UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA

Escuela Profesional de Ingeniería de Sistemas

Proyecto *Apis y Funciones Jarro_Valle*

Curso: *Tópicos de Base de Datos Avanzados*

Docente: *Mag. Patrick Cuadros*

Integrantes:

Jose Luis Jarro Cachi (2020067148)
Gustavo Alonso Valle Bustamante (2020066916)

Tacna – Perú
2024

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	MPV	ELV	ARV	10/10/2020	Versión Original

Sistema *Proyecto Apis y Funciones Jarro_Valle*
Documento de Visión

Versión *1.0*

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	MPV	ELV	ARV	10/10/2020	Versión Original

INDICE GENERAL

1. Introducción

El presente Documento de Visión describe el proyecto Apis y Funciones Jarro_Valle, el cual tiene como objetivo principal el desarrollo de una API para gestionar eventos dentro de un entorno académico. Este proyecto utiliza tecnologías modernas, como .NET Core, MongoDB, Docker, y GitHub Actions, con el fin de proporcionar una solución robusta y eficiente para el manejo de eventos mediante operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

La necesidad de este proyecto surge del deseo de ofrecer a los usuarios una herramienta sencilla y escalable para gestionar eventos de manera efectiva. Dado que los eventos son fundamentales en muchas actividades académicas, el sistema permite una gestión centralizada y de fácil acceso a los registros de eventos. La API se integra con MongoDB, una base de datos NoSQL, lo que permite una estructura flexible para almacenar la información de los eventos, y se empaqueta en contenedores Docker para facilitar su despliegue y portabilidad.

Este documento de visión tiene como propósito proporcionar una visión clara de los objetivos, funcionalidades y el alcance de la API Apis y Funciones Jarro_Valle, sirviendo como guía para su desarrollo, pruebas y despliegue. Además, define los términos técnicos, el flujo de trabajo del proyecto y las herramientas utilizadas para garantizar que todos los involucrados en el proyecto, desde los desarrolladores hasta los usuarios finales, tengan una comprensión clara del propósito y las características del sistema.

Con el fin de facilitar la adopción y el uso de esta API, el sistema incluye documentación interactiva a través de Swagger UI, permitiendo a los usuarios y desarrolladores explorar y probar la API de manera intuitiva.

1.1 Propósito

El propósito del proyecto **Apis y Funciones Jarro_Valle** es desarrollar una API robusta y escalable para la gestión de eventos dentro de un entorno académico. Esta API permite a los usuarios crear, leer, actualizar y eliminar (CRUD) eventos de manera eficiente, utilizando tecnologías modernas como **.NET Core** para el desarrollo de la API, **MongoDB** como base de datos NoSQL para el almacenamiento de información,

y **Docker** para facilitar el despliegue y la portabilidad del sistema.

1.1. Alcance

El proyecto Apis y Funciones Jarro_Valle tiene como alcance la creación de una API para la gestión de eventos en un entorno académico, utilizando .NET Core como framework de desarrollo, MongoDB como base de datos NoSQL, y Docker para la implementación y portabilidad del sistema. La API estará disponible para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los registros de eventos, que son almacenados en la base de datos MongoDB.

Funcionalidades Clave:

1. Gestión de Eventos:
    - Crear nuevos eventos académicos.
    - Leer la información de eventos existentes.
    - Actualizar los detalles de eventos previamente creados.
    - Eliminar eventos de la base de datos.
2. Interacción con la Base de Datos (MongoDB):
    - La API se conecta a MongoDB, donde se almacenan los eventos. MongoDB se utiliza por ser una base de datos NoSQL, que ofrece flexibilidad y escalabilidad para manejar los datos de eventos.
3. Despliegue con Docker:
    - El proyecto está empaquetado en contenedores Docker, lo que permite un despliegue sencillo y consistente en diferentes entornos. Docker facilita la portabilidad del sistema, permitiendo que la API pueda ejecutarse en cualquier máquina que soporte Docker, sin importar el entorno o la configuración local.
4. Automatización y CI/CD:
    - GitHub Actions se emplea para la integración continua (CI) y despliegue continuo (CD) del proyecto. Esto asegura que las versiones

más recientes del código se construyan, prueben y desplieguen automáticamente en el entorno de producción, garantizando la calidad y disponibilidad del sistema.

5. Documentación Interactiva con Swagger:
    - Se incluye Swagger UI para la documentación interactiva de la API, permitiendo a los desarrolladores explorar y probar las funcionalidades de la API de manera fácil y visual. Esto facilita la integración de la API en otros sistemas y aplicaciones.

