Logotipo

Descripción generada automáticamente





Contenido

[**1.**](#_heading=h.gjdgxs) **Introducción** 3

[**2.**](#_heading=h.30j0zll) **Propósito** 3

[**3.**](#_heading=h.1fob9te) **Alcance** 3

[**4.**](#_heading=h.3znysh7) **Equipo de Trabajo-Actores del Desarrollo** 4

[**5.**](#_heading=h.2et92p0) **Recomendaciones de conformidad con esta práctica.** 4

[Para asegurar que la arquitectura del sistema cumpla con los requisitos y estándares establecidos, se recomiendan las siguientes prácticas: 4](#_heading=h.tyjcwt)

[1. Implementar revisiones de arquitectura cada xxxx para asegurar que se ajusten a los cambios de requisitos. 4](#_heading=h.3dy6vkm)

[2. Utilizar patrones de diseño desacoplados como MVC para modularidad y escalabilidad o por capas. 4](#_heading=h.1t3h5sf)

[3. Adoptar metodologías ágiles como Scrum para la entrega continua de mejoras y funcionalidades. 4](#_heading=h.4d34og8)

[**6.**](#_heading=h.2s8eyo1) **Referencias y estándares aplicables a este documento:** 5

[**7.**](#_heading=h.17dp8vu) **Definiciones, acrónimos y abreviaciones.** 5

[**8.**](#_heading=h.3rdcrjn) **DESCRIPCIÓN DE ARQUITECTURA:** La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas. La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas. 6

[8.1 VISTAS: 6](#_heading=h.26in1rg)

[8.2 TIPOS DE VISTAS: 6](#_heading=h.lnxbz9)

[8.3 Framework Conceptual 7](#_heading=h.35nkun2)

[8.3.1 Descripción de la arquitectura en contexto 7](#_heading=h.1ksv4uv)

[8.3.1 Actores Usuarios y sus roles: 8](#_heading=h.44sinio)

[8.3.2 Actividades de arquitectura en el ciclo de vida 9](#_heading=h.2jxsxqh)

[8.3.3 Descripciones prácticas de arquitectura. 9](#_heading=h.z337ya)

[8.4 Documentación de la arquitectura 10](#_heading=h.3j2qqm3)

[8.5 Selección de puntos de vista de la arquitectura 10](#_heading=h.1y810tw)

[8.6 Vistas de la arquitectura 11](#_heading=h.4i7ojhp)

[UML-VISTA DE ESCENARIO 11](#_heading=h.2xcytpi)

[UML-VISTA LÓGICA-DIAGRAMA DE CLASES 12](#_heading=h.1ci93xb)

[UML-VISTA LÓGICA-DIAGRAMA DE BASE DE DATOS 14](#_heading=h.qsh70q)

[UML-VISTA DE PROCESO-DIAGRAMAS DE ACTIVIDAD 16](#_heading=h.1pxezwc)

[UML-VISTA DE DESPLIEGUE 18](#_heading=h.49x2ik5)

[UML-VISTA FÍSICA – DIAGRAMA INFRAESTRUCTURA 19](#_heading=h.2p2csry)

[Usabilidad y Calidad de software- Front-end -Vistas Principales 21](#_heading=h.147n2zr)

[Usabilidad y Calidad de software – Vistas específicas de usuario 22](#_heading=h.3o7alnk)

[8.7 Consistencia en la cantidad de vistas de la arquitectura. 23](#_heading=h.23ckvvd)

[8.7.1 DESCRIPCION DE MODULOS 23](#_heading=h.ihv636)

[8.7.2 DESCRIPCIÓN DE COMPONENTES 23](#_heading=h.32hioqz)

[8.7.3 DESCRIPCIÓN DE CONECTORES 23](#_heading=h.1hmsyys)

[9.1 Acceso a fuentes de información del proyecto 25](#_heading=h.41mghml)

1. **Introducción**

El presente informe tiene como objetivo describir en detalle la arquitectura de un sistema de inventario inteligente diseñado para automatizar la administración de tareas y procesos comunes dentro de un almacén de barrio.

El sistema propuesto busca mejorar el control de las existencias de cada producto de la tienda.

El sistema desarrollado será implementado en el almacén de barrio “La Martita” en la comuna de Maipú. El sistema también proporcionará una interfaz intuitiva y accesible para todos los usuarios, asegurando que las gestiones se realicen de manera eficiente y organizada basada en los estándares de las normas ISO/IEC 25010.

1. **Propósito**

El propósito de este documento es servir como guía para el diseño e implementación de la arquitectura del sistema inteligente de inventario. En el se detallan las decisiones técnicas, los componentes principales y los puntos de integración que permitirán implementar una solución robusta y escalable, capaz de gestionar múltiples comunidades bajo un esquema de administración unificado.

Este informe es esencial para los desarrolladores y stakeholders del proyecto, ya que les proporcionará una visión detallada de la estructura y comportamiento del sistema, así como las relaciones entre las diferentes funcionalidades y componentes. Asimismo, establece las bases para futuras expansiones y mejoras del sistema, considerando factores de disponibilidad, seguridad y usabilidad.

