INSTITUTO NEBRIJA | Formación Profesional

Creación de un CRM

Hugo García Álvarez

Alejandro García Álvarez

Sergio Cuadrado

Curso académico:

ÍNDICE PAGINADO

1. JUSTIFICACIÓN DEL PROYECTO

El proyecto intermodular se plantea como una oportunidad para integrar, en un entorno realista y completo, los conocimientos adquiridos a lo largo del ciclo formativo. La necesidad de desarrollar una aplicación funcional que combine frontend, backend, persistencia de datos, diseño de interfaces, automatización de procesos y servicios concurrentes responde directamente a las competencias profesionales que se demandan actualmente en el sector tecnológico.

1. La elección de un sistema basado en arquitectura cliente-servidor, con un frontend desarrollado en Angular y un backend implementado en Spring Boot, permite al alumnado enfrentarse a un escenario de desarrollo moderno, alineado con las prácticas utilizadas en empresas de software. Asimismo, el uso de una base de datos relacional y la aplicación de patrones como DAO, Repository o MVC refuerzan la comprensión de la persistencia y del diseño limpio orientado a la mantenibilidad del código.

2. Este proyecto también se justifica por su carácter integrador: combina los contenidos de diferentes módulos —Acceso a Datos, Diseño de Interfaces, Sistemas de Gestión Empresarial y Programación de Servicios y Procesos— para construir una solución completa, modular y escalable. Gracias a esta integración, el alumnado puede comprender cómo cada área del desarrollo contribuye a un producto final coherente, algo imprescindible para su futura inserción laboral.

3. Además, el enfoque práctico del proyecto fomenta el trabajo en equipo, la gestión del tiempo, el uso de control de versiones y la documentación técnica, habilidades transversales que forman parte del día a día en cualquier empresa de desarrollo. La evaluación individual asegura que cada miembro del grupo demuestre su aportación personal, garantizando un aprendizaje significativo y equitativo.

4. En definitiva, este proyecto se justifica como una experiencia formativa clave que permite aplicar conocimientos teóricos en un contexto práctico, profesional y completo, acercando al alumnado a las metodologías y tecnologías usadas en la industria del

software y consolidando las competencias necesarias para su futuro desempeño profesional.

2. INTRODUCCIÓN

El presente proyecto intermodular tiene como objetivo principal el desarrollo de una aplicación completa que integre los conocimientos adquiridos en los distintos módulos del ciclo formativo de Desarrollo de Aplicaciones Multiplataforma / Desarrollo de Aplicaciones Web. A través de un enfoque práctico y orientado a la realidad profesional, el alumnado debe diseñar, implementar y documentar un sistema que combine un frontend moderno, un backend seguro y estructurado, y una gestión eficiente de datos y procesos.

1. La propuesta obliga a aplicar tecnologías actuales como Angular en la parte cliente y Spring Boot en el servidor, junto con una base de datos relacional o documental. Esta arquitectura permite construir una solución escalable y mantenible, similar a las utilizadas en entornos empresariales reales. Además, se incorporan elementos clave del desarrollo profesional como el control de versiones con GitHub, el diseño de interfaces centradas en la experiencia de usuario y la ejecución de procesos concurrentes y servicios automatizado.

2. El proyecto no solo busca evaluar conocimientos técnicos, sino también promover competencias transversales como el trabajo colaborativo, la planificación, la documentación del desarrollo y la defensa oral del trabajo realizado. Cada alumno aporta una parte específica al sistema, garantizando así la implicación individual y una comprensión global del producto software.

3. A lo largo del desarrollo se combinan diferentes disciplinas —Acceso a Datos, Diseño de Interfaces, Sistemas de Gestión Empresarial y Programación de Servicios y Procesos— para dar lugar a una aplicación funcional, coherente y alineada con las necesidades actuales del sector. De esta forma, este proyecto se convierte en un pilar fundamental dentro del proceso formativo, preparando al alumnado para afrontar retos reales en su futura carrera profesional.

