Pontificia Universidad Javeriana Cali
Facultad de Ingeniería.
Maestría en Ingeniería de Software
Proyecto de Grado.

Diseño de una Arquitectura de Software en la Nube Mantenible y Escalable para Optimizar la Siembra de Aguacate Hass en la Región Andina de Colombia

EDWIN DAVID TORO ECHEVERRIA

Director(a): Juan Pablo Giraldo Rendón

20 de abril de 2025

Santiago de Cali, 20 de 04 de

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 "Diseño de una Arquitectura de Software en la Nube Mantenible y Escalable para Optimizar la Siembra de Aguacate Hass en la Región Andina de Colombia" realizado por el estudiante de Magister en Ingeniería de Software EDWIN DAVID TORO ECHEVERRIA (cod: 8988329), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,

Juan Pablo Giraldo Rendón

Santiago de Cali, 20 de 04 de

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

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Diseño de una Arquitectura de Software en la Nube Mantenible y Escalable para Optimizar la Siembra de Aguacate Hass en la Región Andina de Colombia" 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,

EDWIN DAVID TORO ECHEVERRIA
Código: 8988329

Ficha Resumen

Trabajo de Grado Maestría en Ingeniería de Software

Título: Diseño de una arquitectura de Software en la nube mantenible y escalable para optimizar la siembra de Aguacate Hass en la Región Andina de Colombia

1. Énfasis: Ingeniería de Software
2. Área de trabajo: Tecnología
3. Tipo de proyecto (Aplicado, Innovación, Investigación): Investigación
4. Estudiante: Edwin David Toro Echeverria
5. Correo electrónico: ingeniero39@javerianacali.edu.co
6. Dirección y teléfono: carrera 109-71B-43 - 3123531898
7. Director: (Ph. D) Juan Pablo Giraldo Rendón
8. Vinculación del director: (Planta/Cátedra/Externo): Externo
9. Correo electrónico del director: jpgiraldo@gmail.com
10. Co-Director (Si aplica): No aplica
11. Grupo o empresa que lo avala (Si aplica): No aplica
12. Otros grupos o empresas: No aplica
13. Palabras clave (al menos 5): Arquitectura de software, aguacate, siembra, tácticas de arquitectura, patrones de diseño, base de datos, atributos de calidad, diagramas de arquitectura de software.
14. Fecha de inicio: 01-05-2024

Agradecimientos

Quiero expresar mi más profundo agradecimiento por la oportunidad de haber desarrollado este proyecto, el cual representa un reflejo de mi dedicación, esfuerzo y compromiso con la excelencia. Este proyecto ha sido posible gracias a la pasión y perseverancia que puse en cada etapa del proceso, desde la investigación inicial hasta el diseño final. Este trabajo ha sido no solo un reto técnico, sino también una valiosa experiencia personal que me ha permitido crecer profesionalmente, enfrentar desafíos con creatividad y reafirmar mi pasión por el desarrollo de soluciones tecnológicas.

Agradecimientos específicos:

Quiero aprovechar esta oportunidad para expresar mi más sincero agradecimiento a mi director, Juan Pablo Giraldo Rendón, por su valiosa orientación y por brindarme el apoyo necesario durante todo el proceso de investigación y ejecución de este proyecto de grado. Su dedicación y profesionalismo fueron fundamentales para llevar a cabo este trabajo.

Asimismo, deseo agradecer profundamente a mi familia por su constante apoyo moral y emocional, que me dio la fortaleza para seguir adelante en todo momento.

Finalmente, agradezco a la Universidad Javeriana de Cali por proporcionarme las bases académicas y el entorno necesario para el desarrollo de este proyecto, contribuyendo significativamente a mi formación profesional.

Resumen

El proyecto de grado se centra en el diseño de una arquitectura de software adaptativa, mantenible y escalable para optimizar la siembra de Aguacate Hass en la región Andina de Colombia. Este enfoque adquiere gran importancia dado al crecimiento de la industria del aguacate en el país y la necesidad de proteger el medio ambiente. La problemática que se aborda es la dificultad que tienen los agricultores para tomar las mejores decisiones al momento de realizar la siembra de aguacate Hass al no contar con herramientas tecnológicas suficientes que les brinden información precisa como el estado del suelo, fuentes hídricas cercanas, presencia de bacterias en el medio ambiente, entre otras variables de importancia en el proceso de siembra. Los objetivos del proyecto se enfocan en definir los atributos de calidad, aplicar tácticas y patrones de arquitectura, diseñar módulos en los cuales se registren datos claves para la siembra del aguacate Hass en la región Andina de Colombia integrando sistemas de información geográfica. Con esto se espera alcanzar resultados como la escalabilidad, disponibilidad seguridad y rendimiento de la arquitectura propuesta en este proyecto.

Palabras Clave: Arquitectura de software, aguacate, siembra, tácticas de arquitectura, arquitectura de software, patrones de diseño, proveedores de servicio en la nube, base de datos, atributos de calidad diagramas de arquitectura de software.

Abstract

The degree project focuses on the design of an adaptive, maintainable and scalable Software Architecture to optimize the planting of Hass Avocado in the Andean region of Colombia. This approach is of great importance due to the growth of the avocado industry in the country and the need to take care of the environment. The problem that is addressed is the difficulty that farmers have in making the best decisions when planting Hass avocado since they do not have sufficient technological tools that can provide them with precise information such as the state of the soil, nearby water sources, presence of bacteria in the environment, etc. The objectives of the project focus on defining quality attributes, applying architectural tactics and patterns, designing modules in which key data for the planting of Hass avocado in the Andean region of Colombia are recorded, integrating geographic information systems. With this, it is expected to achieve results such as scalability, availability and performance of the architecture proposed in this project.

Keywords:
Software architecture, avocado, planting, architecture tactics, software architecture, design patterns, cloud service providers, database, quality attributes software architecture diagrams.

Índice general

Contenido

Índice de tablas

Índice de imágenes

Capítulo 1

Introducción

Durante la última década, el cultivo de aguacate en Colombia ha crecido a un ritmo acelerado. Según el documento Cadena productiva Aguacate" del Ministerio de agricultura, entre el año 2015 y el año 2020, la superficie cultivada de aguacate aumentó en un 60%, pasando de 57.826 hectáreas a más de 93.045 hectáreas. Este crecimiento se debe en gran parte al aumento de la demanda nacional e internacional. En el año 2020, las exportaciones de aguacate Hass en Colombia alcanzaron las 11,492 toneladas, aumentando en más del 100 % con respecto al año 2015, que fueron de 5.332 toneladas. Este documento indica que el aguacate Hass colombiano se ha posicionado en mercados exigentes como el europeo y el estadounidense, gracias a su calidad. (Ministerio de Agri-cultura, 2020)

Debido a esto, se observa que para muchos agricultores colombianos, la siembra de aguacate Hass, que incluye la preparación del terreno y la plantación inicial de los árboles, se convierte en una de las actividades económicas más rentables, con un retorno de inversión que puede alcanzar hasta el 20 % anual, según datos de la Asociación de Exportadores de Aguacate Hass.(Asesores en Producción y Comercio, 2021)

Con el objetivo de aportar en la optimización de la siembra de aguacate Hass, se considera de valor diseñar una arquitectura de software adaptada a las condiciones específicas de la topografía colombiana, ya que en las regiones de Antioquia, Tolima y el Eje Cafetero es donde se concentra el 60% de la producción nacional de aguacate, las variaciones altitudinales y climáticas requieren soluciones tecnológicas avanzadas para maximizar la eficiencia de la siembra.

El diseño de la arquitectura está orientado pensando en una futura implementación, alineándose con los principios de la Industria 4.0, mediante la integración de tecnologías como el Internet de las Cosas (IoT), la analítica de datos e inteligencia artificial. La propuesta planteada tiene la capacidad de ser mantenible, escalable y altamente disponible permitiendo la recopilación, el análisis y la presentación de los datos útiles para optimizar la siembra aguacate Hass en la región Andina de Colombia.

Este documento examina los desafíos actuales, identifica las oportunidades que existen desde una perspectiva tecnológica y establece los principios básicos de la arquitectura de software. En particular, al centrarse en la integración de datos, proporcionando un enfoque unificado que permitirá a los agricultores tener herramientas informáticas para la toma de decisiones en el momento de plantar aguacate Hass basadas en datos.

1.1. Definición del problema

En la actualidad, el proceso de toma de decisiones para la siembra de aguacates Hass presenta diversas limitaciones que dificultan la optimización de recursos, el rendimiento de las cosechas y la sostenibilidad del cultivo. Estas limitaciones se derivan principalmente de la falta de herramientas tecnológicas integradas que permitan analizar en tiempo real factores clave como las condiciones climáticas, las características del suelo y otros datos ambientales críticos.

Situación actual que se desea mejorar

Los agricultores carecen de una solución que centralice y analice datos relevantes para determinar la viabilidad y el rendimiento óptimo de la siembra de aguacates en ubicaciones específicas. Aunque existen fuentes de información como datos climáticos y edafológicos, estos no están integrados ni fácilmente accesibles para los agricultores.

A pesar del crecimiento acelerado del cultivo de aguacate Hass en Colombia, especialmente en regiones como Antioquia, Caldas, Quindío y Risaralda, la mayoría de los agricultores aún operan bajo prácticas tradicionales, con bajo nivel de adopción tecnológica. Según el Ministerio de Agricultura y desarrollo rural, solo el 12% de los productores agrícolas en el país acceden regularmente a plataformas digitales para apoyo a la toma de decisiones *(minciencias, 2017)*. Esta falta de digitalización se traduce en decisiones empíricas, mal uso de recursos hídricos y pérdidas por plagas o cambios climáticos imprevistos.

Además, herramientas como estaciones meteorológicas físicas o plataformas SIG privadas resultan costosas o inaccesibles para pequeños y medianos productores. En este contexto, la necesidad de una solución tecnológica basada en servicios en la nube se vuelve evidente, ya que permite un acceso flexible, en tiempo real, sin necesidad de infraestructura física local, y con escalabilidad acorde a la demanda. Este diseño busca entonces cerrar esa brecha tecnológica, integrando información geoespacial pública, APIs meteorológicas y módulos de procesamiento que optimicen las decisiones agronómicas a bajo costo.

Situación actual indeseable

La falta de acceso a información precisa y en tiempo real resulta en decisiones de siembra basadas en estimaciones subjetivas o datos incompletos. Esto genera riesgos como baja productividad, desperdicio de recursos (agua, fertilizantes, energía) y problemas ambientales asociados al manejo ineficiente del terreno.

1.1.2 Planteamiento del problema

La industria aguacatera actualmente carece de herramientas tecnológicas en el momento de identificar problemas como: variaciones climáticas, infestaciones de plagas y enfermedades bacterianas, lo que afecta la siembra de los árboles de aguacate. Esto tiene como resultado pérdidas significativas en tiempo y dinero para los agricultores colombianos.

En este contexto, se plantea la siguiente pregunta: ¿cómo podría el diseño de una arquitectura de software identificar los problemas mencionados y así ayudar a mejorar la toma de decisiones de los agricultores en la siembra de aguacate Hass en región la andina de Colombia?

1.2. Objetivos del proyecto

A continuación, se presentan el objetivo general y los objetivos específicos del proyecto, orientados al diseño de una arquitectura de software que integre tecnologías avanzadas para optimizar la toma de decisiones en la siembra de aguacate Hass en la región andina de Colombia.

1.2.1. Objetivo General

Diseñar una arquitectura de software que integre tecnologías 4.0, como el análisis de datos y tecnologías de información geográfica (SIG), para proporcionar información que permita a los agricultores tomar las mejores decisiones en el momento de plantar aguacate Hass en la región andina de Colombia.

1.2.2. Objetivos específicos

1. Establecer los diferentes atributos de calidad los cuales buscan potenciar la concepción de la arquitectura.

2. Definir las diferentes tácticas Arquitectónicas considerando la integración de servicios en la nube.

3. Diseñar un módulo parametrizable para gestionar las siguientes variables: condiciones climáticas, niveles de humedad del suelo, fuentes hídricas, presencia de plagas y enfermedades.

4. Diseñar módulo de gestión de datos que analice la información en tiempo real, entregando resultados que faciliten la toma de decisiones en el momento de sembrar aguacate Hass en la región Andina de Colombia.

5. Diseñar módulo que pueda acceder a sistemas de información geográfica (SIG) para poder obtener datos topográficos y climáticos de la zona donde se pretende sembrar aguacate Hass.

1.3. Delimitaciones y alcances

Este proyecto se centra exclusivamente en el diseño de una arquitectura de software para la toma de decisiones en la siembra de aguacate Hass en la región andina de Colombia, utilizando fuentes de datos geoespaciales y climáticos obtenidos por Apis públicas.

El alcance definido contempla lo siguiente:

1. Diseño arquitectónico bajo el enfoque de arquitectura hexagonal combinada con microservicios.
2. Diseño de módulos como parametrización, integración SIG y gestión de datos.
3. Integración conceptual con servicios como Google Cloud Pub/Sub, Firestore, PostgreSQL, BigQuery y Redis.
4. Definición de atributos de calidad y su evaluación por medio de sesiones técnicas.
5. Desarrollo de artefactos como: diagramas de arquitectura (modelo C4), tácticas arquitectónicas, matriz de evaluación conceptual y glosario técnico.

El proyecto no contempla:

1. La implementación de sensores físicos o dispositivos IOT.
2. La adquisición o análisis de imágenes satelitales crudas.
3. El desarrollo de una PoC funcional o código de backend/frontend.
4. El diseño o desarrollo de una aplicación de usuario final.
5. El entrenamiento de modelos de inteligencia artificial o aprendizaje automático.
6. Este diseño no contempla la integración directa con dispositivos IOT o sensores físicos; la información se obtiene exclusivamente de fuentes externas y servicios geográficos públicos.

1.4. Justificación del trabajo de grado

El diseño de una arquitectura en la nube específica para mejorar la toma de decisiones en el cultivo de aguacate Hass en Colombia surge como respuesta urgente a la necesidad de abordar una serie de desafíos y limitaciones existentes. Estos desafíos incluyen la escasez de herramientas tecnológicas y la falta de datos del ecosistema, entre otros. La justificación de esta iniciativa se fundamenta en varias razones clave:

1. **Optimización de Recursos:** A través del análisis de datos y algoritmos especializados se obtienen resultados los cuales permitirán a los cultivadores poder optimizar el uso de recursos críticos como el agua, los pesticidas y los fertilizantes. Al entregar información precisa en tiempo real sobre las condiciones topográficas, se podrá minimizar el impacto ambiental asociado con prácticas agrícolas ineficientes y reducir costos.

2. **Mejora en la Toma de Decisiones:** Con el diseño de una arquitectura en la nube que tenga un componente centralizado de gestión de datos y herramientas de análisis, los agricultores podrán tomar decisiones informadas basadas con datos precisos y contextualizados, lo que les permitirá abordar los desafíos de manera más efectiva.

3. **Sostenibilidad Agrícola:** La integración de tecnologías con sistemas de información geográfica (SIG) promoverá prácticas agrícolas más respetuosas con el medio ambiente, impulsando así la conservación de recursos naturales y la biodiversidad.

4. **Resiliencia ante Amenazas Agrícolas**: La arquitectura está diseñada para que los agricultores puedan tomar decisiones más acertadas frente a amenazas agrícolas como variaciones climáticas, plagas y enfermedades, protegiendo así sus cultivos de manera más efectiva.

5. **Innovación Tecnológica:** El diseño de una arquitectura de software avanzada para el sector aguacatero colombiano no solo impulsa la innovación tecnológica y el cuidado del ecosistema, si no que sienta las bases para futuras integraciones con tecnologías sociales. Esta arquitectura podrá evolucionar como un componente clave para fomentar un modelo de trabajo colaborativo entre cultivadores. En un futuro será posible integrar modelos de inteligencia artificial y aprendizaje automático que incluyan practicas ancestrales.

1.5. Metodología de la investigación

Este proyecto se enmarca en una metodología de investigación aplicada, cuyo propósito es proponer una solución concreta a un problema real del sector agrícola colombiano. En este caso, la solución consiste en el diseño de una arquitectura de software en la nube orientada a facilitar la toma de decisiones informadas sobre la siembra de aguacate Hass en la región andina del país.

Dentro de esta metodología, se adoptó un enfoque de diseño arquitectónico guiado por atributos de calidad, apoyado en buenas prácticas de ingeniería de software. Para ello, se utilizó el método ADD (Attribute-Driven Design), que orienta la toma de decisiones arquitectónicas con base en la priorización de atributos de calidad como escalabilidad, rendimiento, seguridad, disponibilidad, confiabilidad, mantenibilidad y resiliencia.

Este enfoque parte de la premisa de que la arquitectura no debe diseñarse únicamente desde los requisitos funcionales, sino considerando de manera explícita los atributos de calidad que impactan en la sostenibilidad y desempeño del sistema a lo largo del tiempo.

La investigación no se orienta a validar empíricamente una implementación, sino a estructurar una solución sólida a nivel arquitectónico, evaluada conceptualmente mediante escenarios, diagramas y criterios de calidad.

.

El proceso seguido se resume en las siguientes etapas:

1. **Definición del problema y objetivos**: Se identificó la necesidad de mejorar el proceso de toma de decisiones en la siembra de aguacate Hass en la región andina de Colombia.

2. **Recolección de información contextual**: Incluyó entrevistas a agricultores y revisión documental sobre condiciones climáticas, del suelo y mejores prácticas de la siembra.

3. **Identificación de atributos de calidad**: Se establecieron atributos prioritarios mediante un análisis de trade-offs, los cuales orientaron todo el diseño arquitectónico.

4. **Asignación de responsabilidades y tácticas arquitectónicas**: Se implementaron tácticas como aislamiento, encapsulamiento, indirección, adaptación y uso de recursos compartidos para garantizar el cumplimiento de los atributos definidos.

5. **División del sistema en módulos**: Se establecieron componentes como microservicios y microfrontends, permitiendo una distribución lógica y flexible alineada con los atributos priorizado.

6. **Diseño conceptual de la arquitectura**: Se estructuró la solución usando los principios de arquitectura hexagonal y microservicios, integrando tecnologías como Firestore, Redis, BigQuery, Apigee y Google Cloud Pub/Sub.

7. **Selección de estilo arquitectónico:** Se eligió el estilo hexagonal combinado con microservicios, lo que permite desacoplar la lógica de negocio de las tecnologías externas y facilita la mantenibilidad, escalabilidad y evolución del sistema.

8. **Documentación de resultados**: Se produjeron artefactos como el árbol de decisiones arquitectónicas, diagramas C4, tácticas empleadas, glosario técnico, y matrices de evaluación conceptual para sustentar la validez de la propuesta.

1.6 Resultados obtenidos

Se han obtenido una serie de documentos que detallan el diseño de la arquitectura y su ciclo de vida, que abordan específicamente los siguientes elementos:

1.Documento de Escalabilidad: Donde se describe detalladamente los mecanismos utilizados para maximizar la escalabilidad horizontal y vertical, además se incluyen diagramas de arquitectura y pruebas de concepto. *(Enlace del documento)*

2.Documento de Disponibilidad: El cual contiene mecanismos específicos de recuperación ante posibles fallos para garantizar la operatividad continua de todos los componentes de la arquitectura. *(Enlace del documento)*

3.Documento de Rendimiento: El cual describe los mecanismos y componentes utilizados para poder tener un rendimiento optimo en la arquitectura. *(Enlace del documento)*

4.Documento con Políticas de Seguridad: Donde se describe las políticas de seguridad imple- mentadas para proteger datos sensibles, incluyendo medidas específicas. *(Enlace del documento)*

5.Documento de Fiabilidad: El cual describe los mecanismos y componentes utilizados para garantizar la fiabilidad continua de la arquitectura. *(Enlace del documento)*

1.7 Cronograma de actividades

El siguiente cronograma presenta la planificación del proyecto. Se detallan las actividades realizadas desde la formulación del anteproyecto, pasando por la fase de diseño arquitectónico, hasta la entrega final del documento. Cada actividad está organizada por mes y año, permitiendo evidenciar el seguimiento continuo del proceso de investigación y diseño.

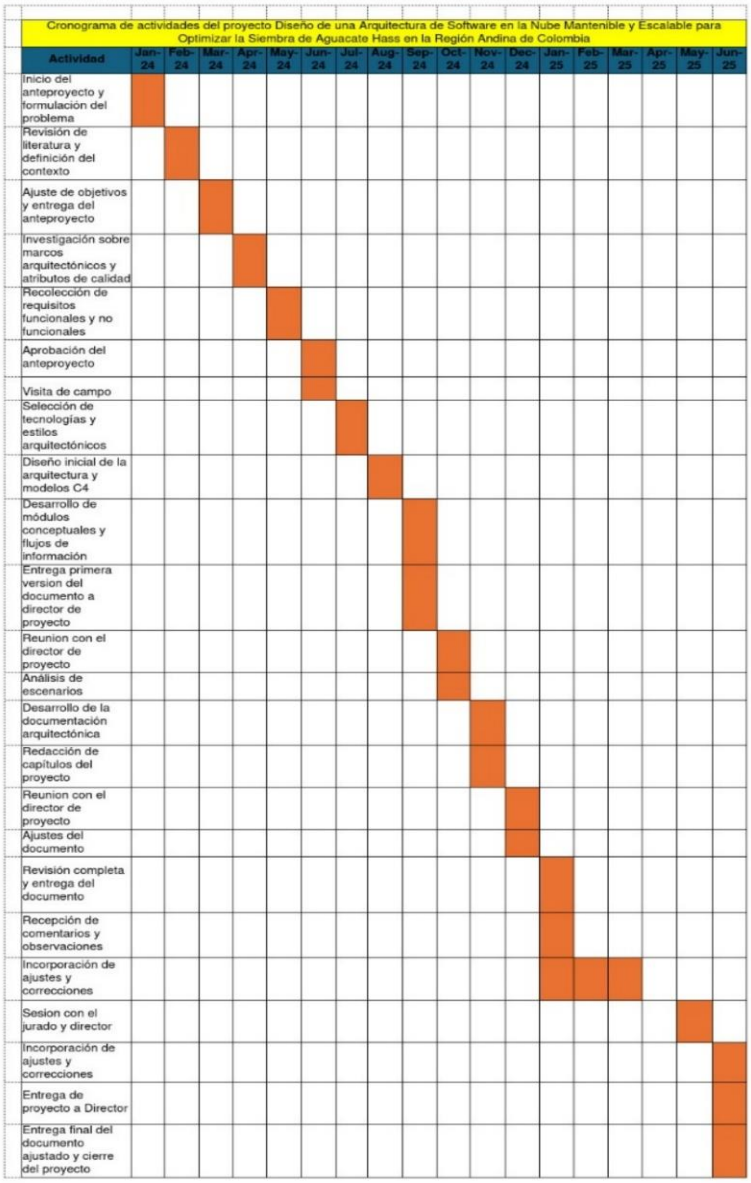


Ilustración 1 Diagrama de actividades

Fuente: Construcción Propia

CAPÍTULO 2
Marco de referencia

2.1. Marco Teórico

Cuando se habla del diseño de una arquitectura de software para el cultivo de aguacate Hass, es esencial tener en cuenta los últimos avances en agricultura y tecnología de la información. Estos avances ayudan a asegurar que la arquitectura diseñada realmente cumpla con los objetivos propuestos.

En primer lugar, en lo que respecta al cultivo de aguacate Hass, es importante comprender los elementos clave que influyen en su crecimiento y desarrollo. Desde el clima hasta el suelo y los nutrientes, cada detalle cuenta cuando se trata de obtener aguacates de alta calidad. Por otro lado, en el ámbito de la arquitectura de software, también hay elementos clave que debemos considerar. Esto incluye no solo la tecnología en sí misma, sino también la forma en que se estructura y organiza el software para que funcione de manera eficiente y efectiva. Al combinar estos dos mundos, agricultura y tecnología, podemos crear una arquitectura de software que realmente se adapte a las necesidades de los agricultores y ayude a maximizar el cultivo de aguacate Hass. Es un proceso emocionante que demuestra cómo con la colaboración entre diferentes disciplinas se pueden llevar a cabo resultados verdaderamente innovadores y beneficiosos.

