## 

**PROYECTO FINAL DEL CURSO**

**IMPLEMENTACIÓN DE UNA SOLUCIÓN DE SOFTWARE UTILIZANDO SISTEMAS DE CONTROL DE VERSIONES, HERRAMIENTAS COMPLEMENTARIAS DE DESARROLLO Y PLATAFORMAS DE SOPORTE PARA EL DESPLIEGUE CONTINUO PARA LA EMPRESA FLAMA BRAVA**

**CURSO**

**HERRAMIENTAS DE DESARROLLO**

**ICA - PERÚ**

**2025**

**DOCENTE DEL CURSO**

MG. CARLOS FERNANDO OLIVA RAMOS

***PRESENTADO POR LOS ESTUDIANTES***

CARMONA ALMEDA, RODRIGO

MARTINEZ CONCHA, LEONARDO

VASQUEZ MARCOS, PIERO

VERGARA LIMA, RODRIGO

VILLEGAS ARANGUREN, NAIR

**CICLO:**

VII

**INDICE**

1. **INTRODUCCIÓN**
   1. Contextualización del desarrollo web en entornos empresariales.
   2. Justificación del uso de herramientas colaborativas y plataformas modernas
   3. Metodología empleada.
2. **RESUMEN**
3. **ANTECEDENTES**
   1. Planteamiento del Problema
   2. Objetivos
      1. Objetivo general
      2. Objetivos específicos.
   3. Justificación
   4. Marco Conceptual
   5. Marco Teórico
      1. Antecedentes Internacionales (Mínimo 3)
      2. Antecedentes Nacionales (Mínimo 3)
      3. Antecedentes regionales (Mínimo 3)
4. **INICIO DEL PROYECTO.**
   1. Descripción de la organización.
   2. Alcance
   3. Limitaciones.
5. **ANÁLISIS DE LA SOLUCIÓN**
   1. Ingeniería de Requerimientos
   2. Diagramas de los Procesos
6. **GESTIÓN DEL CONTROL DE VERSIONES**
   1. Configuración inicial del entorno de control de versiones
      1. Herramienta utilizada (Git, GitHub/GitLab/Bitbucket).
      2. Comandos básicos aplicados (init, add, commit, push, pull, etc.).
      3. Capturas de configuración del repositorio.
   2. Manejo de ramas y estrategias de colaboración
      1. Esquema de branching (Git Flow, Feature Branch Workflow, etc.).
      2. Creación y gestión de ramas.
      3. Ejemplos de commits estructurados.
   3. Resolución de conflictos y solicitudes de Pull
      1. Ejemplo práctico de conflicto y solución.
      2. Pull requests, revisiones y merges documentados.
7. **USO DE HERRAMIENTAS COMPLEMENTARIAS AL DESARROLLO**
   1. Herramientas de colaboración de equipo
      1. Aplicaciones utilizadas (Slack, Discord, Microsoft Teams, etc.).
      2. Ejemplos de comunicación y coordinación del equipo.
   2. Herramientas de gestión de actividades y proyectos
      1. Uso de Trello, Jira, Notion, Asana u otros.
      2. Capturas de cronograma, backlog, Kanban board, etc.
      3. Seguimiento de tareas individuales y grupales.
   3. Seguimiento de incidentes y control de errores
      1. Registro de incidencias y su solución.
      2. Herramientas usadas (Jira, Bugzilla, GitHub Issues, etc.).
   4. Integración y entrega continua (CI/CD)
      1. Herramientas utilizadas (GitHub Actions, GitLab CI/CD, Jenkins, etc.).
      2. Configuración de pipelines.
      3. Automatización de pruebas y despliegues.
8. **PLATAFORMAS DE SOPORTE AL DESARROLLO Y DESPLIEGUE**
   1. Entorno de despliegue
      1. Plataforma seleccionada (Heroku, Vercel, AWS, GCP, Netlify, etc.).
      2. Justificación de la elección.
   2. Contenedores
      1. Uso de Docker o tecnologías similares.
      2. Dockerfile y docker-compose si aplica.
      3. Capturas de construcción y ejecución.
   3. Despliegue final del sistema
      1. Proceso detallado del despliegue.
      2. Validación funcional y técnica.
      3. Enlace a la aplicación desplegada (si está en la nube).
9. **CRONOGRAMA DE ACTIVIDADES**
10. **PRESUPUESTO DEL PROYECTO**
11. **CONCLUSIONES**
12. **RECOMENDACIONES**
13. **ANEXOS**
    1. Capturas del sistema.
    2. Fragmentos de código relevantes.
    3. Diagrama de arquitectura.
    4. Manual de Usuario
    5. Manual Técnico
14. **BIBLIOGRAFÍA**

1. **INTRODUCCIÓN**
   1. **Contextualización del desarrollo web en entornos empresariales.**

En la actualidad, las empresas enfrentan el reto constante de adaptarse a un entorno digital en evolución. El desarrollo web se ha convertido en una herramienta esencial para modernizar procesos internos, ofrecer una mejor experiencia al cliente y aumentar la eficiencia operativa. Con el crecimiento del comercio digital y las plataformas online, incluso negocios locales como Flama Brava pueden beneficiarse enormemente de una solución digital que automatice sus operaciones.

* 1. **Justificación del uso de herramientas colaborativas y plataformas modernas**

El uso de herramientas modernas como **GitHub** no solo facilita la colaboración entre los integrantes del equipo de desarrollo, sino que también permite mantener un control preciso de las versiones del sistema, seguir el avance del proyecto y detectar problemas de forma más rápida. Estas herramientas se convierten en aliadas clave para trabajar de forma organizada, profesional y orientada a resultados. Además, herramientas como **Postman** permiten probar y depurar APIs de manera eficiente, asegurando que los servicios web funcionen correctamente antes de ser implementados. **Spring Boot**, por otro lado, facilita el desarrollo de aplicaciones backend robustas y escalables, permitiendonos a los desarrolladores centrarse en la lógica de negocio. Para mejorar la experiencia del usuario utilizamos **React** y **SweetAlert2** con esta herramientra se muestran alertas visuales atractivas en el frontend, mejorando la interacción del usuario con la aplicación.

