UNIVERSIDAD NACIONAL DE JULIACA

VICERRECTORADO DE INVESTIGACIÓN

ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE Y SISTEMAS

"Sistema Web para la Gestión y Almacenamiento de Datos Ambientales"

TIPO DE INVESTIGACIÓN

Investigación Científica de Desarrollo Tecnológico

Aplicado

LÍNEA DE INVESTIGACIÓN

Desarrollo de Tecnología para la Mitigación y Adaptación al Cambio Climático

INFORME FINAL DE INVESTIGACIÓN FORMATIVA

Juliaca, Perú

2025

UNIVERSIDAD NACIONAL DE JULIACA

COMISIÓN ORGANIZADORA

Dr. Edelfre Flores Velazques

Presidente de Comisión Organizadora

Dr. Arrufo Alcántara Hernández

Vicepresidente Académico

Dr. Richard Dante Ramírez Ormeño

Vicepresidente de Investigación

DIRECCIÓN DE INNOVACIÓN Y TRANSFERENCIA TECNOLÓGICA

Dr. Richard Dante Ramírez Ormeño

Vicepresidente de Investigación

EQUIPO DE INVESTIGACIÓN FORMATIVA

Dra. Vilma Sarmiento Mamani

Investigador principal

Esau Carlo Paredes Taipe

Representante del Equipo de Investigación Formativa

Dario Daniel Quispe Quispe

Asistente de Investigación

Aldo Kerson Quispe Chambi

Asistente de Investigación

Edison David Supo Cruz

Asistente de Investigación

DEDICATORIA

Dedicamos el presente informe de investigación, en primer lugar, a Dios, por brindarnos la fortaleza, la salud y la sabiduría necesarias para culminar este trabajo.

A los docentes de la Universidad Nacional de Juliaca, por su apoyo incondicional, comprensión y motivación constante a lo largo del proceso académico.

Finalmente, a todas las personas que, de una u otra manera, contribuyeron a la realización de la presente investigación.

AGRADECIMIENTOS

Agradecemos profundamente al equipo de investigación, por su dedicación, paciencia y valiosos aportes durante el desarrollo de este proyecto.

De igual manera, agradecemos a nuestros compañeros y amigos por el apoyo brindado y por compartir experiencias que enriquecieron este proceso.

Finalmente, agradecemos a la Universidad Nacional de Juliaca por proporcionar los recursos necesarios para la realización de esta investigación.

ÍNDICE DE CONTENIDOS

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

La investigación denominada “Sistema Web para la Gestión y Almacenamiento de Datos Ambientales” tuvo como objetivo general desarrollar un Sistema Web para la Gestión y Almacenamiento de Datos Ambientales, conforme lo programado en el cronograma establecido, dicho sistema permite una verificación rápida, segura y capaz optimizar la gestión de acceso y almacenamiento mejorando la seguridad de acceso. La metodología aplicada siguió un enfoque cualitativo, tipo Investigación Científica de Desarrollo Tecnológico Aplicado, diseño no experimental. Se utilizó la metodología SCRUM, en donde se estructuró en varias fases: diagnóstico y análisis de requisitos, donde se identificaron las necesidades para el diseño del sistema, que incluyó la planificación de la arquitectura y tecnologías a utilizar; desarrollo del software, con la codificación de las funcionalidades principales y la integración con la base de datos; pruebas del sistema en un entorno controlado. Los resultados incluyen la implementación de una página web dinámica que incluye bases de datos. La metodología Scrum demostró ser efectivo para gestionar cambios en requisitos y mantener entregas incrementales, garantizando alineación con las expectativas del cliente y la priorización con MoSCoW y el uso de Jira optimizaron la asignación de recursos y el seguimiento de tareas. El Diseño de base de datos para la Gestión y Almacenamiento de Datos Ambientales fue desarrollado en función a los requerimientos obtenidos para lograr Almacenamiento eficiente y seguro de datos ambientales. En cuanto al Diseño de interfaz de Usuario es interactiva para la Gestión y Almacenamiento de Datos Ambientales, por lo que la integración de tecnologías modernas (React.js, Supabase) aseguró escalabilidad y rendimiento, así como la evaluación de la escalabilidad y rendimiento de la base de datos.

Palabras clave: datos ambientales, base de datos, radiación solar, datos climatológicos.

ABSTRACT

The research project, entitled “Web System for the Management and Storage of Environmental Data,” had the overall objective of developing a web system for the management and storage of environmental data, according to the established schedule. This system allows for fast, secure verification and optimizes access and storage management, improving access security. The methodology employed followed a qualitative approach, specifically a Scientific Research of Applied Technological Development, with a non-experimental design. The Scrum methodology was used, structured in several phases: requirements analysis and diagnosis, where the needs for system design were identified, including planning the architecture and technologies to be used; software development, with the coding of the main functionalities and integration with the database; and system testing in a controlled environment. The results include the implementation of a dynamic website that incorporates databases. The Scrum methodology proved effective in managing changes to requirements and maintaining incremental deliveries, ensuring alignment with client expectations. Prioritization with MoSCoW and the use of Jira optimized resource allocation and task tracking. The database design for Environmental Data Management and Storage was developed based on the requirements obtained to achieve efficient and secure environmental data storage. The user interface design is interactive for Environmental Data Management and Storage, and the integration of modern technologies (React.js, Supabase) ensured scalability and performance, as well as the evaluation of the database's scalability and performance.

Keywords: environmental data, database, solar radiation, climatological data.

INTRODUCCIÓN

En un entorno tecnológico en constante evolución, la gestión eficiente de datos se ha convertido en un elemento crucial para el éxito de las aplicaciones y sistemas. La diversidad de herramientas y el avance de las tecnologías de almacenamiento generan requisitos específicos que los sistemas de bases de datos deben satisfacer, lo cual representa un desafío significativo, ya que la elección tecnológica depende en gran medida de factores de contexto. En el ámbito de los datos ambientales, una gestión robusta es fundamental para el monitoreo, análisis y apoyo a la toma de decisiones. Sin embargo, en la región de Puno se identifica una adopción limitada de tecnologías digitales avanzadas para la visualización y el almacenamiento sistematizado de dicha información a través de plataformas web.

Este contexto revela un problema la carencia en la región de Puno de un sistema web integrado que permita la gestión, el almacenamiento seguro y la visualización accesible de datos ambientales. Esta deficiencia obstaculiza el monitoreo efectivo y el análisis oportuno, generando problemas de accesibilidad, seguridad en el manejo de la información y capacidad para transformar datos crudos en conocimiento comprensible mediante visualizaciones claras. Para abordar esta problemática, esta investigación se planteó como objetivo general desarrollar un Sistema Web para la Gestión y Almacenamiento de Datos Ambientales. Este propósito se desglosó en objetivos específicos dirigidos a: 1) Analizar los requisitos técnicos y funcionales necesarios para el diseño Web para la Gestión y Almacenamiento de Datos Ambientales; 2) Diseñar la arquitectura del sistema Web para la Gestión y Almacenamiento de Datos Ambientales; y 3) Implementar el Sistema Web para la Gestión y Almacenamiento de Datos Ambientales, gestión de acceso y visualización mediante tablas y gráficos. Estas metas buscaban responder a las preguntas de investigación sobre los requisitos clave, la estructura arquitectónica idónea y el grado de mejora que un sistema de este tipo puede aportar en términos de accesibilidad, seguridad y utilidad frente a los métodos tradicionales.

La justificación de este estudio radica en su potencial para ofrecer una solución escalable y adaptada al contexto local, que mejore la accesibilidad y seguridad de la información ambiental, facilite su análisis y contribuya al conocimiento sobre la implementación de tecnologías web y de bases de datos en entornos con recursos y necesidades específicas. La investigación se llevó a cabo en el contexto de la provincia de San Román, de la región de Puno, Perú, y el desarrollo técnico del sistema web se ejecutó durante los meses de enero a mayo, siguiendo una metodología estructurada

de desarrollo de software, como es la metodología SCRUM.

Se sostuvo que el desarrollo e implementación de este Sistema Web permitiría una verificación más rápida, un almacenamiento más seguro y una gestión de acceso optimizada, mejorando significativamente la accesibilidad y utilidad de la información ambiental, en comparación con las prácticas o herramientas preexistentes. No obstante, el trabajo reconoció ciertas limitaciones, como el alcance delimitado por el tiempo disponible, el enfoque en un conjunto específico de datos ambientales, la posible limitación en la calidad de los datos históricos para la carga inicial, y el hecho de que un estudio exhaustivo de adopción y usabilidad a largo plazo por los usuarios finales queda como ámbito para futuras investigaciones.

CAPITULO I
MARCO TEÓRICO

1.1. Objetivos del estudio

Indicar lo que se obtuvo en la investigación dando respuesta al problema de investigación formulado. Se redactan un objetivo general y tres a cinco objetivos específicos. Se redacta en tiempo infinitivo.

1.1.1. Objetivo General

Desarrollar un Sistema Web para la Gestión y Almacenamiento de Datos Ambientales.

1.1.2. Objetivos específicos

- Analizar los requisitos técnicos y funcionales necesarios para el diseño Web para la Gestión y Almacenamiento de Datos Ambientales.
- Diseñar la arquitectura del sistema Web para la Gestión y Almacenamiento de Datos Ambientales.
- Implementar el Sistema Web para la Gestión y Almacenamiento de Datos Ambientales.

1.2. Fundamentos Tecnológicos para Sistemas de Gestión de Datos Ambientales

1.2.1. Evolución de los Sistemas de Gestión de Datos

La gestión de datos ha experimentado una transformación radical en las últimas décadas, pasando de sistemas centralizados y monolíticos a arquitecturas distribuidas y especializadas. Según Date (2003), los fundamentos teóricos de las bases de datos relacionales establecidos por Edgar F. Codd en 1970 sentaron las bases para la estandarización en el manejo de datos estructurados. Sin embargo, la explosión de datos no estructurados y semiestructurados en la era digital ha requerido la evolución hacia paradigmas más flexibles (Stonebraker & Hellerstein, 2005).

En el contexto específico de datos ambientales, la complejidad aumenta debido a la naturaleza multidimensional de la información (datos geográficos, series temporales, mediciones de múltiples variables) y la necesidad de integración de fuentes heterogéneas. Como señala Fegraus et al. (2011), los sistemas de información ambiental enfrentan desafíos únicos en cuanto a escalabilidad, interoperabilidad y mantenimiento de la calidad de los datos a largo plazo.

1.2.2. Tecnologías Web Modernas para Aplicaciones de Datos

El desarrollo de aplicaciones web ha evolucionado significativamente desde las páginas estáticas hasta las Single Page Applications (SPAs) complejas. React, desarrollado por Facebook, representa un paradigma importante en este desarrollo al introducir un modelo de programación basado en componentes reutilizables y un DOM virtual que optimiza el rendimiento (Facebook Open Source, 2020). Según Banks y Porcello (2017), React facilita la creación de interfaces de usuario dinámicas y responsivas, características esenciales para sistemas de visualización de datos ambientales que requieren actualizaciones en tiempo real y representaciones gráficas complejas.

La arquitectura de componentes de React permite la construcción modular de interfaces, donde cada componente puede gestionar su propio estado y recibir datos a través de propiedades. Este enfoque es particularmente valioso para aplicaciones de datos ambientales, donde diferentes visualizaciones (gráficos, mapas, tablas) pueden implementarse como componentes independientes que se comunican a través de un estado global gestionado por herramientas como Redux o Context API (Eve Porcello & Alex Banks, 2020).

1.2.3. Sistemas de Gestión de Bases de Datos Relacionales (RDBMS)

Los Sistemas de Gestión de Bases de Datos Relacionales (RDBMS) han sido la piedra angular de la gestión de datos estructurados durante más de cuatro décadas. Basados en el modelo relacional propuesto por Codd (1970), estos sistemas organizan los datos en tablas con filas y columnas, manteniendo relaciones entre ellas mediante claves primarias y foráneas. Entre los gestores más populares se encuentran:

MySQL: Desarrollado originalmente por MySQL AB, ahora propiedad de Oracle Corporation, es conocido por su velocidad y facilidad de uso en aplicaciones web (Widenius & Axmark, 2002).

PostgreSQL: Sistema de código abierto que destaca por su robustez, cumplimiento de estándares SQL y soporte para tipos de datos avanzados (Momjian, 2001).

Oracle Database: Solución comercial líder en el mercado empresarial, reconocida por su escalabilidad y características avanzadas (Loney, 2008).

Microsoft SQL Server: Sistema propietario de Microsoft que se integra estrechamente con otras tecnologías de la misma compañía (Ben-Gan et al., 2019).

DB2: Desarrollado por IBM, ofrece capacidades avanzadas de procesamiento analítico (Chong et al., 2005).

La elección entre estos sistemas depende de múltiples factores, incluyendo requisitos de escalabilidad, presupuesto, características técnicas específicas y ecosistema tecnológico existente. Como señalan Garcia-Molina et al. (2008), cada RDBMS presenta ventajas en contextos específicos, y la selección debe alinearse con los objetivos del sistema a desarrollar.

1.2.4. PostgreSQL: Características y Aplicaciones

PostgreSQL se distingue entre los RDBMS de código abierto por su compromiso con el cumplimiento de estándares y su arquitectura extensible. Según Obe y Hsu (2017), PostgreSQL ofrece características avanzadas que lo hacen adecuado para aplicaciones complejas:

- Soporte para tipos de datos complejos: Incluye tipos nativos para arrays, JSON, XML, y datos geométricos/spatial mediante la extensión PostGIS.
- Concurrencia mediante MVCC (Multiversion Concurrency Control): Permite operaciones de lectura y escritura concurrentes sin bloqueos excesivos.
- Extensibilidad: Los usuarios pueden definir sus propios tipos de datos, funciones, y operadores.
- Integridad transaccional: Cumple completamente con las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).
- Replicación y alta disponibilidad: Soporta diversas estrategias de replicación para redundancia y escalabilidad.

