UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Aplicación web para la gestión de inventario de la empresa American Lambster utilizando la tecnología Ruby on Rails.

Trabajo de Titulación para optar al título de Ingeniero en Tecnologías de la Información

Autor:
Yugcha Tisalema, Luis Javier

Tutor:
Ing. Ximena Alexandra Quintana, Ph.D.

Riobamba, Ecuador. 2025

DECLARATORIA DE AUTORÍA

Yo, **Luis Javier Yugcha Tisalema** con cédula de ciudadanía **1805342886**, autor del trabajo de investigación titulado: **Aplicación web para la gestión de inventario de la empresa "American Lambster" utilizando la tecnología Ruby on Rails**, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 15 de septiembre de 2024.

Luis Javier Yugcha Tisalema

C.I: 1805342886

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la Ciudad de Riobamba, a los 15 días del mes de septiembre de 2024, luego de haber revisado el Informe Final del Trabajo de Investigación presentado por el estudiante **Luis Javier Yugcha Tisalema** con CC: **1805342886**, de la carrera Ingeniería en Tecnologías de la Información y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado **"APLICACIÓN WEB PARA LA GESTIÓN DE INVENTARIO DE LA EMPRESA "AMERICAN LAMBSTER" UTILIZANDO LA TECNOLOGÍA RUBY ON RAILS"**, por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.

PhD. Ximena Quintana
TUTORA

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación: Aplicación web para la gestión de inventario de la empresa "American Lambster" utilizando la tecnología Ruby on Rails, presentado por Luis Javier Yugcha Tisalema, con cédula de identidad número 1805342886, bajo la tutoría de Ing. Ximena Quintana, Ph.D.; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 18 de diciembre de 2024

Ana Congacha, Mgs.
PRESIDENTE DEL TRIBUNAL DE GRADO

Pamela Buñay, Mgs.
MIEMBRO DEL TRIBUNAL DE GRADO

Diego Reina, Mgs.
MIEMBRO DEL TRIBUNAL DE GRADO

CERTIFICACIÓN

Que, **Yugcha Tisalema Luis Javier** con CC: **1805342886**, estudiante de la Carrera **Ingeniería en Tecnologías de la Información**, Facultad de **Ingeniería**; ha trabajado bajo mi tutoría el trabajo de investigación titulado " **APLICACIÓN WEB PARA LA GESTIÓN DE INVENTARIO DE LA EMPRESA "AMERICAN LAMBSTER" UTILIZANDO LA TECNOLOGÍA RUBY ON RAILS**", cumple con el 2 %, de acuerdo al reporte del sistema Anti plagio **Turnitin**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 25 de octubre de 2024

XIMENA ALEXANDRA QUINTANA LOPEZ

PhD. Ximena Quintana
TUTORA

DEDICATORIA

En primer lugar, a Dios por su amor incondicional hacia mi por darme fuerza y valor para afrontar cualquier situación que se me ha presentado a lo largo de mi vida universitaria.

A mis padres Gonzalo y Gladys quienes fueron los principales responsables de que yo pueda estudiar ya que sin si apoyo este logro nunca hubiera sido posible.

A mis hermanas Jessica y Mayra por su apoyo incondicional a lo largo de este tiempo.

A Tefa por su apoyo y su ánimo en todo momento.

A mis amigos que he conocido a lo largo de mi vida universitaria ya que de todos he aprendido.

AGRADECIMIENTO

Agradezco y glorifico a Dios.

A mis padres y hermanas quien fueron principalmente las personas que a lo largo de mi vida universitaria me ayudaron a seguir adelante

A mi tutora la Ing Ximena Quintana a su guía durante el proceso de elaboración de la tesis, por sus consejos como amiga y también como docente.

A mis panas Jona, Andrés, Jhona quien fueron mis mejores amigos con los cuales compartí toda la carrera, con los cuales pasé buenos y malos momentos pero siempre estuvimos para apoyarnos.

A la empresa Americam lambster quienes me brindaron su apoyo para poder realizar mi trabajo de grado.

A la facultad de Ingeniería principalmente a la carrera de Tecnologías de la información quine me ha brindado todos los conocimientos para poder obtener este logro.

Javier Yugcha

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La investigación previa sobre el framework Ruby on Rails desempeñó un papel fundamental en el desarrollo del sistema de inventario para American Lambster. Esta investigación proporcionó la base tecnológica necesaria para crear una aplicación web sólida y escalable. La elección de Ruby on Rails se basó en su conocida eficiencia, versatilidad y capacidad para facilitar el desarrollo ágil y seguro de aplicaciones web complejas. Además, se llevó a cabo un análisis detallado de las características y beneficios de Ruby on Rails, lo que orientó el desarrollo del sistema de manera óptima. La metodología de investigación tuvo un enfoque cuantitativo porque se obtuvieron resultados numéricos de los indicadores de rendimiento. Para la gestión y organización del proceso de desarrollo, se adoptó la metodología ágil SCRUM. El sistema de inventario resultante se destacó por ser una solución integral y flexible, capaz de adaptarse fácilmente a las necesidades cambiantes de la empresa. El sistema fue evaluado rigurosamente en términos de rendimiento utilizando la herramienta JMeter, analizando métricas clave como la eficiencia, el tiempo de respuesta y el uso de recursos del sistema. Los resultados indicaron un rendimiento del 100%, cumpliendo con los estándares del modelo de calidad FURPS. El tiempo de respuesta promedio fue de 2456 ms, por debajo del umbral sugerido por FURPS, y el uso de recursos del sistema se mantuvo en un 15%, alineándose con las expectativas del modelo de calidad. Estos resultados confirmaron el rendimiento óptimo del sistema.

Palabras claves: FURPS, Jmeter, Inventario, Metodología Ágil, Rendimiento, Ruby on Rails, SCRUM.

ABSTRACT

Previous research on the Ruby on Rails framework played a critical role in developing the inventory system for American Lambster. This research provided the technological foundation for a robust and scalable web application. The choice of Ruby on Rails was based on its known efficiency, versatility, and ability to facilitate the agile and secure development of complex web applications. In addition, a detailed analysis of the features and benefits of Ruby on Rails was carried out, which optimally guided the development of the system. The research methodology was quantitative because numerical results were obtained from the performance indicators. The agile SCRUM methodology was adopted to manage and organize the development process. The resulting inventory system stood out to be a comprehensive and flexible solution, capable of quickly adapting to the company's changing needs. The system was rigorously evaluated in terms of performance using the JMeter tool, analyzing key metrics such as efficiency, response time, and system resource usage. The results indicated a performance of 100%, meeting the standards of the FURPS quality model. The average response time was 2456 ms, below the threshold suggested by FURPS, and system resource usage remained at 15%, aligning with quality model expectations. These results confirmed the optimal performance of the system.

Keywords: FURPS, Jmeter, Inventory, Agile Methodology, Performance, Ruby on Rails, SCRUM.

Reviewed by:
Ms.C. Ana Maldonado León
ENGLISH PROFESSOR
C.I.0601975980

CAPÍTULO I. INTRODUCCION

En la era actual, los sistemas web para la gestión de inventarios se han convertido en una herramienta esencial para empresas de diversos sectores, incluyendo la industria textil. American Lambster, una destacada empresa de producción de prendas de vestir busca implementar una solución tecnológica que le permita mantener un control preciso y en tiempo real de su amplia gama de productos textiles. Este tipo de sistema no solo agiliza las operaciones internas, sino que también mejora significativamente la visibilidad y el seguimiento de las existencias, permitiendo a la empresa responder con mayor rapidez y eficacia a las demandas del mercado.

La adopción de un sistema web para la gestión de inventarios en American Lambster está estrechamente relacionada con la utilización de nuevas tecnologías y enfoques en el desarrollo de software. Herramientas modernas y frameworks de desarrollo permiten crear soluciones robustas y escalables, capaces de manejar la complejidad y el volumen de datos de una empresa textil en crecimiento. Sin embargo, esta transformación también plantea desafíos para los desarrolladores, quienes deben gestionar la complejidad del software y seleccionar adecuadamente las herramientas de desarrollo que mejor se adapten a las necesidades específicas de la empresa.

Para optimizar los resultados y maximizar la eficiencia, American Lambster puede beneficiarse enormemente de la automatización de sus operaciones mediante el uso de frameworks de software. Estos frameworks permiten reducir el código redundante y repetitivo, asegurando una mayor estandarización y calidad en el desarrollo del sistema. Al implementar estas tecnologías, la empresa puede no solo mejorar el control y la precisión de su inventario, sino también facilitar la integración de nuevas funcionalidades y adaptaciones futuras, garantizando así una solución tecnológica robusta y adaptable a las dinámicas cambiantes del mercado textil.

Ruby on Rails, a menudo abreviado como Rails, es conocido por ser un framework que promueve la convención sobre la configuración, lo que significa que proporciona un conjunto de convenciones predeterminadas para simplificar y acelerar el proceso de desarrollo [1]. Al ofrecer una arquitectura robusta y una serie de herramientas integradas, Rails permite a los desarrolladores centrarse más en la implementación de la lógica empresarial en lugar de lidiar con la configuración y la infraestructura básica del proyecto [2]. Esto facilita la creación de aplicaciones web de alta calidad en un tiempo significativamente reducido, optimizando así la eficiencia y la productividad del equipo de desarrollo [3].

Para llevar a cabo este trabajo de investigación se desarrolló un aplicativo web de inventario para la empresa American Lambster, haciendo uso del Framework Ruby on Rails. Además,

se implementó la metodología ágil de desarrollo de software SCRUM y utilizando el modelo de calidad FURPS, se llevó a cabo una evaluación detallada del rendimiento del sistema.

La investigación está organizada de la siguiente manera: El Capítulo I presenta el planteamiento del problema. En el Capítulo II se exploran los conceptos fundamentales relacionados con el tema. El Capítulo III detalla la metodología empleada y las herramientas utilizadas. El Capítulo IV presenta los resultados obtenidos al evaluar el rendimiento del sistema usando el modelo de calidad FURPS. Por último, el Capítulo V ofrece las conclusiones derivadas de la investigación y las recomendaciones propuestas.

1.1 Planteamiento y justificación de la investigación

La empresa "American Lambster" ha experimentado un notable crecimiento en su línea de productos de ropa en los últimos años, lo que ha incrementado la complejidad en la gestión de su inventario. Actualmente, la empresa utiliza un sistema manual para controlar sus existencias, lo que conlleva diversos desafíos operativos y financieros.

El equipo responsable de la gestión de inventarios enfrenta dificultades para realizar un seguimiento preciso de las cantidades disponibles de cada artículo en las diferentes ubicaciones, debido a la ausencia de una herramienta centralizada que permita acceder a la información en tiempo real. Esta situación provoca que algunos productos se agoten rápidamente, mientras que otros se acumulen en exceso sin ser detectados a tiempo, resultando en pérdidas por ventas perdidas y costos adicionales por mantener inventario no vendido. Además, el proceso de reabastecimiento y pedidos a los proveedores se basa en pronósticos manuales, lo que incrementa la probabilidad de errores en la cantidad de mercancía solicitada. Esto puede llevar a una acumulación innecesaria de inventario, ocupando espacio de almacenamiento e inmovilizando capital que podría ser invertido en otras áreas clave de la empresa.

Para lograr los objetivos de la investigación, se desarrolló una aplicación web para la gestión de inventarios utilizando el framework Ruby on Rails, que se destaca por su enfoque en la convención sobre configuración y su capacidad para agilizar el desarrollo de aplicaciones. Además, se adoptó la metodología SCRUM para garantizar un proceso de desarrollo ágil y eficiente. Finalmente, se evaluó el rendimiento de la aplicación empleando el modelo de calidad FURPS.

1.2 Formulación del problema

¿En qué medida la tecnología Ruby on Rails influirá en el rendimiento de la aplicación web para la gestión de inventarios de la empresa American Lambster?

1.3 Objetivos

General

Implementar una Aplicación Web para la Gestión de inventario de la empresa “American Lambster” utilizando el framework Ruby on Rails.

Específicos

- Investigar la tecnología Ruby on Rails orientado al desarrollo de las aplicaciones web.
- Desarrollar una aplicación web para la gestión de inventario de la empresa “American Lambster” utilizando la tecnología Ruby on Rails.
- Evaluar el rendimiento del sistema utilizando la herramienta JMeter

CAPÍTULO II. MARCO TEÓRICO

2.1 Ruby

Ruby es un lenguaje orientado a objetos y de propósito general, influenciado por Perl y Smalltalk. Fue creado en Japón en los años 90 por Yukihiro Matsumoto [4]. Aunque no cuenta con una especificación completa, su implementación original se considera la referencia principal [5]. Desde 2008, han surgido varias implementaciones alternativas como YARV, JRuby, Rubinius, IronRuby y MacRuby. Se diseñó pensando en las necesidades humanas, priorizando la productividad del programador sobre la eficiencia computacional [5]. En Ruby, todo es un objeto, sin tipos primitivos como los enteros o booleanos de Java. Por ejemplo, un entero es un objeto de la clase Integer.

Es un lenguaje interpretado, sin separación entre compilación y ejecución. Implementaciones como JRuby e IronRuby ofrecen compilación en tiempo de ejecución, mientras que otras usan una interpretación de un solo paso [6]. Ruby tiene tipado dinámico, permitiendo la definición y redefinición de clases en tiempo de ejecución. Los programadores pueden modificar el comportamiento de una clase durante la ejecución de un programa, afectando a todos los objetos de esa clase, incluidas las clases integradas como Integer o String [6].

Los métodos en Ruby pueden tratarse como objetos, pasarse como parámetros a otros métodos y devolver nuevos métodos como resultado [7]. Se le considera un lenguaje multiparadigmático por su capacidad de incorporar constructos de programación funcional, como las funciones de orden superior [7].

2.1.1 Ruby on Rails

Ruby on Rails (Rails) es un framework de desarrollo web de código abierto y de pila completa, desarrollado en el lenguaje de programación Ruby [8]. Este framework permite a los desarrolladores crear aplicaciones capaces de manejar solicitudes HTTP, interactuar con bases de datos para consultas o actualizaciones, y generar respuestas HTTP [8]. La Figura 1 ilustra los componentes de una aplicación web típica que utiliza Rails. Rails requiere un servidor web para gestionar las conexiones HTTP entrantes y está compuesto por cuatro módulos principales: Dispatcher, ActionController, ActiveRecord y ActionView. Además, Rails utiliza una base de datos para almacenar datos de manera persistente y es compatible con diferentes bases de datos y servidores web [8].

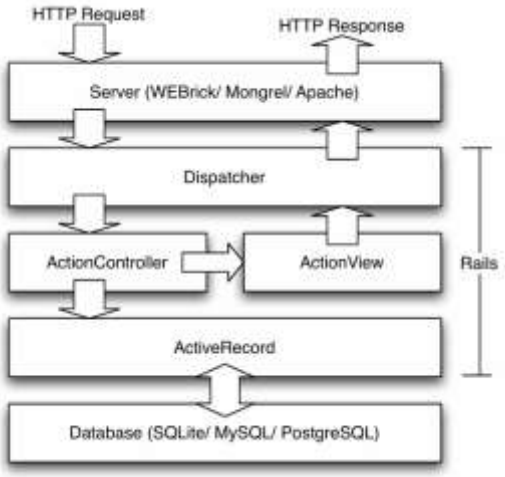


Figura 1: Pila de rieles, las flechas indican una interacción HTTP típica [8]

2.1.2 El procesamiento HTTP y la generación de respuesta

El framework Rails recibe solicitudes HTTP desde un servidor web. Cuando una solicitud es recibida, se redirige a una instancia de ActionController. ActionController puede instruir la consulta o manipulación de datos en una base de datos mediante ActiveRecord. Asimismo, puede dar instrucciones para la generación de una respuesta a través de ActionView [9].

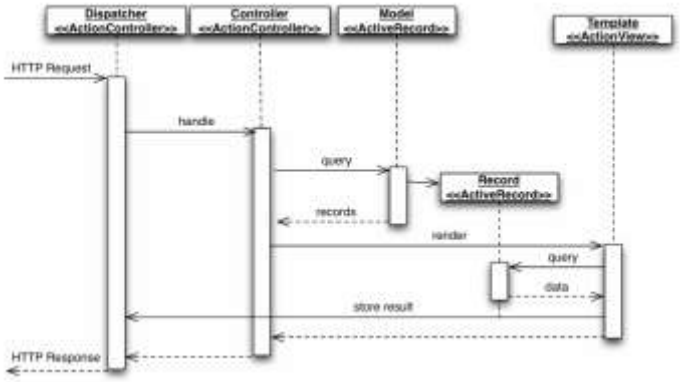


Figura 2: Solicitud HTTP típica [9]

La Figura 2 ilustra el procesamiento típico de una solicitud HTTP por parte de ActionController. La clase ActionController maneja el bucle principal de eventos [9]. Mediante un conjunto de reglas de enrutamiento, una solicitud se mapea a un método en una instancia de ActionController según su URL y su acción HTTP (GET, PUT, POST o DELETE) [9]. ActionController llama automáticamente a este método, que es definido por el programador. Este método puede utilizar información de la solicitud para acceder a la base de datos y preparar los datos para la generación de la respuesta. Luego, puede llamar explícitamente a un método para renderizar la respuesta (como se muestra en la Figura 2) o simplemente finalizar, en cuyo caso ActionController retoma el control y decide cómo generar una respuesta utilizando ciertas convenciones. ActionView es el módulo encargado de generar la respuesta, lo cual se realiza a través de un sistema de plantillas [9]. El programador puede definir una plantilla para la respuesta en un lenguaje específico de

dominio (como ERB, RJS). La respuesta resultante puede tener varios formatos (por ejemplo, HTML, XML, etc.). Una vez generada la respuesta, ActionController retoma el control y la envía al servidor web [9].