1.3 Definiciones, Siglas y Abreviaturas

A continuación se presentan las definiciones, siglas y abreviaturas utilizadas en este documento, que son esenciales para la comprensión del proyecto Apis y Funciones Jarro_Valle:

Definiciones

- API (Interfaz de Programación de Aplicaciones): Es un conjunto de reglas y especificaciones que permiten que una aplicación interactúe con otras. En este proyecto, la API permite gestionar los eventos mediante operaciones CRUD (Crear, Leer, Actualizar, Eliminar).
- CRUD (Crear, Leer, Actualizar, Eliminar): Son las cuatro operaciones básicas que se pueden realizar sobre los datos en una base de datos. Este proyecto permite realizar estas operaciones en los registros de eventos almacenados en MongoDB.
- MongoDB: Es una base de datos NoSQL orientada a documentos. En este proyecto, se utiliza para almacenar los registros de eventos. Su modelo flexible de datos permite una fácil escalabilidad y manejo de grandes volúmenes de datos.

Siglas y Abreviaturas

- API: Application Programming Interface (Interfaz de Programación de Aplicaciones).
- CRUD: Create, Read, Update, Delete (Crear, Leer, Actualizar, Eliminar).

- MongoDB: Database Management System (Sistema de Gestión de Bases de Datos NoSQL).
- Docker: Plataforma para empaquetar y ejecutar aplicaciones.
- GitHub Actions: Herramienta de CI/CD para automatización de flujos de trabajo.
- Swagger UI: Interfaz de documentación interactiva de APIs.
- CI: Continuous Integration (Integración Continua).
- CD: Continuous Deployment (Despliegue Continuo).
- .NET Core: Framework de desarrollo de aplicaciones basado en .NET, utilizado para crear la API.

1.4 Referencias

A continuación se presentan las referencias clave utilizadas para el desarrollo del proyecto Apis y Funciones Jarro_Valle.

a) Documentación Oficial de .NET Core

   a. URL:https://learn.microsoft.com/en-us/dotnet/core/ Descripción: Guía oficial de Microsoft para el desarrollo de aplicaciones con .NET Core, el framework utilizado para la creación de la API.

b) MongoDB Documentation

   a. URL:https://www.mongodb.com/docs/ Descripción: Documentación completa de MongoDB, la base de datos NoSQL

c) Docker Documentation

   a. Descripción: Documentación oficial de Docker, la plataforma utilizada para la creación y despliegue de contenedores. Explica cómo empaquetar aplicaciones y ejecutar contenedores de manera eficiente.

d) Swagger UI Documentation

   a. Descripción: Documentación de Swagger UI, la herramienta utilizada para generar una interfaz interactiva de documentación y prueba de la API.

e) Docker Hub

   a. URL:Descripción: Plataforma de distribución de imágenes Docker, utilizada para almacenar y distribuir las imágenes de contenedores creadas en el proyecto.

f) GitFlow Workflow

a. URL : Descripción: Recurso que explica el flujo de trabajo GitFlow, el cual es utilizado en el proyecto para la gestión de ramas en el control de versiones con Git.

1.5 Visión General

El proyecto Apis y Funciones Jarro_Valle tiene como objetivo principal el desarrollo de una API para la gestión de eventos académicos, proporcionando una solución robusta, escalable y fácil de usar para administrar eventos en un entorno académico. Esta API está diseñada para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre registros de eventos almacenados en MongoDB, una base de datos NoSQL, con el fin de permitir un manejo eficiente de los datos en entornos educativos.

Alcance del Proyecto: La API permite a los usuarios gestionar eventos mediante operaciones CRUD. Los eventos pueden ser de diversa índole, como conferencias, seminarios o actividades académicas. Sin embargo, el alcance del proyecto se limita a la gestión de estos eventos, sin incluir funcionalidades adicionales como la gestión de usuarios o la integración con otros sistemas académicos más allá de la base de datos de eventos.

2. Posicionamiento

2.1 Oportunidad de negocio

La gestión de eventos es una actividad esencial dentro de las instituciones educativas, especialmente en el ámbito académico, donde se realizan conferencias, seminarios, talleres y actividades extracurriculares. Sin embargo, muchas veces los procesos involucrados en la gestión de estos eventos son manuales, fragmentados o dependen de sistemas obsoletos que no ofrecen la eficiencia ni la escalabilidad necesarias. La oportunidad de negocio que ofrece el proyecto Apis y Funciones Jarro_Valle radica en la automatización y centralización de la gestión de eventos académicos a través de una API moderna, escalable y fácil de integrar.

Al proporcionar una plataforma que facilite la creación, consulta, actualización y eliminación de eventos a través de una API, el proyecto satisface la necesidad de optimizar la administración de eventos dentro de las instituciones académicas. Las principales oportunidades de negocio incluyen:

- Reducción de la Carga Administrativa:
- Escalabilidad y Flexibilidad:
- Acceso Remoto y en Tiempo Real:
- Integración con Otros Sistemas:
- Facilidad de Uso:

2.2 Definición del problema

Las instituciones académicas, como universidades y centros educativos, enfrentan varios desafíos al gestionar eventos de forma manual o mediante sistemas tradicionales. Estos problemas incluyen la falta de automatización, la ineficiencia en el manejo de datos y la dificultad para acceder a la información en tiempo real. Algunos de los problemas clave que este proyecto busca resolver incluyen:

a) Gestión Manual de Eventos:
b) Falta de Centralización:
c) Escalabilidad Limitada:
d) Dificultad en la Integración con Otros Sistemas:
e) Falta de Accesibilidad y Visibilidad en Tiempo Real:

El proyecto Apis y Funciones Jarro_Valle resuelve estos problemas al proporcionar una API que centraliza la información de eventos en una base de datos flexible como MongoDB. La automatización de las operaciones CRUD permite a las instituciones gestionar eventos de manera más eficiente, mientras que la escalabilidad y flexibilidad de la solución garantizan que pueda adaptarse al crecimiento y a las necesidades cambiantes de la institución. Además, la facilidad de integración con

otros sistemas académicos y la documentación interactiva proporcionada por Swagger UI mejoran la accesibilidad y la adopción del sistema, ofreciendo una solución moderna y accesible para la gestión de eventos.

