UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN

SOFTWARE CON MENCIÓN EN DISEÑO DE ARQUITECTURA DE SISTEMAS

RPC-SO-34-NO.778-2021

OPCIÓN DE TITULACIÓN:

PROYECTO DE TITULACIÓN CON COMPONENTES DE INVESTIGACIÓN APLICADA Y/O DE DESARROLLO

TEMA:

DESARROLLO DE UN PROTOTIPO DE ECOSISTEMA PARA LA INTEGRACIÓN EFECTIVA DE TECNOLOGÍAS POPULARES EN EL DESARROLLO DE SISTEMAS WEB: ANGULAR, NODE.JS, JWT, BCRYPT Y MYSQL.

AUTOR:

DANIEL ALEXANDER PATIÑO VÁSQUEZ

DIRECTOR:

JOE FRAND LLERENA IZQUIERDO

CUENCA – ECUADOR
2024

Autor:

Daniel Alexander Patiño Vásquez
Ingeniero en Sistemas.
Candidato a Magíster en Software por la Universidad Politécnica Salesiana – Sede Cuenca.
dpatinov@est.ups.edu.ec

Dirigido por:

Joe Frand Llerena Izquierdo
Ingeniero en Computación.
Magister en Sistemas de Información Gerencial.
Magister en Administración de Empresas.
Magister en Educación.
jllerena@ups.edu.ec

DEDICATORIA

Dedico este trabajo a mis queridos padres y hermanos, cuyo constante apoyo y estímulo han sido el motor detrás de cada uno de mis logros. Su inquebrantable fe en mí ha sido la luz que me ha guiado en este camino hacia la formación personal y académica.

Asimismo, quiero extender esta dedicatoria a mis sobrinos, con el firme propósito de inspirarles el valor del estudio y el poder de perseguir sus sueños. Que este trabajo sirva como un recordatorio de que, con esfuerzo y dedicación, pueden alcanzar cualquier meta que se propongan.

A todos ustedes, mi familia, les agradezco infinitamente por ser mi fuente inagotable de amor y motivación.

AGRADECIMIENTO

Quiero expresar mi profundo agradecimiento, en primer lugar, a mis padres, Lincoln y Claudia, cuya presencia ha sido fundamental a lo largo de todo mi proceso de formación, tanto a nivel personal como profesional. Su apoyo, sabiduría y amor incondicional han sido pilares invaluables que han guiado cada paso de mi camino.

Asimismo, agradezco de corazón a mis queridos hermanos, Diego, Leonardo y Claudia, por su constante respaldo y aliento durante este viaje. Su incondicionalidad ha sido una fuente de fuerza y motivación que nunca he subestimado.

No puedo dejar de mencionar mi gratitud hacia la Universidad Politécnica Salesiana y a sus distinguidos docentes. Gracias por brindarme la oportunidad de aprender y crecer académicamente en esta prestigiosa institución. Su dedicación y compromiso con la educación han sido fundamentales en mi desarrollo como profesional y como persona.

Tabla de Contenido.

Listado de figuras.

Listado de tablas.

Desarrollo de un prototipo de ecosistema para la integración efectiva de tecnologías populares en el desarrollo de sistemas web: Angular, Node.js, JWT, Bcrypt y MySQL.

Autor(es):

Daniel Alexander Patiño Vásquez

Resumen

Desde la creación de la World Wide Web en 1990, la tecnología ha experimentado una evolución continua y significativa. Esta progresión ha subrayado la importancia de la innovación constante en el ámbito tecnológico, destacando la necesidad de desarrollar herramientas y sistemas cada vez más seguros, interoperables y eficaces. En este contexto de cambio y avance perpetuo, se hace imprescindible investigar y desarrollar soluciones que respondan a las demandas actuales de seguridad y eficiencia en la gestión de información y recursos.

Partiendo del hito histórico de la creación de la World Wide Web en 1990 y la evolución hacia las generaciones sucesivas, se contextualiza la importancia de la innovación continua en el panorama tecnológico. Para el presente trabajo, la investigación se basó en una revisión de la literatura que incluya el desarrollo práctico de sistemas enfocados a la seguridad, la interoperabilidad y la documentación; durante el proceso de investigación, sobresalieron tecnologías que son generalmente usadas para el desarrollo de sistemas robustos, dichas tecnologías ayudaron a guiar a la propuesta del desarrollo de un ecosistema integral para una gestión eficiente empleando tecnologías clave como Angular, Node.js, JWT, Bcrypt y MySQL.

La metodología empleada incluyó la implementación de buenas prácticas de seguridad, la dockerización del proyecto y la documentación completa del proceso de instalación y configuración. Los hallazgos principales incluyen la creación de un backend robusto y seguro, así como un frontend dinámico y altamente interactivo. La documentación detallada y comprensible, junto con la versión controlada del proyecto en GitHub y su dockerización en la nube, garantizan la accesibilidad y el aprovechamiento completo de sus capacidades.

Para finalizar, el proyecto ofrece una solución integral para la gestión de establecimientos, demostrando la interoperabilidad fluida entre las tecnologías

clave y la importancia de implementar prácticas de seguridad sólidas en todos los niveles del desarrollo. Las implicaciones incluyen la mejora de la eficiencia operativa, la protección de la información sensible y la promoción de estándares de seguridad en el desarrollo de aplicaciones web modernas.

Palabras clave:

Desarrollo web, Patrones de diseño, Arquitectura de sistemas, Aplicaciones empresariales.

ABSTRACT

Since the creation of the World Wide Web in 1990, technology has undergone continuous and significant evolution. This progression has underlined the importance of constant innovation in the technological field, highlighting the need to develop increasingly secure, interoperable and efficient tools and systems. In this context of perpetual change and progress, it is essential to research and develop solutions that meet today's demands for security and efficiency in the management of information and resources.

Starting from the historical milestone of the creation of the World Wide Web in 1990 and the evolution towards successive generations, the importance of continuous innovation in the technological landscape is contextualized. For the present work, the research was based on a literature review that includes the practical development of systems focused on security, interoperability and documentation; during the research process, technologies that are generally used for the development of robust systems stood out, these technologies helped guide the proposal for the development of an integral ecosystem for efficient management using key technologies such as Angular, Node.js, JWT, Bcrypt and MySQL.

To conclude, the project provides a comprehensive solution for facility management, demonstrating seamless interoperability between key technologies and the importance of implementing sound security practices at all levels of development. The implications include improving operational efficiency, protecting sensitive information and promoting security standards in the development of modern web applications.

Palabras clave:

Web development, Design patterns, System architecture, Business application.

1. Introducción

1.1. Contexto previo.

1.1.1. Evolución histórica del Desarrollo de sistemas web.

En el contexto de la evolución histórica, un hito fundamental se produjo en noviembre de 1990 con la propuesta de Tim Berners-Lee y Robert Cailliau de un sistema de hipermedia denominado "The World-Wide Web" (Santos & Fundação Oswaldo Cruz. Instituto de Comunicação e Informação Científica e Tecnológica em Saúde. Rio de Janeiro, 2022). Esta iniciativa marcó el comienzo del desarrollo de la primera aplicación web; más tarde, en 1991, Tim Berners-Lee introdujo el primer navegador y servidor web, dando lugar al funcionamiento práctico de la World Wide Web.

Esta innovación permitió crear las primeras páginas web y sentó las bases para los sistemas web más complejos. La aplicación web inicial, aunque rudimentaria en comparación con las tecnologías actuales, introdujo la noción de documentos interconectados a través de enlaces hipertextuales, allanando el camino para la expansión y la evolución continua de las aplicaciones web a lo largo de las décadas. Este hito histórico es crucial para comprender el surgimiento y la transformación de las aplicaciones y sistemas web en el panorama tecnológico actual (Grigar Dene & O'Sullivan James, 2021).

Gracias a este hito histórico, se da paso a la llamada Web 1.0, la primera generación de la web, surgida en 1991, se caracterizó por ser un entorno informativo donde los usuarios solo podían leer y compartir datos estáticos. Aunque ofrecía ventajas como el acceso único y autonomía para los creadores de contenido, presentaba desafíos tecnológicos y limitaciones de interactividad. La incapacidad de los usuarios para editar datos y la falta de colaboración de ideas y conocimientos entre ellos condujeron a menos tráfico y publicidad. La Web 1.0 se considera una etapa cerrada

y menos amigable para el usuario, marcando el inicio de la necesidad de transformar a la web más interactiva y participativa (Nath, 2022).

Debido a la necesidad de interactuar y compartir conocimientos nace la Web 2.0 alrededor del año 2003; la Web 2.0 fue usada especialmente a través de Virtual Communities of Practice (VCoPs), que son grupos innovadores que aprovechan las tecnologías de la información para crear y compartir conocimientos. En aquella época tuvo mucha importancia la creación y gestión del conocimiento, teniendo un papel crucial las tecnologías de la información y la Web 2.0 en la transformación de la forma en que las personas interactúan y comparten conocimientos en diversos contextos organizativos, educativos y políticos (Ziegler, 2022).

Continuando con la reseña sobre la evolución de la web, a partir del año 2007 surge la conocida web 3.0 la cual propone un cambio hacia una internet descentralizada, donde los usuarios sean los principales creadores y árbitros de valor. La web 3.0 surge como respuesta a las críticas de la centralización en Web 2.0. Sin embargo, los defensores buscan una estructura basada en blockchain, destacando cambios en lo social, económico y cultural. Es así que, la web 3.0 socialmente, redefine interacciones y comunidades; económicamente, introduce la creación de valor centrada en el usuario con tecnologías como NFT y DeFi; culturalmente, empodera a los creadores (Chohan, 2022).

Como última generación se tiene la web 4.0 la cual presenta sus primeras impresiones a inicios del año 2016, y representa una revolución en la forma en que se interactúa con la información en línea. La web 4.0 se vislumbra como una red inteligente y simbiótica que implica la interacción entre humanos y máquinas; se espera que integre tecnologías como el big data, la realidad aumentada, la comunicación máquina a máquina, la computación en la nube y la inteligencia artificial con agentes inteligentes. Es así que, la Web 4.0 se percibe como una tecnología revolucionaria que conecta el Internet con nuevos objetos, permitiendo interacciones avanzadas en diversos contextos (Ersöz et al., 2020).

La evolución de los sistemas web, desde la concepción de "The World-Wide Web" en 1990 hasta la actualidad, ha marcado hitos significativos. Desde la Web 1.0, enfocada en la información estática, hasta la participativa Web 2.0 y la descentralizada Web 3.0, cada fase refleja la creciente interactividad y la transformación de la experiencia online. La Web 4.0, emergente desde 2016, promete una revolución al integrar tecnologías como inteligencia artificial y realidad aumentada. En conjunto, estas generaciones delinean una historia de constante innovación y cambio en el panorama tecnológico.

1.1.2. Métodos empleados en la creación de aplicaciones web.

En la evolución del desarrollo de software, las metodologías surgieron en los años 70 para abordar problemas inherentes a la falta de control en las etapas de desarrollo, resultando en productos deficientes (Molina et al., 2017). Con el auge del desarrollo de aplicaciones funcionales en la web, la implementación de metodologías ha mejorado significativamente la calidad del desarrollo; se destaca la comparativa de metodologías web, siendo la tecnología hipermedia orientado a objetos, por sus siglas en ingles OOHDM, se identifica como la más completa y eficiente (Molina et al., 2018). Adicionalmente, se están adoptando metodologías ágiles como programación extrema, por sus siglas en ingles XP, y Scrum para ajustarse a los requisitos cambiantes, proporcionando flexibilidad y eficiencia en la creación de aplicaciones web (Jacome Carrasco, 2022).

La metodología XP es una respuesta a la demanda de creación de aplicaciones informáticas, especialmente dentro de la rama del diseño de páginas web, que requiere sistemas adaptables y responsivos a diversos dispositivos (Triatama et al., 2023) . La metodología XP, desarrollada por Kent Beck, se enfoca en abordar la complejidad de los requisitos cambiantes y se basa en principios probados en la ingeniería de software (Bautista-Villegas, 2022). Se caracteriza por promover la comunicación efectiva, la simplicidad en el diseño, el feedback continuo con el cliente, el respeto dentro del equipo y la valentía para afrontar cambios. Sus características incluyen un ciclo dinámico, trabajo en parejas y 10 a 15 iteraciones

en un proyecto típico. Entre sus ventajas se encuentran la eficiencia en la planificación y pruebas, aplicabilidad a cualquier lenguaje de programación y facilidad de implementación en tecnologías actuales, aunque puede tener limitaciones en proyectos a largo plazo y complejos, así como en la falta de evidencia en cambios dinámicos frecuentes (Jacome Carrasco, 2022).

Continuando con las metodologías, La metodología Scrum es una estrategia eficiente para la creación de proyectos que busca adaptarse a entornos dinámicos y fomentar la colaboración en equipos. Se organiza en sprints, períodos de desarrollo de 2 a 4 semanas, con planificación y revisión al final de cada uno. La metodología se basa en roles como Maestro scrum (Scrum Master), dueño del producto (Product Owner), interesados (Stakeholders) y Equipo de programadores (developer group). Scrum se caracteriza por su enfoque en la autoorganización del equipo, la flexibilidad para adaptarse a cambios y la gestión sistemática de riesgos. Sus beneficios incluyen la obtención temprana de resultados, la capacidad de adaptarse a distintos entornos y la gestión estructurada de riesgos. No obstante, puede presentar desafíos en equipos pequeños y demanda una gestión intensiva de tareas y plazos, además de un equipo altamente capacitado para lograr su implementación exitosa (Jacome Carrasco, 2022). La metodología Scrum ha ganado aceptación generalizada y se ha adaptado a diversas disciplinas, aunque estas adaptaciones han generado una variedad de modificaciones y enfoques contextuales. Su popularidad lo convierte en una metodología para adaptaciones en diversos contextos y objetivos, y un componente clave para otros métodos. Como desventaja se puede indicar que a pesar de la amplia documentación, la falta de integración con un enfoque fijado en el método limita la generación acumulativa del aprendizaje (Hron & Obwegeser, 2022).

Otra metodología empleada es la OOHDM que se enfoca en seguir un proceso de desarrollo distribuido en 5 fases. Esta metodología combina gráficos en UML con otras propias de la metodología. Originalmente concebida para aplicaciones hipermedia, OOHDM fue adaptada con el crecimiento de Internet para la creación de desarrollos hipermedia enfocadas a la web, tales como los motores de búsqueda.

Esta metodología se enfoca en simplificar y mejorar la eficacia del diseño de aplicaciones web, empleando modelos específicos como conceptual, de navegación e de interfaz del usuario final. Las 5 etapas de OOHDM abarcan desde el mapeo de requerimientos hasta la implementación (Molina et al., 2018).

Metodologías como OOHDM, XP y Scrum han sido fundamentales en el desarrollo de aplicaciones informáticas, mejorando la calidad y la adaptabilidad de las mismas. OOHDM se destaca por su enfoque en aplicaciones web hipermedia, simplificando el proceso de diseño mediante fases como la obtención de requerimientos y el diseño conceptual. XP, centrada en la adaptabilidad, promueve la comunicación efectiva y la simplicidad. Scrum, con su gestión ágil basada en sprints, destaca por la obtención de resultados anticipados. Cada metodología contribuye de manera única al dinámico panorama de la creación de aplicaciones web.

1.1.3. Direcciones actuales en el Desarrollo de los sistemas web.

En la actualidad, el desarrollo de sistemas web está experimentando una rápida evolución impulsada por las tendencias tecnológicas emergentes. El internet, como poderoso medio de comunicación e influencia, ha desencadenado avances significativos que exigen una continua implementación tanto por parte de los clientes como de los programadores. Este dinamismo se refleja en la constante búsqueda de modelos y servicios web más eficientes y avanzados. La intersección de diversos lenguajes de programación, herramientas y plataformas ha dado lugar a un panorama tecnológico diversificado que optimiza la velocidad y precisión en el proceso de creación de diversos sistemas enfocados en la web. En este contexto de transformación digital, a continuación, se exploran las tendencias más destacadas que están configurando el presente y futuro del desarrollo de sistemas web.

En cuanto a JavaScript, se destaca su posición como el lenguaje interpretado más empleado en la creación de páginas web robustas, con una sintaxis que guarda similitudes con Java y C. De igual manera, se destaca su capacidad para ejecutarse en el lado del cliente, lo que contribuye a la agilidad y eficiencia en el desarrollo web. Además, se resalta que, a diferencia de PHP, JavaScript no implica intercambio

de datos con el servidor, lo que influye en el rendimiento (Valarezo Pardo et al., 2018).

Se menciona que el lenguaje de marcas de hipertexto, conocido por sus siglas en ingles HTML, es fundamental para el desarrollo de aplicaciones web estáticas, aunque su combinación con otros lenguajes permite crear aplicaciones dinámicas. Se introduce HTML5 como una evolución que introduce elementos dinámicos para configurar el entorno web y sus contenidos (Valarezo Pardo et al., 2018).

En relación a la autenticación y autorización, JSON Web Token (JWT) se considera especialmente conveniente para aplicaciones Serverless, esto debido a que JWT permite almacenar información adicional directamente en el token, más allá de sólo las credenciales del usuario, el servidor ya no necesita buscar esa información en la base de datos. Cuando un usuario inicia sesión con sus credenciales, estas se envían al Autorizador, y el JWT se devuelve en caso de éxito, es decir, la plataforma valida el JWT antes de otorgar acceso a los recursos protegidos optimizando la seguridad y la eficiencia en la gestión de la autenticación de usuarios en arquitecturas Serverless (Huynh, 2020).

Continuando con la seguridad, Bcrypt se emplea como un algoritmo de hash en el contexto de la autenticación basada en contraseñas en aplicaciones web; esta elección de Bcrypt se justifica por su capacidad para agregar una capa adicional de seguridad mediante el uso de "Salt", una cadena aleatoria añadida a la contraseña. Este procedimiento fortalece la seguridad al prevenir ataques directos e indirectos a la contraseña almacenada en la base de datos. Cuando un usuario intenta iniciar sesión, las credenciales ingresadas se comparan con la información de la base de datos utilizando la función de comparación proporcionada por el paquete Bcrypt. La implementación de Bcrypt asegura que, incluso si alguien obtiene acceso a la base de datos, las contraseñas permanezcan seguras y protegidas (Pant et al., 2022).

Como se aprecia, el panorama actual del desarrollo de sistemas web refleja una dinámica transformación impulsada por las tendencias tecnológicas emergentes. La evolución constante, impulsada por la influencia del Internet, ha llevado a la búsqueda de modelos y servicios más eficientes y avanzados. En este contexto, JavaScript destaca como el lenguaje interpretado más utilizado, proporcionando agilidad y eficiencia en el desarrollo web. La combinación de HTML y HTML5 sigue siendo fundamental para los nuevos desarrollos enfocados en aplicaciones web, ofreciendo tanto estructuras estáticas como dinámicas. En el contexto de la verificación y autorización, JSON Web Token (JWT) emerge como una solución que mejora tanto a la seguridad como a la eficiencia en la gestión de la autenticación de usuarios. Además, la implementación de Bcrypt para el hash de contraseñas refuerza la seguridad al agregar una capa adicional mediante el uso de "Salt", mitigando riesgos asociados con ataques directos e indirectos a las contraseñas almacenadas. En conjunto, estas tendencias reflejan el compromiso continuo con la innovación y la seguridad en la creación de sistemas web.

2. Establecimiento del Problema

2.1 Detalle del problema.

El problema aborda la necesidad de comprender las tendencias contemporáneas y las prácticas óptimas en la creación de aplicaciones web, considerando elementos como la arquitectura, los flujos de trabajo, la seguridad y la utilización de tecnologías específicas como Node.js, Angular, Bcrypt, JWT, MySQL y otros frameworks. La interrogante central se enfoca en cómo estos factores influyen en el transcurso de desarrollo de aplicaciones web en su conjunto. Esta formulación del problema guiará la investigación hacia una evaluación de las prácticas actuales en este campo, buscando identificar tanto las innovaciones más recientes como las estrategias de desarrollo más efectivas.

2.2 Planteamiento del problema.

¿Cuáles son las tendencias actuales y las mejores prácticas durante el desarrollo de las aplicaciones web, considerando aspectos como arquitectura, flujos de trabajo, seguridad, y la utilización de tecnologías específicas como Node.js, Angular, Bcrypt, JWT, MySQL y otros frameworks?

El planteamiento, consiste en una declaración clara y precisa que aborda los elementos fundamentales PICO (Población, Intervención, Comparación y Resultados) al explorar las tendencias o patrones en la creación de aplicaciones web. La pregunta se centra en la población general del desarrollo de aplicaciones web y busca entender cómo diferentes elementos, como arquitectura, flujos de trabajo, seguridad y tecnologías específicas como Node.js, Angular, Bcrypt, JWT, MySQL y otros frameworks, impactan en estos procesos. La formulación de la pregunta guiará la investigación hacia una evaluación comprensiva de las prácticas y tendencias actuales en el ámbito mencionado.

2.3 Argumentación del problema.

El inicio del presente desarrollo de ecosistema surge de la necesidad urgente de enfrentar los desafíos actuales relacionado con la creación de nuevas aplicaciones web empresariales. En un panorama tecnológico donde Angular, Node.js, JWT, Bcrypt y MySQL son reconocidas como tecnologías clave, se presenta la oportunidad de aprovechar su sinergia para ofrecer soluciones eficientes, escalables y seguras.

La relevancia de esta combinación reside en su capacidad para cambiar la manera en que las empresas proporcionan servicios en línea, lo cual marca un hito significativo en el avance tecnológico. La implementación eficaz de estas tecnologías promete no solo optimizar el rendimiento de las aplicaciones empresariales, sino también garantizar niveles superiores de seguridad y autorización. La concepción y creación de un prototipo funcional del sistema web constituye la piedra angular de este trabajo, aspirando a demostrar de manera tangible como la integración armoniosa de estas tecnologías puede generar soluciones de vital importancia. Estas soluciones, a su vez, están diseñadas para abordar la creciente demanda del entorno empresarial contemporáneo.

