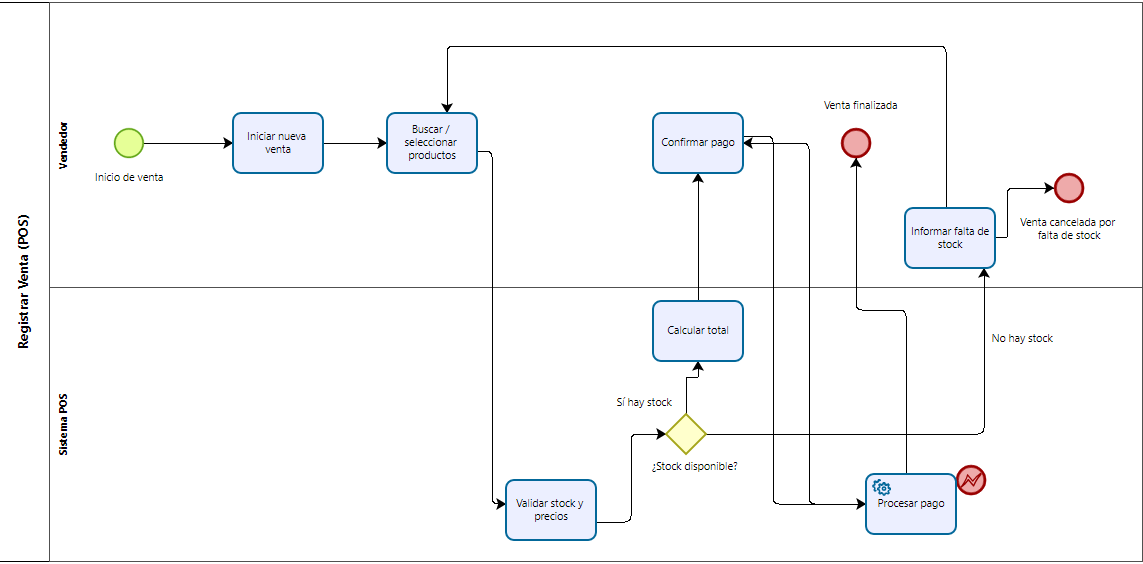
# Tecnologías para implementar

|  |  |  |
| --- | --- | --- |
| Componente | Tecnología | Descripción / Justificación |
| Lenguaje principal | Python (Django Framework) | Para la creación de APIs REST para la conexión entre frontend y backend. |
| Frontend | HTML5, CSS, JavaScript (Bootstrap 5) | Facilitan una interfaz moderna, adaptable y fácil de mantener. Bootstrap mejora la experiencia visual y la compatibilidad multiplataforma. |
| Base de Datos | MySQL | Gestor de base de datos relacional robusto, de alto rendimiento y gratuito, compatible con Django. |
| Servidor Web |  |  |
| Inteligencia de Negocios (BI) | Power BI - Python (pandas, matplotlib) | Permite generar reportes gráficos, análisis de ventas y paneles de control interactivos para la toma de decisiones. |
| Machine Learning | Scikit-learn - Pandas | Para la creación de modelos predictivos que detecten patrones de venta por mes y generan alertas automáticas. |
| Control de versiones | GitHub | Se utilizará para la colaboración y control de cambios entre las versiones del proyecto. |