2.1.3 El Patrón de Inversión de Control

Rails permite a los desarrolladores definir cómo se manejan las solicitudes mediante el uso del patrón de Inversión de Control (IoC) [10]. IoC implica que, en lugar de que el programador defina la secuencia de operaciones a realizar ante una solicitud, el programador registra las respuestas deseadas para ciertos eventos, y Rails se encarga de controlar el orden exacto y el conjunto de eventos a activar [10]. Este enfoque es empleado por numerosos frameworks populares de aplicaciones web, como Spring, PicoContainer, HiveMind y EJB. En Rails, los desarrolladores definen respuestas a las solicitudes HTTP dentro de ActionController y ActionView, y también pueden registrar respuestas a eventos de manipulación de datos en ActiveRecord [10].

2.1.4 El Patrón MVC de Push

Rails adopta el patrón arquitectónico Model-View-Controller (MVC), ampliamente utilizado en los frameworks de desarrollo de aplicaciones web. Según MVC, el estado de la aplicación debe residir en un modelo [11]. En las aplicaciones de Rails, este estado generalmente se almacena en una base de datos y se accede a través del módulo ActiveRecord, que lleva a cabo el Mapeo Objeto-Relacional (ORM). Este ORM, junto con la base de datos, conforman el componente del modelo [11]. Aunque ActiveRecord es una parte integral de Rails, puede utilizarse de forma independiente.

MVC también dicta que la vista debe ser la representación gráfica del modelo. Existen dos variantes de MVC: Push y Pull [11]. En el MVC de Pull, la vista accede directamente al modelo para extraer la información que se mostrará al usuario. En el MVC de Push, el controlador es quien se encarga de enviar los datos al componente de vista, es decir, los datos son empujados hacia la vista por el controlador [11]. En ambas variantes, el controlador tiene la responsabilidad de realizar operaciones que modifiquen el modelo. Rails emplea el MVC de Push.

2.1.5 El Principio DRY

Rails sigue el principio DRY (Don't Repeat Yourself), lo que implica que cada funcionalidad debería, en lo posible, estar definida solo una vez [12]. Este principio está íntimamente relacionado con la Separación de Preocupaciones (SoC) y conceptos como la Programación Orientada a Aspectos. Un ejemplo de la aplicación del principio DRY en Rails es la capacidad de definir filtros [12]. Una solicitud puede pasar por un filtro antes de ser redirigida a un método de acción específico, proporcionando así una forma de centralizar comportamientos comunes a múltiples acciones en un solo lugar [12].

2.1.6 Desarrollo Ágil de Software y Rails

Rails es particularmente adecuado para los métodos de Desarrollo Ágil de Software. Ágil es un término que engloba un conjunto de métodos de desarrollo de software diseñados para proporcionar un proceso de desarrollo ligero y adaptable [13]. Los métodos ágiles representan una alternativa a los métodos tradicionales basados en planes, que a menudo resultan ser demasiado rígidos y mecanicistas para su uso efectivo. Entre los muchos métodos ágiles se encuentran:

- Proceso de desarrollo Scrum
- Programación Extrema (XP)
- Programación Pragmática (PP)

Los autores [13] de estos métodos también son coautores y firmantes del Manifiesto Ágil, que enumera varias características comunes de los métodos ágiles, tales como:

- El objetivo es entregar software funcional en el menor tiempo posible, de manera rápida y continua.
- El enfoque en el software funcional como la principal medida de progreso y la fuente principal de documentación.
- Permitir cambios en los requisitos en cualquier fase del desarrollo.
- Fomentar una interacción estrecha entre el cliente y el desarrollador.
- Utilizar la comunicación cara a cara como el método principal para transmitir información dentro del equipo de desarrollo.

Rails no impide el uso de ningún método de desarrollo, incluidos los no ágiles. No prescribe un proceso específico, pero apoya las prácticas y procesos comunes en muchos métodos ágiles [13]. Los principios de CoC y DRY ayudan a mantener la base de código pequeña. Los cambios en las etapas avanzadas del desarrollo afectan menos código en los proyectos de Rails en comparación con otros proyectos, facilitando así la respuesta a cambios de último momento, una capacidad esencial en los métodos ágiles [13].

Los generadores permiten a los desarrolladores entregar software funcional desde las primeras etapas del ciclo de desarrollo, mejorando la interacción entre cliente y desarrollador [14]. No obstante, los desarrolladores pueden optar por no utilizar generadores y construir sus aplicaciones desde cero siguiendo un diseño detallado.

Para proporcionar una visión clara y concisa de las opciones disponibles para el desarrollo de aplicaciones web de inventario, se presenta a continuación la Tabla 1 con la comparativa de los frameworks más comúnmente utilizados en este ámbito[14]. Esta tabla tiene como objetivo destacar las características clave, ventajas y desventajas de cada framework, facilitando así la elección del más adecuado según los requerimientos específicos del proyecto y las preferencias del equipo de desarrollo [14]. La selección del framework

correcto es crucial para asegurar la eficiencia, escalabilidad y mantenibilidad de la aplicación a largo plazo.

Tabla 1: Ruby on Rails vs Frameworks

Framework	Lenguaje	Descripción	Ventajas	Desventajas
Ruby on Rails	Ruby	Framework de desarrollo rápido con convenciones sobre configuración.	Rápida curva de desarrollo, gran cantidad de gemas y plugins disponibles.	Pesado para aplicaciones ligeras, rendimiento puede ser un problema en aplicaciones muy grandes.
Django	Python	Framework de alto nivel con patrón MTV y potente administración de bases de datos.	Alta seguridad, excelente documentación y comunidad.	Puede ser excesivo para aplicaciones pequeñas, ORM puede ser restrictivo en casos complejos.
Laravel	PHP	Framework con sintaxis elegante y características integradas.	Gran comunidad, muchos paquetes disponibles, desarrollo rápido con características integradas.	Rendimiento puede ser un problema sin optimización, requiere conocimiento de PHP.
Spring Boot	Java	Simplifica el desarrollo con Spring Framework, configuración automática.	Altamente escalable, adecuado para aplicaciones empresariales grandes, soporte robusto.	Curva de aprendizaje pronunciada, puede ser excesivo para aplicaciones más simples.

Fuente: [14]

2.2 SOA en Rails

La Tabla 2, ilustra las seis posibles tareas que Rails puede realizar en relación con estos protocolos, junto con los nombres de los componentes comúnmente utilizados para cada

tarea. A continuación, se detallará cada una de estas tareas y se explicará cómo pueden ser implementadas en Rails [15].

Tabla 2: Componentes disponibles para SOA clasificados según tareas

		Protocolos		
		HTTP	**XML-RPC**	**SOAP**
Roles	**Invoca**	ActiveResource, Restclient, HTTParty, NET::HTTP, Open-uri	AWS, xmlrpc4r	AWS, soap4r
	Provee	Rails basic functionality	AWS, xmlrpc4r	AWS, soap4r

2.2.1 Provisión de servicios RESTful

Rails es especialmente adecuado para ofrecer servicios RESTful mediante HTTP. Aunque inicialmente fue diseñado para crear sitios web en HTML y solo soportaba las operaciones HTTP GET y POST en URIs, ahora Rails incluye funcionalidades que permiten ofrecer casi cualquier tipo de servicio RESTful sobre HTTP. Esto incluye el soporte completo para todas las operaciones HTTP, como PUT y DELETE [15].

En el estilo REST, un ActionController en Rails puede considerarse responsable de gestionar un recurso específico o un conjunto de recursos [15]. Las solicitudes pueden ser enrutadas al controlador basándose en su URI y luego redirigidas dentro del controlador según la operación HTTP correspondiente. El controlador tiene la capacidad de consultar o actualizar un recurso y generar una respuesta, que puede ser legible para humanos (como imágenes, HTML, texto plano) o procesable por máquinas (como XML, YAML, JSON) [15].

2.2.2 Invocación de servicios RESTful

Rails ofrece funcionalidades para invocar servicios RESTful. El módulo ActiveResource de Rails permite trabajar con recursos remotos sobre HTTP como si fueran locales, convirtiendo llamadas a métodos en solicitudes HTTP y asumiendo representaciones en formato XML específico. Sin embargo, ActiveResource está limitado a ciertos servicios RESTful y espera resultados en formatos específicos, lo que restringe la realización de operaciones arbitrarias en recursos arbitrarios [16].

Al desarrollar una aplicación Rails, es fácil crear un servicio compatible con ActiveResource de otra aplicación [16]. Al generar una entidad con un generador, se crea automáticamente un servicio que maneja un ActiveRecord con representaciones HTML y XML que ActiveResource puede utilizar. Existen otras opciones para invocar servicios RESTful, como

restclient, HTTParty, open-uri y la biblioteca NET::HTTP del núcleo de Rails, las cuales no dependen de Rails [16].

Restclient es una biblioteca sin estado que permite realizar operaciones en cualquier recurso remoto con cualquier representación de entrada, devolviendo las respuestas HTTP con sus encabezados y datos del cuerpo sin procesar, además de manejar errores y redirecciones [17].

HTTParty facilita la creación de objetos que representan servicios dentro del programa, configurados para generar solicitudes HTTP y procesar respuestas en formatos XML y JSON [17].

Open-uri es una biblioteca simple que realiza operaciones HTTP GET con autenticación básica HTTP cuando se le proporciona una URL [17].

NET::HTTP es una biblioteca sin estado que ejecuta solicitudes HTTP sin procesar datos de respuesta, generar datos de solicitud o manejar errores, proporcionando solo funciones básicas para enviar mensajes HTTP. RESTclient y HTTParty utilizan esta biblioteca [17].

2.3 Provisión e Invocación de Servicios usando XML-RPC y SOAP

Para proporcionar o invocar servicios mediante XML-RPC o SOAP, los programadores de Rails tienen varias opciones [18]. La más común es usar ActionWebService. Este módulo formaba parte del framework Rails hasta la versión 2.0, y aunque ya no es compatible, todavía se puede usar en proyectos Rails 2.0 con algunos ajustes. ActionWebService es un módulo que permite la provisión e invocación de servicios tanto XML-RPC como SOAP, y está limitado a la interacción de estilo RPC [18].

Con ActionWebService, los programadores pueden definir procedimientos y sus parámetros en un archivo API, que se asemeja a una interfaz en Java al solo especificar las firmas de los procedimientos. El manejo de las llamadas a procedimiento se realiza mediante métodos de ActionController. ActionWebService expone los procedimientos definidos en un archivo API a través de interfaces XML-RPC y SOAP [18].

Otra opción es usar las bibliotecas Ruby que subyacen a ActionWebService: xmlrpc4r y soap4r. Estas bibliotecas pueden utilizarse para proporcionar o consumir servicios de forma independiente a Rails, funcionando como servidores o clientes XML-RPC o SOAP completos y autónomos. Aunque no están integradas en Rails, pueden ser utilizadas desde la plataforma [18].

Adicionalmente, existe un port del popular framework de servicios web WSO2 al lenguaje Ruby, pero aún es altamente experimental y tiene muy poca documentación disponible [18].

2.4 Metodología de desarrollo de software

El desarrollo ágil de software se ha convertido en una alternativa popular a los métodos tradicionales, permitiendo a las organizaciones adaptarse rápidamente a nuevos desafíos y aumentar la productividad [19]. Metodologías como Scrum, Lean, Extreme Programming y Kanban son ampliamente utilizadas. Scrum, en particular, estructura el trabajo en ciclos cortos llamados sprints, promoviendo la colaboración, la auto-organización y la entrega frecuente de productos [19]. Las reuniones diarias y las revisiones al final de cada sprint aseguran que los equipos se mantengan alineados y adapten sus procesos continuamente. A pesar de sus beneficios, como la mejora de la calidad del software y la rápida respuesta a cambios, también presenta desafíos como la necesidad de una comunicación constante y la gestión de las expectativas del cliente [19].

2.4.1 Metodologías ágiles

La metodología ágil permite el desarrollo rápido de un producto de software en un breve período. Los métodos ágiles siguen los pasos del ciclo de vida de desarrollo de software (SDLC) y aplican diversas prácticas para la gestión de productos de software [20]. Estos procesos promueven la gestión disciplinada y la colaboración en equipo, realizando inspecciones y adaptaciones regulares para mantener los productos en sintonía con los requisitos comerciales cambiantes. La técnica ágil se fundamenta en 12 principios, entre los cuales destacan: priorizar al cliente, adaptarse a los cambios en los requisitos en cualquier momento, desplegar el producto funcional con frecuencia, trabajar en equipo y cumplir con los tiempos establecidos [20].

2.4.2 SCRUM

Ken Schwaber y Jeff Sutherland desarrollaron Scrum y la Guía Scrum y definen a esta metodología como "un marco que permite a las personas resolver problemas complejos de compatibilidad y entregar productos de alto valor de manera productiva y creativa". Scrum es un marco de desarrollo ágil colaborativo que incluye una serie de reuniones, herramientas y roles que ayudan a los equipos a trabajar conjuntamente en la gestión del desarrollo de productos [21].

Jeff Sutherland y Ken Schwaber, en 1993, junto con Takeuchi y Nonaka, quienes introdujeron el término Scrum en 1986, basaron las políticas de Scrum en el juego de rugby, caracterizado por una estructura de equipo integral y auto-organizada [21].

Según diversos estudios, Scrum es una práctica ligera que aumenta la capacidad del equipo para abordar desafíos complejos, adaptándose a los requisitos cambiantes en el desarrollo y despliegue de proyectos importantes, mejorando la colaboración, la creatividad y la eficiencia [21].

Según la Figura 3 representa una estructura que ilustra la relación entre Agile y Scrum, destacando los elementos fundamentales de Scrum. La estructura se asemeja a un edificio con diferentes componentes que sustentan el marco de trabajo [21].

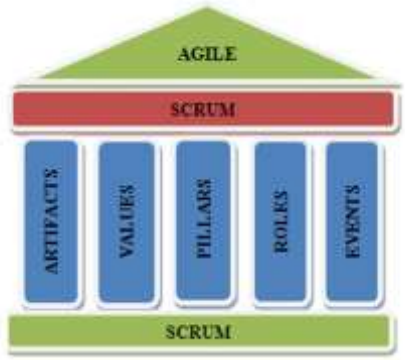


Figura 3: Base y techo de SCRUM [21]

Agile: Representado como el techo del edificio, Agile es la metodología amplia y flexible que engloba varias prácticas de desarrollo de software, incluyendo Scrum.

Scrum: Representado como una viga principal justo debajo del techo y también como la base del edificio, indicando que Scrum es un marco de trabajo dentro de Agile. Scrum está construido sobre los principios y valores ágiles y, a su vez, sirve como base para los elementos específicos que lo componen.

Columnas de Scrum: Estas columnas sostienen la estructura de Scrum y representan los cinco componentes clave del marco de trabajo:

- **Artifacts (Artefactos):** Elementos tangibles utilizados y producidos durante el desarrollo, como el Product Backlog, Sprint Backlog y el Incremento.
- **Values (Valores):** Los valores fundamentales de Scrum, como el compromiso, coraje, enfoque, apertura y respeto, que guían el comportamiento del equipo.
- **Pillars (Pilares):** Los tres pilares de Scrum son transparencia, inspección y adaptación. Estos pilares aseguran que el proceso sea visible, evaluado regularmente y ajustado según sea necesario.
- **Roles:** Los roles dentro de Scrum, que incluyen el Scrum Master, el Product Owner y el equipo de desarrollo. Cada rol tiene responsabilidades específicas para garantizar el éxito del proyecto.
- **Events (Eventos):** Los eventos o ceremonias de Scrum, como las reuniones diarias (Daily Standups), la planificación del sprint (Sprint Planning), las revisiones del sprint (Sprint Review) y las retrospectivas del sprint (Sprint Retrospective), que estructuran el proceso de trabajo y promueven la mejora continua.

2.5 Calidad del Software

Investigadores españoles y colombianos [22] subrayan que el software debe poseer ciertas características que aseguren a los usuarios un producto seguro y confiable en términos de funcionalidad y eficiencia. Estas características mejoran el desempeño del software a lo largo de su ciclo de vida, contribuyendo a la calidad del software. La calidad debe ser garantizada mediante un modelo de calidad que se aplique durante todo el proceso de desarrollo, gestionando las características del software y asegurando coherencia con los requisitos iniciales [22].

El IEEE define una métrica de calidad de software como una función que toma datos del software y produce un valor numérico que representa el grado en que el software posee un atributo específico que afecta su calidad [23]. Esta definición se puede aplicar tanto a productos como a servicios de software e infraestructura. El uso adecuado de estas métricas puede reducir el número de fallos en el software.

Las métricas de calidad del software se agrupan en tres categorías principales: métricas de proyectos, métricas de procesos y métricas de productos. Las métricas de productos son las más comunes en la evaluación del software, ya que se centran en el desarrollo del producto y se utilizan en la fase final del proceso. Según algunos autores [24], [23], las métricas deben ser validadas teóricamente para asegurar que miden correctamente los atributos, validadas empíricamente para evaluar su utilidad en la medición de atributos externos, y deben ser fáciles de realizar, preferiblemente de forma automatizada.

Para medir la calidad del software, el sistema de medición debe seleccionarse considerando estándares que aseguren resultados verificables, reales y no subjetivos, con precisión y significancia [4]. Los modelos de calidad de software siguen una estructura [25], [24], y contienen diversos atributos de calidad evaluables mediante métricas, facilitando una evaluación objetiva al asignar valores cualitativos o cuantitativos a los atributos analizados.

Las métricas de calidad para productos de software abarcan varias medidas, incluyendo el tamaño del software (evaluado mediante las métricas de Halstead [25]), la complejidad del programa (evaluada mediante la complejidad ciclomática [25]), y el rendimiento del programa. Además, se incluyen métricas de accesibilidad y seguridad, así como métricas relacionadas con el proceso de pruebas, como el esfuerzo invertido en las pruebas y su eficiencia.

2.5.1 Modelo FURPS

El modelo FURPS, inicialmente propuesto por Robert Grady [26], fue extendido a FURPS+ por IBM Rational Software [26][27], abarcando requisitos como necesidades físicas, requisitos de interfaz, requisitos de implementación y restricciones de diseño. El modelo clasifica las características en requisitos funcionales y no funcionales. Los requisitos funcionales se definen por las entradas y salidas esperadas o la funcionalidad (F). Por otro

lado, los requisitos no funcionales se agrupan en usabilidad (U), fiabilidad (R), rendimiento (P) y soporte del producto (S) [26]. Estas características se ilustran en la Figura 4.

