Desarrollo de un Aplicativo Web y móvil que permita la gestión de procesos y control del comedor de La Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE"

Vallejo Pacheco, Jonathan Alexander

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Tecnología Superior en Redes y Telecomunicaciones

Trabajo de Unidad de Integración Curricular, previo a la obtención del título de Tecnólogo Superior en Redes y Telecomunicaciones

Ing. Andaluz Espinosa, Diego Fernando

29 de febrero del 2024

Latacunga

Copyleaks

Plagiarism report

VALLEJO_JONATHAN _TESIS_FINAL.pdf

Scan details

Scan time:	Total Pages:	Total Words:
February 28th, 2024 at 16:37 UTC	63	15534

Plagiarism Detection

9.3%

Types of plagiarism		Words
Identical	8.7%	1356
Minor Changes	0%	0
Paraphrased	0%	0
Omitted Words	6.1%	949

AI Content Detection

N/A

Text coverage
- AI text
- Human text

Plagiarism Results: (112)

TEMA

Desarrollo de un Aplicativo Web y móvil que permita la gestión de procesos y control del comedor de La Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE"

Ing. Diego ~~Fernando~~ Andaluz ESpinosa
TUTOR

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Tecnología Superior en Redes y Telecomunicaciones

Certificación

Certifico que el trabajo de unidad de integración curricular: **"Desarrollo de un Aplicativo Web y móvil que permita la gestión de procesos y control del comedor de La Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE."** fue realizado por el Señor **Vallejo Pacheco, Jonathan Alexander**, el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Latacunga, 29 de febrero del 2024

Firma:

Ing. Diego Fernando, Andaluz Espinosa, Mgtr.
C. C. 0502166135

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Tecnología Superior en Redes y Telecomunicaciones

Responsabilidad de Autoría

Yo, **Vallejo Pacheco, Jonathan Alexander** con cédula de ciudadanía N°1724397144, declaro que el contenido, ideas y criterios del trabajo de unidad de integración curricular: **"Desarrollo de un Aplicativo Web y móvil que permita la gestión de procesos y control del comedor de La Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE"**, es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga 29 de febrero de 2024

Firma

Vallejo Pacheco, Jonathan Alexander

C.C.: 1724397144

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Tecnología Superior en Redes y Telecomunicaciones

Autorización de Publicación

Yo, **Vallejo Pacheco, Jonathan Alexander**, con cédula de ciudadanía N°1724397144, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de unidad de integración curricular: **"Desarrollo de un Aplicativo Web y móvil que permita la gestión de procesos y control del comedor de La Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Latacunga 29 de febrero de 2024

Firma

.....

Vallejo Pacheco, Jonathan Alexander

C.C.: 1724397144

Dedicatoria

Me permito dedicar primeramente el presente trabajo de investigación a Dios todopoderoso, siendo él, el que me da la fuerza para realizar todo lo que me propongo.

También agradezco a mi familia y mis familiares quiénes me han dado el apoyo y la ayuda para poder llevar a cabo esta investigación y que está haya sido terminada a tiempo con ayuda de sus esfuerzos y mi dedicación.

Vallejo Pacheco Jonathan Alexander

Agradecimiento

En primer lugar, doy gracias a Dios por permitirme continuar con mis estudios y darme la fuerza necesaria para concretar este proyecto también agradece agradezco a todas las personas que le han ayudado en especial a mi familia, quiénes han sido la fuerza que utilizado para continuar la investigación.

Agradezco a mi tutor de tesis quién fue la persona que se encargó de darme las ideas y a portarme correcciones para poder realizar un trabajo que sea factible y que cumpla con las funciones con las cuales está siendo desarrollado.

Vallejo Pacheco Jonathan Alexander

ÍNDICE DE CONTENIDO

Carátula **1**

Reporte de Verificación de Contenidos **2**

Certificación **3**

Responsabilidad de Autoría **4**

Autorización de Publicación **5**

Dedicatoria **6**

Agradecimiento **7**

Índice de Contenido **8**

Índice de Figuras **13**

Índice de tablas **15**

Resumen **16**

Abstract **16**

Capítulo I: Introducción **18**

Antecedentes **18**

Planteamiento del problema **19**

Justificación **20**

Objetivos **21**

Objetivo General ***21***

Objetivos Específicos ***21***

Alcance **22**

Capítulo II: Marco teórico **24**

Fundamentación legal **24**

Fundamentación Teórica 25

Procesos 25

Definición 25

Tipos de procesos 25

Gestión de procesos 26

Modelamiento de procesos 26

Procesos Productivos 27

Calidad 27

Definición 27

Gestión de la calidad 28

La Industria de la Restauración 28

Servicios Gastronómicos 28

Optimización de recursos en un restaurante 29

Software 29

Definición 29

Importancia del Software 29

Clasificación del Software 29

Software Libre 30

Arquitectura Cliente-Servidor 30

Aplicaciones Desktop 31

Aplicaciones Web 31

Web estáticas 31

Web Dinámicas 32

Esquema básico de un servicio web 33

Desarrollo Web *34*

FrontEnd *34*

BackEnd *35*

Diseño web responsivo *36*

Desarrollo web *36*

Herramientas de desarrollo web *37*

Framework de Desarrollo *37*

Laravel *38*

Manejo de datos en Laravel con Eloquent *39*

Manejo de vistas con Blade *39*

Composer *40*

Artisan *40*

Estructura de laravel *40*

Servidor Web *42*

Base de datos *43*

Postgress *44*

Docker *45*

Cómo funciona Docker: *45*

Capítulo III: Desarrollo y resultados **46**

Metodología **46**

Modalidad de investigación *46*

Propuesta de implementación **47**

Materiales *47*

Encuesta *47*

Entrevista *48*

Tabulación de datos..... 49

Resultados Encuesta..... 49

Tabulación datos Encuesta..... 54

Análisis de resultados de la encuesta..... 57

Resultado datos entrevista..... 59

Métodos..... 60

Método de Investigación..... 60

Modelo de desarrollo de software..... 60

Análisis de requerimientos..... 64

Requerimientos funcionales..... 64

Requerimiento no funcionales..... 65

Requerimiento de hardware y software..... 65

Casos de Uso..... 67

Diagrama de Clases..... 71

Diagrama de red..... 72

Diseño de la Base de Datos..... 73

Modelo Lógico..... 73

Modelo Físico..... 74

Diccionario de datos..... 75

Diseño e interfaz del sistema..... 76

Estándar de la interfaz de Usuario..... 76

Interfaces del aplicativo web..... 77

Estructura del proyecto web..... 79

Migraciones..... 80

Modelos *84*

Controladores *89*

Rutas *94*

Implementación **96**

Control de versiones **100**

Pruebas **101**

Capítulo IV: Conclusiones y Recomendaciones **105**

Conclusiones **105**

Recomendaciones **106**

Bibliografía **107**

Anexos **111**

ÍNDICE DE FIGURAS

Figura 1 *Modelo de Sistema de Producción* 27

Figura 2 *Estructura Cliente-Servidor*..... 31

Figura 3 *Arquitectura Web Estáticas* 32

Figura 4 *Arquitectura de una Web Dinámica* 33

Figura 5 *Frameworks de desarrollo* 38

Figura 6 *Estructura de proyecto con laravel*..... 42

Figura 7 *Pregunta 1* 50

Figura 8 *Pregunta 2* 50

Figura 9 *Pregunta 3* 51

Figura 10 *Pregunta 4* 51

Figura 11 *Pregunta 5* 52

Figura 12 *Pregunta 6* 52

Figura 13 *Pregunta 7* 53

Figura 14 *Pregunta 8* 53

Figura 15 *Pregunta 9* 54

Figura 16 *Pregunta 10* 54

Figura 17 *Diagrama de procesos* 64

Figura 18 *UML General del comedor*..... 68

Figura 19 *UML de usuario Militar* 69

Figura 20 *UML usuario Administrador* 70

Figura 21 *UML del Ranchero o cocinero* 71

Figura 22 *Diagrama de Clases*..... 72

Figura 23 *Diagrama de red* 73

Figura 24 *Modelo Físico de BDD* 75

Figura 25 *Interfaz de Login* 78

Figura 26 *Interfaz de Panel Principal* 79

Figura 27 *Estructura del proyecto en Laravel 10x* 79

Figura 28 *Migraciones* 80

Figura 29 *Modelos* 85

Figura 30 *Controladores* 89

Figura 31 *Docker Desktop* 99

Figura 32 *Logs Docker* 99

Figura 33 *Despliegue de la Aplicación* 100

Figura 34 *Repositorio Github* 101

ÍNDICE DE TABLAS

Tabla 1 *Encuesta a usuarios* 47

Tabla 2 *Entrevista* 49

Tabla 3 *Resultados Entrevista* 59

Tabla 4 *Detalle de UML usuario Militar* 69

Tabla 5 *Detalle UML usuario Administrador* 70

Tabla 6 *UML usuario Cocinero* 71

Tabla 7 *Diccionario de Datos* 75

Tabla 8 *Estándar de desarrollo de interfaces* 76

Resumen

En la Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE", la gestión del comedor enfrenta desafíos significativos. Los reportes de gastos de comida muestran inconsistencias, hay un exceso de saldo de comida desperdiciado y a veces se realizan cobros innecesarios. La falta de un sistema digitalizado dificulta el control del inventario de alimentos y la gestión de los servicios de comedor, generando pérdida de recursos y disminuyendo la eficiencia operativa. El propósito de este proyecto es implementar un sistema de aplicativo web y móvil para la gestión del comedor en la AGRUCOMGE. Se busca optimizar las operaciones diarias, garantizando un manejo eficiente de los recursos alimenticios y estableciendo un acceso digital en línea a la información del comedor. Basándonos en investigaciones previas sobre sistemas de gestión de activos y control de inventarios, se plantea desarrollar un sistema que permita un control interno eficiente, ofreciendo una interfaz intuitiva y generando alertas para una gestión más efectiva de los recursos. La metodología incluye el análisis de requerimientos, el diseño e implementación del sistema, y su posterior evaluación. Se utilizarán técnicas de desarrollo de software y se realizarán pruebas exhaustivas para garantizar su funcionamiento óptimo. El sistema implementado ha permitido un seguimiento en tiempo real de la disponibilidad de alimentos, un registro detallado de las comidas prestadas y devueltas, y una planificación efectiva para evitar desperdicios y cobros indebidos. La implementación del sistema ha resultado en una mejora significativa en la eficiencia del proceso de gestión del comedor en la AGRUCOMGE.

Palabras clave: aplicativo web, bases de datos, gestión de procesos, agrupación AGRUCOMGE,

Abstract

In the Communications and Electronic Warfare Group "AGRUCOMGE," the management of the dining facility faces significant challenges. Food expense reports exhibit inconsistencies, there is an excess of wasted food balance, and unnecessary charges are occasionally incurred. The absence of a digitized system hampers the control of food inventory and dining facility management, resulting in resource loss and diminished operational efficiency. The purpose of this project is to implement a web and mobile application system for dining facility management in AGRUCOMGE. The aim is to optimize daily operations, ensuring efficient handling of food resources, and establishing digital online access to dining facility information. Drawing from previous research on asset management systems and inventory control, the intention is to develop a system enabling efficient internal control, offering an intuitive interface, and generating alerts for more effective resource management. The methodology involves requirement analysis, system design and implementation, followed by evaluation. Software development techniques will be utilized, and exhaustive testing conducted to ensure optimal functionality. The implemented system has facilitated real-time tracking of food availability, detailed record-keeping of meals borrowed and returned, and effective planning to mitigate waste and unnecessary charges. The implementation of the system has resulted in a significant improvement in the efficiency of dining facility management processes in AGRUCOMGE.

Keywords: web application, databases, process management, AGRUCOMGE grouping.

Capítulo I

Introducción

Antecedentes

Considerando el avance constante de las tecnologías y la creciente necesidad de digitalizar y optimizar los sistemas de gestión de activos en diversas entidades, se han desarrollado sistemas que abarcan desde aplicaciones de escritorio hasta aquellas ejecutables en navegadores web o dispositivos móviles. Estas aplicaciones, desde sus inicios centradas en el control de activos (entrada y salida del almacén), han evolucionado hasta la actualidad, integrándose con bases de datos remotas y ofreciendo una variedad de funcionalidades, como calendarios y alertas, entre otras.

Al analizar investigaciones anteriores, Morales y Torres (2015) llevaron a cabo el diseño de un sistema de inventario para la empresa Service Lunch, basándose en el modelo COSO 1. Este modelo, un estándar para el control interno, permitió evaluar y mejorar los sistemas en función de las observaciones operativas (Morales Catherin, 2015). Los resultados aseguraron un aumento en la eficiencia y productividad de la empresa en la gestión de insumos y recursos.

Rodríguez (2016) enfocó su investigación en la empresa MAGREB S.A., señalando que los procesos industriales carecían de una sistematización adecuada, lo que resultaba en pérdidas significativas y baja eficiencia (Rodriguez, 2016). Actualizando y optimizando el sistema de control de inventarios, se logró una mejora en el uso de la materia prima, reflejándose en beneficios financieros y una gestión más eficaz de los recursos.

Quizhpi Campoverde (2018) desarrolló e implementó un sistema de inventario que controla de manera permanente productos y clientes, aplicando criterios específicos para cada uno (Quizhpi, 2018). Destacable en esta investigación fue la consideración de criterios de seguridad, implementando un servidor local para resguardar la información privada de productos y clientes.

Carreño Dueñas et al. (2019) sugirieron la optimización del sistema de gestión en una empresa Pymes, especialmente en el área de producción de dulces, aunque el sistema contaba con una memoria de cálculo para dosificación y control, se propuso expandir y mejorar el sistema según los resultados obtenidos en la producción de dulces (Carreño Dueñas et al., 2019).

En base a estas investigaciones previas, se plantea implementar un sistema que permita realizar un control interno para mejorar la eficiencia del proceso de comedor de la Agrupación de Comunicaciones y Guerra Electrónica “AGRUCOMGE”. El sistema debe ofrecer una interfaz intuitiva asi como también la información debe estar disponible para diferentes reportes con la finalidad de toma de decisiones oportunas.

Planteamiento del problema

Los sistemas de gestión para el control y seguimiento de procesos son cruciales para mantener el orden y eficiencia en diversas áreas, tal como se evidencia en el caso de la Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE". Actualmente, se enfrenta a desafíos en la administración del comedor, donde se registran inconsistencias en los reportes de gasto de comidas, excesivo saldo de comida desperdiciado y cobros de comidas a veces innecesarias.

En la actualidad, el manejo de los procesos del comedor se realiza de forma manual, lo que dificulta un control total sobre el inventario de alimentos y la gestión de los servicios de comedor. La falta de un sistema digitalizado impide un seguimiento en tiempo real de la disponibilidad de alimentos, el registro de comidas prestadas y devueltas, así como la planificación efectiva para evitar desperdicios y cobros indebidos.

Este problema se agrava con la falta de alertas acerca de fechas de vencimiento de comidas o posibles excesos de inventario. La revisión manual de registros se convierte en una

tarea laboriosa y propensa a errores, lo que afecta negativamente el rendimiento del comedor y genera gastos innecesarios.

Si no se resuelve el problema de la ineficiente administración del comedor en la Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE", los informes de gastos de comida seguirán presentando inconsistencias, se continuará desperdiciando un exceso de saldo de comida y se seguirán realizando cobros innecesarios, lo cual afectará negativamente la operatividad del comedor. La falta de un sistema digitalizado seguirá dificultando el control del inventario de alimentos, la gestión de los servicios de comedor y la planificación efectiva para evitar desperdicios y cobros indebidos, lo que podría resultar en una mayor pérdida de recursos y una disminución en la eficiencia operativa de la agrupación.

Justificación

La implementación del sistema de aplicativo web y móvil para la gestión de procesos y control del comedor en la Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE" representa una medida esencial para optimizar las operaciones diarias y garantizar un manejo eficiente de los recursos alimenticios. Al adoptar esta solución tecnológica, se establecerá un acceso digital en línea a la información del comedor, lo que facilitará una administración ágil y precisa de los procesos vinculados a las comidas del personal militar.

Este enfoque digital es especialmente relevante al considerar la necesidad de llevar un control de las personas disponibles para acceder a las comidas, con énfasis en las obligatorias como el desayuno y almuerzo, y la confirmación previa para las opcionales como la merienda. El sistema permitirá, mediante credenciales previamente registradas, gestionar eficientemente la información del personal autorizado, así como llevar un registro detallado de las comidas solicitadas y consumidas.

La aplicación web y móvil ofrecerá opciones intuitivas para realizar confirmaciones de asistencia, llevar un seguimiento de las personas que han accedido al comedor, y generar informes detallados sobre las comidas servidas. Además, se podrán establecer alertas para gestionar la preparación de las comidas, asegurando que el personal encargado tenga la información necesaria para planificar y preparar las porciones requeridas.

Los beneficiarios directos de esta iniciativa son el personal militar de la Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE", quienes experimentarán una mejora significativa en la eficiencia del proceso de gestión del comedor. Este control efectivo de los procesos alimenticios contribuirá a la disposición estratégica de los recursos, garantizando una distribución óptima y eficaz.

Además, este sistema, con su diseño intuitivo y capacidad de generar informes detallados, puede ser adaptado para su implementación en otras áreas, generando bases de datos específicas y ajustando los parámetros del sistema según las necesidades particulares de cada contexto. Así, la tecnología propuesta no solo mejorará la gestión del comedor, sino que también establecerá un modelo eficaz para la optimización de procesos en otras áreas de la agrupación militar.

Objetivos

Objetivo General

Desarrollar una aplicación web y móvil que permita la gestión de procesos y control del comedor del Agrupamiento de Comunicaciones y Guerra Electrónica, “AGRUCOMGE”.

Objetivos Específicos

- Realizar un estudio detallado del proceso actual, identificando los puntos críticos mediante trabajo colaborativo con el encargado del comedor.

- Definir los requisitos de interfaz de usuario para la aplicación web y diseñar prototipos utilizando herramientas de diseño de wireframes.
- Definir casos de prueba que cubran todos los aspectos funcionales y de usabilidad de la aplicación mediante pruebas de integración y aceptación por parte de los usuarios finales.

Alcance

El desarrollo del sistema de aplicativo web y móvil para la gestión de procesos y control del comedor en la Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE" tiene como objetivo principal mejorar la eficiencia operativa y el control de los recursos alimenticios. En este contexto, se establecen tres roles principales con funcionalidades específicas:

- **Usuarios (Personal Militar):**
  - ✓ Confirmar asistencia al comedor.
  - ✓ Seleccionar comidas según horario de consumo (Desayuno, almuerzo y merienda).
  - ✓ Recibir ticket o Código de asistencia.
  - ✓ Recibir reportes de sus consumos.
- **Administrador (Oficial de administración de personal):**
  - ✓ Verificar los cupos disponibles de cada una de las comidas.
  - ✓ Controlar la validez de los tickets
  - ✓ Habilitar el ticket y confirmar asistencia del personal militar.
  - ✓ . Emitir reporte de número de personas que asistirán a cada una de las comidas.
- **Cocinero (Ranchero):**

- ✓ Recibir la información de número de personas por cada comida
- ✓ Indicar disponibilidad de cupos para comidas.

El sistema proporcionará una navegación fluida entre opciones como:

- Confirmación y devolución de asistencia.
- Generación de reportes generales o por tipo de comida.
- Revisión de alertas del comedor.
- Manipulación de datos desde la base de datos.
- Respaldos digitales de información.

El presente proyecto no aborda el manejo de inventario y control de ingredientes y sus costos. Se prioriza la optimización de recursos al evitar cocinar en exceso en base a la confirmación de asistencia del personal militar y la toma oportuna de decisiones en base a reportes.

Capítulo II

Marco teórico

Fundamentación legal

Existen varias normas y reglamentos en la legislación ecuatoriana que abordan aspectos como protección de datos personales, derechos de autor, leyes de accesibilidad, seguridad informática entre otras. El presente proyecto ha considerado varios de estos aspectos para determinar la factibilidad de implementar un sistema web fundamentado en base a principios legales y jurídicos.

En la ley de protección de datos personales Art. 3, manifiesta que la presente ley se aplicara en cualquier parte del territorio nacional(Asamblea Nacional, 2021). Siendo la AGRUCOMGE un estamento militar, es pertinente considerar este reglamento ya que se manejará información personal.

