PORTAL WEB PARA LA COMPAÑÍA MASGAS PRESTADORA DE SERVICIOS RELACIONADOS CON GASODOMÉSTICOS

DIEGO ALEJANDRO ZAPATA MONSALVE

Trabajo de grado para optar al título de Tecnólogo en Desarrollo de Software

Asesor Metodológico y Técnico

Oscar Julián Galeano Echeverri

Msc. en Gestión de la Tecnología Educativa

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO

FACULTAD DE INGENIERÍA

TECNOLOGÍA EN DESARROLLO DE SOFTWARE

MEDELLÍN

2025

Contenido

Pág.

Contenido

Lista de figuras

Pág.

Lista de anexos

Pág.

Resumen

El presente trabajo de grado tiene como objetivo desarrollar un portal web para la empresa MasGas, dedicada a la prestación de servicios relacionados con gasodomésticos en Medellín y sus alrededores. Actualmente, la compañía carece de procesos sistematizados, lo que genera dificultades en la asignación y control de citas técnicas, así como en el registro de operaciones. La propuesta busca implementar una solución tecnológica que optimice la gestión de servicios, mejore la experiencia del cliente y permita un control eficiente de las actividades internas. Para ello, se empleará una metodología ágil basada en SCRUM y KANBAN, que facilitará la planificación iterativa y la adaptación a cambios durante el desarrollo. El proyecto contempla la integración de funcionalidades como registro de usuarios, programación de citas y seguimiento de órdenes, contribuyendo a la digitalización de procesos y al fortalecimiento competitivo de la empresa. Se espera que la implementación del portal incremente la eficiencia operativa, reduzca errores en la asignación de recursos y proporcione información confiable para la toma de decisiones estratégicas.

Palabras clave: portal web, MasGas, gasodomésticos, metodología ágil, SCRUM, KANBAN, gestión de servicios, digitalización.

Abstract

This undergraduate project aims to develop a web portal for MasGas, a company that provides services related to gas appliances in Medellín and surrounding areas. Currently, the company lacks standardized processes, which leads to difficulties in scheduling technical appointments and recording operations. The proposed solution seeks to implement a technological platform that optimizes service management, enhances customer experience, and ensures efficient control of internal activities. An agile methodology based on SCRUM and KANBAN will be applied to enable iterative planning and adaptability throughout development. The project includes features such as user registration, appointment scheduling, and order tracking, contributing to process digitalization and strengthening the company's competitiveness. The implementation of the portal is expected to increase operational efficiency, minimize resource allocation errors, and provide reliable information for strategic decision-making.

Key words: web portal, MasGas, gas appliances, agile methodology, SCRUM, KANBAN, service management, digitalization.

Glosario

Gasodoméstico: Artefacto de uso doméstico o industrial que funciona a gas. (Ministerio de Comercio, Industria y Turismo, 2015)

Calentador de agua: Artefacto que calienta el agua mientras esta pasa a través de él o la almacena para calentarla posteriormente. (Ministerio de Comercio, Industria y Turismo, 2015)

Consumidor: Persona natural o jurídica que hace uso de un determinado producto. (Ministerio de Comercio, Industria y Turismo, 2015)

Horno: Gasodoméstico que sirve para preparar asados, repostería, pastelería, etc. (Ministerio de Comercio, Industria y Turismo, 2015)

Base de datos: Sistema de administración de datos relacionales que permite almacenar y gestionar información de manera estructurada. (West, Cai, & Ray, 2024)

HTML: Lenguaje de marcado que define la estructura y el significado del contenido web. (Mozilla, 2025)

CSS: Lenguaje de estilos utilizado para diseñar y dar formato a las páginas web. (Mozilla, 2024)

Tailwind: Framework de CSS que escanea archivos HTML para facilitar el diseño de páginas web. (Tailwindcss, s.f.)

JavaScript: Lenguaje de programación ligero, interpretado y orientado a objetos, utilizado para crear scripts dinámicos en la web. (Mozilla, 2024)

React: Biblioteca de JavaScript para construir interfaces de usuario mediante componentes reutilizables. (React, s.f.)

Node.js: Entorno de ejecución de JavaScript que permite crear servidores y aplicaciones web. (Nodejs, s.f.)

Express.js: Framework minimalista y flexible para aplicaciones web y móviles en Node.js. (Express, s.f.)

API REST: Conjunto de reglas y protocolos que permite la comunicación entre aplicaciones siguiendo principios de diseño REST. (Red Hat, 2023)

Git: Sistema de control de versiones distribuido para gestionar y rastrear cambios en proyectos de software. (Git, s.f.)

GitHub: Plataforma en la nube para almacenar, compartir y colaborar en proyectos de código. (GitHub, s.f.)

Azure: Plataforma de nube pública de Microsoft que ofrece servicios de infraestructura y plataforma. (¿Cómo funciona Azure?, 2025)

SCRUM: Metodología ágil basada en buenas prácticas para trabajar en equipo y obtener mejores resultados en proyectos. (Schwaber & Sutherland, 2020)

Sprint: Periodo corto en SCRUM donde se completan tareas específicas del proyecto. (Schwaber & Sutherland, 2020)

Backlog: Lista priorizada de tareas pendientes en un proyecto ágil. (Cohn, 2010)

Kanban: Método de gestión visual del flujo de trabajo basado en principios Lean, que utiliza tableros para optimizar la eficiencia y fomentar la mejora continua. (Naydenov, s.f.)

SQLite: Sistema de gestión de bases de datos relacional, ligero y embebido, que almacena datos en archivos locales sin necesidad de servidor. (SQLite, s.f.)

Apidog: Plataforma para diseño, prueba y documentación de APIs, que facilita la colaboración y gestión de interfaces de programación. (Apidog, s.f.)

Introducción

El presente trabajo de grado aborda la problemática de la sistematización y optimización de los procesos operativos en la empresa MasGas, dedicada a la prestación de servicios de mantenimiento, reparación, venta e instalación de gasodomésticos en Medellín y el Valle de Aburrá. Actualmente, la gestión de servicios se realiza de manera manual, lo que genera dificultades en el registro, asignación y control de citas técnicas, así como en la trazabilidad de las actividades realizadas. Esta situación limita el crecimiento del emprendimiento y afecta la calidad del servicio ofrecido a los clientes.

La propuesta consiste en el desarrollo e implementación de un sistema de información web orientado a digitalizar y automatizar la gestión de servicios, facilitando el acceso a la información, el control eficiente de las operaciones y la mejora en la experiencia del usuario. Para el desarrollo del proyecto se emplean metodologías ágiles, específicamente SCRUM y KANBAN, que permiten una planificación iterativa y la adaptación continua durante el proceso. Se espera que la solución tecnológica contribuya al fortalecimiento competitivo de MasGas, incrementando la eficiencia operativa y proporcionando herramientas para la toma de decisiones estratégicas.

1. Planteamiento del problema

1.1 Descripción

MasGas es un emprendimiento personal que opera en la ciudad de Medellín y sus alrededores prestando servicios de mantenimiento, reparación, venta e instalación de gasodomésticos. Este es un emprendimiento que no cuenta con ningún tipo de sistematización y toda su operación se realiza a lápiz y papel, y en algunos casos, no se lleva registro alguno de sus operaciones. No se cuenta con un sistema estandarizado que permita registrar la asignación de citas a los respectivos técnicos generando un desbalance al no controlar la cantidad de citas asociadas a los mismos, dificultando la posterior consulta de estas y las actividades y notas relacionados en caso de ser requerido. También permite que se omitan registros de servicios sin que haya evidencia de ello. Todo lo antes mencionado se debe principalmente a 3 causas, la primera de ellas es la falta de capital de inversión, la segunda es la falta de personal, ya que sólo se cuenta con una (1) sola persona que es quién realiza toda la operación y la tercera es el poco tiempo del emprendimiento, que lleva aproximadamente un (1) año operando de manera formal. Esto tiene un impacto negativo ya que no se cuenta con una gestión correcta de los trabajos y/o servicios suministrados al cliente lo cual conlleva a un mal servicio postventa al no tener un registro histórico preciso que permita hacer una trazabilidad cronológica y detallada de los trabajos y/o servicios prestados por MasGas relacionados con los gasodomésticos.

No hacer una transición a esta sistematización y mejoría de la operación podría llevar al cese de operaciones por parte de la empresa ya que no sería rentable debido al bajo flujo de clientes y su fidelización. Adicional a lo antes mencionado, si pensamos en el crecimiento

que puede llegar a tener la empresa, el sistema nos serviría de ayuda para gestionar un gran número de clientes y personal técnico.

1.2 Formulación

¿Cómo gestionar la prestación de servicios y/o venta de gasodomésticos por parte de la compañía MasGas de una manera eficiente, detallada, cronológica y persistente en el tiempo a través de un sistema de información web?