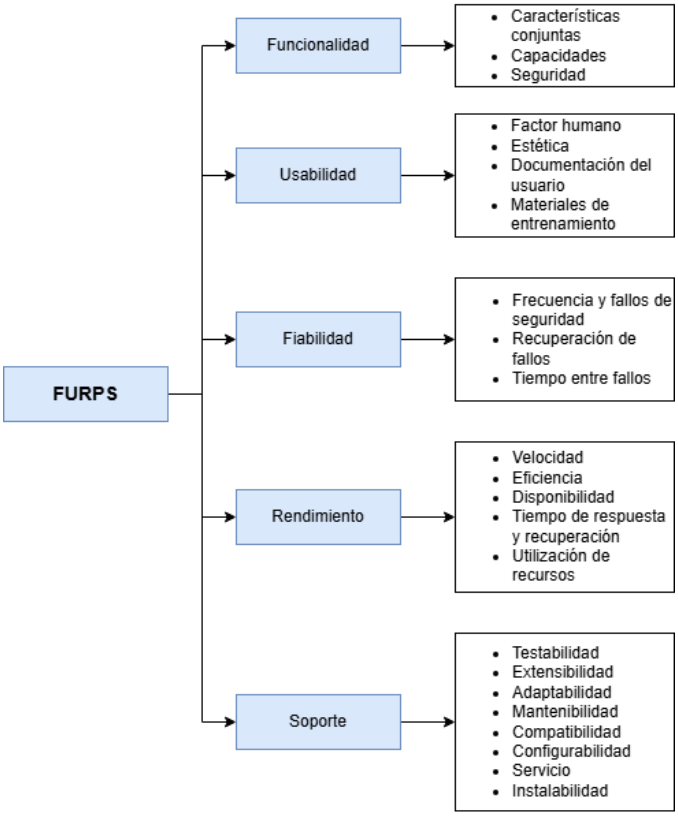


Figura 4: Modelo FURPS [26]

Para evaluar el rendimiento de la aplicación web de inventario desarrollada en Ruby on Rails, se han definido métricas específicas basadas en el modelo FURPS. Este modelo permite una evaluación integral del software considerando tanto sus características funcionales como no funcionales [27]. La Tabla 3 presenta los parámetros clave y los valores asociados que se utilizarán para medir el rendimiento de la aplicación. Estas métricas han sido seleccionadas para garantizar que el sistema no solo funcione correctamente bajo condiciones normales, sino que también mantenga un uso eficiente de los recursos y proporcione tiempos de respuesta adecuados para los usuarios finales [27].

Tabla 3: Métricas de Rendimiento del Modelo FURPS

Parámetros	Valores
Tiempo de respuesta	5s
Uso de recursos	25%

Fuente: [27]

2.6 JMeter

JMeter es una herramienta de código abierto desarrollada por Apache, diseñada principalmente para realizar pruebas de rendimiento y carga en aplicaciones web. Su

objetivo principal es simular múltiples usuarios concurrentes accediendo a una aplicación, lo que permite evaluar el comportamiento del sistema bajo condiciones de carga [28]. JMeter puede generar una gran cantidad de tráfico de usuarios y medir varios aspectos del rendimiento, como tiempos de respuesta, ancho de banda, uso de recursos del servidor y capacidad de procesamiento. Utilizando JMeter, es posible identificar cuellos de botella y problemas de rendimiento en la aplicación antes de que esta se despliegue en un entorno de producción [28].

Para evaluar el rendimiento de una aplicación web con JMeter, se configuran elementos como "Thread Groups", que simulan la cantidad de usuarios concurrentes, y "Samplers", que representan las solicitudes HTTP enviadas a la aplicación [28]. JMeter permite personalizar estos elementos para replicar escenarios de uso realistas, incluyendo variaciones en los tiempos de espera entre las solicitudes y diferentes tipos de peticiones. Los resultados de las pruebas se pueden visualizar mediante diversos "Listeners", que presentan los datos en gráficos y tablas detalladas, facilitando el análisis del rendimiento. Estas capacidades hacen de JMeter una herramienta esencial para asegurar que la aplicación web pueda manejar eficientemente la carga esperada y proporcionar una experiencia de usuario satisfactoria [29].

CAPÍTULO III. METODOLOGIA

El proyecto de investigación se centró en llevar a cabo un análisis cuantitativo con el objetivo de evaluar el rendimiento mediante la aplicación del modelo de calidad FURPS.

3.1 Tipo de investigación

3.1.1 Según la fuente de la investigación

- Investigación bibliográfica: se llevó a cabo a través de la exhaustiva revisión de producción bibliográfica como artículos en revistas indexadas, libros y estudios de caso, con la finalidad de comprender de manera adecuada la aplicación del framework Ruby on Rails en el desarrollo de aplicaciones web, debido a que es el componente principal de este proyecto de investigación.

3.1.2 Según el objeto de estudio

- Investigación aplicada: este trabajo de investigación tiene un enfoque aplicado, puesto que se desarrolló una aplicación web de inventario para la empresa "American Lambster" a través del framework Ruby on Rails. Se consiguió una sinergia entre los conocimientos teóricos y la implementación práctica para brindar una solución adecuada.
- Investigación descriptiva: la investigación se enmarcó como descriptiva, con la finalidad de generar un análisis de los resultados obtenidos en la evaluación del rendimiento usando las métricas del modelo FURPS.

3.2 Técnicas de recolección de datos

La herramienta utilizada para la recopilación de datos en este estudio fue JMeter.

3.3 Población de estudio y tamaño de la muestra

El rendimiento de la aplicación web fue evaluado utilizando la herramienta JMeter, determinándose, así como una población infinita.

3.4 Métodos de análisis, y procesamiento de datos.

El rendimiento se midió utilizando la aplicación JMeter, con la que se obtuvieron resultados numéricos sobre los tiempos de respuesta y el uso de recursos.

3.5 Identificación de variables

3.5.1 Desarrollo de la aplicación web utilizando SCRUM

La aplicación web para la gestión de inventario de la empresa "American Lambster" utilizando la tecnología Ruby on Rails se desarrolló haciendo uso de la metodología ágil de desarrollo SCRUM, en primera instancia se definieron los roles de SCRUM, como se

muestra en la Tabla 7 y posteriormente las actividades a implementar para el desarrollo de software.

Tabla 4: Requisitos no funcionales

Rol	Encargado	Descripción
Product owner		
Scrum master	Ing. Ximena Quintana	Encargada de ejecución
Scrum team	Luis Javier Yugcha	Desarrollador de la aplicación web

Historias de usuario

La Tabla 8, muestra las historias de usuario para la elaboración del sistema con requerimientos del Administrador.

Tabla 5 Historias de usuario para administrador

HISTORIAS DE USUARIO ADMINISTRADOR	
ID	HU01
Nombre	Registro y autenticación de administrador
Prioridad	Alta
Riesgo	Bajo
Descripción	El administrador, debe poder registrarse y autenticarse en la aplicación para acceder a todas las funcionalidades.
Validación	Debe existir un formulario de registro y login para administradores. Debe poder iniciar sesión y acceder a las funciones de administrador.
ID	HU02
Nombre	Ingreso de información de productos
Prioridad	Alta
Riesgo	Medio
Descripción	El administrador, debe poder ingresar la información de los productos disponibles para que los usuarios puedan ver y gestionar los productos.
Validación	Debe existir un formulario para ingresar la información de los productos. La información del producto debe ser almacenada en la base de datos.
ID	HU03
Nombre	Ingreso de información de materia prima
Prioridad	Alta
Riesgo	Medio
Descripción	El administrador, debe poder ingresar la información referente a la materia prima para mantener un registro actualizado de los materiales disponibles.

Validación	Debe existir un formulario para ingresar la información de la materia prima. La información de la materia prima debe ser almacenada en la base de datos.
ID	HU04
Nombre	Generación de Reportes Generales
Prioridad	Media
Riesgo	Alto
Descripción	El administrador, debe poder visualizar los reportes de inventario que se generan en la aplicación.
Validación	Debe existir una vista que muestre los reportes generales de inventario. Los reportes deben incluir información detallada y actualizada sobre los productos y la materia prima.

La Tabla 9, muestra las historias de usuario para la elaboración del sistema con requerimientos del Usuario.

Tabla 6: Historias de usuario para el usuario

HISTORIAS DE USUARIO PARA USUARIO	
ID	HU01
Nombre	Registro y autenticación de usuario
Prioridad	Alta
Riesgo	Bajo
Descripción	El usuario, debe poder registrarse y autenticarse en la aplicación para acceder a las funcionalidades de gestión de productos y materia prima.
Validación	Debe existir un formulario de registro y login para usuarios. Debe poder iniciar sesión y acceder a las funciones de usuario.
ID	HU02
Nombre	Ingreso de información de productos
Prioridad	Alta
Riesgo	Medio
Descripción	El usuario, debe poder ingresar la información de los productos disponibles para mantener actualizado el inventario.
Validación	Debe existir un formulario para ingresar la información de los productos. La información del producto debe ser almacenada en la base de datos.
ID	HU03
Nombre	Ingreso de información de materia prima
Prioridad	Alta
Riesgo	Medio
Descripción	El usuario, debe poder ingresar la información referente a la materia prima para mantener un registro actualizado de los materiales disponibles.

Validación	Debe existir un formulario para ingresar la información de la materia prima. La información de la materia prima debe ser almacenada en la base de datos.
ID	HU04
Nombre	Generación de reportes individuales
Prioridad	Media
Riesgo	Alto
Descripción	El usuario, debe poder visualizar los reportes individuales que se generan en la aplicación.
Validación	Debe existir una vista que muestre los reportes individuales de inventario. Los reportes deben incluir información detallada y actualizada sobre los productos y la materia prima gestionados por el usuario.

Planificación de Sprints

La Tabla 10 explica los sprints que se planificaron para el desarrollo de la aplicación web.

Tabla 7: Planificación de Sprints

Inicio-Sprint 1: Configuración inicial y autenticación de usuarios		
Semanas	**Actividades**	**Responsable**
2 semanas	Configuración del entorno de desarrollo Ruby on Rails	Scrum team
	Configuración de la base de datos.	Scrum team
	Implementación de la funcionalidad de registro y autenticación de usuarios (administrador y usuario).	Scrum team
	Diseño básico de la interfaz de usuario (UI) para el ingreso de usuarios.	Scrum team Scrum master
Planificación-Sprint 2: Diseño de la interfaz intuitiva		
Semanas	**Actividades**	**Responsable**
2 semanas	Diseño detallado y maquetación de la interfaz de usuario.	Scrum team Scrum master
	Implementación de una UI intuitiva y fácil de usar.	Scrum team
	Pruebas de usabilidad para asegurar que la interfaz sea accesible para todos los tipos de usuarios.	Scrum team Product owner
Implementación-Sprint 3: Gestión de productos y materia prima (Administrador)		
Semanas	**Actividades**	**Responsable**
3 semanas	Creación de modelos y controladores para la gestión de productos y materia prima.	Scrum team Scrum master

	Implementación de formularios para que el administrador ingrese la información de productos y materia prima.	Scrum team
	Validación y pruebas de los formularios de ingreso.	Scrum team Product owner
Implementación-Sprint 4: Gestión de productos y materia prima (Usuario)		
Semanas	**Actividades**	**Responsable**
2 semanas	Implementación de formularios para que el usuario ingrese la información de productos y materia prima.	Scrum team
	Validación y pruebas de los formularios de ingreso para usuarios.	Scrum team Product owner
	Ajustes de permisos y roles para diferenciar entre administradores y usuarios.	Scrum team
Validación-Sprint 5: Generación de reportes (Administrador)		
Semanas	**Actividades**	**Responsable**
2 semanas	Implementación de la funcionalidad para la generación de reportes generales de inventario.	Scrum team
	Diseño de vistas para visualizar los reportes.	Scrum team Scrum master
	Pruebas de generación y visualización de reportes.	Scrum team
Validación-Sprint 6: Generación de reportes (Usuario)		
Semanas	**Actividades**	**Responsable**
2 semanas	Implementación de la funcionalidad para la generación de reportes individuales.	Scrum team
	Diseño de vistas para que los usuarios visualicen sus reportes.	Scrum team Scrum master
	Pruebas de generación y visualización de reportes individuales.	Scrum team
Validación-Sprint 7: Integración y pruebas finales		
Semanas	**Actividades**	**Responsable**
2 semanas	Integración de todas las funcionalidades desarrolladas.	Scrum team
	Pruebas de integración y aseguramiento de la calidad.	Scrum team Scrum master
	Resolución de bugs y ajustes finales.	Scrum team
Lanzamiento-Sprint 8: Despliegue y documentación		
Semanas	**Actividades**	**Responsable**

2 semanas	Preparación del entorno de producción y despliegue de la aplicación.	Scrum team
	Documentación del código y de la aplicación.	Scrum team Scrum master
	Formación de usuarios y administradores sobre el uso de la aplicación.	Scrum team

3.5.2 Variable independiente

Framework Ruby on Rails

3.5.3 Variable dependiente

Rendimiento de la aplicación web para la gestión de inventarios de la empresa “American Lambster”.

3.6 Operacionalización de variables

A continuación, en la Tabla 4, se describe la operacionalización de variables.

Tabla 8: Operacionalización de variables

PROBLEMA	TEMA	OBJETIVOS	VARIABLES	DIMENSION	INDICADORES
¿En qué medida la tecnología Ruby on Rails influirá en el rendimiento de la aplicación web para la gestión de inventarios de la empresa Americam Lambster?	Aplicación web para la gestión de inventario de la empresa Americam lambster utilizando la tecnología Ruby on Rails.	**General** Implementar una Aplicación Web para la Gestión de inventario de la empresa "American Lambster" utilizando el framework Ruby on Rails.	**Independiente** Framework Ruby on Rails	Framework	• Número de requerimientos generados. • Número de módulos generados.
		Específicos • Investigar la tecnología Ruby on Rails orientado al desarrollo de las aplicaciones web. • Desarrollar una aplicación web para la gestión de inventario de la empresa "American Lambster" utilizando la tecnología Ruby on Rails. • Evaluar el rendimiento del sistema utilizando la herramienta JMeter.	**Dependiente** Rendimiento de la aplicación web para la Gestión de inventario de la empresa "American Lambster"	Rendimiento	• % Consumo de recurso • Tiempo de respuesta.

3.7 Desarrollo

3.7.1 Análisis de requerimientos

En la Figura 5, se detalla el flujo de la aplicación web.

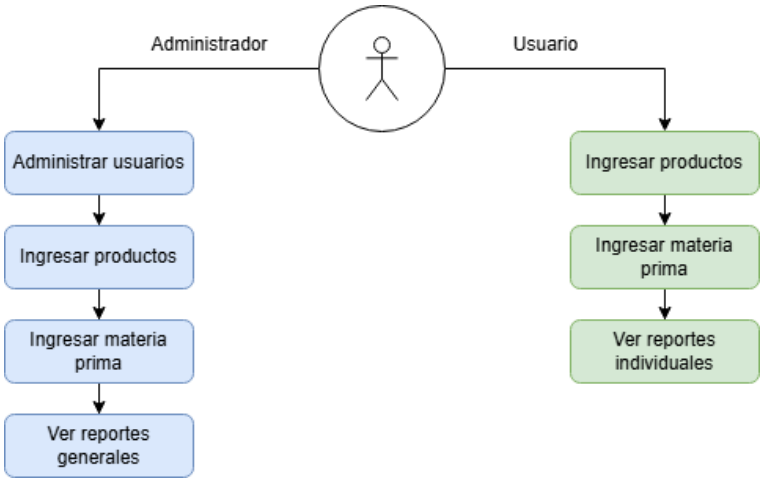


Figura 2: Flujo de la aplicación

Requisitos funcionales

Los requisitos funcionales de la aplicación web se describen en la Tabla 5, a continuación:

Tabla 9: Requisitos funcionales

Requisito	Descripción
Interfaz Intuitiva	Diseñar una interfaz de usuario que sea intuitiva y fácil de usar para todo tipo de usuarios.
Ingreso de Usuarios	Permitir que los usuarios se registren como administrador o como usuario.
Ingreso de productos (Administrador)	El administrador debe ingresar la información de los productos disponibles.
Ingreso de materia prima (Administrador)	El administrador debe ingresar la información referente a la materia prima.
Generación de reportes generales (Administrador)	El administrador debe visualizar los reportes de inventario que se generan en la aplicación.
Ingreso de productos (Usuario)	El usuario debe ingresar la información de los productos disponibles.
Ingreso de materia prima (Usuario)	El usuario debe ingresar la información referente a la materia prima.

Generación de reportes individuales (Usuario)	El usuario debe visualizar los reportes individuales que se generan en la aplicación.

Requisitos no funcionales

Los requisitos no funcionales de la aplicación web se describen en la Tabla 6:

Tabla 10: Requisitos no funcionales

Requisito	Descripción
Seguridad	Implementar procedimientos robustos de seguridad para proteger los datos internos y externos contra accesos no autorizados y vulnerabilidades.
Rendimiento	Asegurar tiempos de respuesta ágiles y un procesamiento eficiente de los datos, incluso durante periodos de alta demanda.
Disponibilidad	Mantener la aplicación disponible para los usuarios, minimizando las interrupciones del servicio.
Usabilidad	Diseñar una interfaz de usuario intuitiva y fácil de navegar para todo tipo de usuarios, con un diseño atractivo y consistente.
Funcionalidad	Garantizar que todas las funciones requeridas en la aplicación funcionen de manera coherente y sin problemas.

3.7.2 Diagramas de casos de uso

Actores

En la Figura 6, se observa el diagrama del administrador, que es el actor principal de la aplicación.

