Sistema Atlas

Daniel Alejandro Molina Yepes

Informe de práctica para optar al título de Ingeniero de Sistemas

Asesor

Jaime Humberto Fonseca Espinal, Especialista en Ciencias Electrónicas e Informáticas área Telemática, Telecomunicaciones

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín, Antioquia, Colombia

2022

Cita	Daniel Molina [1]
Referencia Estilo IEEE (2020)	[1] D. Molina Yepes, “Sistema Atlas”, **Trabajo de grado profesional**, Ingeniería de Sistemas, Universidad de Antioquia, Medellín, Antioquia, Colombia, ***2022***.

Agradezco al Profesor Jaime Humberto Fonseca por su guía y ayuda durante este trabajo de grado, el asesor externo Jhonatan Jimenez Hincapie por sus enseñanzas técnicas y a la líder de proyecto Lucía Valencia por su buena gestión y enseñanza durante el desarrollo del proyecto Atlas.

Dedicatoria

Este trabajo de grados está dedicado a las personas que confiaron en mi esfuerzo y me ayudaron durante todo el proceso de aprendizaje. Mi amada familia, mis amigos y a cada profesor de quien tuve el privilegio de aprender.

Agradecimientos

Agradezco a cada uno de los miembros de mi familia, a mis amigos y a mis tutores, por todo su apoyo en este camino donde encontré grandes profesionales y aún más importante grandes seres humanos.

Tabla de contenido

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

IEEE	Institute of Electrical and Electronics Engineers
QA	Quality Assurance
SPA	Single page application
API	Application programming interface
IA	Inteligencia Artificial
TDD	Test Driven Development
UI	User Interface
UX	User Experience
IDE	Integrated Development Environment
SCM	Source Control Management
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
OOP	Oriented Object Programming
CI	Continuous Integration
CD	Continuous Delivery
CRUD	Create Read Update Delete
ML	Machine Learning
DB	Database

RESUMEN

El presente trabajo de grado tiene como objetivo describir el proceso de desarrollo del sistema Atlas, un proyecto nacido de la necesidad de unificar un conjunto de procesos y aplicaciones legadas cuya función principal era la gestión de las tiendas pertenecientes a la compañía salvadoreña Unicomer. El producto fue diseñado por la empresa Ingeneo S.A.S, y se trata de un sitio web con la capacidad de gestionar información de los productos distribuidos en los diferentes países donde opera la empresa Unicomer: Costa Rica, Honduras, Salvador y Guatemala; La construcción del producto fue ejecutada por un grupo de 11 personas; para su planificación y control se usaron metodologías agiles del desarrollo de software; en específico SCRUM. El proyecto se desarrolló usando tecnologías de última vanguardia para construcción de aplicaciones web obteniendo como resultado una web funcional, robusta, intuitiva y con una funcionalidad excepcional para el control de tiendas y productos en diferentes países de Centro América.

Palabras clave — **Trabajo de grado, Sistema Atlas, Sistema de gestión de tiendas.**

ABSTRACT

The present paper aims to describe the software development process to build the Atlas system, this project was created to solve a big issue related to the Unicomer management operations, issues like information silos, manual information management, several information systems to control the same data, different databases for saving products and suppliers data located in different cities and countries across Central America, inaccurate information given by the systems they handled; bringing with it problems with customers.

The software was realized by Ingeneo S.A.S using agile methodologies and the most popular frameworks for building web applications; as a final result, we get a system with plenty of valuable features that let our clients be more productive and profitable managing their stores and products across countries like Costa Rica, Honduras, Salvador, and Guatemala. The system works as a web page and our clients could access it through their phones or their computers.

Keywords — **Degree work, Atlas System, Store Management System.**

I. INTRODUCCIÓN

La empresa Unicomer es una empresa líder en el sector de venta minorista en diferentes países de Latino América y El Caribe, opera con más de 25 marcas en 27 países y cuenta con un total de 15,000 empleados 1, debido al tamaño de la empresa, esta cuenta con un departamento de sistemas y un conjunto de aplicaciones muy bien consolidado que le permite mejorar su gestión del día a día , sin embargo la empresa no cuenta con un sistema específico para gestionar de una forma organizada y eficiente las tiendas a través de los diferentes países que opera 2, teniendo en cuenta lo anterior, la empresa Ingeneo S.A.S 3, una empresa dedicada al desarrollo de software a la medida le propone la construcción de un sitio web con las siguientes características:

- Gestión de tiendas, proveedores y productos.
- Soporte para varios idiomas.
- Soporte para hacer diferentes gestiones según el país seleccionado.

La empresa Ingeneo S.A.S para el desarrollo de la aplicación dispuso de un equipo de 11 miembros trabajando durante un año para entregarle a su cliente Unicomer una aplicación que soluciona su necesidad específica.

1 Cuenta con un total de 20 años de experiencia y fue fundada en El Salvador.

2 actualmente Unicomer cuenta con sistemas que manejan información para cada país por lo cual busca unificar todas estas gestiones a través de un único sistema.

3 empresa colombiana con fuerte presencia en la prestación de servicios de software en Colombia y en varios países de centro América.

II. PLANTEAMIENTO DEL PROBLEMA

La empresa salvadoreña Unicomer es cliente de Ingeneo S.A.S, En la actualidad, Unicomer administra la información relacionada con el área de mercancías por medio de distintos sistemas y procesos. De esta manera, se plantea la necesidad de una herramienta a través de la cual se pueda manejar de una manera ágil y automatizada la información.

En conclusión, se requiere un sistema para gestionar su cadena de suministros: proveedores, productos y tiendas. Se requiere que cada una de estas operaciones se pueda distinguir por localización, de esta forma cada proveedor y tienda deben estar asociados a un país.

III. JUSTIFICACIÓN

El manejo de la cadena de suministros realizada por varios sistemas informáticos afecta de forma negativa la operación de la empresa Unicomer a nivel global, puesto que en ocasiones las aplicaciones de las que disponían los operadores daban resultados distintos para determinados clientes, lo cual daba como resultado final una mala experiencia para el cliente y pérdidas económicas para la empresa; Unicomer decide entonces unificar los procesos de distribución en una sola aplicación, a causa de ello se decide construir el sistema Atlas, un sistema que tiene como propósito la administración global de la cadena de suministros; mejorando la experiencia de los clientes de Unicomer y reportando una mejora en sus índices económicas.

IV. OBJETIVOS

A. Objetivo general

Analizar, diseñar e implementar el sistema Atlas mediante el uso de tecnologías de desarrollo de software con el fin de suplir la necesidad del cliente Unicomer para poder gestionar su cadena de tiendas y proveedores en cada país donde opera.

B. Objetivos específicos

- Estudiar las necesidades del cliente mediante el análisis lógico del sistema con el fin de entregarles un software que cumpla sus requisitos respecto a la administración de proveedores y tiendas.

- Construir una aplicación mediante el uso de tecnologías como SpringBoot y Angular, obteniendo con ello un software seguro, robusto y con capacidad para una gran cantidad de datos, garantizando de esta manera una mejor experiencia para los usuarios de la aplicación.

- Ejecutar la implementación del software de un modo disciplinado e iterativo mediante una gestión correcta y el uso de metodologías ágiles.

- Realizar un proceso de QA riguroso mediante el uso de TDD, pruebas unitarias, pruebas integrales y pruebas automatizadas garantizando de esta manera un correcto funcionamiento del aplicativo.

V. MARCO TEÓRICO

Fundamentación conceptual que sustenta el objeto de la práctica.

En la empresa Ingeneo S.A.S para el desarrollo de aplicaciones empresariales el practicante debe conocer los siguientes conceptos:

Metodología Scrum: Su nombre no corresponde a una sigla, sino a un concepto deportivo, propio del rugby, relacionado con la formación requerida para la recuperación rápida del juego ante una infracción menor. Su primera referencia en el contexto de desarrollo data de 1986, cuando Takeuchi y Nonaka utilizan el Rugby Approach para definir un nuevo enfoque en el desarrollo de productos, dirigido a incrementar su flexibilidad y rapidez, a partir de la integración de un equipo interdisciplinario y múltiples fases que se traslapan entre sí [1].

Microservicios: Es un enfoque para el desarrollo de una aplicación única como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y mecanismos ligeros de comunicación, a menudo un recurso de una interfaz de programación de aplicaciones (API) sobre protocolo de transferencia de hipertexto (HTTP). Estos servicios están construidos alrededor de las capacidades del negocio y con independencia de despliegue e implementación totalmente automatizada. Existe un mínimo de gestión centralizada de estos servicios, los que pueden estar escritos en lenguajes de programación diferentes y utilizar diferentes tecnologías de almacenamiento de datos [2].

Lenguaje Java: Java está diseñado para enfrentar los desafíos del desarrollo de aplicaciones en el contexto de entornos heterogéneos distribuidos en toda la red. Apto para el desarrollo de aplicaciones que consumen el mínimo de recursos del sistema, puede ejecutarse en cualquier plataforma, y se puede extender dinámicamente [3].

Sistema Gestor de bases de datos: “Definimos un Sistema Gestor de Bases de Datos o SGBD, también llamado DBMS (Data Base Management System) como una colección de datos relacionados entre sí, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina Base de Datos o BD, (DB Data Base)” [4].

SpringBoot: Las tecnologías web avanzan de una manera exorbitante en la actualidad, donde los frameworks juegan un papel fundamental en ese desarrollo gracias al conjunto de librerías, buenas prácticas y herramientas que facilitan la creación de sitios web a nivel empresarial y entre estas se encuentra SpringBoot, el cual se ha convertido desde su creación en uno de los mejores framework para el desarrollo de aplicaciones web profesionales en el lenguaje de programación Java. Uno de los aspectos más relevantes de SpringBoot es la facilidad que ofrece en la creación de microservicios, los cuales son utilizados por grandes empresas como Netflix, Amazon, Google y otras [5].

Angular: Angular es un framework JavaScript, gratuito y Open Source, creado por Google y destinado a facilitar la creación de aplicaciones web modernas de tipo SPA (Single Page Application) [6].