El propósito fundamental del presente trabajo es contribuir al corpus de conocimiento en torno a la optimización de tecnologías específicas, enfocadas especialmente en el desarrollo de aplicaciones empresariales. La meta es ofrecer a las empresas una ventaja competitiva palpable en un mercado en constante evolución. Esto se logrará mediante la implementación de un prototipo de sistema integral, seguro, eficiente y escalable, que responda de manera directa a los retos actuales y a las expectativas crecientes del entorno empresarial moderno. De esta manera, esta investigación aspira no solo a abordar problemáticas puntuales, sino también a proporcionar soluciones prácticas y sostenibles que impulsen la excelencia en el desarrollo de aplicaciones web.

2.4 Delimitación del problema.

El enfoque principal del presente proyecto radica en el diseño de componentes esenciales del ecosistema previo a su construcción, garantizando una interoperabilidad sin fisuras entre las tecnologías, es decir, el diseño y desarrollo implica la implementación de un sistema de comunicación eficiente entre el lado del cliente como en el lado del servidor, la integración de autenticación segura mediante JWT, el cifrado de datos sensibles con Bcrypt y el establecimiento de una gestión de datos eficiente en MySQL. Aunque el proyecto desarrollará componentes básicos, no abarcará el desarrollo de aplicaciones completas ni la optimización exhaustiva de componentes individuales ya que el propósito fundamental es identificar los posibles puntos de apoyo como los puntos débiles de la implementación de un sistema mediante la combinación de las tecnologías anteriormente mencionadas, es por esto que el presente trabajo se encuentra limitado al desarrollo de un prototipo funcional.

2.5 Importancia del trabajo.

Para la ejecución del presente trabajo, se planea la creación de un prototipo que incorpora las funciones esenciales como es la creación, modificación, consultas y eliminación de registros, dichas funciones serán necesarias para su eficiente funcionamiento. La relevancia de esta iniciativa radica en la oportunidad de mostrar cómo una integración eficiente de las tecnologías mencionadas puede manifestarse como un sistema robusto y escalable. Una vez que el prototipo esté completamente implementado, se procederá a documentar todos los aspectos pertinentes para facilitar su replicación en otros contextos o áreas relacionadas. El propósito principal de este prototipo de ecosistema es destacar tanto a las empresas como a los desarrolladores la importancia crítica de una implementación precisa y cohesionada de las tecnologías, fomentando así una sinergia efectiva entre ellas y evitando futuros problemas de mantenimiento o escalamiento del sistema.

Además, se tiene previsto que el proyecto esté accesible al público en general a través de un repositorio dedicado, lo que permitirá a otros desarrolladores de aplicaciones web examinar, adoptar e incluso optimizar dicha implementación. Este enfoque de disponibilidad abierta no solo promueve la transparencia y el intercambio de conocimientos, sino que también fomenta la colaboración y el aprendizaje continuo en la comunidad de creación de aplicaciones web.

2.6 Objetivos.

2.6.1 Objetivo general.

Desarrollar un ecosistema integrando de manera efectiva las tecnologías populares en el desarrollo de sistemas web incluyendo Angular, Node.js, JWT, Bcrypt y MySQL, con el propósito de agilizar y estandarizar la creación de proyectos web seguros y eficientes fomentando las mejores prácticas.

2.6.2 Objetivos específicos.

- Realizar una investigación de trabajos relacionados, mediante un análisis en bases de datos académicas y recursos online relacionados al tema, con el propósito de comprender las capacidades, ventajas y sinergias de las tecnologías clave: Angular, Node.js, JWT, Bcrypt y MySQL.
- Diseñar y prototipar los componentes esenciales, mediante el uso de: Angular, Node.js, JWT, Bcrypt y MySQL, asegurando la interoperabilidad fluida entre las tecnologías, con el propósito de implementar un sistema de comunicación efectivo entre el frontend y el backend.
- Documentar la instalación, configuración y uso del ecosistema, mediante una documentación completa y comprensible, con el propósito de fomentar la adopción y el aprovechamiento completo de sus capacidades.

3. Marco teórico referencial

Se exploran proyectos de desarrollo web que destacan la integración de diversas tecnologías para abordar desafíos específicos. Desde el mejoramiento de la gestión de historias clínicas en el ámbito de la salud mediante el uso de Angular framework, hasta la creación de aplicaciones web completas utilizando Node.js para gestionar conexiones a Internet de clientes, cada proyecto demuestra la eficacia de la integración tecnológica en el desarrollo web. Además, la implementación de MySQL en una aplicación basada en CodeIgniter y el uso de JSON Web Token (JWT) y bcrypt para garantizar la seguridad en la autenticación de usuarios en aplicaciones web, resaltan la diversidad de enfoques y tecnologías utilizadas en el desarrollo web moderno.

En un proyecto titulado “Desarrollo de un sistema web utilizando angular framework y rest (Transferencia de estado representacional) para la gestión de historias electrónicas” (Conza Ccolque, 2019), aborda la problemática de la gestión ineficiente de historias clínicas en el ámbito de la salud en Perú, específicamente en la ciudad de Juliaca. Se destaca que gran parte de los gerentes de organizaciones de salud desconocen los beneficios de una administración adecuada de historias clínicas. Ante la necesidad de gestionar electrónicamente las historias clínicas, se menciona que Angular se integró en el proyecto como un framework de JavaScript que facilita la creación de aplicaciones web. Angular proporciona características como enlace de datos, enrutamiento y animaciones, simplificando el desarrollo de aplicaciones modernas. Además, se explica que la adopción de Angular busca mejorar la accesibilidad y gestión de la información en las historias clínicas, optimizando el proceso de diagnóstico y tratamiento de pacientes en la Clínica Dermacenter Ríos en Juliaca.

Por otro lado, en el trabajo “Designing a Node.js full stack web application” (Janne Kinnunen, 2023), Node.js es un entorno de ejecución de JavaScript que se utiliza para construir servidores web seguros y versátiles. En este caso, se empleó para

desarrollar una aplicación web completa para un proveedor de servicios de Internet. Gracias a su amplio ecosistema, Node.js permitió la construcción de un servidor robusto y versátil, junto con una API para ingresar a la base de datos. El ecosistema resultante, diseñada para funcionar en diversos dispositivos, se implementó con éxito en la intranet de la empresa, brindando a los empleados la capacidad de gestionar y controlar las conexiones a Internet de los clientes. Este enfoque full-stack demuestra la eficacia de Node.js en un entorno de desarrollo de aplicaciones escalables y seguras.

La integración de MySQL desempeña un rol crucial en el desarrollo del Aplikasi Website Portal Manajemen Informatika, basado en el framework CodeIgniter. La aplicación, diseñada para ofrecer información actualizada sobre la gestión de la informática, aprovecha las capacidades de MySQL para gestionar eficazmente diversos conjuntos de datos. La implementación de MySQL permite una estructura organizada y accesible, contribuyendo así a la rapidez y eficacia en la obtención de información tanto para los estudiantes de informática como para la comunidad en general (Ramadhan et al., 2020).

En el desarrollo de un trabajo titulado Development of a modern full stack web application (Aleksi Kujala, 2023), en relación de la autenticación de usuarios en aplicaciones web, se ha empleado JSON Web Token (JWT), y para afianzar la seguridad de datos sensibles, como las contraseñas, se ha implementado bcrypt para el hash de las contraseñas de los usuarios. Al registrarse, la aplicación verifica la no existencia de usuario y correo proporcionados en la base de datos antes de agregar al usuario con el hash de la contraseña correspondiente. Al utilizar bcrypt para el hash de contraseñas, el desarrollador debe determinar el número de rondas de sal utilizadas para calcular la complejidad y el tiempo de procesamiento de la operación de hash. Las rondas de sal representan el factor de costo, determinando cuánto tiempo se necesita para calcular un solo hash bcrypt, con cada ronda adicional incrementando el factor de costo y duplicando el tiempo de cálculo.

Se han analizado proyectos que ilustran la combinación de diferentes tecnologías en entornos de desarrollo web. Desde la mejora de la gestión de historias clínicas

con Angular hasta la creación de aplicaciones web completas utilizando Node.js, estos casos destacan la efectividad de combinar tecnologías para enfrentar desafíos específicos. La adopción de MySQL en CodeIgniter y la implementación de medidas de seguridad como JSON Web Token (JWT) y bcrypt refuerzan la idea de que la integración tecnológica es fundamental en el desarrollo web contemporáneo. Estos ejemplos resaltan la versatilidad y el impacto positivo que puede tener la combinación estratégica de herramientas en la resolución de incidencia y la optimización de procesos en diversos entornos.

A continuación, se indican términos importantes para una mejor comprensión del presente trabajo:

3.1 Tecnología en desarrollo frontend avanzado - Angular,

Angular es una tecnología en desarrollo frontend avanzado que se basa en un marco conceptual para desarrollar aplicaciones web sofisticadas centradas en el cliente. Desarrollado por Google y basado en el lenguaje TypeScript; Angular utiliza TypeScript como su principal herramienta de codificación debido a su verificación de tipos y mejor estructuración de código. El framework estructura el código alrededor de componentes que tienen baja dependencia entre sí, donde cada uno cuenta con su propio HTML, CSS y controlador escrito en TypeScript. Además de los componentes de Angular, también facilita la creación de servicios, directivas y otros elementos fundamentales para el desarrollo de aplicaciones web (Cincovic et al., 2019).

3.2 Tecnología en servidor y programación eficiente - Node.js.

En la tecnología de Node.js, se implementa mediante el uso del lenguaje de desarrollo JavaScript para el servidor el cual sigue el paradigma "JavaScript en todas partes". Se basa en una arquitectura dirigida por eventos y permite la ejecución

asíncrona de código. En la pila MEAN, Node.JS se integra como el entorno del lado del servidor junto con Express, un marco minimalista que facilita la creación y enrutamiento de servicios web. La implementación de servicios en Node.JS sigue una estructura de capas separadas para lograr un código más seguro y comprensible (Cincovic et al., 2019).

3.3 Tecnología en gestión de identidad y acceso - JSON Web Token (JWT).

JWT es un medio ligero para intercambiar datos entre 2 partes, facilitando la autenticación, autorización y seguridad. Cada declaración JWT se almacena como una entidad json, utilizada como texto sin formato en el cifrado de Json Web Encryption o como carga útil de JSON Web Signature (JWS), lo que permite que las afirmaciones se aseguren y autentiquen digitalmente con el Código de Autenticación de Mensajes (MAC). Antes de la revolución de JWT, un 'token' era simplemente una cadena sin un valor inherente, con JWT, los tokens codifican y verifican sus propias afirmaciones, lo que resulta en JWTs autónomos de corta duración y sin necesidad de acceso al repositorio de datos, simplificando el diseño y eliminando la carga (Mahindrakar & Pujeri, 2020).

3.4 Tecnología en cifrado y protección de datos - Bcrypt.

La tecnología de cifrado y protección de datos bcrypt, es una función hash utilizada en la programación de contraseñas en UNIX, creada a partir de la combinación de Blowfish y Crypt. Su proceso de inicialización, llamado "eksblowfish", ralentiza el descifrado de contraseñas al incorporar una clave costosa de Blowfish. Esta implementación de Bcrypt es resistente a las tablas arcoíris gracias al uso de una sal de 128 bits en el proceso de hash. Consta de 2 pasos: primero, establece una clave inicial utilizando eksblowfish y, luego, cifra con OrpheanBeholderScryDoubt usando una clave de 192 bits generada previamente. Este enfoque brinda seguridad

adicional mediante el uso de una sal y una estructura que dificulta ataques potenciales (Giffary & Ramadhani, 2022).

3.5 Tecnología en almacenamiento de datos - MySQL.

MySQL es un software para administrar repositorios de datos ampliamente utilizado en proyectos web dinámicos; es esencial para la creación de proyectos web, permitiendo la búsqueda eficiente y manipulación de datos. Su popularidad entre los programadores se debe a su capacidad multiplataforma, facilidad de uso y sistema de seguridad confiable con licencia gratuita. MySQL mantiene un gran impacto en el desarrollo tanto del frontend como del backend de proyectos web, siendo fundamental para almacenar y gestionar datos de manera efectiva (Sotnik et al., 2023).

4. Desarrollo del proyecto.

4.1 Enfoques metodológicos.

Entre los métodos a emplear se encuentran:

- Revisión bibliográfica y análisis de trabajos relacionados: Este método implica investigar y revisar estudios académicos, artículos científicos y recursos en línea asociados con el uso de tecnologías para la creación de aplicaciones web. Esto enriquece la comprensión del estado actual del campo, buenas prácticas, desafíos comunes y soluciones propuestas.
- Desarrollo: Se llevó a cabo por medio de la combinación del desarrollo iterativo e incremental, junto con el desarrollo basado en componentes. Mediante el enfoque iterativo, se logró crear múltiples versiones de diversos componentes, permitiendo así la incorporación gradual de nuevas funcionalidades más específicas. Por otro lado, la metodología del desarrollo basado en componentes posibilitó la reutilización de

secciones de código tanto en el backend como en el frontend, con el propósito de aprovechar componentes de software preexistentes siempre que fuera factible. Esta estrategia contribuyó significativamente a acelerar el proceso de desarrollo. La implementación del prototipo funcional del sistema se realizó utilizando las tecnologías objeto de investigación. Esto permitió explorar de manera práctica la aplicación de estas tecnologías, así como su interacción entre sí, y cómo pueden satisfacer los requisitos específicos del sistema prototipo.

- Almacenamiento en repositorio GitHub: Se podría afirmar que se adoptó una metodología ágil, caracterizada por la colaboración continua y la adaptación a los cambios a lo largo del desarrollo. Esta elección se fundamenta en el hecho de que el repositorio GitHub registra de manera sistemática los cambios y las entregas sucesivas del proyecto. El almacenamiento en este repositorio no solo facilita la accesibilidad al trabajo por parte de otros desarrolladores, sino también brinda la facilidad de revisar el código, entender la metodología utilizada e incluso contribuir con sus propias mejoras. Esta práctica promueve la difusión del proyecto y fomenta la colaboración en la comunidad de desarrollo de software.

4.2 Reconocimiento de requerimientos de datos.

Para la selección de fuentes de información se realizó mediante el método prisma iniciando con la pregunta de investigación definida en la sección de Planteamiento del problema. para luego definir ciertos puntos clave de exclusión y de inclusión:

- Fecha de Publicación:
  - Se incluyen estudios publicados en los años 2019-2023.
  - Se excluyen estudios publicados antes del 2019.
- Tipo de Estudio:

- Se incluyen estudios empíricos, revisiones sistemáticas y estudios de caso.
- Se excluyen estudios con casos puntuales que no indiquen recomendaciones de implementación.

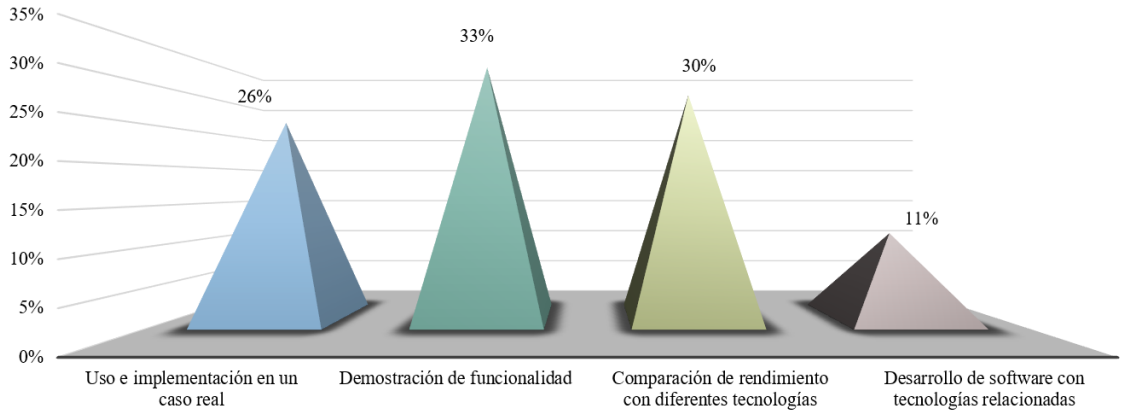
- Metodología:
  - Se incluyen solo estudios que utilicen una metodología cualitativa o cuantitativa.
  - Se excluyen estudios que no cumplan con los términos de inclusión.
- Relevancia Temática:
  - Se incluyen estudios que abordan aspectos específicos de un tema o tecnología.
  - Se excluyen estudios que no estén directamente relacionados con el enfoque de investigación.
- Calidad del Estudio:
  - Se incluyen sólo estudios con altos estándares metodológicos.
  - Se excluyen estudios con limitaciones metodológicas significativas.

Asi mismo, como expresión de búsqueda se utilizó la siguiente cadena:

- ("Information System" OR "Extreme Programming") AND ("AWS" OR "Restful APIs" OR "Postman") AND ("Angular" OR "Node.js" OR "MySQL" OR "bcrypt" OR "JWT") AND ("Web development" OR "Scrum" OR "XP" OR "Agile methodologies").

Una vez aplicados dichos criterios, se obtuvieron un total de 105 estudios, sin embargo, luego de realizar una revisión individual de cada trabajo se lograron filtrar un total de 27 estudios relevantes por lo cual se realizó un análisis obteniendo el siguiente resultado:

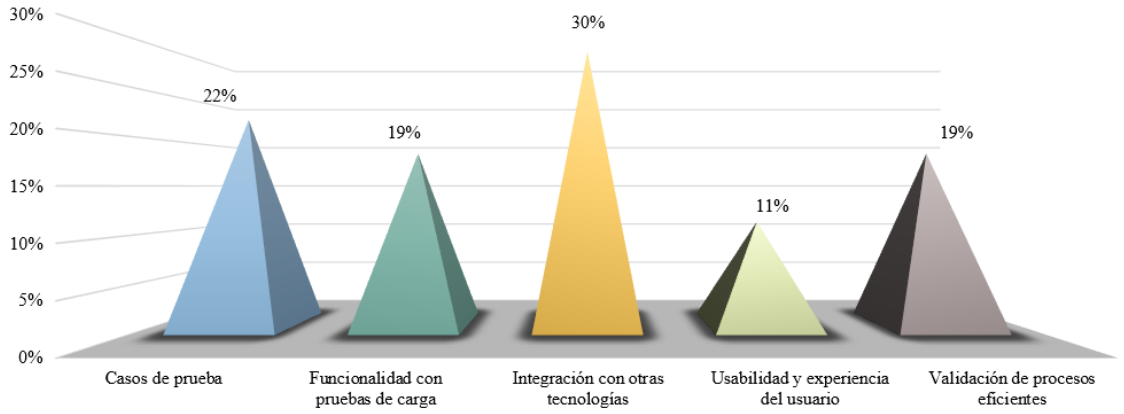
Figura 1: Resultados enfocados en temáticas y objetivos.



Elaborado por: Autor.

De acuerdo al análisis de temáticas y objetivos se obtuvo que los trabajos relacionados con el uso e implementación en un caso real 26%, demostración de funcionalidad 33%, comparación de rendimiento con diferentes tecnologías 30% y el desarrollo de software con tecnologías relacionadas 11%.

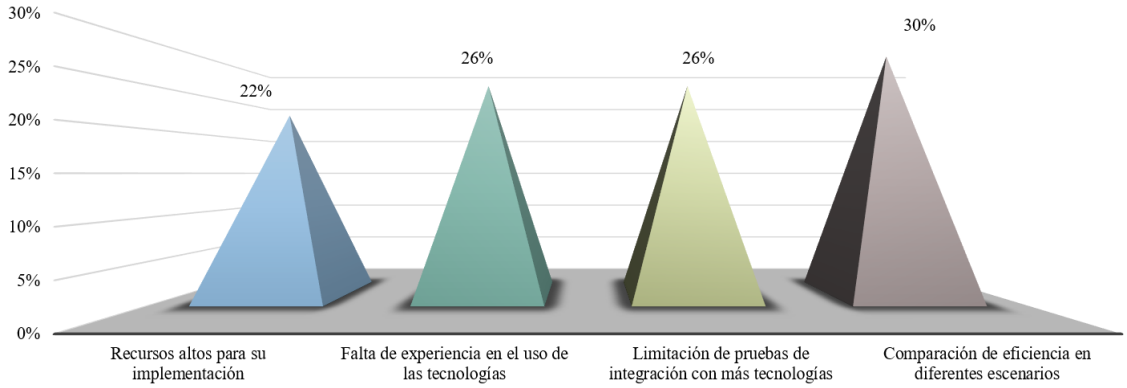
Figura 2: Resultados enfocados en formulación del problema y datos implicados.



Elaborado por: Autor.

De acuerdo al análisis de formulación del problema y datos implicados se obtuvo que los trabajos relacionados con casos de prueba 22%, funcionalidad con pruebas de carga 19%, integración con otras tecnologías 30%, usabilidad y experiencia del usuario 11% y la validación de procesos eficientes 19%.

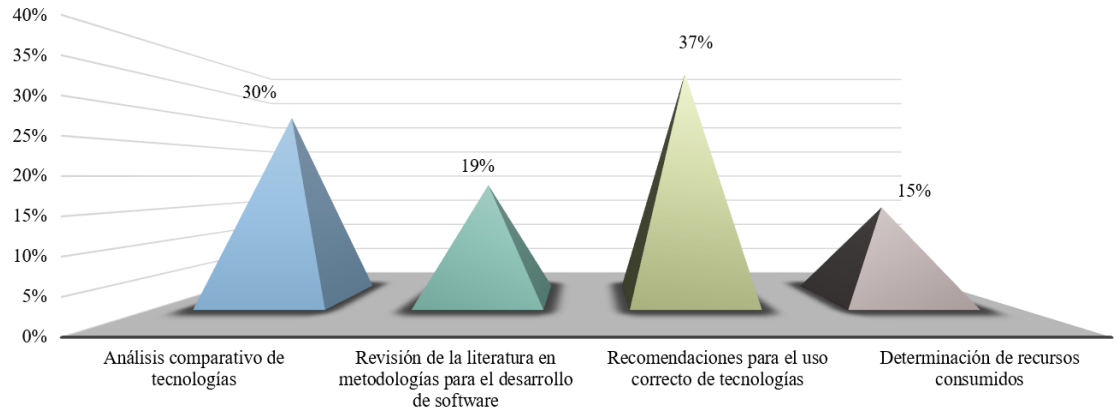
Figura 3: Resultados enfocados en limitaciones encontradas.