1. **Área del predio**: El concepto de área del predio se refiere simplemente al tamaño total de un terreno y los límites que lo rodean, que marcan la extensión de una propiedad. En otras palabras, es como la huella que deja una propiedad en el mundo físico: define el espacio que ocupa y hasta dónde se extiende. Cuando hablamos de los linderos de un predio, nos referimos a las líneas imaginarias que delimitan su perímetro y establecen sus fronteras. Estas líneas son importantes porque marcan dónde termina un terreno y comienza otro, lo que ayuda a evitar disputas y conflictos sobre la propiedad de la tierra. *(Ministerio de Justicia, 2022)*

2. **Altitud**: El concepto de altitud, hace referencia a la altura o elevación de un lugar en relación con un punto de referencia específico, que generalmente se toma como el nivel medio del mar. Es como medir lo alto que está un lugar desde el nivel del mar hasta su posición actual. Por ejemplo, si una persona está en una montaña y quisiera saber a qué altura esta, mediría la altitud desde el nivel del mar hasta su ubicación en la montaña. Esta medida ayuda a comprender cuán alto o bajo esta esa persona en

comparación con otros lugares. *(Matemáticas: Problemas introductorios tipo PISA para el primer ciclo de Educación Secundaria Obligatoria, 2014)*

3. **Humedad del suelo**: Es un factor crucial en la agricultura, ya que afecta directamente el crecimiento y la salud de las plantas, incluida la planta de aguacate Hass. Cuando hay demasiada o muy poca agua en el suelo, puede causar problemas para las plantas, lo que afecta su desarrollo y producción. La cantidad de humedad en el suelo puede variar debido a diferentes factores, como el clima y los cambios meteorológicos. Por ejemplo, períodos de lluvia intensa pueden aumentar la humedad del suelo, mientras que períodos de sequía pueden hacer que el suelo esté más seco. Es importante monitorear y gestionar cuidadosamente la humedad del suelo para garantizar condiciones óptimas para el crecimiento de las plantas de aguacate Hass. Esto puede implicar el uso de técnicas de riego adecuadas y el seguimiento regular de las condiciones del suelo para asegurarse de que las plantas reciban la cantidad adecuada de agua en todo momento. *(Cherlinka V, 2024)*

4. **Salinidad del suelo**: se refiere a la acumulación de sales en el suelo, lo que puede tener un impacto significativo en el cultivo de plantas como el aguacate Hass. Cuando hay demasiada sal en el suelo, puede interferir con la capacidad de las plantas para absorber agua, lo que puede causar una serie de problemas para su crecimiento y desarrollo. *(Cherlinka V, 2024)*

5. **Condiciones climáticas**: Las condiciones climáticas comprenden un conjunto de variables atmosféricas como temperatura, presión barométrica, humedad relativa, precipitaciones y velocidad del viento que caracterizan el estado del clima en una región determinada. En el ámbito de la arquitectura de software, estas variables adquieren relevancia en el diseño de sistemas, plataformas de monitoreo ambiental y soluciones basadas en IoT. La integración de datos climáticos en sistemas inteligentes permite anticipar eventos meteorológicos, optimizar recursos y adaptar comportamientos operativos en tiempo real. Por ejemplo, la presión atmosférica y la humedad pueden ser utilizadas como indicadores predictivos en algoritmos de detección de tormentas, mientras que la velocidad del viento y las precipitaciones pueden condicionar la ejecución de tareas en sistemas agrícolas automatizados. Así, las condiciones climáticas no solo afectan la vida cotidiana, sino que también representan entradas críticas para arquitecturas orientadas a eventos, sistemas de toma de decisiones y modelos de simulación ambiental. *(Climate Technologies:, 2019)*

6. **Temperatura Ambiental**: La temperatura ambiental se refiere al valor térmico medido en una zona geográfica específica, el cual influye directamente en las condiciones operativas de sistemas físicos y computacionales. En el contexto de arquitectura de software, especialmente en aplicaciones que interactúan con sensores o dispositivos IoT (Internet of Things), este parámetro puede ser crítico para la toma de decisiones automatizadas, la gestión energética o el control de

ambientes inteligentes. Por ejemplo, en sistemas de climatización inteligente, la lectura de temperatura ambiental permite ajustar dinámicamente el comportamiento del software para optimizar el confort térmico y la eficiencia energética. Así, la temperatura no solo representa una condición física, sino también un dato relevante que puede ser modelado, procesado y utilizado como entrada en arquitecturas orientadas a eventos, sistemas embebidos o plataformas de monitoreo ambiental. *(iotforall.com, 2025)*

7. **variaciones en la temperatura ambiental** Las fluctuaciones en la temperatura ambiental constituyen un factor relevante en el diseño de sistemas adaptativos dentro de la arquitectura de software, especialmente en entornos sensibles al contexto como la agricultura de precisión, la gestión urbana inteligente o el monitoreo ambiental. Estas variaciones pueden desencadenar cambios en la lógica de negocio de aplicaciones que dependen de datos climáticos, afectando desde la programación de actividades automatizadas hasta la activación de alertas en sistemas críticos. Por ejemplo, en plataformas de planificación agrícola, los cambios térmicos pueden modificar algoritmos de riego, fertilización o cosecha, mientras que, en sistemas de gestión de recursos naturales, pueden influir en la modelación de hábitats y en la toma de decisiones para la conservación de la biodiversidad. En este sentido, la temperatura ambiental no solo condiciona actividades humanas, sino que también representa una variable clave en arquitecturas orientadas a datos contextuales. *(cambio climatico, 2024)*

8. **Principios de Arquitectura de Software**: Los principios de arquitectura de software son fundamentales para establecer la modularidad, cohesión y acoplamiento, entre otros aspectos. Proporcionan una guía invaluable para que el diseño de un sistema sea disponible, mantenible y escalable. Cuando por ejemplo estás construyendo una casa: antes de comenzar la construcción, necesitas establecer principios de diseño que determinen cómo se organizarán los espacios, qué materiales se utilizarán y cómo se integrarán las diferentes partes de la casa. De manera similar, en la arquitectura de software, los principios ayudan a estructurar el sistema de manera que sea fácil de entender y modificar a medida que evolucionan los requisitos del negocio. Estos principios pueden incluir la separación de preocupaciones, la minimización de dependencias, la maximización de la cohesión y modularidad, entre otros. Al seguir estos principios, los arquitectos de software pueden crear sistemas que sean adaptables a los cambios y que puedan crecer y evolucionar con el tiempo. Los principios de arquitectura de software son fundamentales para establecer la base sobre la cual se construirá el sistema, garantizando que sea flexible, mantenible y escalable a medida que cambian las necesidades del negocio. *(Cervantes H., 2016)*

9. **Patrones de diseño arquitectónico**: Son enfoques estructurados que ofrecen soluciones proba- das para desafíos recurrentes en el diseño de sistemas de software. Imaginemos que estás construyendo una casa: hay diferentes formas de organizar los

espacios y estructuras para que funcionen de manera eficiente y satisfagan tus necesidades. De manera similar, en el diseño de software, los patrones de diseño arquitectónico ofrecen diferentes plantillas que puedes seguir para organizar y estructurar tu sistema de manera efectiva. Por ejemplo, el patrón de arquitectura hexagonal se centra en separar la lógica de negocio del resto del sistema, mientras que el patrón por capas divide el sistema en capas lógicas y funcionales. Por otro lado, el patrón de microservicios se basa en dividir el sistema en componentes independientes y escalables. *(Bass, L., Kazman, R., Clements, P., 2013)*

10. **Atributo de calidad**: Es una característica o propiedad de un sistema que puede ser medida y evaluada para determinar en qué medida satisface las necesidades y expectativas de sus usuarios. por ejemplo, cuando vamos a comprar un automóvil: considerarías atributos de calidad como la eficiencia del combustible, la seguridad, la comodidad y la confiabilidad antes de tomar una decisión. Del mismo modo, en el diseño de arquitectura de software, los atributos de calidad son aspectos clave que se deben tener en cuenta para garantizar que la arquitectura funcione de manera efectiva y satisfaga las necesidades de los usuarios. Estos atributos pueden incluir la usabilidad, el rendimiento, la seguridad, la escalabilidad y la confiabilidad, entre otros. Cada uno de estos atributos puede ser medido y evaluado para determinar si la arquitectura cumple con los estándares y requisitos establecidos. *(Bass, L., Kazman, R., Clements, P., 2013)*

11. **Tácticas arquitectónicas**: Son decisiones de diseño que se emplean específicamente para poder optimizar atributos de calidad particulares en la arquitectura de software. Imaginemos que estamos construyendo una casa: para garantizar que la estructura sea segura y resistente, podrías utilizar materiales específicos, como acero reforzado o concreto, y aplicar técnicas de construcción especializadas. De manera similar, en la arquitectura de software, las tácticas arquitectónicas se eligen y aplican para mejorar aspectos clave del sistema, como la seguridad, el rendimiento o la escalabilidad. Estas tácticas pueden incluir la segmentación de funcionalidades en componentes independientes, la implementación de redundancia para mejorar la disponibilidad del sistema, o la aplicación de técnicas de caché para mejorar el rendimiento. *(Bass, L., Kazman, R., Clements, P., 2013)*

12. **Gestión de datos**: La gestión de datos desempeña un papel fundamental en la arquitectura de software, donde es crucial para la toma de decisiones informadas y la optimización de procesos. sí por ejemplo estamos administrando una granja: para tomar decisiones sobre siembra, riego o cosecha, necesitas tener acceso a datos relevantes sobre el clima, el suelo, el estado de los cultivos, entre otros. La gestión eficaz de estos datos te permite recopilar, almacenar, procesar y analizar información crítica para tomar decisiones estratégicas y mejorar la eficiencia de las operaciones agrícolas. En el ámbito de la arquitectura de software, la gestión de datos implica diseñar sistemas que puedan manejar grandes volúmenes de información de manera eficiente y confiable. Esto puede incluir el uso de bases de datos robustas, la

implementación de técnicas de almacenamiento y recuperación de datos optimizadas, y la integración de herramientas de análisis de datos para extraer información valiosa de los conjuntos de datos. La gestión de datos en la arquitectura de software es fundamental para habilitar la toma de decisiones informadas y optimizar las operaciones agrícolas, asegurando que los sistemas puedan manejar y aprovechar eficazmente la gran cantidad de información disponible. *(Qué es la gestión de datos, 2024)*

13. **Calidad de software**: Es un aspecto crítico en el diseño de la arquitectura de software, ya que se refiere al conjunto de características necesarias para satisfacer las necesidades del sistema de software. Imagina que estás construyendo una casa: la calidad de la construcción, la durabilidad de los materiales y la funcionalidad de los espacios son aspectos fundamentales que determinan si la casa cumplirá con tus necesidades y expectativas. De manera similar, en la arquitectura de software, la calidad se refiere a la capacidad del sistema para cumplir con los requisitos de rendimiento, seguridad, confiabilidad, usabilidad, disponibilidad, modificabilidad, integrabilidad, deplegabilidad, etc. Para garantizar la calidad del software, es necesario adoptar prácticas de desarrollo y diseño que se centren en la prevención de defectos y la mejora continua. Esto puede incluir la realización de pruebas exhaustivas, la adopción de estándares de codificación, y la aplicación de principios de diseño robusto y modular. La calidad de software es un aspecto crítico en el diseño de la arquitectura de software, ya que garantiza que el sistema pueda cumplir con las necesidades y expectativas de los usuarios de manera efectiva y confiable. *(Salud Electrónica, 2021)*

14. **Diseño de arquitectura de software:** Todo diseño que se comienza tiene requisitos funcionales y no funcionales, si no se tienen claros estos requisitos desde el principio se pueden correr riesgos, para evitar estos riesgos se debe: comenzar con una visión general para los cuales se utilizan diferentes conocimientos y se sigue un lineamiento donde se encapsulan todos los conocimientos arquitectónicos y de desarrollo de software, comenzar en pensar en cada componente, considerar la documentación que se necesita, identificar los atributos de calidad, realizar diseño de la arquitectura teniendo en cuenta que el primer diseño es solo la primera iteración, *(Ronaldo, L. Donni, R, 2024)*

15. **Proveedores de servicio en la nube**: Un proveedor de servicios en la nube es una empresa de TI que proporciona recursos de procesamiento a pedido, por lo general , los modelos de servicio en la nube se definen como IaaS (Infraestructura como servicio), PaaS(Plataforma como servicio), SaaS (Software como servicio), existen varios proveedores de servicio pero sin ninguna duda los tres más grandes en este momento al generar este documento son Google Cloud, Amazon Web Services, Microsoft Azure.*(Google Cloud, sfc)*

16. **Requerimientos funcionales**: Los requerimientos funcionales describen acciones

específicas los cuales se dividen en reglas de negocio y casos de uso comprendiendo la necesidad del cliente o partes interesadas para poder establecer los objetivos principales de un sistema de software. *(Google Cloud, sfb)*

17. **Arquitectura de alto nivel**: Establece una visión general de la arquitectura del producto donde se identifican los actores que interactúan con el sistema lo que permite a los desarrolladores y arquitectos comprender su estructura y tomar decisiones durante el proceso de desarrollo, en este nivel se enfocan los patrones de diseño y las tecnologías utilizadas en cada componente *(Google Cloud, sfa)*

18. **Base de datos**: Es una recopilación organizada de datos que se almacenan, editan, recuperan y actualizan electrónicamente, normalmente una base datos está controlada por un sistema de gestión de base de datos, los datos guardados están estructurados por medio de filas y columnas en una seria de tablas lo que permite aumentar la eficacia en el procesamiento y la consulta de datos de esta manera se puede gestionar fácilmente la información. *(Amazon Web Services, 2023)*

19. **Arquitectura de referencia**: Una arquitectura de referencia identifica los contornos normales de un sistema. El núcleo de toda arquitectura de referencia es el modelo. Los modelos muestran los componentes del sistema, sus relaciones y los atributos que deben especificarse. Una arquitectura de referencia puede abarcar cualquier dominio y cualquier parte de una empresa. Puede ser abstracta o detallada. La arquitectura de referencia mejorará la calidad y acelerará el desarrollo de la arquitectura. *(Arquitectura de Referencia, 2025)*

20. **Micro Frontend:** Es un estilo arquitectónico para el desarrollo web donde una aplicación se divide en varias funcionalidades y se entrega de forma independiente. *(Euristiq, 2012)*

21. **Lenguaje de programación**: Es una forma de comunicarnos con la computadora donde le indicamos que queremos hacer. Existen diferentes lenguajes como Java, Python, PHP. *(CILSA, 2017)*

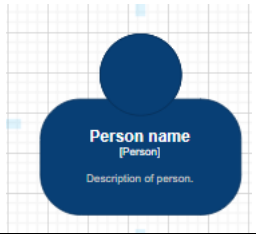
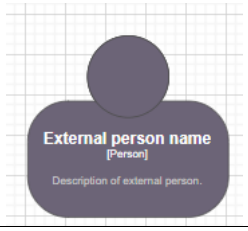
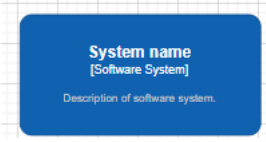
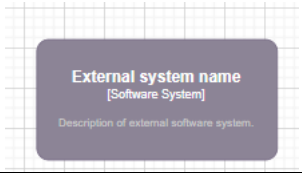
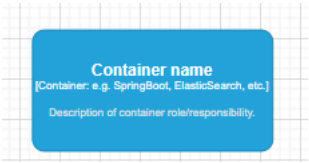
22. **Contenedor de software**: Hace referencia a las estructuras estáticas es decir son paquetes de software que incluyen todos los elementos necesarios para que se ejecuten en cualquier entorno *(cloud.google.com,2025)*

23. **Componente de software**: Agrupación lógica de código fuente tanto de Frontend como de Backend. son los elementos que conforman cualquier programa o aplicación informática. Son los bloques de construcción que permiten que el software funcione correctamente. *(agenciarococrm.com,2023)*

24. **Tradeoffs**: Son aquellas situaciones en las que se deben balancear los atributos de calidad, considerando beneficio y pérdida, con esto se decide finalmente cuál será la

estructura y arquitectura de un sistema de software. *(Pablo Cruz, 2021)*

25. **Modelo C4**: es una forma sencilla de comunicar la arquitectura del software en diferentes niveles de abstracción, de modo que se pueden contar diferentes historias a diferentes audiencias. Cuenta con una convención propia sencilla que se explica en la siguiente Figura:

SIMBOLO	DESCRIPCION
	Usuario interno que interactúa con un sistema, este icono puede ser utilizado en todos los niveles del modelo c4 con la descripción de su rol.
	Usuario externo que interactúa con un sistema, este icono puede ser utilizado en todos los niveles del modelo c4.
	Es el sistema de información general, usado en el diagrama de contexto al realizar una vista más detallada el siguiente nivel de inspección serán los contenedores.
	Sistema de información externo, no se suele hacer un detalle muy específico.
	Contenedor de código genérico usado en el diagrama de contenedores, al realizar una vista más detallada el siguiente nivel de inspección serán los componentes.
	Contenedor de base de datos generalmente se usa en el diagrama de contenedores, al realizar una vista más detallada el siguiente nivel de inspección serán los componentes.

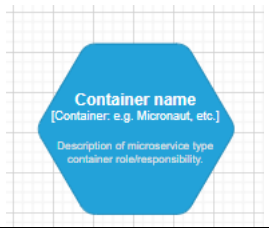
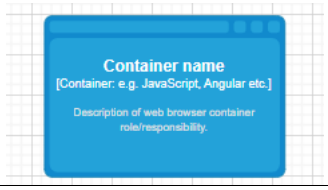
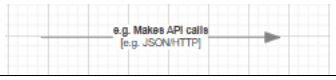
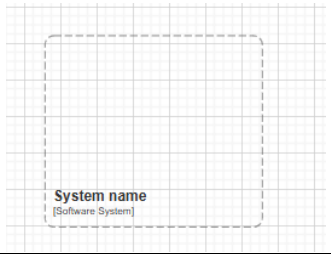
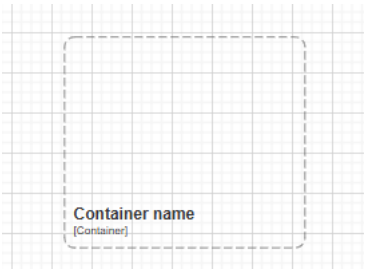
	Contenedor de código de microservicios, se utiliza en el diagrama de contendores, al realizar una vista más detallada el siguiente nivel de inspección serán los componentes.
	Contenedor de bus de mensajes usado en el diagrama de contenedores, al realizar una vista más detallada el siguiente nivel de inspección serán los componentes.
	Contenedor para el frontend se utiliza en el diagrama de contendores, al realizar una vista más detallada el siguiente nivel de inspección serán los componentes.
	Componente utilizado tanto en backend como en frontend representa la manera en la cual se organiza el código, al realizar una vista más detallada el siguiente nivel de inspección serán los diagramas de clases en UML.
	Relaciones entre elementos y su protocolo de comunicación.
	Relaciones entre elementos con mensajes dicientes de una acción.
	Relaciones entre elementos.
	Representa el límite de un sistema, se usa en el diagrama de contendores si el diagrama contiene diferentes sistemas externos.
	Representa el límite que puede existir en uno o varios contenedores.

Tabla 1 Cuadro de Convenciones Modelo C4

Fuente: Construcción Propia

25. **Arquitectura hexagonal**: También conocida como arquitectura de puertos y adaptadores, tiene como función principal separar en capas la aplicación lo que permite que pueda evolucionar de forma independiente. Gracias a este desacoplamiento se pueden obtener ventajas en el momento de realizar pruebas ya que ninguna capa interviene con otra. (edusalguero, 2022)

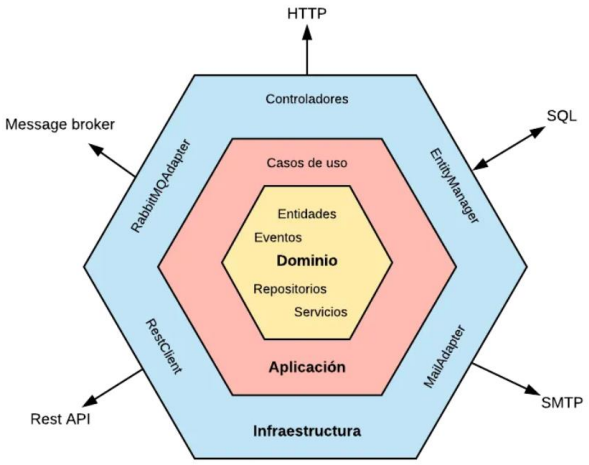


Ilustración 2 Ejemplo Arquitectura Hexagonal

Fuente: Edu Salguero, 2018, Arquitectura Hexagonal, disponible en: https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f

26. **Arquitectura de Microservicios**: Esta arquitectura tiene como función principal crear componentes desacoplados ligeros los cuales son capaces de escalar independientemente operando en varias instancias de forma dinámica. Uno de los mayores beneficios es la velocidad y su fácil implementación. (intel, 2012)

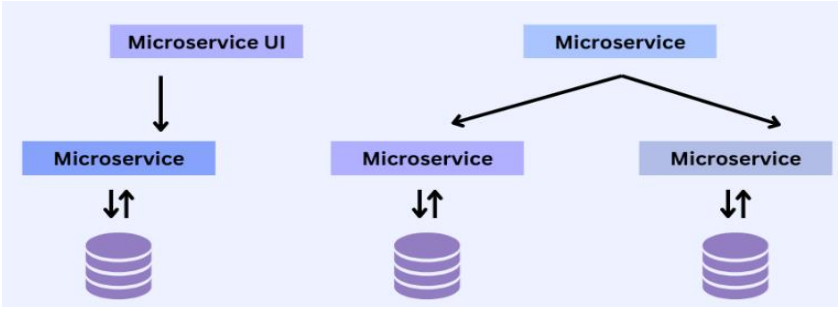


Ilustración 3 Ejemplo Arquitectura de Microservicios

Fuente: NamLabs Technologies, 2025, *What is Microservices Architecture*, disponible en: https://www.atatus.com/blog/microservices-architecture/

27. **Atributo de calidad de Escalabilidad**: Este atributo se refiere a la capacidad que tiene el sistema para manejar el aumento de carga sin que disminuya su rendimiento. Existen dos tipos de escalamiento existe el escalado horizontal y vertical en los servicios de procesamiento y almacenamiento según la demanda. *(syndcode, 2012)*

28. **Atributo de calidad de Disponibilidad**: Este atributo hace referencia a la capacidad de un sistema para estar siempre disponible y listo para desempeñar funciones cuando se necesite. *(syndcode, 2012)*

29. **Atributo de calidad de Rendimiento**: Muestra la respuesta del sistema a la realización de determinadas acciones durante un determinado periodo de tiempo. *(syndcode, 2012)*

30. **Atributo de calidad de Seguridad:** Este atributo contempla la protección de la información que se encuentra en un Sistema. *(syndcode, 2012)*

31. **Atributo de calidad de Confiabilidad**: Este atributo se refiere al grado de estabilidad que tiene un sistema cuando se expone a condiciones inesperadas teniendo una excelente integridad en los datos. *(syndcode, 2012)*

32. **Atributo de calidad de Recuperación ante fallos**: Este atributo se refiere a la capacidad que tiene un sistema de recuperarse ante un fallo en el menor tiempo posible *(syndcode, 2012)*

33. **Atributo de calidad de Mantenibilidad**: Capacidad de los componentes para ser modificados debido a necesidades evolutivas o correctivas. *(syndcode, 2012)*

34. **Arquitectura de Microservicios**: Es un estilo arquitectónico en el cual existen módulos de ligero acoplamiento los cuales operan de forma independiente. *(intel, 2012)*

35. **API REST**: Es un mecanismo que permite que una aplicación o servicio acceda a un recurso dentro de otra aplicación. (IBM, 2012)

2.2 Estado del Arte

El diseño de arquitecturas de software aplicadas al cultivo del aguacate Hass ha sido explorado en diferentes proyectos a nivel nacional e internacional. A continuación, se presentan distintas iniciativas y desarrollos tecnológicos que, si bien no resuelven por completo el problema planteado en esta investigación, aportan elementos relevantes. Cada propuesta es descrita con su enfoque principal contribución y limitaciones frente al problema abordado en este proyecto.

Aplicativo TECNOHASS de AGROSAVIA

Aplicación diseñada para el manejo productivo del aguacate Hass en el departamento del Cauca. Incluye módulos de fertilización, riego, manejo de plagas y nichos productivos con base en datos climáticos y edáficos.

El aplicativo TECNOHASS es una herramienta digital gratuita que está orientada a formular planes de manejo productivo en los cultivos de aguacate Hass en el departamento del Cauca.

El aplicativo cuenta con tres módulos de uso:

Manejo del riego y fertilización: Genera la necesidad sugerida de fertilizante para cada macro y micronutriente y genera tres opciones de planes de fertilización en función de algunas fuentes de fertilizante sugeridas.