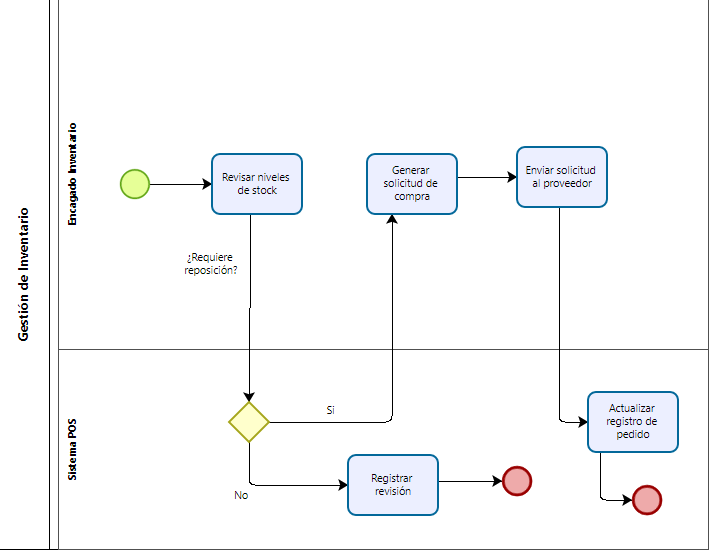
# Arquitectura por implementar.

### Confecciona y describe los BPMN de los procesos de negocio (Marcelo)

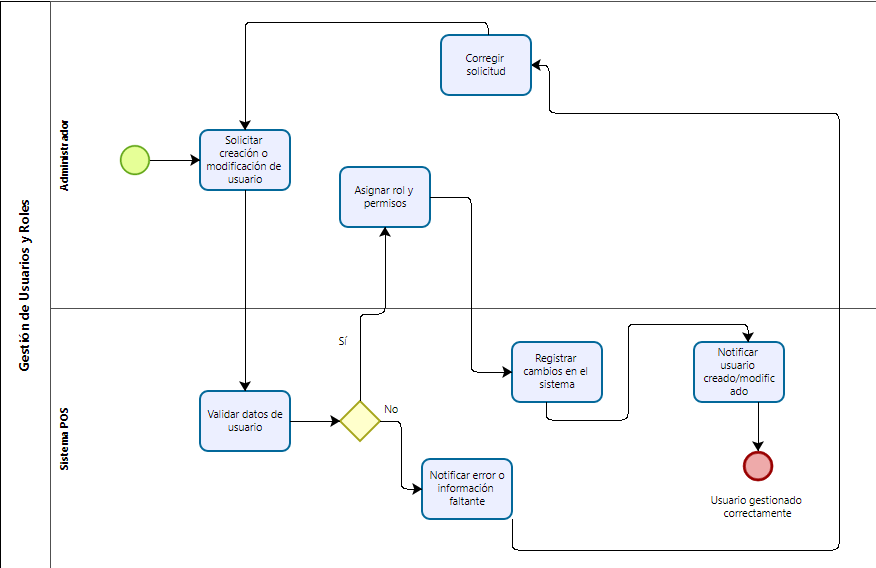
* Registro de ventas



* Gestión de Inventario



* Gestión de Usuarios y Roles



### Confecciona y describe los diagramas UML de los casos de uso y diagrama de componentes (Joaquín)

**1. Diagrama de Casos de Uso UML**

**Actores y sus casos de uso:**

**Administrador**

* Iniciar sesión
* Gestionar usuarios (crear, editar, eliminar)
* Gestionar productos (agregar, editar, eliminar)
* Consultar reportes de ventas e inventario

**Cajero**

* Iniciar sesión
* Registrar venta
* Consultar stock
* Actualizar inventario (cuando se vende o llega nueva mercancía)

