CITDEV Express: Solución tecnológica para impulsar el Desarrollo de Software a través de Citizen Developers

Breiner Eduardo Hernández García
John Jairo Cardona Echeverry

Proyecto de grado entregado para obtener el título de
Magíster en Ingeniería de Software

Dirigida por
Ph.D. Luisa Rincón

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Maestría en Ingeniería de Software
Santiago de Cali
27 de octubre de 2025

Santiago de Cali, 27 de octubre de 2025.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "CITDEV Express: Solución tecnológica para impulsar el Desarrollo de Software a través de Citizen Developers" realizado por los estudiantes de Magister en Ingeniería de Software Breiner Eduardo Hernández García (cod: 1114822973) y John Jairo Cardona Echeverry (cod: 94447491), el cual se encuentran terminado y considero que cumplen con los requisitos para ser sustentado.

Atentamente,

Firmado digitalmente por Luisa Rincón
DN: OU=Dpto Elect y Ciencias Comput-Facultad Ing y Ciencias, O=Pontificia Universidad Javeriana Cali, CN=Luisa Rincón, E=lfrincon@javerianacali.edu.co
Razón: Soy el autor de este documento
Ubicación:
Fecha: 2025.10.29 16:21:40-05'00'
Foxit PDF Reader Versión: 2025.1.0

Ph.D. Luisa Rincón

Santiago de Cali, 27 de octubre de 2025.

Señores
Pontificia Universidad Javeriana Cali
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software
Cali.

Cordial Saludo.

Nos permitimos presentar a su consideración el proyecto de grado titulado "CITDEV Express: Solución tecnológica para impulsar el Desarrollo de Software a través de Citizen Developers" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Magister en Ingeniería de Software.

Atentamente,

Breiner Eduardo Hernández García
Código: 1114822973

John Jairo Cardona Echeverry
Código: 94447491

0.1. Resumen

En el dinámico ámbito del desarrollo de software, la codificación de operaciones CRUD (Create, Read, Update, Delete) constituye una fase esencial para las operaciones básicas de gestión de datos. Sin embargo, la complejidad y la gran cantidad de código generado por herramientas de andamiaje (scaffolding) -como Laravel, Symfony, y otros— las cuales producen código repetitivo (boilerplate code) orientado a desarrolladores con experiencia pueden limitar la partcipación de perfiles no técnicos, especialmente en un contexto global marcado por la escasez de talento tecnológico.

Este trabajo de grado aborda este desafío proponiendo una solución tecnológica (*CITDEV Express*) que no solo simplifica el proceso de desarrollo y promueve la interoperabilidad, sino que también empodera a los *Citizen Developers*. De esta manera, se contribuye a cerrar la brecha de habilidades tecnológicas y se facilita la adaptación de las organizaciones ante la creciente demanda de transformación digital, maximizando la eficiencia y minimizando costos asociados al desarrollo de software.

Palabras claves: Citizen developer, Scaffolding, Boilerplate, Interoperability.

0.2. Abstract

In the dynamic field of software development, coding CRUD operations (Create, Read, Update, Delete) constitutes an essential phase for basic data management operations. However, the complexity and the substantial amount of code generated by scaffolding tools —such as Laravel, Symfony, and others— which produce repetitive code (Boilerplate code) and are designed for experts, can limit the involvement of non-technical profiles, especially in a global context of technological talent scarcity.

This thesis addresses this challenge by proposing a technological solution (*CITDEV Express*) that not only simplifies the development process and promotes interoperability but also empowers *Citizen Developers*. In this way, it contributes to bridging the technological skills gap and facilitates organizations in adapting to the increasing demand for digital transformation, maximizing efficiency, and minimizing costs associated with software development.

Keywords: Citizen developer, Scaffolding, Boilerplate, Interoperability.

0.3. Introducción

En el vertiginoso panorama del desarrollo de software, la implementación de operaciones CRUD (Create, Read, Update, Delete) emerge como una fase crítica para la gestión eficiente de opreaciones básicas de datos. Este proceso, fundamental en el ámbito tecnológico, a menudo se ve potenciado por diversas herramientas de Scaffolding, como Ruby on Rails, Laravel, Symfony y Django, entre otras; que buscan acelerar el desarrollo de aplicaciones mediante la generación automática de código inicial.

No obstante, estas herramientas, aunque eficaces para profesionales experimentados, presentan desafíos considerables para quienes no cuentan con conocimientos avanzados en programación. La creación automática de código redundante (Boilerplate) puede volverse extensa y compleja, lo que puede limitar la participación de desarrolladores no técnicos. En un contexto donde la demanda de talento tecnológico supera la oferta, la incorporación de nuevos actores, conocidos como *Citizen Developers*, se presenta como una respuesta estratégica a la falta de habilidades técnicas.

Este trabajo de grado abordó dichas problemáticas proponiendo a *CITDEV Express* como una solución tecnológica que facilita la creación de funcionalidades para la gestión de operaciones básicas de datos, promoviendo la agilidad, la interoperabilidad y empoderando a *Citizen Developers* para contribuir al desarrollo de software. Asimismo, se sustenta en la coyuntura global de los últimos años que han estado caracterizados por la acelerada adopción de las Tecnologías de la Información (TI) por parte de las empresas. Esta tendencia, que se intensificó durante la pandemia de 2020, catalizó una transformación digital acelerada y obligó a las empresas a buscar soluciones tecnológicas que les permitan adaptarse rápidamente a nuevas oportunidades y necesidades del mercado, buscando de esta manera mantener su competitividad en un entorno empresarial en constante cambio. Esta evolución, a su vez, ha agudizado el déficit de talento en tecnologías de la información, manifestándose como una brecha significativa entre la oferta y la demanda de talento en TI.

La implementación de *CITDEV Express* se percibe como esencial en el contexto actual, donde las organizaciones buscan maximizar recursos, aumentar la productividad, reducir los costos y mejorar la experiencia de los clientes.

La propuesta en este trabajo de grado no solo facilita la participación de *Citizen Developers*, sino que también se alinea con estándares de calidad al incorporar buenas prácticas de desarrollo y evitar la generación de código repetitivo.

Finalmente, este trabajo de grado llevó a cabo un análisis de caso de estudio orientado a cómo los profesionales del área de desarrollo de software percibieron *CITDEV Express*. La evaluación se centró en múltiples dimensiones, como cambio cultural en la forma en que concebimos y ejecutamos proyectos de desarrollo de software, promoviendo la inclusión y la participación de *Citizen Developers* para asegurar un enfoque colaborativo y participativo en el desarrollo. La experiencia de usuario (UX) fue otro punto de análisis, priorizándose la creación de interfaces intuitivas y eficientes que potenciaran la productividad, mitigaran los errores y aumentaran la satisfacción de los usuarios. La estrategia de evitar boilerplates, eliminando la redundancia de código, se examinó con aten-

ción, buscando una implementación que promoviera la mantenibilidad y escalabilidad del producto. Además, se profundizó en la interoperabilidad, asegurándose de que la solución proporcionara una integración sin inconvenientes con otras tecnologías o plataformas. A través de este enfoque integral, se buscó no solo obtener percepciones significativas de los profesionales del desarrollo, sino también identificar áreas de mejora y buenas prácticas que contribuyeran a la evolución de *CITDEV Express*.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Descripción del problema

1.1. Definición del problema de investigación

En el proceso de desarrollo de software, suele ser necesario realizar la codificación de operaciones básicas para el manejo de datos, más conocido como operaciones *CRUD* (Pantelelis and Kalloniatis, 2022). Estas funcionalidades en la mayoría de los casos son numerosas y de baja complejidad. No obstante, su necesidad se extiende más allá del mero manejo de datos, ya que ante la posibilidad de tener sistemas heterogéneos incluso dentro de una misma organización, es esencial contar con mecanismos que faciliten un intercambio seguro de datos (Amazon Web Services, sfa; Tolk, 2013). Este principio, conocido como interoperabilidad, destaca la importancia de que diferentes sistemas puedan comunicarse entre sí (ISO/IEC, 2011; IEEE, 1990).

Se ha identificado que en el contexto actual de desarrollo de software web, se destacan diversas *Scaffolding Tools* (Brisaboa et al., 2016) como Laravel (Laravel, sf), Symfony (Symfony, sf), Ruby on Rails (Ruby on Rails, sf), Django (Django, sf), entre otras, que buscan acelerar significativamente el proceso de construcción de aplicaciones proporcionando una estructura inicial estandarizada, la cual a menudo resulta en la creación automática de código redundante (*Boilerplate*) (Misu and Satter, 2022) que puede llegar a ser extenso y complejo de entender en algunos casos. Es importante destacar que estas herramientas van dirigidas a personal experto, ya que por lo general ofrecen una Interfaz de Línea de Comandos (CLI) (Amazon Web Services, sfd) para la generación automática de código fuente.

Sin embargo, este enfoque puede representar una barrera para aquellos que no poseen conocimientos avanzados en programación, limitando su capacidad para contribuir en el proceso de desarrollo de software. Teniendo en cuenta que desde hace años el mundo en general arrastra un rezago de talento humano para el sector de las tecnologías (García Rico, 2023), una tendencia reciente involucra personas no técnicas conocidas como *Citizen Developers* (Avishahar-Zeira and Lorenz, 2023; Hurlburt, 2021).

Involucrar a *Citizen Developers* en tareas más simples, ayudaría al área de tecnología en las organizaciones para optimizar la asignación de recursos especializados a centrarse en aspectos más complejos del desarrollo, lo que podría traducirse en una mayor eficiencia y una reducción significativa de costos (Khorram et al., 2020).

Por consiguiente, es necesario abordar estos desafíos mediante el diseño de una solución tecnológica que facilite la creación de funcionalidades simples mientras se evita la redundancia de código, se mantiene la interoperabilidad y se habilitan los *Citizen Developers* para que puedan contribuir al área de tecnología en las organizaciones.

1.1.1. Planteamiento del problema

¿Cómo hacer una solución tecnológica que facilite la creación de operaciones básicas para el manejo de datos mientras se evita la redundancia de código, se considera la interoperabilidad y se habilitan los *Citizen Developers* para que puedan contribuir al área de tecnología en las organizaciones?

1.1.2. Sistematización

- ¿Cuáles son las especificaciones y características críticas de las operaciones básicas para el manejo de datos, considerando las necesidades y habilidades de los *Citizen Developers*?
- ¿Cómo diseñar e implementar las funcionalidades de gestión de operaciones simples considerando las habilidades de los *Citizen Developers* evitando la redundancia del código fuente?
- ¿Cómo diseñar e implementar un marco de interoperabilidad que garantice una integración efectiva y segura con otros sistemas de información, sin la intervención de personal experto en programación?
- ¿Cómo evaluar la usabilidad, eficiencia e interoperabilidad de la solución tecnológica propuesta?

1.2. Objetivos

1.2.1. Objetivo general

Proponer una solución tecnológica que facilite la creación de operaciones básicas para el manejo de datos mientras se evita la redundancia de código, se considera la interoperabilidad y se habilitan los *Citizen Developers* para que puedan contribuir al área de tecnología en las organizaciones.

1.2.2. Objetivos específicos

- Definir las especificaciones y características críticas de las operaciones básicas para el manejo de datos, considerando las necesidades y habilidades de los *Citizen Developers*.
- Diseñar e implementar las funcionalidades de gestión de operaciones simples considerando las habilidades de los *Citizen Developers* evitando la redundancia del código fuente.
- Diseñar e implementar un marco de interoperabilidad que garantice una integración efectiva y segura con otros sistemas de información, sin la intervención de personal experto en programación.
- Evaluar la usabilidad, eficiencia e interoperabilidad de la solución tecnológica propuesta por medio de un caso de estudio.

1.3. Justificación

La pandemia del coronavirus del año 2020 ha tenido un impacto significativo en el comportamiento de las personas y la economía a nivel mundial. El PIB real se contrajo en un 3.1 % a escala global (Jung and Katz, 2022), lo que ha sido notable en la demanda en los sectores digitales. Como resultado, las organizaciones han tenido que adaptar sus estrategias para responder a este entorno empresarial de constante cambio (Alvarado-Morales and Zambrano-Roldán, 2020). Ha habido una acelerada transformación digital que ha generado una demanda sin precedentes de soluciones tecnológicas que les permitan mantenerse competitivas. De esta manera, las organizaciones pueden adaptarse rápidamente a los cambios del mercado, identificar nuevas oportunidades, ofrecer productos y/o servicios novedosos que se alineen con las necesidades de los clientes (Sulbarán, 2023).

El crecimiento de la demanda de transformación digital ha intensificado el déficit de talento en Tecnologías de la Información, en España un informe reciente reveló que pierde alrededor de 15.000 millones de euros al año por la escasez de talento tecnológico, lo que equivale a un 1,5 % de su PIB. Colombia no es exenta a esta situación y también enfrenta un déficit de profesionales en tecnología, que se sitúa para el año 2025 entre los 90.000 y los 130.000 puestos. Esta situación, de no contar con una fuerza laboral alrededor de las tecnologías, puede tener altas implicaciones socioeconómicas negativas para el desarrollo del país (García Rico, 2023). La creación de *CITDEV Express*, dirigida a *Citizen Developers* se presenta como una respuesta estratégica a esta brecha de habilidades, permitiendo que individuos sin un trasfondo técnico extenso contribuyan al proceso de desarrollo de software.

La creación de operaciones básicas en el contexto actual se ha convertido en una práctica esencial para las organizaciones que buscan maximizar sus recursos, reducir tiempos de desarrollo y minimizar costos asociados al desarrollo de software. *CITDEV Express* la solución tecnológica propuesta en este trabajo de grado se posiciona como una solución integral que no solo se centra en la facilidad de uso, sino también en la abstracción de la complejidad inherente en el desarrollo de software, que no solo facilita la participación de *Citizen Developers*, sino que también contribuye a la reducción de errores asociados con la codificación manual, al automatizar procesos y proporcionar interfaces visuales intuitivas.

Además, la adopción de mejores prácticas de desarrollo se convierte en un componente fundamental de *CITDEV Express*. Al fomentar la implementación del uso de código eficientemente, ya que no se genera código repetitivo, la solución tecnológica no solo es incluyente de los *Citizen Developers*, sino que también ayudará a que las soluciones creadas cumplan con los requisitos de usabilidad, eficiencia e interoperabilidad necesarios en el contexto empresarial actual.

En conclusión, el desarrollo de *CITDEV Express*, no solo aborda la creciente demanda de transformación digital y la escasez de talento en tecnologías de la información, sino que también responde a la necesidad de las organizaciones de reducir costos y tiempos de desarrollo a través de la democratización del proceso de desarrollo al empoderar a *Citizen Developers*.

1.4. Delimitaciones y alcances

A continuación, se describen los elementos que fueron contemplados en el proyecto con el objetivo de delimitar el alcance de los componentes implementados.
Se desarrolló la primera versión de *CITDEV Express* bajo las siguientes restricciones:

- Se construyó un core que permite a los *Citizen Developers* gestionar operaciones básicas para el manejo de datos de manera centralizada a partir de la configuración de una entidad. Se entiende por operaciones básicas de datos la codificación de operaciones *CRUD* (Create, Read, Update, Delete).
- Se implementó una funcionalidad que permite la creación y configuración de entidades para gestionar información sin necesidad de escribir código.
- Se garantizó la integridad de los datos conforme a la configuración definida en la entidad para cada uno de sus atributos. Por ejemplo, en el caso que un atributo se declare como obligatorio, *CITDEV Express* realiza una validación para asegurar que dicho atributo se considere realmente como necesario.
- Se desarrolló una capa de servicios API para interoperar con otros sistemas.

1.4.1. Entregables

- Documentación técnica.
- Repositorio.
- Documento Proyecto de Grado.
- Manual de usuario para el uso de *CITDEV Express*.

Capítulo 2

Desarrollo del proyecto

2.1. Marco de referencia

2.1.1. Marco teórico

2.1.1.1. Scaffolding Tool

El concepto de *Scaffolding Tool* encuentra su origen en el ámbito de la construcción de edificios, donde se emplean estructuras de soporte temporales conocidas como andamios. En el contexto del desarrollo de software, el andamiaje se refiere a la creación de una estructura fundamental y predefinida que sirve como cimiento para una aplicación. Este enfoque de desarrollo acelerado se combina frecuentemente con técnicas de generación de código, plantillas preestablecidas y una especificación proporcionada por el desarrollador (Brisaboa et al., 2016). Con esto se busca agilizar de manera significativa el proceso de creación de aplicaciones, en lugar de emprender la laboriosa tarea de escribir manualmente todos los elementos y componentes de una aplicación desde cero.

El *Scaffolding Tool* toma gran importancia al proveer una vía rápida para generar el código base, permitiendo que los desarrolladores se enfoquen en dedicar su tiempo y esfuerzo a la personalización y al desarrollo de características más avanzadas que respalden la lógica del negocio.

No existe una instancia específica que pueda ser señalada como el punto de partida o la "primera vez" en que se utilizó esta técnica en el desarrollo de aplicaciones CRUD (Crear, Leer, Actualizar y Borrar), pero el origen está intrínsecamente ligado al desarrollo de frameworks y herramientas específicas diseñadas para facilitar la generación de código. Un ejemplo notable de un framework que incorporó *Scaffolding Tool* es Ruby on Rails, que ganó popularidad entre los años 2005 y 2008 (Google, 2024) destacándose como un precursor temprano.

2.1.1.2. Atributos de calidad

La calidad de un producto de software se define por el grado en que dicho producto logra satisfacer las necesidades y expectativas de sus usuarios (ISO/IEC, 2011). Dichas necesidades, también identificadas como atributos de calidad, son factores que afectan el comportamiento en tiempo de ejecución, el diseño del sistema y la experiencia del usuario (Microsoft, 2023). La idea de atributos de calidad es relativamente genérica y se relaciona prácticamente con todos los sistemas de software, desde pequeños scripts hasta grandes aplicaciones empresariales (Peruma and Krutz, 2018).

A continuación se presentan los atributos de calidad más relevantes que se aplican a *CITDEV Express*:

- **Interoperabilidad**: es la capacidad que tienen dos o más sistemas informáticos para intercambiar información (ISO/IEC, 2011; IEEE, 1990), de tal manera que no sea necesario conocer las características únicas de cada sistema (Diallo et al., 2011).
- **Seguridad**: es la capacidad que posee un sistema para proteger la información, de tal manera que personas o sistemas no autorizados no puedan leerla o modificarla (ISO/IEC, 2011).
- **Confidencialidad**: es la capacidad de protección contra el acceso a datos e información no autorizados, ya sea accidental o deliberadamente (ISO/IEC, 2011).
- **Disponibilidad**: es la capacidad que posee el sistema de estar operativo y accesible para su uso cuando se requiere (ISO/IEC, 2011).
- **Integridad**: es la capacidad que posee el sistema para prevenir modificaciones no autorizadas sobre un conjunto de datos (ISO/IEC, 2011).
- **Usabilidad**: es la capacidad que posee un sistema para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones (ISO/IEC, 2011).

2.1.1.3. APIs, API RESTful & JWT

- **Interfaces de Programación de Aplicaciones (APIs)**: las *APIs* han transformado la manera en que interactuamos con la tecnología en todos los ámbitos de nuestra vida. Desde el uso de las aplicaciones móviles que utilizamos diariamente hasta los dispositivos conectados en nuestros hogares a través del Internet de las cosas (IoT), las *APIs* se han convertido en el mecanismo de enlace que potencia la comunicación entre sistemas heterogéneos. Se puede afirmar que el uso de *APIs* es relativamente sencillo, lo cual las hace susceptibles a posibles ataques y genera la necesidad de implementar mecanismos de seguridad, como por ejemplo, la autenticación basada en tokens que en la actualidad es un enfoque dominante para proteger las *APIs* (Madden, 2020).
- **API RESTful**: en el caso de una API RESTful, el cliente solicitará datos al servidor, donde estos datos se identifican mediante un Identificador Universal de Recursos (URI). La información obtenida desde el servidor puede presentarse en formatos como texto, JSON o XML, siendo JSON el formato más utilizado actualmente. REST se emplea como la interfaz de la *API* para acceder a un recurso, permitiendo a los desarrolladores de aplicaciones interactuar con dicho recurso sin necesidad de comprender la estructura interna de la *API* en el servidor (de Rosal Ignatius Moses Setiadi et al., 2019).
- **JSON Web Token (JWT)**: es un método de autenticación basado en tokens que codifica datos en formato de cadena JSON. Consta de tres cadenas de texto codificadas en Base64 y separadas por caracteres de puntos: Header, Payload y Signature, como se muestra en la Figura 2.1.

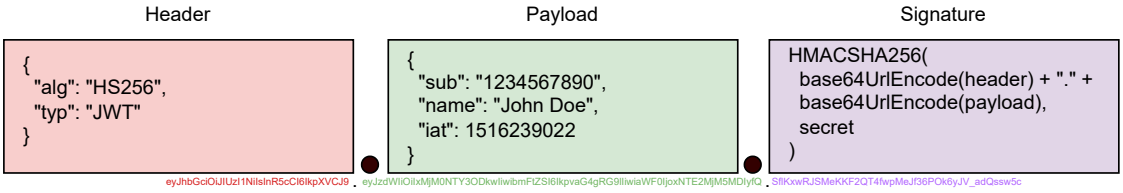


Figura 2.1: Estructura de un JSON Web Token. Adaptada de (Nicerova, 2017)

La sección Header especifica el tipo de token y el algoritmo que se utiliza para firmar el *JWT*. La sección Payload es donde se almacena la información real para que se utilice el token. Dicha información contiene datos arbitrarios, como el emisor del token, el tiempo de vencimiento y la información del usuario. La sección Signature permite a los emisores de tokens firmarlos utilizando un código de autenticación de mensajes basado en hash (HMAC), RSA o ECDSA para mantener la integridad del token (Jones et al., 2015; Pramono and Yana Javista, 2021).

El funcionamiento consiste en los siguientes pasos que se describen en la Figura 2.2:

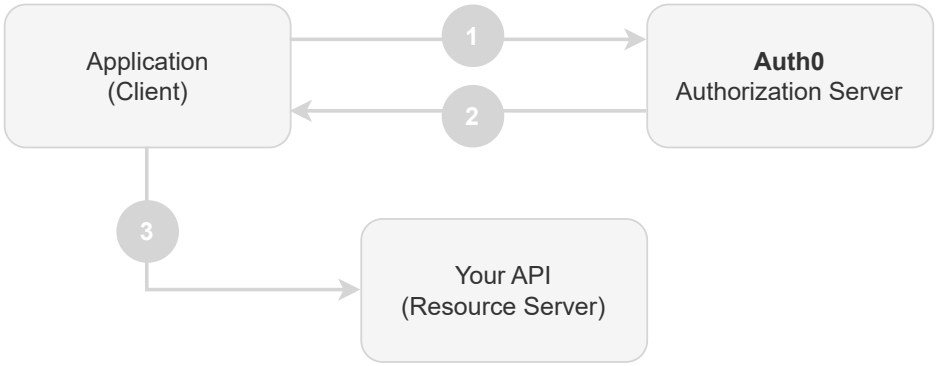


Figura 2.2: Obtener y usar un JSON Web Token. Adaptada de (jwt.io, sf)

1. La aplicación o el cliente solicita autorización al servidor.
2. Cuando se concede la autorización, el servidor devuelve un token de acceso a la aplicación.
3. La aplicación utiliza el token de acceso para acceder a un recurso protegido (como una API).

2.1.1.4. Modelo C4 para documentación arquitectónica

El *Modelo C4* (Contexto, Contenedor, Componente, Código) fue propuesto por Simon Brown como una estrategia estructurada para la documentación de arquitecturas de software desde diferentes niveles de abstracción (Brown, 2018). Su objetivo consiste en facilitar la comunicación efectiva entre los distintos perfiles que intervienen en el desarrollo, operación y mantenimiento de sistemas

de software. Este modelo no prescribe una notación formal, sino que la complementa y simplifica las vistas tradicionales mediante diagramas que responden a preguntas específicas sobre el sistema.

- **Nivel de contexto**: el diagrama de contexto representa el sistema como una "caja negra" y muestra cómo interactúan los actores externos con él. Este nivel es útil para comprender el propósito del sistema y su relación con el entorno.
- **Nivel de contenedor**: el diagrama de contenedores descompone el sistema en sus principales ejecuciones en tiempo de ejecución (contenedores), tales como aplicaciones web, servicios backend o bases de datos. Este nivel describe qué partes componen el sistema, cómo se comunican entre sí y qué tecnologías se emplean.
- **Nivel de componente**: este nivel detalla los componentes internos de cada contenedor. Un componente es una agrupación lógica de código que cumple una función específica, como controladores, servicios o repositorios. En este nivel, se hace explícito cómo se organiza internamente el sistema desde la perspectiva de la implementación.
- **Nivel de código (opcional)**: este nivel se emplea cuando se requiere visualizar la estructura interna de un componente, por ejemplo, a nivel de clases o funciones. Se emplea en proyectos donde la mantenibilidad y la trazabilidad del código fuente son especialmente relevantes.

El uso del *Modelo C4* en contextos educativos y proyectos académicos ha demostrado ser útil para representar de forma estructurada la arquitectura de un sistema, permitiendo articular los requisitos y funcionalidades del software desde una perspectiva progresiva, que abarca desde el contexto general hasta los componentes específicos de la solución (Vázquez-Ingelmo et al., 2020). En esta tesis, el *Modelo C4* se ha utilizado para representar los componentes que conforman la solución tecnológica y las interacciones que se establecen entre ellos.

2.1.2. Trabajos previos

2.1.2.1. Sistemas de gestión de bases de datos (SGBD)

Es un software especializado que facilita la organización, almacenamiento, recuperación y manipulación de datos en una base de datos. Actúa como una capa intermedia entre las aplicaciones y los datos, permitiendo a los usuarios y programas acceder y gestionar información de manera eficiente. Los SGBD ofrecen características como seguridad, integridad de datos, gestión de concurrencia y un lenguaje de consulta que permite realizar consultas y modificaciones en la base de datos de manera estructurada. La historia de los SGBD refleja una constante evolución para satisfacer las cambiantes necesidades de almacenamiento y gestión de datos, desde aplicaciones empresariales tradicionales hasta sistemas de big data y aplicaciones en la nube. Cada década ha traído innovaciones significativas en el campo de la gestión de bases de datos, y hoy en día, hay una amplia variedad de SGBD disponibles, cada uno diseñado para cubrir necesidades específicas en el mundo de la informática y la gestión de datos.

Algunos de los SGBD más representativos y ampliamente utilizados:

- Oracle Database: desarrollada por Oracle Corporation, es conocida por su escalabilidad y robustez, y se utiliza comúnmente en aplicaciones empresariales.
- Microsoft SQL Server: desarrollada por Microsoft, es una opción popular en entornos Windows y es ampliamente utilizada en aplicaciones empresariales y sistemas de administración de bases de datos.
- MySQL: un sistema de gestión de bases de datos de código abierto ampliamente utilizado para aplicaciones web y empresariales.
- PostgreSQL: otro sistema de gestión de bases de datos de código abierto con características avanzadas de extensibilidad y soporte para lenguajes de programación.
- MongoDB: un sistema de base de datos NoSQL orientado a documentos, adecuado para aplicaciones web y móviles.

2.1.2.2. Oracle Forms

Es una herramienta de desarrollo de software utilizada para crear aplicaciones empresariales basadas en formularios. Esta herramienta es parte de la suite de desarrollo de Oracle y se centra en la creación de aplicaciones de interfaz gráfica de usuario (GUI) para acceder y gestionar bases de datos Oracle. Las aplicaciones desarrolladas con Oracle Forms generalmente se utilizan para tareas como la entrada y actualización de datos en bases de datos Oracle, la generación de informes y la automatización de procesos empresariales. Oracle Forms se basa en un modelo de desarrollo de tres capas, que incluye la capa de presentación (donde se diseñan los formularios), la capa de lógica de negocios (donde se definen las reglas empresariales y la funcionalidad) y la capa de acceso a datos (que se conecta a la base de datos Oracle subyacente). Esto permite una separación clara de las preocupaciones y facilita el mantenimiento y la escalabilidad de las aplicaciones. Oracle Forms es una herramienta que se utiliza comúnmente para desarrollar aplicaciones empresariales que involucran operaciones CRUD en bases de datos Oracle. Los formularios desarrollados con Oracle Forms se utilizan para realizar operaciones CRUD en los datos de la base de datos, permitiendo a los usuarios crear nuevos registros (Crear), leer información existente (Leer), actualizar registros existentes (Actualizar) y eliminar registros (Borrar).

2.1.2.3. Microsoft Access

Es una herramienta de base de datos que permite a los desarrolladores crear aplicaciones de escritorio que incluyen formularios interactivos para interactuar con datos almacenados en bases de datos de Microsoft Acces o SQL Server, así como en otros motores de bases de datos.

Al igual que Oracle Forms, Microsoft Access ofrece un entorno de desarrollo visual que permite a los usuarios diseñar formularios de entrada de datos y aplicaciones interactivas de usuario sin necesidad de escribir código manualmente. Estos formularios pueden utilizarse para realizar operaciones CRUD (Crear, Leer, Actualizar y Borrar) en los datos almacenados en SQL Server.

