Plataforma web y móvil para cálculos de costos de producción.

Web and mobile platform for production cost calculations.

Recibido: 2024/10/20 - Aceptado: 2024/11/15 – Publicado: 2024/11/22

Jorge Andrés Cabrera Malavé
Universidad Tecnológica Israel
e1750065458@uisrael.edu.ec

Edgar Vinicio Ramos de la Cruz
Universidad Tecnológica Israel
e1726776956@uisrael.edu.ec

Renato Mauricio Toasa Guachi
Universidad Tecnológica Israel
e1726776956@uisrael.edu.ec
https://orcid.org/0000-0002-2138-300X

Resumen

La innovación y avance tecnológico es una de las grandes virtudes de la industria 4.0, la cual, en su afán de automatizar los procesos, permite que las tareas se desarrollen con el mínimo esfuerzo posible, en este ámbito, las empresas y organizaciones han optado por el uso de tecnologías accesibles, cómodas y sencillas de utilizar, para ofrecer a sus usuarios clientes, la mejor experiencia posible, es en este contexto, con el cual el desarrollar software funcional para organizaciones que no tengan automatizados los procesos básicos de la misma. Es, por tanto, que el desarrollo de un software capaz de calcular costos de producción para emprendedores rurales dentro de la ciudad de Quito, no solo tiene un impacto local, puesto que, este puede ser utilizado de manera genérica

por quien quiera calcular sus propios costos con los cuales realizan sus productos o servicios, para su beneficio no solo económico, sino sociales.

Palabras clave

industria 4.0, automatización de procesos, innovación tecnológica, software de cálculo de costos, emprendimiento rural.

Abstract

Innovation and technological progress is one of the great virtues of industry 4.0, which, in its eagerness to automate processes, allows tasks to be developed with the least possible effort, in this area, companies and organizations have opted for the use of accessible, comfortable and easy to use technologies, to offer their customers users the best possible experience, it is in this context, with which to develop functional software for organizations that do not have automated the basic processes of the same. It is, therefore, that the development of a software capable of calculating production costs for rural entrepreneurs in the city of Quito, not only has a local impact, since it can be used in a generic way by anyone who wants to calculate their own costs with which they make their products or services, for their benefit not only economically, but socially.

Keywords

industry 4.0, process automation, technological innovation, costing software, rural entrepreneurship.

1. Introducción

El sector rural de Ecuador enfrenta grandes desafíos en la gestión de costos de producción debido a la falta de acceso a herramientas tecnológicas modernas. Con la iniciativa de la organización de referencia, se busca cerrar esta brecha mediante el desarrollo de una aplicación móvil y web diseñada para facilitar el cálculo de costos para emprendedores rurales. Estudios recientes han mostrado cómo estas tecnologías pueden empoderar a los pequeños productores (Cabrera & Ramos, 2024), permitiéndoles mejorar la eficiencia y precisión en la gestión de sus procesos productivos y decisiones financieras.

Esta herramienta tiene como objetivo principal empoderar a los emprendedores al proporcionarles una solución fácil de usar que les permita gestionar de manera eficiente

los costos asociados a la producción agrícola y artesanal. Al mejorar la precisión en la gestión de los costos, los emprendedores pueden optimizar sus procesos productivos, minimizar pérdidas y mejorar su rentabilidad. Además, esta iniciativa promueve la sostenibilidad económica a largo plazo al reducir la dependencia de los métodos tradicionales y mejorar la integración de los productores rurales en mercados más amplios.

Proyectos similares en Ecuador y América Latina destacan el papel crucial de la tecnología en la mejora del sector rural. El proyecto AgroTech Ecuador es un claro ejemplo de cómo las aplicaciones móviles pueden ayudar a los agricultores a gestionar de manera más eficiente sus operaciones, mejorando la productividad en las zonas rurales del país (Mena & Rodríguez, 2019). Esta iniciativa permitió a los productores calcular sus costos de producción y el rendimiento de los cultivos, integrando datos en tiempo real que optimizan sus procesos.

De manera similar, en América Latina, el proyecto AgroEmpresario ha logrado conectar a pequeños agricultores con mercados locales y nacionales, proporcionando una plataforma digital para calcular costos de producción y gestionar sus recursos de manera más eficiente (Gutiérrez & Paredes, 2021). Este proyecto ha tenido un impacto positivo en la reducción de desperdicios y ha mejorado la capacidad de los productores para tomar decisiones basadas en datos precisos.

A nivel mundial, iniciativas como FarmLogs en Estados Unidos han demostrado el poder transformador de la tecnología en la agricultura. FarmLogs permitió a los agricultores realizar un seguimiento detallado de sus costos de producción, facilitando la planificación y el uso eficiente de los recursos agrícolas (Smith, Johnson, Harris, 2018). Esta experiencia muestra que, al igual que en otros lugares, el uso de herramientas tecnológicas para la gestión de costos puede mejorar significativamente la competitividad de los productores rurales.

En conclusión, el desarrollo de la aplicación móvil y web de la organización objetivo se alinea con las tendencias globales y regionales en cuanto a la incorporación de tecnología en el sector rural. Al proporcionar a los emprendedores rurales una herramienta para gestionar mejor sus costos de producción, se espera mejorar su competitividad y contribuir al desarrollo económico sostenible en las comunidades rurales de Ecuador.

2. Metodología

2.1 Requisitos y Análisis

El trabajo fue desarrollado como parte de un trabajo de vinculación, para una organización que apoya a emprendedores del sector rural. El objetivo principal era optimizar el proceso de costos de producción, proporcionando una herramienta web y móvil que permitiera a los usuarios calcular y gestionar los costos de sus productos.

Los requisitos del sistema fueron identificados mediante reuniones mantenidas con el dueño del proceso, realizadas a través de Microsoft Teams. Durante estas reuniones, se definieron las principales funcionalidades que el sistema debía ofrecer, centradas en el cálculo de costos de materias primas, mano de obra, costos indirectos y maquinaria (opcional). También se estableció la necesidad de generar reportes en formato Excel para facilitar el análisis de los datos por parte de los emprendedores.