**Sistema de alertas**

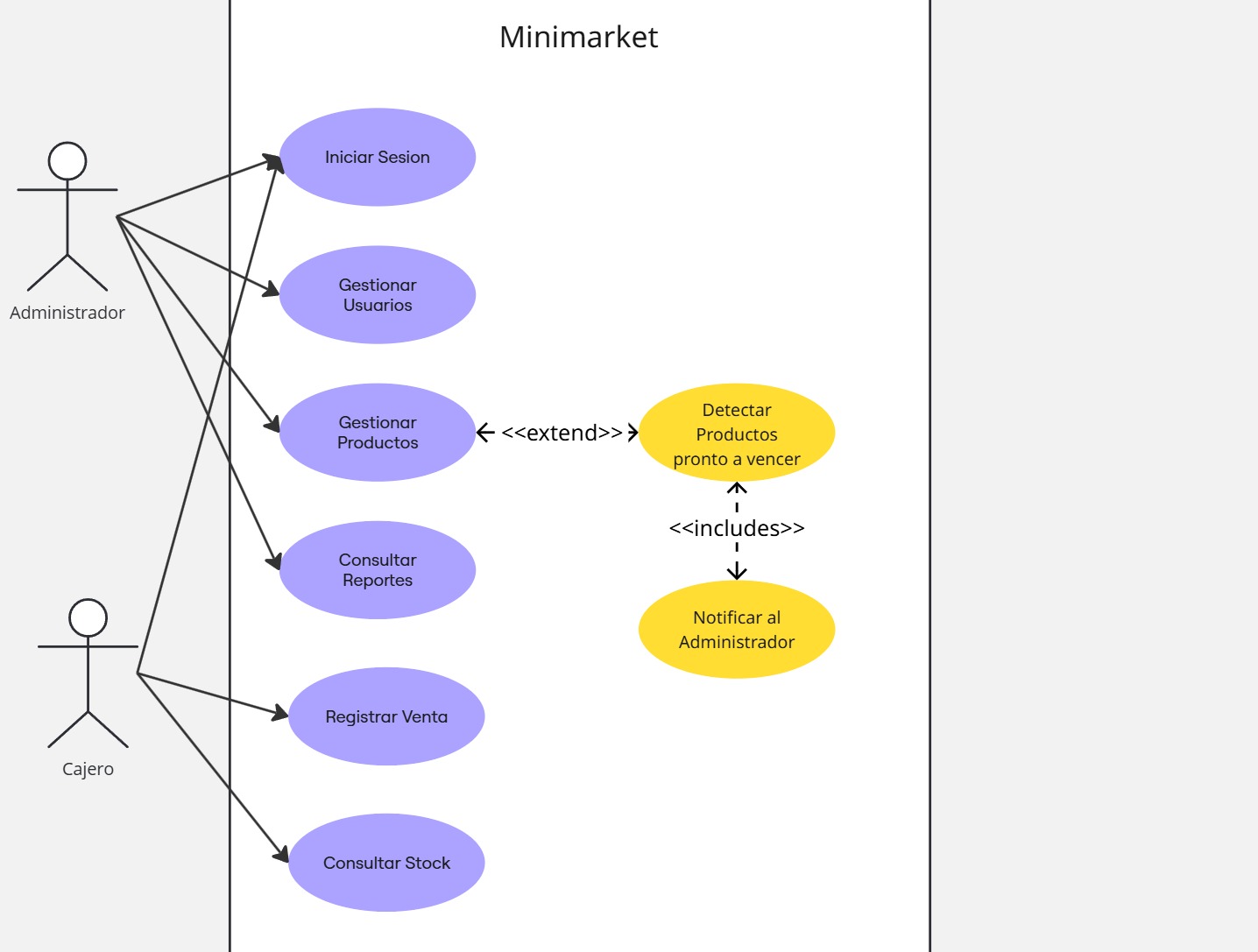
* Detectar productos próximos a vencer
* Notificar al administrador

**Descripción breve del diagrama:**

El **Administrador** tiene control total del sistema: gestiona usuarios, productos y revisa reportes.  
 El **Cajero** realiza las operaciones diarias de venta e inventario.  
 El **Sistema de alertas** funciona de forma automática, monitoreando las fechas de vencimiento en la base de datos y enviando alertas cuando sea necesario.

**Relaciones clave:**

* “Iniciar sesión” es un caso común (usado por ambos actores).
* “Notificar al administrador” depende de “Detectar productos próximos a vencer” (relación *include*).