2.1.2.4. FastAPI

FastAPI se destaca como un framework de desarrollo web diseñado específicamente para la creación de APIs en Python. Su rendimiento lo posiciona al mismo nivel que otras tecnologías líderes como NodeJS y Go. Una de sus principales fortalezas es su capacidad para acelerar el proceso de desarrollo de nuevas funcionalidades, proporcionando una experiencia ágil y eficiente para los desarrolladores. Además, FastAPI contribuye a la reducción de errores humanos al validar datos de forma automáticamente mediante el uso de Pydantic, mejorando así la confiabilidad de las aplicaciones desarrolladas (Ramírez Montaño, sf).

2.1.3. Comparación de soluciones

Característica	Scaff. tool	SGBD	Oracle Forms	Ms Access	FastAPI
Enfoque en *Citizen Developer*	○	○	○	○	○
Evita código Boilerplate	○	◑	◑	◑	◑
Fácil configuración	○	○	○	◑	◑
Fácil creación de CRUD	○	◑	◑	◑	◑
Fácil interoperabilidad	◑	○	○	○	●
Sin despliegue continuo	○	◑	◑	◑	○

Tabla 2.1: Comparación de soluciones

●: Cumple, ◑: Cumple parcialmente, ○: No cumple

Esta tabla 2.1 comparativa proporciona una visión general de las diferencias entre las soluciones de software, presentadas en trabajos previos, en función de las características consideradas valiosas en el enfoque del desarrollo de *CITDEV Express* como herramienta para impulsar el Desarrollo de Software a través de *Citizen Developers*s.

En cada fila se encuentran las siguientes características:

- Enfoque en *Citizen Developer*: indica si la solución está diseñada para ser utilizada por personas no técnicas, sin una experiencia previa en programación.
- Evita código Boilerplate: se refiere a si la solución reduce o elimina la necesidad de escribir código repetitivo y estándar.
- Fácil configuración: evalúa la facilidad con la que se puede configurar y personalizar la solución tecnológica según los requisitos específicos para la gestión de datos.
- Fácil creación de CRUD: mide la facilidad para crear operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la solución tecnológica, lo cual resulta esencial para la gestión de datos.

- Fácil interoperabilidad: indica qué tan fácilmente la solución puede integrarse con otros sistemas y soluciones tecnológicas, facilitando la comunicación entre ellos.
- Sin despliegue continuo: determina si la solución tecnológica requiere o permite el despliegue continuo de los cambios que se realizan, lo cual puede afectar la estabilidad y disponibilidad del sistema.

Basándose en esta tabla comparativa, se puede observar que cada solución de software presenta sus propias fortalezas y debilidades en cuanto a las características evaluadas, pero no se identifica una de ellas que demuestre fortalezas de manera consistente a través de todas las características.

2.2. Metodología

Se propone la adopción de una metodología de desarrollo híbrido que combine elementos de la metodología tradicional y del marco ágil Scrum.

Esta elección se fundamenta en la necesidad de aprovechar las fortalezas de ambas metodologías para garantizar una gestión eficiente del proyecto, combinando la estructura y planificación de metodologías más tradicionales, así como la flexibilidad y adaptabilidad a cambios durante la ejecución de metodologías ágiles.

2.2.1. Actividades

- Identificar a los stakeholders pertenecientes a la industria de la tecnología, específicamente en áreas de desarrollo.
- Determinar las características mínimas de las operaciones básicas para el manejo de datos, considerando las habilidades de los *Citizen Developers*.
- Consultar las posibles soluciones, con enfoque en *Citizen Developer*, disponibles en el mercado para analizar las características relacionadas con este trabajo de grado.
- Analizar y evaluar las diferentes arquitecturas de software que puedan representar bases sólidas para *CITDEV Express* como solución tecnológica enfocada a *Citizen Developers*.
- Representar los esquemas del software a través de los procesos, estructura de las clases y los objetos; y su comportamiento dentro de *CITDEV Express*.
- Determinar el modelo de datos para un óptimo almacenamiento de la información de *CITDEV Express*.
- Definir el stack tecnológico para implementar *CITDEV Express*.
- Diseñar e implementar *CITDEV Express* como una solución tecnológica usable, mantenible, escalable e interoperable, que se ajuste a través de módulos, clases, estructuras de manejo de datos y modelos de datos a los requisitos que se definan.

- Diseñar una interfaz de usuario sencilla e intuitiva enfocada en el uso de un *Citizen Developer*.
- Diseñar el o los reportes de la información almacenada más relevantes para las organizaciones.
- Determinar los recursos, actividades y costos fijos que *CITDEV Express* requiere para poder entrar en funcionamiento.
- Evaluar *CITDEV Express* a través de pruebas funcionales de caja negra.
- Evaluar *CITDEV Express* a través de un caso de estudio.

2.3. Resultados esperados

Al concluir el proyecto, se espera que *CITDEV Express* esté lista para ingresar a la fase de producción, lo que posibilitará la realización de operaciones CRUD. A continuación, se relacionan los resultados esperados por cada una de las etapas del proyecto, los cuales se encuentran directamente asociados a los objetivos específicos.

- Planeación: durante esta etapa, se espera llevar a cabo actividades destinadas a definir, analizar y documentar los requisitos inherentes a *CITDEV Express*. Dichas actividades incluyen los procesos de educción, la recopilación de datos y el análisis de los requisitos identificados, así como su refinamiento. Además, se espera identificar a los actores, las funcionalidades relevantes, los riesgos potenciales y las restricciones pertinentes. La documentación en esta fase establecerá las bases para el desarrollo, garantizando una comprensión de los objetivos y contribuyendo al éxito de la implementación.

- Diseño: en la fase de diseño se tomarán decisiones para estructurar e implementar *CITDEV Express*. Durante esta etapa, con base en los requisitos se definirá la arquitectura de software y hardware para respaldar las funcionalidades. Asimismo, se establecerán el diseño de la interfaz de usuario, el modelamiento de la base de datos, las medidas de seguridad y los casos de prueba.
  En conclusión, esta etapa proporcionará las bases que facilitará la implementación, evaluación y mantenimiento del software, asegurando su cumplimiento eficiente y efectivo con los requisitos.

- Implementación: en esta fase, se espera como resultado la materialización de las definiciones conceptuales de la etapa de diseño de *CITDEV Express* en forma de software ejecutable. En esta etapa se realizará un análisis de los diseños para su refinamiento e identificar tareas de codificación granulares que se abordaran en el desarrollo de cada sprint (Amazon Web Services, sfc). La complejidad y la duración de esta etapa está directamente relacionada al lenguaje de programación utilizado, así también como a la calidad del diseño previamente realizado (Maida and Pacienzia, 2015).

- Pruebas: en esta etapa se espera garantizar que *CITDEV Express* cumpla con las metas establecidas para cada entrega incremental. Se ejecutará continuamente en el ciclo de vida del proyecto, por lo que se harán las pruebas en cada sprint, abarcando desde la evaluación de la funcionalidad de los prototipos hasta la verificación de la estabilidad, cobertura y rendimiento de la arquitectura (Maida and Pacienzia, 2015). Finalmente, se aspira a realizar pruebas finales y retroalimentación por parte de terceros de la industria del desarrollo de software.

Capítulo 3

Diseño de la solución

En este capítulo se detalla el proceso de diseño de *CITDEV Express*, que tiene como objetivo facilitar la creación de operaciones básicas de un CRUD (Crear, Leer, Actualizar y Eliminar) a través de la participación activa de los *Citizen Developers*. El diseño de *CITDEV Express* incluye una descripción detallada de la arquitectura del sistema y los componentes principales para garantizar la usabilidad, interoperabilidad y eficiencia de la plataforma.

Además, se presentan los requisitos funcionales y no funcionales que guían la implementación, asegurando que *CITDEV Express* responda a las necesidades de usuarios con conocimientos técnicos limitados en el desarrollo de software. A lo largo de este capítulo se abordarán los diferentes aspectos técnicos que sustentan el diseño de una plataforma ágil y flexible, capaz de adaptarse a las demandas actuales del sector tecnológico.

3.1. Planeación

El desarrollo de este proyecto está motivado por la experiencia personal de los autores y percibida recurrentemente en el ámbito de la ingeniería de software: las dificultades asociadas a la contratación y capacitación de personal para la ejecución de tareas CRUD básicas (Crear, Leer, Actualizar y Eliminar).

Gracias a esta experiencia, se ha enfrentado a la realidad de un largo proceso de capacitación en el stack tecnológico, lo cual representa un proceso largo, costoso y que requiere una inversión considerable de tiempo y recursos, representado en una curva de aprendizaje considerable. Esta situación, sumada a la escasez de mano de obra calificada y al costo de contratación, motivó a buscar una solución tecnológica que pudiera automatizar y optimizar estas tareas, liberando tiempo y recursos valiosos para enfocarnos en aspectos más estratégicos del desarrollo de software.

El desarrollo de este proyecto responde a estas necesidades apremiantes. Con el fin de ampliar y refinar las funcionalidades propuestas, fueron analizados los requisitos adicionales en colaboración con otros expertos del sector tecnológico.

Metodología de recolección de datos

Con el fin de ampliar y refinar las funcionalidades propuestas, se optó por realizar entrevistas para recolectar y analizar requisitos adicionales en colaboración con otros expertos del sector tecnológico. La recolección de datos se realizó con entrevistas estructuradas en tres etapas las cuales se realizaron tanto virtuales como presenciales.

En esta exploración se entrevistó a cinco profesionales del sector tecnológico y/o usuarios potenciales de la propuesta: *CITDEV Express*, incluyendo:

- Un gerente estratégico de una empresa de software.
- Un director de innovación.
- Un desarrollador senior.
- Un líder técnico de célula de desarrollo.
- Un analista de datos de una empresa farmacéutica.

El diseño de las entrevistas de tres etapas, desarrolladas por los autores, se encuentra en los Anexo: Entrevistas de tres etapas. A continuación, se describe cada etapa:

Etapa 1: Evaluación del estado actual. En esta primera fase, se exploró la situación actual de las empresas participantes en relación con el desarrollo de operaciones CRUD y la incorporación de nuevos desarrolladores. Se profundizó en aspectos como el tiempo de capacitación necesario, la curva de aprendizaje, la calidad del software resultante y los desafíos más comunes enfrentados en estos procesos.

Etapa 2: Introducción al concepto de *Citizen Developer*, en la segunda etapa, se presentó el concepto de *Citizen Developers* y se exploró cómo esta nueva figura podría impactar en el área de desarrollo de software de las empresas participantes.

Etapa 3: Presentación de la solución y prototipos, se presentó el objetivo principal del proyecto y los prototipos desarrollados. A través de esta presentación se buscó obtener opiniones sobre la potencial implementación de *CITDEV Express*.

Principales hallazgos: a partir de las entrevistas realizadas, se identificaron varios hallazgos clave:

- Capacitación y curva de aprendizaje: todos los entrevistados coincidieron en que la capacitación de nuevos desarrolladores en el stack tecnológico actual consume una cantidad significativa de tiempo y recursos, afectando la productividad inicial.
- Déficit de mano de obra calificada: existe una escasez notable de desarrolladores con las habilidades necesarias, lo cual incrementa los costos de contratación y prolonga el proceso de selección.
- Pertenencia: existe una necesidad generalizada de optimizar y automatizar las tareas CRUD básicas, liberando tiempo y recursos para actividades más estratégicas.
- Interés en *"Citizen Developers"*: la mayoría de los entrevistados manifestaron interés en la figura del *"Citizen Developer"*, una figura poco conocida por ellos, que se presenta como una solución potencial para aliviar la carga de trabajo de los desarrolladores expertos. Esta tendencia permite la democratización del desarrollo de software, empoderando a los usuarios sin formación técnica para que puedan contribuir en la creación de soluciones de software, especialmente en tareas de menor complejidad.

- Aceptación de *CITDEV Express*: la propuesta de una solución que facilite la creación y gestión de operaciones CRUD fue bien recibida. Se destacó la necesidad de una interfaz intuitiva y fácil de usar que les permita capacitar rápidamente a nuevos empleados en tareas CRUD básicas.

Adicionalmente, según las entrevistas realizadas, se identificaron necesidades como:

- Desarrollo de funcionalidades avanzadas de análisis de datos.
- Implementación de mecanismos de inteligencia artificial.
- Integración especifica con software en el back office de las empresas de los entrevistados.
- Uso de bases de datos dispuestas en el stack tecnológico de las empresas de los entrevistados.

Sin embargo, estas necesidades no fueron consideradas, ya que excedían el alcance del proyecto y requerían una infraestructura tecnológica más avanzada y compleja, lo cual habría impactado tanto en el tiempo como en el presupuesto, aspectos que no estaban contemplados. Además, se priorizarán funcionalidades esenciales para la ejecución de operaciones CRUD, que están más alineadas con los objetivos específicos, y no en funcionalidades avanzadas.

3.1.1. Identificación de stakeholders

La identificación y análisis de los stakeholders (partes interesadas) es un paso crucial en el desarrollo de cualquier proyecto de ingeniería de software.

En la tabla 3.1 se presenta una matriz que resume la información clave sobre los stakeholders entrevistados para este proyecto, incluyendo su nombre, función/rol/cargo, empresa, correo electrónico, teléfono, disponibilidad (espacial y horaria), relevancia y experiencia.

Nombre	Función/Rol/Cargo	Empresa	Correo Electrónico	Disponibilidad (Espacial y horario)	Relevancia	Experticia
James M. Martínez	Gerente Estratégico	Nexura Internacional	jmartinez@nexura.com	Carrera 28 No. 5B-71, Cali	Alta	Alta
Javier Vargas Occa	Director de desarrollo e innovación	SSI	jvargas@ssi.com.co	Carrera 72 No. 10Bis–153, Cali	Alta	Alta
Jhonatham Samir Salgado	Desarrollador	Desarrollador Sr.	jhonnatham@pleno.digital	Calle 34 No. 94-39, Cali	Media	Alta
David Salazar	Engineering manager	Milo.io	: david.salazar@milo.io	Calle 37N No. 2BN- 93, Prados del Norte, Cali	Alta	Media
Camilo Muñoz	Analista Químico	Abbott	juancamilo.munoz@abbott.com	Carrera 1 A No. 46a-34, Cali	Alta	Media

Tabla 3.1: Matriz de stakeholders

La identificación y análisis de stakeholders permite:

- Comprender las diferentes perspectivas y necesidades de las partes involucradas en el proyecto.
- Priorizar las necesidades y expectativas de los stakeholders más relevantes.
- Desarrollar estrategias de comunicación y gestión de stakeholders efectivas.
- Mitigar los riesgos potenciales asociados con la falta de participación o apoyo de los stakeholders.

- Aumentar las posibilidades de éxito del proyecto al garantizar que se satisfacen las necesidades de todas las partes involucradas.

La matriz de stakeholders que se presenta sirve como punto de partida para la gestión de stakeholders en este proyecto. Por lo tanto, es importante mantener actualizada dicha matriz y realizar análisis periódicos para identificar nuevos stakeholders o cambios en las necesidades o expectativas de los stakeholders existentes.

3.1.2. Casos de uso

En el desarrollo del proyecto de grado, se han definido los casos de uso que se pude observar en la figura 3.1, los cuales han sido esenciales para estructurar y guiar el diseño del sistema. Los casos de uso proporcionaron una base sólida para entender cómo interactuarán los actores principales — Admin, CitDev y CRUD, todos ellos generalizaciones de Usuario — con el sistema, así como las interacciones con sistemas externos a través del API. Además, la definición precisa de estos casos de uso ha permitido asegurar que todas las funcionalidades críticas sean contempladas y correctamente implementadas.

Para el Admin, se han identificado los casos de uso de Autenticación, Gestión de usuarios y roles, asegurando el control y la seguridad del acceso al sistema. Para el CitDev, se definió el caso de uso de Gestión dinámica de entidades, permitiendo la administración y configuración de las estructuras de datos. El CRUD, por su parte, se encargó de la Gestión de registros de entidades dinámicas, facilitando la manipulación de los datos almacenados. Todos estos casos de uso extienden del caso de uso base de "Gestión", que define las acciones esenciales como: Listar registros, Crear registro, Ver registro, Actualizar registro y Eliminar registro.

Finalmente, se ha incluido el caso de uso de Auditoría, que se encarga de mantener la trazabilidad de todas las acciones gestionadas en el sistema. Este caso de uso ha sido crucial para asegurar la integridad y el seguimiento de las operaciones realizadas, proporcionando un historial detallado de las actividades y contribuyendo significativamente a la transparencia y seguridad del sistema.

3.1.3. Definición y análisis de requisitos funcionales

La matriz de requisitos fue esencial para el éxito de este proyecto, estableciendo una directriz para el desarrollo del sistema y asegurando que cumpliera con las expectativas de los usuarios. Facilitó la alineación de expectativas entre todos los involucrados y el equipo de desarrollo, evitando malentendidos y desviaciones del alcance original del proyecto. Además, la matriz sirvió como una guía invaluable para la gestión del desarrollo del software proporcionando una visión clara de las características y comportamientos esperados del sistema. Al priorizar los requisitos y asignar recursos de manera eficiente, se optimizó el proceso de desarrollo y se garantizó la implementación de las funcionalidades más importantes en las fases iniciales. También fue fundamental para la validación del producto final, verificando que el sistema cumpliera con las especificaciones establecidas y contribuyendo al logro de los objetivos del proyecto.

La tabla 3.2 presenta los requisitos funcionales del sistema, describiendo las funcionalidades que este debe ofrecer.

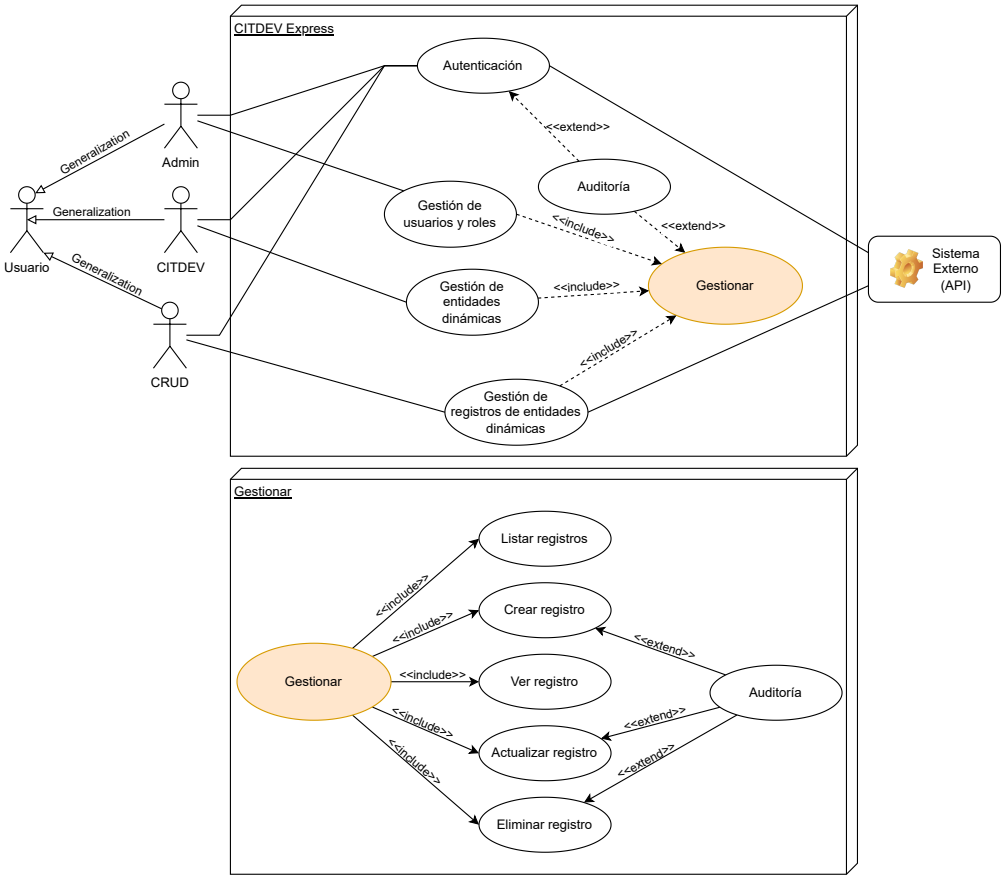


Figura 3.1: Diagrama de casos de uso

Estructuralmente, la tabla esta compuesta de las siguientes columnas:

- Identificador (♯RQM): es un código único asignado a cada requisito funcional. Su propósito es facilitar la referencia, mantener la trazabilidad, la organización y la comunicación eficiente entre los miembros del equipo de desarrollo sobre cada uno de los requisitos a lo largo del ciclo de vida del proyecto, permitiendo una gestión efectiva de los cambios.
- Requisito: clara y concisa que especifica la característica o cualidad que el sistema debía poseer. Esta claridad aseguró que todos los interesados comprendieran exactamente qué se necesitaba, ayudando en la evaluación y validación del cumplimiento del requisito.
- Prioridad: indica el nivel de importancia de cada requisito funcional, clasificándolo como alta, media o baja. Esta clasificación permitió asignar recursos y esfuerzos de manera eficiente, enfocando los esfuerzos en los aspectos más críticos en las primeras fases del desarrollo.
- Caso de uso: Identifica la categoría o el contexto en el que se aplica el requisito funcional dentro del sistema. Esta clasificación facilita la segmentación y gestión de los requisitos en función

de su relación con los procesos clave del sistema. Los valores posibles para esta columna son:

- Autenticación.
- Gestión de usuarios y roles.
- Gestión de entidades dinámica.
- Gestión de registros.
- Servicios Web (API).

♯RQM	Requisito	Caso de uso	Prioridad
RF01	*CITDEV Express* debe permitir el ingreso solamente con usuario y contraseña	Autenticación	ALTA
RF02	Cuando no hay una sesión de usuario iniciada, *CITDEV Express* debe redireccionar al usuario a la funcionalidad de inicio de sesión.	Autenticación	MEDIA
RF03	En la funcionalidad de inicio de sesión, l*CITDEV Express* debe mostrar un formulario al usuario para que ingrese sus credenciales de acceso.	Autenticación	MEDIA
RF04	Cuando se crea una sesión correctamente, *CITDEV Express* debe redireccionar al usuario a la funcionalidad del panel de administración.	Autenticación	ALTA
RF05	Cuando se tiene una sesión de usuario iniciada, *CITDEV Express* debe ofrecer una opción mediante la cual el usuario pueda cerrar su sesión.	Autenticación	BAJA
RF06	En el panel de administración, *CITDEV Express* debe mostrar las diferentes opciones disponibles al usuario para que pueda gestionar sus respectivos datos.	Gestión de usuarios y roles	MEDIA
RF07	En listar usuarios, *CITDEV Express* debe mostrar un listado con los registros existentes al usuario para que los pueda modificar, eliminar o crear otros registros.	Gestión de usuarios y roles	MEDIA
RF08	En la funcionalidad de crear usuario, *CITDEV Express* debe mostrar un formulario al usuario mediante el cual pueda ingresar la información del nuevo registro.	Gestión de usuarios y roles	MEDIA
RF09	En la opción de modificar usuario, *CITDEV Express* debe mostrar al usuario un formulario prediligenciado con los datos del registro.	Gestión de usuarios y roles	MEDIA

♯RQM	Requisito	Caso de uso	Prioridad
RF10	Cuando el formulario de crear usuario es enviado, *CITDEV Express* debe validar que el correo electrónico ingresado sea único y mostrar al usuario un mensaje de error si ya existe o una confirmación del registro exitoso.	Gestión de usuarios y roles	MEDIA
RF11	Cuando se hace clic en el botón de eliminar usuario, *CITDEV Express* debe validar que el registro existe y mostrar al usuario un mensaje de error si el registro no existe o una confirmación de eliminación.	Gestión de usuarios y roles	MEDIA
RF12	En la gestión de colecciones, *CITDEV Express* debe ofrecer una vista con la lista de registros de colecciones existentes para que los pueda crear, modificar, eliminar, gestionar campos y ver documentación de la API.	Gestión de entidades dinámica	ALTA
RF13	En crear colección, *CITDEV Express* debe mostrar un formulario al usuario mediante el cual puede ingresar la información del nuevo registro.	Gestión de entidades dinámica	ALTA
RF14	En la opción de modificar colección, *CITDEV Express* debe mostrar al usuario un formulario prediligenciado con los datos del registro.	Gestión de entidades dinámica	ALTA
RF15	Cuando se hace clic en el botón de eliminar colección, *CITDEV Express* debe validar que el registro existe y mostrar al usuario un mensaje de error si el registro no existe o una confirmación de eliminación.	Gestión de entidades dinámica	BAJA
RF16	Desde la funcionalidad de modificar colección, *CITDEV Express* debe ofrecer una opción mediante la cual el usuario pueda habilitar la interoperabilidad con otros sistemas al configurar los servicios web de obtener registro por identificador, listar, crear, modificar y eliminar un registro.	Gestión de entidades dinámica	MEDIA
RF17	En la configuración de campos de una colección, *CITDEV Express* debe ofrecer una opción mediante la cual el usuario pueda adicionar o eliminar dinámicamente los campos de una colección.	Gestión de entidades dinámica	MEDIA
RF18	Cuando se hace clic en el botón de eliminar campo, *CITDEV Express* debe mostrar una ventana emergente de confirmación al usuario para aceptar o rechazar la eliminación del campo.	Gestión de entidades dinámica	BAJA

♯RQM	Requisito	Caso de uso	Prioridad
RF19	En listar registros de un crud dinámico, *CITDEV Express* debe mostrar el listado de los registros existentes al usuario para que los pueda modificar, eliminar o crear otros registros.	Gestión de registros	ALTA
RF20	En crear registro de un CRUD dinámico, debe mostrar al usuario un formulario mediante el cual puede ingresar la información del nuevo registro.	Gestión de registros	ALTA
RF21	En modificar registro de un CRUD dinámico, *CITDEV Express* debe mostrar al usuario un formulario prediligenciado con los datos actuales del registro.	Gestión de registros	MEDIA
RF22	Cuando se hace clic en el botón de eliminar registro de un CRUD dinámico, *CITDEV Express* debe validar que el registro existe y mostrar al usuario un mensaje de error si el registro no existe o una confirmación de la eliminación.	Gestión de registros	BAJA
RF23	Cuando se llama el servicio web de inicio de sesión, *CITDEV Express* debe validar que las credenciales enviadas sean correctas y que pertenezcan a un usuario activo para entregar al consumidor un JSON Web Token válido.	Servicios Web (API)	BAJA
RF24	Cuando se llama a cualquier servicio web del CRUD dinámico, *CITDEV Express* debe verificar si el JSON Web Token enviado por el consumidor es válido y no ha expirado.	Servicios Web (API)	BAJA
RF25	Cuando se llama a cualquier servicio web del CRUD dinámico, *CITDEV Express* debe verificar si la operación solicitada ha sido habilitada por el usuario antes de entregar los datos; de lo contrario, entregará un error indicando que el servicio no está disponible.	Servicios Web (API)	BAJA
RF26	Cuando se llama el servicio web listar del CRUD dinámico, *CITDEV Express* debe entregar un listado al consumidor con los registros en formato JSON.	Servicios Web (API)	ALTA
RF27	Cuando se llama el servicio web crear del CRUD dinámico, *CITDEV Express* debe validar que los datos enviados por el consumidor correspondan con los campos configurados para la colección o CRUD, y se entregará un mensaje en formato JSON indicando si el resultado de la operación fue exitoso o si se presentó algún error.	Servicios Web (API)	ALTA

♯RQM	Requisito	Caso de uso	Prioridad
RF28	Cuando se llama al servicio web modificar del CRUD dinámico, *CITDEV Express* debe validar que el registro existe y los datos enviados por el consumidor corresponden a los campos configurados para la colección o CRUD, y se entregará un mensaje en formato JSON indicando si el resultado de la operación fue exitoso o si se presentó algún error.	Servicios Web (API)	MEDIA
RF29	Cuando se llama al servicio web eliminar del CRUD dinámico, *CITDEV Express* debe validar que el registro existe y entregar al consumidor un mensaje en formato JSON indicando si el resultado fue exitoso o si se presentó algún error.	Servicios Web (API)	BAJA
RF30	Cuando se llama el servicio web de consulta por identificador del CRUD dinámico, *CITDEV Express* debe validar que el registro existe y entregar al consumidor los datos del registro en formato JSON.	Servicios Web (API)	MEDIA
RF31	Cuando se hace clic en el botón ver documentación de la API, *CITDEV Express* debe mostrar el contrato con el estándar OpenAPI con la operación de login de la API y las operaciones CRUD de la entidad dinámica.	Gestión de entidades dinámica	BAJA

Tabla 3.2: Listado de requisitos funcionales.

3.1.4. Definición y análisis de requisitos no funcionales

Con la finalidad de que *CITDEV Express* no solo cumpliera con las funcionalidades esperadas, sino que también operara de manera eficiente y eficaz en un entorno de producción, se elaboró la tabla 3.3 de requisitos no funcionales. Estos requisitos abarcaban aspectos críticos como el rendimiento, la interoperabilidad, la seguridad y la usabilidad del sistema, entre otros.
La tabla conserva la estructura de la tabla de requisitos funcionales en cuanto a columnas de identificación, requisito y prioridad, pero además incorpora la columna Atributo de calidad, que describe la característica específica del sistema que se busca mejorar o garantizar. De esta forma, cada requisito se alinea con un objetivo de calidad concreto, lo cual ayuda a identificar áreas críticas que requieren atención especial.

