UNIVERSIDAD DON BOSCO

TECNICO EN INGENIERIA EN COMPUTACION

INTEGRANTES:

Francisco Antonio López Mejía	LM240054
Tatiana Gabriela Sánchez Hernández	SH191154
Josue Rafael Siliezar Chavez	SC240158
Angel Marcelo Delgado Estrada	DE241507

MATERIA:

Desarrollo de Aplicaciones con Software Propietario DAS901 G01T

DOCENTE:

Ing. Karen Medrano

FECHA DE ENTREGA: 13 DE OCTUBRE 2024

Introducción

Este documento describe el desarrollo de una aplicación web para la gestión de eventos y tareas colaborativas en una empresa organizadora de eventos. El software está basado en una arquitectura de microservicios, lo que permite una modularidad eficiente y un mantenimiento flexible. Las funciones clave incluyen la creación y administración de eventos, asignación de tareas, y notificaciones automáticas. Esta aplicación está diseñada para mejorar la organización y seguimiento de tareas, facilitando la colaboración entre los miembros de la empresa.

Objetivo General

Desarrollar una aplicación web en ASP.NET Core que permita a los usuarios gestionar eventos y tareas colaborativas dentro de una empresa organizadora de eventos, utilizando una arquitectura de microservicios para asegurar escalabilidad y eficiencia.

Objetivos Específicos

Implementar un sistema de gestión de eventos que permita a los usuarios crear, modificar, eliminar y visualizar eventos.

Desarrollar un módulo de gestión de tareas, permitiendo la asignación y el seguimiento de tareas relacionadas con los eventos.

Asegurar la comunicación entre los componentes del sistema a través de un API Gateway que gestione las solicitudes a los microservicios.

Garantizar la seguridad de las transacciones a través de HTTPS y prácticas de autenticación y autorización.

Proporcionar una documentación clara del sistema utilizando Swagger para facilitar la integración y el uso de la API.

Cronograma de actividades de aplicación de organización de eventos

ID	Tarea / Diagrama	Descripción	Fecha de Inicio	Fecha de Finalización	Duración	Responsable(s)	Estado
1	Planificación	Definir las fases y secuencia de creación de los diagramas	29/09/2024	10/1/2024	3 día	Tatiana Sanchez (Estudiante), Fracisco Lopez (Estudiante), Josue Siliezar (Estudiante)	Completado
2	Diagrama de Casos de Uso	Identificación de actores y descripción de los casos de uso del sistema	10/1/2024	10/2/2024	1días	Josue Siliezar (Estudiante)	Completado
3	Diagrama de Secuencia	Modelado de la interacción temporal entre objetos para cumplir funciones específicas	10/2/2024	10/3/2024	1 días	Marcelo Delgado (Estudiante)	Completado
4	Diagrama de Módulo (Componentes)	Diseño de los módulos y componentes del sistema y sus interfaces	10/3/2024	10/4/2024	1 días	Fracisco Lopez (Estudiante)	Completado
5	Diagrama de Modelo Físico de Datos	Definición de la estructura física de la base de datos (tablas, relaciones, índices)	10/7/2024	10/8/2024	1 días	Tatiana Sanchez (Estudiante)	Completado
6	Diagrama Entidad-Relación (ER)	Modelado de entidades y relaciones a nivel lógico	10/8/2024	10/9/2024	1 días	Fracisco Lopez (Estudiante)	Completado
7	Revisión Final	Revisión de todos los diagramas para asegurar consistencia y cumplimiento de requisitos	10/9/2024	10/11/2024	2 día	Tatiana Sanchez (Estudiante), Fracisco Lopez (Estudiante), Josue Siliezar (Estudiante)	Completado
8	Presentación Final	Presentación de todos los diagramas UML al equipo de stakeholders para su aprobación	10/11/2024	13/10/2024	2 día	Tatiana Sanchez (Estudiante), Fracisco Lopez (Estudiante), Josue Siliezar (Estudiante)	Completado