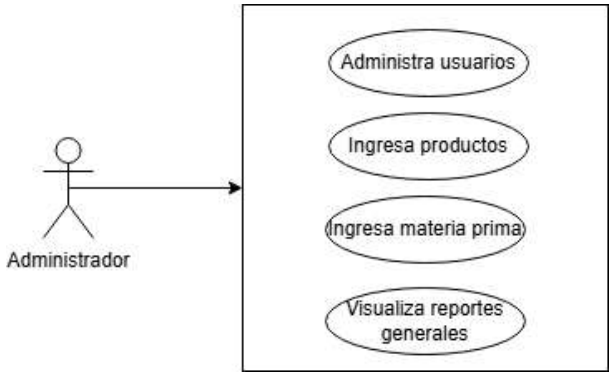


Figura 3: Caso de uso administrador

Usuario

En la Figura 7, se observa el diagrama del usuario, quien tiene 3 acciones dentro de la aplicación.

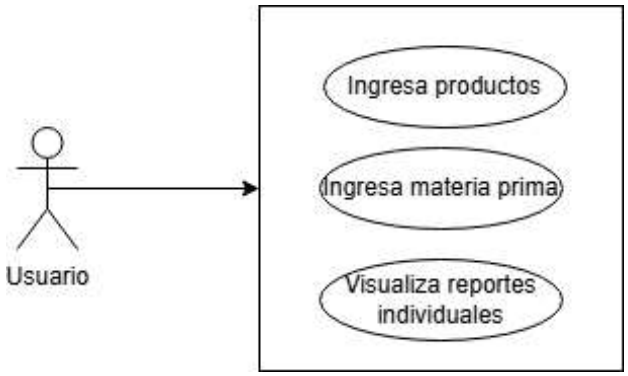


Figura 4: Caso de uso usuario

3.7.3 Diseño de arquitectura

Vista conceptual

La estructura de la aplicación web responde a los siguientes módulos: módulo de inicio de sesión, módulo de control de usuarios, módulo de ingreso de productos, módulo de ingreso de materia prima, módulo de reportes. En la Figura 8, se aprecia la vista conceptual.

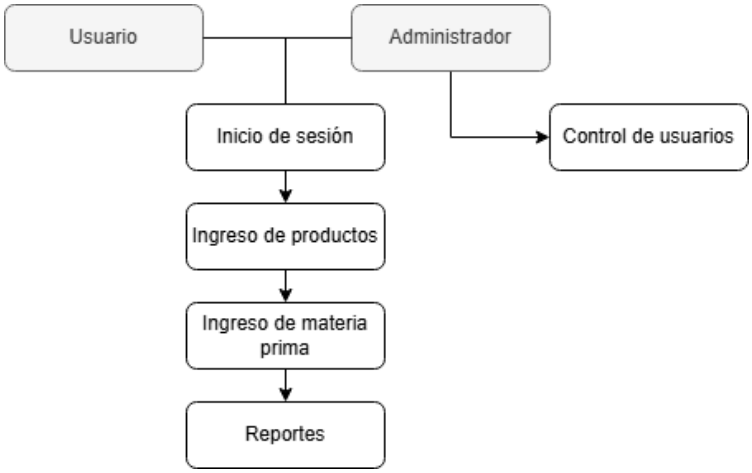


Figura 5: Vista conceptual

El concepto de los módulos se describe a continuación:

- **Módulo de inicio de sesión:** ingreso con credenciales para diferenciar entre usuario y administrador.
- **Módulo de control de usuarios:** el usuario podrá administrar la información de los registrados como usuarios.

- **Módulo de ingreso de productos:** el administrador y el usuario llenar formularios referentes a la información de los productos.
- **Módulo de ingreso de materia prima:** el administrador y el usuario pueden llenar formularios con información de la materia prima.
- **Módulo de reportes:** para graficar los inventarios de productos y de materia prima de la empresa.

Vista lógica

Al estar utilizando el framework Ruby on Rails, la arquitectura de software que se sigue por defecto es la MVC Modelo-Vista-Controlador, como se observa en la Figura 9.

- **Modelo:** aquí se manejan los productos del inventario, materia prima y la representación de los reportes.
- **Vista:** se implementan los formularios de ingreso de información de productos y materia prima, se hace la autenticación de usuarios y la visualización de los reportes.
- **Controlador:** aquí se manejan las solicitudes de usuario, como el control de usuarios, control de productos, materia prima y el controlador para generar los reportes.

Figura 9: Vista lógica

Vista física

En la Figura 10, se visualiza la vista física de la aplicación web.

Figura 10: Vista física

3.7.4 Modelado

La Figura 11 ilustra el diagrama de clases, cuyo propósito es representar las entidades y sus relaciones.

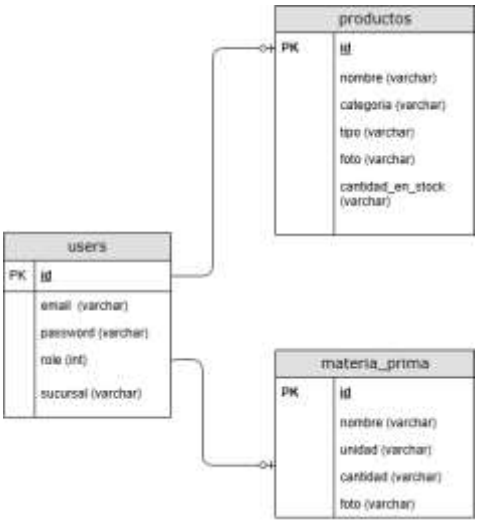


Figura 6: Diagrama de clase

Diagrama de Base de Datos

La Figura 12, muestra la diagramación de los registros con el objetivo de representar la estructura de registros obtenida de SQL Server.

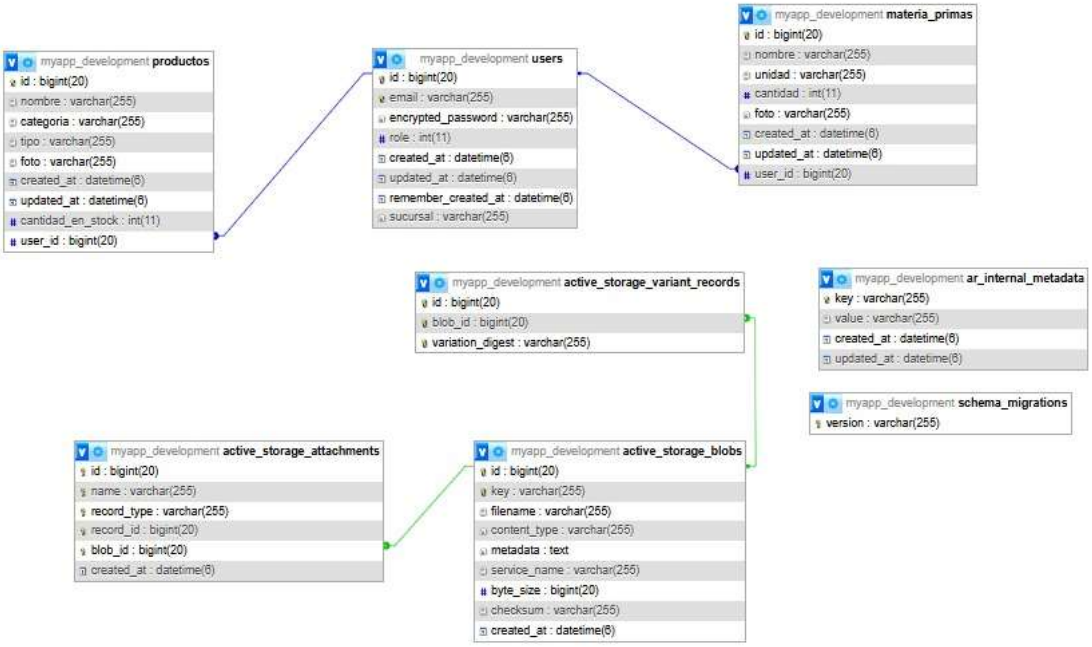


Figura 7: Base de Datos

3.7.5 Inicio

A continuación, se presenta la Tabla 11, que detalla el Product Backlog utilizado durante el desarrollo del sistema de inventario basado en Ruby on Rails. En esta tabla se detallan los ítems de trabajo, las tareas correspondientes y el esfuerzo estimado en puntos.

Tabla 11: Product backlog

Ítem	Tarea	Esfuerzo
Autenticación de Administrador (A01)	Implementar registro y login de administrador	5 puntos
	Validar credenciales del administrador	3 puntos
Autenticación de Usuario (U01)	Implementar registro y login de usuario	5 puntos
	Validar credenciales del usuario	3 puntos
Gestión de Productos (Administrador)	Crear formulario para ingreso de productos (Administrador)	4 puntos
	Implementar validaciones para campos de productos	3 puntos
Gestión de Materia Prima (Administrador)	Crear formulario para ingreso de materia prima (Administrador)	4 puntos
	Implementar validaciones para campos de materia prima	3 puntos
Gestión de Productos (Usuario)	Crear formulario para ingreso de productos (Usuario)	4 puntos
	Implementar validaciones de campos para productos	3 puntos
Gestión de Materia Prima (Usuario)	Crear formulario para ingreso de materia prima (Usuario)	4 puntos
	Implementar validaciones de campos para materia prima	3 puntos
Generación de Reportes (Administrador)	Implementar vista de reportes generales de inventario	6 puntos
	Validar datos en reportes de inventario	4 puntos
Generación de Reportes (Usuario)	Implementar vista de reportes individuales de inventario	6 puntos
	Validar datos en reportes individuales	4 puntos
Diseño de Interfaz Intuitiva	Crear y diseñar interfaz de usuario intuitiva	7 puntos
	Pruebas de usabilidad para interfaz	5 puntos

Sprint 1: Configuración inicial y autenticación de usuarios

Durante el primer sprint, se completaron exitosamente varias actividades clave para establecer la base del proyecto. Se configuró el entorno de desarrollo en Ruby on Rails y se estableció la base de datos, asegurando un entorno robusto para el desarrollo futuro. Además, se implementaron las funcionalidades de registro y autenticación de usuarios, permitiendo tanto a administradores como a usuarios registrarse y acceder al sistema de manera segura. Finalmente, se diseñó una interfaz de usuario básica para el ingreso de usuarios, proporcionando una base intuitiva y accesible para futuras mejoras en la experiencia del usuario, como se observa en la Figura 13.

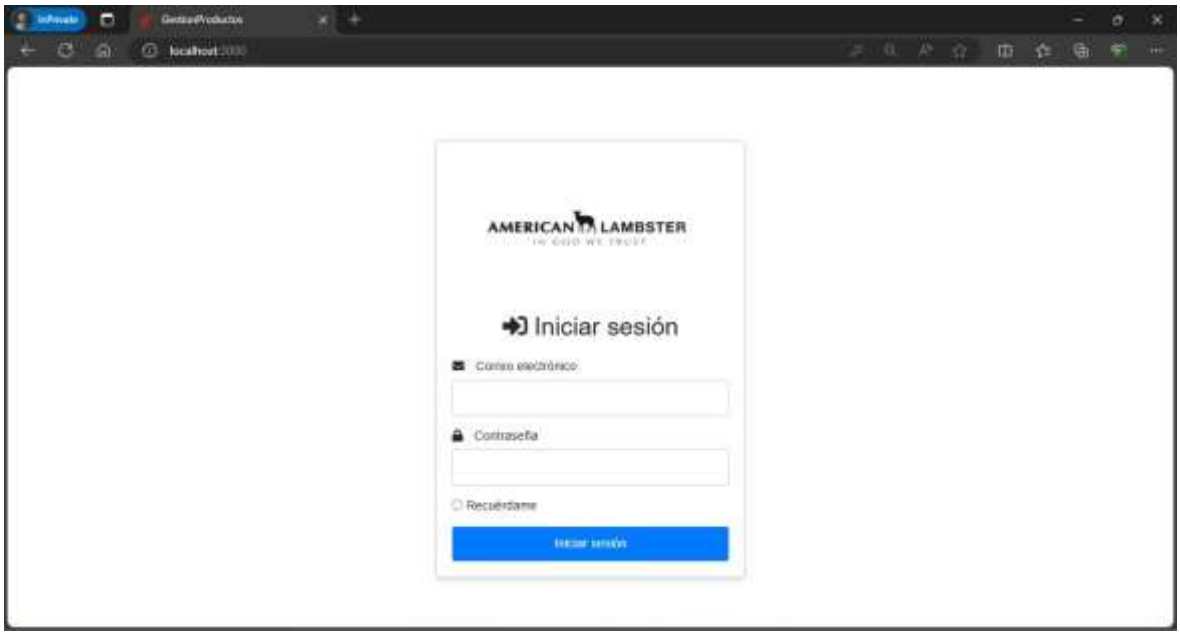


Figura 13: Pantalla de Inicio de sesión

3.7.6 Planificación

Sprint 2: Diseño de la interfaz intuitiva

En el segundo sprint, se realizaron actividades esenciales enfocadas en mejorar la experiencia del usuario. Se completó el diseño detallado y la maquetación de la interfaz de usuario, asegurando que cada elemento visual esté alineado con los objetivos de usabilidad del proyecto. Se implementó una interfaz de usuario intuitiva y fácil de usar, diseñada para facilitar la navegación y el uso del sistema por parte de todos los tipos de usuarios se visualiza en la Figura 14. Además, se llevaron a cabo pruebas de usabilidad exhaustivas para asegurar que la interfaz sea accesible y comprensible para todos los usuarios, independientemente de su nivel de experiencia técnica, incluso se agregó un modo noche para transformar los colores de la interfaz, esta funcionalidad se aprecia en la Figura 15.

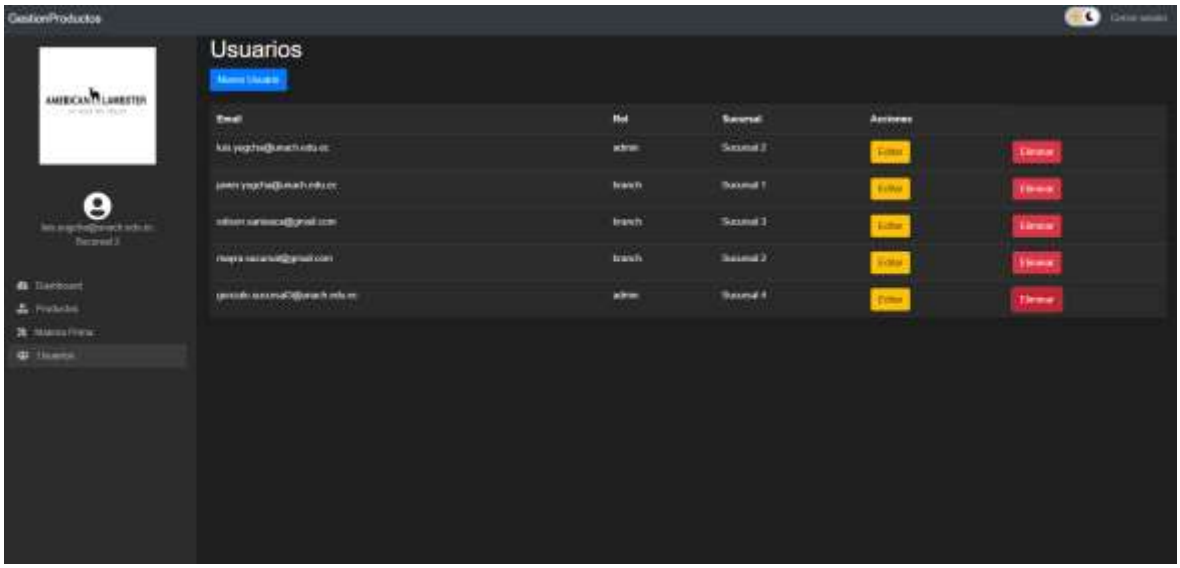


Figura 14: Pantalla de usuarios

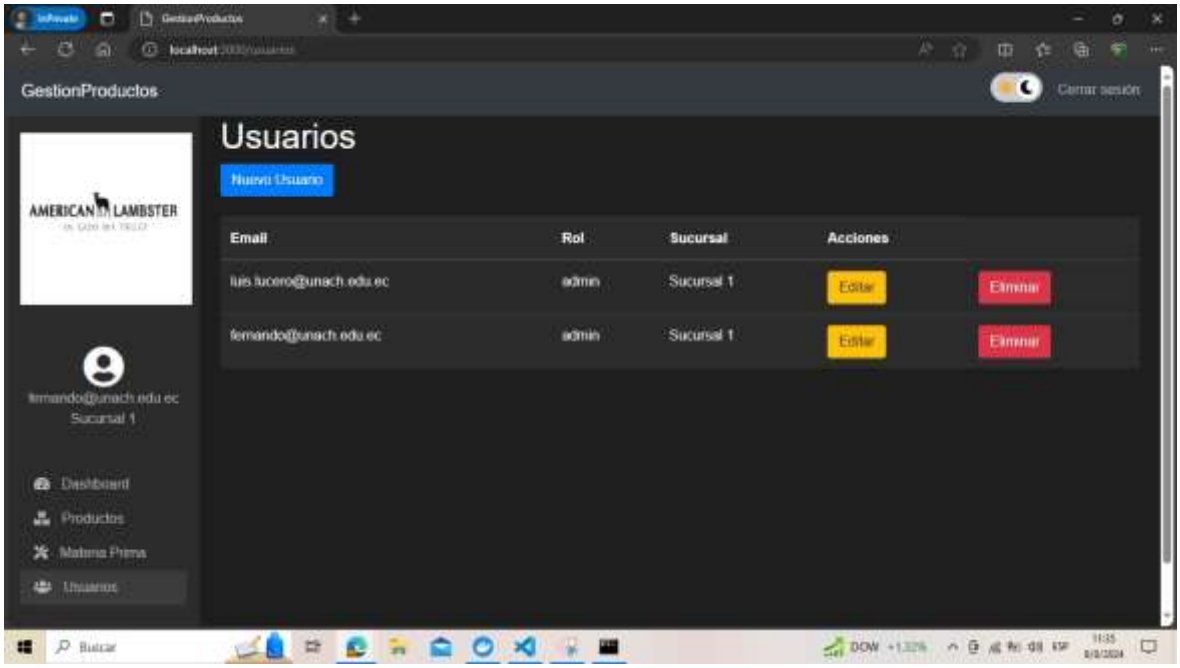


Figura 15: Pantalla de usuarios modo oscuro

3.7.7 Implementación

Sprint 3: Gestión de productos y materia prima (Administrador)

Para el tercer sprint, como se demuestra en la Figura 16 y Figura 17, se avanzó significativamente en la funcionalidad de gestión del inventario. Se crearon los modelos y controladores necesarios para la gestión de productos y materia prima, estableciendo la estructura fundamental para el manejo de datos dentro del sistema. Se implementaron formularios que permiten a los administradores ingresar información detallada sobre productos y materia prima, facilitando una gestión precisa y actualizada del inventario. Además, se realizaron validaciones y pruebas exhaustivas de estos formularios para asegurar su correcto funcionamiento y la integridad de los datos ingresados, garantizando así una experiencia de usuario fiable y eficiente.