El sistema debía incluir dos tipos de roles de usuario: administradores, que podrían ver todos los productos registrados, y emprendedores, que solo tendrían acceso a sus propios productos y cálculos. A lo largo del proceso de análisis, se documentaron las siguientes historias de usuario, que capturaron los requisitos clave del sistema:

Tabla 1

Historias de Usuario,

Identificador (ID) de la Historia	Característica/Funcionalidad	Responsable	Tiempo invertido
HU-SCPR-0001	Cálculo total de costos de producción, incluyendo materias primas, mano de obra, costos indirectos y maquinaria.	Edgar Ramos	8 HORAS

HU-SCPR-0002	Autenticación con usuario y contraseña	Andrés Cabrera	8 HORAS
HU-SCPR-0003	Visualización de todos los productos creados (para administradores).	Andrés Cabrera	8 HORAS
HU-SCPR-0004	Registro de productos y sus componentes de costos.	Edgar Ramos	8 HORAS
HU-SCPR-0005	Exportación de costos a un reporte en Excel.	Edgar Ramos	8 HORAS
HU-SCPR-0006	Sincronización entre versiones web y móvil.	Andrés Cabrera	8 HORAS
HU-SCPR-0007	Notificaciones de campos incompletos durante la creación de productos.	Edgar Ramos	4 HORAS
HU-SCPR-0008	Escalabilidad del sistema para manejar más usuarios y datos.	Andrés Cabrera	8 HORAS

2.2 Diseño y Arquitectura

La arquitectura del sistema fue diseñada siguiendo un enfoque basado en microservicios para garantizar flexibilidad y escalabilidad. Se dividieron los módulos en componentes específicos: seguridad, producción, web y móvil, utilizando REST APIs para la comunicación entre los diferentes servicios.

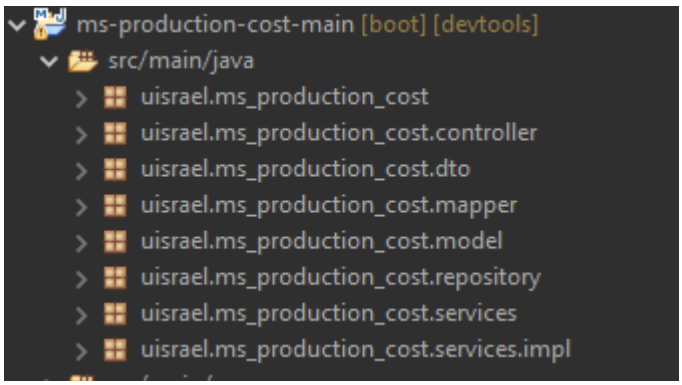
El frontend de la aplicación fue desarrollado con Angular y Angular Material, mientras que el backend se construyó en Spring Boot. Para la aplicación móvil, se integró Microsoft MAUI, permitiendo a los usuarios interactuar con el sistema desde dispositivos móviles y sincronizar los datos con la versión web. La base de datos se implementó en PostgreSQL utilizando Docker Compose, lo que facilitó la replicación del entorno y su despliegue.

2.2.1 Patrón de diseño

El patrón de diseño utilizado en el backend es comúnmente conocido como el Patrón de Arquitectura en Capas o Layered Architecture Pattern. Este patrón separa las responsabilidades en capas bien definidas, lo que mejora la mantenibilidad, escalabilidad y testabilidad del sistema. (Refactoring Guru, n.d.)

Figura 1

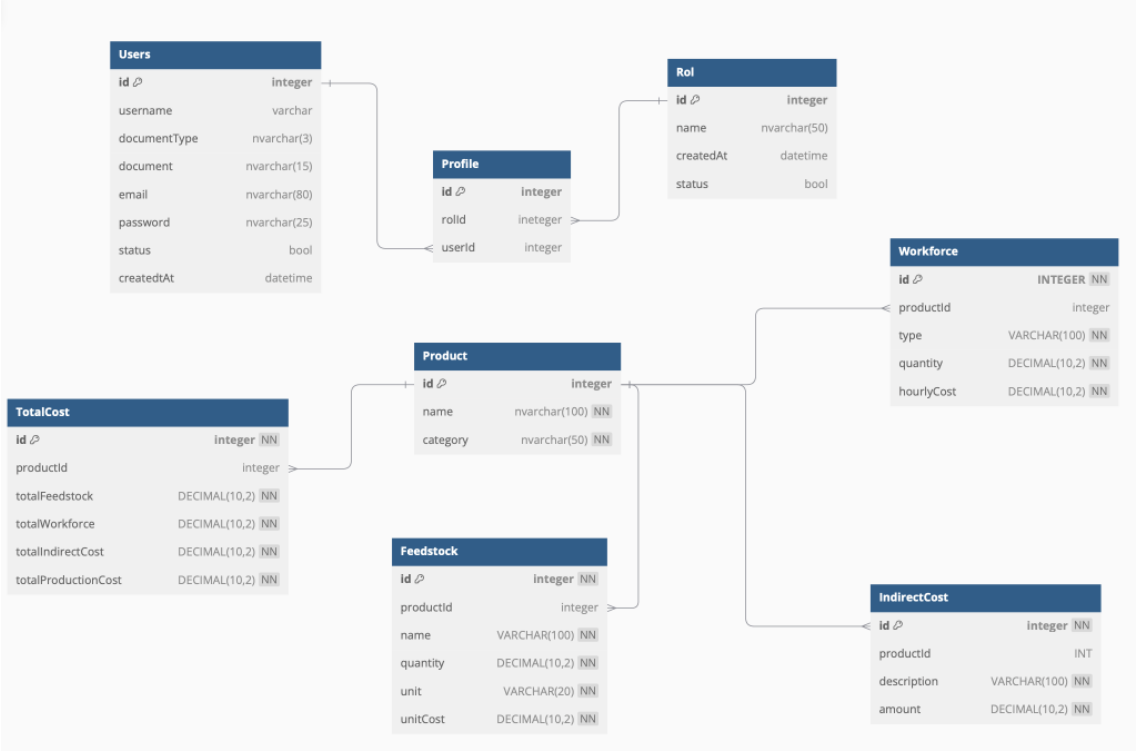
Arquitectura del diseño en N-capas en Spring Boot.



Este enfoque sigue una separación clara de preocupaciones, facilitando la modificación y mejora del sistema sin afectar el resto de las capas.

Figura 2

Modelo entidad relación,



2.3 Desarrollo

El desarrollo del producto se llevó a cabo de manera estructurada y colaborativa, aprovechando las mejores prácticas en control de versiones, metodologías ágiles y herramientas modernas de desarrollo. Este enfoque permitió asegurar la trazabilidad, la colaboración fluida entre los miembros del equipo y la entrega de un producto final.