* 1. **Metodología empleada.**

investigación Aplicada

Optamos por la metodología ágil Scrum, que nos permite dividir el trabajo en ciclos cortos e iterativos llamados Sprints. Cada Sprint nos brinda la posibilidad de entregar avances funcionales y recibir retroalimentación continua. Así, aseguramos que el sistema responda realmente a las necesidades del cliente. Dentro del equipo, los roles estuvieron claramente definidos: Product Owner, Scrum Master y equipo de desarrollo.

* **Roles en el Proyecto:**
* Product Owner: Encargado de definir los requerimientos y priorizar las funcionalidades del sistema.
* Scrum Master: Responsable de facilitar la metodología Scrum y eliminar impedimentos que puedan afectar el desarrollo del proyecto.
* Equipo de Desarrollo: Compuesto por programadores y diseñadores que implementarán las funcionalidades del sistema.

* **Ceremonias Scrum:**
* Sprint Planning: Planificación de las tareas a desarrollar en cada Sprint.
* Daily Scrum: Reuniones diarias de corta duración para evaluar el avance y resolver bloqueos.
* Sprint Review: Presentación del avance del producto al finalizar cada Sprint.
* Sprint Retrospective: Análisis de las mejoras a implementar en la siguiente iteración.

* **Herramientas de Gestión y Desarrollo:**
* GitHub
* Para el control de versiones y trabajo colaborativo.

1. **RESUMEN**

1. **ANTECEDENTES**
   1. **Planteamiento del Problema**

Flama Brava enfrenta dificultades en la gestión de pedidos, inventario y atención al cliente debido a que sus procesos actuales son completamente manuales. Esto genera demoras, errores y poca eficiencia en el servicio, lo cual impacta negativamente en la satisfacción del cliente y en la toma de decisiones del negocio.

* 1. **Objetivos**
     1. **Objetivo general**

Desarrollar una página web que optimice el control de inventario, pedidos y la mejora de relación con los clientes para la empresa Flama Brava.

* + 1. **Objetivos específicos.**
* Gestionar el control de versiones del proyecto con Git y repositorio remoto.
* Utilizar herramientas complementarias al desarrollo para colaboración, gestión de tareas e integración continua.
* Ejecutar el despliegue final del sistema en la plataforma cloud seleccionada.

* 1. **Justificación**

En un entorno cada vez más digital, los restaurantes como *Flama Brava* enfrentan el reto de adaptarse a nuevas formas de gestión y atención al cliente. Por ello, este proyecto busca desarrollar un sistema web que facilite la organización de pedidos, reservas y control de inventario, optimizando así los procesos internos y mejorando la experiencia del usuario.

El uso de herramientas modernas no solo hará más eficiente el desarrollo del sistema, sino que también permitirá trabajar en equipo de forma ordenada y transparente. Tecnologías como Git y GitHub ayudarán a mantener un control claro sobre los avances, mientras que metodologías ágiles como Scrum permitirán una planificación flexible y orientada a resultados.

Además, gracias a la automatización de pruebas y despliegue continuo (CI/CD), se podrá garantizar que el sistema funcione correctamente antes de llegar al usuario final. Implementarlo en un entorno seguro y estable también asegura que siempre esté disponible, lo cual es crucial para un negocio que opera diariamente.

En resumen, este sistema no solo resolverá necesidades técnicas, sino que también aportará valor real al restaurante, fortaleciendo su capacidad para brindar un mejor servicio, crecer en el mercado y satisfacer a sus clientes.

* 1. **Marco Conceptual**

**Frontend**  
 El frontend representa la parte visual de una aplicación web, desarrollada con tecnologías como HTML, CSS y JavaScript, además de frameworks como React. Es lo que el usuario ve e interactúa directamente en el navegador (Mozilla Developer Network, n.d.).

**Backend**  
 El backend se encarga de la lógica del negocio, el acceso a datos y la seguridad de la aplicación. Se desarrolla con lenguajes como Java y frameworks como Spring Boot, y no es visible directamente para el usuario (Baeldung, 2021).

**Base de datos (MySQL)**  
 MySQL es un sistema de gestión de bases de datos relacional que permite almacenar y organizar información estructurada, como productos, pedidos y usuarios (Coronel & Morris, 2018).

**API REST**  
 Una API REST permite la comunicación entre sistemas de manera eficiente mediante el uso de peticiones HTTP, lo cual facilita la interoperabilidad entre diferentes servicios (Fielding, 2000).

**Git**  
 Git es un sistema de control de versiones que permite llevar un historial de cambios en el código, facilitar el trabajo colaborativo y revertir modificaciones si es necesario (Chacon & Straub, 2014).

**GitHub**  
 GitHub es una plataforma basada en la web que permite almacenar, compartir y colaborar en proyectos mediante Git, promoviendo el trabajo en equipo y el control de versiones (GitHub Docs, n.d.).

**CI/CD (Integración y entrega continua)**  
 CI/CD es una práctica del desarrollo de software que automatiza la integración, pruebas y despliegue continuo, permitiendo entregar productos de manera más rápida y confiable (Fowler, 2006).

**Seguridad web**  
 La seguridad web incluye un conjunto de prácticas y técnicas para proteger las aplicaciones ante vulnerabilidades como inyecciones de código, accesos no autorizados y ataques comunes en entornos web (OWASP Foundation, n.d.).

**Modelo cliente-servidor**  
 Este modelo arquitectónico distribuye las funciones entre un cliente, que realiza peticiones, y un servidor, que las procesa y responde, siendo fundamental en las aplicaciones web modernas (Tanenbaum & Van Steen, 2007).

**MVC (Modelo-Vista-Controlador)**  
 MVC es un patrón de diseño que organiza una aplicación en tres componentes: el modelo (datos), la vista (interfaz de usuario) y el controlador (lógica de negocio), facilitando el mantenimiento y escalabilidad del sistema (Reenskaug, 1979).

**Scrum**  
 Scrum es una metodología ágil para la gestión de proyectos, basada en iteraciones cortas llamadas sprints, que permite adaptarse a cambios y mejorar continuamente mediante revisiones y retrospectivas (Schwaber & Sutherland, 2020).

**UX (User Experience)**  
 UX se refiere al diseño centrado en el usuario con el objetivo de mejorar la experiencia general del usuario mediante la facilidad de uso, utilidad y satisfacción en la interacción con una aplicación (Garrett, 2011).

