**Proyecto MagikStock**

**(SAD) Software Architecture Document**

**Versión 1.3**

**Integrantes:**

Nicolas Moreno

Cristian Núñez

Catary Rodriguez

**Docente:**

Rocio Contreras

**Asignatura:**

Capstone

**Sección:**

003D

**Identificación del documento**

| **Identificación** | 5 |
| --- | --- |
| **Proyecto** | MagikStock |
| **Versión** | 1.3 |

| **Documento mantenido por** | Nicolas Moreno, Cristian Núñez y Catary Rodriguez |
| --- | --- |
| **Fecha de última revisión** | 20/10/2024 |
| **Fecha de próxima revisión** | 10/11/2024 |

| **Documento aprobado por** | Catary Rodriguez |
| --- | --- |
| **Fecha de última aprobación** | 10/11/2024 |

**Historial de cambios**

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 09/09/2024 | 1.0 | Se comienza el desarrollo del documento. | Catary Rodriguez |
| 20/10/2024 | 1.2 | Se añaden los modelos finales | Nicolás Moreno |
| 10/11/2024 | 1.3 | Cierre del documento | Catary Rodriguez |

**Introducción**

Magikoffee es una cafetería que recientemente ha expandido su negocio al abrir una nueva sucursal, una heladería, lo que ha generado la necesidad de un sistema que permita gestionar su inventario de manera eficiente y centralizada. Debido a que ambas sucursales comparten un inventario común, la gestión manual se ha vuelto ineficiente y propensa a errores. Por ello, se desarrollará un sistema de control de stock automatizado y centralizado con una aplicación web, que permitirá el monitoreo en tiempo real, la sincronización de inventarios y la generación de reportes para ambas sucursales.

1. **Alcance**

El alcance de este documento incluye la definición de los casos de uso y escenarios de calidad que serán la base para el desarrollo del sistema. El software se enfocará en la gestión del inventario compartido entre la cafetería y la heladería, proporcionando una solución integral para la administración del stock, la sincronización entre sucursales, la generación de alertas y el análisis predictivo.

El proyecto está dirigido a los administradores de Magikoffee, quienes podrán gestionar el inventario de manera eficiente, y a los empleados, quienes interactúan con el sistema para registrar ingresos y salidas del inventario.

1. **Referencias**

A continuación, se listan las referencias utilizadas para el desarrollo de este documento y del sistema:

* ISO/IEC 25010:2011 – Systems and software quality requirements and evaluation (SQuaRE) – System and software quality models.
* Metodología PMBOK 6ª edición
* Documentación del sistema de control de inventario de Magikoffee

**Arquitectura de Software**

La arquitectura del sistema de control de stock de Magikoffee está diseñada siguiendo el enfoque del framework 4+1, con un modelo de capas que facilita la organización y visualización de la gestión del sistema. Las vistas incluidas en esta versión del documento son:

* Vista de Casos de Uso y Escenarios de Calidad: Describe los casos de uso más significativos, presenta los actores y una descripción de sus casos de uso asociados. De igual forma describe los escenarios de calidad más relevantes para la arquitectura.

Modelo de Capas:

El sistema de control de stock de Magikoffee está diseñado siguiendo un modelo de capas que incluye:

* Capa de Presentación: Esta capa se encarga de la interfaz de usuario y la interacción con el sistema. Su función principal es proporcionar un medio mediante el cual los usuarios, ya sean administradores o empleados, puedan visualizar y manipular los datos de manera intuitiva y eficiente. En este caso, se utiliza Django para crear una aplicación web que ofrece una serie de funcionalidades clave, como la visualización de movimientos de stock, la gestión de inventarios mediante un CRUD (productos, categorías, proveedores), y la interacción con un dashboard interactivo que muestra predicciones de productos. La autenticación de usuarios y la asignación de roles (Empleado / Administrador) se gestionan mediante funciones nativas de Django y decoradores.
* Capa de Lógica de Negocio: Esta capa se ocupa de la gestión de las reglas y procesos que permiten al sistema cumplir su propósito funcional. En este caso, incluye la sincronización en tiempo real del inventario entre sucursales, la manipulación de datos de stock (ingresos y salidas), y la gestión de alertas de stock bajo mediante reglas definidas (rojo < 5, amarillo entre 5 y 10, verde > 10). Además, esta capa es responsable de generar predicciones de impacto para los productos en base a los datos históricos, que luego se visualizan en el dashboard interactivo. Toda esta lógica se implementa utilizando el framework Django con Python, asegurando que las operaciones y procesos se realicen de manera eficiente.
* *Capa de Datos:* Esta capa gestiona la persistencia y acceso a los datos del sistema. Utiliza una base de datos relacional SQL (PostgreSQL) para almacenar la información relacionada con los productos, proveedores, empleados, movimientos de stock y configuraciones del sistema. En esta capa se gestionan las consultas y actualizaciones de los datos, garantizando que las transacciones se realicen correctamente y que la integridad de los datos esté asegurada. Se implementan triggers y JavaScript para validar y mantener la consistencia de los datos en la base de datos durante las operaciones, como la manipulación del inventario y la actualización de los movimientos. Esta capa también asegura que la base de datos se maneje de manera eficiente y que la información esté siempre disponible para las demás capas del sistema.

| **Casos de uso** | | | | |
| --- | --- | --- | --- | --- |
| **Código** | **Nombre** | **Actores** | **Descripción** | **Prioridad** |
| CU-001-001 | Monitorear Stock en Tiempo Real | Administradores, Empleados | Permite a los usuarios monitorear el inventario en tiempo real para ambas sucursales. | Alta |
| CU-002-001 | Gestionar Productos | Administradores | Permite agregar, modificar y eliminar productos en el sistema de inventario. | Alta |
| CU-003-001 | Sincronizar Inventario | Sistema | Sincroniza el inventario de ambas sucursales en un sistema centralizado. | Alta |
| CU-004-001 | Configurar Alertas de Stock Bajo | Administradores | Configura alertas cuando el inventario de un producto se encuentre bajo un umbral predefinido. | Alta |
| CU-005-001 | Registrar Entrada de Stock | Empleados, Administradores | Permite registrar la entrada de nuevos productos al inventario al recibir mercancía. | Media |
| CU-006-001 | Administrar Proveedores | Administradores | Permite el registro y la gestión de proveedores asociados al inventario. | Media |
| CU-007-001 | Gestionar Accesos y Roles | Administradores | Gestiona los permisos y roles de acceso de los usuarios del sistema. | Alta |
| CU-008-001 | Acceso Remoto al Sistema | Administradores | Permite a los administradores acceder al sistema desde cualquier dispositivo con conexión a internet. | Alta |
| CU-009-001 | Realizar Copia de Seguridad | Sistema | Realiza copias de seguridad automáticas del sistema de inventario para prevenir la pérdida de datos. | Media |

