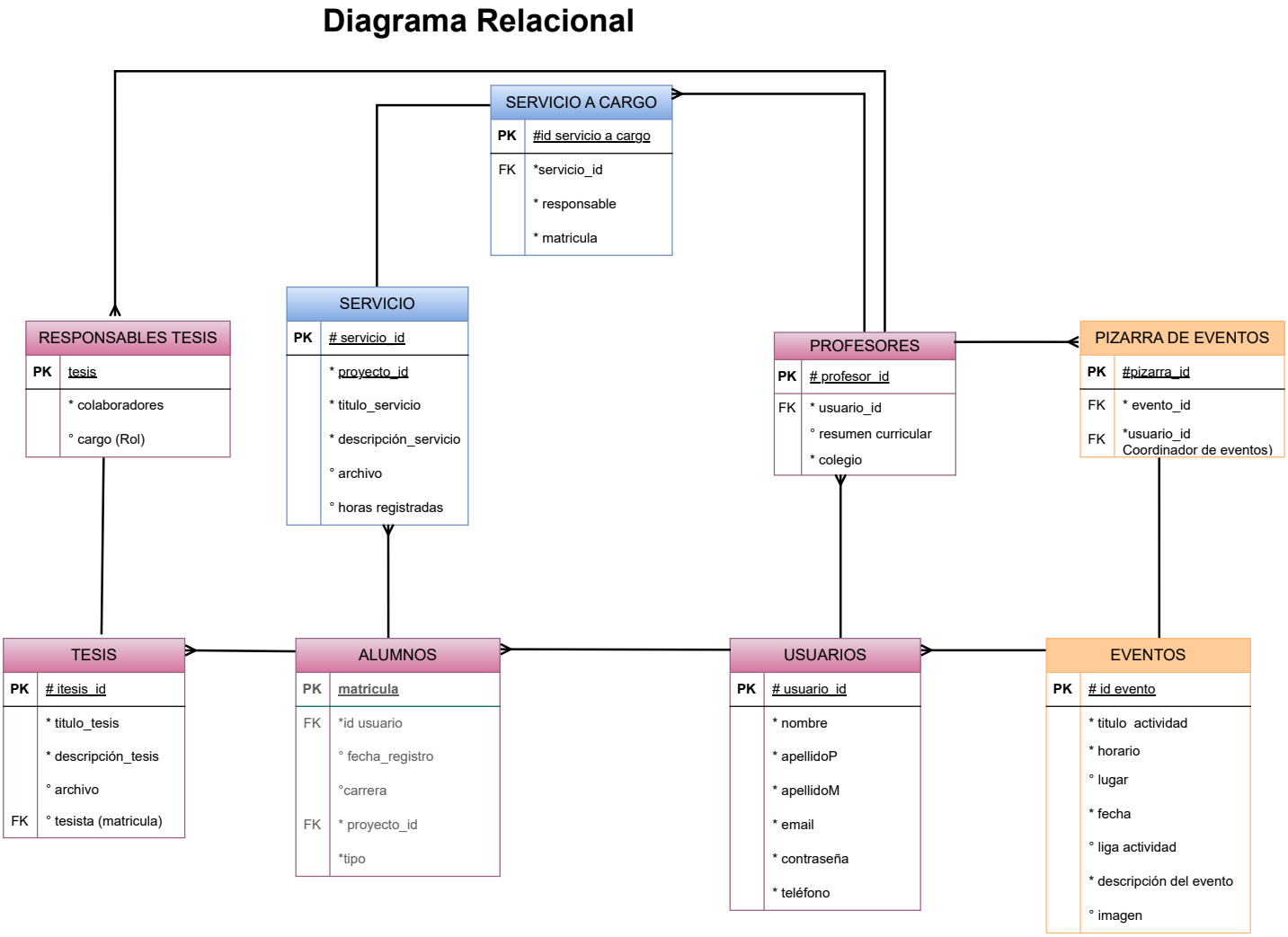


Diagrama Relacional: Este diagrama ilustra la estructura relacional de la base de datos, especificando las tablas con sus respectivas llaves primarias (PK) y llaves foráneas (FK). A diferencia del modelo conceptual, aquí se definen los tipos de datos y la normalización de las tablas, mostrando cómo la tabla ÜSUARIOS"sirve como núcleo para identificar roles específicos (Alumnos o Profesores) y cómo se vinculan de manera única con módulos operativos como "TESIS", "SERVICIOz .EVENTOS".