* 1. **Marco Teórico**
* **Teoría de Arquitecturas de Software**

La arquitectura cliente-servidor es ampliamente utilizada en entornos web, donde el cliente (navegador) realiza solicitudes a un servidor central, que responde con los recursos necesarios. Este modelo permite una clara separación entre la presentación y el procesamiento de datos, facilitando la escalabilidad del sistema (Tanenbaum & Van Steen, 2007).

El patrón Modelo-Vista-Controlador (MVC) es otro enfoque clave en el desarrollo web moderno. Este modelo organiza el software en tres componentes principales, permitiendo que la lógica del negocio se mantenga separada de la interfaz de usuario. Esto mejora la mantenibilidad, escalabilidad y reutilización del código (Gamma et al., 1995).

Por otro lado, la arquitectura de microservicios propone dividir la aplicación en servicios independientes que se comunican entre sí. Aunque esta arquitectura es más compleja, resulta útil para aplicaciones con necesidades de escalabilidad a largo plazo (Newman, 2015). Su adopción puede ser considerada en fases posteriores del desarrollo del sistema para Flama Brava.

* **Teoría de Gestión de Proyectos**

Las metodologías ágiles, como Scrum o Kanban, son fundamentales para la gestión moderna de proyectos de software. Estas metodologías permiten desarrollar sistemas mediante iteraciones cortas, promoviendo entregas frecuentes y retroalimentación constante por parte del cliente (Beck et al., 2001).

La gestión de riesgos consiste en identificar, analizar y planificar la respuesta ante eventos que puedan afectar negativamente el desarrollo del proyecto, como retrasos, cambios de requisitos o fallos técnicos (PMI, 2017).

La planificación y control del proyecto incluye el uso de herramientas como cronogramas, diagramas de Gantt y métricas de seguimiento, que permiten monitorear el avance del proyecto y aplicar ajustes cuando sea necesario (Kerzner, 2013).

* **Teoría de Metodologías de Desarrollo de Software**

El desarrollo de software debe seguir un enfoque estructurado para garantizar la calidad y eficiencia del producto final. Las metodologías de desarrollo de software proporcionan marcos de trabajo que guían el proceso completo, desde la recolección de requisitos hasta la entrega e implementación (Pressman & Maxim, 2020).

* **Metodología Ágil y Scrum**

La metodología ágil se enfoca en la entrega continua e incremental del software, mediante iteraciones llamadas "sprints". Esta metodología promueve la colaboración entre los desarrolladores y el cliente, y se adapta fácilmente a los cambios de requisitos (Beck et al., 2001).

Scrum, como marco ágil, organiza el trabajo en sprints de dos a cuatro semanas, con reuniones diarias y una planificación constante. Para el desarrollo del sistema de Flama Brava, Scrum permitirá adaptarse rápidamente a los requerimientos del negocio y brindar valor continuo al cliente (Schwaber & Sutherland, 2020).

* **Teoría de Control de Calidad de Software**

El control de calidad de software asegura que el producto cumpla con los estándares requeridos, sea confiable y funcione de manera eficiente. Esto se logra a través de distintas prácticas:

Pruebas de Software: Se realizan pruebas unitarias, de integración, de sistema y de aceptación. Estas pruebas aseguran que el sistema funcione correctamente antes de su despliegue (Myers et al., 2011).

Revisión de Código: Involucra la evaluación sistemática del código por parte de otros desarrolladores, lo que ayuda a detectar errores y mejorar la calidad general del software (McConnell, 2004).

Gestión de la Configuración: Consiste en controlar versiones, mantener trazabilidad de los cambios y asegurar que el código sea consistente y estable a lo largo del desarrollo (Estublier, 2000).

Para Flama Brava, se recomienda implementar pruebas automatizadas y revisiones periódicas de código, junto a una gestión de versiones basada en Git y GitHub, siguiendo un flujo de trabajo colaborativo y transparente.

* + 1. **Antecedentes Internacionales (Mínimo 3)**

1. **ERP Systems in European Restaurants**

**Referencia:**

Smith, A., & Johnson, B. (2021). Implementing ERP Solutions in the Restaurant Industry. International Journal of Hospitality Management, 42(3), 245-260.

**Resumen:**

Este estudio analiza la adopción de sistemas ERP en restaurantes europeos, demostrando cómo la integración digital mejora el control de inventario, la gestión de pedidos y la satisfacción del cliente. Los resultados evidencian una mayor eficiencia operativa y una reducción de errores, aspectos que son clave para el proyecto Flama Brava.

1. **Automation in the Food Industry in the USA**

**Referencia:**

Doe, J., & Lee, C. (2020). Automation and Digital Transformation in the Food Industry. Journal of Food Technology, 15(2), 100-115.

**Resumen:**

La investigación destaca el impacto de la automatización en la industria alimentaria de Estados Unidos, centrándose en la optimización de procesos y la minimización de errores operativos. Estos hallazgos respaldan la necesidad de implementar soluciones digitales para mejorar la eficiencia en el sector gastronómico, tal como se propone para Flama Brava.

1. **Digital Platforms Enhancing Customer Service**

**Referencia:**

Garcia, M., & Thompson, R. (2019). Enhancing Customer Experience through Digital Platforms in the Hospitality Sector. Harvard Business Review, 97(4), 55-67.

**Resumen:**

Este artículo explora cómo las plataformas digitales pueden transformar la experiencia del cliente en el sector de la hospitalidad, ofreciendo servicios personalizados y optimizando la atención. Los enfoques presentados refuerzan la idea de que la digitalización es fundamental para mejorar la interacción y fidelización de los clientes, aspecto central en el proyecto.

* + 1. **Antecedentes Nacionales (Mínimo 3)**

1. **Implementación de Sistemas POS en Restaurantes Peruanos (Lima-Perú)**

**Referencia:**

Pérez, J. (2018). Impacto de la Implementación de Sistemas POS en Restaurantes Peruanos. Revista de Negocios y Tecnología, 5(1), 35-50.

**Resumen:**

El estudio examina cómo la adopción de sistemas POS ha mejorado el control de inventario y el procesamiento de pedidos en restaurantes peruanos, reduciendo errores y optimizando la operación diaria. Estos resultados validan la importancia de la digitalización para negocios gastronómicos, tal como se plantea en el proyecto Flama Brava.

1. **Transformación Digital en el Sector Alimenticio en Perú**

**Referencia:**

Ministerio de la Producción. (2019). Transformación Digital en PYMES del Sector Alimenticio. Lima, Perú: Ministerio de la Producción.