3. OBJETIVOS

A. OBJETIVO GENERAL

Desarrollar un sistema CRM completo y funcional que integre un frontend moderno basado en Angular, un backend robusto implementado con Spring Boot y una base de

datos SQL, garantizando una comunicación eficiente entre todas las capas y aplicando los conocimientos adquiridos en los diferentes módulos del ciclo formativo.

B. OBJETIVOS ESPECÍFICOS

-Diseñar una arquitectura cliente-servidor que permita separar correctamente la lógica de presentación, la lógica de negocio y la capa de persistencia.

-Implementar una interfaz de usuario intuitiva, responsiva y accesible utilizando Angular 18, aplicando principios de UX/UI y estructura modular basada en componentes.

-Desarrollar un backend seguro y estructurado con Spring Boot, capaz de gestionar entidades del CRM, procesar peticiones del cliente y aplicar las reglas de negocio correspondientes.

-Configurar y gestionar una base de datos SQL para almacenar de forma persistente la información del CRM, asegurando la integridad y coherencia de los datos.

-Establecer la comunicación entre Angular y Spring Boot mediante servicios REST, garantizando un flujo fluido de datos basado en solicitudes HTTP y respuestas en formato JSON.

-Transformar los datos de SQL a JSON mediante Spring, permitiendo que Angular consuma la información de forma estandarizada y pueda representarla en la interfaz.

-Aplicar buenas prácticas de programación, incluyendo uso de control de versiones, documentación técnica, modularidad y patrones de diseño.

-Fomentar el trabajo colaborativo y la planificación, organizando tareas de forma eficiente mediante metodologías propias del desarrollo ágil.

-Realizar pruebas funcionales y validaciones, asegurando el correcto funcionamiento del CRM en sus distintas partes: frontend, backend y base de datos.
-Documentar el proceso de desarrollo, incluyendo análisis, diseño, implementación, despliegue y conclusiones individuales y grupales.

4. DESARROLLO

1. FUNDAMENTACIÓN TEÓRICA

Para el desarrollo del sistema de gestión de relaciones con clientes (CRM) se ha seleccionado el framework Angular 18. Esta versión se considera adecuada debido a su madurez, estabilidad y compatibilidad con las bibliotecas más utilizadas en el ecosistema Angular. Aunque actualmente existe Angular 20, se ha optado por la versión 18 al ofrecer un equilibrio óptimo entre innovación tecnológica y fiabilidad en entornos de producción.

Angular 18 introduce mejoras sustanciales respecto a versiones anteriores, entre ellas el soporte para la detección de cambios sin Zone.js (zoneless), la implementación del nuevo sistema de reactividad mediante Signals, y optimizaciones en Server-Side Rendering (SSR) y Static Site Generation (SSG). Estas características permiten optimizar el rendimiento del sistema y mejorar la experiencia de usuario, especialmente en aplicaciones con un elevado número de componentes y actualizaciones dinámicas, como ocurre en un CRM.

Asimismo, esta versión facilita el uso de componentes independientes (standalone components), lo que contribuye a una arquitectura más modular, escalable y mantenible. Esta característica resulta esencial para proyectos de larga duración que requieren una evolución continua y una gestión eficiente del código fuente.

Comunicación asíncrona y reactividad en Angular

En el contexto del desarrollo de aplicaciones modernas, es necesario diferenciar entre la comunicación asíncrona y la reactividad, conceptos que, aunque relacionados, cumplen funciones distintas dentro del framework Angular.

La comunicación asíncrona se gestiona principalmente mediante la librería RxJS (Reactive Extensions for JavaScript), la cual permite manejar flujos de datos que se producen en distintos momentos del tiempo, como peticiones HTTP, eventos del usuario o transmisión de datos en tsiempo real. RxJS se basa en la utilización de Observables, que representan secuencias de valores que pueden emitirse de forma continua o eventual. Su sistema de operadores (map, filter, mergeMap, entre otros) permite transformar, combinar y gestionar de forma eficiente estos flujos de información. En el contexto de un CRM, RxJS resulta especialmente útil para mantener sincronizados los datos entre el cliente y el servidor, garantizando la actualización inmediata de la información sobre clientes, contactos, oportunidades o tareas.