Restricciones Arquitectónicas

El desarrollo del sistema debe ajustarse a los siguientes requerimientos y restricciones:

- La arquitectura debe estar basada en microservicios, lo que implica una separación clara entre los servicios de eventos, tareas, usuarios y notificaciones.
- La base de datos será de tipo relacional (SQL) y cada microservicio tendrá acceso exclusivo a su propio almacenamiento para mantener la independencia.
- La seguridad es prioritaria, por lo que las comunicaciones entre servicios deben estar cifradas utilizando HTTPS.
- El API Gateway manejará todas las solicitudes de los usuarios hacia los microservicios correspondientes.

Plataforma Tecnológica

Para el desarrollo de la aplicación se utilizaron las siguientes tecnologías:

- Lenguaje de programación: ASP.NET Core 6.0 para el backend, Blazor para el frontend.
- Base de datos: SQL Server para la persistencia de datos.
- Seguridad: Implementación de HTTPS para cifrar las comunicaciones y autenticación por tokens JWT.
- API Gateway: Para gestionar las solicitudes entre los usuarios y los microservicios.
- Swagger: Para documentar la API y facilitar la integración.

Descripción Completa del Modelo

El sistema está diseñado con base en los principios de la programación orientada a objetos y utiliza diagramas UML para representar las interacciones entre los diferentes componentes. A continuación, se presentan los diagramas de caso de uso y de secuencia.

Diagrama de Casos de Uso:

Este diagrama muestra las principales funcionalidades que el sistema proporciona a los usuarios (tanto usuarios comunes como administradores). Las interacciones clave incluyen la creación, modificación y eliminación de eventos, la asignación de tareas, y el envío de notificaciones. Los actores principales son el Usuario y el Administrador, quienes tienen diferentes niveles de acceso y funcionalidades según su rol.

- Usuario: Puede crear eventos, visualizar eventos y asignar tareas. También tiene la capacidad de marcar tareas como completadas.
- Administrador: Además de todas las funcionalidades del usuario, puede modificar y eliminar eventos, así como enviar notificaciones.

El siguiente diagrama de casos de uso refleja estas interacciones:

- Crear, modificar, eliminar y visualizar eventos.
- Asignar tareas y marcarlas como completadas.
- Enviar notificaciones sobre cambios en eventos o tareas.

