DOCUMENTO DE ARQUITECTURA DE SOFTWARE

Arazaka CyberSolutions

Bar: “Mi primera Borrachera”

Versión: 2.0

Tabla de contenido

1. Introducción

1.1 Propósito

El propósito de este documento es definir y presentar la Arquitectura de Software que será utilizada en el desarrollo del sistema de gestión para los establecimientos de la cadena “Mi primera borrachera”. En esta fase del proyecto, correspondiente al sprint 5, se implementarán las funcionalidades de Login y CRUD de usuarios. Este documento describe las vistas de arquitectura de componentes, despliegue, y el modelo de datos, que son fundamentales para la correcta operación de estas funcionalidades.

Está dirigido tanto al equipo de desarrollo como a otros interesados, con el objetivo de proporcionar una visión clara y comprensible de la estructura y organización del sistema. Asimismo, este documento servirá como referencia para la implementación y asegurará que se cumplan los requisitos no funcionales del sistema, como la seguridad, mantenibilidad y escalabilidad.

Con esta arquitectura, buscamos optimizar el proceso de autenticación y la gestión de usuarios, asegurando que el sistema pueda crecer de manera eficiente y mantener la integridad de los datos a lo largo del tiempo.

1.2 Definiciones y siglas.

DAS: Documento de arquitectura de software.
MySQL: Software gestor de base de datos.
Sprint Boot: Framework que nace con la finalidad de simplificar el desarrollo de aplicaciones basadas en el framework Spring Core.
Front-End: Capa de una aplicación que se encarga de la interfaz gráfica y la interacción directa con el usuario.
Back-End: Parte del sistema que gestiona el procesamiento de datos y la lógica interna, respondiendo a las solicitudes del Front-End.
JWT (JSON Web Token): Es un estándar abierto basado en JSON para la creación de tokens de acceso que permiten la propagación de identidad y privilegios.

2. Buenas Prácticas de Desarrollo

Arquitectura en Capas: La separación de responsabilidades mejora la modularidad del sistema. Al dividir la presentación, la lógica de negocio y el acceso a datos, es más fácil mantener y actualizar cada parte sin afectar las demás, lo que es esencial para la escalabilidad a largo plazo en sistemas como el que se propone para las sedes de "Mi Primera Borrachera".

Principios SOLID: Estos principios aseguran un código flexible, fácil de entender y modificar. Al aplicar SOLID, garantizamos que el sistema sea adaptable a futuros cambios, lo cual es crucial para agregar funcionalidades como las que se detallan en la propuesta, como la gestión de inventario y roles

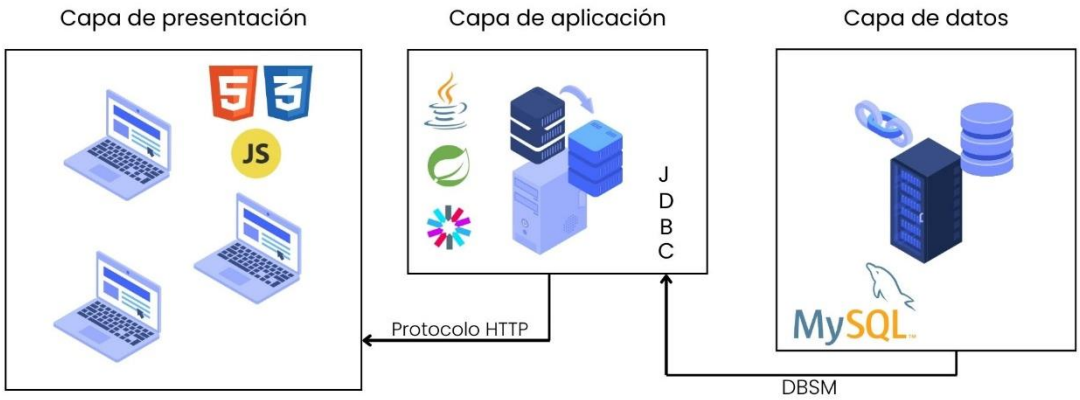
Manejo de Excepciones: Un manejo adecuado de excepciones asegura que el sistema sea robusto y fácil de depurar. Esto es crítico para un entorno de bares donde cualquier falla en el sistema podría afectar la operación diaria. Implementar un manejo centralizado de excepciones y logging mejorará la estabilidad del sistema.

Pruebas Automatizadas: Garantizan la calidad del código, detectando errores tempranamente. Esto es vital en un entorno con múltiples roles y funcionalidades específicas, como la toma de pedidos, reportes, y gestión de inventario.

Seguridad: En un sistema que maneja pagos y datos de clientes, es esencial aplicar medidas de seguridad, como encriptación de datos sensibles y uso de HTTPS. Esto protege contra vulnerabilidades comunes y asegura la confianza de los usuarios y del negocio.