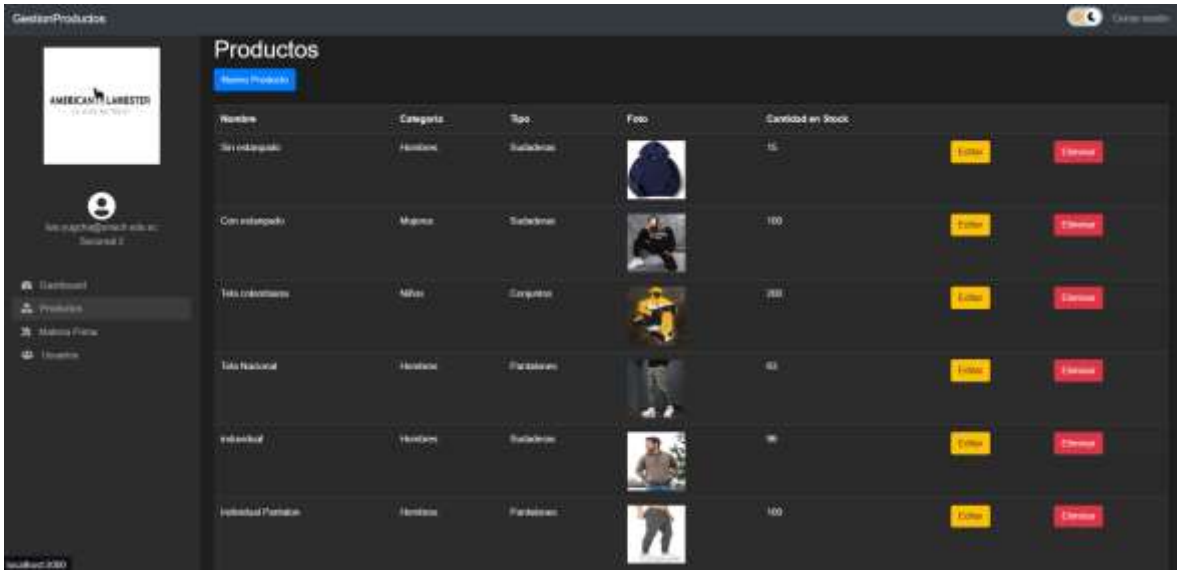


Figura 16: Interfaz de gestión de productos

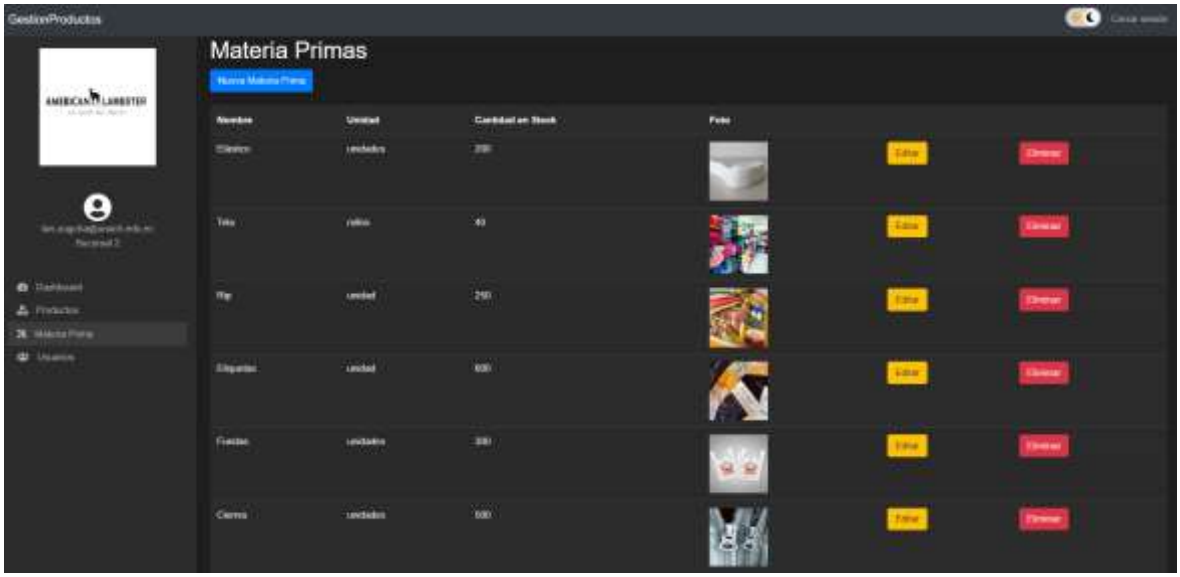


Figura 17: Interfaz de gestión de materia prima

Sprint 4: Gestión de productos y materia prima (Usuario)

El cuarto sprint, se enfocó en habilitar funcionalidades clave para los usuarios estándar del sistema. Se implementaron formularios específicos para que los usuarios pudieran ingresar información relacionada con productos y materia prima, asegurando que el inventario se mantenga actualizado desde múltiples puntos de entrada, como se puede visualizar en la Figura 18 y Figura 19. Además, se llevaron a cabo validaciones y pruebas exhaustivas de estos formularios para verificar su correcto funcionamiento y asegurar la precisión de los datos ingresados por los usuarios. También se realizaron ajustes en los permisos y roles dentro del sistema para diferenciar claramente las capacidades y accesos entre administradores y usuarios, estableciendo un control de acceso adecuado y mejorando la seguridad y la gestión del sistema. Por ejemplo, poder editar detalles del producto como se muestra en la Figura 20 y poder eliminar productos como en la Figura 21.

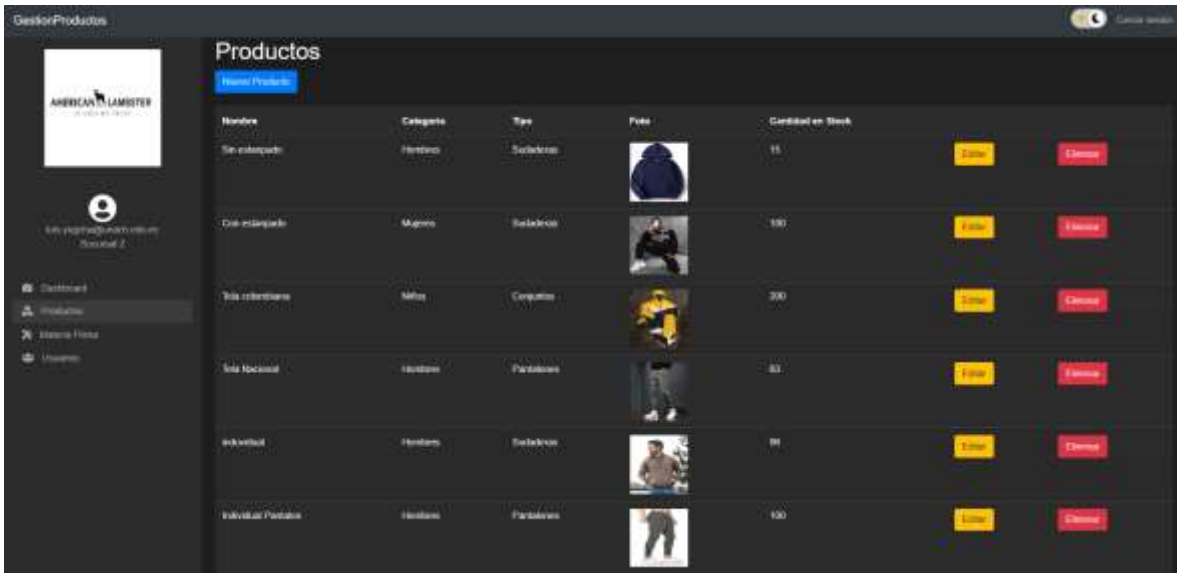


Figura 18: Nuevo producto

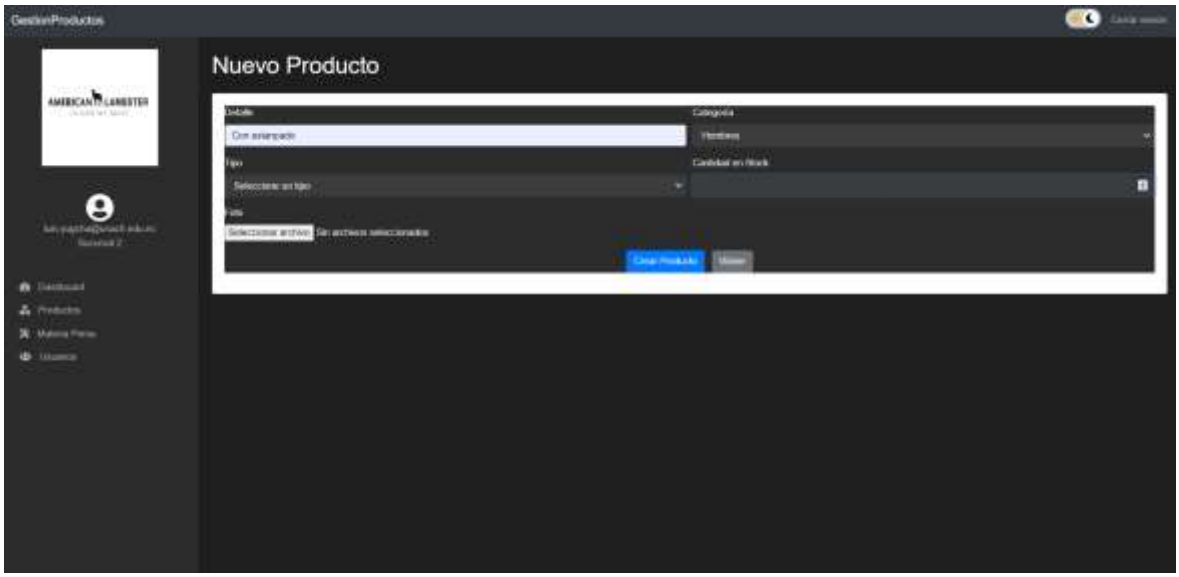


Figura 19: Formulario para crear nuevo producto

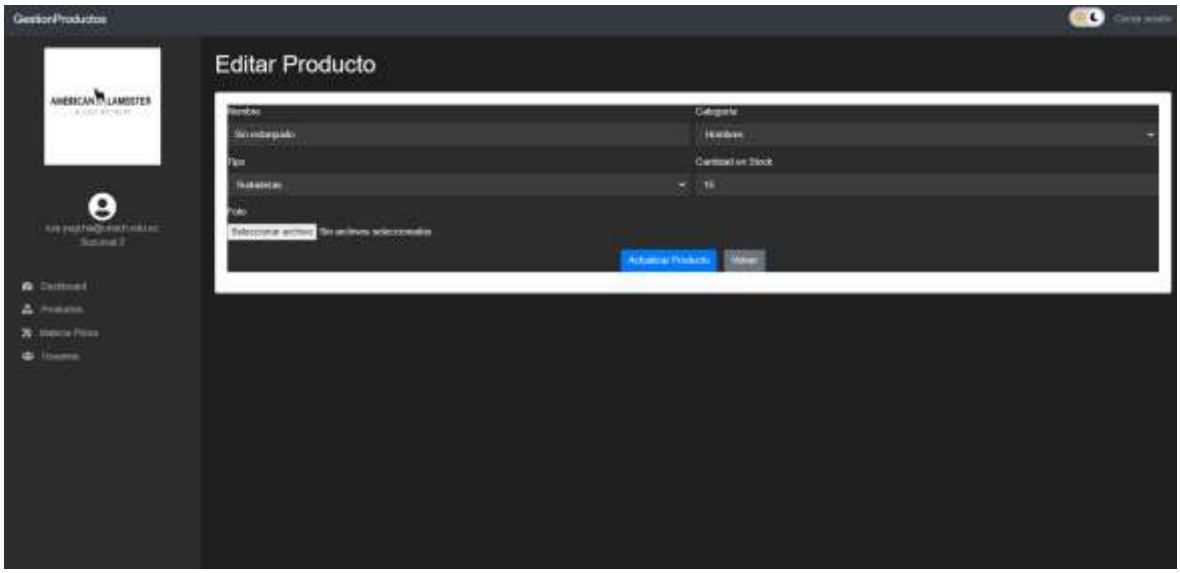


Figura 20: Editar nuevo producto

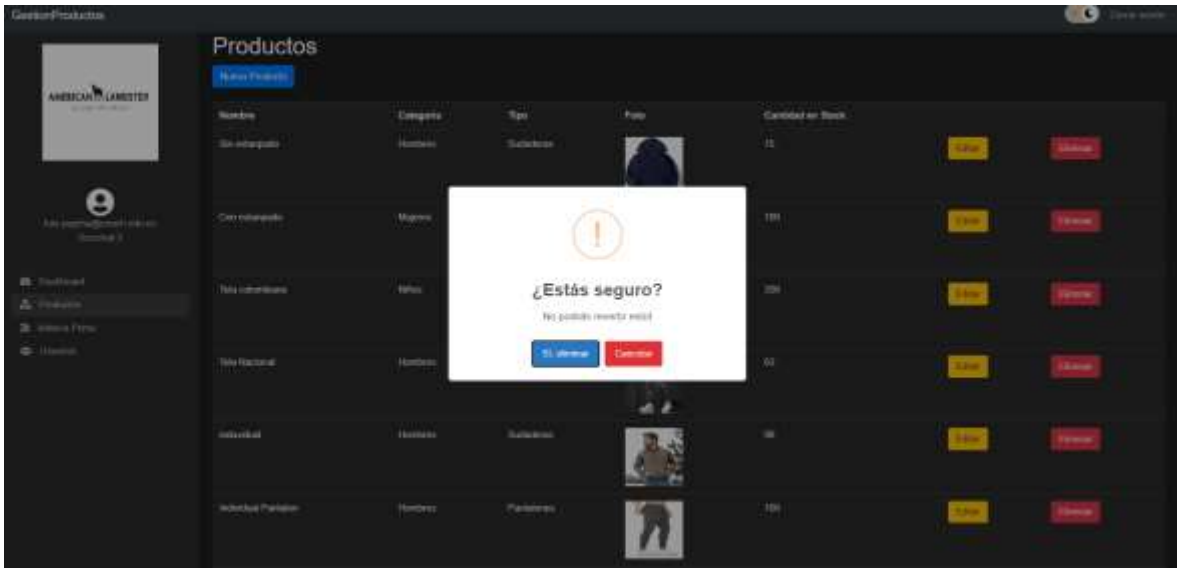


Figura 21: Eliminar nuevo producto

3.7.8 Validación

Sprint 5: Generación de reportes (Administrador)

Durante el quinto sprint, se implementaron funciones cruciales para la gestión y visualización del inventario. Se desarrolló la funcionalidad para la generación de reportes generales de inventario, permitiendo obtener una visión integral y detallada del estado actual de los productos y la materia prima, lo que se muestra en la Figura 22. Se diseñaron vistas específicas para la visualización de estos reportes, asegurando que la información se presente de manera clara y accesible para los administradores. Además, se realizaron pruebas exhaustivas tanto en la generación como en la visualización de los reportes, verificando la precisión de los datos y la facilidad de uso de las interfaces diseñadas para este propósito.

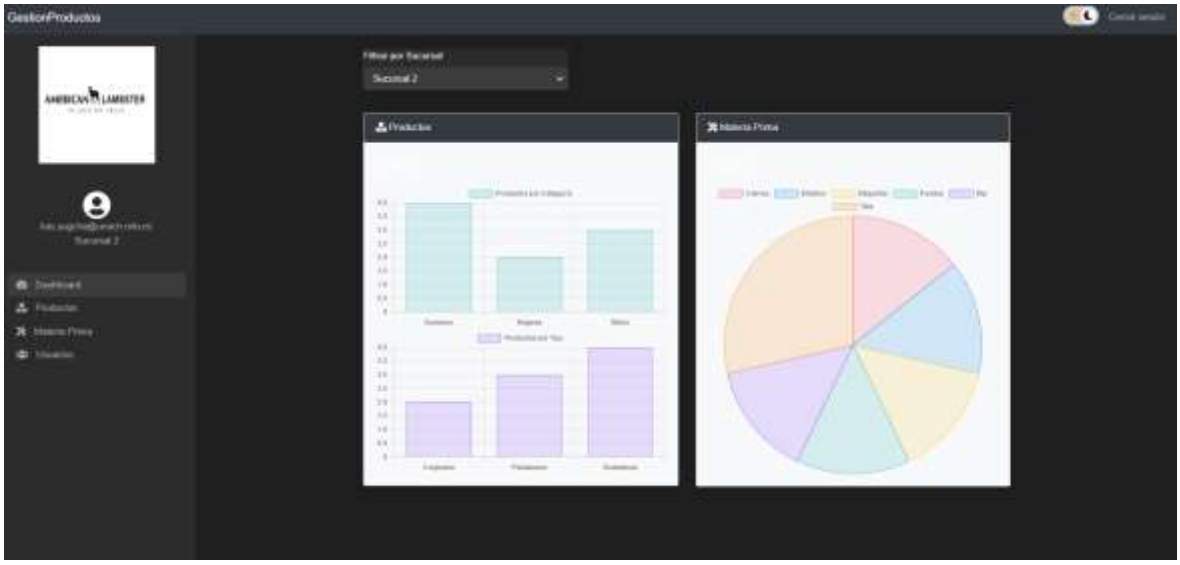


Figura 22: Reporte de productos y materia prima para el administrador

Sprint 6: Generación de reportes (Usuario)

En el sexto sprint, se implementaron funcionalidades esenciales para la personalización y gestión individual de datos de inventario. Se desarrolló la funcionalidad para la generación de reportes individuales, permitiendo a los usuarios obtener información detallada y específica sobre los productos y la materia prima que han gestionado. Se diseñaron vistas dedicadas para que los usuarios puedan visualizar estos reportes de manera clara y accesible, como se visualiza en la Figura 23. Además, se llevaron a cabo pruebas exhaustivas tanto en la generación como en la visualización de los reportes individuales, asegurando la precisión de los datos y la usabilidad de las interfaces diseñadas para este fin.