Elaborado por: Autor.

De acuerdo al análisis de limitaciones encontradas se obtuvo que los trabajos relacionados con recursos altos para su implementación 22%, falta de experiencia en el uso de las tecnologías 26% y comparación de eficiencia en diferentes escenarios 30%.

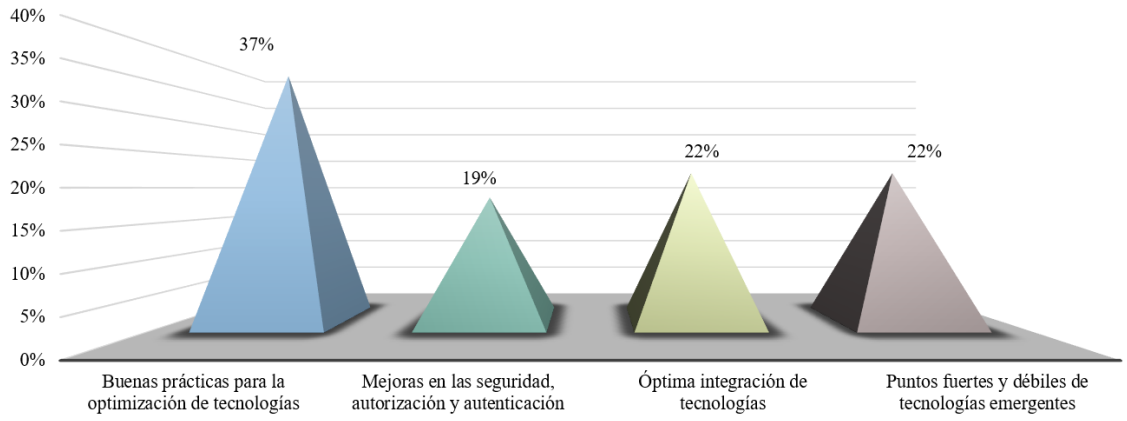
Figura 4: Resultados enfocados en propuestas y metodología aplicada.



Elaborado por: Autor.

De acuerdo al análisis de propuestas y metodologías empleadas se obtuvo que los trabajos relacionados con el análisis comparativo de tecnologías 30%, revisión de la literatura en metodologías para el desarrollo de software 19%, recomendaciones para el uso correcto de tecnologías 37% y determinación de recursos obtenidos 15%.

Figura 5: Resultado enfocados en la solución encontrada, resultados y desafíos.



Elaborado por: Autor.

De acuerdo al análisis de solución encontrada, resultados y desafíos se obtuvo que los trabajos relacionados con buenas prácticas para la optimización de tecnologías 37%, mejoras en la seguridad, autorización y autenticación 19%, óptima integración de tecnologías 22% y puntos fuertes y débiles de tecnologías emergentes 22%.

4.3 Métodos de evaluación de procesos.

Para una funcionalidad eficiente se usaron diagramas de flujo en el cual se reflejan los pasos a seguir para cada escenario:

- Login / Register: El diagrama para el presente escenario de login / register, se enfoca en el correcto flujo del sistema cuando un usuario realiza la acción de iniciar sesión o de registrarse, asi mismo, se indican los posibles flujos en caso de presentar algún error durante dicho proceso.
- Registrar un producto: El diagrama de registrar un producto indica los pasos a seguir y validar para que un producto sea creado correctamente, siempre y cuando el rol del usuario en sesión sea el requerido.
- Actualizar un producto: En el presente diagrama de actualizar un producto, se valida el nivel de permisos que tiene el rol del usuario

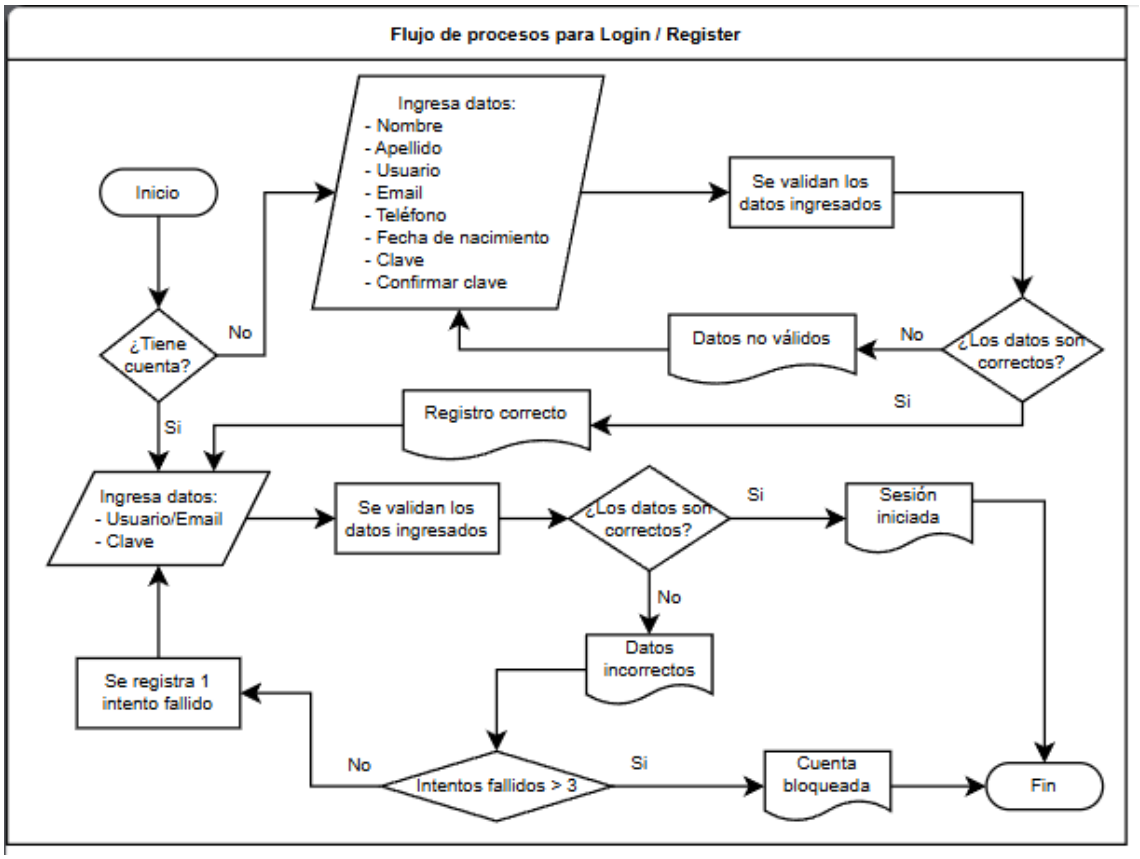
iniciado sesión, caso contrario, no tendrá posibilidad de realizar dicha acción.

- Eliminar un producto: Para la eliminación de un producto, el sistema verifica el nivel de permisos del usuario en sesión para permitir la acción de eliminar, caso contrario esta acción estará deshabilitada.

Los diagramas de flujo en programación son representaciones gráficas de algoritmos o procesos, utilizados como herramientas visuales para la enseñanza y comprensión de la lógica de programación. Estos diagramas permiten modelar de manera intuitiva y estructurada el flujo de control de un algoritmo, mostrando las diferentes etapas, decisiones y acciones que se realizan durante su ejecución (Jesús et al., 2022).

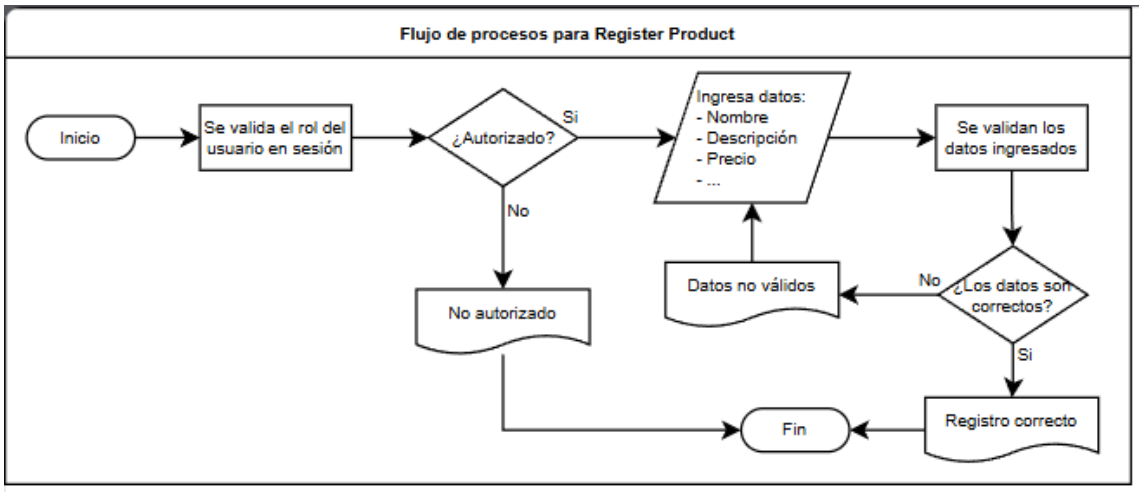
Además de su aplicación en la enseñanza de la programación, los diagramas de flujo también se utilizan como parte fundamental de técnicas para diagnosticar procesos. Se emplean para representar los pasos a seguir en diferentes escenarios, como se indica en la Figura 6, Figura 7, Figura 8 y Figura 9

Figura 6: Flujo de procesos para iniciar sesión y registro.



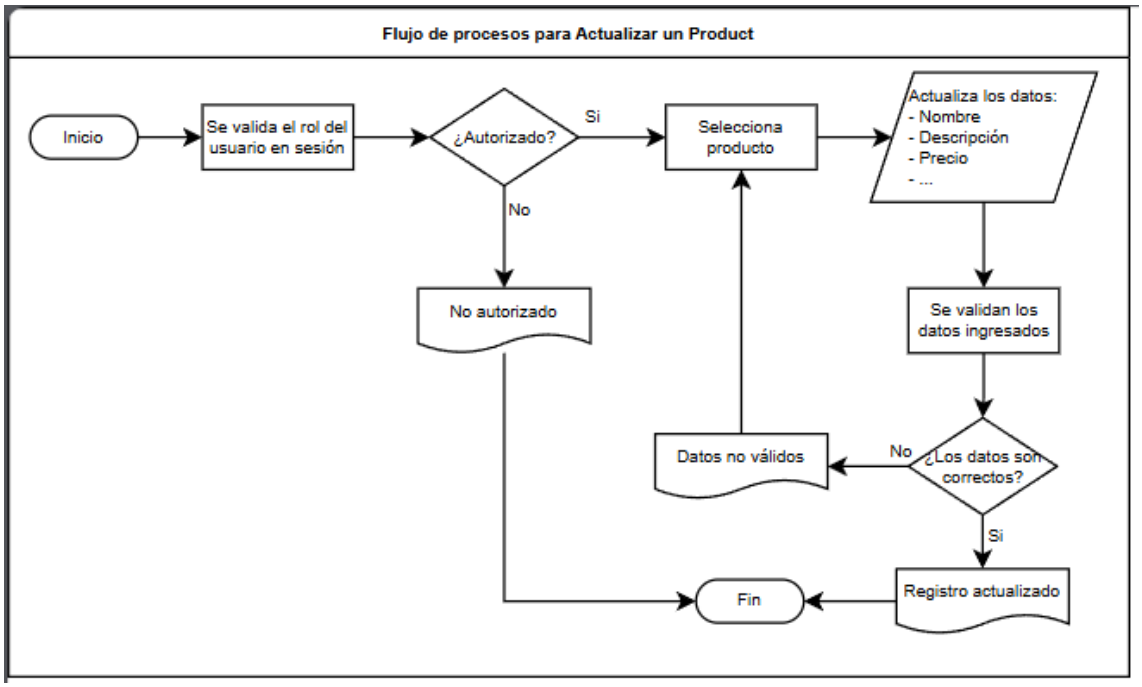
Elaborado por: Autor.

Figura 7: Flujo de procesos para registrar un producto.



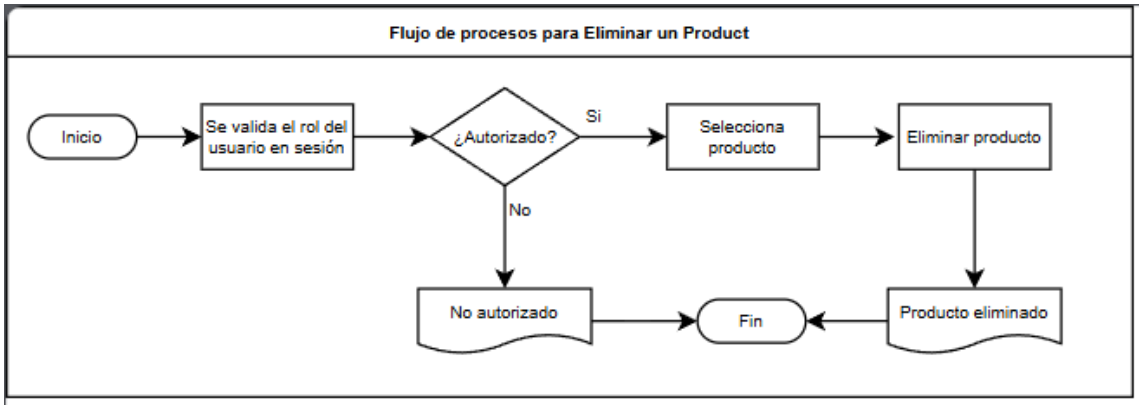
Elaborado por: Autor.

Figura 8: Secuencia de procesos para actualizar un producto.



Elaborado por: Autor.

Figura 9: Secuencia de procesos para eliminar un producto.



Elaborado por: Autor.

Por otro lado, para la verificación de funciones o métodos ya sea del lado del frontend o backend, se optó por el uso de un check list en el cual se registraron los campos necesarios y validaciones previas a su ejecución tal como se indica en la Tabla 1 y Tabla 2.

Tabla 1: Check List para registrar un usuario.

Check List para registrar un usuario.
Nombre.
☐ Campo de entrada para ingresar texto.

☐ Validación de campo no vacío. ☐ Longitud máxima y mínima.
Apellido.
☐ Campo de entrada para ingresar texto. ☐ Validación de campo no vacío. ☐ Longitud máxima y mínima.
Usuario.
☐ Campo de entrada para ingresar texto. ☐ Validación de campo no vacío. ☐ Longitud máxima y mínima.
Correo electrónico.
☐ Campo de entrada para ingresar texto. ☐ Validación de campo no vacío. ☐ Longitud máxima y mínima. ☐ Validación de campo de tipo email.
Género.
☐ Validación de campo no vacío. ☐ Selección de registro de tipo lista.
Teléfono.
☐ Campo de entrada para ingresar número. ☐ Validación de campo no vacío. ☐ Longitud máxima y mínima.
Fecha de Nacimiento.
☐ Campo de tipo date para seleccionar una fecha. ☐ Validación de campo no vacío.
Clave.
☐ Campo de entrada para ingresar texto. ☐ Validación de campo no vacío. ☐ Longitud máxima y mínima. ☐ Validación para incluir al menos un carácter especial. ☐ Validación para incluir al menos un número. ☐ Validación para incluir al menos una letra mayúscula.
Confirmar clave.
☐ Validación de campo que contenga la misma información de la clave. ☐ Validación de campo no vacío.

Elaborado por: Autor.

Tabla 2: Checklist para registrar un producto.

Check List para registrar un producto.
Nombre.
☐ Campo de entrada para ingresar texto. ☐ Validación de campo no vacío. ☐ Longitud máxima y mínima.
Descripción.

▯ Campo de entrada para ingresar texto. ▯ Validación de campo no vacío. ▯ Longitud máxima y mínima.
Precio.
▯ Campo de entrada para ingresar un valor numérico con decimales. ▯ Validación de campo no vacío.
Selección de archivo.
▯ Validación de campo no vacío. ▯ Tamaño máximo de 1080px X 1080px.

4.4 Instrumentos para el estudio y comprensión de datos.

Para la interpretación de los datos del lado del backend, se empleó la herramienta Postman debido a su versatilidad y amplias configuraciones que permiten realizar pruebas en diversas funcionalidades del backend desarrollado en Node.js. Postman facilitó la ejecución de pruebas funcionales para los métodos implementados, tales como iniciar sesión, registrar usuarios, gestionar productos, realizar lecturas y modificaciones de datos, eliminar registros, verificar permisos y validar información encriptada. Su interfaz intuitiva y la posibilidad de configurar diferentes escenarios de prueba fueron fundamentales para garantizar el eficiente funcionamiento y seguridad del sistema backend. Además, Postman es una herramienta de prueba que proporciona funcionalidades para probar APIs y es utilizada por los equipos de Aseguramiento de Calidad (QA) y Operaciones. Facilita la creación de patrones de prueba para casos de uso de pruebas de producción (Ranta, n.d.).

Por otro lado, en el ámbito del frontend, se optó por utilizar el navegador web Edge gracias a su flexibilidad y capacidad para adecuarse a diferentes exigencias del usuario. En este caso, era crucial contar con un diseño responsive, ya que el prototipo del ecosistema debía ser accesible desde una variedad de dispositivos, como laptops, teléfonos móviles y tabletas. Gracias a esta elección, se pudo visualizar cómo se presentaría el resultado final, permitiendo una interacción óptima para los usuarios finales, independientemente de su dispositivo. Además, se aprovechó esta plataforma para mostrar las diferentes interfaces según el rol del

usuario, ya sea cliente, encargado o administrador, proporcionando accesos diferenciados según sus necesidades y permisos.

Finalmente, para el manejo de datos en MySQL, se optó por utilizar MySQL Workbench debido a los beneficios que ofrece como herramienta de administración de bases de datos. Esta elección se basó en su amplia gama de funcionalidades, que incluyen la facilidad de diseño y modelado de repositorio de datos, la ejecución de consultas SQL, la administración de usuarios y permisos, la optimización del rendimiento y la visualización de datos mediante gráficos y tablas. Además, su interfaz intuitiva y su capacidad para gestionar eficientemente los datos fueron fundamentales para garantizar un manejo eficaz y seguro de la información almacenada en MySQL.

4.5 Limitaciones y dificultades encontradas a lo largo del desarrollo de trabajo.

Una limitación crucial que se enfrentó fue la falta de recursos y tiempo, lo que afectó la capacidad para presentar un producto completamente finalizado y listo para ser implementado en un entorno productivo. Esta limitación fue a causa de la necesidad de realizar pruebas exhaustivas en una variedad de entornos y circunstancias para garantizar el rendimiento y estabilidad general del sistema en casos reales.

Además, dada la naturaleza del ecosistema como una aplicación web, surgió la necesidad de múltiples instancias para su lanzamiento en producción. Esto se debe a que la arquitectura del sistema está diseñada para integrar tecnologías como React.js en el frontend, Node.js en el backend y MySQL para la gestión de datos. Cada una de estas tecnologías requiere su propia configuración y despliegue, lo que añade complejidad al proceso de implementación y aumenta la necesidad de recursos adicionales, tanto en términos de tiempo como de infraestructura.

4.6 Descripción completa del desarrollo del trabajo.

Para el desarrollo del prototipo de ecosistema, se seleccionó una temática base que permitiera plasmar las principales funcionalidades de la aplicación web. La temática elegida debía estar relacionada con un entorno que pudiera replicarse en múltiples ocasiones, y para este proyecto se optó por la gestión de bares o licoreras. Esta elección se fundamentó en la observación de que, en una localidad típica, suelen existir varios establecimientos de este tipo, cada uno con características similares, como un menú o carta, promociones y eventos.

El objetivo fue diseñar una aplicación que permitiera a los administradores gestionar eficientemente estos establecimientos, asignar roles a los usuarios correspondientes y garantizar una experiencia óptima tanto para los encargados como para los clientes. Las funciones principales de la aplicación se dividieron en 3 roles: administrador, encargado y cliente.

El administrador tendría la capacidad de registrar y modificar los establecimientos, así como de asignar roles a los usuarios según sea necesario. Por su parte, los encargados podrían agregar menús con sus respectivas características, como título, descripción y precio, así como también gestionar promociones y eventos, especificando horarios y detalles relevantes, para más detalles como controles y permisos se pueden observar los diagramas de flujo mencionados anteriormente en la sección de Métodos de evaluación de procesos.

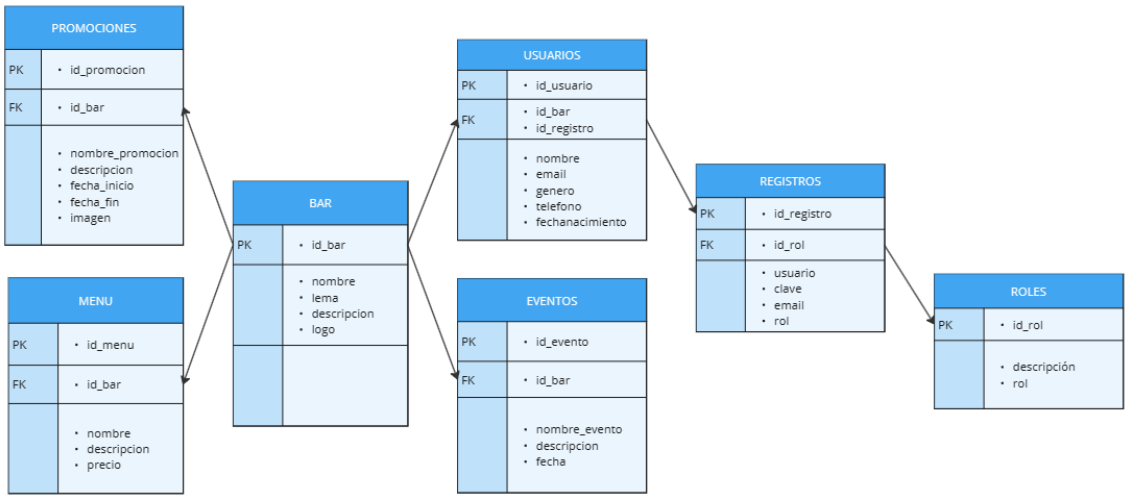
Finalmente, los clientes tendrían acceso a todos los establecimientos registrados en la plataforma, pudiendo explorar los diferentes menús, promociones y eventos disponibles en cada uno de ellos. Esta descripción general del proyecto establece la base para el desarrollo de un prototipo funcional que satisfaga las necesidades de gestión y experiencia de usuario en el contexto de bares y licoreras.