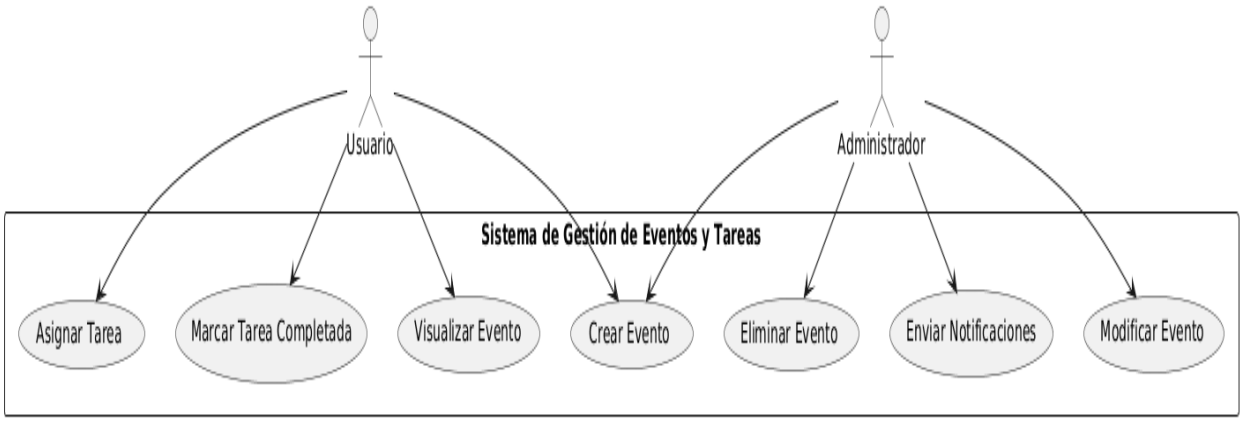
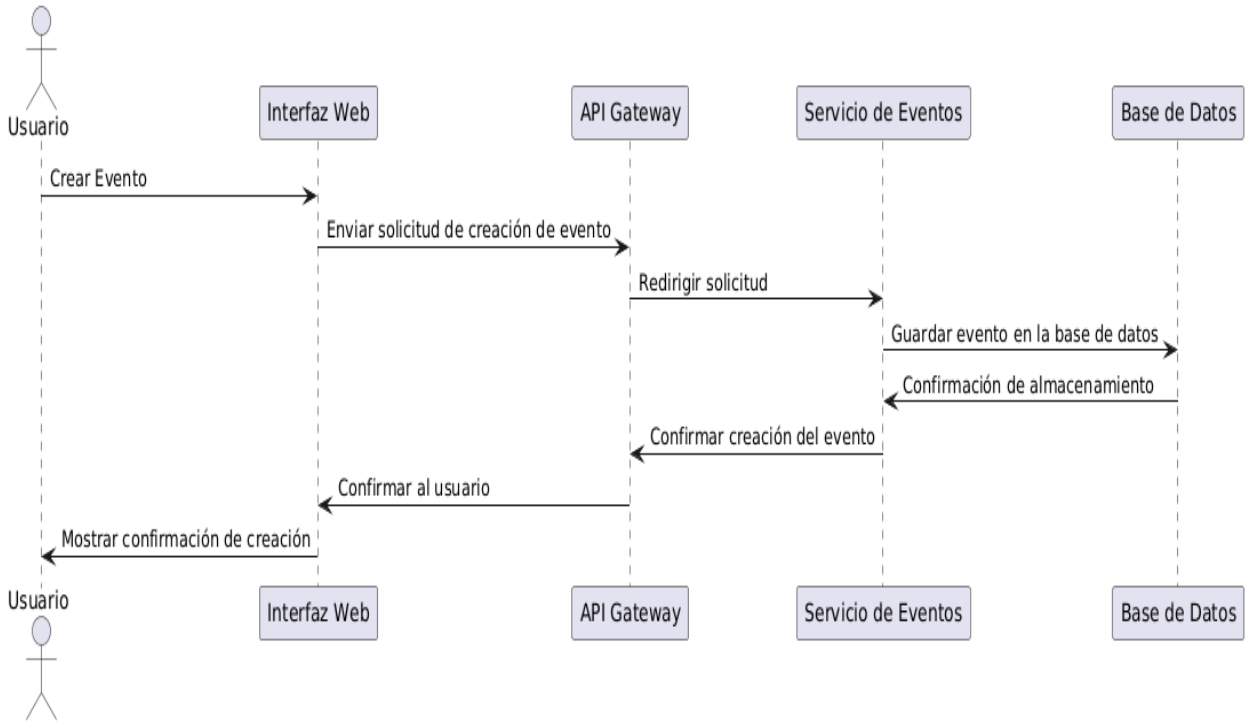


DIAGRAMA DE SECUENCIA:

El diagrama de secuencia ilustra el flujo de interacción entre el usuario, la interfaz web, el API Gateway, los microservicios y la base de datos. En este ejemplo, se detalla el proceso de creación de un evento por parte del usuario. El flujo incluye los siguientes pasos:

1. El Usuario realiza una solicitud para crear un evento a través de la Interfaz Web.
2. La Interfaz Web envía la solicitud al API Gateway, que se encarga de redirigir la solicitud al microservicio correspondiente.
3. El Servicio de Eventos recibe la solicitud, valida los datos y envía la información a la Base de Datos para almacenar el nuevo evento.
4. Una vez almacenado el evento, el Servicio de Eventos envía una confirmación de éxito al API Gateway, que a su vez informa a la Interfaz Web.
5. La Interfaz Web muestra una confirmación al Usuario.