VI. METODOLOGÍA

Para la construcción del sistema se usan las distintas metodologías de desarrollo. En el caso del proyecto Atlas se usó Scrum como marco de trabajo.

El proceso para la creación del sistema informático Atlas fue el siguiente:

1. El equipo de trabajo tiene una reunión inicial con el cliente, en dicha reunión se discuten las características iniciales para la aplicación (kickoff).
2. El equipo de trabajo se reúne nuevamente con el cliente con el fin de definir a detalle cada uno de los requisitos indicados en el paso anterior (grooming meeting).
3. El equipo de trabajo escribe el conjunto de historias de usuario (Backlog).
4. El equipo de trabajo se reúne con el cliente para definir las historias prioritarias, luego de ello se asigna cada historia a uno o más miembros del equipo de trabajo (sprint planning).
5. Se inicia la ejecución del sprint, este suele tener una duración de tres o cuatro semanas.
6. Se realizan reuniones diarias para evaluar el estado de desarrollo de cada historia de usuario (scrum daily).
7. Se finaliza el sprint y el equipo se reúne para hacer un análisis retrospectivo de la ejecución del sprint (sprint retrospective).
8. Se vuelve a repetir el proceso de forma iterativa entregando en cada cierre de sprint nuevas funcionalidades al cliente.

De esta forma cíclica se van cumpliendo los objetivos planteados.

A continuación, se detalla el cronograma de las actividades realizadas durante todo el semestre de industria, las actividades tienen como resultado final la construcción, pruebas y documentación de los siguientes módulos: Algoritmos, Proveedores y Tiendas.

TABLA I

CRONOGRAMA

Punto	Nombre	Días	Porcentaje	Fecha inicio	Fecha fin
7.11	Módulo 1 - relación entre tiendas e ítems	20	100		
7.11.1	Seguridad - gestión de usuarios - roles	1.44	100	25/06/2021	29/06/2021
7.11.2	Proceso sustituto - Frontend	4.11	100	29/06/2021	07/07/2021
7.11.3	Proceso sustituto - Backend	5	100	07/07/2021	14/07/2021
7.11.4	Proceso similar - Frontend	3.67	100	14/07/2021	22/07/2021
7.11.5	Proceso similar - Backend	2.89	100	22/07/2021	27/07/2021
7.11.6	Proceso nuevo - Frontend	4.22	100	27/07/2021	03/08/2021
7.11.7	Proceso nuevo - Backend	3	100	03/08/2021	06/08/2021
7.11.8	Proceso mantenimiento - Frontend	3.67	100	06/08/2021	12/08/2021
7.11.9	Proceso mantenimiento - Backend	5.44	100	12/08/2021	23/08/2021
vacaciones	Vacaciones	6	100	23/08/2021	31/08/2021
7.11.10	Proceso consultas - Frontend	3	100	01/09/2021	06/09/2021
7.11.11	Proceso consultas - Backend	2.22	100	06/09/2021	09/09/2021
7.14	Módulo 3 - proveedores	10.56	100		
7.14.1	Seguridad - gestión de usuarios y roles	0.89	100	10/09/2021	13/09/2021
7.14.2	Proceso - Frontend	5.89	100	13/09/2021	20/09/2021
7.14.3	Proceso - Backend	3.78	100	20/09/2021	24/09/2021
7.22	Módulo algoritmos				
7.22.1	Seguridad - gestión de usuarios y roles	0.67	100	24/09/2021	24/09/2021
7.22.2	Proceso consulta - Frontend	7.22	100	27/09/2021	04/10/2021
7.22.3	Proceso consulta - Backend	4.56	100	04/10/2021	11/10/2021
7.23	Pruebas				
7.23.1	Pruebas módulo tiendas e ítems	9	100	11/10/2021	22/10/2021
7.23.2	Pruebas módulo proveedores	9	100	25/10/2021	05/11/2021
7.23.3	Pruebas módulo algoritmos	9	100	08/11/2021	19/11/2021

VII RESULTADOS

Resultados Esperados: Para cuando el pasante termine su proceso se espera que el módulo de proveedores, algoritmos y tiendas estén finalizado; a continuación, se describe el proceso necesario para el cumplimiento de dichos resultados.

Ejecución del proceso:

Para el cumplimiento del resultado esperado fue necesaria la ejecución de una serie de procesos por parte del pasante, entre dichos procesos se encuentran:

1. Análisis de requisitos:

   Durante esta etapa el pasante participo en reuniones con el cliente y con el equipo de desarrollo, cada una de estas reuniones estaban orientadas a discutir sobre la mejor forma de solucionar la necesidad del cliente, el resultado final de este proceso eran las historias del usuario con un mayor nivel de detalle, grabaciones, actas y otros documentos que servían como evidencia para reflejar lo acordado acerca del funcionamiento de la aplicación.

2. Diseño de los requisitos funcionales de la aplicación:

   Durante esta etapa el pasante participo en la discusión de ideas relacionadas con la mejor forma de solucionar técnicamente un conjunto de requisitos, una vez se tenia concebida la idea con mejores beneficios se procedía a realizar el diseño de la solución a través de diagramas UML; a continuación, se mencionan de manera resumida los requerimientos funcionales.

   A nivel de implementación serán considerados las siguientes categorías:

   - Seguridad (Autenticación y Autorización)
   - Core

- DD (Demand)
- FF (Fullfilment)

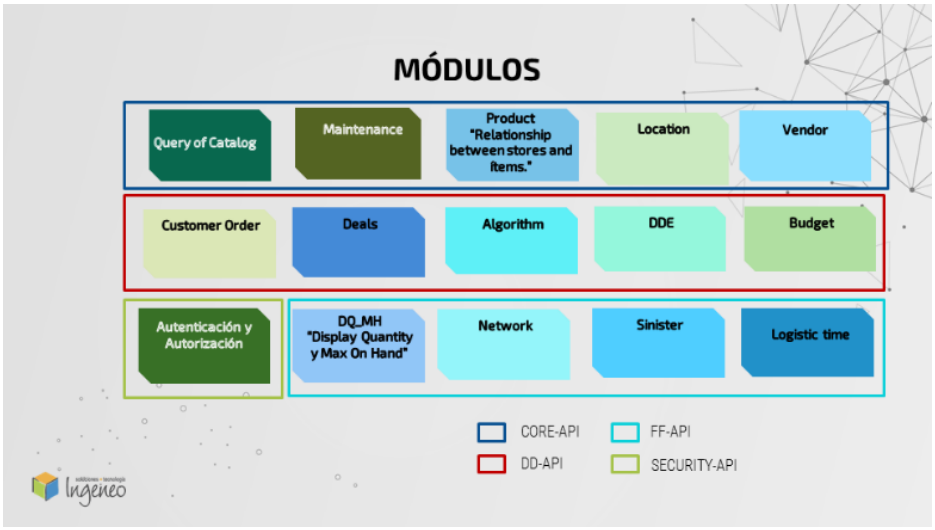


Fig. 1. Módulos Atlas.

CORE: Contiene todas aquellas funcionalidades consideradas transversales para las categorías de FF y DD. En este orden ideas los siguientes módulos integrarán la categoría Core:

- Consulta de catálogos
- Mantenimientos
- Relación entre tiendas e ítems.
- Locaciones
- Proveedor

DEMAND (DD): Comprende todos aquellos módulos relacionados con la gestión de la demanda. En consecuencia, contiene:

- Customer order
- Deals
- Algoritmos
- DDE

- Budget

FULLFILMENT (FF): Fulfillment básicamente es el término utilizado para definir el proceso de recepción, empaquetado y envío de mercancías. En consecuencia, serán incluidos los siguientes módulos:

- Display Quantity y Max on Hand
- Network
- Siniestros
- Tiempos administrativos

Se establece como las funcionalidades núcleo a Países, Locaciones e Items Vs Store dado que los demás módulos se asocian con estas.

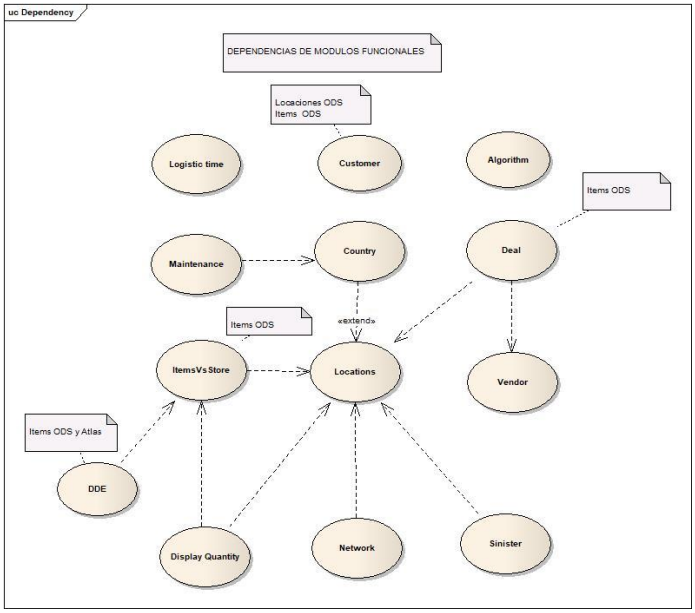


Fig. 2. Dependencia entre Módulos Atlas

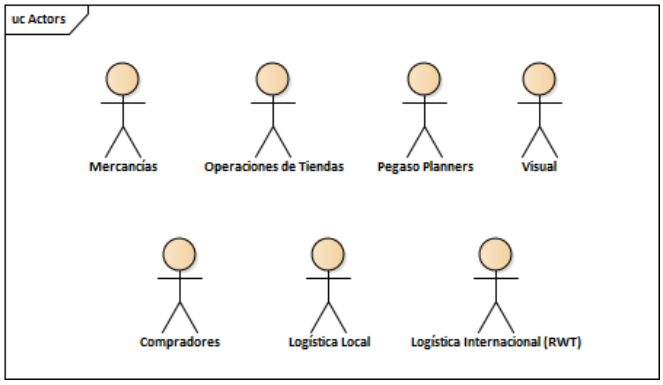


Fig. 3. Actores atlas

En el sistema se han identificado los siguientes actores principales:

- **Mercancías**: Realiza las operaciones de planeación y configuración de acuerdo con su rol asignado.