Documentación del Código: La buena documentación asegura que otros desarrolladores puedan entender y mantener el código en el futuro, un aspecto importante para el crecimiento y evolución del sistema, especialmente cuando se involucren nuevos equipos.

3. Diagrama de arquitectura



Descripción General

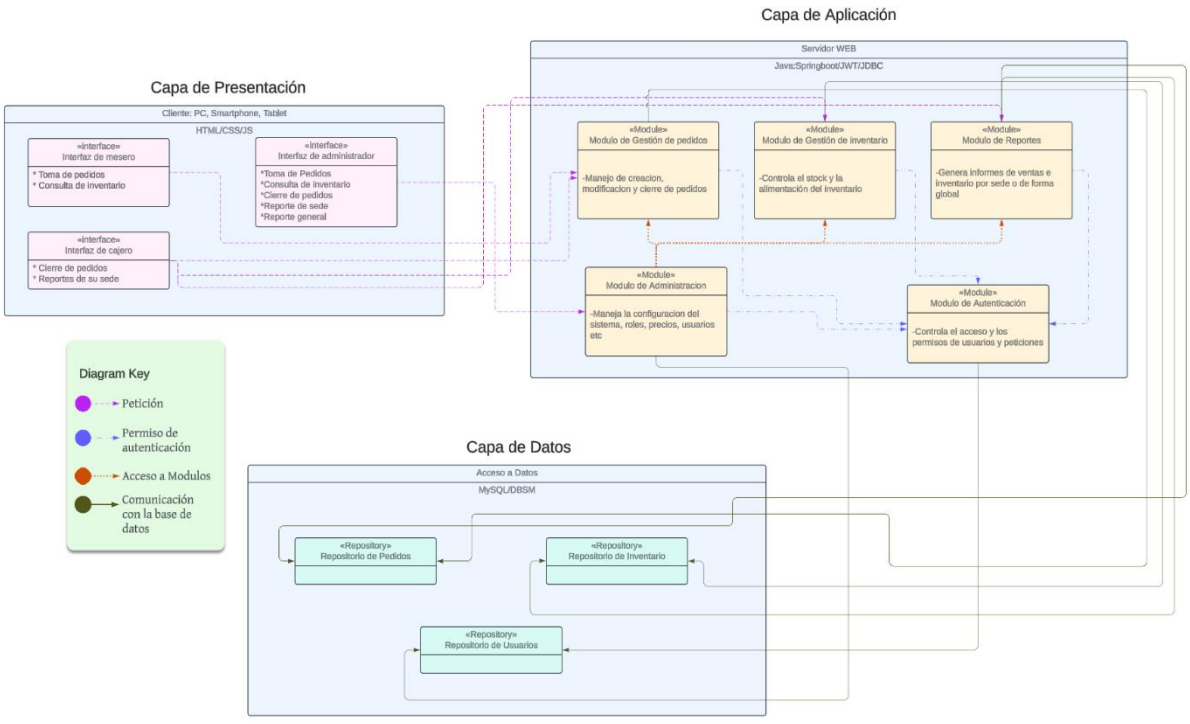
El diagrama presentado representa una arquitectura de software de una aplicación web de tres capas: presentación, aplicación y datos. Esta estructura modular permite una clara separación de responsabilidades y facilita el desarrollo, mantenimiento y escalabilidad de la aplicación.

Capas de la Arquitectura

1. **Capa de Presentación:**
    - **Interfaz de Usuario:** Se compone de páginas web (representadas por los íconos de laptops) que interactúan directamente con el usuario.
    - **Tecnologías:** Emplea HTML5 y JavaScript (JS) para construir interfaces dinámicas y responsivas.
    - **Función:** Se encarga de la presentación visual de la información y la captura de las interacciones del usuario.

2. **Capa de Aplicación:**
    - **Lógica de Negocio:** Contiene la lógica que define el comportamiento de la aplicación, como las reglas de negocio, los cálculos y las validaciones.
    - **Tecnologías:** Utiliza Java como lenguaje principal para gestionar las solicitudes hechas por la cpaa de presentación, JWT (Json Web Token) para la seguridad y autenticación de cada solicitud y Springboot como Framework para usar características como Hibernate.
    - **Función:** Procesa las solicitudes de la capa de presentación, accede a la capa de datos y genera las respuestas correspondientes.
3. **Capa de Datos:**
    - **Base de Datos:** Emplea MySQL como sistema de gestión de bases de datos (DBMS).
    - **Función:** Almacena de forma persistente la información de la aplicación, como usuarios, productos, pedidos, etc.
    - **Conexión:** Se comunica con la capa de aplicación a través del protocolo JDBC (Java Database Connectivity).