3. Descripción de los interesados y usuarios

3.1 Resumen de los interesados

Los interesados son todas aquellas personas, grupos o entidades que tienen un interés en el desarrollo y éxito del proyecto, pero que no necesariamente interactúan directamente con la solución tecnológica. En el caso del proyecto Apis y Funciones Jarro_Valle, los principales interesados incluyen:

a) Institución Educativa (Universidades/Facultades):
b) Desarrolladores del Proyecto:
c) Equipo de Infraestructura y Soporte Técnico
d) Usuarios Administrativos:
e) Proveedores de Tecnología:

3.2 Resumen de los usuarios

Los usuarios son aquellos que interactúan directamente con la API, ya sea para gestionar eventos, obtener información o realizar tareas específicas. En el caso de este proyecto, los principales usuarios incluyen:

a) Administradores de Eventos: Son los usuarios responsables de crear, actualizar, y eliminar eventos. También gestionan las fechas, descripciones y resultados de los eventos académicos.
b) Estudiantes: Aunque no tienen acceso a la gestión de eventos, los estudiantes pueden interactuar con la API para consultar eventos futuros o pasados, conocer los resultados de los eventos y obtener detalles relacionados.
c) Desarrolladores e Integradores: Usuarios técnicos que se encargarán de integrar la API con otras plataformas académicas o servicios adicionales, como sistemas de inscripción o plataformas de gestión de cursos.

3.3 Entorno de usuario

El entorno de usuario se refiere al contexto en el cual los usuarios interactúan con la API, considerando las herramientas y tecnologías disponibles. En este proyecto, el

entorno de usuario incluye:

a) Interfaz de Usuario Web/Móvil: Los usuarios acceden a la API a través de una interfaz web o móvil que consume los puntos finales de la API para visualizar y gestionar eventos.
b) API RESTful: Los usuarios interactúan con la API a través de solicitudes HTTP.
c) Swagger UI: La API está documentada usando Swagger UI, lo que facilita a los usuarios y desarrolladores la visualización y prueba de los diferentes puntos finales de la API.
d) Entorno de Desarrollo y Despliegue: La API se desarrolla en un entorno .NET Core, y se despliega en contenedores Docker, lo que asegura que los desarrolladores puedan trabajar en un entorno controlado y escalable.

3.4 Perfiles de los interesados

Institución Educativa:

- Responsabilidades: Supervisar el correcto uso y la implementación del sistema.
- Intereses: Garantizar la eficiencia en la gestión de eventos académicos y la satisfacción de los usuarios.
- Expectativas: Obtener una solución escalable, segura y fácil de integrar con otros sistemas existentes.

Desarrolladores del Proyecto:

- Responsabilidades: Desarrollar y mantener la API, implementando nuevas funcionalidades y resolviendo problemas técnicos.
- Intereses: Asegurar la calidad, rendimiento y escalabilidad de la API.
- Expectativas: Desarrollar una API eficiente y documentada, con un código limpio y fácil de mantener.

Equipo de Infraestructura y Soporte Técnico:

- Responsabilidades: Gestionar la infraestructura tecnológica, servidores y bases de datos.

- Intereses: Asegurar que la infraestructura sea confiable, escalable y esté bien configurada para el rendimiento de la API.
- Expectativas: Que el sistema sea fácil de implementar, administrar y escalar.

3.5 Perfiles de los Usuarios

Administrador de Eventos:

- Responsabilidades: Gestionar y organizar eventos académicos, actualizar detalles de los eventos, y realizar tareas CRUD a través de la API.
- Intereses: Tener una herramienta eficiente para gestionar eventos de manera centralizada y fácil de usar.
- Expectativas: Contar con una interfaz simple y funcional que permita realizar tareas administrativas de manera rápida y sin errores.

Estudiantes:

- Responsabilidades: Consultar eventos académicos, ver detalles de los eventos a los que pueden asistir y revisar resultados.
- Intereses: Estar informados sobre los eventos académicos y sus resultados.
- Expectativas: Tener acceso a una lista clara de eventos disponibles y resultados actualizados.