1. **Alcance**

El sistema gestionará las siguientes áreas:

* Inventario del almacén: Crear orden en el inventario, ya que en la actualidad no se lleva un conteo de los productos que entran y salen.
* Métricas de ventas y de los productos: Generación de reportes (productos más o menos vendidos, etc) y alertas automáticas (stock bajo de productos)
* Ventas del almacén: Registro de todas las ventas que realizar los vendedores

Contexto: el almacén de barrio la Martita sufre pérdida de ¼ de sus productos lácteos, ya sean: flanes, yogurt, leches, etc. Esto se traduce en 25 a 35 mil pesos de pérdida al mes. Por otra parte, no se sabe con exactitud que productos se venden más o cuales se venden menos. Tampoco se mantienen un control del monto total de ventas por día, por lo cual no se sabe cuánto dinero genera el negocio con certeza.

Se espera que el sistema ayude a mejorar en un 80% a evitar las pérdidas de estos productos en el primer mes de su implementación. También, se espera que el sistema ayude a saber con exactitud cuales son los productos con mayor y menos rotación dentro del primer mes de funcionamiento y cuánto dinero genera el negocio mensualmente.

1. **Equipo de Trabajo-Actores del Desarrollo**

El desarrollo del sistema requiere la participación de un equipo multidisciplinario con roles claramente definidos para asegurar que el proyecto se lleve a cabo de manera eficiente. A continuación, se presenta una tabla con los principales roles y sus responsabilidades:

| Rol | Responsabilidades |
| --- | --- |
| Cliente-Stakeholder | Marta Millacán |
| Product Owner | Víctor Fuenzalida |
| Desarrolladores Backend | Víctor Fuenzalida |
| Desarrolladores Frontend | Geannelee Araya |
| DBA-Especialista en Base de Datos | Geannelee Araya |
| Tester/QA (cruzado) | Víctor Fuenzalida/Geannelee Araya |

1. **Recomendaciones de conformidad con esta práctica.**

Para asegurar que la arquitectura del sistema cumpla con los requisitos y estándares establecidos, se recomiendan las siguientes prácticas:

1. Implementar revisiones de arquitectura cada módulo para asegurar que se ajusten a los cambios de requisitos.
2. Utilizar patrones de diseño desacoplados como MVC para modularidad y escalabilidad o por capas.
3. Adoptar metodologías ágiles como Scrum para la entrega continua de mejoras y funcionalidades.
4. **Referencias y estándares aplicables a este documento:**

El documento sigue los siguientes estándares de calidad para asegurar la robustez y escalabilidad del sistema:

* **IEEE 830-1998 ST**: Estandarización de requisitos de software.
* **ISO 9126-2001**: Métricas de calidad del software.
* **Modelo 4+1 de Kruchten**: Modelo estructural basado en múltiples vistas de escenarios.
* **ATAM (Architecture Tradeoff Analysis Method)**: Metodología para evaluar y documentar decisiones arquitectónicas

1. **Definiciones, acrónimos y abreviaciones.**

| Término | Definición |
| --- | --- |
| DAS | Documento de Arquitectura de Software |
| MVC | Modelo-Vista-Controlador, patrón de diseño utilizado para separar la lógica de negocio de la interfaz de usuario |
| UML | Unified Modeling Language, lenguaje gráfico para representar la arquitectura del sistema |
| CRUD | Operaciones básicas de Create, Read, Update, Delete para la gestión de datos. |

1. **DESCRIPCIÓN DE ARQUITECTURA:**La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas. La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas.
   1. VISTAS:

Cada vista se representará en un diagrama UML para describir la estructura y comportamiento del sistema.

* 1. TIPOS DE VISTAS:

1. **Vista de Escenarios (Diagrama de Caso de Uso)**: Describe las funcionalidades del sistema y cómo interactúan los diferentes actores con él.
2. **Vista Lógica (Diagrama de Clases)**: Detalla la estructura de datos y las relaciones entre las principales entidades del sistema.
3. **Vista de Procesos (Diagrama de Actividad)**: Muestra el flujo de actividades dentro del sistema y cómo se coordinan los diferentes procesos.
4. **Vista de Despliegue (Diagrama de Componentes)**: Representa la distribución de componentes en la infraestructura física.
5. **Vista Física (Diagrama Topológico)**: Representa la infraestructura de red y la comunicación entre los diferentes servicios del sistema
   1. Framework Conceptual
      1. Descripción de la arquitectura en contexto

El sistema será diseñado y desarrollado utilizando un enfoque ágil con metodologías como Scrum, lo que permitirá la entrega incremental de funcionalidades, la retroalimentación constante y la capacidad de adaptarse a cambios en los requisitos a lo largo del ciclo de vida del proyecto. Esto asegurará que se puedan abordar necesidades emergentes y maximizar el valor del producto para los usuarios.

El diseño del sistema se basará en la separación de vistas, que ofrecerán diferentes perspectivas de la arquitectura para una comprensión integral. A continuación, se describen las principales vistas:

* **Vista de Escenarios**: Representada mediante diagramas de Casos de Uso, que describen las interacciones del usuario con el sistema y los flujos principales.
* **Vista Lógica**: Incluye el Modelo de Clases, que muestra las entidades y relaciones del sistema, así como el Modelo de Datos que detalla la estructura de la base de datos.
* **Vista de Procesos**: Utiliza Diagramas de Actividad para ilustrar los flujos internos del sistema y las interacciones entre los distintos módulos.
* **Vista de Despliegue**: Se implementará a través de Diagramas de Componentes o Modelos de Capas para reflejar la distribución de la lógica de negocio, la interfaz de usuario y la capa de acceso a datos.
* **Vista Física**: Utiliza Diagramas Topológicos o de Servicios para mostrar la distribución del sistema en el entorno físico, como servidores, redes y otros componentes de infraestructura.