### **2. Diagrama de Componentes UML**

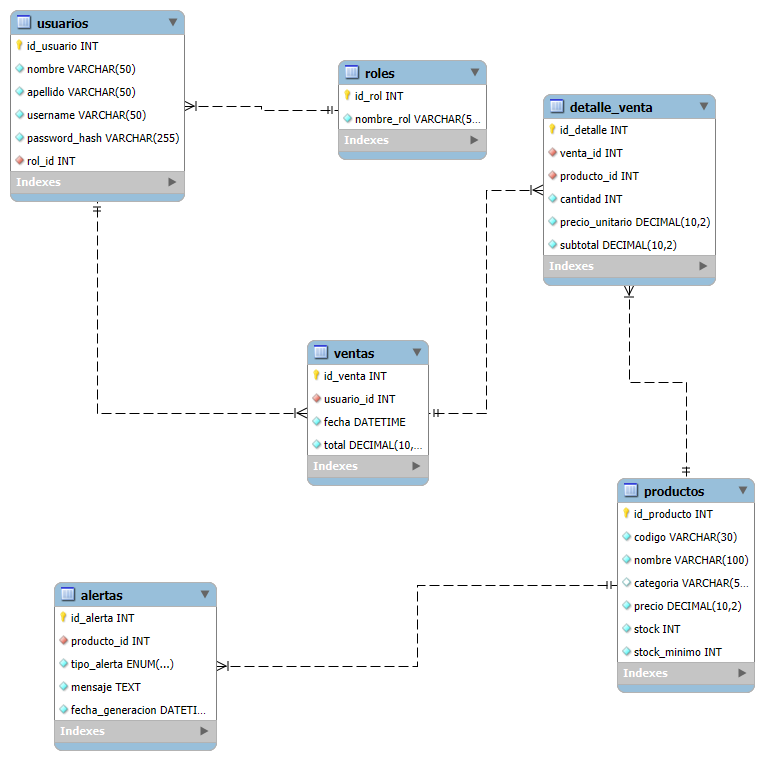
**Componentes principales del sistema RapidMarket:**

1. **Interfaz Web (Frontend)**
   1. HTML, CSS, JavaScript
   2. Pantallas: login, gestión de productos, ventas, reportes
2. **Lógica de Negocio (Backend Django)**
   1. Módulo de autenticación
   2. Módulo de inventario
   3. Módulo de ventas
   4. Módulo de alertas
   5. Módulo de reportes
3. **Base de Datos (MySQL)**
   1. Tablas: usuarios, productos, ventas, alertas
4. **Servidor de Alertas Automáticas**
   1. Proceso en segundo plano que revisa vencimientos y genera notificaciones
5. **Administrador / Cajero (Usuarios externos)**
   1. Acceden vía navegador al sistema

**Descripción del diagrama:**

El **Frontend** se comunica con el **Backend Django** mediante solicitudes HTTP.  
 El **Backend** procesa las peticiones, aplica la lógica de negocio y accede a la **Base de Datos MySQL** para leer o modificar información.  
 El **Servidor de alertas automáticas** corre de manera independiente (puede ser un script o tarea programada en Django) y envía notificaciones al **Administrador**.

### Confecciona el diagrama del modelo de datos. (Marcelo)



### Diccionario de datos.

**1. Roles**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Campo | Tipo de dato | Longitud | Nulo | Clave | Descripción |
| id\_rol | INT | — | NO | PK | Identificador único del rol. |
| nombre\_rol | VARCHAR | 50 | NO | — | Nombre del rol asignado al usuario (ADMIN o VENDEDOR). |

**Relaciones:**

1:N con la tabla usuarios.  
(Un rol puede estar asignado a varios usuarios).