3.6 Necesidades de los interesados y usuarios

Interesados:

1. Institución Educativa: Necesita una solución que centralice la gestión de eventos, ahorre tiempo y recursos, y se integre fácilmente con otros sistemas.
2. Desarrolladores del Proyecto: Necesitan una arquitectura escalable, un código bien estructurado y herramientas que faciliten el despliegue y mantenimiento de la API.
3. Equipo de Infraestructura y Soporte Técnico: Necesita una infraestructura confiable, fácil de gestionar y con capacidad para escalar según sea necesario.
4. Proveedores de Tecnología: Necesitan que sus servicios sean utilizados de

manera eficiente y sin inconvenientes técnicos.

Usuarios:

1. Administrador de Eventos: Necesita una herramienta sencilla para gestionar eventos sin problemas técnicos ni complejidades.
2. Estudiantes: Necesitan acceso fácil a la información sobre eventos y resultados de manera rápida y sin complicaciones.

3. Vista General del Producto

La API de Gestión de Eventos Jarro_Valle es una plataforma diseñada para gestionar de manera eficiente y centralizada los eventos .

3.1 Perspectiva del producto

La API está diseñada para integrarse fácilmente con otras plataformas existentes en el entorno académico, como sistemas de inscripción o plataformas de gestión de cursos. Además, es accesible tanto a través de interfaces web y móviles, brindando flexibilidad en su uso.

La arquitectura de la API es escalable y flexible, permitiendo su implementación en instituciones de cualquier tamaño, desde universidades pequeñas hasta grandes universidades con múltiples facultades y eventos. El uso de Docker y GitHub Actions facilita la automatización del proceso de construcción y despliegue, lo que mejora la eficiencia operativa y garantiza un sistema robusto.

3.2 Resumen de capacidades

Las principales capacidades del producto son las siguientes:

a) Gestión de Eventos:
    a. Crear, leer, actualizar y eliminar eventos a través de solicitudes HTTP.
    b. Permitir la gestión de atributos de eventos como nombre, fecha, facultad, descripción, entre otros.
    c. Asociar resultados y detalles adicionales a los eventos académicos.
b) Acceso Remoto:
    a. Acceso a la API a través de Swagger UI, que permite visualizar y probar los diferentes puntos finales de la API.

b. Accesibilidad desde aplicaciones móviles, web y otros sistemas que consumen la API.

c) Integración con Otros Sistemas:

a. Posibilidad de integración con otros sistemas académicos, como sistemas de inscripción o plataformas de gestión de estudiantes.

b. Facilidad para que los desarrolladores integren la API con servicios de terceros a través de su arquitectura RESTful.

d) Escalabilidad:

a. Despliegue en contenedores Docker, permitiendo una escalabilidad eficiente tanto vertical como horizontal.

b. Almacenamiento en MongoDB, que permite gestionar grandes volúmenes de datos de manera flexible y escalable.

e) Autenticación y Seguridad:

a. Implementación de mecanismos de autenticación para proteger el acceso a la API y garantizar que solo los usuarios autorizados puedan realizar operaciones sensibles.

f) Interfaz de Usuario Interactiva:

a. Uso de Swagger UI para interactuar con la API, facilitando su aprendizaje y prueba.

b. Interfaces móviles y web para los usuarios finales (administradores, docentes, estudiantes).

3.3 Suposiciones y dependencias

Suposiciones

1. Conocimiento Técnico de los Usuarios: Se asume que los administradores y desarrolladores tendrán el conocimiento técnico necesario para integrar y utilizar la API, especialmente en lo que respecta a las operaciones CRUD.
2. Consistencia en los Datos: Se supone que la base de datos MongoDB se mantendrá consistente y sin conflictos, garantizando que las operaciones de la API sean fiables y consistentes.
3. Acceso Web y Móvil: Se supone que los usuarios tendrán acceso a dispositivos móviles o equipos con navegador web para interactuar con la API de manera remota.

Dependencias

1. Docker: La API depende de Docker para su despliegue y gestión en contenedores. La infraestructura debe ser capaz de soportar la ejecución de contenedores Docker.
2. MongoDB: La API depende de MongoDB como su base de datos principal para almacenar la información de eventos. Cualquier problema con MongoDB podría afectar el funcionamiento de la API.
3. Swagger UI: La documentación interactiva de la API se genera a través de Swagger UI, lo que permite a los usuarios probar y entender fácilmente los puntos finales de la API.

3.4 Costos y precios

El modelo de costos y precios dependerá principalmente de los siguientes factores:

Infraestructura de Servidores:

Si la API se aloja en servidores propios de la institución educativa, se tendrán costos asociados al mantenimiento de la infraestructura.

Si se opta por usar un servicio en la nube (AWS, Azure, etc.), habrá costos relacionados con la provisión de instancias de servidor y almacenamiento.

Licencias y Herramientas de Desarrollo:

MongoDB: La versión básica de MongoDB es de código abierto, pero si se necesita una versión con más características empresariales (como soporte o mayor capacidad), pueden aplicarse costos adicionales.

Docker: Docker es gratuito para la mayoría de los usos, pero las empresas pueden optar por suscripciones premium para obtener características avanzadas.

GitHub Actions: Dependiendo del uso, GitHub Actions puede tener costos adicionales si se superan los límites gratuitos proporcionados por GitHub.