| **Escenarios de Calidad** | | | |
| --- | --- | --- | --- |
| **Número** | **Nombre** | **Atributo de calidad** | **Descripción** |
| 1 | Interfaz Web Operativa | Usabilidad | El sistema debe proporcionar una interfaz web que sea operativa y fácil de usar para que los empleados y administradores gestionen el inventario eficientemente. Este requisito es observable y se asocia con la operabilidad. |
| 2 | Facilidad de Uso de la Interfaz | Usabilidad | La interfaz debe ser intuitiva y fácil de entender, permitiendo a los empleados y administradores usar el sistema con una mínima curva de aprendizaje. Esto es observable y está asociado a la comprensibilidad y aprendizaje. |
| 3 | Accesibilidad Remota | Portabilidad | El sistema debe ser accesible desde cualquier dispositivo con conexión a internet, adaptándose a diferentes plataformas y resoluciones. Es un atributo observable asociado con la capacidad de ser instalada y la adaptabilidad. |
| 4 | Seguridad de Acceso | Seguridad | El sistema debe asegurar la confidencialidad y autenticidad, protegiendo la información sensible del inventario mediante control de accesos basado en roles. Es un requisito no observable que garantiza la seguridad del sistema. |
| 5 | escalabilidad del Sistema | Mantenibilidad | El sistema debe ser modular y capaz de expandirse para adaptarse al crecimiento del negocio sin requerir grandes modificaciones. Es un atributo no observable asociado con la modularidad y capacidad de expansión. |
| 6 | Backup y Recuperación Automática | Fiabilidad | El sistema debe realizar copias de seguridad automáticas periódicamente y permitir la recuperación de datos en caso de falla, asegurando la continuidad de las operaciones. Es un atributo no observable relacionado con la capacidad de recuperación. |

* Vista de Metas y Restricciones: Describe restricciones tecnológicas, normativas, estándares, etc., los cuales influyen sobre las decisiones arquitectónicas, del producto y del proceso de desarrollo.

**Metas del Proyecto**

1. Optimización del Control de Inventario:

* Automatizar la gestión de inventario para minimizar errores humanos y mejorar la eficiencia operativa de ambas sucursales.

1. Monitoreo en Tiempo Real:

* Proveer a los administradores y empleados la capacidad de monitorear el inventario en tiempo real para tomar decisiones informadas y oportunas sobre la reposición de stock.

1. Sincronización Centralizada:

* Asegurar que la información del inventario esté sincronizada entre la cafetería y la heladería, evitando duplicidades y desabastecimiento.

1. Seguridad y Acceso Controlado:

* Implementar un sistema de control de acceso basado en roles para proteger la información del inventario y garantizar que solo usuarios autorizados puedan realizar modificaciones.

1. Escalabilidad y Adaptabilidad:

* Diseñar un sistema escalable que permita integrar nuevas sucursales o funcionalidades a medida que el negocio crezca.

1. Accesibilidad Remota:

* Asegurar que el sistema sea accesible desde cualquier dispositivo con conexión a internet para permitir a los administradores monitorear y gestionar el inventario desde cualquier lugar.

1. Facilidad de Uso:

* Desarrollar una interfaz web intuitiva y fácil de usar para minimizar la curva de aprendizaje y los errores de los usuarios.

**Restricciones del Proyecto**

1. Presupuesto Limitado:
   * El desarrollo y la implementación del sistema deben mantenerse dentro del presupuesto asignado por Magikoffee.
2. Tiempo de Implementación:
   * El sistema debe estar completamente implementado y funcional en un plazo máximo de 16 semanas para cumplir con los objetivos.
3. Compatibilidad con Dispositivos Existentes:
   * El sistema debe ser compatible con los dispositivos y sistemas operativos que actualmente utilizan ambas sucursales (PC, tablets, smartphones).
4. Capacidad de Infraestructura:
   * El sistema debe operar dentro de las limitaciones de la infraestructura tecnológica actual de Magikoffee.
5. Normativas de Seguridad:
   * El sistema debe cumplir con las normativas de seguridad de datos y protección de la información, garantizando la confidencialidad y privacidad de la información de inventario.
6. Acceso a Internet:
   * Dado que el sistema es web, depende de una conexión a internet estable para su funcionamiento. Las sucursales deben contar con acceso constante a internet para asegurar la operatividad del sistema.
7. Disponibilidad y Fiabilidad:
   * El sistema debe ser fiable y estar disponible al menos un 98% del tiempo, ya que se utilizará para operaciones críticas en ambas sucursales.
8. Uso de Tecnologías Estándar:
   * Deben usarse tecnologías que sean estándar y ampliamente aceptadas para asegurar la sostenibilidad y la facilidad de mantenimiento del sistema.

**Objetivos y Restricciones de la Arquitectura**

A continuación se revisan las metas y restricciones de la arquitectura.

**Objetivos de la arquitectura**

De acuerdo con las reuniones y al análisis de los requerimientos, se listan los principales conductores iniciales de la arquitectura los cuales corresponden a las metas arquitectónicas iniciales:

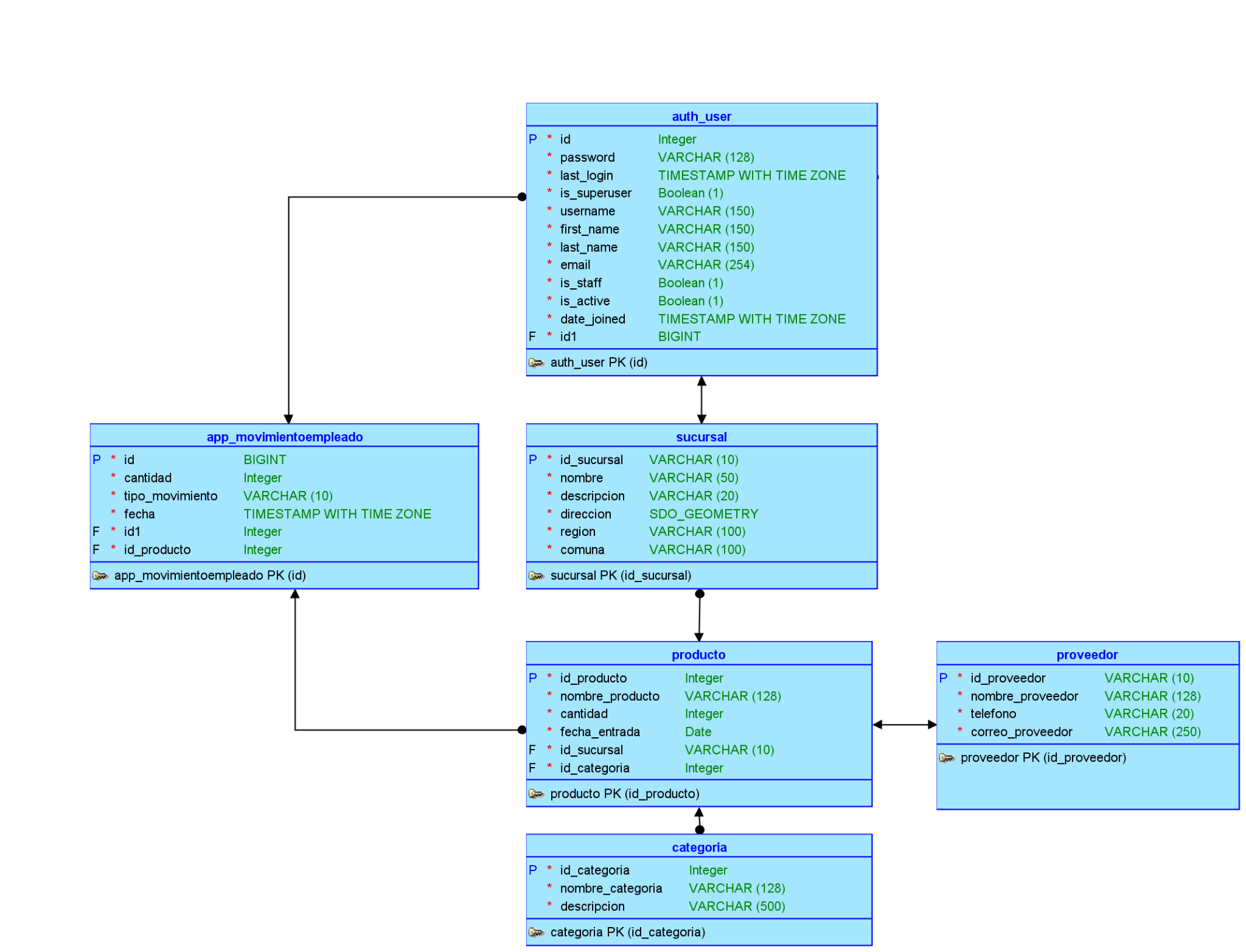
* Usabilidad: Operabilidad, Compresibilidad, Aprendizaje
* Portabilidad: Adaptabilidad.
* Seguridad: Autenticidad, Confidencialidad.
* Compatibilidad:Coexistencia.
* Mantenibilidad: Modularidad, Capacidad de expansión.
* Fiabilidad: Capacidad de recuperación.

**Restricciones de la Arquitectura**

Las restricciones de la arquitectura de software planteada para el sistema contempla aspectos importantes en las etapas del desarrollo, a continuación se describe las siguientes restricciones del diseño de la arquitectura:

* El proyecto se realizará sobre una arquitectura Web
* La construcción de la interfaz Web tendrá un plazo máximo de 12 semanas.
* El software tendrá la facilidad de poder visualizarse mediante una conexión a Internet por medio de un navegador.
* La implementación de la base de datos tendrá un plazo máximo de 2 semanas.
* El sistema operativo de desarrollo es Microsoft Windows.
* El protocolo de comunicación de la capa de aplicaciones y la base de datos será TCP/IP.

1. **Arquitectura Lógica**

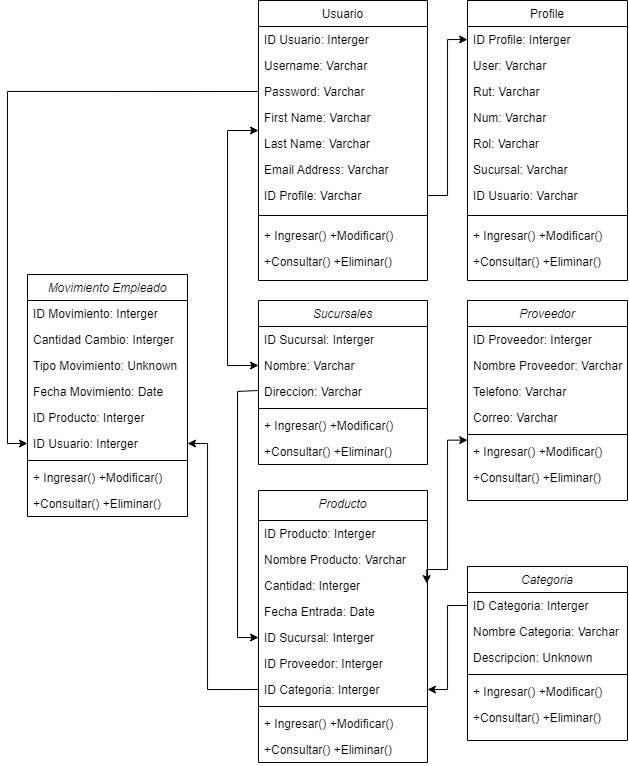


A continuación se presenta una vista lógica de la aplicación expresado en dos diagramas, uno de ellos que muestra la parte estructural o estática de la aplicación (módulos), y otra vista que representa la parte dinámica (componentes y conectores).

[BDD\_V5\_Logical\_Bechman.png](https://drive.google.com/file/d/1rIeNLCtig8x1vDhFlIo9s4BdKxnw1zwo/view?usp=sharing)