Figura 3

Arquitectura móvil.

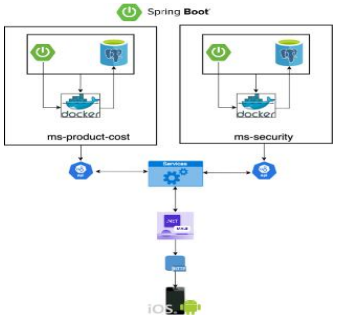
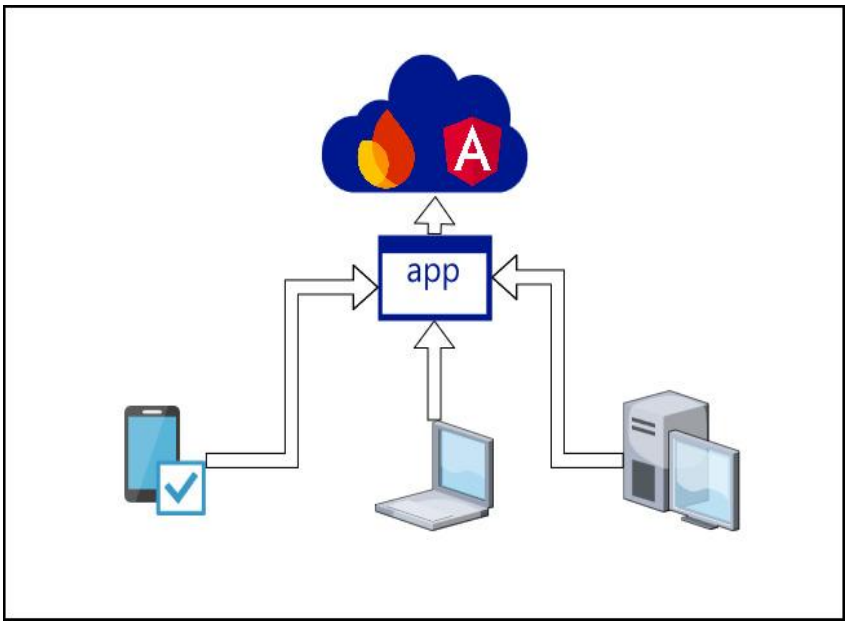


Figura 4

Arquitectura web.

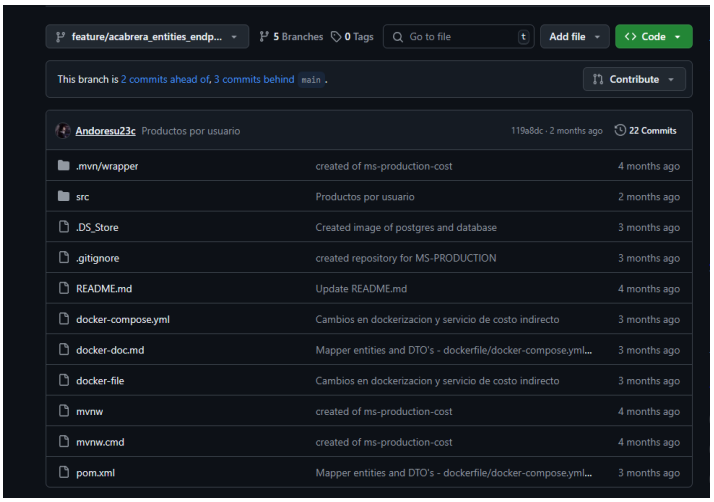


2.3.1 Control de versiones con Github

Para el control de versiones, se utilizó GitHub como plataforma central. Git es una de las herramientas más utilizadas para el control de versiones distribuidas, permitiendo un manejo eficiente de los cambios en el código fuente (Chacon & Straub, 2014). La estrategia de ramificación (branching) permitió que el equipo trabajará en diferentes características de manera simultánea, creando ramas independientes para cada nueva funcionalidad o corrección de errores. Este proceso de ramificación y la fusión de ramas a través de pull requests garantiza la integridad del código, ya que cada contribución fue revisada antes de integrarse en la rama principal, minimizando la posibilidad de conflictos o errores.

Figura 5

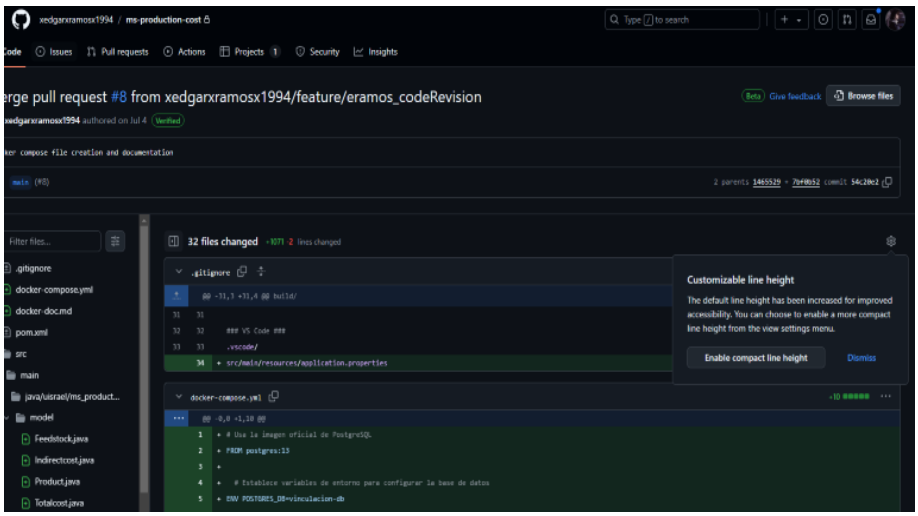
Github con sus ramas de desarrollo independientes.



Cada funcionalidad o corrección de errores se trabajó en ramas individuales, y una vez completada la tarea, se realizaba un pull request para integrar los cambios en la rama principal. Edgar Ramos, como QA del equipo, se encargaba de revisar estos cambios y decidir si aceptarlos o de negarlos, asegurando que el código cumpliera con los estándares de calidad requeridos antes de ser fusionado (Chacon & Straub, 2014).

Figura 6

Merge pull request del proyecto.