2. Justificación

El proyecto es importante porque ayudará a balancear de forma eficiente y equitativa la carga laboral sobre los técnicos, además de contar con un sistema confiable para la consulta de los servicios realizados por los técnicos y las ventas de los equipos. La utilidad del proyecto radica en que va a permitir que la empresa MasGas agilice y optimice el sistema de asignación de citas. A su vez que brindará la posibilidad de acceder de forma rápida, eficiente y confiable al histórico de los servicios prestados por los técnicos. Hoy en día, MasGas no cuenta con un sistema digital en la nube que permita gestionar los datos. Este proyecto le aportará el aspecto tecnológico cómo se ha mencionado anteriormente. Para este pequeño emprendimiento, la novedad de este proyecto radica en el mismo desarrollo del sistema web para poder gestionar su operación ya que actualmente cómo se ha mencionado antes, todo se realiza de una manera muy rudimentaria contando con registros a lápiz y papel por la falta de capital económico para la operación de la compañía.

3. Objetivos

3.1 Objetivo general

Desarrollar un sistema de información web que permita gestionar los servicios de venta, reparación, mantenimiento e instalación de gasodomésticos prestados por la compañía MasGas en el Valle de Aburrá y sus alrededores.

3.2 Objetivos específicos

Identificar los requisitos (funcionales y no funcionales) del proyecto.

Diseñar un sistema de información web que permita gestionar los servicios de venta, reparación, mantenimiento e instalación de gasodomésticos prestados por la compañía MasGas.

Implementar las funcionalidades de los módulos diseñados de acuerdo a los requisitos funcionales.

4. Marco teórico

MasGas es un microemprendimiento que opera de manera informal en la ciudad de Medellín, el Valle de Aburrá y sus alrededores desde el año 2020 liderado por una sola persona que antes trabajó cómo técnico en diferentes empresas de la misma ciudad. A comienzos del año 2024 se le dio un toque más formal al establecer su nombre, su logo y asociando una línea de WhatsApp para atención al cliente. Su actividad principal se enfoca en la prestación de servicios de mantenimiento, reparación, venta e instalación de gasodomésticos, específicamente cubiertas, hornos, calentadores de agua, calentadores para turco y calentadores para jacuzzi. Su operación es dirigida y ejecutada por una única persona que se encarga de recibir las solicitudes y prestar el servicio requerido.

5. Metodología

5.1 Tipo de proyecto

Desarrollo tecnológico aplicado para la implementación de un sitio web en MasGas**,** orientado a resolver una necesidad concreta mediante la construcción y despliegue de una solución software funcional. No busca generar teoría, sino resultados operativos medibles (uso, desempeño y satisfacción).

5.2 Método

5.2.1. SCRUM

Scrum es una metodología ágil, de las más utilizadas, puede ser aplicada tanto en desarrollo de software como en otras especialidades. Consiste en un conjunto de buenas prácticas para trabajar en equipo y obtener un proyecto con un mejor resultado.

Esta metodología cuenta con cinco (5) fases (Fases y procesos de Scum, s.f.), las cuales se definen a continuación:

Fase de Inicio: La primera fase de la metodología Scrum, se encarga de estudiar y analizar, se define la visión del proyecto, se determina que es lo que el usuario final solicita, se establece los roles y los integrantes del equipo Scrum, los cuales estarán encargados de elaborar los entregable de cada uno de los sprint, para este caso no se cuenta con equipo ya que el proyecto será desarrollado por una sola persona; se desarrollan las épicas y se ordenan por prioridades (Backlog Priorizado del Producto). Finalmente, se realiza la planificación del lanzamiento. (Fases y procesos de Scum, s.f.)

Fase de Planeación y Estimación: Para realizar un buen control de todos los recursos se realiza una correcta planificación de las tareas del proyecto. Se elaboran las historias de usuario, es necesario hacer una estimación inicial de esfuerzo de las historias de usuario, se identificas las tareas y se estima el esfuerzo inicial de dichas tareas a realizar en cada sprint, finalmente, se desarrolla el Sprint Backlog, que compone todas las tareas que se deben completar en cada sprint. (Fases y procesos de Scum, s.f.)

Fase de Implementación: En la presente fase es donde se ejecutan las tareas y actividades recopiladas de la fase anterior. Se crean los entregables y finalmente se revisa el backlog priorizado, para analizar, realizar cambios y actualizaciones. (Fases y procesos de Scum, s.f.)

Fase de Revisión y Retrospectiva: En la cuarta fase de la metodología, se busca la revisión de los entregables, además de los procesos que se realizaron durante el sprint, con el fin de establecer mejoras. (Fases y procesos de Scum, s.f.)

Fase de Lanzamiento: Esta última fase de Scrum, se busca entregar el resultado final al cliente, luego de la previa aprobación del producto. Así mismo, se busca retroalimentación para próximos proyectos, con el objetivo de mejorar la eficiencia del equipo Scrum. (Fases y procesos de Scum, s.f.)

Para este proyecto se implementará los sprints para controlar avances semanalmente y poder corregir posibles errores o realizar mejoras de ser necesario.

5.2.2. KANBAN

Kanban es un método de gestión del flujo de trabajo basado en principios Lean, diseñado para ayudar a las organizaciones a visualizar, gestionar y mejorar sus procesos de trabajo. Su objetivo principal es optimizar la eficiencia y fomentar la mejora continua en la entrega de servicios o productos. (Naydenov, s.f.)

Características principales de Kanban

Visualización del trabajo: Kanban utiliza tableros para representar visualmente las tareas y su estado dentro del flujo de trabajo, lo que proporciona claridad y transparencia al equipo. (Naydenov, s.f.)

Gestión del flujo de trabajo: Permite a los equipos controlar y mejorar el flujo de trabajo, identificando cuellos de botella y áreas de mejora para optimizar la eficiencia.

Mejora continua: Fomenta una cultura de evaluación y ajuste constante de los procesos para lograr una entrega de valor más efectiva y eficiente. (Naydenov, s.f.)

Para este proyecto se tendrá en consideración el uso de tableros Kanban para el control de los avances del proyecto y su facilidad para la visualización teniendo en cuenta que es un proyecto pequeño y que sólo se cuenta con una persona para todo su desarrollo

y además permite:

Visualizar el flujo de trabajo: Representa las diferentes etapas del proceso mediante columnas (por ejemplo, "Por hacer", "En proceso", "Hecho"), facilitando la comprensión del estado de cada tarea. (Naydenov, s.f.)

Identificar cuellos de botella: Al observar la acumulación de tareas en ciertas columnas, el equipo puede detectar rápidamente problemas en el flujo de trabajo y tomar medidas correctivas. (Naydenov, s.f.)

Mejorar la eficiencia: Al limitar el trabajo en curso y enfocarse en completar tareas antes de iniciar nuevas, se reduce el tiempo de ciclo y se mejora la entrega de valor al cliente. (Naydenov, s.f.)

6. Resultados

El desarrollo del sistema de información web para la empresa MasGas se centró en la implementación y validación de la capa backend, utilizando una arquitectura modular basada en los módulos services, controllers y routes. La base de datos utilizada fue SQLite, en la que se crearon siete tablas principales: users, roles, services, locations, users_by_type, services_by_type y service_type.

Se implementaron y probaron cinco endpoints fundamentales para la gestión de los datos:

6.1. Crear registro (POST): Permite la creación de nuevos registros en las tablas correspondientes, validando los datos ingresados y asegurando su persistencia en la base de datos.