Manejo de plagas: A partir de modelos generados por la correlación de variables climáticas y niveles de población.

Nichos productivos: En este módulo se pueden consultar las zonas aptas para el desarrollo del cultivo acorde a la información recopilada y procesada por AGROSAVIA. (AGROSAVIA, 2018)

Contribución al proyecto: Aporta una visión del impacto que tiene la tecnología sobre la productividad agrícola, reforzando el valor que puede tener una arquitectura pensada para decisiones previas como la siembra de aguacate Hass.

Limitación del proyecto: Este software, aunque se acerca mucho a la solución del planteamiento del problema, solo esta delimitada para la zona del Cauca.

Modelo Predictivo para la oferta de aguacate Hass en la región del Tolima.

Este proyecto propone el diseño de un modelo predictivo orientado a anticipar la oferta de aguacate Hass mediante la identificación de variables claves de producción. Se recopilan datos desde fuentes gubernamentales y directamente de los agricultores.
Uno de los enfoques es la creación e implementación de este modelo predictivo ya que puede marcar diferencia en la producción agrícola. No solo ayudará a los agricultores a anticipar posibles problemas, como condiciones climáticas adversas, sino que también les permitirá tomar decisiones más informadas para mejorar la calidad de sus cultivos.

El objetivo general de este proyecto es diseñar un modelo predictivo específicamente para el aguacate Hass, con objetivos específicos que incluyen definir las variables que influyen en su producción. Este proceso de implementación del modelo predictivo se centra en la recopilación y análisis de información relevante. El cual busca información en diversas fuentes, como las alcaldías, el Ministerio de Agricultura y otras entidades gubernamentales. Además, se establece un contacto directo con los agricultores, solicitando información adicional para garantizar la precisión y la completitud de los datos recopilados. (Camero, J. D. G, 2020)

Contribución al proyecto: Contribuye como base para el análisis de información en el módulo de gestión de datos, lo que refuerza la necesidad de integración de fuentes de información confiables y contextualizadas.

Limitación del proyecto: Esta revisión exhaustiva de la propuesta no aborda completamente el planteamiento del problema del proyecto, ya que no se define claramente una arquitectura específica. Aunque se contempla el diseño de un modelo predictivo, no se especifica de qué manera está estructurado.

Sistema IS-AR para Programación de Riego con Imágenes Satelitales

Este sistema emplea datos de radar SAR de la Agencia Espacial Europea para estimar humedad del suelo y facilitar el riego del aguacate Hass. Utiliza modelos climáticos y funciona a través de una plataforma digital.

Imagina un sistema que simplifica la vida de los agricultores al programar el riego para el cultivo de aguacate Hass. Este sistema, conocido como IS-AR, es una herramienta tecnológica innovadora que hace precisamente eso: facilita la programación del riego con solo unos pocos datos. Lo más sorprendente es que IS-AR se integra con la Agencia Espacial Europea (ESA), lo cual le permite acceder a imágenes de radar (SAR) de toda la superficie terrestre.

Estas imágenes se actualizan cada 3 o 10 días, dependiendo de la región, proporcionando información crucial sobre el suelo, como la humedad, que es fundamental para el cultivo del aguacate. Gracias a la combinación de modelos e innovadoras técnicas de software, el sistema ha sido adaptado de manera que funcione de manera óptima. Esto significa que los agricultores pueden confiar en IS-AR para programar el riego de manera eficiente y efectiva, sin tener que preocuparse por complicaciones técnicas.

Es una herramienta que realmente hace la diferencia en el día a día de los agricultores, permitiéndoles centrarse en lo que más importa: cultivar aguacates de alta calidad. El acceso a la plataforma se realiza a través del enlace http://www.is-sar.com/, el cual está diseñado para ser compatible con diversos dispositivos.

Contribución al proyecto: Destaca la relevancia del uso información geoespacial, que en nuestro caso se integran mediante el módulo de integración SIG y el uso de APIs geográficas como parte de la arquitectura propuesta.

Limitación del proyecto: Sin embargo, esta propuesta no resuelve por completo el problema planteado en el proyecto, ya que al intentar acceder al sistema utilizando la URL proporcionada, esta no está disponible.

SupPlant Inteligencia Artificial y Sensores para el Monitoreo Agrícola

SupPlant combina inteligencia artificial y sensores para detectar estrés hídrico y crecimiento en cultivos de aguacate en algunas regiones del país de México.

La inteligencia artificial está revolucionando la forma en que los agricultores cultivan aguacates Hass. Ahora, en lugar de depender únicamente de su intuición, tienen a su disposición una herramienta poderosa que les ayuda a determinar el momento y la mejor manera de regar sus cultivos. Este avance se logra gracias a la integración de la inteligencia artificial con los sensores de SupPlant. Estos dispositivos no solo pueden detectar el estrés de las plantas, sino que también siguen de cerca su crecimiento y desarrollo. Actualmente, esta tecnología ya se está utilizando con éxito en el estado de Michoacán, México. El objetivo es expandir este sistema para que todos los agricultores de la región puedan monitorear y rastrear las necesidades de sus cultivos en tiempo real. Esto significa que, independientemente de su experiencia o conocimientos previos, los agricultores tendrán acceso a información valiosa que les permitirá tomar decisiones más informadas y mejorar la calidad de sus cultivos. Es un cambio revolucionario que está transformando la agricultura tal como la conocemos. (THE FOOD TECH, 2021)

Contribución al proyecto: Refuerza la importancia de incorporar análisis en tiempo real y procesamiento de datos climáticos, elementos fundamentales en el diseño del módulo de gestión de datos de la arquitectura.

Limitación del proyecto: Esta técnica no puede resolver el planteamiento del problema, ya que no se observa una arquitectura o por lo menos un diseño que justifique el cómo se aborda el tema, siendo la inteligencia artificial un tema que puede abarcar todo un contexto en el mundo del software.

Dispositivo Clasificador de Madurez del Aguacate Hass

Permite identificar el punto óptimo de cosecha a través de la apariencia de la fruta. Mejora los ingresos de los agricultores.

El dispositivo clasificador de la madurez del aguacate Hass, mediante el cual se evalúa la apariencia externa de la cáscara del fruto del aguacate para determinar el momento óptimo de la cosecha, determinó que el uso de dicha tecnología logra que se incrementen los ingresos netos de los agricultores de aguacate en 91 %. (Nigrinis, P. D. P, 2023)

Contribución al proyecto: Aporta una visión del impacto que tiene la tecnología sobre la productividad agrícola, reforzando el valor que puede tener una arquitectura pensada para que los agricultores puedan tomar buenas decisiones en el momento de sembrar aguacate Hass.

Limitación del proyecto: Esta tecnología, si bien puede indicarle al agricultor en qué momento se puede recoger la cosecha, no considera la siembra, que es el problema abordado en este proyecto.

Plataforma Avolab Información Técnica para Agricultores

Se enfoca en la tecnificación y modernización del agro. Entregando información técnica para productores y exportadores.

Avolab es un laboratorio y una plataforma tecnológica la cual se encarga de brindar información técnica a todos los actores de la cadena agrícola, especialmente productores, comercializadores y exportadores. Son pioneros en el futuro tecnológico de la agroindustria en Colombia. Acompaña a cada productor y le indica cómo hacer que sus predios sean más tecnificados. (Ciencia y tecnología al servicio del agro, 2020)

Contribución al proyecto: Refuerza la necesidad de contar con plataformas tecnológicas integradas y enfocadas en el apoyo al agricultor y el cuidado del medio ambiente.

Limitación del proyecto: Esta plataforma tecnológica, si bien entrega información técnica, no parece estar solucionando el problema de este proyecto, ya que no son muy enfáticos en qué tipo de información técnica brindan a los agricultores.

Hecterra Aplicación para el Control de Operaciones Agrícolas

Permite controlar operaciones agrícolas mediante datos telemáticos y GPS. Brinda información sobre rotación de cultivos, historial de campos y productos utilizados.

Hecterra es una aplicación sencilla y eficiente para el sector agrícola que permite controlar la ejecución de operaciones de campo basándose en los datos telemáticos. La solución GPS agrícola ofrece a los usuarios los datos fiables y transparentes sobre campos, rotación de cultivos, operaciones de campo y muchos más. Mediante el cálculo automático del área cultivada, el registro de operaciones y productos agrícolas en guías especiales, el almacenamiento del historial de campos y los informes detallados, los usuarios tienen todo lo necesario para una eficiente gestión agrícola. (Winlon, 2024)

Contribución al proyecto: Valida el uso de tecnologías, motivando la integración de datos geoespaciales para el diseño de la arquitectura.

Limitaciones del proyecto: Es una herramienta tecnológica que aborda temas agrícolas, pero no se enfoca en el aguacate Hass lo que conlleva a que es una plataforma muy generalizada que no puede resolver el problema del proyecto.

Proyecto de Reflectancia en Aguacates Afectados por Plagas

Busca crear una plataforma que utilice sensores remotos para identificar afectaciones por plagas, enfermedades o deficiencias nutricionales mediante análisis de reflectancia.

Proyecto de determinación de los grados de reflectancia de plantas de aguacate afectadas por disturbios como plagas, enfermedades y deficiencias nutricionales. Este proyecto se enfoca en generar una plataforma tecnológica que dispone de datos regionales para la toma de decisiones, la cual aplica técnicas de sensores remotos para identificar y controlar disturbios anteriormente mencionados. (Asociación de Productores y Empacadores Exportadores de Aguacate de México (APEAM), 2024)

Contribución al proyecto: Aporta perspectiva sobre el valor del monitoreo remoto y análisis de condiciones adversas, reforzando la inclusión de APIs climáticas y edáficas para análisis de condiciones de siembra.

Limitación del proyecto: Es un proyecto que hasta ahora no está implementado por lo tanto no resuelve el problema que se aborda en este proyecto.

2.3. Resumen del capítulo

En este capítulo se investigaron diferentes proyectos, plataformas y tecnologías aplicadas a la siembra de cultivo del aguacate Hass, tanto a nivel nacional como internacional. Estas iniciativas permitieron identificar enfoques relevantes que han abordado problemáticas similares a las del presente proyecto. Aunque algunos de estos proyectos, como TECNOHASS o SupPlant, ofrecen funcionalidades valiosas, también presentan limitaciones importantes como cobertura geográfica restringida, falta de diseño arquitectónico documentado o enfoque exclusivo en etapas específicas de la siembra, no obstante, aportan insumos fundamentales para estructurar la propuesta actual, como la identificación de variables críticas, uso de datos geoespaciales y climáticos.

Este análisis comparativo del estado del arte permitió delimitar con mayor claridad el alcance del proyecto y fundamentar el diseño arquitectónico propuesto, el cual busca integrar componentes tecnológicos para facilitar la toma de decisiones durante la etapa de siembra del aguacate Hass, especialmente en la región Andina de Colombia.

CAPÍTULO 3

Desarrollo del Proyecto

3.1 Introducción al desarrollo

Este capítulo presenta de forma detallada el desarrollo del proyecto. A partir del problema y los objetivos planteados, se describe la selección de tecnologías, el diseño de los módulos, las decisiones arquitectónicas adoptadas, y se ilustran con diagramas funcionales y estructurales.

3.2 Visita de campo

Durante el desarrollo del proyecto, se realizó una visita de campo en el mes de junio del año 2024 a la finca El Recreo, ubicada en la vereda La Camelia, en los límites entre San Sebastián de Mariquita y Fresno Tolima. En esta visita, se entrevistó al agricultor Javier Carvajal el cual nos ha dado su permiso para publicar su nombre, quien lleva más de 15 años cultivando aguacate como su principal fuente de sustento. Durante la entrevista, se identificaron los desafíos que existen en la siembra de aguacate Hass, entre los que se destacan la falta de certeza sobre las áreas óptimas de la finca para sembrar, la dificultad para determinar cuántos árboles de aguacate podría ser sembrados de manera eficiente. Estos hallazgos resaltan la importancia de comprender las necesidades que tienen los agricultores como el señor Javier Carvajal y orientar las soluciones propuestas en el proyecto.

3.3 Fundamentos de Diseño Arquitectónico Aplicados al Proyecto

La arquitectura diseñada en este proyecto se apoya en un conjunto de decisiones conceptuales que definen su estructura general, tomando como base buenas prácticas reconocidas en la ingeniería de software. Estas prácticas no se presentan como una arquitectura de referencia genérica, sino como el marco específico adoptado para orientar el diseño de la solución planteada.

A partir del análisis del problema y los objetivos del proyecto, se establecieron los siguientes elementos fundamentales de diseño:

- Atributos de calidad priorizados**:** Mediante un proceso de análisis de tradeoffs, se priorizaron diferentes atributos de calidad prioritarios: escalabilidad, disponibilidad, rendimiento, seguridad, mantenibilidad, confiabilidad y resiliencia.

- Estrategia de selección tecnológica: Se definió un conjunto de criterios rigurosos para la selección de tecnologías, privilegiando soluciones modernas y orientadas a la nube. Estas elecciones se encuentran con una alineación directa con los atributos de calidad priorizados y asegura la capacidad de crecimiento y evolución de la arquitectura.

- Estilo arquitectónico: Después de un análisis riguroso se eligió para el diseño una arquitectura basada en microservicios combinada con el estilo hexagonal, porque con esta elección se permite desacoplar la lógica de negocio de las tecnologías externas, mejorar la mantenibilidad, y facilitar la integración con APIS externas.

- Patrones y principios de diseño: Se realizo el análisis de principios como SOLID, para separación de responsabilidades y bajo acoplamiento entre los componentes que se encuentran diseñados en la arquitectura, así como patrones de adaptación, encapsulamiento e indirección que fortalecen la calidad del diseño. Se consideraron también los patrones Observer/Event Bus para facilitar la comunicación desacoplada entre servicios, permitiendo emitir eventos ante cambios en los datos sin que los productores conozcan a los consumidores, lo cual mejora la escalabilidad y extensibilidad si se llega a una implementación futura de la arquitectura. Asimismo, se contempló el uso del patrón Factory en los módulos para facilitar la creación dinámica de objetos. Finalmente, se incorporaron principios de arquitectura limpia y hexagonal, permitiendo la separación entre lógica de negocio y mecanismos externos, y se utilizó inyección de dependencias para facilitar el mantenimiento y pruebas conceptuales de los módulos.

- Decisiones de arquitectura: Se optó por una arquitectura basada en microservicios en lugar de una arquitectura monolítica debido a la necesidad de escalabilidad independiente que debe tener cada módulo. Dado a que se maneja diferentes fuentes de información clima, humedad, localización, cada módulo puede evolucionar, desplegarse y escalar de manera aislada. Esta decisión también permite una mejor mantenibilidad, ya que los cambios en un módulo, como por ejemplo integración SIG no afectan directamente a otros módulos. Se consideraron los riesgos asociados como la complejidad en la orquestación y se propuso el uso de Kubernetes como mitigación. Otra decisión ha sido la de integrar arquitectura hexagonal, se evaluaron diferentes estilos arquitectónicos en capas y basada en eventos seleccionando la arquitectura hexagonal por su capacidad de separar los detalles de la infraestructura del núcleo de dominio, lo que mejora la mantenibilidad. Esta decisión no solo fue técnica, sino estratégica, considerando las integraciones con APIS externas. este patrón fue seleccionado para aislar la lógica de negocio de las tecnologías externas, lo cual favorece el desacoplamiento, las pruebas unitarias y la posibilidad de adaptar a distintos entornos tecnológicos. En el módulo de gestión de datos, por ejemplo, los adaptadores externos manejan la lógica para consumir APIS externas, sin afectar el núcleo de reglas de negocio. Para mejorar el rendimiento en la consulta de datos frecuentes como zonas agrícolas y parámetros de clima, se eligió Redis como cache en memoria. Esta decisión permite reducir la latencia en la recuperación de datos y alivia la carga sobre la base de datos relacional. Se consideró su volatilidad como riesgo, mitigado mediante persistencia parcial y TTL. Se diseñó la arquitectura para operar en un entorno de nube pública debido

a su elasticidad, alta disponibilidad y capacidad de escalar automáticamente. Aunque el proyecto no contempló una implementación real, se eligieron componentes compatibles con plataformas como GCP.

3.3.1 Diseño de la arquitectura basado en atributos de calidad

El diseño arquitectónico propuesto se desarrolló con base en el método ADD (Attribute-Driven Design), que prioriza decisiones guiadas por atributos de calidad. Para este proyecto, dichos atributos fueron seleccionados mediante análisis de tradeoffs, considerando su impacto en arquitecturas distribuidas, escalables y orientadas al análisis en tiempo real.

A continuación, se describen los atributos de calidad priorizados y su relevancia específica dentro de la arquitectura junto con un árbol de decisiones:

- Escalabilidad: Se priorizo analizando que múltiples usuarios distribuidos geográficamente pueden llegar a necesitar información en tiempo real, por esta razón se diseñó una arquitectura modular permitiendo un crecimiento horizontal ya que se encuentra basada conceptualmente en microservicios desplegados en la nube. Se espera que, en el futuro con una correcta implementación, se puedan manejar cargas simultáneas de hasta 100 solicitudes por segundo a APIS externas sin afectar la latencia.

- Disponibilidad: Dado que los agricultores requieren acceso en tiempo real, incluso ante condiciones de conectividad variables, el acceso a los datos debe ser constante y tolerante a fallos. Por esta razón se definió un diseño con mecanismos y componentes distribuidos para evitar puntos únicos de falla.

- Rendimiento: Es un atributo muy importante ya que los datos deben procesarse y presentarse con baja latencia para ser útiles en tiempo real. El diseño de la arquitectura contempla desacoplamiento entre servicios y técnicas de optimización para asegurar tiempos de respuesta eficientes.

- Seguridad: Este atributo se priorizo ya que el acceso y la protección a los datos debe estar restringido como una necesidad crítica. El diseño incluye mecanismos conceptuales de autenticación y autorización, para mantener los datos confidenciales.

- Mantenibilidad: Con este atributo se busca que el diseño de la arquitectura en un futuro pueda adaptarse a cambios como nuevos módulos o distintas regiones. La separación de responsabilidades y uso de principios de diseño facilitan su evolución.

- Confiabilidad: Este atributo fue considerado para asegurar que el diseño arquitectónico propuesto pueda, en teoría, proporcionar datos consistentes. En esta etapa de diseño, la

confiabilidad se abordó mediante el concepto de transacciones ACID (atomicidad, consistencia, aislamiento y durabilidad), lo que evita inconsistencias y pérdida de información. (ver anexo 1)

- Resiliencia: Es atributo fue priorizado ya que en teoría pueden existir fallas de red, interrupciones de servicios o APIs externas, y lo que el diseño de manera conceptual busca es mitigar estas fallas mediante estrategias de recuperación, reintentos y tolerancia a errores.

3.3.2 Árbol de Decisiones Técnicas

- La siguiente estructura resume las decisiones tomadas en el diseño arquitectónico, alineadas con los atributos de calidad priorizados. Estas decisiones se basan en la metodología ADD, y reflejan cómo se eligieron tecnologías, estilos arquitectónicos y patrones para cumplir con los objetivos del proyecto.

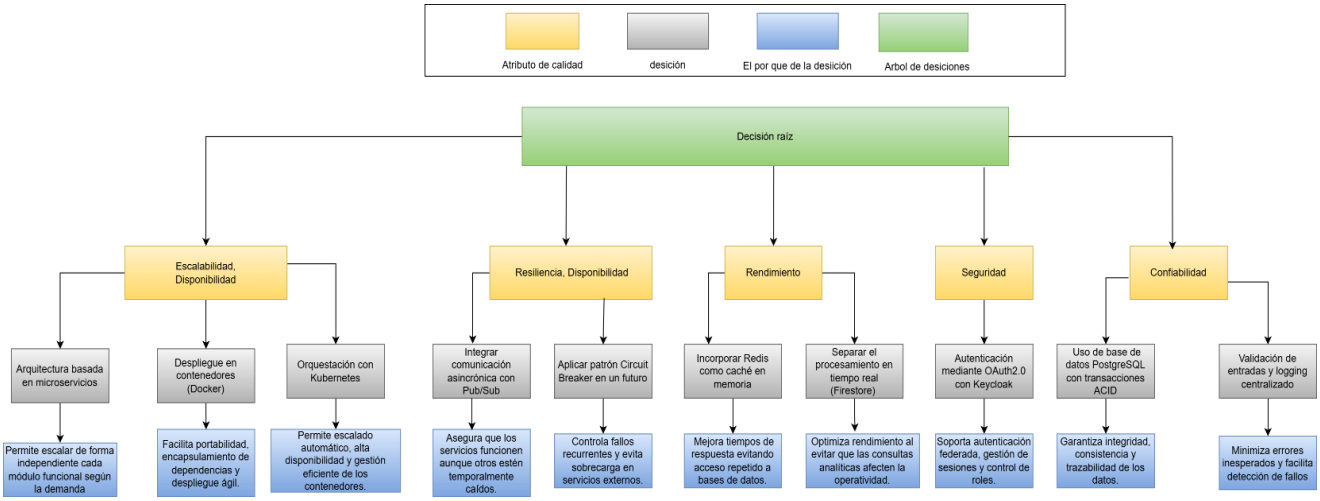


Ilustración 4 Árbol de decisiones

Fuente: Construcción Propia

3.3.3 Tecnologías para Front-end Interfaz de Usuario

En el diseño de la arquitectura, se contempla el uso del enfoque de microfrontends para la futura construcción de la interfaz de usuario, dado que este estilo arquitectónico permite abordar de manera efectiva algunos de los atributos de calidad priorizados como modularidad, escalabilidad y mantenibilidad. Cada microfrontend representa un módulo funcional independiente de la interfaz, lo cual permite que equipos de desarrollo distintos puedan desarrollar y mantener los siguientes microfrontends:

- Parametrización
- Administrador
- Usuario
- Gestion-Datos
- Histórico

En este diseño, se contempla el uso de React.js como biblioteca de JavaScript base para cada microfrontend, dada su capacidad de construir interfaces altamente reutilizables y reactivas. Para la orquestación de los distintos microfrontends, se propone emplear Single-SPA, que permite su integración dentro de una misma aplicación manteniendo independencia funcional. Asimismo, se considera el uso de plantillas como Material, con el fin de garantizar coherencia visual entre los distintos módulos y facilitar el mantenimiento de estilos a lo largo del sistema.

La arquitectura de microfrontends está alineada con el principio de independencia de despliegue. Esta característica permite que, ante una futura evolución de la arquitectura, se puedan actualizar microfrontends específicos sin afectar otros microfrontends, contribuyendo a la mantenibilidad y escalabilidad.

Además, este enfoque permite facilitar la gestión de roles y accesos diferenciados: por ejemplo, un agricultor y un técnico podrán ver diferentes vistas según sus permisos, lo que se facilita desde el diseño modular de la interfaz.

3.3.4 Tecnologías para Back-end

En el diseño arquitectónico se ha optado por una arquitectura de microservicios, que permite desarrollar y desplegar de forma independiente los módulos diseñados, tales como Parametrización, IntegracionSig, Gestión de datos. Con esta decisión se abordan los atributos de escalabilidad, mantenibilidad y recuperación ante fallos.

En un futuro para implementar los servicios del backend, se propone el uso del lenguaje de programación Node.js con el framework Express, ya que este lenguaje está formado con una tecnología ligera y eficiente que permite crear microservicios orientados a eventos,

esto es ideal ya que el entorno de la arquitectura es distribuida y con requerimientos de respuesta en tiempo real. Node.js es adecuado para gestionar múltiples solicitudes concurrentes, mientras que Express permite definir rutas, middlewares y controladores de manera modular.

La comunicación entre los microservicios está diseñada bajo el estilo RestFull, utilizando los métodos HTTP (GET, POST, PUT, DELETE) para facilitar la interoperabilidad entre servicios. Este enfoque permite mantener una estructura clara y desacoplada, compatible

con mecanismos de autenticación y facilita la integración entre componentes desarrollados en diferentes tecnologías.

3.3.5 Tecnologías para Base de Datos

En el diseño de la arquitectura se contempló el uso combinado de distintas tecnologías de bases de datos, cada una seleccionada según su función específica dentro del diseño los módulos, buscando optimizar conceptualmente atributos de calidad como rendimiento, disponibilidad, escalabilidad, confiabilidad, seguridad y recuperación ante fallos. A continuación, se detallan estas tecnologías junto con su propósito específico en el contexto del proyecto.