Desarrollo y Mantenimiento:

Los costos asociados al desarrollo, mantenimiento y actualizaciones periódicas de la API también deben ser considerados. Esto incluye el tiempo y los recursos de los desarrolladores, así como la capacitación de usuarios y administradores.

Soporte y Consultoría:

En caso de que se necesite soporte adicional o consultoría técnica para la implementación, integraciones o personalización de la API, se pueden incurrir en costos por servicios externos.

Modelo de Precios:

El modelo de precios podría ser basado en la suscripción, donde las instituciones educativas pagan una tarifa mensual o anual por el uso de la API. Los precios podrían variar dependiendo del número de usuarios, eventos gestionados o volumen de datos. Alternativamente, la institución podría elegir una opción de pago único, donde se paga una tarifa fija por el acceso y uso de la API.

3.5 Licenciamiento e instalación

Licenciamiento

El producto Apis y Funciones Jarro_Valle puede estar licenciado bajo una licencia de código abierto (por ejemplo, MIT License o GPLv3), lo que permite a las instituciones educativas modificar el código, realizar personalizaciones y contribuir al proyecto.

Instalación

1. Requisitos Previos:
    - Docker: Para ejecutar la API en un contenedor, se requiere tener instalado Docker en el sistema.
    - MongoDB: Debe configurarse una instancia de MongoDB para almacenar los datos de los eventos.
    - .NET Core: La API está desarrollada utilizando .NET Core, por lo que debe estar instalado en el sistema para desarrolladores que deseen realizar cambios en el código fuente.
2. Instrucciones de Instalación:
    - Clonar el repositorio del proyecto desde GitHub.
    - Construir la imagen Docker del proyecto utilizando el comando docker build -t eventos_vj ..
    - Ejecutar el contenedor Docker utilizando docker run -d -p 9091:80 eventos_vj.
    - Acceder a la API a través de la URL proporcionada en Swagger UI: http://localhost:9091/swagger.

La instalación es sencilla y está diseñada para ser ejecutada en entornos locales o de producción con configuraciones mínimas.

4. Características del producto

La API de Gestión de Eventos Jarro_Valle está diseñada para proporcionar una solución eficiente y flexible para la gestión de eventos académicos. A continuación, se describen las características principales que hacen de esta API una herramienta robusta y fácil de usar.

4.1 Gestión de Eventos

Operaciones CRUD:

La API permite realizar operaciones de Crear, Leer, Actualizar y Eliminar eventos académicos mediante solicitudes HTTP estándar (GET, POST, PUT, DELETE).

Cada evento tiene atributos como: nombre, fecha de inicio, fecha de término, facultad, resultado y descripción.

- Creación de Eventos:
- Actualización de Eventos:
- Eliminación de Eventos:

- Acceso a la

  API Acceso

  Restful:

  - La API sigue el estilo de arquitectura REST, lo que garantiza una comunicación eficiente entre el cliente (móvil, web o backend) y el servidor.

- Swagger UI:
  - Se ha integrado Swagger UI para proporcionar una interfaz interactiva donde los desarrolladores y usuarios pueden probar los puntos finales de la API.
  - Swagger proporciona una visualización clara de todos los endpoints disponibles, sus parámetros y respuestas esperadas.

4.2 Integración con MongoDB

Base de Datos NoSQL:

Los eventos se almacenan en una base de datos MongoDB, una base de datos NoSQL, que es flexible y adecuada para manejar grandes volúmenes de datos no estructurados.

MongoDB facilita el almacenamiento y la consulta rápida de eventos, incluso

en instituciones con una gran cantidad de eventos y usuarios.

Modelo de Datos Flexible:

La base de datos se adapta a cambios en el modelo de datos sin necesidad de realizar migraciones complejas, lo que permite la evolución del sistema conforme cambian los requisitos.

4.3 Despliegue y Escalabilidad

Docker:

La API está empaquetada en un contenedor Docker, lo que facilita su despliegue en diferentes entornos (locales, servidores de producción, plataformas en la nube).

Docker permite la escalabilidad horizontal, lo que significa que se pueden agregar más contenedores según sea necesario para manejar el aumento en el tráfico o la carga.

4.4 Interfaz de Usuario

Interfaces Web y Móviles:

Aunque la API está diseñada para ser consumida por otros sistemas, también ofrece interfaces de usuario web y móviles para los administradores de eventos, docentes y otros usuarios que deseen interactuar directamente con la API.

Documentación Interactiva:

La API está completamente documentada utilizando Swagger, lo que permite a los desarrolladores y usuarios acceder a documentación interactiva, realizar pruebas de los puntos finales y entender cómo interactuar con la API sin necesidad de leer documentación extensa.

4.5 Validación y Manejo de Errores

Validaciones de Entrada:

La API realiza validaciones de entrada para asegurar que los datos recibidos sean correctos y completos antes de ser procesados o almacenados en la base de datos.

Las validaciones incluyen la comprobación de campos obligatorios, formatos de fecha, y coherencia entre los datos proporcionados.