- **Operaciones de Tiendas**: Encargados del mantenimiento de tiendas y las relaciones de estas con los artículos.

- **Pegaso Planners**: Planeadores asignados a una categoría de producto encargadas de dar mantenimiento de parámetros y establecer relaciones de productos y proveedores.

- **Visual**: Encargados de la apariencia y establecimiento de cantidades para muestra de productos en tienda.

- **Compradores**: Realizan el ingreso de órdenes especiales, negociaciones y mantenimiento de parámetros relacionados.

- **Logística Local**: Darán mantenimiento a los reportes de tiempos administrativos a nivel de país.

- **Logística Internacional (RWT)**: Darán mantenimiento a los reportes de siniestro, tiempos administrativos a nivel de puntos de consolidación.

- **Administrador ATLAS**: Encargado de configuraciones y mantenimiento de la plataforma en general, tendrá acceso a los mantenimientos de asignación de roles y todo lo relacionado a acceso dentro del aplicativo ATLAS.

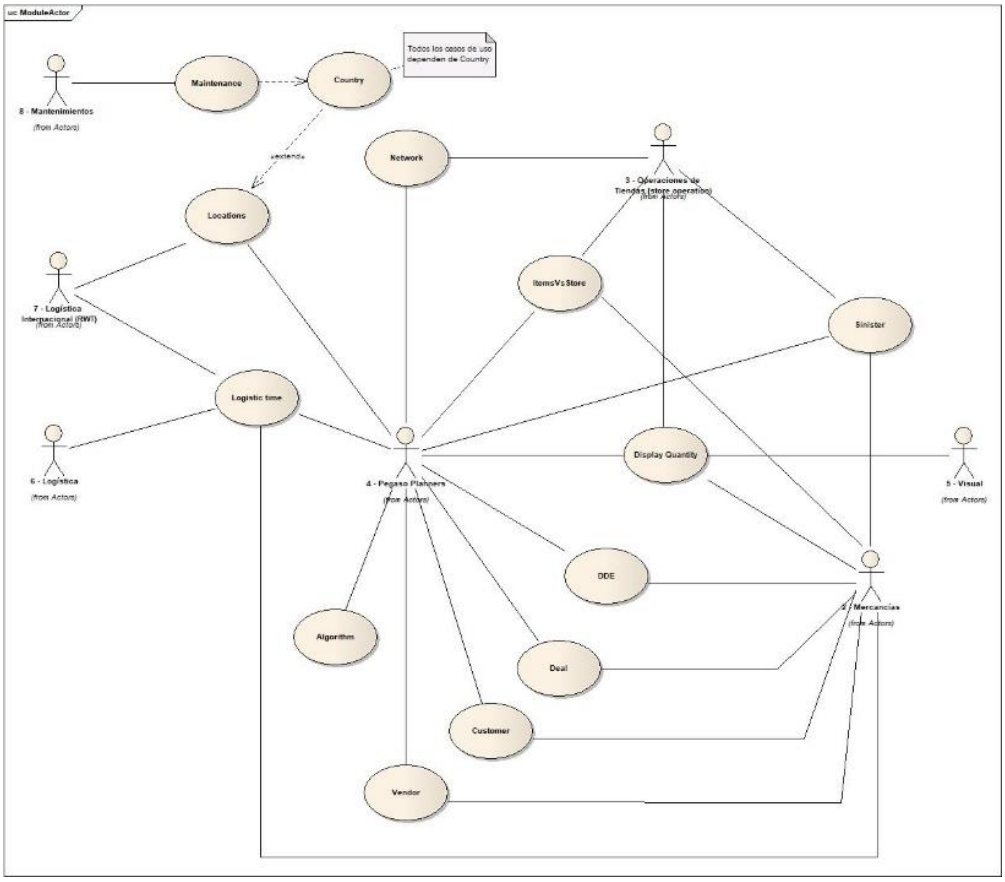


Fig. 4. Actores Atlas interacción con módulos

3. Diseño del módulo de seguridad (autenticación y autorización):

En esta etapa se discutió con el equipo de desarrollo como se abordarían las distintas capas del sistema para garantizar el mayor nivel de seguridad posible a los datos del cliente.

El módulo de seguridad incluye las funcionalidades necesarias para efectuar la autenticación y la autorización. La autorización es el procedimiento mediante el cual se valida la identidad de un usuario. Por su parte la autorización es el procedimiento encargado de comprobar que un usuario tenga los privilegios adecuados para usar un recurso, de no ser exitosa esta comprobación, el acceso al recurso solicitado será rechazado.

En Atlas, la autenticación siempre se llevará a cabo usando el Active Directory, debido a ello no es necesaria la gestión de ningún tipo de contraseña. La autorización por su parte estará basada en el rol y perfiles asociados con el usuario.

A continuación, se presenta el flujo propuesto para realizar el procedimiento de autenticación.

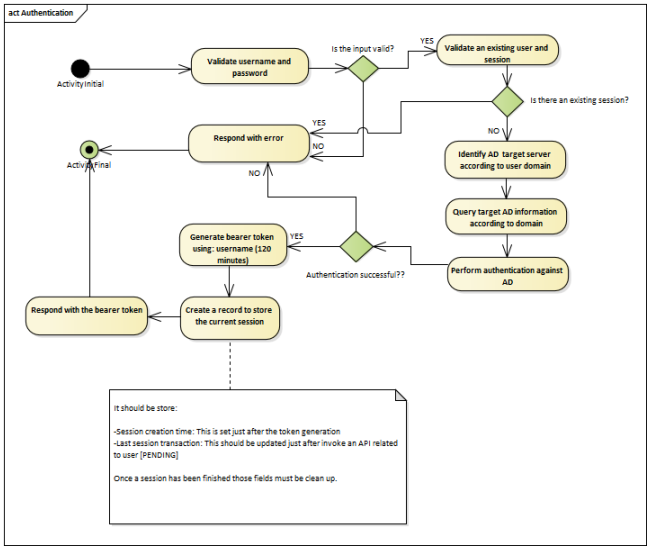


Fig. 5. Diagrama de flujo Autenticación

Si la autenticación es exitosa un JSON Web Token (JWT) será generado almacenando como “Claims” el identificador del usuario y el rol asociado. Adicionalmente, algunas variables relacionadas con la sesión serán almacenadas en base de datos, con el fin de realizar una gestión centralizada de las sesiones. Por otro lado, solo será posible abrir una sesión de manera concurrente para cada usuario, por lo tanto, si el usuario desea iniciar sesión en otro dispositivo deberá cerrar su sesión actual.

Por su parte, la autorización seguirá el siguiente flujo:

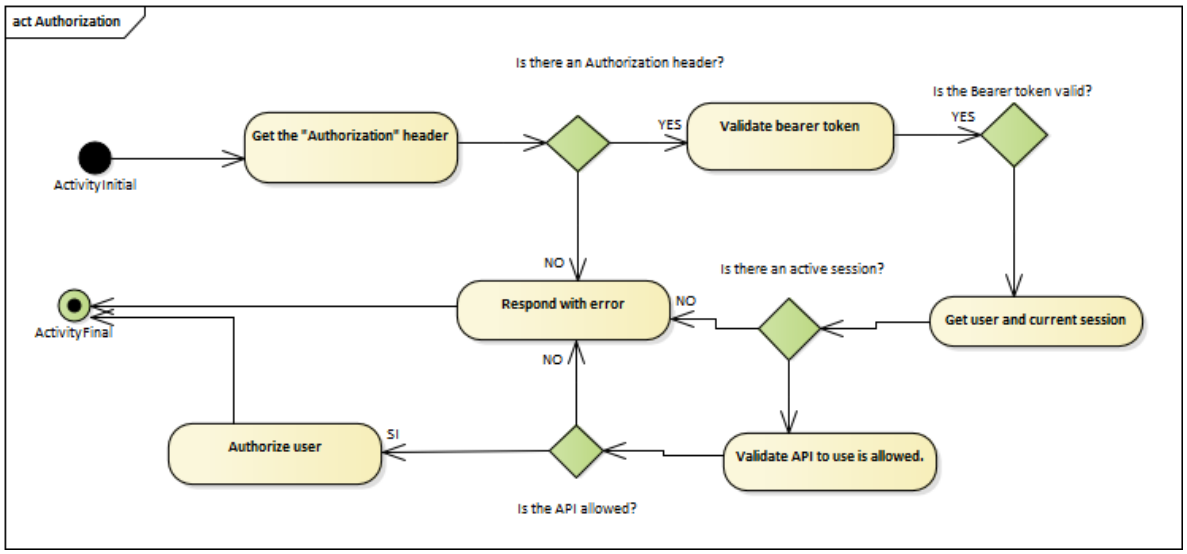


Fig. 6. Diagrama de flujo Autorización

La autenticación siempre se efectuará en el microservicio de seguridad y será transparente para cualquier otro microservicio negocio; un flujo resumido de la autenticación y la autorización se presenta a continuación:

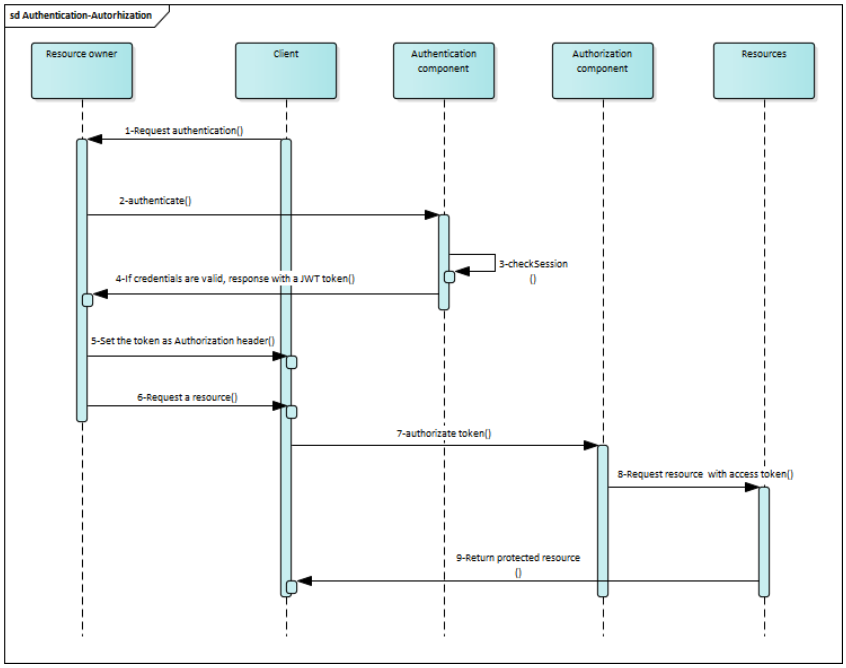


Fig. 7. Diagrama de flujo Autorización

A continuación, se describe el flujo de autorización y autenticación realizado por la aplicación:

1. El cliente solicita al propietario (usuario) autenticación para el acceso a los recursos.
2. El usuario ingresa y envía sus credenciales.
3. El componente de autenticación valida la existencia de una sesión activa para el mismo usuario.
4. Un token en formato JWT es generado siempre y cuando las credenciales sean válidas.
5. Un encabezado “Authorization” es creado para ser usado en las siguientes peticiones a partir del token obtenido.
6. Un nuevo recurso es solicitado incluyendo el encabezado creado en el paso anterior.
7. El token es verificado, si este es correcto se permite la petición al recurso solicitado.
8. Se realiza la petición al recurso solicitado.
9. Se retorna el recurso solicitado al cliente.

4. Diseño técnico de la aplicación:

En esta etapa el equipo de desarrollo en conjunto con los técnicos discutió la arquitectura que tendría la aplicación, los componentes principales y la división en módulos.

Orienta la visión de la solución, esta se plasma y materializa en los componentes que harán parte de la definición estructural, relaciones y dependencias internas. Además, garantiza la alineación con las definiciones corporativas determinantes para el uso de tecnologías de vanguardia.

Para la implementación del proyecto Atlas, se ha seleccionado como estilo de arquitectura la Arquitectura orientada a Microservicios. El estilo de Arquitectura de Microservicios evolucionó de forma natural a partir de dos fuentes principales:

- Aplicaciones monolíticas desarrolladas utilizando el patrón de arquitectura en capas.
- Aplicaciones distribuidas a través del patrón de arquitectura orientada a servicios SOA.

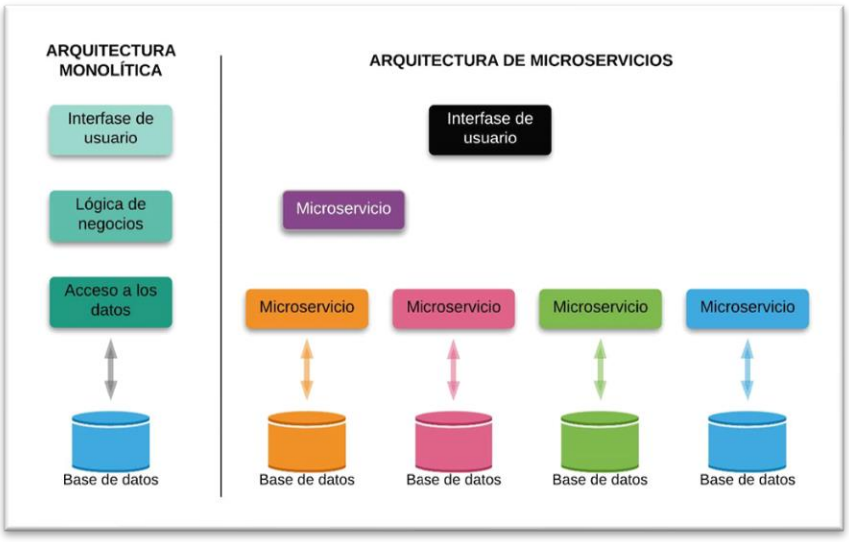


Fig. 8. Comparación Arquitectura monolítica – Arquitectura de Microservicios

La Arquitectura orientada a Microservicios se basa en la noción de unidades desplegadas por separado. De acuerdo con esta noción, cada componente se debe implementar como una unidad independiente, permitiendo de esta manera una implementación más sencilla. Adicionalmente, el modelo de Arquitectura de Microservicios se define como una arquitectura distribuida, ello significa que todos los componentes dentro de la arquitectura están totalmente desacoplados e interactúan mediante un protocolo específico de manera remota. La naturaleza distribuida de este patrón de arquitectura le concede características superiores de escalabilidad e implementación con respecto a otros modelos.

Por otro lado, se emplea la “contenerización”, técnica que posibilita operar bajo el concepto de Multiplataforma, para obtener y aprovechar las mejoras ventajas de cada una de las tecnologías existentes.

VISTA DE IMPLEMENTACIÓN

Esta vista presenta los estados de los objetos principales de negocio, así como la interacción entre los diferentes componentes pertenecientes al sistema.

DIAGRAMA DE COMPONENTES

A continuación, se da una perspectiva general de los diferentes componentes pertenecientes al sistema Atlas. Se explicará Atlas desde una perspectiva externa mostrando las relaciones entre otros sistemas y desde una perspectiva interna describiendo los componentes principales del sistema.

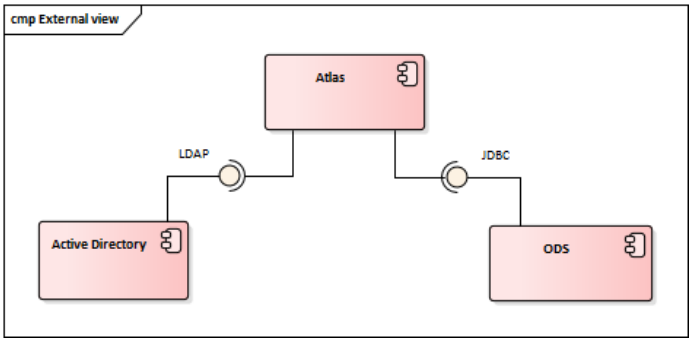


Fig. 9. Vista externa

En esta vista es posible apreciar la interacción de Atlas con cada uno de los sistemas y/o componentes externos.

Atlas: Aplicación construida bajo arquitectura orientada a microservicios, dotada de un conjunto de “capacidades” para ser consumidas tanto a nivel interno como a nivel

externo por parte de otras plataformas. Para su normal operación, Atlas necesita de integraciones con otras plataformas.

Active Directory: Implementación de servicio de directorio de Microsoft. Es en este repositorio donde se almacenan las identidades de los usuarios. Atlas debe comunicarse con esta plataforma usando el protocolo LDAP con el fin de autorizar a los usuarios que deseen usar alguna de sus funcionalidades.

ODS: Operational Data Store o también conocido como Almacén Operacional de Datos. Se trata del contenedor de datos activos de Unicomer cuyo propósito general es integrar los datos de origen dispares en una sola estructura. Atlas debe interactuar con esta plataforma para obtener información acerca de entidades base de negocio y catálogos. Esta comunicación se realiza a través del protocolo JDBC.

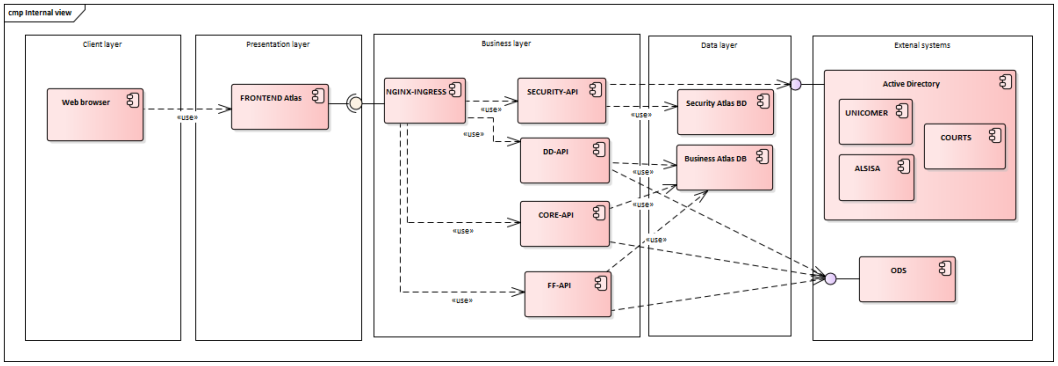


Fig. 10. Vista interna

CAPA DE PRESENTACIÓN

FRONTEND-Atlas: Aplicación Angular que contendrá las páginas web con las cuales interactuarán los usuarios. Esta aplicación se comunicará con la capa de negocio a través de un catálogo de microservicios.

CAPA DE NEGOCIO

NGINX-INGRESS: Ingress controller que servirá de intermediario entre las solicitudes del cliente y los microservicios de negocio.

SECURITY-API: Microservicio donde se implementarán los flujos de autenticación y autorización.

DD-API: Microservicio donde se implementarán todas aquellas funcionalidades relacionadas con Demand.

FF-API: Microservicio donde se implementarán todas las funcionalidades relacionadas con la categoría de Fullfilment.

CORE-API: Microservicio para todas las funcionalidades transversales que podrán ser aprovechadas por otros microservicios.

CAPA DE DATOS

Security Atlas BD: Esquema de base de datos Oracle para gestionar las entidades de seguridad (Configuración de usuarios, perfiles, permisos, APIs).

Business Atlas BD: Esquema de base de datos Oracle para gestionar las entidades de negocio.