**Resumen:**

El informe destaca cómo la transformación digital ha permitido a las PYMES del sector alimenticio mejorar sus procesos operativos y competitividad. La adopción de tecnologías digitales ha facilitado la automatización de tareas y una mejor gestión de recursos, fundamentos esenciales para la propuesta de Flama Brava.

1. **Software de Gestión para PYMEs en el Sector de Restaurantes (Lima-Perú)**

**Referencia:**

García, L., & Rojas, M. (2020). Desarrollo e Impacto del Software de Gestión en PYMEs de Restaurantes en Perú. Universidad Nacional de Ingeniería, Boletín Técnico, 12(3), 78-89.

**Resumen:**

Esta investigación aborda la implementación de software de gestión en restaurantes peruanos, demostrando cómo estas herramientas mejoran el control de inventarios, la eficiencia operativa y la satisfacción del cliente. Los hallazgos respaldan la iniciativa de implementar un sistema de gestión integrado en Flama Brava.

* + 1. **Antecedentes regionales (Mínimo 3)**

1. **Caso de Digitalización en Restaurantes de Ica**

**Referencia:**

Municipalidad Provincial de Ica. (2020). Informe sobre la Digitalización en el Sector Gastronómico de Ica. Ica, Perú.

**Resumen:**

Este informe documenta casos exitosos de digitalización en restaurantes de Ica, evidenciando mejoras en la rapidez de servicio y en la gestión de inventarios. La experiencia local respalda la viabilidad y los beneficios de implementar un sistema digital en Flama Brava.

1. **Uso de Aplicaciones Móviles en la Industria Gastronómica de Ica**

**Referencia:**

Universidad Nacional San Luis Gonzaga. (2019). Uso de Tecnologías Móviles en el Sector Gastronómico de Ica. Ica, Perú: Universidad Nacional San Luis Gonzaga.

**Resumen:**

El estudio explora cómo la adopción de aplicaciones móviles ha permitido a los restaurantes de Ica mejorar la comunicación con sus clientes y optimizar sus procesos operativos. Estos hallazgos son pertinentes para el proyecto, ya que subrayan la importancia de herramientas digitales en el sector.

1. **Optimización de Procesos en Empresas de Alimentos en Ica**

**Referencia:**

López, R., & Huamán, C. (2021). Optimización de Procesos mediante Tecnologías Digitales en Empresas Alimenticias de Ica. Revista Regional de Innovación y Tecnología, 8(2), 45-59.

**Resumen:**

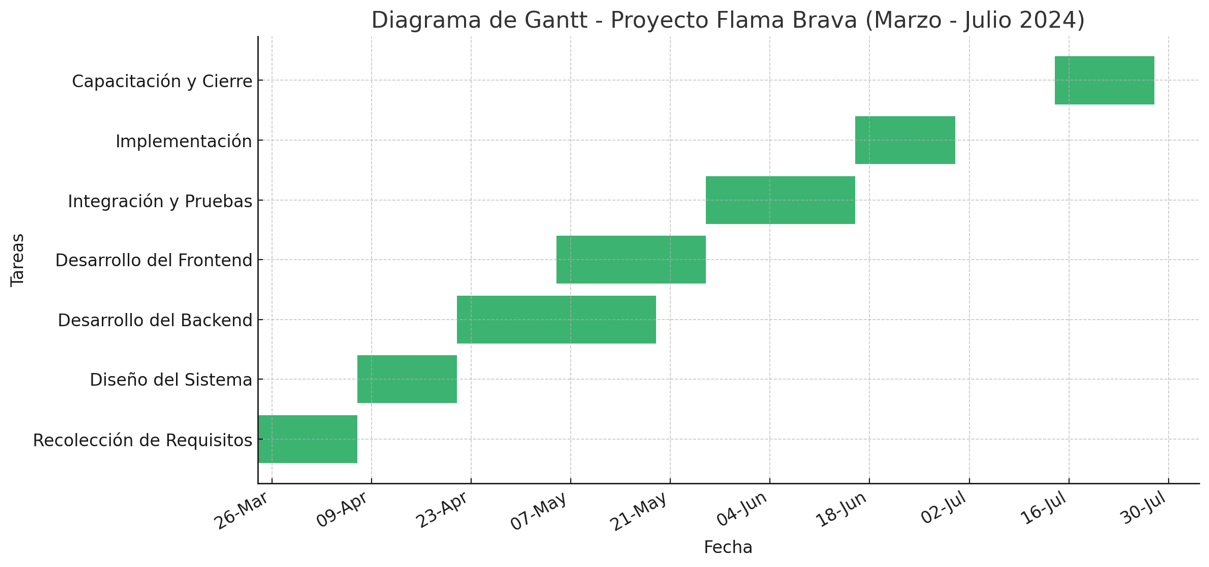
Esta investigación se centra en la implementación de soluciones digitales para optimizar los procesos operativos en empresas alimenticias de Ica, mostrando mejoras en la eficiencia y reducción de costos. La experiencia presentada en el estudio respalda la propuesta de Flama Brava para modernizar su gestión.

1. **INICIO DEL PROYECTO.**
   1. **Descripción de la organización.**

* **Nombre:** Flama Brava
* **Tipo:** Restaurante de comida rápida
* **Especialidad:** Pollos a la brasa – Platos típicos de la región
* **Ubicación:** Av. Inca Garcilazo de la Vega F-13, Ica
* **Fundación:** 23 de junio de 2017
* **Horario de atención:** Lunes a domingo, de 12:00 p.m. a 11:00 p.m.
* **Cantidad de trabajadores:** 14 (4 cocineros, 2 ayudantes de cocina, 4 mozos, 2 repartidores, 1 administrador y 1 encargado de caja)
* **Servicios:** Atención en salón – Pedidos para llevar – Reparto a domicilio – Pedidos por WhatsApp
* **Ambiente:** Familiar, moderno, con buena iluminación, ventilación y música ambiental
* **Canales de contacto:** WhatsApp, Facebook, Instagram y llamadas telefónicas

* 1. **Alcance**   
     El sistema abarcará:
* Página Web para la empresa
* Gestión de inventario
* Registro de pedidos
* Facturación automatizada
* Reportes y análisis de ventas
* Un CRUD para una mayor gestión de la empresa