A continuación, se detalla el proceso que fue llevado a cabo con el propósito de ajustar correctamente la implementación del proyecto, desde la gestión de datos hasta las pantallas expuestas al usuario.

4.6.1 Base de datos.

Se utilizó MySQL Workbench en su versión 8.0.33. Se procedió a diseñar las principales tablas del repositorio de datos en base a una recopilación de requerimientos e información general presentes en los establecimientos de interés. El resultado final de esta integración de tablas en el repositorio de datos se presenta de manera visual en la Figura 10.

Figura 10: Estructura funcional de la base de datos.



Elaborado por: Autor.

A continuación en la Figura 11, Figura 12, Figura 13, Figura 14, Figura 15, Figura 16 y Figura 17, se procedió a crear las principales tablas funcionales en MySQL utilizando MySQL Workbench.

Figura 11: Elaboración de tabla BAR

```sql
DROP TABLE IF EXISTS `tbar`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tbar` (
  `id_bar` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL,
  `lema` varchar(45) NOT NULL,
  `descripcion` varchar(45) NOT NULL,
  `logo` varchar(100) NOT NULL,
  `fcreacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `fmodificacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id_bar`)
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Elaborado por: Autor.

Figura 12: Elaboración de tabla EVENTOS

```sql
DROP TABLE IF EXISTS `teventos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `teventos` (
  `id_evento` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL,
  `descripcion` varchar(255) NOT NULL,
  `fecha` date NOT NULL,
  `fcreacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `fmodificacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `id_bar` int NOT NULL,
  `image` varchar(100) NOT NULL,
  PRIMARY KEY (`id_evento`),
  KEY `fk_id_bar_eventos` (`id_bar`),
  CONSTRAINT `fk_id_bar_eventos` FOREIGN KEY (`id_bar`) REFERENCES `tbar` (`id_bar`) ON DELE
TE RESTRICT
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

```

Elaborado por: Autor.

Figura 13: Elaboración de tabla MENÚ.

```sql
DROP TABLE IF EXISTS `tmenu`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tmenu` (
  `id_menu` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL,
  `descripcion` varchar(255) NOT NULL,
  `precio` decimal(10,2) NOT NULL,
  `fcreacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `fmodificacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `id_bar` int NOT NULL,
  `image` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id_menu`),
  KEY `fk_id_bar_menu` (`id_bar`),
  CONSTRAINT `fk_id_bar_menu` FOREIGN KEY (`id_bar`) REFERENCES `tbar` (`id_bar`) ON DELETE RESTRICT
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Elaborado por: Autor.

Figura 14: Elaboración de tabla PROMOCIONES.

```sql
DROP TABLE IF EXISTS `tpromociones`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tpromociones` (
  `id_promocion` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL,
  `descripcion` varchar(225) NOT NULL,
  `fecha_inicio` timestamp(6) NOT NULL,
  `fecha_fin` timestamp(6) NOT NULL,
  `id_bar` int NOT NULL,
  `fcreacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `fmodificacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `image` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id_promocion`),
  KEY `fk_id_bar_promocion` (`id_bar`)
) ENGINE=MyISAM AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Elaborado por: Autor.

Figura 15: Elaboración de tabla REGISTROS.

```sql
DROP TABLE IF EXISTS `tregistros`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tregistros` (
  `id_registro` int NOT NULL AUTO_INCREMENT,
  `usuario` varchar(100) NOT NULL,
  `clave` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `rol` varchar(50) DEFAULT '1',
  `fcreacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `fmodificacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `intentoslogin` int DEFAULT '0',
  PRIMARY KEY (`id_registro`),
  KEY `tregistros_fk_idx` (`rol`),
  CONSTRAINT `tregistros_fk` FOREIGN KEY (`rol`) REFERENCES `troles` (`rol`)
) ENGINE=InnoDB AUTO_INCREMENT=81 DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Elaborado por: Autor.

Figura 16: Elaboración de tabla ROLES.

```sql
DROP TABLE IF EXISTS `troles`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `troles` (
  `id_rol` int NOT NULL AUTO_INCREMENT,
  `descripcion` varchar(45) NOT NULL,
  `rol` varchar(50) NOT NULL,
  PRIMARY KEY (`id_rol`),
  UNIQUE KEY `uk_rol` (`rol`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Elaborado por: Autor.

Figura 17: Elaboración de tabla USUARIOS.

```sql
DROP TABLE IF EXISTS `tusuarios`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tusuarios` (
  `id_usuario` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL,
  `apellido` varchar(100) NOT NULL,
  `genero` varchar(10) NOT NULL,
  `telefono` varchar(20) NOT NULL,
  `fechanacimiento` date NOT NULL,
  `fcreacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `fmodificacion` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `id_registro` int NOT NULL,
  `id_bar` int DEFAULT NULL,
  PRIMARY KEY (`id_usuario`),
  KEY `tusuarios_irfk_1_idx` (`id_registro`),
  KEY `tusuarios_fk_2_idx` (`id_bar`),
  CONSTRAINT `tusuarios_irfk_1` FOREIGN KEY (`id_registro`) REFERENCES `tregistros` (`id_registro`)
) ENGINE=InnoDB AUTO_INCREMENT=77 DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Elaborado por: Autor.

4.6.2 Creación del Backend.

Para la generación del backend, se empleó Node.js en su versión 18.16.0, junto con el entorno de desarrollo Visual Studio Code. Al iniciar el proyecto, fue esencial tener en cuenta la incorporación de ciertas dependencias para asegurar el funcionamiento adecuado y eficiente del sistema.

Entre estas dependencias, se encuentran:

- bcrypt: Utilizado para la encriptación de información sensible, como contraseñas de usuarios, proporcionando una capa adicional de seguridad al almacenar datos confidenciales en la base de datos.
- cookie-parser: Empleado para el manejo de cookies en las solicitudes HTTP, permitiendo la gestión eficiente de la sesión de usuario y la persistencia de datos en el navegador del cliente.
- Express: Framework de Node.js ampliamente utilizado para la construcción de aplicaciones web y APIs. Express simplifica la creación de rutas, respuestas y manejo de solicitudes HTTP, middleware, entre otras tareas, permitiendo un desarrollo rápido y eficiente.

- express-jwt: Empleado para la generación de JSON Web Tokens (JWT), necesarios para la autorización y autenticación entre los diferentes roles disponibles dentro del sistema. Los JWT proporcionan un mecanismo seguro para la gestión de sesiones y la protección de rutas privadas en la aplicación.
- Multer: Utilizado para el manejo de archivos en las solicitudes HTTP, permitiendo la carga de archivos desde el cliente al servidor de manera eficiente. Multer es eficiente en aplicaciones que demandan de la manipulación de archivos multimedia, como imágenes o archivos de audio.
- mysql: Empleado para la comunicación con el repositorio de datos MySQL, facilitando realizar manipulaciones en el repositorio de datos desde el backend.

Estas dependencias desempeñan un papel crucial en el desarrollo del backend, incluyendo las herramientas para implementar funcionalidades clave como la fuerte seguridad de los datos y optimizar la comunicación entre el lado del servidor y el lado del cliente en la aplicación web.

Al abordar la configuración inicial del backend, se emprendió un análisis detallado de las clases, funciones y métodos a implementar, guiados por un patrón de diseño altamente reconocido en la industria: el Modelo Vista Controlador (MVC). Este enfoque se seleccionó debido a sus ventajas en la separación de responsabilidades y la facilitación del mantenimiento y la escalabilidad del sistema. El patrón MVC se destaca por su capacidad para organizar la lógica del sistema en 3 componentes diferentes:

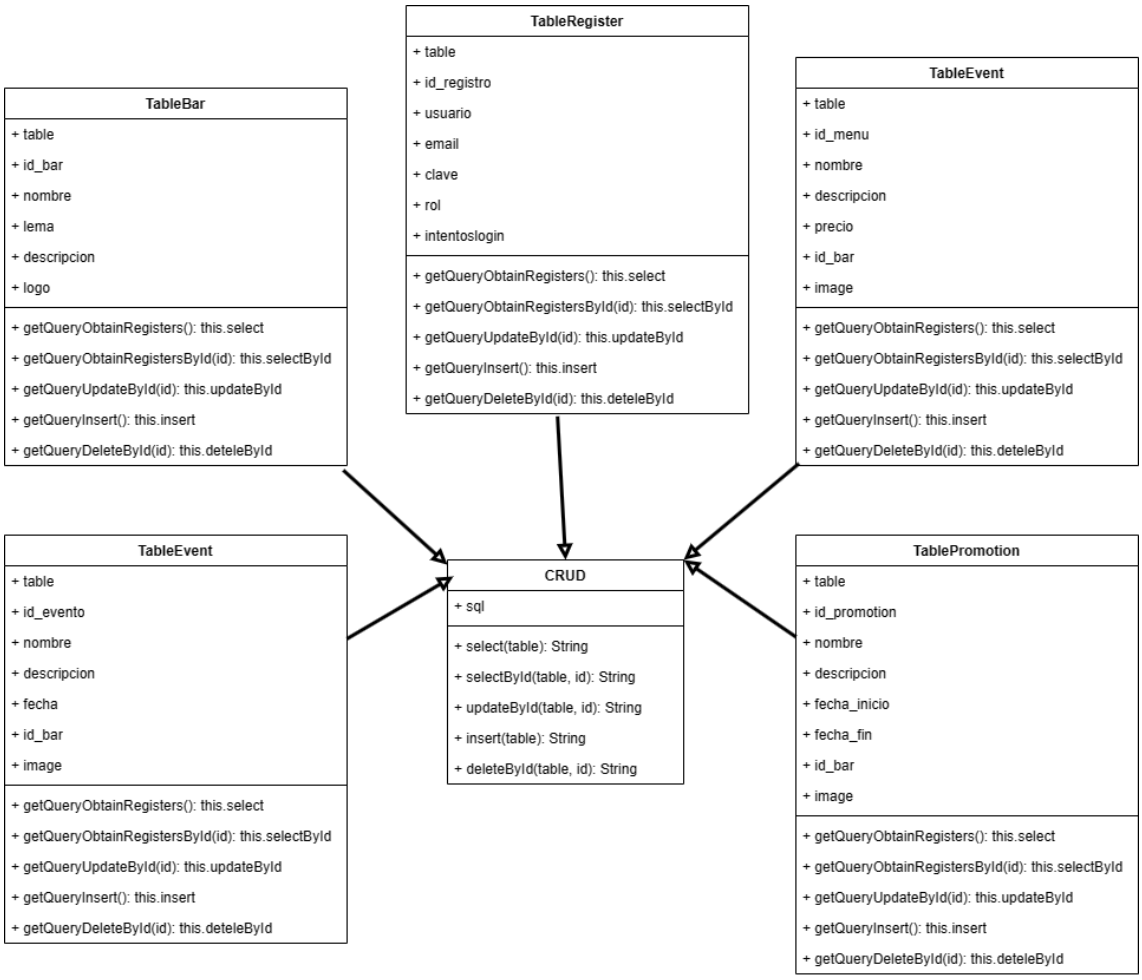
- Modelo: Es el encargado de representar los datos y la lógica empresarial de la aplicación. Aquí se define la lógica de acceso hacia datos, estructuras de datos y las operaciones relacionadas con el negocio.
- Vista: Es el responsable para presentar la información al cliente. Este componente maneja la pantalla de usuario y la representación visual de los datos, asegurando una experiencia de usuario coherente.

- Controlador: Se encuentra justo en medio del modelo y la vista, manejando las solicitudes del usuario y coordinando las acciones pertinentes. Aquí se implementa la lógica de control de la aplicación, incluida la validación de datos y la coordinación de las operaciones del modelo y la vista.

Al separar las preocupaciones relacionadas con la presentación, la lógica empresarial y el manejo de solicitudes, el sistema se vuelve más flexible y adaptable a futuros cambios y expansiones.

El resultado final de este desarrollo de diseño se plasma en un diagrama general (ver Figura 18) que ilustra la arquitectura y la organización del backend, brindando una visión panorámica de cómo interactúan los diferentes componentes y cómo se distribuye la funcionalidad en el sistema. Este diagrama sirve como guía para la implementación subsiguiente, afianzando una estructura coherente y bien diseñada para el manejo de lectura, actualización, inserción y eliminación de datos.

Figura 18: Herencia de métodos del CRUD a todas las clases para manejos SQL.



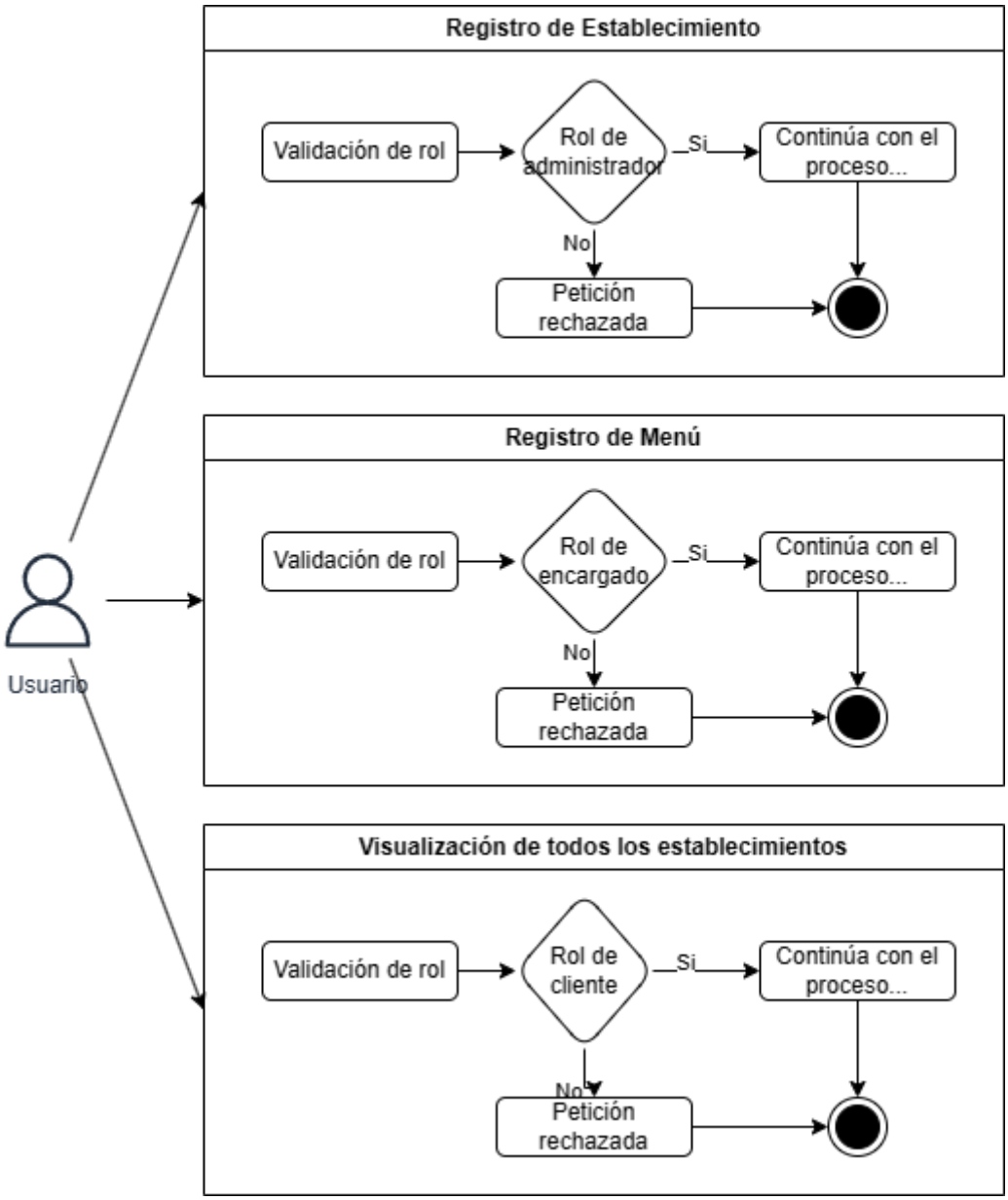
Elaborado por: Autor.

Además, se requirió de la creación de funciones adicionales para abordar aspectos específicos del manejo de la aplicación, como la gestión de roles de usuario. Esta funcionalidad resulta fundamental para garantizar un adecuado control de acceso y seguridad dentro del sistema. La administración de roles permite definir qué acciones y recursos están disponibles para cada tipo de usuario, como administradores, empleados o clientes.

La implementación de esta característica implicó el diseño y desarrollo de mecanismos para asignar, modificar y revocar roles de manera eficiente y segura. Estas funciones adicionales se integran estrechamente con el patrón de diseño MVC mencionado anteriormente, asegurando una coherencia en la arquitectura y una separación clara de responsabilidades.

La Figura 19 muestra un esquema representativo de cómo se estructuran y gestionan los roles dentro del sistema; en ella se pueden identificar las diferentes entidades involucradas, como usuarios, roles y permisos, así como las relaciones y acciones asociadas a cada una de ellas.

Figura 19: Manejo de autorizaciones de acuerdo al rol.



Elaborado por: Autor.

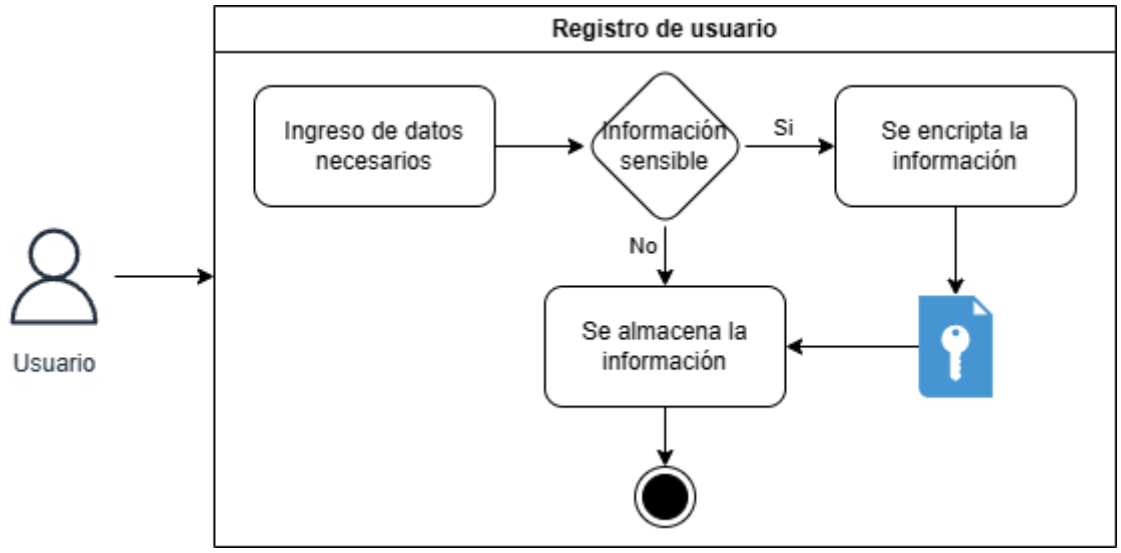
Del mismo modo, para potenciar la seguridad de los datos del usuario, se implementó un sistema de encriptación utilizando bcrypt. Esta medida es

fundamental para proteger datos confidenciales, como contraseñas, contra posibles ataques de ingeniería inversa o intrusión no autorizada.

La encriptación con bcrypt ofrece una fuerte capa extra de integridad y seguridad al momento de guardar las claves del cliente en el repositorio de datos. Este algoritmo de encriptación es reconocido por su robustez y resistencia a técnicas de descifrado, lo que brinda una mayor tranquilidad tanto a los usuarios como a los administradores del sistema.

La Figura 20 representa visualmente el proceso de encriptación implementado con bcrypt; en ella se ilustran los pasos involucrados en el cifrado de contraseñas, desde la entrada del texto plano hasta la generación de un hash encriptado. Esta visualización ayuda a comprender el flujo de datos y las transformaciones realizadas durante la fase de encriptación, lo que mejora la auditoría y la comprensión de la seguridad implementada en el sistema.

Figura 20: Encriptación de información sensible.



Elaborado por: Autor.

4.6.3 Creación del frontend.

Para el desarrollo del frontend, se optó por utilizar la versión 16.0.6 de Angular CLI, una herramienta robusta y ampliamente utilizada en la creación de aplicaciones web modernas. Esta elección se fundamentó en la flexibilidad y la potencia que Angular proporciona para la implementación de pantallas de usuario dinámicas y altamente interactivas.

El entorno de desarrollo seleccionado fue el editor de código Visual Studio Code, altamente versátil y popular entre los desarrolladores por su amplia gama de extensiones y su integración fluida con herramientas de desarrollo modernas.

Durante el proceso de desarrollo del frontend, fue esencial la integración de ciertas dependencias para asegurar el óptimo funcionamiento de la aplicación; entre las principales dependencias utilizadas se encuentran:

- Forms: Esta dependencia fue fundamental para la gestión y validación de formularios en la aplicación, permitiendo una experiencia de usuario fluida y segura en la interacción con los distintos elementos de entrada de datos.
- router: La implementación del enrutamiento fue posible gracias a esta dependencia, que facilitó la navegación entre las diferentes vistas y componentes de la aplicación, garantizando una experiencia de usuario cohesiva y fácil de seguir.
- jquery: Se utilizó jQuery para simplificar la manipulación del DOM y la interacción con elementos HTML en la aplicación, proporcionando una sintaxis concisa y poderosa para realizar operaciones comunes de frontend.
- rxjs: La utilización de Reactive Extensions for JavaScript (RxJS) permitió gestionar de manera eficiente la asincronía en la aplicación, simplificando el desarrollo reactivo y la administración de flujos de datos complejos en tiempo real.
- tslib: Esta dependencia fue necesaria para aprovechar las funcionalidades de TypeScript de manera óptima, proporcionando soporte para

características específicas del lenguaje y mejorando la eficiencia y legibilidad del código.