**2. Usuarios**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Campo | Tipo de dato | Longitud | Nulo | Clave | Descripción |
| id\_usuario | INT | — | NO | PK | Identificador único del usuario. |
| nombre | VARCHAR | 50 | NO | — | Nombre del usuario. |
| apellido | VARCHAR | 50 | NO | — | Apellido del usuario. |
| username | VARCHAR | 50 | NO | UNIQUE | Nombre de usuario utilizado para iniciar sesión. |
| password | VARCHAR | 255 | NO | — | Contraseña en formato cifrado (hash). |
| rol\_id | INT | — | NO | FK | Identificador del rol, relacionado con la tabla roles. |

**Relaciones:**

* N:1 con roles.
* 1:N con ventas (un usuario puede registrar varias ventas).

**3. Productos**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Campo | Tipo de dato | Longitud | Nulo | Clave | Descripción |
| id\_producto | INT | — | NO | PK | Identificador único del producto. |
| codigo | VARCHAR | 30 | NO | — | Código o SKU interno del producto. |
| nombre | VARCHAR | 100 | NO | — | Nombre comercial del producto. |
| categoria | VARCHAR | 50 | SÍ | — | Categoría o tipo de producto (bebidas, abarrotes, etc.). |
| precio | DECIMAL | (10,2) | NO | — | Precio de venta unitario del producto. |
| stock | INT | — | NO | — | Cantidad actual disponible. |
| stock\_minimo | INT | — | NO | — | Cantidad mínima antes de generar alerta. |

**Relaciones:**

* 1:N con detalle\_venta.
* 1:N con alertas.

**4. Ventas**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Campo | Tipo de dato | Longitud | Nulo | Clave | Descripción |
| id\_venta | INT | — | NO | PK | Identificador único de la venta. |
| usuario\_id | INT | — | NO | FK | Usuario responsable de la venta. |
| fecha | DATETIME | — | NO | — | Fecha y hora de la transacción. |
| total | DECIMAL | (10,2) | NO | — | Total, de la venta. |

**Relaciones**

* N:1 con usuarios.
* 1:N con detalle\_venta.

**5. Detalle\_venta**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Campo | Tipo de dato | Longitud | Nulo | Clave | Descripción |
| id\_detalle | INT | — | NO | PK | Identificador del registro de detalle. |
| venta\_id | INT | — | NO | FK | Referencia a la venta correspondiente. |
| producto\_id | INT | — | NO | FK | Producto vendido en la transacción. |
| cantidad | INT | — | NO | — | Cantidad de unidades vendidas. |
| precio\_unitario | DECIMAL | (10,2) | NO | — | Precio del producto al momento de la venta. |
| subtotal | DECIMAL | (10,2) | NO | — | Resultado de (cantidad × precio\_unitario). |

**Relaciones:**

* N:1 con ventas.
* N:1 con productos.

**6. Alertas**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Campo | Tipo de dato | Longitud | Nulo | Clave | Descripción |
| id\_alerta | INT | — | NO | PK | Identificador único de la alerta. |
| producto\_id | INT | — | NO | FK | Producto asociado a la alerta. |
| tipo\_alerta | ENUM | — | NO | — | Tipo de alerta: BAJO\_STOCK o TENDENCIA\_VENTA. |
| mensaje | TEXT | — | NO | — | Descripción o recomendación generada automáticamente. |
| fecha\_generacion | DATETIME | — | NO | — | Fecha y hora en que se generó la alerta. |

**Relaciones:**

* N:1 con productos.