El Art. 4 de la ley de protección de datos personales define a una base de datos como *"Conjunto estructurado de datos cualquiera que fuera la forma, modalidad de creación, almacenamiento, organización, tipo de soporte, tratamiento, procesamiento, localización o acceso, centralizado, descentralizado o repartido de forma funcional o geográfica"* (Asamblea Nacional, 2021). En este sentido el aplicativo web gestionara los datos y la información en una base de datos digital, la cual se debe manejar con los aspectos de confidencialidad, disponibilidad y seguridad de la información.

La ley de propiedad intelectual del Ecuador declara en su Art. 7 que software es *"Toda secuencia de instrucciones o indicaciones destinadas a ser utilizadas, directa o indirectamente, en un dispositivo de lectura automatizada, ordenador, o aparato electrónico o similar con capacidad de procesar información, para la realización de una función o tarea, u obtención de un resultado determinado, cualquiera que fuere su forma de expresión o fijación"*(Congreso Nacional, 2006). En este contexto el desarrollo del proyecto se ha realizado considerando

recursos open source y de uso libre, con la finalidad de precautelar el uso indebido de software sin licenciamiento.

Partiendo de la lógica de la interconexión persona-máquina, es imprescindible el uso del internet como catalizador para el desarrollo global, más aún cuando se ha definido al internet como el recurso esencial para el desarrollo de las sociedades modernas (Santos, 2022). En este sentido, se debe hacer uso correcto de las herramientas tecnológicas como son los aplicativos web. Existen varios tratados y marco regulatorio en el cual se deben enmarcar los productos software.

Fundamentación Teórica

Procesos

Definición

El proceso es considerado como el conjunto de eventos necesarios para realizar una actividad, sea de forma individual o grupal(Mendoza-Fernandez & Sobeida Moreira-Chóez, 2021). En informática, un proceso refiere a distintas combinaciones operativas que ocurren simultáneamente para alcanzar un resultado o un producto (Hossian, 2012).

Tipos de procesos

Estratégicos: procesos destinados a definir y controlar las metas de la empresa, sus políticas y estrategias. Estos procesos son gestionados directamente por la alta dirección en conjunto (Zaratiegui, 1999).

Operativos: procesos destinados a llevar a cabo las acciones que permiten desarrollar las políticas y estrategias definidas para la empresa para dar servicio a los clientes. De estos procesos se encargan los directores funcionales, que deben contar con la cooperación de los otros directores y de sus equipos humanos (Zaratiegui, 1999).

De apoyo: procesos no directamente ligados a las acciones de desarrollo de las políticas, pero cuyo rendimiento influye directamente en el nivel de los procesos operativos (Zaratiegui, 1999).

Gestión de procesos

La gestión de procesos es la visión sistémica actual a una visión futura en la organización, con el fin de obtener una sinergia en los conceptos de gestión y procesos (Bravo Carrasco, 2011).

La Gestión de Procesos es una metodología corporativa y de disciplina de gestión, cuyo objetivo es mejorar el desempeño (eficiencia y eficacia) y la optimización de cada uno de los procesos de negocio de una organización, a través de la gestión de los procesos que se deben diseñar, modelar, organizar, documentar y optimizar de forma continua (Maldonado, 2018). En este caso se desea mejorar el proceso de comedor en el AGRUCOMGE.

Modelamiento de procesos

Curtis et al. (1992), afirman que existen cuatro puntos de vista en cuanto al modelado de los procesos de negocio: vista funcional (qué), la cual representa la dependencia funcional entre los elementos del proceso; vista dinámica (cuándo, cómo), que proporciona una secuenciación y control de la información sobre el proceso; vista informacional, que incluye la descripción y relación entre las entidades que son producidas, consumidas o incluso manipuladas por los procesos, y la vista organizacional (quién, dónde) que describe quién desarrolla cada tarea o función y dónde se desarrolla dentro de la organización.

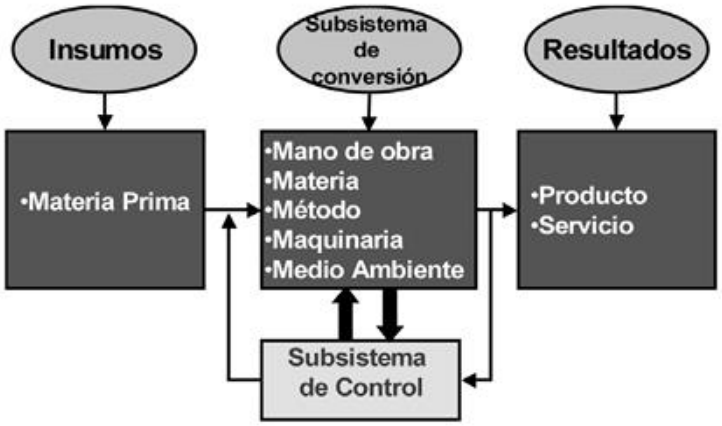
El modelar los procesos dentro de la organización, permite conocer las áreas problemáticas y susceptibles a mejoras, los niveles y la delegación de autoridad, las áreas de alto riesgo, el volumen de sus operaciones y el ciclo de vida de sus procesos, incluyendo el contenido tecnológico y la problemática social (Ríos & Leticia, 2001).

Procesos Productivos

Un proceso de producción recibe insumos en forma de materiales, personal, capital, servicios e información, y los transforma dentro de un subsistema de conversión en los productos y/o servicios deseados. Además, existen unos productos indirectos que se suelen pasar por alto. Los sistemas productivos generan impuestos, desperdicios, contaminación, empleos, sueldos, y adelantos tecnológicos; estos son algunos ejemplos de productos indirectos de un sistema (Tejeda, 2011). En el presente proyecto se busca la implementación del producto de software para optimizar los recursos en el proceso de comedor y menorar los desperdicios de alimentos al tener un control de cuanta comida cocinar.

Figura 1

Modelo de Sistema de Producción



Nota. El gráfico representa un modelo de un sistema de producción. Tomado de (Tejeda, 2011)

Calidad

Definición

Según (Bravo Carrasco, 2011b), la calidad se comprende como la creación de valor con la máxima eficiencia, lo cual implica la permanente búsqueda de brindar valor útil al cliente anticipándose a sus necesidades y reduciendo la cantidad de recursos y esfuerzo durante su elaboración.

Gestión de la calidad

Es un sistema que relaciona un conjunto de variables relevantes para la puesta en práctica de una serie de principios, prácticas y técnicas para la mejora de la calidad. Así pues, el contenido de los distintos enfoques de Gestión de la Calidad se distingue por tres dimensiones.(Camisón Zornoza et al., 2007)

1. Los principios que asumen y que guían la acción organizativa.

2. Las prácticas –actividades– que incorporan para llevar a la práctica estos principios.

3. Las técnicas que intentan hacer efectivas estas prácticas (Camisón Zornoza et al., 2007).

La Industria de la Restauración

Con el término restauración nos referimos a los negocios dedicados a la elaboración de comidas y bebidas y que son preparadas para su consumo. Incluiríamos pues diferentes tipos de negocios como restaurantes, casas de comidas, cafeterías, bares, mesones, vinotecas, bodega, chiringuitos, etc.

La restauración hoy en día se considera y se incluye dentro de la actividad turística ya que es un complemento, como lo menciona Tamames (2010) quien también aduce que es “aquella actividad que se desarrolla en establecimientos abiertos al público, y que consiste en ofrecer habitualmente y mediante precio, servicio de comidas y bebidas, para su consumo en el mismo local, independientemente de que esta actividad se desarrolle de forma principal” (Chancay, 2015).

Servicios Gastronómicos

De acuerdo con el criterio de (Arizmendi ,2014) los servicios gastronómicos “Es la mera forma de satisfacer al comensal de manera que todo funciona como un engranaje, desde su principal necesidad hasta el buen funcionamiento de la empresa”(Chancay, 2015).

Optimización de recursos en un restaurante

La gestión de compras en los restaurantes es una pieza fundamental para el éxito y la rentabilidad de estos establecimientos. Los restaurantes, ya sean pequeños establecimientos locales o cadenas internacionales de renombre, se enfrentan a una serie de desafíos en su proceso de compras (Macías, 2023). Estos desafíos pueden manifestarse de varias formas, desde fluctuaciones en los precios de los alimentos y la disponibilidad de productos hasta la falta de control sobre los consumos diarios. Estos problemas pueden erosionar los márgenes de ganancia y afectar negativamente la satisfacción del cliente y, en última instancia, poner en peligro la viabilidad del negocio.

Software

Definición

El software es el equipamiento lógico e intangible de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware (Pacienzia, 2015).

Importancia del Software

El software es ahora la clave del éxito de muchas empresas y negocios, ya que sin él sería casi imposible el mantenimiento y crecimiento de estos. Lo que diferencia una compañía de otra es la suficiencia, exactitud y oportunidad de la información dada por el software (Pacienzia, 2015).

Clasificación del Software

El software se clasifica según su función en:

Software de sistema. Programas que dan al usuario la capacidad de relacionarse con el sistema, para ejercer control sobre el hardware. El software de sistema también se ofrece como soporte para otros programas. Por ejemplo: sistemas operativos o servidores.

Software de programación. Programas diseñados como herramientas que le permiten a un programador desarrollar programas informáticos. Se valen de técnicas y un lenguaje de programación específico. Por ejemplo: compiladores o editores multimedia.

Software de aplicación. Programas diseñados para realizar una o más tareas específicas a la vez, pueden ser automáticos o asistidos. Por ejemplo: videojuegos o reproductores multimedia.

Software Libre

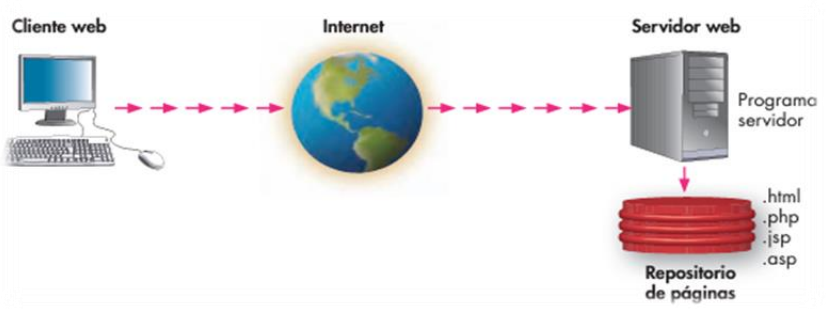
El "software libre" es una cuestión de libertad, no de precio. Para comprender este concepto, debemos pensar en la acepción de libre como en "libertad de expresión" y no como en "barra libre de cerveza". Con software libre nos referimos a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Nos referimos especialmente a cuatro clases de libertad para los usuarios de software (Stallman, 2019):

Arquitectura Cliente-Servidor

La tecnología Cliente/Servidor se refiere a una arquitectura de diseño de software de aplicación que es el resultado de la subdivisión de un sistema de información, en conjunto de procesos servidores, generalmente especializados, que pueden ejecutarse en variadas plataformas, tanto hardware como software, y que provee a un gran número de procesos cliente, sobre diferentes plataformas físicamente interconectadas por una red local o de área extendida, utilizando uno o varios protocolos de comunicación (Pacheco & Rosero, 1999).

Figura 2

Estructura Cliente-Servidor



Nota. Muestra la estructura Cliente-Servidor. Tomado de (Pacheco & Rosero, 1999)

Aplicaciones Desktop

Varios autores manifiestan que las aplicaciones de escritorio son el conjunto de programas o herramientas que tenemos instalados en nuestros ordenadores de sobremesa o portátiles y que únicamente podemos usar en dichos dispositivos. No los podemos trasladar a otro distinto, por lo que poseen esta limitación.

Aplicaciones Web

La arquitectura de las aplicaciones web consta de máquinas conectadas a una red, por lo general, Internet o una Intranet corporativa que sigue el esquema cliente-servidor en nuestro caso de servidores web. Surgió a mediados de la década de 1990, durante la etapa de la Web 1.0 con la aparición de las primeras conexiones de acceso conmutado (RTC, RDSI, GSM, GPRS) y de las etiquetas multimedia del estándar HTML y la incorporación de pequeños programas realizados en Java, llamados applets (Lerma-Blasco et al., 2013).

Web estáticas

Como es conocido, toda web tradicional (estática) es un conjunto de páginas HTML entrelazadas de alguna forma entre sí y que contienen referencias a otros elementos dependientes: ficheros de imágenes, sonido, video, hojas de estilo, scripts…. Ante una petición

de una página por parte de un usuario, el servidor que aloja la web se limitará a buscarla y, si la encuentra, a remitirla al ordenador peticionario con el resto de los elementos referenciados. Una vez recibidos, el navegador del solicitante se encargará de interpretar (presentar visualmente) el código HTML (véase ilustración 1). Por tanto, existe una correspondencia entre los archivos o páginas HTML que se encuentran en el servidor y los que recibe e interpreta el cliente (Torres-Del-Rey & Rodríguez-V, 2014).

Figura 3

Arquitectura Web Estáticas

Nota. Muestra la estructura de las web estáticas. Tomado de (Gutiérrez Gallardo, 2010)

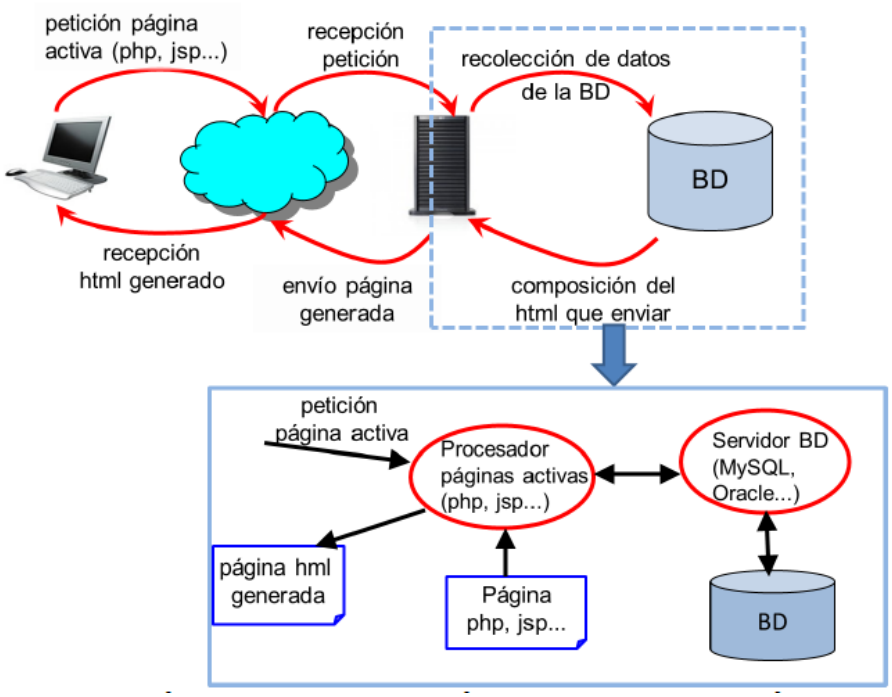
Web Dinámicas

Una web dinámica como un conjunto de datos almacenados en una base de datos (BD), gestionada por la capa de software conocida como sistema de gestión de base de datos (SGBD) y, por otro lado, un conjunto de programas conocidos como páginas activas (en lenguaje PHP, ASPX, JSP) con su correspondiente software procesador o intérprete del lenguaje, que constituyen la interfaz para que los usuarios del sitio manipulen la BD (Torres-Del-Rey & Rodríguez-V, 2014).

Todo el contenido de una web dinámica que los usuarios editores incorporan está de alguna forma alojado en una BD, y las páginas activas se encargan, a través del entorno más o menos sencillo o ergonómico que ofrecen (menús, botones, iconos, cuadros de texto…), de acceder a la BD para reconstruir una página web solicitada por un usuario; o bien modificar aquélla y, por ende, ésta, si el usuario que interactúa a través de un navegador tiene privilegios para ello (Torres-Del-Rey & Rodríguez-V, 2014).

Figura 4

Arquitectura de una Web Dinámica



Nota. Muestra la estructura de las web dinámicas. Tomado de (Gutiérrez Gallardo, 2010)

Esquema básico de un servicio web

- ✓ Clientes: Son quienes hacen las peticiones (navegador web).
- ✓ Servidores: Programas que se ejecutan continuamente en un ordenador, a la espera de que les lleguen las solicitudes de los clientes y responden mediante una página web (servidor web).

- ✓ Aplicaciones de software: Mecanismos que utilizan los clientes y los servidores para comunicarse (navegador web).
- ✓ Lenguaje de programación: Forma de crear las aplicaciones software (C++).
- ✓ Red de ordenadores: conjunto de ordenadores y/o dispositivos conectados por medio de cables, señales, ondas o cualquier otro método de transporte de datos que comparten información (Internet).
- ✓ Protocolos: Conjunto de reglas utilizadas en la comunicación entre los clientes y los servidores HTTP (HyperText Transfer Protocol), diseñado para transferir páginas HTML.
- ✓ Estándares: Conjunto de reglas normalizadas que describen los requisitos que deben ser cumplidos por un producto, proceso o servicio, con el objetivo de establecer un mecanismo base para permitir que distintos elementos hardware o software que lo utilicen sean compatibles entre sí (Camazón, 2009).

Desarrollo Web

En el ámbito del desarrollo web, el FrontEnd está conformado por todas aquellas tecnologías que corren del lado del cliente, es decir, todas aquellas tecnologías que corren del lado del navegador web, generalizándose más que nada en tres lenguajes, Html, CSS y JavaScript. La persona encargada del FrontEnd, trabaja con estas tres tecnologías, aunque esto no significa que desconozca como trabaja el BackEnd (lado servidor), ya que es necesario para que pueda consumir datos y pueda estructurar correctamente un maquetado en HTML y CSS para su mejor comodidad (Graciela et al., 2021).

FrontEnd

FrontEnd se encarga de estilizar la página de tal manera que la misma pueda presentar la información de forma agradable para el usuario. El responsable del FrontEnd, debe de conocer las técnicas de experiencia de usuario para brindar una mejor interacción entre la

persona y la página que visita, así mismo debe tener conocimientos de diseño de Interacción para colocar los elementos de tal manera que el usuario las pueda ubicar de forma rápida y cómoda.

Las aplicaciones FronEnd, son aquellas que se utilizan en el lado Cliente, que se utilizan en los diferentes dispositivos que utilizados para conectarse con el servidor a través de internet. Esta tecnologías y lenguajes de programación vienen implementados en los distintos navegadores web que existen, que son interpretadores de estos códigos. Las tecnologías más utilizadas en el Frontend son HTML, CSS, JavaScript, jQuery, Ajax, BootStrap, Angular, etc (Aranda, 2018).

BackEnd

Se denomina BackEnd a la capa de acceso a los datos de un software que no es accesible para el usuario final. Además, esta capa contiene toda la lógica de la aplicación que maneja los datos. Cabe destacar que los datos de una aplicación se encuentran almacenados en una base de datos dentro de un servidor. El responsable del BackEnd es la persona que trabaja del lado del servidor y debe formarse como desarrollador de aplicaciones web o como desarrollador de aplicaciones multiplataforma. Debe estudiar los diferentes lenguajes de programación que son necesarios para desarrollar su trabajo y según la empresa en la que trabaje serán necesarios unos u otros. Además, necesita conocer las interacciones con diferentes bases de datos, saber las diferencias entre estas y cualidades de las más utilizadas.

Las aplicaciones BacKend, son aquellas que se utilizan en el lado Servidor, las que utiliza el Servidor para realizar gestión de peticiones de información que le llegan y para gestionar las bases de datos alojadas en los mismos. La información una vez tratada le es devuelta al dispositivo para que sea visualizada en el dispositivo a través de las tecnologías Frontend. Las tecnologías más utilizadas en el Backend son PHP, Java, .NET, Phyton, MySQL, etc (Aranda, 2018).

Diseño web responsivo

Actualmente, el uso generalizado de smartphones y tablets, y por qué no incluir también los smart TV han obligado a que los sitios web se ajusten a las diferentes resoluciones de visualización. De esta manera, un factor principal es la necesidad de acceso desde un dispositivo inteligente a sitios web (Arce, 2016).