- zone.js: La integración de zone.js resultó fundamental para la detección de cambios en la aplicación y la actualización dinámica de la pantalla de usuario en respuesta de acciones y eventos del usuario, garantizando una experiencia interactiva y receptiva.

Para la creación del frontend, se usó una arquitectura basada en componentes, lo cual permitió la creación de los elementos fundamentales de la aplicación. Cada componente encapsulaba tanto la lógica como la presentación relacionada con una parte específica de la pantalla de usuario. Estos componentes son capaces de comunicarse entre sí mediante entradas y salidas, y pueden contener tanto lógica de presentación, como de negocio, incluyendo manipulación de HTML, CSS y TypeScript.

Además de los componentes, se utilizaron los servicios proporcionados por Angular. Estos servicios fueron empleados para recuperar datos del backend, así como para mantener una lógica compartida y funcionalidades entre diferentes partes del ecosistema de la aplicación.

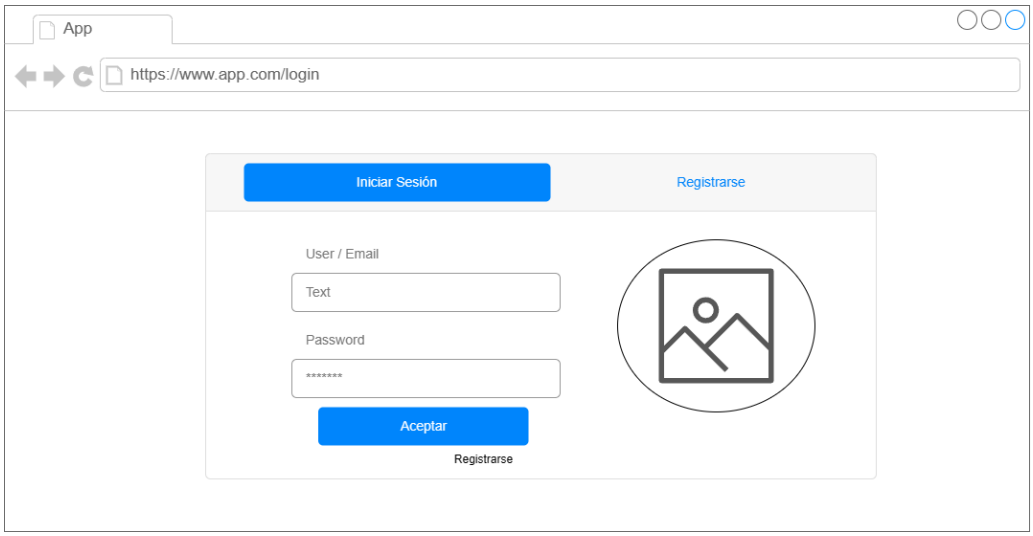
Para la navegación entre vistas y componentes de la aplicación, se emplearon las rutas definidas con el enrutador de Angular. Estas rutas fueron configuradas mediante un mecanismo basado en árbol, lo que permitía una navegación jerárquica dentro de la aplicación.

Finalmente, se implementó un nivel adicional de seguridad en el frontend utilizando el servicio AuthGuard, el cual implementa la interfaz CanActivate de Angular. Esta medida permitió controlar el acceso a las rutas dentro de la aplicación en función de ciertas condiciones. Este sistema de seguridad fue configurado a nivel de raíz, lo que significa que estaba disponible de manera global en toda la aplicación. Dentro del método canActivate, se verifica la presencia de un token de autenticación en la sesión del navegador. Dependiendo de la configuración del token, los usuarios

tienen acceso a ciertas rutas o vistas, mientras que su acceso será limitado para otras rutas no asignadas a su perfil o rol.

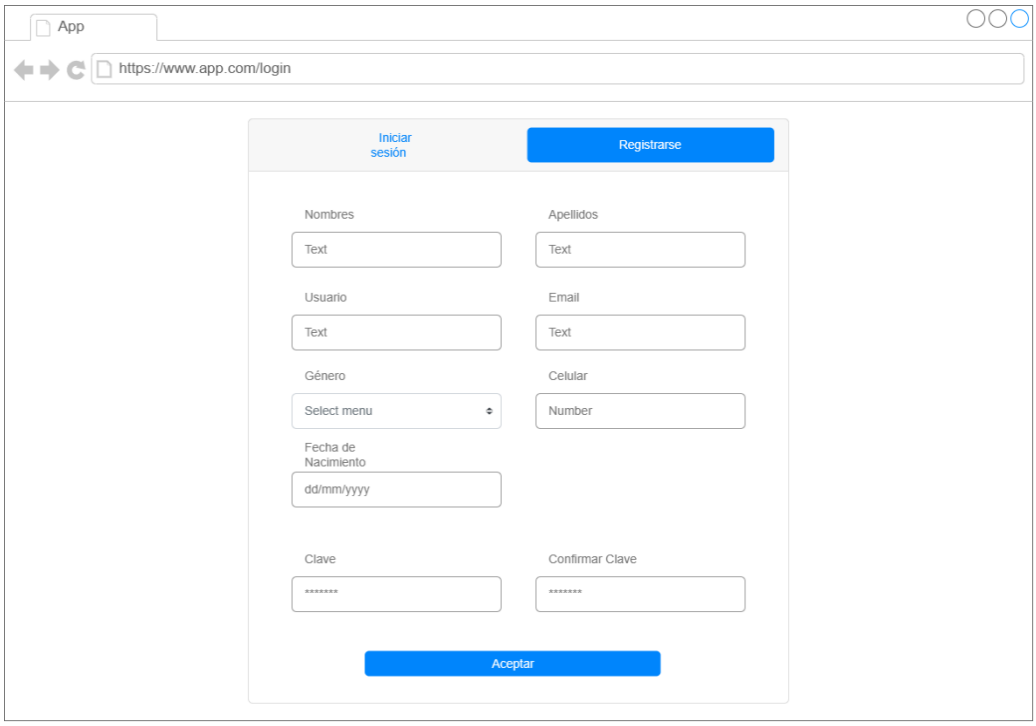
Continuando con la implementación del frontend, se llevó a cabo la creación de diseños que capturan el funcionamiento del ecosistema funcional, detallando las acciones principales para una comunicación efectiva entre vistas y funcionalidades. Estos diseños se presentan en la Figura 21, Figura 22, Figura 23, Figura 24, Figura 25, Figura 26, Figura 27 y Figura 28.

Figura 21: Pantalla de inicio de sesión.



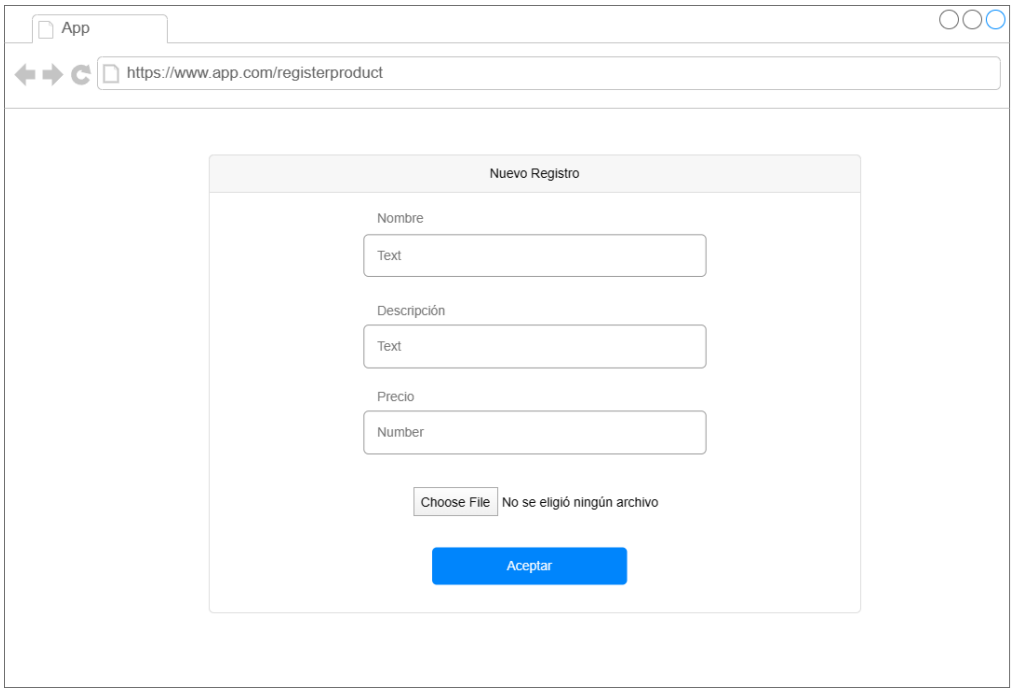
Elaborado por: Autor.

Figura 22: Pantalla de Registro.



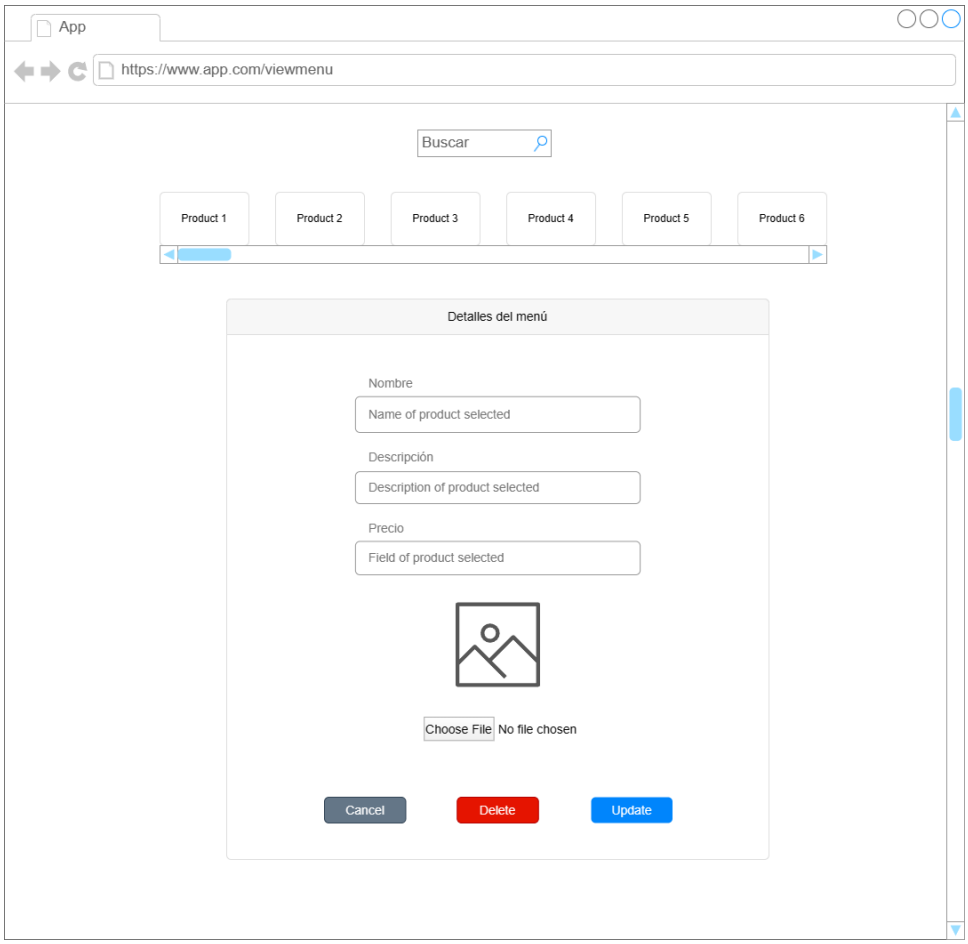
Elaborado por: Autor.

Figura 23: Interfaz para registrar producto.



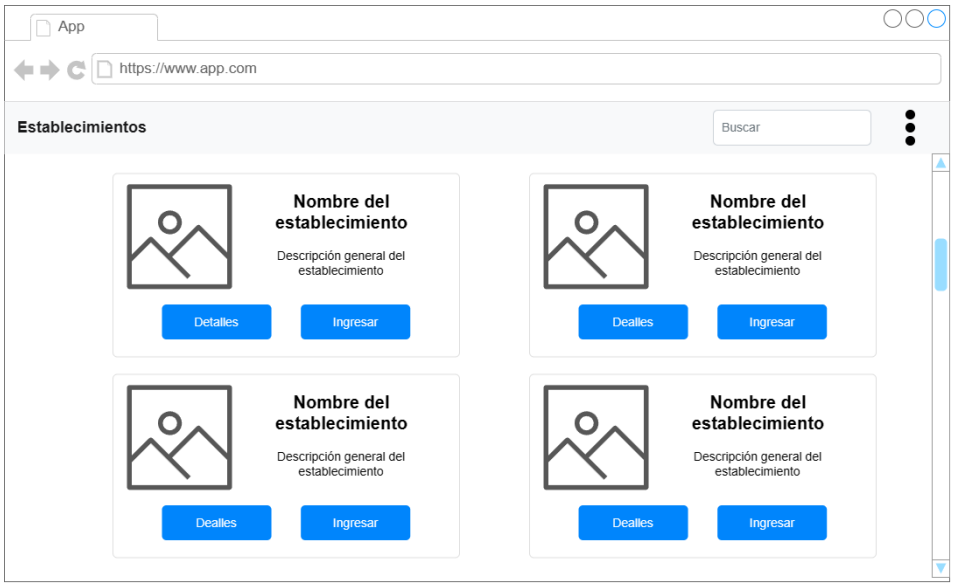
Elaborado por: Autor.

Figura 24: Interfaz de detalles y modificación del producto.



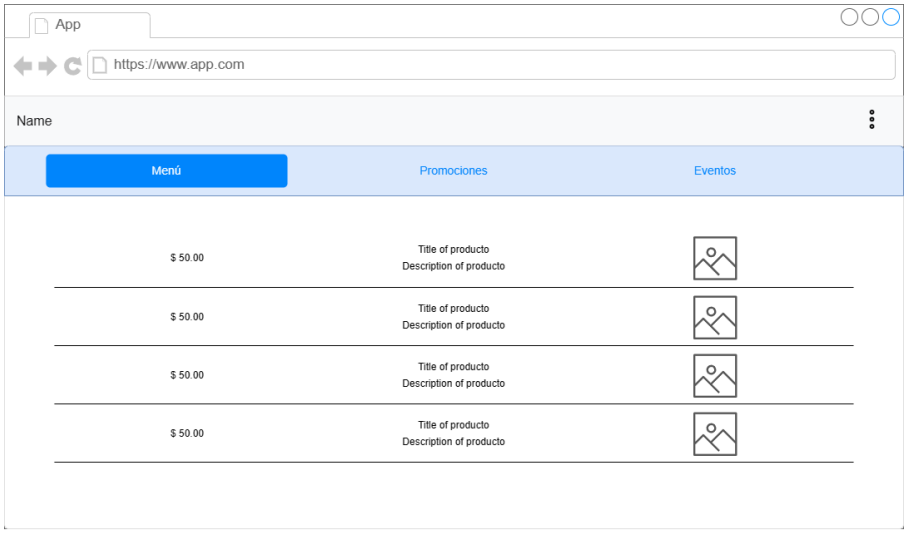
Elaborado por: Autor.

Figura 25: Interfaz inicial.



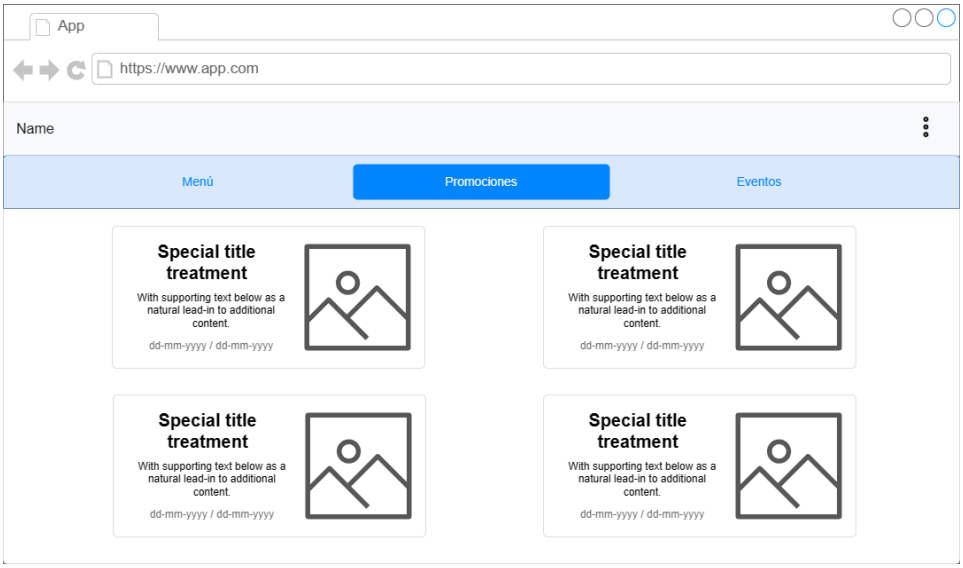
Elaborado por: Autor.

Figura 26: Elaboración de menú.



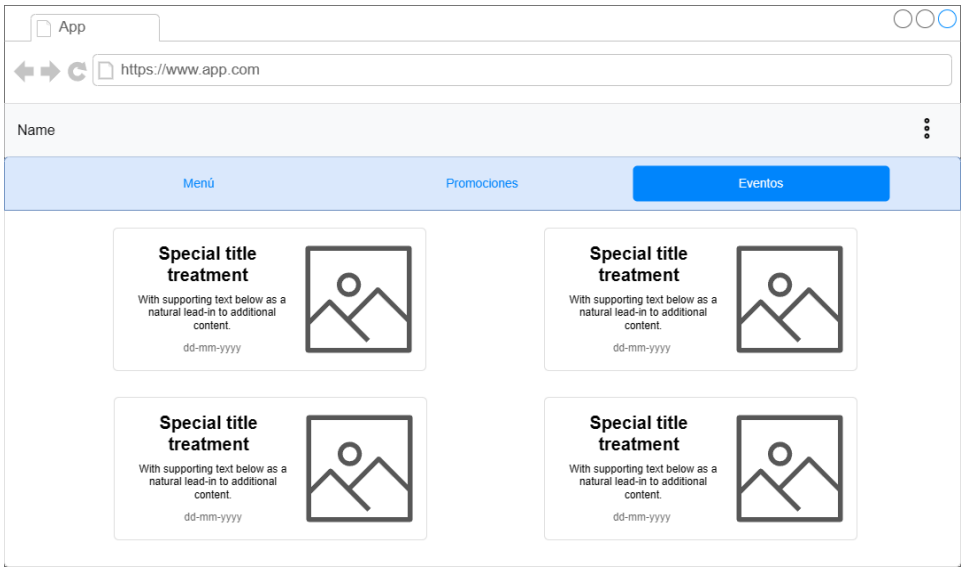
Elaborado por: Autor.

Figura 27: Elaboración de promociones.



Elaborado por: Autor.

Figura 28: Elaboración de eventos.



Elaborado por: Autor.

La creación de diseños previos a la implementación resultó de gran ayuda, ya que permitió anticipar posibles fallos de comunicación entre vistas y funcionalidades. Gracias a estas interfaces, se logró obtener una visión más amplia de los requisitos de implementación, abarcando y considerando los escenarios fundamentales para garantizar el correcto funcionamiento del ecosistema funcional.

4.6.4 Dockenizar el proyecto.

La segmentación del proyecto en diferentes partes del ecosistema implica la necesidad de satisfacer requisitos específicos para su ejecución. Sin embargo, configurar estos requisitos puede resultar complicado y propenso a errores. Para abordar esta problemática, se optó por la dockerización del proyecto; esta decisión permitirá simplificar considerablemente el despliegue y mantenimiento del sistema, al encapsular cada componente en contenedores independientes. Con la implementación de Docker, se garantiza una ejecución más fluida y sin las complicaciones derivadas de la gestión de múltiples versiones incompatibles.

Para que los contenedores funcionen correctamente y puedan comunicarse entre sí, es necesario realizar ciertos ajustes en la configuración. En la Figura 29 se detalla la configuración necesaria para el archivo Dockerfile en la carpeta del backend.

Figura 29: Dockerfile del backend.

```
# Establece la imagen base a utilizar, en este caso, Node.js en su versión 18.16.0.
FROM node:18.16.0

# Establece el directorio de trabajo dentro del contenedor en /app.
WORKDIR /app

# Copia el archivo package.json del directorio local al directorio de trabajo del contenedor.
COPY package.json .

# Ejecuta el comando npm install dentro del contenedor para instalar las dependencias especificadas en package.json.
RUN npm install

# Copia todos los archivos y carpetas del directorio local al directorio de trabajo del contenedor.
COPY . .

# Define el comando por defecto a ejecutar cuando se inicie un contenedor basado en esta imagen, en este caso, npm start.
CMD npm start
```

Elaborado por: Autor.

Por otro lado, para el lado del frontend, se requiere ubicar la configuración en un nuevo archivo ubicado en la carpeta raíz del proyecto, el contenido del archivo se detalla en la Figura 30.

Figura 30: Dockerfile del frontend.

```
#Define una etapa de construcción llamada BUILD y establece la imagen base como Node.js en su versión 18.16.0.
FROM node:18.16.0 AS BUILD

# Establece el directorio de trabajo dentro de la etapa de construcción en /usr/src/app.
WORKDIR /usr/src/app

# Copia los archivos package.json y package-lock.json (si existen) del directorio
# local al directorio de trabajo del contenedor.
COPY package*.json ./

# Instala las dependencias especificadas en los archivos package.json y package-lock.json.
RUN npm install

# Copia todos los archivos y carpetas del directorio local al directorio de trabajo del contenedor.
COPY . .

# Ejecuta el comando npm run build --prod dentro del contenedor para compilar
# la aplicación con opciones de producción.
RUN npm run build --prod

# Etapa de producción: Define una nueva etapa utilizando la imagen base de nginx.
FROM nginx

# Copia los archivos generados en la etapa de construcción desde /usr/src/app/dist/catalogo-web-front
# al directorio donde Nginx sirve archivos estáticos por defecto.
COPY --from=BUILD /usr/src/app/dist/catalogo-web-front /usr/share/nginx/html

# Expone el puerto 80 para que Nginx pueda recibir tráfico web.
EXPOSE 80
```

Elaborado por: Autor.