Vista de Despliegue

La aplicación se desplegará en un servidor web configurado para soportar ASP.NET Core. La base de datos estará alojada en un servidor SQL dedicado. Cada microservicio se ejecutará de manera independiente en contenedores Docker, lo que facilita la escalabilidad del sistema. La topología del despliegue incluye:

1. Servidor de aplicación: Servirá la aplicación web desarrollada en ASP.NET Core.
2. Servidor de base de datos: SQL Server para la persistencia de los datos.
3. API Gateway: Responsable de dirigir las solicitudes a los microservicios correspondientes.
4. Servicios de microservicios: Cada servicio se ejecuta de manera aislada en contenedores.

Vista de Implementación

La implementación del sistema está dividida en varios componentes independientes:

1. Microservicio de eventos: Gestiona el CRUD de eventos.
2. Microservicio de tareas: Gestiona la creación y seguimiento de tareas relacionadas con los eventos.
3. Microservicio de usuarios: Gestiona la autenticación y autorización de los usuarios.
4. Microservicio de notificaciones: Envía notificaciones a los usuarios sobre cambios en los eventos y tareas. Cada uno de estos componentes interactúa a través del API Gateway.

Vista Logica

La vista lógica de la arquitectura del sistema se organiza en tres niveles, cada uno representando un refinamiento del anterior. Este enfoque asegura una clara separación de responsabilidades y una mayor modularidad del sistema.

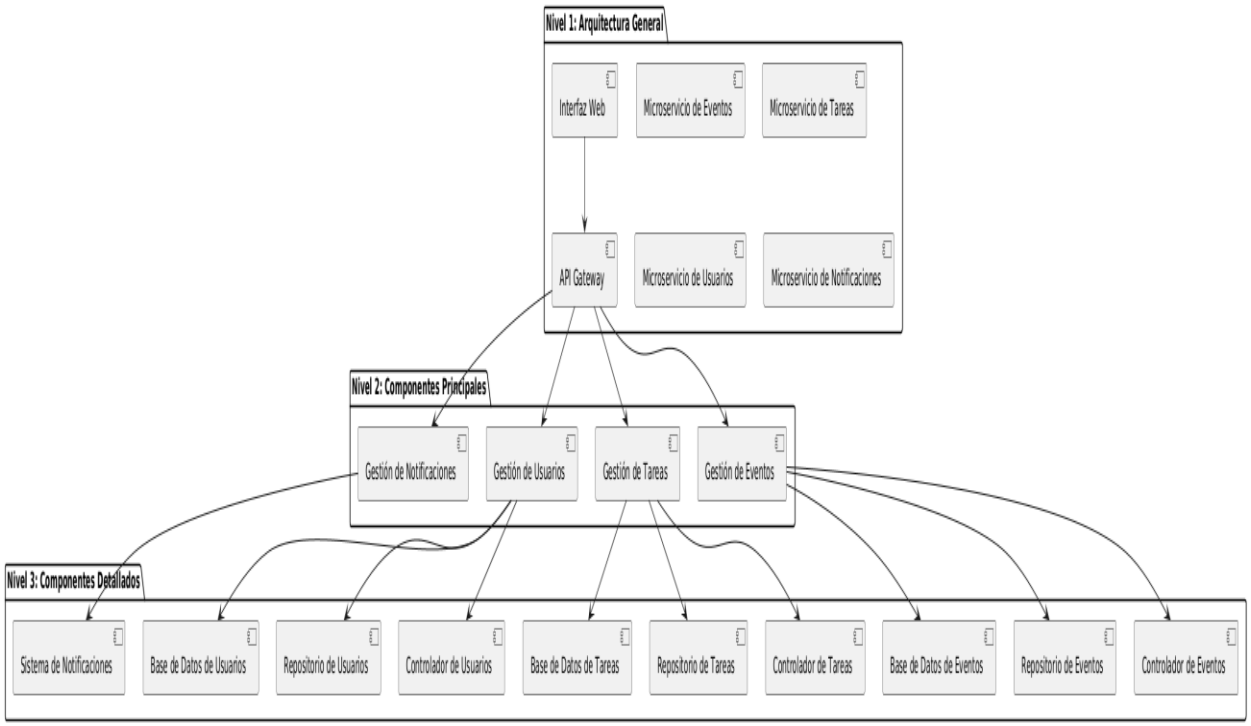
Nivel 1 (Arquitectura General): En este nivel se encuentran los componentes de alto nivel que representan las partes principales del sistema. Los elementos clave incluyen la Interfaz Web, que permite a los usuarios interactuar con el sistema, y el API Gateway, que actúa como intermediario para las solicitudes enviadas a los diferentes microservicios. Estos microservicios están organizados según las funcionalidades principales: Microservicio de Eventos, Microservicio de Tareas, Microservicio de Usuarios y Microservicio de Notificaciones.