Nombre	Tipo	Función en el modulo
Google Firestore	No relacional	Se emplea en el módulo IntegraciónSig para almacenar datos operativos generados por las APIS externas en tiempo real, como temperatura, radiación solar, humedad del ambiente entre otros. En el módulo de Gestión de datos también se emplea para almacenar datos de configuración y de siembra creados por el usuario administrador. Firestore permite consultas rápidas y consistentes sin afectar el rendimiento.
PostgreSQL	Relacional	Se contempla para almacenar información estructurada y sensible que requiere integridad, como los perfiles de usuario, nombre de fincas, roles, y configuraciones del sistema. Este motor de base de datos soporta relaciones complejas y validación de reglas, siendo clave para la gestión centralizada de datos persistentes.
Google BigQuery	Relacional cooperativo	Se contempla para el procesamiento y análisis de grandes volúmenes de datos históricos, incluyendo reportes climáticos, tendencias de siembra o consultas pasadas. Esta tecnología forma parte del módulo histórico y respalda el análisis

		retrospectivo, entregando datos para reportes periódicamente.
Redis	No relacional	Se propone como almacenamiento en memoria para mejorar conceptualmente el rendimiento general de la arquitectura, facilitando el acceso rápido a resultados temporales, configuraciones de usuario y respuestas repetitivas. Se integra en los módulos de Parametrización y Usuario, reduciendo la carga en la base de datos PostgreSQL.

Tabla 2 Tecnologías para Gestión de Datos

3.3.6 Tecnologías para Procesamiento y Análisis

En el diseño arquitectónico se han considerado diversas tecnologías para cubrir necesidades de análisis, procesamiento de datos y orquestación de eventos. A continuación, se detallan cada una, especificando su función dentro de la arquitectura y su rol en la propuesta:

Nombre	Función en el diseño
Apigee	Plataforma de gestión de APIS contemplada para exponer de forma segura los servicios desarrollados en el backend. Su función en el diseño es actuar como una fachada, permitiendo aplicar políticas de autenticación, control de tráfico, trazabilidad y consumo, especialmente relevante si en el futuro se permite interoperabilidad con terceros como entidades gubernamentales o redes de productores agrícolas.
Cloud Pub/Sub	Este servicio de mensajería desacoplada permite diseñar flujos asincrónicos dentro de la arquitectura. En el diseño se contempla para la transmisión de eventos entre los módulos IntegracionSig y Gestion de datos, facilitando la resiliencia y escalabilidad de la arquitectura, dado que los mensajes pueden ser almacenados temporalmente y procesados de manera tolerante a fallos.
Cloud	Este Servicio de almacenamiento de objetos utilizado para guardar

Storage	archivos no estructurados como informes, resultados de análisis se utiliza en módulo de histórico para la entrega de archivos en tiempo real.
Cloud Load Balancer	Este componente es esencial en el diseño de la arquitectura ya que su función es la de distribuir el tráfico entrante entre múltiples instancias de microservicios. Esto permite mejorar el rendimiento general, asegurar la continuidad operativa ante fallos, y soportar cargas elevadas de peticiones.

Tabla 3 Tecnologías para Procesamiento y Análisis

3.3.7 Tácticas de Arquitectura

Para lograr una arquitectura sólida, mantenible y alineada con los objetivos del proyecto, se han utilizado diversas tácticas arquitectónicas. A continuación, se explican de manera clara y sencilla.

- **Aislamiento**: El principio base de la arquitectura hexagonal es mantener separada la lógica del negocio (ejemplo: las manos de los pies) de los elementos externos como bases de datos, APIS o pantallas. Esto permite que, si en el futuro se cambia una tecnología externa, como una base de datos, no haya que reescribir la lógica central. Esto se logra mediante el uso de puertos y adaptadores. Los servicios como ServicioGestionDatos y ServicioIntegracionSIG actúan como adaptadores que traducen la información que llega desde afuera. Los servicios ServicioGestionDatos, ServicioIntegracionSIG actúan como intermediarios ósea los adaptadores que traducen las solicitudes entrantes y salientes al formato que entiende el núcleo El núcleo en sí no conoce los detalles de la base de datos, las APIS externas o el frontend.

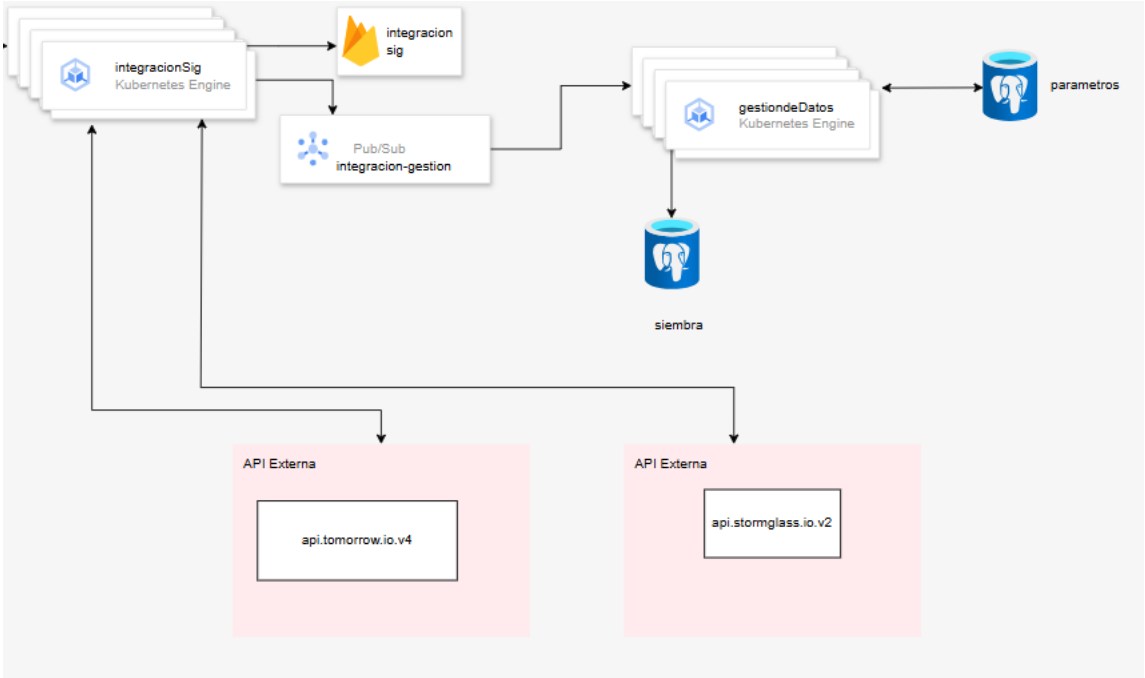


Ilustración 5 Aislamiento

Fuente: Construcción Propia

- Indirección: Esta táctica se utiliza para desacoplar componentes mediante la introducción de una capa intermedia. En la arquitectura hexagonal, los puertos actúan como esta capa de indirección donde los componentes no se comunican directamente entre sí, esto hace que el sistema sea más flexible. Ejemplo: Los servicios no hablan directamente con la base de datos o una API. Hablan con una interfaz (puerto), que es como una promesa de que ciertas acciones se pueden hacer (por ejemplo: guardar, buscar).

  Luego, otra parte del sistema (como un DAO o cliente de API) cumple esa promesa, implementando la acción real. Los servicios no interactúan directamente con la base de datos o las APIs. En su lugar, interactúan con interfaces (puertos) que definen las operaciones que se pueden realizar. Los Daos y los clientes de las APIs implementan estas interfaces, proporcionando la indirección necesaria.

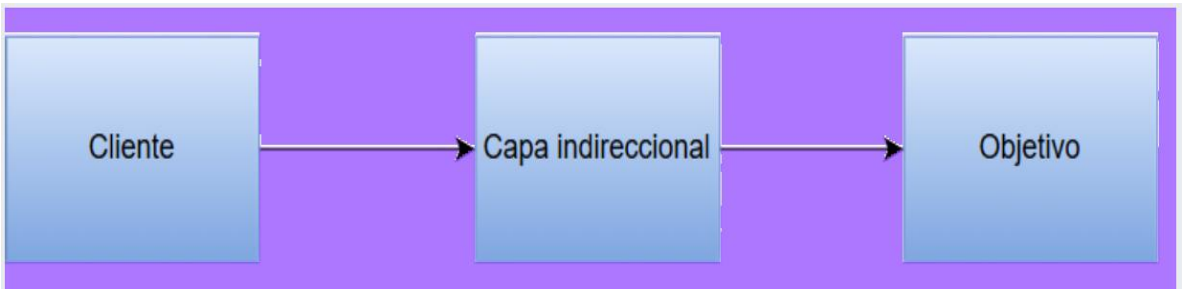


Ilustración 6 Indirección

Fuente: Construcción Propia

- Encapsulamiento: Esta táctica se centra en ocultar los detalles de implementación de un componente a otros componentes. En la arquitectura hexagonal, los adaptadores encapsulan la lógica específica de interacción con las tecnologías externas. Por ejemplo, SiembraDAO encapsula la lógica de acceso a la base de datos PostgreSQL. Los clientes de las APIS como (API- TomorrowClient la cual entrega parámetros del clima como temperatura, humedad etc. en tiempo real con solo enviar la ubicación de un usuario) y (weatherapi la cual entregue parámetros como la densidad, la presión atmosférica, la calidad del aire etc. en tiempo real con solo enviar la ubicación de un usuario) encapsulan la lógica de comunicación con las APIS externas. Esto permite cambiar la base de datos o las APIS sin afectar al núcleo.

Ilustración 7 Encapsulamiento

Fuente: Construcción Propia

- Abstracción: La abstracción es una táctica que permite simplificar la complejidad al ocultar los detalles internos y mostrar solo lo esencial de un componente o funcionalidad. En arquitectura de software, se utiliza para representar interacciones o comportamientos de forma general, sin necesidad de conocer su implementación específica. Por ejemplo, en este proyecto se define una interfaz llamada SiembraRepository que representa las operaciones básicas relacionadas con las siembras, como obtener, guardar, consultar o editar información. Esta interfaz actúa como una capa intermedia entre el servicio y la base de datos. Gracias a esta abstracción, el servicio puede interactuar con SiembraRepository sin preocuparse por si los datos se almacenan en PostgreSQL. Lo importante es que sabe qué operaciones puede realizar, no cómo están implementadas.

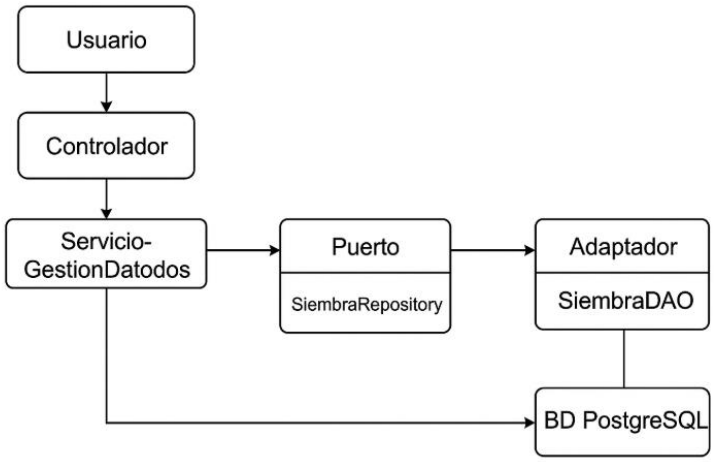


Ilustración 8 Abstracción Fuente: Construcción Propia

- Adaptación: Esta táctica se centra en convertir la interfaz de un componente a la interfaz que otro componente espera. En la arquitectura hexagonal, los adaptadores cumplen esta función.

En este proyecto Los adaptadores traducen los datos entre el formato del núcleo y el formato de las tecnologías externas. Por ejemplo, el componente usuario traduce lo que el sistema necesita a un formato que entiende la base de datos y viceversa.

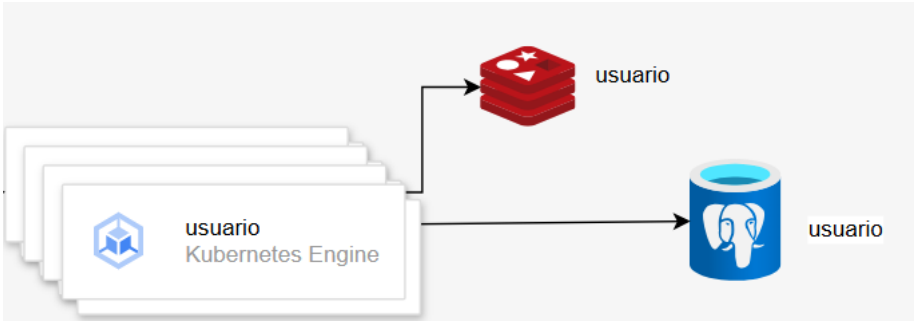


Ilustración 9 Adaptación

Fuente: Construcción Propia

- Recursos compartidos: Para no duplicar recursos innecesariamente, se comparten clientes de ingreso común para acceso rápido a datos en memoria, datos no estructurados en tiempo real, el envío y recepción de mensajes entre módulos.
  Esto mejora el rendimiento y reduce el uso innecesario de memoria para el proyecto ya que muchos usuarios pueden consultar por ejemplo en que áreas de sus fincas es factible cultivar aguacate Hass.

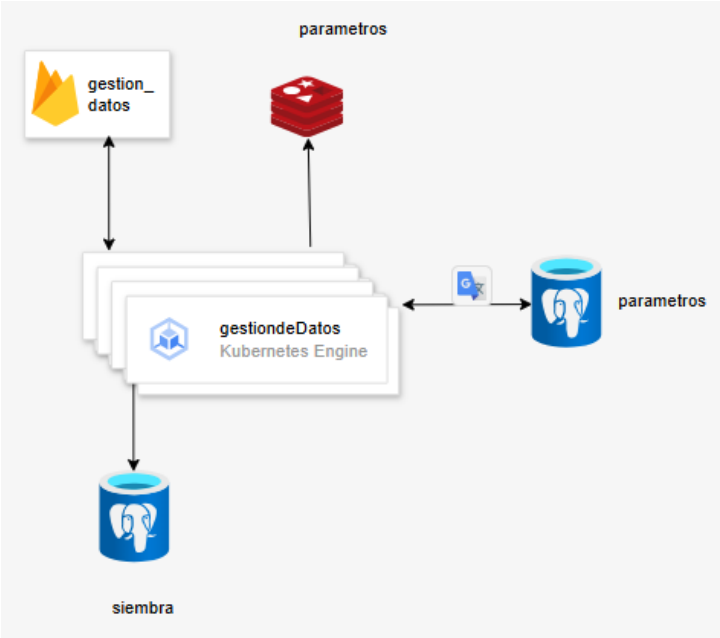


Ilustración 10 Recursos Compartidos

Fuente: Construcción Propia

- Mediador: En vez de que un componente hable directamente con otro, se puede usar un "intermediario" que reciba y reparta los mensajes. Eso hace el sistema más flexible y tolerante a fallos, un ejemplo es Google Cloud Pub/Sub actúa como mediador. Si el componente integracionSig necesita mandar un mensaje al componente gestionDatos, lo hace a través de Pub/Sub (integración-gestion). Si gestionDatos está caído, el mensaje se guarda y se reintenta más tarde.

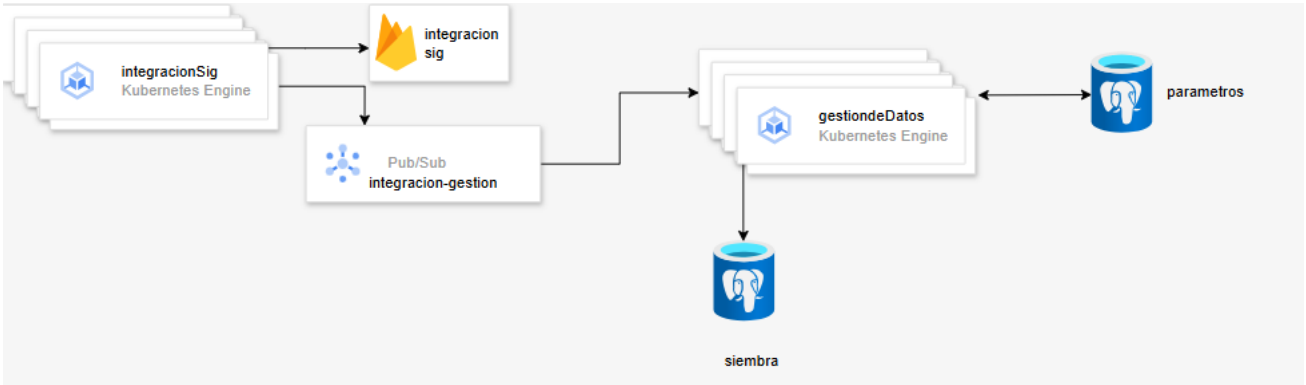


Ilustración 11 Mediador

Fuente: Construcción Propia

- Rate Limiting: Se establece un límite de peticiones a cada módulo para evitar sobrecarga en toda la arquitectura.

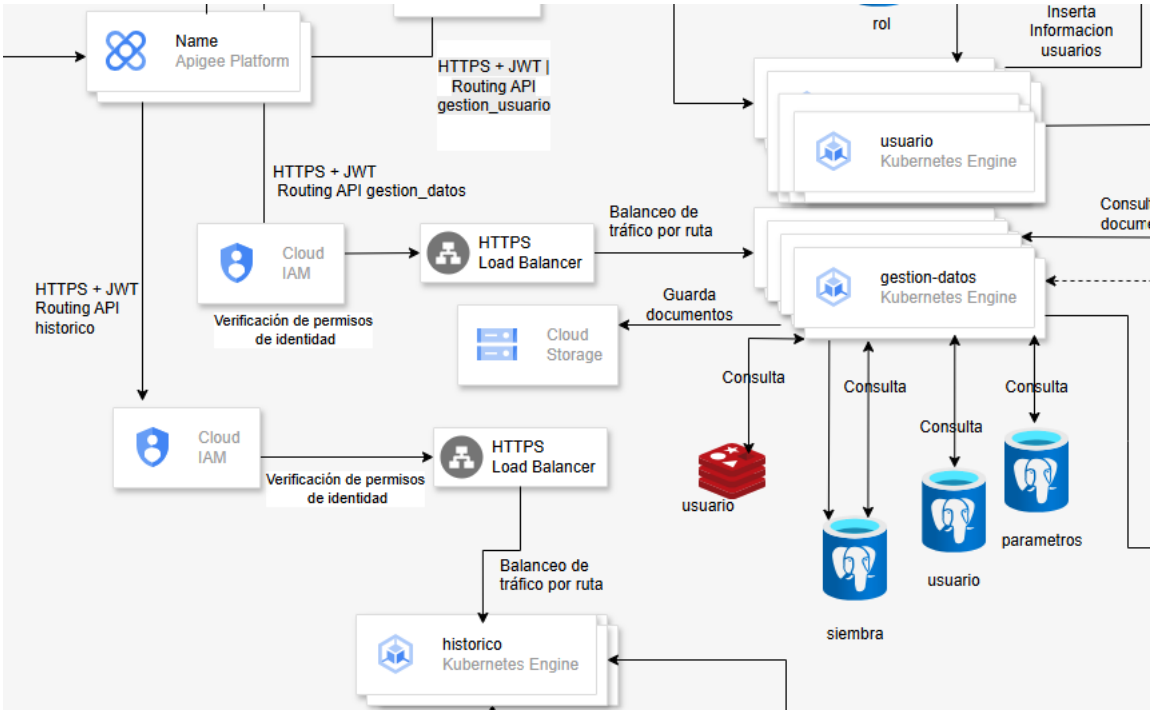


Ilustración 12 Rate Limiting

Fuente: Construcción Propia

- Expiración de Tokens: todos Los tokens en la arquitectura tienen una expiración controlada esto se evidencia en el componente de keycloak, después de este tiempo, el token deja de ser reconocido por la arquitectura y el usuario debe autenticarse nuevamente para obtener uno nuevo. Este mecanismo es crucial para la seguridad, ya que limita la ventana de tiempo en la que un token robado o comprometido puede ser utilizado.

```
const axios = require('axios');

let accessToken = null;
let tokenExpiration = null;

// Simulación de obtención inicial del token
async function fetchToken() {
  const response = await axios.post('https://api.externalsource.com/auth/token', {
    client_id: 'your-client-id',
    client_secret: 'your-client-secret',
    grant_type: 'client_credentials'
  });

  accessToken = response.data.access_token;
  const expiresIn = response.data.expires_in; // segundos

  // Se  Establece fecha de expiración
  tokenExpiration = Date.now() + (expiresIn - 60) * 1000; // renovar 1 minuto antes
}

// Verifica si el token está expirado
async function ensureValidToken() {
  if (!accessToken || Date.now() >= tokenExpiration) {
    console.log('Token expirado o no disponible. Renovando...');
    await fetchToken();
  }
}

// Hace una solicitud usando el token válido
async function fetchClimateData() {
  await ensureValidToken();
```

Ilustración 13 codigo token

Fuente: Construcción Propia

- Reintentos controlados: con el componente de la arquitectura de pub sub se repite automáticamente una operación fallida, como una llamada al servicio gestión de datos, con el objetivo de que esta tenga éxito en un intento posterior. Esta táctica se contempló ya que la arquitectura está enfocada a sistemas distribuidos donde las fallas son inevitables debido a problemas de red, sobrecargas o errores transitorios.

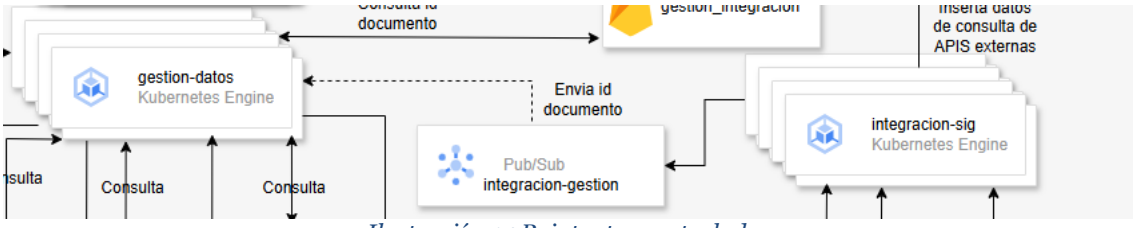


Ilustración 14 Reintentos controlados

Fuente: Construcción Propia

- Event Bus / Observer: Como evolución futura se propone el uso de un patrón Observer o Event Bus para que, por ejemplo, el módulo de gestión de datos notifique eventos cuando detecte condiciones críticas como alerta de sequía, permitiendo que otros componentes como el de parametrización, y usuario reaccionen sin acoplamientos directos.

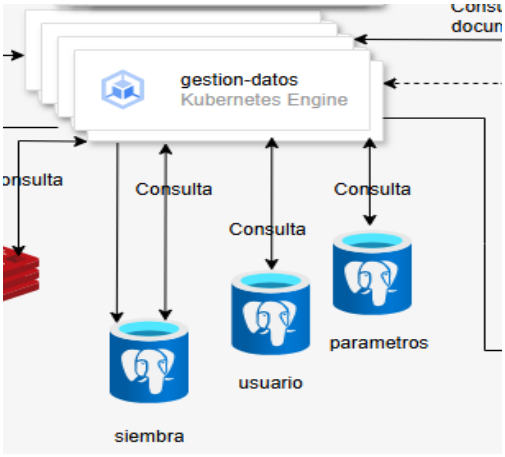


Ilustración 15 Event bus gestión datos

Fuente: Construcción Propia

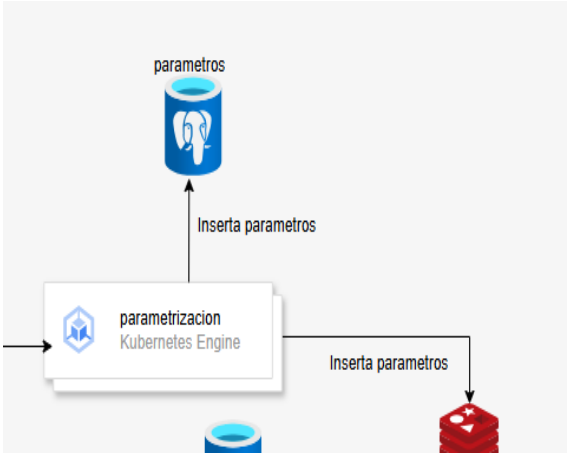


Ilustración 16 Parametrización

Fuente: Construcción Propia

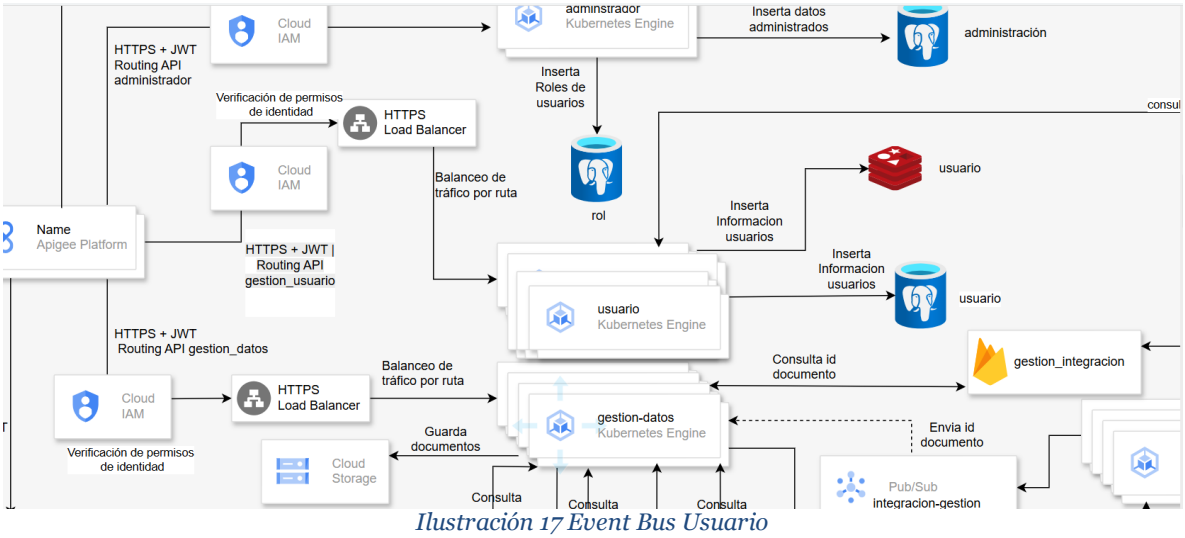


Ilustración 17 Event Bus Usuario

Fuente: Construcción Propia

- Fail-Fast: en todos los módulos se detectan errores y se lanzan excepciones que permite detectar una o varias condiciones anómalas en la arquitectura un ejemplo claro es el de las peticiones que se realizan alas APIS externas, si las APIS no responden las peticiones se abortan rápidamente.