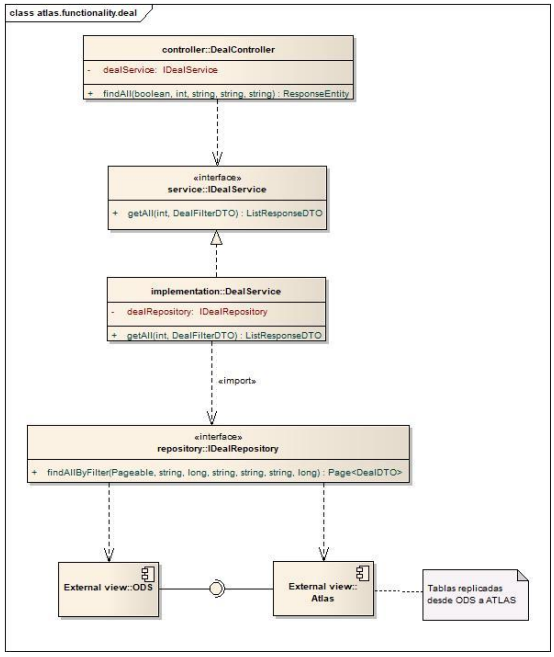


Fig. 11. Diagrama de clases

Toda la construcción de lógica de negocio en las Apis tendrá el siguiente estándar de codificación respetando el paquete y estructura del punto anterior, por lo tanto, se tiene:

- RestController
- Interface de Servicio
- Interface de Repositorio

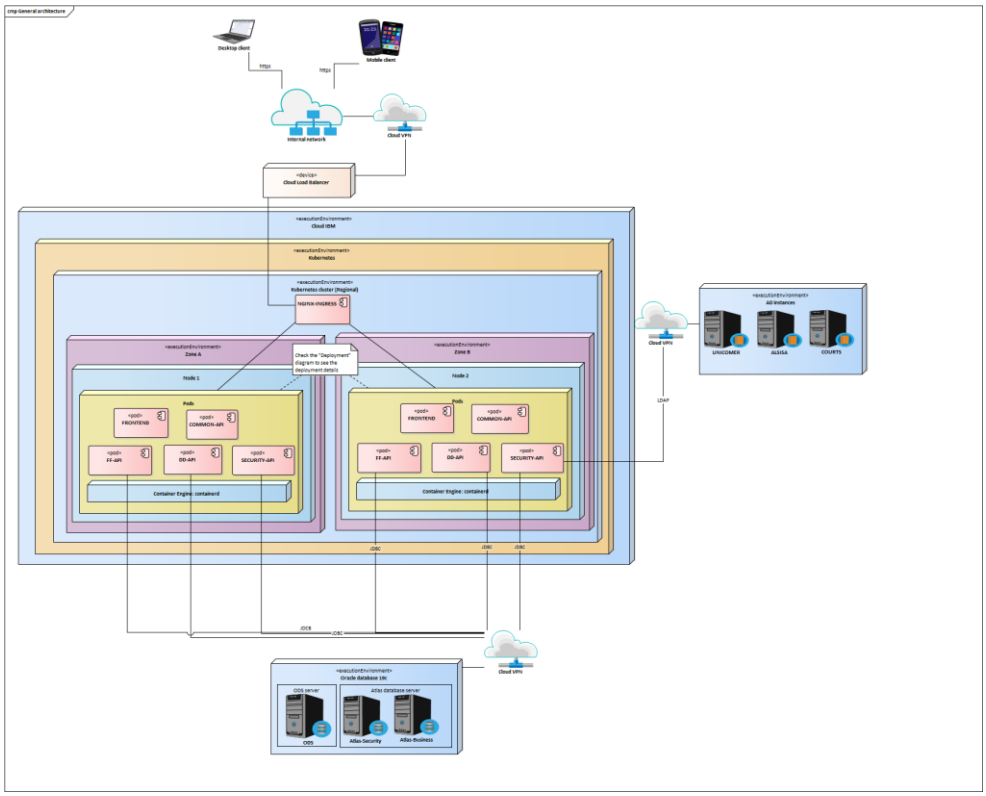


Fig. 12. Diagrama de despliegue

En esta vista se muestra la disposición física de los artefactos de software en los nodos asociados. Para el caso particular de Atlas es posible destacar:

Cloud IBM: Corresponde al proveedor de la nube seleccionado por Unicomer para el despliegue de la solución. Es aquí donde se realizará la configuración de Kubernetes

Kubernetes: Es una plataforma open-source para el despliegue automático, escalabilidad y administración de aplicaciones contenerizadas.

Cloud load balancer: Se encarga de hacer una distribución de carga sobre los nodos. El servidor soporta la detección de fallo de un nodo y hace el redireccionamiento del tráfico a otro nodo disponible.

NGINX-Ingress: Objeto de Kubernetes, este se encarga de proporcionar reglas para gestionar el acceso a servicios desplegados en un Kubernetes clúster a través de los protocolos HTTP/HTTPS.

Cloud VPN: Servicio de VPN mediante el cual se establece conexión con servicios localizados en otras nubes o en ambientes On-premise. Este debe ser un servicio de alta disponibilidad para evitar latencias en las comunicaciones.

Zone X: Corresponde a cada una de las zonas de alta disponibilidad del proveedor de nube. Las zonas de disponibilidad son ubicaciones físicas únicas con recursos de alimentación, red y refrigeración independientes.

Node X: Corresponde a cada uno de los nodos pertenecientes al clúster de Kubernetes para el despliegue de la solución. Son sugeridos como mínimo dos nodos, cada uno de ellos debe encontrarse en una zona diferente para ofrecer una mejor disponibilidad del servicio.

Pods: Los "Pods" son las unidades de despliegue más pequeñas administradas por Kubernetes y tecnologías relacionadas. Básicamente, un "Pod" es un grupo de uno o más contenedores que comparten: almacenamiento, recursos y una especificación de cómo deben ejecutarse tales contenedores. Cada Pod tendrá un solo contenedor, esto facilitará la escalabilidad de la solución.

Active Directory: Implementación de servicio de directorio de Microsoft. Es en este repositorio se almacenan las identidades de los usuarios. 3 instancias serán configuradas inicialmente, cada una de las cuales asociadas con los siguientes dominios:

- **UNICOMER**
- **ALSISA**
- **COURTS**

Atlas-Security: Esquema de Oracle (Oracle 19c) para almacenar las tablas relacionadas con el módulo de Autenticación y Autorización de Atlas.

Atlas-Business: Esquema de Oracle (Oracle 19c) para almacenar las tablas relacionadas con los módulos de negocio de Atlas.

ODS: Operational Data Store o también conocido como Almacén Operacional de Datos. Se trata del contenedor de datos activos de Unicomer cuyo propósito general es integrar los datos de origen dispares en una sola estructura. Este repositorio de datos se encuentra alojado en Oracle (Oracle 19c).

Containerd: Container Engine usado por parte de Kubernetes para administrar los contenedores.

Contenedores: Conjunto de aplicaciones pertenecientes a la solución, los cuales se listan a continuación:

- **FRONTEND**
- **COMMON-API**
- **FF-API**

- **DD-API**
- **SECURITY-API**

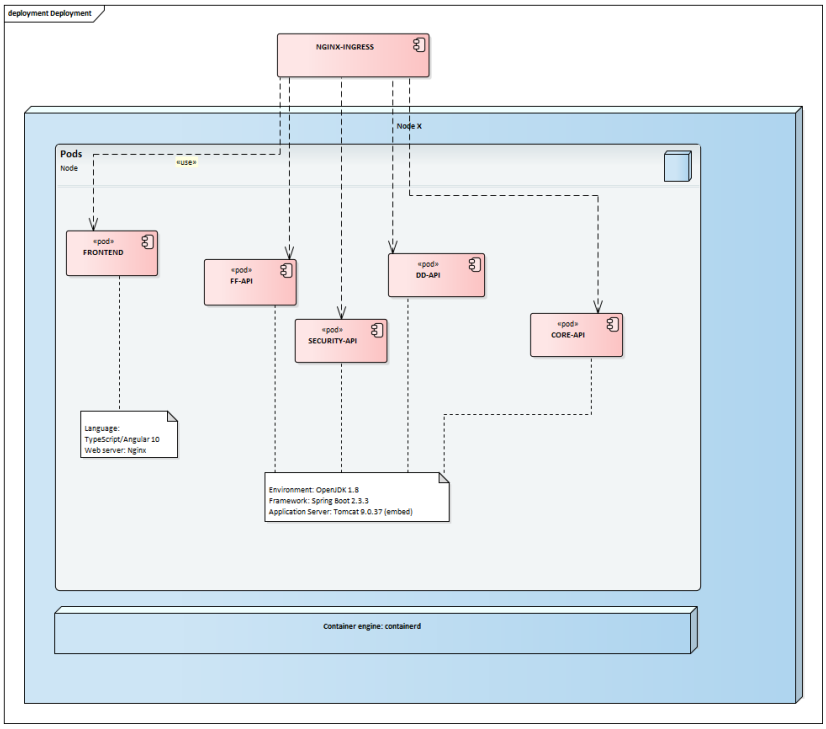


Fig. 13. Despliegue de contenedores

5. Diseño de los requisitos no funcionales de la aplicación:

Durante esta etapa el equipo de desarrollo en conjunto con los lideres técnicos y los arquitectos discutieron los requisitos transversales a toda la aplicación, aquellos requisitos que asegurarán el mejor funcionamiento de la aplicación a lo largo del tiempo; aquellos que definirán la calidad del sistema.

Un requisito no funcional o atributo de calidad, corresponde a un requisito que especifica criterios usados para juzgar la operación de un sistema en lugar de sus comportamientos específicos; por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar.

Interoperabilidad: Aunque la solución se desea desplegar en la nube, debe poderse desplegar independientemente del proveedor de nube, inclusive en un datacenter on Premise. Por esta razón, la aplicación está diseñada para ser desplegada en contenedores orquestados por un servicio Kubernetes, facilitando así su interoperabilidad.

Tiempos de respuesta: Los tiempos de respuesta no pueden exceder los 5 segundos. Dado que se trata de una aplicación basada en microservicios, las métricas serán extraídas tomando como base los tiempos de respuesta de los microservicios.

Comprobabilidad:

- Se contará con pruebas unitarias automatizadas en un flujo de integración continua, para facilitar la ejecución de pruebas regresivas ante la liberación de nuevas versiones.
- Se contará con pruebas funcionales automatizadas, ejecutadas como pasos de validación después de desplegar una actualización.

Concurrencia: La aplicación estará diseñada para soportar hasta 100 usuarios concurrentes. Pruebas de carga serán efectuadas para comprobar esta cantidad de usuarios esperada.