La importancia de esta tabla radicó en su capacidad para:

- La gestión de los requisitos no funcionales a lo largo del ciclo de vida del proyecto.
- Definir atributos de calidad en rendimiento, seguridad, usabilidad, compatibilidad e interoperabilidad.
- Evaluar el sistema mediante métricas que identifiquen áreas de mejora.

- Garantizar la satisfacción del usuario al asegurar una experiencia de uso positiva.
- Reducir riesgos, anticipando problemas potenciales en cuanto al rendimiento y la calidad.
- Optimizar el desarrollo, priorizando los aspectos que más afectan la calidad final del producto.

♯RQM	Atributo calidad	Requisito	Criterios de Aceptación	Prioridad
RNF01	Compatibilidad	El sistema debe ser compatible con navegadores (Chrome, Firefox, Safari, Edge) y funcionar correctamente en laptops, tablets y smartphones; la interfaz debe ajustarse a diferentes resoluciones y tamaños de pantalla.	- Pruebas de renderizado exitosas en navegadores y dispositivos móviles. - Diseño responsivo que garantice la accesibilidad básica.	MEDIA
RNF02	Fiabilidad	El sistema debe validar entradas, manejar excepciones y registrar fallos sin afectar la estabilidad global. Debe poder recuperarse de errores sin pérdida de datos críticos.	- Logs de errores de fácil consulta y diagnóstico. - Manejo seguro de datos corruptos y mensajes claros de error en la interfaz.	ALTA
RNF03	Rendimiento	El sistema debe soportar al menos 5 solicitudes concurrentes por segundo sin degradación significativa; las consultas a la base de datos deben ser eficientes para que el tiempo de respuesta no supere los 2000 ms en promedio.	- Pruebas de carga con resultados máximo de 2000 ms en el 90 % de las transacciones.	ALTA

♯RQM	Requisito	Atributo calidad	Criterios de Aceptación	Prioridad
RNF04	Interoperabilidad	La solución debe permitir el intercambio de información con sistemas externos, utilizando JSON como formato de comunicación. Deben existir pruebas de integración que garanticen la correcta transferencia de datos.	- APIs documentadas (OpenAPI/Swagger o similar). - Pruebas de integración con sistemas externos: envío/recepción de datos en JSON, validación exitosa de la respuesta.	MEDIA
RNF05	Seguridad (No repudio)	El sistema debe generar registros de auditoría y permitir la trazabilidad de acciones de los usuarios, almacenando información relevante para cumplir con normas legales o políticas internas.	- Registro automático de accesos y acciones (fechas, usuarios y detalles). - Almacenamiento seguro de bitácoras, accesible para eventuales inspecciones internas o externas.	ALTA
RNF06	Seguridad	El sistema debe gestionar roles y privilegios para cada usuario, asegurando que solo los autorizados puedan ejecutar determinadas acciones. Las contraseñas deben almacenarse de manera segura (hashing).	- Tabla de roles y permisos claramente definida. - Contraseñas no en texto plano, verificación de que el método de cifrado cumpla con estándares de seguridad (p. ej., bcrypt, Argon2).	ALTA
RNF07	Usabilidad	Las funciones principales deben ser accesibles con un máximo de 3 clics desde la página inicial; la interfaz debe ser intuitiva y accesible para usuarios no técnicos, realizándose pruebas de experiencia de usuario y accesibilidad.	- Pruebas de usabilidad para confirmar máximo de 3 clics a funciones clave. - Retroalimentación de usuarios no técnicos en pruebas piloto.	MEDIA

Tabla 3.3: Listado de requisitos no funcionales.

3.1.5. Integración de principios de usabilidad

Durante la elaboración de este trabajo de grado, se tuvieron en cuenta los puntos en común entre los principios básicos de la heurística de usabilidad de Jakob Nielsen y los principios de diseño de Don Norman, dos enfoques fundamentales en la experiencia de usuario (UX). Al analizar cómo ambos convergen en áreas clave como la visibilidad, consistencia, prevención de errores, simplicidad y apoyo al usuario, se busca proporcionar un marco que optimice la usabilidad y la satisfacción en el diseño de interfaces. A continuación, se resalta la importancia de integrar ambos conjuntos de principios para crear experiencias digitales efectivas y accesibles en *CITDEV Express* (Nielsen, 1994; Norman, 2013).

De ambas metodologías nos centraremos en:

- Mantener a los usuarios informados mediante visibilidad y retroalimentación.
- Facilitar la comprensión y el uso del sistema mediante consistencia, descubribilidad (capacidad de descubrimiento) y reconocimiento en lugar de recuerdo.
- Prevenir errores y mejorar la eficiencia mediante restricciones, usabilidad intuitiva y flexibilidad
- Mantener la simplicidad y claridad en el diseño a través del minimalismo y la estética.
- Proporcionar apoyo adicional mediante ayuda y documentación accesible.

En el desarrollo de *CITDEV Express* como una solución tecnológica dirigida a los *Citizen Developers*, se implementaron cinco puntos en común entre los principios de Don Norman y las heurísticas de Jakob Nielsen. Estos principios fueron seleccionados por su capacidad para mejorar la experiencia del usuario y garantizar la eficiencia y efectividad del sistema, especialmente en la creación de aplicaciones *CRUD* (Crear, Leer, Actualizar, Eliminar).

3.1.5.1. Mantener a los usuarios informados mediante visibilidad y retroalimentación

La visibilidad del estado del sistema y la retroalimentación constante se consideran esenciales para mantener a los usuarios informados sobre lo que está ocurriendo en todo momento. Esto se logra mediante indicadores visuales, mensajes de confirmación y/o notificaciones que aseguran que los usuarios no se sientan confundidos durante su interacción con *CITDEV Express*.

Ejemplo: En *CITDEV Express*, cada vez que un usuario crea, actualiza o elimina un registro, el sistema muestra una notificación emergente que confirma la acción y actualiza la lista de registros en tiempo real. Además, la navegación Breadcrumb (rastro de migas de pan) permite que el usuario se oriente fácilmente dentro de *CITDEV Express*, mostrando su ubicación exacta en todo momento. Complementariamente, los mensajes claros y directos ayudan a evitar confusiones al proporcionar información sobre el resultado de sus acciones.

3.1.5.2. Facilitar la comprensión y el uso del sistema mediante consistencia, intuición y reconocimiento en lugar de recuerdo. (estandarización)

La consistencia en el diseño y la utilización de elementos familiares facilita la comprensión y el uso del sistema. Al emplear patrones de diseño conocidos y asegurarse de que las funciones sean fáciles de encontrar, se reduce la carga cognitiva del usuario, permitiéndole reconocer acciones y elementos en lugar de tener que recordarlos.

Ejemplo: En *CITDEV Express*, los iconos de botones para "Crear", "Leer", "Actualizar" y "Eliminar" utilizan colores y ubicaciones consistentes en todas las interfaces de usuario. Además, se emplean iconos familiares (como un lápiz para editar y un basurero para eliminar) facilitando la navegación y el uso del sistema.

3.1.5.3. Prevenir errores y mejorar la eficiencia mediante restricciones, affordances (señales) y flexibilidad

Para prevenir errores y mejorar la eficiencia, se implementaron restricciones que limitan las acciones del usuario a aquellas que son válidas en el contexto actual. Además, se utilizaron señas claras que indican cómo interactuar con los elementos del sistema.

Ejemplo: En *CITDEV Express*, los formularios de creación y actualización de registros incluyen validaciones que alertan a los usuarios sobre errores en la entrada de datos (como campos obligatorios no completados). Además, se utilizan elementos del sistema con un diseño claro, como botones de acción grandes y bien etiquetados, y flexibilidad al crear un collection para nuevos formularios.

3.1.5.4. Mantener la simplicidad y claridad en el diseño a través del minimalismo y la estética

Un diseño minimalista y estéticamente agradable ayuda a mantener la simplicidad y claridad, lo que facilita la navegación y el uso del sistema. Al eliminar elementos innecesarios y enfocarse en lo esencial, se crea una interfaz limpia y fácil de entender.

Ejemplo: En *CITDEV Express*, la interfaz se mantiene minimalista, con un diseño limpio y estético. Solo se muestran los elementos esenciales, como la lista de registros y los botones de acción principales. Los colores suaves y el uso de espacios en blanco ayudan a mantener la claridad y a evitar la sobrecarga de información.

3.1.5.5. Proporcionar apoyo adicional mediante ayuda y documentación accesibles

La disponibilidad de ayuda y documentación accesible es crucial para apoyar a los usuarios en caso de dudas o problemas. Esto incluye tutoriales, guías de usuario y soporte en línea que están fácilmente disponibles y son comprensibles para todos los niveles de habilidad.

Ejemplo: En *CITDEV Express*, se incluye una sección de ayuda accesible desde cualquier página.

En conclusión, la familiaridad en el diseño permite que los usuarios interactúen de forma intuitiva con productos digitales, reduciendo la carga cognitiva al saber de inmediato cómo utilizarlos.

Al seguir convenciones de diseño comunes, se aprovecha la "Ley de Jakob", que indica que los usuarios desarrollan expectativas basadas en su experiencia previa acumulada con otros sitios. Esto les permite centrarse más en el objetivo del contenido, el mensaje o el producto del sitio, sin tener que aprender a usar una nueva interfaz (Yablonski, 2024). Los modelos mentales, basados en estas experiencias previas, también son clave: permiten que los usuarios predigan cómo funciona un sistema (Yablonski, 2024). Al respetar estos modelos, los usuarios pueden ser productivos de inmediato, lo que mejora la usabilidad y la satisfacción en soluciones tecnológicas, especialmente para *Citizen Developers*.

3.2. Estrategia para implementar las funcionalidades de gestión de operaciones simples considerando las habilidades de los *Citizen Developers*

A lo largo de esta sección, se presentan las estrategias aplicadas en *CITDEV Express* para la gestión de operaciones simples, detallando la configuración de entidades, las habilidades de los usuarios y la persistencia de los datos. Además, se explica cómo estas funcionalidades permiten a los *Citizen Developers* gestionar información sin necesidad de programar.

La implementación de esta estrategia responde a los requisitos funcionales RF07–RF16 y RF18–RF23, los cuales permiten la configuración de entidades y gestión de datos mediante una interfaz gráfica adaptable a las habilidades de los *Citizen Developers*. En cuanto a los requisitos no funcionales, se abordan RNF01, RNF02, RNF06 y RNF07, que garantizan la compatibilidad con diferentes dispositivos, la facilidad de uso y el almacenamiento de registros de auditoría. Esta sección también contribuye al cumplimiento del objetivo específico OE2[1], relacionado con la implementación de una solución tecnológica que optimice la gestión de operaciones *CRUD* sin redundancia de código y considerando las habilidades de los *Citizen Developers*.

3.2.1. Analogía de recetas

Con el fin de facilitar la comprensión del funcionamiento de las entidades dinámicas dentro de *CITDEV Express*, se presenta a continuación una analogía basada en un entorno cotidiano: un restaurante. Esta analogía permite representar, de manera conceptual, los roles, componentes y acciones que intervienen en la configuración y uso de las entidades dinámicas. Al establecer paralelismos entre elementos del sistema y funciones propias de un restaurante, se busca ilustrar cómo los *Citizen Developers* y usuarios finales interactúan con la solución, diferenciando claramente sus responsabilidades dentro del proceso. Esta representación no técnica facilita la apropiación del modelo incluso por parte de actores sin experiencia en desarrollo de software.

Imagina un restaurante especializado que busca ofrecer un menú personalizado a sus clientes acorde a su estilo o temática. En este restaurante existen diferentes roles clave para que todo

[1] OE2: Diseñar e implementar las funcionalidades de gestión de operaciones simples considerando las habilidades de los *Citizen Developers* evitando la redundancia del código fuente.

funcione de manera organizada y eficiente:

- El **dueño del restaurante**, quien representa el administrador del sistema, es quien define el estilo o la temática general del lugar, como si fuera italiano, vegetariano, comida rápida, etc. También es el encargado de contratar el personal y asignar el rol de **administrador del restaurante** o **chef** (crear y administrar los usuarios).
- El **administrador del restaurante** quien representa al *Citizen Developer*, es quien se encarga de definir cada tipo de plato (o colecciones dentro de *CITDEV Express*), tal como: las entradas, sopas, platos fuertes, postres, bebidas, etc. Según la temática del restaurante, decide qué ingredientes estarán disponibles para cada tipo de plato, asegurándose de que cada chef tenga solo lo necesario para crear la receta en esta parte del menú. De esta manera, el administrador del restaurante garantiza qué ingredientes y cómo son utilizados en las sopas o platos fuertes, para que se mantengan alineados con la temática del restaurante. Por ejemplo, en un restaurante de comida rápida, el administrador podría definir colecciones para hamburguesas, perros calientes y sándwiches; y los respectivos ingredientes para cada tipo. En otro caso, en un restaurante vegetariano, se definirán tipos de platos como ensaladas, sopas y platos fuertes y a su vez limitaría los ingredientes a aquellos de origen vegetal para crear sus recetas.
- El **chef** representa al usuario final del sistema, que luego pueden acceder a cada tipo de plato (colección) creado por al **administrador del restaurante**, y solo podrá usar los ingredientes permitidos para crear la receta del plato, como un plato fuerte o un postre. Los chefs pueden ser creativos dentro de los límites establecidos por el administrador del restaurante, pero no pueden mezclar ingredientes de un tipo de plato con otro. Por ejemplo, un chef podrá usar cebollas para crear una receta para tipos de plato como ensaladas o sopas, pero no para las bebidas puesto que no se definió de esta manera. Así, el chef solo usará el conjunto de ingredientes limitado y organizado por el administrador del restaurante (*Citizen Developer*) para cada tipo de plato.

La **Figura 3.2** ilustra visualmente esta analogía, evidenciando cómo se distribuyen las responsabilidades dentro del sistema.

Con esta estructura, el **dueño del restaurante** (administrador del sistema) establece la visión y gestiona el personal; el **administrador del restaurante** (*Citizen Developer*) define los tipos de platos y organiza los ingredientes permitidos para cada uno; y los **chefs** (usuario final) crean las recetas respetando las reglas y los ingredientes definidos, evitando combinaciones fuera de lugar. En resumen, esta estructura organiza el menú, ahorra tiempo, permite innovación dentro de un marco claro y da flexibilidad a quienes, sin ser expertos, pueden contribuir con recetas alineadas con los estándares del restaurante.

3.2.2. Habilidades de los *Citizen Developers*

Los *Citizen Developers* son usuarios con habilidades centradas en la gestión de datos y la configuración de sistemas a través de interfaces de usuario, sin requerir conocimientos de programación

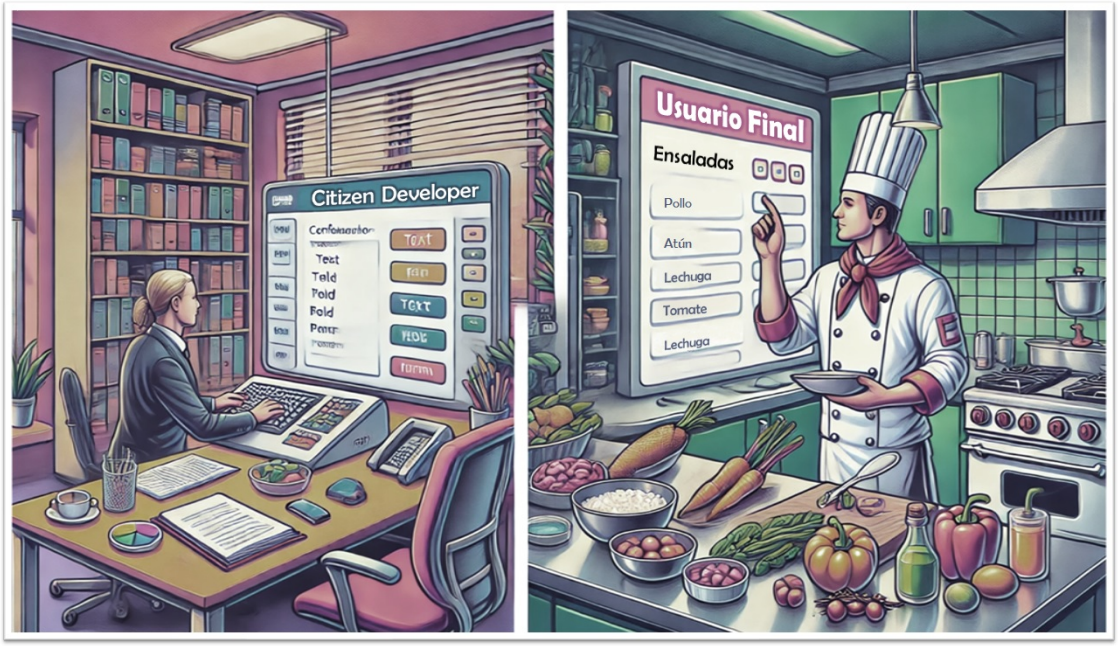


Figura 3.2: *Citizen Developer* & Usuario - Elaborado con IA generativa

(Hurlburt, 2021). *CITDEV Express* permite a estos usuarios acceder al sistema y gestionar entidades dinámicas de manera intuitiva.

De acuerdo con la analogía de recetas, el *Citizen Developer* actúa como un **administrador** que define los atributos (ingredientes) de un *CRUD* específico (tipo de plato), los cuales puede ajustar en cualquier momento de acuerdo a las necesidades.

Al definir los atributos de un *CRUD*, el *Citizen Developer* realiza, de manera implícita, una configuración en la base de datos. Esta configuración alimenta un componente del sistema que interpreta dicha información, generando automáticamente formularios para capturar datos y aplicar las validaciones necesarias sobre los atributos. Dichos datos pueden originarse de un formulario dinámico generado por *CITDEV Express* o a través de una interacción con la *API*.

Volviendo a la analogía, un *CRUD* representa un tipo de plato, y éste permite el uso de diferentes ingredientes según su propósito. Esto implica que los tipos de plato son heterogéneos y pueden variar entre sí. En términos de software, esto se traduce en estructuras de datos heterogéneas, que se adaptan a los atributos definidos en la gestión de entidades dinámicas.

3.2.3. Entidades

Dado que las estructuras de datos son heterogéneas y se necesita implementar una gestión simple de operaciones para el *Citizen Developer*, la estrategia para lograr esta flexibilidad y adaptabilidad se basa en el uso de **entidades dinámicas**, facilitando así la configuración y administración de los datos sin requerir codificación.

En el ámbito de las bases de datos, una entidad representa un conjunto de datos organizados para facilitar su almacenamiento y manipulación. En las bases de datos relacionales, este concepto se materializa como tablas, que almacenan registros en filas y columnas con un esquema predefinido (Date, 2004). En cambio, en las bases de datos no relacionales como MongoDB, los datos se organizan en colecciones, las cuales agrupan documentos con estructuras flexibles que no requieren seguir un esquema fijo (Chodorow, 2019). Esta flexibilidad permite manejar tipos de datos heterogéneos, adaptándose a las necesidades específicas de cada aplicación. En la solución tecnológica *CITDEV Express*, existen dos tipos de entidades: estáticas y dinámicas.

En la **Figura 3.3** se observa a modo de ejemplo que la entidad estática podría representarse como un rompecabezas terminado, donde las piezas encajan de una única manera y no se pueden hacer cambios. Por otra parte, la entidad dinámica podría representarse como un conjunto de piezas tipo bloques, listos para ensamblarse y generar objetos, con la capacidad de agregar y/o quitar piezas fácilmente en cualquier momento.

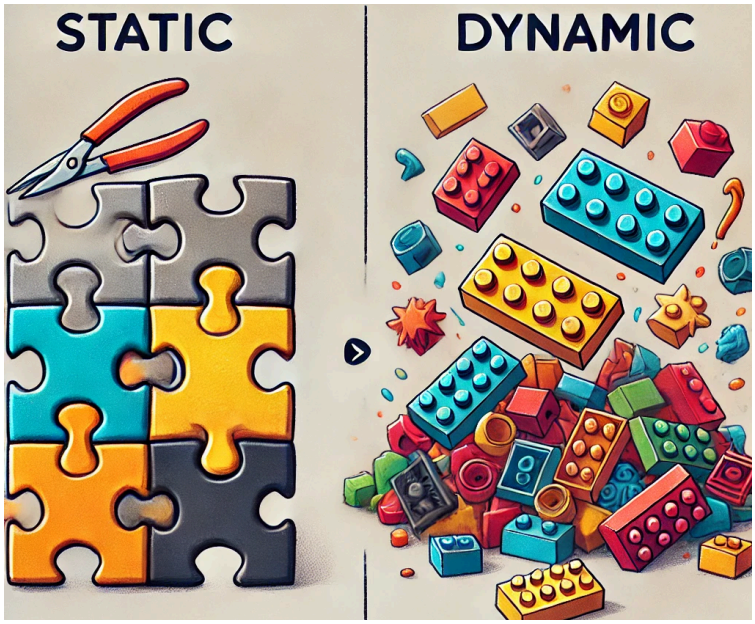


Figura 3.3: Entidad estática vs entidad dinámica - Elaborado con IA generativa

A continuación se explicarán los detalles de cada tipo de entidad.

3.2.3.1. Entidades estáticas

Las entidades estáticas son aquellas cuya estructura está predefinida y escrita directamente en el código fuente de *CITDEV Express*. Debido a que su esquema se encuentra codificado, no puede ser modificado dinámicamente por los *Citizen Developers* y cualquier cambio en su estructura requiere la intervención de un desarrollador. Estas entidades se utilizan en funcionalidades clave del sistema, tales como la gestión de usuarios, el log de auditoría y la gestión de entidades dinámicas. En la

Figura 3.4 se representa la relación de composición entre la colección de logs de auditoría (CoreAuditLog) y la colección de usuarios (CoreUsers). Esta relación se justifica en la medida en que los registros de auditoría dependen directamente de las acciones realizadas por los usuarios, quienes actúan sobre los distintos recursos del sistema. Por otro lado, entre la colección *CoreAuditLog* y la colección de configuraciones para la gestión de entidades dinámicas (CoreCollections) se establece una relación de agregación, dado que *CoreAuditLog* puede almacenar eventos relacionados con modificaciones en la configuración de las entidades, sin que ello implique una dependencia estructural directa entre ambas colecciones.

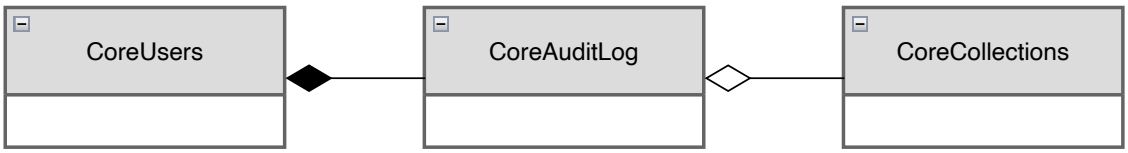


Figura 3.4: Colecciones de entidades estáticas

La naturaleza rígida de este tipo de entidades responde a la necesidad de estabilidad, donde la consistencia y seguridad son prioritarias. Al estar definidas en el código, las entidades estáticas garantizan que el sistema funcione de manera confiable con una estructura fija que no puede ser alterada en cualquier momento.

3.2.3.2. Entidades dinámicas

Las entidades dinámicas ofrecen una mayor flexibilidad y adaptabilidad dado que no tienen un esquema predefinido en el código fuente. En lugar de eso, su configuración se realiza dinámicamente a través de una interfaz gráfica. Esto permite que los *Citizen Developers* definan los atributos necesarios para cada entidad sin necesidad de escribir código fuente, considerando que esta habilidad no hace parte del conjunto de habilidades detectadas para su rol.

Cada vez que se configura una entidad dinámica, los atributos definidos forman un esquema que se genera en memoria cuando es necesario interactuar con los datos de dicha entidad. A diferencia de las entidades estáticas, este esquema no hace parte del código fuente, sino que se crea y ajusta en tiempo de ejecución conforme a las configuraciones realizadas por el *Citizen Developer*. Este enfoque dinámico permite a *CITDEV Express* adaptarse a las necesidades de negocio en constante evolución, ya que los esquemas pueden modificarse de manera fácil y rápida a medida que cambian los requisitos.

Es importante destacar que tanto los esquemas de las entidades estáticas como los de las dinámicas se crean en memoria y están disponibles para ser utilizados cada vez que se necesiten. La diferencia principal radica en que, en el caso de las entidades dinámicas, si la configuración se actualiza, el esquema en memoria se elimina y se regenera con la nueva configuración en tiempo de ejecución.

3.2.4. Configuración de una entidad dinámica

A continuación se describe cómo se configura una entidad dinámica en *CITDEV Express* y en la **Sección 3.2.5** se muestra un ejemplo de la misma con los datos de un *CRUD*:

3.2.4.1. Estructura general

- **_id**: identificador único de la entidad dinámica, generado automáticamente por el motor de base de datos.
- **path_name**: ruta que se utiliza para acceder a los servicios de esta entidad, como un endpoint de API.
- **collection_name**: nombre de la colección de objetos que almacenará los registros en la base de datos.
- **title**: título descriptivo para identificar la entidad en la interfaz de usuario.
- **showAdmin**: indica si la gestión de registros para la entidad dinámica será visible en la interfaz administrativa. Por defecto no se visualiza dado que al crear la entidad aún no tiene campos definidos.

3.2.4.2. Formulario de configuración (*form*)

El formulario está compuesto por una lista de campos (*fields*) con los siguientes atributos:

- **name**: nombre técnico del campo.
- **type**: tipo de dato HTML (por ejemplo: Texto, Lista desplegable, Numérico, etc).
- **label**: título visible en la interfaz de usuario.
- **cols**: configuración visual de la interfaz basada en un sistema de rejillas para indicar qué espacio ocupa el campo.
- **default_value**: valor predeterminado que tendrá el campo.
- **projection**: indica si el campo será incluido en las vistas de consulta de registros.
- **others**: configuraciones adicionales:
  - **rules**: reglas de validación, como `required` para que el campo sea obligatorio.
  - **options**: configuración de valores disponibles para los campos que poseen varias opciones de respuesta, como es el caso de las *Casillas de selección*, *Lista desplegable*, *Opción única*, etc. Esta funcionalidad permite que la información se pueda extraer desde otras colecciones de la base de datos, es decir, se pueden relacionar entidades dinámicas con otras entidades estáticas o dinámicas ya que dichas entidades relacionadas solo se usan para leer información.

- **type**: se indica "CUSTOM" para valores definidos manualmente y separados por coma o "COLLECTION" para valores que se encuentran en una colección de la base de datos.
- **collection_name**: si las opciones provienen de otra colección, aquí se indica su nombre. Por defecto no hay una colección asignada.
- **values**: si es tipo "CUSTOM", solo se agregan las diferentes opciones separadas por coma. Si es tipo "COLLECTION", se debe indicar un par de atributos de la colección indicada separados por coma, donde el primer atributo corresponde al dato que se desea almacenar y el segundo al dato que se desea mostrar.

- **config**: se encuentra la configuración del tipo de dato en la base de datos (*database_type*), por ejemplo `String`, `Number`, etc.

3.2.4.3. Configuración para mostrar información (*collectionConfig*)

- **projection**: define qué campos se mostrarán por defecto en las listas de registros.
- **labels**: etiquetas asociadas a los campos de proyección (en el mismo orden) para mostrarlas de forma amigable al usuario.

3.2.4.4. Servicios permitidos (*allow_services*)

Este atributo es un objeto que define las operaciones CRUD habilitadas vía API. Todos los atributos son de tipo *boolean* y por defecto tienen asignado el valor *false*. En la **Sección 3.3** se presentan los detalles sobre la interoperabilidad, lo cual está relacionado con la configuración de los *Servicios permitidos*.

- **list**: permite listar todos los registros.
- **getById**: permite consultar un registro por su identificador único.
- **create**: permite crear nuevos registros.
- **update**: permite modificar registros existentes.
- **delete**: permite eliminar registros.

3.2.5. Resultado de la configuración de una entidad dinámica

La **Figura 3.7** ilustra un ejemplo de la estructura completa de un objeto JSON, configurado para representar una entidad dinámica que almacena la información de los equipos que participarán en la próxima Copa Libertadores.

La información se visualiza organizada mediante tablas, atributos y relaciones:

- Las tablas representan objetos, que están compuestos por atributos con tipos de datos primarios (String, Number, Boolean, etc.) y complejos (Array, Object, etc).

- Los atributos de tipo primario presentan directamente el valor que almacenan.
- Los atributos de tipo complejo se identifican con un punto que conecta, mediante una flecha, a una tabla secundaria que define la estructura del objeto correspondiente.
- Las relaciones se representan mediante flechas de línea discontinua.
- Para una configuración de un solo atributo en una entidad dinámica se requieren 11 estructuras de objetos en el JSON. Por cada uno de los atributos configurados (*form.fields*) se requieren 5 objetos. Por lo tanto, en la **Figura 3.7** se aprecian 36 objetos en total.