2.1.2 Metodología Kanban para el control del desarrollo.

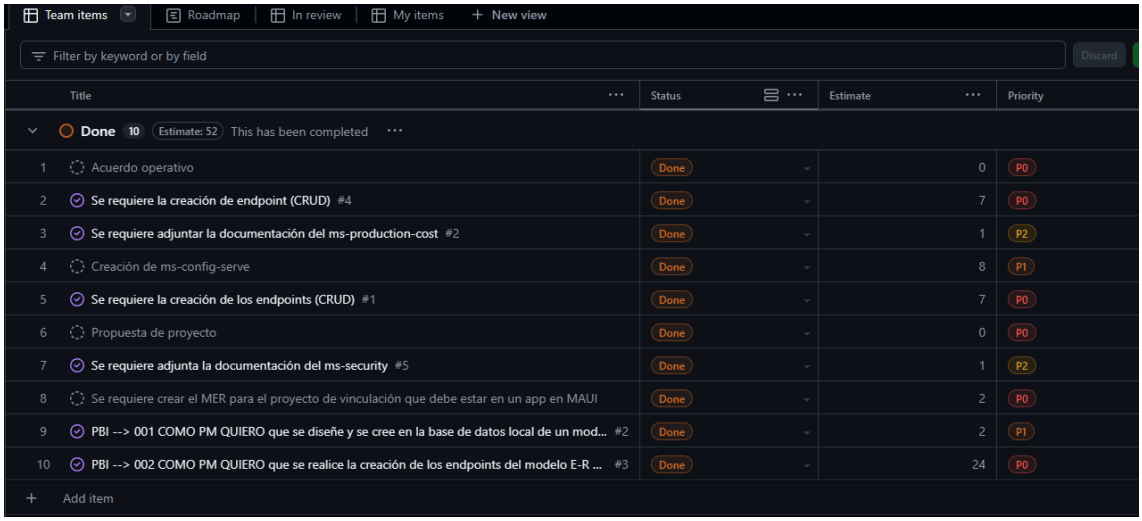
Kanban, una metodología ágil, se centró en la visualización del flujo de trabajo mediante tableros que incluían columnas como "Por hacer", "En progreso" y "Completado". Esta estructura permitió que el equipo mantuviera un enfoque claro sobre qué tareas tenían prioridad en cada momento. Andrés Cabrera y Edgar Ramos colaboraron activamente en el desarrollo, y esta metodología les permitió trabajar en paralelo sin generar conflictos, dado que cada tarea estaba claramente definida y gestionada de manera visual. La flexibilidad de Kanban, al no estar restringido por ciclos de tiempo cerrados como en Scrum, permitió que el equipo se ajustará a cambios inesperados o nuevas demandas sin comprometer el flujo de trabajo continuo (Leopold & Kaltenecker, 2015).

Andrés Cabrera se enfocó en el desarrollo backend utilizando Spring Boot y Docker. Su trabajo se centró en la creación de microservicios escalables que se desplegaron utilizando Docker, lo cual facilitó la gestión de entornos de desarrollo consistentes tanto en desarrollo como en producción. Además, desarrolló la interfaz de usuario móvil utilizando MAUI, que permitió una implementación simultánea para Android e iOS, asegurando así la consistencia de la experiencia del usuario en ambas plataformas (MacDonald & Nathan, 2022).

Por su parte, Edgar Ramos fue el responsable del desarrollo de la plataforma web utilizando Angular y TypeScript. Angular, una plataforma altamente eficiente para la construcción de aplicaciones web de una sola página (SPA), permitió a Edgar crear una interfaz dinámica y modular que facilita la interacción del usuario con la aplicación web (Freeman, 2018). Además, Edgar Ramos asumió el rol de QA, donde se encargó de revisar y aprobar los cambios antes de que fueran integrados en la rama principal del repositorio de GitHub. Este proceso de revisión de pull requests garantizó que solo los cambios que cumplían con los criterios de calidad fueran aceptados, manteniendo así la integridad y la estabilidad del proyecto. Su rol en la validación y aceptación de los merges resultó fundamental para asegurar la calidad continua del código.

Figura 7

Tareas realizadas.



2.1.3 Desarrollo web y móvil

El desarrollo se subdivide en dos partes, la primera es una plataforma web que cumple con los requisitos y funcionalidades requeridas en una sola solución para su despliegue, desarrollada en Angular y TypeScript. La segunda es la solución móvil que igualmente cumple con el requerimiento de calcular los costos de producción y definirlos, desarrollado su backend como un microservicio de Spring Boot y su base de datos en Postgres, ambas alojadas a su vez en un Docker con puerto local el cual se comunica a través de comunicación lazy asíncrona a la interfaz de MAUI.NET.

2.1.3.1 Desarrollo web en angular

Primero, se estructuró el proyecto utilizando Angular CLI para organizar módulos que gestionan distintos aspectos de los costos de producción, como materias primas, mano de obra, y costos indirectos. En cuanto a la interfaz de usuario, se diseñaron componentes con formularios interactivos para que los emprendedores puedan ingresar los detalles específicos de su producción. Estos formularios utilizarían un formulario reactivo de Angular para validar los datos de entrada, asegurando que los usuarios proporcionen la información necesaria antes de avanzar a la siguiente etapa del cálculo.

Todo el proceso se centró en aprovechar las capacidades del framework para crear una aplicación eficiente y dinámica. Se utilizaron componentes para manejar la interfaz de

usuario, formularios reactivos para capturar los datos de producción, y servicios en TypeScript para implementar la lógica de cálculo. La aplicación gestionaría el flujo de datos mediante rutas, con una clara organización modular, permitiendo a los usuarios navegar entre las secciones y realizar los cálculos necesarios, además se incorporó una funcionalidad para poder realizar la exportación de los datos a una plantilla de Excel predeterminada, puesto que, la organización no poseía los recursos necesarios para la implementación de un servidor dedicado con una base de datos.