Integración continua: La liberación de los containers se realizará mediante un flujo de despliegue continuo. Para esta finalidad se implementará un Pipeline usando GitLab.

Gobernabilidad: Todo el código fuente será propiedad de Unicomer y será entregado en el GitLab de Unicomer.

Usabilidad:

- Dado que el sistema tendrá usuarios concurrentes de diferentes regiones, se mostrará un mensaje a los usuarios una vez intenten realizar modificaciones a registros que otro ha comenzado a editar. Esto no bloqueará la capacidad de los usuarios para consultar los registros en cuestión.

- El sistema va a ser utilizado en múltiples países, por tanto, será multilenguaje, en su primera versión la aplicación soportará inglés y español. La internacionalización aplicará sin restricciones para todas las etiquetas de la aplicación, así como para los mensajes de retroalimentación al usuario, no obstante, para la información obtenida desde base de datos, tal como los catálogos de ODS, la internacionalización dependerá de los idiomas en los que se haya configurado dicha información.

Escalabilidad: La aplicación podrá escalar fácilmente tanto de forma horizontal como vertical. Una de las ventajas de hacer uso de Kubernetes, es la facilidad para definir políticas de auto escalabilidad basadas en el uso de recursos.

Seguridad: No habrá categorías de usuario que puedan autenticarse a la aplicación sin pertenecer a alguno de los dominios de AD integrados con la aplicación.

6. Calidad de código

Durante esta etapa los desarrolladores realizaron programación en parejas y también usaron herramientas como Sonarqube para evaluar la calidad del código.

Un código fuente con calidad es aquel que, además de hacer lo que tiene que hacer en términos funcionales, es fácil de comprender y modificar por otros programadores.

Un código fuente con calidad permite ser reutilizado, extendido y mantenido de forma fácil, y no únicamente al equipo que implemente las funcionalidades, sino también a cualquier persona que utilice el código fuente.

El código fuente debe ser simple, no debe necesitar explicación o documentación más allá de la funcional y debe estar centralizado alrededor de los 7 ejes de calidad de código.

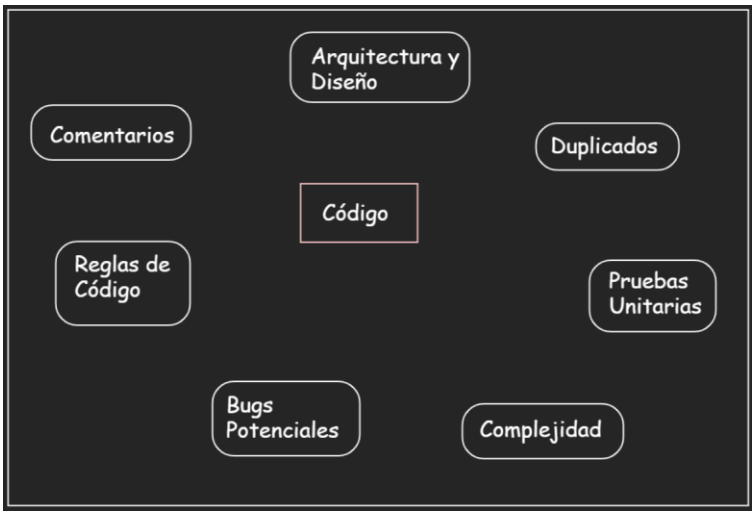


Fig. 14. Ejes de calidad de código

7. Pruebas

Durante esta etapa se realizaron pruebas unitarias y pruebas automatizadas con el fin de detectar tempranamente cualquier error dentro de la funcionalidad del aplicativo.

Pruebas automatizadas: Las implementaciones están dotadas de un conjunto de pruebas automatizadas, permitiendo identificar errores mediante una herramienta llamada Serenity.

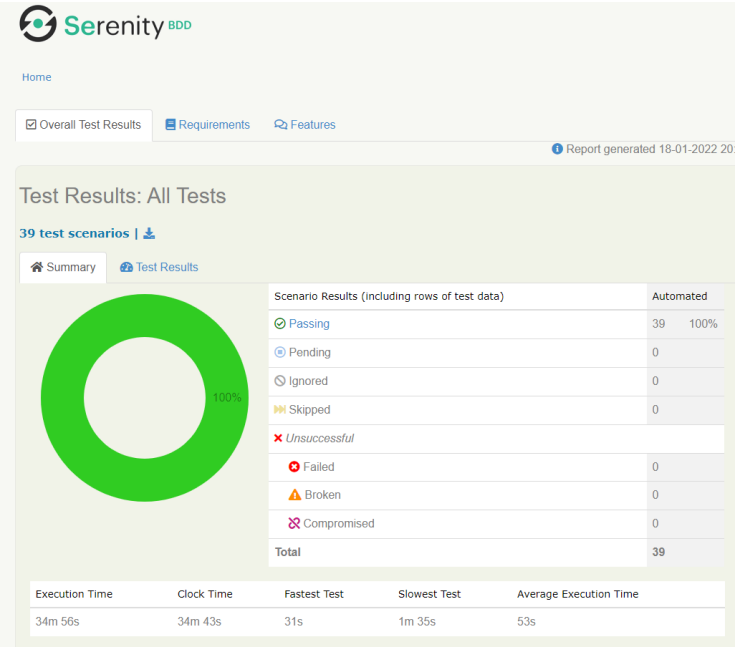
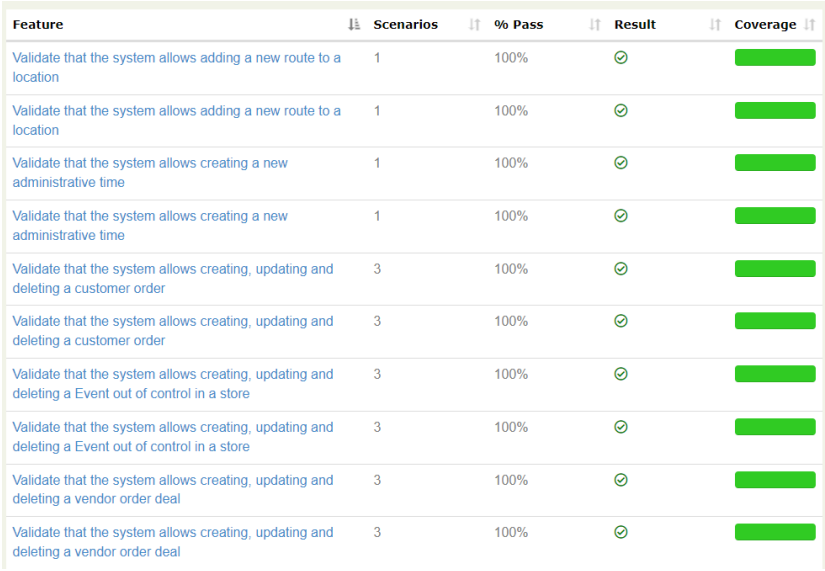


Fig. 15. Calidad de código. Reporte de pruebas automatizadas



Feature	Scenarios	% Pass	Result	Coverage
Validate that the system allows adding a new route to a location	1	100%		
Validate that the system allows adding a new route to a location	1	100%		
Validate that the system allows creating a new administrative time	1	100%		
Validate that the system allows creating a new administrative time	1	100%		
Validate that the system allows creating, updating and deleting a customer order	3	100%		
Validate that the system allows creating, updating and deleting a customer order	3	100%		
Validate that the system allows creating, updating and deleting a Event out of control in a store	3	100%		
Validate that the system allows creating, updating and deleting a Event out of control in a store	3	100%		
Validate that the system allows creating, updating and deleting a vendor order deal	3	100%		
Validate that the system allows creating, updating and deleting a vendor order deal	3	100%		

Fig. 16. Calidad de código. Features

Pruebas unitarias: Las implementaciones estarán dotadas de un conjunto de pruebas unitarias, permitiendo identificar errores en etapas tempranas –antes de pruebas funcionales y despliegues productivos, además, se establecerá una política de cobertura de código.

Las pruebas unitarias deberán estar orientadas a probar los comportamientos probables discretos, cuya característica radica en ser probados como unidades independientes.

Cuando se realicen modificaciones al código probado, es necesario realizar la comprobación del nuevo código implementado y/o modificado, para verificar escenarios que impiden que las pruebas unitarias se ejecuten con los resultados correctos.

La inyección de dependencias y los marcos de datos de prueba –Mocks- harán parte de esta práctica, a fin de garantizar cobertura y la facilidad de prueba del código. Este enfoque permite probar cada una de las implementaciones concretas de forma aislada, y sólo comprobar que invoca la abstracción –Mocks- correctamente.

Cobertura de código: Las coberturas de código permiten determinar la cantidad de código abarcada con las pruebas unitarias, la métrica ofrece un nivel de certidumbre sobre el impacto obtenido al no incorporar la práctica de las pruebas unitarias.

Duplicidad de código: El análisis de código duplicado o CPD (Cut and Paste Detection) permite encontrar bloques duplicados en el código fuente de los proyectos.

La aparición de este tipo de situaciones en los proyectos es un indicativo de malas prácticas. Si el código se debe repetir, debería estar encapsulado en un método de una clase. Esto permite reducir el tamaño del código y facilita el mantenimiento ya que cualquier cambio posterior sólo se tiene que hacer en un sitio localizado.

Code smell: Un code smell consiste en un síntoma en el código fuente que puede indicar un problema profundo.

Sonar Qube: A continuación, se ilustra la ejecución del código usando SonarQube, se realiza análisis con las reglas de sonar establecidas en Ingeneo con el estándar de proyectos Java.

Configuración: A continuación, se visualiza la configuración establecida en la plataforma SonarQube para el aplicativo Atlas.