### Confecciona y describe el diagrama de la Topología de comunicación que da soporte a la solución (Joaquín)

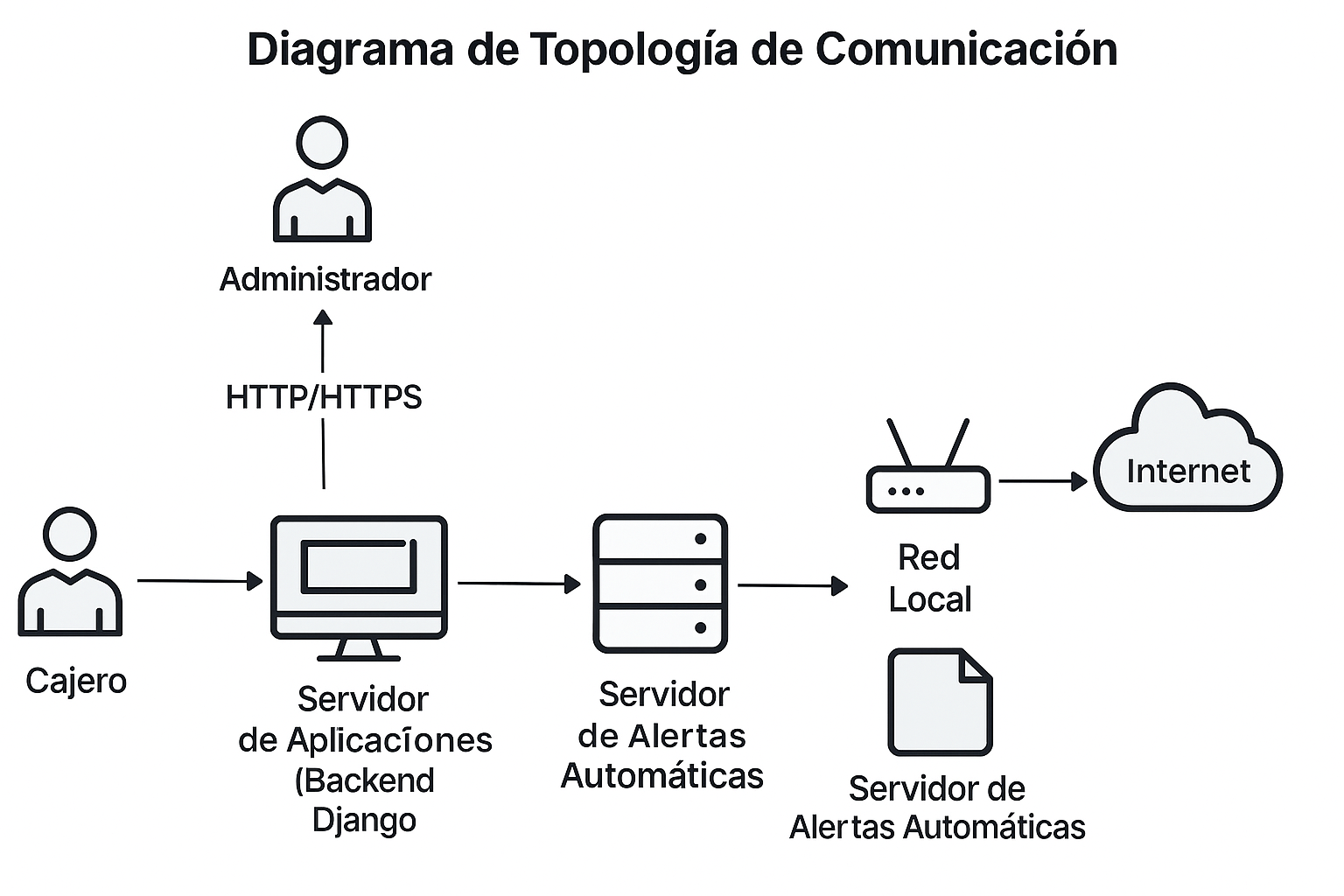
**Diagrama de Topología de Comunicación**