Para aplicaciones de datos ambientales, PostgreSQL es particularmente adecuado debido al soporte de PostGIS para datos georreferenciados. Como demuestra Obe y Hsu (2020), PostGIS permite almacenar, consultar y manipular datos espaciales con funciones especializadas para análisis geográfico, una capacidad esencial para sistemas de monitoreo ambiental.

1.2.5. Supabase como Plataforma Backend-as-a-Service

Supabase representa una evolución en el desarrollo de aplicaciones al ofrecer una capa de abstracción sobre PostgreSQL que simplifica tareas comunes de backend. Según su documentación oficial (Supabase, 2023), la plataforma proporciona:

- Base de datos PostgreSQL gestionada: Incluye todas las capacidades de PostgreSQL con herramientas adicionales de administración.
- Autenticación y autorización: Sistema completo de gestión de usuarios con soporte para múltiples proveedores de identidad.
- API REST y GraphQL automáticas: Genera automáticamente APIs basadas en el esquema de la base de datos.
- Suscripciones en tiempo real: Permite a los clientes suscribirse a cambios en la base de datos mediante WebSockets.
- Almacenamiento de objetos: Sistema para archivos y medios con control de acceso granular.

La arquitectura de Supabase se alinea con la tendencia hacia el desarrollo de aplicaciones sin servidor (serverless), donde los desarrolladores pueden enfocarse en la lógica de negocio mientras la plataforma gestiona la infraestructura subyacente. Como explica Roberts (2022), este enfoque reduce significativamente la barrera de entrada para desarrollar aplicaciones robustas con funcionalidades avanzadas de backend.

1.2.6. Bases de Datos NoSQL

El término "NoSQL" abarca una amplia gama de sistemas de bases de datos que divergen del modelo relacional tradicional. Según Sadalage y Fowler (2012), la aparición de NoSQL respondió a las limitaciones de los RDBMS para escalar horizontalmente y manejar datos no estructurados en aplicaciones web a gran escala.

1.2.7. Características Fundamentales de los Sistemas NoSQL

Las bases de datos NoSQL presentan características distintivas que las diferencian de los sistemas relacionales tradicionales:

a. Consistencia Eventual: A diferencia de los RDBMS que priorizan la consistencia fuerte (propiedad ACID), muchos sistemas NoSQL adoptan el modelo BASE (Basically Available, Soft-state, Eventual consistency). Según Pritchett (2008), este enfoque acepta que la consistencia puede no ser inmediata en sistemas distribuidos, permitiendo mayor disponibilidad y tolerancia a particiones de red.

b. Estructura Distribuida: Emplean mecanismos como tablas de hash distribuidas (DHT) para particionar datos automáticamente entre múltiples nodos. Como explica Decandia et al. (2007) en el contexto de DynamoDB de Amazon, esta arquitectura permite escalabilidad masiva sin puntos únicos de fallo.

c. Escalabilidad Horizontal: La capacidad de aumentar la capacidad del sistema añadiendo más nodos es fundamental en sistemas NoSQL. Según Chang et al. (2008), arquitecturas como Bigtable de Google permiten escalar a miles de nodos manteniendo rendimiento consistente.

d. Arquitecturas Especializadas para Cargas de Trabajo Específicas: A diferencia de los RDBMS que son sistemas de propósito general, las bases de datos NoSQL suelen optimizarse para patrones de acceso específicos. Como señala Leavitt (2010), esta especialización permite mejor rendimiento para casos de uso particulares.

e. Flexibilidad en el Esquema: Los sistemas NoSQL permiten estructuras de datos dinámicas donde diferentes registros pueden tener diferentes atributos. Esta característica, según Moniruzzaman y Hossain (2013), facilita la evolución del esquema sin costosas operaciones de migración.

f. Tolerancia a Fallos y Redundancia: Implementan replicación automática de datos entre nodos para garantizar disponibilidad incluso ante fallos de hardware. Según Lakshman y Malik (2010), sistemas como Cassandra emplean estrategias de replicación configurable para balancear consistencia, disponibilidad y rendimiento.

1.2.8. Clasificación de Bases de Datos NoSQL

Las bases de datos NoSQL se categorizan generalmente en cuatro tipos principales, cada uno optimizado para diferentes patrones de acceso y estructuras de datos:

- Almacenes Clave-Valor: Sistemas como Redis, DynamoDB y Riak que ofrecen alto rendimiento para operaciones simples de lectura/escritura mediante un modelo de datos minimalista (Vogels, 2009).
- Almacenes de Documentos: MongoDB, CouchDB y Firebase Firestore organizan datos en documentos (generalmente JSON o BSON) que pueden contener estructuras anidadas complejas (Chodorow, 2013).
- Bases de Datos de Columnas Anchas: Cassandra, HBase y Bigtable organizan datos por columnas en lugar de filas, optimizando consultas analíticas sobre grandes conjuntos de datos (Lakshman & Malik, 2010).
- Bases de Datos de Grafos: Neo4j, Amazon Neptune y ArangoDB modelan datos como nodos y relaciones, excelentes para datos altamente interconectados (Robinson et al., 2015).

1.2.9. Comparativa RDBMS vs. NoSQL para Sistemas de Datos Ambientales

La elección entre tecnologías relacionales y NoSQL para sistemas de datos ambientales requiere considerar múltiples dimensiones. Como analizan Gessert et al. (2017), ningún paradigma es superior universalmente; cada uno ofrece ventajas en contextos específicos:

Ventajas de RDBMS para Datos Ambientales:

- Integridad y Consistencia: Fundamental para datos científicos donde la precisión es crítica (Abadi, 2009).
- Consultas Complejas: SQL permite expresar consultas analíticas complejas sobre múltiples dimensiones (temporal, espacial, categórica).
- Madurez y Estándares: Décadas de desarrollo y estandarización garantizan estabilidad y disponibilidad de talento (Stonebraker, 2010).
- Soporte para Transacciones: Operaciones ACID garantizan que las actualizaciones de datos relacionados se completen consistentemente.

Ventajas de NoSQL para Datos Ambientales:

- Escalabilidad para Datos Masivos: Adecuado para sistemas de monitoreo que generan volúmenes masivos de datos de sensores (Hecht & Jablonski, 2011).
- Flexibilidad de Esquema: Permite incorporar nuevos tipos de mediciones sin reestructuración costosa (Cattell, 2011).
- Alto Rendimiento de Escritura: Optimizado para ingesta continua de datos de múltiples fuentes (Moniruzzaman & Hossain, 2013).
- Modelos de Datos Especializados: Bases de datos de series temporales (InfluxDB, TimescaleDB) optimizadas específicamente para datos ambientales (Fouquier et al., 2012).

1.2.10. Arquitecturas Híbridas y Políglotas

Reconociendo que ninguna tecnología satisface todos los requisitos, las arquitecturas modernas frecuentemente adoptan enfoques híbridos. Según Sadalage y Fowler (2013), la persistencia políglota implica utilizar múltiples sistemas de bases de datos, cada uno especializado para un aspecto particular de la aplicación.

Para sistemas de datos ambientales, una arquitectura común podría incluir:

- PostgreSQL/PostGIS: Para datos maestros estructurados, relaciones complejas y consultas geoespaciales.

- Base de datos de series temporales: Para almacenamiento eficiente de lecturas periódicas de sensores.
- Almacén de documentos: Para datos semiestructurados como informes, metadatos de sensores y configuraciones.

1.2.11. Consideraciones para la Selección Tecnológica

El contexto específico para esta investigación introduce consideraciones adicionales en la selección tecnológica:

- Recursos Técnicos Disponibles: Como señalan Andrade et al. (2020), en regiones con limitada infraestructura técnica, la simplicidad de implementación y mantenimiento es crucial. PostgreSQL, al ser código abierto y ampliamente documentado, reduce barreras de adopción.
- Escalabilidad Progresiva: El sistema debe permitir crecimiento incremental según aumenten las necesidades, sin requerir rearquitectura completa. La combinación de PostgreSQL (escalable verticalmente) con estrategias de particionamiento permite este crecimiento controlado (Kempiak, 2022).
- Sostenibilidad a Largo Plazo: Los sistemas ambientales requieren mantenimiento continuo por décadas. Tecnologías con amplia comunidad y trayectoria, como PostgreSQL y React, ofrecen mayor garantía de soporte continuo (Nagy, 2023).

1.2.12. Estado del Arte en Sistemas Web para Datos Ambientales

- La revisión de literatura revela tendencias significativas en el desarrollo de sistemas web para gestión de datos ambientales:
- Visualización Interactiva: Sistemas como Kepler.gl (por Uber) y herramientas basadas en D3.js han establecido estándares para visualización geoespacial interactiva en navegadores web (Murray, 2017).
- APIs Estándar: La adopción de estándares como OGC Sensor Observation Service (SOS) y WaterML facilita la interoperabilidad entre sistemas (Bröring et al., 2011).
- Computación en el Lado del Cliente: Frameworks modernos como React permiten procesamiento significativo en el navegador, reduciendo carga en servidores (Marcin & Nabiałek, 2018).

- Autenticación y Control de Acceso: Plataformas como Supabase simplifican la implementación de sistemas granulares de permisos, esenciales para datos ambientales que pueden tener restricciones de acceso (Supabase, 2023).

1.2.13. Contribución Teórica de la Investigación

Esta investigación contribuye al cuerpo teórico existente en múltiples dimensiones:

- Validación de Arquitecturas Modernas en Contextos Específicos: Documenta la aplicabilidad de la combinación React + Supabase/PostgreSQL para sistemas de datos ambientales en regiones con recursos técnicos limitados.
- Patrones de Diseño para Sistemas Ambientales Web: Identifica y formaliza patrones específicos para visualización, gestión y almacenamiento de datos ambientales en aplicaciones web.
- Metodología de Selección Tecnológica Contextualizada: Desarrolla un marco para seleccionar tecnologías de bases de datos considerando no solo aspectos técnicos sino también factores contextuales como disponibilidad de habilidades técnicas y infraestructura local.
- Integración de Paradigmas de Persistencia: Explora estrategias para combinar fortalezas de diferentes paradigmas de bases de datos (relacional, documental, series temporales) en un sistema coherente.

CAPITULO II
METODOLOGÍA

2.1. Enfoque

La presente investigación se desarrolla bajo un enfoque cualitativo, orientado a comprender y resolver una problemática específica mediante la aplicación del conocimiento científico y tecnológico. Según Hernández-Sampieri y Mendoza (2018), el enfoque cualitativo se caracteriza por comprender fenómenos desde la perspectiva de quienes los experimentan, explorando contextos naturales y profundizando en situaciones particulares. En este estudio, dicho enfoque permite analizar en detalle los requerimientos técnicos, funcionales y contextuales para el desarrollo de un Sistema Web de Gestión y Almacenamiento de Datos Ambientales.

2.2. Tipo de investigación

El tipo de investigación corresponde a Investigación Científica de Desarrollo Tecnológico Aplicado. De acuerdo con la Organización para la Cooperación y el Desarrollo Económicos (OCDE, 2015), la investigación aplicada se define como aquella emprendida para determinar los posibles usos de los hallazgos de la investigación básica, o para determinar nuevos métodos o formas de alcanzar objetivos específicos predeterminados. En este caso, se parte de fundamentos teóricos sobre bases de datos relacionales, sistemas NoSQL y desarrollo web moderno, para aplicarlos en la construcción de una solución tecnológica funcional que responda a una necesidad concreta en el ámbito ambiental de Puno.

Asimismo, Espinoza (2020) señala que el desarrollo tecnológico aplicado implica un proceso sistemático de diseño, implementación y validación de soluciones tecnológicas basadas en conocimiento científico preexistente, con el propósito de resolver problemas prácticos. Esta investigación se enmarca en dicha definición al desarrollar un sistema web funcional que integra PostgreSQL (mediante Supabase) y React, respondiendo a requisitos específicos de almacenamiento, seguridad y visualización de datos ambientales.

2.3. Contexto de la Investigación

La investigación se llevó a cabo en el ámbito de la provincia de San Román, región de Puno, Perú, durante los meses de enero a mayo. El contexto geográfico y sociotécnico se caracteriza por:

- Limitada infraestructura tecnológica para la gestión de datos ambientales en instituciones públicas y académicas.
- Carencia de sistemas web especializados para el almacenamiento y visualización de datos ambientales recopilados por diferentes fuentes.
- Presencia de datos ambientales dispersos en formatos heterogéneos y sin estandarización.
- Interés institucional por parte de la Universidad Nacional de Juliaca en desarrollar capacidades tecnológicas aplicadas al monitoreo ambiental.

Como señalan Andrade et al. (2020), los contextos con limitaciones de recursos técnicos requieren aproximaciones metodológicas que prioricen la sostenibilidad, la facilidad de mantenimiento y la transferencia tecnológica. Estos elementos fueron considerados transversalmente en el diseño metodológico del presente estudio.

2.4. Diseño de la Investigación

El estudio adopta un diseño no experimental de tipo descriptivo y proyectivo. Es no experimental porque no se manipulan deliberadamente variables; en su lugar, se observan y analizan fenómenos en su contexto natural para, posteriormente, desarrollar una propuesta tecnológica fundamentada (Hernández-Sampieri y Mendoza, 2018). Es descriptivo porque caracteriza los requisitos, funcionalidades y arquitecturas posibles para sistemas de datos ambientales; y es proyectivo porque culmina en la elaboración de una solución tecnológica concreta: el sistema web desarrollado.

Hurtado de Barrera (2012) define la investigación proyectiva como aquella que propone soluciones a situaciones específicas mediante la creación, diseño o elaboración de propuestas, planes, programas o proyectos. Esta investigación se ajusta a dicha categoría, ya que no solo analiza el problema, sino que desarrolla un artefacto tecnológico que constituye la solución propuesta.

Complementariamente, se incorpora un componente comparativo para la evaluación de sistemas de gestión de bases de datos. Según Yin (2018), los diseños comparativos en estudios cualitativos permiten identificar similitudes, diferencias y fundamentos subyacentes entre casos o fenómenos. En esta investigación, dicho componente se aplica al análisis de PostgreSQL (RDBMS) en dimensiones críticas para datos ambientales: autenticación, control de acceso, confidencialidad, integridad y disponibilidad.