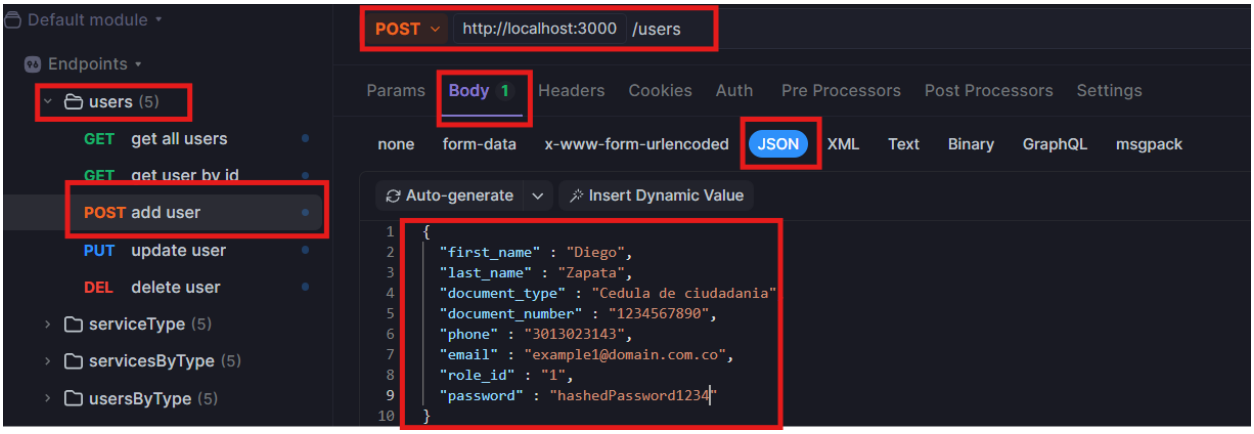


Ilustración 1 Endpoint: /users(POST)

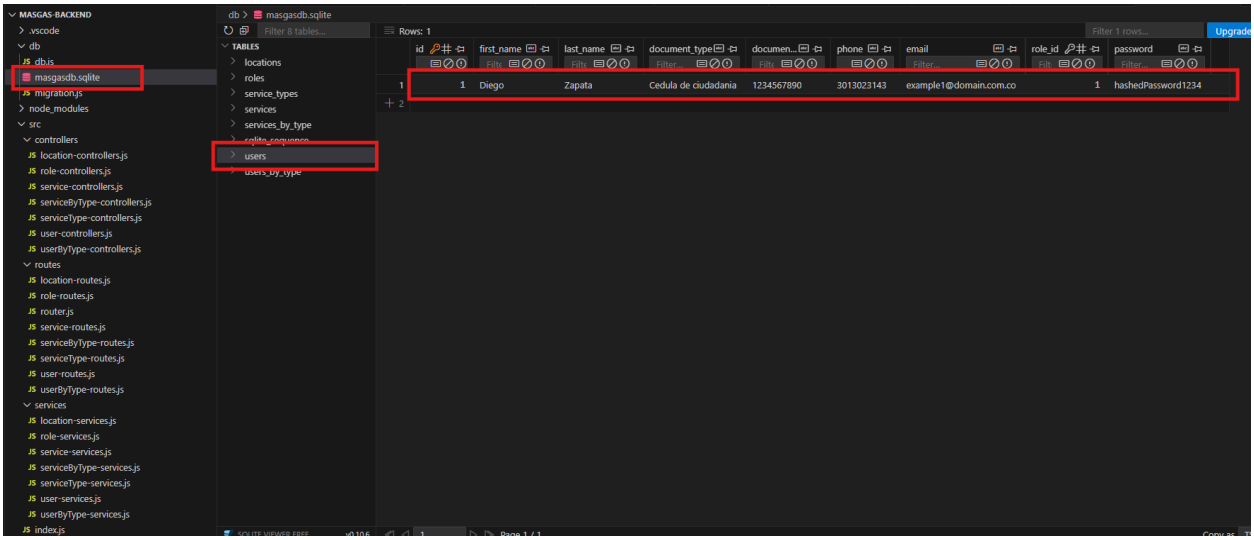


Ilustración 2 Registro agregado en la tabla users en la base de datos

6.2. Leer todos los registros (GET): Recupera todos los registros de una tabla específica, facilitando la visualización y consulta de la información almacenada.

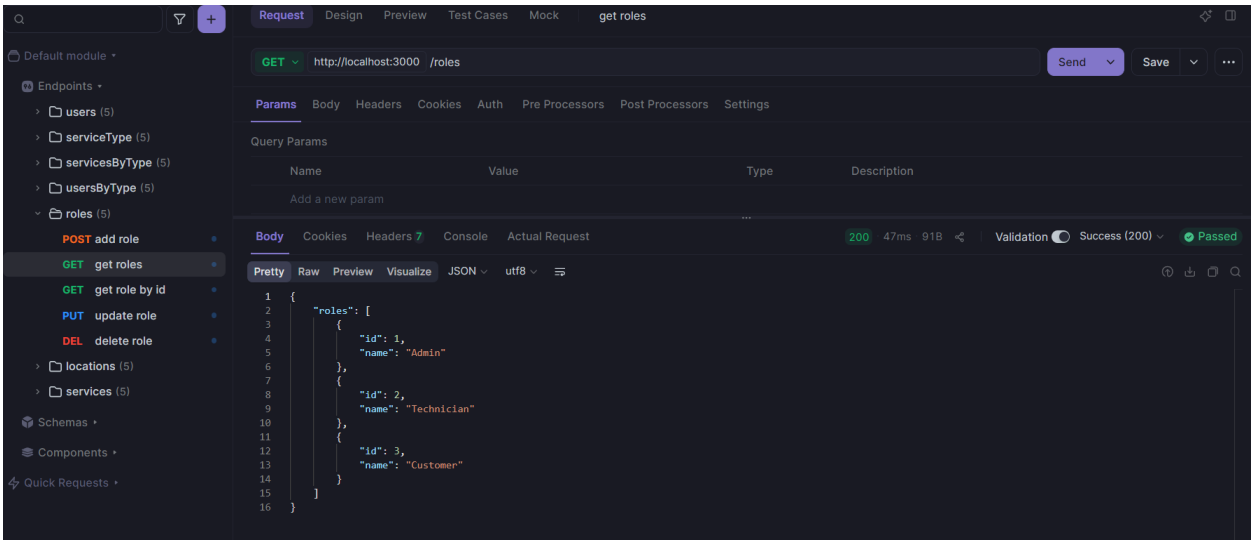


Ilustración 3 endpoint: /roles (GET)

6.3.Leer un registro por ID (GET): Permite consultar un registro específico utilizando su identificador único, mejorando la trazabilidad y el acceso a información puntual.

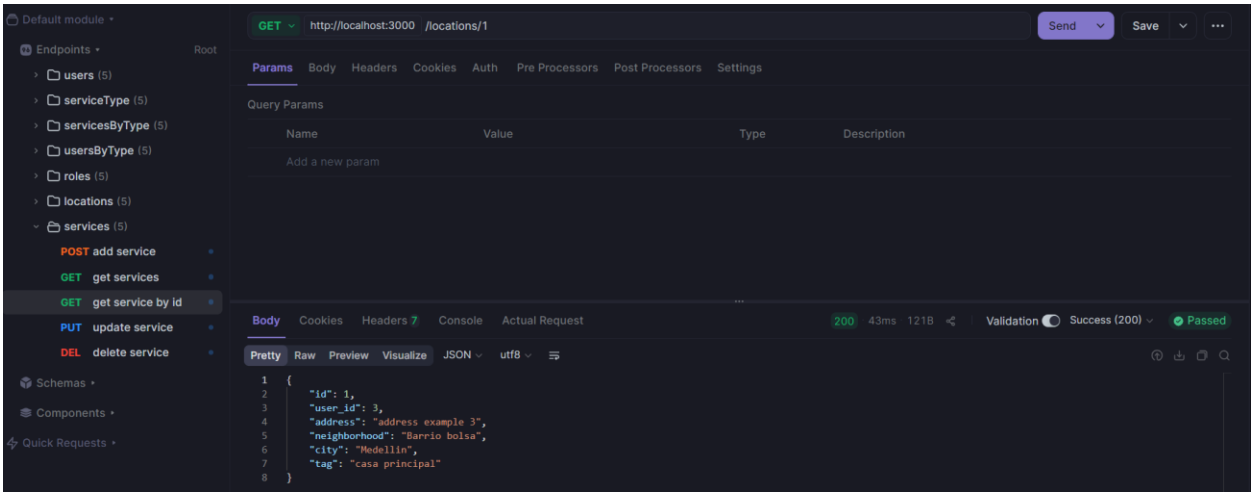


Ilustración 4 endpoint: /services/:id (GET)

6.4.Modificar un registro completo por ID (PUT): Facilita la actualización de la información de un registro existente, garantizando la integridad y consistencia de los datos.