El término Responsive Web Design fue creado por Ethan Marcotte en 2008. El concepto One Web había partido del World Wide Web Consortium (W3C) en 2008 y hace referencia a construir una web para todos (Web for all), permitiendo la accesibilidad a esta única web desde cualquier dispositivo (web for everything). Uno de los sitios web considerado como insignia del Responsive Web Design es el de The Boston Globe, al ser uno de los primeros grandes sitios en internet en adoptar el cambio y también por ser partícipe Ethan Marcotte en el equipo de desarrolladores encargados del rediseño (Arce, 2016).

Desarrollo web

Los estándares Web, definidos por el grupo W3, son las respuestas más eficaces a la rápida y continua evolución tecnológica que experimenta la red. Adecuarse a ellos hace posible que el trabajo de hoy constituya una base efectiva en el futuro y ayude a evolucionar tecnológicamente con el medio.

Estándares definidos por el grupo W3C, como CSS, XHTML, JavaScript y el DOM W3C, permiten a los diseñadores de interfaz tener un mayor control sobre el diseño, la disposición y la tipografía de las páginas web, así como desarrollar comportamientos sofisticados que funcionen en diferentes navegadores y plataformas. Además, garantizan la accesibilidad, facilitan la separación de estilo, estructura y comportamiento, y aseguran la compatibilidad con navegadores actuales y futuros.

Herramientas de desarrollo web

Hoy en día los desarrolladores de software suelen utilizar aplicaciones que permiten realizar de forma rápida y sencilla el diseño y codificación de interfaces gráficas de usuario. Algunos de los IDE de desarrollo más conocidos son Visual Studio Net, NetBeans, entre otros. También se suelen utilizar Editores de código, los cuales, si bien no son IDE de desarrollo, facilitan al programador sus tareas de desarrollo, entre estos tenemos Visual Studio Code, Atom, Sublime Text.

Los lenguajes de programación utilizados para desarrollar software basado en la web son de tipo intérprete; es decir, son lenguajes que analizan el programa fuente y lo ejecutan directamente utilizando otro programa que normalmente es un explorador de Internet. Los intérpretes no generan código equivalente al lenguaje de máquina; dentro de los lenguajes de programación utilizados para la web se encuentran, HTML, JavaScript, PHP, Python, Cobol, ASP.NET, entre otros(Arbeláez, 2011) .

Framework de Desarrollo

Se describe como un recurso que proporciona un marco de organización y sistematización para el desarrollo de aplicaciones o recursos informáticos, facilitando la creación de proyectos y mejorando la operatividad de los diversos equipos involucrados.

Uno de los aspectos relevantes al momento de elegir un framework de desarrollo de software es determinar la curva de aprendizaje que requiere. En los proyectos de desarrollo de software, en general, los desarrolladores disponen de poco tiempo para entregar un producto estable y usualmente requieren de frameworks de desarrollo que tengan una curva de aprendizaje baja. Actualmente, existen diversos frameworks para soportar el desarrollo web; sin embargo, elegir el más adecuado puede ser una tarea compleja, debido a que los criterios de selección pueden ser diversos, poco claros e incluso inexistentes (Pantoja & Pardo, 2016).

Figura 5

Frameworks de desarrollo

N°	Nombre del Framework
1	Laravel
2	Django
3	VueJS
4	React
5	Ruby on Rails
6	Node.js
7	Angular
8	Express
9	jQuery
10	Electron
11	ExtJS
12	Jenkins
13	EmberJS
14	Knockout
15	A-Frame
16	Mithril
17	Mybatis
18	Codeigniter
19	BackboneJS
20	CakePHP
21	Spring
22	SSM framework
23	VAADIN Framework
24	Aurelia
25	ZK

Nota. Lista de frameworks de desarrollo más utilizados en el 2020. Tomado de (Espinosa-Hurtado, 2021)

Laravel

Framework Laravel es un framework de código abierto basado en el lenguaje PHP, este posee una filosofía muy clara, que está enfocada para que el código desarrollado sea lo más expresivo y elegante posible, para desarrollar aplicaciones y servicios web. Laravel propone una forma de desarrollar aplicaciones web de un modo mucho más ágil, simplificando el trabajo

con tareas comunes como la autenticación, el enrutamiento, gestión sesiones, entre otras (Cabrera León, 2019).

Laravel es un framework de desarrollo de PHP y podemos crear aplicaciones web fácilmente con esta aplicación. Utiliza el popular patrón de diseño MVC (modelo–vista–controlador) y está basado en el sistema Symfony. Laravel utiliza un sistema de paquetes modular, por lo tanto, podemos expandir nuestra aplicación con nuevos módulos. Reutiliza varios componentes existentes de otros marcos, lo que ayuda a crear una aplicación operable y segura rápidamente. Admite el acceso a diferentes bases de datos; presenta utilidades que ayudan en el desarrollo de aplicaciones web; motor de enrutamiento simple y rápido; la aplicación web se vuelve más escalable; se ahorra considerable tiempo en el diseño. Laravel es un software de código abierto, con licencia MIT(Subecz, 2021).

Manejo de datos en Laravel con Eloquent

Laravel incluye una valiosa pieza de software, llamada Eloquent ORM. Este ORM se funda en patrón active record y su funcionamiento es en extremo sencillo. Un ORM (Object Relational Mapper) en PHP es un software que permite tratar la capa de persistencia de los datos, como simples accesos a métodos de una Clase u Objeto en PHP. La funcionalidad interna del ORM es mapear los objetos de PHP a las tablas en la base de datos, para el caso en que la persistencia de los datos de la aplicación es proporcionada por una DB (Ovando, 2019).

Manejo de vistas con Blade

Manejo de Vistas en Laravel El manejo de las vistas es un punto demasiado importante si hablar de realizar una aplicación web, Laravel incluye de paquete un sistema de procesamiento de plantillas llamado Blade. Este sistema de plantillas, Blade favorece un código mucho más limpio en las Vistas, además de incluir un sistema de Caché que lo hace más

rápido. El sistema Blade de Laravel, permite usar una sintaxis más reducida en su escritura. Por ejemplo, en vez de pintar la vista usando el código PHP (Ovando, 2019).

Composer

Composer es una herramienta de gestión de dependencias para PHP. Permite a los desarrolladores de PHP declarar e instalar las bibliotecas externas (llamadas "paquetes") que sus proyectos necesitan. Esto simplifica en gran medida la gestión de las dependencias de un proyecto PHP, ya que Composer maneja automáticamente la descarga de las bibliotecas necesarias, sus versiones y sus dependencias.

Al utilizar Composer, los desarrolladores pueden definir las dependencias de su proyecto en un archivo llamado composer.json, donde especifican qué paquetes son necesarios y en qué versiones. Luego, al ejecutar el comando composer install, Composer leerá este archivo y descargará automáticamente las bibliotecas requeridas en la carpeta del proyecto.

Composer también gestiona las dependencias de forma recursiva, lo que significa que si un paquete depende de otros paquetes, Composer descargará automáticamente todas las dependencias necesarias para garantizar que el proyecto funcione correctamente.

Artisan

Es una herramienta de línea de comandos incluida en Laravel que facilita varias tareas comunes de desarrollo de aplicaciones. Artisan proporciona una serie de comandos predefinidos que permiten realizar acciones como la generación de controladores, modelos, migraciones de base de datos, semillas de datos, entre otros.

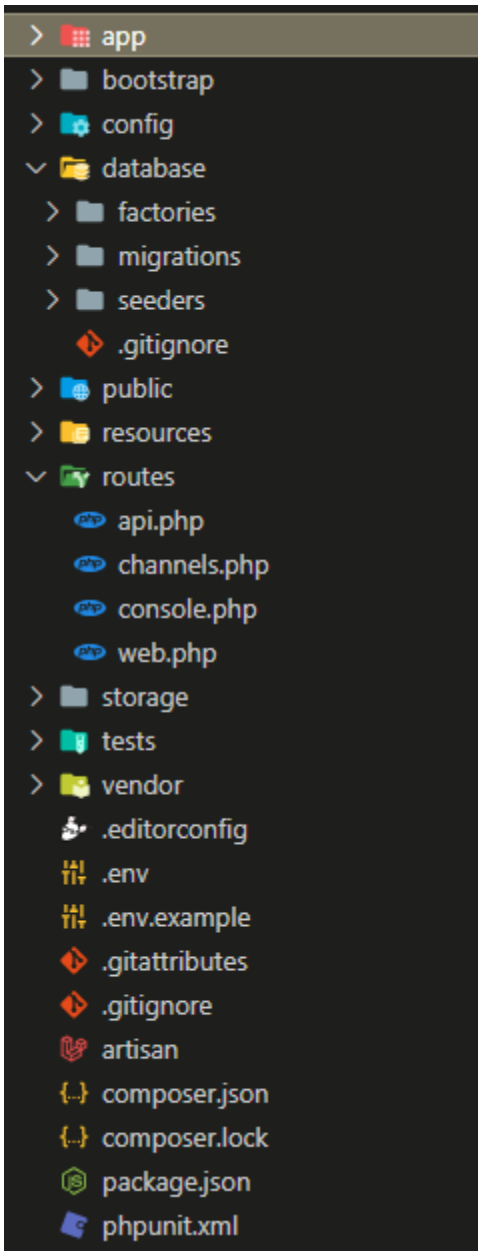
Estructura de laravel

A continuación, se muestra una estructura básica de un proyecto con laravel y las carpetas más importantes:

- ✓ app/: Contiene la lógica de la aplicación, incluyendo controladores, modelos, middleware y proveedores de servicio.
- ✓ bootstrap/: Contiene los scripts de inicio de la aplicación.
- ✓ config/: Almacena archivos de configuración de la aplicación.
- ✓ database/: Contiene las migraciones de base de datos, los factories para la generación de datos de prueba y los seeders para poblar la base de datos con datos iniciales.
- ✓ public/: Es el directorio raíz accesible públicamente. Aquí se encuentran los activos estáticos, como archivos CSS, JavaScript e imágenes.
- ✓ resources/: Contiene archivos de recursos que serán compilados o procesados, como archivos de vistas Blade, archivos de idioma y archivos Sass.
- ✓ routes/: Contiene archivos de definición de rutas para la aplicación.
- ✓ storage/: Contiene archivos generados por la aplicación, como archivos de sesiones, archivos cargados y archivos de caché.
- ✓ tests/: Contiene archivos de pruebas automatizadas para la aplicación.
- ✓ vendor/: Es el directorio donde Composer instala las dependencias de la aplicación.
- ✓ .env: Archivo de configuración de entorno que contiene variables de entorno específicas de la instalación.
- ✓ .env.example: Archivo de ejemplo de configuración de entorno.
- ✓ artisan: Es el script de consola de Laravel para ejecutar comandos de Artisan.
- ✓ composer.json y composer.lock: Archivos de configuración y bloqueo de Composer para gestionar las dependencias del proyecto.
- ✓ README.md: Archivo de documentación del proyecto.

Figura 6

Estructura de proyecto con laravel



Nota. Estructura básica de un proyecto con laravel.

Servidor Web

.

Los servidores web utilizan diversos protocolos de datos para la comunicación, siendo el más común o estándar el HTTP (Protocolo de Transferencia de Hipertexto) o su variante segura HTTPS (Protocolo de Transferencia de Hipertexto Seguro). Cabe mencionar que el término "servidor web" también puede referirse al ordenador en el que se almacenan los archivos que componen un sitio web, junto con el software necesario para facilitar la conexión de datos web. (MOSELEY, 2007).

El servidor web Apache HTTP es el servidor web más utilizado en el mundo. Apache proporciona muchas características poderosas, incluyendo módulos cargables dinámicamente, un sólido soporte multimedia y una amplia integración con otro software popular. Apache es un tipo de servidor web que puede ejecutarse en varios sistemas operativos, como Microsoft Windows, Linux, Unix, Novell Netware y otras plataformas que se utilizan para servir y gestionar instalaciones web utilizando un protocolo conocido como HTTP (Protocolo de Transferencia de Hipertexto).. Inicialmente, Apache era un software de código abierto utilizado únicamente como reemplazo del servidor web Netscape. Sin embargo, Apache ha crecido en popularidad desde abril de 1996. Hasta mayo de 1999, Apache se utilizaba ampliamente en una variedad de servidores web en todo el mundo. Ahora, Apache es uno de los software de código abierto, lo que significa que es respaldado por desarrolladores de todo el mundo, lo que hace que Apache sea más mantenido y actualizado regularmente con nuevas características y funcionalidades para mejorar la calidad de la entrega de servicios HTTP (Haris, 2022).

Base de datos

Una base de datos constituye un repositorio de información de diversa índole, cuya organización se ajusta a los requisitos del programador o usuario final. Esta organización permite establecer enlaces y relaciones coherentes al realizar consultas o modificar la información almacenada. En una base de datos, es posible almacenar una amplia variedad de

datos, que van desde textos, números y fechas hasta archivos con extensiones como ".pdf", ".mp4" y ".jpge", entre otras.

Las bases de datos están vinculadas a las aplicaciones web, lo que facilita la modificación de las tablas presentes en la base de datos y mantiene el archivo de origen en el servidor. Una de sus principales ventajas radica en la indexación de la información, lo que agiliza la búsqueda y obtención de resultados de manera eficiente. (Pisco Gómez, 2017).

Postgress

El Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos conocido como PostgreSQL (y brevemente llamado Postgres95) está derivado del paquete Postgres escrito en Berkeley. Con cerca de una década de desarrollo tras él, PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl y python). Postgres, desarrollada originalmente en el Departamento de Ciencias de la Computación de la Universidad de California en Berkeley, fue pionera en muchos de los conceptos de bases de datos relacionales orientadas a objetos que ahora empiezan a estar disponibles en algunas bases de datos comerciales. Ofrece suporte al lenguaje SQL: 2003, integridad de transacciones, y extensibilidad de tipos de datos. PostgreSQL es un descendiente de dominio público y código abierto del código original de Berkeley (López, 2016).

Docker

Docker es una plataforma de contenedores que facilita la creación, implementación y gestión de aplicaciones. En lugar de virtualizar el hardware como las máquinas virtuales, Docker virtualiza el sistema operativo, lo que lo hace más ligero y eficiente.

Beneficios de Docker:

Portabilidad: Las aplicaciones se ejecutan en contenedores que son portables entre diferentes entornos (desarrollo, pruebas, producción).

Eficiencia: Los contenedores son más ligeros que las máquinas virtuales y consumen menos recursos.

Escalabilidad: Los contenedores se pueden escalar fácilmente para aumentar o disminuir la capacidad según sea necesario.

Aislamiento: Las aplicaciones se ejecutan en un entorno aislado, lo que reduce el riesgo de conflictos y errores.

Cómo funciona Docker:

1. Imágenes: Se crea una imagen que contiene el código de la aplicación, las dependencias y las configuraciones necesarias. Las imágenes se pueden compartir y almacenar en repositorios como Docker Hub.

2. Contenedores: A partir de una imagen, se crea un contenedor que es una instancia ejecutable de la aplicación. Los contenedores se pueden iniciar, detener, eliminar y escalar según sea necesario.

3. Docker Engine: El motor de Docker se instala en cada máquina donde se desea ejecutar contenedores. El motor proporciona comandos para crear, iniciar, detener y gestionar contenedores.

Capítulo III

Desarrollo y resultados

Metodología

Modalidad de investigación

La investigación se basa en las siguientes modalidades:

Investigación Cualitativa: Se considera que el proyecto es un tipo de investigación cualitativa por que se realizaron de encuestas y la recolección de opiniones de expertos, los cuales ayudaron a una comprensión profunda y detallada de los procesos en el AGUCROMGE. Las entrevistas en persona o en línea con los usuarios (Personal militar), ranchero y oficial de administración, proporcionan información valiosa sobre sus experiencias, percepciones y sugerencias para mejorar el proceso.

Investigación Cuantitativa: También se considera como una investigación cuantitativa, ya que, el análisis de datos cuantitativos de las encuestas puede proporcionar información sobre tendencias, patrones y estadísticas relevantes para entender mejor el funcionamiento del comedor en el AGUCROMGE y las necesidades de los clientes. Por ejemplo, se podría analizar datos sobre tiempos de espera, satisfacción del cliente, frecuencia de pedidos, desperdicios entre otros.

Investigación de Desarrollo: Dado que también estás contemplando el desarrollo de un aplicativo web para gestionar el proceso en el restaurante, se considera como una investigación de desarrollo ya que implica la creación de prototipos, pruebas de usabilidad y evaluación iterativa del aplicativo para garantizar que cumpla con las necesidades y expectativas de los usuarios.

Propuesta de implementación

Materiales

Los materiales utilizados para el proceso de recolección de la información fueron mediante encuestas, las cuales estaban dirigidas a las entidades que intervienen en el proceso del comedor del AGUCROMGE. Esto con el fin de obtener las necesidades de los usuarios y de esta manera implementar el sistema web.

Encuesta

Se necesita obtener datos cualitativos, los cuales luego de ser procesados nos permitan orientar nuestra investigación. En la tabla 1, se muestra una matriz con las preguntas que se aplicaran a los usuarios.

Tabla 1

Encuesta a usuarios

Nro. de Pregunta	Pregunta	Opciones
1	¿Con qué frecuencia utilizas el comedor de AGRUCOMGE?	a) Diariamente b) Varias veces por semana c) Ocasionalmente d) Nunca
2	¿Qué aspectos te parecen positivos del actual sistema de gestión del comedor?	a) Variedad de opciones b) Horarios convenientes c) Calidad de alimentos d) Rapidez en el servicio e) Otro (Especificar)
3	¿Qué aspectos consideras problemáticos o ineficientes en el sistema actual de gestión del comedor?	a) Inconsistencias en reportes de gastos b) Desperdicio de alimentos c) Cobros innecesarios d) Dificultad para seguir el consumo e) Otro (Especificar)
4	¿Qué funciones consideras esenciales en un sistema de gestión del comedor?	a) Registro de consumos b) Alertas de vencimiento c) Planificación de menús d) Control de inventario e) Otro (Especificar)
5	¿Qué características te gustaría ver en un sistema digitalizado para el comedor?	a) Interfaz fácil de usar b) Acceso remoto c) Notificaciones en tiempo real d) Personalización de menús e) Otro (Especificar)

Nro. de Pregunta	Pregunta	Opciones
6	¿Qué información consideras más importante para tener acceso en tiempo real en relación con el comedor?	a) Disponibilidad de alimentos b) Registros de comidas prestadas y devueltas c) Alertas de vencimiento d) Otro (Especificar)
7	¿Has experimentado alguna inconsistencia en los reportes de gasto de comidas o en los cobros del comedor?	a) Sí b) No
8	¿Has notado desperdicio de alimentos en el comedor? ¿Cómo crees que se podría reducir este desperdicio?	a) Sí, limitando las porciones b) Sí, mejorando la planificación de menús c) Sí, implementando medidas de control de inventario d) No, no he notado desperdicio
9	¿Has tenido dificultades para realizar seguimiento de tus consumos en el comedor?	a) Sí b) No
10	¿Crees que la implementación de un sistema digitalizado ayudaría a solucionar los problemas actuales? ¿Por qué?	a) Sí, facilitaría el seguimiento y la gestión b) Sí, reduciría errores y desperdicios c) No, no estoy seguro d) No, creo que no resolvería los problemas existentes

Nota. Matriz de preguntas utilizadas para la encuesta.

Entrevista

Una técnica de recolección de informacion es la entrevista. En este proyecto se aplica entrevista al personal involucrado con la finalidad de obtener información detallada acerca del proceso de comedor, a través de testimonios directos de los especialistas en la gestión del comedor. Esta entrevista nos permitirá la identificación de necesidades y requisitos, identificación de problemas y desafíos, generación de ideas para funcionalidades del aplicativo, validación de la propuesta. En la tabla 2, se puede observar un cuestionario con una serie de preguntas las cuales se aplicaron para identificación de requerimientos funcionales del aplicativo.

Tabla 2

Entrevista

Preguntas	Respuestas	Observaciones
¿Cuál es tu nombre y cuál es tu rol en la Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE"?		
¿Puedes describir cómo es el proceso actual de administración del comedor?		
¿Cómo se lleva a cabo el control del inventario de alimentos y la gestión de los servicios de comedor?		
¿Has notado alguna inconsistencia en los reportes de gasto de comidas? Si es así, ¿puedes proporcionar algunos ejemplos?		
¿Has observado algún desperdicio de comida o cobros indebidos? Si es así, ¿puedes proporcionar algunos ejemplos?		
¿Cómo se manejan las fechas de vencimiento de las comidas y los posibles excesos de inventario?		
¿Qué desafíos has enfrentado debido a la falta de un sistema digitalizado?		
¿Cómo crees que un sistema digitalizado podría mejorar la administración del comedor?		
¿Hay algo más que te gustaría compartir sobre tu experiencia en la administración del comedor?		