2.1.3.2 Desarrollo móvil con microservicios

En primer lugar se definen las herramientas utilizadas para el desarrollo de este apartado de la soluciòn fue:

- **SpringToolsSuite4 de SpringBoot:** Se escogió springboot para realizar los microservicios, tanto el módulo de seguridades como el de cálculo de costos de producción, puesto que, posee herramientas flexibles que poseen personalización y mejora en el proceso de desarrollo. Dentro de este mismo se instalaron las dependencias necesarias para poder manejar de manera cómoda tanto la persistencia de datos de JPA, como el mapeo de la estructura de los datos con los DTO.
    - Lombok
    - MapStruct
    - mockito
    - JPA de Spring Boot
- **Docker:** Este proporciona el manejo de imágenes virtuales, por lo tanto, permite la contenerización del proyecto completo en una sola imagen del contenedor, con su propia base de datos y su puerto local para pruebas.
- **Postman:** Permite verificar si los métodos de la API funcionan correctamente según la lógica programada.
- **MAUI.NET:** Framework de Microsoft para el desarrollo móvil de interfaces activas.
- **Postgres16:** Base de datos gratuita que permite el uso de librerías

2.1.3.2.1 Compatibilidad de las tecnologías y su desarrollo.

En el desarrollo del sistema se puso un gran énfasis en seleccionar tecnologías que no solo fueran compatibles entre sí, sino también lo suficientemente sólidas como para garantizar que la solución se mantuviera eficiente y activa en el tiempo. PostgreSQL fue elegido como el gestor de base de datos debido a su capacidad para mantener la integridad de la información, algo clave en cualquier sistema que maneje datos críticos. Para lograrlo, utilizamos Spring JPA, que no solo nos permitió gestionar la persistencia de datos de manera confiable, sino que lo hizo de manera asíncrona, lo que significa que la información se actualiza en segundo plano, sin interrumpir la experiencia del usuario.

Por otro lado, la interfaz fue desarrollada en MAUI.NET, una plataforma que permite crear aplicaciones con una interfaz amigable y una experiencia fluida, asegurando que los usuarios pudieran interactuar sin demoras, independientemente de la complejidad de las operaciones en segundo plano.

Lo más interesante es cómo todo esto se conecta a través de un microservicio dedicado exclusivamente a la gestión de los costos de producción, una parte central del proyecto. Esta arquitectura basada en microservicios no solo permitió dividir las responsabilidades del sistema de manera clara, sino que además hace que el mantenimiento y las futuras ampliaciones sean mucho más manejables. En resumen, este enfoque garantiza que cada pieza del sistema se comunique de manera eficaz y confiable, brindando una solución robusta y preparada para crecer junto con las necesidades del usuario.

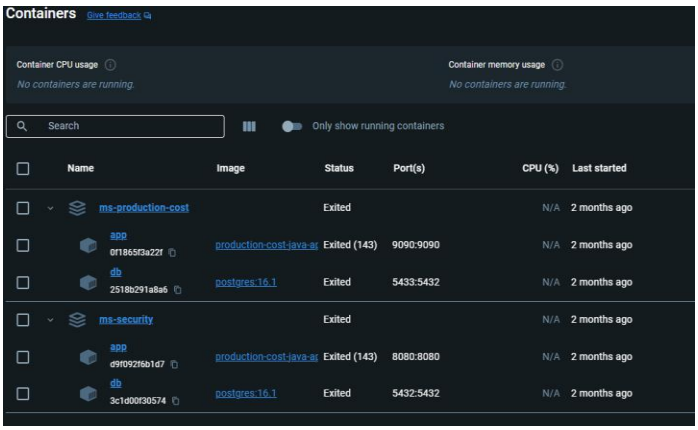
Dockerizacion

El propio proyecto y base de datos se optaron por realizarlos en imágenes virtuales separadas de un mismo contenedor docker, el cual permite la flexibilidad del propio proyecto en cuanto a pruebas y despliegue se refiere, puesto que, el propio microservicio se conectaba directamente a la imagen de la base de datos, evitando totalmente instalaciones innecesarias tanto del proyecto como de la propia base, que causan problemas de compatibilidad descritas anteriormente. Según N.J. (2023), el uso

de Docker en aplicaciones Spring Boot permite una mayor escalabilidad y facilita el despliegue en entornos distribuidos.

Figura 8

Contenedor con ambas imágenes de los módulos.



3. Pruebas y resultados

El proceso de pruebas es un paso crucial para garantizar la calidad del producto, y en este proyecto se realizaron principalmente pruebas unitarias y pruebas de integración, además de pruebas de usabilidad con el dueño del producto. Las pruebas de seguridad no fueron implementadas debido a que se trataba de un prototipo, pero se obtuvieron comentarios valiosos para posibles mejoras en futuras versiones del sistema.

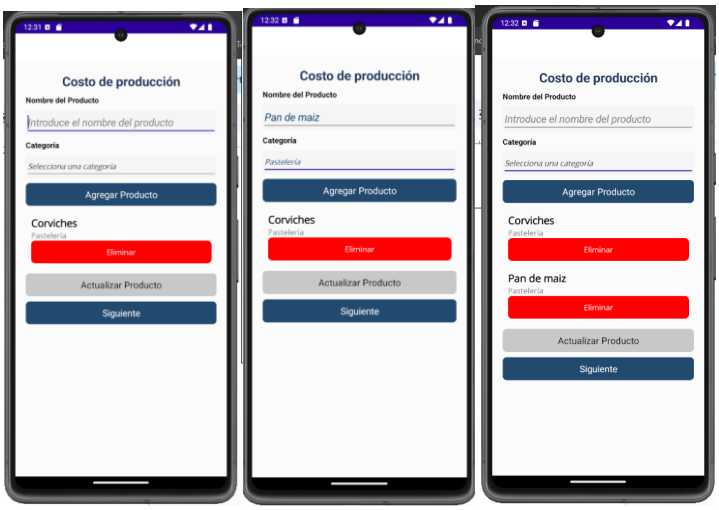
3.1 Pruebas Unitarias

En el backend, desarrollado con Spring Boot, se implementaron pruebas unitarias utilizando JUnit. Estas pruebas se enfocaron en validar el correcto funcionamiento de los componentes clave del sistema, particularmente en los procesos de creación de productos y la adición de ítems al producto final. A través de estas pruebas, se aseguró que cada método y servicio funcionará correctamente en diferentes casos de uso, lo que permitió identificar errores antes de la integración con otros módulos. Aunque no se estableció un umbral de cobertura de código específico, las pruebas fueron exhaustivas,

abarcando todos los casos de uso principales para la gestión de productos (Martin, 2017).