Sonar way BUILT-IN

Conditions

Conditions on New Code

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

Fig. 17. Configuración establecida para Atlas

Apis analizadas:

- Common (ing4482-atlas-common): Librería de utilitarios y código en común entre APIs
- Core- API (ing4482-atlas-core)
- DD-API (ing4482-atlas-dd)
- FF-API (ing4482-atlas-ff)
- Security- API (ing4482-atlas-security)

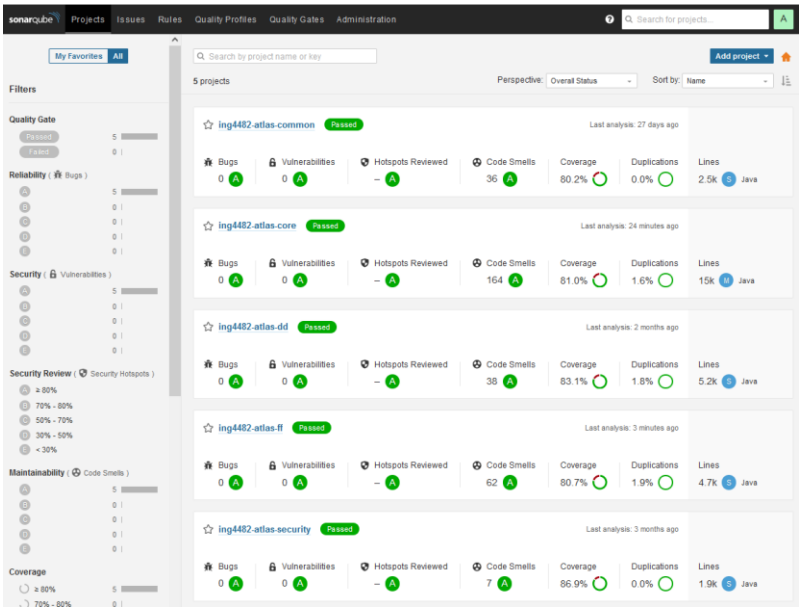


Fig. 18. Ejecución de SonarQube Atlas

8. Integración continua:

Durante esta etapa se realizan procesos de integración continua con el fin de desplegar la aplicación en los ambientes de aseguramiento de calidad y de producción.

La integración continua (CI) ayuda a que los desarrolladores fusionen los cambios que introducen en el código para incorporarlos a una división compartida (o "rama") con más frecuencia. Una vez que se fusionan los cambios implementados por un desarrollador en una aplicación, se validan con el desarrollo automático de la aplicación y la ejecución de distintos niveles de pruebas automatizadas (generalmente, pruebas de unidad e integración) para verificar que los cambios no hayan dañado la aplicación. Esto significa probar todo, desde las clases y el funcionamiento hasta los distintos módulos que conforman toda la aplicación.

Con el fin de aplicar las bondades de la integración continua en el proyecto Atlas, será creado un Pipeline para cada microservicio propuesto, el cual se encargará de:

- Realizar un escaneo de calidad de código, de acuerdo con los apartados expuestos en la sección “Calidad de código”.
- Realizar una ejecución de todas las pruebas unitarias del microservicio.
- Guardar los artefactos de instalación.

CD (DISTRIBUCIÓN CONTINUA)

Después de la automatización de los diseños y las pruebas de unidad e integración de la CI, la distribución continua automatiza la liberación de ese código validado hacia un repositorio. Por eso, para que el proceso de distribución continua sea eficaz, es importante que la CI ya haya sido incorporada.

Con el fin de implementar las ventajas propuestas por la distribución continua, un nuevo pipeline será construido para esta finalidad, el cual se encargará de:

- Descargar e instalar los artefactos de instalación en el clúster de Kubernetes.
- Ejecutar un conjunto de pruebas funcionalidades automatizadas para verificar la integridad del despliegue.

9. Soporte de la aplicación:

Las aplicaciones desarrolladas por Ingeneo cuentan con un soporte de seis meses después de que el cliente certifica su correcto funcionamiento en el ambiente de producción. La aplicación Atlas ya se encuentra actualmente en producción, el ultimo ajuste realizado correspondiente a temas de garantía fue la actualización de la librería Log4j con el fin de evitar la vulnerabilidad “Log4Shell”, una vulnerabilidad encontrada en ciertas versiones de dicha librería [7].

IX. CONCLUSIONES

Como resultado final del proceso de pasantía del estudiante, se obtuvieron tres módulos de software completos, dichos módulos hacen parte del Sistema Atlas, cada uno de los módulos cumplió con los criterios de aceptación por parte del cliente y del equipo de QA. Supliendo de esta manera las necesidades del cliente; los módulos brindaban el siguiente grupo de funciones: registro de proveedores, registro de regiones, países, red de suministro, promoción de proveedores, ubicaciones de bodegas y tiendas.

La empresa Unicomer dispone de una nueva herramienta que le permite gestionar su red de suministros en los 27 países que opera.

El periodo de pasantía tuvo una duración de seis meses, el costo de los recursos humanos y materiales relativos al pasante para ese periodo fue de $8.101.156, el tiempo fue suficiente para tener el producto (Atlas) a tiempo.

El practicante adquiere experiencia en las siguientes herramientas durante su pasantía:

- Java.
- SpringBoot.
- Angular.
- Nginx.
- Scrum.
- Microservicios.
- SPA.
- Oracle.

Debido a la falta de experiencia del estudiante con algunas de las tecnologías cada módulo tuvo un retraso promedio de cinco días, resultando en un retraso total de tres semanas lo que implica que el estudiante se desfaso en un 15% de lo planeado, a pesar de ello el tutor externo señala que el estudiante tiene potencial y que el retraso era natural debido a su falta de experiencia.

A continuación se muestra el resultado final, pantallas con la interfaz grafica desarrollada por el estudiantes durante su proceso de pasantia.

Módulo de algoritmos: La siguiente pantalla muestra el resultado final del módulo de algoritmos, se trata de un CRUD que registra los diferentes tipos de algoritmos y su estado de activación, esta tabla es usada por otro sistema para determinar qué tipo de algoritmo de ML será ejecutado.

Fig. 19. UI Algoritmos

Módulo de proveedores: Las siguientes pantallas muestran el resultado final del módulo de proveedores, se trata de un conjunto de formularios que permite buscar y registrar la información asociada a los proveedores, además permite mantener registros separados según el país asociado.

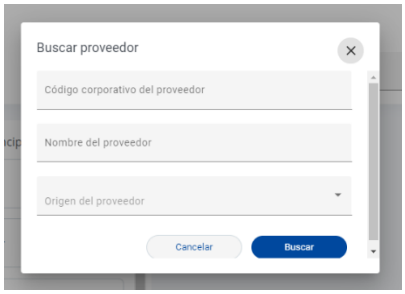


Fig. 20. UI Búsqueda proveedor

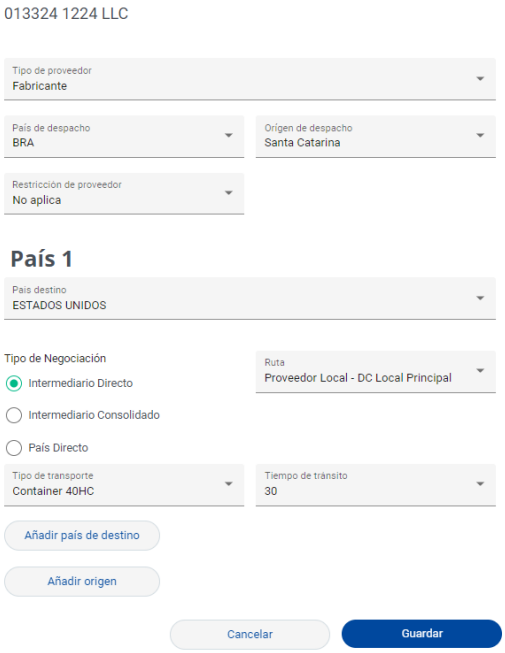


Fig. 21. UI Registro proveedor

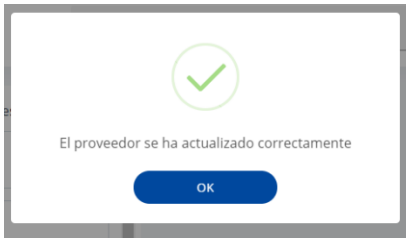


Fig. 22. UI Registro exitoso

Módulo de tiendas: Las siguientes pantallas muestran el resultado final del módulo de tiendas y ubicaciones, se trata de un conjunto de formularios que permite registrar tiendas y bodegas.