**Componentes principales:**

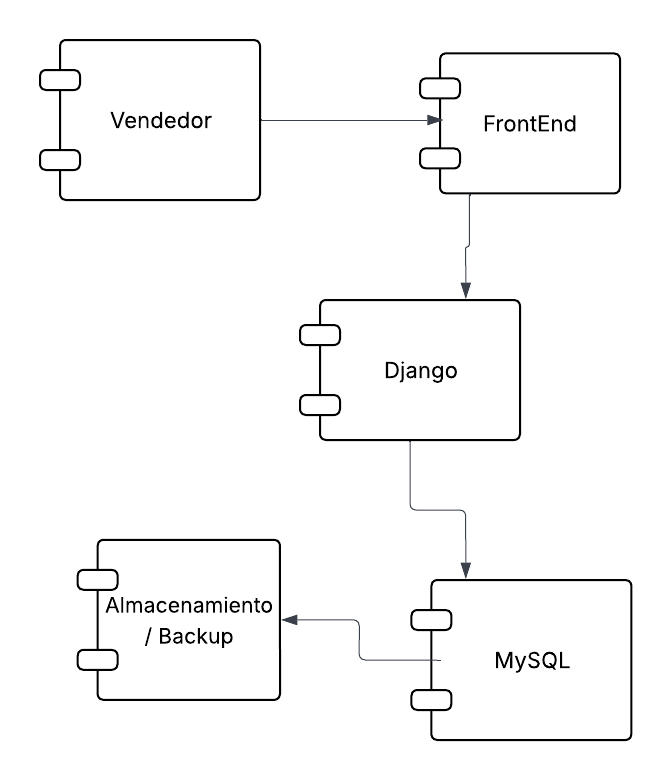
1. **Usuarios finales (Administrador / Cajero)**
   1. Utilizan computadores o tablets dentro del minimarket.
   2. Acceden al sistema mediante navegador web (HTTP/HTTPS).
2. **Servidor de Aplicaciones (Backend Django)**
   1. Alojado en un servidor local o en la nube.
   2. Ejecuta la lógica del sistema y gestiona las solicitudes.
3. **Servidor de Base de Datos (MySQL)**
   1. Puede estar en el mismo equipo que el backend o separado.
   2. Almacena información de usuarios, productos, ventas y alertas.
4. **Red local del Minimarket (LAN)**
   1. Conecta los equipos del administrador y cajeros al servidor mediante router o switch.
5. **Conexión a Internet (opcional)**
   1. Permite acceso remoto al sistema o respaldo en la nube.
   2. Soporta notificaciones por correo o servicios web.
6. **Servidor de Alertas Automáticas**
   1. Servicio o script programado que se ejecuta en el servidor.
   2. Envía notificaciones internas o externas (por correo).

**Descripción de la comunicación:**

1. El **Cajero o Administrador** ingresa al sistema desde su navegador.
2. La solicitud viaja por la **red local (LAN)** hacia el **Servidor Django**.
3. Django procesa la petición, consulta o actualiza la **base de datos MySQL**.
4. La respuesta (por ejemplo, stock actualizado o reporte generado) retorna al navegador del usuario.
5. El **módulo de alertas automáticas** monitorea la base de datos y envía notificaciones al **Administrador** cuando detecta productos próximos a vencer.



### Confecciona el diagrama de diseño de la infraestructura que da soporte a la solución. (Marcelo)



### Confecciona y describe el diagrama de Arquitectura que da soporte a la solución. especificando los componentes de infraestructura TI. tanto de software como de hardware. (Joaquín)

**1. Descripción general de la arquitectura**

La solución sigue una **arquitectura de tres capas (3-tier architecture)**:

1. **Capa de presentación (Frontend / Cliente)**
   1. Interfaz web accesible desde navegadores.
   2. Se comunica con el servidor mediante peticiones HTTP/HTTPS.
   3. Permite a **administradores** y **cajeros** interactuar con el sistema.
2. **Capa de aplicación (Backend Django)**
   1. Contiene la lógica de negocio.
   2. Gestiona peticiones de los usuarios, controla las operaciones CRUD, las alertas y genera reportes.
   3. Desarrollada en **Python (Django Framework)**.
3. **Capa de datos (Base de datos MySQL)**
   1. Almacena toda la información estructurada (usuarios, productos, ventas, alertas).
   2. Garantiza integridad, seguridad y consistencia de datos.

