

Proyecto

***Soporte Digit@l***

| **EQUIPO DE TRABAJO** | |
| --- | --- |
| **Número** | **Apellidos y Nombres** |
| 1 | Medina Quispe, Licia |
| 2 | Barco Peña Sofia, Antonela |
| 3 | Sanchez Rodas, Neyder Jose |
| 4 |  |
| 5 |  |

**ÍNDICE**

[**1. Resumen 3**](#_cxfrntswh6gp)

[**2. Problema 3**](#_zbx4i2g61ss4)

[**3. Objetivo 4**](#_8lbmd7so5fx6)

[Objetivo General: 4](#_umrj3pmz661)

[Objetivos Específicos (Modelo SMART): 4](#_nmvtke76m31a)

[**4. Fundamento Teórico 4**](#_bm77a37bza31)

[Modelado de Información y Base de Datos 4](#_igwuqwj4deee)

[Tecnologías de Desarrollo Web 4](#_vorrs4vype91)

[**5. Planteamiento de Solución 5**](#_wjllhchiquw8)

[Modelo de Información 5](#_4pj8u3hply7e)

[Diseño de Base de Datos 6](#_4lgtko7ndy8i)

[Script DDL de Base de Datos 10](#_eaov5isgy2qu)

[Diagrama de Clases 10](#_uvg3hvmkx9hj)

[Modelo MVC 10](#_qymk55nishfj)

[**6. Conclusión 10**](#_awb146ck9jlh)

[**7. Recomendaciones 10**](#_csjjcgqcrhw1)

[**8. Referencias Bibliográficas 11**](#_tgx1z1302hzs)

### Resumen

En la segunda etapa del desarrollo del proyecto de la plataforma web de la empresa 123digit@l, se abordó la problemática de la gestión ineficiente de solicitudes de soporte técnico mediante procesos manuales y desorganizados. Esta situación derivó en una baja productividad y en una experiencia deficiente para los clientes. La solución propuesta fue la implementación de una plataforma web basada en la arquitectura Modelo-Vista-Controlador (MVC), utilizando Java, PostgreSQL, JSP y servlets. El sistema desarrollado permite gestionar las solicitudes de los clientes de manera centralizada, asignar actividades a colaboradores, registrar avances y generar reportes de estadísticas. Además, se han implementado funcionalidades para la gestión de perfiles de usuarios, la actualización de datos personales y la subida de una foto de perfil. La base de datos ha sido diseñada para soportar toda la lógica del negocio, con tablas como usuarios, colaboradores, actividades y roles, entre otras.

El desarrollo de este proyecto reafirma la importancia de la automatización de procesos y el uso de patrones de diseño para la mejora continua de la eficiencia empresarial. Con la implementación de esta plataforma, se logró reducir significativamente los tiempos de respuesta y mejorar la experiencia de los usuarios, cumpliendo así con los objetivos planteados. Se recomienda, para etapas futuras, la incorporación de notificaciones automáticas y mejoras en la interfaz para lograr un uso más intuitivo.

### Problema

123digit@l, empresa de desarrollo de software, no cuenta con una plataforma digital para gestionar eficientemente la atención a sus clientes. Actualmente, las solicitudes de soporte, capacitación o requerimientos no se registran de forma estructurada, y no hay un sistema que permite asignarlas, hacer seguimiento ni registrar el trabajo realizado por los colaboradores.

Esto genera desorganización, falta de trazabilidad en las actividades, demoras en la atención y deficiente comunicación con el cliente. Además, la empresa no dispone de herramientas para analizar el desempeño del área de soporte. Por ello, se requiere una solución computacional que automatice y optimice todo el proceso de atención, desde la solicitud hasta su cierre.

### Objetivo

#### Objetivo General:

Desarrollar e implementar una plataforma web para gestionar las solicitudes de asistencia de los clientes de 123digit@l, que permita registrar, asignar, monitorear y finalizar los casos de soporte, capacitación o requerimientos, con trazabilidad y reportes gráficos, en un plazo máximo de 18 semanas, mejorando la eficiencia del proceso de atención en un 100%.