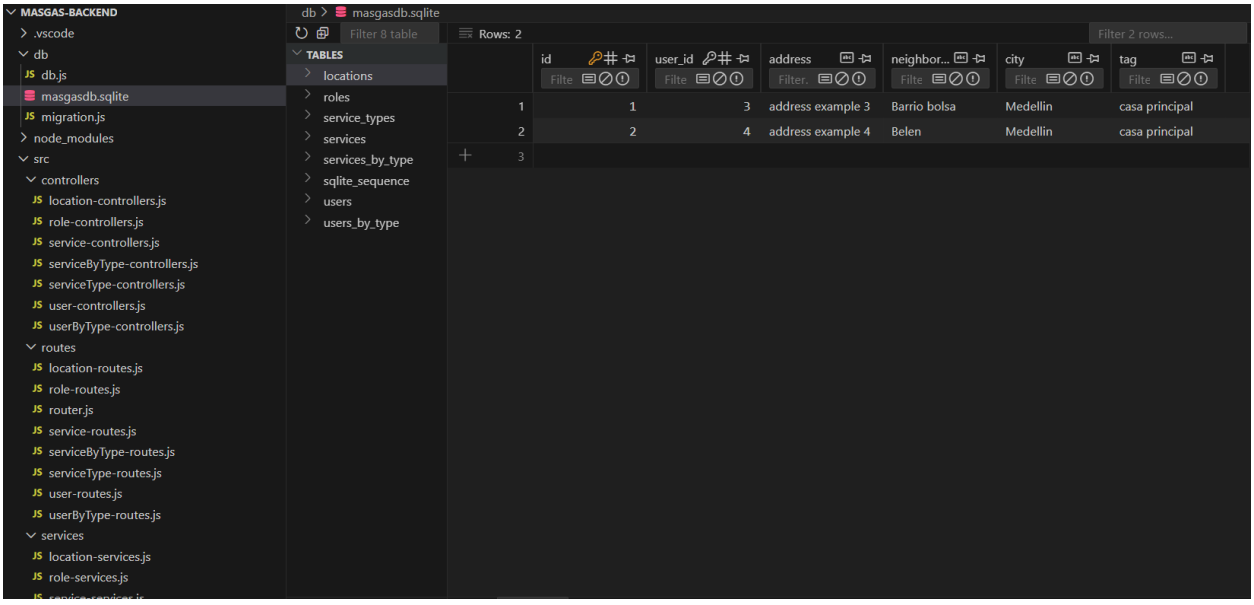


Ilustración 5 Registros de la tabla locations en la base de datos antes de actualizar

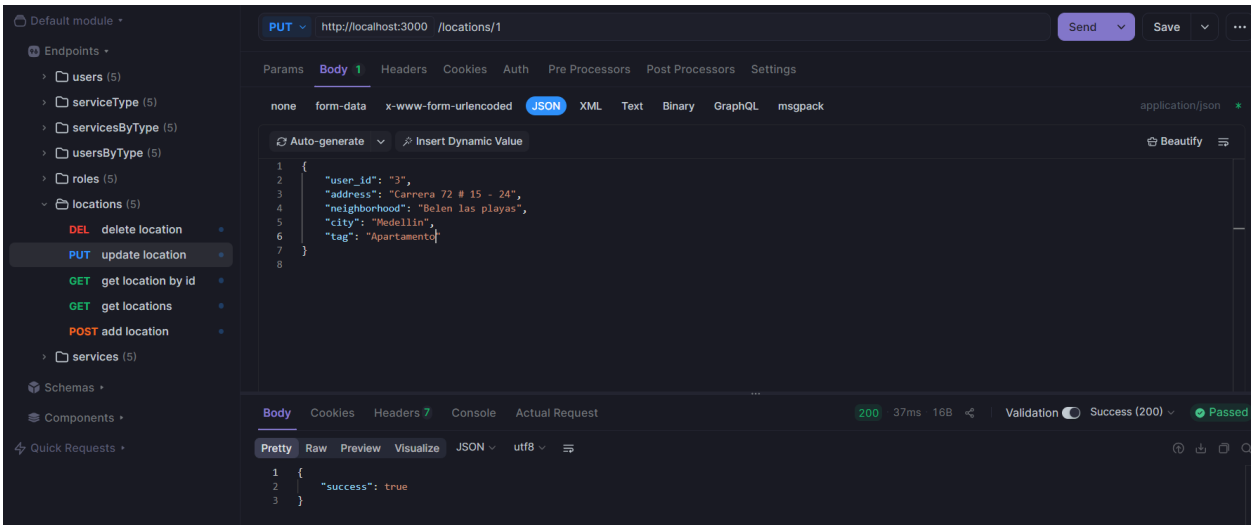


Ilustración 6 endpoint: /locations/:id (PUT)

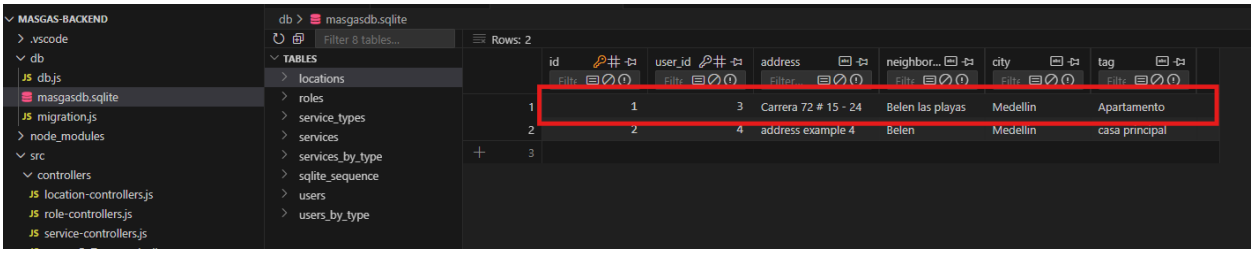


Ilustración 7 Actualizacion en la base de datos para registro 1 de la tabla locations

6.5. Eliminar un registro por ID (DELETE): Permite la eliminación segura de registros, asegurando que los datos obsoletos o incorrectos sean removidos de la base de datos.

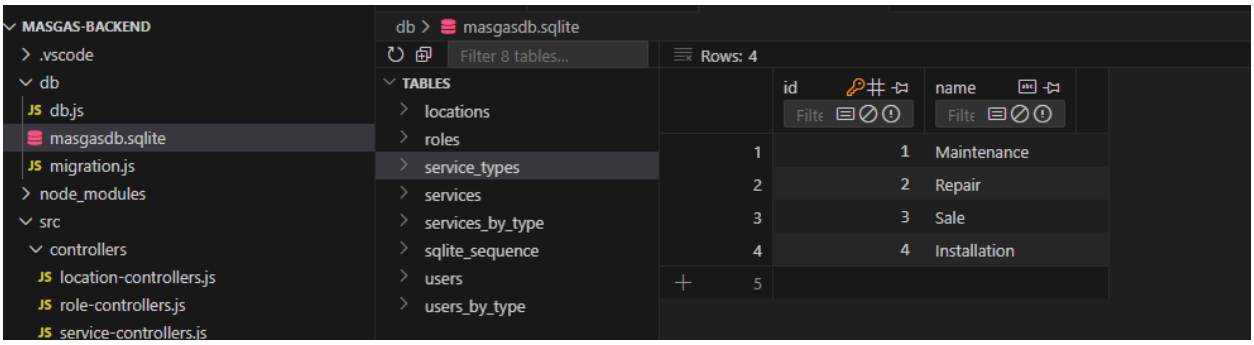


Ilustración 8 Registros en la tabla service_types antes de eliminar

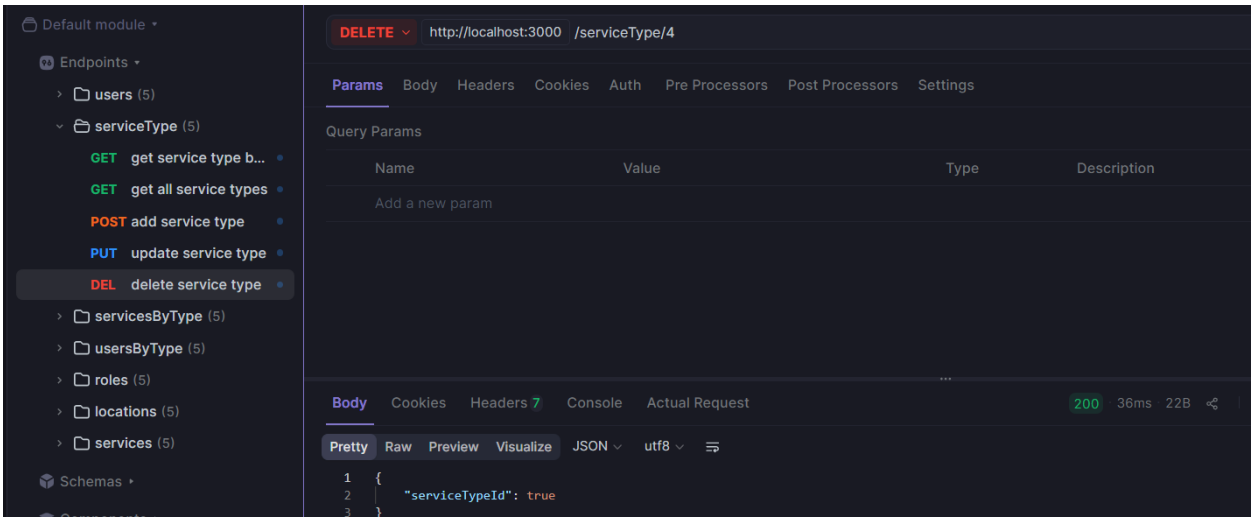


Ilustración 9 endpoint: /serviceType:id (DELETE)