Cada una de estas vistas está interconectada para garantizar la coherencia y la alineación con los requisitos de negocio. No hay ninguna vista separada de una implementación específica, sino que todas contribuyen a una visión holística de la arquitectura del sistema descrita en este documento.

Estas vistas se han creado utilizando el Lenguaje de Modelado Unificado (UML) en su versión 2.0 y han sido desarrolladas empleando herramientas de modelado estándar como **StarUML**, **Creately**, **Visual Paradigm** y en front-end **Figma, Bootstrap** y **Sitejet** para facilitar la creación de diagramas visuales que respalden la documentación arquitectónica.

El estilo arquitectónico se centra en una combinación de patrones de diseño orientados a servicios xxxx (ejemplo: SOA) y arquitectura por capas, en donde cada módulo del sistema se corresponde con un conjunto específico de casos de uso que resuelven los procesos definidos para la operación.

Estos procesos corresponden a:

* **Gestión de Usuarios**: Facilita la creación y administración de los diferentes roles de usuarios dentro del sistema.
* **Generación de Informes y Reportes**: Permite generar informes sobre la rotación de los productos y las ventas.
* **Administración de ventas**: Permite facilitar el proceso de venta y mantener un registro de las ventas del almacén. También facilitará el cuadre de caja e inventario
* **Gestión de Inventario:** Permite modificar o eliminar las características de un producto.
  + 1. Actores Usuarios y sus roles:

Este documento representa la identificación de Actores/Usuarios Stakeholders y sus roles a partir de la interpretación de los casos de uso del Negocio asociados.

| ACTOR | ROL / CARGO | FUNCION |
| --- | --- | --- |
| usuario | Administrador | Encargado de administrar los usuarios que hacen uso del sistema, administrar las existencias de productos en el sistema, trabajar con la información que entrega el sistema para la toma de decisiones. |
| usuario | Vendedor | Genera registro de las ventas que realiza |
| usuario | Bodeguero | Ingresa productos y existencias al sistema |

8.3.2 Actividades de arquitectura en el ciclo de vida

El desarrollo de la arquitectura del sistema de Inventario Inteligente, se abordó utilizando un ciclo de vida **iterativo incremental** basado en **Scrum**. Esta metodología fue seleccionada porque permite realizar entregas parciales y frecuentes de software funcional, facilitando la validación temprana de los requisitos por parte de los stakeholders y el product owner con la adaptación a cambios en el entorno de negocio.

El ciclo de vida se organizó en **sprints** de 2 a 3 semanas, con cada iteración centrada en la construcción y mejora de un módulo específico del sistema (por ejemplo, el módulo de gestión de gastos comunes o el módulo de reservas). Al final de cada sprint, se realiza una revisión detallada de los resultados para asegurar la alineación con las expectativas del cliente y se priorizan las tareas para el siguiente sprint.

8.3.3 Descripciones prácticas de arquitectura.

Durante el proceso de desarrollo, la arquitectura del sistema fue escalada desde un **escenario principal de caso de uso** hasta un diseño modular a través de los siguientes pasos:

1. **Análisis de Requerimientos**: Se analizaron los requisitos iniciales del sistema para identificar los casos de uso principales, tales como "Administrar inventario", "Generar reportes",”Realizar venta” y “Actualizar stock”.
2. **Definición de la Vista Lógica**: A partir del escenario principal, se identificaron las entidades y relaciones claves en un diagrama de clases UML, representando el núcleo del sistema.
3. **Modelo de Datos**: Se desarrolló un diagrama de base de datos para definir la estructura de la información, manteniendo la coherencia con la vista lógica.
4. **Vista de Procesos**: Se modelaron los flujos de las actividades y transacciones principales del sistema mediante diagramas de actividad.
5. **Despliegue**: Se definió la infraestructura física y lógica para alojar el sistema en la nube, detallando los servicios necesarios, como servidores de aplicaciones y la base de datos.
6. **Definición de Planes de pruebas y técnicas de pruebas testing**.
   1. Documentación de la arquitectura

* **ISO/IEC 25010**: Se utilizó para evaluar la calidad del sistema en términos de funcionalidad, seguridad y eficiencia.
* **IEEE 1471-2000**: Proporcionó la guía para la representación de vistas arquitectónicas y la identificación de stakeholders.
* **Modelo 4+1 de Kruchten**: Estructura la arquitectura en vistas lógicas, de desarrollo, de procesos y física, facilitando la visualización desde múltiples perspectivas.
* **Planes de prueba**: (Declarar los tipos de pruebas a realizar)
  1. Selección de puntos de vista de la arquitectura

| VISTAS | UML | Cantidad |
| --- | --- | --- |
| Escenario Principal | Diagrama de Caso de uso | 1 (2-3) |
| Vista Lógica | Diagrama de Clases  MER | 1 |
| Vista de Proceso | Diagrama de Actividad:  Actividad Administrador.  Actividad Vendedor.  Actividad Bodeguero | 1 (2-3) |
| Vista Física | Diagrama Topológico de Servicios | 1 |
| Vista de Despliegue | Diagrama de componentes  Diagrama de Capas (MVC) | 1  1 |

* 1. Vistas de la arquitectura

| UML-VISTA DE ESCENARIO |
| --- |
| Diagrama de Caso de uso extendido /Subproyecto [Inventario Inteligente] |
|  |

**Gestión de Usuarios**: Facilita la creación y administración de los diferentes roles de usuarios dentro del sistema.

**Generación de Informes y Reportes**: El sistema permitirá al administrador generar informes sobre la rotación de los productos y las ventas.