**Parte Estructural**

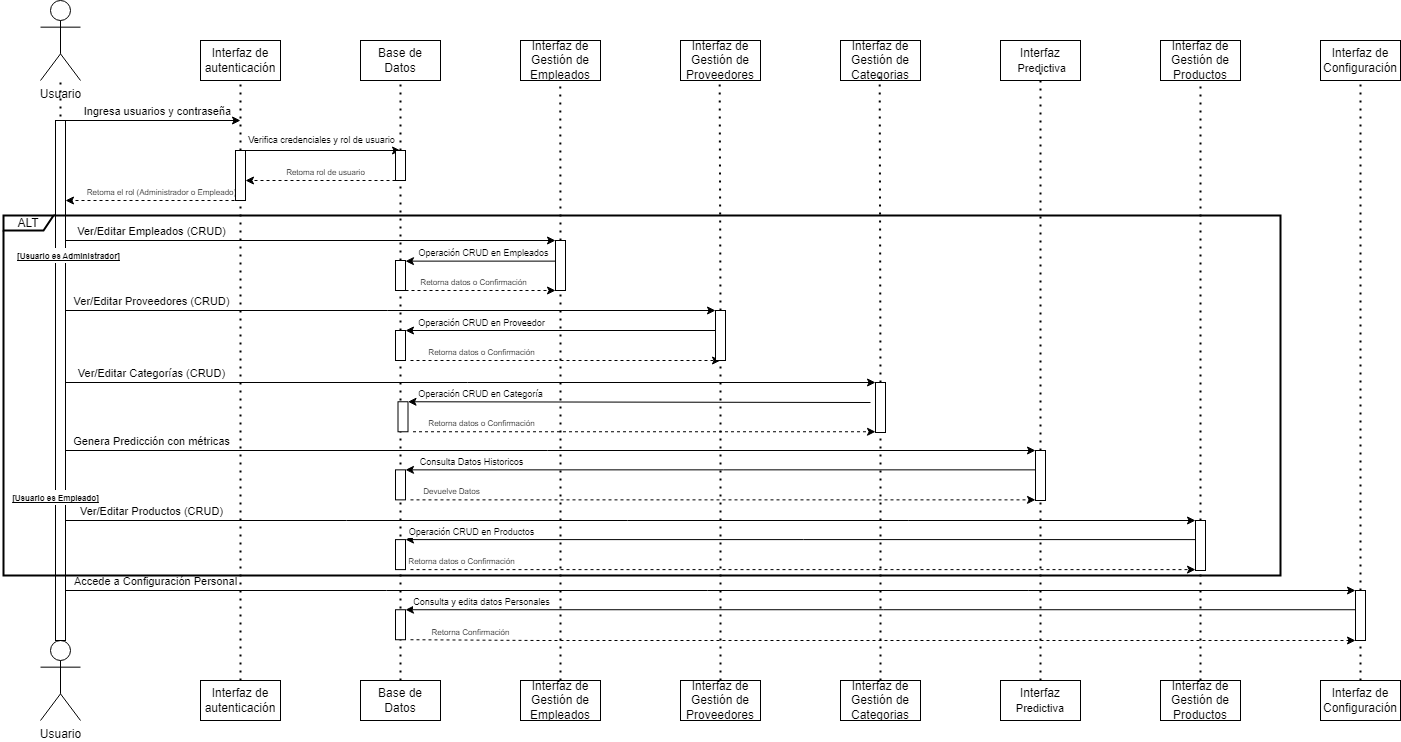
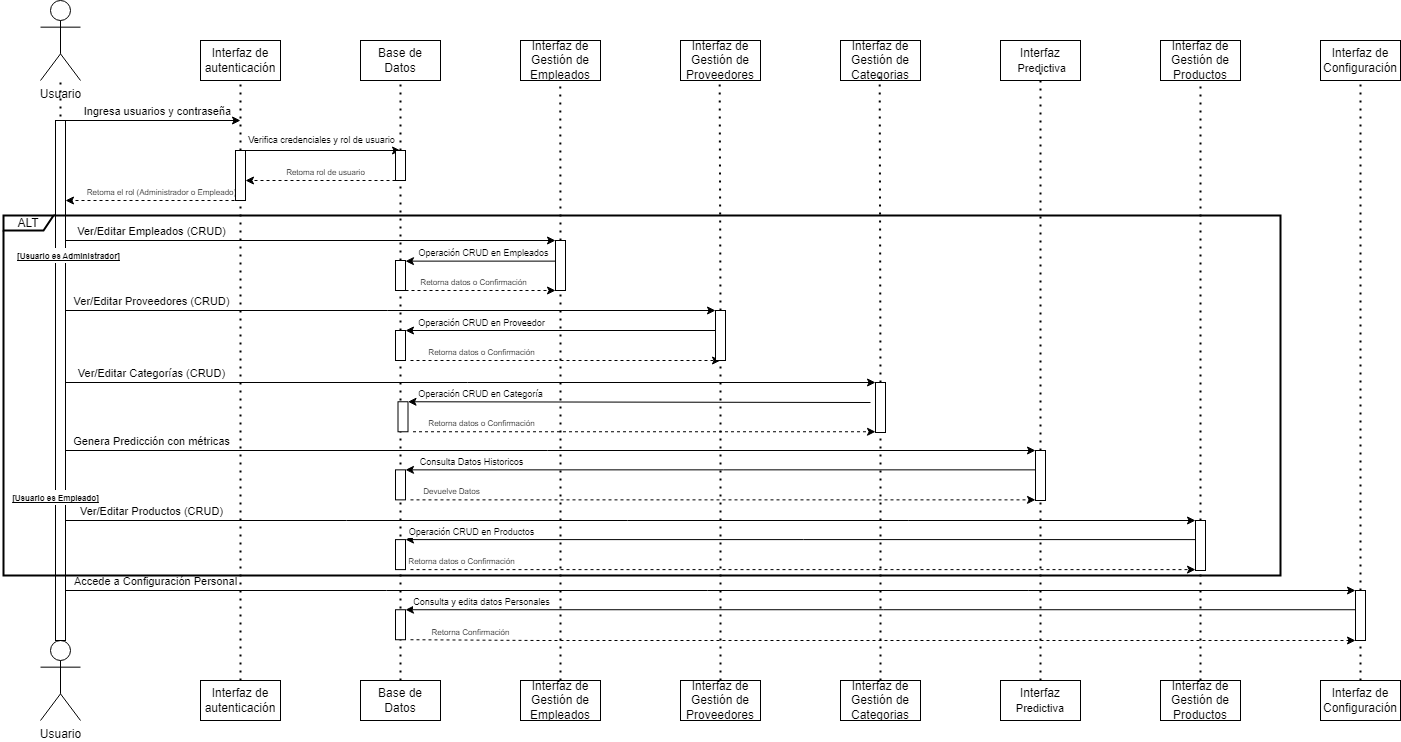
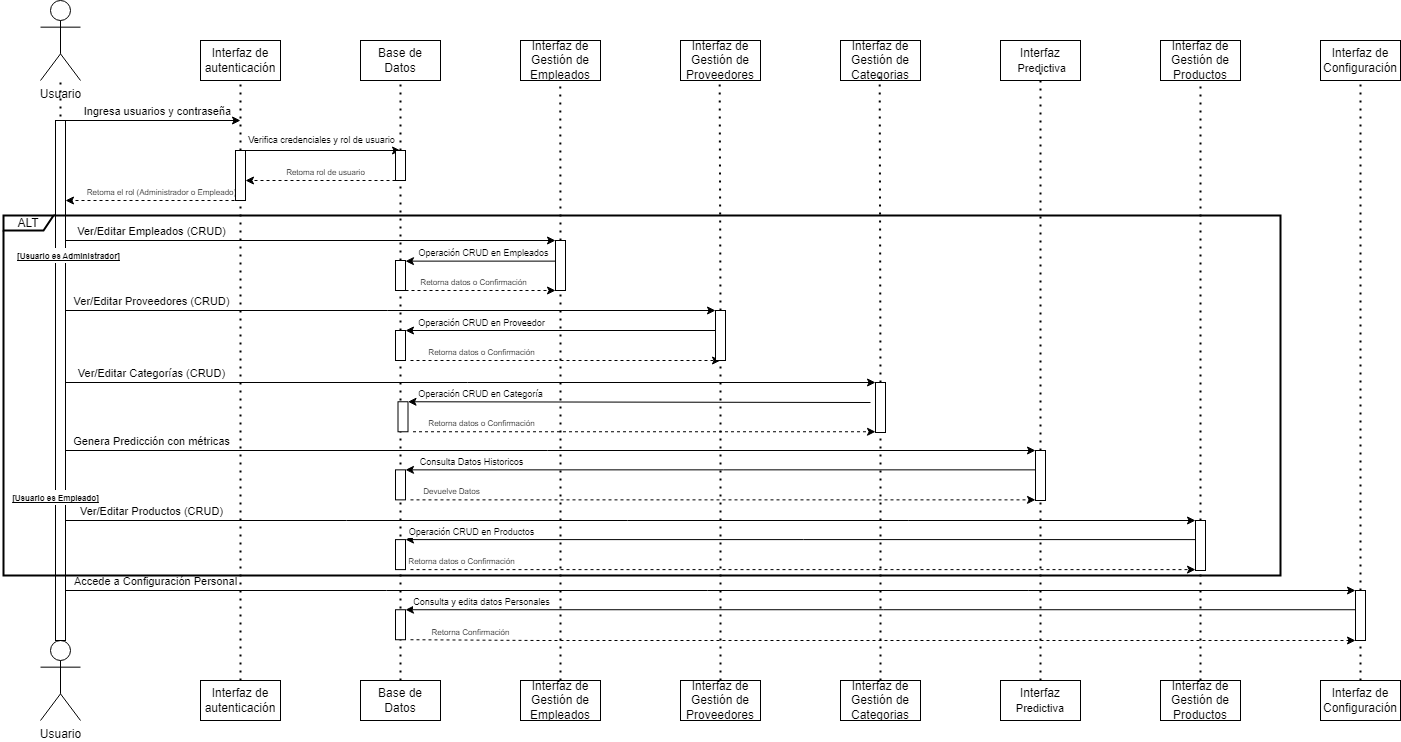
**Ilustración 1: Diagrama de clases**