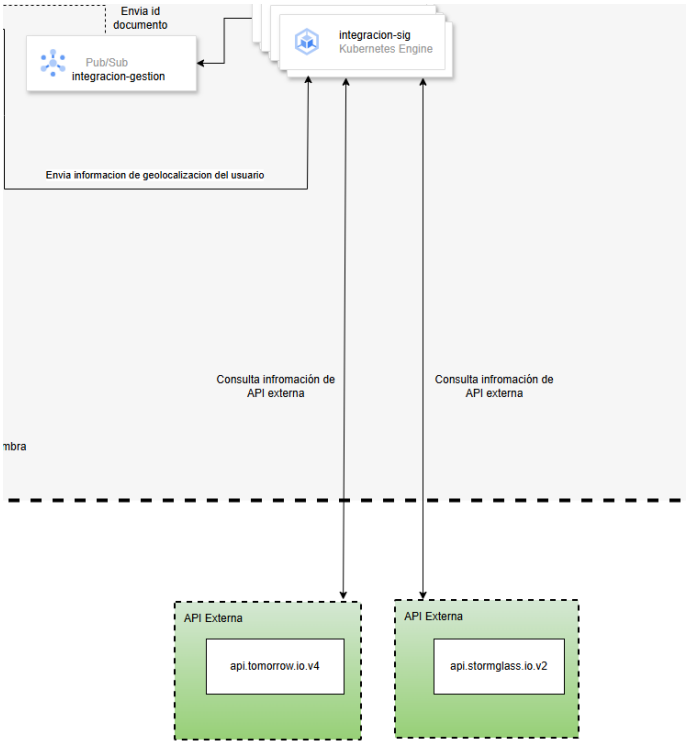


Ilustración 18 Fail fast

Fuente: Construcción Propia

3.3.8 Diseño módulo Parametrización

Objetivo del módulo: Permitir la configuración dinámica de los criterios y las condiciones necesarias para evaluar la viabilidad de la siembra de aguacate Hass. Este módulo forma parte del núcleo de la arquitectura y proporciona los parámetros que alimentan el componente de gestión de datos. La determinación de los parámetros a configurar en este módulo se basa en una revisión de literatura técnica, guías agroclimáticas para el cultivo de aguacate Hass. Se consideraron tanto factores climáticos como relacionados con el suelo, sanitarios y ambientales que inciden directamente en el proceso de la siembra. A continuación, se agrupan y justifican los principales criterios:

- **Condiciones climáticas (temperatura, humedad, precipitación, viento, radiación solar)**: Son parámetros críticos que afectan el desarrollo fisiológico del aguacate. Se basan en rangos recomendados por entidades como el ICA (Instituto Colombiano Agropecuario) y estudios del Agrosavia. (agrosavia, 2010)
- **Características del suelo (pH, humedad, altitud)**: Estas condiciones son esenciales para la absorción de nutrientes, desarrollo radicular y resistencia a enfermedades del árbol. Los valores se derivan de estudios de suelos en zonas productoras del país. (agrosavia, 2010)
- **Factores sanitarios (plagas, enfermedades)**: La presencia de enfermedades y plagas que pueden hacer inviable la siembra si no se controlan adecuadamente. Este criterio se incluyó tras analizar estudios de impacto fitosanitario en cultivos comerciales. (agrosavia, 2010)
- **Factores sociales y de infraestructura (ubicación, fuente hídrica, densidad de siembra)**: Estos permiten ajustar la evaluación a las condiciones específicas de cada predio, garantizando que los recursos estén disponibles para el manejo eficiente de la siembra. (agrosavia, 2010)

Estos parámetros no son arbitrarios: son el resultado de analizar qué condiciones determinan el éxito o fracaso de un cultivo de aguacate, especialmente en la región andina de Colombia. Además, la configuración flexible permite adaptar el sistema a diferentes regiones productoras o actualizar los criterios ante nuevas condiciones climáticas o descubrimientos técnicos. (agrosavia, 2020)

Estructura funcional y flujo del módulo:

El módulo de parametrización en su diseño se compone de los siguientes elementos:

1. **Interfaz de Usuario (Microfrontend):**

- Formularios para ingresar o modificar parámetros.

2. **Microservicio de Parametrización (Backend):**

- API que expone los métodos para guardar, actualizar y consultar parámetros.
- Encapsula la lógica de negocio y garantiza la integridad de los datos.

3. **Integración Redis**

- Redis se utiliza para almacenar en memoria los parámetros de uso frecuente o recientemente consultados, con el fin de optimizar tiempos de respuesta en las interfaces del usuario.
- Esta estrategia también reduce la carga sobre la base de datos de PostgreSQL.

4. **Diagrama de flujo**

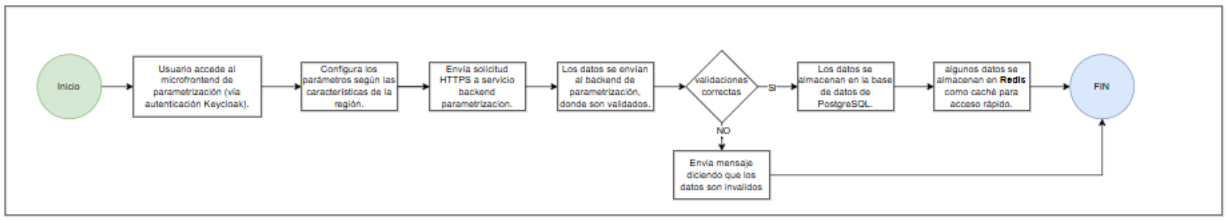


Ilustración 19 Diagrama de flujo parametrización

Fuente: Construcción Propia

3.3.9 Diseño módulo de gestión de datos

Objetivo del módulo: Procesar y analizar información basada en consultas realizadas a la base de datos PostgreSQL y Firestore, recepción de datos geoespaciales y climáticos de la API de integración SIG, para determinar si una zona es adecuada para sembrar aguacate Hass, y que cantidad de árboles se podrían sembrar.

Funcionalidades Clave

- Recibe la localización del usuario.
- Envía la localización al servicio IntegracionSig
- Consulta la base de datos PostgreSQL para obtener los datos parametrizados.
- Consulta la información en el base de datos de Redis.
- Consulta la información en el base de datos de Firestore.
- Compara los datos recibidos con los parámetros definidos en la base de datos de PostgreSQL (ejemplo. Tipo de suelo, precipitación mínima, temperatura ideal).
- Escribe archivos en Cloud Storage.
- Realiza cálculos para determinar la cantidad de árboles que se pueden sembrar. Para calcular la cantidad de árboles de aguacate Hass que pueden sembrarse en una zona determinada, el sistema aplica una fórmula basada en la densidad de siembra recomendada por prácticas agronómicas. La fórmula utilizada es:

Cantidad de árboles = Área Útil del terreno(m²) / Área requerida del Árbol (m²)

- El valor del área requerida por árbol depende del marco de siembra recomendado, que puede variar entre 6x6 metros (36 m² por árbol), 7x7 metros (49 m² por árbol), entre otros, según las condiciones del terreno. Este cálculo permite entregar al agricultor una estimación precisa de cuántos árboles puede sembrar. (agroptima, 2021)

- Realiza cálculos para determinar la viabilidad de la siembra esto se genera tras comparar los datos geoespaciales y climáticos actuales alojados en Firestore con los parámetros configurados previamente (humedad, precipitación, pH, temperatura, etc.). La viabilidad de siembra se calcula en función del porcentaje de cumplimiento de estos parámetros. La fórmula utilizada es:

  Viabilidad (%) = (Parámetros que cumplen condiciones / Total parámetros evaluados) X 100

  Esta fórmula es una adaptación lógica del concepto de cumplimiento de criterios ya que es una forma objetiva de decir que tan buena es una zona para sembrar sin depender de juicios subjetivos

- Entrega al microfronted gestión de datos información clave para el agricultor donde se incluye, viabilidad de la siembra, cantidad de árboles que se pueden plantar. Este es un ejemplo de la representación del reporte en formato JSON:


```
{
  "zona": "Lote 4 - Vereda San Luis",
  "viabilidad_siembra": "Alta",
  "porcentaje_viabilidad": 85,
  "area_total": 10000,
  "area_util": 8500,
  "marco_siembra": "6x6",
  "cantidad_arboles_estimados": 236,
  "fecha_evaluacion": "2025-04-18"
}
```


Ilustración 20 Representación de reporte en formato JSON

Fuente: Construcción Propia

- Guarda la información procesada en la tabla de siembra. (ver Anexo 2)

- Diagrama de Interacción del Módulo

El siguiente diagrama representa la interacción lógica del módulo de gestión de datos con los diferentes componentes del sistema en el diseño arquitectónico. Este módulo actúa como núcleo funcional para la consolidación y procesamiento de información relacionada con la siembra de aguacate Hass.

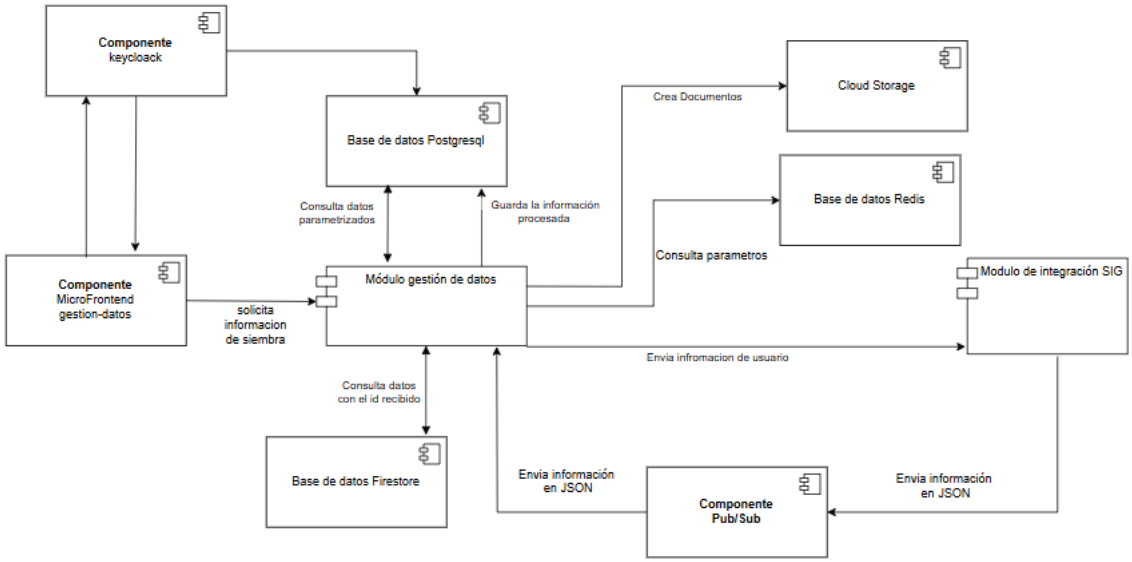


Ilustración 21 Diagrama de Interacción

Fuente: Construcción Propia

A continuación, se presentan ejemplos representativos del diccionario de datos correspondiente al modelo entidad relación de la base de datos del diseño de la arquitectura. El diccionario completo puede consultarse en el Anexo 1.

Tabla: usuario

Campo	Tipo	Descripción	Restricciones
id_usuario	Serial	Identificador único de usuario	PRIMARY KEY
nombres	Varchar	Nombres del usuario	NOT NULL
apellidos	Varchar	Apellidos del usuario	NOT NULL
correo	Varchar	Correo electrónico del usuario	NOT NULL
contrasena	Varchar	Contraseña del usuario	NOT NULL
fecha_creacion	Timestamp	Fecha de creación del registro	NOT NULL

Tabla: rol

Campo	Tipo	Descripción	Restricciones
id_rol	Serial	Identificador del rol	PRIMARY KEY
descripcion	Varchar	Nombre o descripción del rol	NOT NULL
fecha_creacion	Timestamp	Fecha de creación del rol	NOT NULL
id_usuario	Integer	Usuario que creó el rol	FK usuario
fecha_actualizacion	Timestamp	Última modificación	NOT NULL

- Diagrama de flujo

El diagrama de flujo representa el recorrido lógico de la información dentro del módulo de gestión de datos, diseñado para apoyar la toma de decisiones sobre la siembra de aguacate Hass. El flujo describe cómo, tras una solicitud del usuario autenticado, se recopilan parámetros y datos operacionales desde diferentes fuentes, los procesa y genera respuestas estructuradas para su visualización y análisis. (ver anexo 3)

- Datos que se muestran al usuario

A continuación, se presenta una representación gráfica de la información que recibe el usuario al realizar una consulta de datos climáticos y geográficos sobre un predio, entendido en este caso como una o varias hectáreas de fincas ubicadas en la región andina de Colombia. Esta visualización incluye datos relevantes sobre las características del terreno, las condiciones climáticas y los factores técnicos asociados a la siembra, constituyéndose en una herramienta fundamental para orientar al agricultor en la toma de decisiones relacionadas con la siembra de aguacate Hass. La representación no solo facilita la interpretación de la información, sino que también evidencia de manera práctica cómo la arquitectura propuesta respalda el objetivo del proyecto, garantizando que se aporte valor real en el proceso de siembra de aguacate Hass.

Campo	Descripción
Ubicación	Coordenadas o nombre del lugar analizado
Viabilidad de siembra	Resultado del análisis (Apta / No apta / Parcialmente apta)
Motivo de viabilidad	Breve explicación del resultado (Ej. "Temperatura fuera del rango ideal")
Cantidad de árboles sugerida	Número calculado de árboles por hectárea en base al análisis
Humedad ambiente	Rango promedio durante el periodo evaluado
Humedad del suelo	Valor promedio registrado o proyectado
Temperatura	Rango de temperatura mínima y máxima esperada
Precipitación	Nivel de lluvias promedio (mm)
Radiación solar	Nivel promedio de radiación (W/m²)
Velocidad del viento	Rango de velocidad registrado (km/h)
pH del suelo	Nivel de acidez/alcalinidad promedio
Enfermedades detectadas	Enfermedades presentes en la zona (si las hay)
Plagas detectadas	Plagas reportadas por sistemas SIG o bases comunitarias
Acceso a fuente hídrica	Si / No – según cercanía a fuentes de agua
Altitud	Altura sobre el nivel del mar en metros
Valor óptimo por variable	Comparativa de los valores obtenidos frente a los rangos recomendados
Fecha del análisis	Fecha en que se generó el reporte
Recomendaciones técnicas	Consejos automáticos según el análisis (Ejemplo ajustar densidad de siembra)

Ilustración 22 Representación gráfica de los datos que se muestran al usuario

Fuente: Construcción Propia

3.3.10 Diseño de módulo integración SIG

Objetivo del módulo: Consultar datos climáticos relevantes desde las fuentes WeatherAPI y TomorrowClient (APIS externas) para su análisis.

Para lograr una gestión eficiente y precisa de la información climática y geoespacial, se optó por integrar dos APIS complementarias: TomorrowClient (Tomorrow.io) y WeatherAPI. TomorrowClient se seleccionó por su alta granularidad y especialización en variables clave para la agricultura, como la humedad del suelo, la radiación solar y modelos predictivos de enfermedades, que resultan esenciales para determinar la viabilidad de siembra y realizar cálculos detallados en el módulo de gestión de datos. Por otro lado, WeatherAPI se utiliza para proporcionar información climática general y rápida, como temperatura, precipitaciones o condiciones del día, ideal para alimentar reportes simplificados y notificaciones al agricultor. Esta combinación estratégica permite balancear precisión técnica con eficiencia operativa, garantizando así tanto la calidad analítica como la accesibilidad de la información para todos los usuarios.

Funcionalidades Clave

- Recibe la localización del usuario.
- Consulta con esa localización los datos geoespaciales y climáticos de la zona.
- Guarda los datos obtenidos en la base de datos Firesbase.
- envía el id documento a un Pub Sub para que el módulo de gestión de datos consulte la información en el Firesbase.

- Diagrama de Interacción del Módulo

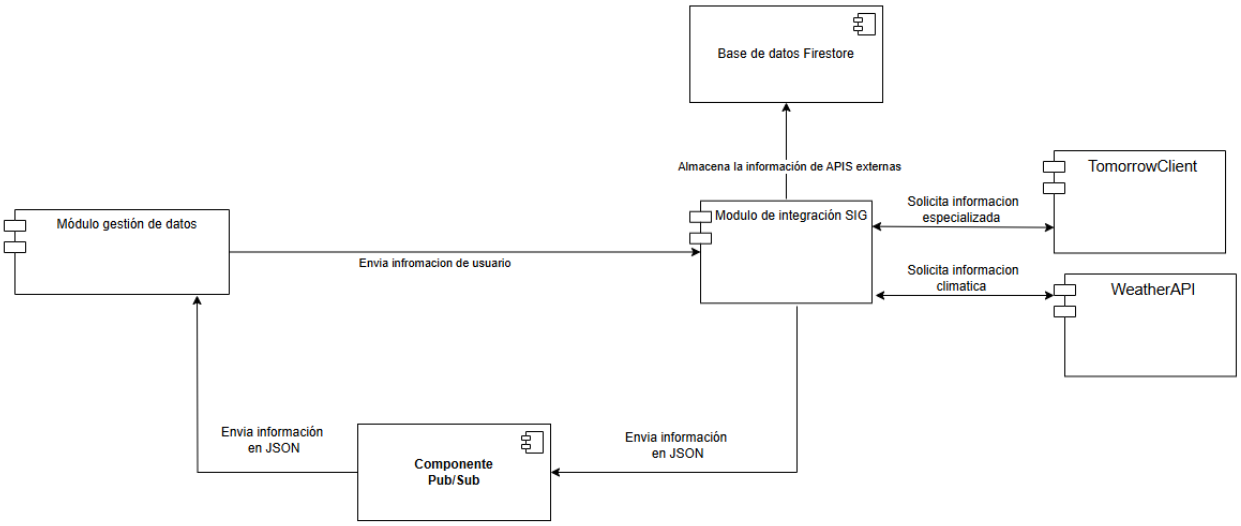


Ilustración 23 Diagrama de módulo de integración SIG

Fuente: Construcción Propia

- Diagrama de flujo

El siguiente diagrama de flujo describe el comportamiento lógico del módulo de integración SIG, cuya función principal es la recolección y gestión de información proveniente de APIS externas. Este flujo permite enriquecer los datos operacionales con información climática y topográfica, la cual es esencial para la evaluación de zonas aptas para el cultivo de aguacate Hass.

El proceso comienza con una solicitud del servicio de gestión de datos, que es recibida y procesada por el servicio de integraciónSIG. Este último consulta a las APIS externas y

maneja su respuesta con tolerancia a fallos, aplicando el patrón circuit breaker para mitigar errores. Posteriormente, la información recuperada es almacenada en Firestore y distribuida de manera asincrónica a través de Pub/Sub. El flujo contempla la validación del envío del mensaje y la configuración de reintentos en caso de fallos, asegurando así confiabilidad y resiliencia en el intercambio de datos entre módulos.

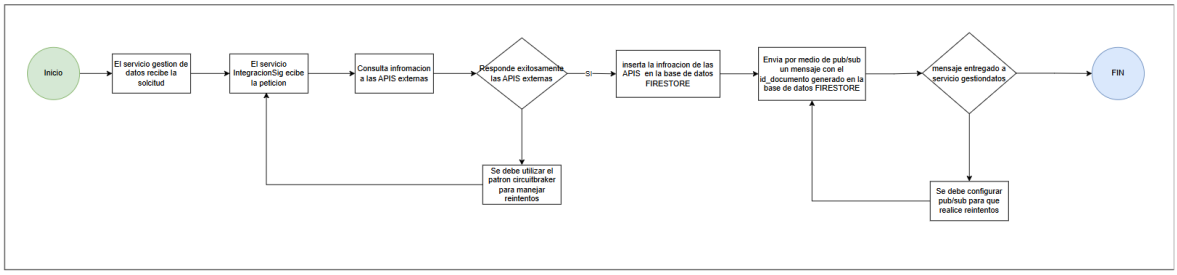


Ilustración 24 Diagrama de flujo modulo integración sig

Construcción propia

3.3.11 Diseño Funcional

Interfaz amigable y accesible que permite a los usuarios gestionar parámetros clave utilizando formularios dinámicos que validen los datos antes de enviarlos al backend.

Funcionalidades: Agregar, consultar editar y eliminar registros de la base de datos.

Implementación técnica: Servicio RestFull con los siguientes end points.

Método	EndPoint	Acción
GET	parámetro	Listar todos los parámetros de un registro.
PUT	parámetro	Actualizar registros
POST		Crear nuevo registro
DELETE	parámetro	Eliminar registro

Tabla 4 Métodos Rest Full

Seguridad y permisos: Restringir el acceso al módulo de parametrización solo a usuarios con roles específicos.

- **Rol Administrador:** Puede crear, actualizar y eliminar registros.

- **Rol Usuario Experto:** Usuario con permisos avanzados de consulta, pero sin acceso completo como administrador.

Caché: Uso Redis para mejorar la eficiencia de acceso a La información. En lugar de hacer una consulta a la base de datos cada vez que un usuario necesita información, el backend puede guardar temporalmente los resultados en Redis, con esto se reduce la carga de la base de datos y mejora los tiempos de respuesta.

Creación de tabla: Debe contener campos básicos y configurables. Ejemplo de esquema.