**Alertas Automáticas**: Notificaciones a todos los usuarios sobre las existencias de productos en el inventario: productos a caducar y productos con bajas existencias de alta rotación.

**Administración de ventas**: Permite facilitar el proceso de venta y mantener un registro de las ventas del almacén. También facilitará el cuadre de caja e inventario

**Búsqueda de producto:** Permite al usuario buscar un producto por su nombre o código en el sistema.

**Ingreso de productos:** Controla el ingreso de productos al inventario, a través de una clasificación predeterminada.

**Gestión de Inventario:** Permite al administrador y al bodeguero modificar o eliminar las características de un producto.

| UML-VISTA LÓGICA-DIAGRAMA DE CLASES |
| --- |
| [Diagrama de Clases] |
|  |

El **Diagrama de Clases** para el sistema de Inventario inteligente muestra las clases involucradas, sus atributos, métodos y las relaciones entre ellas. Este diagrama ayuda a modelar la estructura y el comportamiento del sistema de una manera orientada a objetos.

**2. Elementos Principales**

* **Clases**:
  + **Usuario**:
    - **Atributos**: numrun, p\_nombre, s\_nombre, p\_apellido, s\_apellido, fechanacimiento, sexo, nacionalidad, numero\_telefonico, direccion, email, password\_usuario, token, fyh creacion, fyh\_actualizacion, id\_rol
    - **Métodos**: iniciarSesion(), cerrarSesion()
  + **Rol:**
    - **Atributos**: id\_rol, rol, fyh creacion, fyh\_actualizacion.
    - **Métodos**: asignarRol(), crearRol(), modificarRol(), EliminarRol()
  + **Venta**
    - **Atributos**: id\_venta, fyhventa, tipopago, cantidad, totalventa, id\_usuario, id\_producto, id\_reporte.
    - **Métodos**: realizarVenta()
  + **Producto**:
    - **Atributos**: id\_producto, nombreproducto, marca, descripción, imagen, código, fechavencimiento, existencias, precio\_venta, precio\_compra, fyh creacion, fyh\_actualizacion, id\_venta, id\_reporte
    - **Métodos**: ingresarProducto(), eliminarProducto(), modificarProducto()
  + **Categoria**:
    - **Atributos**: id\_categoria, categoria, fyh creacion, fyh\_actualizacion, ubicacion, id\_producto
    - **Métodos**: crearCategoria(), modificarCategoria(), eliminarCategoria ()
  + **Reporte**:
    - **Atributos**: id\_reporte, nombrereporte, fechareporte, tiporeporte,id\_venta, id\_producto
    - **Métodos**: generarReporte(), generarAlertas()
  + **Proveedor**:
    - **Atributos**: id\_proveedor, nombreproveedor, numero\_telefonico, direccion, correo, id\_producto
    - **Métodos**: crearProveedor(), modificarProveedor(), eliminarProveedor()

**3. Relaciones:**

* **Usuario** tiene una relación de **asociación** con **ventas** (múltiples ventas pueden estar asociadas a un vendedor).
* **Rol** tiene una relación de **asociación** con **Usuario** (múltiples usuarios pueden tener un mismo rol).
* **Ventas** tiene una relación de **asociación** con **reporte y usuario.** Y una relación de **asociación** con producto(una venta se constituye de varios productos**)**
* **Producto** tiene una relación de **asociación** con **categoría, proveedor y venta**
* **Proveedor**  tiene una relación de **asociación** con **Producto** (múltiples productos pueden asociarse con un mismo proveedor)
* **Categoría**  tiene una relación de **composición** con Producto (múltiples productos pueden constituir una categoría)
* **Reportes** tiene una relación de **composición** con **ventas y productos** (está compuesto por varios registros de ventas y/o productos)

| UML-VISTA LÓGICA-DIAGRAMA DE BASE DE DATOS |
| --- |
| [Diagrama de BD - MER] |
|  |

El **Modelo Entidad-Relación (MER)** para el sistema de Inventario inteligente representa las entidades involucradas, sus atributos y las relaciones que existen entre ellas. Este modelo permite visualizar cómo se organiza la información relacionada con la gestión del inventario, las ventas y los reportes.

**2. Elementos Principales**

* **Entidades**:
  + **Usuario**:
    - **Atributos**: : numrun (PK), p\_nombre, s\_nombre, p\_apellido, s\_apellido, fechanacimiento, sexo, nacionalidad, numero\_telefonico, direccion, email, password\_usuario, token, fyh creacion, fyh\_actualizacion, id\_rol (FK).
  + **Rol:**
    - **Atributos**: id\_rol (PK), rol, fyh creacion, fyh\_actualizacion.
  + **Venta**
    - **Atributos**: id\_venta (PK), fyhventa, tipopago, cantidad, totalventa, id\_usuario (FK), id\_producto (FK), id\_reporte (FK).
  + **Producto**:
    - **Atributos**: id\_producto (PK), nombreproducto, marca, descripción, imagen, código, fechavencimiento, existencias, precio\_venta, precio\_compra, fyh creacion, fyh\_actualizacion, id\_venta, id\_proveedor (FK), idcategoria (FK)
  + **Categoria**:
    - **Atributos**: id\_categoria, categoria, fyh creacion, fyh\_actualizacion.
  + **Reporte**:
    - **Atributos**: id\_reporte, nombrereporte, fechareporte, tiporeporte, id\_venta (FK).
  + **Proveedor**:
    - **Atributos**: id\_proveedor, nombreproveedor, numero\_telefonico, direccion, correo, id\_producto (FK)
* **Relaciones**:
  + **Usuario** está asociado a un único **rol**
  + **Rol** un **rol** puede ser asignado a **múltiples usuarios**
  + **Venta** **múltiples ventas** pueden estar asociadas a **un vendedor**, pero varios vendedores no pueden estar asociados a una sola venta.
  + **Producto** un **producto** puede pertenecer a **múltiples ventas** y a **múltiples proveedores**. Por el contrario, **un producto** puede pertenecer a una sola **categoría** y a una sola **ubicación**
  + **Categoria** una **categoría** puede tener **múltiples productos**
  + **Reporte** un **reporte** puede contener información de **múltiples ventas y múltiples productos**
  + **Proveedor** un **proveedor** puede tener **múltiples productos**