[Diagrama de clases.drawing](https://drive.google.com/file/d/14dBVDBkofPsB8SZDtUa_xioXh2u8oORx/view?usp=sharing)

**Parte Dinámica**

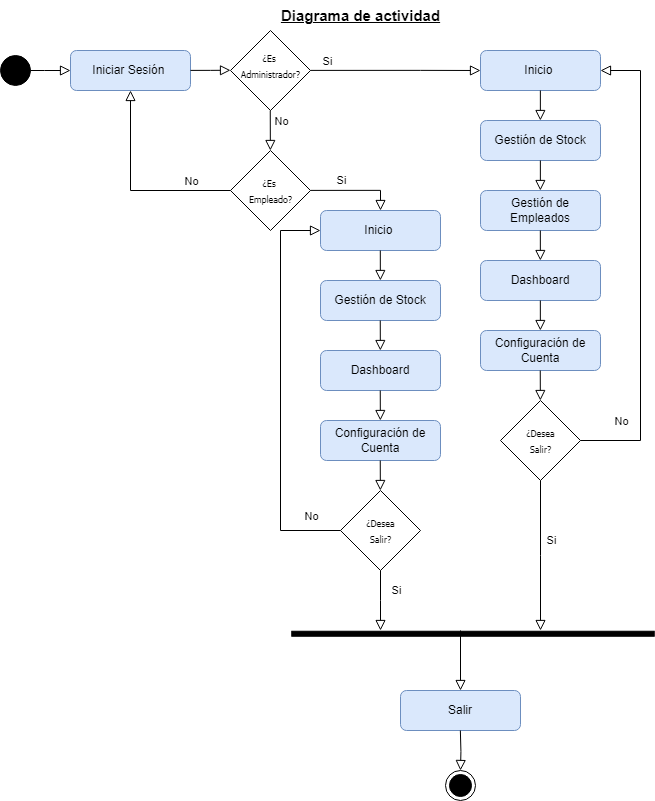
**Ilustración 2: Diagrama de secuencia**

****

[Diagrama de secuencia UML.drawio](https://drive.google.com/file/d/1l9OYdUAgivdmyWG4KT-O7K6lqmptTIMG/view?usp=sharing)

1. **Arquitectura de Procesos**

**Ilustración 3: Diagrama de actividad**

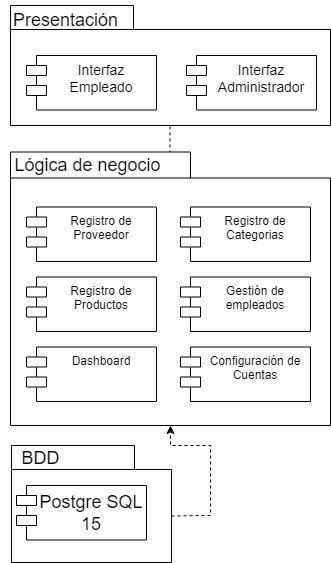


[Diagrama de actividad.drawio](https://drive.google.com/file/d/1RKLYUlhZxGRw5agIdaQtKa_qggdoSwv-/view?usp=sharing)

1. **Arquitectura de desarrollo**

En esta vista se aprecia que existirán dos módulos principales que contendrán distintas funcionalidades de la aplicación. A continuación se describen:

**Ilustración 4: Vista de Implementación (Diagramas de componentes o paquetes)**

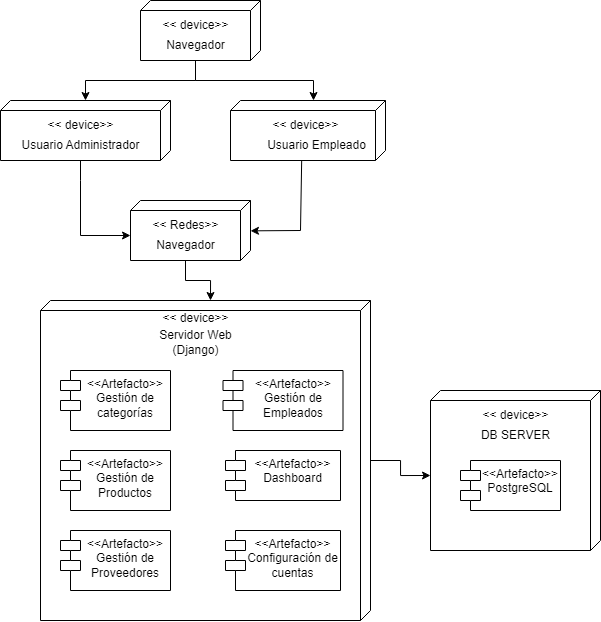
****

[**Diagramas de componentes .drawio**](https://drive.google.com/file/d/1GNeF9oHrVVk0esQwJVdzsYZ0RMvA6V7X/view?usp=sharing)

1. **Arquitectura física**

En esta vista se despliegan los nodos que participan con el sistema. Los nodos principales son los nodos Servidor de Integración. Características a continuación:

**Ilustración 5: Diagrama de Despliegue**

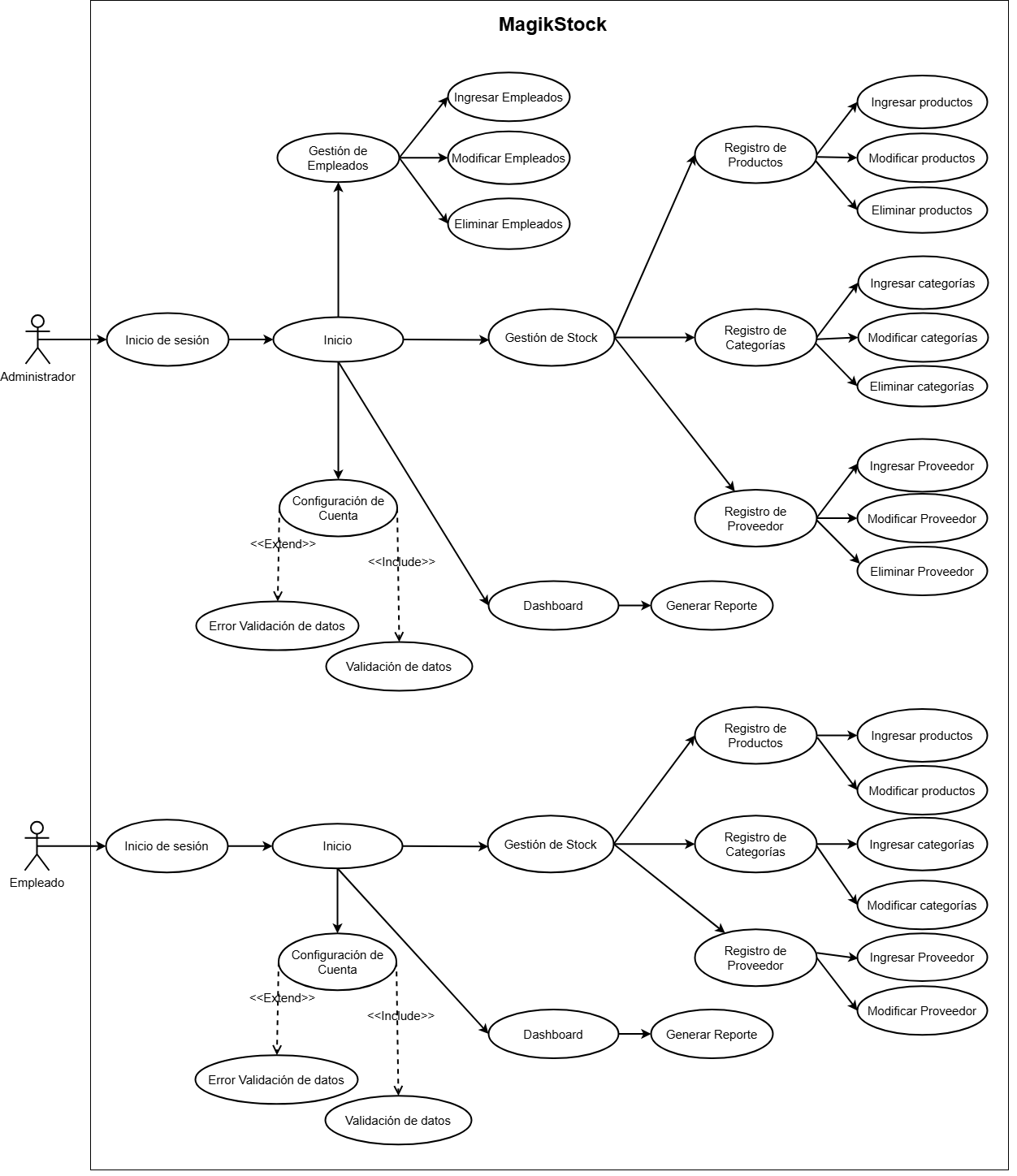
****

[**Diagrama de Despliegue.drawio**](https://drive.google.com/file/d/1ik6hANXyVzfgeh2rfx0zVFtHChZtquq-/view?usp=sharing)

1. **Escenarios**

Esta sección describe en detalle el conjunto de escenarios funcionales y no funcionales que obtuvieron la mayor prioridad en el análisis. Para esto se presenta y describe el diagrama de casos de uso y los casos de uso prioritarios, así como los escenarios en que uno o más atributos de calidad se ven involucrados de manera significativa.

El modelo de casos de uso puede ser encontrado en el documento “Casos de Uso”.



[Modelo de Casos de Uso.drawio](https://drive.google.com/file/d/1FBwTaqdjqSWCIbT5qrvBEKokr8zPW833/view?usp=sharing)

**Especificación de Casos de Uso Relevantes**

Los casos de uso considerados los más relevantes para el desarrollo de la arquitectura fueron determinados. Los criterios usados para dicha determinación fueron:

* Su implementación implica varios nodos de la vista de despliegue.
* Su implementación es de alto riesgo.
* Incluye muchos conceptos y relaciones del dominio.
* Incluye posibles escenarios críticos de calidad.

A continuación se listan los casos de uso relevantes, los cuales pueden ser encontrados con su especificación detallada en el documento “Casos de Uso”.

| **Casos de uso** | | | |
| --- | --- | --- | --- |
| **Código** | **Nombre** | **Actores** | **Prioridad** |
| CU-001-001 | Iniciar sesión | Administrador, Empleado | Muy alta |
| CU-002-001 | Gestión de Stock | Administrador, Empleado | Alta |
| CU-002-002 | Registro de proveedor | Administrador | Alta |
| CU-002-003 | Registro de Categorías | Administrador | Alta |
| CU-002-004 | Registro de Productos | Administrador, Empleado | Alta |
| CU-003-001 | Gestión de Empleados | Administrador | Muy Alta |
| CU-004-001 | Dashboard | Administrador, Empleado | Alta |
| CU-002-007 | Configuración de cuenta | Administrador, Empleado | Alta |