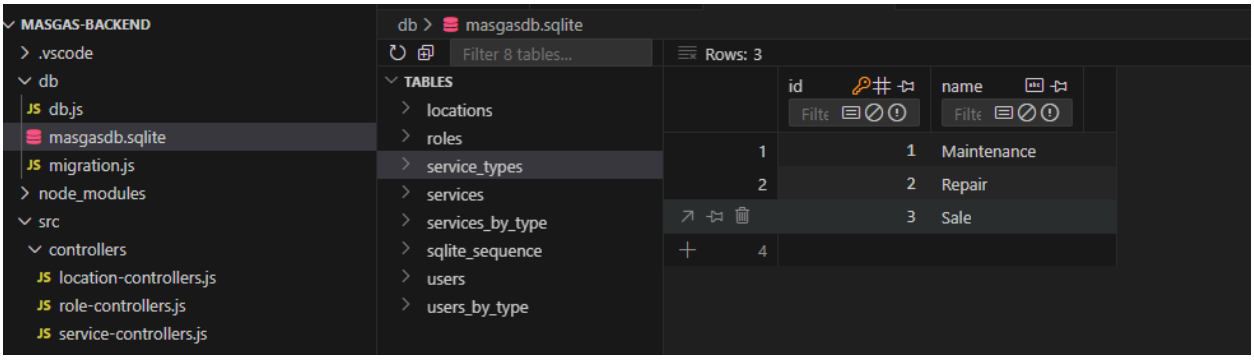


Ilustración 10 Registro eliminado de la tabla service_types

Cada uno de estos endpoints fue desarrollado siguiendo buenas prácticas de programación y probado exhaustivamente utilizando la plataforma Apidog, lo que permitió validar el correcto funcionamiento de las APIs y la interacción con la base de datos SQLite. Las pruebas incluyeron la verificación de respuestas, manejo de errores y validación de los flujos de datos entre el servidor y la base de datos.

Cabe destacar que, debido a limitaciones de tiempo, no se desarrolló la interfaz frontend, por lo que se obvió el uso de tecnologías como HTML, CSS, Tailwind y React mencionadas en el marco teórico. Sin embargo, JavaScript sí fue utilizado en el backend, específicamente con Node.js y Express.js para la lógica de negocio y la gestión de los endpoints.

La estructura modular y la correcta implementación de los endpoints permiten una gestión eficiente de la información y preparan el sistema para futuras integraciones con una interfaz de usuario. La

utilización de Apidog y SQLite contribuyó significativamente a la calidad y confiabilidad del sistema.

7. Conclusiones

7.1 Modernización y sistematización de procesos: La implementación del sistema de información web para MasGas representa un avance significativo en la modernización de los procesos internos, permitiendo la gestión eficiente de servicios de venta, reparación, mantenimiento e instalación de gasodomésticos. La transición de registros manuales a una plataforma digital mejora la trazabilidad y la calidad del servicio postventa, alineándose con las necesidades actuales de la empresa y su potencial crecimiento (Galindo Melo et al., 2022).

7.2 Arquitectura modular y escalabilidad: El diseño basado en módulos (services, controllers y routes) facilita la escalabilidad y el mantenimiento del sistema, permitiendo futuras integraciones, como la incorporación de una interfaz frontend. Esta estructura modular es coherente con las mejores prácticas en desarrollo de software y prepara la solución para adaptaciones tecnológicas posteriores (Red Hat, 2023).

7.3 Gestión robusta de datos con SQLite: La utilización de SQLite y la definición de siete tablas principales aseguran una gestión robusta y persistente de la información, permitiendo operaciones CRUD (crear, leer, actualizar y eliminar) sobre los datos de usuarios, roles, servicios y ubicaciones. Esto contribuye a la integridad y consistencia de los datos, aspectos fundamentales en sistemas de información (West, Cai & Ray, 2024).

7.4 Validación exhaustiva de endpoints y calidad del backend: La implementación y prueba de los cinco endpoints fundamentales mediante la plataforma Apidog garantizan la correcta interacción entre el servidor y la base de datos, así como el manejo adecuado de errores y la validación de flujos de datos. Este enfoque asegura la confiabilidad y calidad del backend, cumpliendo con los requisitos funcionales del proyecto.

7.5 Limitaciones y oportunidades de mejora: Aunque el proyecto se centró en el desarrollo del backend, la ausencia de una interfaz frontend limita la experiencia del usuario final. Sin embargo, la arquitectura propuesta permite la integración futura de tecnologías como React y Tailwind, lo que representa una oportunidad clara para ampliar la funcionalidad y accesibilidad del sistema.

7.6 Contribución académica y profesional: El desarrollo del sistema bajo metodologías ágiles como Scrum y Kanban, y el uso de herramientas modernas (Node.js, Express.js,

Apidog, GitHub), evidencia la aplicación de conocimientos técnicos y metodológicos adquiridos durante la formación como tecnólogo en desarrollo de software. El proyecto constituye una contribución relevante tanto para la empresa MasGas como para el campo académico, al demostrar la viabilidad de soluciones tecnológicas en microempresas.

8. Recomendaciones

8.1 **Desarrollar la interfaz frontend**
Se recomienda priorizar la implementación de la interfaz de usuario utilizando tecnologías como React, HTML, CSS y Tailwind. Esto permitirá mejorar la experiencia del usuario final, facilitar la interacción con el sistema y aprovechar la arquitectura modular ya establecida en el backend.

8.2 **Implementar mecanismos de seguridad y autenticación**
Es fundamental incorporar controles de acceso y autenticación robustos para proteger los datos sensibles de la empresa y los usuarios. Se sugiere el uso de estándares como JWT (JSON Web Tokens) y cifrado de contraseñas en la base de datos.

8.3 **Realizar pruebas de integración y usabilidad**
Además de las pruebas unitarias y de endpoints, se recomienda efectuar pruebas de integración y usabilidad una vez se disponga del frontend, para asegurar la correcta interacción entre todas las capas del sistema y validar la experiencia del usuario.

8.4 **Documentar exhaustivamente el sistema**
Se aconseja mantener una documentación técnica detallada sobre la arquitectura, endpoints, flujos de datos y procedimientos de despliegue. Esto facilitará el mantenimiento, la capacitación de futuros desarrolladores y la escalabilidad del proyecto.

8.5 **Planificar la migración a una base de datos más robusta**
Aunque SQLite es adecuada para el prototipo y primeras fases, se recomienda evaluar la migración a sistemas de gestión de bases de datos más robustos como PostgreSQL o SQL Server, especialmente si se prevé un crecimiento significativo en el volumen de datos y usuarios.

8.6 **Monitorear y optimizar el rendimiento**
Es importante implementar herramientas de monitoreo y análisis de rendimiento para identificar posibles cuellos de botella y optimizar el uso de recursos, garantizando la estabilidad y eficiencia del sistema en producción.

8.7 **Fomentar la retroalimentación continua**
Se recomienda establecer canales de comunicación con los usuarios finales y técnicos para recibir retroalimentación sobre el funcionamiento del sistema, permitiendo ajustes y mejoras continuas en función de las necesidades reales de la empresa.

9. Referencias bibliográficas

Aguillón Capacho, C. O. (2021). Diseño e implementación de un sistema web y una aplicación móvil para la gestión de pedidos y entregas de domicilios. Recuperado de https://repository.libertadores.edu.co/server/api/core/bitstreams/1f9e9136-8471-4390-bae1-4e0f875d6dad/content

Fases y procesos de SCRUM. (s.f.). Scrum study. Recuperado de https://www-scrumstudy-com.translate.goog/whyscrum/scrum-phases-and-processes?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Galindo Melo, F. C., Linares Ospina, Y. F., & Perdomo Pardo, L. S. (2022). Desarrollo de un sistema de información para la microempresa Lavamar. Universidad El Bosque. Recuperado de https://repositorio.unbosque.edu.co/server/api/core/bitstreams/75f079dd-525d-4474-ae17-fd9eee640448/content

Git. (s.f.). Recuperado de https://git-scm.com/

GitHub. (s.f.). Recuperado de https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

Hernández, H. A., Cruz Gil, Y. L., Puentes Saavedra, M. D., & Mendoza Patiño, D. E. (2021). Diseño de un sistema de gestión de inventarios para un almacén. Revista de investigaciones Universidad del Quindío. Recuperado de https://revistas.uniquindio.edu.co/ojs/index.php/riuq/article/view/562/752

HTML: Lenguaje de etiquetas de hipertexto. (2025, 11 de abril). MDN Web Docs. Recuperado de https://developer.mozilla.org/es/docs/Web/HTML

JavaScript. (2024, 17 de diciembre). MDN Web Docs. Recuperado de https://developer.mozilla.org/es/docs/conflicting/Web/JavaScript

Kanban: ¿Qué es kanban? Explicación para principiantes. Naydenov, P. (s.f.). Businessmap. Recuperado de https://businessmap.io/es/recursos-de-kanban/primeros-pasos/que-es-kanban