El **diagrama de actividad** representa el flujo de actividades que se llevan a cabo en un proceso específico. En este caso, los procesos de gestión de usuarios, gestión de inventario, venta, y generación de reportes. Este diagrama es útil para visualizar los pasos secuenciales y las decisiones que los usuarios debe tomar al utilizar el sistema.

| UML-VISTA DE PROCESO-DIAGRAMAS DE ACTIVIDAD |
| --- |
| Diagramas de Actividad: administrador |
|  |

**Componentes**:

Diagrama Administrador

El administrador puede llevar a cabo tres grandes procesos: Gestión de usuarios, Gestión de Inventario y Generación de reportes.

Gestión de Usuarios:

1. El administrador inicia la aplicación.

2. Ingresa su usuario y contraseña en la interfaz de login.

3. El sistema valida los datos:

- Si los datos son inválidos, se solicita al administrador volver a ingresarlos.

4. El administrador selecciona la opción "Administrar Usuarios".

5. Busca un usuario:

- Si el usuario no existe, el sistema indica que no hay resultados.

- Si el usuario existe, se despliega un menú de opciones.

6. Opciones para el usuario encontrado:

- Eliminar al usuario.

- Modificar datos del usuario.

- Consultar el registro de ventas del usuario.

7. Se guardan los cambios realizados en el sistema.

8. Si el administrador desea crear un nuevo usuario, selecciona la opción correspondiente.

Administración de Existencias o gestión de inventario

1. El administrador elige la opción de "Administrar Existencias".

2. Busca un producto en el inventario:

- Si el producto no existe, puede agregar uno nuevo.

- Si el producto existe, se muestra en la interfaz.

3. Si el producto tiene existencias:

- El administrador puede modificar la cantidad en el inventario.

- Se guardan los cambios en el sistema.

4. Si el administrador desea agregar otro producto, puede hacerlo; de lo contrario, se finaliza esta tarea.

Administración de Reportes

1. El administrador selecciona "Administrar Reportes".

2. Se despliega la interfaz de parámetros, donde selecciona los criterios necesarios para el reporte.

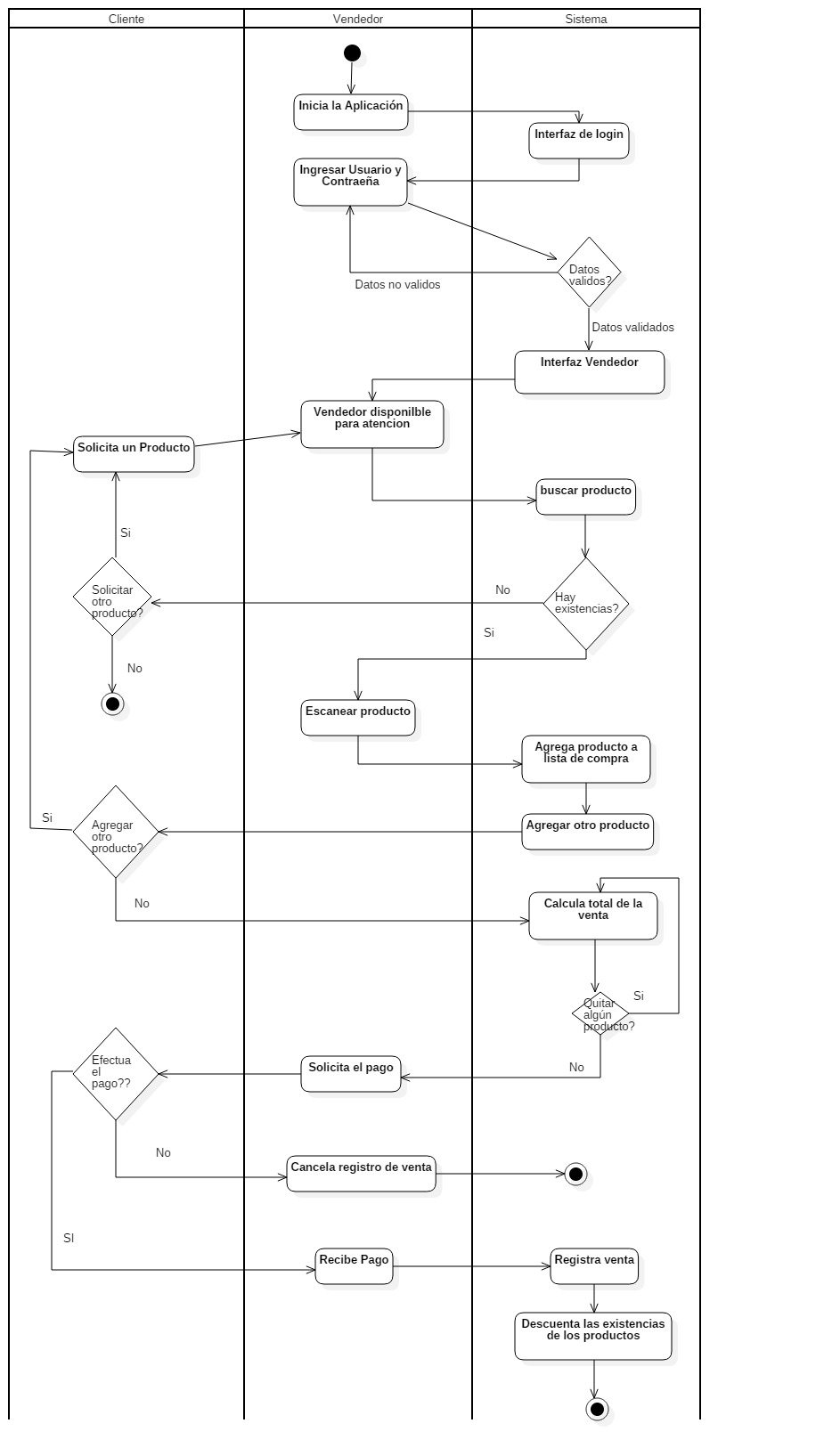
3. El sistema genera e imprime el reporte según los parámetros seleccionados.