Por otro lado, la reactividad mediante Signals, introducida a partir de Angular 16 y consolidada en Angular 18, ofrece un enfoque más simple y predecible para la gestión del estado interno de la aplicación. Los Signals permiten que los componentes de la interfaz se actualicen automáticamente cuando los valores asociados cambian, sin necesidad de recurrir a suscripciones manuales o al sistema tradicional de detección de cambios basado en Zone.js. Este modelo proporciona un comportamiento determinista y mejora tanto el rendimiento como la legibilidad del código.

En síntesis:

RxJS y Observables se orientan a la comunicación asíncrona con el servidor y la gestión de datos externos.

Signals se utilizan para la reactividad interna de la aplicación y el control del estado local.

La combinación de ambas tecnologías proporciona un flujo de datos coherente y eficiente, permitiendo desarrollar una interfaz dinámica, fluida y sincronizada en tiempo real con el backend.

Funcionalidad destacada en el proyecto

Dentro del desarrollo del CRM, la funcionalidad más destacada es la gestión reactiva de clientes y oportunidades, ya que representa la integración práctica de los conceptos de comunicación asíncrona y reactividad.

Esta funcionalidad permitirá que las modificaciones realizadas por los usuarios (como la creación o actualización de registros de clientes, tareas o contactos) se reflejen de

manera inmediata en la interfaz sin necesidad de recargar la aplicación. Para lograrlo, se combinará el uso de RxJS para la comunicación en tiempo real con el servidor y Signals para la actualización automática de la interfaz de usuario.

De esta forma, el sistema ofrecerá una experiencia moderna, eficiente y fluida, acorde con los estándares actuales de las aplicaciones web. Asimismo, la adopción de componentes independientes y el uso de Angular Material facilitarán la creación de una interfaz modular, accesible y visualmente atractiva, contribuyendo a la escalabilidad, mantenibilidad y sostenibilidad técnica del proyecto a largo plazo.

2. MATERIALES Y METODOS

Para continuar con el desarrollo del proyecto, comenzamos reflexionando sobre la estructura general del sistema y la manera en que se relacionan sus diferentes componentes. A partir de este análisis inicial, se presentan a continuación los diagramas que permiten visualizar la arquitectura, el flujo de datos y la organización interna tanto del frontend como del backend.