Finalmente, es crucial establecer correctamente la conexión con el repositorio de datos. Esta información se especifica directamente en el archivo docker-compose.yml, ubicado en la raíz del proyecto. La configuración de dicho archivo se describe en la Figura 31.

Figura 31: Docker compose yml del proyecto.

```yaml
version: '3'

services:
  db:
    image: mysql:8.3.0
    environment:
      MYSQL_ROOT_PASSWORD: 123456
      MYSQL_DATABASE: catalogowebbdd
    ports:
      - "3307:3306"
    volumes:
      - /tmp/mysql-data:/var/lib/mysql
      - ./bdd.sql:/docker-entrypoint-initdb.d/script.sql

  backend:
    build: ./catalogo-web-bares-back-end
    depends_on:
      - db
    links:
      - db
    ports:
      - "3000:3000"

  frontend:
    build: ./catalogo-web-bares-front-end
    depends_on:
      - backend
    ports:
      - "4200:80"
```

Elaborado por: Autor.

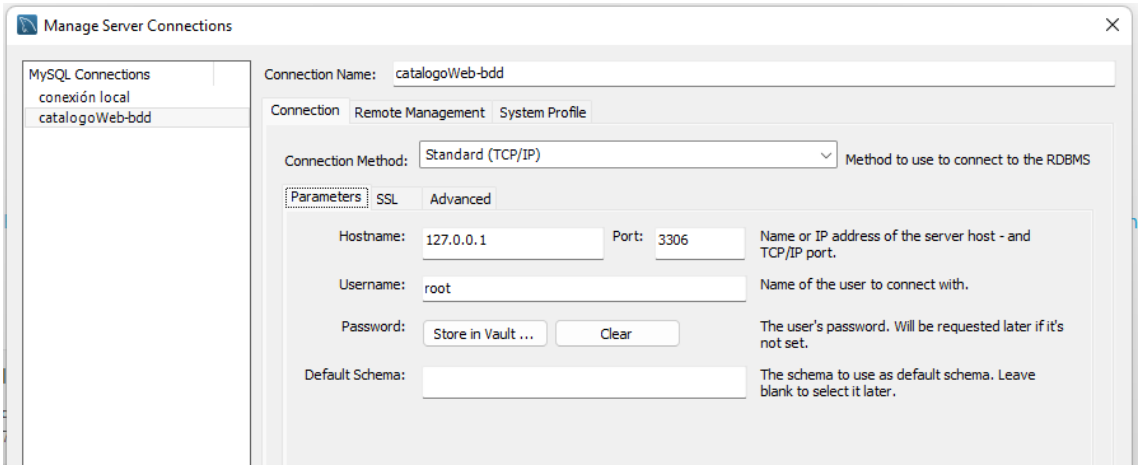
En la línea 13 del archivo docker-compose.yml, se ejecuta un script que contiene toda la información base para la creación de tablas, campos y registros en el ecosistema. Este archivo SQL debe colocarse al mismo nivel que el archivo docker-compose.yml.

5. Resultados y discusión

5.1 Efectos de la integración con la base de datos.

La configuración de la conexión a la base de datos (ver Figura 32) se llevó a cabo localmente, siguiendo los procedimientos detallados en la imagen adjunta. Este paso fue crucial para establecer una comunicación efectiva entre el ecosistema y el repositorio de datos, asegurando así que el sistema pudiera ingresar y modificar los datos de manera eficiente.

Figura 32: Configuración de MySQL WORKBENCH



Elaborado por: Autor.

A continuación, se presenta el resultado de la elaboración o generación de las tablas en el repositorio de datos. La Figura 33, Figura 34, Figura 35, Figura 36, Figura 37, Figura 38 y Figura 39 muestran las tablas recién creadas, junto con sus respectivas columnas y configuraciones. Este paso es importante para el desarrollo del sistema, proporcionando la estructura fundamental sobre la cual se almacenarán y gestionarán los datos relacionados con el sistema.

Figura 33: Tabla BAR.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_bar	INT	☑	☑	☐	☐	☐	☐	☑	☐	
nombre	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
lema	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
descripcion	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
logo	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
fcreacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
fmodificacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

Elaborado por: Autor.

Figura 34: Tabla EVENTOS.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_evento	INT	☑	☑	☐	☐	☐	☐	☑	☐	
nombre	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
descripcion	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
fecha	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
fcreacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
fmodificacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
id_bar	INT	☐	☑	☐	☐	☐	☐	☐	☐	
image	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	

Elaborado por: Autor.

Figura 35: Tabla MENU.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_menu	INT	☑	☑	☐	☐	☐	☐	☑	☐	
nombre	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
descripcion	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
precio	DECIMAL(10,2)	☐	☑	☐	☐	☐	☐	☐	☐	
fcreacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
fmodificacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
id_bar	INT	☐	☑	☐	☐	☐	☐	☐	☐	
image	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Elaborado por: Autor.

Figura 36: Tabla PROMOCIONES.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_promocion	INT	☑	☑	☐	☐	☐	☐	☑	☐	
nombre	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
descripcion	VARCHAR(225)	☐	☑	☐	☐	☐	☐	☐	☐	
fecha_inicio	TIMESTAMP(6)	☐	☑	☐	☐	☐	☐	☐	☐	
fecha_fin	TIMESTAMP(6)	☐	☑	☐	☐	☐	☐	☐	☐	
id_bar	INT	☐	☑	☐	☐	☐	☐	☐	☐	
fcreacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
fmodificacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
image	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Elaborado por: Autor.

Figura 37: Tabla REGISTROS.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_registro	INT	☑	☑	☐	☐	☐	☐	☑	☐	
usuario	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
clave	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
email	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
rol	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	☐	'1'
fcreacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
fmodificacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
intentoslogin	INT	☐	☐	☐	☐	☐	☐	☐	☐	'0'

Elaborado por: Autor.

Figura 38: Tabla ROLES.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_rol	INT	☑	☑	☐	☐	☐	☐	☑	☐	
descripcion	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
rol	VARCHAR(50)	☐	☑	☑	☐	☐	☐	☐	☐	

Elaborado por: Autor.

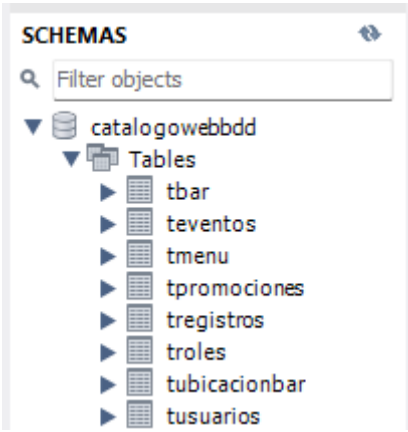
Figura 39: Tabla USUARIOS.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_usuario	INT	☑	☑	☐	☐	☐	☐	☑	☐	
nombre	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
apellido	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
genero	VARCHAR(10)	☐	☑	☐	☐	☐	☐	☐	☐	
telefono	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
fechanacimiento	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
fcreacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
fmodificacion	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
id_registro	INT	☐	☑	☐	☐	☐	☐	☐	☐	
id_bar	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Elaborado por: Autor.

La creación de las tablas en la base de datos fue fundamental en la construcción del sistema integral y funcional para la gestión de registros en el ecosistema. La Figura 40 ofrece una visión general de cuál fue el resultado de estas creaciones.

Figura 40: Esquema general de las tablas en MySQL Workbench.



Elaborado por: Autor.

Para completar la integración, se estableció la conexión entre el servidor desarrollado en Node.js y el repositorio de datos. La Figura 41 y Figura 42 muestran el resultado de la conexión, evidenciando la correcta vinculación entre ambos componentes. Este paso es crucial para garantizar que el backend pueda acceder y gestionar los datos de manera eficiente. Dichas manipulaciones se detallarán en la siguiente sección de Resultados de la integración con backend.

Figura 41: Configuración en el backend para la conexión con MySQL WORKBENCH.

```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: '127.0.0.1',      // Dirección del servidor MySQL
  user: 'root',       // Usuario de la base de datos
  password: '', // Contraseña del usuario
  database: 'catalogowebbdd'  // Nombre de la base de datos
});

/* The `connection.connect()` function is used to establish a connection to the MySQL database server.
It takes a callback function as a parameter, which will be executed once the connection is
established or if there is an error. */
connection.connect((err) => {
  if (err) {
    console.error('Error al conectar con MySQL:', err);
  } else {
    console.log('Conexión exitosa a la base de datos.');
  }
});

module.exports = connection;
```

Elaborado por: Autor.

Figura 42: Resultado de la conexión.

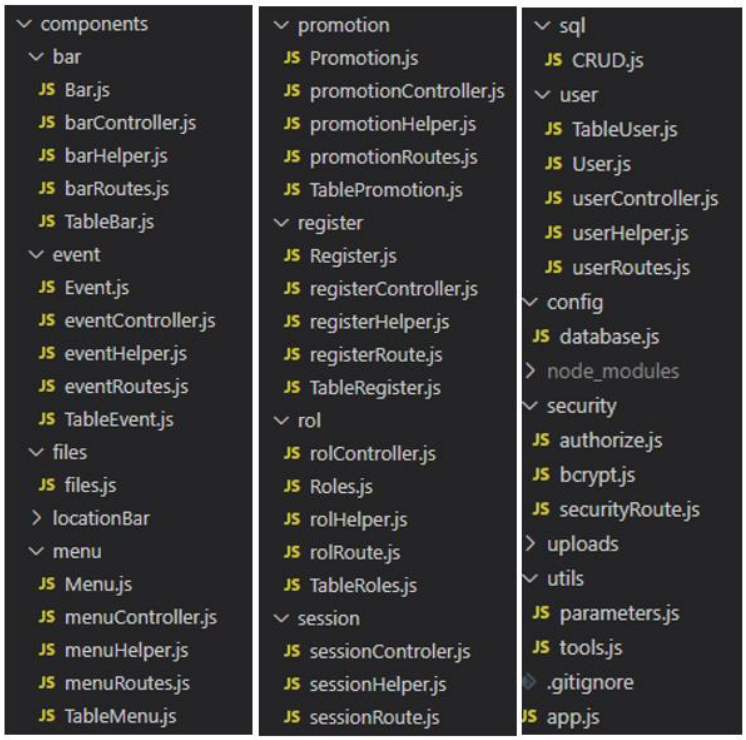
Elaborado por: Autor.

Como resultado final de la creación de tablas se puede observar en el anexo 9.1. Creación de tablas.

5.2 Resultados de la integración con backend.

En los resultados de la integración con el backend, tras haber definido todas las clases, métodos y funciones necesarias para el funcionamiento eficiente del prototipo, se logró obtener la siguiente estructura principal (ver Figura 43) dentro del entorno de desarrollo Visual Studio Code.

Figura 43: Estructura general del backend.



Elaborado por: Autor.

Después de completar la integración de todos los componentes requeridos, se avanzó con la siguiente etapa del proyecto: realizando pruebas del manejo de datos utilizando la herramienta Postman. Estas pruebas no solo incluyeron el inicio de sesión y la eliminación de productos, sino que también abarcaron una serie de escenarios representativos para validar la funcionalidad integral del sistema.

Cada prueba se diseñó para simular situaciones reales de uso, lo que implicaba no solo la ejecución de acciones básicas como el inicio de sesión, sino también la verificación de la interacción entre diferentes roles de usuario y la aplicación de los controles de seguridad correspondientes.

Por ejemplo, se realizaron pruebas para garantizar que solo los usuarios con roles específicos tuvieran acceso a determinadas funcionalidades, como la capacidad de eliminar productos o modificar ciertos aspectos de la interfaz. Esto se logró mediante la validación adecuada de los JSON Web Tokens (JWT) asociados con cada

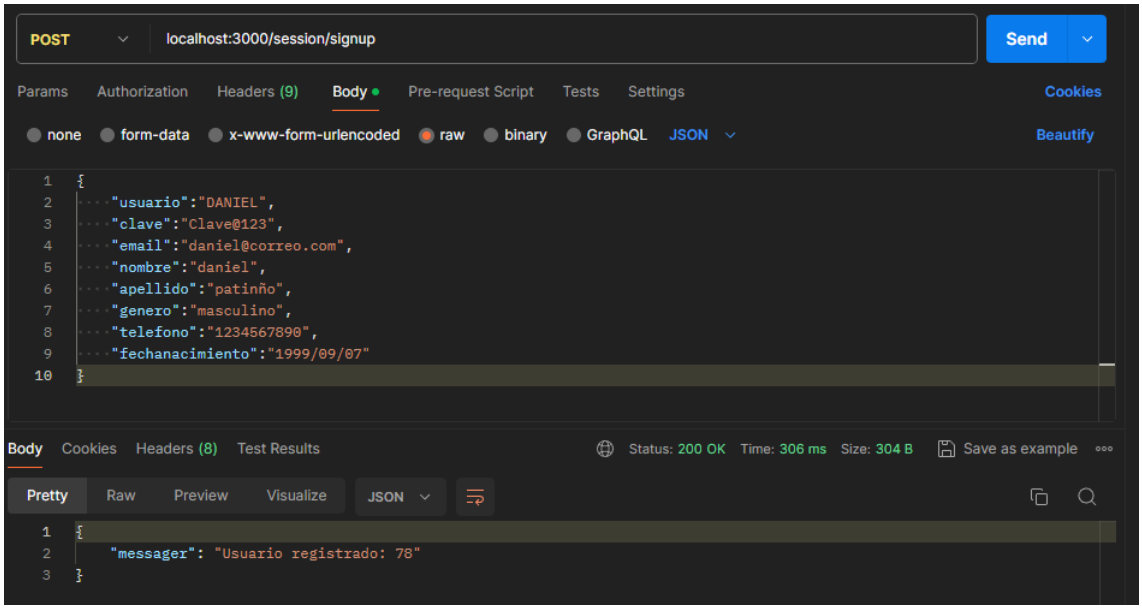
solicitud, lo que garantizaba que solo los usuarios autorizados pudieran llevar a cabo ciertas acciones.

Pruebas realizadas:

- Registro de usuario.
- Acceso del cliente al sistema.
- Inicio de sesión de un encargado del establecimiento.
- Inicio de sesión de un administrador.
- Registro de un establecimiento.
- Intento de modificar sin autorización

Cada prueba se documentó, iniciando con la prueba de registro de usuario (ver Figura 44), y los resultados se analizaron para identificar cualquier anomalía o incluso realizar mejoras. Este proceso de pruebas no solo garantizó la estabilidad y confiabilidad del sistema, sino que también proporcionó información valiosa para futuras iteraciones y mejoras en el desarrollo.

Figura 44: Prueba de registro de usuario.



Elaborado por: Autor.

En este caso particular, es fundamental destacar las medidas implementadas para salvaguardar la información sensible. Internamente, se aplican técnicas de

encriptación para mejorar la seguridad de los registros sensibles previo a ser guardados. Esto asegura que, incluso en el supuesto caso de que el repositorio de datos sea comprometido, la información permanezca inaccesible para personas no autorizadas. La implementación de esta medida refleja un compromiso con la fuerte protección y alta privacidad de los registros sensibles de usuarios, cumpliendo con estándares de seguridad y regulaciones de privacidad pertinentes. Esto fortalece la confianza de los clientes en la aplicación y a mitigar posibles riesgos relacionados con la manipulación de datos sensibles.

La información encriptada en la base de datos se refleja en la Figura 45 y Figura 46.

Figura 45: Datos de la tabla REGISTROS.

id_registro	usuario	clave	email	rol	fcreacion	fmodificacion	intentoslogin
81	DANIEL	$2b$12$xv7Fvn92HLHNYyuQ...	daniel@correo.com	1	2024-02-26 18:49:19	2024-02-26 18:49:19	0

Elaborado por: Autor.

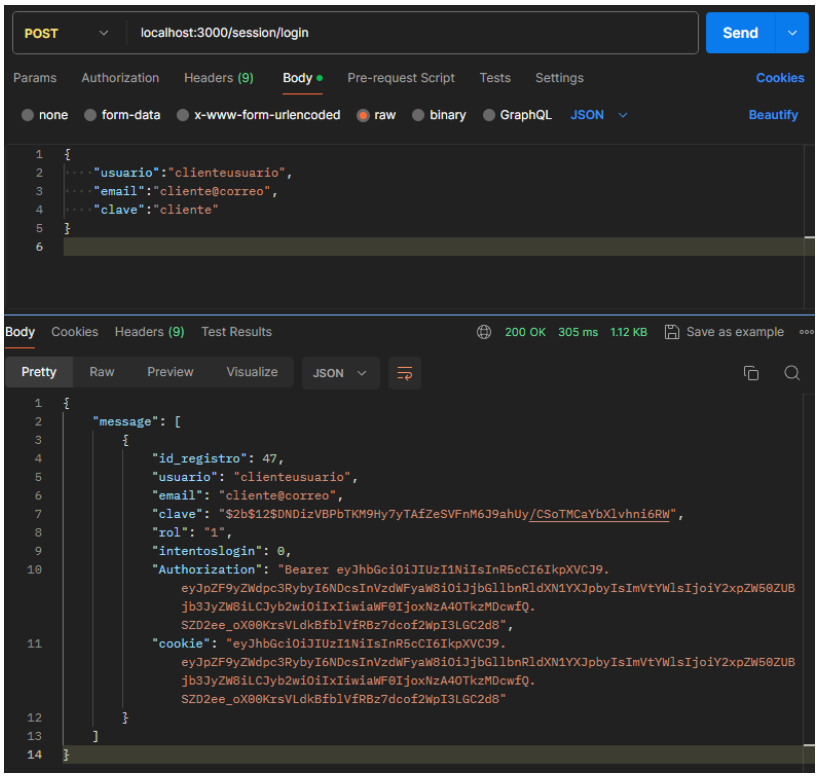
Figura 46: Datos de la tabla USUARIOS.

id_usuario	nombre	apellido	genero	telefono	fechanacimiento	fcreacion	fmodificacion	id_registro	id_bar
77	daniel	patiño	masculino	1234567890	1999-09-07	2024-02-26 18:4...	2024-02-26 18:49:19	81	NULL

Elaborado por: Autor.

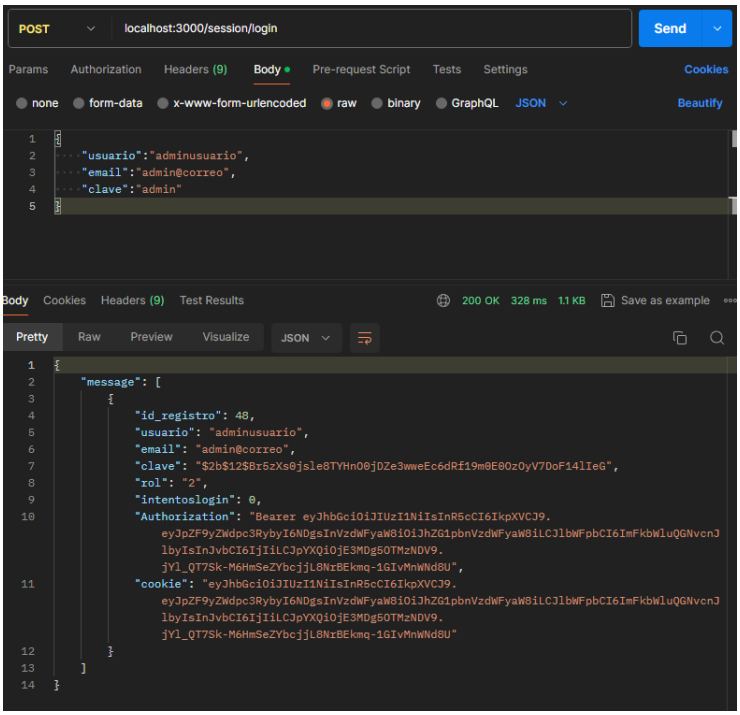
Continuando, se detalla el proceso de iniciar sesión (ver Figura 47, Figura 48 y Figura 49) de diversos usuarios en el sistema. Es importante destacar que los datos se ilustran completos con fines didácticos y de ilustración.

Figura 47: Prueba de acceso del cliente al sistema.



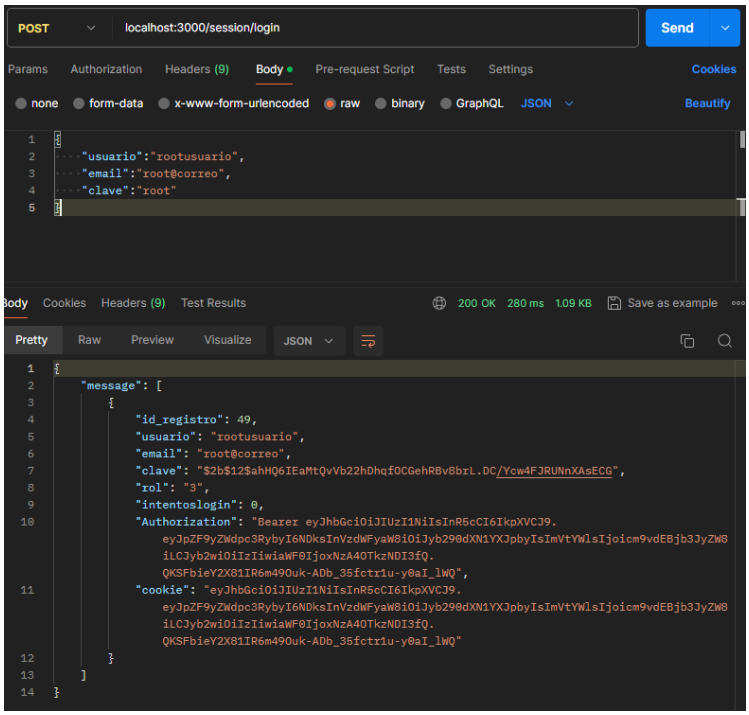
Elaborado por: Autor.