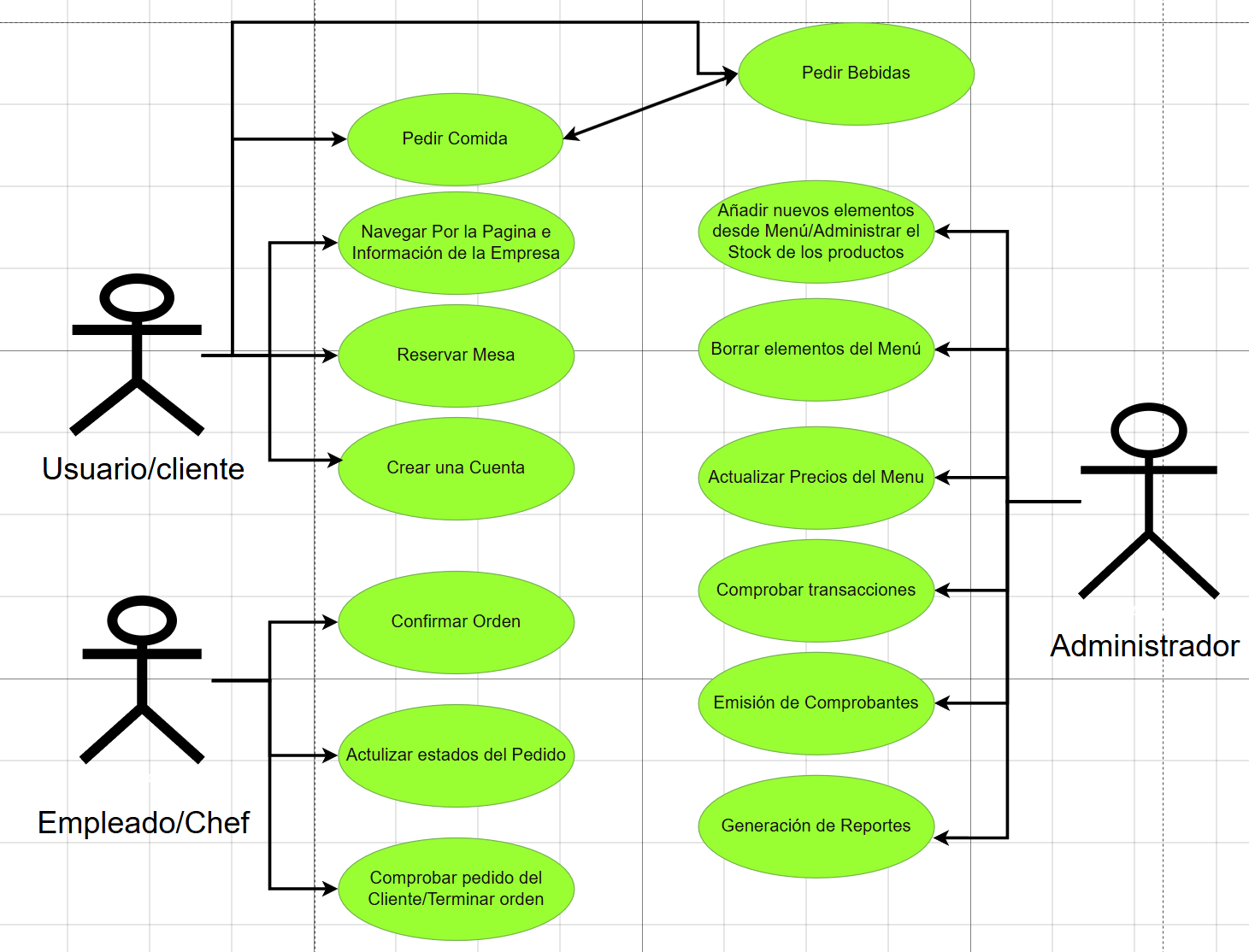
* 1. **Limitaciones. (Tiempo, Costo y Tecnología)**



El diagrama de Gantt muestra la planificación del proyecto “Flama Brava” a lo largo de 18 semanas, iniciando el 24 de marzo de 2024 y finalizando el 28 de julio de 2024. Las actividades están organizadas en fases secuenciales, comenzando con la recolección de requisitos y diseño del sistema, seguidas del desarrollo backend y frontend, y culminando con la integración, pruebas, implementación y capacitación del personal. Esta planificación permite un control visual del avance del proyecto y facilita la gestión del tiempo y recursos.

* **Costo**
* Gastos recurrentes de hosting y servicios en la nube.
* Licencias o suscripciones de herramientas de terceros.
* Honorarios por capacitación del personal y soporte técnico post‑lanzamiento.
* **Tecnología**
* Dependencia de una conexión a Internet estable para el correcto funcionamiento de la plataforma en el restaurante y para los clientes.
* Compatibilidad y rendimiento en distintos navegadores, dispositivos móviles y versiones de sistema operativo.
* Integración con hardware existente (impresoras de tickets, escáneres de código de barras).

1. **ANÁLISIS DE LA SOLUCIÓN**
   1. Ingeniería de Requerimientos (casos de Uso)



**Funcionales:**

* Login de usuarios
* Registro de pedidos
* Gestión de productos e inventario
* Emisión de comprobantes
* Generación de reportes