Nodejs. (s.f.). Recuperado de https://nodejs.org/es

Olivos Varcarcel, J. S., & Castillo Palacio, M. S. (2021, abril). Automatización de procesos de pedidos en restaurantes. Universidad Distrital Francisco José de Caldas. Recuperado de http://hdl.handle.net/11349/27777

React. (s.f.). Recuperado de https://es.react.dev/

Red Hat. (2023, 31 de julio). ¿Qué es una API de REST? Recuperado de https://www.redhat.com/es/topics/api/what-is-a-rest-api

Tailwindcss. (s.f.). Recuperado de https://tailwindcss.com/

Tibaduiza Álvarez, D. F., Gómez García, C., & Camargo, M. F. (2021). Aplicación móvil para la gestión de PQRS en la Superintendencia de Servicios Públicos Domiciliarios de Cali. Recuperado de https://repository.unad.edu.co/bitstream/handle/10596/40362/dftibaduizaa.pdf?sequence=1&isAllowed=y

West, R., Cai, S., & Ray, M. (2024, 9 de abril). Microsoft SQL Server. Recuperado de https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16

¿Como funciona Azure? (2025, 21 de abril). Azure learn. Recuperado de https://learn.microsoft.com/es-es/azure/cloud-adoption-framework/get-started/what-is-azure

CSS. (2024, 17 de diciembre). MDN Web Docs. Recuperado de https://developer.mozilla.org/es/docs/Web/CSS

Express. (s.f.). Recuperado de https://expressjs.com/

Ministerio de Comercio, Industria y Turismo. (2015, 6 de marzo). Resolución de 680 del 2015. Recuperado de https://www.mincit.gov.co/temas-interes/reglamentos-tecnicos/ministerio-de-comercio-industria-y-turismo/resolucion-de-680-del-2015.aspx

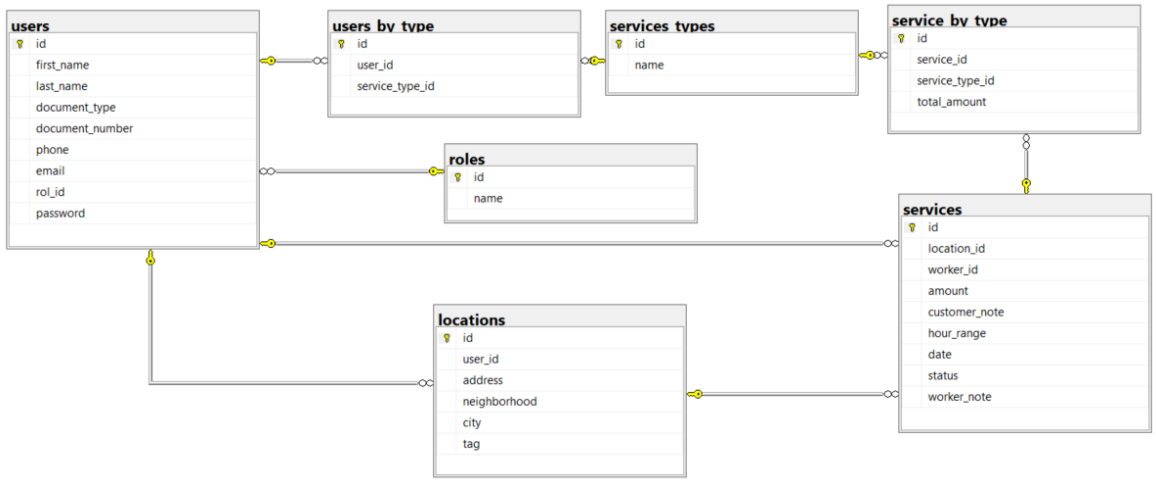
11. Anexos

Anexo A. Tabla de historias de usuario

ID	Historia de Usuario	Prioridad	Rol
HU-01	Como cliente, quiero registrarme en el sistema para acceder a la solicitud y seguimiento de servicios.	Alta	Cliente
HU-02	Como cliente, quiero iniciar sesión con mis credenciales para acceder a mi cuenta y ver mis servicios.	Alta	Cliente
HU-03	Como cliente, quiero solicitar un servicio (venta, instalación, mantenimiento o reparación) indicando la fecha y ubicación, para recibir atención.	Alta	Cliente
HU-04	Como cliente, quiero consultar el estado de mis solicitudes de servicio para saber si están pendientes, en proceso o finalizadas.	Alta	Cliente
HU-05	Como cliente, quiero evaluar la atención del técnico después del servicio para contribuir a la mejora del servicio.	Baja	Cliente
HU-06	Como técnico, quiero visualizar las solicitudes de servicio asignadas para organizar mis visitas y tiempos de atención.	Alta	Técnico
HU-07	Como técnico, quiero actualizar el estado del servicio (en camino, en proceso, finalizado) para mantener informado al cliente y al administrativo.	Alta	Técnico
HU-08	Como técnico, quiero registrar observaciones o recomendaciones al finalizar el servicio para mantener trazabilidad del trabajo realizado.	Media	Técnico
HU-09	Como técnico, quiero adjuntar evidencias fotográficas del servicio realizado para respaldar la atención brindada.	Media	Técnico
HU-10	Como administrativo, quiero visualizar todas las solicitudes de servicio para asignarlas a los técnicos disponibles según zona y tipo de	Alta	Administrativo
HU-11	Como administrativo, quiero registrar proveedores externos asociados a la venta de gasodomésticos para gestionar las solicitudes de	Alta	Administrativo
HU-12	Como administrativo, quiero generar reportes de servicios atendidos, pendientes y calificaciones de clientes para analizar el desempeño general.	Alta	Administrativo
HU-13	Como administrativo, quiero registrar y actualizar la información de técnicos (nombre, zona de cobertura, especialidad) para gestionar las asignaciones.	Media	Administrativo
HU-14	Como administrativo, quiero recibir alertas o notificaciones cuando existan servicios sin asignar o atrasados para garantizar la atención oportuna.	Alta	Administrativo
HU-15	Como administrativo, quiero registrar los pagos realizados por los clientes y asociarlos a cada servicio para mantener el control financiero del sistema.	Media	Administrativo