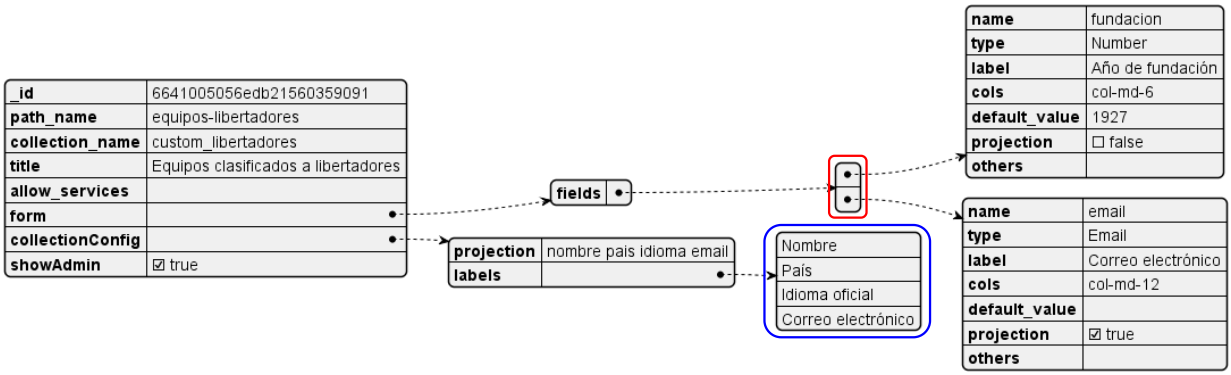


Figura 3.5: Atributo que almacena una lista de objetos

La **Figura 3.5** muestra el caso específico del atributo *form.fields* y *collectionConfig.labels*:

- El atributo *form.fields* es de tipo complejo y su estructura apunta a una tabla (marcada en rojo) donde los elementos individuales son representados por puntos que a su vez se relacionan con otros objetos. Esto significa que el atributo *form.fields* es de tipo *Array* y almacena una lista de objetos.
- El atributo *collectionConfig.labels* es de tipo complejo y está relacionado a una tabla de valores (marcada en azul). Esto indica que su tipo de dato es un *Array*, el cual almacena una lista de elementos de tipo primitivo. En este caso se trata de cadenas de texto (*String*).

A continuación se presenta un fragmento del objeto JSON que corresponde al de la **Figura 3.7**:

```
{
  "_id": "6641005056edb21560359091",
  "path_name": "equipos-libertadores",
  "collection_name": "custom_libertadores",
  "title": "Equipos clasificados a libertadores",
  "allow_services": {
    "list": true,
    "getById": true,
```

```
    "create": true,
    "update": true,
    "delete": false
  },
  "form": {
    "fields": [
      {
        "name": "nombre",
        "type": "Text",
        "label": "Nombre",
        "cols": "col-md-12",
        "default_value": "América de Cali",
        "projection": true,
        "others": {
          "rules": {
            "required": true
          },
          "options": {
            "type": "CUSTOM",
            "values": "OPTION 1,OPTION 2,OPTION 3",
            "collection_name": ""
          },
          "config": {
            "database_type": "String"
          }
        }
      }
    ]
  },
  "collectionConfig": {
    "projection": "nombre pais idioma email",
    "labels": ["Nombre", "País", "Idioma oficial", "Correo electrónico"]
  },
  "showAdmin": true
}
```


3.2.5.1. Ejemplo práctico: Gestionar equipos de fútbol

Para gestionar la información de los equipos de fútbol que participarán en la próxima Copa Libertadores de América, se deben almacenar los siguientes datos de cada equipo: nombre, país que representa, idioma oficial, método de clasificación (directa o por eliminatoria), año de fundación y un correo electrónico de contacto.

El país se seleccionará desde una lista desplegable, cuyos valores se obtendrán a partir de otra colección. El idioma se elegirá desde una lista desplegable con las opciones “Español” y “Portugués”. El tipo de clasificación se seleccionará utilizando opciones de tipo radio, con los valores “Directa” y

"Eliminatoria".

El nombre del equipo, el año de fundación y el correo electrónico se capturarán en campos de texto. Además, el año de fundación se almacenará en la base de datos como un dato numérico, mientras que los demás datos se almacenarán como cadenas de texto.

3.2.5.2. Solución del ejemplo práctico

En la **Figura 3.6** se muestra un diagrama a alto nivel que extiende la representación previamente presentada en la **Figura 3.4**, incorporando dos colecciones dinámicas, *CustomLibertadores* y *CustomCountries* (resaltadas en color rojo), las cuales ejemplifican la solución al caso práctico.

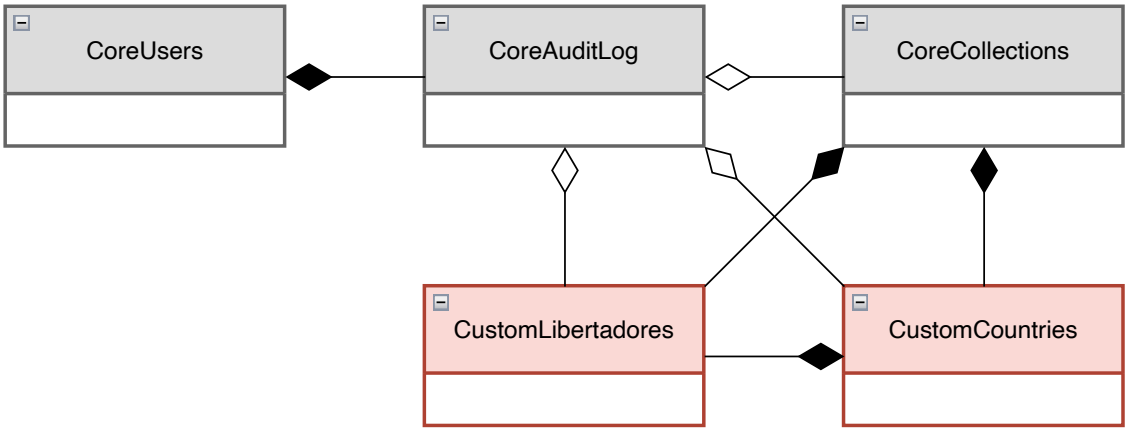


Figura 3.6: Colecciones estáticas y dinámicas del ejemplo práctico

Estas nuevas entidades dinámicas permiten observar cómo se materializa la configuración realizada a través de la colección *CoreCollections*. En este contexto, se establece una relación de composición entre *CoreCollections* y cada una de las colecciones dinámicas, lo cual implica que estas últimas no pueden existir sin una configuración previa en *CoreCollections*, de la cual dependen estructuralmente para definir sus atributos y comportamientos.

Además, se mantiene la relación de agregación entre *CoreAuditLog* y las colecciones dinámicas, dado que los registros de auditoría capturan eventos derivados de la interacción de los usuarios sobre estas entidades configuradas, sin que exista una dependencia estructural entre ellas. Asimismo, se observa una relación de composición entre *CustomLibertadores* y *CustomCountries*, en donde la existencia de registros en *CustomLibertadores* requiere la presencia de valores previamente definidos en *CustomCountries*, ya que estos son utilizados como opciones en un campo de tipo *Lista desplegable* de un formulario para capturar la información.

Las entidades dinámicas representadas en la **Figura 3.6** siguen la estructura descrita en la **Sección 3.2.4**, donde se detallan los atributos que las conforman. En particular, la **Figura 3.7** ilustra la configuración completa de la colección *CustomLibertadores*, evidenciando cómo se definen los campos, las reglas de validación y las opciones de interoperabilidad asociadas a esta entidad dinámica.

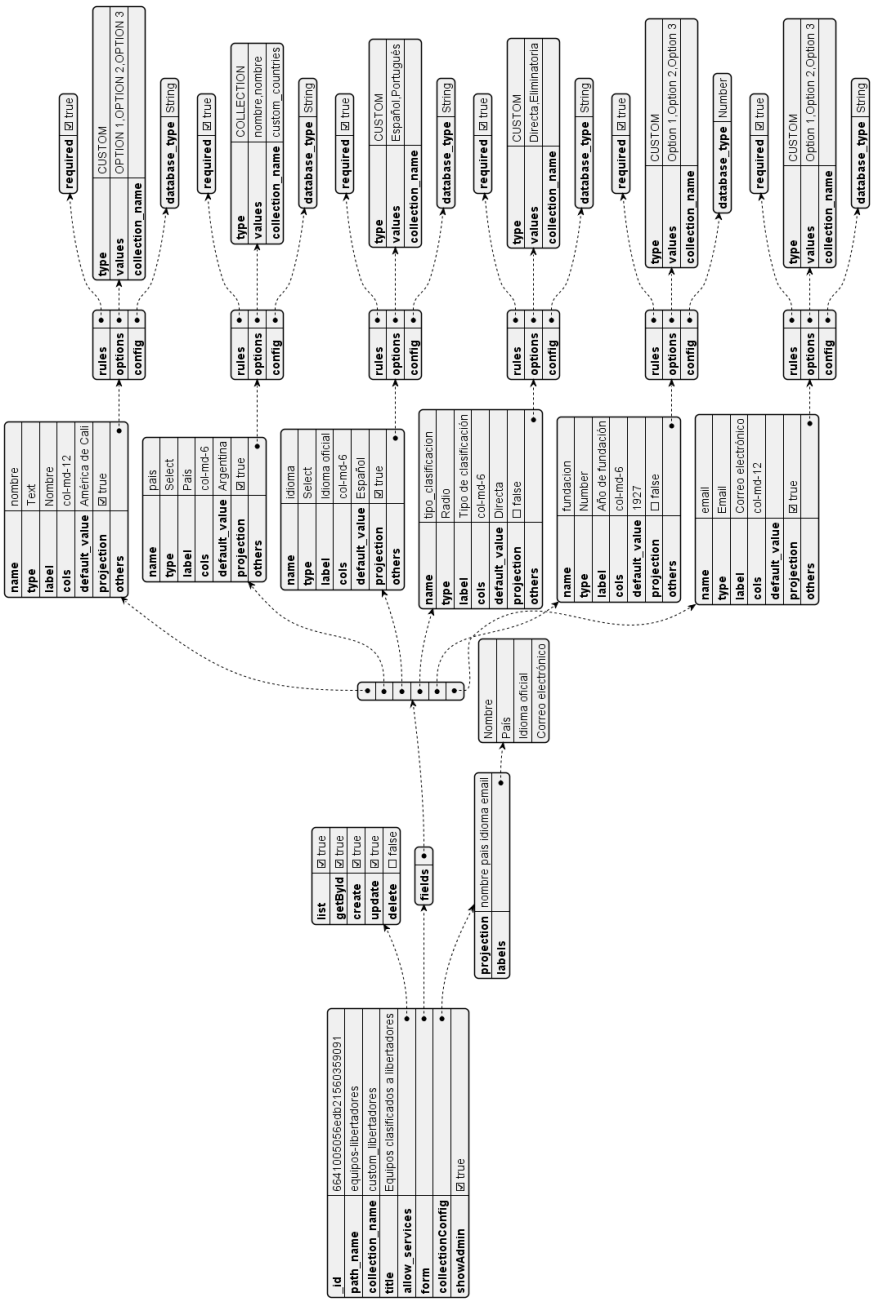


Figura 3.7: Configuración de la entidad dinámica

3.2.6. Persistencia de datos con una entidad dinámica

En la **Sección 3.2.4** se explicó cómo se configura una entidad dinámica, mientras que en la **Sección 3.2.5** se presentó un ejemplo práctico de cómo se define dicha configuración, incluyendo diversos tipos de campos.

Resumiendo lo abordado en las secciones anteriores, la **Figura 3.8** ilustra un objeto JSON que contiene los atributos, tipos de datos y configuraciones requeridas para definir una *Entidad dinámica*. Este objeto puede ser enviado o consultado en la base de datos mediante una entidad estática encargada de gestionar las configuraciones de las entidades dinámicas.

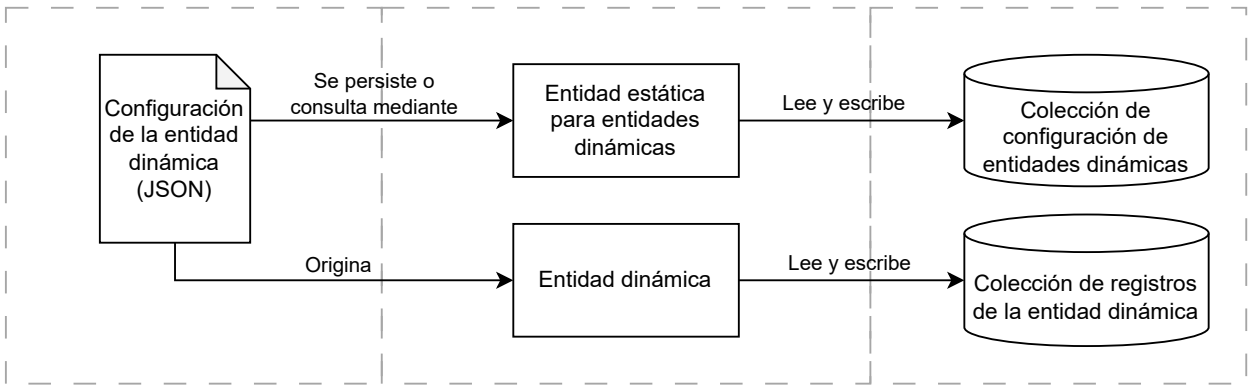


Figura 3.8: Persistencia de datos con una entidad dinámica

En relación con el proceso de persistencia de datos a través de la entidad dinámica, este inicia con la configuración almacenada en la *Colección de configuración de entidades dinámicas*. En un momento posterior, dicha configuración es consultada para generar la *Entidad dinámica* correspondiente y utilizarla para gestionar datos desde y hacia la *Colección de registros de la entidad dinámica*, la cual es nombrada con el valor configurado en el atributo *collection_ name*.

3.2.7. Gestión de usuarios

La gestión de usuarios en *CITDEV Express* permite administrar y controlar el acceso a las funcionalidades del sistema mediante la asignación de roles. Esta capacidad incluye la creación, edición, inactivación de usuarios y la definición de sus privilegios. A través de los roles, se delimita el alcance de las operaciones que cada usuario puede realizar, garantizando así un acceso seguro, controlado y alineado con las responsabilidades asignadas.

3.2.7.1. Roles

Los roles definen los permisos asociados a cada tipo de interacción dentro del sistema. Un mismo usuario puede contar con múltiples roles, lo cual le permite acceder a diferentes funcionalidades según el contexto y sus responsabilidades. Esta flexibilidad facilita una gestión precisa y adaptable de los privilegios, asegurando un modelo de acceso alineado con los principios de control basado en roles.

En *CITDEV Express* se han definido los siguientes roles de usuario:

- **ADMIN**: permite la gestión de usuarios y asignación de roles.
- **CITDEV**: permite la gestión de entidades dinámicas, incluyendo su configuración, atributos y exposición vía API.
- **CRUD**: permite la gestión de registros (crear, leer, actualizar, eliminar) en las entidades configuradas por el rol *CITDEV*, únicamente desde la interfaz gráfica.
- **API**: permite la gestión de registros mediante el consumo de la API REST configurada por el rol *CITDEV*.

Esta funcionalidad está respaldada por una colección estática denominada *core_ users*, la cual almacena la información de autenticación y los roles asignados a cada usuario. Los registros contenidos en esta colección son utilizados durante el proceso de inicio de sesión y en la generación de tokens JWT para el acceso autenticado a los servicios expuestos vía API REST. Adicionalmente, esta información es empleada en el módulo de auditoría para registrar de forma trazable las operaciones de escritura ejecutadas por cada usuario.

3.2.7.2. Atributos de una cuenta de usuario

Las cuentas de usuario en *CITDEV Express* se representan mediante documentos almacenados en la colección estática *core_ users*. En la **Tabla 3.4** se describen los atributos definidos para cada usuario.

Atributo	Tipo de dato	Descripción
`_id`	ObjectId	Identificador único generado automáticamente por la base de datos.
`name`	String	Nombre completo del usuario registrado.
`email`	String	Dirección de correo electrónico utilizada como identificador único para autenticación.
`password`	String (Hash)	Contraseña cifrada del usuario, almacenada mediante algoritmo de hashing seguro.
`status`	Boolean	Indica si la cuenta se encuentra activa (`true`) o inactiva (`false`). Solo los usuarios activos pueden iniciar sesión o generar tokens JWT.
`rols`	Array de String	Lista de roles asignados al usuario. Un mismo usuario puede tener múltiples roles de forma simultánea.
`updatedAt`	Date	Fecha y hora de la última actualización del documento. Este campo es útil para propósitos de auditoría y trazabilidad.

Tabla 3.4: Atributos definidos para una cuenta de usuario

3.2.7.3. Gestión de sesiones desde la GUI

CITDEV Express implementa un flujo de autenticación específico para usuarios que acceden a través de la interfaz gráfica (GUI). Este proceso permite la gestión de sesiones activas utilizando cookies de sesión seguras, sin necesidad de emitir tokens JWT.

Cuando un usuario accede a la plataforma a través de la GUI, se presenta un formulario de autenticación donde debe ingresar su correo electrónico y contraseña. Si las credenciales son válidas y el usuario se encuentra activo en la colección *core_ users*, el sistema crea una sesión de navegación asociada a su cuenta. Esta sesión mantiene al usuario autenticado mientras navega por la plataforma, y sus permisos se validan en cada vista con base en los roles asignados.

El cierre de sesión puede realizarse explícitamente mediante una opción visible en la interfaz. Esta funcionalidad revoca la sesión actual, invalidando la cookie correspondiente y redirigiendo al usuario a la pantalla de inicio de sesión.

Todas las acciones de escritura realizadas durante una sesión activa son registradas en el módulo de auditoría, incluyendo el inicio de sesión, lo que contribuye a mantener la trazabilidad de accesos y operaciones realizadas por cada usuario.

3.2.7.4. Registro de auditoría

CITDEV Express incorpora un mecanismo de auditoría que permite registrar las operaciones de escritura realizadas por los usuarios sobre las entidades del sistema. Este registro es gestionado por una colección estática denominada `core_auditlog`, la cual se actualiza automáticamente cada vez que se ejecuta una acción que modifica el estado de un recurso.

Cada evento de auditoría captura información relevante sobre la entidad afectada, el usuario que realizó la acción, el tipo de operación y el detalle de los cambios efectuados. Este mecanismo es esencial para garantizar la trazabilidad, el cumplimiento normativo y la supervisión operativa.

En la **Tabla 3.5** se describen los atributos definidos para el registro de auditoría.

Este mecanismo de auditoría se activa automáticamente para todas las operaciones de escritura, ya sea que se ejecuten desde la interfaz gráfica o desde la API, siempre que el usuario autenticado posea los privilegios requeridos para realizar la acción. Adicionalmente, el sistema registra eventos de inicio de sesión y la generación de tokens JWT, con el fin de mantener una trazabilidad completa sobre el acceso y uso del sistema por parte de los usuarios.

3.3. Implementación del marco de interoperabilidad

Esta sección presenta los lineamientos adoptados en *CITDEV Express* para garantizar la interoperabilidad entre sistemas, abordando desde su concepción teórica hasta su implementación técnica. Se describe el contexto actual en términos de integración de sistemas, el diseño de rutas dinámicas, el uso correcto de códigos de estado *HTTP*, los mecanismos de autenticación basados en *JSON Web Token (JWT)* y la generación automática de documentación mediante *OpenAPI*[2].

[2]OpenAPI es una especificación estándar, abierta e independiente del lenguaje, utilizada para describir de manera estructurada los recursos y operaciones de una API REST (Casas et al., 2021).

Atributo	Tipo de dato	Descripción
`_id`	ObjectId	Identificador único del registro de auditoría.
`collection_name`	String	Nombre de la colección sobre la cual se realizó la acción.
`user`	Object	Información del usuario que ejecutó la acción: identificador, correo, nombre y roles.
`documentId`	ObjectId	Identificador del documento específico que fue modificado.
`action`	String	Tipo de operación realizada sobre el recurso. Por ejemplo: `CREATE`, `UPDATE`, `DELETE`.
`originAction`	String	Medio a través del cual se ejecutó la operación. Por ejemplo: `GUI` o `API`.
`detail`	Array de objetos	Lista de campos modificados. Cada objeto incluye el nombre del campo, su valor original y el valor actualizado.
`createdAt`, `updatedAt`	Date	Fecha y hora en que se creó o actualizó el registro de auditoría.

Tabla 3.5: Atributos definidos para los registros de auditoría

Esta sección también explica cómo los *Citizen Developers* pueden controlar, a través de la interfaz gráfica, qué operaciones expone cada entidad dinámica a través de la *API REST*, asegurando una integración segura, estructurada y bajo control de acceso. Este marco de interoperabilidad permite que la plataforma se comunique con otros sistemas, cumpliendo los requisitos funcionales y no funcionales establecidos, y contribuyendo directamente al logro del objetivo específico OE3[3].

La implementación del marco de interoperabilidad descrito en esta sección da cumplimiento a los requisitos funcionales RF16 y RF23–RF31, al permitir exponer operaciones sobre entidades dinámicas a través de una API REST, generar documentación técnica automatizada y gestionar rutas de acceso mediante configuración. Asimismo, se encuentran involucrados los requisitos no funcionales RNF02 y RNF04–RNF06, al garantizar una experiencia de integración segura, documentada y con trazabilidad de acciones.

3.3.1. Interoperabilidad en el contexto actual

La interoperabilidad es un principio clave en el desarrollo de software moderno, ya que permite la comunicación efectiva entre distintos sistemas. La interoperabilidad se define como la capacidad de los sistemas para intercambiar información sin restricciones tecnológicas, garantizando así una integración fluida y eficiente (Tolk, 2013).

En el diseño de *Interfaces de Programación de Aplicaciones (APIs)*, la interoperabilidad se

[3]OE3: Diseñar e implementar un marco de interoperabilidad que garantice una integración efectiva y segura con otros sistemas de información, sin la intervención de personal experto en programación.

sustenta en el uso de estándares abiertos que facilitan la integración con sistemas heterogéneos. En este contexto, los estándares de interfaz, como REST, han sido ampliamente adoptados debido a su simplicidad y compatibilidad con protocolos como HTTP, lo que permite una comunicación eficiente y uniforme entre diferentes plataformas (Fielding and Taylor, 2000).

Con el auge de las arquitecturas basadas en microservicios[4] y soluciones en la nube, la interoperabilidad se ha convertido en un requisito para evitar dependencias rígidas entre sistemas (Taibi et al., 2017). Para abordar este desafío, el uso de APIs con rutas dinámicas ha demostrado ser una solución que permite la gestión flexible de recursos y facilita la escalabilidad y el mantenimiento del software.

El presente marco de interoperabilidad ha sido diseñado para asegurar una estructura adaptable y permitiendo la integración con otros sistemas. Además, la implementación de mecanismos estándar y el uso de buenas prácticas en el diseño de *APIs* garantizan un ecosistema de software interconectado y sostenible en el tiempo.

3.3.2. Rutas dinámicas

El diseño de la *API REST* que incluye *CITDEV Express* sigue principios *RESTful*, utilizando rutas que permiten gestionar distintos tipos de entidades dinámicas configuradas en el sistema. Estas rutas garantizan la extensibilidad y simplifican el mantenimiento, ya que permiten manejar múltiples recursos sin necesidad de definir nuevas rutas para cada tipo de entidad.

Para las rutas que requieren de parámetros se sigue la siguiente estructura:

- ***path_name*:** define el nombre de la ruta que identifica a una entidad sobre la cual se ejecutan las operaciones *CRUD*. Este valor se recibe dinámicamente en el sistema y permite la reutilización del mismo conjunto de rutas para distintas entidades.

- ***id*:** corresponde al identificador único de un registro.

En la Tabla 3.6 se describe cada una de las rutas que se han implementado y pueden ser habilitadas en la *API* de *CITDEV Express*.

Alias	Verbo HTTP	Ruta	Descripción
Index	GET	/api/crud/*{path_name}*	Obtiene todos los registros
Create	POST	/api/crud/*{path_name}*	Crea un nuevo registro
Read	GET	/api/crud/*{path_name}*/*{id}*	Recupera un registro
Update	PUT	/api/crud/*{path_name}*/*{id}*	Modifica un registro
Delete	DELETE	/api/crud/*{path_name}*/*{id}*	Elimina un registro

Tabla 3.6: Rutas dinámicas

[4]Los microservicios son aplicaciones desarrolladas como un conjunto de servicios relativamente pequeños, consistentes, aislados y autónomos que se implementan de forma independiente, con un propósito único y claramente definido (Taibi et al., 2017).

3.3.3. Códigos de estado HTTP

El uso adecuado de códigos de estado HTTP en la implementación de *API REST* es fundamental para garantizar una comunicación clara entre el cliente y el servidor. Estos códigos permiten que los consumidores de la API interpreten correctamente el resultado de una solicitud, facilitando la depuración, mejorando la experiencia de usuario y asegurando la interoperabilidad con otros sistemas. Un uso incorrecto de los códigos HTTP puede generar confusión en los clientes y dificultar la gestión de errores en las aplicaciones que dependen de la API. Para evitar estos problemas, las mejores prácticas recomiendan el uso de códigos estándar definidos por el *Internet Engineering Task Force (IETF)* y evitar respuestas ambiguas o genéricas (Fielding and Reschke, 2014).

El sistema implementado en este proyecto sigue estas prácticas y utiliza los siguientes códigos HTTP en sus respuestas:

- **200 OK**: indica que la solicitud se procesó correctamente y que el servidor devuelve la respuesta esperada.
- **201 Created**: se devuelve cuando un nuevo recurso ha sido creado correctamente. Este código se utiliza en solicitudes POST, indicando que la operación de inserción se completó sin errores.
- **400 Bad Request**: se usa cuando la solicitud enviada por el cliente es inválida o contiene datos incorrectos. Permite que el cliente corrija la solicitud antes de reenviarla.
- **401 Unauthorized**: indica que la solicitud requiere autenticación y que el cliente no ha proporcionado un token válido. Se emplea para rutas protegidas que requieren autenticación JWT.
- **404 Not Found**: se devuelve cuando el recurso solicitado no existe o no puede ser encontrado. Es útil para indicar que un ID de registro no es válido o que una URL es incorrecta.
- **500 Internal Server Error**: se usa cuando ocurre un error inesperado en el servidor. Este código no debe exponer detalles específicos sobre el fallo para evitar vulnerabilidades de seguridad.
- **503 Service Unavailable**: indica que el servicio no está disponible temporalmente, ya sea por mantenimiento o por sobrecarga en el servidor.

La **Tabla 3.7** expone los códigos de estado HTTP asignados a las rutas definidas en *CIT-DEV Express*, permitiendo una gestión estructurada de las respuestas del servidor ante diferentes solicitudes.

	200	201	400	401	404	500	503
Index	X			X		X	X
Create		X	X	X		X	X
Read	X			X	X	X	X
Update	X		X	X	X	X	X
Delete	X			X	X	X	X

Tabla 3.7: Códigos de estado HTTP por ruta considerados en CITDEV Express

3.3.4. Autenticación

Dado que la *API* maneja múltiples entidades dinámicas, requiere una estrategia de seguridad para proteger los recursos y evitar accesos no autorizados. Esta sección detalla la estrategia de autenticación basada en *JSON Web Tokens (JWT)*.

La *API* emplea un mecanismo de autenticación basado en *JWT* para gestionar el acceso a los recursos protegidos. *JWT* es una tecnología utilizada en la autenticación de APIs, ya que permite verificar la identidad del usuario y autorizar su acceso a distintos servicios sin la necesidad de almacenar sesiones en el servidor (Jones and Hardt, 2015).

El proceso de autenticación comienza cuando el usuario envía sus credenciales a través de la ruta ***/api/auth/login***. Si las credenciales son correctas, el sistema genera un *token JWT* que contiene información sobre el usuario y los permisos asociados a su cuenta. Este token debe ser enviado en cada solicitud protegida mediante el encabezado HTTP Authorization, utilizando el esquema Bearer[5].

Ejemplo de generación de un token JWT:

```
// generar un token JWT
curl --location 'http://localhost:3000/api/auth/login' \
    --header 'Content-Type: application/json' \
    --data-raw '{
        "email":"username@example.com",
        "password":"123*!Yed",
    }'
```

Cuando un usuario intenta acceder a un recurso protegido sin un token válido, la API devuelve un código de estado **401 Unauthorized**, indicando que la autenticación es requerida. Si el token ha expirado o es inválido, la API rechaza la solicitud y el cliente debe obtener un nuevo token autenticándose nuevamente.