Figura 9

Pruebas unitarias



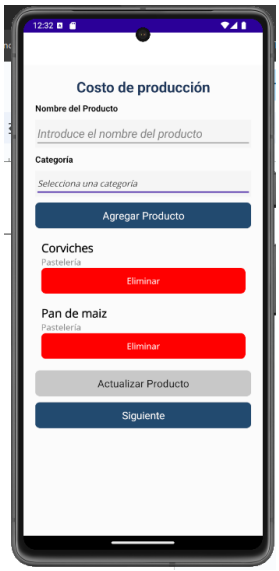
3.2 Pruebas de Integración

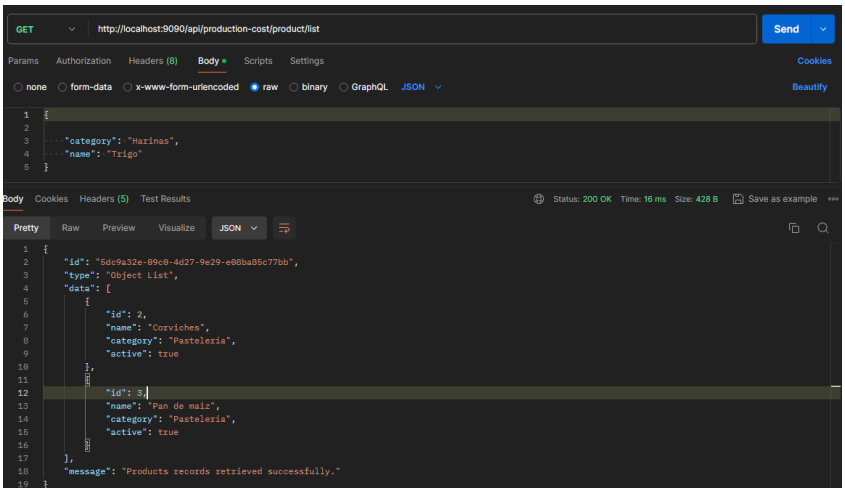
Para las pruebas de integración, se utilizó Postman como herramienta principal para validar la comunicación entre el frontend y los microservicios de Spring Boot, los cuales se encontraban desplegados en un entorno Dockerizado. Estas pruebas aseguraron que los APIs REST del backend recibieran y procesaran correctamente las solicitudes provenientes de la interfaz web, desarrollada en Angular. La contenerización del entorno mediante Docker permitió simular un entorno de producción, donde cada servicio escuchaba en puertos específicos para garantizar la correcta comunicación entre el frontend y el backend (Pressman, 2020).

Las pruebas se realizaron específicamente sobre los endpoints de creación de productos y adición de ítems, verificando la correcta interacción entre las interfaces y los microservicios en tiempo real. Este enfoque permitió detectar y corregir cualquier discrepancia entre los módulos, asegurando una integración fluida en el entorno Dockerizado (Chacon & Straub, 2014).

Figura 10

Prueba de integración





3.3 Pruebas de Usuario

En cuanto a las pruebas de usabilidad, estas fueron realizadas directamente con el dueño del producto, quien probó todo el flujo de la aplicación, desde la creación de productos hasta la generación de reportes de costos. Aunque no se recopilaron formalmente datos ni se realizó un análisis exhaustivo de la experiencia de usuario, se obtuvo feedback cualitativo que permitió identificar posibles mejoras en futuras versiones del sistema. Este tipo de retroalimentación, aunque no sistematizada, fue valiosa para ajustar la interfaz y mejorar la experiencia del usuario final en base a sus necesidades y expectativas (Anderson, 2010).

3.4 Resultados

Se precisaron los resultados en una plataforma web y móvil que cumplen lo establecido en un inicio, para poder realizar el cálculo correcto de los costos de producción. La plataforma web, desarrollada con Angular, maneja su propia lógica para el cálculo de costos de producción, enfocada en ofrecer una experiencia de usuario optimizada para navegadores web. Por otro lado, la aplicación móvil, construida con MAUI (.NET), está diseñada para dispositivos móviles, con una lógica separada para el manejo de la interacción del usuario, adaptada a las características de plataformas móviles como Android e iOS.

Aunque ambas plataformas cumplen con el mismo objetivo (calcular costos de producción), cada una fue desarrollada con sus propias particularidades y no comparten directamente la lógica de negocio. Esto permite mayor flexibilidad en la evolución de cada plataforma y su adaptación a diferentes necesidades de los usuarios, asegurando una experiencia óptima en cada entorno.

3.4.1 Despliegue

Despliegue en Firebase: Firebase ofrece una plataforma robusta para desplegar aplicaciones web, incluyendo la automatización del hosting. El despliegue incluye el uso del CLI de Firebase para conectar el proyecto, configurar archivos como `firebase.json`, y finalmente ejecutar `firebase deploy`, lo cual coloca la aplicación en producción. Firebase facilita la gestión de versiones, dominios, y servicios de autenticación integrados.

Release en MAUI.NET: El proceso de publicación de una aplicación MAUI.NET implica compilar el proyecto para Android e iOS, ajustando configuraciones de `Release`. Se emplean herramientas como Visual Studio para empaquetar la app y publicarla en tiendas como Google Play o App Store, asegurando compatibilidad con los dispositivos móviles. Para este apartado sólo llegó a un release de prueba de la aplicación en entorno local por lo cual, no se publicó ningún APK ni IPA para sus distintas versiones de Android y IOS para su distribución.