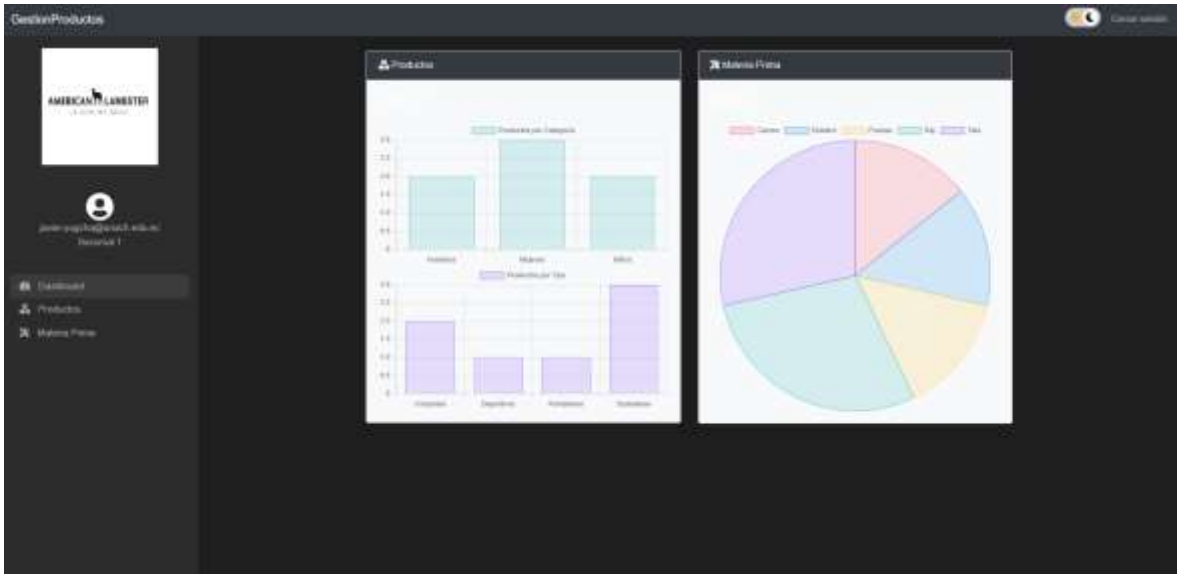


Figura 23: Reporte de productos y materia prima para el usuario

Sprint 7: Integración y pruebas finales

Durante el séptimo sprint, se centró en consolidar y afinar el sistema para asegurar su funcionamiento integral y libre de errores. Se realizó la integración de todas las funcionalidades desarrolladas hasta el momento, garantizando que todos los componentes del sistema trabajen de manera cohesiva. Se llevaron a cabo pruebas de integración y aseguramiento de la calidad para identificar y corregir cualquier incompatibilidad o problema emergente. Además, se procedió a la resolución de bugs y se hicieron los ajustes finales necesarios, asegurando que el sistema esté completamente operativo y optimizado para su uso final.

3.7.9 Lanzamiento

Sprint 8: Despliegue y documentación

Finalmente, en el octavo sprint, se llevaron a cabo las actividades necesarias para el despliegue exitoso de la aplicación en un entorno de producción. Se preparó el entorno de producción y se desplegó la aplicación, asegurando que todos los componentes estén configurados y funcionando correctamente en el nuevo entorno. Se completó la documentación del código y de la aplicación, proporcionando guías detalladas para el uso y

mantenimiento del sistema. Además, se llevó a cabo la formación de usuarios y administradores, instruyéndolos sobre el uso de la aplicación y sus funcionalidades, para asegurar una transición suave y una adopción eficiente del sistema en el entorno de trabajo real.

3.7.10 Planificación de pruebas

Para llevar a cabo la etapa de evaluación del sistema de inventario desarrollado en Ruby on Rails, se seleccionó Apache JMeter como la herramienta principal para llevar a cabo las pruebas de rendimiento. JMeter es una herramienta ampliamente reconocida y utilizada para realizar pruebas de rendimiento en aplicaciones web. La razón principal por la que seleccionó esta herramienta es que JMeter simula condiciones de uso real en un entorno de solicitudes concurrentes, permitiendo evaluar la respuesta del sistema bajo diferentes niveles de carga. La Tabla 11 presenta los indicadores medidos con JMeter.

Tabla 12: Indicadores medidos con JMeter

Indicador	Definición
% Consumo de recursos	Se refiere a la proporción de recursos del sistema, como la CPU y la memoria, que son utilizados por la aplicación durante su funcionamiento. Este indicador es crucial para evaluar la eficiencia del software, asegurando que el sistema no sobrecargue los recursos disponibles y mantenga un rendimiento óptimo.
% Tiempo de respuesta	Mide la proporción de tiempo que la aplicación tarda en responder a las solicitudes de los usuarios. Este indicador es esencial para evaluar la rapidez con la que el sistema procesa y responde a las interacciones, lo que impacta directamente en la experiencia del usuario y la eficiencia operativa del software.

3.7.11 Ejecución de las pruebas

Cada módulo se sometió a pruebas planificadas, se simularon condiciones de uso real a través de actividad de usuarios concurrentes. Para cada método disponible, tanto GET como POST, se establecieron parámetros relevantes que aseguran una representación precisa de los escenarios de uso. Este enfoque permitió evaluar la capacidad del sistema para gestionar cargas elevadas y detectar posibles limitaciones o áreas de mejora en el rendimiento. La Figura 24, se observa cómo se configuró la herramienta JMeter para la realización de pruebas del sistema.

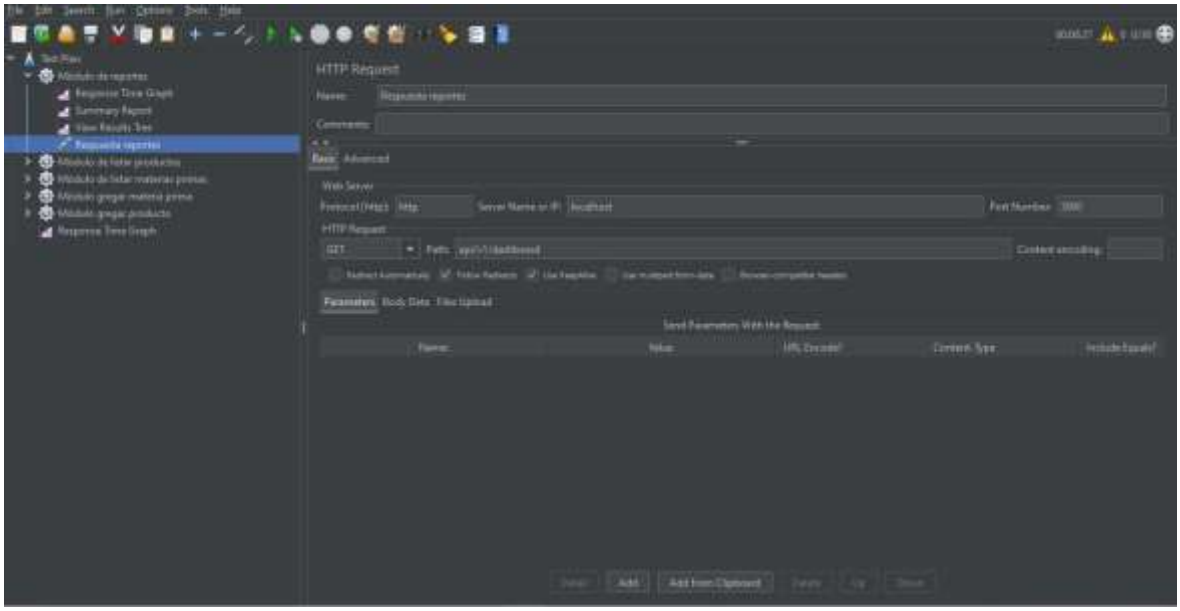


Figura 24: Pruebas HTTP Request en Jmeter

En la figura 25, se visualiza muestra la configuración del protocolo HTTP request en los métodos POST y GET, en el módulo Listar Productos.

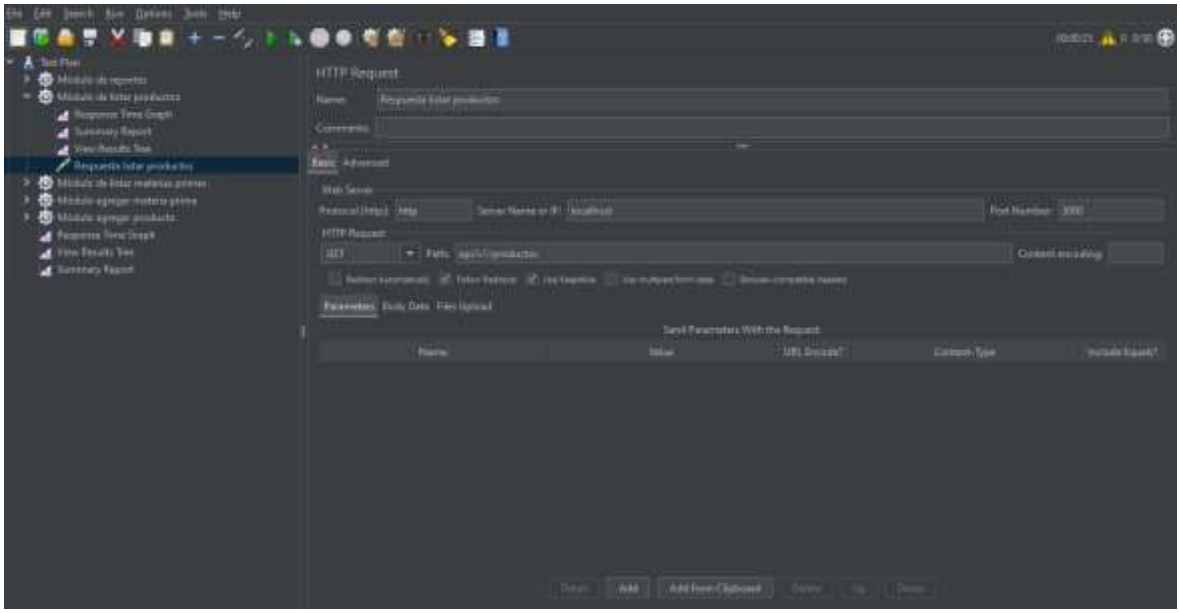


Figura 25: Pruebas HTTP Request en JMeter

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

4.1 Resultados

Una vez culminada la etapa de elaboración del sistema de inventario, se realizaron las pruebas de rendimiento haciendo uso de la herramienta JMeter. Los resultados que se obtuvieron se alinean al modelo de calidad FURPS. El resumen generado por la herramienta, se representa en la Figura 26, y revela que no se registraron errores en las solicitudes, lo cual es un indicio positivo de la estabilidad de la aplicación, los tiempos de respuesta presentan un tiempo máximo de 10.262 ms, el rendimiento global de la aplicación, medido en términos de throughput, es alto, con una tasa promedio de procesamiento de solicitudes que varía entre 2,4 y 3,5 solicitudes por segundo.

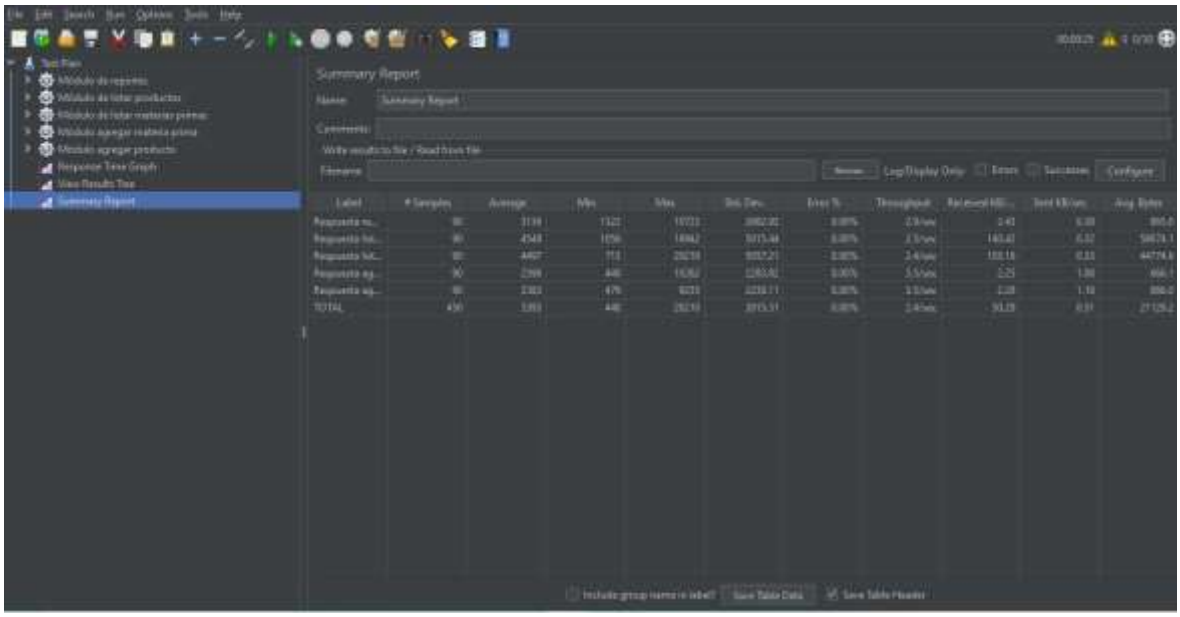


Figura 26: Pruebas HTTP Request en Jmeter

La Figura 27 corresponde al árbol de resultados de JMeter, que permite analizar en detalle cada una de las solicitudes enviadas durante la prueba de rendimiento de la aplicación web. En este caso, todas las solicitudes de las diferentes funcionalidades del sistema, como el módulo de reportes y el listado de productos y materias primas, han sido marcadas con un símbolo verde de éxito. Esto indica que todas las peticiones fueron procesadas correctamente por el servidor sin ningún tipo de error. La ausencia de errores sugiere que la aplicación es estable bajo las condiciones de prueba implementadas, lo que es un buen indicador de su capacidad para manejar las operaciones solicitadas.

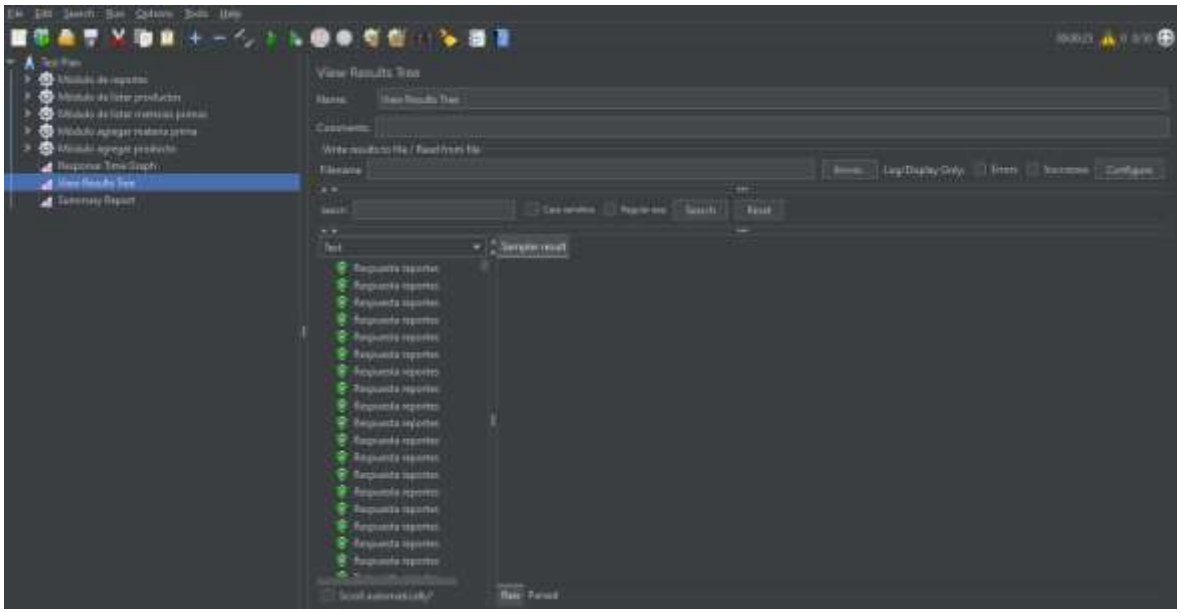


Figura 27: Pruebas HTTP Request en Jmeter

En la tabla 12, se resume la cantidad de solicitudes realizadas en la ejecución de pruebas, que corresponde a ejecutar 3 bucles de carga con 90 solicitudes en 15 segundos, siendo asi que se realizaron un total de 450 procedimientos.

Tabla 13: Total de procedimientos realizados

Módulo	Cantidad de Pruebas
Módulo de reportes	90
Módulo de listar productos	90
Módulo de listar materias primas	90
Módulo de agregar materia prima	90
Módulo de agregar producto	90
TOTAL	**450**

La figura 28, es un gráfico de anillo que representa el porcentaje de pruebas exitosas que se obtuvieron.