Apéndice C

Mockups

En este capítulo se presentan los bocetos y prototipos iniciales del portal web desarrollados con la herramienta Balsamiq Mockup. Estos esquemas fueron diseñados con el propósito de definir la estructura y organización de la interfaz de usuario, asegurando una experiencia intuitiva y funcional para los diferentes actores involucrados en el sistema.

Los bocetos permiten visualizar la disposición de los elementos, la navegación entre secciones y la interacción de los usuarios con la plataforma. Además, sirven como una guía para la implementación del portal web, facilitando la comunicación entre la creadora y otros interesados en el proyecto.

A lo largo de este capítulo, se describirán los principales componentes de la interfaz, justificando las decisiones de diseño tomadas en función de la usabilidad y accesibilidad del sistema. Finalmente, se destacarán las mejoras y ajustes que surgieron a partir de la retroalimentación obtenida durante el proceso de diseño.

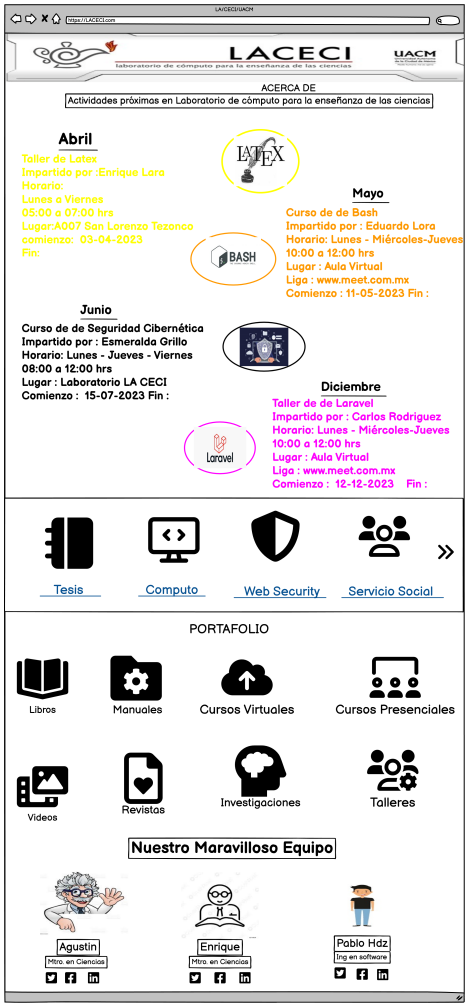


Figura C.1: Propuesta de la Página principal.

Figura C.2: Propuesta de registro para servicio social.

Figura C.3: Propuesta inicio de sesión.

Figura C.4: Propuesta registro para tesistas.

Figura C.5: Propuesta de inicio de sesión tesistas.

Figura C.6: Propuesta menú.

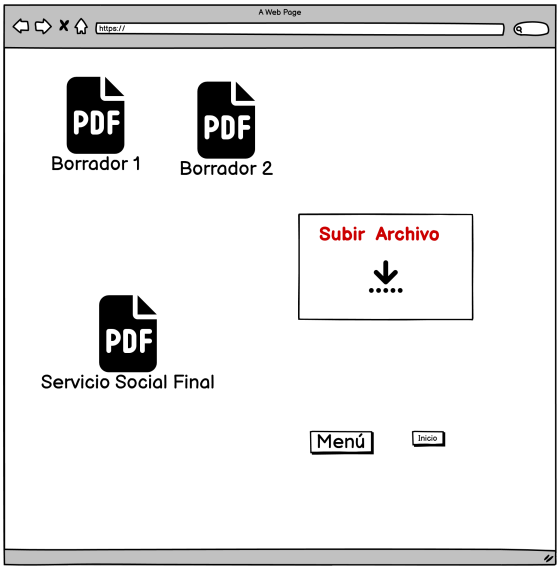


Figura C.7: Propuesta pagína.