Nota. Cuestionario aplicado en la entrevista.

Tabulación de datos

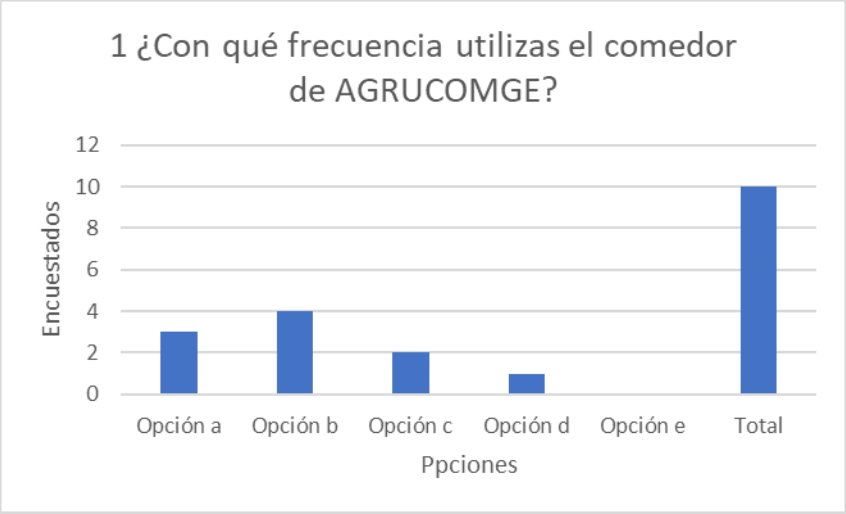
Resultados Encuesta

Se procede a analizar y tabular la información de la encuesta, en donde se muestran los siguientes resultados:

Pregunta 1. ¿Con qué frecuencia utilizas el comedor de AGRUCOMGE?

Figura 7

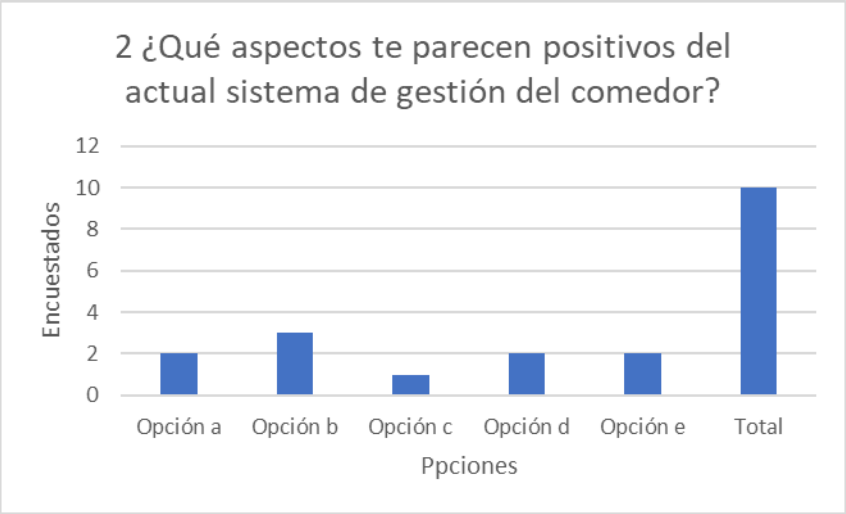
Pregunta 1



Pregunta 2. ¿Qué aspectos te parecen positivos del actual sistema de gestión del comedor?

Figura 8

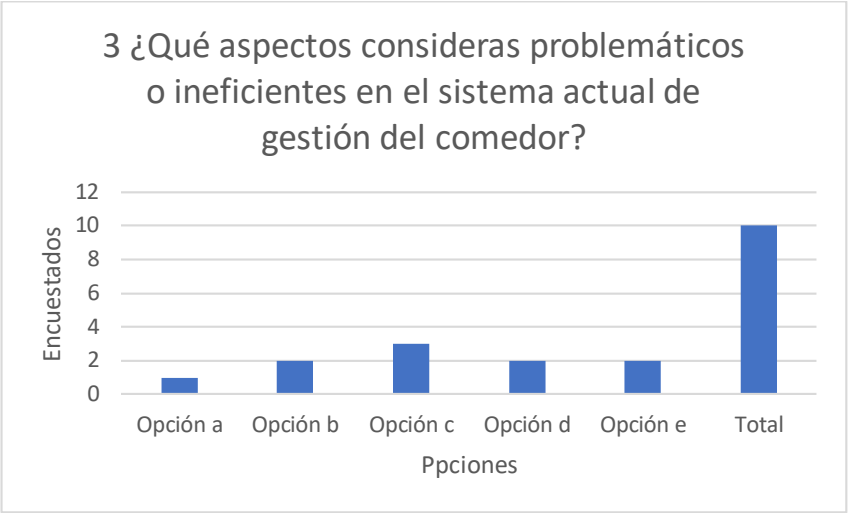
Pregunta 2



Pregunta 3. ¿Qué aspectos consideras problemáticos o ineficientes en el sistema actual de gestión del comedor?

Figura 9

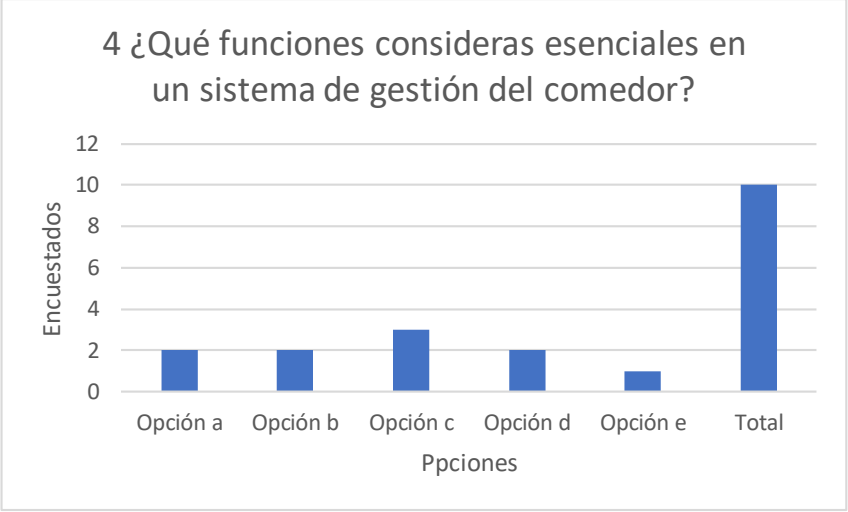
Pregunta 3



Pregunta 4. ¿Qué funciones consideras esenciales en un sistema de gestión del comedor?

Figura 10

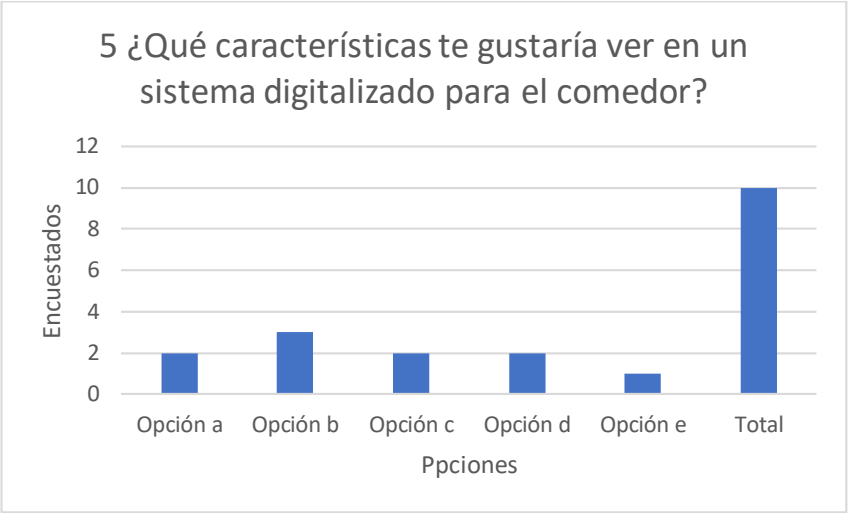
Pregunta 4



Pregunta 5. ¿Qué características te gustaría ver en un sistema digitalizado para el comedor?

Figura 11

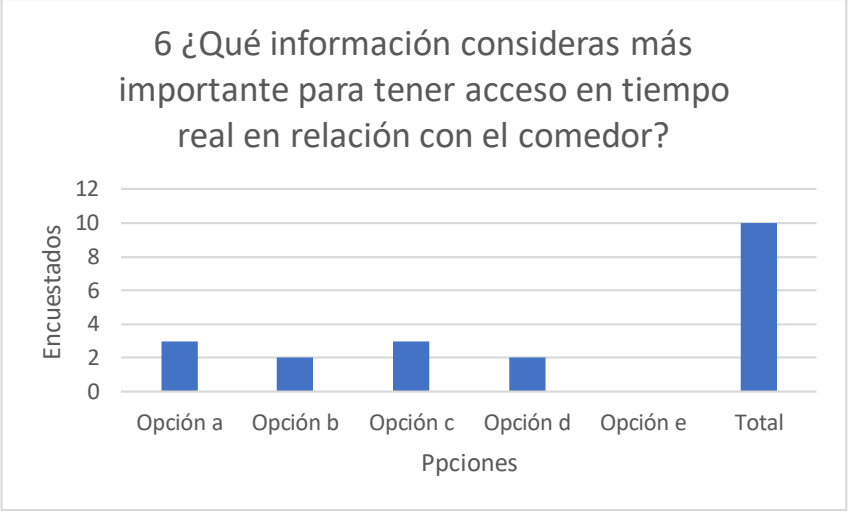
Pregunta 5



Pregunta 6. ¿Qué información consideras más importante para tener acceso en tiempo real en relación con el comedor?

Figura 12

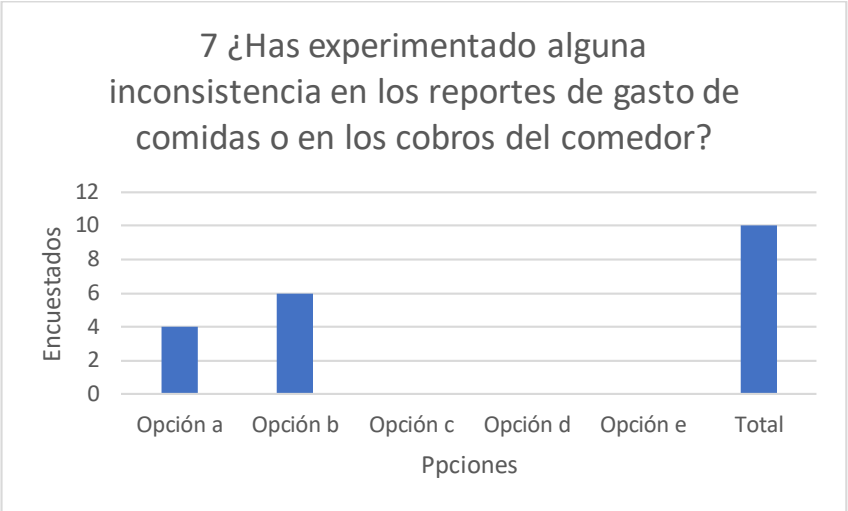
Pregunta 6



Pregunta 7. ¿Has experimentado alguna inconsistencia en los reportes de gasto de comidas o en los cobros del comedor?

Figura 13

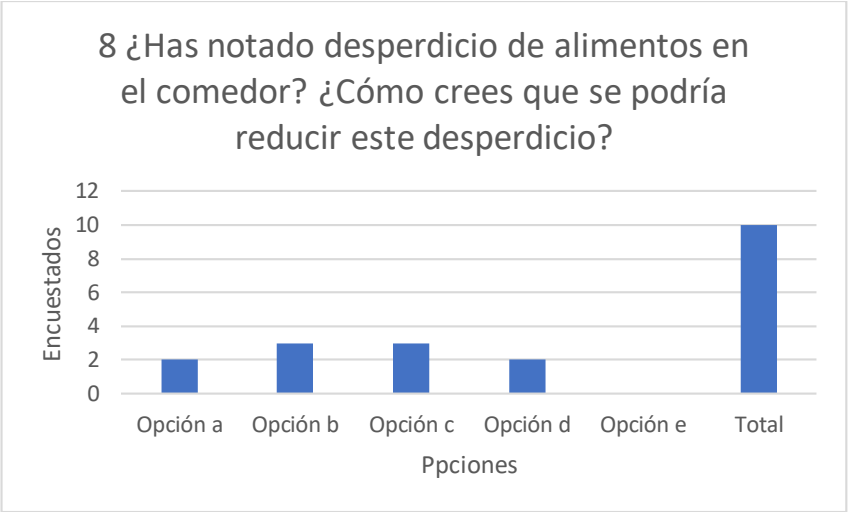
Pregunta 7



Pregunta 8. ¿Has notado desperdicio de alimentos en el comedor? ¿Cómo crees que se podría reducir este desperdicio?

Figura 14

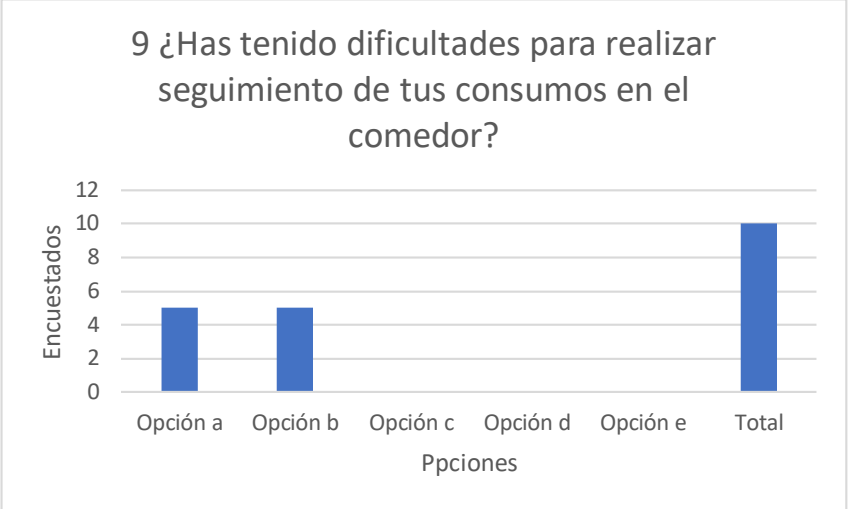
Pregunta 8



Pregunta 9. ¿Has tenido dificultades para realizar seguimiento de tus consumos en el comedor?

Figura 15

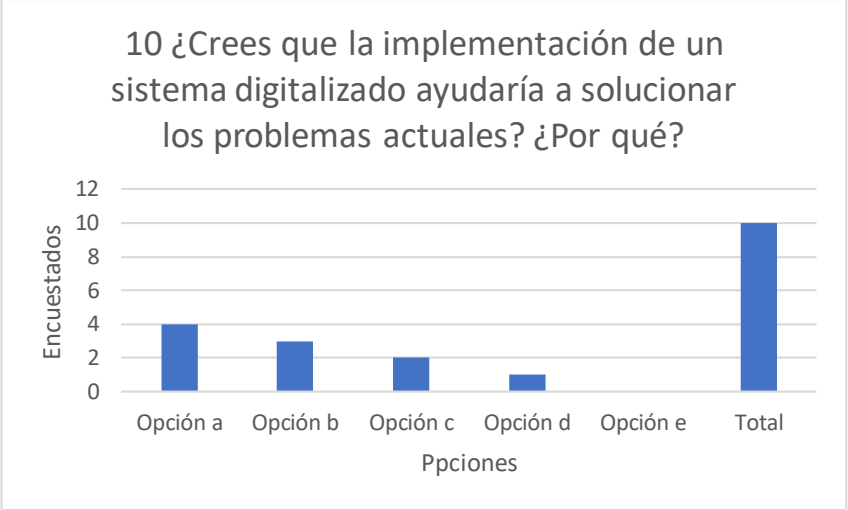
Pregunta 9



Pregunta 10. ¿Crees que la implementación de un sistema digitalizado ayudaría a solucionar los problemas actuales?

Figura 16

Pregunta 10



Tabulación datos Encuesta

Pregunta 1: ¿Con qué frecuencia utilizas el comedor de AGRUCOMGE?

Diariamente: 30%

Varias veces por semana: 40%

Ocasionalmente: 20%

Nunca: 10%

Pregunta 2: ¿Qué aspectos te parecen positivos del actual sistema de gestión del comedor?

Variedad de opciones: 20%

Horarios convenientes: 30%

Calidad de alimentos: 10%

Rapidez en el servicio: 20%

Otro (Especificar): 20%

Pregunta 3: ¿Qué aspectos consideras problemáticos o ineficientes en el sistema actual de gestión del comedor?

Inconsistencias en reportes de gastos: 10%

Desperdicio de alimentos: 20%

Cobros innecesarios: 30%

Dificultad para seguir el consumo: 20%

Otro (Especificar): 20%

Pregunta 4: ¿Qué funciones consideras esenciales en un sistema de gestión del comedor?

Registro de consumos: 20%

Alertas de vencimiento: 20%

Planificación de menús: 30%

Control de inventario: 20%

Otro (Especificar): 10%

Pregunta 5: ¿Qué características te gustaría ver en un sistema digitalizado para el comedor?

Interfaz fácil de usar: 20%

Acceso remoto: 30%

Notificaciones en tiempo real: 20%

Personalización de menús: 20%

Otro (Especificar): 10%

Pregunta 6: ¿Qué información consideras más importante para tener acceso en tiempo real en relación con el comedor?

Disponibilidad de alimentos: 30%

Registros de comidas prestadas y devueltas: 20%

Alertas de vencimiento: 30%

Otro (Especificar): 20%

Pregunta 7: ¿Has experimentado alguna inconsistencia en los reportes de gasto de comidas o en los cobros del comedor?

Sí: 40%

No: 60%

Pregunta 8: ¿Has notado desperdicio de alimentos en el comedor? ¿Cómo crees que se podría reducir este desperdicio?

Sí, limitando las porciones: 20%

Sí, mejorando la planificación de menús: 30%

Sí, implementando medidas de control de inventario: 30%

No, no he notado desperdicio: 20%

Pregunta 9: ¿Has tenido dificultades para realizar seguimiento de tus consumos en el comedor?

Sí: 50%

No: 50%

Pregunta 10: ¿Crees que la implementación de un sistema digitalizado ayudaría a solucionar los problemas actuales? ¿Por qué?

Sí, facilitaría el seguimiento y la gestión: 40%

Sí, reduciría errores y desperdicios: 30%

No, no estoy seguro: 20%

No, creo que no resolvería los problemas existentes: 10%

Análisis de resultados de la encuesta

Basado en los resultados de la encuesta, se puede observar que la mayoría de los encuestados utilizan el comedor de AGRUCOMGE varias veces por semana. Los aspectos

positivos más valorados del sistema de gestión actual son los horarios convenientes y la variedad de opciones. Sin embargo, los encuestados también identificaron problemas, siendo los más destacados los cobros innecesarios y las inconsistencias en los reportes de gastos.

En cuanto a las funciones esenciales en un sistema de gestión del comedor, la planificación de menús y el registro de consumos fueron considerados muy importantes. Los encuestados también expresaron su deseo de ver una interfaz fácil de usar y acceso remoto en un sistema digitalizado para el comedor.

La mayoría de los encuestados consideran que la disponibilidad de alimentos y las alertas de vencimiento son la información más importante para tener acceso en tiempo real. Además, la mitad de los encuestados han experimentado inconsistencias en los reportes de gasto de comidas o en los cobros del comedor, y han tenido dificultades para realizar seguimiento de sus consumos en el comedor.

Finalmente, la mayoría de los encuestados creen que la implementación de un sistema digitalizado facilitaría el seguimiento y la gestión, y reduciría errores y desperdicios. Sin embargo, también hay una proporción significativa de encuestados que no están seguros de si un sistema digitalizado resolvería los problemas existentes.