Ejemplo de llamado a una ruta con token JWT:

```
// llamar una ruta con el token JWT
curl --location 'http://localhost:3000/api/crud/equipos-libertadores/67
   b4b1a562e9a14cd7dbb479' \
```

[5]El esquema Bearer es un mecanismo de autenticación definido en el estándar OAuth 2.0 que permite transmitir tokens de acceso en las solicitudes HTTP (Jones et al., 2012).

```
--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJ1c2VyIjp...'
```

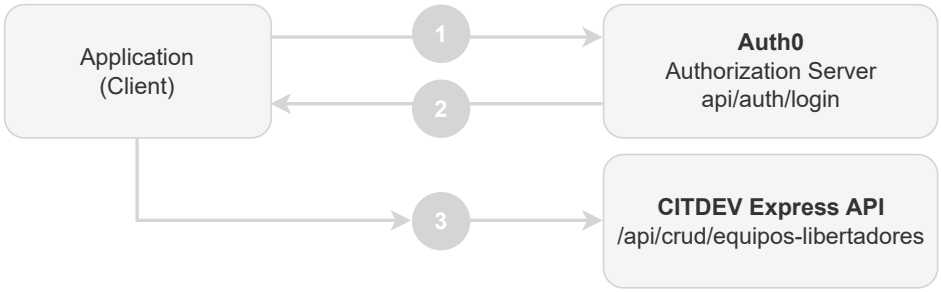


Figura 3.9: Obtener y usar un JWT en CITDEV Express. Adaptada de (jwt.io, sf)

El uso de *JWT* facilita una gestión segura del acceso a la *API*, eliminando la necesidad de mantener sesiones persistentes en el servidor y reduciendo la carga en los sistemas backend. La **Figura 3.9** ilustra este proceso, donde la aplicación cliente primero solicita un token al servidor de autorización (Auth0), posteriormente recibe el *JWT* válido, y finalmente utiliza dicho token para consumir los servicios de la *API* de *CITDEV Express*. Esta estrategia permite incluir información relevante sobre el usuario dentro del token, como roles y permisos, lo cual optimiza la validación de privilegios sin necesidad de realizar consultas adicionales a la base de datos en cada solicitud, mejorando así la eficiencia general del sistema.

3.3.5. Documentación

La documentación es una parte fundamental en el desarrollo de APIs, ya que permite a los desarrolladores comprender la estructura, los métodos disponibles y los parámetros que se pueden utilizar en cada endpoint. Sin una documentación adecuada, la integración con otros sistemas se vuelve más compleja y propensa a errores (Corradini et al., 2024).

OpenAPI es una especificación estándar para describir los componentes de una APIs REST y genera la documentación de forma estructurada y legible, facilitando la integración de servicios y el uso de herramientas automatizadas para pruebas y validación de APIs (Pergl and Jíša, 2024).

Swagger es un conjunto de herramientas que permite visualizar, generar y consumir documentación basada en *OpenAPI*. Incluye Swagger Editor, Swagger UI y Swagger Codegen, los cuales ayudan a los desarrolladores a escribir, visualizar y generar código cliente respectivamente (Yuncharoen and Suwannasart, 2024).

La adopción de *OpenAPI* y *Swagger* en el desarrollo de APIs REST ofrece múltiples beneficios para los desarrolladores y consumidores de servicios web. En primer lugar, la estandarización permite que diferentes APIs sigan un formato unificado, facilitando la integración con otros sistemas y asegurando una estructura homogénea en la definición de endpoints. Además, la automatización posibilita la generación de documentación interactiva y pruebas automatizadas, lo que reduce la carga manual en la validación y mantenimiento de las APIs. Otro beneficio clave es la facilidad de

uso, ya que *OpenAPI* y *Swagger* proporcionan interfaces visuales que permiten a los desarrolladores explorar y probar los endpoints sin necesidad de herramientas externas (Pergl and Jíša, 2024).

Verbo HTTP	Ruta	Descripción
GET	/admin/api-doc/*{path_name}*	Visualiza la definición de una API

Tabla 3.8: Detalle de la ruta de documentación

En *CITDEV Express* se ha integrado *Swagger UI* para visualizar la documentación que corresponde a una API REST desde la *GUI*. Se ha definido una ruta dinámica que se reusa para generar y visualizar la documentación de la API REST requerida. En la Tabla 3.8 se presentan los detalles de la ruta dinámica para documentación y en la Figura 3.10 se visualiza el resultado obtenido al consultar la documentación para la API *Equipos clasificados a libertadores*.

Figura 3.10: Visualizar documentación de una API en CITDEV Express

3.3.6. Habilitar la interoperabilidad

CITDEV Express ofrece a los *Citizen Developers* la capacidad de habilitar o deshabilitar las operaciones *CRUD* disponibles en cada entidad dinámica. Esta configuración se realiza a través de una interfaz intuitiva que permite definir qué operaciones de la *API* estarán disponibles para la integración con sistemas externos. Antes de procesar cualquier solicitud, se verifica que la operación correspondiente haya sido habilitada, rechazando aquellas solicitudes que no cumplan con los criterios establecidos. Esto permite controlar las funcionalidades expuestas y minimiza riesgos de seguridad.

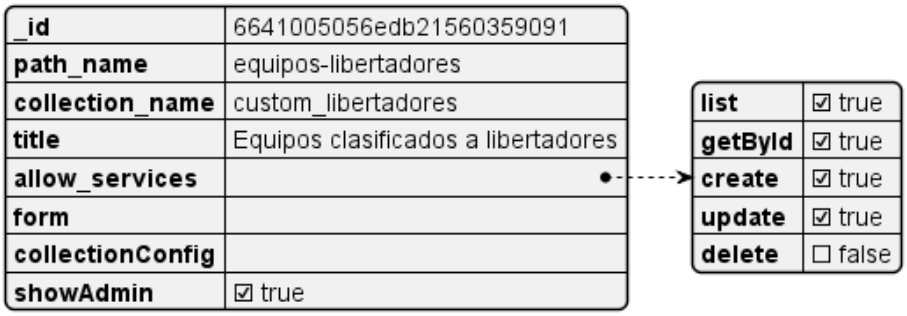


Figura 3.11: Servicios habilitados (allow_services)

Retomando lo anteriormente mencionado en la **Sección 3.2.4.4**, en el atributo **allow_services** de la configuración de la entidad dinámica se indica qué operaciones se exponen a través de la *API*. En la **Figura 3.11** se puede apreciar un ejemplo, en el cual se han habilitados todas las operaciones excepto la de eliminar registros (delete).

3.4. Conclusiones

Este capítulo expuso las decisiones técnicas y conceptuales que fundamentan el diseño de la solución *CITDEV Express*, orientada a habilitar la creación de funcionalidades *CRUD* sin intervención directa de desarrolladores expertos. A partir del análisis realizado, se destacan las siguientes conclusiones:

- La solución tecnológica propuesta permite abstraer la complejidad técnica del desarrollo de software tradicional, habilitando a usuarios sin formación en ingeniería a configurar funcionalidades operativas sin escribir código.
- La evidencia recopilada en el proceso de análisis confirmó que las principales barreras para adoptar enfoques tradicionales radican en la curva de aprendizaje prolongada y la limitada disponibilidad de personal técnico calificado.
- El levantamiento estructurado de requisitos funcionales y no funcionales brindó una base sólida para tomar decisiones de diseño orientadas a la seguridad e interoperabilidad del sistema.

- Las entidades dinámicas, como mecanismo de configuración de estructuras de datos, demostraron ser una solución para adaptar el sistema a múltiples contextos sin incurrir en redundancia ni necesidad de mantenimiento sobre código fuente.
- La flexibilidad del modelo se ve reforzada por el uso de una interfaz gráfica para la definición de entidades, lo cual permite modificar y extender las estructuras de datos en tiempo de ejecución, sin interrumpir el funcionamiento general del sistema.
- La estrategia de interoperabilidad implementada mediante rutas dinámicas, autenticación con *JWT* y documentación automática garantiza que la solución pueda integrarse fácilmente con plataformas externas.

Capítulo 4

Implementación de la solución

Este capítulo presenta la implementación técnica de la solución tecnológica *CITDEV Express*, la cual busca facilitar la creación y gestión de operaciones básicas para el manejo de datos sin requerir conocimientos avanzados en programación. Se describe el conjunto de tecnologías utilizadas en cada capa del sistema, conforme al modelo MVC y al modelo C4 adoptado para su arquitectura. Además, se explican los componentes desarrollados, su funcionalidad y las decisiones técnicas tomadas durante el proceso. Esta implementación responde a los requisitos funcionales y no funcionales definidos previamente, así como a los objetivos específicos del proyecto, asegurando la usabilidad e interoperabilidad de la solución. Finalmente, se incluye una demostración funcional del sistema, evidenciando su comportamiento frente a las tareas clave asociadas al rol de los *Citizen Developers*.

4.1. Stack tecnológico

En la Figura 4.1 se proporciona una visión integral de las tecnologías utilizadas en el desarrollo de *CITDEV Express*. Esta tabla describe las herramientas, lenguajes de programación, frameworks y servicios que se integraron en las diferentes capas de la aplicación. Al detallar el stack tecnológico, se busca ofrecer una comprensión clara y estructurada de los componentes que sustentan el sistema, así como de sus interrelaciones y funcionalidades específicas.

Figura 4.1: Stack tecnológico definido para *CITDEV Express*

Cuando se inició la concepción de *CITDEV Express*, la elección del stack tecnológico fue una decisión crucial puesto que se quería construir una solución sólida, escalable y eficiente, por lo que se seleccionó cuidadosamente cada componente. En las siguientes subsecciones se describe cada una de las tecnologías.

4.1.1. Presentación

La capa de presentación es responsable de gestionar la interacción entre el usuario y el sistema, ofreciendo una interfaz gráfica que permite visualizar los datos y ejecutar acciones mediante formularios, botones o menús. Su función principal es recibir las entradas del usuario y mostrar los resultados provenientes de la lógica de negocio de manera clara y estructurada. Esta capa se fundamenta en el patrón de diseño *Modelo-Vista-Controlador (MVC)*[1], actuando como la vista que genera la salida a partir de los datos entregados por el controlador, sin involucrarse en la manipulación directa de datos ni en la aplicación de reglas de negocio (Khaliluzzaman and Chowdhury, 2016).

- *HTML, CSS y Javascript* se usaron en el desarrollo del *front-end* de *CITDEV Express* debido a varias razones técnicas y de sostenibilidad. *HTML* se emplea para estructurar semánticamente cada vista, garantizando compatibilidad con navegadores y contribuyendo al posicionamiento web mediante marcado accesible y ordenado (Vaughan-Nichols, 2010). *CSS* es un lenguaje ampliamente utilizado para definir la presentación de documentos estructurados e interfaces de usuario (Mazinanian and Tsantalis, 2017). *Javascript* resulta indispensable para dotar de interactividad a la aplicación web, facilitando validaciones en tiempo real, actualizaciones parciales sin recarga y comunicación asincrónica con el backend mediante llamadas *AJAX* (Wang, 2011). Este conjunto de tecnologías, estándar en la industria, asegura compatibilidad con navegadores actuales, promueve la mantenibilidad del código y evita la dependencia de frameworks adicionales en tiempo de ejecución.

- La incorporación de *jQuery* permitió implementar interacciones dinámicas con el usuario sin necesidad de recargar la vista, incluyendo funcionalidades como el ordenamiento de campos en la *Gestión de campos* de las entidades dinámicas, la gestión de eventos sobre formularios y la ejecución de llamados asíncronos hacia servicios del backend o funcionalidades internas del sistema (OpenJS Foundation, 2025a,b).

- *Bootstrap* se eligió por su framework sólido y pre-construido, que proporcionó componentes UI listos para usar y una base consistente para la interfaz de usuario. También es importante mencionar su gran comunidad.

4.1.2. Aplicación

La capa de aplicación constituye el núcleo funcional del sistema, donde se implementan las reglas de negocio y los casos de uso específicos del dominio. Esta capa orquesta la interacción entre la capa de presentación y la capa de datos, procesando las solicitudes recibidas, aplicando la lógica necesaria y preparando los datos para ser visualizados o almacenados. Corresponde al controlador dentro del enfoque *MVC*, encargado de contener la lógica de negocio, realizar operaciones computacionales y coordinar el acceso a las fuentes de datos (Khaliluzzaman and Chowdhury, 2016).

[1] El patrón arquitectónico MVC organiza el software en Modelo, Vista y Controlador, separando datos, presentación y lógica de control para facilitar el mantenimiento y la escalabilidad del sistema (Yingda et al., 2018).

- Se decidió utilizar *Node.js* por su arquitectura de eventos y su capacidad para manejar muchas conexiones simultáneas. Esto era crucial para el alto volumen de tráfico que se puede presentar al implementar *CITDEV Express* dentro de un entorno laboral (Tilkov and Vinoski, 2010).

- *Express.js* fue el framework elegido por su sencillez y flexibilidad para crear rutas, controladores y middleware. Express.js es uno de los frameworks más maduros, ampliamente aceptados y mejor documentados para diversos escenarios de aplicación (Wu et al., 2024).

4.1.3. Datos

La capa de datos se encarga de gestionar el almacenamiento, consulta y actualización de la información utilizada por el sistema. Proporciona mecanismos para acceder de manera estructurada a las bases de datos, permitiendo recuperar y modificar los registros requeridos por la capa de aplicación. Esta capa se asocia al modelo en el patrón *MVC*, asumiendo la responsabilidad de interactuar con los repositorios de datos y garantizar la integridad y consistencia de la información manejada por el sistema (Khaliluzzaman and Chowdhury, 2016).

MongoDB fue seleccionado como sistema de gestión de base de datos para *CITDEV Express* debido a su naturaleza *NoSQL* orientada a documentos, que permite almacenar datos estructurados y no estructurados de forma flexible. Esta característica resulta especialmente adecuada para gestionar las entidades dinámicas del sistema. A diferencia de los sistemas relacionales tradicionales, las bases de datos *NoSQL* ofrecen un enfoque más robusto para aplicaciones con requisitos de adaptabilidad y crecimiento continuo (Araujo et al., 2021). En particular, *MongoDB* proporciona un modelo de almacenamiento basado en documentos similares a *JSON*, lo que facilita la representación de esquemas dinámicos y la integración eficiente con aplicaciones web modernas (Oonhawat and Nupairoj, 2017).

4.2. Arquitectura

En el diseño de la arquitectura de software de *CITDEV Express*, se ha empleado el *Modelo C4*. Este modelo facilita la descripción y comprensión de los elementos que conforman el componente de software, desde una perspectiva general hasta una particular, en cuatro niveles: contexto, contenedores, componentes y, opcionalmente, un nivel de código (Brown, 2018). Esta estructura jerárquica permite representar progresivamente los distintos aspectos del sistema, brindando una vista coherente y detallada del mismo. Cada nivel proporciona información relevante para diferentes actores del equipo técnico, lo que favorece la comunicación, el análisis y la toma de decisiones sobre la arquitectura. La utilización del *Modelo C4* también responde a su creciente adopción en entornos reales de desarrollo de software, donde ha demostrado ser efectivo para estructurar y documentar sistemas de forma clara y adaptable a distintos dominios (Vázquez-Ingelmo et al., 2020).

4.2.1. Contexto

El diagrama de contexto presenta una visión a alto nivel del sistema *CITDEV Express* y sus interacciones principales con actores externos. En este modelo se distinguen dos tipos de actores: los **Usuarios** del sistema, que interactúan a través de una *Interfaz Gráfica de Usuario (GUI)*, y los **Sistemas externos**, que acceden mediante una *API REST*.

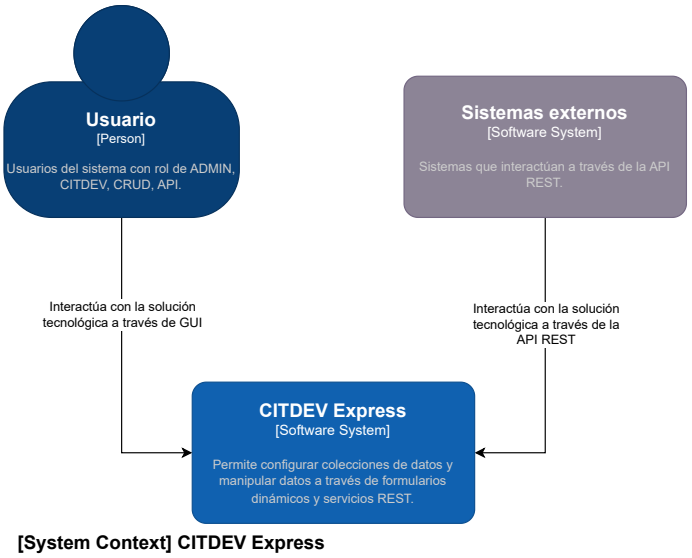


Figura 4.2: Diagrama de contexto

Los **Usuarios** interactúan con *CITDEV Express* mediante una *GUI*. De acuerdo con su rol (ADMIN, CITDEV, CRUD o API), tienen acceso a diferentes funcionalidades tales como la gestión de usuarios, la configuración de entidades dinámicas y la gestión de registros.

Paralelamente, los **Sistemas externos** interactúan con *CITDEV Express* a través de solicitudes HTTP dirigidas a la API REST. Cada petición se procesa de forma autenticada, utilizando tokens JWT, y su acceso es validado contra las configuraciones establecidas por el *Citizen Developer*.

4.2.2. Contenedores

El diagrama de contenedores permite representar los elementos principales que conforman *CITDEV Express* y cómo se comunican entre sí. Esta vista, corresponde al segundo nivel del *Modelo C4* y proporciona una descomposición lógica del sistema en contenedores de software, donde cada uno cuenta con una responsabilidad claramente definida. La arquitectura se compone de los siguientes contenedores:

- *Frontend*: es el contenedor responsable de ofrecer una *Interfaz Gráfica de Usuario (GUI)* basada en tecnologías web (HTML, CSS, Javascript, jQuery y Bootstrap). A través de esta

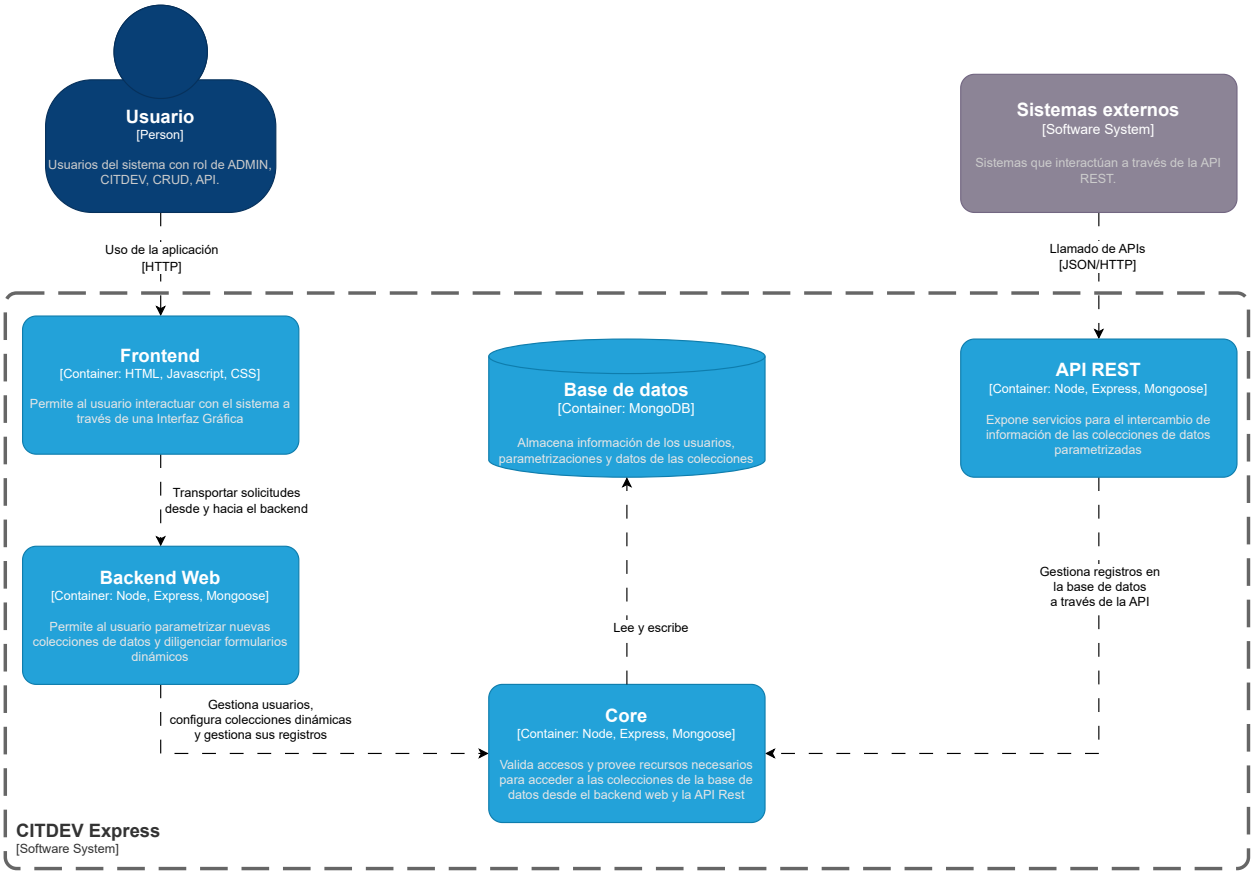


Figura 4.3: Diagrama de contenedores

interfaz, los usuarios del sistema pueden interactuar con las funcionalidades de configuración, gestión y uso de colecciones dinámicas.

- *Backend Web*: implementado con *Node.js* y *Express.js*, este contenedor actúa como intermediario entre la interfaz de usuario y la lógica de negocio del sistema. Gestiona sesiones, autenticación de usuarios, validación de permisos, y gestiona las operaciones sobre usuarios, entidades dinámicas y sus registros.

- *API REST*: contenedor encargado de exponer los servicios web que permiten realizar operaciones sobre las entidades dinámicas configuradas. A través de esta API, otros sistemas externos pueden consultar, crear, modificar o eliminar registros mediante rutas dinámicas validadas. Las solicitudes son procesadas utilizando el mismo núcleo de la aplicación para garantizar integridad y consistencia.

- *Core*: contenedor transversal que encapsula funcionalidades compartidas, como la generación de entidades dinámicas, control de autenticación y autorización, almacenamiento de logs de

auditoría, y validaciones sobre las estructuras de datos. Esta capa ofrece soporte tanto al Backend Web como a la API REST.

- *Base de datos*: sistema de almacenamiento basado en *MongoDB*, encargado de gestionar los datos de las entidades estáticas (usuarios, auditoría, configuración de colecciones) y las entidades dinámicas generadas por los usuarios. Su estructura flexible permite manejar esquemas heterogéneos adaptados a las necesidades del *Citizen Developer*.

Todas las interacciones entre contenedores se realizan mediante el protocolo HTTP, y los datos se intercambian en formato JSON. La comunicación entre los módulos es orquestada de forma síncrona desde el Frontend hacia el Backend Web y de forma asíncrona para las solicitudes a la API REST.

4.2.3. Componentes

El diagrama de componentes representa la descomposición interna de los contenedores definidos en la arquitectura de *CITDEV Express*. Este nivel describe los principales componentes software que conforman la solución y las responsabilidades que desempeñan dentro de su respectivo contenedor. Se hace énfasis en los contenedores *Backend Web* y *Core*, dado que encapsulan la lógica fundamental del sistema. El detalle de estos contenedores se puede visualizar en la **Figura 4.4** y la **Figura 4.5** respectivamente, y se explicará en las siguientes subsecciones.

4.2.3.1. Backend Web

Este contenedor se implementó bajo el patrón de diseño Modelo-Vista-Controlador (MVC) utilizando *Node.js* y *Express.js*. Está conformado por controladores que procesan las solicitudes de los usuarios, gestionan la lógica asociada a cada funcionalidad y determinan las vistas que deben renderizarse. Los componentes principales son:

- *UserController*: expone operaciones para gestionar usuarios del sistema, como creación, lectura, actualización y eliminación (CRUD), así como la asignación de roles.
- *CollectionsController*: administra las colecciones o entidades dinámicas configuradas por los usuarios, incluyendo su creación, edición y eliminación.
- *FieldsConfigController*: gestiona los atributos (campos) asociados a una colección dinámica, permitiendo su configuración personalizada.
- *AutoCrudController*: facilita la interacción con los registros de las colecciones dinámicas a través de formularios generados automáticamente teniendo en cuenta sus respectivos atributos configurados.
- *PanelAdminController*: presenta una vista consolidada de las colecciones habilitadas, actuando como punto de entrada al sistema administrativo.

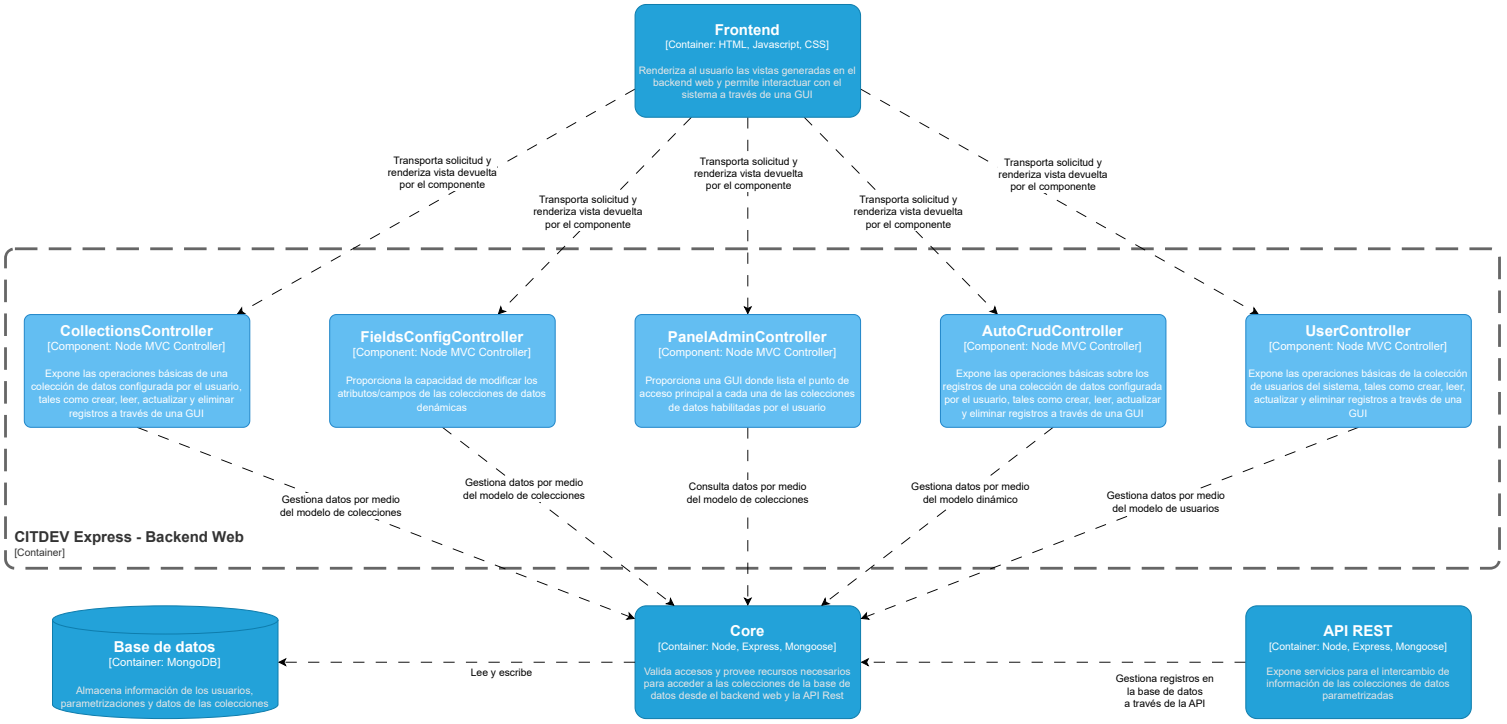


Figura 4.4: Diagrama de componentes - Backend Web

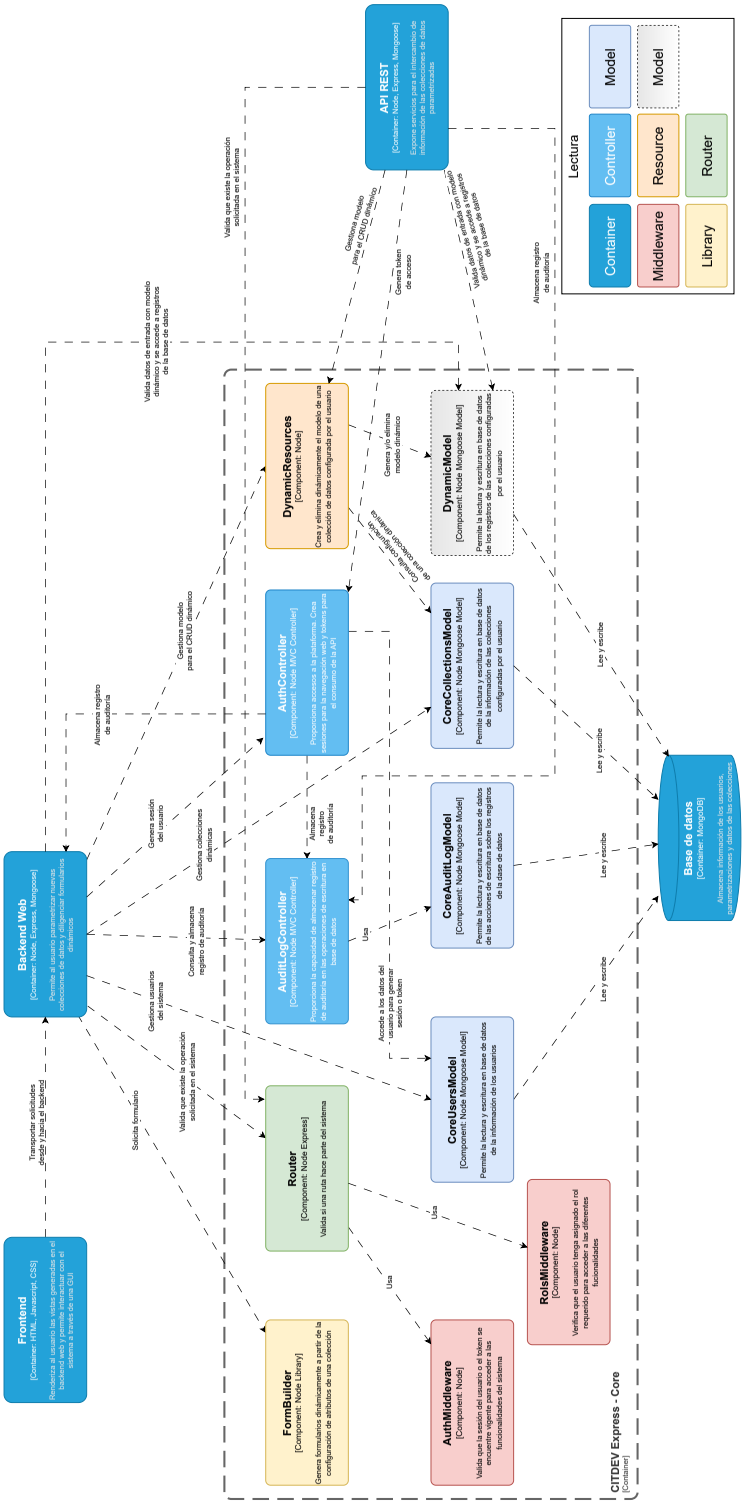


Figura 4.5: Diagrama de componentes - Core

4.2.3.2. Core

El contenedor *Core* ofrece funcionalidades compartidas para el *Backend Web* y la *API REST*. Está compuesto por componentes responsables de la autenticación, control de acceso, construcción de modelos dinámicos y registro de auditoría, entre otros. Sus componentes clave son:

- *AuthController:* controlador que se encarga de gestionar el proceso de autenticación de usuarios. Permite la creación de sesiones para la navegación en el entorno web y la generación de tokens *JWT* para el consumo de servicios vía *API*. Este componente es fundamental para garantizar el acceso seguro al sistema.
- *AuthMiddleware:* middleware que intercepta las solicitudes HTTP y verifica si existe una sesión activa o un token válido antes de permitir el acceso a los recursos. De esta forma, implementa un mecanismo de protección de rutas basado en roles.
- *Router:* componente encargado de validar si una ruta solicitada por el usuario o por un sistema externo forma parte del conjunto de rutas habilitadas por el sistema.
- *DynamicResources:* contiene la lógica para crear y eliminar modelos dinámicos en memoria. A partir de la configuración almacenada en la colección de colecciones, genera estructuras de datos adaptadas a cada entidad dinámica, sin necesidad de reiniciar el sistema.
- *DynamicModel:* modelo dinámico de base de datos generado en tiempo de ejecución utilizando *Mongoose*. Permite realizar operaciones *CRUD* sobre los datos de las colecciones dinámicas previamente configuradas.
- *CoreUsersModel:* modelo de base de datos encargado de gestionar la información de los usuarios del sistema, incluyendo credenciales y roles. Este modelo pertenece a las colecciones estáticas de la solución.
- *CoreCollectionsModel:* modelo utilizado para almacenar y consultar las configuraciones de las entidades dinámicas. Es a partir de este modelo que se construyen los esquemas dinámicos utilizados por otros componentes.
- *AuditLogController:* controlador responsable de registrar las acciones de escritura realizadas por los usuarios. Permite almacenar datos como el tipo de operación, el usuario responsable, el timestamp y el estado de la operación.
- *CoreAuditLogModel:* modelo de base de datos encargado de almacenar los registros generados por el *AuditLogController*. Permite la trazabilidad de las acciones sobre los datos y contribuye a cumplir con requisitos de seguridad y auditoría.
- *FormBuilder:* librería encargada de generar formularios dinámicos a partir de las configuraciones definidas en cada colección. Utiliza los atributos definidos por el Citizen Developer para construir formularios consistentes en la interfaz gráfica del sistema.