#### Objetivos Específicos (Modelo SMART):

Diseñar e implementar los módulos de registro, asignación y seguimiento de solicitudes en un entorno web funcional, dentro de las primeras 5 semanas del proyecto.

Desarrollar funcionalidades de registro de actividades, control de tiempos y notificaciones automáticas, asegurando trazabilidad y comunicación con el cliente, antes del avance final

Incorporar un panel de estadísticas gráficas para el administrador, que permita evaluar el desempeño del soporte técnico, al finalizar el cuarto mes del proyecto. (OPCIONAL)

### Fundamento Teórico

La solución propuesta para 123digit@l está sustentada en conocimientos fundamentales de bases de datos, programación orientada a objetos y desarrollo web utilizando tecnologías Java. Su objetivo es automatizar el registro, seguimiento y cierre de solicitudes de soporte técnico mediante una plataforma eficiente y escalable.

#### Modelado de Información y Base de Datos

Se utilizó un Diagrama Entidad–Relación (ER) con notación de Chen para identificar entidades clave como usuarios, solicitudes, asignaciones y actividades. A partir de este modelo, se diseñó una base de datos relacional normalizada, y se elaboró un script DDL para la creación de tablas, claves primarias y relaciones, que puede ejecutarse en gestores como MySQL o PostgreSQL.

#### Tecnologías de Desarrollo Web

La plataforma fue desarrollada en Java utilizando NetBeans como entorno de desarrollo, y desplegada en el servidor Apache Tomcat. Se aplicaron tecnologías como:

* JavaServer Pages (JSP): Permite generar interfaces dinámicas en el navegador, integrando código Java dentro del HTML para mostrar información desde el servidor.
* Servlets: Gestionan las peticiones del cliente, ejecutan la lógica del negocio y actúan como intermediarios entre la vista y el modelo.
* Java Persistence API (JPA): Facilita la conexión con la base de datos mediante el mapeo objeto-relacional, evitando el uso de sentencias SQL directas y manteniendo una estructura más orientada a objetos.
* JavaMail API: Se utiliza para automatizar el envío de correos electrónicos, como notificaciones a los clientes sobre el estado de sus solicitudes.

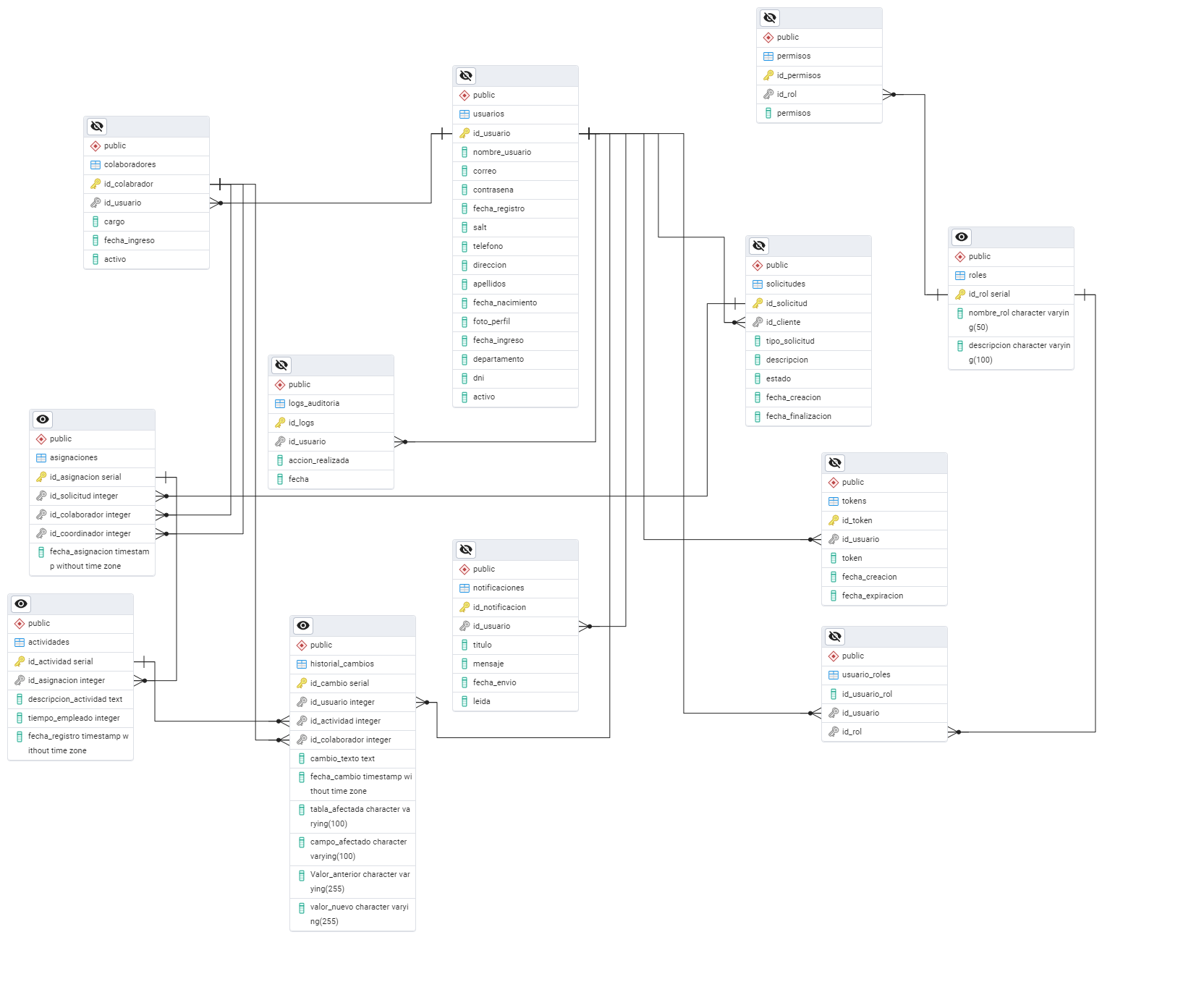
Esta combinación de herramientas y técnicas permite implementar una aplicación modular, segura y funcional, orientada a mejorar el proceso de atención a clientes dentro de la empresa.