2.5. Población estudiada

En este trabajo de investigación formativa la población está considerado:

- Estudiantes de la Universidad Nacional de Juliaca involucrados en el desarrollo del sistema.
- Datos ambientales recopilados por sensores y bases de datos anteriores.
- La muestra considerada es la base de datos, de acuerdo al interés de estudio para el presente trabajo de investigación formativa.

2.6. Mediciones realizadas

- Las mediciones se realizaron en el marco del desarrollo web para datos ambientales considerando las siguientes características:
- Estudiantes de la Universidad Nacional de Juliaca involucrados en el desarrollo del sistema.
- Datos ambientales recopilados por sensores y fuentes manuales.

2.7. Materiales y procedimientos utilizados

2.7.1. Materiales tecnológicos

- Lenguaje de programación: React.js, Vite, Chakra-UI, Chart.js.
- Base de datos: **PostgreSQL** para datos estructurados.
- Framework frontend: **React.js**.
- Herramientas de despliegue: **AWS Cloud** o plataformas similares

2.7.2. Aplicación web con base de datos relacional MySQL

a. Crear el modelo entidad relación a:

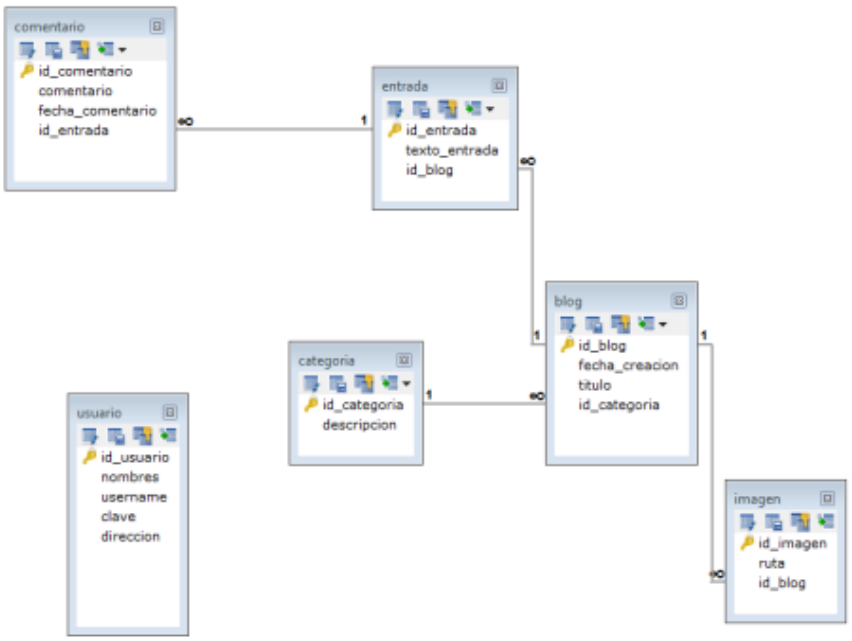

Figura 1. Modelo entidad relación de la investigación

2.8. Procedimiento

- **Análisis de Requisitos:** Se realizó una reunión con los interesados para identificar necesidades específicas del sistema.
- **Diseño del Sistema:** Se diseñó la arquitectura del sistema utilizando herramientas como diagramas de flujo y mockups.
- **Implementación:** Desarrollo del sistema integrando backend, frontend y base de datos.
- **Pruebas:** Evaluación de la funcionalidad, rendimiento y seguridad del sistema.

2.9. Análisis estadístico

No se realizó análisis estadístico, por ser trabajo aplicado.

CAPITULO III
RESULTADOS Y DISCUSIÓN

3.1. Resultados

Para el desarrollo del proyecto se establecieron los principales componentes para Implementar el Sistema Web para la Gestión y Almacenamiento de Datos Ambientales, para lo cual utilizamos la herramienta Impact Mapping utilizado en el contexto del Scrum. Por lo que se obtuvieron como resultados la implementación del Sistema Web para la Gestión y Almacenamiento de Datos Ambientales, aplicando metodología Scrum, además de ello se realizaron una serie de procedimientos que se detalla a continuación.

3.1.1. Determinación de Requerimientos

Para el presente proyecto se establece los requerimientos para el cual se procedió a seleccionarlos en requerimientos de información, funcionales y no funcionales, a los cuales a su vez se procedió a darles niveles de prioridad, con la finalidad de encontrar aquellas historias de usuario que fueran relevantes para formar versiones del sistema Web, todo esto de acuerdo a reuniones con el cliente.

Para desglosar los requerimientos del sistema se analizó con base en los objetivos de Investigación Formativa y visión del proyecto de acuerdo al análisis realizado con el Impact Mapping.

No Funcionales:

- Seguridad: Encriptación de contraseñas, autenticación JWT.
- Rendimiento: Respuesta en <2 segundos para consultas.
- Escalabilidad: Soporte para 10+ sensores simultáneos.

Funcionales

- Diseño y Estructuración de la Base de Datos
- Gestión de Usuarios y Roles
- Gestión de Datos Ambientales
- Visualización y Reportes
- Seguridad y Auditoría
- Infraestructura y Experiencia de Usuario

3.1.2. Desarrollo de épicas

En esta sección se presentan las épicas que orientan la Implementación el Sistema Web para la Gestión y Almacenamiento de Datos Ambientales. Cada épica abarca un conjunto de funcionalidades esenciales para mejorar la eficiencia en la gestión de información ambiental. Estas épicas han sido diseñadas para cubrir los aspectos fundamentales del sistema, desde la administración de la base de datos hasta la seguridad y el acceso a la información.

Tabla 1
Épicas

Código	Épica	Descripción de la Épica
E01	Diseño y Estructuración de la Base de Datos	Modelado y creación de la base de datos para la conexión con el frontend.
E02	Gestión de Usuarios y Roles	Sistema para administrar roles, autenticación y seguridad básica.
E03	Gestión de Datos Ambientales	Captura, almacenamiento y validación de datos desde múltiples fuentes.
E04	Visualización y Reportes	Herramientas interactivas para análisis y reportes ambientales.
E05	Seguridad y Auditoría	Trazabilidad de acciones y prevención de accesos no autorizados
E06	Infraestructura y Experiencia de Usuario	Configuración de entornos, despliegue e interfaz unificada.

3.1.3. Creación del Backlog Priorizado del Producto

El Backlog Priorizado del Producto, es un punto importante para guiar la Implementación el Sistema Web para la Gestión y Almacenamiento de Datos Ambientales.

Aplicando el método de los 100 puntos, los stakeholders clave, incluyendo responsables del proyecto de investigación formativa, han asignado puntajes a cada

épica en función de su importancia y urgencia. Este enfoque garantiza que los esfuerzos de desarrollo se enfoquen primero en las funcionalidades más críticas, mejorando así el impacto del sistema desde las primeras etapas del proyecto.

3.1.4. Historias de usuario

En esta parte se explica la elaboración de las historias de usuario, que han sido cuidadosamente diseñadas para representar las necesidades y metas estratégicas del proyecto de investigación formativa. Estas experiencias de usuario se han orientado según la visión del Product Owner y el equipo de desarrollo, y se encuentran en consonancia con los requerimientos funcionales y operativos del proyecto las experiencias de usuario representan el eje central del desarrollo ágil, dado que conectan directamente las demandas empresariales con las características del sistema, garantizando que cada evolución aporte a los objetivos estratégicos fijados desde el comienzo del proyecto.

Así mismo se considera la escala de MosCow en las historias de usuario para determinar la prioridad en su implementación.

Épica E01: Diseño y Estructuración de la Base de Datos

Tabla 2
Tarjeta de Historia de Usuario H01

TARJETA DE HISTORIA DE USUARIOS	
Código	H01
Nombre de Historia	Diseñar modelo E-R
Prioridad en el negocio	Alta
Importancia de desarrollo	1
Puntos estimados	1 2 3 5 8 13 21
Prioridad MOSCOW	
Tiempo estimado	M
Módulo asignado	E01
Como	Administrador de BD
Quiero	Diseñar un modelo entidad-relación
Para poder	Garantizar integridad y escalabilidad de los datos

Criterios de validación	1. Diagrama ER validado 2. Normalización hasta 3FN

Tabla 3
Tarjeta de Historia de Usuario H02

TARJETA DE HISTORIA DE USUARIOS	
Código	H02
Nombre de Historia	Crear BD en Supabase
Prioridad en el negocio	Alta
Importancia de desarrollo	2
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	
Módulo asignado	E01
Como	DevOps
Quiero	implementar la base de datos en Supabase
Para poder	Almacenar datos de forma segura
Criterios de validación	1. Tablas creadas en PostgreSQL 2. Conexión exitosa con React

Tabla 4
Tarjeta de Historia de Usuario H03

TARJETA DE HISTORIA DE USUARIOS	
Código	H03
Nombre de Historia	Optimizar modelo de BD
Prioridad en el negocio	Media
Importancia de desarrollo	3
Puntos estimados	

Prioridad MOSCOW	**S**
Tiempo estimado	
Módulo asignado	E01
Como	Desarrollador Backend
Quiero	optimizar scripts SQL
Para poder	mejorar el rendimiento de consultas
Criterios de validación	1. Índices aplicados 2. Tiempos de respuesta <500ms.

Tabla 5
Tarjeta de Historia de Usuario H04

TARJETA DE HISTORIA DE USUARIOS	
Código	H04
Nombre de Historia	Integrar BD con frontend
Prioridad en el negocio	Alta
Importancia de desarrollo	4
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	
Módulo asignado	E01
Como	Desarrollador Full Stack
Quiero	conectar React con Supabase
Para poder	mostrar datos en tiempo real
Criterios de validación	1. Componentes consumiendo API 2. Pruebas de carga exitosas

Épica E02: Gestión de Usuarios y Autenticación

Tabla 6
Tarjeta de Historia de Usuario H05

TARJETA DE HISTORIA DE USUARIOS	
Código	H05
Nombre de Historia	Crear usuarios con roles
Prioridad en el negocio	Alta
Importancia de desarrollo	1
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	
Módulo asignado	E02
Como	Super Administrador
Quiero	Asignar roles predefinidos
Para poder	delegar responsabilidades
Criterios de validación	1. Campos obligatorios: email, rol, contraseña 2. Notificación por email

Tabla 7
Tarjeta de Historia de Usuario H06

TARJETA DE HISTORIA DE USUARIOS	
Código	H06
Nombre de Historia	Editar roles de usuarios
Prioridad en el negocio	Media
Importancia de desarrollo	2
Puntos estimados	
Prioridad MOSCOW	S
Tiempo estimado	

Módulo asignado	E02
Como	Administrador
Quiero	modificar permisos
Para poder	ajustar accesos operativos
Criterios de validación	1. Restricción: no editar Super Administrador 2. Actualización instantánea

Tabla 8
Tarjeta de Historia de Usuario H07

TARJETA DE HISTORIA DE USUARIOS	
Código	H07
Nombre de Historia	Recuperar contraseña
Prioridad en el negocio	Alta
Importancia de desarrollo	3
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	
Módulo asignado	E02
Como	Usuario
Quiero	restablecer credenciales
Para poder	recuperar acceso al sistema
Criterios de validación	1. Token JWT válido por 1 hora 2. Enlace enviado al email registrado

Tabla 9
Tarjeta de Historia de Usuario H08

TARJETA DE HISTORIA DE USUARIOS	
Código	H08
Nombre de Historia	Limitar acceso a invitados
Prioridad en el negocio	Media
Importancia de desarrollo	4
Puntos estimados	
Prioridad MOSCOW	S
Tiempo estimado	
Módulo asignado	E02
Como	Invitado
Quiero	ver solo datos públicos
Para poder	navegar sin registro
Criterios de validación	1. Ocultar botones de edición 2. Redirección a registro en acciones restringidas

Épica E03: Gestión de Datos Ambientales

Tabla 10
Tarjeta de Historia de Usuario H09

TARJETA DE HISTORIA DE USUARIOS	
Código	H09
Nombre de Historia	Subir archivos CSV/Excel
Prioridad en el negocio	Alta
Importancia de desarrollo	1
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	

Módulo asignado	E03
Como	Analista
Quiero	cargar datos masivos
Para poder	centralizar información
Criterios de validación	1. Validar formato (columnas: fecha, temperatura, ubicación) 2. Almacenar en PostgreSQL

Tabla 11
Tarjeta de Historia de Usuario H10

TARJETA DE HISTORIA DE USUARIOS	
Código	H10
Nombre de Historia	Registrar datos manualmente
Prioridad en el negocio	Alta
Importancia de desarrollo	2
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	
Módulo asignado	E03
Como	Técnico
Quiero	ingresar datos manuales
Para poder	completar información ante fallos
Criterios de validación	1. Campos obligatorios: fecha, sensor, valor 2. Validación en tiempo real

Tabla 12
Tarjeta de Historia de Usuario H11

TARJETA DE HISTORIA DE USUARIOS	
Código	H11
Nombre de Historia	Integrar sensores externos
Prioridad en el negocio	Media
Importancia de desarrollo	3
Puntos estimados	
Prioridad MOSCOW	S
Tiempo estimado	
Módulo asignado	E03
Como	Administrador
Quiero	conectar APIs de sensores
Para poder	automatizar recolección
Criterios de validación	1. Sincronización cada 1 hora 2. Alertas por fallos de conexión

Épica E04: Visualización y Reportes

Tabla 13
Tarjeta de Historia de Usuario H12

TARJETA DE HISTORIA DE USUARIOS	
Código	H12
Nombre de Historia	Generar gráficos interactivos
Prioridad en el negocio	Alta
Importancia de desarrollo	1
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	
Módulo asignado	E04

Como	Analista
Quiero	visualizar tendencias
Para poder	analizar datos ambientales
Criterios de validación	1. Selección de rangos de fecha 2. Exportación a PNG/PDF con Chart.js