Manejo de Errores:

La API proporciona respuestas claras y detalladas en caso de errores, facilitando la resolución de problemas.

Se manejan diferentes tipos de errores, como errores de validación, errores de autenticación y errores de conexión a la base de datos.

4.6 Seguridad

- Autenticación y Autorización:
  - La API implementa un sistema de autenticación utilizando JSON Web Tokens (JWT) para validar y autorizar a los usuarios.
  - Esto garantiza que solo los usuarios autenticados y autorizados puedan realizar ciertas operaciones, como crear, actualizar o eliminar eventos.
- Protección de Datos Sensibles:
  - Los datos sensibles, como contraseñas o información personal, se manejan de forma segura utilizando técnicas de cifrado.
  - Se sigue un enfoque de seguridad en capas para proteger tanto la comunicación como el almacenamiento de datos.

4.7 Soporte Multiplataforma

Compatibilidad con Diferentes

Plataformas:

La API es compatible con aplicaciones móviles y web, lo que permite a los usuarios interactuar con el sistema desde diferentes dispositivos y entornos.

Los desarrolladores pueden integrar la API en diferentes sistemas académicos, independientemente de la plataforma que utilicen.

Manejo de Diferentes Versiones:

Se mantiene un control de versiones en la API para permitir a los desarrolladores adaptar sus sistemas a las versiones más recientes o utilizar versiones anteriores si lo requieren.

4.8 Desempeño y Optimización

Desempeño Rápido:

Se implementan técnicas de optimización en las consultas a la base de datos y en la comunicación entre los componentes del sistema para garantizar tiempos de respuesta rápidos.

El uso de MongoDB como base de datos NoSQL ayuda a gestionar grandes volúmenes de datos sin comprometer el rendimiento.

Escalabilidad:

La arquitectura de la API permite escalar el sistema fácilmente para satisfacer la demanda en entornos de producción con alto tráfico o grandes cantidades de eventos.

5. Restricciones

Las restricciones del proyecto son los límites o condiciones que afectan el desarrollo, funcionamiento y uso de la API de gestión de eventos. A continuación, se describen las principales restricciones que deben tenerse en cuenta durante el diseño y operación del sistema.

Restricciones Tecnológicas

- Dependencia de MongoDB:
- Compatibilidad con .NET Core:
- Dockerización y Contenedores:

Restricciones de Rendimiento

- Tiempos de Respuesta en Consultas:
- Escalabilidad Limitada en el Ambiente Local:

Restricciones de Seguridad

- Autenticación y Autorización Básica:
- Protección de Datos Sensibles:

Restricciones de Infraestructura

- Dependencia de AWS o Plataforma Similar:
- Uso de Docker Hub:

Restricciones Funcionales

- Acceso y Control de Usuarios Limitado:
- Soporte para Solo un Tipo de Evento:

Restricciones de Usuario

- Interfaz Limitada:
- Dependencia del Entorno Local para Pruebas:

Restricciones Legales y de Licenciamiento

- Licencia de Uso:
  El proyecto está licenciado bajo una licencia de código abierto, lo que significa que los usuarios pueden modificar y distribuir el código según los términos de la licencia.
- Cumplimiento Normativo:
  El sistema debe cumplir con las leyes locales relacionadas con la protección de datos personales (como la Ley de Protección de Datos Personales en Perú o el GDPR en Europa).

6. Rangos de calidad

Los rangos de calidad describen los criterios y estándares que el sistema debe cumplir para garantizar su efectividad, fiabilidad, usabilidad, seguridad y rendimiento. Estos rangos definen los niveles de calidad que se esperan del sistema, y son esenciales para evaluar el éxito del producto y para asegurar que se cumplan las expectativas de los usuarios finales. A continuación, se detallan los rangos de calidad para la API de gestión de eventos.

Fiabilidad

- Disponibilidad:
- Tolerancia a Fallos:
- Pruebas de Confiabilidad:

Rendimiento

- Tiempo de Respuesta:
- Escalabilidad:
- Capacidad de Manejo de Carga:

Usabilidad

- Documentación:
    - Descripción de cada endpoint.
    - Ejemplos de solicitudes y respuestas.
    - Instrucciones claras sobre cómo utilizar los parámetros de la API.
- Interfaz de Usuario:
    - Si se proporciona una interfaz de usuario web o móvil, esta debe ser intuitiva y fácil de usar, permitiendo a los usuarios gestionar eventos sin dificultad.
- Accesibilidad:
    - El sistema debe ser accesible para usuarios con discapacidad visual o auditiva.