Diagrama de arquitectura de la aplicacion

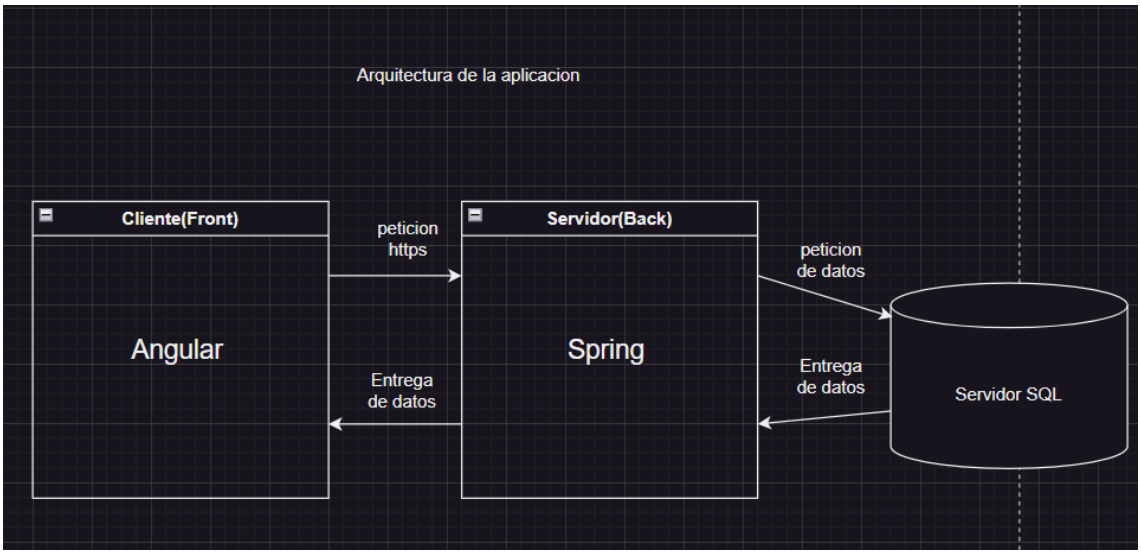


Diagrama de comunicación Angular-spring-sql

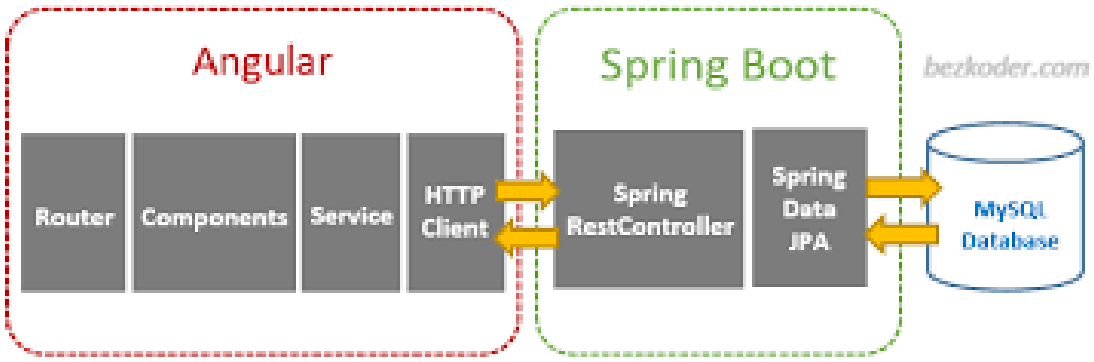
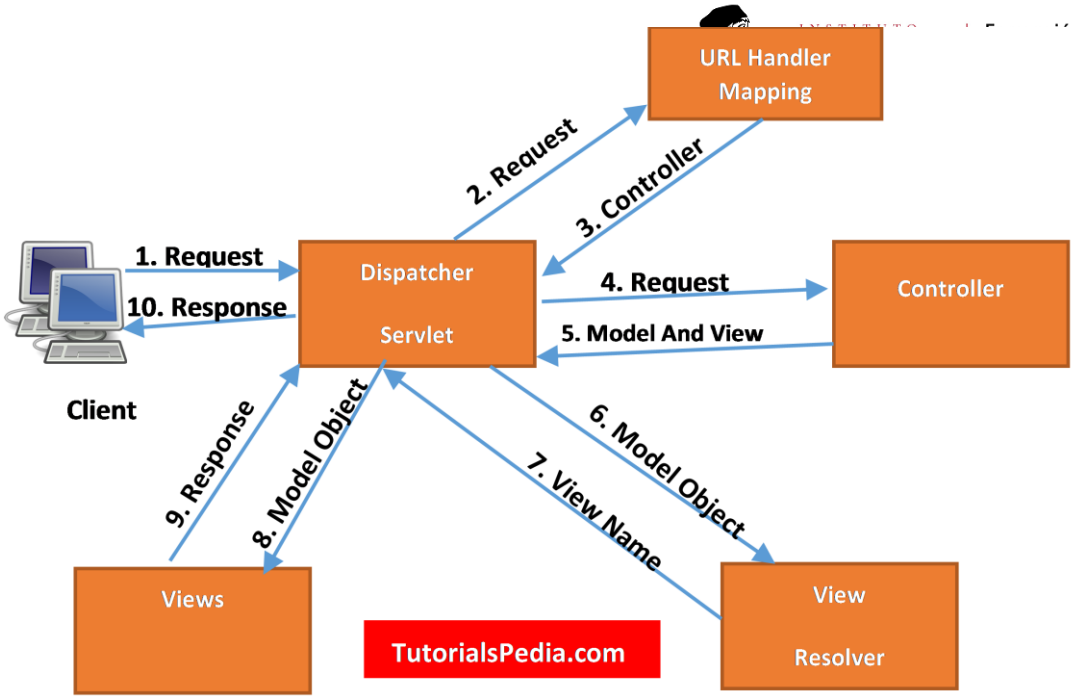