Tabla 14
Tarjeta de Historia de Usuario H13

TARJETA DE HISTORIA DE USUARIOS	
Código	H13
Nombre de Historia	Dashboard unificado
Prioridad en el negocio	2
Importancia de desarrollo	2
Puntos estimados	
Prioridad MOSCOW	C
Tiempo estimado	
Módulo asignado	E04
Como	Usuario
Quiero	ver métricas clave
Para poder	monitorear en tiempo real
Criterios de validación	1. Widgets ajustables 2. Actualización automática cada 5 min

Tabla 15
Tarjeta de Historia de Usuario H14

TARJETA DE HISTORIA DE USUARIOS	
Código	H14
Nombre de Historia	Exportar reportes en PDF/Excel
Prioridad en el negocio	Media
Importancia de desarrollo	3
Puntos estimados	
Prioridad MOSCOW	S
Tiempo estimado	
Módulo asignado	E04
Como	Administrador
Quiero	generar documentos formales
Para poder	compartir análisis
Criterios de validación	1. Filtros por sensor/fecha 2. Logo institucional en PDF

Épica E05: Seguridad y Auditoría

Tabla 16
Tarjeta de Historia de Usuario H15

TARJETA DE HISTORIA DE USUARIOS	
Código	H15
Nombre de Historia	Auditar actividades críticas
Prioridad en el negocio	Alta
Importancia de desarrollo	1
Puntos estimados	
Prioridad MOSCOW	M

Tiempo estimado	
Módulo asignado	E05
Como	Auditor
Quiero	revisar logs
Para poder	garantizar trazabilidad
Criterios de validación	1. Filtrar por usuario/acción 2. Exportar logs a CSV

Tabla 17
Tarjeta de Historia de Usuario H16

TARJETA DE HISTORIA DE USUARIOS	
Código	H16
Nombre de Historia	Encriptar contraseñas
Prioridad en el negocio	Alta
Importancia de desarrollo	2
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	
Módulo asignado	E05
Como	Super Admin
Quiero	proteger credenciales
Para poder	cumplir estándares de seguridad
Criterios de validación	1. Usar bcrypt con salt 2. Validar 8+ caracteres, mayúsculas y números

Tabla 18
Tarjeta de Historia de Usuario H17

TARJETA DE HISTORIA DE USUARIOS	
Código	H17
Nombre de Historia	Alertar actividades sospechosas
Prioridad en el negocio	Baja
Importancia de desarrollo	3
Puntos estimados	
Prioridad MOSCOW	C
Tiempo estimado	
Módulo asignado	E05
Como	Sistema
Quiero	detectar intentos fallidos
Para poder	prevenir accesos no autorizados
Criterios de validación	1. Trigger en PostgreSQL tras 5 intentos 2. Notificación por email al Admin

Épica E06: Infraestructura y Experiencia de Usuario

Tabla 19
Tarjeta de Historia de Usuario H18

TARJETA DE HISTORIA DE USUARIOS	
Código	H18
Nombre de Historia	Desplegar en servidor
Prioridad en el negocio	Alta
Importancia de desarrollo	1
Puntos estimados	
Prioridad MOSCOW	M
Tiempo estimado	

Módulo asignado	E06
Como	DevOps
Quiero	publicar la aplicación
Para poder	permitir acceso global
Criterios de validación	1. URL funcional en Vercel/Netlify 2. Configuración CI/CD

Tabla 20
Tarjeta de Historia de Usuario H19

TARJETA DE HISTORIA DE USUARIOS	
Código	H19
Nombre de Historia	Componentes reutilizables
Prioridad en el negocio	Media
Importancia de desarrollo	2
Puntos estimados	
Prioridad MOSCOW	S
Tiempo estimado	
Módulo asignado	E06
Como	Desarrollador Frontend
Quiero	estandarizar UI
Para poder	acelerar el desarrollo
Criterios de validación	1. Forms con Chakra-UI 2. Documentación en Storybook

Tabla 21
Tarjeta de Historia de Usuario H20

TARJETA DE HISTORIA DE USUARIOS	
Código	H20
Nombre de Historia	Interfaz responsiva
Prioridad en el negocio	Baja
Importancia de desarrollo	3
Puntos estimados	
Prioridad MOSCOW	C
Tiempo estimado	
Módulo asignado	E06
Como	Usuario
Quiero	navegar fácilmente
Para poder	usar el sistema en cualquier dispositivo
Criterios de validación	1. Diseño adaptable a móvil/desktop 2. Gráficos responsivos

3.2. Implementación

Se llevó a cabo actividades clave para la ejecución efectiva del sprint, incluyendo el desarrollo de tareas, la coordinación constante a través de reuniones diarias y el ajuste continuo de prioridades en el product backlog para asegurar la alineación con los objetivos del proyecto.

3.2.1. Desarrollo de tareas del sprint Backlog

Vamos a ilustrar el manejo del proyecto diseño e implementación del sistema de Gestión de datos ambientales través de la herramienta Jira, una plataforma de gestión de proyectos que facilitó la organización y seguimiento de tareas mediante el uso de tableros ágiles. Este enfoque visual permite monitorizar el avance del desarrollo en tiempo real, mostrando claramente las distintas etapas del proceso desde el Product Backlog hasta las tareas completadas.

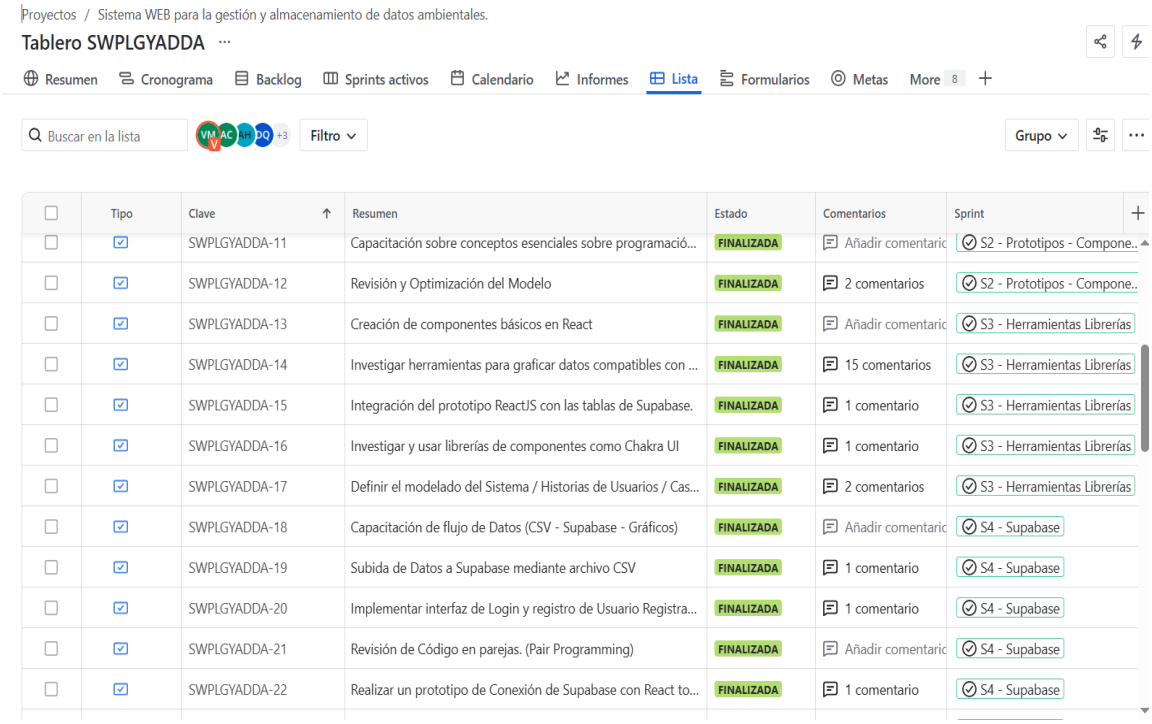

Figura 1. Vista de tablero del trabajo SCRUM

Modelo Entidad Relación

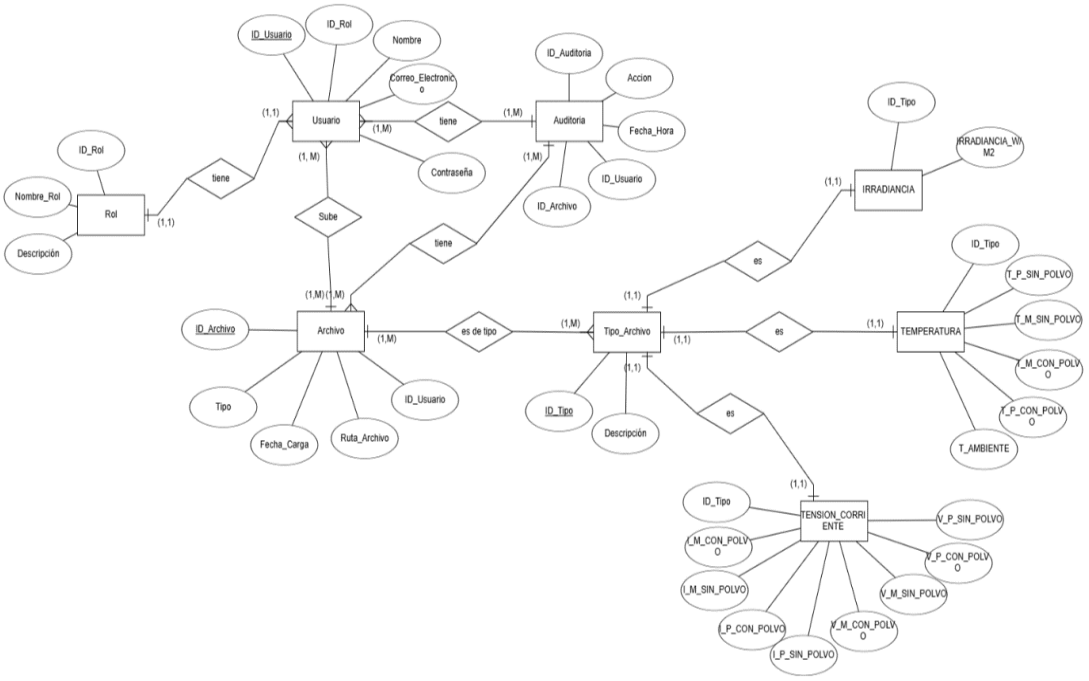

Figura 2. Muestra del diseño de diagrama de Entidad – Relacion en su representación conceptual