Figura C.8: Propuesta de menú servicio social.

Figura C.9: Propuesta para mostrar tesis disponibles.

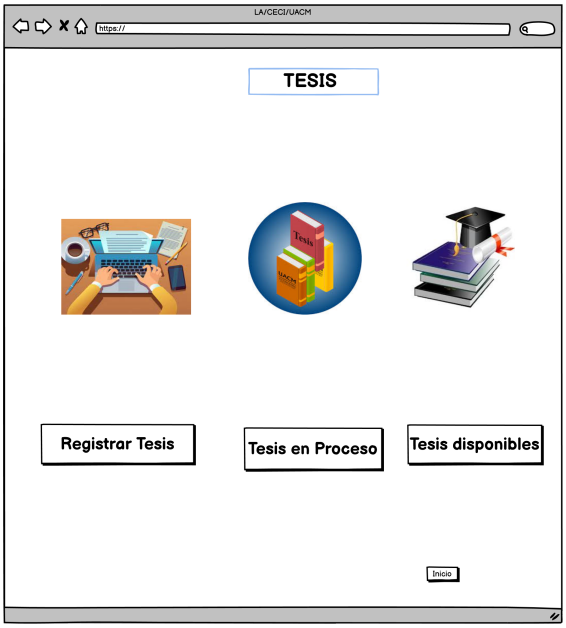


Figura C.10: Propuesta de menú tesis.

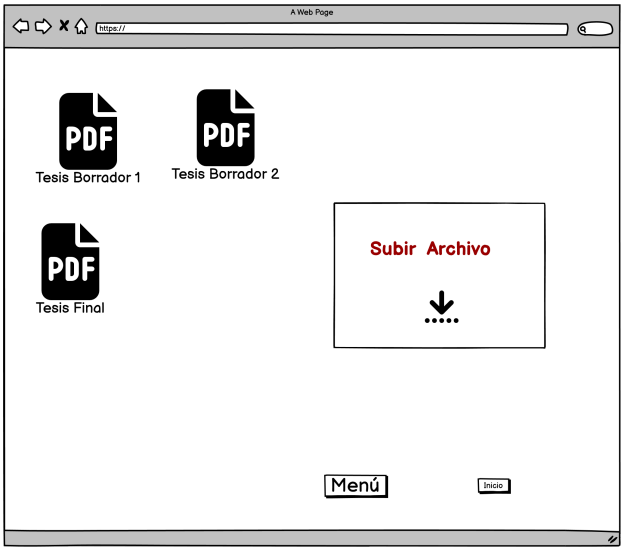


Figura C.11: Propuesta para página tesis.

Bibliografía

[1] Node.js Foundation. (s.f.). *Node.js v20.x Documentation.* https://nodejs.org/docs/latest/api/

[2] *ENIAC: La primera computadora electrónica programable.* https://www.fayerwayer.com/2010/09/158-eniac-la-primera-computadora-electronica FayerWayer (2010).

[3] IBM i 7.5. (n.d.), ¿Qué es Express.js? https://www.ibm.com /docs/es/i/7.5topic=languages-nodejs (2025-01-17)

[4] Kinsta. (2022, December 19) Todo lo que Debes Saber. Kinsta®. https://kinsta.com/es/base-deconocimiento/que-es-express/ Diciembre 19 2022.

[5] IBM i 7.5. (n.d.), ttps://www.ibm.com/docs/es/i/7.5?topic=languages-nodejs 2025-01-17.