Seguridad

- Autenticación y Autorización:
- Protección de Datos Sensibles:
- Protección contra Ataques Comunes:

Mantenibilidad

- Código Modular y Legible:
- Pruebas Unitarias:
- Actualización y Soporte:

Compatibilidad

- Compatibilidad con Navegadores y Dispositivos:
- Compatibilidad con Versiones de API:

Costos

- Eficiencia en Costos:
  - La implementación de la solución debe ser eficiente en términos de recursos, para minimizar el uso de infraestructura y costos operativos.
  - El uso de tecnologías como Docker y MongoDB debe garantizar que el sistema sea escalable sin incurrir en costos excesivos de infraestructura, especialmente en escenarios de alta demanda.
- Licencias de Software:
  - El uso de tecnologías de código abierto como .NET Core y MongoDB debe asegurarse de que no haya costos de licencias imprevistos. Además, cualquier dependencia de terceros debe ser bien gestionada para evitar costos adicionales.

7. Precedencia y Prioridad

La precedencia y prioridad son conceptos fundamentales en la planificación y gestión de proyectos, especialmente al desarrollar un producto complejo como una API para la gestión de eventos. Definir claramente las prioridades y la relación de precedencia entre las tareas y funcionalidades del sistema es esencial

para la correcta ejecución del proyecto y para garantizar que los recursos se asignen de manera efectiva.

Precedencia

La precedencia se refiere a la relación temporal y dependiente entre las actividades del proyecto, en la que ciertas tareas deben completarse antes de que otras puedan iniciarse. Es crucial entender qué elementos dependen de otros para garantizar que las fases del desarrollo se sigan de forma lógica y eficiente.

Relación de Precedencia para el Proyecto:

1. Análisis y Diseño Inicial:
    - El análisis de requerimientos y el diseño de la arquitectura deben completarse antes de la codificación de la API
2. Desarrollo de la API:
    - El desarrollo de la API debe seguir el diseño previamente establecido. Las funcionalidades críticas de la API, como las operaciones CRUD
3. Integración de Base de Datos (MongoDB):
    - La configuración de MongoDB debe completarse antes de poder comenzar las operaciones de interacción con la base de datos.
4. Desarrollo de la Interfaz de Usuario:
    - La interfaz de usuario (web o móvil) debe ser desarrollada después de que la API esté funcional.
5. Pruebas Unitarias y de Integración:
    - Las pruebas deben comenzar después de que se haya completado la funcionalidad básica del sistema
6. Despliegue y Publicación en DockerHub:
    - El despliegue de la API debe realizarse después de completar las pruebas y antes de que la API sea accesible para los usuarios finales.

Prioridad

Prioridades en el Proyecto:

1. Alta Prioridad:
    - Funcionalidad CRUD (Crear, Leer, Actualizar, Eliminar):
    - Conexión de la API a la base de datos MongoDB:
    - Autenticación y Autorización:

2. Media Prioridad:
    - Manejo de Errores y Respuestas Adecuadas:
    - Documentación de la API (Swagger):
3. Baja Prioridad:
    - Interfaz de Usuario (Web/Móvil):
    - Optimización y Escalabilidad:

8. Otros requerimientos del producto

En esta sección se describen los **otros requerimientos del producto** que no están directamente relacionados con la funcionalidad básica del sistema, pero que son esenciales para asegurar su correcta operación, su alineación con las normativas legales, su calidad y seguridad, así como su capacidad de comunicación con otros sistemas o plataformas.

b) Estandares legales

El sistema debe cumplir con todas las normativas legales y regulaciones locales e internacionales aplicables a la gestión de datos, la privacidad de los usuarios, y la operatividad de la API en el entorno académico y empresarial. Los requisitos legales pueden variar según la ubicación de los usuarios y la naturaleza de los datos gestionados. A continuación, se detallan los estándares legales a considerar:

- Protección de Datos Personales (GDPR):
    - Si el sistema maneja datos personales de usuarios, debe cumplir con el Reglamento General de Protección de Datos (GDPR, por sus siglas en inglés) de la Unión Europea, que regula el almacenamiento, tratamiento y uso de datos personales de los ciudadanos de la UE. Esto incluye:
        - Consentimiento explícito para recolectar datos.
        - Derecho a la eliminación de datos (derecho al olvido).
        - Transparencia sobre cómo se utilizan los datos personales.
- Ley de Protección de Datos Personales en Perú (Ley N° 29733):
    - En el contexto peruano, el sistema también debe alinearse con la Ley N° 29733 que regula la protección de los datos personales. Esto implica:
        - Consentimiento informado para la recolección de datos personales.

        - Implementación de medidas de seguridad adecuadas para proteger los datos almacenados.

- Licencias de Software:
    - El sistema debe cumplir con las licencias de software de las tecnologías utilizadas (por ejemplo, .NET Core, MongoDB, Docker), garantizando que el uso de estas herramientas no infrinja las leyes de propiedad intelectual ni los derechos de los creadores del software.

c) Estandares de comunicación

Los estándares de comunicación son fundamentales para garantizar la interoperabilidad entre los componentes del sistema y la correcta integración con otros servicios y plataformas. A continuación, se mencionan algunos de los estándares de comunicación clave para el proyecto:

- Protocolo HTTP/HTTPS:
    - La API debe comunicarse utilizando HTTP o, preferiblemente, HTTPS para garantizar la seguridad de las transacciones. HTTPS debe estar habilitado para garantizar que todas las solicitudes y respuestas se cifren durante la transmisión.
- JSON como Formato de Intercambio de Datos:
    - La API debe utilizar JSON (JavaScript Object Notation) como formato estándar para la entrada y salida de datos. JSON es ampliamente utilizado por su facilidad de lectura y compatibilidad con la mayoría de los lenguajes de programación.
- RESTful API:
    - La API debe seguir los principios de un diseño RESTful (Representational State Transfer), lo que implica:
        - Uso de métodos HTTP estándar (GET, POST, PUT, DELETE).
        - URLs claras y descriptivas que representen recursos (por ejemplo, /api/eventos).
        - Respuestas coherentes con códigos de estado HTTP

apropiados (200 OK, 201 Created, 400 Bad Request, etc.).

d) Estandaraes de cumplimiento de la plataforma

El producto debe estar alineado con los estándares de cumplimiento de la plataforma para garantizar su correcta operación y conformidad en los entornos en los que se despliegue. Esto incluye:

- Cumplimiento con Docker
- Despliegue en la Nube
- Cumplimiento con las mejores prácticas de CI/CD
- Compatibilidad con Sistemas Operativos y Navegadores:

e) Estandaraes de calidad y seguridad

Los estándares de calidad y seguridad son fundamentales para asegurar que el sistema funcione correctamente y esté protegido frente a vulnerabilidades. Estos incluyen:

- Normas de Calidad del Código:
- Pruebas Automatizadas:
- Seguridad de la API:
- Protección de Datos:
- Manejo de Errores y Respuestas Seguras:

CONCLUSIONES

- El sistema está diseñado para cumplir tanto con los requerimientos funcionales (gestionar eventos académicos) como con los no funcionales (seguridad, escalabilidad, usabilidad y rendimiento). El uso de tecnologías como .NET Core y MongoDB permite crear un sistema ágil, escalable y robusto.
- Se utilizarán principios de diseño RESTful para la API, asegurando que las interacciones sean intuitivas y eficientes. La implementación de GitFlow y GitHub Actions asegura un flujo de trabajo eficiente en el desarrollo y despliegue continuo del proyecto.
- El proyecto enfatiza la importancia de seguridad mediante el uso de mecanismos de autenticación robustos como JWT, así como el cumplimiento de normativas de protección de datos como GDPR y la Ley de Protección de Datos Personales en Perú.
- La integración de Docker y DockerHub facilita la automatización del despliegue, lo que asegura que el producto sea fácilmente distribuible y escalable en entornos de producción, alineado con los estándares de la plataforma.
- La solución proporciona una interfaz accesible a través de Swagger UI, permitiendo una fácil integración para los desarrolladores y usuarios interesados en interactuar con la API. La solución se despliega en plataformas en la nube, permitiendo una alta disponibilidad.

RECOMENDACIONES

- Se recomienda realizar pruebas unitarias e integradas para validar cada funcionalidad de la API, garantizando que todas las operaciones CRUD funcionen correctamente antes de poner el sistema en producción.
- Se debe establecer un sistema de monitoreo para observar el desempeño de la API y detectar posibles problemas en tiempo real. Las actualizaciones periódicas y el mantenimiento de las dependencias de software son cruciales para mantener el sistema seguro y funcional.
- Aunque la arquitectura está diseñada para ser escalable, es importante realizar pruebas de carga para verificar el rendimiento del sistema bajo condiciones de alta demanda. La capacidad de escalar horizontalmente utilizando contenedores Docker debe ser evaluada y ajustada según las necesidades.
- La documentación del proyecto debe ser continua y mejorada, tanto para desarrolladores como para usuarios finales, asegurando que todos los aspectos del sistema estén claramente explicados y sean fácilmente accesibles.
- Es crucial que el equipo de desarrollo se mantenga actualizado con las normativas de protección de datos y las leyes locales o internacionales aplicables. Esto incluye revisar los procesos de recolección y almacenamiento de datos personales, y realizar auditorías de seguridad periódicas.

BIBLIOGRAFIA

- Microsoft. .NET Documentation. https://learn.microsoft.com/en-us/dotnet/
- MongoDB, Inc. MongoDB Documentation. https://www.mongodb.com/docs/
- European Commission. General Data Protection Regulation (GDPR). https://gdpr.eu/
- PERÚ. Ley N° 29733 - Ley de Protección de Datos Personales. https://www.gob.pe/institucion/jne/normas-legales/31699-ley-29733

- Docker, Inc. Docker Documentation. https://docs.docker.com/
- GitHub, Inc. GitHub Actions Documentation. https://docs.github.com/en/actions

WEBGRAFIA

- Docker Hub. Repositorio de Docker Images. https://hub.docker.com/
- Swagger. Swagger API Documentation. https://swagger.io/
- GitHub. Repositorio del Proyecto SI-8811. https://github.com/tuusuario/proyecto-si-8811.git
- MongoDB Atlas. MongoDB Atlas Cloud Database. https://www.mongodb.com/cloud/atlas
- Elastic Beanstalk Documentation. https://aws.amazon.com/es/elasticbeanstalk/