Modelo Entidad Relación

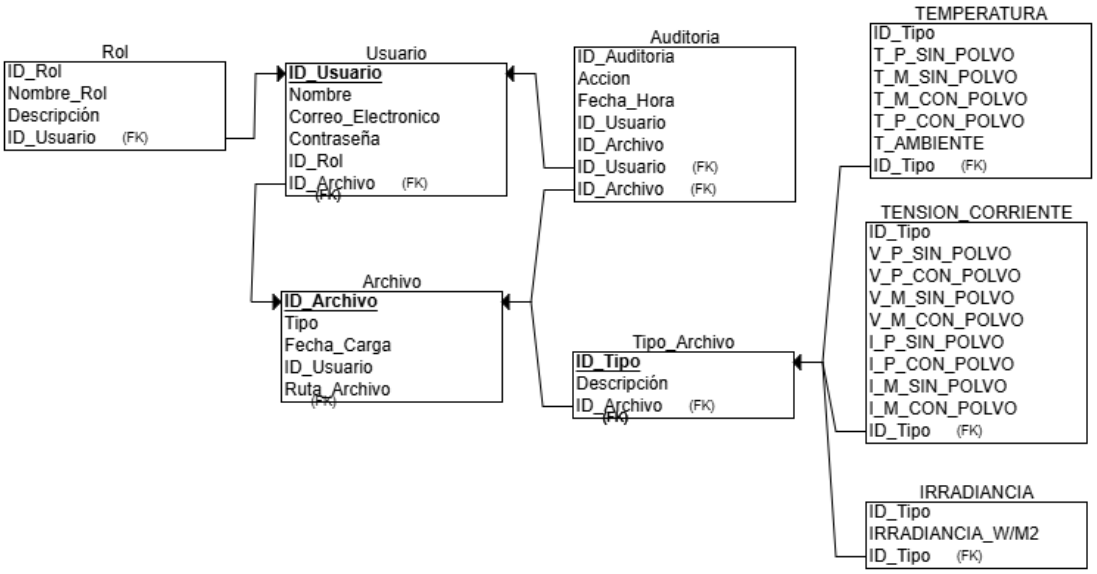

Figura 3. Modelo Entidad- Relacion

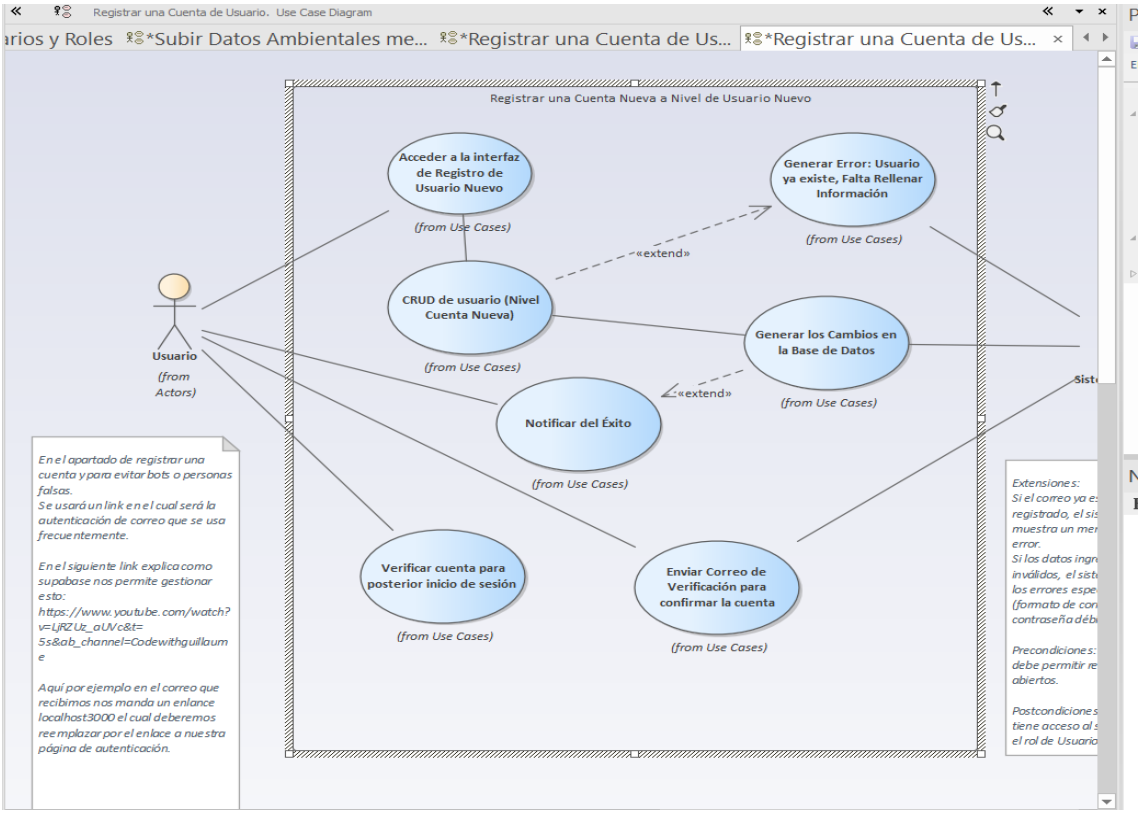

Figura 4. Diagrama de casos de uso

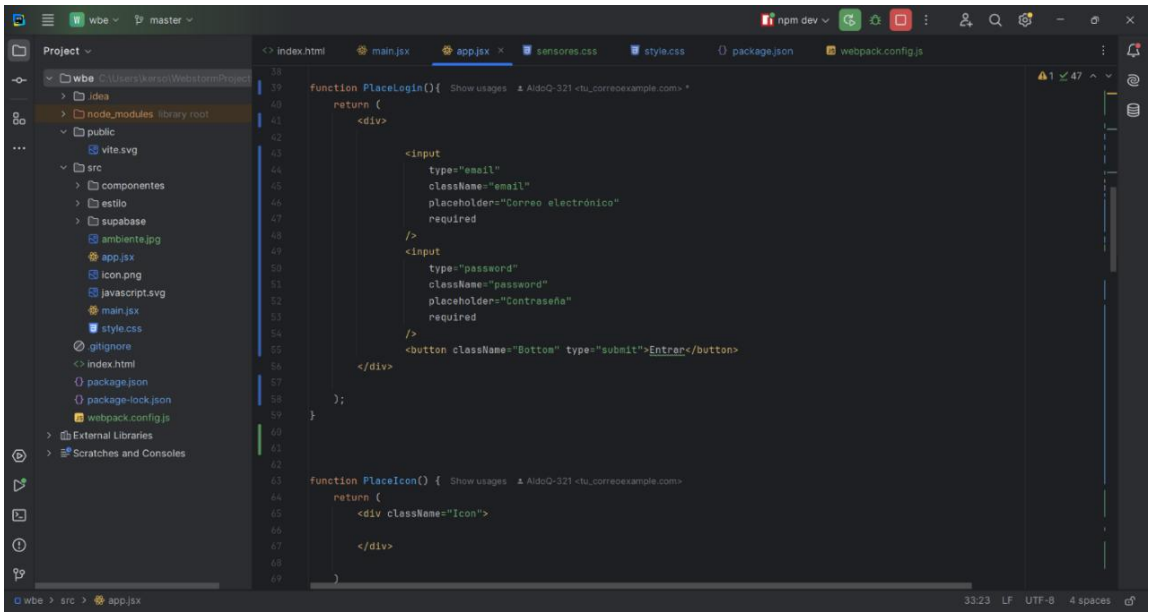

Figura 5. Captura de pantalla de programación backend

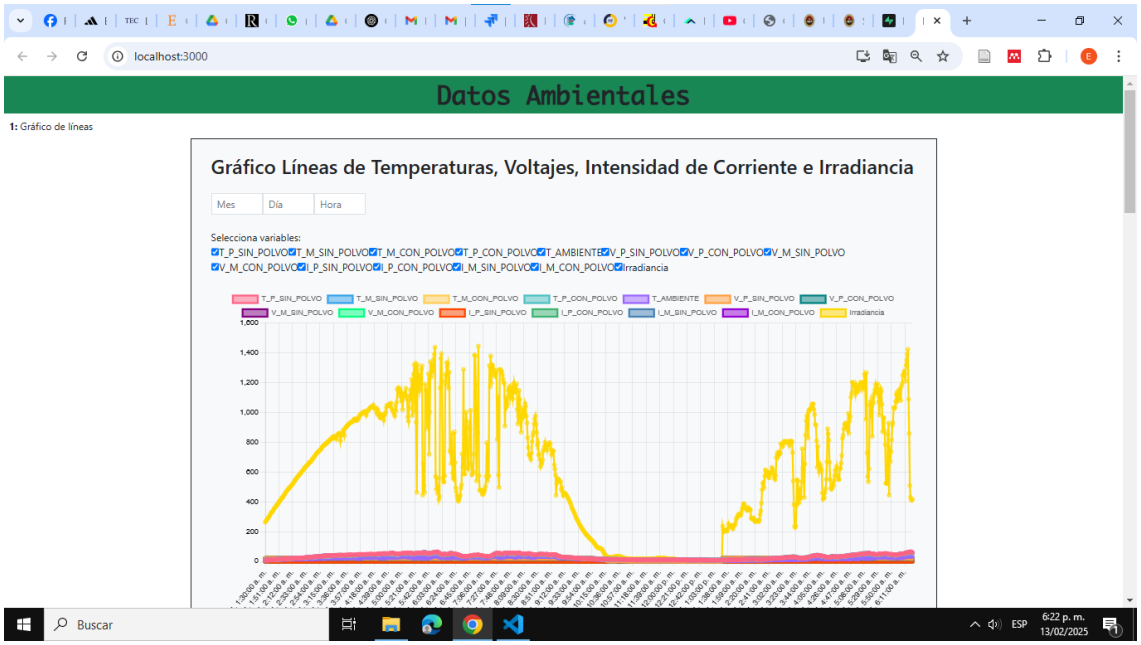

Figura 6.Modelo de vista de gráficos

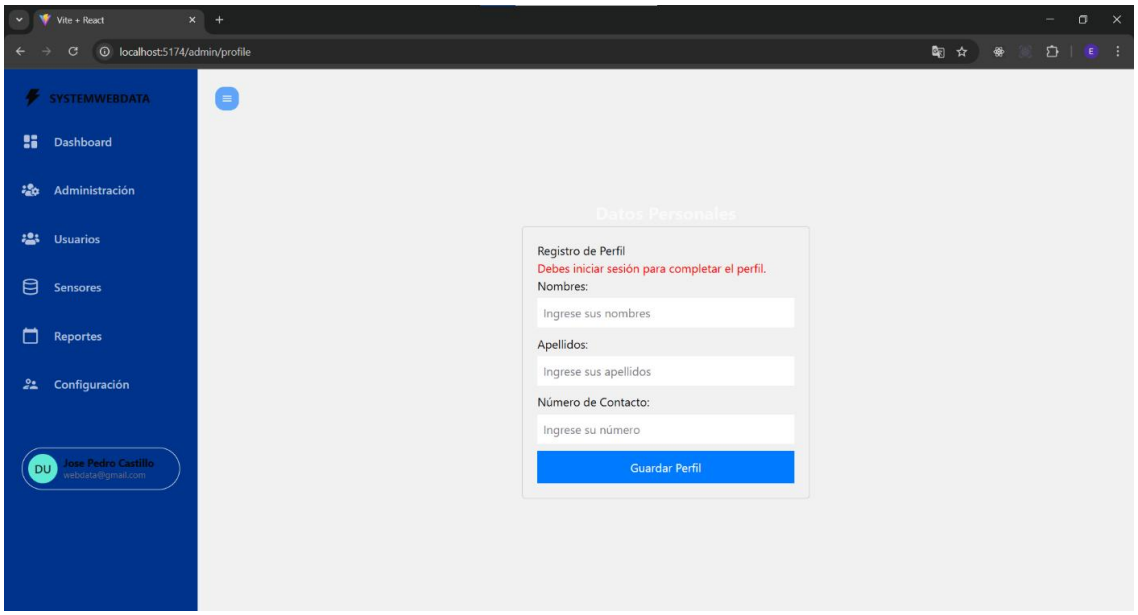

Figura 7. Vista de Dashboard

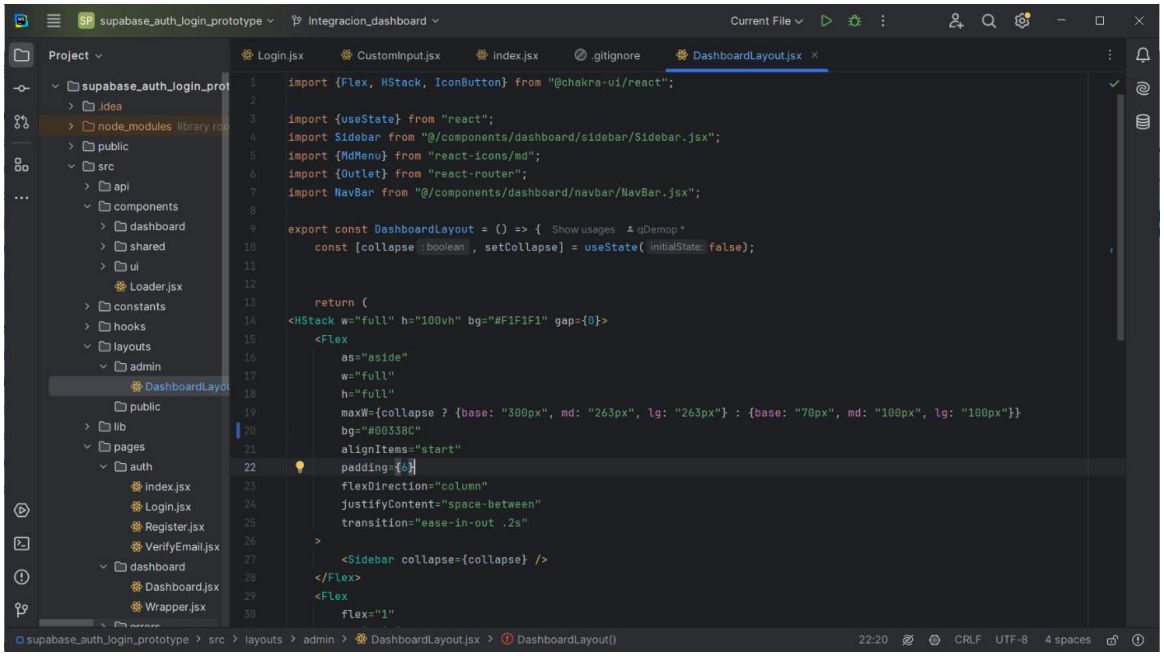

Figura 8. Código de programación de los gráficos a mostrar

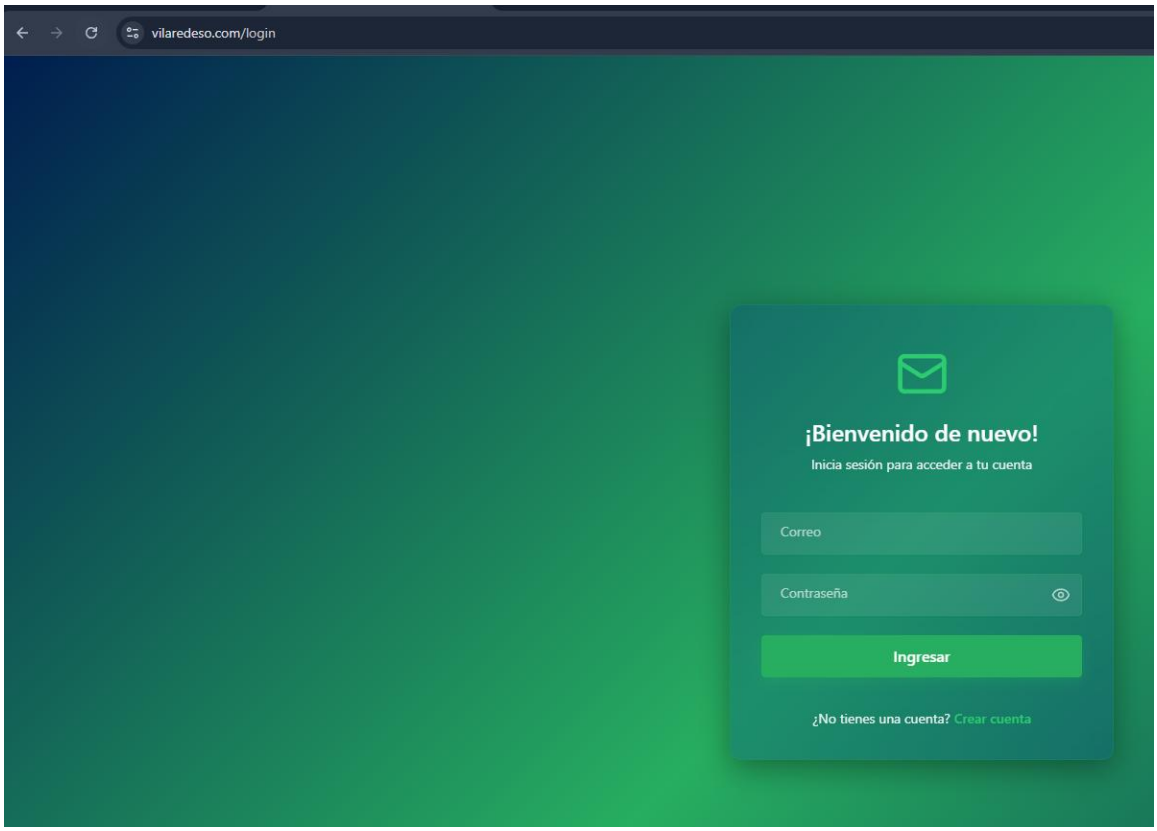

Figura 9. Vista de ingreso a la plataforma

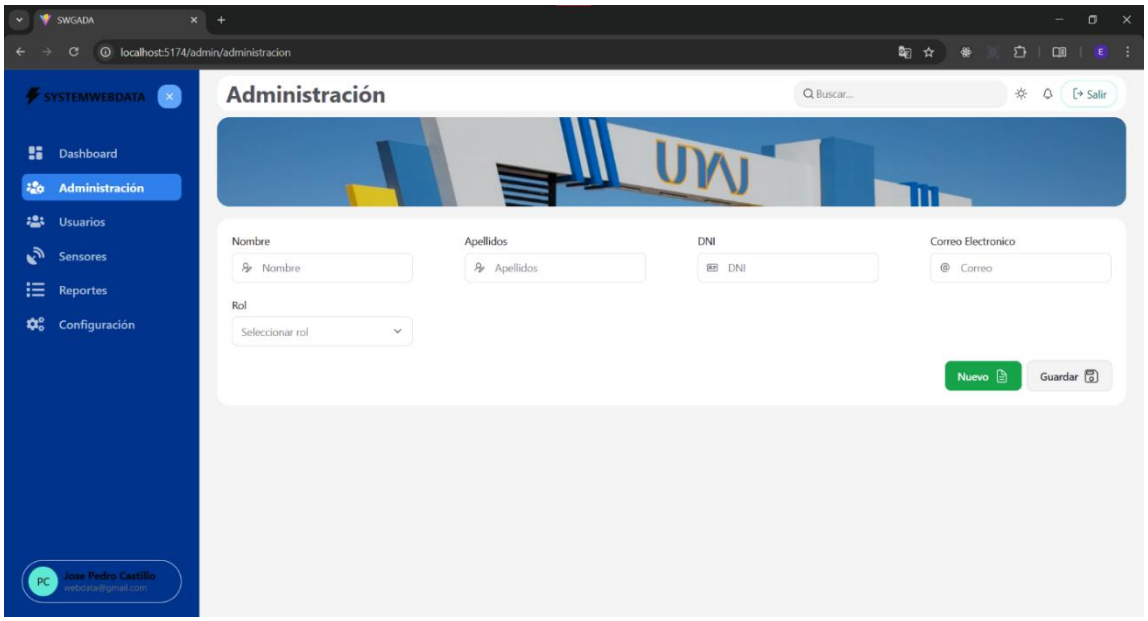

Figura 10. Vista de panel de administrador

3.3. Discusión

En la presente investigación formativa para el desarrollo de un Sistema Web para la Gestión y Almacenamiento de Datos Ambientales, para la implementación del sistema se ha utilizado la herramienta JIRA, el cual permitió el monitoreo y supervisión en tiempo real de la investigación formativa. Los resultados obtenidos sugieren que esta metodología puede ser muy efectiva para gestionar proyectos de investigación formativa.

La presente investigación presenta los hallazgos relevantes que se han obtenido en el marco de la presente investigación, lo cual indica que la aplicación de SCRUM con JIRA en proyectos de investigación formativa mejora significativamente el nivel de productividad, satisfacción del cliente, cumplimiento del presupuesto y cumplimiento del cronograma. Los resultados obtenidos sugieren que esta metodología puede ser muy efectiva para gestionar proyectos de investigación formativa.

La aplicación de la metodología Scrum, permitió una gestión ágil del proyecto. La priorización de requerimientos con MoSCoW aseguró enfocarse en funcionalidades críticas (MUST-HAVE), como la autenticación JWT y la integración con la base de datos. La transparencia en sprints facilitó detectar obstáculos tempranos. La retroalimentación en revisiones de sprint mejoró la usabilidad del dashboard. Por lo tanto los resultados validaron que Scrum optimiza la productividad en entornos dinámicos, aunque requiere compromiso del equipo y stakeholders.

En cuanto al Diseño de base de datos para la Gestión y Almacenamiento de Datos Ambientales, se desarrolló el modelo Entidad Relación, luego se ha pasado al modelo relacional. Todo esto tomando en cuenta la teoría mencionada anteriormente.

CONCLUSIONES

En relación con el análisis de requisitos, la metodología Scrum demostró ser altamente efectiva para gestionar los cambios en los requisitos y mantener entregas incrementales. La aplicación de técnicas como la priorización con MoSCoW y el uso de Jira optimizaron la asignación de recursos y el seguimiento de las tareas, lo que garantizó una clara alineación con las expectativas del cliente y sentó una base sólida para las siguientes etapas.

Respecto al diseño de la arquitectura del sistema, este se realizó en función de los requerimientos analizados, dando como resultado un diseño de base de datos que prioriza el almacenamiento eficiente y seguro de los datos ambientales. Asimismo, la selección de una stack tecnológico moderno, que integró React.js para el frontend y Supabase (PostgreSQL) para el backend, aseguró desde su concepción la escalabilidad y el rendimiento del sistema web.

Sobre la implementación del sistema web, se logró desarrollar con éxito la interfaz de usuario interactiva y toda la funcionalidad del sistema. La implementación práctica confirmó la efectividad de la arquitectura diseñada, culminando en un Sistema Web para la Gestión y Almacenamiento de Datos Ambientales completamente funcional.

Finalmente, el proyecto permitió fortalecer habilidades clave, como la investigación aplicada, el diseño de arquitecturas escalables y la programación full-stack. Un logro destacable fue el dominio práctico de Sistemas de Gestión de Bases de Datos Relacionales, utilizando PostgreSQL en Supabase, una herramienta fundamental para el desarrollo de aplicaciones modernas.

RECOMENDACIONES

Se recomienda institucionalizar un programa de capacitación continua para el equipo de desarrollo, con el fin de mantener su competitividad y capacidad de respuesta ante un panorama tecnológico en constante evolución. Así como, en la profundización de metodologías ágiles avanzadas y herramientas de gestión de proyectos como Jira, para optimizar aún más los flujos de trabajo y la colaboración. Segundo, en la adopción de nuevos frameworks de frontend y mejores prácticas de desarrollo, lo que permitiría enriquecer la experiencia de usuario, mejorar el rendimiento de la aplicación y garantizar la mantenibilidad del código a largo plazo. Esta inversión en conocimiento no solo protege la inversión realizada en el sistema, sino que también constituye un activo invaluable para la organización.

Implementar un protocolo de monitoreo post-implementación. Este proceso debe evaluar el rendimiento del sistema en entornos reales de producción, recopilando métricas clave sobre los tiempos de respuesta, la estabilidad bajo carga y la experiencia de los usuarios finales. Los datos obtenidos de esta evaluación continua serán la fuente más fidedigna para identificar cuellos de botella, priorizar mejoras iterativas y validar que el sistema sigue cumpliendo con los requisitos operativos en un contexto dinámico.