4. El administrador decide si desea realizar otra operación:

- Si sí, vuelve al inicio de las opciones de administración.

- Si no, finaliza la actividad.

| UML-VISTA DE PROCESO-DIAGRAMAS DE ACTIVIDAD |
| --- |
| Diagramas de Actividad: administrador |



**Componentes**:

Diagrama Vendedor

El Vendedor puede llevar a cabo un proceso: Realizar venta. Este se divide en 3 sub procesos descritos a continuación:

Proceso de Venta

1. El vendedor agrega un producto a la lista de compra.

2. presiona “agregar otro producto” si el cliente desea más productos:

- Si sí, vuelve al paso anterior y agrega el siguiente producto.

- Si no, continúa con el proceso.

3. El sistema calcula el “total de la venta”.

4. El cliente revisa los productos en la lista y decide si quiere “quitar producto”:

- Si sí, el sistema vuelve a calcular el total.

- Si no, continúa con el proceso.

Solicitud de Pago

1. El vendedor solicita al cliente que efectúe el pago (efectivo o tarjeta).

2. El cliente “efectúa el pago”:

- Si no realiza el pago, el vendedor “cancela” el registro de la venta.

- Si realiza el pago, el vendedor da click en “aceptar” y pasa al siguiente paso.

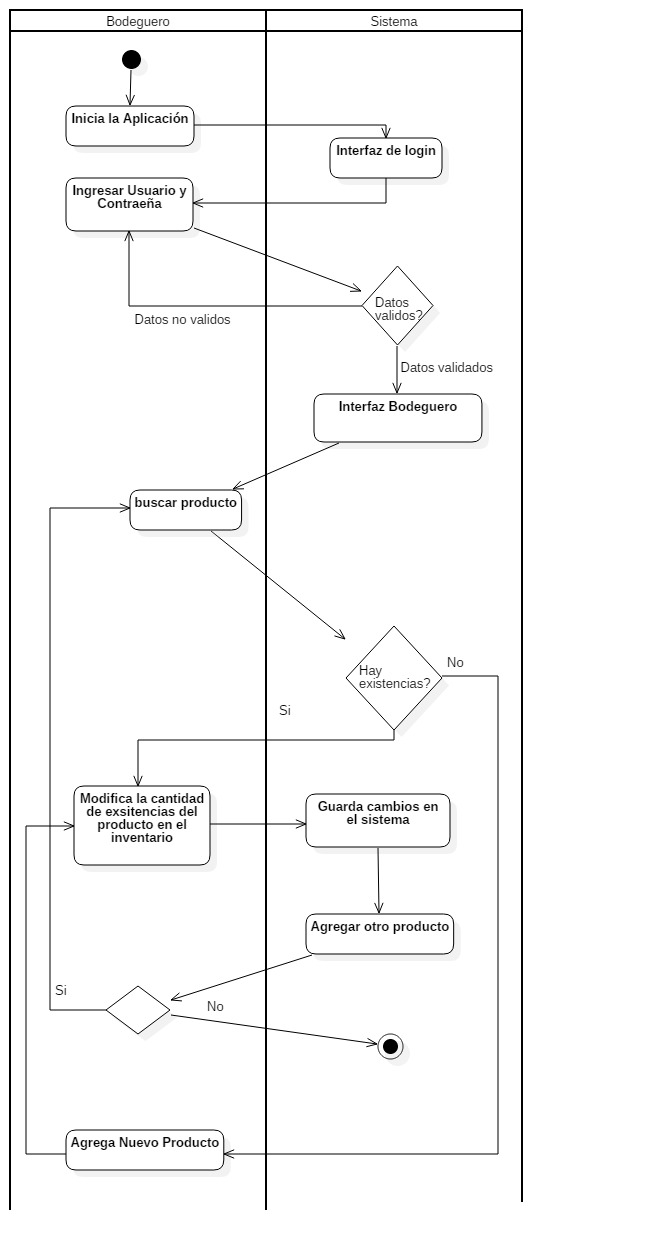
Registro y Finalización de la Venta

1. El sistema recibe el pago.

2. Registra la venta en el sistema.

3. Descuenta las existencias de los productos vendidos del inventario.

| UML-VISTA DE PROCESO-DIAGRAMAS DE ACTIVIDAD |
| --- |
| Diagramas de Actividad: bodeguero |



**Componentes**:

Diagrama Bodeguero

El Vendedor puede llevar a cabo un proceso: Gestión de inventario. Este se divide en 4 sub procesos descritos a continuación:

1. Buscar Producto:

- Una vez dentro de la interfaz del bodeguero, este busca el producto que desea consultar o modificar.