Diagrama MVC de spring



3. RESULTADOS Y ANALISIS

A continuación, se muestra el resultado del análisis hecho para la creación del CRM

Diagrama de casos de uso

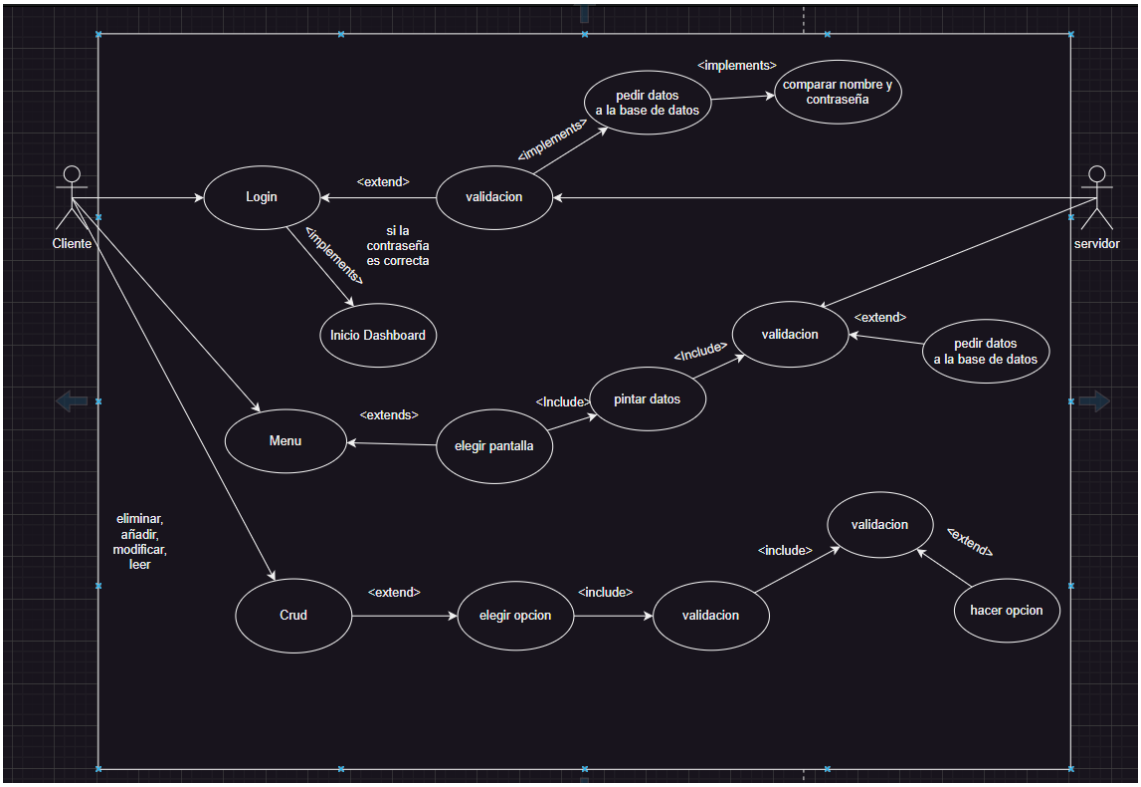
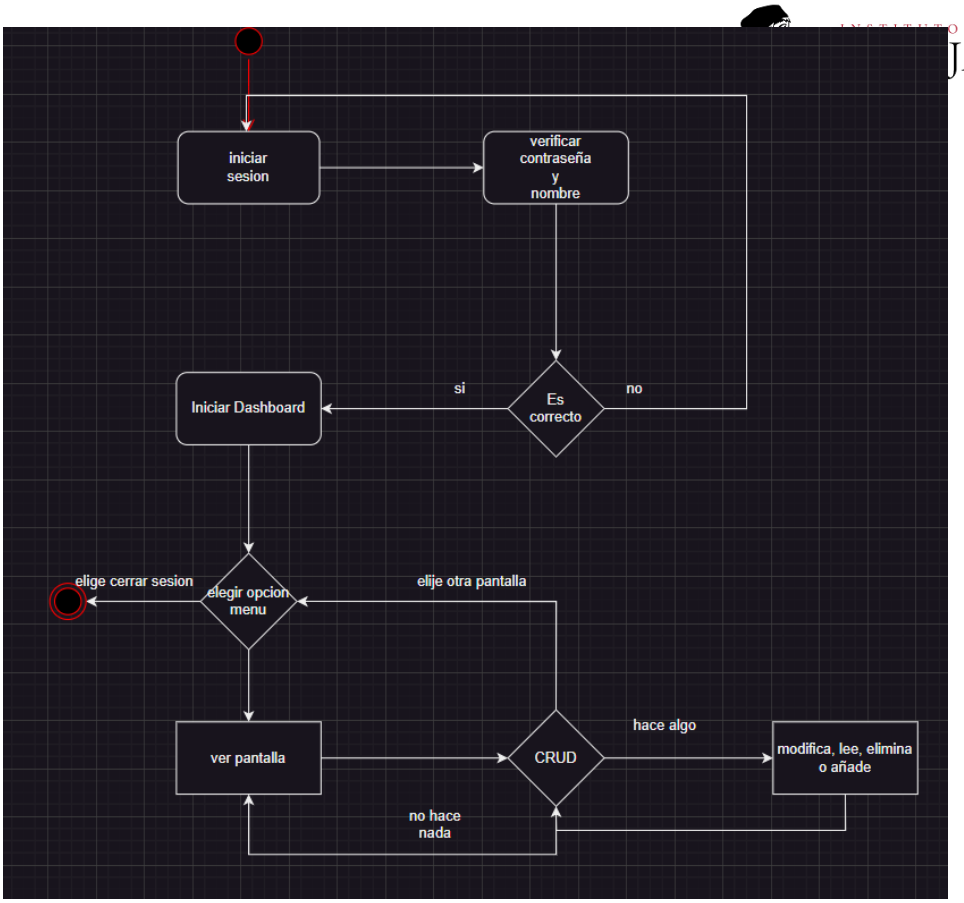
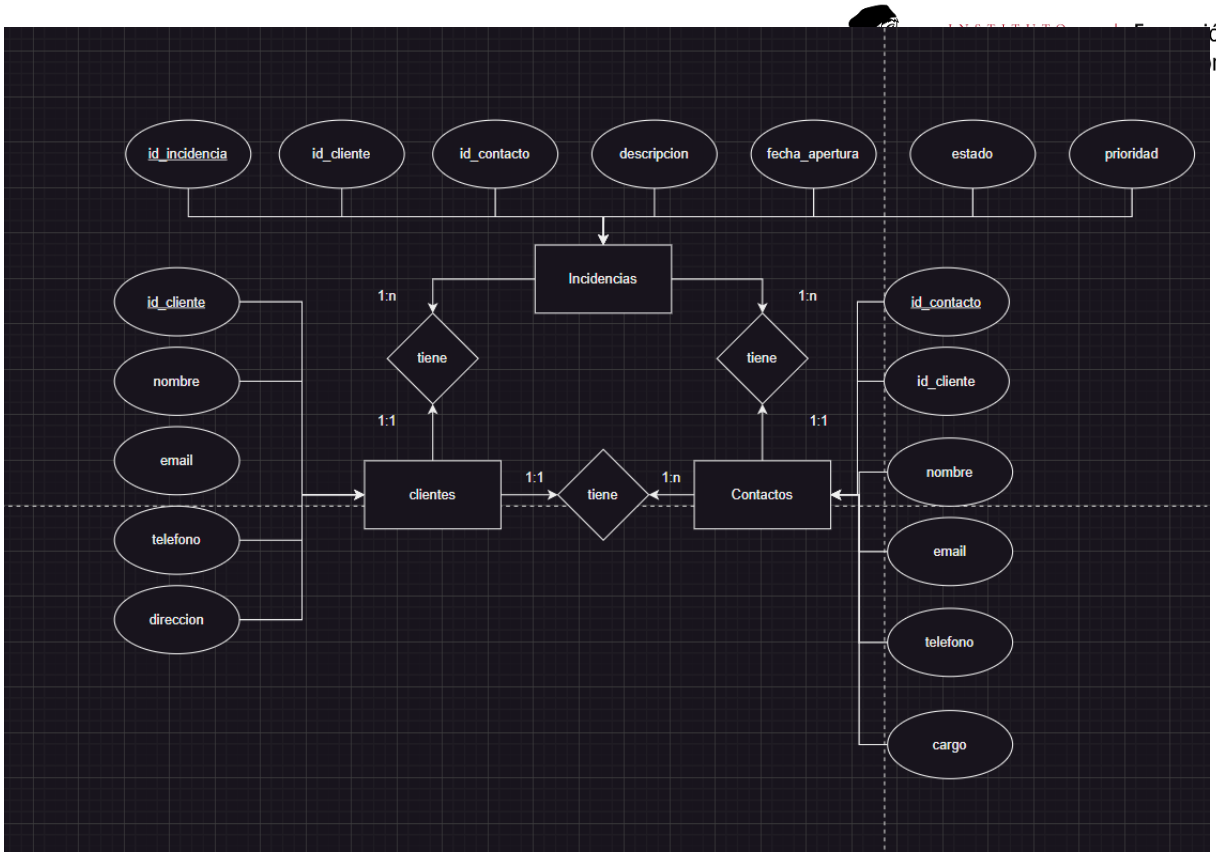