Figura 28: Pruebas HTTP Request en JMeter

4.2 Valoración de indicadores

4.2.1 Número de requerimientos generados

El número de requerimientos generados en este proyecto asciende a un total de nueve, divididos entre funcionalidades para administradores, usuarios y la interfaz de usuario. Estos requerimientos abarcan aspectos esenciales como la autenticación, la gestión de productos y materia prima, la generación de reportes, y el diseño de una interfaz intuitiva, asegurando así una cobertura integral de las necesidades de la aplicación web para inventario.

4.2.2 Número de módulos generados

La aplicación web de inventario tiene un total de cinco módulos para su adecuado funcionamiento: módulo de inicio de sesión, módulo de control de ususarios, módulo de ingreso de productos, módulo de ingreso de materia prima y módulo de reportes.

4.2.3 Tiempo de respuesta

La Figura 29 muestra los tiempos de respuesta medidos en milisegundos de los diferentes módulos de la aplicación web, utilizando JMeter. Se observan cinco módulos de operaciones: agregar productos, agregar materia prima, listar materias primas, listar productos, y generar reportes.

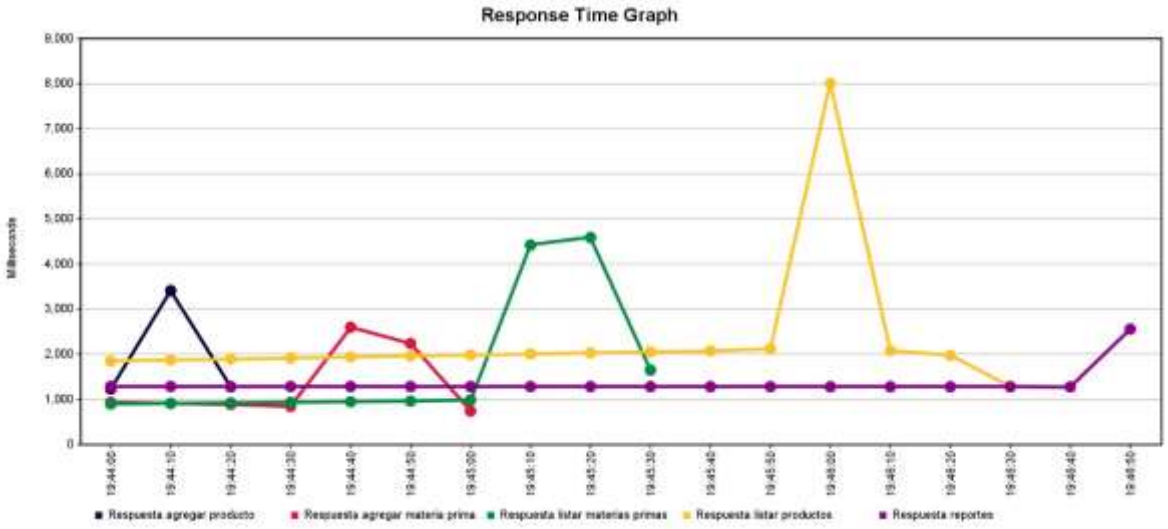


Figura 29: Gráfica de tiempo de respuesta

Respuesta agregar producto (línea azul): Los tiempos de respuesta para esta operación son relativamente estables, con picos ocasionales que alcanzan hasta 2,000 milisegundos, indicando que el sistema mantiene un rendimiento consistente, aunque con algunas fluctuaciones bajo ciertas condiciones.

Respuesta agregar materia prima (línea roja): Similar a la operación de agregar productos, los tiempos de respuesta también son estables, con un ligero aumento alrededor

de los 2,000 milisegundos en un punto, lo que sugiere que el rendimiento es generalmente estable, pero podría estar sujeto a pequeñas variaciones.

Respuesta listar materias primas (línea verde): Esta operación muestra una mayor variabilidad en los tiempos de respuesta, con un notable pico que alcanza aproximadamente 5,000 milisegundos. Este incremento indica que el sistema podría enfrentar dificultades al gestionar la operación de listado de materias primas, especialmente bajo condiciones de carga.

Respuesta listar productos (línea amarilla): Este es el que presenta mayor variabilidad, con un pico significativo que alcanza cerca de 8,000 milisegundos. Este comportamiento sugiere que listar productos es la operación más costosa en términos de tiempo de respuesta y podría ser un área clave para optimizar en el sistema.

Respuesta reportes (línea morada): Los tiempos de respuesta para la generación de reportes son los más consistentes y bajos, manteniéndose alrededor de los 1,000 milisegundos. Esto indica que la operación de generación de reportes es manejada eficientemente por el sistema.

4.2.4 Consumo de recursos

La Tabla 12 permite visualizar el porcentaje de utilización de recursos que exige la aplicación web para inventario.

Tabla 14: Porcentaje de utilización de recursos

Parámetro	Indicador	% Consumo
Uso de recursos	Uso de CPU	20%
	Uso de RAM	10%
	Uso de disco duro	15%

4.2.5 Valores obtenidos del estudio en base al modelo de calidad FURPS

La Tabla 13 representa los obtenidos en las pruebas, basándose en el modelo de calidad FURPS.

Tabla 15: Valores obtenidos en las pruebas

Parámetros	Valores
Tiempo de respuesta	2456 ms
Uso de recursos	15%

4.3 Discusión

La elección del framework Ruby on Rails en el desarrollo del sistema de inventario tuvo un impacto significativo. Ruby on Rails fue seleccionado por su eficiencia y por facilitar el desarrollo ágil de aplicaciones web robustas. Para la evaluación de los indicadores de rendimiento de la aplicación, se utilizó la herramienta JMeter. Esta decisión permitió llevar a cabo un análisis detallado y preciso del comportamiento del sistema bajo distintas condiciones de uso, garantizando que la aplicación desarrollada cumpliera con los estándares del modelo de calidad FURPS.

Al comparar los resultados obtenidos con los criterios establecidos por el modelo FURPS, se comprobó que el desempeño del sistema fue satisfactorio y acorde a las expectativas de rendimiento. En cuanto a los tiempos de respuesta, el sistema mostró un promedio de 2456 ms, situándose dentro de los límites aceptables según las directrices de FURPS. Además, el consumo promedio de recursos del sistema, incluyendo CPU, RAM y almacenamiento, fue del 15%, lo que refleja un uso eficiente y optimizado de los recursos disponibles.

Estos resultados subrayan la alta calidad y eficiencia del sistema desarrollado, validando la elección de Ruby on Rails como framework y confirmando la efectividad de las metodologías utilizadas para el desarrollo y evaluación de la aplicación. La capacidad del sistema para manejar múltiples solicitudes concurrentes sin errores, manteniendo tiempos de respuesta adecuados y un consumo moderado de recursos, es indicativa de su estabilidad y rendimiento robusto.

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Las investigaciones analizadas sobre diferentes implementaciones de Ruby on Rails han demostrado que este framework es altamente eficiente y versátil para el desarrollo de sistemas de inventario. Su estructura basada en Ruby ha facilitado la creación de una aplicación web robusta y escalable. Rails, con su ORM integrado y sistema de plantillas, ha sido la solución ideal para este proyecto, permitiendo un desarrollo rápido y organizado que se ajusta perfectamente a las necesidades específicas del sistema de inventario.
- En la implementación del sistema de inventario para American Lambster, se utilizó PostgreSQL para la gestión de la base de datos, mientras que en el frontend se emplearon plantillas y formularios de Rails, junto con lenguajes como JavaScript, HTML y CSS para manejar las interacciones. En el backend, el entorno de desarrollo de Rails fue aprovechado de manera eficiente, beneficiándose de su compatibilidad con metodologías ágiles como SCRUM. Este enfoque integral permitió una organización efectiva del proyecto, asegurando que cada componente del sistema funcione de manera cohesiva.
- Las pruebas de rendimiento realizadas con JMeter, fundamentadas en el modelo de calidad FURPS, demostraron que la aplicación web desarrollada cumple con todos los criterios establecidos. Con un total de 450 iteraciones distribuidas en 5 módulos, los resultados mostraron un tiempo de respuesta promedio de 2456 ms y un consumo de recursos del 15%. Estos resultados confirman que la aplicación es estable y capaz de manejar las operaciones solicitadas.
- Durante las pruebas de rendimiento, se observó que el sistema de inventario para American Lambster tiene un rendimiento adecuado, pero existe la posibilidad de que en escenarios de alta demanda o crecimiento en el volumen de datos, el sistema pueda experimentar una mayor carga. Por lo tanto, la escalabilidad del sistema es un factor crucial para asegurar su estabilidad a largo plazo.
- Dado que el sistema de inventario es una herramienta clave para la gestión operativa de American Lambster, es fundamental que los usuarios y administradores del sistema estén familiarizados con sus funcionalidades para asegurar un uso eficiente y efectivo. Esto también garantiza una mejor gestión del inventario y la toma de decisiones informadas dentro de la empresa.

5.2 Recomendaciones

- Dado el éxito de Ruby on Rails en este proyecto, se recomienda continuar utilizando este framework para futuras ampliaciones o desarrollos adicionales dentro del sistema de inventario. Además, se sugiere explorar nuevas funcionalidades y bibliotecas que el ecosistema de Rails ofrece, como herramientas de optimización de rendimiento y seguridad, para maximizar la eficiencia del desarrollo.

- Para garantizar una mayor cohesión y rendimiento en futuros proyectos, se recomienda realizar revisiones periódicas del código y la arquitectura del sistema, centrándose en el uso eficiente de PostgreSQL y Rails. También sería beneficioso optimizar las interacciones frontend-backend para asegurar que la comunicación entre ambos sea rápida y eficiente. Esto puede incluir mejoras en la carga asíncrona de datos y optimización de la gestión de formularios.
- Aunque los resultados de las pruebas fueron satisfactorios, se recomienda realizar una optimización adicional en las operaciones relacionadas con el listado de productos y materias primas, que mostraron picos significativos en los tiempos de respuesta. Se sugiere implementar técnicas de optimización de consultas SQL y revisar la eficiencia del código en estas áreas para asegurar un rendimiento más uniforme, especialmente en escenarios de alta demanda.
- Es aconsejable implementar herramientas de monitoreo continuo una vez que el sistema esté en producción. Esto permitirá identificar en tiempo real posibles problemas de rendimiento y ajustarse a la demanda de los usuarios. Además, se debería planificar la escalabilidad del sistema para asegurar que pueda manejar un aumento en el volumen de datos y usuarios sin comprometer su estabilidad y rendimiento.
- Se recomienda establecer un programa de capacitación continua para los usuarios y administradores del sistema. Este programa debe incluir sesiones periódicas de actualización sobre las nuevas funcionalidades, mejores prácticas en el uso del sistema y soporte técnico. Esto asegurará que todo el personal esté capacitado para aprovechar al máximo las capacidades del sistema, contribuyendo así a la optimización de la gestión del inventario.

BIBLIOGRAFIA

[1] D. Thomas, «Agile Web Development with Rails 6,» Pragmatic Bookshelf, 2019.

[2] M. Hartl, Ruby on Rails Tutorial (6th Edition): Learn Web Development with Rails, Addison-Wesley Professional, 2020.

[3] R. Miller y L. Tate, The Ruby on Rails Way, Addison-Wesley Professional, 2017.

[4] Askins, Ben, Gree, Alan. “A Rails/Django Comparison.” http://docs.google.com/ View?docid=dcn8282p_1hg4sr9.

[5] “Benefits of Using Open Source Software.” June 2009. http://open-source.gbdirect.co.uk/migration/benefit. html.

[6] “MVC: The Most Vexing Conundrum.” Slash 7. February 2, 2005.

[7] http://slash7.com/articles/2005/2/22/mvc the-most-vexing-conundrum.

[8] JRuby. July 2009. http://jruby.codehaus.org.

[9] Von Rotz, Bruno. “Open Source Year 2008 in Review: More Adoption, Success, Innovation, and Alterna tives”. Optaros. December 21, 2008. http://www. optaros.com/blogs/open-source-year-2008-review more-adoption-success-innovation-and-alternatives.

[10] Scrum Roles, http://www.agile42.com/, [Online], *Available*: http://www.agile42.com/en/agile-info-center/scrum roles/.

[11] Dr. Mark C. Paulk, Noopur Davis, " On Empirical Research Into Scrum, https://www.scrumalliance.org , [Online]. Available: https://www.scrumalliance.org/resource_download/989.

[12] Scrum https://www.mountaingoatsoftware.com. [Online].

[13] Ivan Bocic, Tevfik Bultan, "Symbolic Model Extraction for Web Application Verification", *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pp.724-734, 2017.

[14] Ivan Bocić, Tevfik Bultan, "Finding access control bugs in web applications with CanCheck", *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp.155-166, 2016.

[15] Takanori Kobashi, Hironori Washizaki, Nobukazu Yoshioka, Haruhiko Kaiya, Takao Okubo, Yoshiaki Fukazawa, "Designing Secure Software by Testing Application of Security Patterns", *Exploring Security in Software Architecture and Design*, pp.136, 2019.

[16] Micael Gallego-Carrillo, I. G.-A.-H. (2005). Applying Hierarchical MVC Architecture

To High Interactive Web Application . Third International Conference I.TECH - 2005, (pp. 1-6).

[17] Kingsley Sage. (n.d.). Model View Controller (MVC) architecture. 18-27

[18] Vedavyas J , Venkatesulu.S IJCET(2013) Volume 4 Issue 3-A Study of MVC – A Software Design Pattern for Web Application Development on J2EE Architecture

[19] Official Documentation of Laravel at http://www.laravel.com

[20] Officail Documentaion of Ruby on Rails at http://guides.rubyonrails.org/

ANEXOS

Anexo 1: Acta de entrega-recepción del sistema

AMERICAN LAMBSTER
IN GOD WE TRUST

CERTIFICADO

A petición de la parte interesada, yo Sr. Edison Sanisaca Yugcha administrador de la empresa American lambster encargado de la parte administrativa de nuestra empresa.

Certifico:

Que el señor **LUIS JAVIER YUGCHA TISALEMA**, portador de la cedula de ciudadania No **1805342886**, estudiante de la carrera de tecnologías de la información de la facultad de ingeniería de la Universidad Nacional de Chimborazo, entrego el sistema web como parte de su trabajo de titulación denominado: **APLICACIÓN WEB PARA LA GESTIÓN DE INVENTARIO DE LA EMPRESA "AMERICAN LAMBSTER" UTILIZANDO LA TECNOLOGÍA RUBY ON RAILS.**

Es todo cuanto puedo certificar en honor a la verdad. Facultando al interesado hacer uso del presente, para los fines que crea conveniente.

Ambato. 22 de octubre de 2024

.....

Sr. Edison Sanisaca

Administrador Americam Lambster

Anexo 2: Manual de usuario

Manual del sistema web de inventario para la empresa "American Lamsbter"

Versión 1.0

1. Introducción

Este manual de usuario proporciona una guía detallada para utilizar el sistema de inventario desarrollado en Ruby on Rails para American Lambster. Diseñado para facilitar la gestión de productos y materias primas, el sistema permite a los administradores y usuarios realizar tareas esenciales como el ingreso de datos, la generación de reportes y la consulta de inventarios de manera eficiente. A través de este manual, los usuarios encontrarán instrucciones claras y concisas sobre cómo navegar por las distintas funciones del sistema, asegurando un uso óptimo y eficaz de la plataforma. Este manual está dirigido tanto a usuarios nuevos como a administradores que buscan maximizar el rendimiento y la utilidad del sistema en su operativa diaria.

2. Visión general del sistema

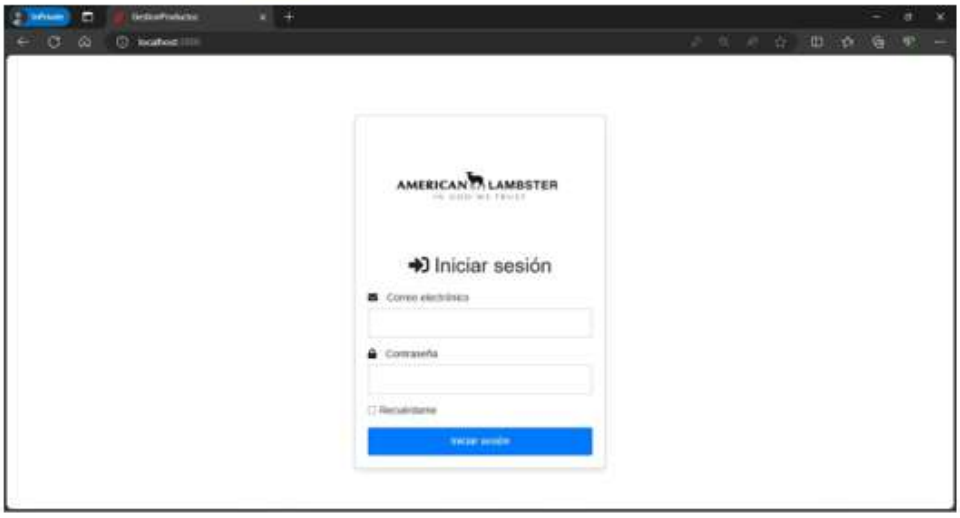
El sistema de inventario desarrollado para American Lambster tiene como objetivo optimizar la gestión de productos y materias primas, proporcionando a los administradores y usuarios una herramienta eficiente y fácil de usar. Basado en el framework Ruby on Rails, el sistema permite realizar funciones críticas como el registro de nuevos productos, la actualización de inventarios y la generación de reportes detallados. Además, el sistema está diseñado para asegurar una experiencia de usuario intuitiva, garantizando que tanto los administradores como los usuarios puedan acceder y gestionar la información de manera rápida y precisa.

El sistema se estructura en varios módulos clave, que incluyen el manejo de usuarios, el ingreso de productos y materias primas, y la generación de reportes. Cada módulo ha sido diseñado para maximizar la eficiencia y ofrecer una funcionalidad completa, con controles de acceso según el rol del usuario. El uso de la metodología ágil SCRUM durante el desarrollo ha permitido asegurar la adaptabilidad y escalabilidad del sistema, lo cual es esencial para responder a las necesidades cambiantes de American Lambster. Además, el sistema ha sido evaluado exhaustivamente en términos de rendimiento, garantizando así su estabilidad y confiabilidad en condiciones de uso intensivo.