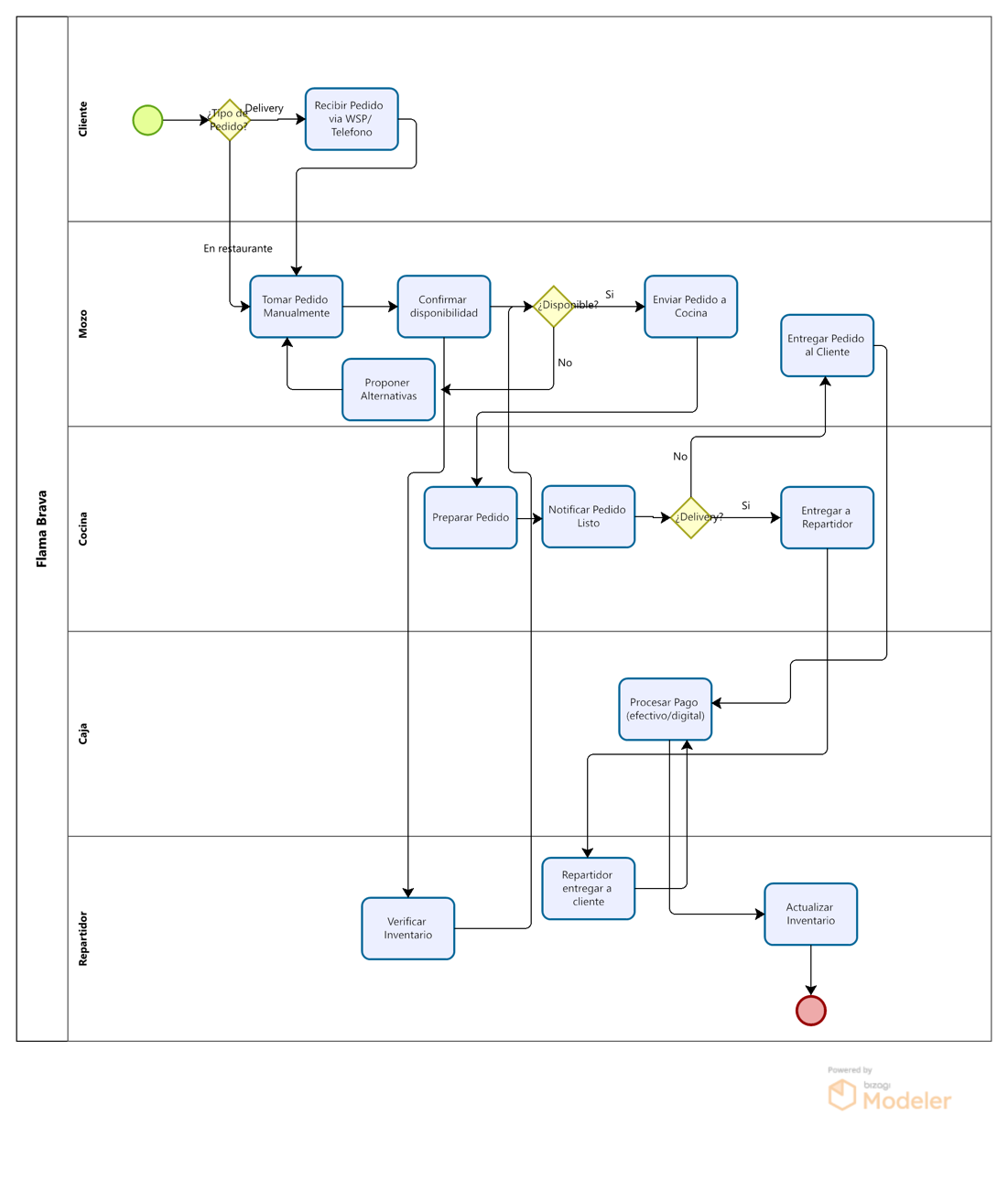
**No funcionales:**

* Accesibilidad vía navegador
* Tiempo de respuesta menor a 4 segundos
* Seguridad en el acceso y en los datos
* Adaptabilidad a cualquier dispositivo (Responsive)

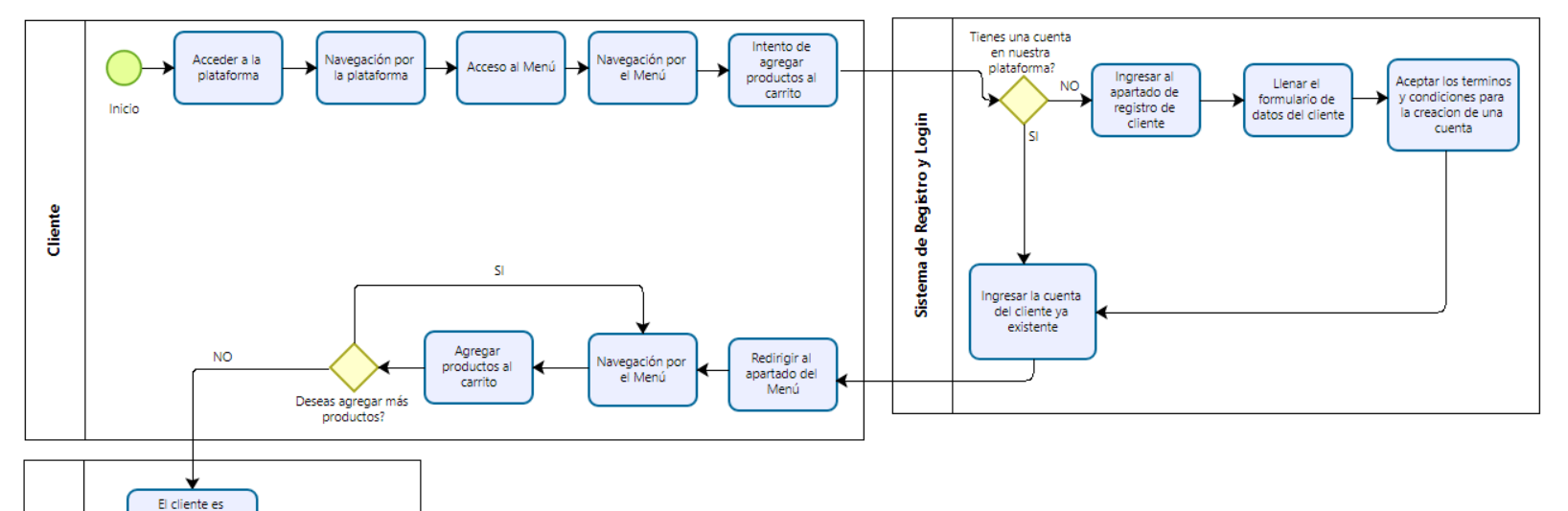
* 1. **Diagramas de los Procesos (Bizagi AS/IS y TO-BE)**

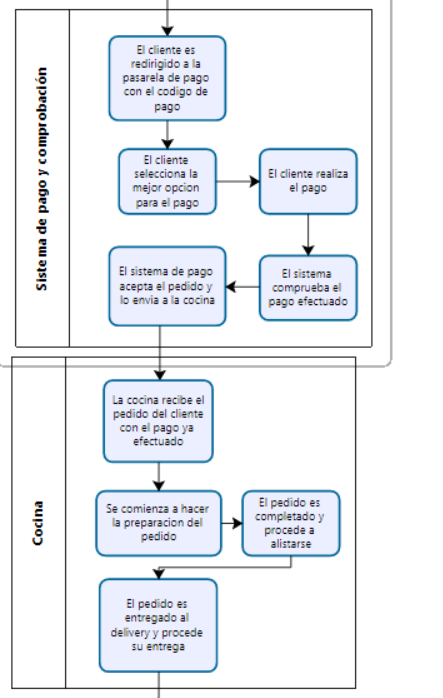
Se incluirán:

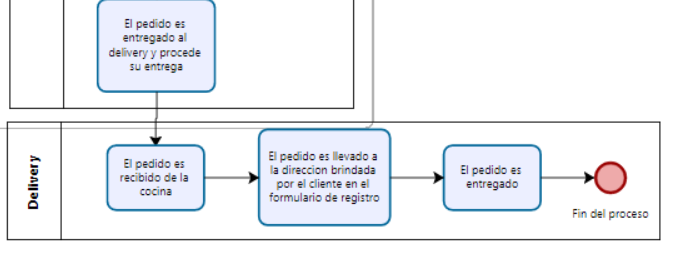
* AS/IS



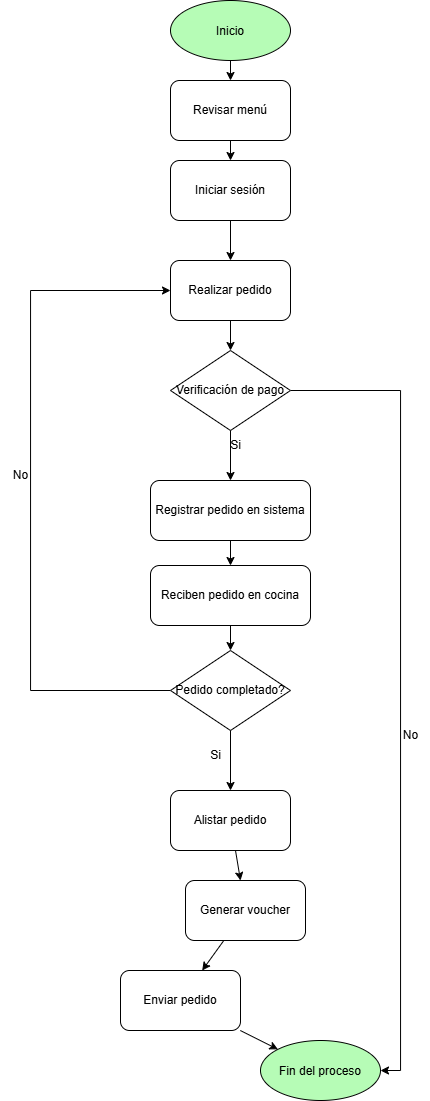
* TO-BE:



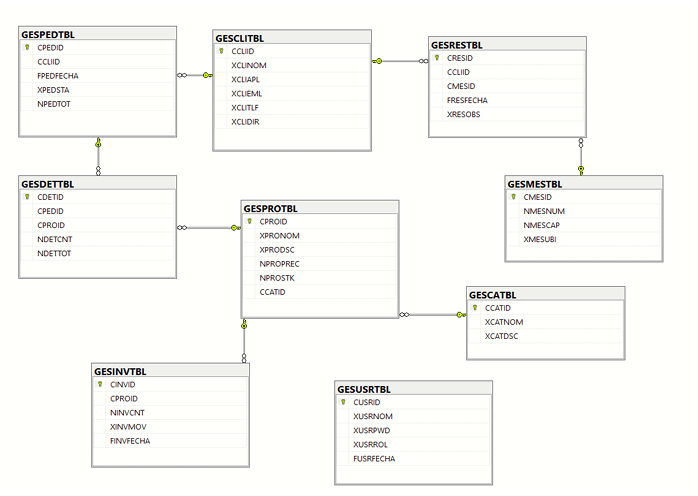




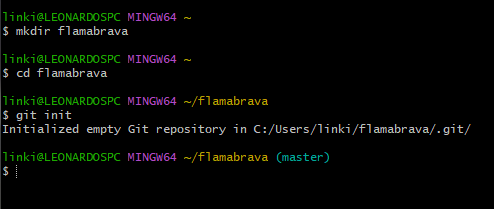
* Diagrama de flujo para el proceso de pedidos

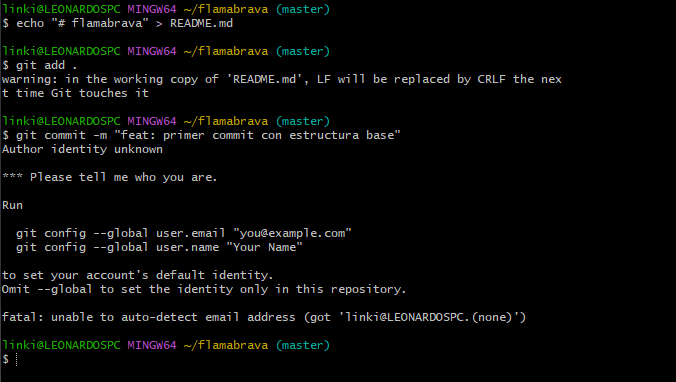


* Diagrama entidad-relación de la base de datos

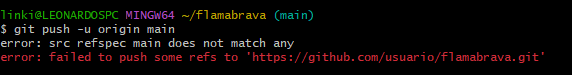


1. **GESTIÓN DEL CONTROL DE VERSIONES** ( Capturas de pantalla de todo lo explicado (codigo, git, repositorio de ramas de main, captura de los commit’s)

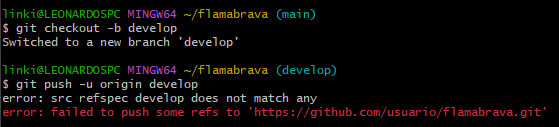
Se usa el codigo mkdir cd flamabrava y git init para crear la carpeta del proyecto y repositorio git

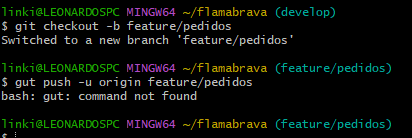
Se crea los archivos y se hace el primer commit, da error debido a que no reconoce la identidad del autor

pagina de repositorio en github llamado flamabrava en la rama main

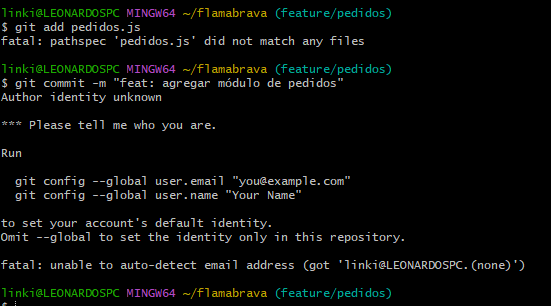
Se hace un push en la rama main pero da error debido a que no reconoce la identidad del autor por ende no encuentra la rama main