En conclusión, los resultados de la encuesta sugieren que, aunque hay aspectos positivos en el sistema de gestión actual del comedor de AGRUCOMGE, también existen áreas problemáticas que podrían beneficiarse de la digitalización. Sin embargo, cualquier implementación de un sistema digitalizado debe tener en cuenta las preocupaciones y necesidades específicas de los usuarios para ser efectiva.

Resultado datos entrevista

En la tabla se puede observar las respuestas de la entrevista realizada. Con esta información se ha podido establecer los requerimientos iniciales de la aplicación y levantar el proceso apegado a las necesidades de la AGRUCOMGE.

Tabla 3

Resultados Entrevista

Preguntas	Respuestas	Observaciones
¿Cuál es tu nombre y cuál es tu rol en la Agrupación de Comunicaciones y Guerra Electrónica "AGRUCOMGE"?	Mi nombre es Juan y soy el administrador del comedor en AGRUCOMGE.	-
¿Puedes describir cómo es el proceso actual de administración del comedor?	Actualmente, la administración del comedor se realiza manualmente, lo que a veces puede llevar a errores y dificultades para rastrear los datos.	Un sistema digitalizado podría mejorar la eficiencia y la precisión.
¿Cómo se lleva a cabo el control del inventario de alimentos y la gestión de los servicios de comedor?	El control del inventario y la gestión de los servicios se realizan a través de registros escritos.	Este método puede ser propenso a errores humanos y dificultades para rastrear los datos.
¿Has notado alguna inconsistencia en los reportes de gasto de comidas? Si es así, ¿puedes proporcionar algunos ejemplos?	Sí, a veces hay discrepancias entre los registros escritos y las existencias actuales.	Un sistema digitalizado podría minimizar estos errores.
¿Has observado algún desperdicio de comida o cobros indebidos? Si es así, ¿puedes proporcionar algunos ejemplos?	Sí, hemos tenido problemas con la sobre ordenación y la falta de seguimiento adecuado del inventario.	Un sistema digitalizado podría optimizar el inventario para evitar desperdicios.
¿Cómo se manejan las fechas de vencimiento de las comidas y los posibles excesos de inventario?	Es un desafío mantener un registro preciso debido al sistema manual actual.	La implementación digital garantizaría una gestión precisa del inventario.
¿Qué desafíos has enfrentado debido a la falta de un sistema digitalizado?	La falta de un sistema digitalizado ha llevado a errores en los reportes de gasto y dificultades para rastrear el inventario.	Un sistema digitalizado podría ayudar a superar estos desafíos.

Preguntas	Respuestas	Observaciones
¿Cómo crees que un sistema digitalizado podría mejorar la administración del comedor?	Creo que un sistema digitalizado podría mejorar la eficiencia, reducir los errores y facilitar el seguimiento del inventario y los gastos.	-
¿Hay algo más que te gustaría compartir sobre tu experiencia en la administración del comedor?	Creo que la implementación de un sistema digitalizado es esencial para mejorar la gestión del comedor.	

Métodos

Método de Investigación

Investigación de campo

Se realizará una investigación de campo, ya que se recolectará información necesaria de los involucrados, en este caso personal militar que son los usuarios, el oficial administrador que es el encargado de la gestión del comedor y el cocinero que es el ranchero.

Investigación Bibliográfica-Documental

Se realizará investigación bibliográfica – documental, siendo que se tomará información importante como libros virtuales, tesis, artículos científicos, con el objetivo de recopilar información sobre técnicas en los procesos de gestión de comedor y sobre el desarrollo de aplicaciones web, lo cual, será de utilidad durante el proceso de desarrollo del proyecto.

Modelo de desarrollo de software

A continuación, se describe las metodologías de desarrollo de software, con la finalidad de ampliar el panorama de los lectores, posteriormente se destallará la metodología seleccionada.

Prototipo: Se enfoca en entender los requisitos del usuario y mejorar la calidad de estos (Sommerville, 2005). Inicia con la recolección de requisitos, luego se desarrolla rápidamente un prototipo que el cliente evalúa para refinar los requisitos.

Desarrollo basado en componentes (reutilización): Se basa en la reutilización de componentes de software existentes (Pons et al., 2010). Permite un desarrollo más rápido, pero implica compromisos en los requisitos ya que no todo se desarrolla de cero.

Desarrollo en espiral: Combina actividades de especificación, desarrollo y validación en forma de espiral en lugar de etapas sucesivas (Boehm, 1988). Permite la entrega temprana de funcionalidades y la evaluación del cliente. Gestiona explícitamente los riesgos del proyecto.

RAD (Desarrollo Rápido de Aplicaciones): Modelo iterativo que enfatiza el uso de herramientas RAD para acelerar el desarrollo (Salazar et al., 2011). Su duración es relativamente corta (60-90 días). Requiere alto compromiso de los equipos.

Cascada: Representa las actividades como fases separadas y secuenciales (requisitos, diseño, implementación, etc.) (Royce, 1987). Es fácil de manejar, pero poco flexible a cambios. Requiere congelar partes del desarrollo mientras se avanza.

Luego de conocer algunos de los modelos utilizados para el desarrollo de software, se ha determinado realizar el desarrollo de un Aplicativo Web y móvil que permita la gestión de procesos y control del comedor de La Agrupación de Comunicaciones y Guerra Electrónica “AGRUCOMGE”, mediante el modelo de prototipo. Las fases de desarrollo utilizando el modelo de prototipo son:

Recolección de requisitos del cliente: Se reúnen los requisitos conocidos y necesidades de los usuarios.

Definición de objetivos: Se identifican los objetivos principales que debe cumplir el software con base en los requisitos recolectados.

Desarrollo de prototipo: Se crea rápidamente una versión preliminar o prototipo del sistema con la funcionalidad básica, enfocándose en la interfaz y aspectos visibles para el usuario.

Evaluación del prototipo: El cliente utiliza el prototipo, experimenta con él, y provee retroalimentación sobre qué requisitos faltan, qué se debe mejorar o cambiar.

Refinamiento de requisitos: Con base en la evaluación, se refinan los requisitos recolectados inicialmente y se aclaran dudas.

Desarrollo del sistema: Una vez que el prototipo es aprobado y los requisitos están claros, se desarrolla el sistema real cumpliendo con todos los requerimientos.

Pruebas y entrega: Se realizan pruebas exhaustivas al software y se entrega la versión final al cliente.

Este ciclo de desarrollo de prototipo y refinamiento se repite en iteraciones hasta que el cliente quede completamente satisfecho.

Análisis y modelado de procesos

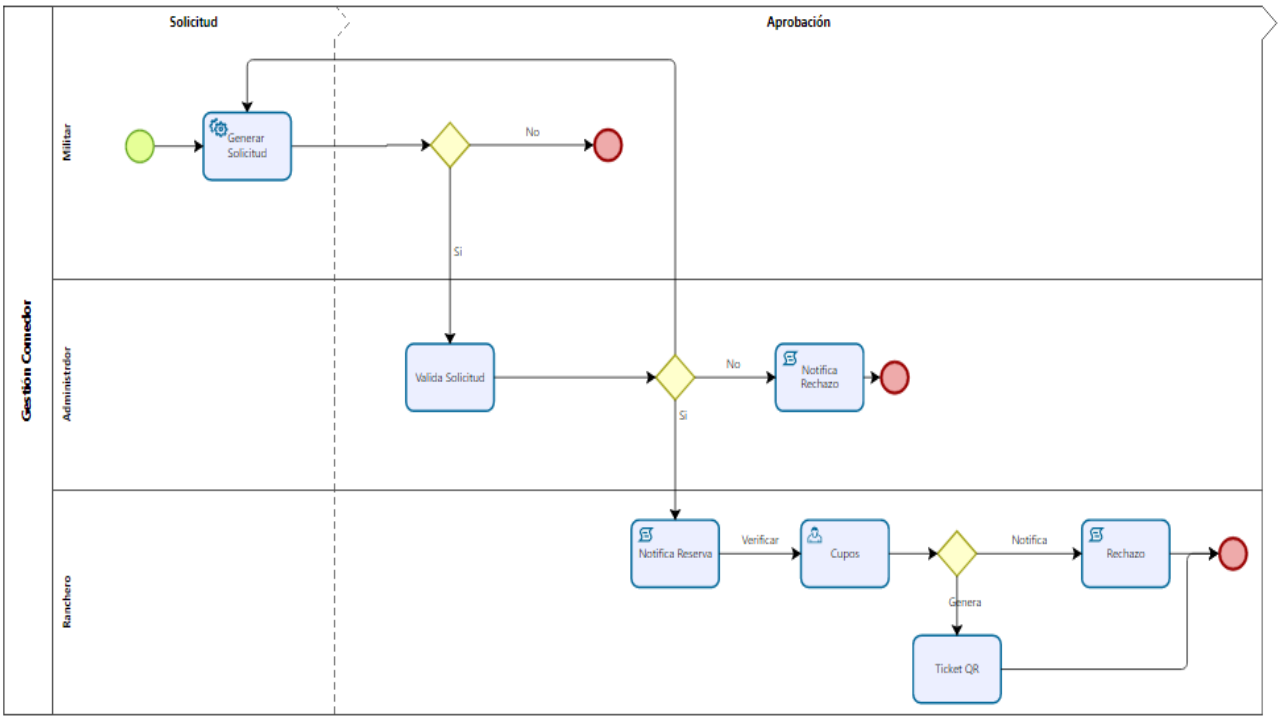
En este aparatado se va a describir el proceso de gestión de comedor de la AGRUCOMGE, luego de realizar el estudio y análisis de los procesos actuales. El objetivo del modelado/levantamiento de procesos es documentar el flujo de actividades, tareas, actores, sistemas y datos involucrados en los procesos clave del negocio. Esto permite analizar y optimizar dichos procesos, proponer mejoras, cambios o automatizaciones. También sirve como base para desarrollar software que soporte esos procesos.

Flujo del Proceso de Gestión de Comidas:

1. Inicio del Proceso
2. El Personal Militar (Usuarios) confirma su asistencia al comedor y selecciona sus comidas (desayuno, almuerzo, cena).
3. La solicitud del Personal Militar llega al Administrador.
4. El Administrador verifica que la solicitud sea coherente. Adicional validara el ticket generado cuando el usuario militar se acerque a el comedor.
5. El Administrador indica la reserva al ranchero o cocinero.
6. El Cocinero revisa la información de asistencia a comidas y confirma y genera los tickets a todos los usuarios que realizaron la reserva.
7. El sistema centraliza la información de asistencia y selección de comidas para generar reportes.
8. El sistema también maneja alertas, respaldos de información y permite manipulación de la base de datos.
9. Fin del Proceso.

Figura 17

Diagrama de procesos



Nota. Proceso de gestión de comidas en el comedor del AGRUCOMGE.

Análisis de requerimientos.

En este apartado se identifican los requerimientos funcionales y no funcionales del sistema, para lo cual nos servirá de base, los datos tabulados de las entrevistas y las encuestas descritas en la sección materiales. Esto permite tener una visión clara de las necesidades a cubrir.

Requerimientos funcionales

- ✓ Autenticación de usuarios (comensales administrador y ranchero)
- ✓ Módulo de reserva de comidas por parte de comensales
- ✓ Módulo de gestión de reservas por parte del cocinero (validación, rechazo por falta de cupo, etc.)
- ✓ Generación de tickets o códigos para comensales.

- ✓ Escaneo y validación de tickets de comida al ingreso al comedor
- ✓ Reportes de comidas servidas, ingresos, etc.

Requerimiento no funcionales

- ✓ El sistema debe tener una interfaz responsive para funcionar en desktop y dispositivos móviles
- ✓ El sistema debe soportar al menos 100 usuarios concurrentes con un tiempo de respuesta menor a 3 segundos.
- ✓ La interfaz de usuario debe seguir los estándares corporativos de la institución y ser intuitiva.
- ✓ La información del sistema debe respaldarse en una base de datos Postgress
- ✓ El sistema debe operar correctamente con los navegadores Chrome, Firefox.
- ✓ La autenticación debe ser segura, cifrando las contraseñas en la BD.
- ✓ Se debe proveer documentación de usuario final y técnica.

Requerimiento de hardware y software

Algunos ejemplos de requerimientos de hardware/software que podrías considerar desde el inicio para tu proyecto son:

Software de Servidor Web

- ✓ Apache 2.4.x (última versión estable)
- ✓ Nginx (También puede usarse como infraestructura para proxy inverso, balanceador de carga, y caché HTTP)

Entorno de Desarrollo

- ✓ Visual Studio Code: Editor de código fuente.
- ✓ Extensión de PHP para VS Code: Mejora la productividad al desarrollar en PHP.
- ✓ Composer: Gestor de dependencias de PHP.

Tecnologías de Backend

- ✓ PHP 8.1: Lenguaje de programación.
- ✓ Laravel 10.x: Framework de desarrollo web en PHP.
- ✓ Blade Template Engine: Motor de plantillas para PHP.
- ✓ Eloquent ORM: Mapeo objeto-relacional para trabajar con bases de datos.
- ✓ Migraciones de base de datos: Permite versionar la estructura de la base de datos.
- ✓ Colas y Jobs: Para la ejecución de tareas en segundo plano.
- ✓ Autenticación, autorización y políticas: Manejo de acceso y permisos dentro de la aplicación.

Base de Datos

- ✓ PostgreSQL: Sistema de gestión de bases de datos relacional.

Control de Versiones

- ✓ Git / GitHub: Se mantiene Git como herramienta de control de versiones con GitHub para la gestión de repositorios.

Pruebas

- ✓ PHPUnit: Framework de pruebas para PHP.

Infraestructura

- ✓ Docker: Plataforma de contenedores para simplificar la creación y administración de entornos de desarrollo.
- ✓ Docker Compose: Herramienta para definir y ejecutar aplicaciones Docker multi-contenedor.

Seguridad

- ✓ Encriptación en .env con Hash::make(): Para la encriptación de contraseñas y datos sensibles.
- ✓ CORS: Middleware para habilitar CORS (Compartición de Recursos de Origen Cruzado).

Documentación

- ✓ Postman: Plataforma para el desarrollo de APIs que permite probar, documentar y monitorear APIs.

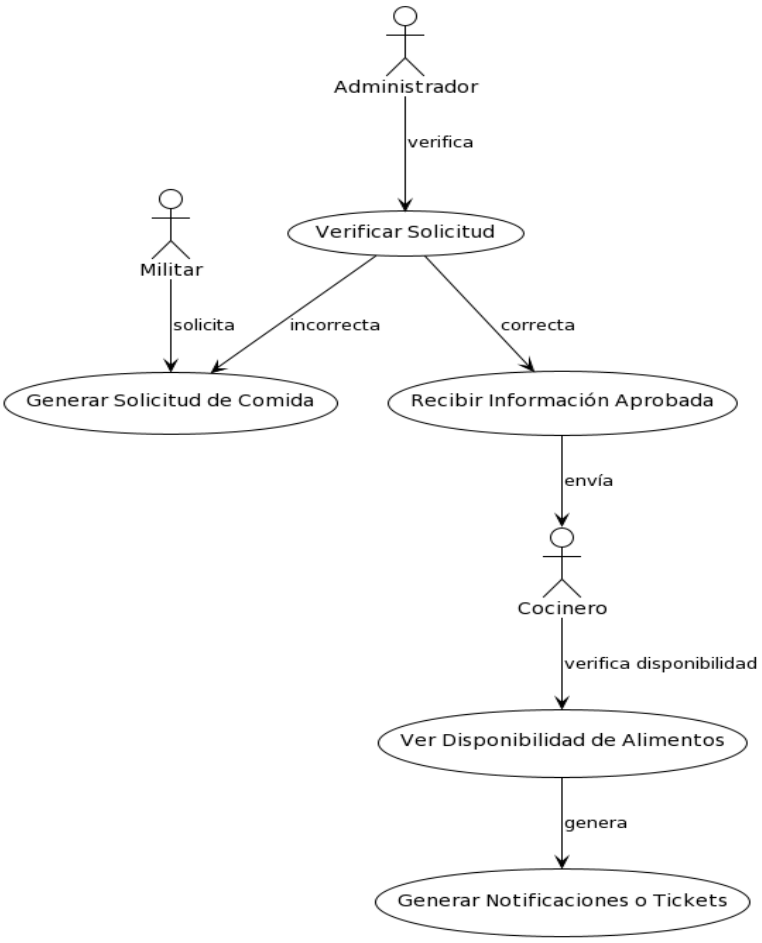
De esta manera se tiene una pila tecnológica robusta, escalable y que promueve las buenas prácticas de desarrollo con Laravel, uno de los frameworks web más populares actualmente en PHP.

Casos de Uso

Los proyectos de desarrollo de software suelen fallar, por diversos motivos, siendo los más comunes, la entrega tardía y el aumento de los costos de desarrollo, superando lo planificado y presupuestado. La causa de estos problemas es la falta de concordancia o pobre relación entre las necesidades reales de los usuarios finales del sistema y los requisitos solicitados (Inés Lund et al., s. f.). En este sentido se procederán a elaborar los caos de uso con la finalidad de obtener un producto software que satisfaga las necesidades reales de los usuarios. En la figura 18 se describe el UML del proceso general del comedor con sus usuarios involucrados.

Figura 18

UML General del comedor



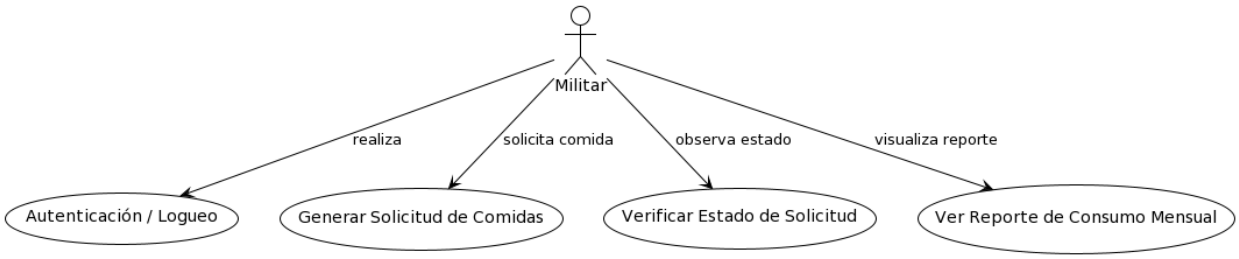
Nota. UML en el que describe de forma general el proceso del comedor.

Desglosar un diagrama general en diagramas más pequeños por cada rol de usuario mejora la claridad al concentrarse en las interacciones específicas entre cada actor del sistema y simplifica la gestión de cambios, ya que es más fácil identificar cómo las actualizaciones afectan a cada actor por separado, lo que reduce la complejidad y los riesgos de errores.

En la figura 19, se procede a detallar el caso de uso que corresponde a los usuarios militares.

Figura 19

UML de usuario Militar



Nota. Descripción de caso de uso de los usuarios militares.

En la tabla 4 se describe de manera detallada el caso de uso de la figura 19.

Tabla 4

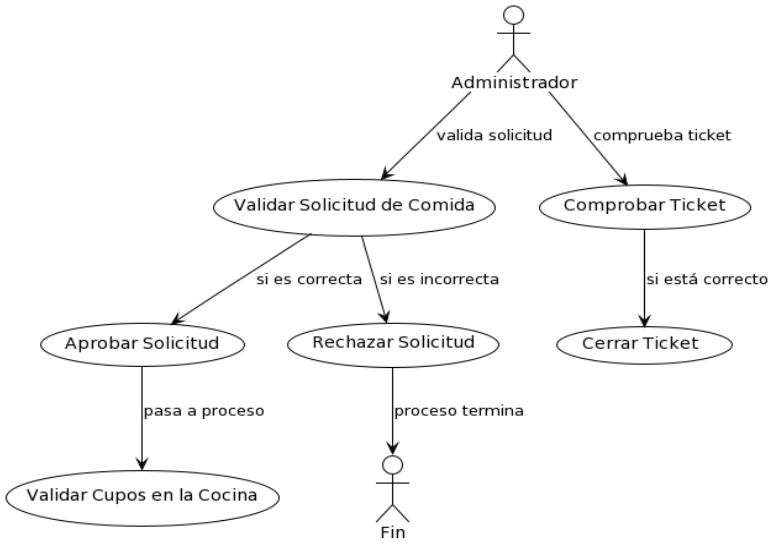
Detalle de UML usuario Militar

CASOS DE USO	**CASO 1:. CREA SOLICITUD**
DESCRIPCIÓN	El usuario militar puede crear solicitudes de comidas como desayuno, almuerzo o merienda.
PRE-CONDICIÓN	El militar debe estar registrado el sistema web Ingresar al sistema Autenticarse o Login
SECUENCIA NORMAL	1. Ingresar al aplicativo web 2. Ingresar los datos de su cuenta personal 3. Proceso de verificación de credenciales. 4. Crear solicitud indicando la comida a solicitar. 5. Enviar solicitud.
POSTCONDICIÓN	La solicitud generada será enviada para validación.
EXCEPCIONES	1. Si el militar no se encuentra registrado en el aplicativo, no podrá ingresar al aplicativo ni realizar procesos.