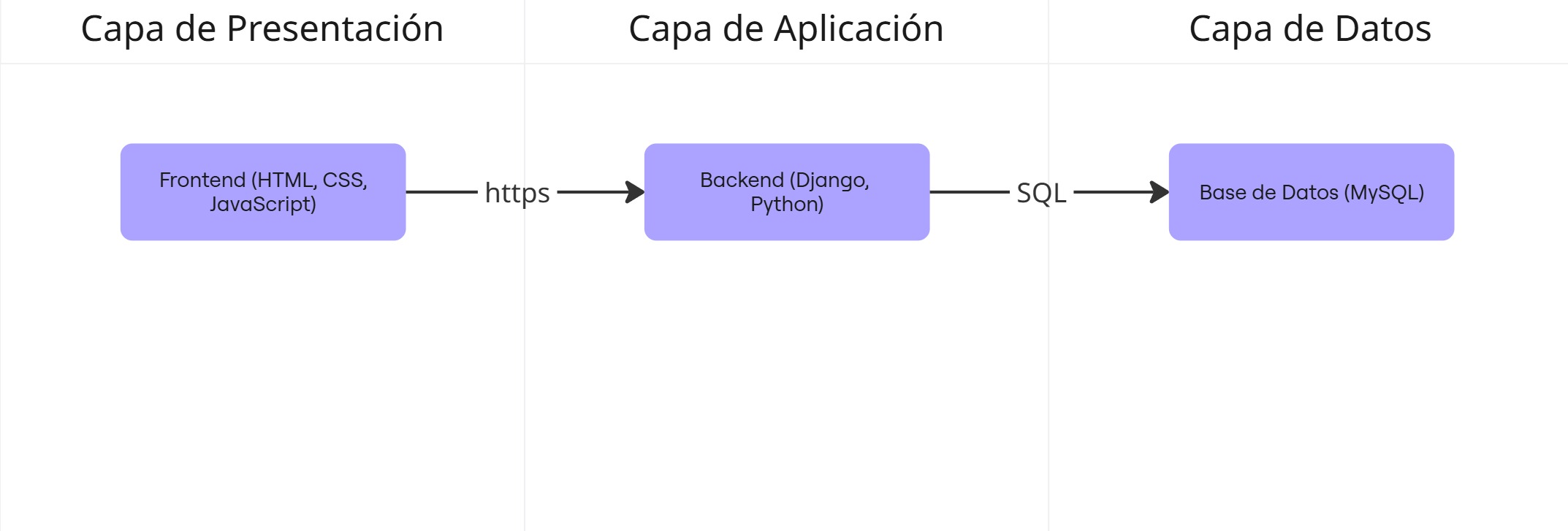
**2. Componentes de infraestructura**

**Software**

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Componente** | **Descripción** |
| Sistema Operativo | Linux o Windows Server | Soporte del servidor de aplicaciones y base de datos. |
| Framework | Django (Python) | Lógica de negocio, autenticación y control de inventario. |
| Servidor Web | Apache o Nginx | Intermediario entre navegador y aplicación Django. |
| Base de Datos | MySQL | Almacena los registros del sistema. |
| Navegador Web | Chrome, Edge o Firefox | Interfaz de acceso de usuarios. |
| Notificaciones | Script de alertas o servicio de correo SMTP | Envía alertas automáticas de vencimientos. |

**Hardware**

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Componente** | **Descripción** |
| Servidor de Aplicaciones | CPU 4 núcleos, 8 GB RAM, SSD 256 GB | Aloja Django y los servicios web. |
| Servidor de Base de Datos | CPU 2 núcleos, 4 GB RAM, SSD 128 GB | Guarda los datos en MySQL. |
| Equipos de Usuario | PC o tablets con acceso LAN/WiFi | Cajero y administrador acceden al sistema. |
| Router / Switch | Red local interna | Conecta los equipos y servidores. |
| Conexión a Internet | Fibra óptica / ADSL | Permite acceso remoto y envío de alertas. |



# Indicadores KPI y SLA del negocio.