4.3. Demostración funcional

La presente sección tiene como objetivo evidenciar el cumplimiento de los requisitos funcionales definidos mediante una serie de imágenes que ilustran el comportamiento del sistema *CITDEV Express*. Estas imágenes no solo validan la implementación técnica de la solución, sino que también demuestran la aplicación de los principios de usabilidad definidos en la **Sección 3.1.5**. Cada interfaz fue diseñada teniendo en cuenta la visibilidad del estado del sistema, la consistencia en la presentación de la información, la prevención de errores mediante restricciones en los formularios, y el reconocimiento en lugar del recuerdo, reduciendo la carga cognitiva del usuario. Asimismo, se priorizó un diseño minimalista y estético que facilite la interacción con la herramienta, complementado con mensajes de ayuda y retroalimentación después de cada acción realizada por el usuario. De este modo, se garantiza una experiencia accesible e intuitiva, especialmente para los *Citizen Developers*, quienes son el público objetivo.

4.3.1. Aplicación de principios de usabilidad

En la **Figura 4.9** se muestra una vista correspondiente al módulo de gestión de entidades dinámicas. Esta interfaz aplica los principios de usabilidad descritos en la **Sección 3.1.5**, los cuales se describen a continuación:

- **Visibilidad y retroalimentación:** el sistema proporciona mensajes emergentes tipo *toast*[2] tras acciones como crear o eliminar registros, indicando claramente el resultado. Por ejemplo, ante una creación exitosa, se muestra una alerta en color verde con el texto *"El registro ha sido creado"*, mientras que, si ocurre un error como la duplicidad del nombre de recurso, se presenta una alerta en color rojo con un mensaje explicativo. Además, las acciones destructivas despliegan una ventana modal de confirmación antes de eliminar un registro. Finalmente, el uso de *breadcrumbs*[3] permite al usuario conocer su posición dentro del sistema en todo momento (ver la **Figura 4.7**).

- **Consistencia, descubribilidad y reconocimiento:** los botones para las operaciones *CRUD* emplean iconografía universal (por ejemplo, lápiz para modificar, papel para ver detalles y basurero para eliminar), acompañados de colores estandarizados (azul para acciones seguras y rojo para acciones destructivas). Además, la distribución horizontal de los botones se mantiene en todas las vistas del sistema, facilitando la familiarización y reduciendo la necesidad de aprendizaje adicional.

- **Prevención de errores y mejora de la eficiencia mediante restricciones y señales visuales:** se aplican validaciones en tiempo real para prevenir errores en los formularios. Por ejemplo, se muestran bordes rojos y mensajes como *"Este campo es obligatorio"* cuando un campo no es diligenciado. Además, si se intenta registrar una ruta ya existente, el sistema

[2] *Toast* es una notificación que proporciona retroalimentación a los usuarios (Wang et al., 2022).
[3] *Breadcrumbs* indica la ubicación de la página actual dentro de una jerarquía de navegación (Bootstrap, sf).

muestra un mensaje de advertencia específico. Estas validaciones se complementan con íconos y colores que permiten al usuario interpretar fácilmente el estado del formulario. Estos mecanismos permiten al usuario corregir los errores antes de enviar la información.

- **Simplicidad y claridad en el diseño mediante minimalismo y estética:** la interfaz evita el exceso de elementos visuales, presentando únicamente la información necesaria, como la tabla de registros, campos de búsqueda y botones de acción. El uso de espacios en blanco, tipografía clara y colores suaves contribuye a una experiencia visual limpia y ordenada. Las **Figuras 4.6** a **4.20** que se explicarán más adelante, muestran la interfaz gráfica de la solución.
- **Apoyo adicional mediante ayuda y documentación accesibles:** la interfaz incorpora un ícono de ayuda representado por un signo de interrogación. Al interactuar con este ícono, se muestra un mensaje que orienta al usuario sobre la funcionalidad del sistema. También se utilizan guías visuales en forma de mensajes flotantes que explican el propósito de los botones. Para acceder a estas guías, en todas las pantallas de la solución se dispone de un botón ubicado en la parte superior derecha, identificado con el ícono de un signo de interrogación, que permite al usuario activar la ayuda contextual correspondiente.

La implementación de estos principios contribuye a que los *Citizen Developers* usan la plataforma *CITDEV Express* sin necesidad de conocimientos técnicos especializados.

4.3.2. Formulario de inicio de sesión

La **Figura 4.6** muestra la interfaz gráfica de inicio de sesión de *CITDEV Express*. A través de este formulario, los usuarios registrados pueden autenticarse proporcionando sus credenciales de acceso (correo electrónico y contraseña). Esta funcionalidad permite iniciar una sesión válida en el sistema y redirige al usuario al panel de administración una vez autenticado exitosamente. Contribuye al cumplimiento de los requisitos RF01–RF04, al validar las credenciales y establecer una sesión de ingreso al sistema. Asimismo, soporta los requisitos no funcionales RNF02 y RNF06, al manejar errores de autenticación y restringiendo el acceso a usuarios autenticados. Esta funcionalidad se relaciona con el objetivo específico OE2.

4.3.3. Panel de administración

La **Figura 4.7** presenta el panel de administración de *CITDEV Express*, el cual se despliega una vez se valida correctamente la sesión del usuario. Desde este panel se habilitan accesos directos a las funcionalidades correspondientes a la gestión de usuarios, la gestión de entidades dinámicas y la gestión de registros asociados a dichas entidades (mediante la sección denominada *Entidades dinámicas del sistema*). Adicionalmente, se proporciona la opción para cerrar sesión del usuario. Este componente contribuye al cumplimiento de los requisitos RF05 y RF06, al garantizar que los usuarios autenticados accedan y visualicen las opciones disponibles acorde a sus privilegios. Además, se vincula con los requisitos no funcionales RNF07 y RNF06, al ofrecer una navegación clara e intuitiva y respetar los niveles de acceso definidos por roles. Esta funcionalidad se alinea con el objetivo específico OE2.

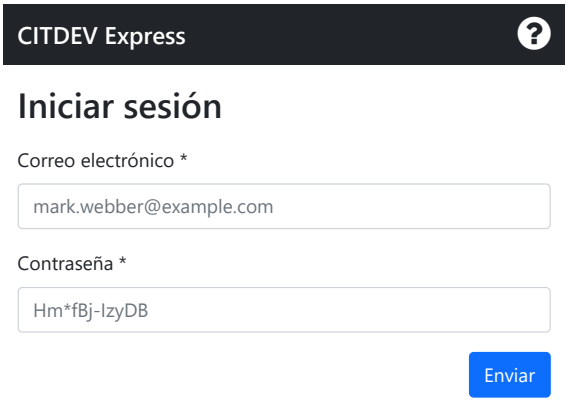


Figura 4.6: Formulario de inicio de sesión

4.3.4. Gestión de entidades dinámicas

Esta sección presenta las funcionalidades orientadas a la creación, modificación y eliminación de las entidades dinámicas configuradas por el rol *CITDEV*. Estas entidades representan estructuras de datos personalizadas que se almacenan en la base de datos y se exponen mediante rutas generadas automáticamente en la API. A través de una interfaz gráfica, el usuario puede definir los atributos de cada entidad, configurar su comportamiento y controlar su visibilidad. Las funcionalidades descritas en esta sección permiten cumplir con diversos requisitos funcionales y no funcionales, y están alineadas con el objetivo específico OE2, al facilitar la administración de modelos de datos sin necesidad de escribir código.

4.3.4.1. Lista de entidades dinámicas

La **Figura 4.8** muestra la vista general de las entidades dinámicas configuradas por el rol *CITDEV*, accesible desde el panel de administración. Esta funcionalidad permite visualizar todas las entidades disponibles en el sistema, cada una representando una estructura de datos personalizada para la gestión de información. Desde esta vista, los usuarios con los permisos adecuados pueden acceder a opciones para modificar o eliminar la configuración de la entidad, gestionar sus campos y ver la documentación técnica de la API que se expone. Esta funcionalidad da cumplimiento a los requisitos RF06 y RF12, al presentar las entidades disponibles y facilitar su gestión. Asimismo, se relaciona con el requisito no funcional RNF07, al permitir un acceso ágil a las operaciones más frecuentes.

4.3.4.2. Crear entidad dinámica

La **Figura 4.9** muestra la interfaz utilizada por el rol *CITDEV* para registrar una nueva entidad dinámica en el sistema. En este formulario se definen tres aspectos fundamentales: el título visible de la entidad, el nombre técnico del recurso utilizado en las rutas de la API, y el nombre de la colección en la base de datos. Esta funcionalidad da cumplimiento al requisito RF13, al permitir

Figura 4.7: Panel de administración

Figura 4.8: Lista de entidades dinámicas

definir nuevas entidades dinámicas desde la interfaz gráfica. También se relaciona con los requisitos no funcionales RNF05–RNF07, al registrar las acciones realizadas, controlar el acceso según el rol autorizado y ofrecer una experiencia de uso clara.

4.3.4.3. Modificar entidad dinámica

La **Figura 4.10** muestra la vista mediante la cual un usuario con el rol *CITDEV* puede editar los atributos principales de una entidad dinámica. En esta interfaz es posible modificar atributos como el título de la entidad, el nombre del recurso usado en las rutas (*path_name*) y su visibilidad dentro del panel de administración. El nombre de la colección de la base de datos es de solo lectura ya que no se permite modificar. Esta funcionalidad permite ajustar la configuración de la entidad sin requerir intervención de un desarrollador. Da cumplimiento al requisito RF14, al permitir actualizar la configuración previamente definida para una entidad dinámica. Además, se relaciona con los requisitos no funcionales RNF05 y RNF06, al garantizar trazabilidad sobre los cambios realizados al registro modificado y controlar que solo los usuarios con privilegios adecuados puedan acceder a esta funcionalidad. También contribuye al cumplimiento del requisito RNF07, al integrar accesos directos hacia funcionalidades relacionadas como la gestión de campos y la visualización de la documentación

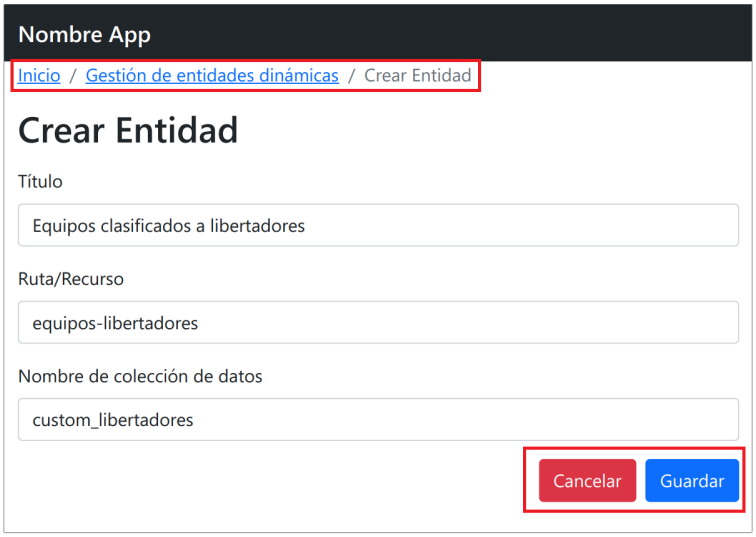


Figura 4.9: Crear entidad dinámica

técnica de la *API REST*, mejorando así la experiencia de uso.

4.3.4.4. Gestionar operaciones expuestas de la API

La **Figura 4.11** presenta la interfaz que permite al rol *CITDEV* habilitar o deshabilitar las operaciones disponibles para cada entidad a través de la API. Las operaciones configurables incluyen la creación, lectura, actualización y eliminación de registros. Esta funcionalidad da cumplimiento al requisito RF16, al permitir definir el comportamiento de exposición de cada entidad mediante la *API REST* sin necesidad de programación. Además, se relaciona con los requisitos no funcionales RNF05 y RNF06, al restringir el acceso a esta configuración exclusivamente al rol autorizado y registrar los cambios realizados. También contribuye al cumplimiento del requisito RNF07, al integrar accesos directos hacia funcionalidades relacionadas como la gestión de campos y la visualización de la documentación técnica de la *API REST*, mejorando así la experiencia de uso.

4.3.4.5. Ver documentación de la API

La **Figura 4.12** muestra la vista donde el rol *CITDEV* puede consultar la documentación técnica generada automáticamente para cada entidad dinámica. Esta documentación, presentada en formato *Swagger*, incluye las rutas del *API REST*, los métodos HTTP habilitados, los esquemas de los datos y los parámetros requeridos. Su propósito es facilitar la integración de otros sistemas con la API expuesta por la plataforma. Esta funcionalidad da cumplimiento al requisito RF20, al permitir consultar en tiempo real la documentación de las operaciones configuradas. Se relaciona con los requisitos no funcionales RNF06 y RNF07, al restringir el acceso exclusivamente al rol autorizado y proporcionar una interfaz estandarizada para facilitar la comprensión.

Figura 4.10: Modificar entidad dinámica

4.3.4.6. Gestión de campos

La **Figura 4.13** muestra la vista que permite al rol *CITDEV* administrar los campos que conforman una entidad dinámica. A través de esta interfaz es posible crear, actualizar, eliminar y reorganizar los atributos que describen la estructura de datos de una entidad. Cada campo puede configurarse con su tipo de dato, etiquetas, validaciones y otras propiedades asociadas. Esta funcionalidad da cumplimiento a los requisitos funcionales RF17 y RF18. Asimismo, se relaciona con los requisitos no funcionales RNF05–RNF07, al garantizar que solo los usuarios con los permisos adecuados puedan acceder a esta funcionalidad, registrar las modificaciones aplicadas y asegurar una experiencia de uso eficiente.

4.3.5. Gestionar registros de una entidad dinámica

Esta sección presenta las funcionalidades que permiten a los usuarios con rol *CRUD* interactuar con los datos almacenados en las entidades dinámicas configuradas por el rol *CITDEV*. A través de una interfaz generada automáticamente, el usuario puede listar, crear, modificar y eliminar registros asociados a una entidad, respetando la estructura previamente definida.

4.3.5.1. Listar registros de una entidad dinámica

La **Figura 4.14** muestra la interfaz que permite visualizar los registros almacenados de una entidad dinámica previamente configurada. Esta vista se genera automáticamente a partir de la estructura definida por el rol *CITDEV*, y presenta cada registro en forma de tabla con sus respectivos

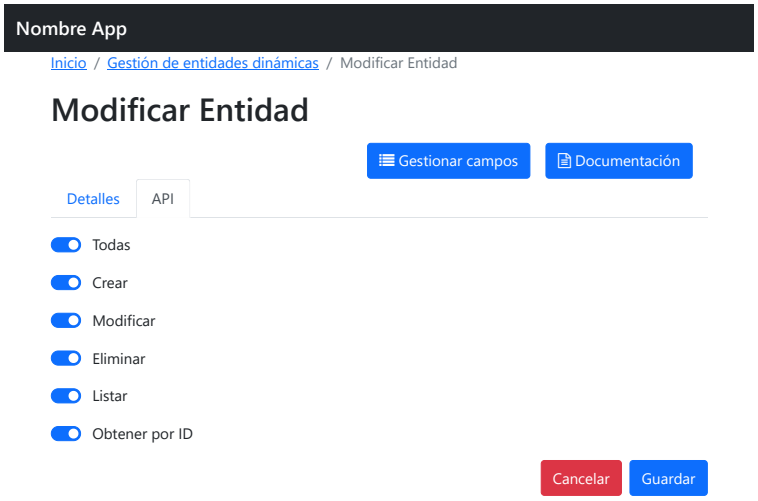


Figura 4.11: Gestionar operaciones expuestas de la API

campos. Desde esta pantalla, el rol *CRUD* puede consultar los datos existentes y acceder a opciones para modificarlos o eliminarlos. Esta funcionalidad da cumplimiento al requisito funcional RF19. Además, se relaciona con los requisitos no funcionales RNF01, RNF05–RNF07, al garantizar que solo usuarios con permisos accedan a esta funcionalidad, registrar las operaciones realizadas y ofrecer una experiencia de uso eficiente.

4.3.5.2. Crear registro de una entidad dinámica

La **Figura 4.15** muestra la interfaz generada automáticamente para registrar un nuevo dato de una entidad dinámica. El formulario se construye a partir de la estructura definida previamente por el rol *CITDEV*, y contiene los campos configurados con sus respectivas restricciones. El rol *CRUD* es el encargado de diligenciar esta información. Esta funcionalidad da cumplimiento al requisito funcional RF20, al permitir insertar nuevos registros en la colección asociada a la entidad dinámica. También se relaciona con los requisitos no funcionales RNF01, RNF05–RNF07, al controlar el acceso por rol, registrar las acciones realizadas y ofrecer una experiencia de uso eficiente.

4.3.5.3. Modificar registro de una entidad dinámica

La **Figura 4.16** muestra la interfaz utilizada para actualizar un registro previamente creado en una entidad dinámica. Esta vista es generada automáticamente con base en la estructura definida por el rol *CITDEV*, y permite al rol *CRUD* modificar los valores de los campos. Esta funcionalidad da cumplimiento al requisito funcional RF21, al permitir la actualización de registros ya existentes para una entidad dinámica. Además, se relaciona con los requisitos no funcionales RNF01, RNF05–RNF07, al controlar el acceso por rol, registrar las acciones realizadas y ofrecer una experiencia de uso eficiente.

Figura 4.12: Ver documentación de la API

4.3.5.4. Eliminar registro de una entidad dinámica

La funcionalidad para eliminar un registro de una entidad dinámica se encuentra accesible desde el botón *Eliminar* ubicado en la columna de acciones de la interfaz presentada en la **Figura 4.14**, correspondiente a la lista de registros en cumplimiento al requisito RF19. Al hacer clic en dicho botón, se presenta una ventana de confirmación como la que se puede visualizar en la **Figura 4.17**. Si el usuario confirma la acción, el registro es eliminado de forma definitiva de la base de datos. Esta operación da cumplimiento al requisito funcional RF22, al permitir eliminar un registro individual desde la interfaz gráfica. Asimismo, se relaciona con los requisitos no funcionales RNF05–RNF07, al controlar el acceso por rol, registrar las acciones realizadas y ofrecer una experiencia de uso eficiente.

4.3.6. Gestionar usuarios y roles

Esta sección presenta las funcionalidades relacionadas con la administración de los usuarios y los roles que determinan los niveles de acceso al interior de *CITDEV Express*. A través de una interfaz gráfica, los usuarios con el rol *ADMIN* pueden registrar nuevos usuarios, modificar la información de usuarios existentes, eliminar cuentas y asignar roles que habilitan el uso de funcionalidades específicas. Esta operación está alineada con los objetivos específicos OE2 y OE3, y se relaciona con los requisitos no funcionales RNF05, RNF06 y RNF07, al garantizar el control de acceso según el rol autorizado, registrar las acciones realizadas y proporcionar una experiencia de usuario eficiente y clara.

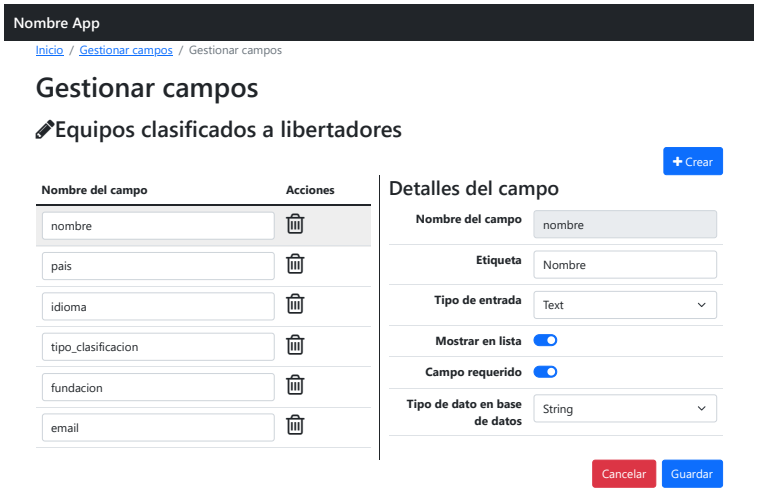


Figura 4.13: Gestionar campos

Figura 4.14: Lista de registros de una entidad dinámica

4.3.6.1. Listar usuarios

La **Figura 4.18** muestra la vista que permite consultar los usuarios registrados en el sistema. Esta interfaz presenta los datos relevantes de cada usuario en formato de tabla, como su nombre, correo electrónico, estado y roles asignados. Desde esta pantalla, los usuarios que poseen el rol *ADMIN* pueden acceder a opciones para modificar la información de los usuarios o eliminarlos si es necesario. Esta funcionalidad da cumplimiento al requisito funcional RF07, al permitir la gestión centralizada de usuarios desde la interfaz gráfica.

4.3.6.2. Crear usuario

La **Figura 4.19** muestra la interfaz que permite registrar un nuevo usuario en el sistema. A través de este formulario, los usuarios con el rol *ADMIN* pueden diligenciar los datos requeridos como nombre completo, correo electrónico, contraseña y roles asignados. El acceso a esta funcionalidad está

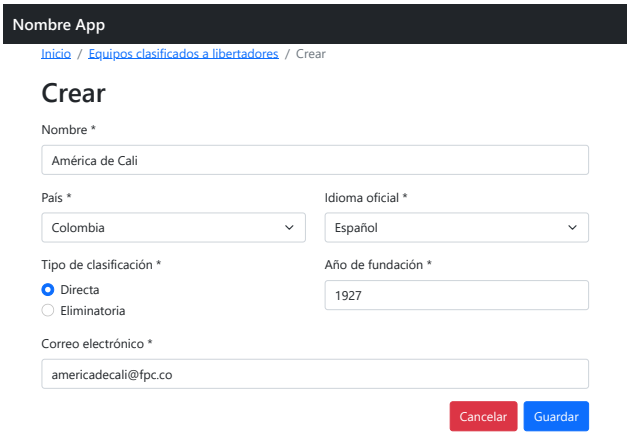


Figura 4.15: Crear registro de una entidad dinámica

restringido exclusivamente a los usuarios que poseen el rol *ADMIN*, y permite mantener actualizado el control de acceso al sistema. Da cumplimiento a los requisitos funcionales RF08 y RF10, al permitir el registro de nuevos usuarios desde la interfaz gráfica.

4.3.6.3. Modificar usuario

La **Figura 4.20** presenta la interfaz utilizada para actualizar la información de un usuario previamente registrado en el sistema. Esta vista permite modificar atributos como el nombre, el correo electrónico y los roles asignados. El acceso a esta funcionalidad está restringido exclusivamente a los usuarios que poseen el rol *ADMIN*. Esta operación da cumplimiento al requisito funcional RF09, al permitir la edición de la información de los usuarios existentes.