[6] TJ Berners-Lee, R. Cailliau, JF Groff, B. Pollermann, World Wide Web, http://info.cern.ch/ (Primavera de 1992)

[7] WorldWideWeb, TJ Berners-Lee, R. Cailliau, JF Groff, B. Pollermann, World Wide Web.http://info.cern.ch, (Primavera de 1992)

[8] TCP/IP ©Copyright IBM Corporation, AIX 7.2. (s. f.). https://www.ibm.com/docs/es/aix/7.2?topic=protocol-tcpip-protocols 1992-11-03.

[9] Noelia Beltrán, Word Wide Web. LaPrimerapaginawebdelmundo. Terceto.https://tercetocomunicacion.es/la-primera-pagina-web-del-mundo/ Editorial, 2020.

[10] Hypertext Robert Balzer. Hypertext and Software Engineering. 3ª ed. Sevilla.1989.

[11] Páginas Web CIS INFORMÁTICA, La Evolución de las Páginas Web https://www.cisinformatica.cat/es/evolucion-de-las-paginas-web marzo 20, 2024.

[12] CSS Avanzado. Javier Eguíluz Pérez, https://unaexperiencia20.com/libros-gratis-html-css-js/#google_vignette,

[13] Alex Zap Chernyak
Pruebas de integración
Profundización en los tipos
https://www.zaptest.com/es/que-son-las-pruebas-de-integracion
profundizacion-en-los-tipos-el-proceso-y-la-aplicacion
2023.

[14] Dr.C. José Felipe Ramírez Pérez Revista Cubana de Informática Médica
ISSN 1684-1859
Estrategia de entrenamiento y acompañamiento a usuarios para el Sistema
http://scielo.sld.cu/scielo
php?script=sci_arttext&pid=S1684-18592020000100076 Ciudad de la Habana ene.-jun. 2020.

[15] Equipo editorial de IONOS
frontend y el backend
https://www.ionos.com/es-us/digitalguide/paginas-web
/creacion-de-paginas-web/que-es-el-frontend
(06/07/2024)

[16] MDN Web Docs. Modelo de Objetos del Documento (DOM)
https://developer.mozilla.org/es/docs/Web/API/Document_Object
_Model
Introduction
(1998-2025).

[17] Jorge Ferre. Curso completo de HTML
https://unaexperiencia20.com/libros-gratis-html-css-js/
#google_vignette

[18] Maria Coppola, HubSpot Selectores CSS
https://blog.hubspot.es/website/selectores-css
16demayode2023.

[19] RockContentBloc, Bootstrap
https://rockcontent.com/es/blog/bootstrap/
Abril-12-20

[20] Antonio Rodríguez Manual de JavaScript
https://unaexperiencia20.com/libros-gratis-html-css-js/
#google_vignette

[21] GITBOOK
Libro desarrollo de software,conceptosrest https://rcasalla.gitbooks.io
/libro-desarrollo-de-software/ content/libro
/temas/t_rest/rest_ conceptosrest.html, 2024

[22] AWS. (s. f.). Amazon Web Services, Inc Contenedores de Docker
https://aws.amazon.com/es/what-is-aws/

[23] NetApp. (2024c, julio 22). Qué son los contenedores? NetApp
https://www.netapp.com/es/devops-solutions/what-are-containers/

[24] MDN Web Docs, Tecnología para desarrolladores web,Referencia de la
API Web, https://developer.mozilla.org/es/docs/Web/API, 1998-2025

[25] Deyimar A. (29 de junio de 2023). *¿Qué es React? Una guía para principiantes.* Hostinger Tutoriales. https://www.hostinger.mx/tutoriales/que-es-react

[26] OpenAI ChatGPT: Modelo de lenguaje IA
https://openai.com/chatgpt

[27] Libro JavaScript author:Negrino, Tom,JavaScript PARA DISEÑO
WEB,2007,PEARSON prentice halll, Madrid España

[28] author = Next U. (s.f.), title = Interacción entre JavaScript,
Node.js, Express.js y Base de Datos, url = https://www.nextu.com/,
note =Imagen digital. Licencia: CC BY 2.0