4. Diagrama de componentes / despliegue



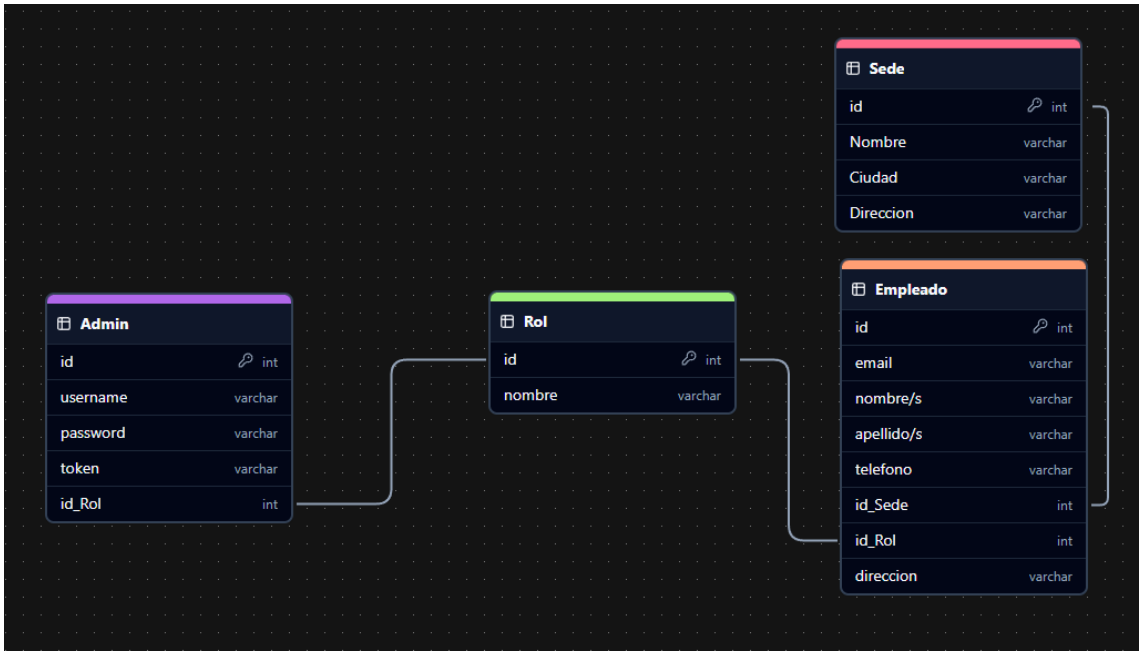
“Haga clic aquí para ver el diagrama más a detalle”

El diagrama de componentes ilustra la arquitectura de tres capas del sistema:

1. Capa de Presentación: Consta de interfaces específicas para cada rol de usuario (Mesero, Cajero, Administrador), facilitando una experiencia de usuario adaptada a sus responsabilidades.
2. Capa de Aplicación: Comprende módulos especializados para Gestión de Pedidos, Inventario, Reportes y Administración. Un módulo central de Autenticación y Autorización asegura el control de acceso adecuado.
3. Capa de Datos: Incluye repositorios para Pedidos, Inventario y Usuarios, centralizando el almacenamiento y acceso a datos.

Esta estructura modular promueve la separación de responsabilidades, facilitando el mantenimiento y la escalabilidad. Las interacciones entre componentes están claramente definidas, permitiendo una gestión eficiente de las operaciones del negocio, desde la toma de pedidos hasta la generación de reportes, mientras se mantiene un control robusto sobre el acceso y la seguridad del sistema

5. Modelo Entidad Relación



El modelo entidad-relación presentado estructura un sistema de gestión de usuarios basado en roles, integrando cuatro entidades principales: **Admin**, **Empleado**, **Rol** y **Sede**.

- **Admin**: Almacena los datos de los administradores del sistema, incluyendo credenciales y tokens de autenticación. Cada administrador está asociado a un rol específico mediante el campo **id_Rol**, lo que facilita la asignación de permisos.
- **Empleado**: Registra a los empleados con su información de contacto y se asocia a un rol específico a través de **id_Rol**, garantizando la consistencia en la gestión de roles. Además, cada empleado se asocia a una sede particular mediante el campo **id_Sede**, lo que permite organizar a los empleados por ubicación.
- **Rol**: Define los distintos niveles de acceso y permisos del sistema, centralizando la información de los roles tanto para administradores como para empleados, lo que facilita la gestión y escalabilidad.
- **Sede**: Almacena la información de las distintas ubicaciones o sedes de la organización, incluyendo el nombre, ciudad y dirección. La relación entre las sedes y los empleados se gestiona mediante el campo **id_Sede** en la tabla **Empleado**, permitiendo así una estructura

clara y organizada para gestionar a los empleados por su ubicación física.

Relaciones: Tanto los administradores como los empleados tienen una relación de muchos a uno con la tabla **Rol**, lo que permite gestionar permisos de manera eficiente y flexible. Además, la relación entre **Empleado** y **Sede** organiza y asigna a los empleados según su ubicación, mejorando la escalabilidad y claridad en la estructura del sistema.