**Especificación de los Escenarios de Calidad Relevantes**

Después de un análisis en conjunto con los stakeholders, los escenarios de calidad se expresan a continuación:

| Escenario de Calidad N°1 | |
| --- | --- |
| Atributo de Calidad Asociado (Característica): Usabilidad  Sub Característica: Operabilidad | |
| Descripción: Un sistema de software cuya interfaz sea web, accesible para los empleados y administradores, permitiendo el manejo eficiente del stock. | |
| **Diagrama  Descripción generada automáticamente** | |
| Fuente del Estímulo: | Usuario |
| Estímulo: | El usuario quiere acceder al sistema desde un navegador web. |
| Artefacto: | Sistema |
| Ambiente: | Producción |
| Respuesta: | El sistema debe permitir la gestión completa del inventario desde una interfaz web. |
| Medida de Respuesta: | El usuario podrá interactuar con el sistema en un entorno web, con acceso completo a las funcionalidades de gestión de inventario. |

| Escenario de Calidad N°2 | |
| --- | --- |
| Atributo de Calidad Asociado (Característica): Usabilidad  Sub Característica: Compresibilidad, Aprendizaje | |
| Descripción: La interfaz debe ser intuitiva y fácil de usar para reducir errores humanos en el manejo del stock. | |
| **Diagrama  Descripción generada automáticamente** | |
| Fuente del Estímulo: | Usuario (Administrador/Empleado) |
| Estímulo: | El usuario realiza tareas de gestión de inventario y necesita entender la interfaz sin capacitación extensa. |
| Artefacto: | Interfaz de usuario |
| Ambiente: | Producción |
| Respuesta: | El sistema proporciona una interfaz clara y fácil de aprender, reduciendo el tiempo de capacitación y errores operativos. |
| Medida de Respuesta: | El usuario es capaz de utilizar las funciones clave sin asistencia en un tiempo reducido, y con una tasa de errores mínima en las primeras interacciones. |

| Escenario de Calidad N°3 | |
| --- | --- |
| Atributo de Calidad Asociado (Característica): Portabilidad  Sub Característica: Capacidad de ser instalada, Adaptabilidad | |
| Descripción: El sistema debe ser accesible desde cualquier dispositivo con conexión a internet para monitoreo remoto. | |
| **Diagrama  Descripción generada automáticamente** | |
| Fuente del Estímulo: | Administrador |
| Estímulo: | El administrador desea acceder al sistema desde un dispositivo móvil o fuera de la red interna. |
| Artefacto: | Sistema web |
| Ambiente: | Producción |
| Respuesta: | El sistema permite el acceso remoto, adaptándose a múltiples dispositivos y resoluciones de pantalla. |
| Medida de Respuesta: | El sistema es completamente funcional en al menos 95% de los dispositivos móviles y navegadores más comunes, sin pérdida de funcionalidad. |

| Escenario de Calidad N°4 | |
| --- | --- |
| Atributo de Calidad Asociado (Característica): Seguridad  Sub Característica: Confidencialidad, Autenticidad | |
| Descripción: El sistema debe contar con control de accesos basado en roles para proteger la información sensible del inventario. | |
| **Diagrama  Descripción generada automáticamente** | |
| Fuente del Estímulo: | Administrador |
| Estímulo: | Un usuario intenta acceder a datos sensibles sin los permisos correspondientes. |
| Artefacto: | Sistema de control de acceso |
| Ambiente: | Producción |
| Respuesta: | El sistema deniega el acceso a usuarios sin privilegios. |
| Medida de Respuesta: | El 100% de los intentos no autorizados son bloqueados, y los intentos se registran para auditoría. |

| Escenario de Calidad N°5 | |
| --- | --- |
| Atributo de Calidad Asociado (Característica): Mantenibilidad  Sub Característica: Modularidad, Capacidad de expansión | |
| Descripción: El sistema debe ser escalable para soportar el crecimiento futuro del negocio (nuevas sucursales). | |
| **Diagrama  Descripción generada automáticamente** | |
| Fuente del Estímulo: | Desarrollador |
| Estímulo: | Se planea abrir nuevas sucursales y el sistema debe adaptarse sin grandes modificaciones. |
| Artefacto: | Arquitectura del sistema |
| Ambiente: | Desarrollo y producción |
| Respuesta: | El sistema debe ser ampliado para gestionar el inventario de las nuevas sucursales con mínimo impacto en la arquitectura actual. |
| Medida de Respuesta: | El sistema puede integrar nuevas sucursales con menos del 10% de esfuerzo en reestructuración o ajustes al código existente. |

| Escenario de Calidad N°6 | |
| --- | --- |
| Atributo de Calidad Asociado (Característica): Fiabilidad  Sub Característica: Capacidad de recuperación | |
| Descripción: El sistema debe realizar copias de seguridad automáticas periódicamente para prevenir la pérdida de datos. | |
| **Diagrama  Descripción generada automáticamente** | |
| Fuente del Estímulo: | Sistema |
| Estímulo: | El sistema experimenta una pérdida de datos o falla crítica. |
| Artefacto: | Sistema de backup |
| Ambiente: | Producción |
| Respuesta: | El sistema restaura los datos desde la copia de seguridad más reciente sin pérdida significativa de información. |
| Medida de Respuesta: | El tiempo de recuperación de datos es menor a 15 minutos, y la pérdida de datos es inferior a 1 hora de trabajo. |

**Tamaño y desempeño**

Las principales decisiones arquitectónicas se tomaron considerando la restricción del tiempo de construcción, que es de 16 semanas. Dado que la aplicación web para la gestión de stock de Magikoffe debe implementarse en este tiempo ajustado, se optó por una arquitectura simple y conocida que minimice los riesgos de implementación.

Se modelará la aplicación dividiéndola en tres componentes principales: backend, frontend y base de datos. Esta modularización permite un diseño claro y un desarrollo enfocado en las funcionalidades esenciales. Se decidió implementar el backend y frontend utilizando Django, mientras que la base de datos estará en PostgreSQL. Adicionalmente, para el servidor se utilizará Heroku, una solución rentable que garantiza disponibilidad 24/7 y escalabilidad suficiente para las necesidades del proyecto.

En cuanto al diseño de la arquitectura, se prioriza la seguridad basada en roles, dado que los usuarios principales serán empleados y administradores. Esto permite gestionar accesos y permisos de manera clara, manteniendo la información sensible protegida. Asimismo, el sistema se implementará con una solución monolítica, ya que no se requiere una separación compleja de servicios ni una arquitectura distribuida.

Para garantizar la calidad del software, se optará por realizar pruebas unitarias utilizando pytest, un marco de pruebas sencillo y eficaz que asegura que cada módulo del sistema funcione correctamente. Esta estrategia permitirá que, aunque los recursos sean limitados, se mantenga un nivel aceptable de calidad antes de la puesta en producción.

Un elemento adicional es el uso de un modelo de forecasting implementado con Prophet, desarrollado en Python, para aportar valor añadido a la gestión de stock. Esto permitirá a los administradores predecir necesidades futuras basándose en patrones históricos.

Por último, los datos entrantes se manejaron exclusivamente a través de interacciones con la base de datos PostgreSQL, evitando la complejidad de integraciones con sistemas externos. Dado que no se requieren procesos automáticos como temporizadores, la solución propuesta es sencilla pero efectiva para las necesidades de Magikoffe.