Campo	Tipo	Descripción
id	serial	Identificador único
nombre	VARCHAR	Nombre del parámetro (ejemplo: Humedad mínima")
valor_min	DECIMAL	Valor mínimo permitido
valor_max	DECIMAL	Valor máximo permitido
fecha_creacion	TIMESTAMP	Fecha de creación del registro
fecha_actualizacion	TIMESTAMP	Última vez que fue modificado
usuario	VARCHAR	Usuario que creó o actualizó el parámetro

Tabla 5 Modelo de tabla en base de datos

3.3.10 Modelo de Dominio

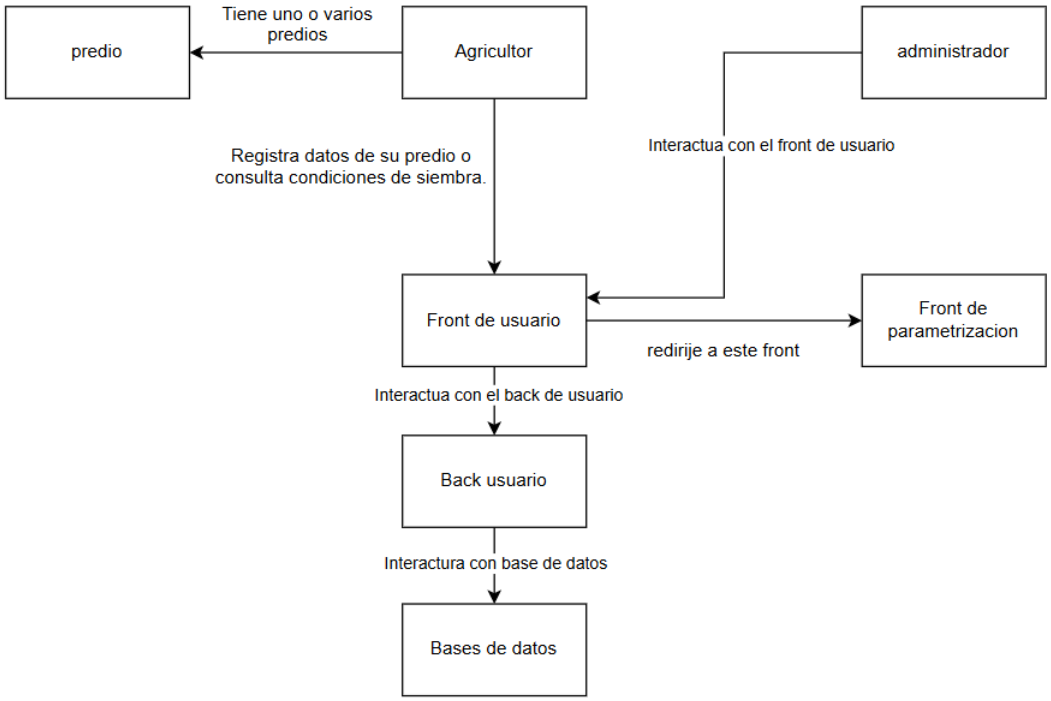


Ilustración 25 Modelo de Dominios

Construcción propia

3.4 Diagramas Arquitectura de Software

En el desarrollo de este proyecto, se consideró fundamental representar visualmente el diseño de la arquitectura para facilitar su comprensión y análisis por parte de los diferentes actores involucrados. Aunque las metodologías ágiles tienden a minimizar la documentación formal, se reconoce que los diagramas bien estructurados siguen siendo una herramienta valiosa para comunicar la visión técnica y funcional de un sistema.

Por esta razón, se optó por utilizar el modelo C4 (Contexto, Contenedores, Componentes y Código), ya que permite describir la arquitectura de forma jerárquica y en distintos niveles de detalle. Esta aproximación facilita la comunicación efectiva con diversos perfiles, como directivos, arquitectos, desarrolladores, testers e ingenieros de infraestructura. Cada nivel del modelo C4 aborda un grado de abstracción distinto, lo que permite adaptar la representación arquitectónica a las necesidades específicas de cada audiencia.

El uso del modelo C4 se justifica por su claridad, su enfoque visual coherente y su capacidad para mostrar tanto la estructura general como los detalles técnicos sin perder la coherencia global del sistema. Para evitar confusiones, todos los diagramas incluyen

leyendas y descripciones que explican las notaciones empleadas, lo cual mejora su interpretación y reduce el riesgo de ambigüedad.

3.4.1 Diagrama de Contexto

El diagrama ilustra una propuesta conceptual de la interacción entre los usuarios, las API externas y la arquitectura diseñada. Es importante aclarar que este esquema no corresponde a un sistema implementado, sino a una representación de alto nivel que facilita la comprensión de cómo se integran los diferentes componentes. Para una visualización detallada del diagrama. (Ver anexo 4)

3.4.2 Diagrama de Contenedores

Los siguientes diagramas de contenedores muestran la interacción que existe entre los artefactos internos y externos del sistema.

- **Diagrama de contenedores del módulo Usuarios**

El módulo de Usuarios constituye uno de los componentes de la arquitectura, dado que gestiona los procesos de autenticación, y direccionamiento a otros módulos. Para comprender de manera estructurada su funcionamiento, se presenta el diagrama de contenedores, el cual describe cómo se organiza este módulo en términos de aplicaciones, servicios, bases de datos y componentes externos con los que interactúa. Esta representación permite identificar claramente los límites de cada contenedor, sus responsabilidades y las relaciones que se establecen entre ellos, brindando una visión de alto nivel que facilita tanto la comprensión del diseño arquitectónico como su futura implementación y mantenimiento. (Ver anexo 5)

- **Diagrama de contenedores del módulo parametrización**

El módulo de Parametrización es el encargado de gestionar la configuración dinámica de la arquitectura, permitiendo ajustar reglas, valores y condiciones sin necesidad de modificar directamente el código fuente. Su objetivo principal es ofrecer flexibilidad y adaptabilidad, de manera que la solución tecnológica pueda responder con agilidad a cambios en los procesos o en las necesidades de los usuarios. El diagrama de contenedores de este módulo muestra la organización de sus componentes principales, la interacción con bases de datos internas y externas como Redis. Esta representación facilita comprender cómo se estructura el módulo dentro de la arquitectura global y resalta su importancia en la administración eficiente y escalable. (Ver anexo 6)

- **Diagrama de contenedores del módulo gestión de datos**

El módulo de Gestión de Datos se encarga de centralizar, almacenar y administrar la información proveniente de las diferentes fuentes de la arquitectura, garantizando su integridad, disponibilidad y consistencia. Este módulo permite realizar operaciones de consulta, actualización y validación de datos, sirviendo como núcleo para la toma de decisiones y el soporte a otros módulos funcionales. El diagrama de contenedores asociado describe la organización de los componentes principales, los mecanismos de comunicación con las bases de datos y servicios internos, así como las interacciones con los módulos de negocio que dependen de la información procesada. Esta representación facilita comprender el rol estratégico del módulo dentro de la arquitectura global y evidencia su importancia en la eficiencia, escalabilidad y confiabilidad de la arquitectura. (Ver anexo 7)

- **Diagrama de contenedores del módulo integración SIG**

El diagrama del módulo de integración SIG muestra cómo la arquitectura propuesta incorpora herramientas y APIS externas para el análisis, consulta y visualización de datos territoriales. Este módulo permite relacionar información alfanumérica con la ubicación geográfica de los predios, posibilitando la generación de capas de información que apoyan la toma de decisiones. La integración se convierte en un componente clave para vincular datos técnicos y espaciales, garantizando que los usuarios cuenten con una visión más completa y contextualizada del entorno en el cual operan. El diagrama evidencia la interacción entre la arquitectura y los servicios externos especializados. (Ver anexo 8)

El diagrama de contenedores representa de manera estructurada cómo se distribuyen los componentes principales de la arquitectura en distintos niveles funcionales. A través de esta vista se visualizan claramente los microservicios, las interfaces de usuario (microfrontends), las bases de datos y los servicios externos, así como las relaciones entre ellos. Esta organización respalda la escalabilidad, mantenibilidad y seguridad del diseño, al tiempo que garantiza una separación clara de responsabilidades, coherente con los atributos de calidad priorizados en el proyecto.

3.4.3 Diagrama de Componentes

El siguiente diagrama de componentes muestra una vista más detallada de la integración de cada componente de la arquitectura.

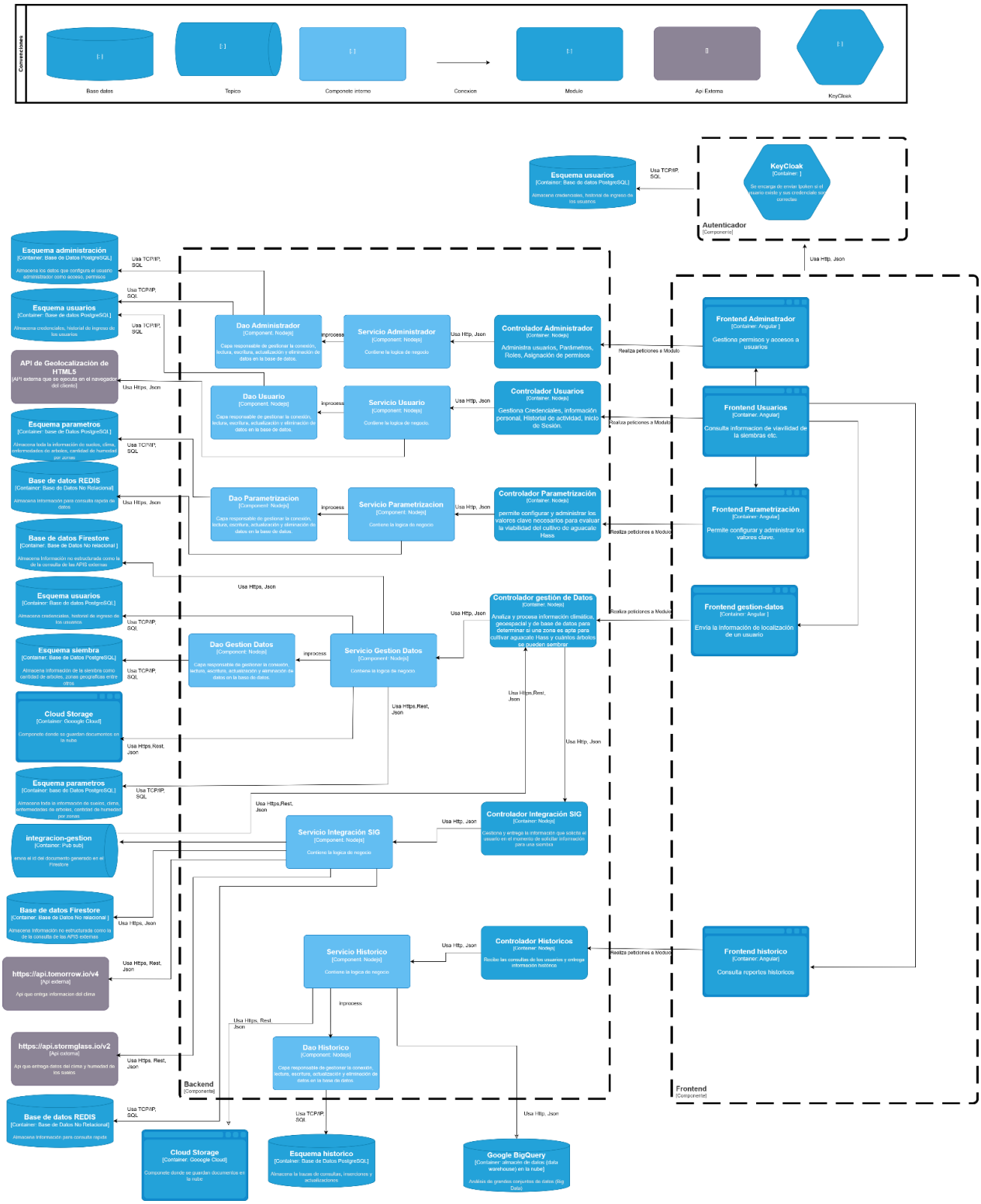


Ilustración 26 Diagrama de componentes

Fuente: Construcción Propia

3.4.4 Diagrama dinámico de componentes en la nube

El diagrama dinámico muestra una visión conceptual del comportamiento de la arquitectura diseñada frente a un escenario de interacción típica, como una consulta o acción desde la interfaz de usuario. En él se representa el flujo entre microfrontends, servicios backend, bases de datos y componentes de integración de la nube.

El diseño contempla la comunicación segura entre los módulos mediante el uso de JWT, pasarelas de APIS gestionadas como la de Apigee, y mecanismos de control de acceso como IAM. Adicionalmente, se incluyen elementos para asegurar la escalabilidad, resiliencia y eficiencia en el manejo de datos.

El flujo también evidencia cómo se consulta información a APIS externas, Tomorrow.io y Stormglass.io desde el módulo de integraciónSIG, y cómo esa información es usada por los distintos componentes que conforman la arquitectura.

Este diagrama no representa una implementación operativa, sino una visión lógica del comportamiento esperado de los componentes, de acuerdo con los atributos de calidad priorizados durante el diseño arquitectónico. (Ver anexo 9)

3.4.4 Diagrama de Clases

El siguiente diagrama de clases muestra una vista detallada de la estructura estática, mostrando las clases, sus atributos, métodos y las relaciones entre ellas, como herencia, asociación y dependencia.

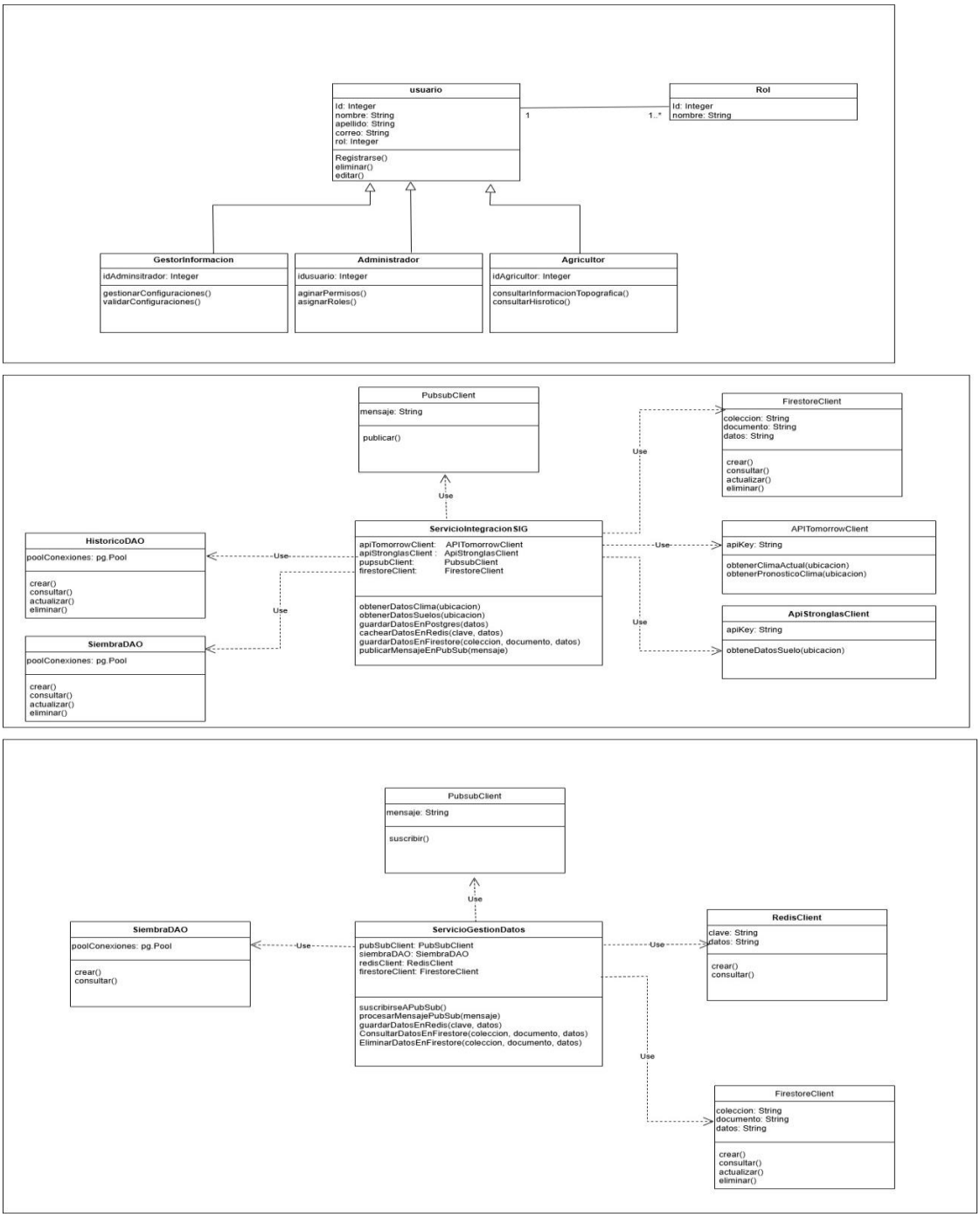


Ilustración 27 Diagrama de clases

Fuente: Construcción Propia

3.4.5 Diagrama de Arquitectura Hexagonal

En este diagrama de arquitectura hexagonal muestra una vista que representa un diseño centrado en el dominio, donde se organizan los componentes en capas: el núcleo (dominio y lógica de negocio) está rodeado por puertos (interfaces) y adaptadores (implementaciones) que conectan el sistema con sus usuarios, servicios externos o infraestructura, promoviendo la independencia tecnológica y la mantenibilidad.

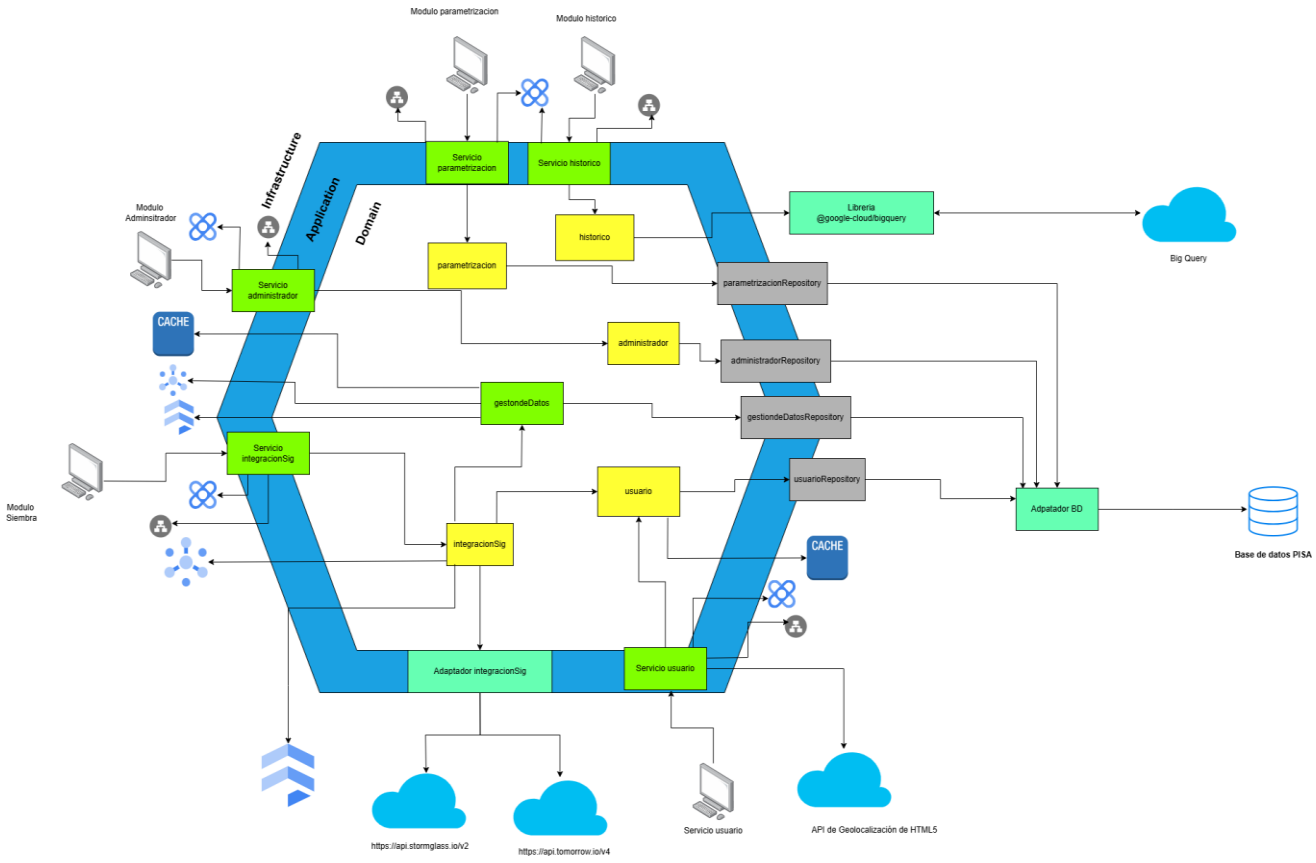


Ilustración 28 Diagrama de arquitectura hexagonal

Fuente: Construcción Propia

3.4.6 Diagrama de Despliegue

En este diagrama de despliegue muestra una vista para representar cómo los componentes de la arquitectura (software y hardware) se distribuyen y ejecutan en un entorno físico. Muestra nodos (servidores, dispositivos, máquinas virtuales) y sus conexiones, así como la ubicación de los artefactos como aplicaciones, servicios o bases de datos. (Ver anexo 10)

3.4.7 Diagrama de Plataformas

En estos diagramas de plataformas se muestra una vista que detalla las tecnologías, herramientas y servicios que soportan el desarrollo, despliegue y operación de la arquitectura. Estos diagramas incluyen plataformas de hardware, sistemas operativos, frameworks, servicios en la nube, bases de datos y otros componentes tecnológicos que forman parte de la infraestructura. Es útil para documentar la arquitectura técnica y garantizar compatibilidad entre los distintos elementos del ecosistema.

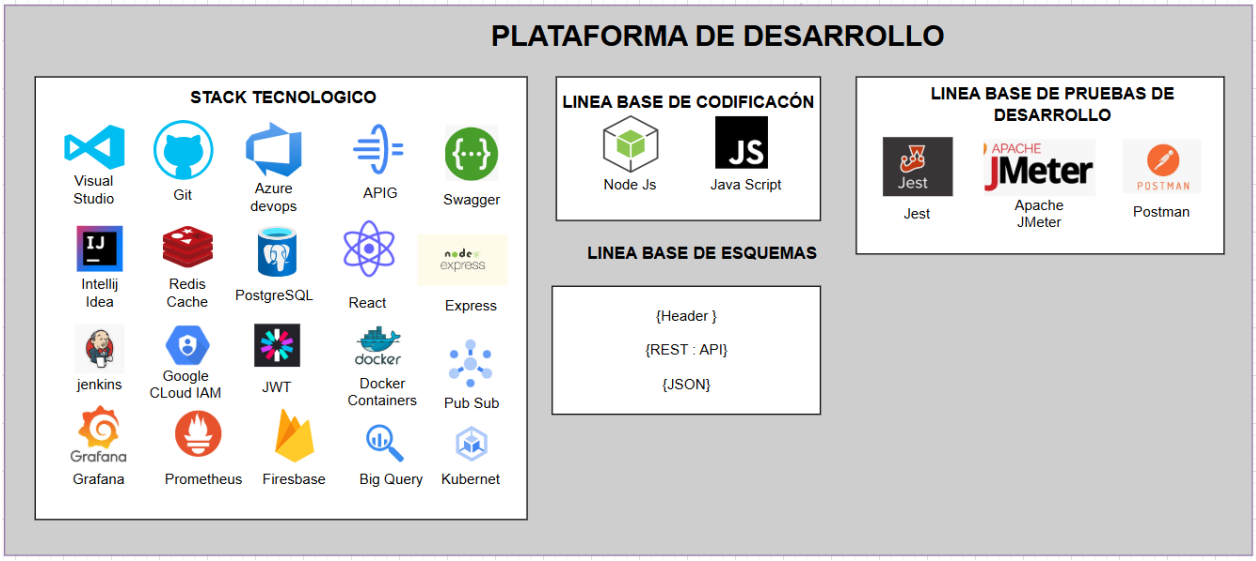


Ilustración 29 Diagrama de plataforma de desarrollo

Fuente: Construcción Propia

Este diagrama representa la plataforma de gestión tecnológica diseñada conceptualmente para el monitoreo del comportamiento de las APIS. Las solicitudes gestionadas por Apigee son distribuidas a microservicios como parametrización, usuarios o integración SIG. Cada uno registra eventos relevantes en un repositorio de eventos central, desde donde Prometheus recolecta métricas de manera periódica. Finalmente, estas métricas se visualizan en Grafana, permitiendo un seguimiento eficiente del estado de los servicios, sus errores y tiempos de respuesta. (Ver anexo 11)

El siguiente diagrama representa conceptualmente cómo se gestiona el monitoreo de los tiempos de respuesta de las bases de datos en la arquitectura. Las solicitudes realizadas por los usuarios (web o móvil) se enrutan a través del API Gateway (Apigee), que las distribuye hacia los microservicios diseñados. Estos servicios interactúan con la base de datos PostgreSQL para recuperar o almacenar información.

Simultáneamente, los microservicios exponen métricas clave como tiempos de respuesta, errores y estado de las consultas, las cuales son recolectadas por Prometheus. Finalmente, Grafana se conecta a Prometheus para ofrecer una visualización centralizada y en tiempo

real del comportamiento de los servicios, lo cual resulta esencial para mantener la calidad del servicio y anticipar problemas de rendimiento en el entorno de producción. (Ver anexo 12)

3.4.8 Resumen del capítulo

En este capítulo se presentó el desarrollo completo del diseño arquitectónico, partiendo de la identificación de atributos de calidad como escalabilidad, disponibilidad, seguridad y mantenibilidad. Se explicaron las decisiones conceptuales adoptadas para seleccionar tecnologías acordes con estos atributos, y se detallaron los módulos diseñados parametrización, gestión de datos e integración SIG, junto con sus flujos y funcionalidades previstas.