En la figura 20, se describe el UML que corresponde al usuario militar.

Figura 20

UML usuario Administrador



Nota. Descripción de caso de uso del usuario administrador.

En la tabla 5, se describe de manera detallada el caso de uso de la figura 20.

Tabla 5

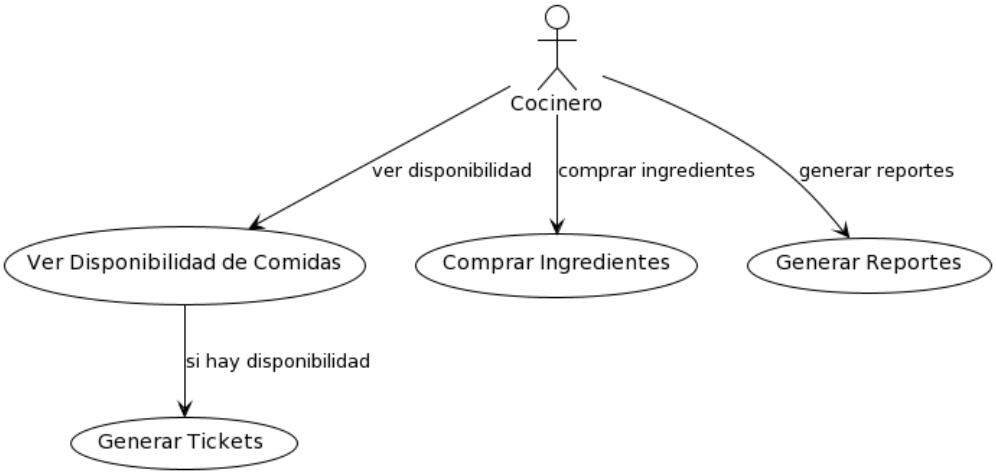
Detalle UML usuario Administrador

CASOS DE USO	**CASO 2:. VERIFICA TICKETS**
DESCRIPCIÓN	El usuario administrador puede verificar los tickets a los usuarios militares
PRE-CONDICIÓN	Se debe haber creado solicitudes de comida por parte de los militares Ingresar al sistema Autenticarse o Login
SECUENCIA NORMAL	1. Ingresar al aplicativo web 2. Ingresar número de ticket 3. Proceso de verificación de código de ticket 4. Cerrar ticket. 5. Generar reportes.
POSTCONDICIÓN	Se actualizarán datos en la BDD
EXCEPCIONES	1. Si el ticket no es válido no se podrá servir la comida.

En la figura 21, se describe el UML que corresponde al usuario ranchero o cocinero.

Figura 21

UML del Ranchero o cocinero



Nota. Descripción de caso de uso del usuario Ranchero.

En la tabla 6 se describe de manera detallada el caso de uso de la figura 21.

Tabla 6

UML usuario Cocinero

CASOS DE USO	**CASO 3:. GENERAR TICKETS**
DESCRIPCIÓN	El usuario cocinero puede revisar las solicitudes aprobadas por el administrador y en base a la disponibilidad de alimentos genera tickets con codigos únicos a los militares.
PRE-CONDICIÓN	Se debe haber aprobado las solicitudes de comida por parte del administrador Ingresar al sistema Autenticarse o Logiin
SECUENCIA NORMAL	1. Ingresar al aplicativo web 2. revisar el listado de solicitudes aprobadas 3. Ver disponibilidad de alimentos 4. Generar notificaciones con tickets 5. Generar reportes de consumo diarios y mensuales.
POSTCONDICIÓN	Se actualizarán datos en la BDD
EXCEPCIONES	1. Si el ticket no es válido no se podrá servir la comida.

Diagrama de Clases

Un diagrama de clases modela diferentes aspectos de las clases en un sistema orientado a objetos y las relaciones entre ellas, permitiendo visualizar y entender mejor su

diseño. Es una parte importante del modelado OO durante el desarrollo de software. En la figura 22, se puede observar el diagrama de clases del aplicativo.

Figura 22

Diagrama de Clases

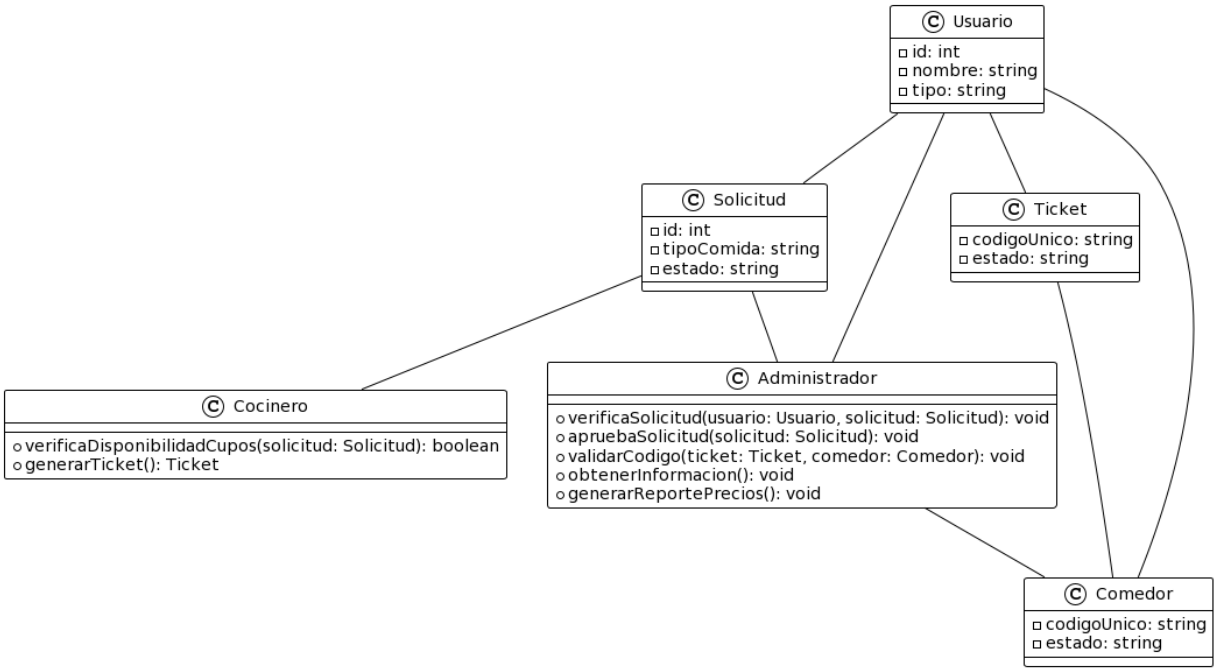
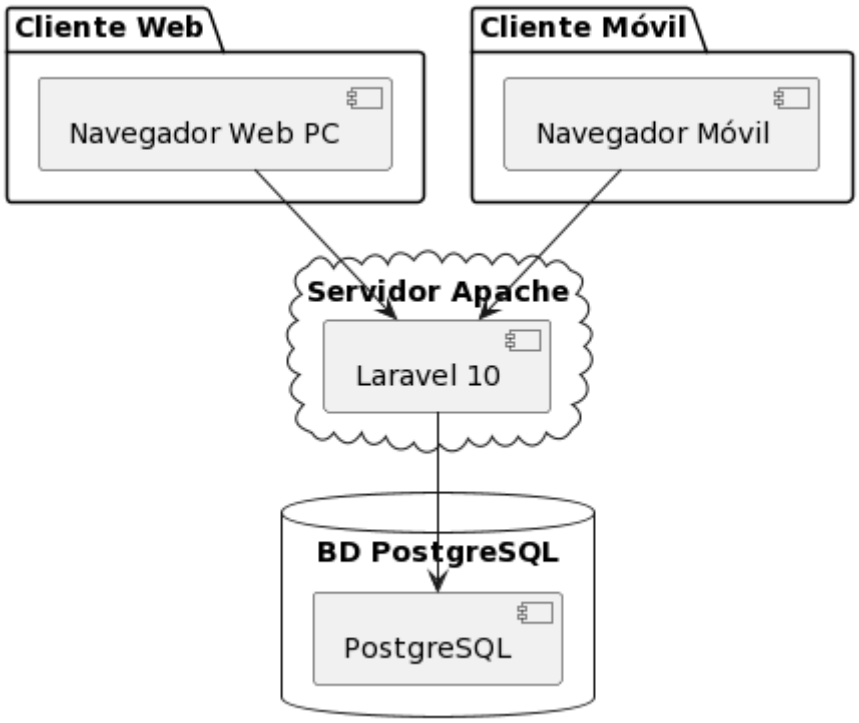


Diagrama de red

Un diagrama de red sirve para visualizar la interacción y la comunicación entre diferentes componentes de una red, como servidores, clientes y bases de datos. En este contexto en la figura se muestra el diagrama, donde se puede visualizar cómo los clientes web y móviles interactúan con un servidor Apache que utiliza Laravel 10 y una base de datos PostgreSQL. Esto puede ayudar a entender el flujo de información, identificar posibles cuellos de botella o problemas de seguridad, y facilitar la planificación de cambios o mejoras en la red.

Figura 23

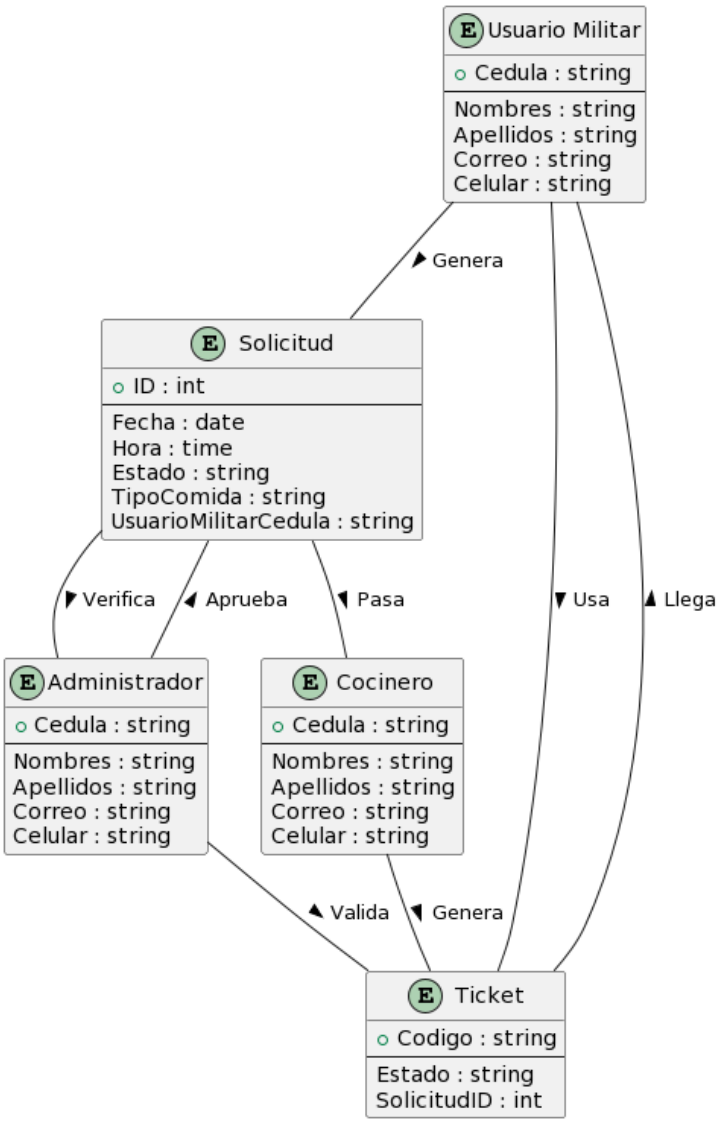
Diagrama de red



Diseño de la Base de Datos

Modelo Lógico

El modelo lógico de base de datos se centra en la estructura y organización de los datos de manera independiente del hardware o software subyacente. En esta etapa, se ha definido la estructura de la base de datos en términos de entidades, atributos, relaciones y restricciones. El objetivo principal es capturar la lógica y la semántica de los datos, sin preocuparse por detalles de implementación.

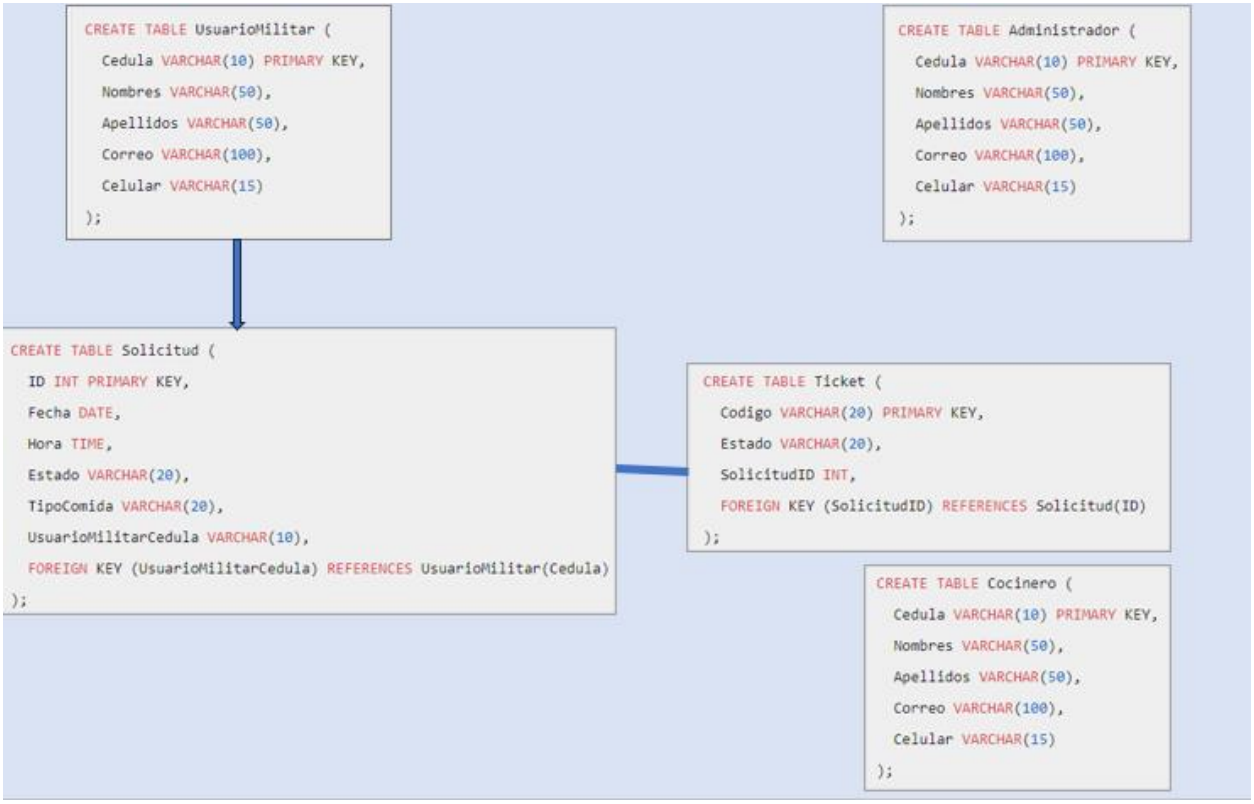


Modelo Físico

El modelo lógico de base de datos suele representarse utilizando diagramas de entidad-relación (ER), que muestran las entidades principales en la base de datos y sus relaciones. También puede incluir diagramas de modelo relacional, que muestran cómo se traducen las entidades y relaciones a tablas y claves primarias y foráneas. En este contexto se procede a mostrar el diccionario de datos.

Figura 24

Modelo Físico de BDD



Diccionario de datos

En la tabla 7, se puede observar el diccionario de datos, en donde se detalla cada uno de los campos con su tipo de datos, tamaño y descripción.

Tabla 7

Diccionario de Datos

Tabla	Campo	Tipo	Tamaño	Descripción
UsuarioMilitar	Cedula	VARCHAR	10	Cédula del usuario militar, llave primaria
UsuarioMilitar	Nombres	VARCHAR	50	Nombres del usuario militar
UsuarioMilitar	Apellidos	VARCHAR	50	Apellidos del usuario militar
UsuarioMilitar	Correo	VARCHAR	100	Correo electrónico del usuario militar
UsuarioMilitar	Celular	VARCHAR	15	Número de celular del usuario militar
Administrador	Cedula	VARCHAR	10	Cédula del administrador, llave primaria
Administrador	Nombres	VARCHAR	50	Nombres del administrador
Administrador	Apellidos	VARCHAR	50	Apellidos del administrador

Tabla	Campo	Tipo	Tamaño	Descripción
Administrador	Correo	VARCHAR	100	Correo electrónico del administrador
Administrador	Celular	VARCHAR	15	Número de celular del administrador
Cocinero	Cedula	VARCHAR	10	Cédula del cocinero, llave primaria
Cocinero	Nombres	VARCHAR	50	Nombres del cocinero
Cocinero	Apellidos	VARCHAR	50	Apellidos del cocinero
Cocinero	Correo	VARCHAR	100	Correo electrónico del cocinero
Cocinero	Celular	VARCHAR	15	Número de celular del cocinero
Solicitud	ID	INT	-	ID de la solicitud, llave primaria
Solicitud	Fecha	DATE	-	Fecha de la solicitud
Solicitud	Hora	TIME	-	Hora de la solicitud
Solicitud	Estado	VARCHAR	20	Estado de la solicitud
Solicitud	TipoComida	VARCHAR	20	Tipo de comida solicitada
Solicitud	UsuarioMilitarCedula	VARCHAR	10	Cédula del usuario militar que realizó la solicitud, clave foránea a UsuarioMilitar
Ticket	Codigo	VARCHAR	20	Código del ticket, llave primaria
Ticket	Estado	VARCHAR	20	Estado del ticket
Ticket	SolicitudID	INT	-	ID de la solicitud relacionada con el ticket, clave foránea a Solicitud

Diseño e interfaz del sistema

Estándar de la interfaz de Usuario

Mediante las reuniones mantenidas con el usuario administrador del aplicativo, se definió las características de las interfaces del aplicativo web, aspectos como, color, tipo de letra, logos entre otros. De esta manera ofrecer una interfaz sencilla y amigable para los usuarios finales. A continuación de detalla estos aspectos en la tabla 8.

Tabla 8

Estándar de desarrollo de interfaces

Aspecto	Descripción
Colores	Los colores se seleccionarán en base a la marca y la identidad visual del producto. Se definirá una paleta de colores principal y secundaria.
Tipos de letra	Se seleccionará un tipo de letra que sea legible y esté en línea con la identidad visual del producto. Se definirán tipos de letra para los encabezados, el texto del cuerpo, los botones, etc.
Tamaño de fuente	Se definirán tamaños de fuente estándar para diferentes elementos de la interfaz, como encabezados, texto del cuerpo, subtítulos, etc.

Aspecto	Descripción
Espaciado	Se establecerán reglas para el espaciado entre los elementos de la interfaz para asegurar una apariencia limpia y ordenada.
Imágenes y gráficos	Se definirán directrices para el uso de imágenes y gráficos, incluyendo el tamaño, la resolución y el estilo.
Interactividad	Se definirán las interacciones estándar del usuario, como clics, desplazamiento, arrastrar y soltar, etc.
Responsividad	La interfaz se diseñará para ser responsive, es decir, se verá y funcionará bien en una variedad de tamaños de pantalla y dispositivos.
Menús	Se definirán los estilos para los menús, incluyendo menús desplegables, menús laterales, etc. Esto incluirá colores, tipos de letra, tamaños y espaciado.
Alertas	Se establecerán directrices para las alertas y mensajes de error, incluyendo colores, iconografía, texto y posicionamiento.
Botones	Se definirán los estilos para los botones, incluyendo colores, tamaños, tipos de letra y estados (por ejemplo, hover, active, disabled).
Paneles	Se establecerán directrices para los paneles, incluyendo colores de fondo, bordes, sombras, tamaños de fuente y espaciado.
Formularios	Se definirán los estilos para los formularios y sus elementos, como campos de entrada, casillas de verificación, selectores, etc. Esto incluirá colores, tamaños, tipos de letra y espaciado.
Iconografía	Se seleccionará un conjunto de iconos que se utilizará en toda la interfaz. Los iconos deben ser claros, reconocibles y consistentes en estilo.