### Planteamiento de Solución

#### Modelo de Información

* **Diagrama Entidad-Relación**: El modelo de información está representado mediante un Diagrama Entidad-Relación (ERD), donde se definen las entidades principales (Usuario, Colaborador, Solicitud, Asignación, Actividad, etc.) y las relaciones entre ellas. Se utilizó la notación Crow's Foot para ilustrar las relaciones.

Este diagrama tiene las siguientes entidades principales.



#### Diseño de Base de Datos

La base de datos fue diseñada siguiendo las reglas de normalización hasta

la tercera forma normal (3NF), asegurando la eliminación de redundancias y

la integridad de los datos.

1. **Usuarios**

**Tabla: usuarios**

* Atributos: id\_usuario, nombre\_usuario, correo, contrasena, telefono, direccion, dni, fecha\_registro, etc.
* Descripción: Esta tabla almacena los datos básicos de cada usuario registrado en el sistema. Incluye datos personales como el correo y contraseña (almacenada de forma segura, encriptada), la dirección y demás detalles.
* Propósito: Permite gestionar la información de los usuarios, sirviendo como base para identificar a los colaboradores y clientes dentro del sistema.

1. **Roles y Permisos**

**Tabla: roles**

* Atributos: id\_rol, nombre\_rol.
* Descripción: Define los roles que existen dentro del sistema, como 'Administrador', 'Analista', etc.
* Propósito: Los roles ayudan a definir los permisos que tendrá cada usuario según su rol.

**Tabla: permisos**

* Atributos: id\_permiso, permiso.
* Descripción: Define los permisos que cada rol puede tener. Esto se asocia a acciones específicas que se pueden realizar en el sistema.

**Tabla: usuarios\_roles**

* Atributos: id\_usuario, id\_rol.
* Descripción: Tabla intermedia que asocia cada usuario a uno o varios roles. Esto permite tener flexibilidad en la asignación de permisos.
* Propósito: Implementar un modelo de permisos flexible basado en roles.

1. **Colaboradores**

**Tabla: colaboradores**

* Atributos: id\_colaborador, id\_usuario, cargo, fecha\_ingreso.
* Descripción: Almacena información específica de los colaboradores, que son usuarios con permisos para gestionar solicitudes y realizar actividades.
* Propósito: Permitir extender la información de los usuarios que son colaboradores con atributos específicos, como el cargo y la fecha de ingreso.