El trabajo realizado abre la puerta a la integración de técnicas de Inteligencia Artificial y Análisis Predictivo sobre los datos ambientales almacenados, lo que permitiría pasar de la simple gestión a la modelación de tendencias de contaminación o la predicción de eventos ambientales. Otra línea de gran relevancia es la investigación e implementación de estrategias de interoperabilidad con otros sistemas de monitorización ambiental, fomentando la creación de una red de datos más comprehensiva y poderosa.

REFERENCIAS BIBLIOGRÁFICAS

Abadi, D. J. (2009). Data management in the cloud: Limitations and opportunities. *IEEE Data Engineering Bulletin*, *32*(1), 3–12.

Andrade, M. C., Silva, J. M. y Ribeiro, C. (2020). Technological adoption in environmental monitoring systems in developing regions. *Journal of Environmental Informatics*, *35*(2), 112–125. https://doi.org/10.3808/jei.202000435

Andrade, M. C., Silva, J. M. y Ribeiro, C. (2020). Technological adoption in environmental monitoring systems in developing regions. *Journal of Environmental Informatics*, *35*(2), 112–125.

Banks, A. y Porcello, E. (2017). *Learning React: Functional web development with React and Redux*. O'Reilly Media.

Bröring, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T., Stasch, C., Liang, S. y Lemmens, R. (2011). New generation sensor web enablement. *Sensors*, *11*(3), 2652–2699. https://doi.org/10.3390/s110302652

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, *39*(4), 12–27. https://doi.org/10.1145/1978915.1978919

Chodorow, K. (2013). *MongoDB: The definitive guide*. O'Reilly Media.

Date, C. J. (2003). *An introduction to database systems* (8.ª ed.). Addison-Wesley.

Decandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. y Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, *41*(6), 205–220. https://doi.org/10.1145/1323293.1294281

Espinoza, E. E. (2020). La investigación formativa en la educación superior. *Revista Conrado*, *16*(77), 47–53.

Facebook Open Source. (2020). *React documentation*. https://reactjs.org/docs/getting-started.html

Fegraus, E. H., Andelman, S., Jones, M. B. y Schildhauer, M. (2011). Maximizing the value of ecological data with structured metadata: An introduction to ecological

metadata language (EML) and principles for metadata creation. *Bulletin of the Ecological Society of America*, *86*(3), 158–168. https://doi.org/10.1890/0012-9623(2005)86[158:MTVOED]2.0.CO;2

Garcia-Molina, H., Ullman, J. D. y Widom, J. (2008). *Database systems: The complete book* (2.ª ed.). Prentice Hall.

Gessert, F., Wingerath, W., Friedrich, S. y Ritter, N. (2017). NoSQL database systems: A survey and decision guidance. *Computer Science and Information Systems*, *14*(1), 1–45. https://doi.org/10.2298/CSIS160315002G

Hecht, R. y Jablonski, S. (2011, diciembre). *NoSQL evaluation: A use case oriented survey* [Ponencia de conferencia]. 2011 International Conference on Cloud and Service Computing, Hong Kong, China. https://doi.org/10.1109/CSC.2011.6138544

Hernández-Sampieri, R. y Mendoza, C. P. (2018). *Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta*. McGraw-Hill.

Hurtado de Barrera, J. (2012). *Metodología de la investigación: Guía para la comprensión holística de la ciencia* (4.ª ed.). Quirón Ediciones.

Kempiak, M. (2022). Scalable data architectures for environmental monitoring systems. *Journal of Data Science and Environmental Informatics*, *18*(3), 45–67.

Lakshman, A. y Malik, P. (2010). Cassandra: A decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, *44*(2), 35–40. https://doi.org/10.1145/1773912.1773922

Leavitt, N. (2010). Will NoSQL databases live up to their promise? *Computer*, *43*(2), 12–14. https://doi.org/10.1109/MC.2010.58

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

Moniruzzaman, A. B. M. y Hossain, S. A. (2013). NoSQL database: New era of databases for big data analytics-classification, characteristics and comparison. *International Journal of Database Theory and Application*, *6*(4), 1–14.

Murray, S. (2017). *Interactive data visualization for the web* (2.ª ed.). O'Reilly Media.

Nagy, J. (2023). Long-term sustainability in environmental data systems. *Environmental Modelling & Software*, *159*, 105567. https://doi.org/10.1016/j.envsoft.2022.105567

Obe, R. O. y Hsu, L. S. (2017). *PostgreSQL: Up and running* (3.ª ed.). O'Reilly Media.

Obe, R. O. y Hsu, L. S. (2020). *PostGIS in action* (3.ª ed.). Manning Publications.

Pressman, R. S. (2014). *Software engineering: A practitioner's approach* (8.ª ed.). McGraw-Hill.

Pritchett, D. (2008). BASE: An ACID alternative. *ACM Queue*, *6*(3), 48–55. https://doi.org/10.1145/1394127.1394128

Roberts, M. (2022). *Serverless architectures on AWS* (2.ª ed.). Manning Publications.

Robinson, I., Webber, J. y Eifrem, E. (2015). *Graph databases* (2.ª ed.). O'Reilly Media.

Sadalage, P. J. y Fowler, M. (2012). *NoSQL distilled: A brief guide to the emerging world of polyglot persistence*. Addison-Wesley.

Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM*, *53*(4), 10–11. https://doi.org/10.1145/1721654.1721659

Stonebraker, M. y Hellerstein, J. M. (2005). What goes around comes around. En *Readings in database systems* (4.ª ed., pp. 2–41). MIT Press.

Supabase. (2023). *Supabase documentation*. https://supabase.com/docs

Talbot, D. (2021). Progressive web apps for environmental monitoring in low-connectivity regions. *International Journal of Environmental Research and Public Health*, *18*(9), 4567. https://doi.org/10.3390/ijerph18094567

Vogels, W. (2009). Eventually consistent. *Communications of the ACM*, *52*(1), 40–44. https://doi.org/10.1145/1435417.1435432

Yin, R. K. (2018). *Case study research and applications: Design and methods* (6.ª ed.). Sage Publications.