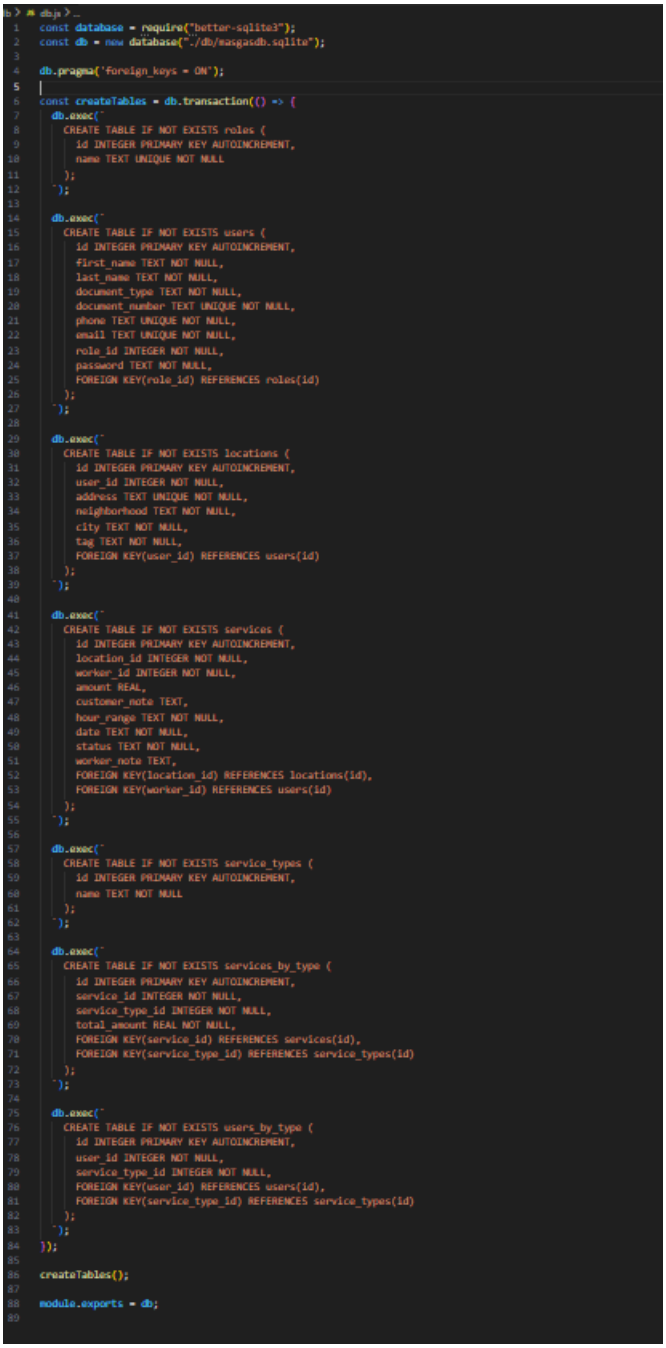
Anexo 1 Historias de usuario

Anexo B. Modelo Entidad relación



Anexo 2 Modelo entidad relación

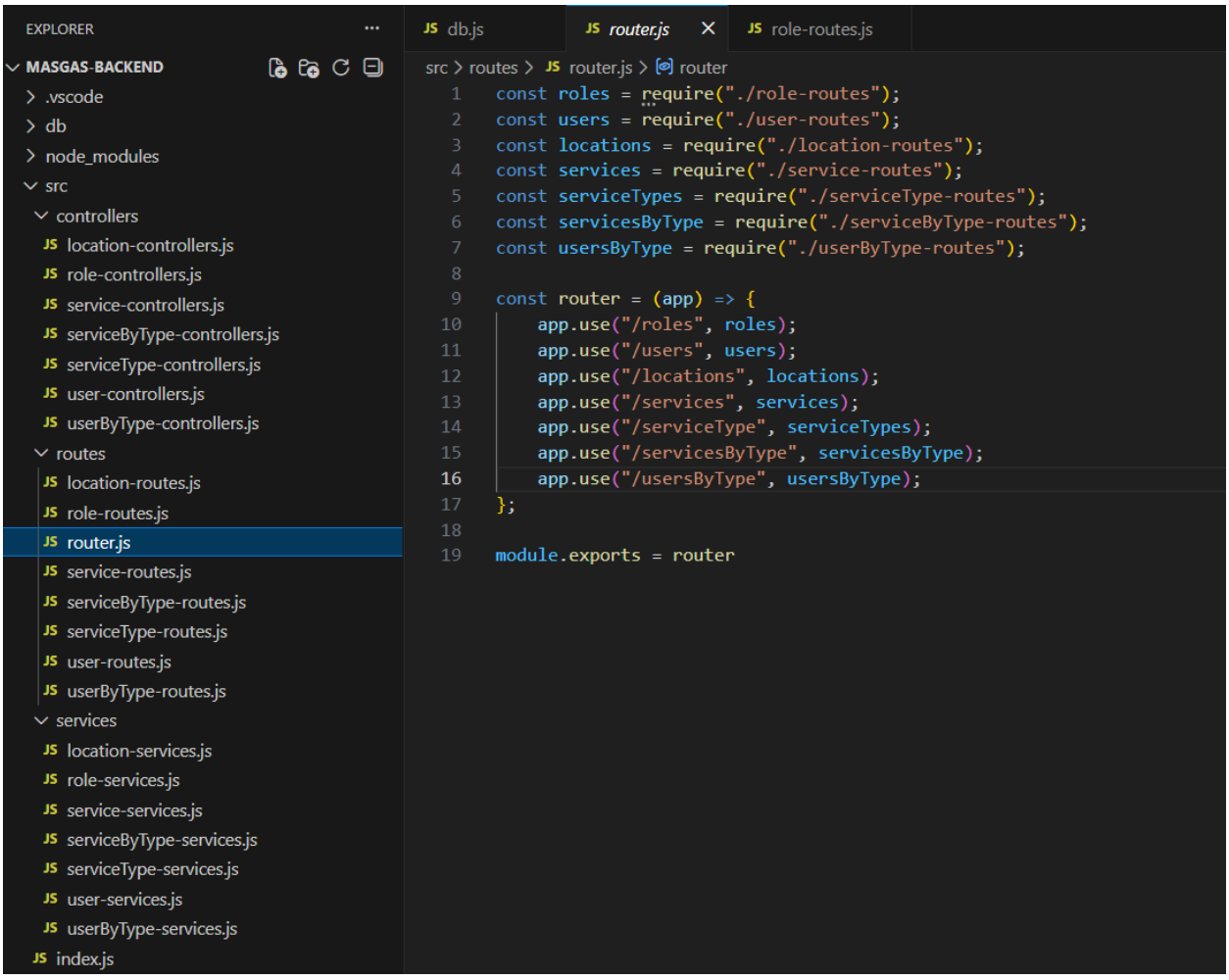
Anexo C. Creación de la base de datos



```
const database = require("better-sqlite3");
const db = new database("./db/masgasdb.sqlite");

db.pragma('foreign_keys = ON');

const createTables = db.transaction(() => {
  db.exec(`
    CREATE TABLE IF NOT EXISTS roles (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT UNIQUE NOT NULL
    );
  `);

  db.exec(`
    CREATE TABLE IF NOT EXISTS users (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      first_name TEXT NOT NULL,
      last_name TEXT NOT NULL,
      document_type TEXT NOT NULL,
      document_number TEXT UNIQUE NOT NULL,
      phone TEXT UNIQUE NOT NULL,
      email TEXT UNIQUE NOT NULL,
      role_id INTEGER NOT NULL,
      password TEXT NOT NULL,
      FOREIGN KEY(role_id) REFERENCES roles(id)
    );
  `);

  db.exec(`
    CREATE TABLE IF NOT EXISTS locations (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      user_id INTEGER NOT NULL,
      address TEXT UNIQUE NOT NULL,
      neighborhood TEXT NOT NULL,
      city TEXT NOT NULL,
      tag TEXT NOT NULL,
      FOREIGN KEY(user_id) REFERENCES users(id)
    );
  `);

  db.exec(`
    CREATE TABLE IF NOT EXISTS services (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      location_id INTEGER NOT NULL,
      worker_id INTEGER NOT NULL,
      amount REAL,
      customer_note TEXT,
      hour_range TEXT NOT NULL,
      date TEXT NOT NULL,
      status TEXT NOT NULL,
      worker_note TEXT,
      FOREIGN KEY(location_id) REFERENCES locations(id),
      FOREIGN KEY(worker_id) REFERENCES users(id)
    );
  `);

  db.exec(`
    CREATE TABLE IF NOT EXISTS service_types (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL
    );
  `);

  db.exec(`
    CREATE TABLE IF NOT EXISTS services_by_type (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      service_id INTEGER NOT NULL,
      service_type_id INTEGER NOT NULL,
      total_amount REAL NOT NULL,
      FOREIGN KEY(service_id) REFERENCES services(id),
      FOREIGN KEY(service_type_id) REFERENCES service_types(id)
    );
  `);

  db.exec(`
    CREATE TABLE IF NOT EXISTS users_by_type (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      user_id INTEGER NOT NULL,
      service_type_id INTEGER NOT NULL,
      FOREIGN KEY(user_id) REFERENCES users(id),
      FOREIGN KEY(service_type_id) REFERENCES service_types(id)
    );
  `);
});

createTables();

module.exports = db;
```

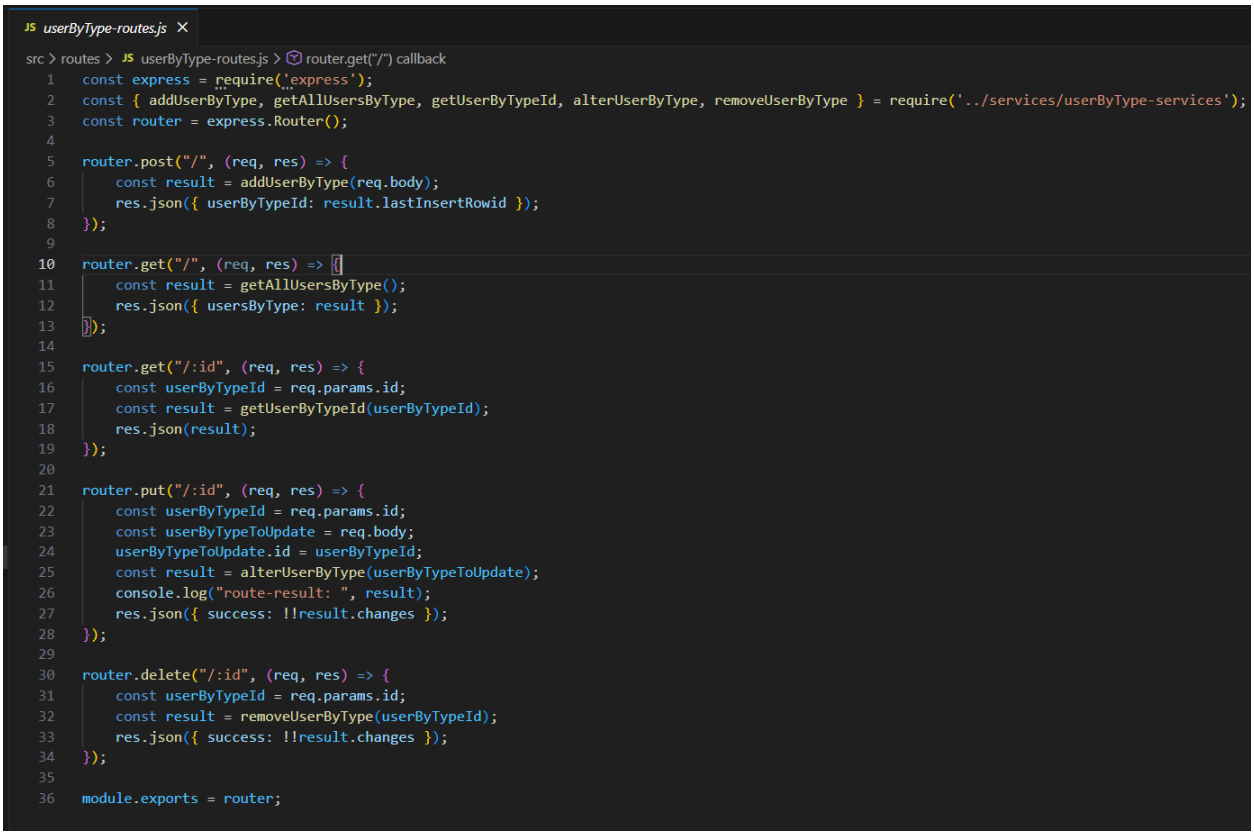
Anexo 3 Sentencias SQL para la creacion de la base de datos y sus tablas