Figura 11

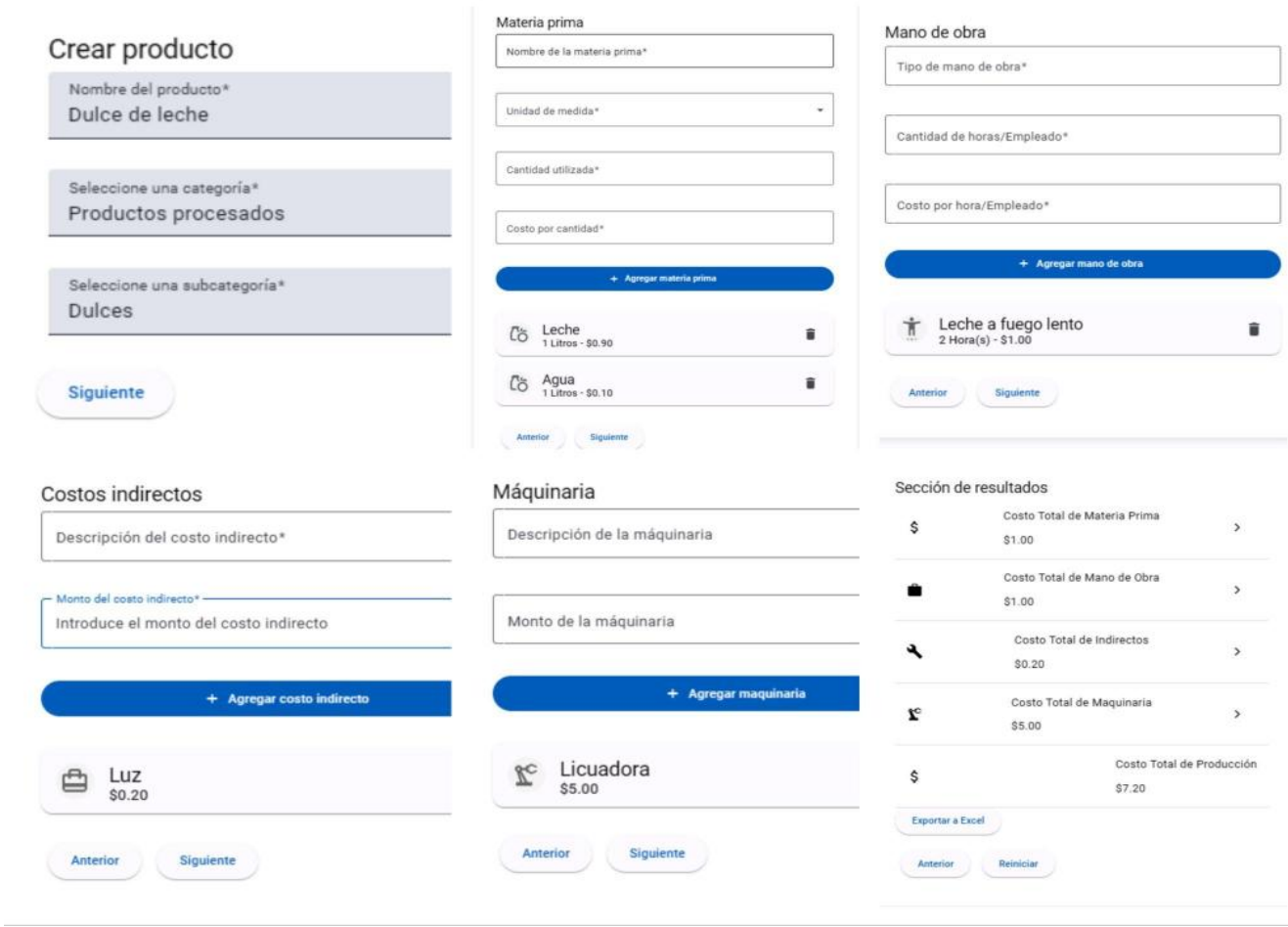
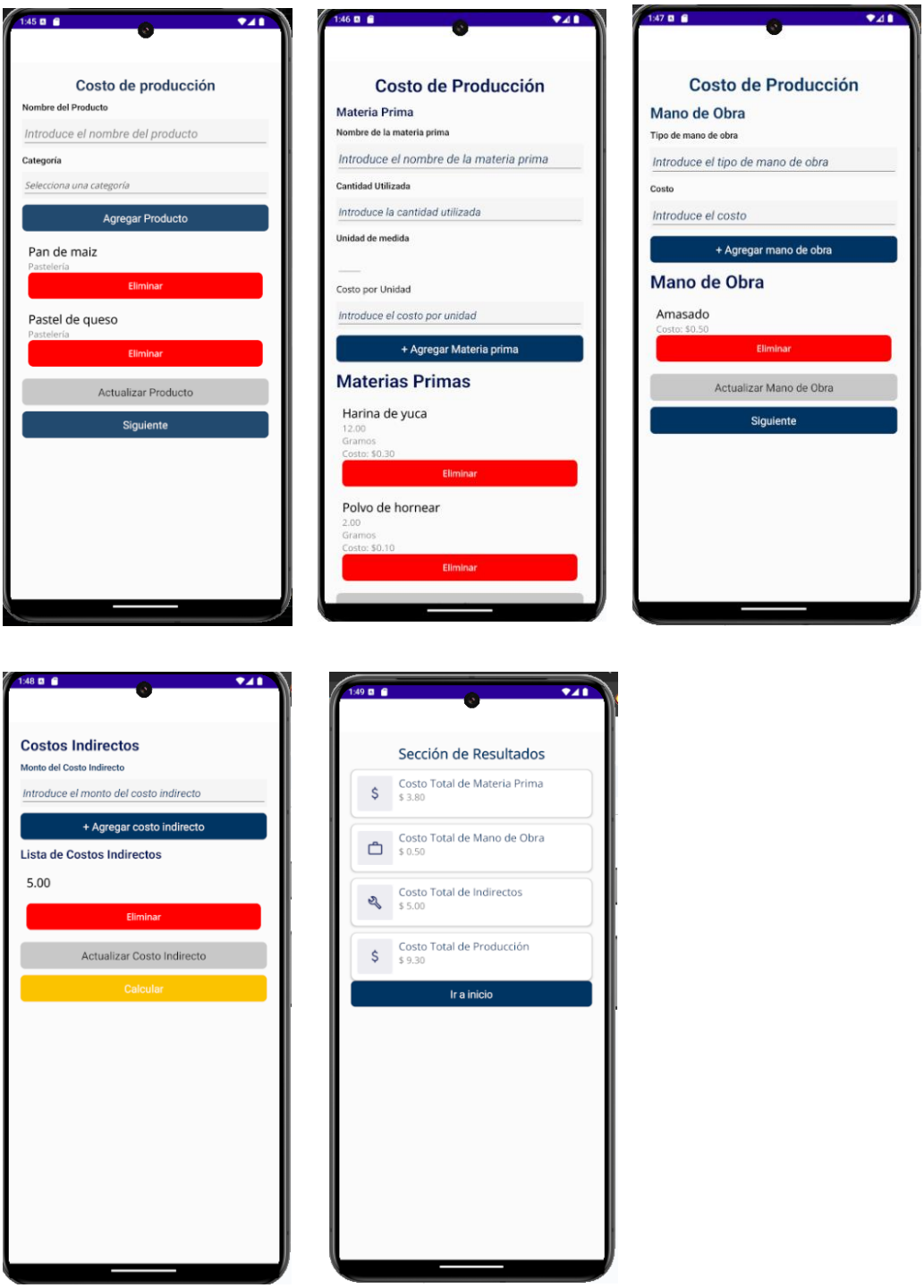