Interfaces del aplicativo web

El aplicativo web cuenta con interfaces graficas que satisfacen los siguientes criterios:

Seguridad:

Protección de datos: El login seguro protege la información confidencial de los usuarios.

Prevención de accesos no autorizados: El login seguro evita que usuarios no autorizados accedan a la información o funcionalidades del sistema.

Reducción del riesgo de ataques: El login seguro reduce el riesgo de ataques cibernéticos como phishing, malware y ataques de fuerza bruta.

Usabilidad:

Mejora la experiencia del usuario: El login seguro proporciona una experiencia de usuario más segura y confiable.

Personalización: La página de menú permite personalizar la experiencia del usuario.

Acceso rápido a la información: El dashboard ofrece una vista general de la información más importante.

Eficiencia:

Mejora la productividad: El dashboard permite a los usuarios acceder a la información que necesitan de forma rápida y eficiente.

Toma de decisiones más rápidas: El dashboard facilita la toma de decisiones al presentar la información de forma organizada y visual.

Mejora la comunicación: El dashboard facilita la comunicación entre los usuarios al proporcionar una vista compartida de la información.

En las figuras 25 y figura 26 se puede observar las interfaces de inicio del sistema.

Figura 25

Interfaz de Login

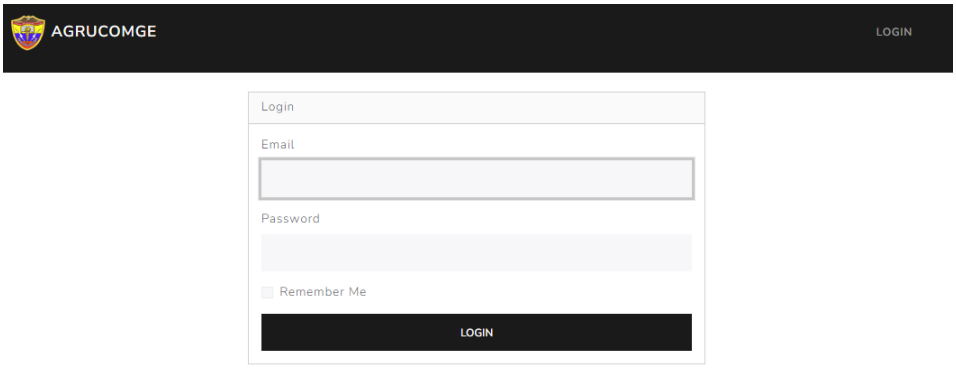
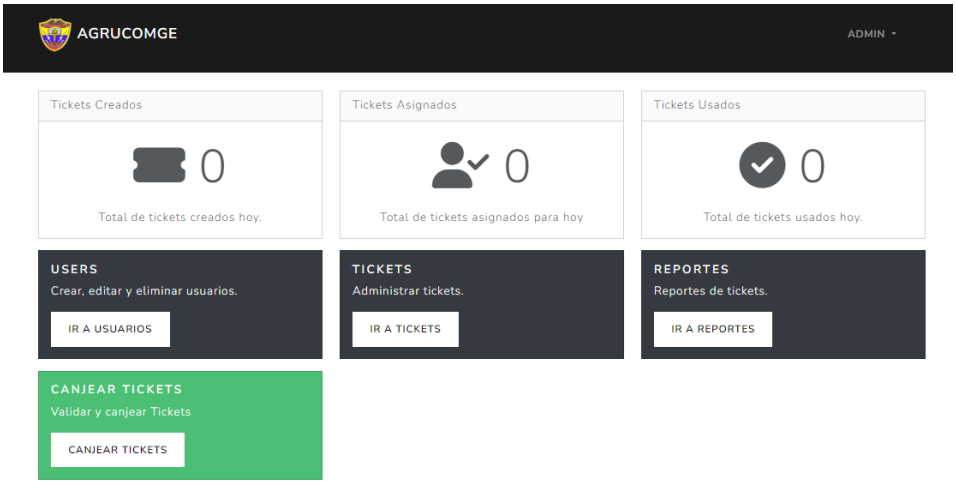


Figura 26

Interfaz de Panel Principal

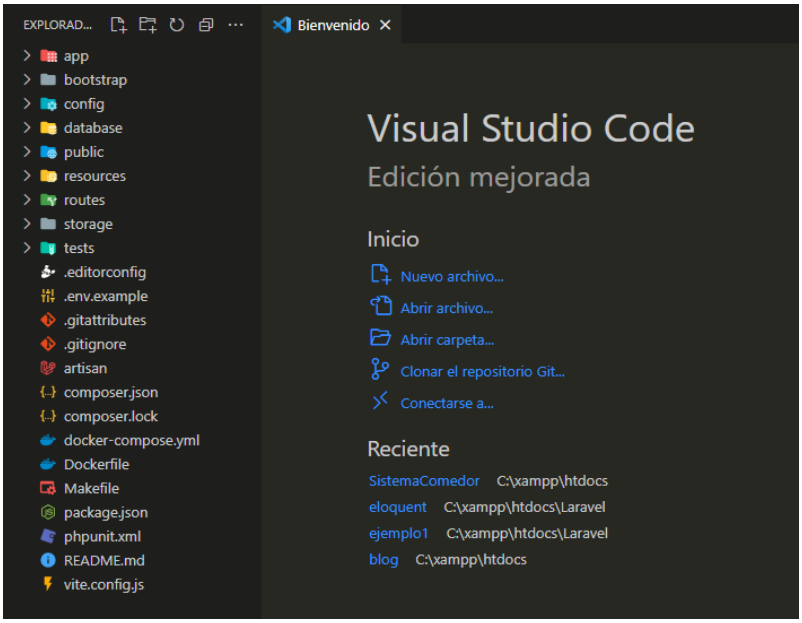


Estructura del proyecto web

Creamos nuestro proyecto con el comando: composer create-project laravel/laravel meal-tracker. La estructura del proyecto se muestra en la figura 27.

Figura 27

Estructura del proyecto en Laravel 10x

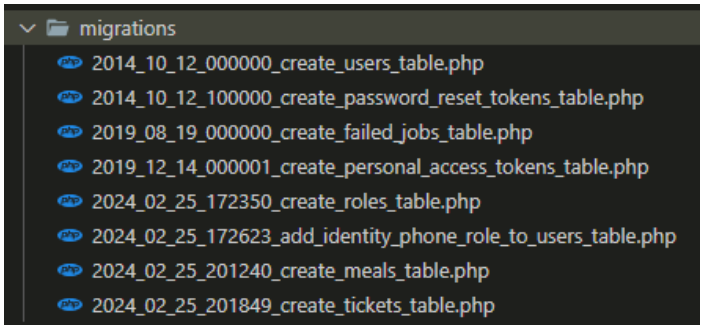


Migraciones

Las migraciones son una característica que facilita la administración y el versionado de la base de datos. Laravel utiliza migraciones para crear y modificar la estructura de la base de datos de manera programática, en lugar de realizar cambios directamente en el esquema de la base de datos. En la figura 26 podemos observar algunos de los modelos principales del proyecto.

Figura 28

Migraciones



Migración Users

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
```

```
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
};
```

Migración Roles

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('roles', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('roles');
```

```
    }
};
```

Migración Tickets

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('tickets', function (Blueprint $table) {
            $table->id();
            $table->string('code', 4);
            $table->date('valid_for');
            $table->boolean('redeemed')->default(false);
            $table->foreignId('meal_id')->constrained('meals')->onDelete('set null');
            $table->foreignId('user_id')->nullable()->constrained('users')->onDelete('set null');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('tickets');
    }
};
```

Migración Meals

```
<?php
```

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('meals', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('meals');
    }
};
```

Migración Token

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
```

```
    {
        Schema::create('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamp('expires_at')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('personal_access_tokens');
    }
};
```

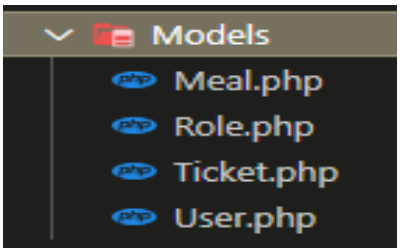
De esta manera se permite al desarrollador definir los cambios en la estructura de la base de datos utilizando código PHP en lugar de escribir consultas SQL directamente. Esto hace que sea más fácil colaborar en equipo, mantener un historial de cambios en la estructura de la base de datos y desplegar aplicaciones en diferentes entornos sin tener que preocuparse por sincronizar manualmente la estructura de la base de datos.

Modelos

Los modelos son clases de PHP que representan y encapsulan la lógica de negocio y la interacción con la base de datos. Los modelos en Laravel suelen corresponder a tablas específicas de la base de datos, lo que facilita la manipulación de los datos de esas tablas mediante operaciones como la creación, lectura, actualización y eliminación (CRUD). En la figura se puede ver algunos de los modelos más importantes del proyecto.

Figura 29

Modelos



Modelo Meals

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Meal extends Model
{
    use HasFactory;

    public function tickets()
    {
        return $this->hasMany(Ticket::class);
    }
}
```

Modelo Role

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Role extends Model
{
    /**
     * The attributes that are mass assignable.
```

```
    *
    * @var array<int, string>
    */
    protected $fillable = [
        'name'
    ];

    /**
    * Get the users for the role.
    */
    public function users()
    {
        return $this->hasMany(User::class);
    }
}
```

Modelo Ticket

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Str;

class Ticket extends Model
{
    use HasFactory;

    /**
    * The attributes that are mass assignable.
    *
    * @var array<int, string>
    */
    protected $fillable = [
        'code',
        'valid_for',
        'meal_id',
    ];

    public function meal()
```

```
    {
        return $this->belongsTo(Meal::class);
    }

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    /**
     * Generate a unique 5-character alphanumeric code.
     *
     * @return string
     */
    public static function generateCode()
    {
        $code = '';
        do {
            $code = strtoupper(Str::random(4));
        } while (self::where('code', $code)->exists());

        return $code;
    }
}
```

Modelo User

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
```

```
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'identity',
        'name',
        'email',
        'phone',
        'rank',
        'password',
        'role_id',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];

    /**
     * Get the role that owns the user.
     */
    public function role()
    {
        return $this->belongsTo(Role::class);
    }

    /**
     * Check if the user has a specific role.
     *
     * @param string $roleName
```

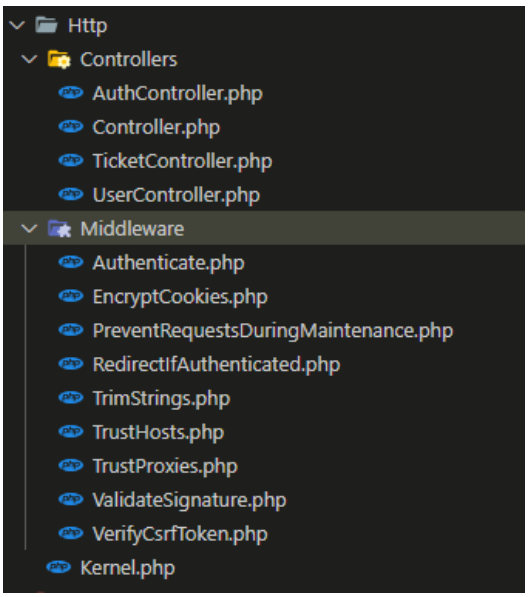
```
    * @return bool
    */
    public function hasRole($roleName)
    {
        return $this->role->name === $roleName;
    }
}
```

Controladores

Los controladores son clases de PHP que se utilizan para manejar las solicitudes HTTP y ejecutar la lógica de la aplicación. Los controladores actúan como intermediarios entre las rutas definidas en la aplicación y la lógica que debe ejecutarse en respuesta a esas solicitudes. En la figura tenemos los controladores más importantes de nuestro proyecto.

Figura 30

Controladores



Controlador AuthController

```
<?php

namespace App\Http\Controllers;
```

```php
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    public function show()
    {
        return view('auth.login');
    }

    public function login(Request $request)
    {
        $credentials = $request->validate([
            'email'    => ['required', 'email'],
            'password' => ['required'],
        ]);

        if (Auth::attempt($credentials, $request->filled('remember'))) {
            $request->session()->regenerate();

            return redirect()->intended('/');
        }

        return back()->withErrors([
            'email' => 'The provided credentials do not match our records.',
        ])->onlyInput('email');
    }

    public function logout(Request $request)
    {
        Auth::logout();

        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return redirect('/');
    }
}
```

Controlador TicketController

```php
<?php

namespace App\Http\Controllers;

use App\Models\Meal;
use App\Models\Ticket;
use App\Models\User;

use Illuminate\Http\Request;

class TicketController extends Controller
{
    public function create()
    {
        return view('tickets.create');
    }

    public function generate(Request $request)
    {
        $request->validate(['valid_for' => 'required|date']);

        $diners = User::whereHas('role', function ($query) {
            $query->where('name', 'Diner');
        })->get();

        $meals = Meal::all();

        foreach ($diners as $diner) {
            foreach ($meals as $meal) {
                Ticket::create([
                    'code'      => Ticket::generateCode(),
                    'valid_for' => $request->valid_for,
                    'meal_id'   => $meal->id,
                ]);
            }
        }

        return back()->with('success', 'Tickets generated successfully.');
    }
}
```

Controlador UserController

```php
?php

namespace App\Http\Controllers;

use App\Models\User;
use App\Models\Role;

use Illuminate\Http\Request;

class UserController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $users = User::all();
        return view('users.index', compact('users'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        $roles = Role::all();
        return view('users.create', compact('roles'));
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $request->validate([
            'identity' => 'required|unique:users,identity',
            'name'     => 'required',
            'email'    => 'required|email|unique:users,email',
            'password' => 'required|min:6',
        ]);

        $user           = new User($request->all());
        $user->password = bcrypt($request->password);
        $user->save();
```

```
    return redirect()->route('users.index')->with('success', 'User created successfully.');
}

/**
 * Display the specified resource.
 */
public function show(string $id)
{
    $user = User::find($id);
    return view('users.show', compact('user'));
}

/**
 * Show the form for editing the specified resource.
 */
public function edit(string $id)
{
    $user = User::find($id);
    return view('users.edit', compact('user'));
}

/**
 * Update the specified resource in storage.
 */
public function update(Request $request, string $id)
{
    $user = User::find($id);

    $rules = [
        'identity' => 'required|unique:users,identity,' . $user->id,
        'name'     => 'required',
        'email'    => 'required|email|unique:users,email,' . $user->id,
    ];

    if ($request->filled('password')) {
        $rules['password'] = 'min:6';
    }

    $request->validate($rules);

    $data = $request->only(['identity', 'name', 'email', 'phone']);

    if ($request->filled('password')) {
        $data['password'] = bcrypt($request->password);
    }
```

```
        $user->update($data);

        return redirect()->route('users.index')->with('success', 'User updated successfully.');
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(string $id)
    {
        $user->delete();
        return redirect()->route('users.index')->with('success', 'User deleted successfully.');
    }
}
```

Rutas

Las rutas son definiciones de URL que mapean las solicitudes HTTP entrantes a acciones específicas en la aplicación. En otras palabras, las rutas determinan cómo la aplicación responde a las diferentes URL que los usuarios solicitan a través de su navegador web u otras herramientas de cliente.

En Laravel, las rutas se definen generalmente en el archivo routes/web.php para las rutas web y en el archivo routes/api.php para las rutas de la API. Estos archivos contienen declaraciones de rutas que especifican la URL, el verbo HTTP (GET, POST, PUT, DELETE, etc.) y el controlador o la función de cierre que se debe ejecutar cuando se accede a esa URL.

Web Route

```
<?php

use App\Http\Controllers\AuthController;
use App\Http\Controllers\DashboardController;
use App\Http\Controllers\TicketController;
use App\Http\Controllers\UserController;

use Illuminate\Support\Facades\Route;
```

```
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/
Route::middleware(['auth'])->group(function () {
    Route::get('/', [DashboardController::class, 'showDailySummary'])->name('dashboard');

    Route::resource('users', UserController::class);

    Route::get('tickets', [TicketController::class, 'index'])->name('tickets.index');
    Route::get('tickets/create', [TicketController::class, 'create'])->name('tickets.create');
    Route::post('tickets/generate', [TicketController::class, 'generate'])->name('tickets.generate');

    Route::get('/user/tickets', [TicketController::class, 'showAssignedTickets'])->name('user.tickets');
    Route::get('/user/tickets/assign', [TicketController::class, 'showRequestForm'])->name('user.tickets.show.assign');
    Route::post('/user/tickets/assign', [TicketController::class, 'handleRequest'])->name('user.tickets.assign');

    Route::get('/tickets/redeem', [TicketController::class, 'showRedeemForm'])->name('tickets.redeem.show');
    Route::post('/tickets/redeem', [TicketController::class, 'redeemTicket'])->name('tickets.redeem');

    Route::get('/tickets/report', [TicketController::class, 'showReportForm'])->name('tickets.report.form');
    Route::get('/tickets/report/generate', [TicketController::class, 'generateReport'])->name('tickets.report');

    Route::get('/tickets/report/single-date', [TicketController::class, 'generateSingleDateReport'])-
>name('tickets.report.single_date');
    Route::get('/tickets/report/date-range', [TicketController::class, 'generateDateRangeReport'])-
>name('tickets.report.date_range');

    Route::get('/user/tickets/select-monthly-report', function () {
        return view('user.select_monthly_report');
    })->name('user.tickets.select_monthly_report');
    Route::get('/user/tickets/monthly-report', [TicketController::class, 'monthlyReportForUser'])-
>name('user.tickets.monthly_report');
});

Route::get('login', [AuthController::class, 'show'])->name('login');
Route::post('login', [AuthController::class, 'login']);
Route::post('logout', [AuthController::class, 'logout'])->name('logout');
```

Implementación

Docker juega un rol muy importante en el proceso de implementación de proyectos de software modernos. Algunas de las principales formas en que Docker facilita la implementación son:

Estandarización del entorno: Docker permite empaquetar una aplicación junto con todas sus dependencias en un contenedor estandarizado. Esto asegura que la aplicación se ejecutará de la misma manera en cualquier entorno (desarrollo, pruebas, producción) que tenga Docker instalado.

Portabilidad: Los contenedores Docker se pueden ejecutar en cualquier máquina con Docker instalado, facilitando el despliegue de la aplicación en diferentes entornos. No es necesario preocuparse por incompatibilidades de dependencias o versiones del sistema operativo.

Escalabilidad: Es fácil escalar aplicaciones en contenedores creando nuevas instancias a medida que aumenta la demanda. Los contenedores también facilitan la implementación en entornos de nube elásticos.

Despliegues rápidos: Los contenedores Docker son muy livianos en comparación con las máquinas virtuales, por lo que se pueden desplegar muy rápidamente. Esto facilita entregas e implementaciones continuas de la aplicación.

Seguridad: Docker aísla procesos en contenedores, agregando una capa adicional de seguridad a implementaciones en producción.

Gestión: Docker viene con herramientas incorporadas para administrar y monitorear contenedores a lo largo de todo el ciclo de vida de la aplicación.

En resumen, en el proyecto utiliza Docker, para estandarizar, agilizar y facilitar el despliegue de la aplicación.

El proceso para utilizar Docker consiste en los siguientes pasos:

Desarrollar la aplicación y crear un Dockerfile

El Dockerfile es un archivo de texto que contiene los comandos para ensamblar la imagen de Docker. Aquí se definen el sistema operativo base, las dependencias y librerías necesarias, los archivos y directorios a copiar desde el host, puertos a exponer, variables de entorno, etc. A continuación podemos observar el dockerFile de nuestra aplicación.