1. **Solicitudes**

**Tabla: solicitudes**

* Atributos:

id\_solicitud,

tipo\_solicitud

descripcion

estado

id\_cliente.

* Descripción: Almacena todas las solicitudes realizadas por los clientes. Cada solicitud tiene un estado que define su progreso dentro del sistema.
* Propósito: Centralizar las solicitudes de soporte técnico que ingresan al sistema para facilitar su gestión.

1. **Asignaciones**

**Tabla: asignaciones**

* Atributos:

id\_asignacion

id\_solicitud

id\_colaborador

fecha\_asignacion

* Descripción: Contiene las asignaciones de solicitudes a colaboradores específicos.
* Propósito: Registrar quién es responsable de atender cada solicitud, asegurando la trazabilidad y la gestión adecuada de cada caso.

1. **Actividades**

**Tabla: actividades**

* Atributos:

id\_actividad

id\_asignacion, descripcion\_actividad, tiempo\_empleado, fecha\_registro.

* Descripción: Almacena las actividades realizadas en una solicitud.
* Cada actividad está relacionada con una asignación específica.
* Propósito: Permitir el seguimiento detallado del progreso de cada solicitud y del trabajo realizado por cada colaborador.

1. **Notificaciones**

**Tabla: notificaciones**

* Atributos:

id\_notificacion

titulo, mensaje

fecha\_envio

usuario\_destino

* Descripción: Almacena las notificaciones que se envían a los usuarios, ya sea para informarles sobre el estado de sus solicitudes, nuevas asignaciones, etc.
* Propósito: Mantener informados a los usuarios sobre el estado y el
* progreso de las solicitudes.

1. **Tokens**

**Tabla: tokens**

* Atributos:

id\_token, id\_usuario

token

fecha\_creacion

fecha\_expiracion

* Descripción: Almacena tokens de autenticación utilizados para restablecer contraseñas o confirmar correos electrónicos.
* Propósito: Implementar un mecanismo seguro para gestionar la autenticación y verificación de usuarios.

1. **Historial de Cambios**

**Tabla: historial\_cambios**

* Atributos:

id\_cambio, descripcion\_cambio, fecha\_cambio,

tabla\_afectada, campo\_afectado, valor\_anterior, valor\_nuevo,

usuario\_responsable.

* Descripción: Almacena un registro de cada cambio importante realizado en el sistema.
* Propósito: Facilitar la auditoría de cambios y mejorar la trazabilidad de las operaciones en la base de datos.

**10. Logs de Auditoría**

**Tabla: logs\_auditoria**

* Atributos:

id\_log

accion\_realizada

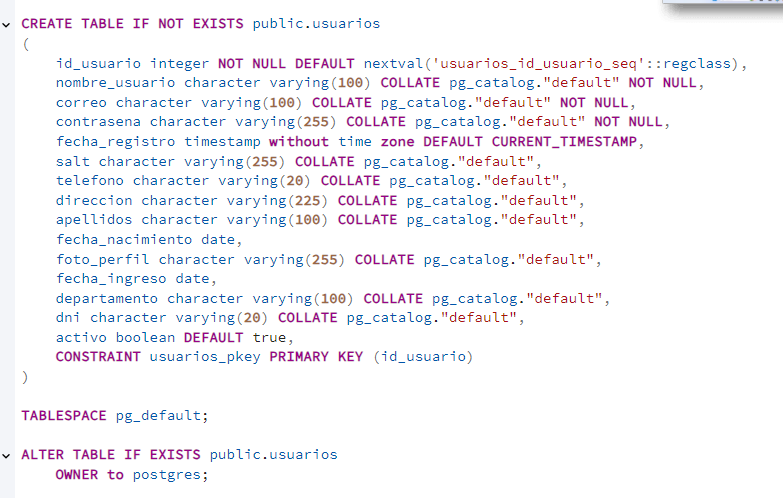
fecha\_accion

usuario\_responsable.