Diagrama de actividad

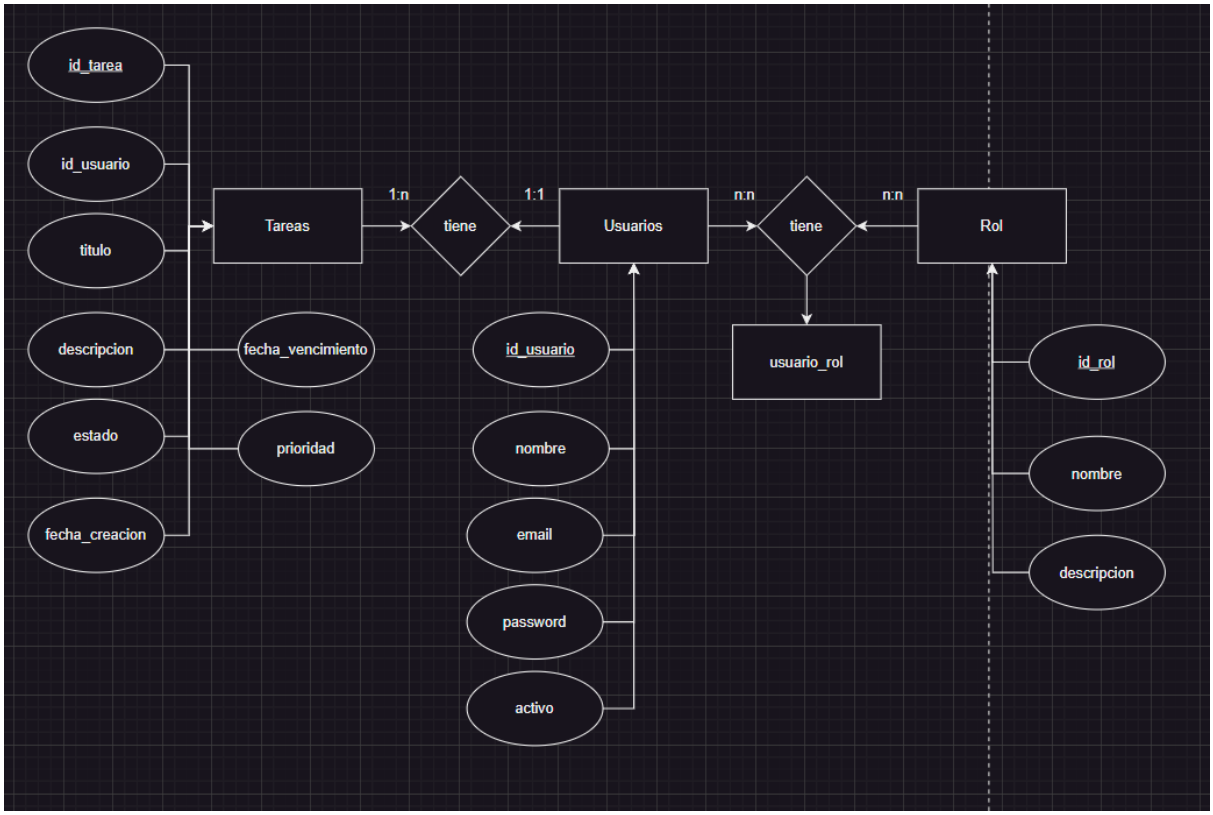


Base de datos

Un cliente tiene muchos contactos y cada contacto pertenece a un cliente (1:n)
Un cliente y contacto puede tener varias incidencias, pero cada incidencia es de un cliente o contacto(1:n)



Relación usuarios-rol-tareas



Link al figma https://www.figma.com/design/IArvpjeYSFtIq6Zr502zNc/prototype?node-id=0-1&t=f4AiLi7DKXS9PCrS-1

5. CONCLUSIONES

El desarrollo del proyecto ha culminado de manera satisfactoria, logrando implementar un CRM plenamente funcional gracias a la integración efectiva de Angular 18 en el frontend, Spring Boot en el backend y SQL como sistema de persistencia. La combinación de estas tecnologías ha permitido construir una arquitectura sólida, escalable y alineada con los estándares profesionales del desarrollo web actual.

Durante el proceso, Angular 18 ha sido fundamental para crear una interfaz dinámica, moderna y adaptable, facilitando la interacción fluida del usuario con el sistema. Su estructura basada en componentes, junto con el uso de servicios y comunicación reactiva, ha permitido organizar el frontend de forma clara y modular, simplificando la gestión de datos y la navegación dentro de la aplicación.

Por su parte, Spring Boot ha actuado como el núcleo lógico del proyecto. Gracias a sus módulos —como Spring Web, Spring Data JPA y Spring Security— ha sido posible desarrollar una API robusta, segura y capaz de gestionar todas las operaciones necesarias del CRM. Spring se ha encargado de comunicar el sistema con la base de datos SQL, enviando consultas, ejecutando transacciones y obteniendo la información requerida. Una vez que los datos son recuperados desde la base de datos, Spring los transforma en un formato estándar y universal: JSON.