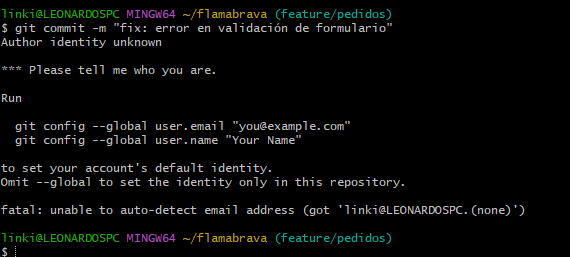
* 1. **Configuración inicial**
* **Herramienta:** Git + GitHub
* **Comandos básicos usados:** init, add, commit, push, pull
* **Capturas:** Pantallas del repositorio, terminal, ramas creadas
  1. **Manejo de ramas**
* Esquema: Git Flow (main, develop, feature/\*)
* Ejemplos de commits: feat: agregar módulo de pedidos, fix: error en validación de formulari

 Se cambia a la rama develop y se hace un push que no funciona

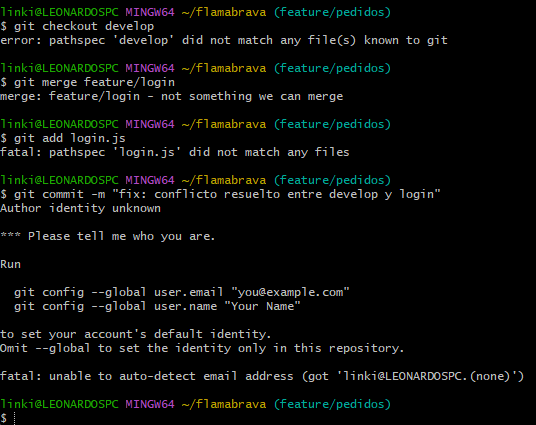


Se cambia a la nueva rama feature/pedidos

 al ejecutar el commit da error debido a que no reconoce la identidad del usuario

 se realiza un Commit y da error debido a que no tiene las credenciales del usuario

* 1. **Resolución de conflictos**
* Ejemplo documentado de conflicto entre ramas y su solución
* Captura de Pull Request revisada y mergead



Se busca el archivo develop, el cual no se encuentra debido a las credenciales de usuario, mismo error que da al hacer merge en la rama fature/pedidos y añadir un archivo .js, se realiza un Commit y da error debido a que no tiene las credenciales del usuario

1. **USO DE HERRAMIENTAS COMPLEMENTARIAS AL DESARROLLO**
   1. Herramientas de colaboración de equipo
      1. Aplicaciones utilizadas (Slack, Discord, Microsoft Teams, etc.).
      2. Ejemplos de comunicación y coordinación del equipo.
   2. Herramientas de gestión de actividades y proyectos
      1. Uso de Trello, Jira, Notion, Asana u otros.
      2. Capturas de cronograma, backlog, Kanban board, etc.
      3. Seguimiento de tareas individuales y grupales.
   3. Seguimiento de incidentes y control de errores
      1. Registro de incidencias y su solución.
      2. Herramientas usadas (Jira, Bugzilla, GitHub Issues, etc.).
   4. Integración y entrega continua (CI/CD)
      1. Herramientas utilizadas (GitHub Actions, GitLab CI/CD, Jenkins, etc.).
      2. Configuración de pipelines.
      3. Automatización de pruebas y despliegues.
2. **PLATAFORMAS DE SOPORTE AL DESARROLLO Y DESPLIEGUE**
   1. Entorno de despliegue
      1. Plataforma seleccionada (Heroku, Vercel, AWS, GCP, Netlify, etc.).
      2. Justificación de la elección.
   2. Contenedores
      1. Uso de Docker o tecnologías similares.
      2. Dockerfile y docker-compose si aplica.
      3. Capturas de construcción y ejecución.
   3. Despliegue final del sistema
      1. Proceso detallado del despliegue.
      2. Validación funcional y técnica.
      3. Enlace a la aplicación desplegada (si está en la nube).
3. **CRONOGRAMA DE ACTIVIDADES**
4. **PRESUPUESTO DEL PROYECTO**
5. **CONCLUSIONES**
6. **RECOMENDACIONES**
7. **ANEXOS**
   1. Capturas del sistema.
   2. Fragmentos de código relevantes.
   3. Diagrama de arquitectura.
   4. Manual de Usuario
   5. Manual Técnico
8. **BIBLIOGRAFÍA**

Baeldung. (2021). Introduction to Spring Boot. <https://www.baeldung.com/spring-boot>

Chacon, S., & Straub, B. (2014). Pro Git (2nd ed.). Apress. <https://git-scm.com/book/en/v2>

Coronel, C., & Morris, S. (2018). Database systems: Design, implementation, & management (13th ed.). Cengage Learning. <https://www.cengage.com/c/database-systems-design-implementation-management-13e-coronel/9781337627900>

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation, University of California, Irvine). <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>

Fowler, M. (2006). Continuous Integration. <https://martinfowler.com/articles/continuousIntegration.html>

Garrett, J. J. (2011). The elements of user experience: User-centered design for the web and beyond (2nd ed.). New Riders. <https://www.amazon.com/Elements-User-Experience-User-Centered-Design/dp/0321683684>

GitHub Docs. (n.d.). About GitHub. <https://docs.github.com/en>

Mozilla Developer Network. (n.d.). Introduction to web development. <https://developer.mozilla.org/en-US/docs/Learn>

OWASP Foundation. (n.d.). OWASP Top Ten Web Application Security Risks. <https://owasp.org/www-project-top-ten/>

Reenskaug, T. (1979). Models-Views-Controllers. Xerox PARC. <https://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide. <https://scrumguides.org>

Tanenbaum, A. S., & Van Steen, M. (2007). Distributed systems: Principles and paradigms (2nd ed.). Prentice Hall. <https://www.amazon.com/Distributed-Systems-Principles-Andrew-Tanenbaum/dp/0132392273>