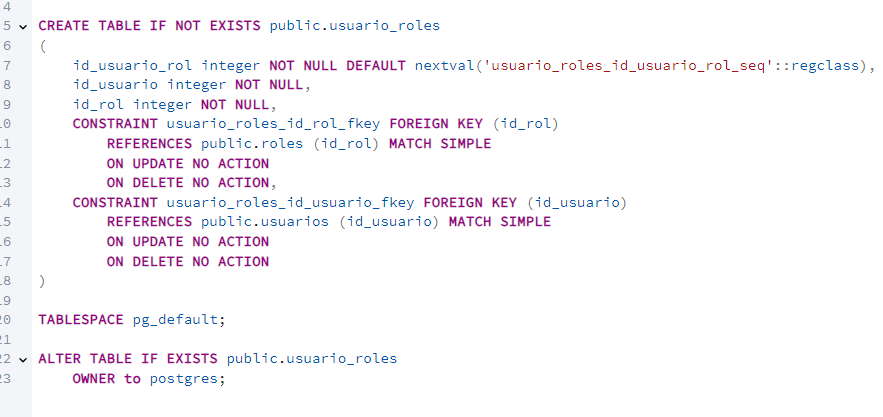
* Descripción: Almacena acciones realizadas en el sistema, como inicios de sesión, cambios importantes, etc.
* Propósito: Ayudar a monitorizar las actividades de los usuarios y a mejorar la seguridad del sistema mediante auditoría.