**Anexos**

**Anexo 1: código de creación de la BD ORACLE POSTGRESQL**

| **public.auth\_user** | **public.app\_profile** |
| --- | --- |
| **CREATE TABLE** public.auth\_user (  **id** SERIAL PRIMARY KEY,  **password** VARCHAR,  **last\_login** TIMESTAMP,  **is\_superuser** BOOLEAN,  **username** VARCHAR,  **first\_name** VARCHAR,  **last\_name** VARCHAR,  **email** VARCHAR,  **is\_staff** BOOLEAN,  **is\_active** BOOLEAN,  **date\_joined** TIMESTAMP  ); | **CREATE TABLE** public.app\_profile (  **id** SERIAL PRIMARY KEY,  **rut** VARCHAR,  **num** VARCHAR,  **rol** VARCHAR,  **user\_id** INTEGER **UNIQUE REFERENCES public.auth\_user(id)**,  **sucursal\_id** INTEGER **REFERENCES public.sucursal(id\_sucursal)**  ); |
| **public.sucursal** | **public.producto** |
| **CREATE TABLE** public.sucursal  (  **id\_sucursal** SERIAL PRIMARY KEY,  **nombre** VARCHAR,  **descripcion** TEXT,  **direccion** VARCHAR,  **region** VARCHAR,  **comuna** VARCHAR  **);** | **CREATE TABLE public.producto**  **(**  **id\_producto** SERIAL PRIMARY KEY**,**  **nombre\_producto** VARCHAR**,**  **cantidad** INTEGER**,**  **fecha\_entrada** DATE**,**  **id\_categoria\_id** INTEGER **REFERENCES public.categoria(id\_categoria),**  **id\_proveedor\_id** INTEGER **REFERENCES public.proveedor(id\_proveedor),**  **id\_sucursal\_id** INTEGER **REFERENCES public.sucursal(id\_sucursal)**  **);** |
| **public.app\_movimientoempleado** | **public.categoria** |
| **CREATE TABLE public.app\_movimientoempleado**  **(**  **id** SERIAL **PRIMARY KEY,**  **cantidad** INTEGER**,**  **tipo\_movimiento** VARCHAR**,**  **fecha** DATE**,**  **producto\_id** INTEGER **REFERENCES public.producto(id\_producto),**  **empleado\_id** INTEGER **REFERENCES public.app\_profile(id)**  **);** | **CREATE TABLE public.categoria**  **(**  **id\_categoria** SERIAL **PRIMARY KEY,**  **nombre\_categoria** VARCHAR**,**  **descripcion** TEXT  **);** |
| **public.proveedor** |  |
| **CREATE TABLE public.proveedor (**  **id\_proveedor SERIAL PRIMARY KEY,**  **nombre\_proveedor VARCHAR,**  **telefono VARCHAR,**  **correo\_proveedor VARCHAR**  **);** |  |

**Anexo 2: Código de creación de las clases**

| **public.sucursal** | **public.app\_profile** |
| --- | --- |
| **class** **Sucursal**(**models**.**Model**):  id\_sucursal = **models**.**CharField**(max\_length=10, primary\_key=True)  **DESCRIPCION\_SUCURSALES** = [  **('cafeteria', 'Cafetería'),**  **('heladeria', 'Heladería')**,  ]  nombre = models.CharField(max\_length=50, unique=False)  descripcion = models.CharField(max\_length=20, choices=DESCRIPCION\_SUCURSALES)  direccion = models.CharField(max\_length=255)  region = models.CharField(max\_length=100)  comuna = models.CharField(max\_length=100)  **class** Meta:  db\_table = **'sucursal'**  **def** \_\_str\_\_(self):  **return** self.get\_descripcion\_display() | **class** **Profile**(**models.Model**):  **USER\_TYPES** = [  **('empleado', 'Empleado'),**  **('administrador', 'Administrador')**,  ]  user = models.OneToOneField(User, on\_delete=models.CASCADE)  rut = models.CharField(max\_length=20)  num = models.CharField(max\_length=20)  rol = models.CharField(max\_length=20, choices=USER\_TYPES, default=**'empleado'**)  sucursal = models.ForeignKey(Sucursal, on\_delete=models.SET\_NULL, null=True, blank=True)  **def** \_\_str\_\_(self):  **return** f"{self.user.username} - {self.sucursal.nombre **if** self.sucursal **else** **'Sin Sucursal'**}" |
| **public.proveedor** | **public.categoria** |
| **class** **Proveedor**(**models.Model**):  id\_proveedor = models.CharField(max\_length=10, primary\_key=True)  nombre\_proveedor = models.CharField(max\_length=128)  telefono = models.CharField(max\_length=20, null=True, blank=True)  correo\_proveedor = models.EmailField(max\_length=250, null=True, blank=True)  **def** save(self, \*args, \*\*kwargs):  self.nombre\_proveedor = self.nombre\_proveedor.strip().lower()  super(Proveedor, self).save(\*args, \*\*kwargs)  **class** Meta:  db\_table = **'proveedor'**  **def** \_\_str\_\_(self):  **return** self.nombre\_proveedor | **class** **Categoria**(**models.Model**):  id\_categoria = models.AutoField(primary\_key=True)  nombre\_categoria = models.CharField(max\_length=128)  descripcion = models.CharField(max\_length=500, blank=True)  **class** Meta:  db\_table = **'categoria'**  **def** \_\_str\_\_(self):  **return** self.nombre\_categoria |
| **public.producto** | **public.app\_movimientoempleado** |
| c**class** **Producto**(**models.Model**):  id\_producto = models.AutoField(primary\_key=True)  nombre\_producto = models.CharField(max\_length=128)  cantidad = models.IntegerField()  id\_categoria = models.ForeignKey(Categoria, on\_delete=models.CASCADE)  id\_proveedor = models.ForeignKey(Proveedor, on\_delete=models.CASCADE)  id\_sucursal = models.ForeignKey(Sucursal, on\_delete=models.CASCADE)  fecha\_entrada = models.DateField(auto\_now\_add=True)  **class** Meta:  db\_table = **'producto'**  **def** \_\_str\_\_(self):  **return** self.nombre\_producto | **class** **MovimientoEmpleado**(**models.Model**):  **TIPO\_MOVIMIENTO\_CHOICES** = [  (**'agregar'**, **'Agregar'**),  (**'restar'**, **'Restar'**),  ]  empleado = models.ForeignKey('Profile', on\_delete=models.CASCADE)  producto = models.ForeignKey('Producto', on\_delete=models.CASCADE)  cantidad = models.IntegerField()  tipo\_movimiento = models.CharField(max\_length=10, choices=**TIPO\_MOVIMIENTO\_CHOICES**)  fecha = models.DateTimeField(default=timezone.now)  **def** \_\_str\_\_(self):  **return** f"{self.empleado.user.username} {self.tipo\_movimiento} {self.cantidad} de {self.producto.nombre\_producto} el {self.fecha}"    **class** Meta:  permissions = [  (**"view\_movimiento\_empleado\_history"**, **"Puede ver el historial de movimientos de empleados"**)  ] |

* Código extraído del archivo **app/models.py**