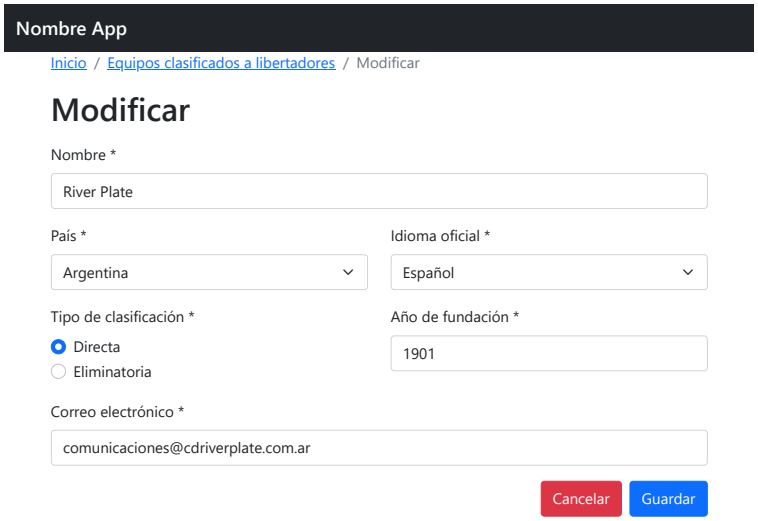


Figura 4.16: Modificar registro de una entidad dinámica

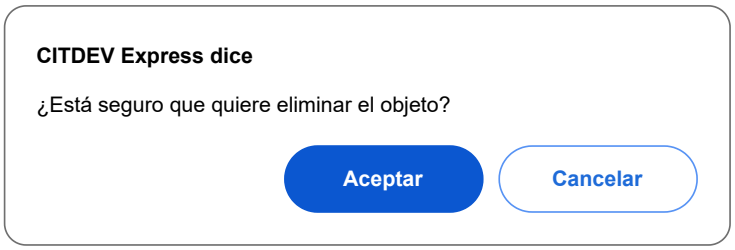


Figura 4.17: Listar usuarios

4.4. Conclusiones

La implementación de *CITDEV Express* ha sido orientada a cumplir con los requisitos funcionales y no funcionales previamente definidos, así como con los objetivos específicos del proyecto. En este capítulo se abordaron las tecnologías utilizadas, la arquitectura adoptada y las funcionalidades desarrolladas. A partir del análisis del proceso de implementación, se concluye lo siguiente:

- La solución tecnológica permite abstraer la complejidad técnica del desarrollo de software tradicional, habilitando a usuarios sin formación en ingeniería para configurar funcionalidades operativas sin escribir código.
- Las principales barreras para adoptar enfoques tradicionales fueron mitigadas mediante una interfaz gráfica que facilita la interacción con el sistema, reduciendo la curva de aprendizaje y promoviendo la autonomía de los *Citizen Developers*.
- La definición estructurada de requisitos funcionales y no funcionales permitió tomar decisiones

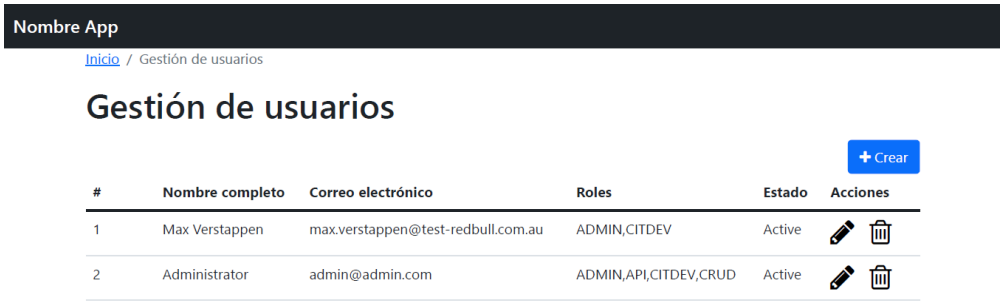


Figura 4.18: Listar usuarios

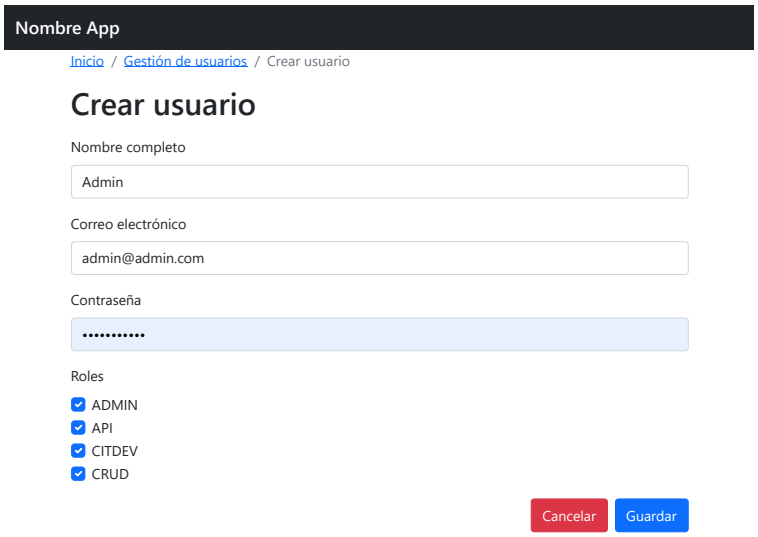


Figura 4.19: Crear usuario

de diseño orientadas a garantizar la seguridad, la interoperabilidad y la experiencia de usuario.

- El uso de entidades dinámicas como mecanismo de configuración flexible permitió adaptar el sistema a diferentes contextos de uso sin necesidad de modificar el código fuente.
- La interfaz gráfica implementada para la definición y gestión de entidades y registros permitió modificar y extender estructuras de datos en tiempo de ejecución, lo cual aporta a la mantenibilidad y evolución del sistema.
- La estrategia de interoperabilidad basada en rutas dinámicas, autenticación mediante *JWT* y documentación automática conforme al estándar *OpenAPI*, garantiza la integración de la plataforma con otros sistemas externos de forma segura.

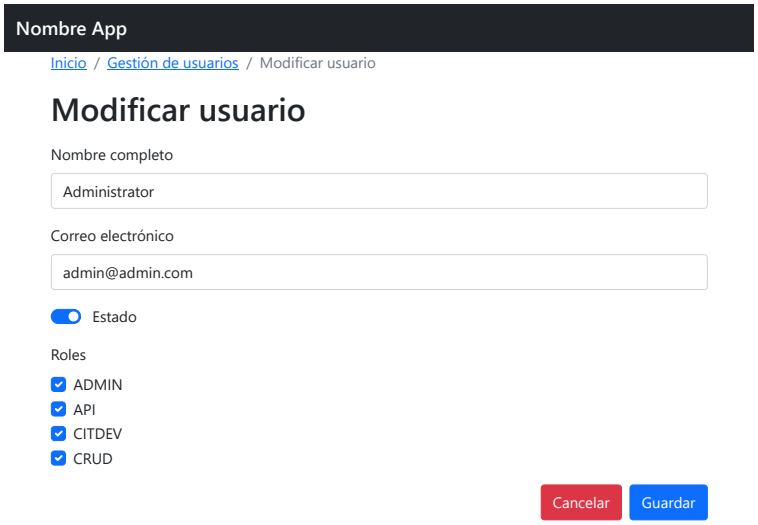


Figura 4.20: Modificar usuario

Capítulo 5

Evaluación

El desarrollo de soluciones tecnológicas dirigidas a Citizen Developers requiere una validación enfocada en evaluar tanto la usabilidad como la confiabilidad del software. Con este fin, el proyecto *CITDEV Express* fue evaluado siguiendo un proceso secuencial que abarcó desde pruebas funcionales hasta la medición de la experiencia del usuario. Los objetivos principales fueron analizar la usabilidad de la plataforma (en términos de efectividad, eficiencia y satisfacción) y evaluar su interoperabilidad.

Estos aspectos clave se establecieron para garantizar una experiencia de usuario satisfactoria y una integración efectiva con otros sistemas.

5.1. Metodología general

La evaluación de *CITDEV Express* se llevó a cabo mediante un enfoque multidimensional, que combinó técnicas cuantitativas y cualitativas para ofrecer un análisis global del sistema. Este proceso se estructuró en las siguientes fases:

- Pruebas funcionales: diseñadas para verificar que el software cumple con los requisitos especificados y comprobar que las funcionalidades implementadas responden a los objetivos.
- Pruebas no funcionales: enfocadas en evaluar atributos de calidad como el rendimiento, la interoperabilidad, la seguridad y la usabilidad. Estas pruebas permitieron determinar la robustez en condiciones de carga, evaluar la seguridad frente a controles de acceso y verificar la capacidad para interoperar con APIs mediante el formato estándar de intercambio de datos JSON, conforme a la norma ISO/IEC 25010.
- User Testing: basándose en los hallazgos de la revisión sistemática de Paz y Pow-Sang (Paz and Pow-Sang, 2015),que señalan que los métodos más recurrentes en la evaluación de software son los cuestionarios estandarizados y las pruebas con usuarios, se implementaron dos métodos:
  - *User Testing*: se realizaron pruebas con usuarios finales, quienes ejecutaron tareas específicas para identificar barreras en la interacción y proponer mejoras, de acuerdo con las recomendaciones de Rubin y Chisnell (Rubin and Chisnell, 2008).
  - Cuestionario de Satisfacción (*System Usability Scale - SUS*): diseñado por Brooke (Brooke, 1995), este cuestionario estandarizado permite medir la usabilidad percibida y el nivel de satisfacción del usuario.

Este enfoque metodológico permitió integrar perspectivas diversas, lo que permitió alcanzar un análisis equilibrado de los aspectos técnicos y de la experiencia del usuario.

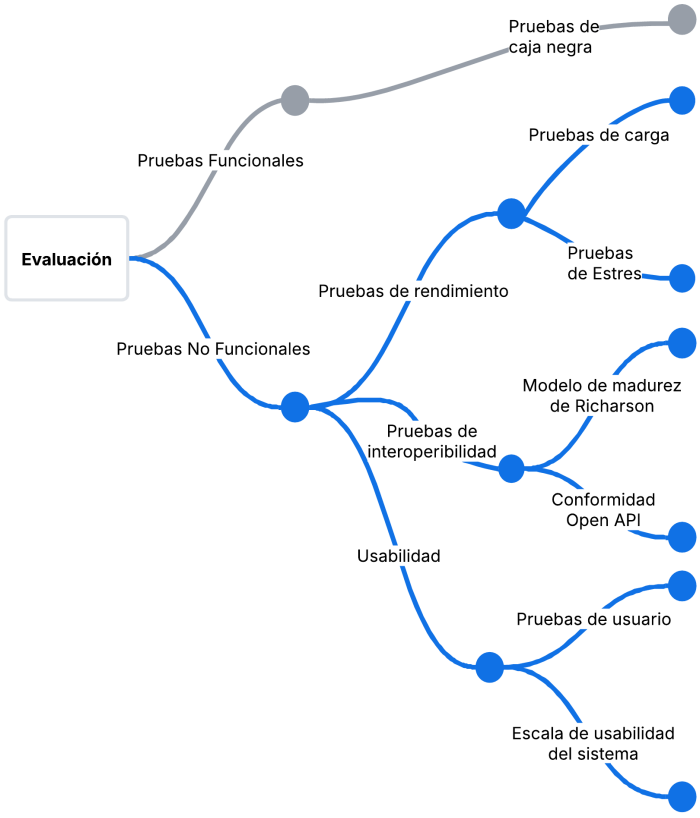


Figura 5.1: Metodología general de la evaluación

5.2. Pruebas funcionales

Las pruebas funcionales tuvieron como objetivo verificar que las funcionalidades definidas en los requerimientos fueran implementadas correctamente y que *CITDEV Express* operara de acuerdo con las expectativas del usuario. Para ello, se diseñaron casos de prueba basados en los requerimientos funcionales, lo que permitió la cobertura de los escenarios más críticos.

5.2.1. Metodología

La metodología de pruebas aplicada corresponde a un enfoque de caja negra, en el cual se evalúa la funcionalidad del sistema con base en las entradas y salidas definidas, sin considerar la estructura interna del software ni su implementación (Ostrand, 2002).

5.2.2. Casos de prueba y cobertura

En términos generales, las pruebas se ejecutaron considerando los escenarios principales de uso y las posibles interacciones de los usuarios con el sistema.

5.2.3. Cobertura de pruebas

Para medir la efectividad de la evaluación, se utilizaron métricas de cobertura de pruebas:

- Cobertura de requerimientos: Se ejecutaron pruebas sobre el 100 % de los requerimientos funcionales, independientemente de su nivel de prioridad (ALTA, MEDIA o BAJA). Sin embargo, aquellos marcados como ALTA y MEDIA se les dio un énfasis especial para garantizar su correcto funcionamiento.
- Cobertura de caminos de usuario: Se revisaron los flujos principales de navegación para garantizar la continuidad de los procesos y la accesibilidad de las funcionalidades sin interrupciones:
  - Autenticación y control de acceso: validación del inicio de sesión y restricción de acceso a funcionalidades protegidas según el perfil del usuario.
  - Gestión de usuarios: listar, crear, editar y eliminar usuarios, con validaciones específicas sobre la unicidad del correo electrónico.
  - Gestión de colecciones dinámicas: listar, crear, editar y eliminar colecciones, permitiendo definir y configurar campos dinámicos.
  - Operaciones CRUD dinámicas: validación de las operaciones básicas sobre los registros dentro de cada colección, asegurando la correcta persistencia y recuperación de datos.

También algunos de los flujos alternos, por ejemplo: validaciones de campos obligatorios, inexistencia de registros y mensajes informativos acordes a las acciones ejecutadas.

5.2.4. Hallazgos

Las pruebas funcionales se ejecutaron con un enfoque de caja negra y cubrieron los requerimientos establecidos. Se verificaron todos los caminos de usuario descritos en 5.2.3 (autenticación, gestión de usuarios, gestión de colecciones y operaciones CRUD), además de los escenarios alternativos de validación de datos, ausencia de registros y mensajes informativos. Los requerimientos con prioridad ALTA y MEDIA recibieron atención prioritaria.

Las pruebas funcionales permitieron validar el comportamiento del sistema, optimizar la experiencia del usuario y mejorar la prevención de errores mediante las siguientes mejoras:

- Flujos de navegación: los flujos principales se completaron sin interrupciones. En los flujos secundarios, se agregó una redirección automática a «Gestionar campos» después de crear una entidad dinámica, lo que eliminó un clic o paso intermedio. Esto, a su vez, sirvió para verificar el requerimiento no funcional que busca reducir la cantidad de clics para acceder a las operaciones principales.

- Creación de entidades: para agilizar el registro y disminuir errores al registrar entidades, los campos «Ruta/Recurso» y «Nombre de colección» se autocompletan con base en el título que ingresa el usuario. Sin embargo, estos campos siguen siendo editables.
- Unicidad de identificadores: el backend agrega un identificador único (UID) al final de «Ruta/Recurso» y «Nombre de colección» en el momento de su creación. Se recomienda mejorar el manejo de este UID para que no sea visible al usuario final, manteniendo la unicidad sin afectar la experiencia de uso.
- Visibilidad por defecto al crear la entidad: la opción «Mostrar en el grupo de entidades dinámicas del sistema» permanece activada por defecto para evitar que nuevas entidades no sean localizadas en la interfaz.
- Definición de campos dinámicos: el valor «Nombre del campo» se genera automáticamente a partir de la etiqueta proporcionada; el sistema valida los caracteres permitidos. El usuario conserva la posibilidad de modificarlo.

Estas mejoras respaldan la adecuación funcional de la aplicación y orientan las tareas de mejora continua centradas en la experiencia del usuario, la minimización de los flujos, la prevención de errores y la estandarización o consistencia de vistas y datos.

5.3. Pruebas no funcionales

5.3.1. Atributos de calidad

El desarrollo de *CITDEV Express*, debido a su orientación a Citizen Developers, requirió un enfoque en la calidad del software para permitir su efectividad, fiabilidad y facilidad de uso.

En este contexto, la norma ISO/IEC 25010 fue clave para identificar los atributos de calidad antes mencionados, que sirvieron como base para evaluar aspectos críticos del sistema. Estos atributos surgieron de los requerimientos no funcionales y se vincularon con los objetivos del trabajo de grado, como la implementación de las operaciones CRUD y la evaluación de la usabilidad, la eficiencia y la interoperabilidad, de modo que la solución atendiera las necesidades técnicas y operativas. A continuación, se describen los atributos identificados:

- Compatibilidad: dado que los usuarios operan en diferentes dispositivos y navegadores, el sistema debe ofrecer una experiencia uniforme sin importar el entorno. Para lograrlo, se adoptó un diseño responsivo y consistente, asegurando que los elementos visuales y las interacciones mantuvieran coherencia en todas las vistas (Nielsen, 1994).
- Interoperabilidad: en cuanto a la interoperabilidad, *CITDEV Express* implementó una arquitectura basada en servicios RESTful, utilizando JSON como formato de intercambio de datos. Esta combinación facilita la comunicación segura y eficiente entre sistemas a través de Internet, gracias al uso de estándares abiertos como HTTP y formatos ligeros como JSON, que permiten una integración flexible y escalable entre aplicaciones (Amazon Web Services, sfb).

- Fiabilidad: para mantener la estabilidad del sistema y la integridad de los datos, se implementaron mecanismos de manejo adecuado de excepciones y registro de errores. Se aplicaron principios (Nielsen, 1994) como la prevención de errores, asegurando que las acciones críticas contaran con confirmaciones y advertencias claras. Además, la retroalimentación inmediata en caso de fallos permitió que los usuarios comprendieran el estado del sistema en todo momento, alineándose con la visibilidad y la transparencia en la interacción.

- Rendimiento: la eficiencia en el procesamiento de operaciones fue priorizada para favorecer tiempos de respuesta óptimos y una ejecución fluida. Se implementaron estrategias de optimización como la automatización de procesos y mapeo, enfocadas en reducir cuellos de botella y evitar sobrecargas innecesarias. Asimismo, se establecieron restricciones para mejorar la gestión de recursos, en línea con las recomendaciones de IBM sobre eficiencia operativa (IBM, 2024)

- Seguridad: la protección de la información y la gestión de accesos se basaron en principios de autenticación segura (JWT) (jwt.io, sf), almacenamiento de credenciales cifradas y auditoría de acciones. De esta manera, se garantiza la confidencialidad, la integridad y el no repudio. Además, se incluyeron restricciones en las interacciones para prevenir errores (Nielsen, 1994) y garantizar que solo los usuarios autorizados ejecutaran determinadas acciones.

- Usabilidad: la interfaz fue diseñada para reducir la carga cognitiva mediante el uso de patrones de diseño estandarizados y consistentes (Norman, 2013). Se priorizó la descubribilidad y la estandarización de ventanas, iconos y posiciones, asegurando que la disposición de los elementos fuera homogénea en todo el sistema (Nielsen, 1994). Esto permitió que los usuarios identificaran rápidamente las funciones sin necesidad de recordar comandos complejos .

5.3.2. Pruebas de rendimiento

Con el fin de evaluar el comportamiento de la solución tecnológica *CITDEV Express* bajo diferentes condiciones de carga, se diseñaron y ejecutaron pruebas de rendimiento enfocadas en dos escenarios: prueba de carga y prueba de estrés. Ambas pruebas se realizaron en la plataforma *Heroku*[1] sobre un *dyno (Basic)*[2], el cual presenta recursos compartidos y limitados respecto a procesamiento y memoria.

Las operaciones seleccionadas para la evaluación fueron dos solicitudes representativas del sistema: la operación de inicio de sesión `(POST /api/auth/login)`, necesaria para la obtención del token *JWT*, y una operación de consulta sobre una entidad dinámica que retorna 200 registros `(GET /api/crud/inventario-de-equipos-7140)`.

El requerimiento no funcional RNF03, asociado al rendimiento del sistema, establece que este debe soportar al menos cinco solicitudes concurrentes por segundo sin degradación significativa, y

[1] *Heroku* es una plataforma que permite desplegar aplicaciones de forma rápida (Heroku, sfb).

[2] Un *dyno* es un contenedor ligero e independiente basado en Linux donde *Heroku* ejecuta las aplicaciones y *Basic* es uno de los planes de precios de *Heroku* diseñado para ejecutar pequeños proyectos o pruebas de concepto. (Heroku, sfa).

que el tiempo promedio de respuesta no debe superar los 2000 ms en el 90 % de las transacciones.

5.3.2.1. Prueba de carga

Esta prueba consistió en incrementar progresivamente el número de usuarios concurrentes hasta alcanzar un máximo de 7 hilos, simulados durante un período de prueba total de 7 minutos. Durante los primeros dos minutos se aumentó la carga de forma escalonada, y los cinco minutos siguientes se mantuvo la concurrencia constante.

Los resultados mostraron un tiempo de respuesta promedio global de 840 ms y un rendimiento sostenido de 7.3 transacciones por segundo. El 90 % de las transacciones fueron respondidas en un tiempo menor a 1618 ms, y no se presentaron errores de ejecución, tal como se evidencia en la **Figura 5.2**.

Etiqueta ↑	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
Estrés01:Inventario-equipos	1537	644	583	1502	1519	1579	96	1748	0,00%	3,7/sec	110,90
Estrés01:Login	1533	1037	1042	1633	1684	1730	310	1816	0,00%	3,7/sec	4,66
Total	3070	840	798	1618	1635	1720	96	1816	0,00%	7,3/sec	115,55

Figura 5.2: Resumen de la prueba de carga

Estos resultados permiten concluir que el sistema cumple con el requerimiento RNF03 en condiciones de carga media, alcanzando más de cinco solicitudes por segundo y manteniendo los tiempos de respuesta dentro de los límites establecidos en al menos el 90 % de los casos.

5.3.2.2. Prueba de estrés

En la prueba de estrés se buscó determinar el punto de fallo del sistema. Para ello, se programó una ejecución con incremento gradual de carga hasta alcanzar 30 hilos durante cinco minutos, agregando un nuevo hilo cada 10 segundos.

El rendimiento máximo registrado fue de 7.8 transacciones por segundo. Sin embargo, se observó un aumento progresivo en los tiempos de respuesta, alcanzando un promedio de 1958 ms y un 90 % de las transacciones con tiempos menores a 4034 ms. El porcentaje de error fue del 1.29 % y los picos máximos de respuesta superaron los 10 s, como se resume en la **Figura 5.3**.

Etiqueta ↑	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec
Estrés01:Inventario-equipos	1174	1630	870	3880	4400	5586	4	8868	2,47%	4,0/sec	117,48
Estrés01:Login	1145	2294	2187	4039	4703	5872	313	10401	0,09%	3,9/sec	4,94
Total	2319	1958	1687	4034	4557	5872	4	10401	1,29%	7,8/sec	122,41

Figura 5.3: Resumen de la prueba de estrés

Aunque el sistema logra mantener una tasa de procesamiento aceptable hasta cierto umbral, los tiempos de respuesta comienzan a degradarse a partir de los 20 hilos aproximadamente, indicando que esta es la capacidad límite para el entorno en que se realizaron las pruebas. Se evidencia así la necesidad de escalar la infraestructura en caso de proyectarse un uso mayor por parte de los usuarios.

5.3.2.3. Hallazgos

Ambas pruebas permiten establecer una línea base para el comportamiento del sistema en el entorno bajo el cual se realizaron las pruebas. En particular, se confirma que:

- El sistema cumple el requerimiento RNF03 bajo condiciones de carga media.
- El rendimiento sostenido alcanza hasta 7.3 transacciones por segundo sin errores ni degradación significativa.
- La operación de autenticación y consulta se comportan de forma estable en escenarios controlados.
- Bajo carga extrema, se identifica el punto de fallo después de 25 usuarios concurrentes, útil para definir estrategias de escalabilidad horizontal o vertical de acuerdo a las necesidades del negocio.
- Estos resultados son relevantes para futuras decisiones de mejora continua y optimización de recursos.

5.3.3. Pruebas de interoperabilidad

La interoperabilidad se ha convertido en un requisito esencial para los sistemas de información, especialmente en entornos donde diferentes sistemas deben comunicarse de manera eficiente y confiable. En este contexto, *CITDEV Express* adoptó una arquitectura RESTful y el uso de JSON como formato de intercambio, lo que favorece la integración con servicios.

Para valorar la madurez y la calidad del servicio, se aplicaron dos referentes complementarios que permitieron obtener una visión integral.

- **Modelo de Madurez de Richardson (RMM):** permite determinar el grado de adopción de principios REST mediante cuatro niveles (0–3), desde la simple exposición de recursos hasta la incorporación de hipermedios.
- **Conformidad con OpenAPI 3.0:** garantiza que el contrato del servicio sea verificable por herramientas automáticas y generadores de clientes, aportando portabilidad a los procesos de desarrollo y pruebas.

La documentación generada por *CITDEV Express* se publica automáticamente mediante Swagger, lo que facilita la inspección visual de los recursos.

Resultados del RMM

La API alcanza el nivel 2, lo que demuestra un uso adecuado de los verbos HTTP y recursos bien definidos. Sin embargo, aún no incorpora hipermedia que habilite la navegación dinámica entre estados, un requisito necesario para alcanzar el nivel 3 según el modelo de madurez de Richardson.

Nivel	Criterio principal	Evidencia en la API	Cumple
0	Acceso remoto sin recursos diferenciados	No aplica: las rutas se basan en sustantivos	—
1	Identificación explícita de recursos mediante URI cohesivas	`/api/crud/marcas-de-automoviles-{id}`	✓
2	Uso semántico de métodos HTTP y códigos de estado	Implementa `GET`, `POST`, `PUT`, `DELETE` con códigos 200, 201, 4xx, 5xx	✓
3	Hipermedia como motor de estado (HATEOAS)	Las respuestas no incluyen enlaces navegables (`_links`) ni perfiles HAL/JSON API	×

Tabla 5.1: Resultados del Modelo de Madurez de Richardson (RMM)

Resultados de la validación OpenAPI

Se eligieron *Suppa.ai OpenAPI AI Tools* y *Redocly CLI* porque, de las cinco herramientas evaluadas, fueron las únicas que proporcionaron retroalimentación detallada.

- El análisis con *Suppa.ai OpenAPI Tools* evidenció inconsistencias estructurales y omisiones en la definición de parámetros y respuestas de error. Estas observaciones no impiden el consumo manual del servicio, pero invalidan la especificación ante validadores estrictos, dificultan la generación automática de pruebas y documentación, y reducen la interoperabilidad en los procesos de integración y entrega continua..

- El análisis con *Redocly CLI* reportó advertencias de severidad media relacionadas con prácticas de documentación (descripciones incompletas, ejemplos poco ilustrativos y metadatos inconsistentes). Estas observaciones no afectan la ejecución del servicio, pero sí disminuyen la claridad y la mantenibilidad del contrato.

Hallazgos principales

- **Conformidad estructural:** la especificación cumple globalmente con la sintaxis de *OpenAPI 3.0.0*; no obstante, se identificaron inconsistencias puntuales en parámetros y respuestas de error que deben corregirse para garantizar la interoperabilidad plena con los pipelines de documentación, pruebas e integración continua.

- **Cobertura documental:** las advertencias señaladas por *Redocly* evidencian la necesidad de fortalecer las descripciones de operaciones, parámetros y esquemas para mejorar la comprensión por parte de los consumidores externos.

- **Madurez REST:** alcanzar el nivel 3 del RMM permitiría incorporar hipermedia (links) y habilitar HATEOAS, aumentando el desacoplamiento cliente–servidor y la capacidad de evolución sin cambios disruptivos.
- **Estrategia de mejora continua:**
  - **Corto plazo:** solventar las advertencias de documentación y normalizar los ejemplos.
  - **Mediano plazo:** introducir enlaces navegables y perfiles HAL/JSON API para escalar al nivel 3.
  - **Largo plazo:** evaluar mecanismos de descubrimiento de servicios y versionado semántico para fortalecer la mantenibilidad.
- **Evaluación integrada:** la utilización conjunta de conformidad OpenAPI y RMM resultó efectiva para mejorar progresivamente la calidad técnica de los servicios web ofrecidos.

En síntesis, la API de *CITDEV Express* demuestra una base sólida de interoperabilidad; no obstante, avanzar hacia hipermedia y mejorar la consistencia documental potenciará su extensibilidad y la experiencia de los desarrolladores que la consumen.

5.4. Evaluación de usabilidad

Para evaluar la usabilidad de *CITDEV Express*, se implementó una combinación de métodos cualitativos y cuantitativos. Se aplicaron *User Testing* para analizar la eficacia y la eficiencia del sistema, mientras que la satisfacción fue medida mediante la aplicación del *SUS*, un cuestionario ampliamente validado en estudios de usabilidad.

5.4.1. Métricas de evaluación

Para la evaluación de la usabilidad se utilizaron las siguientes métricas:

- Eficacia: Se midió en términos de la tasa de éxito de cada tarea. Un usuario se consideraba exitoso si lograba completar la tarea sin errores críticos.
- Eficiencia: Se comparó el tiempo que cada usuario empleó en completar las tareas con un referente de 17,2 h, calculado como la media de 13 estimaciones de profesionales del sector tras descartar dos valores atípicos (2 h y 48 h). Las estimaciones abarcaron un rango de 8 h a 24 h, con una desviación estándar muestral[3] de 6,4 h. Los datos de la estimación se pueden encontrar en los anexos, tabla 3.3.

[3]La desviación estándar muestral se calcula como:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

donde x_i es el dato del participante i, $\bar{x}$ es la media muestral de los datos y N es el número total de datos considerados.

- Satisfacción: Se midió a través de *SUS*[4], con una escala de 0 a 100, donde puntajes superiores a 68 indican una buena usabilidad.

5.4.2. Protocolo de diseño de la evaluación

Se entrevistaron cinco usuarios, siguiendo la recomendación del Nielsen Norman Group, basada en el modelo matemático propuesto por Nielsen and Landauer (1993), que indica que con cinco participantes es posible identificar aproximadamente el 85 % de los problemas de usabilidad.[5]

Las entrevistas se llevaron a cabo a través de reuniones virtuales, facilitando la participación remota de los usuarios. Para recopilar las respuestas, se desarrolló en *Microsoft Forms* la encuesta sobre usabilidad de software, estructurada en tres secciones:

- **Información Básica**: recopila datos sobre la experiencia laboral, el sector de actividad, el nivel de familiaridad con herramientas de creación de formularios y bases de datos, y estimaciones de tiempos para tareas CRUD.
- **Ejecución de actividades o *User testing***: incluye instrucciones para completar una serie de tareas prácticas en *CITDEV Express* (flujos CRUD y gestión de entidades dinámicas).
- **Encuesta de usabilidad (SUS)**: contiene las 10 afirmaciones del *System Usability Scale*, con una escala Likert de 1 ("Totalmente en desacuerdo") a 5 ("Totalmente de acuerdo"), para medir la percepción de facilidad de uso, eficacia y satisfacción.

5.4.3. Participantes y criterios de selección

Se seleccionaron cinco participantes para evaluar *CITDEV Express* en los escenarios definidos. Se consideró que todos tuvieran una experiencia laboral similar, aunque procedieran de sectores distintos (3 en la industria tecnológica, 1 en la industria manufacturera y 1 en la educación básica).

- **Experiencia laboral:**
  - 4 participantes con más de 6 años (promedio >8 años).
  - 1 participante con 4–6 años.
- **Experiencia en desarrollo web:**
  - Con experiencia: 3.
  - Sin experiencia: 2.

[4]System Usability Scale es un cuestionario estandarizado desarrollado por John Brooke en 1986 para evaluar la usabilidad de sistemas y productos tecnológicos mediante una escala Likert de 10 ítems.