Anexo D. Arquitectura Modular



Anexo 4 Carpetas de arquitectura modular y su contenido

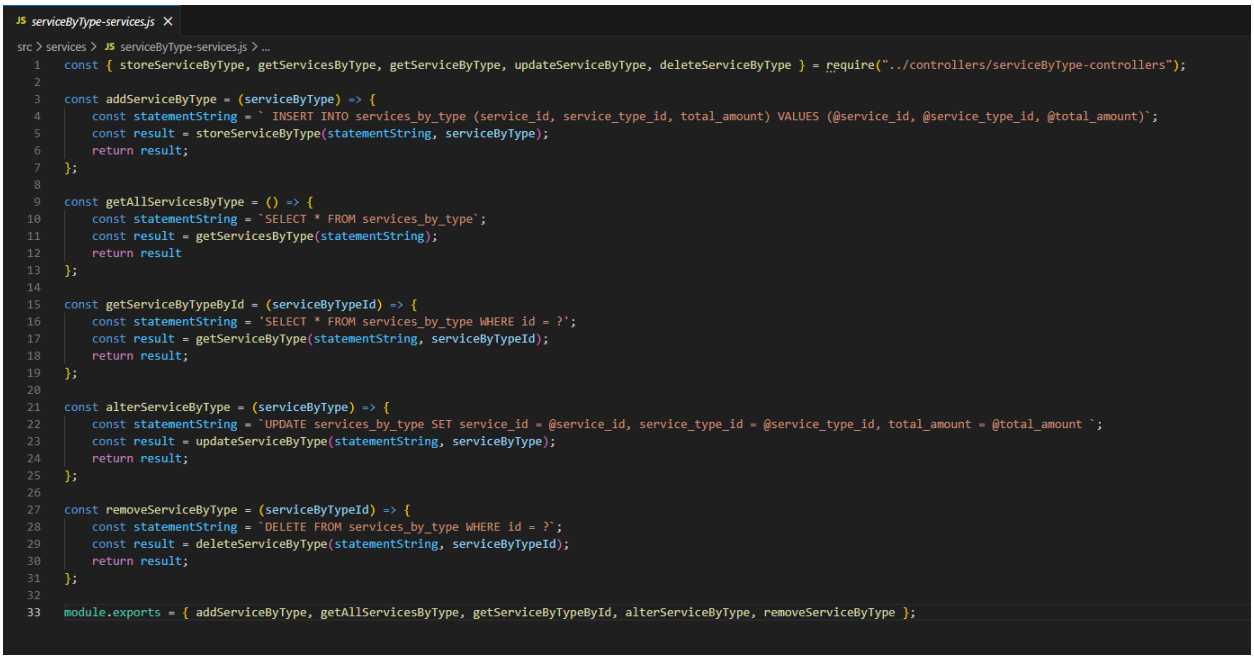
Anexo E. Código módulo routes



```
JS userByType-routes.js ×
src > routes > JS userByType-routes.js > router.get("/") callback
1   const express = require('express');
2   const { addUserByType, getAllUsersByType, getUserByTypeId, alterUserByType, removeUserByType } = require('../services/userByType-services');
3   const router = express.Router();
4
5   router.post("/", (req, res) => {
6       const result = addUserByType(req.body);
7       res.json({ userByTypeId: result.lastInsertRowid });
8   });
9
10  router.get("/", (req, res) => {
11      const result = getAllUsersByType();
12      res.json({ usersByType: result });
13  });
14
15  router.get("/:id", (req, res) => {
16      const userByTypeId = req.params.id;
17      const result = getUserByTypeId(userByTypeId);
18      res.json(result);
19  });
20
21  router.put("/:id", (req, res) => {
22      const userByTypeId = req.params.id;
23      const userByTypeToUpdate = req.body;
24      userByTypeToUpdate.id = userByTypeId;
25      const result = alterUserByType(userByTypeToUpdate);
26      console.log("route-result: ", result);
27      res.json({ success: !!result.changes });
28  });
29
30  router.delete("/:id", (req, res) => {
31      const userByTypeId = req.params.id;
32      const result = removeUserByType(userByTypeId);
33      res.json({ success: !!result.changes });
34  });
35
36  module.exports = router;
```

Anexo 5 Ejemplo modulo userByType-routes

Anexo F. Código módulo services



```
src > services > serviceByType-services.js > ...
1   const { storeServiceByType, getServicesByType, getServiceByType, updateServiceByType, deleteServiceByType } = require("../controllers/serviceByType-controllers");
2
3   const addServiceByType = (serviceByType) => {
4       const statementString = ` INSERT INTO services_by_type (service_id, service_type_id, total_amount) VALUES (@service_id, @service_type_id, @total_amount)`;
5       const result = storeServiceByType(statementString, serviceByType);
6       return result;
7   };
8
9   const getAllServicesByType = () => {
10      const statementString = `SELECT * FROM services_by_type`;
11      const result = getServicesByType(statementString);
12      return result
13  };
14
15  const getServiceByTypeById = (serviceByTypeId) => {
16      const statementString = 'SELECT * FROM services_by_type WHERE id = ?';
17      const result = getServiceByType(statementString, serviceByTypeId);
18      return result;
19  };
20
21  const alterServiceByType = (serviceByType) => {
22      const statementString = `UPDATE services_by_type SET service_id = @service_id, service_type_id = @service_type_id, total_amount = @total_amount `;
23      const result = updateServiceByType(statementString, serviceByType);
24      return result;
25  };
26
27  const removeServiceByType = (serviceByTypeId) => {
28      const statementString = `DELETE FROM services_by_type WHERE id = ?`;
29      const result = deleteServiceByType(statementString, serviceByTypeId);
30      return result;
31  };
32
33  module.exports = { addServiceByType, getAllServicesByType, getServiceByTypeById, alterServiceByType, removeServiceByType };
```

Anexo 6 Ejemplo modulo serviceByType-services

Anexo G. Código modulo controllers

```
JS user-controllers.js ×
src > controllers > JS user-controllers.js > ...
1   const db = require("../../db/db");
2
3   const storeUser = (statement, user) => {
4       try {
5           const stmt = db.prepare(statement);
6           const result = stmt.run(user);
7           return result;
8       } catch (error) {
9           return error.message;
10      }
11  };
12
13  const getUsers = (statement) => {
14      try {
15          const stmt = db.prepare(statement);
16          const result = stmt.all();
17          return result;
18      } catch (error) {
19          return error.message;
20      }
21  };
22
23  const getUser = (statement, userId) => {
24      try {
25          const stmt = db.prepare(statement);
26          const user = stmt.get(userId);
27          return user;
28      } catch (error) {
29          return error.message;
30      }
31  };
32
33  const updateUser = (statement, user) => {
34      try {
35          const stmt = db.prepare(statement);
36          const result = stmt.run(user);
37          return result.lastInstertRowid;
38      } catch (error) {
39          return error.message;
40      }
41  };
42
43  const deleteUser = (statement, userId) => {
44      try {
45          const stmt = db.prepare(statement);
46          const result = stmt.run(userId);
47          return result.lastInstertRowid;
48      } catch (error) {
49          return error.message;
50      }
51  };
52
53  module.exports = { storeUser, getUsers, getUser, updateUser, deleteUser }
```

Anexo 7 Ejemplo modulo user-controllers