Fig. 23. UI Registro exitoso

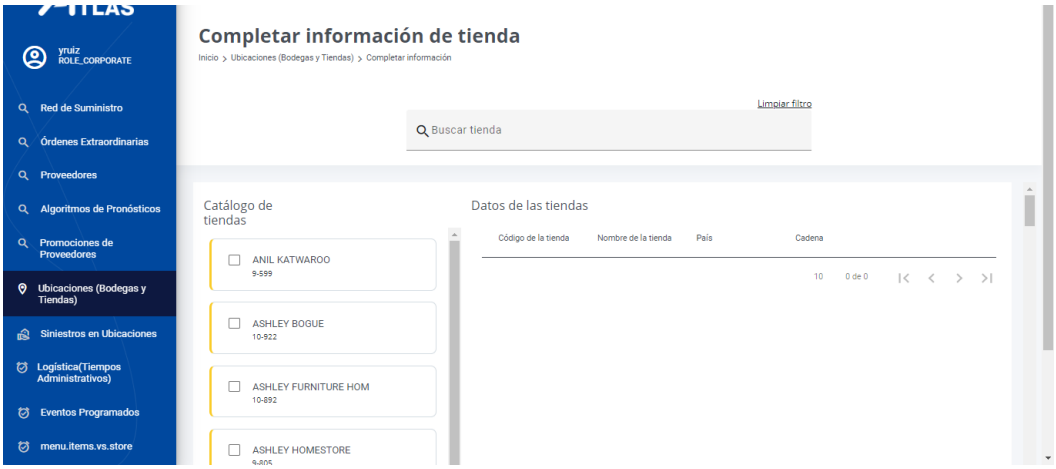


Fig. 24. UI Registro exitoso

Fig. 25. UI Registro exitoso

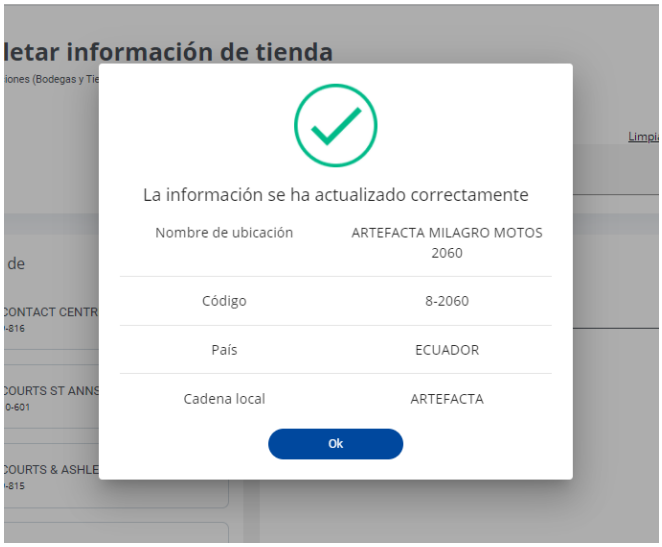


Fig. 26. UI Registro exitoso

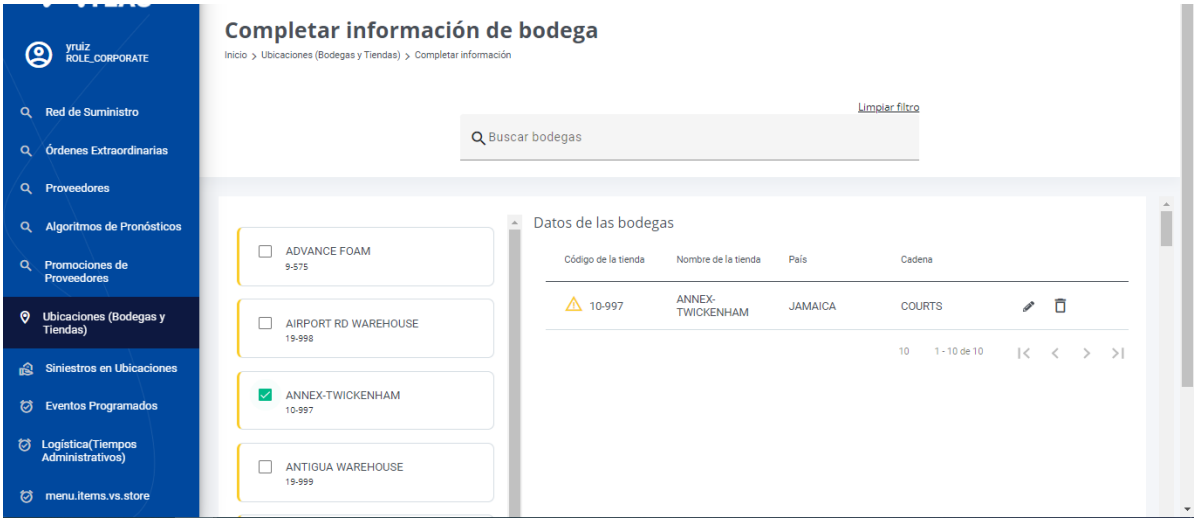


Fig. 27. UI Búsqueda bodega.

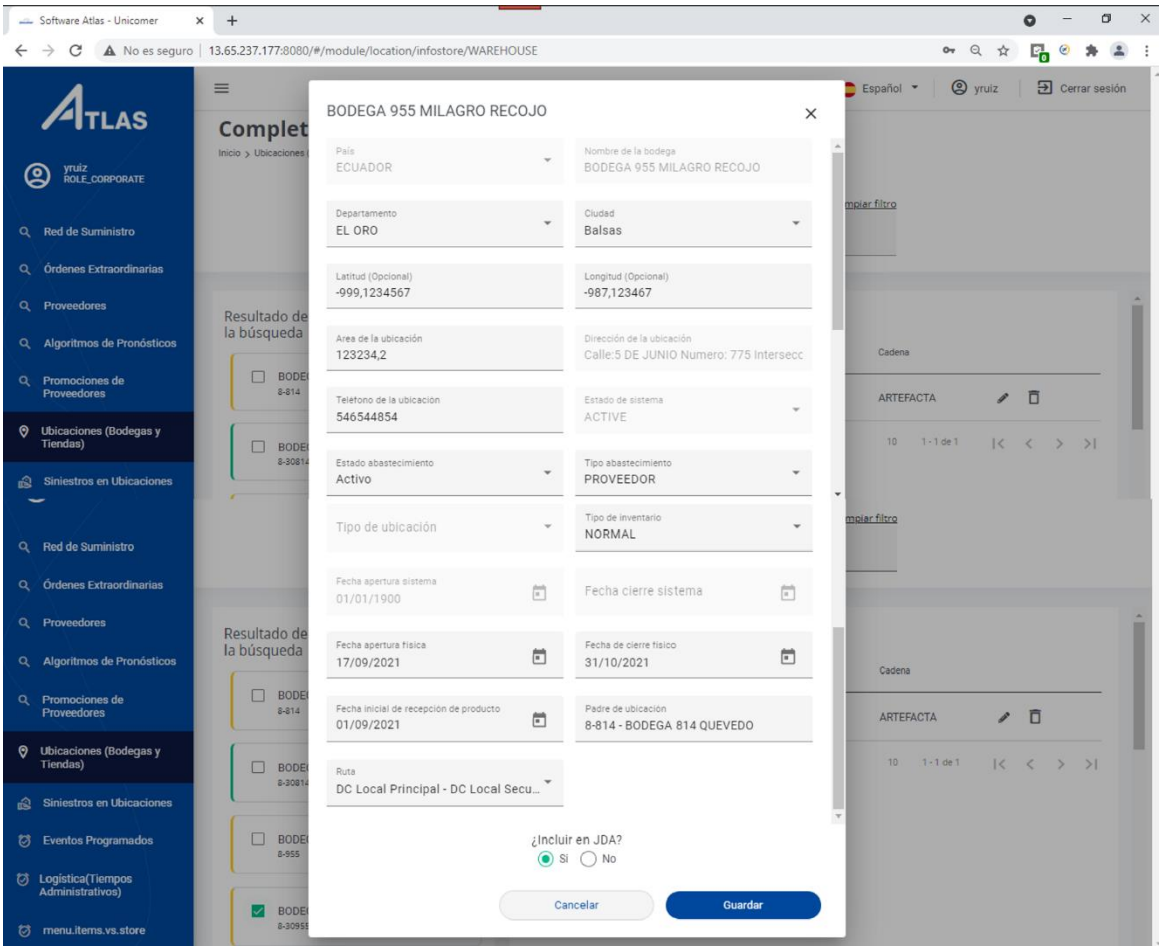


Fig. 28. UI Registro bodega.

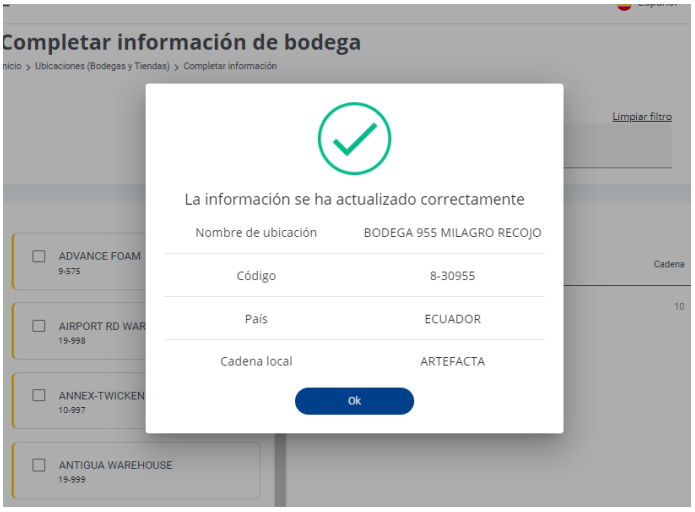


Fig. 29. UI Bodega completada con éxito.

REFERENCIAS

[1] A. N. Cadavid, J. D. F. Martínez, and J. M. Vélez, “Revisión de metodologías ágiles para el desarrollo de software,” *Prospectiva*, vol. 11, no. 2, pp. 30–39, 2013.

[2] M. Fowler, “Lewis J,” *Microservices. Viittattu*, vol. 28, p. 2015, 2014.

[3] J. Gosling and H. McGilton, “The Java language environment,” *Sun Microsystems Computer Company*, vol. 2550, p. 38, 1995.

[4] M. J. R. Martín, F. M. Rodríguez, and A. R. Martin, *Sistemas gestores de bases de datos*. McGraw-Hill Interamericana, 2006.

[5] S. Ramírez Pérez, “Estudio del framework Spring, Spring Boot y microservicios,” 2020.

[6] F. P. Arámburo, “Implementación de framework enfocado al desarrollo rápido de aplicaciones móviles para medianas empresas”.

[7] M. H. Pérez, “¿Qué es Log4Shell? ¿Por qué es la peor vulnerabilidad informática de la década?,” *El País*, Dec. 22, 2021.

ANEXOS

Anexo A. Requisitos de interfaces

A continuación, se describirán todos aquellos requisitos relacionados con integraciones a sistemas existentes dentro de la organización.

Integración Active Directory: Debido a la existencia de varias instancias de Active Directory, Unicomer deberá suministrar para cada una de ellas, la siguiente información:

- Host: Host que se usará para establecer conexión con la instancia Active Directory.
- Puerto: Puerto que se usará para establecer conexión con la instancia Active Directory.
- Credenciales: Usuario con su correspondiente contraseña que será usado para ejecutar consultas sobre el Active Directory.
- Base path: Path dentro del estructura del Active Directory donde se efectuarán las búsquedas.

La comunicación se realizará usando el protocolo LDAP.

Integración ODS: Se configurará un Data Source con el fin de interactuar por medio del protocolo JDBC con este repositorio de información. Básicamente será necesario:

- Host: Se usará para establecer conexión con la instancia de base de datos.
- Puerto: Se usará para establecer conexión con la instancia de base de datos.
- Credenciales: Usuario con su correspondiente contraseña.
- Service name: Nombre del servicio Oracle.