[5]El modelo propuesto fue: $U(n) = 1 - (1 - L)^n$, donde $U(n)$ representa la proporción acumulada de problemas descubiertos, L es la tasa promedio de descubrimiento por usuario ($\sim$0.31), y n es el número de usuarios. Con $n = 5$, se tiene: $U(5) \approx 1 - (1 - 0{,}31)^5 \approx 85$ %.

Al integrar participantes de diversos sectores, se evaluó el sistema en distintos niveles de experiencia, lo que permitió comprobar que la interfaz y las funcionalidades fueran prácticas para un amplio rango de usuarios.

5.4.4. Diseño de pruebas y casos de uso

Los participantes fueron guiados a través de una serie de tareas representativas dentro de *CITDEV Express*, centradas en la gestión de colecciones CRUD y la administración de entidades dinámicas. Se definieron cinco escenarios prácticos:

- Crear dos entidades relacionadas mediante un campo de referencia, especificando los atributos de cada una.
- Registrar datos para cada una de las entidades a partir de la información previamente suministrada.
- Modificar la estructura de una colección existente, eliminando un campo y reorganizando el orden del formulario.
- Editar un registro existente, modificando al menos uno de sus valores.
- Eliminar un registro existente a partir de un valor clave (como el número de serie).

Cada caso incluyó instrucciones paso a paso y criterios de éxito, como la completitud de los datos y la ausencia de asistencia, para comprobar que el usuario completara el flujo de forma autónoma.

5.4.5. Triangulación de datos

Para reforzar la validez de los resultados obtenidos, se implementó una triangulación metodológica, utilizando más de un método de evaluación para medir la usabilidad de *CITDEV Express*. Este enfoque permite contrastar los hallazgos obtenidos a través del *SUS* y *User Testing*.

- La aplicación del *SUS* permitió obtener una medición estandarizada de la satisfacción percibida por los usuarios, utilizando una escala validada y ampliamente adoptada en la industria para evaluar la usabilidad de sistemas interactivos.
- *User Testing* proporcionó datos objetivos sobre la eficacia y la eficiencia, permitiendo identificar cuellos de botella en la interacción con la plataforma.
- Al comparar los resultados obtenidos en ambos métodos, fue posible confirmar tendencias recurrentes y detectar discrepancias, lo que permitió refinar las conclusiones sobre la usabilidad del sistema.

Esta triangulación metodológica fortaleció la confiabilidad de la evaluación, asegurando que los hallazgos fueran representativos y aplicables en contextos similares.

5.4.6. Hallazgos

- Tres de los participantes trabajan en TI/Software y dos en industrias distintas.
- La mayoría de los participantes el *User testing* estimó entre 4 y 6 horas para crear un CRUD de cinco campos y alrededor de 3 horas para modificar una entidad (5,33 h y 3,33 h). Las omisiones de respuesta (40 %) correspondieron a los participantes sin experiencia en desarrollo de software.
- Durante la sesión de *User testing*, los participantes completaron las siguientes tareas en 39.2 minutos:
  - Creación de dos CRUDs (3 y 5 campos).
  - Creación de un registro en ambos CRUDs.
  - Relación entre entidades.
  - Modificación de la estructura de una entidad existente.
  - Edición y eliminación de registros.
- Una segunda encuesta a 15 profesionales del sector tecnológico estimó un promedio de 17,2 horas para crear un solo CRUD de 5 campos básicos (ver anexos).
- Las actividades realizadas en el *User testing* fueron más amplias y variadas que las contempladas en la encuesta a los 15 profesionales.
- La puntuación media SUS fue de 80,5 (rango 70–95), con una desviación estándar (SD)[6] de aproximadamente 9,8; lo que sitúa la usabilidad en niveles "Buenos" y "Excelentes".
- Analizando grupos con/sin experiencia en desarrollo de software, respectivamente:
  - Con experiencia: Media SUS ≈ 81,7; SD muestral ≈ 12,6. Alta dispersión relativa: puntuaciones entre 70 y 95.
  - Sin experiencia: Media SUS ≈ 78,8; SD muestral ≈ 12,4. Alta dispersión relativa: rango 70–87,5.

[6] La desviación estándar poblacional es:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2},$$

donde x_i es la puntuación SUS del participante i, μ es la media poblacional de las puntuaciones y N el número total de participantes.

5.5. Sesgos y/o riesgos en la evaluación

Como en toda evaluación empírica, es fundamental considerar los posibles sesgos y limitaciones que pueden influir en los resultados obtenidos. A continuación, se detallan algunos de los principales riesgos identificados y las estrategias para su control:

- Sesgo de selección:
  - La muestra de usuarios seleccionados pertenece a un contexto específico (desarrolladores en Cali, Colombia), lo que podría limitar la generalización de los resultados a otros entornos o perfiles de Citizen Developers en distintas regiones.
  - Medida de control: Se aplicó una segmentación por niveles de experiencia para asegurar la diversidad en los perfiles de usuarios evaluados. Sin embargo, futuras evaluaciones deberían incluir participantes de diferentes países y sectores.
- Sesgo de expectativa del evaluador: Existe el riesgo de que los autores, al estar involucrados en el desarrollo del sistema, interpreten los resultados de manera favorable.

Existe el riesgo de que los investigadores, al estar involucrados en el desarrollo del sistema, interpreten los resultados de manera favorable.

5.6. Recomendaciones

- Como parte del plan de mejora continua, se debería, a mediano plazo, subsanar las advertencias de documentación y, a largo plazo, añadir enlaces navegables (*HATEOAS*).
- Realizar iteraciones de prototipos enfocadas en las vistas de configuración para reducir la variabilidad en la experiencia reportada por los usuarios.
- Para garantizar un funcionamiento adecuado, el sistema debe ejecutarse sobre una infraestructura con una capacidad mínima de 512 MB de memoria RAM, 0.5 unidades de CPU y 1 GB de espacio en disco.

5.7. Conclusiones

- *CITDEV Express* reduce en más del 96 % el tiempo requerido para crear y modificar CRUDs en comparación con el desarrollo tradicional estimado por profesionales del desarrollo.
- La amplitud de las tareas realizadas en el user testing valida la cobertura funcional de CitDev Express frente a escenarios reales.
- *CITDEV Express* cumple con el requerimiento no funcional RNF03 bajo condiciones de carga media, alcanzando un rendimiento estable de hasta 7.3 transacciones por segundo sin la presencia de errores.

- Las pruebas evidenciaron que las operaciones críticas, como la autenticación y la consulta de registros, mantienen tiempos de respuesta adecuados y un comportamiento estable en escenarios controlados.
- Se identificó el punto de fallo del sistema al superar los 25 usuarios concurrentes, lo cual representa un insumo para definir estrategias de escalabilidad en entornos de producción.
- La definición actual del swagger del API satisface la sintaxis de *OpenAPI 3.0* y permite el consumo del servicio. Sin embargo, las inconsistencias detectadas impiden la validación en entornos CI/CD. Corregirlas otorgaría conformidad plena.
- El API emplea recursos bien identificados y verbos HTTP con semántica adecuada, permitiendo la interoperabilidad con los clientes. Para alcanzar el nivel 3 en *MMR* se requiere incorporar hipermedia (*HATEOAS*).
- La combinación de verificar la conformidad con *OpenAPI 3.0* y el *Modelo de Madurez de Richardson* permitió detectar brechas y orientar la evolución del API hacia una mayor interoperabilidad en el futuro.
- A partir de la puntuación SUS global (80,5) y la dispersión observada (SD $\approx$ 9,8), la plataforma presenta buena aceptación, aunque con variabilidad en la percepción de algunos usuarios.
- Aunque los grupos de usuarios con/sin experiencia en desarrollo de software valoran positivamente la plataforma (medias $>$75), la alta desviación estándar ($\approx$ 12 puntos) en ambos grupos revela diferencias significativas en la percepción de usabilidad. Estas diferencias puntuales sugieren la necesidad de afinar los elementos de la interfaz, los flujos de trabajo y la asistencia para conseguir una experiencia más homogénea.

Capítulo 6

Conclusiones

El presente capítulo expone las conclusiones generales del proyecto de grado titulado "CITDEV Express: Solución tecnológica para impulsar el Desarrollo de Software a través de Citizen Developers", resultado del análisis de los objetivos planteados, el desarrollo de la solución tecnológica y las evaluaciones realizadas. Se presentan además las lecciones aprendidas más relevantes identificadas durante el proceso de diseño, implementación y evaluación de *CITDEV Express*, así como las líneas de trabajo futuras orientadas a fortalecer y evolucionar la plataforma. El contenido de este capítulo se fundamenta en la relación directa con los objetivos específicos formulados al inicio del proyecto, proporcionando un cierre estructurado y coherente con el alcance definido.

6.1. Conclusiones generales

El proyecto de grado logra cumplir los objetivos propuestos, demostrando su pertinencia académica y práctica en el contexto de la ingeniería de software. En primer lugar, *CITDEV Express* permite que personas sin conocimientos técnicos avanzados configuren y gestionen operaciones básicas de datos (*CRUD*) mediante una interfaz gráfica, sin necesidad de escribir código. Esta característica representa un aporte para la inclusión de perfiles no técnicos, contribuyendo a la mitigación del déficit de talento en tecnologías de la información en entornos organizacionales.

En segundo lugar, la implementación de entidades dinámicas como mecanismo central de configuración evidencia ser una estrategia para garantizar la flexibilidad y adaptabilidad del sistema, favoreciendo su mantenibilidad y evolución sin requerir modificaciones al código fuente. Esto se traduce en un modelo de desarrollo accesible para organizaciones de diferentes tamaños.

En cuanto a la arquitectura técnica, se confirma que el enfoque modular y el stack tecnológico seleccionado permiten cumplir con los requisitos funcionales y no funcionales definidos, destacando el rendimiento estable registrado en las pruebas de carga (7.3 transacciones por segundo) y el cumplimiento del requisito no funcional **RNF03**, aspectos documentados en la **Sección 5.3**.

Adicionalmente, el uso conjunto de la verificación de conformidad con *OpenAPI 3.0* y el Modelo de Madurez de Richardson, permitió identificar deficiencias puntuales en el contrato de la *API* y trazar el camino técnico para avanzar al nivel 3 del modelo mediante la incorporación de hipermedia (*HATEOAS*). Este paso aportará descubrimiento de recursos durante la ejecución y menor dependencia entre cliente y servidor.

La *API* se sitúa en el nivel 2, los recursos están definidos y los métodos HTTP se emplean con la semántica prevista, lo que facilita la interoperabilidad con distintos clientes. Se recomienda completar la transición al nivel 3 mediante hipermedia y añadir validación automática en entornos de integración continua (*CI*) a fin de fortalecer la calidad del contrato y facilitar su evolución.

Desde la perspectiva de la experiencia de usuario, los resultados de la evaluación de usabilidad indican una aceptación positiva (puntuación *SUS* 80,5), aunque con variabilidad en la percepción de los usuarios según su nivel de experiencia previa. Esta observación subraya la necesidad de continuar refinando aspectos de la interfaz y flujos de trabajo para obtener una experiencia más homogénea entre diferentes perfiles de usuario.

Finalmente, se destaca como valor agregado del proyecto el enfoque metodológico adoptado, basado en la integración de metodologías ágiles y tradicionales, complementado con recolección de datos mediante entrevistas a stakeholders. Este enfoque permitió alinear el desarrollo de *CITDEV Express* con necesidades reales del sector, fortaleciendo su aplicabilidad y relevancia práctica.

6.2. Lecciones aprendidas

Durante el desarrollo del proyecto de grado, se identificaron las siguientes lecciones aprendidas:

- **Importancia de la integración de enfoques ágiles y tradicionales:** la combinación de metodologías tradicionales para la planificación inicial con técnicas ágiles como *Scrum* facilitó una mayor adaptabilidad frente a cambios en los requisitos, evidenciando la necesidad de flexibilidad en proyectos de software orientados a investigación aplicada.
- **Valor de la validación temprana con stakeholders:** las entrevistas estructuradas realizadas durante la etapa de planeación permitieron ajustar funcionalidades y priorizar características alineadas con las necesidades reales de la industria, resaltando la importancia de involucrar a usuarios finales desde etapas tempranas.
- **Desafíos en el diseño de plataformas para *Citizen Developers*:** se observó que diseñar soluciones dirigidas a usuarios sin formación técnica avanzada requieren de un enfoque centrado en la usabilidad y la simplificación de interfaces de usuario.
- **Limitaciones técnicas en la interoperabilidad:** aunque se logró implementar un marco de interoperabilidad funcional mediante *API RESTful* y *JWT*, se identificaron retos relacionados con estándares avanzados como *HATEOAS*, subrayando la necesidad de explorar modelos de integración más robustos en futuras versiones.
- **Importancia de la documentación arquitectónica:** la aplicación del *Modelo C4* para documentar la arquitectura de *CITDEV Express*, facilitó la comunicación entre los integrantes del equipo de desarrollo y evaluadores externos, demostrando que una documentación estructurada es útil para proyectos de software académicos y profesionales.
- **Gestión de expectativas frente a alcance limitado:** se constató que es fundamental definir y comunicar claramente las limitaciones del proyecto a los stakeholders, especialmente cuando se identifican requisitos que exceden el alcance inicial.

Estas lecciones aprendidas constituyen un insumo para futuros desarrollos en el área de plataformas orientadas a *Citizen Developers*, así como para iniciativas académicas que busquen articular investigación aplicada con necesidades prácticas del sector tecnológico.

6.3. Trabajos futuros

Con el objetivo de fortalecer y evolucionar *CITDEV Express*, se plantean los siguientes trabajos futuros:

- **Ampliación de la capacidad de configuración:** incorporar nuevos tipos de campos de entrada, habilitar la definición de campos lógicos calculados dinámicamente, implementar controles de grilla avanzados y almacenamiento de versiones históricas de las configuraciones de entidades dinámicas.
- **Mejoras en la validación de datos:** desarrollar mecanismos que permitan definir restricciones avanzadas, tales como expresiones regulares, rangos numéricos, dependencias condicionales entre campos y validaciones basadas en reglas personalizadas configurables por el usuario.
- **Extensión de funcionalidades programables:** permitir la ejecución de lógica personalizada antes o después de las operaciones CRUD, integrando eventos configurables a través de la interfaz gráfica, para potenciar la flexibilidad del sistema sin necesidad de intervención directa en el código fuente.
- **Optimización de la usabilidad y experiencia de usuario:** incorporar características adicionales tales como soporte para temas oscuros, opciones de personalización visual, accesibilidad para usuarios con capacidades diversas y mejoras continuas basadas en pruebas de usabilidad formalizadas.
- **Integración de Inteligencia Artificial (IA):** explorar el uso de modelos de IA para sugerir configuraciones de campos y entidades dinámicas, detectando patrones y proponiendo configuraciones automatizadas a partir del propósito declarado por el *Citizen Developer*.
- **Automatización de despliegue y mantenimiento:** incorporar prácticas de *Continuous Integration* y *Continuous Deployment* (CI/CD) para facilitar la actualización continua de la plataforma sin afectar su disponibilidad, garantizando procesos de mantenimiento eficientes y controlados.

Capítulo 7

Anexos

En el desarrollo de este proyecto, se ha producido una serie de documentos complementarios que respaldan las metodologías aplicadas, los resultados obtenidos y las herramientas utilizadas durante el proceso. Estos archivos incluyen desde reportes técnicos hasta conjuntos de datos y recursos digitales que ofrecen contexto adicional para el lector interesado en profundizar en los procedimientos descritos.

A continuación se listan los documentos de anexo disponibles en el **repositorio de código fuente** del proyecto en la carpeta */anexos*:

- Entrevistas de tres etapas.
- Manual de usuario.
- Proceso de evaluación de usabilidad.
  - Encuesta de usabilidad del software.
  - Guía de actividades.
  - Escala de usabilidad System Usability Scale (SUS).
  - Encuesta de estimación de esfuerzo para el desarrollo de funcionalidades CRUD.
- Script de las pruebas de rendimiento.
- Documentos técnicos.
  - Documento de despliegue.
  - Diagrama de clases.
- Repositorio de código fuente: `https://github.com/breinerhdez/form-base`
- Las instrucciones de instalación se encuentran en el `README.md` del repositorio.

Bibliografía

Alvarado-Morales, L. M. and Zambrano-Roldán, K. (2020). Perfil del consumidor digital y aceptación de oferta en pandemia. *Revista Científica Arbitrada de Investigación en Comunicación, Marketing y Empresa*, 3.

Amazon Web Services (s.f.a). ¿Qué es la interoperabilidad? Disponible en: https://aws.amazon.com/es/what-is/interoperability/. Accedido: 2023-11-25.

Amazon Web Services (s.f.b). ¿qué es api restful? Consultado el 21 de abril de 2025.

Amazon Web Services (s.f.c). ¿Qué es el SDLC? - Explicación del ciclo de vida del desarrollo de software - AWS. Disponible en: https://aws.amazon.com/es/what-is/sdlc/. Accedido: 2023-11-13.

Amazon Web Services (s.f.d). ¿Qué es una CLI? (Interfaz de línea de comandos). Disponible en: https://aws.amazon.com/es/what-is/cli/. Accedido: 2023-10-30.

Araujo, J. M. A., de Moura, A. C. E., da Silva, S. L. B., Holanda, M., Ribeiro, E. d. O., and da Silva, G. L. (2021). Comparative performance analysis of nosql cassandra and mongodb databases. In *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6.

Avishahar-Zeira, A. and Lorenz, D. H. (2023). Could No-Code Be Code? Toward a No-Code Programming Language for Citizen Developers. *Onward! 2023: Proceedings of the 2023 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pages 103–119.

Bootstrap (s.f.). Breadcrumb. https://getbootstrap.com/docs/5.1/components/breadcrumb/.

Brisaboa, N. R., Cortinas, A., Luaces, M. R., and Pedreira, O. (2016). Aplicando scaffolding en el desarrollo de lıneas de producto software. In *Aplicando scaffolding en el desarrollo de Lıneas de Producto Software*.

Brooke, J. (1995). Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189.

Brown, S. (2018). *The C4 model for visualising software architecture*. Leanpub.

Casas, S., Cruz, D., Vidal, G., and Constanzo, M. (2021). Uses and applications of the openapi/swagger specification: a systematic mapping of the literature. In *2021 40th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–8.

Chodorow, K. (2019). *MongoDB: The Definitive Guide*. O'Reilly Media, 3rd edition.

Corradini, D., Montolli, Z., Pasqua, M., and Ceccato, M. (2024). DeepREST: Automated Test Case Generation for REST APIs Exploiting Deep Reinforcement Learning. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ASE '24, page 1383–1394, New York, NY, USA. Association for Computing Machinery.

Date, C. J. (2004). *An Introduction to Database Systems*. Pearson Education, 8th edition.

de Rosal Ignatius Moses Setiadi, Najib, A. F., Rachmawanto, E. H., Sari, C. A., Sarker, M. K., and Rijati, N. (2019). A comparative study MD5 and SHA1 algorithms to encrypt REST API authentication on mobile-based application. *2019 International Conference on Information and Communications Technology, ICOIACT*, pages 206–211.

Diallo, S. Y., Herencia-Zapana, H., Padilla, J. J., and Tolk, A. (2011). Understanding interoperability. *Emerging M and S Applications in Industry and Academia Symposium 2011, EAIA 2011 - 2011 Spring Simulation Multiconference*, pages 84–91.

Django (s.f.). Django makes it easier to build better web apps more quickly and with less code. Disponible en: https://www.djangoproject.com/. Accedido: 2023-11-13.

Fielding, R. T. and Reschke, J. (2014). Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. https://datatracker.ietf.org/doc/html/rfc7231.

Fielding, R. T. and Taylor, R. N. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine.

García Rico, J. C. (2023). Seguimos en grave déficit en talento TIC / Análisis. Disponible en: https://www.eltiempo.com/tecnosfera/novedades-tecnologia/analisis-de-jose-carlos-garcia-seguimos-en-grave-deficit-en-talento-tic-811409. Accedido: 2023-11-13.

Google (2024). Ruby on rails, ruby.

Heroku (s.f.a). Heroku Pricing. Disponible en: https://www.heroku.com/pricing/. Consultado el 20 de mayo de 2025.

Heroku (s.f.b). Heroku Solutions. Disponible en: https://www.heroku.com/solutions/. Consultado el 20 de mayo de 2025.

Hurlburt, G. F. (2021). Low-Code, No-Code, What's Under the Hood? *IT Professional*, 23(December):4–7.

IBM (2024). ¿qué es la eficiencia operativa? https://www.ibm.com/mx-es/topics/operational-efficiency. Consultado el 21 de julio de 2025.

IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, pages 1–84.

ISO/IEC (2011). ISO/IEC 25010:2011: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Norma ISO/IEC 25010:2011.

Jones, M., Bradley, J., and Sakimura, N. (2015). JSON Web Token (JWT). https://www.rfc-editor.org/rfc/rfc7519.

Jones, M. and Hardt, D. (2015). The JSON Web Token (JWT) Profile for OAuth 2.0. https://datatracker.ietf.org/doc/html/rfc7523.

Jones, M., Hardt, D., and Recordon, D. (2012). The OAuth 2.0 Authorization Framework: Bearer Token Usage. https://datatracker.ietf.org/doc/html/rfc6750.

Jung, J. and Katz, R. (2022). Impacto del covid-19 en la digitalización de américa latina.

jwt.io (s.f.). Introduction to JSON Web Tokens. Disponible en: https://jwt.io/introduction/. Accedido: 2023-10-30.

Khaliluzzaman, M. and Chowdhury, I. I. (2016). Pre and post controller based mvc architecture for web application. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 552–557.

Khorram, F., Mottu, J.-m., and Sunyé, G. (2020). Challenges & opportunities in low-code testing. *MODELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems*, pages 1–10.

Laravel (s.f.). Forms & html. Disponible en: https://laravel.com/docs/4.2/html. Accedido: 2023-10-30.

Madden, N. (2020). *API Security in Action*. Manning.

Maida, E. G. and Pacienzia, J. (2015). Metodologías de desarrollo de software.

Mazinanian, D. and Tsantalis, N. (2017). Cssdev: Refactoring duplication in cascading style sheets. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 63–66.

Microsoft (2023). Chapter 16: Quality Attributes. Disponible en: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ee658094(v=pandp.10)?redirectedfrom=MSDN. Accedido: 2023-12-15.

Misu, M. R. H. and Satter, A. (2022). An Exploratory Study of Analyzing JavaScript Online Code Clones. *2022 IEEE/ACM 30th International Conference on Program Comprehension (ICPC)*, pages 94–98.

Nicerova (2017). Autenticación con Node.js y JWT. https://nicerova.wordpress.com/2017/11/22/autenticacion-con-nodejs-y-jwt/. Accedido: 2025-02-27.

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.

Nielsen, J. and Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, pages 206–213. ACM.

Norman, D. (2013). *The DESIGN of EVERYDAY THINGS*. Basic Books.

Oonhawat, B. and Nupairoj, N. (2017). Hotspot management strategy for real-time log data in mongodb. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 221–227.

OpenJS Foundation (2025a). jQuery. Disponible en: https://jquery.com/. Consultado el 11 de junio de 2025.

OpenJS Foundation (2025b). jQuery UI. Disponible en: https://jquery.com/. Consultado el 11 de junio de 2025.

Ostrand, T. (2002). Black-box testing. *Encyclopedia of Software Engineering*.

Pantelelis, M. and Kalloniatis, C. (2022). Mapping CRUD to Events - Towards an object to event-sourcing framework. *PCI '22: Proceedings of the 26th Pan-Hellenic Conference on Informatics*, pages 285–289.

Paz, F. and Pow-Sang, J. A. (2015). Usability evaluation methods for software development: A systematic mapping review. In *2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA)*, pages 165–178.

Pergl, R. and Jíša, N. (2024). Semantic Analysis of API Blueprint and OpenAPI Specification. In *World Conference on Information Systems and Technologies*. Springer.

Peruma, A. and Krutz, D. E. (2018). Security: A Critical Quality Attribute in Self-Adaptive Systems. *2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 188–189.

Pramono, L. H. and Yana Javista, Y. K. (2021). Firebase Authentication Cloud Service for RESTful API Security on Employee Presence System. *2021 4th International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2021*, pages 1–6.

Ramírez Montaño, S. (s.f.). FastAPI. Disponible en: https://fastapi.tiangolo.com/. Accedido: 2024-01-28.

Rubin, J. and Chisnell, D. (2008). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests - Jeffrey Rubin, Dana Chisnell - Google Libros*. Wiley Publishing, Inc., second edition.

Ruby on Rails (s.f.). Compress the complexity of modern web apps. Disponible en: https://rubyonrails.org/. Accedido: 2023-11-13.

Sulbarán, I. (2023). ¿Qué es la transformación digital y cuáles son sus 4 tecnologías base? Disponible en: https://global.tiffin.edu/noticias/que-es-la-transformacion-digital. Accedido: 2023-11-25.

Symfony (s.f.). Forms. Disponible en: https://symfony.com/doc/2.4/book/forms.html. Accedido: 2023-10-30.

Taibi, D., Lenarduzzi, V., and Pahl, C. (2017). Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. *IEEE Cloud Computing*, pages 22–32.

Tilkov, S. and Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, pages 80–83.

Tolk, A. (2013). Interoperability, Composability, and Their Implications for Distributed Simulation: Towards Mathematical Foundations of Simulation Interoperability. *2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications*, pages 3–9.

Vaughan-Nichols, S. J. (2010). Will html 5 restandardize the web? *Computer*, 43(4):13–15.

Vázquez-Ingelmo, A., García-Holgado, A., and García-Peñalvo, F. J. (2020). C4 model in a software engineering subject to ease the comprehension of uml and the software. In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 919–924.

Wang, S., Ling, Z., Zhang, Y., Liu, R., Kraunelis, J., Jia, K., Pearson, B., and Fu, X. (2022). Implication of animation on android security. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 1122–1132.

Wang, X. (2011). Ajax technology applications in the network test system. In *2011 International Conference on Electrical and Control Engineering*, pages 1954–1956.

Wu, B., Cao, Y., and Liu, X. (2024). Research and analysis of commonly used node.js framework. In *2024 Prognostics and System Health Management Conference (PHM)*, pages 104–107. IEEE.

Yablonski, J. (2024). *1. Jakob's Law | Laws of UX, 2nd Edition.* O'Reilly Media, Inc., second edition.

Yingda, L., Jianzhuang, L., and Xiaowen, C. (2018). Variadic parameter command pattern and the utilization in mvc. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pages 374–377.

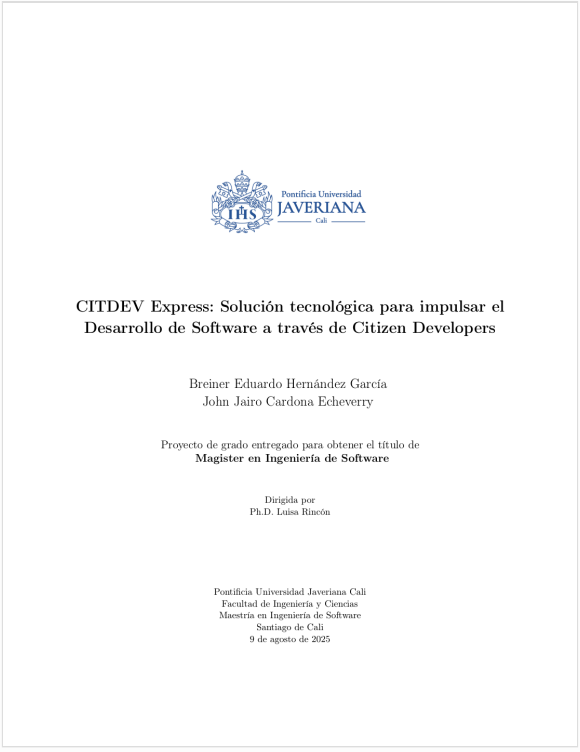
Yuncharoen, N. and Suwannasart, T. (2024). Generating RESTful API Test Scripts with Database Constraints using OpenAPI Specification. In *Proceedings of the 2024 The 6th World Symposium on Software Engineering (WSSE)*, WSSE '24, page 15–26, New York, NY, USA. Association for Computing Machinery.

Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega:	BREINER EDUARDO HERNANDEZ GARCIA
Título del ejercicio:	[Actividad] Valida tus trabajos con Turnitin
Título de la entrega:	20250819_Maestria_ProyectoDeGrado_CitDevExpress.pdf
Nombre del archivo:	20250819_Maestria_ProyectoDeGrado_CitDevExpress.pdf
Tamaño del archivo:	6.26M
Total páginas:	109
Total de palabras:	32,782
Total de caracteres:	187,509
Fecha de entrega:	19-ago-2025 09:48p. m. (UTC-0500)
Identificador de la entrega:	2732172936



Pontificia Universidad JAVERIANA Cali

CITDEV Express: Solución tecnológica para impulsar el Desarrollo de Software a través de Citizen Developers

Breiner Eduardo Hernández García
John Jairo Cardona Echeverry

Proyecto de grado entregado para obtener el título de
Magister en Ingeniería de Software

Dirigida por
Ph.D. Luisa Rincón

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Maestría en Ingeniería de Software
Santiago de Cali
9 de agosto de 2025