ANEXOS

Código de programa

App.jsx

```
import React, { useEffect, useState } from "react";
import BarsChart from "./BarsChart.jsx";
import BarsChart2 from "./BarsChart2.jsx";
import BarsChart3 from "./BarsChart3.jsx";
import TablaChart from "./TablaChart.jsx";

function App() {
    const [isDarkMode, setIsDarkMode] = useState(false);

    useEffect(() => {
        const darkModeMediaQuery = window.matchMedia('(prefers-color-
scheme: dark)');
        setIsDarkMode(darkModeMediaQuery.matches);

        const listener = (e) => setIsDarkMode(e.matches);
        darkModeMediaQuery.addEventListener("change", listener);

        return () => darkModeMediaQuery.removeEventListener("change",
listener);
    }, []);

    const colors = {
        background: isDarkMode ? "#111827" : "#ffffff",
        chartBox: isDarkMode ? "#1f2937" : "#f0f0f0",
        border: isDarkMode ? "#4b5563" : "#cccccc",
        text: isDarkMode ? "#f3f4f6" : "#1f2937",
        titleBg: "#059669",
        titleText: "#ffffff",
        separator: isDarkMode ? "#6b7280" : "#999999",
    };

    const chartBoxStyle = {
        backgroundColor: colors.chartBox,
        margin: "20px auto",
        padding: "20px",
        border: `2px solid ${colors.border}`,
        borderRadius: "10px",
        width: "1200px",
        height: "930px",
        color: colors.text,
        boxSizing: "border-box",
    };

    const tablaBoxStyle = {
        ...chartBoxStyle,
        width: "850px",
        height: "1750px",
    };

    const tituloStyle = {
        backgroundColor: colors.titleBg,
        color: colors.titleText,
        textAlign: "center",
        fontFamily: "monospace",
        fontWeight: "bold",
```

```
        padding: "15px",
        fontSize: "24px",
        borderRadius: "10px",
    };

    const subtituloStyle = {
        marginBottom: "10px",
        fontWeight: "600",
        color: colors.text,
    };

    const separadorStyle = {
        margin: "40px auto",
        border: `1px solid ${colors.separator}`,
        width: "80%",
    };

    const appContainerStyle = {
        padding: "20px",
        backgroundColor: colors.background,
        minHeight: "100vh",
    };

    return (
        <div style={appContainerStyle}>
            <h1 style={tituloStyle}>Visualización de Datos
Ambientales</h1>

            <div style={{ marginTop: "40px" }}>
                <p style={subtituloStyle}>Gráfico de barras: Paneles
solares y ambiente</p>
                <div style={chartBoxStyle}>
                    <BarsChart />
                </div>
            </div>

            <div style={{ marginTop: "40px" }}>
                <p style={subtituloStyle}>Comparación de tensiones</p>
                <div style={chartBoxStyle}>
                    <BarsChart2 />
                </div>
            </div>

            <div style={{ marginTop: "40px" }}>
                <p style={subtituloStyle}>Comparación de
corrientes</p>
                <div style={chartBoxStyle}>
                    <BarsChart3 />
                </div>
            </div>

            <hr style={separadorStyle} />

            <div style={{ marginTop: "40px" }}>
                <p style={subtituloStyle}>Tabla de datos recogidos</p>
                <div style={tablaBoxStyle}>
                    <TablaChart />
                </div>
            </div>
        </div>
    );
```

```
}

export default App;
```