#### Script DDL de Base de Datos

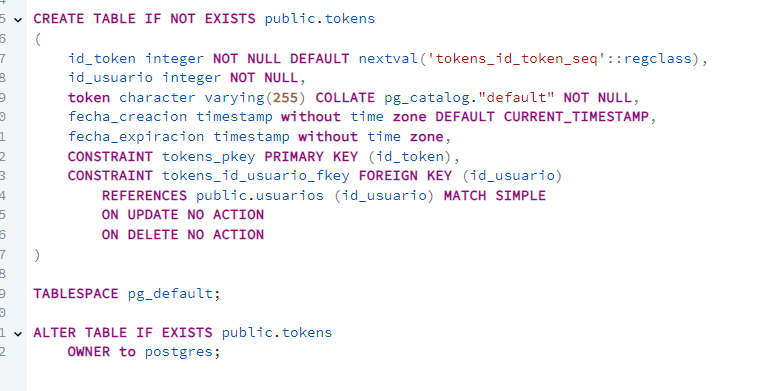
**Usuarios**



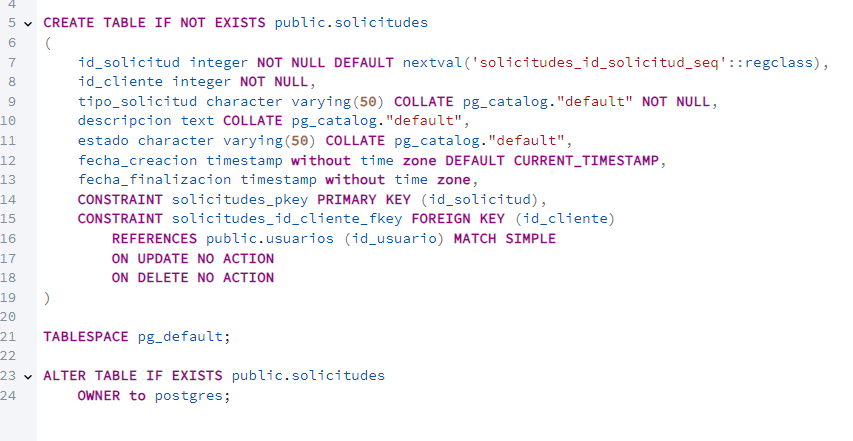
**Usuario\_roles**



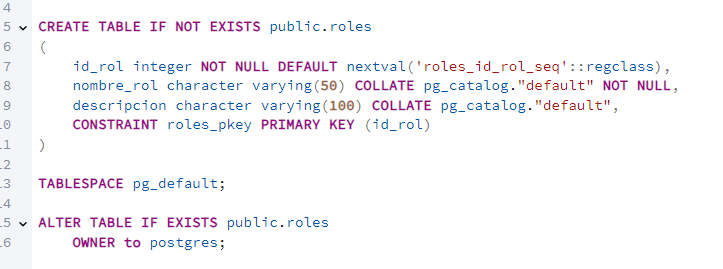
**Tokens**

****

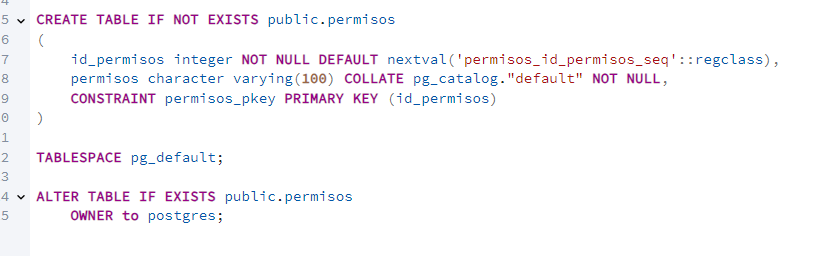
**Solicitudes**

****

**Roles**

****

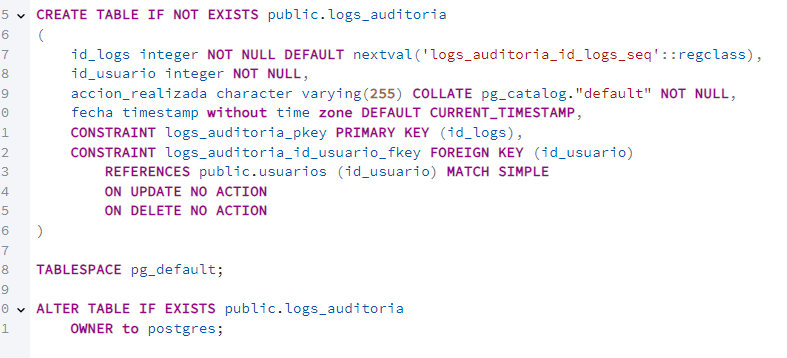
**Permisos**

****

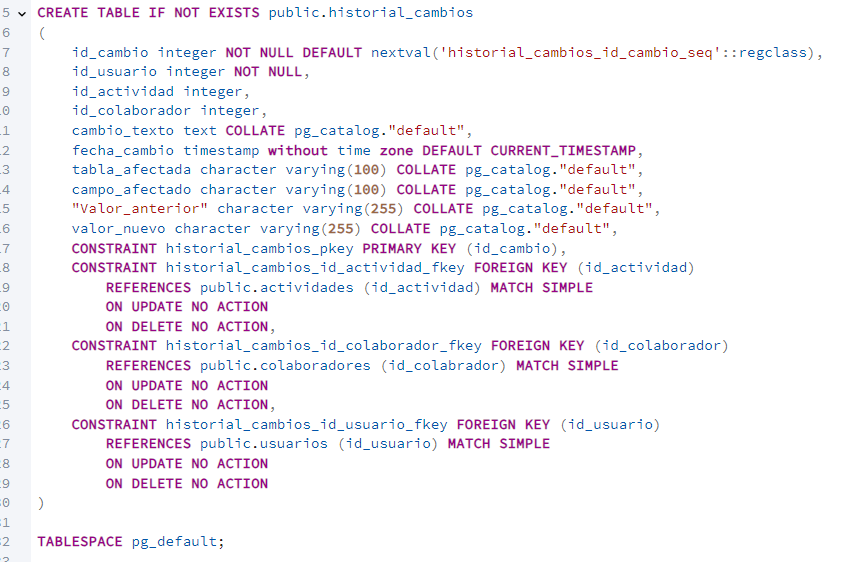
**Notificaciones**

****

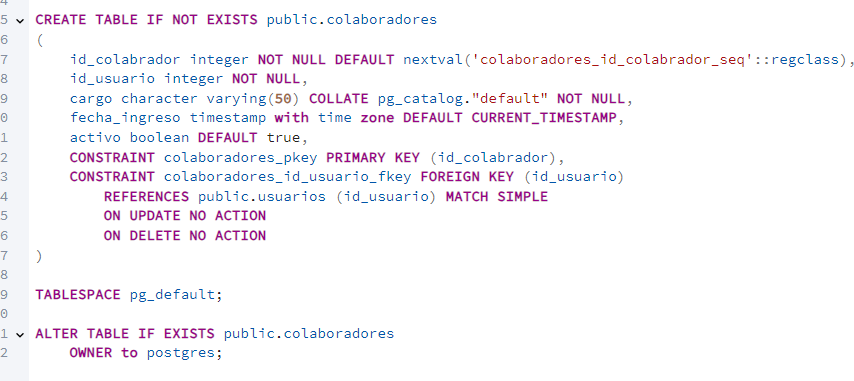
**Logs\_auditoria**

****

**Historial\_cambios**

****

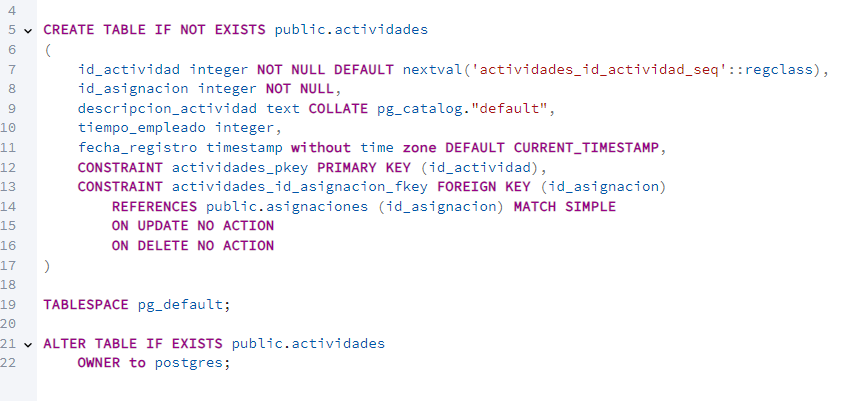
**Colaboradores**

****

**Asignaciones**

****

**Actividades**

****

#### Diagrama de Clases

#### Modelo MVC

### Conclusión

La implementación de una plataforma web para la gestión de solicitudes de soporte técnico en 123digit@l representa un avance significativo en la optimización de procesos internos y mejora de la atención al cliente. La adopción de tecnologías modernas y una arquitectura robusta ha permitido automatizar tareas previamente manuales, reducir errores y mejorar la eficiencia operativa.​

La separación de responsabilidades mediante la arquitectura MVC ha facilitado el desarrollo modular y el mantenimiento del sistema. El uso de Java y PostgreSQL ha proporcionado una base sólida y segura para el manejo de datos críticos. La integración de la API JavaMail ha mejorado la comunicación con los clientes, manteniéndolos informados sobre el estado de sus solicitudes.​