3. Administrador

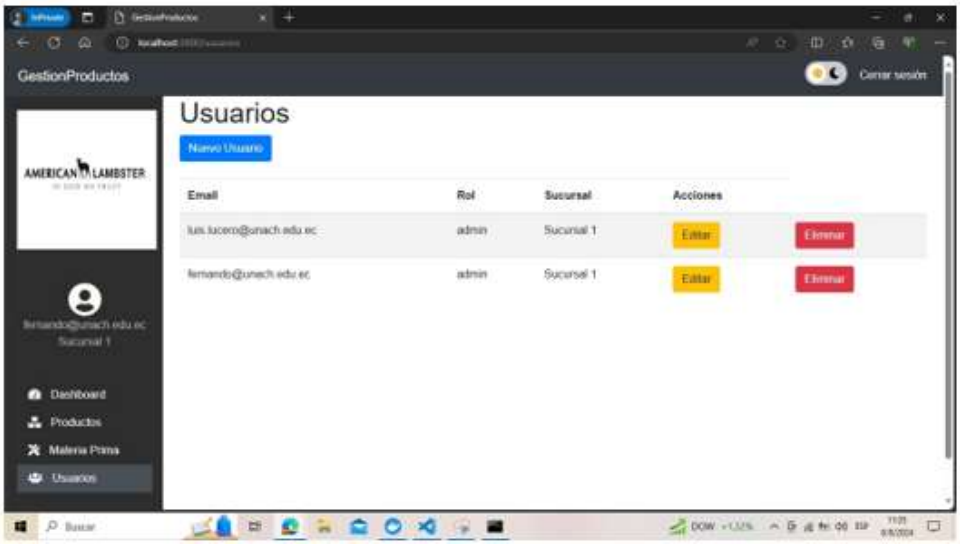
Pantalla principal

La pantalla principal del sistema, es la siguiente, donde se muestra el inicio de sesión



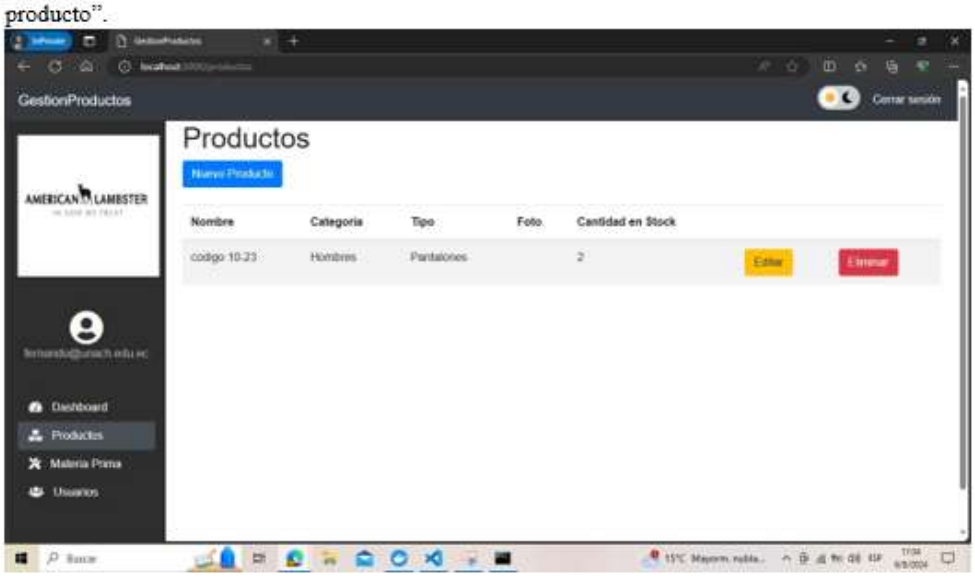
Acceso al panel de administrador

Al iniciar la sesión se accede al panel de administrador, donde se observa el e-mail, rol, sucursal y acciones.



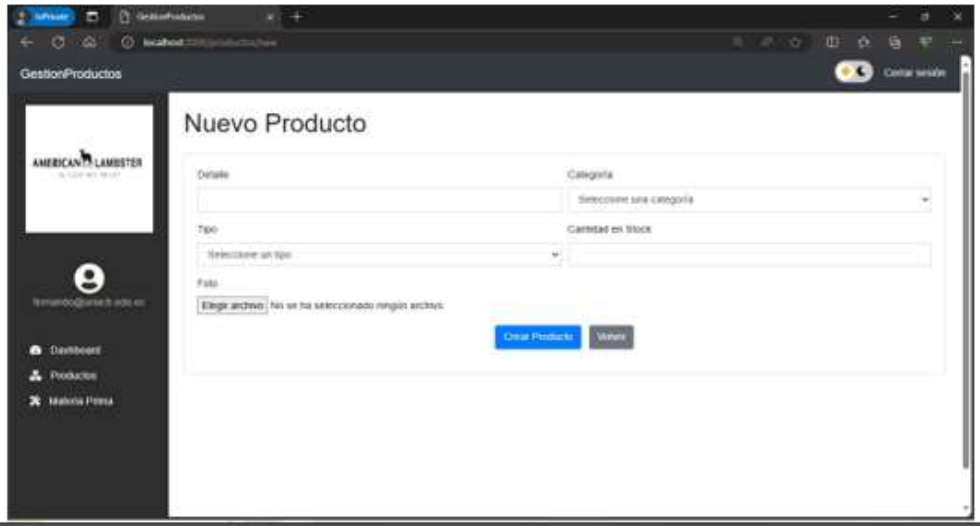
Ingresar productos

Al seleccionar la opción "Productos" se despliega la pantalla para el ingreso de "Nuevo producto".



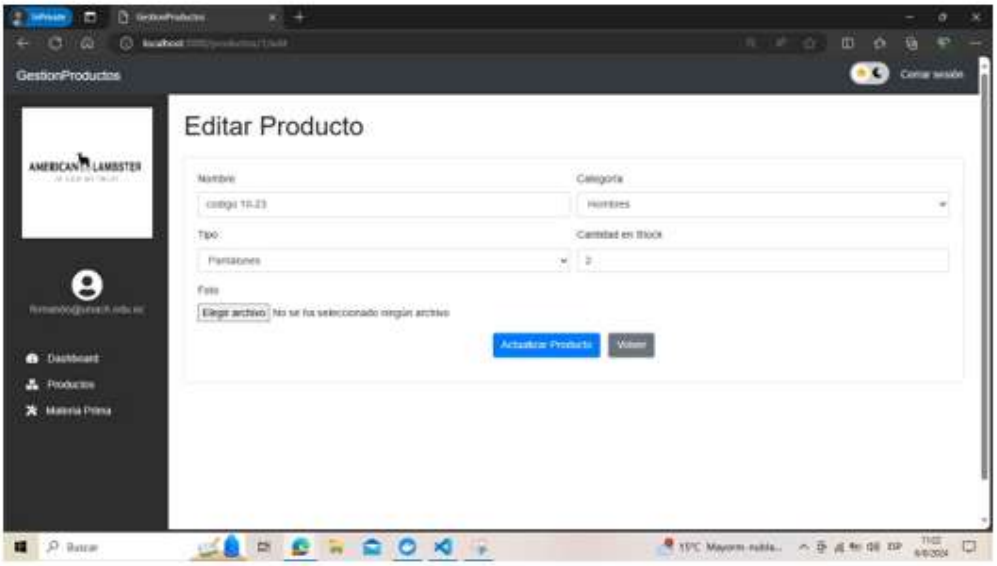
Nuevo Producto

La pantalla para ingresar Nuevo Producto, despliega la información a ingresar: Detalle, Categoría, Tipo, Foto, Cantidad en stock, finalmente se da clic en "Crear producto" para guardar la información.



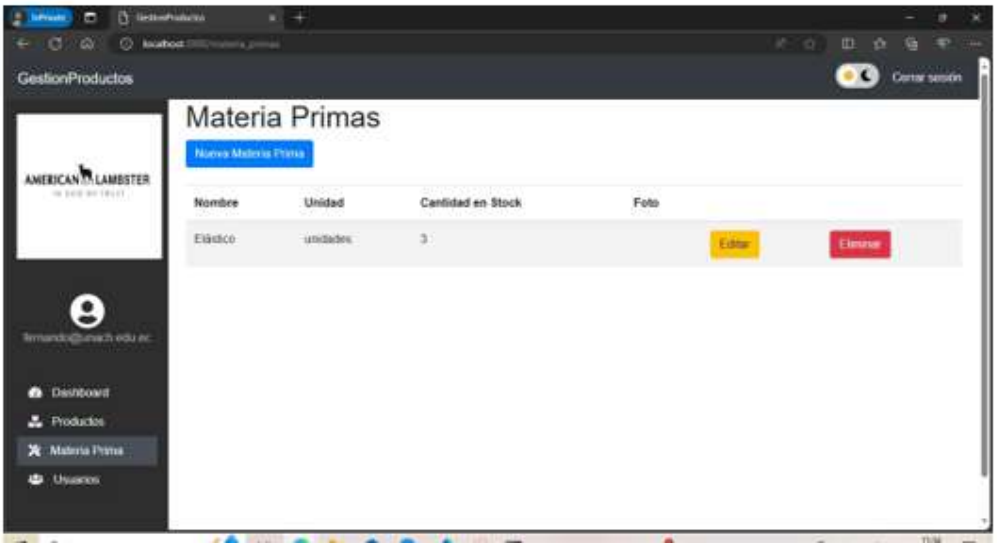
Actualizar producto

La pantalla para editar producto, despliega la información a ingresar: Detalle, Categoría, Tipo, Foto, Cantidad en stock, finalmente se da clic en "Actualizar producto" para guardar la información.



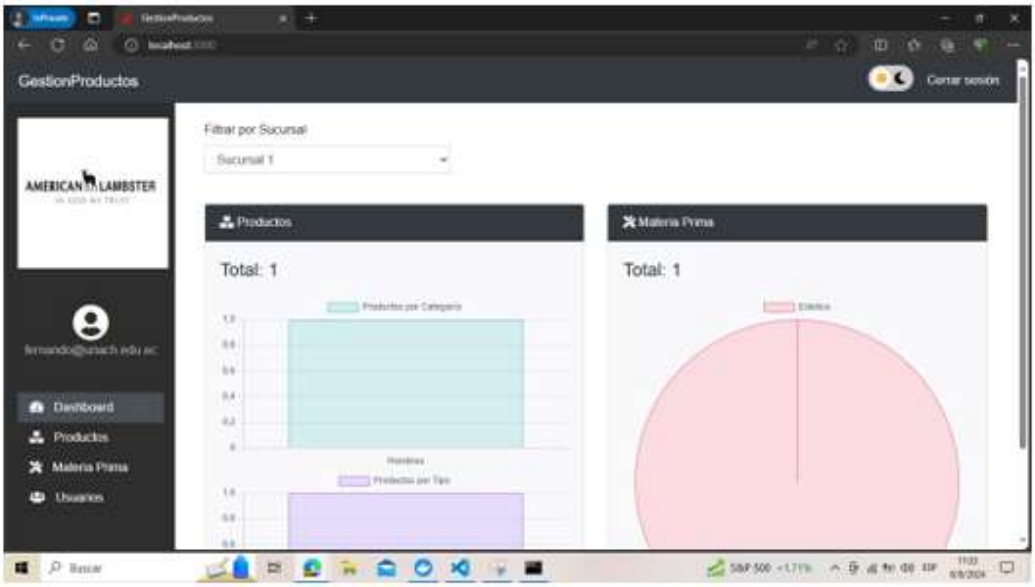
Ingresar Materia Prima

Al seleccionar la opción "Materia Prima" se despliega la pantalla para el ingreso de "Nueva Materia Prima", la información ha ingresar es: Nombre, Unidad, Cantidad en stock y Foto.



Generar reportes

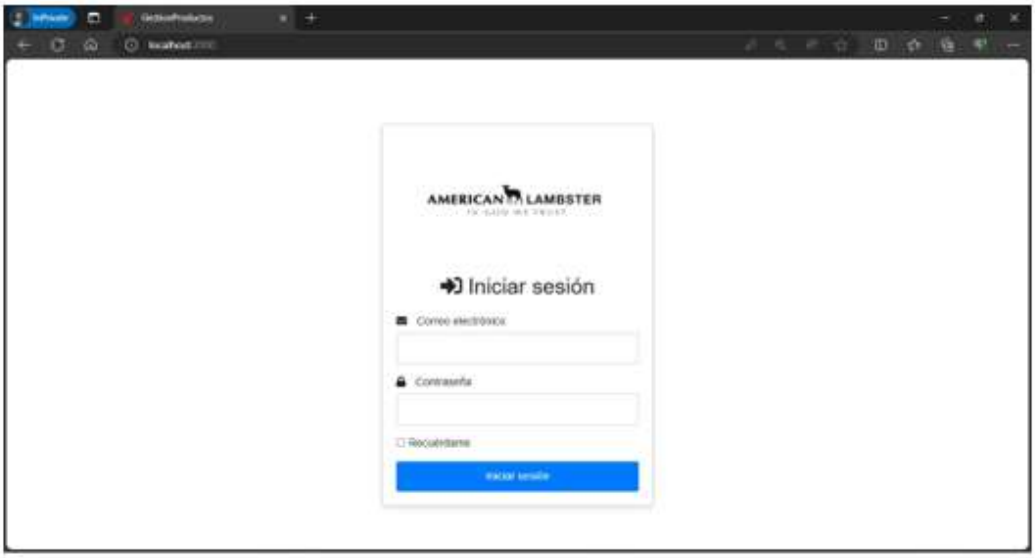
Al seleccionar la opción "Dashboard", el administrador puede usar el Filtro por sucursal para obtener gráficos de Productos y Materia prima.



4. Operario

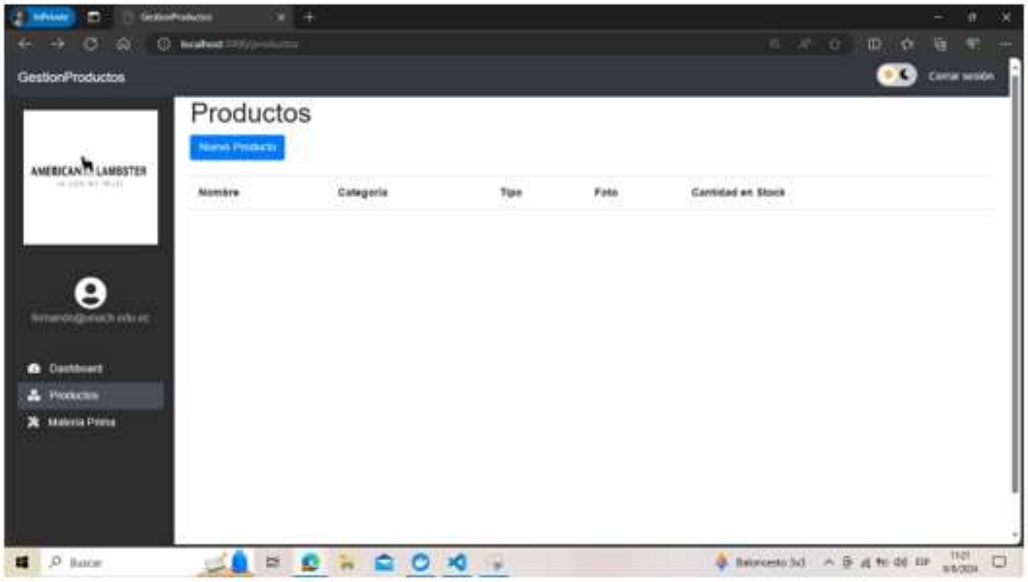
Inicio de sesión

El operario accede a la pantalla de inicio ingresa sus credenciales.



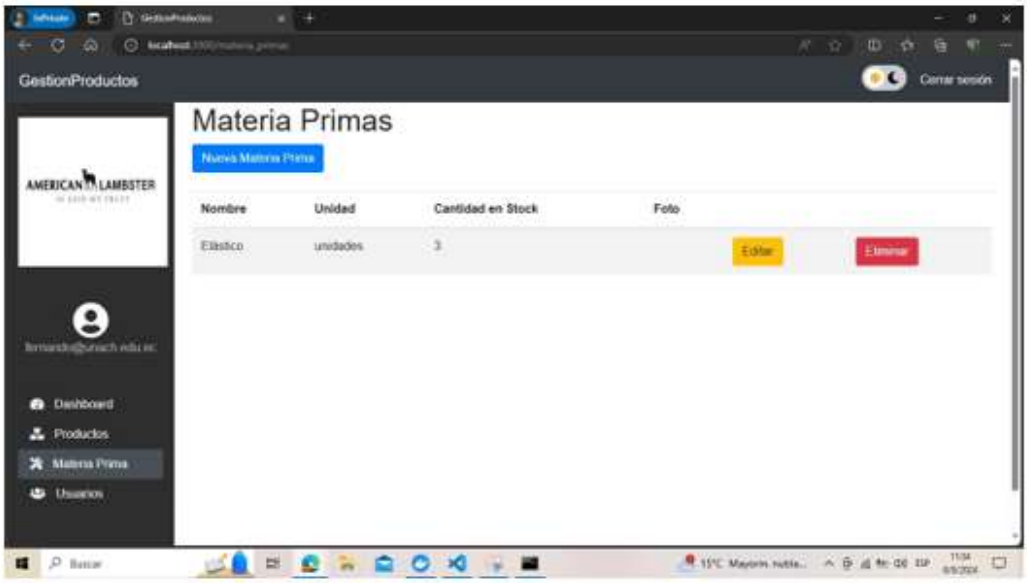
Visualización de productos disponibles

El operario puede visualizar la información de los productos



Visualización de materia prima disponible

El operario puede visualizar la información de la materia prima.



Dashboard de operario

El operario también puede visualizar el reporte de los productos y materia prima.