El diseño adopta un enfoque híbrido basado en arquitectura hexagonal y microservicios, permitiendo una alta cohesión interna y un bajo acoplamiento con tecnologías externas. Se seleccionaron tecnologías modernas como React.js, Node.js, PostgreSQL, Firestore, Redis y BigQuery, distribuidas conceptualmente según su rol dentro de la arquitectura.

Asimismo, se documentaron los patrones arquitectónicos, principios de diseño y tácticas empleadas para soportar los atributos priorizados, y se representó visualmente la solución a través de diagramas C4, incluyendo contexto, contenedores, componentes, clases.

Este diseño se mantiene en un nivel conceptual, sin implementación, pero plantea una base robusta para la futura construcción de una solución tecnológica capaz de apoyar la toma de decisiones para la siembra de aguacate Hass, alineada con los desafíos de sostenibilidad y eficiencia del sector agrícola colombiano.

CAPÍTULO 4

Evaluación

4.1. Diseño de la evaluación

El propósito de esta evaluación ha sido verificar que el diseño arquitectónico para optimizar la siembra de aguacate Hass en la región andina de Colombia cumple con los objetivos específicos del proyecto. Para ello, se describen en detalle la planeación, los métodos de evaluación empleados, los criterios de aceptación definidos para los atributos de calidad, y los resultados obtenidos. Se utilizó una combinación de enfoques cualitativos y cuantitativos, orientados a validar que las decisiones arquitectónicas adoptadas respondieran adecuadamente a dichos objetivos.

4.1.1. Participantes Clave

- **Director de Proyecto (Guía Académica):** Actúa como mentor y guía en el proyecto. Su rol principal es orientar al estudiante en la comprensión de los conceptos, la correcta ejecución de las actividades y la evaluación desde una perspectiva de buenas prácticas en arquitectura de software.
- **Estudiante (Analista de Arquitectura):** Lidera la investigación y el diseño de la arquitectura. Aplica los conocimientos adquiridos identificando los atributos de calidad más relevantes para el proyecto además se plantea escenarios, analiza las decisiones arquitectónicas e identifica posibles riesgos y sensibilidades.

4.1.2. Planeación de la Evaluación

Para llevar a cabo una evaluación del diseño de la arquitectura, se planificaron tres sesiones estructuradas, cada una con un enfoque específico. Estas sesiones permitieron evaluar la alineación entre los atributos de calidad y las decisiones arquitectónicas, como también la consistencia del diseño y los posibles riegos que se puedan generar.

- **Sesión 1: Relación entre atributos de calidad y decisiones arquitectónicas**

En esta sesión se revisaron los atributos de calidad priorizados en el proyecto (rendimiento, escalabilidad, fiabilidad y seguridad) y se verificó que cada uno estuviera respaldado por decisiones arquitectónicas concretas. Se construyó una matriz de relación que permitió evidenciar la trazabilidad entre los atributos de calidad y las decisiones adoptadas, identificando posibles brechas o aspectos que requerían ajustes. Esta revisión ha sido fundamental para asegurar que el diseño arquitectónico propuesto responderá efectivamente a los atributos de calidad establecidos desde la etapa inicial del proyecto.

- **Sesión 2**: **Evaluación de los diagramas de arquitectura y principios de diseño**

En esta sesión se evaluaron los diagramas de arquitectura desarrollados (contexto, contenedores, componentes y despliegue) con el objetivo de validar la coherencia del diseño y su cumplimiento de principios fundamentales como la separación de responsabilidades, bajo acoplamiento y alta cohesión. Se utilizó una lista de verificación para guiar la evaluación, en la que participaron el director del proyecto y el estudiante, identificando fortalezas y aspectos a mejorar.

- **Sesión 3**: **Identificación y análisis de riesgos arquitectónicos**

En esta sesión se realizó una evaluación sistemática de los riesgos técnicos y arquitectónicos que podrían comprometer el éxito del diseño de la arquitectura de software. Se identificaron posibles escenarios de falla, cuellos de botella tecnológicos, dependencias críticas, y riesgos relacionados con decisiones tempranas del diseño. Como parte del análisis, se construyó una matriz de riesgos donde se estimó la probabilidad e impacto de cada riesgo identificado, y se definieron acciones de mitigación asociadas. Adicionalmente, se discutieron alternativas arquitectónicas que podrían disminuir la exposición a dichos riesgos, considerando factores como la complejidad, el uso de tecnologías específicas, y la evolución esperada del diseño de la arquitectura en el tiempo.

4.1.3. Método de evaluación

La evaluación del diseño arquitectónico se realizó con base en una adaptación conceptual del método ATAM (Architecture Tradeoff Analysis Method), enfocado en el análisis cualitativo de decisiones arquitectónicas a partir de escenarios hipotéticos. Esta adaptación es justificada porque el proyecto no contempla implementación ni pruebas funcionales, sino el diseño y documentación de una arquitectura orientada por atributos de calidad. El objetivo fue evidenciar que el diseño propuesto es coherente, viable y que se pueda implementar en un futuro.

Análisis de cumplimiento frente a objetivos específicos: Se evaluó el grado en que el diseño arquitectónico contribuye al cumplimiento de los objetivos específicos planteados en el proyecto. Para ello, se estableció una relación explícita entre cada objetivo y las decisiones arquitectónicas adoptadas, analizando cómo estas decisiones permiten parametrizar datos, analizar datos en tiempo real, interoperar con las APÍS externas TomorrowClient y weatherapi. Esta evaluación permitió asegurar que el diseño responde a los objetivos específicos establecidos

Objetivo Específico	Evidencia en el Diseño Arquitectónico	Cumplimiento (1 a 5)	Observaciones
1. Establecer los diferentes atributos de calidad que potencian la concepción de la arquitectura.	Se identificaron y priorizaron atributos como rendimiento, escalabilidad, disponibilidad y seguridad. Se formularon escenarios de calidad y se relacionaron con decisiones arquitectónicas.	5	Se evidencia una alineación clara entre los atributos seleccionados y las decisiones tomadas.
2. Definir tácticas arquitectónicas considerando la integración de servicios en la nube.	En el diseño Se incorporaron tácticas como escalamiento horizontal y vertical, autenticación centralizada, desacoplamiento mediante colas y servicios RESTque pueden ser desplegados en entornos cloud.	5	El diseño de la arquitectura incluye el uso de servicios en la nube y considera explícitamente tácticas orientadas a la robustez y adaptabilidad del sistema.
3. Diseñar un módulo parametrizable para gestionar condiciones climáticas, humedad, fuentes hídricas, plagas y enfermedades.	El módulo de parametrización fue diseñado con componentes independientes que permiten configurar variables de una forma confiable y disponible 7/24.	4	Aunque el diseño es claro y modular, la validación específica del comportamiento dinámico dependerá de su implementación futura.
4. Diseñar un módulo de gestión de datos que analice información en tiempo real para apoyar la toma de decisiones en la siembra.	Se diseñó un componente específico para la ingesta y análisis en tiempo real, apoyado por un motor de reglas y almacenamiento temporal.	5	El diseño permite flujos de análisis en tiempo real, con escenarios que en implementaciones futruras pueden probar el rendimiento y confiabilidad del módulo.
5. Diseñar un módulo que acceda a sistemas de información geográfica (SIG) para obtener datos topográficos y climáticos.	Se diseño un módulo de integración SIG con conectores hacia servicios externos los cuales entreguen datos meteorológicos.	5	El diseño cubre adecuadamente la integración SIG mediante APIs y estándares compatibles con APIS geográficos.

Ilustración 30 Matriz de evaluación de Análisis de cumplimiento frente a objetivos específico

Fuente: Construcción Propia

- **Revisión de artefactos de diseño**: Se realizo una revisión de forma detallada a los artefactos arquitectónicos generados a lo largo del proyecto, incluyendo los diagramas del modelo C4 (Contexto, Contenedores, Componentes y Despliegue), las decisiones arquitectónicas documentadas, la justificación de tecnologías seleccionadas, y la aplicación de principios SOLID. Esta evaluación permitió analizar la integridad, consistencia y claridad del diseño propuesto, así como su alineación con buenas prácticas de arquitectura de software. También se verificó la correcta separación de responsabilidades, el uso adecuado de patrones de diseño, y la coherencia entre los distintos niveles de abstracción representados en los diagramas.

Artefacto	Criterio Evaluado	Cumplimiento (1-5)	Observaciones
Diagrama de Contexto	Identificación clara de actores y sistemas externos, límites del sistema definidos.	5	Permite comprender de forma efectiva las interacciones globales , ademas representa claramente los actores externos.
Diagrama de Contenedores	Estructura lógica bien definida, separación clara entre frontend, backend y servicios.	5	Refleja adecuadamente los principales contenedores de software y sus responsabilidades.
Diagrama de Componentes	Aplicación de principio de modularidad, uso de interfaces, acoplamiento bajo y alta cohesión entre componentes.	4	Se identifican los componentes clave, aunque algunos podrían detallarse más y ademas podría mejorar la descripción de interfaces.
Diagrama de Clases	Modelado de entidades del dominio, relaciones, herencia y encapsulamiento.	4	Las clases reflejan bien la lógica del dominio, aunque puede profundizarse en relaciones.
Diagrama de Despliegue	Infraestructura considerada, balanceo de carga, redundancia y distribución geográfica.	5	El diseño refleja adecuadamente la arquitectura física y su escalabilidad en la nube.
Decisiones Arquitectónicas	Registro de decisiones, alternativas consideradas, justificación clara y Coherencia con atributos de calidad	5	Se documentaron decisiones clave con base en tácticas y trade-offs, cada decisión responde a un atributo específico.
Tecnologías Seleccionadas	Son pertinentes con respecto a los atributos de calidad, madurez y compatibilidad tecnológica.	5	Las tecnologías elegidas responden a las necesidades de rendimiento, escalabilidad, conmfiabilidad y segurirdad.
Principios SOLID y buenas prácticas	Aplicación de principios de diseño, separación de responsabilidades, mantenibilidad.	4	Se evidencia la intención de aplicar SOLID, aunque en algunos casos puede profundizarse más.

Ilustración 31 Matriz de Revisión de artefactos de diseño

Fuente: Construcción Propia

- **Revisión por pares:** Se realizaron dos revisiones técnicas, ambas lideradas por el director de proyecto centradas en la calidad del diseño de la arquitectura, estas revisiones contaron con la participación del estudiante investigador. Se evaluó la coherencia del diseño y la adecuación a los requisitos del proyecto.

Aspecto Evaluado	Descripción	Observaciones / Sugerencias	Cumplimiento (1-5)	Responsable Revisión
Documentación técnica	Revisión del conjunto de artefactos de arquitectura (diagramas C4, decisiones arquitectónicas, componentes).	Se recomienda mayor nivel de detalle en las relaciones de componentes internos del sistema.	5	Director del proyecto
Coherencia del diseño	Se analizó la alineación entre el diseño arquitectura y los requisitos funcionales y no funcionales	Buena alineación con los objetivos.	5	Director del proyecto
Proceso de revisión	Se entregó documentación al director de forma anticipada, se hizo revisión individual y luego reunión conjunta para discutir hallazgos.	Proceso estructurado y bien documentado; se utilizó plantilla estandarizada para observaciones.	4	Director y estudiante
Simulación de escenarios de calidad	Se formularon escenarios de calidad y se simularon conceptualmente para validar los atributos críticos del diseño arquitectónico.	NO APLICA	4	Estudiante investigador
Instrumentos de trazabilidad conceptual	Se emplearon matrices para vincular atributos de calidad, tácticas arquitectónicas y componentes del diseño.	Se sugiere representar de forma más detallada el impacto de las decisiones arquitectónicas con mapas de trazabilidad o diagramas de causa-efecto.	2	Director del proyecto

Ilustración 32 Matriz de Revisión por pares

Fuente: Construcción Propia

Proceso: Se ha proporcionado la documentación de la arquitectura diagramas C4, descripción de componentes, entre otros y una lista de escenarios de calidad. El director del proyecto reviso la documentación de forma independiente y luego se reunió con el investigador para discutir sus hallazgos. Se ha utilizado una plantilla estandarizada para registrar los comentarios y sugerencias de mejora.

Artefacto Revisado	¿Entregado?	¿Completo?	Observaciones del Revisor
Diagrama de Contexto (C4 nivel 1)	☑ Sí	☑ Sí	Representa claramente los actores externos e interacción inicial.
Diagrama de Contenedores (C4 nivel 2)	☑ Sí	☑ Sí	Representa claramemnte los detalles sobre contenedores en la arquitectura.
Diagrama de Componentes (C4 nivel 3)	☑ Sí	☑ Sí	Componentes están bien definidos; sería ideal indicar flujos de datos principales.
Diagrama de Clases	☑ Sí	☑ Sí	Buena definición de entidades del módulo de datos; sugerencia: indicar clases auxiliares.
Diagrama de Despliegue	☑ Sí	☑ Sí	Representa entornos nube y dispositivos IoT, pero falta incluir redundancia para alta disponibilidad.
Árbol de Decisiones Arquitectónicas	☑ Sí	☑ Sí	Decisiones justificadas frente a atributos de calidad.

Ilustración 33 Plantilla estandarizada de revisión por pares

Fuente: Construcción Propia

- **Requisitos Funcionales y No Funcionales del Diseño Arquitectónico**

Requisitos Funcionales

Los siguientes requisitos funcionales se derivan del análisis del problema planteado y las necesidades del contexto en la dificultad de toma de decisiones para la siembra del aguacate Hass:

Código	Requisito Funcional
RF-01	
RF-01	El diseño arquitectónico debe contemplar componentes que permitan la configuración de parámetros técnicos relacionados con el cultivo, incluyendo factores climáticos como la temperatura, humedad, precipitación, condiciones del suelo, altitud y elementos sanitarios como plagas y enfermedades.
RF-02	El diseño debe ofrecer mecanismos de consulta en tiempo real de condiciones climáticas y geográficas.
RF-03	El diseño debe contemplar el registro en logs los eventos relevantes configuración de parámetros, errores, fallas de servicios externos, para permitir monitoreo y trazabilidad.

RF-04	El diseño debe estructurase para permitir acceso por roles diferenciados, integrando mecanismos de autenticación y autorización desde su concepción.
RF-05	El diseño debe estructurase con componentes para la generación de reportes técnicos de viabilidad de siembra, considerando los parámetros configurados y los datos recopilados desde servicios externos. Estos reportes deben estar disponibles para su visualización en el frontend y como archivos descargables.

Tabla 6 Requisitos Funcionales

Requisitos no Funcionales

Los siguientes requisitos no funcionales han sido utilizados como criterios de calidad en la toma de decisiones arquitectónicas. Su formulación responde a los atributos priorizados y fundamenta la selección de tecnologías, estilos, patrones y tácticas arquitectónicas.

Código	Requisito No Funcional/Atributo de Calidad Asociado
RNF-01	**Rendimiento:** El diseño debe incorporar tácticas como cache en memoria Redis y desacoplamiento de procesos (Pub/Sub), con el objetivo de permitir respuestas ágiles a operaciones de análisis y consulta.
RNF-02	**Disponibilidad:** El diseño arquitectónico debe evitar puntos únicos de falla mediante distribución de servicios, balanceo de carga, y recuperación ante fallos, buscando disponibilidad conceptual continua (24/7).
RNF-03	**Escalabilidad:** El diseño debe contemplar el crecimiento horizontal de componentes en función del número de usuarios o fincas conectadas, aprovechando despliegues en contenedores orquestados por Kubernetes.
RNF-04	**Seguridad:** La arquitectura diseñada debe prever el uso de autenticación, control de acceso por roles y transporte seguro de datos mediante, como medidas integradas desde el diseño.
RNF-05	**fiabilidad:** El diseño de la arquitectura debe contemplar mecanismos de validación de datos y manejo explícito de errores, con el objetivo de garantizar que, ante las mismas condiciones de entrada, la arquitectura produzca resultados consistentes, predecibles y libres de comportamientos inesperados.

Tabla 7 Requisitos no Funcionales

Requisitos Funcionales si en un momento futuro se implementa la arquitectura.

Aunque el presente proyecto se centró exclusivamente en el diseño conceptual de la arquitectura, se consideraron algunos escenarios funcionales hipotéticos que reflejan los posibles comportamientos esperados de la arquitectura si se llegara a implementar. Estos escenarios fueron formulados como una guía para validar la coherencia del diseño con las necesidades reales del usuario agricultor, en el contexto de la siembra de aguacate Hass. Se presentan a continuación como requisitos funcionales, los cuales podrían formar parte de una futura fase de desarrollo e implementación de la arquitectura.

Requisito	Descripción	Evento esperado	Validación esperada
RF01	El usuario podrá consultar las condiciones climáticas actuales como temperatura, humedad, viento, presión desde su ubicación utilizando el módulo de gestión de datos.	La arquitectura recibirá las coordenadas del usuario y devolverá la información climática en tiempo real a través de una interfaz gráfica.	La respuesta debe generarse en menos de 3 segundos y reflejar datos provenientes de una fuente externa validada.
RF02	La arquitectura debe notificar al agricultor si las condiciones climáticas superan umbrales críticos previamente parametrizados.	El sistema debe notificar al agricultor si las condiciones climáticas superan umbrales críticos previamente parametrizado	La arquitectura compara los datos con los valores del módulo de parametrización y activa un flujo de alerta a través del frontend
RF 03	La arquitectura evaluará si un terreno es apto para siembra basándose en la ubicación y parámetros climáticos y geográficos históricos.	El agricultor ingresa una ubicación, y la arquitectura responde con una recomendación (apto/no apto) basada en reglas de negocio definidas.	El módulo de gestión de datos procesa la información y devuelve un resultado razonado basado en la lógica del dominio.

Tabla 8 Requisitos Funcionales en un futuro

Escenarios de calidad simulados:

Dado que el sistema no fue implementado los escenarios de calidad se formularon como situaciones hipotéticas que podrían enfrentar los componentes de la arquitectura, cada uno se relacionó explícitamente con el atributo evaluado.

Rendimiento:

- **Escenario 1:** Simular que 100 usuarios consultan al Modulo Integración SIG, cada consulta debe responder en menos de 3 segundos en el 95% de los casos.

PARTE	VALOR
Fuente	Usuario final (agricultor)
Estímulo	100 usuarios simultáneamente solicitan información de plagas y enfermedades
Artefacto	Microservicio integracionSIG
Ambiente	En ejecución, bajo carga concurrente
Respuesta	El sistema utiliza Redis para respuestas frecuentes
Medida de la respuesta	La arquitectura responde en tiempo menor a 3 segundos por petición, la cache Redis reduce la latencia en las consultas frecuentes a la base de datos relacional

Tabla 9 Escenario 1 Rendimiento

Escalabilidad:

- **Escenario 2**: Simular un incremento de 1 a 100 para consultar información de APIS externas

PARTE	VALOR
Fuente	Usuario final (agricultor)
Estímulo	Incremento del número de solicitudes simultáneas para consultar información de APIS externas
Artefacto	Microservicio integracionSIG
Ambiente	Muchos usuarios realizan la misma petición
Respuesta	Los servicios escalan horizontalmente sin degradación de respuesta; el balanceador distribuye carga eficientemente
Medida de la respuesta	El sistema sigue respondiendo en menos de 3 segundos; los microservicios aumentan instancias dinámicamente.

Tabla 10 Escenario 2 Escalabilidad

Disponibilidad:

- **Escenario 3**: Simular la falla de las APIS externas y medir el tiempo de recuperación del sistema.

PARTE	VALOR
Fuente	Usuario final (agricultor)
Estímulo	Fallo en el servicio de datos climáticos externo (API de terceros no

	responde)
Artefacto	Microservicio integracionSIG Y API externa
Ambiente	Durante una solicitud de viabilidad de la siembra
Respuesta	Se contemplo en el diseño un patrón circuit breaker, reintenta la conexión y en caso de falla, devuelve una respuesta de fallo que será enviada al usuario administrador.
Medida de la respuesta	Disponibilidad mayor al 99.9%, con recuperación ante fallos menor a 30 segundos

Tabla 11 Escenario 3 Disponibilidad

Seguridad:

- **Escenario 4**: Simular que se accede al microservicio de gestión de datos sin token el cual es necesario para ingresar al módulo.

PARTE	VALOR
Fuente	Usuario sin autorización.
Estímulo	Intenta acceder al módulo de gestión de datos sin los permisos necesarios.
Artefacto	Microservicio gestión de datos.
Ambiente	En ejecución, durante una solicitud HTTP.
Respuesta	Se bloquea el acceso y se registra el intento; el usuario recibe un mensaje de "Acceso no autorizado".
Medida de la respuesta	Cero accesos indebidos permitidos.

Tabla 12 Escenario 5 Seguridad

CONFIABILIDAD

- **Escenario 5**: Simular que sé valida la consistencia de los datos y recuperación ante errores controlados

PARTE	VALOR
Fuente	Usuario administrador.
Estímulo	100 usuario realizan múltiples consultas al módulo de gestión de datos.
Artefacto	Microservicio gestión de datos.
Ambiente	En ejecución, durante una solicitud HTTP.
Respuesta	Con el diseño de la arquitectura aplica el

	principio ACID mediante el uso de transacciones atómicas en la base de datos, asegurando que ninguna operación quede incompleta o corrompa la integridad de los datos.
Medida de la respuesta	Se garantizo la integridad de los datos aún ante fallos. En pruebas conceptuales, ninguna transacción quedó en estado intermedio.

Escenario 6: Simular que sé valida la consistencia de los datos de las APIS externas

PARTE	VALOR
Fuente	Servicio API externa.
Estímulo	Envió de datos con campos incompletos o malformateados.
Artefacto	Microservicio integración SIG.
Ambiente	En ejecución, durante una solicitud HTTP.
Respuesta	El módulo valida datos, detecta errores y los descarta.
Medida de la respuesta	Se asegura la consistencia de los datos almacenados.

RESILIENCIA

- **Escenario 8**: Simular que las respuestas de las APIS externas fallan.

PARTE	VALOR
Fuente	APIS externas
Estímulo	Las APIS externas fallan intermitentemente
Artefacto	Microservicio integración SIG.
Ambiente	Fallos de red esporádicos.
Respuesta	La arquitectura se diseñó para realizar reintentos exponenciales con el patrón Circuit braker.
Medida de la respuesta	Se evita la propagación de fallos al resto de la arquitectura.

4.1.4. Análisis críticos de resultados

Aunque los escenarios de calidad definidos permitieron validar teóricamente el cumplimiento de los atributos establecidos, es importante reconocer las limitaciones del entorno de prueba. Las pruebas locales demostraron que la arquitectura, bajo ciertas condiciones simuladas, podría comportarse de manera eficiente, pero aún es necesario implementar la solución para validar los comportamientos en producción y ajustar decisiones arquitectónicas como la elección del tipo de almacenamiento, balanceadores de carga, y mecanismos de cache.

4.1.5. Conclusiones de la evaluación

Aunque no se realizaron pruebas automatizadas ni simulaciones reales, debido al alcance del proyecto, la evaluación se sustentó en el análisis de diseño, documentación técnica de las herramientas propuestas, por esto el diseño arquitectónico es adecuado para las necesidades del proyecto, ya que cumple con los requisitos de escalabilidad, rendimiento y seguridad. Sin embargo, es necesario seguir evaluando ciertos aspectos, como la optimización de recursos y la integración de nuevas tecnologías, para mejorar la eficiencia a largo plazo. En general, no se requieren ajustes mayores, pero se sugiere realizar pruebas adicionales para asegurar la robustez del sistema en condiciones extremas.

4.1.6. Resumen del capítulo

La En este capítulo se desarrolló una evaluación estructurada del diseño arquitectónico propuesto, enfocándose en determinar su alineación con los objetivos del proyecto y los atributos de calidad definidos. La evaluación se basó en escenarios conceptuales, documentación técnica y la aplicación del método ATAM, adaptado al contexto de una arquitectura en fase de diseño.