Crear producto
Nombre del producto*
Dulce de leche
Seleccione una categoría*
Productos procesados
Seleccione una subcategoría*
Dulces
Siguiente
Materia prima
Nombre de la materia prima*
Unidad de medida*
Cantidad utilizada*
Costo por cantidad*
+ Agregar materia prima
Leche
1 Litros - $0.90
Agua
1 Litros - $0.10
Anterior
Siguiente
Mano de obra
Tipo de mano de obra*
Cantidad de horas/Empleado*
Costo por hora/Empleado*
+ Agregar mano de obra
Leche a fuego lento
2 Hora(s) - $1.00
Anterior
Siguiente
Costos indirectos
Descripción del costo indirecto*
Monto del costo indirecto*
Introduce el monto del costo indirecto
+ Agregar costo indirecto
Luz
$0.20
Anterior
Siguiente
Máquinaria
Descripción de la máquinaria
Monto de la máquinaria
+ Agregar maquinaria
Licuadora
$5.00
Anterior
Siguiente
Sección de resultados
Costo Total de Materia Prima
$1.00
Costo Total de Mano de Obra
$1.00
Costo Total de Indirectos
$0.20
Costo Total de Maquinaria
$5.00
Costo Total de Producción
$7.20
Exportar a Excel
Anterior
Reiniciar

Figura 12

Flujo de usuario en plataforma móvil



4. Conclusiones

- El desarrollo de la plataforma web y móvil ha resultado ser una herramienta clave para los emprendedores rurales de Quito, permitiéndoles calcular de manera automatizada los costos de producción. Esta automatización ha reducido considerablemente los errores manuales, mejorado la planificación financiera y promovido la sostenibilidad de sus negocios. La facilidad de uso y el acceso a una herramienta tecnológica moderna ha empoderado a los usuarios, ayudándolos a optimizar sus procesos productivos, lo que impacta directamente en su competitividad y rentabilidad.

- La arquitectura del sistema, basada en microservicios, ha demostrado ser una elección acertada, ya que permite que cada módulo funcione de manera independiente y escalable. Al utilizar Spring Boot en el backend y MAUI para la aplicación móvil, se logró una separación clara de responsabilidades, lo que facilita la escalabilidad del sistema a futuro. Esta estructura modular no solo simplifica el mantenimiento y las mejoras, sino que también ofrece flexibilidad para incorporar nuevas funcionalidades según sea necesario, lo que asegura la evolución del sistema a largo plazo.

- El despliegue de la aplicación en Firebase, aunque limitado por la falta de una base de datos permanente, fue efectivo para las necesidades del proyecto. La aplicación pudo gestionar los datos de manera temporal y cumplir con todo el flujo de trabajo sin complicaciones. Esta solución temporal, aunque básica, permitió ofrecer una experiencia completa a los usuarios sin incurrir en costos adicionales, demostrando que a veces es posible lograr soluciones eficientes con recursos limitados.

- Las pruebas de usabilidad realizadas con el dueño del producto proporcionaron retroalimentación importante que ayudó a mejorar la interfaz y ajustar la experiencia del usuario. Aunque no se realizaron estudios formales ni se implementaron pruebas de seguridad, los comentarios cualitativos recogidos

permitieron identificar mejoras para futuras versiones del sistema. Esto demuestra que, incluso en las fases tempranas de desarrollo, la interacción directa con los usuarios puede ofrecer una visión clara de cómo adaptar mejor la solución a sus necesidades reales.

- Finalmente, aunque el proyecto fue concebido como un prototipo, su diseño y desarrollo muestran un gran potencial para futuras mejoras. La integración de herramientas como Docker y Postman evidencia que el sistema está preparado para un despliegue más complejo y robusto en el futuro. Con el tiempo, la inclusión de una base de datos permanente y la implementación de medidas de seguridad más avanzadas permitirán que la plataforma crezca y se adapte a un mayor número de usuarios, ampliando su impacto en otras comunidades.

5. Referencias

Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press.

Cabrera, J., Chamorro, F., Ramos, E., Sandoval, S., & Valencia, M. (2024). Automatizando el cálculo de costos de producción para emprendedores. *Scientific Software Journal, 1*(1), 12-24. https://esparadigms.com/index.php/ssj/article/view/2/8

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress.

Freeman, A. (2018). *Pro Angular 6*. Apress.

Gutiérrez, M., & Paredes, L. (2021). Impacto de las tecnologías digitales en el desarrollo agrícola en América Latina: el caso de AgroEmpresario. *Revista Latinoamericana de Innovación Agropecuaria, 5*(2), 67-81. https://doi.org/10.1234/agroempresario

Leopold, K., & Kaltenecker, S. (2015). *Kanban Change Leadership: Creating a Culture of Continuous Improvement*. Wiley.

MacDonald, M., & Nathan, J. (2022). *Introduction to .NET MAUI: Build Apps with C# and .NET for iOS, Android, macOS, and Windows*. Apress.

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.

Mena, R., & Rodríguez, J. (2019). Implementación de aplicaciones móviles para mejorar la productividad en la agricultura ecuatoriana: el caso de AgroTech Ecuador. *Revista Ecuatoriana de Ciencias Agrícolas, 3*(1), 45-59. https://doi.org/10.1234/agrotech-ecuador

N.J. (2023, July 5). Dockerizing a Spring Boot 3 application. *HackerNoon*. https://hackernoon.com/dockerizing-a-spring-boot-3-application

Refactoring Guru. (n.d.). Patrones creacionales. *Refactoring.Guru*. Retrieved October 22, 2024, from https://refactoring.guru/es/design-patterns/creational-patterns

Smith, D., Johnson, P., & Harris, T. (2018). FarmLogs: Transforming agricultural management through technology. *Journal of Agricultural Technology, 12*(4), 123-137. https://doi.org/10.5678/farmlogs-tech

Turnbull, J. (2014). The Docker Book: Containerization is the new virtualization. James Turnbull.

Walls, C. (2016). Spring Boot in Action. Manning Publications.