Figura 48: Prueba de inicio de sesión de un encargado del establecimiento.



Elaborado por: Autor.

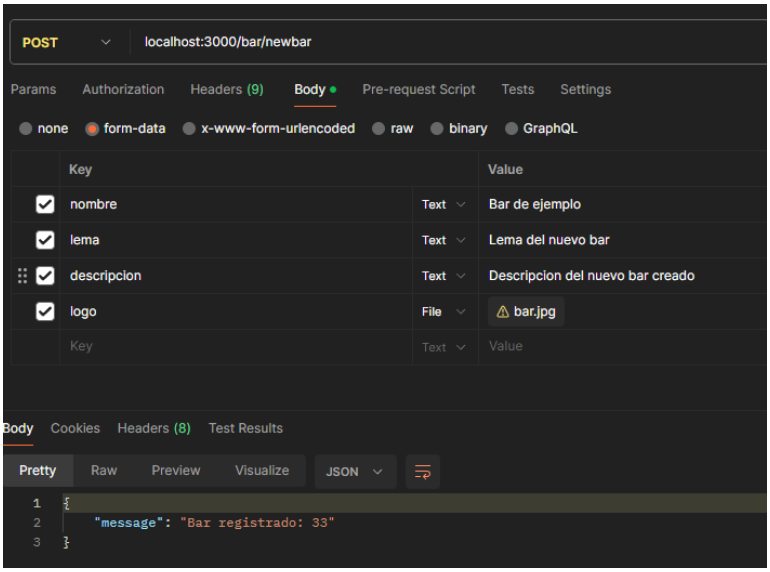
Figura 49: Prueba de inicio de sesión de un administrador.



Elaborado por: Autor.

A continuación, se presenta el proceso de registro de un bar (ver Figura 50), en el cual se destaca que se adjunta un archivo que luego se almacena en una ruta específica del servidor. En este caso particular, el archivo se guarda en la siguiente ruta: `uploads/images/logo`.

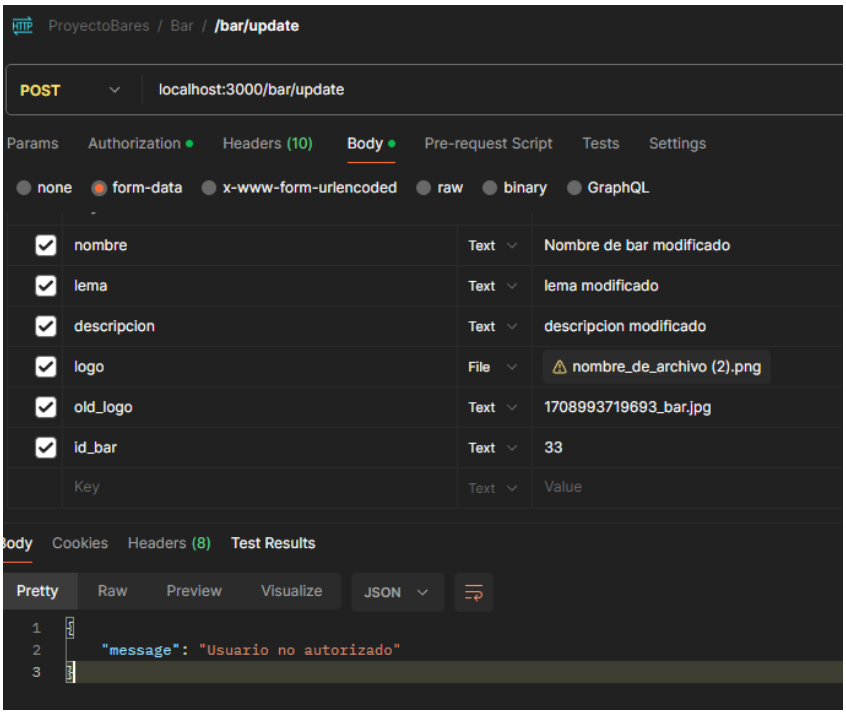
Figura 50: Prueba de registro de un establecimiento.



Elaborado por: Autor.

A continuación, en la Figura 51 se muestra un escenario en el cual un usuario intenta actualizar un registro de un establecimiento, pero su rol no corresponde al de administrador.

Figura 51: Prueba de intento de modificar sin autorización.



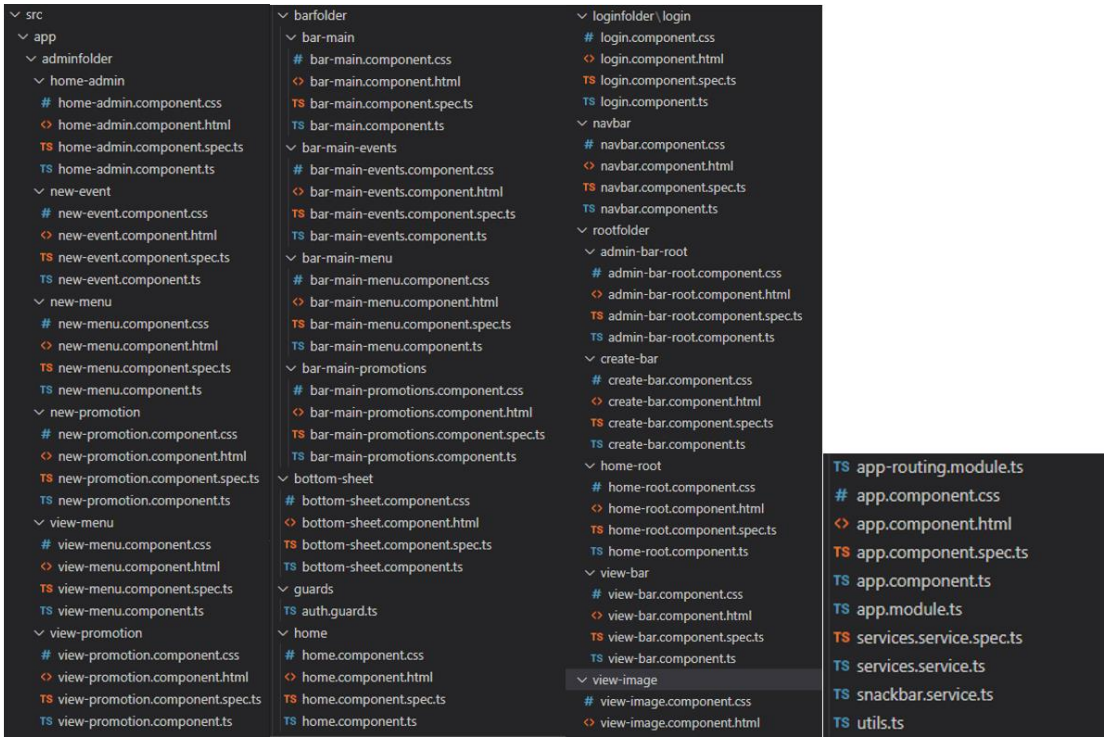
Elaborado por: Autor.

Como resultado final de la creación del backend se puede ver en el anexo 9.2. Creación del backend.

5.3 Resultados de la integración con frontend.

En los resultados de la integración con el frontend, después de identificar las interfaces principalmente funcionales, se procedió a crear cada componente en Angular (ver Figura 52).

Figura 52: Estructura general del frontend.



Elaborado por: Autor.

Para establecer la conexión con el backend y acceder a los datos alojados en MySQL, se usó la conexión HTTP Client de Angular common. Esta conexión permitió la comunicación con los diferentes endpoints expuestos por el backend.

En el servicio ServicesService de angular, se define una serie de URLs que representan los diferentes endpoints del backend. Estas URLs se organizan según las diferentes áreas funcionales de la aplicación, como los bares, las sesiones de usuario, los menús, las promociones, los eventos, entre otros.

Se usan métodos HTTP para realizar manipulaciones (CRUD) en el backend. Cada método corresponde a una acción específica, como registrar un nuevo usuario, iniciar sesión, obtener datos de una sesión, obtener información de un bar, actualizar un menú, etc.

Para garantizar la seguridad de las solicitudes, se agrega un token de autenticación al encabezado de las solicitudes HTTP, lo que permite autenticarse con el backend y acceder a los recursos protegidos.

Además, para realizar operaciones que requieren el envío de archivos, como la carga de imágenes para menús, promociones y eventos, se usa un objeto FormData para adjuntar los datos y las imágenes correspondientes a las solicitudes HTTP.

A continuación, en la Figura 53 se indica una vista general de cómo se encuentran implementados los servicios.

Figura 53: Vista general de los servicios.

```
export class ServicesService {

  constructor(private http: HttpClient) { }

  mainUrl = 'http://localhost:3000';

  //ruta principal de bar
  barUrl = `${this.mainUrl}/bar`
  urlBars = `${this.barUrl}/bars`;
  urlFindBarByIdSession = `${this.barUrl}/barbysessionid`;
  urlRegisterNewBar = `${this.barUrl}/newbar`;
  urlFindBarDataById = `${this.barUrl}/allbarbyid`;
  urlUpdateBar = `${this.barUrl}/update`;
  urlDeleteBar = `${this.barUrl}/delete`;

  //ruta prnicipal de session
  sessionUrl = `${this.mainUrl}/session`;
  urlRegister = `${this.sessionUrl}/signup`;
  urlLogin = `${this.sessionUrl}/login`;
  urlLogout = `${this.sessionUrl}/logout`;
  urlDataSession = `${this.sessionUrl}/datasession`;

  //ruta principal de menu
  menuUrl = `${this.mainUrl}/menu`;
  urlRegisterNewMenu = `${this.menuUrl}/newmenu`;
  urlFindMenuById = `${this.menuUrl}/menubybarid`;
  urlGetMenu = `${this.menuUrl}/menubyid`;
  urlUpdateMenu = `${this.menuUrl}/update`;
  urlDeleteMenu = `${this.menuUrl}/delete`;

  //ruta principal de promocion
  promotionUrl = `${this.mainUrl}/promotion`;
  urlRegisterNewPromotion = `${this.promotionUrl}/newpromotion`;
  urlFindPromotionById = `${this.promotionUrl}/promotionbybarid`;
  urlGetPromotion = `${this.promotionUrl}/promotionbyid`;
  urlUpdatePromotion = `${this.promotionUrl}/update`;
  urlDeletePromotion = `${this.promotionUrl}/delete`;
```

Elaborado por: Autor.

Continuando con la integración del frontend, se incorpora un nivel más de seguridad manejado en el AuthGuard de Angular. Este nivel de seguridad se encarga

de validar y gestionar el permiso a determinadas rutas de la aplicación en función del rol del usuario y la autenticación del mismo.

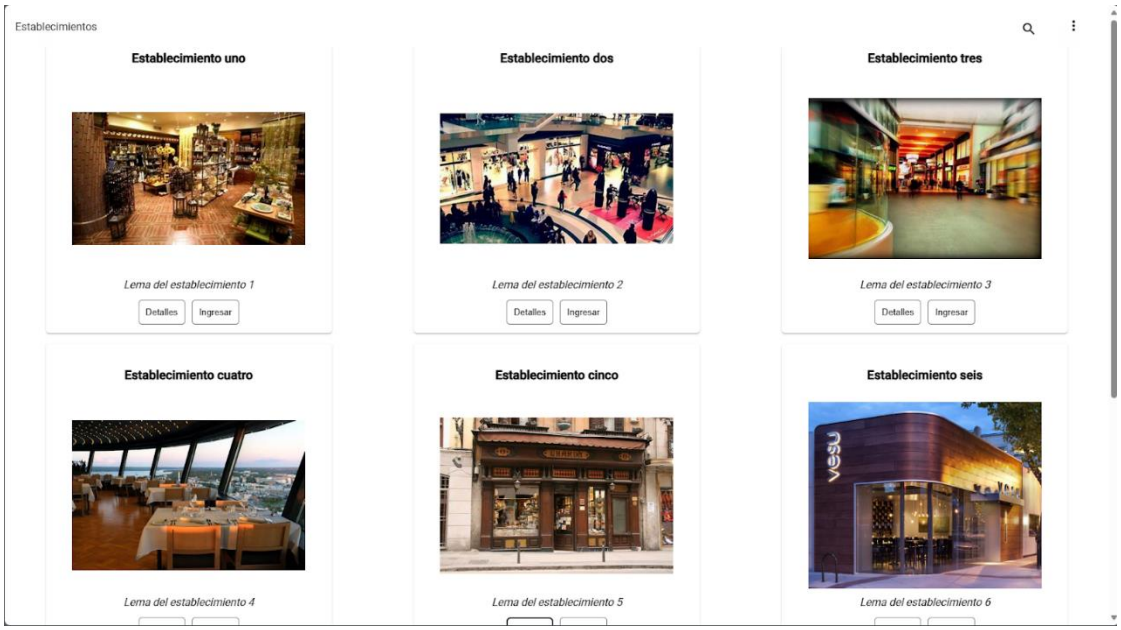
El AuthGuard implementa la interfaz CanActivate de Angular, lo que le permite interceptar las solicitudes de navegación y determinar si el usuario tiene o no el respectivo permiso que le facilita el acceso a la ruta solicitada. Para ello, primero verifica la existencia de un token de autenticación en la sesión del navegador. Si la ruta solicitada requiere autenticación y no se encuentra un token, se redirige a la página principal.

En caso de que exista un token de autenticación, el AuthGuard realiza una llamada al servicio `getCheck()` para obtener el rol del usuario actual. Una vez obtenido el rol, se evalúa mediante un switch-case para determinar si el usuario tiene acceso a la ruta solicitada según su rol. Se definen diferentes conjuntos de rutas permitidas para cada tipo de usuario (usuario, administrador y root), y se comparan con la ruta solicitada. Si el rol y la ruta coinciden, se permite el acceso, caso contrario se redirige a la página principal.

Este enfoque facilita una capa extra de seguridad a la aplicación, asegurando que solo los clientes autenticados y autorizados puedan acceder a las áreas y funcionalidades específicas de la misma.

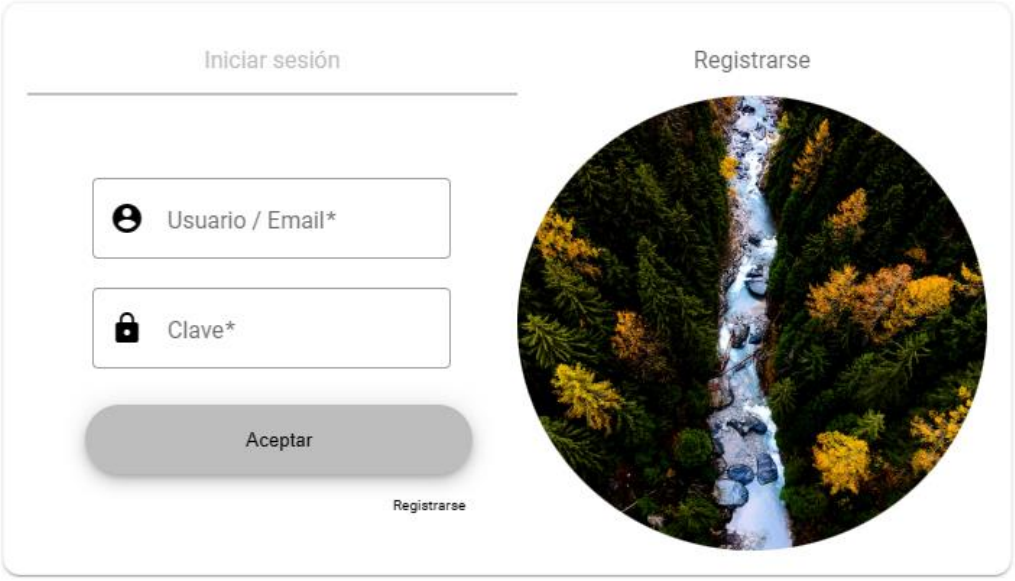
Por otro lado, de acuerdo al diseño indicado previamente en la sección de Creación del frontend., se puede observar el resultado final desde la Figura 54 hasta la Figura 60, de este modo se aprecia la interoperabilidad efectiva de frontend junto al backend y la base de datos establecida.

Figura 54: Interfaz principal.



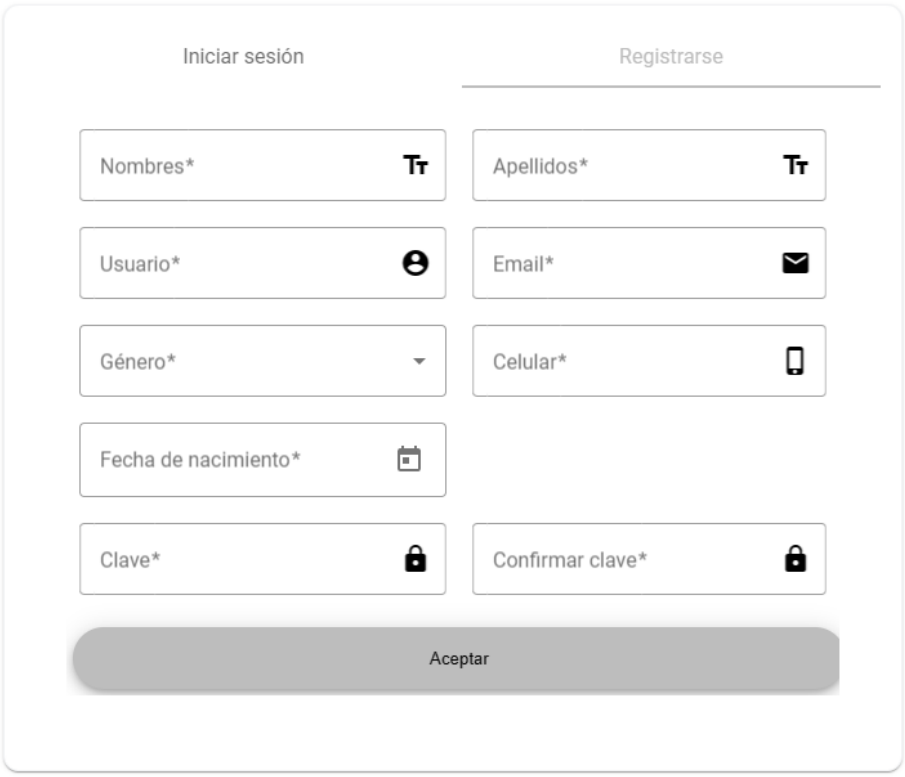
Elaborado por: Autor.

Figura 55: Interfaz de inicio se sesión.



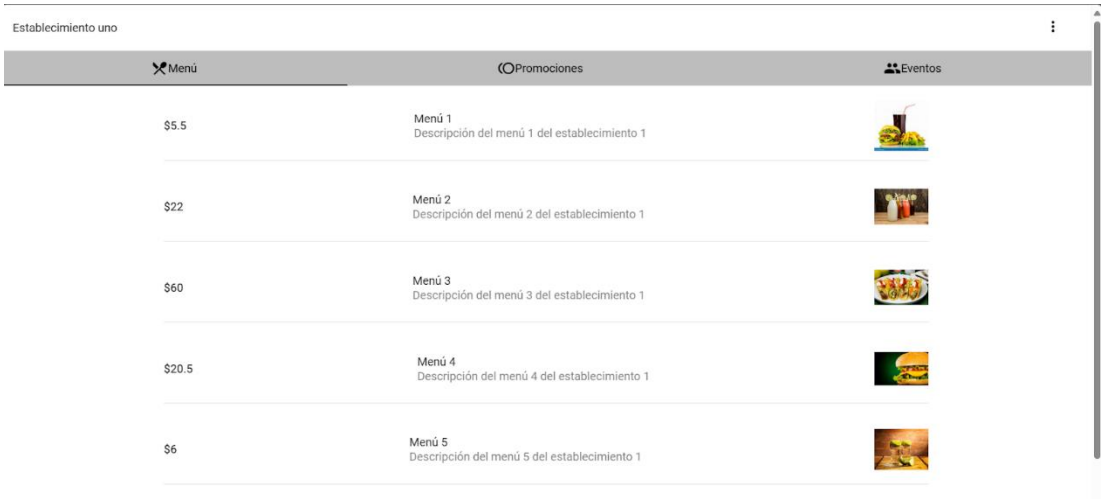
Elaborado por: Autor.

Figura 56: Interfaz de registro.



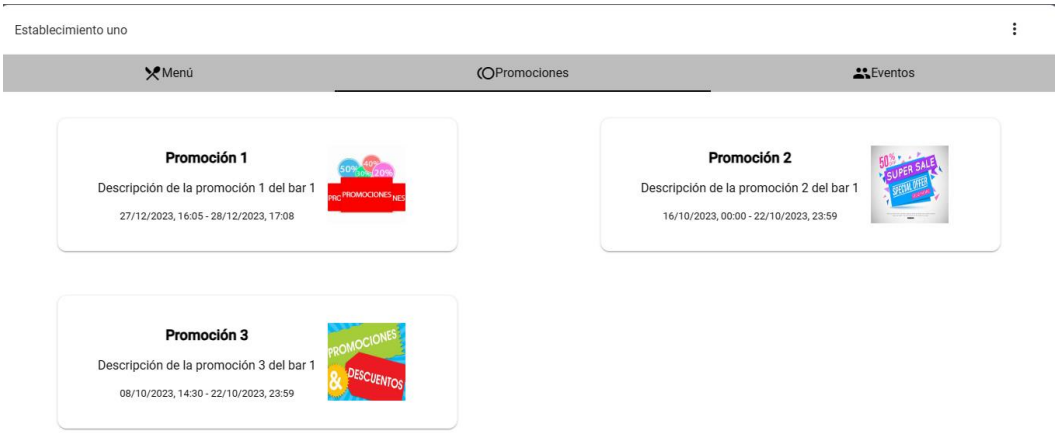
Elaborado por: Autor.

Figura 57: Interfaz de menús.



Elaborado por: Autor.

Figura 58: Interfaz de promociones.