```
---
version: '3.8'

services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: app
    env_file:
      - .env.local
    restart: unless-stopped
    tty: true
    ports:
      - "8000:8000"
    volumes:
      - ./app:/var/www/app
      - ./config:/var/www/config
      - ./database:/var/www/database
      - ./resources:/var/www/resources
      - ./routes:/var/www/routes
      - ./tests:/var/www/tests
      - ./phpunit.xml:/var/www/phpunit.xml
      - ./composer.json:/var/www/composer.json
      - ./composer.lock:/var/www/composer.lock
      - ./.env:/var/www/.env
    depends_on:
      - db
      - redis
```

```
  db:
    image: postgres:alpine
    container_name: db
    restart: unless-stopped
    environment:
      POSTGRES_DB: tracker
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    ports:
      - "6432:5432"
    volumes:
      - dbdata:/var/lib/postgresql/data

  redis:
    image: redis:alpine
    container_name: redis
    restart: unless-stopped
    ports:
      - "6379:6379"

volumes:
  dbdata:
```

Construir la imagen de Docker

Usando el comando docker build para generar la imagen a partir del Dockerfile. Procedemos a construir las imágenes de los servicios definidos en tu archivo docker-compose.yml. Este comando busca el archivo de configuración y crea las imágenes necesarias para tus servicios.

```
docker-compose --project-name escome build
```

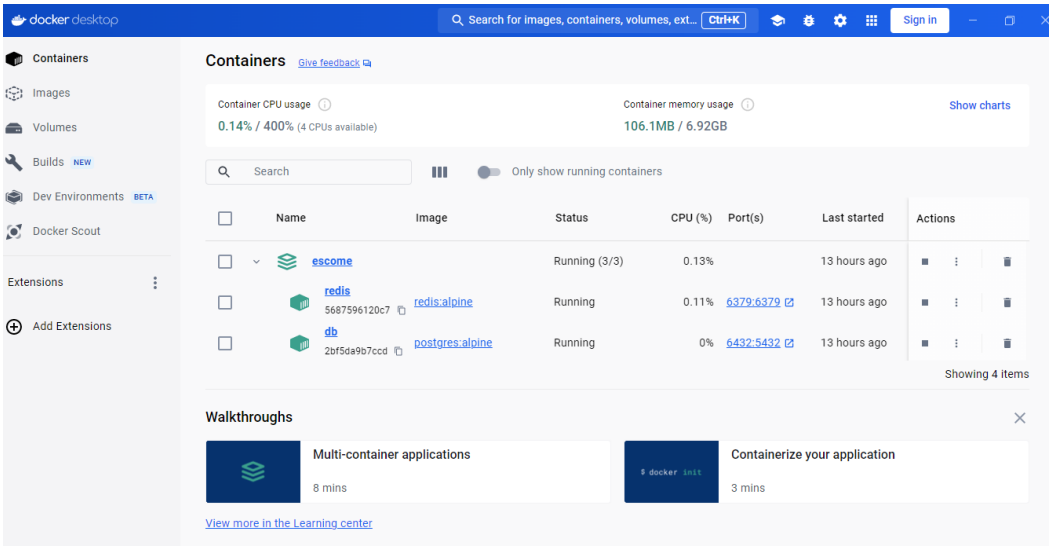
Con el siguiente comando inicia los servicios definidos en tu archivo docker-compose.yml. Los servicios se ejecutan en segundo plano (modo detached) debido al argumento

```
docker compose --project-name escome up --detach --remove-orphans
```

Procedemos a instalar Docker en la Pc de producción como lo muestra la figura 31.

Figura 31

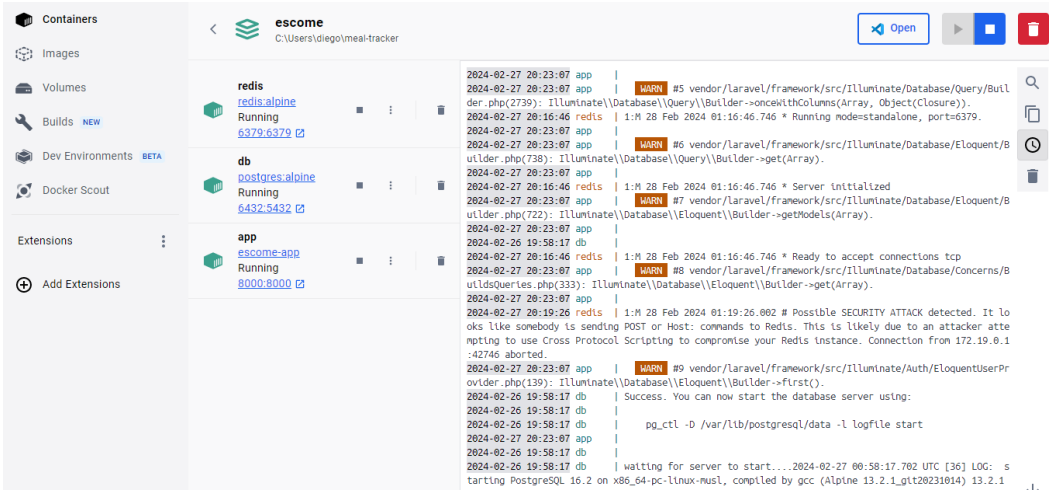
Docker Desktop



Procedemos a revisar los logs, para identificar que todo esté funcionando correctamente, que se hayan levantado todos los servicios e instalado todas las dependencias como lo muestra la figura 38.

Figura 32

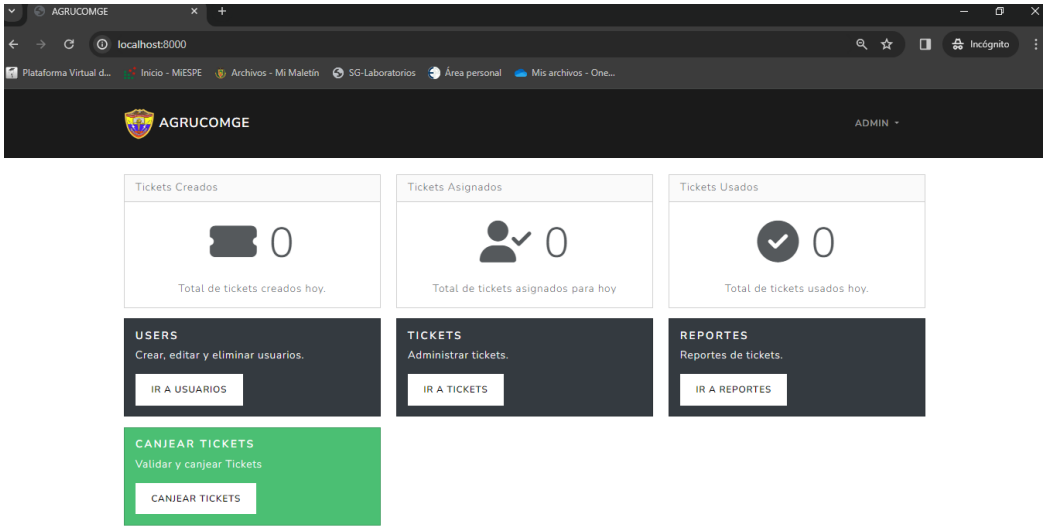
Logs Docker



Procedemos a probar si se ha desplegado la infraestructura Docker con todos los elementos aprovisionados y que son requeridos para que la aplicación funcione.

Figura 33

Despliegue de la Aplicación



Como se puede observar en la figura 33, la aplicación está desplegada de manera correcta. Esto nos evidencia que usar una infraestructura Docker, facilita los procesos de implementación de software, evitando fallos de librerías y componentes.

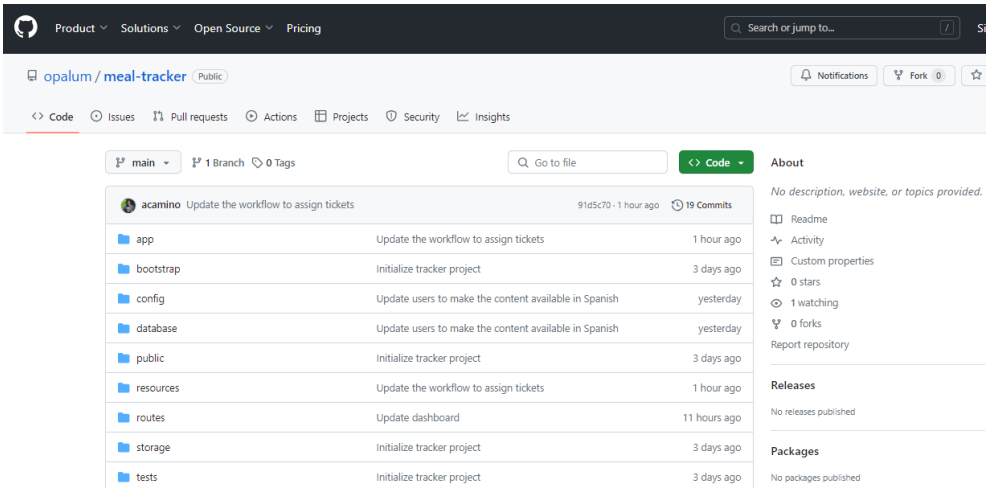
Control de versiones

En el proceso de control de versiones y pruebas para el proyecto se utilizó GitHub, primero se configura un repositorio en la plataforma para el proyecto. Luego, se procede a clonar el repositorio en la máquina local utilizando Git, lo que te permite trabajar con el código de manera independiente. Se desarrolla y modifica el código localmente, agregando y confirmando cambios mediante los comandos de Git. Antes de enviar los cambios al repositorio remoto en GitHub, se realiza pruebas locales para garantizar su funcionalidad. Pensando en un entorno colaborativo, podemos utilizar Pull Requests para enviar los cambios a una rama

específica y solicitar la revisión de otros desarrolladores antes de fusionarlos con la rama principal. En la siguiente imagen podemos observar el repositorio creado en GitHub.

Figura 34

Repositorio Github



Aplicamos los comandos requeridos mediante nuestra interfaz bash de github, con la finalidad de clonar el repositorio y luego descargar los cambios realizados en el código.

```
$ git clone https://github.com/opalum/meal-tracker.git
```

```
$ git pull --ff-only origin main
```

Pruebas

En el proceso de realizar pruebas con los seeders de Laravel, primero se crea un seeder utilizando el comando artisan proporcionado por el framework. Este seeder actuará como una especie de plantilla para los datos de prueba que deseamos insertar en la base de datos de nuestra aplicación. Una vez creado, nos sumergimos en la edición del seeder, donde meticulosamente agregamos los datos que consideramos relevantes para las pruebas de nuestra aplicación. Con cada registro cuidadosamente definido, procedemos a ejecutar los seeders utilizando el comando db:seed de artisan, poblándose así la base de datos con los

datos de prueba que acabamos de definir. Ahora, con un conjunto sólido de datos de prueba en su lugar, preparamos el terreno para la escritura de nuestras pruebas. Configuramos un entorno de prueba dedicado, asegurando que nuestras pruebas no interfieran con los datos de nuestras bases de datos de desarrollo o producción. Con el escenario listo, comenzamos a escribir nuestras pruebas, utilizando PHPUnit y las herramientas de prueba proporcionadas por Laravel para verificar el comportamiento de nuestra aplicación con estos datos.

Seeder User

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;

class AdminUserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $adminRoleId = DB::table('roles')->where('name', 'Admin')
                                ->first()
                                ->id;

        DB::table('users')->insert([
            'identity'   => 'admin',
            'name'       => 'Admin',
            'email'      => 'admin@example.com',
            'phone'      => '',
            'role_id'    => $adminRoleId,
            'password'   => Hash::make('tesseract'),
            'created_at' => now(),
            'updated_at' => now(),
        ]);
    }
```

```
}
```

Seeder Comidas

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class MealSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        DB::table('meals')->insert([
            [
                'name'       => 'Desayuno',
                'created_at' => now(),
                'updated_at' => now(),
            ],
            [
                'name'       => 'Almuerzo',
                'created_at' => now(),
                'updated_at' => now(),
            ],
                    [
                        'name'        => 'Merienda',
                        'created_at' => now(),
                        'updated_at' => now(),
                    ],
            ]);
        }
}
```

Seeder Roles

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class RolesTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        DB::table('roles')->insert([
            [
                'name'       => 'Comensal',
                'created_at' => now(),
                'updated_at' => now(),
            ],
            [
                'name'       => 'Admin',
                'created_at' => now(),
                'updated_at' => now(),
            ],
            [
                'name'       => 'Cocinero',
                'created_at' => now(),
                'updated_at' => now(),
            ],
        ]);
    }
}
```

Capítulo IV

Conclusiones y Recomendaciones

Conclusiones

- Tras realizar un estudio detallado del proceso actual del comedor, en colaboración estrecha con el encargado de este, se logró identificar los puntos críticos que afectaban la eficiencia operativa. Esta fase fue crucial para comprender las necesidades específicas del entorno y establecer una base sólida para el diseño e implementación del sistema de gestión.
- La definición de los requisitos de interfaz de usuario y el diseño de prototipos utilizando herramientas de wireframes fueron pasos fundamentales en el proceso de desarrollo del sistema. Este enfoque garantizó que la aplicación web cumpliera con las expectativas de usabilidad y funcionalidad, al tiempo que proporcionaba una experiencia intuitiva para los usuarios finales.
- Al definir casos de prueba exhaustivos que abarcaran todos los aspectos funcionales y de usabilidad de la aplicación, se garantizó la calidad y fiabilidad del sistema. Las pruebas de integración y aceptación realizadas por los usuarios finales jugaron un papel crucial en la validación del sistema, identificando posibles fallos y áreas de mejora.

Recomendaciones

Implementar el uso de contenedores Docker para crear un entorno de desarrollo reproducible y portátil, facilitará la configuración del entorno de desarrollo en diferentes equipos y garantizará que todos los miembros del equipo estén trabajando en la misma base de código. Además, Docker puede utilizarse para el despliegue del sistema en diferentes entornos, como pruebas y producción, proporcionando consistencia y facilitando la escalabilidad.

Aprovechar las funcionalidades y la estructura de Laravel, un framework PHP moderno y potente, ayuda a acelerar el desarrollo del sistema. Laravel proporciona una amplia gama de herramientas y características, como el enrutamiento simplificado, el sistema de plantillas Blade y la gestión de sesiones, que pueden mejorar la productividad del equipo y la calidad del código

Implementar un flujo de trabajo basado en Git para el control de versiones del código fuente del proyecto. Esto permitirá a los miembros del equipo colaborar de manera eficiente, mantener un historial de cambios y revertir fácilmente a versiones anteriores si es necesario. Además, Git facilita la integración continua y la entrega continua (CI/CD), lo que permite automatizar las pruebas y despliegues del sistema de manera eficiente.

Bibliografía

Aranda, J. (2018). *FORTALECIMIENTO DEL FRONTEND Y BACKEND*.

Arbeláez, O. (2011). HERRAMIENTAS PARA EL DESARROLLO RÁPIDO DE APLICACIONES WEB. *Scientia et Technica Año XVII*, *47*.

Arce, A. E. V. (2016). *De la interfaz del usuario al responsive web design* (Número 37). www.alistapart.com,

Asamblea Nacional. (2021). *LEY ORGÁNICA DE PROTECCIÓN DE DATOS PERSONALES*. www.lexis.com.ec

Bravo Carrasco, Juan. (2011a). *Gestión de procesos: (Alineados con la estrategia).* Evolución.

Bravo Carrasco, Juan. (2011b). *Gestión de procesos: (Alineados con la estrategia).* Evolución.

Camazón, J. N. (2009). *Aplicaciones web*.

Camisón Zornoza, C., Cruz, Sonia., & González, T. (2007). *Gestión de la calidad : conceptos, enfoques, modelos y sistemas*. Pearson/Prentice Hall.

Carreño Dueñas, D. A., Amaya González, L. F., Ruiz Orjuela, E. T., & Javier Tiboche, F. (2019). Diseño de un sistema para la gestión de inventarios de las pymes en el sector alimentario. *Industrial Data*, *22*(1), 113-132. https://doi.org/10.15381/idata.v22i1.16530

Chancay, R. (2015). *GESTIÓN OPERATIVA DE RESTAURANTES PARA EL MEJORAMIENTO DE LOS SERVICIOS GASTRONÓMICOS EN LA PARROQUIA CALCETA DE LA PROVINCIA DE MANABÍ*.

Congreso Nacional. (2006). *LEY DE PROPIEDAD INTELECTUAL*. www.lexis.com.ec

Espinosa-Hurtado, R. (2021). Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web. *CEDAMAZ*, *11*(2). https://doi.org/10.54753/cedamaz.v11i2.1182

Graciela, S., Ibarra, P., Quispe, R., Mullicundo, F. F., & Lamas, D. A. (2021). *HERRAMIENTAS Y TECNOLOGÍAS PARA EL DESARROLLO WEB DESDE EL FRONTEND AL BACKEND*. https://www.campusmvp.es/recursos/post/Desar

Haris, S. (2022). Performance Analysis of Openlitespeed and Apache Web Servers in Serving Client Requests. *Knowbase : International Journal of Knowledge in Database*, *2*(2), 114. https://doi.org/10.30983/ijokid.v2i2.5306

Hossian, A. (2012). *MODELO DE PROCESO DE CONCEPTUALIZACIÓN DE REQUISITOS*.

Inés Lund, M., Ferrarini, C., Aballay, L., Romagnano, M. G., & Meni, E. (s. f.). *CUPIDo-Plantilla para Documentar Casos de Uso*. http://www.vico.org/aRecursosPrivats/UML_TRAD/talleres/mapas/

Lerma-Blasco, R. V., Murcia, J. Alfredo., & Mifsud Talón, A. Elvira. (2013). *Aplicaciones web*. McGraw-Hill/Interamericana de España.

López, P. (2016). *Comparación del desempeño de los Sistemas Gestores de Bases de Datos MySQL y PostgreSQL*.

Macías, M. (2023). *Optimización de procesos en compras para mejorar estrategias en restaurantes*. https://orcid.org/0000-0002-2888-6363

Maldonado, J. Á. (2018). *GESTIÓN DE PROCESOS*.

Mendoza-Fernandez, V. M., & Sobeida Moreira-Chóez, J. (2021). *Procesos de Gestión Administrativa, un recorrido desde su origen Administrative Management Processes, a*

journey from its origin Processos de Gestão Administrativa, uma jornada desde suas origens. *6*(25), 608-620. https://doi.org/10.23857/fipcaec.v6i3.414

Morales Catherin. (2015). *Diseño de un sistema de inventario para la empresa Service Lunch*.

Ovando, D. (2019). *Bootstrap y Laravel, herramientas para el desarrollo de aplicaciones web*.

Pacheco, J., & Rosero, L. (1999). *Arquitectura Cliente Servidor*.

Pacienzia, G. (2015). *Metodologías de desarrollo de software*.

Pantoja, L., & Pardo, C. (2016). Evaluando la Facilidad de Aprendizaje de Frameworks mvc en el Desarrollo de Aplicaciones Web. *Publicaciones e Investigación*, *10*. https://doi.org/10.22490/25394088.1592

Quizhpi, D. (2018). *Diseño de un Sistema de control de inventario y organización de las bodegas de producto terminado de la empresa LAMITEX*.

Rios, F., & Leticia, B. (2001). *Ingeniería y modelado de procesos.*

Rodriguez, J. (2016). *Diseño y Mejoramiento del Sistema de Inventario Informático de la Empresa MAGREB S.A*.

Santos, M. (2022). *Marco Regulatorio de la ciberseguridad y ciberdefensa dentro de la sociedad de la información y el conocimiento*.

Stallman, R. (2019). *La definición de Software libre*.

Subecz, Z. (2021). Web-development with Laravel framework. *Gradus*, *8*(1). https://doi.org/10.47833/2021.1.csc.006

Tejeda, A. (2011). *GESTIÓN OPERATIVA DE RESTAURANTES PARA EL MEJORAMIENTO DE LOS SERVICIOS GASTRONÓMICOS EN LA PARROQUIA CALCETA DE LA PROVINCIA DE MANABÍ*.

Torres-Del-Rey, J., & Rodríguez-V, E. (2014). La localización de webs dinámicas: objetos, métodos, presente y futuro. En *The Journal of Specialised Translation Issue* (Vol. 21).

Zaratiegui, J. (1999). *La gestión por procesos*.

Anexos