2. Verificación de Existencias:

- El sistema verifica si hay existencias del producto:

- Si hay existencias:

- El bodeguero puede “modificar la cantidad de existencias” del producto en el inventario.

- Si no hay existencias:

- El sistema guarda los cambios realizados hasta el momento.

3. Agregar otro Producto:

- Después de guardar los cambios, el sistema pregunta si el bodeguero desea agregar otro producto:

- Si desea agregar otro producto: el flujo vuelve al paso de “Buscar Producto”.

- Si no desea agregar otro producto: el proceso termina.

4. “Agregar Nuevo Producto”:

- Si se ha optado por no agregar otro producto existente, el bodeguero tiene la opción de “agregar un nuevo producto” en el inventario.

| UML-VISTA DE DESPLIEGUE |
| --- |
| Sistema de Gestión de inventario |
|  |

**Servidor de Aplicaciones**:

* **Aplicación Web de Inventario Inteligente**: Proporciona una interfaz amigable para todos los actores del sistema (vendedores, administradores y bodegueros). Permite la gestión de inventario, gestión de ventas, y generación de reportes.

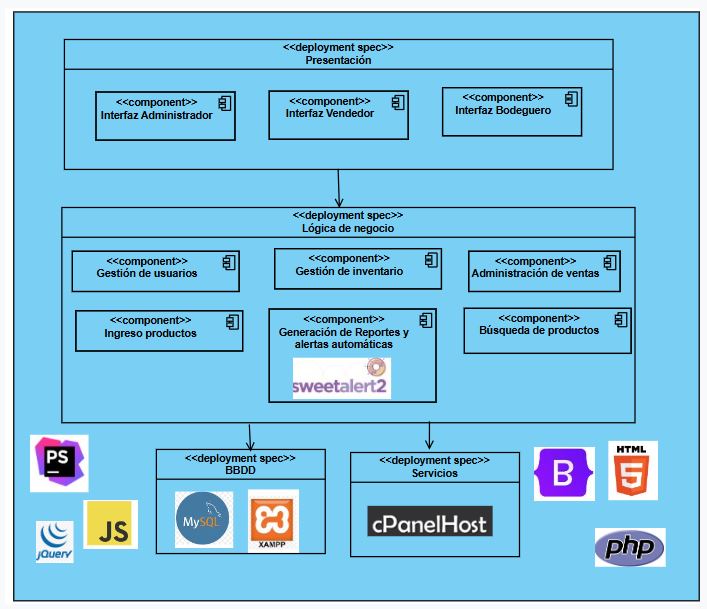
**Servidor de Base de Datos**:

* **Base de Datos del Almacén**: Almacena toda la información relevante, incluyendo detalles de los usuarios, ventas y productos. Este servidor es fundamental para que la aplicación funcione correctamente, ya que procesa las consultas y actualizaciones de datos.

**Servidor de reportes**:

* **Sistema de reportes y alertas**: Se encarga de generar reportes y enviar alertas automáticas a los usuarios sobre las ventas y las existencias y vencimiento de los productos. Esto ayuda a mantener un inventario acorde a la demanda del negocio.

| UML-VISTA DE DESPLIEGUE – DIAGRAMA DE CAPAS (MVC) |
| --- |
| Sistema Gestión de Inventario Inteligente |



| UML-VISTA FÍSICA – DIAGRAMA INFRAESTRUCTURA |
| --- |
| Sistema Gestión de Inventario Inteligente |
|  |

**Centro de Datos Principal**: Es el corazón del sistema donde se aloja la infraestructura central. Incluye servidores dedicados para la aplicación web, base de datos, notificaciones y backups. Todos estos elementos están protegidos por un firewall para garantizar la seguridad de los datos.

**Red del Almacén**: Esta red incluye los dispositivos de acceso dentro del almacén, como las estaciones de trabajo del administrador, del vendedor y el bodeguero. La red local se conecta al Centro de Datos a través del Router Principal.

**Conexiones Externas**: El acceso a Internet está proporcionado por un Proveedor de Servicios de Internet y se distribuye en todo el almacén.

| Usabilidad y Calidad de software- Front-end -Vistas Principales |
| --- |
| Vistas principales del sistema – Vista Home Principal de entrada |
|  |

La **Vista Front-End** del sistema de gestión de condominios ha sido diseñada para ofrecer una experiencia de usuario intuitiva, moderna y fácil de navegar. La interfaz está dividida en diferentes secciones con menús y paneles dinámicos que se adaptan a cada tipo de usuario, brindando acceso a funcionalidades específicas según su rol (Administrador, vendedor y bodeguero). Se presenta un diseño limpio y ordenado que resalta las opciones más importantes, permitiendo a los usuarios interactuar con el sistema de manera eficiente

**Estilo Moderno y Profesional**: Colores sobrios y elementos gráficos que se destacan sin distraer, utilizando una paleta de colores asociada a la identidad visual del almacén.