Este proceso ha permitido establecer una comunicación clara y estructurada entre las distintas capas del proyecto. Angular realiza una solicitud a la API desarrollada en Spring; Spring procesa esa solicitud internamente, accede a la base de datos SQL para recuperar o modificar la información necesaria, y genera una respuesta en formato JSON. Finalmente, Angular consume ese JSON para actualizar la interfaz y mostrar los datos al usuario en tiempo real.

Gracias a esta cadena de comunicación y a la correcta división entre cliente, servidor y almacenamiento, el proyecto ha conseguido un flujo de trabajo eficiente y bien estructurado. La integración entre Angular, Spring y SQL ha demostrado ser no solo viable, sino altamente eficaz para el desarrollo de aplicaciones empresariales como un CRM. En conjunto, este proyecto ha permitido aplicar de forma práctica los conocimientos adquiridos en los distintos módulos, culminando en un producto final coherente, funcional y profesional.

6. LÍNEAS DE INVESTIGACIÓN FUTURAS

(No son obligatorios, pero pueden aparecer)

7. BIBLIOGRAFÍA

- Angular. (2025). *Angular Documentation*. https://angular.io
- Spring. (2025). *Spring Boot Documentation*. https://spring.io/projects/spring-boot
- MySQL. (2025). *MySQL Reference Manual*. https://dev.mysql.com/doc
- OpenAI. (2025). *ChatGPT* https://chat.openai.com
- Google. (2025). *Gemini* https://gemini.google.com
- Microsoft. (2025). *GitHub Copilot* https://github.com/features/copilot

8. ANEXOS

9. OTROS PUNTOS

(No son obligatorios, pero pueden aparecer)

- Aportaciones personales
- Retos profesionales
- Restos personales
- Agradecimientos