En resumen, la solución implementada ha cumplido con los objetivos planteados, mejorando la eficiencia en la gestión de soporte técnico y elevando la calidad del servicio ofrecido a los clientes.

### Recomendaciones

Para asegurar la sostenibilidad y mejora continua del sistema desarrollado, se proponen las siguientes recomendaciones:​

* Capacitación continua: Brindar formación regular a los colaboradores sobre el uso de la plataforma y las mejores prácticas en la atención al cliente.​
* Monitoreo y evaluación: Implementar métricas de desempeño para evaluar la eficiencia del sistema y la satisfacción del cliente, permitiendo ajustes y mejoras continuas.​
* Actualización tecnológica: Mantener la plataforma actualizada con las últimas versiones de las tecnologías utilizadas, garantizando seguridad y compatibilidad.​
* Retroalimentación del usuario: Establecer canales para recibir comentarios de los usuarios, identificando áreas de mejora y adaptando el sistema a sus necesidades.​
* Escalabilidad: Planificar la expansión del sistema para integrar nuevas funcionalidades o adaptarse al crecimiento de la empresa.

### Referencias Bibliográficas

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Horstmann, C. S., & Cornell, G. (2013). Core Java Volume I--Fundamentals (9th ed.). Prentice Hall.

JavaMail API. (2020). JavaMail API Documentation. Oracle. <https://javaee.github.io/javamail/>

Momjian, B. (2001). PostgreSQL: Introduction and Concepts. Addison-Wesley.