**Interactividad y Usabilidad**: Uso de gráficos animados, transiciones suaves y elementos interactivos que facilitan la navegación.

**Iconografía Clara**: Uso de iconos intuitivos para representar cada módulo y función, ayudando a los usuarios a ubicar rápidamente la opción deseada

| Usabilidad y Calidad de software – Vistas específicas de usuario |
| --- |
| Vistas principales del sistema – Vista Principal de Usuario a escritorio de trabajo principal de su componente de sistema |
|  |

Cada tipo de usuario tiene un escritorio personalizado con acceso a las funcionalidades correspondientes a su rol:

* **Administrador**: Control total de los usuarios, gestión de inventario y reportes.
* **Vendedor**: Registro de ventas.
* **Bodeguero**: Gestión de inventario.
  1. Consistencia en la cantidad de vistas de la arquitectura.

Se garantiza que cada vista esté alineada con los requisitos funcionales y no funcionales del sistema, evitando redundancias y manteniendo la coherencia en toda la arquitectura

* + 1. DESCRIPCIÓN DE MÓDULOS

| Nombre del módulo | Descripción | Componentes incluidos |
| --- | --- | --- |
| Gestión de Usuarios | Administración de cuentas y roles | Módulo de registro, Módulo de autenticación, Módulo de perfiles |
| Gestión de Inventario | Permite administrar y establecer un orden de las existencias de los productos | Módulo de administración productos |
| Registro Ventas | Controla y registra las ventas realizadas en el día | Módulo de venta |
| Generación de reportes y control de alarmas automáticas | Permite generar reportes con estadísticas de las ventas y los productos: productos más vendidos o menos vendidos, productos por vencer, etc. | Módulo de generación de reportes y módulo de configuración de alertas automáticas |

* + 1. DESCRIPCIÓN DE COMPONENTES

| Nombre del Componente | Descripción | Componentes Relaciones |
| --- | --- | --- |
| Gestor de Usuarios | Maneja la creación, modificación y eliminación de usuarios | Módulo de Gestión de Usuarios |
| Calculadora de Gastos | Realizar cálculos de gastos basados en tarifas y consumos | Módulo de Gastos, módulo de Reportes |
| Motor de Notificaciones | Envía notificaciones por correo y mensajes emergentes | Módulo de Reservas, Módulo de Gastos |
|  |  |  |

* + 1. DESCRIPCIÓN DE CONECTORES

| Nombre del Conector | Tipo | Propósito |
| --- | --- | --- |
| Rest API | Comunicación | Transferencia de datos entre módulos |
| Socket.io | Comunicación | Actualizaciones en tiempo real |

* + - 1. **Arquitectura lógica**
      2. Performances

El sistema está diseñado para manejar una alta concurrencia, soportando más de 1000 usuarios simultáneamente sin comprometer el rendimiento. Para lograr tiempos de respuesta inferiores a 2 segundos por transacción, se implementarán las siguientes estrategias:

* **Escalabilidad Horizontal**: Se utilizarán contenedores Docker para desplegar microservicios en múltiples instancias en la infraestructura de AWS, permitiendo la adición de recursos según la demanda.
* **Optimización de Consultas**: Se aplicarán técnicas de optimización en las consultas a la base de datos PostgreSQL, incluyendo índices y consultas asíncronas para reducir los tiempos de respuesta.
  + - 1. Criterios de Calidad

El sistema debe cumplir con los siguientes criterios de calidad:

* **Disponibilidad 24/7**: La arquitectura se diseñará para asegurar un tiempo de actividad continuó, implementando redundancia y recuperación ante desastres en la infraestructura de AWS.
* **Respuesta Rápida**: Todas las transacciones deberán completarse en menos de 5 segundos, lo que se logrará mediante el uso de técnicas de caché y un diseño eficiente de la API.
* **Alta Seguridad**: Se implementará cifrado AES-256 para proteger datos sensibles, así como autenticación y autorización robustas para garantizar la integridad de la información y la privacidad del usuario.
  + - 1. Detalles de la implementación

La implementación del sistema se basará en una arquitectura de microservicios, utilizando las siguientes tecnologías:

* **Backend**: Node.js con Express para la creación de APIs RESTful.
* **Frontend**: adminLTE junto con para la creación de una interfaz de usuario moderna y responsiva.
* **Base de Datos**: mySQL para la gestión eficiente de datos estructurados.
* **Infraestructura**: AWS utilizando contenedores Docker para una gestión flexible y escalable de los microservicios.
  + - 1. Lenguajes y plataformas

Las tecnologías y versiones específicas que se utilizarán en el desarrollo del sistema incluyen:

* **Xampp** 3.3.0
* **Sweatalert** 2
* **MySql**
* **ADMINLTE** 3.2.0
* **BOOTSTRAP** 4
* **PHP** 8
* **PHPSTORM**2.1
  1. Acceso a fuentes de información del proyecto

Para acceder a la carpeta de evidencias acumuladas y otros documentos relevantes del proyecto, puede utilizar el siguiente enlace:  
  
Insertar link: Carpeta de Evidencia del Proyecto  
  
Esta carpeta incluye documentación técnica, diagramas, pruebas de rendimiento y cualquier otra evidencia relacionada con el desarrollo y la implementación del sistema