Se analizaron escenarios para atributos clave como escalabilidad, rendimiento, disponibilidad, seguridad, mantenibilidad y resiliencia. Cada escenario fue formulado con base en los requisitos funcionales y no funcionales identificados previamente, y vinculado con tácticas arquitectónicas específicas (por ejemplo, desacoplamiento, cacheo, balanceo de carga, uso de colas de eventos y autenticación delegada).
Adicionalmente, se examinó la coherencia entre las tecnologías seleccionadas Node.js, React.js, Firestore, Redis, BigQuery, Pub/Sub, Apigee, y las decisiones arquitectónicas tomadas, a través de diagramas C4, matriz de atributos, flujos de datos y el árbol de decisiones técnicas.

La evaluación concluyó que el diseño arquitectónico es conceptualmente robusto y factible, pero deberá ser complementado, en etapas posteriores, con pruebas de concepto que permitan validar su comportamiento bajo condiciones reales. La arquitectura está preparada para adaptarse a entornos complejos, distribuidos y cambiantes, como lo

requiere el contexto del sector agrícola colombiano.

CAPÍTULO 5

Conclusiones

5.1. Conclusiones

- La ejecución de este proyecto permitió obtener aprendizajes significativos sobre el diseño arquitectónico en contextos agrícolas, donde la variabilidad de datos, la necesidad de disponibilidad constante y la integración con fuentes externas representan retos técnicos sustanciales.

- A lo largo de la evaluación y diseño arquitectónico, se lograron cumplir los objetivos específicos establecidos. El diseño modular basado en arquitectura hexagonal y microservicios ha demostrado ser adecuado para la gestión de grandes volúmenes de datos provenientes de los sistemas de información geográfica (SIG). Los diagramas de contexto, contenedores, componentes entre otros, se alinearon con los resultados esperados, lo que asegura una alta disponibilidad, seguridad y rendimiento.

- Los enfoques arquitectónicos adoptados se basan en principios ampliamente aceptados en la literatura, como la modularidad y la separación de responsabilidades. Comparando con estudios previos, el diseño de microservicios y de la arquitectura hexagonal han demostrado ser una práctica sólida para sistemas distribuidos y escalables, especialmente en proyectos que requieren un manejo eficiente de datos complejos y climáticos.

- El diseño de esta arquitectura tiene importantes implicaciones prácticas, especialmente en el ámbito agrícola, donde la escalabilidad y la disponibilidad son críticas. Las soluciones tecnológicas propuestas, como el uso de PostgreSQL, BigQuery y Firestore, permiten una integración eficiente de datos climáticos.

- El diseño arquitectónico de esta arquitectura puede servir de modelo para otras iniciativas que busquen optimizar procesos de toma de decisiones en sectores similares.

- El diseño arquitectónico, aunque robusto, presenta ciertas limitaciones en cuanto a la integración total con sistemas externos, como las APIS externas. Además, el proceso de escalabilidad, aunque probado a nivel teórico, necesita ser validado en un entorno de producción con una carga de trabajo real. También se debe considerar la necesidad de ajustar los parámetros de seguridad a medida que se implementan características adicionales.

- El diseño arquitectónico planteado proporciona una base sólida para una futura implementación, especialmente en términos de modularidad y adaptabilidad. Sin embargo, la ausencia de una implementación real limita la validación de aspectos como la resiliencia y la disponibilidad. Se recomienda en etapas posteriores ejecutar pruebas sobre entornos distribuidos en la nube real para confirmar el comportamiento bajo carga y validar la integración con APIS externas en condiciones reales de red.

- Desde el punto de vista del usuario final que es el agricultor, el diseño arquitectónico propuesto busca ofrecer una solución accesible, oportuna y confiable para apoyar la toma de decisiones. No obstante, será necesario en futuras fases de implementación considerar aspectos de usabilidad, conectividad rural y acceso a dispositivos móviles, para asegurar que la tecnología diseñada se traduzca en valor real para el agricultor.

5.1.1 Hallazgos relevantes

- La arquitectura basada en atributos de calidad aporta una metodología clara y defendible para tomar decisiones técnicas con criterios justificables.

- La integración con APIS SIG debe considerarse como un componente crítico y altamente volátil, lo cual requiere estructuras resilientes y desacopladas.

Limitaciones y lecciones aprendidas:

Una de las principales limitaciones fue no contar con un entorno de prueba que permitiera validar empíricamente las decisiones arquitectónicas.

Al tratarse de una tesis de maestría, el alcance estuvo delimitado a la propuesta conceptual del diseño arquitectónico, sin extenderse hacia la implementación ni el despliegue en un entorno productivo.

Se identificó que trabajar exclusivamente con atributos teóricos de calidad, sin una implementación parcial, puede hacer difícil evidenciar el impacto real de algunas decisiones.

5.1.2 Impedimentos durante el desarrollo del proyecto

Aunque el proyecto alcanzó satisfactoriamente el objetivo de formular un diseño arquitectónico sólido y alineado, se identificaron algunos impedimentos inherentes al enfoque teórico y a las condiciones del trabajo académico, los cuales condicionaron el tipo de validación alcanzado:

- **Restricción temporal del trabajo académico:** Dado que este proyecto se realizó en el marco de una tesis de maestría, se debió priorizar la profundidad conceptual y la claridad metodológica, sacrificando posibles desarrollos funcionales que habrían enriquecido la validación del diseño.

- **Limitaciones en recursos humanos y técnicos:** El desarrollo del proyecto fue llevado a cabo por una sola persona, lo cual restringió la posibilidad de implementación del diseño de la arquitectura.

A pesar de estos impedimentos, el trabajo logró cumplir con los objetivos planteados y sentar las bases para futuras extensiones e implementaciones, al estructurar una arquitectura teóricamente validada y alineada con necesidades reales de la toma de decisiones en el momento de la siembra de aguacate Hass en la región andina de Colombia.

5.2. Trabajos futuros

A partir del diseño arquitectónico planteado, se identifican varias líneas de trabajo futuro que pueden enriquecer este proyecto o servir como base para nuevas investigaciones:

1. **Desarrollo de una Prueba de Concepto (PoC):** Implementar un prototipo funcional de uno o varios módulos como parametrización o gestión de datos que permita validar los tiempos de respuesta, integración de APIs SIG, tolerancia a fallos y comportamiento en condiciones reales.

2. **Integración con Inteligencia Artificial:** Ampliar el diseño arquitectónico para incorporar modelos de predicción mediante aprendizaje automático, que permita generar recomendaciones basadas en patrones históricos y condiciones climáticas.

3. **Monitoreo activo y visualización avanzada:** Diseñar un panel de control inteligente que integre visualizaciones geográficas en tiempo real, y que pueda incluir mapas de calor, alertas tempranas y simulaciones predictivas de viabilidad agrícola.

4. **Adaptación del diseño a otros cultivos:** Extender este diseño de arquitectura hacia otros tipos de cultivo que tengan requerimientos similares pero variables críticas diferentes como café, caña, cacao.

5. **Análisis de impacto ambiental:** Evaluar cómo el uso de esta arquitectura puede contribuir a la sostenibilidad, eficiencia en el uso del agua, o reducción de insumos en la siembra, integrando métricas ecológicas en los reportes generados.

5.3. Lecciones aprendidas

- Durante el proyecto una de las lecciones aprendidas fue la importancia de comprender las necesidades reales de los agricultores mediante el trabajo de campo. La visita a la finca El Recreo y la entrevista con el agricultor Javier Carvajal destacaron los desafíos

que enfrentan en la siembra de aguacate Hass, como la falta de herramientas que permitan identificar las áreas óptimas para sembrar y planificar de manera eficiente el número de árboles a cultivar. Este acercamiento permitió validar que las soluciones tecnológicas diseñadas en el proyecto, como el uso de análisis de datos, tienen un impacto directo en la resolución de problemas específicos del agricultor.

- La importancia de garantizar una adecuada separación de responsabilidades en el diseño arquitectónico, lo que facilita el mantenimiento y la escalabilidad del sistema.
- A lo largo del proceso de diseño, se aprendió que validar las decisiones arquitectónicas en etapas tempranas del proyecto es crucial para evitar costosos cambios en fases posteriores.
- Es muy importante realizar un análisis regular para evaluar si las tecnologías utilizadas siguen siendo las más adecuadas para las necesidades del proyecto.
- Mantener una documentación clara y actualizada a lo largo del proceso de diseño es vital para la correcta implementación y mantenimiento de la arquitectura. Esto facilita la incorporación de nuevos desarrolladores al equipo y garantiza que todos los miembros comprendan el flujo y la arquitectura diseñada.
- Este trabajo permitió aplicar y combinar diversas teorías y buenas prácticas de diseño arquitectónico, bajo un enfoque orientado a calidad. El análisis de Trade-Offs entre atributos como rendimiento, disponibilidad y mantenibilidad fue clave para tomar decisiones informadas. Aunque el sistema no fue implementado, el ejercicio de diseño exigió un razonamiento arquitectónico riguroso, propio de un contexto académico de maestría.

Bibliografía

AGROSAVIA (2018). *Aplicativo TECNOHASS®*. Disponible en: https://www.agrosavia.co/productos-y-servicios/oferta-tecnol%C3%B3gica/l%C3%ADnea-agr%C3%ADcola/frutales/sistemas-de-informaci%C3%B3n/805-aplicativo-tecnohass. Consultado en abril de 2024.

Amazon Web Services (2023). *¿Qué es una base de datos?* Disponible en: https://aws.amazon.com/es/whatis/database/#:~:text=Una%20base%20de%20datos%20es,almacenar%2C%20recuperar%20y%20editar%20datos.Consultado en febrero de 2024.

Arquitectura de Referencia. *¿Qué es una Arquitectura de Referencia?*

Disponible en: https://conexiam.com/es/que-es-una-arquitectura-de-referencia/

Consultado en febrero de 2025.

Asesores en Producción y Comercio, G. C. (2021). *La importancia del aguacate Hass en Colombia*. Disponible en:

https://co.gowanco.com/news/la-importancia-del-aguacate-hass-en-colombia.

Consultado en abril de 2024.

Asociación de Productores y Empacadores Exportadores de Aguacate d México (APEAM) (2024). *Programa de aplicación de la tecnología en el cultivo de aguacate*. Disponible en:

https://pruebas.apeamac.com/investigacion/programa-de-aplicacion-de-la-tecnologia-en-el-cultivo-de-aguacate/.

Consultado en abril de 2024.

Bass, L., Kazman, R., Clements, P. (2013). *Software Architecture in Practice*.

Disponible en:

https://edisciplinas.usp.br/pluginfile.php/5922722/mod_resource/content/1/2013%20-%20Book%20-%20Bass%20%20Kazman-Software%20Architecture%20in%20Practice%20%281%29.pdf.

Consultado el 12 mayo de 2024.

Cherlinka V (2024). *Humedad Del Suelo: Cómo Medir Y Controlar Su Nivel.* Disponible en: https://eos.com/es/blog/humedad-del-suelo/.

Consultado en abril de 2024.

Ciencia y tecnología al servicio del agro (2020). Comfama.com. Disponible en:

https://revista. comfama.com/desarrollo-territorial/ciencia-y-tecnologia-al-servicio-deagro/.

Consultado en abril de 2024.

Google Cloud (s/fa). *Arquitectura de software: ¿Qué es y qué tipos hay*? Disponible en: https://www.gluo.mx/blog/arquitectura-de-software-que-es-y-que-tipos-hay.

Consultado en abril de 2024.

Google Cloud (s/fb). *Requerimientos en el desarrollo de s oftware y aplicaciones.*

Disponible en:

https://www.northware.mx/blog/requerimientos-en-el-desarrollo-de-software-aplicaciones/.

consultado en mayo de 2024.

Contenedores *¿Qué son los contenedores?* Disponible en:

https://cloud.google.com/learn/what-are-containers?hl=es

Consultado en junio de 2024.

Syndcode *¿Atributos de calidad de la arquitectura de software?* Disponible en:

https://syndicode.com/blog/12-software-architecture-quality-attributes/

Consultado en junio de 2024.

Google Cloud (s/fc). *¿Qué es un proveedor de servicios de Cloud?* Disponible en:

https://cloud.google.com/learn/what-is-a-cloud-service-provider?hl=es#:~:text=Los%20tres%20grandes%20(Google%20Cloud,Red%20Hat%2C%20DigitalOcean%20y%20Rackspace.

Consultado en abril de 2024.

Matemáticas: Problemas introductorios tipo PISA para el primer ciclo de Educación Secundaria Obligatoria (2014). Carm.es. Disponible en: https://www.carm.es/edu/pub/14_2014/Altitudes%20increibles.html#:~:text=La%20altitud%20es%20la%20distancia,sobre%20el%20nivel%20del%20mar.

Consultado en marzo de 2024.

Ministerio de Agricultura (2020). *Cadena productiva de Aguacate*. Disponible en:

https://sioc.minagricultura.gov.co/aguacate/documentos/2020-03-30%20cifras%20sectoriales.pdf.

Consultado en abril de 2024.

Ministerio de Justicia (2022). *Qué puedo hacer si mi predio tiene un área diferente de la que se encuentra en el folio de matrícula inmobiliaria*. Disponible en:

https://www.minjusticia.gov.co/programas-co/LegalApp/Paginas/aclaracion-de-areas-y-inderos.aspx#:~:text=%E2%80%8BEl%20%C3%A1rea%20de%20un,matr%C3%ADcula%20inmobiliaria%20y% 20otros%20documentos.

Consultado en enero de 2024.

IBM (20205) *Que es la API REST*. Disponible en: *https://www.ibm.com/mx-es/think/topics/rest-apis*

Consultado en enero de 2024.

Salud electrónica (1 agosto, 2021). *Calidad del Software*. Disponible en: https:// saludelectronica.com/calidad-del-software/.

Consultado en mayo de 2024.

Scrumguides.org (2024). *Scrum Guides*. Disponible en: https://scrumguides.org/.

Consultado en abril de 2024.

THE FOOD TECH (2021). *Inteligencia Artificial, la nueva táctica de apoy al cultivo de aguacate*. THE FOOD TECH - Medio de noticias líder en la Industria de Alimentos y Bebidas. Disponible en:

https://thefoodtech.com/seguridad-alimentaria/inteligencia-artificial-la-nueva-tactica-de-apoyo-al-cultivo-de-aguacate/.

Consultado en mayo de 2024.

Tiempo, *Monitoreo del cambio climático* (2019). *RDU UNAM*. Disponible en:

https://www.ctc-n.org/technologies/climate-change-monitoring

Consultado en agosto de 2025.

minciencias (2017). *Pectia* Disponible en:

https://minciencias.gov.co/sites/default/files/upload/noticias/pectia-2017-actualizado.pdf

Consultado en agosto de 2025.

TOGAF (2024). *The Open Group*. Disponible en:

https://www.opengroup.org/togaf.

Consultado en mayo de 2024.

agenciarococrm.com *componentes-del-software* Disponible en: *https://agenciarococrm.com/componentes-del-software/*
Consultado en mayo de 2024.

Pablo Cruz (2021) *Arquitectura de Software*. Disponible en:

https://arquitecturadesoftware.cl/2021/01/14/de-compromisos-tradeoffs-y-sus-resoluciones-en-la-arquitectura-de-software/
Consultado en mayo de 2025.

EDU Salguero (2022). *Arquitectura hexagonal*. Disponible en: https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f

Consultado en junio de 2024.

Intel (2012). *Que son los microservicios*. Disponible en: https://www.intel.la/content/www/xl/es/cloud-computing/microservices.html#:~:text=La%20arquitectura%20de%20microservicios%20es,suelto%20de%20una%20interfaz%20est%C3%A1ndar.
Consultado en junio de 2024.

SYND/CODE (2014). *Atributos de calidad de la arquitectura de software*. Disponible en:
https://syndicode.com/blog/12-software-architecture-quality-attributes/
Consultado en julio de 2024.

Intel (2021) *Arquitectura de microservicios*. Disponible en: https://www.intel.la/content/www/xl/es/cloud-computing/microservices.html#:~:text=La%20arquitectura%20de%20microservicios%20es,suelto%20de%20una%20interfaz%20est%C3%A1ndar.
Consultado en enero de 2025.

AGROSAVIA (2010). *Manual técnico del cultivo del aguacate*. Disponible en: https://repository.agrosavia.co/handle/20.500.12324/13490
Consultado en octubre de 2024.

AGROPTIMA (2020). *Cálculo de árboles del aguacate*. Disponible en: https://blog.agroptima.com/es/blog/calcular-numero-arboles-hectarea/
Consultado en octubre de 2024.

Cervantes H., Velasco E., C. L. (2016). *Arquitectura de Software*

Conceptos y ciclo de desarrollo. Disponible en: https://www.researchgate.net/profile/Perla-elascoElizondo/publication/281137715Arquitecturade__ Software_Conceptos_y_Ciclo_de_Desarrollo/links/57144e1408aeebe07c0641ab/Arquitectura-de-Software-Conceptos-y-Ciclo-de-Desarrollo.pdf.

Consultado enabril de 2024.

Euristiq (2025). *Arquitectura de micro Frontend*. Disponible en: https://euristiq.com/micro-frontend-architecture/
Consultado en enero de 2025.

CILSA (2025). *Que es un lenguaje de programación*. Disponible en: https://desarrollarinclusion.cilsa.org/tecnologia-inclusiva/que-es-un-lenguaje-de-programacion/
Consultado en septiembre de 2024.

Camero, J. D. G (2020). *Diseño de un modelo predictivo de la oferta de aguacate Hass en el municipio de Herveo*. Disponible en:

https://repository.usta.edu.co/bitstream/handle/11634/35564/2021JuanGarzon.pdf?sequence=1.

Consultado en abril de 2024.

Agrosavia (2020). *Generalidades del cultivo*. Disponible en:

https://editorial.agrosavia.co/index.php/publicaciones/catalog/download/162/145/1120-1?inline=1

Consultado en mayo de 202

Winlon (2024). *Hecterra: la solución GPS agrícola*. Disponible en:

https://wialon.com/es/agriculture-gps-tracking?utm_campaign=HECTERRA_ES_SEARCH&utm_source=google&utm_medium=cpc&utm_content=591615101165&utm_term=la%20agricultura%20de%20precisi%C3%B3n&gad_source=1&gclid=Cj0KCQjw0__WyBhDMARIsAL1Vz8tD2pKJiKDHTgt8HRKFk8SarN9fQFHW7ZkgbrT99r44jqT0IdziY9AaAgBCEALw_wcB.

Consultado en mayo de 2024.

IOT (2025) *Hecterra: IoT y Monitoreo Ambiental con Redes de Sensores.*

Disponible en:

https://www.iotforall.com/iot-and-environmental-monitoring-with-sensor-networks

Consultado en agosto de 2025.

Climate nasa (2025) *Nasa: temperatura global*.

Disponible en:

https://climate.nasa.gov/en-espanol/signos-vitales/temperatura-global/?intent=111

Consultado en agosto de 2025.

Anexos

Anexo 1

Tabla: zona

Campo	Tipo	Descripción	Restricciones
id_zona	Serial	Identificador de la zona	PRIMARY KEY
nombre	Varchar	Nombre de la zona	NOT NULL
fecha_creacion	Timestamp	Fecha de creación	NOT NULL
id_usuario	Integer	Usuario que creó la zona	NOT NULL

Tabla: región

Campo	Tipo	Descripción	Restricciones
id_region	Serial	Identificador de región	PRIMARY KEY
nombre	Varchar	Nombre de la región	NOT NULL
id_zona	Integer	Zona a la que pertenece	FK zona
fecha_creacion	Timestamp	Fecha de creación	NOT NULL
id_usuario	Integer	Usuario que creó el registro	NOT NULL

Tabla: país

Campo	Tipo	Descripción	Restricciones
id_pais	Serial	Identificador de país	PRIMARY KEY
nombre	Varchar	Nombre del país	NOT NULL
fecha_creacion	Timestamp	Fecha de creación	NOT NULL
id_usuario	Integer	Usuario creador	NOT NULL

Tabla: siembra

Campo	Tipo	Descripción	Restricciones
id_siembra	Serial	Identificador de la siembra	PRIMARY KEY
ubicacion	Varchar	Ubicación geográfica	NOT NULL
potencial_validez	Integer	Potencial de validez de la siembra	
area_total	Integer	Área sembrada total	
area_util	Integer	Área realmente utilizada	
id_usuario_servicio	Integer	Usuario responsable	
humedad_ambiente	Integer	Nivel de humedad ambiente	
humedad_suelo	Integer	Humedad del suelo	
precipitacion	Integer	Cantidad de lluvia	
radiacion_solar	Integer	Nivel de radiación	
velocidad_viento	Integer	Velocidad del viento	
ph_suelo	Integer	pH del suelo	
acceso_fuentes_hidricas	Boolean	Acceso a agua	
cantidad_sistemas_riego	Integer	Cantidad de sistemas de riego	
altitud	Integer	Altitud de la zona	
fecha_creacion	Timestamp	Fecha de creación del registro	NOT NULL
fecha_actualizacion	Timestamp	Última modificación	

Tabla: reportes

Campo	Tipo	Descripción	Restricciones
id_reporte	Serial	ID del reporte	PRIMARY KEY
fecha_creacion	Timestamp	Fecha del reporte	NOT NULL
id_usuario	Integer	Usuario que reporta	FK usuario

Tabla: plagas

Campo	Tipo	Descripción	Restricciones
id_plaga	Serial	ID de la plaga	PRIMARY KEY
nombre	Varchar	Nombre común de la plaga	NOT NULL
fecha_creacion	Timestamp	Fecha de registro	NOT NULL
id_usuario	Integer	Usuario creador	NOT NULL

Tabla: rel_siembra_plagas

Campo	Tipo	Descripción	Restricciones
id siembra	Integer	Referencia a siembra	FK siembra
id_plaga	Integer	Referencia a plaga	FK plagas

Tabla: recomendaciones_tecnicas

Campo	Tipo	Descripción	Restricciones
id_recomendacion	Serial	ID recomendación	PRIMARY KEY
nombre	Varchar	Título/Nombre	NOT NULL
fecha_creacion	Timestamp	Fecha	NOT NULL
id_usuario	Integer	Usuario creador	NOT NULL

Tabla: rel_siembra_recomendaciones_tecnicas

Campo	Tipo	Descripción	Restricciones
id_recomendacion	Integer	FK a recomendación	NOT NULL
id_siembra	Integer	FK a siembra	NOT NULL

Tabla: enfermedades

Campo	Tipo	Descripción	Restricciones
id_enfermedad	Serial	ID enfermedad	PRIMARY KEY
nombre	Varchar	Nombre	NOT NULL
id_zona	Integer	FK zona afectada	NOT NULL
fecha_creacion	Timestamp	Fecha registro	NOT NULL
id_usuario	Integer	Usuario responsable	NOT NULL

Tabla: rel_siembra_enfermedades

Campo	Tipo	Descripción	Restricciones
id_enfermedad	Integer	FK a enfermedad	NOT NULL
id_siembra	Integer	FK a siembra	NOT NULL

Tabla: rel_usuario_rol

Campo	Tipo	Descripción	Restricciones
id_usuario	Integer	FK a usuario	NOT NULL
id_rol	Integer	FK a rol	NOT NULL

Tabla: rel_zona_enfermedades

Campo	Tipo	Descripción	Restricciones
id_zona	Integer	FK zona	NOT NULL
id_enfermedad	Integer	FK enfermedad	NOT NULL

Tabla: municipios

Campo	Tipo	Descripción	Restricciones
id_municipio	Serial	ID único de municipio	PRIMARY KEY
nombre	Varchar	Nombre	NOT NULL
fecha_creacion	Timestamp	Fecha de creación	NOT NULL
id_zona	Integer	FK zona	NOT NULL
id_usuario	Integer	Usuario creador	NOT NULL

Tabla: rel_zona_plagas

Campo	Tipo	Descripción	Restricciones
id_zona	Integer	FK zona	NOT NULL
id_plaga	Integer	FK plaga	NOT NULL

Anexo 2

Diagrama Modulo Entidad Relación *(Enlace de la imagen)*

Anexo 3

Diagrama de flujo *(Enlace de la imagen)*

Anexo 4

Diagrama de contexto *(Enlace de la imagen)*

Anexo 5

Diagrama de contenedores módulo usuarios *(Enlace de la imagen)*

Anexo 6

Diagrama de contenedores módulo parametrización *(Enlace de la imagen)*

Anexo 7

Diagrama de contenedores módulo de gestión de datos *(Enlace de la imagen)*

Anexo 8

Diagrama de contenedores módulo de integración SIG *(Enlace de la imagen)*

Anexo 9

Diagrama dinámico de componentes en la nube *(Enlace de la imagen)*

Anexo 10

Diagrama dinámico de despliegue *(Enlace de la imagen)*

Anexo 11

Diagrama plataforma de gestión *(Enlace de la imagen)*

Anexo 12

Diagrama gestión respuesta base de datos *(Enlace de la imagen)*