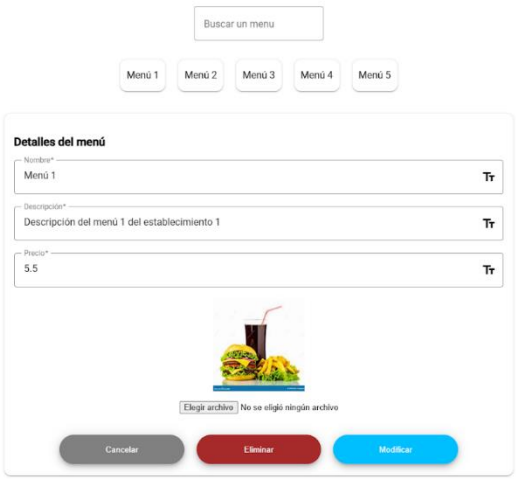


Elaborado por: Autor.

Figura 59: Interfaz de eventos.

Elaborado por: Autor.

Figura 60: Interfaz de vista y modificación de productos.



Elaborado por: Autor.

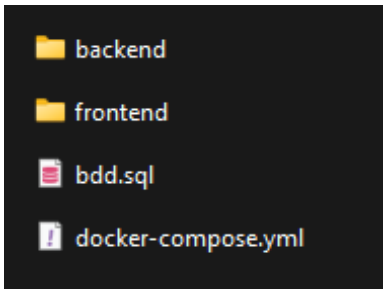
La elección de Angular para el desarrollo del frontend ha demostrado ser altamente beneficiosa en este proyecto. La estructura modular y la facilidad para crear componentes en Angular permitieron una implementación eficiente y organizada de las interfaces de usuario. La conexión con el backend a través de HTTP Client facilitó la comunicación fluida con los endpoints del servidor, garantizando una interacción segura y confiable con la base de datos. Además, la incorporación del AuthGuard proporcionó un nivel adicional de seguridad, asegurando que únicamente los clientes con permisos suficientes puedan acceder a las funcionalidades.

Como resultado final de la creación del frontend se puede ver en el anexo 9.3. Creación del frontend.

5.4 Resultados de dockenizar el proyecto.

Para trabajar de mejor forma el proyecto se crearon los archivos necesarios en la carpeta raíz en la cual se encuentra el ecosistema, la estructura general del proyecto se aprecia en la Figura 61.

Figura 61: Estructura general del ecosistema.



Elaborado por: Autor.

A continuación, se ejecuta el comando "*docker compose up*" (ver Figura 62) para que docker inicie con la construcción de las imágenes y recursos necesarios.

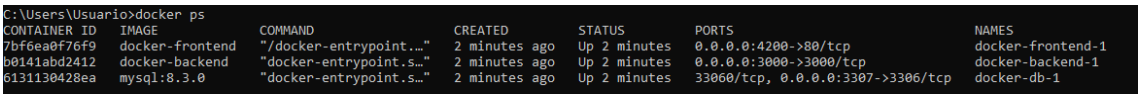
Figura 62: Ejecución del ecosistema con docker.

```
C:\Users\Usuario\Documents\UPS\Proyecto\Titulación\Hito 3\docker>docker compose up
[+] Running 6/11
 - db 10 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]  64.18MB/177.8MB Pulling
   - 9a5c778f631f Downloading [==================================>              ]  34.54MB/51.33MB
   ✔ 9e77c3a95bf2 Download complete
   ✔ 8b279a2086e0 Download complete
   ✔ c8bfbcde7882 Download complete
   ✔ d35b074b68ec Download complete
   ✔ beea5014e6af Download complete
   - dc3791a61558 Downloading [===============>                                 ]  19.39MB/63.09MB
   ✔ 52f9323b9f0e Download complete
   - 7f7391eab49b Downloading [========>                                        ]  10.25MB/63.42MB
   - 8d2f04b287ee Waiting
```

Elaborado por: Autor.

Una vez iniciado todos los contenedores se puede verificar el comando "docker ps" (ver Figura 63) en una terminal nueva.

Figura 63: Comprobación de las imágenes.



Elaborado por: Autor.

Finalmente se evidencia que los servicios están activos y preparados para usar.

- Backend: localhost:3000
- Frontend: localhost:4200

La configuración de los archivos usados se encuentra disponibles en el anexo 9.4. Docker del ecosistema.

5.5 Discusión.

Durante el proceso crítico de integración del repositorio de datos, para el correcto funcionamiento, se estableció una comunicación efectiva y eficiente entre el sistema funcional y la base de datos; esto garantizó que el sistema pudiera acceder y gestionar los registros de forma segura y eficiente.

Se crearon múltiples tablas en la base de datos para respaldar las funcionalidades del sistema; entre estas se incluyen la tabla BAR, EVENTOS, MENU, PROMOCIONES, REGISTROS, ROLES y USUARIOS. Cada una de estas tablas se diseñó con sus respectivas columnas y configuraciones para proporcionar la estructura fundamental sobre la cual se almacenarán y gestionarán los datos relacionados con el sistema. La eficiente integración entre el repositorio de datos y el backend desarrollado en Node.js se estableció de manera sólida. Esto fue esencial para garantizar un acceso eficiente a los datos almacenados, lo que facilita el funcionamiento fluido del sistema.

La integración con el backend fue exitosa, con la definición completa de todas las clases, métodos y funciones necesarias para el funcionamiento eficiente del prototipo. Esto se reflejó en una estructura principal dentro del entorno de desarrollo Visual Studio Code, proporcionando una base sólida para el desarrollo continuo del sistema.

Las pruebas del manejo de datos utilizando la herramienta Postman fueron representativas, además de las pruebas básicas de inicio de sesión y eliminación de productos, se diseñaron escenarios complejos para validar la funcionalidad integral del sistema; estas pruebas incluyeron la interacción entre diferentes roles de usuario y la aplicación de controles de seguridad.

La integración con el frontend se realizó de manera eficiente, con la identificación e implementación de interfaces funcionales en Angular. Se crearon componentes para cada interfaz, proporcionando una experiencia de usuario cohesiva y amigable. Para establecer la conexión con el backend y acceder a los datos alojados en MySQL,

se utilizó la conexión HTTP Client de Angular common. Esto permitió una comunicación fluida con los diferentes endpoints expuestos por el backend, facilitando la interacción segura y confiable con el repositorio de datos. Asimismo, se implementó un nivel adicional de seguridad en el frontend mediante el AuthGuard de Angular. Este nivel de seguridad controla el acceso a determinadas rutas del ecosistema en función del rol del usuario y su autenticación, de este modo se asegura que únicamente los clientes autorizados puedan acceder a las funcionalidades correspondientes.

El diseño del frontend se reflejó en las interfaces principales, incluyendo inicio de sesión, registro, menús, promociones, eventos y gestión de productos. Estas interfaces se implementaron de acuerdo con el diseño especificado, proporcionando una experiencia de usuario coherente y atractiva.

Por último, la dockerización del proyecto se completó con éxito, con la creación de los archivos necesarios en la carpeta raíz y la ejecución de contenedores Docker para el backend y el frontend. Los servicios se levantaron correctamente y están listos para su uso, proporcionando una solución escalable y fácilmente desplegable.

6. Conclusiones

La investigación de trabajos relacionados ha revelado la sólida capacidad y las ventajas de tecnologías clave como Angular, Node.js, JWT, Bcrypt y MySQL en el panorama del desarrollo web. A través de proyectos concretos, se ha evidenciado que Angular, como un framework del lado del cliente, ofrece una estructura flexible y robusta para la creación de aplicaciones web avanzadas, mejorando significativamente la gestión y accesibilidad de la información, como se ha demostrado en la optimización de historias clínicas. Por otro lado, Node.js ha demostrado ser un entorno de ejecución del lado del servidor eficiente y escalable, facilitando la construcción de aplicaciones completas, como aquellas destinadas a la gestión de conexiones a Internet, destacando su versatilidad en la implementación de soluciones web.

Asimismo, la implementación de JWT y Bcrypt ha sido crucial para garantizar tanto la protección como autorización de datos sensibles en aplicaciones web. Estas tecnologías proporcionan un intercambio de datos ligero y una función hash resistente fortaleciendo la integridad de los sistemas. Por último, MySQL ha surgido como un sistema de gestión de bases de datos confiable y efectivo para proyectos web dinámicos, respaldando el almacenamiento y manipulación eficientes de datos.

Es así que, el desarrollo del prototipo de ecosistema logró cumplir con el objetivo de diseñar y prototipar los componentes esenciales utilizando tecnologías clave como Angular, Node.js, JWT, Bcrypt y MySQL. La selección de estas tecnologías permitió asegurar la interoperabilidad fluida entre el frontend y el backend, facilitando así la implementación de un sistema de comunicación efectivo. Mediante el uso de Angular para el frontend, se consiguió desarrollar una pantalla de usuario dinámica. Al mismo tiempo, Node.js facilitó una sólida base para la construcción del lado del servidor, asegurando una gestión eficiente de las solicitudes y respuestas HTTP. Mediante la implementación de JWT y Bcrypt, se

logró asegurar la autenticación y permisos de usuarios, lo que protege la integridad de la información sensible almacenada en el repositorio de datos MySQL. El prototipo desarrollado representa un paso significativo hacia la implementación de un sistema completo que satisfaga las necesidades de gestión y experiencia de usuario demostrando la viabilidad y eficacia de las tecnologías utilizadas en conjunto.

Finalmente, la documentación del proceso de instalación, configuración y uso del ecosistema, respaldada por la disponibilidad del proyecto en GitHub y los archivos dockerizados almacenados en la nube (Google Drive y OneDrive), representa un esfuerzo integral para fomentar la adopción y el aprovechamiento completo de sus capacidades. La completa documentación redactada en los comentarios en inglés en el código para un público de programadores internacionales, garantiza que los usuarios puedan comprender fácilmente el funcionamiento del ecosistema. Esta iniciativa no solo facilita el acceso al desarrollo, sino que también promueve una mayor colaboración y contribución por parte de la comunidad de desarrolladores, fortaleciendo así la robustez y la evolución continua del proyecto. La combinación de una documentación detallada y accesible con la disponibilidad del código fuente en plataformas de código abierto y archivos en la nube establece una base sólida para el éxito y la sostenibilidad a largo plazo del ecosistema.

A continuación, se enumeran algunas recomendaciones y limitaciones que una organización puede enfrentar al implementar el presente ecosistema.

Para robustecer la seguridad de la información, una organización puede implementar las siguientes recomendaciones:

- **Actualizar y parcheo regular:** Mantener actualizados todos los componentes del ecosistema, las bibliotecas de software, las dependencias de código abierto y las herramientas utilizadas. Esto ayuda a mitigar brechas conocidas y a evitar posibles vulnerabilidades relacionadas a la seguridad.
- **Prácticas de seguridad:** Fomentar la adopción de óptimas prácticas de seguridad durante el desarrollo de software, como la verificación de entrada

de datos, la prevención de inyección de código, el uso adecuado de JWT y Bcrypt para la autenticación y el almacenamiento seguro de contraseñas, y el manejo adecuado de sesiones y cookies para la gestión de la sesión de usuario.

- **Auditorías de seguridad:** Se lleva a cabo con el objetivo de detectar puntos críticos en el sistema, lo cual puede abarcar pruebas de penetración, análisis de vulnerabilidades y revisión de código para identificar posibles fallos de seguridad.
- **Implementación de políticas de acceso:** Implementar políticas estrictas de accesos y privilegios para los usuarios del sistema, limitando el acceso solo a las funciones y datos necesarios para realizar sus tareas.
- **Respaldo y recuperación:** Es importante contar con un plan de respaldo y recuperación de datos sólido y que sea probado regularmente. Esto ayuda a fortalecer la confianza relacionada con la disponibilidad de los datos en caso de verse comprometida la seguridad o fallos del sistema.
- **Formación y capacitación:** Recibir capacitaciones recurrentes sobre las prácticas más óptimas enfocadas a la integridad de los datos sensibles, incluyendo la identificación de posibles amenazas, el reconocimiento de correos electrónicos de phishing y el manejo adecuado de datos sensibles.
- **Monitoreo:** Se sugiere la implementación de sistemas de monitoreo de seguridad para detectar y abordar eficazmente posibles intrusiones o actividades sospechosas en el sistema. Esto podría implicar configurar alertas automáticas y coordinar la respuesta a incidentes de seguridad de manera adecuada.

Entre las limitaciones que podrían surgir al implementar estas recomendaciones se encuentran:

- **Costo:** Algunas medidas de seguridad pueden requerir inversiones significativas en tecnología, capacitación y personal especializado.

- **Accesibilidad:** Algunas soluciones de seguridad pueden ser complejas de implementar y pueden requerir conocimientos técnicos avanzados para su configuración y mantenimiento.
- **Tecnología:** La elección de tecnologías específicas puede influir en la efectividad de las medidas de seguridad, y algunas soluciones pueden no ser compatibles con el ecosistema existente.
- **Escalabilidad:** Algunas medidas de seguridad pueden ser difíciles de escalar para adaptarse al crecimiento del negocio o al aumento en el volumen de datos y usuarios.
- **Cumplimiento normativo:** Las organizaciones pueden encontrarse con restricciones relacionadas con el cumplimiento estándares de seguridad de la industria, que podrían imponer requisitos adicionales o limitaciones en la adopción de ciertas medidas de seguridad.

7. Glosario de términos.

- **SQL:** Structured Query Language (Lenguaje de Consulta Estructurada).
- **Frontend:** Desarrollo del lado del cliente.
- **Backend:** Desarrollo del lado del servidor.
- **JWT:** Json Web Token.
- **Bcrypt:** Función de hash de contraseñas y derivación de claves para contraseñas basada en el cifrado Blowfish.
- **VSCode:** Visual Studio Code.
- **Postman:** Herramienta de prueba y desarrollo de API que está diseñada para enviar solicitudes desde el lado del cliente al servidor web.

8. Referencias

Aleksi Kujala. (2023). Development of a modern full stack web application. *Turku Amk*.

Bautista-Villegas, E. (2022). Metodologías agiles XP y Scrum, empleadas para el desarrollo de páginas web, bajo MVC, con lenguaje PHP y framework Laravel. *Revista Amazonía Digital*, *1*(1), e168–e168. https://doi.org/10.55873/RAD.V1I1.168

Chohan, U. W. (2022). Web 3.0: The Future Architecture of the Internet? *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.4037693

Cincovic, J., Delcev, S., & Draskovic, D. (2019). *Architecture of web applications based on Angular Framework: A Case Study*. https://www.eventiotic.com/eventiotic/files/Papers/URL/df6b5054-816e-4bee-b983-663fb87be2cd.pdf

Conza Ccolque, J. L. (2019). Desarrollo de un sistema web utilizando angular framework y rest (Transferencia de estado representacional) para la gestión de historias electrónicas. *Universidad Peruana Unión*. https://repositorio.upeu.edu.pe/handle/20.500.12840/3295

Ersöz, B., Bilimleri, B., & Dergisi, T. (2020). Yeni Nesil Web Paradigması-Web 4.0. *Journal of Computer Science and Technologies*, *1*(2), 58–65. https://dergipark.org.tr/en/pub/bibted/issue/57253/796030

Giffary, R. S., & Ramadhani, E. (2022). Implementasi Bcrypt dengan SHA-256 pada Password Pengguna Aplikasi Golek Kost. *Jurnal Sistem Komputer Dan Informatika (JSON)*, *3*(4), 543–546. https://doi.org/10.30865/json.v3i4.4285

Grigar Dene, & O'Sullivan James. (2021). Electronic Literature as Digital Humanities. *Bloomsbury Academy*, *2*, 151–162. https://library.oapen.org/bitstream/handle/20.500.12657/58859/1/9781501363481.pdf#page=162

Hron, M., & Obwegeser, N. (2022). Why and how is Scrum being adapted in practice: A systematic review. *Journal of Systems and Software*, *183*, 111110. https://doi.org/10.1016/J.JSS.2021.111110

Huynh, K. (2020). The development of a web application : The new trend - Serverless application. *Turku Amk*. http://www.theseus.fi/handle/10024/344206

Jacome Carrasco, H. C. (2022). *Análisis de metodologías ágiles para el desarrollo de aplicaciones Web.* http://dspace.utb.edu.ec/handle/49000/13044

Janne Kinnunen. (2023). *Designing a Node.js full stack web application* [Information and Communication technology]. https://www.theseus.fi/bitstream/handle/10024/793330/Kinnunen_Janne.pdf?sequence=2

Jesús, J., Pimentel, A., Solar González, R., Nieva García, O. S., Patricia, S., & Ibarra, C. (2022). Compilador e intérprete en línea de diagramas de flujo con fines didácticos. *Revista de Investigación En Tecnologías de La Información: RITI, ISSN-e 2387-0893, Vol. 10, Nº. 20, 2022 (Ejemplar Dedicado a: Enero-Junio), Págs. 80-94*, *10*(20), 80–94. https://doi.org/10.36825/RITI.10.20.007

Mahindrakar, P., & Pujeri, U. (2020). Insights of JSON Web Token. *International Journal of Recent Technology and Engineering (IJRTE)*, *6*, 2277–3878. https://doi.org/10.35940/ijrte.F7689.038620

Molina, J. R., Zea, M. P., Contento, M. J., García, F. G., De Metodologías, C., Rolando, J., Ríos, M., Paola, M., Ordóñez, Z., José, M., Segarra, C., Gustavo, F., & Zerda, G. (2018). Comparación de metodologías en aplicaciones web. *3c Tecnología: Glosas de Innovación Aplicadas a La Pyme, ISSN-e 2254-4143, Vol. 7, Nº. 1, 2018, Págs. 1-19*, *7*(1), 1–19. https://doi.org/10.17993/3ctecno.2018.v7n1e25.1-19

Molina, J. R., Zea, M. P., Contento, M. J., García, F. G., Rolando, J., Ríos, M., Paola, M., Ordóñez, Z., José, M., Segarra, C., Gustavo, F., & Zerda, G. (2017). Estado del arte: Metodologías de desarrollo en aplicaciones web. *3c Tecnología: Glosas de Innovación Aplicadas a La Pyme, ISSN-e 2254-4143,*

Vol. 6, Nº. 3, 2017, Págs. 54-71, *6*(3), 54–71. https://doi.org/10.17993/3ctecno.2016.v6n3e23.54-71

Nath, K. (2022). Evolution of the Internet from Web 1.0 to Metaverse: The Good, The Bad and The Ugly. *Tech Rxiv*. https://doi.org/10.36227/TECHRXIV.19743676.V1

Pant, P., Rajawat, A. S., Goyal, S. B., Bedi, P., Verma, C., Raboaca, M. S., & Enescu, F. M. (2022). Authentication and Authorization in Modern Web Apps for Data Security Using Nodejs and Role of Dark Web. *Procedia Computer Science*, *215*, 781–790. https://doi.org/10.1016/J.PROCS.2022.12.080

Ramadhan, W. F., Dewi, W. N., & Nas, C. (2020). APLIKASI WEB PORTAL MANAJEMEN INFORMATIKA BERBASIS WEBSITE DENGAN MENGGUNAKAN FRAMEWORK CODEIGNITER DAN MYSQL PADA UNIVERSITAS CATUR INSAN CENDEKIA. *Jurnal Digit : Digital of Information Technology*, *10*(2), 124–135. https://doi.org/10.51920/JD.V10I2.164

Ranta, J. (n.d.). *Testing AWS hosted Restful APIs with Postman*.

Santos, D. A. B. dos, & Fundação Oswaldo Cruz. Instituto de Comunicação e Informação Científica e Tecnológica em Saúde. Rio de Janeiro, R. Brasil. (2022). *Arquitetura da Informação em ambientes informacionais digitais na Área da Saúde: revisão de escopo*. https://www.arca.fiocruz.br/handle/icict/55980

Sotnik, S., Manakov, V., & Lyashenko, V. (2023). Overview: PHP and MySQL Features for Creating Modern Web Projects. In *International Journal of Academic Information Systems Research* (Vol. 7, Issue 1, pp. 11–17). IJAISR. https://openarchive.nure.ua/handle/document/21601

Triatama, K., Savitri, A., Sintaro, S., Inra Takaendengan, M., Informasi, S., & Sam Ratulangi, U. (2023). Rancang Bangun Sistem Informasi Nilai Akhir Siswa Berbasis Web Menggunakan Extreme Programming. *Jurnal Informatika Dan Rekayasa Perangkat Lunak*, *4*(2), 135–140. https://doi.org/10.33365/JATIKA.V4I2.2581

Valarezo Pardo, M. R., Honores Tapia, J. A., Gómez Moreno, A. S., & Vinces Sánchez, L. F. (2018). Comparación de tendencias tecnológicas en aplicaciones web. *3c Tecnología: Glosas de Innovación Aplicadas a La*

Pyme, ISSN-e 2254-4143, Vol. 7, Nº. 3, 2018, Págs. 28-49, *7*(3), 28–49. https://doi.org/10.17993/3ctecno.2018.v7n3e27.28-49/30

Ziegler, M. G. (2022). Web 2.0 and Knowledge Sharing. A Literature Review. *Intech Open*, *2022*, 1–14. https://doi.org/10.5772/ACRT.03

9. Anexos

9.1. Creación de tablas.

- Repositorio One Drive clic aquí.
  - https://estliveupsedu-my.sharepoint.com/:u:/g/personal/dpatinov_est_ups_edu_ec/EbYjT1DVAdZJmCip4RapdXUBs2-4t-IPGlw-oO-FyJRlhg?e=550UFa
- Repositorio Google Drive clic aquí.
  - https://drive.google.com/file/d/1BeZsjjkD9XEaikOUbqXvPUx5z5-hz813/view?usp=sharing

9.2. Creación del backend.

- Repositorio GitHub clic aquí.
  - https://github.com/Daniel070999/catalogo-web-bares-back-end

9.3. Creación del frontend.

- Repositorio GitHub clic aquí.
  - https://github.com/Daniel070999/catalogo-web-bares-front-end

9.4. Docker del ecosistema.

- Ecosistema con Docker One Drive clic aquí.
  - https://estliveupsedu-my.sharepoint.com/:u:/g/personal/dpatinov_est_ups_edu_ec/ESR4c8vLzJNOsUARrzlC4ukB1WVvmE94AUJIhI12cheqdw?e=N0dHDF
- Ecosistema con Docker Google Drive clic aquí.
  - https://drive.google.com/file/d/1cGka_R05oqTPOG347SnLs7w65HJ9RYax/view?usp=sharing