bars.chart

```
import React, { useState, useRef, useEffect } from "react";
import { Bar, Line } from "react-chartjs-2";
import { supabase } from "./supabaseClient.jsx";
import {
    Chart as ChartJS,
    CategoryScale,
    LinearScale,
    BarElement,
    PointElement,
    LineElement,
    Title,
    Tooltip,
    Legend,
} from "chart.js";
import zoomPlugin from "chartjs-plugin-zoom";

ChartJS.register(
    CategoryScale,
    LinearScale,
    BarElement,
    PointElement,
    LineElement,
    Title,
    Tooltip,
    Legend,
    zoomPlugin
);

const VARIABLES = {
    T_P_SIN_POLVO: "Temperatura panel policristalino sin polvo",
    T_M_SIN_POLVO: "Temperatura panel monocristalino sin polvo",
    T_M_CON_POLVO: "Temperatura panel monocristalino con polvo",
    T_P_CON_POLVO: "Temperatura panel policristalino con polvo",
    T_AMBIENTE: "Temperatura ambiente",
};

const generarDiasDelMes = (mes, anio = 2021) => {
    const ultimoDia = new Date(anio, mes, 0).getDate();
    return Array.from({ length: ultimoDia }, (_, i) => i + 1);
};

const TemperaturasChart = () => {
    const [datosFiltrados, setDatosFiltrados] = useState([]);
    const [loading, setLoading] = useState(false);
    const [filtroDia, setFiltroDia] = useState("31"); // Día por
defecto
    const [filtroHora, setFiltroHora] = useState("");
    const [variableSeleccionada, setVariableSeleccionada] =
useState("T_P_SIN_POLVO");
    const [tipoGrafico, setTipoGrafico] = useState("bar");
    const [darkMode, setDarkMode] = useState(false);
    const [mensaje, setMensaje] = useState("");
    const chartRef = useRef(null);
```

```
    useEffect(() => {
        const mediaQuery = window.matchMedia("(prefers-color-scheme:
dark)");
        setDarkMode(mediaQuery.matches);
        const listener = (e) => setDarkMode(e.matches);
        mediaQuery.addEventListener("change", listener);

        // Carga automática del gráfico al montar
        fetchDatosFiltrados();

        return () => mediaQuery.removeEventListener("change",
listener);
    }, []);

    const getGradient = (ctx, chartArea) => {
        const gradient = ctx.createLinearGradient(chartArea.left, 0,
chartArea.right, 0);
        gradient.addColorStop(0, "#C9EF26");
        gradient.addColorStop(0.5, "#00B5BB");
        gradient.addColorStop(1, "#072C51");
        return gradient;
    };

    const fetchDatosFiltrados = async () => {
        setLoading(true);
        setMensaje("");
        setDatosFiltrados([]);

        const year = 2021;
        let desde = new Date(`${year}-12-01T00:00:00`);
        let hasta = new Date(year, 12, 31, 23, 59, 59);

        if (filtroDia) {
            desde.setDate(parseInt(filtroDia));
            hasta = new Date(desde);
            hasta.setHours(23, 59, 59);
        }

        if (filtroHora) {
            desde.setHours(parseInt(filtroHora), 0, 0);
            hasta = new Date(desde);
            hasta.setHours(parseInt(filtroHora), 59, 59);
        }

        let allData = [];
        let page = 0;
        const pageSize = 1000;
        let finished = false;

        while (!finished) {
            const { data, error } = await supabase
                .from("Datos_fijo")
                .select(`Fecha, ${variableSeleccionada}`)
                .gte("Fecha", desde.toISOString())
                .lte("Fecha", hasta.toISOString())
                .order("Fecha", { ascending: true })
                .range(page * pageSize, (page + 1) * pageSize - 1);

            if (error) {
                console.error("Error al obtener datos:", error);
                setMensaje("Error al consultar la base de datos.");
```

```
                break;
            }

            if (data.length === 0) {
                finished = true;
            } else {
                allData = [...allData, ...data];
                page++;
                if (data.length < pageSize) finished = true;
            }
        }

        if (allData.length === 0) {
            setMensaje("No se encontraron datos.");
        }

        setDatosFiltrados(allData);
        setLoading(false);
    };

    const limpiarFiltros = () => {
        setFiltroDia("31");
        setFiltroHora("");
        setDatosFiltrados([]);
        setMensaje("");
        fetchDatosFiltrados();
    };

    const labels = datosFiltrados.map((d) => new
Date(d.Fecha).toLocaleString());
    const dataValues = datosFiltrados.map((d) =>
d[variableSeleccionada]);

    const data = {
        labels,
        datasets: [
            {
                label: VARIABLES[variableSeleccionada],
                data: dataValues,
                fill: tipoGrafico === "line",
                backgroundColor: (context) => {
                    const { chart } = context;
                    if (!chart.chartArea) return null;
                    return getGradient(chart.ctx, chart.chartArea);
                },
                borderColor: (context) => {
                    const { chart } = context;
                    if (!chart.chartArea) return null;
                    return getGradient(chart.ctx, chart.chartArea);
                },
                tension: 0.3,
                pointBackgroundColor: darkMode ? "#90caf9" :
"#2e7d32",
            },
        ],
    };

    const options = {
        responsive: true,
        devicePixelRatio: 2,
        plugins: {
```

```
            legend: {
                labels: {
                    color: darkMode ? "#ffffff" : "#000000",
                },
            },
            tooltip: {
                backgroundColor: darkMode ? "#333" : "#fff",
                titleColor: darkMode ? "#fff" : "#000",
                bodyColor: darkMode ? "#ccc" : "#333",
            },
            zoom: {
                pan: { enabled: true, mode: "x" },
                zoom: {
                    wheel: { enabled: true },
                    pinch: { enabled: true },
                    mode: "x",
                },
            },
        },
        scales: {
            y: {
                ticks: { color: darkMode ? "#ffffff" : "#000000" },
                grid: { color: darkMode ? "#444" : "#ccc" },
            },
            x: {
                ticks: { color: darkMode ? "#ffffff" : "#000000" },
                grid: { color: darkMode ? "#444" : "#ccc" },
            },
        },
    };

    const ChartComponent = tipoGrafico === "bar" ? Bar : Line;

    const baseStyle = {
        padding: "16px",
        backgroundColor: darkMode ? "#1a1a1a" : "#ffffff",
        color: darkMode ? "#ffffff" : "#000000",
        minHeight: "100vh",
        fontFamily: "sans-serif",
    };

    const selectStyle = {
        border: "1px solid #ccc",
        padding: "5px 10px",
        marginRight: "10px",
    };

    const buttonStyle = {
        padding: "5px 12px",
        marginRight: "10px",
        border: "none",
        borderRadius: "4px",
        cursor: "pointer",
    };

    return (
        <div style={baseStyle}>
            <h2 style={{ fontSize: "20px", fontWeight: "bold",
marginBottom: "16px" }}>
                Gráfico de temperaturas de paneles con y sin polvo
(Diciembre)
```

```
            </h2>

            <div style={{ marginBottom: "16px", display: "flex",
flexWrap: "wrap", gap: "10px" }}>
                <select style={selectStyle} value={filtroDia}
onChange={(e) => setFiltroDia(e.target.value)}>
                    <option value="">Día</option>
                    {generarDiasDelMes(12).map((d) => (
                        <option key={d} value={d}>{d}</option>
                    ))}
                </select>

                <select style={selectStyle} value={filtroHora}
onChange={(e) => setFiltroHora(e.target.value)}>
                    <option value="">Hora</option>
                    {Array.from({ length: 24 }, (_, h) => (
                        <option key={h} value={h}>{`${h}:00`}</option>
                    ))}
                </select>

                <select style={selectStyle}
value={variableSeleccionada} onChange={(e) =>
setVariableSeleccionada(e.target.value)}>
                    {Object.keys(VARIABLES).map((key) => (
                        <option key={key}
value={key}>{VARIABLES[key]}</option>
                    ))}
                </select>

                <select style={selectStyle} value={tipoGrafico}
onChange={(e) => setTipoGrafico(e.target.value)}>
                    <option value="bar">Barras</option>
                    <option value="line">Líneas</option>
                </select>

                <button onClick={fetchDatosFiltrados} style={{
...buttonStyle, backgroundColor: "#007bff", color: "#fff" }}>
                    Mostrar gráfico
                </button>

                <button onClick={limpiarFiltros} style={{
...buttonStyle, backgroundColor: "#6c757d", color: "#fff" }}>
                    Limpiar filtros
                </button>
            </div>

            {loading && <p>Cargando datos...</p>}
            {mensaje && <p style={{ color: "red" }}>{mensaje}</p>}
            {!loading && datosFiltrados.length > 0 && (
                <ChartComponent ref={chartRef} data={data}
options={options} />
            )}
        </div>
    );
};

export default TemperaturasChart;
```

```
import React, { useState, useRef, useEffect } from "react";
import { Bar, Line } from "react-chartjs-2";
import { supabase } from "./supabaseClient.jsx";
import {
    Chart as ChartJS,
    CategoryScale,
    LinearScale,
    BarElement,
    PointElement,
    LineElement,
    Title,
    Tooltip,
    Legend,
} from "chart.js";
import zoomPlugin from "chartjs-plugin-zoom";

ChartJS.register(
    CategoryScale,
    LinearScale,
    BarElement,
    PointElement,
    LineElement,
    Title,
    Tooltip,
    Legend,
    zoomPlugin
);

const VARIABLES = {
    V_P_SIN_POLVO: "Tensión panel policristalino sin polvo",
    V_P_CON_POLVO: "Tensión panel policristalino con polvo",
    V_M_SIN_POLVO: "Tensión panel monocristalino sin polvo",
    V_M_CON_POLVO: "Tensión panel monocristalino con polvo",
    I_P_SIN_POLVO: "Corriente panel policristalino sin polvo",
    I_P_CON_POLVO: "Corriente panel policristalino con polvo",
    I_M_SIN_POLVO: "Corriente panel monocristalino sin polvo",
    I_M_CON_POLVO: "Corriente panel monocristalino con polvo",
};

const generarDiasDelMes = (mes, anio = 2021) => {
    const ultimoDia = new Date(anio, mes, 0).getDate();
    return Array.from({ length: ultimoDia }, (_, i) => i + 1);
};

const TemperaturasChart = () => {
    const [datosFiltrados, setDatosFiltrados] = useState([]);
    const [loading, setLoading] = useState(false);
    const [filtroDia, setFiltroDia] = useState("31"); // Día por
defecto
    const [filtroHora, setFiltroHora] = useState("");
    const [variableSeleccionada, setVariableSeleccionada] =
useState("V_P_SIN_POLVO");
    const [tipoGrafico, setTipoGrafico] = useState("bar");
    const [darkMode, setDarkMode] = useState(false);
    const [mensaje, setMensaje] = useState("");
    const chartRef = useRef(null);

    useEffect(() => {
        const mediaQuery = window.matchMedia("(prefers-color-scheme:
dark)");
        setDarkMode(mediaQuery.matches);
```

```
        const listener = (e) => setDarkMode(e.matches);
        mediaQuery.addEventListener("change", listener);

        // Mostrar por defecto datos del 31 de diciembre
        fetchDatosFiltrados("31");

        return () => mediaQuery.removeEventListener("change",
listener);
    }, []);

    const getGradient = (ctx, chartArea) => {
        const gradient = ctx.createLinearGradient(chartArea.left, 0,
chartArea.right, 0);
        gradient.addColorStop(0, "#C9EF26");
        gradient.addColorStop(0.5, "#00B5BB");
        gradient.addColorStop(1, "#072C51");
        return gradient;
    };

    const fetchDatosFiltrados = async (diaDesdeEfecto = null) => {
        setLoading(true);
        setMensaje("");
        setDatosFiltrados([]);

        const year = 2021;
        const mes = 12;
        const dia = diaDesdeEfecto || filtroDia;

        let desde = new Date(`${year}-${String(mes).padStart(2, "0")}-
01T00:00:00`);
        let hasta = new Date(year, mes, 31, 23, 59, 59);

        if (dia) {
            desde.setDate(parseInt(dia));
            hasta = new Date(desde);
            hasta.setHours(23, 59, 59);
        }

        if (filtroHora) {
            desde.setHours(parseInt(filtroHora), 0, 0);
            hasta = new Date(desde);
            hasta.setHours(parseInt(filtroHora), 59, 59);
        }

        let allData = [];
        let page = 0;
        const pageSize = 1000;
        let finished = false;

        while (!finished) {
            const { data, error } = await supabase
                .from("Datos_fijo")
                .select(`Fecha, ${variableSeleccionada}`)
                .gte("Fecha", desde.toISOString())
                .lte("Fecha", hasta.toISOString())
                .order("Fecha", { ascending: true })
                .range(page * pageSize, (page + 1) * pageSize - 1);

            if (error) {
                console.error("Error al obtener datos:", error);
                setMensaje("Error al consultar la base de datos.");
```

```
                break;
            }

            if (data.length === 0) {
                finished = true;
            } else {
                allData = [...allData, ...data];
                page++;
                if (data.length < pageSize) finished = true;
            }
        }

        if (allData.length === 0) {
            setMensaje("No se encontraron datos.");
        }

        setDatosFiltrados(allData);
        setLoading(false);
    };

    const limpiarFiltros = () => {
        setFiltroDia("31");
        setFiltroHora("");
        setDatosFiltrados([]);
        setMensaje("");
        fetchDatosFiltrados("31");
    };

    const labels = datosFiltrados.map((d) => new
Date(d.Fecha).toLocaleString());
    const dataValues = datosFiltrados.map((d) =>
d[variableSeleccionada]);

    const data = {
        labels,
        datasets: [
            {
                label: VARIABLES[variableSeleccionada],
                data: dataValues,
                fill: tipoGrafico === "line",
                backgroundColor: (context) => {
                    const { chart } = context;
                    if (!chart.chartArea) return null;
                    return getGradient(chart.ctx, chart.chartArea);
                },
                borderColor: (context) => {
                    const { chart } = context;
                    if (!chart.chartArea) return null;
                    return getGradient(chart.ctx, chart.chartArea);
                },
                tension: 0.3,
                pointBackgroundColor: darkMode ? "#90caf9" :
"#2e7d32",
            },
        ],
    };

    const options = {
        responsive: true,
        devicePixelRatio: 2,
        plugins: {
```

```
            legend: {
                labels: {
                    color: darkMode ? "#ffffff" : "#000000",
                },
            },
            tooltip: {
                backgroundColor: darkMode ? "#333" : "#fff",
                titleColor: darkMode ? "#fff" : "#000",
                bodyColor: darkMode ? "#ccc" : "#333",
            },
            zoom: {
                pan: { enabled: true, mode: "x" },
                zoom: {
                    wheel: { enabled: true },
                    pinch: { enabled: true },
                    mode: "x",
                },
            },
        },
        scales: {
            y: {
                ticks: { color: darkMode ? "#ffffff" : "#000000" },
                grid: { color: darkMode ? "#444" : "#ccc" },
            },
            x: {
                ticks: { color: darkMode ? "#ffffff" : "#000000" },
                grid: { color: darkMode ? "#444" : "#ccc" },
            },
        },
    };

    const ChartComponent = tipoGrafico === "bar" ? Bar : Line;

    const baseStyle = {
        padding: "16px",
        backgroundColor: darkMode ? "#1a1a1a" : "#ffffff",
        color: darkMode ? "#ffffff" : "#000000",
        minHeight: "100vh",
        fontFamily: "sans-serif",
    };

    const selectStyle = {
        border: "1px solid #ccc",
        padding: "5px 10px",
        marginRight: "10px",
    };

    const buttonStyle = {
        padding: "5px 12px",
        marginRight: "10px",
        border: "none",
        borderRadius: "4px",
        cursor: "pointer",
    };

    return (
        <div style={baseStyle}>
            <h2 style={{ fontSize: "20px", fontWeight: "bold",
marginBottom: "16px" }}>
                Gráfico de tensión y corriente en paneles con y sin
polvo (Diciembre)
```

```
            </h2>

            <div style={{ marginBottom: "16px", display: "flex",
flexWrap: "wrap", gap: "10px" }}>
                <select style={selectStyle} value={filtroDia}
onChange={(e) => setFiltroDia(e.target.value)}>
                    <option value="">Día</option>
                    {generarDiasDelMes(12).map((d) => (
                        <option key={d} value={d}>{d}</option>
                    ))}
                </select>

                <select style={selectStyle} value={filtroHora}
onChange={(e) => setFiltroHora(e.target.value)}>
                    <option value="">Hora</option>
                    {Array.from({ length: 24 }, (_, h) => (
                        <option key={h} value={h}>{`${h}:00`}</option>
                    ))}
                </select>

                <select style={selectStyle}
value={variableSeleccionada} onChange={(e) =>
setVariableSeleccionada(e.target.value)}>
                    {Object.keys(VARIABLES).map((key) => (
                        <option key={key}
value={key}>{VARIABLES[key]}</option>
                    ))}
                </select>

                <select style={selectStyle} value={tipoGrafico}
onChange={(e) => setTipoGrafico(e.target.value)}>
                    <option value="bar">Barras</option>
                    <option value="line">Líneas</option>
                </select>

                <button onClick={() => fetchDatosFiltrados()} style={{
...buttonStyle, backgroundColor: "#007bff", color: "#fff" }}>
                    Mostrar gráfico
                </button>

                <button onClick={limpiarFiltros} style={{
...buttonStyle, backgroundColor: "#6c757d", color: "#fff" }}>
                    Limpiar filtros
                </button>
            </div>

            {loading && <p>Cargando datos...</p>}
            {mensaje && <p style={{ color: "red" }}>{mensaje}</p>}
            {!loading && datosFiltrados.length > 0 && (
                <ChartComponent ref={chartRef} data={data}
options={options} />
            )}
        </div>
    );
};

export default TemperaturasChart;
```

```
import React, { useState, useRef, useEffect } from "react";
import { Bar, Line } from "react-chartjs-2";
import { supabase } from "./supabaseClient.jsx";
import {
    Chart as ChartJS,
    CategoryScale,
    LinearScale,
    BarElement,
    PointElement,
    LineElement,
    Title,
    Tooltip,
    Legend,
} from "chart.js";
import zoomPlugin from "chartjs-plugin-zoom";

ChartJS.register(
    CategoryScale,
    LinearScale,
    BarElement,
    PointElement,
    LineElement,
    Title,
    Tooltip,
    Legend,
    zoomPlugin
);

const generarDiasDelMes = (mes, anio = 2021) => {
    if (!mes) return [];
    const ultimoDia = new Date(anio, mes, 0).getDate();
    return Array.from({ length: ultimoDia }, (_, i) => i + 1);
};

const TemperaturasChart = () => {
    const [datosFiltrados, setDatosFiltrados] = useState([]);
    const [loading, setLoading] = useState(false);
    const [filtroDia, setFiltroDia] = useState("31"); // Día por
defecto
    const [filtroHora, setFiltroHora] = useState("");
    const [tipoGrafico, setTipoGrafico] = useState("bar");
    const [darkMode, setDarkMode] = useState(false);
    const [mensaje, setMensaje] = useState("");
    const chartRef = useRef(null);

    const filtroMes = "12"; // diciembre fijo
    const variableSeleccionada = "Irradiancia";

    // Modo oscuro y cargar datos por defecto al montar
    useEffect(() => {
        const mediaQuery = window.matchMedia("(prefers-color-scheme:
dark)");
        setDarkMode(mediaQuery.matches);
        const listener = (e) => setDarkMode(e.matches);
        mediaQuery.addEventListener("change", listener);

        // Cargar datos del 31 por defecto al iniciar
        fetchDatosFiltrados("31");

        return () => mediaQuery.removeEventListener("change",
listener);
```

```
    }, []);

    const getGradient = (ctx, chartArea) => {
        const gradient = ctx.createLinearGradient(chartArea.left, 0,
chartArea.right, 0);
        gradient.addColorStop(0, "#C9EF26");
        gradient.addColorStop(0.5, "#00B5BB");
        gradient.addColorStop(1, "#072C51");
        return gradient;
    };

    const fetchDatosFiltrados = async (diaDesdeUseEffect = null) => {
        setLoading(true);
        setMensaje("");
        setDatosFiltrados([]);

        const year = 2021;
        let dia = diaDesdeUseEffect || filtroDia;
        let desde = new Date(`${year}-12-${String(dia).padStart(2,
"0")}T00:00:00`);
        let hasta = new Date(`${year}-12-${String(dia).padStart(2,
"0")}T23:59:59`);

        if (filtroHora) {
            desde.setHours(parseInt(filtroHora), 0, 0);
            hasta = new Date(desde);
            hasta.setHours(parseInt(filtroHora), 59, 59);
        }

        let allData = [];
        let page = 0;
        const pageSize = 1000;
        let finished = false;

        while (!finished) {
            const { data, error } = await supabase
                .from("Datos_fijo")
                .select(`Fecha, ${variableSeleccionada}`)
                .gte("Fecha", desde.toISOString())
                .lte("Fecha", hasta.toISOString())
                .order("Fecha", { ascending: true })
                .range(page * pageSize, (page + 1) * pageSize - 1);

            if (error) {
                console.error("Error al obtener datos:", error);
                setMensaje("Error al consultar la base de datos.");
                break;
            }

            if (data.length === 0) {
                finished = true;
            } else {
                allData = [...allData, ...data];
                page++;
                if (data.length < pageSize) finished = true;
            }
        }

        if (allData.length === 0) {
            setMensaje("No se encontraron datos.");
        }
```

```
        setDatosFiltrados(allData);
        setLoading(false);
    };

    const limpiarFiltros = () => {
        setFiltroDia("31");
        setFiltroHora("");
        setDatosFiltrados([]);
        setMensaje("");
        fetchDatosFiltrados("31");
    };

    const labels = datosFiltrados.map((d) => new
Date(d.Fecha).toLocaleString());
    const dataValues = datosFiltrados.map((d) =>
d[variableSeleccionada]);

    const data = {
        labels,
        datasets: [
            {
                label: "Irradiancia",
                data: dataValues,
                fill: tipoGrafico === "line",
                backgroundColor: (context) => {
                    const { chart } = context;
                    if (!chart.chartArea) return null;
                    return getGradient(chart.ctx, chart.chartArea);
                },
                borderColor: (context) => {
                    const { chart } = context;
                    if (!chart.chartArea) return null;
                    return getGradient(chart.ctx, chart.chartArea);
                },
                tension: 0.3,
                pointBackgroundColor: darkMode ? "#90caf9" :
"#2e7d32",
            },
        ],
    };

    const options = {
        responsive: true,
        devicePixelRatio: 2,
        plugins: {
            legend: {
                labels: {
                    color: darkMode ? "#ffffff" : "#000000",
                },
            },
            tooltip: {
                backgroundColor: darkMode ? "#333" : "#fff",
                titleColor: darkMode ? "#fff" : "#000",
                bodyColor: darkMode ? "#ccc" : "#333",
            },
            zoom: {
                pan: { enabled: true, mode: "x" },
                zoom: {
                    wheel: { enabled: true },
                    pinch: { enabled: true },
```

```
                    mode: "x",
                },
            },
        },
        scales: {
            y: {
                ticks: { color: darkMode ? "#ffffff" : "#000000" },
                grid: { color: darkMode ? "#444" : "#ccc" },
            },
            x: {
                ticks: { color: darkMode ? "#ffffff" : "#000000" },
                grid: { color: darkMode ? "#444" : "#ccc" },
            },
        },
    };

    const ChartComponent = tipoGrafico === "bar" ? Bar : Line;

    const baseStyle = {
        padding: "16px",
        backgroundColor: darkMode ? "#1a1a1a" : "#ffffff",
        color: darkMode ? "#ffffff" : "#000000",
        minHeight: "100vh",
        fontFamily: "sans-serif",
    };

    const selectStyle = {
        border: "1px solid #ccc",
        padding: "5px 10px",
        marginRight: "10px",
    };

    const buttonStyle = {
        padding: "5px 12px",
        marginRight: "10px",
        border: "none",
        borderRadius: "4px",
        cursor: "pointer",
    };

    return (
        <div style={baseStyle}>
            <h2 style={{ fontSize: "20px", fontWeight: "bold",
marginBottom: "16px" }}>
                Gráfico de Irradiancia - Diciembre
            </h2>

            <div style={{ marginBottom: "16px", display: "flex",
flexWrap: "wrap", gap: "10px" }}>
                <select style={selectStyle} value={filtroDia}
onChange={(e) => setFiltroDia(e.target.value)}>
                    <option value="">Día</option>
                    {generarDiasDelMes(12).map((d) => (
                        <option key={d} value={d}>{d}</option>
                    ))}
                </select>

                <select style={selectStyle} value={filtroHora}
onChange={(e) => setFiltroHora(e.target.value)}>
                    <option value="">Hora</option>
                    {Array.from({ length: 24 }, (_, h) => (
```

```
                        <option key={h} value={h}>{`${h}:00`}</option>
                    ))}
                </select>

                <select style={selectStyle} disabled>
                    <option value="Irradiancia">Irradiancia</option>
                </select>

                <select style={selectStyle} value={tipoGrafico}
onChange={(e) => setTipoGrafico(e.target.value)}>
                    <option value="bar">Barras</option>
                    <option value="line">Líneas</option>
                </select>

                <button onClick={() => fetchDatosFiltrados()} style={{
...buttonStyle, backgroundColor: "#007bff", color: "#fff" }}>
                    Mostrar gráfico
                </button>

                <button onClick={limpiarFiltros} style={{
...buttonStyle, backgroundColor: "#6c757d", color: "#fff" }}>
                    Limpiar filtros
                </button>
            </div>

            {loading && <p>Cargando datos...</p>}
            {mensaje && <p style={{ color: "red" }}>{mensaje}</p>}
            {!loading && datosFiltrados.length > 0 && (
                <ChartComponent ref={chartRef} data={data}
options={options} />
            )}
        </div>
    );
};

export default TemperaturasChart;
```