Nivel 2 (Componentes Principales): Cada microservicio del nivel anterior se descompone en sus componentes funcionales. Por ejemplo, el Servicio de Gestión de Eventos maneja las operaciones de creación, lectura, actualización y eliminación de eventos. De manera similar, los demás servicios gestionan las operaciones específicas de sus áreas (tareas, usuarios y notificaciones).

Nivel 3 (Componentes Detallados): En este nivel se presentan los detalles internos de cada microservicio. Cada servicio está compuesto por un Controlador, que se encarga de recibir y gestionar las solicitudes; un Repositorio, que gestiona las interacciones con la base de datos; y una Base de Datos específica para cada servicio, que asegura el almacenamiento adecuado de la información y mantiene la independencia de cada microservicio. Para el servicio de notificaciones, existe un Sistema de Notificaciones encargado de gestionar el envío de alertas a los usuarios.

Este diseño asegura que cada parte del sistema funcione de manera independiente, pero interconectada a través del API Gateway, facilitando la escalabilidad y el mantenimiento de la aplicación.

DIAGRAMA DE MODULOS PARTICIPANTES



(zoom en la imagen para poder visualizar el diagrama)

Vista de Procesos

En la arquitectura de la aplicación, los procesos distribuidos juegan un papel crucial para el correcto funcionamiento del sistema. Cada microservicio se ejecuta en su propio contenedor, lo que permite que los procesos relacionados con los eventos, tareas, usuarios y notificaciones puedan ejecutarse de manera simultánea y distribuida. Los principales procesos activos en el sistema son:

1. Proceso de Gestión de Eventos: Este proceso está a cargo del microservicio de eventos. Permite a los usuarios crear, modificar, eliminar y consultar eventos. Este proceso está activo cada vez que un usuario interactúa con la funcionalidad de eventos.

2. Proceso de Gestión de Tareas: El microservicio de tareas maneja este proceso, que permite la creación, asignación y seguimiento de tareas vinculadas a eventos. Este proceso funciona simultáneamente con el de eventos para garantizar que las tareas asociadas a un evento estén siempre disponibles.

3. Proceso de Autenticación y Gestión de Usuarios: Este proceso lo ejecuta el microservicio de usuarios. Verifica las credenciales de los usuarios y gestiona los roles y permisos de acceso. Es esencial para asegurar que solo usuarios autenticados y autorizados puedan interactuar con el sistema.

4. Proceso de Notificaciones: El microservicio de notificaciones está en ejecución cada vez que se realiza un cambio en los eventos o tareas. Este proceso envía notificaciones automáticas a los usuarios para informarles sobre actualizaciones relevantes.

Los procesos están distribuidos entre varios servidores y contenedores, lo que facilita su ejecución paralela y garantiza la escalabilidad y resiliencia del sistema. Cada proceso es independiente y puede escalarse de manera horizontal (añadiendo más instancias del mismo proceso) según la demanda.

Diagramas del Modelo Físico de Datos

El modelo físico de datos describe cómo se almacenan los datos en la base de datos relacional utilizada por la aplicación. A continuación, se presenta un diagrama Entidad-Relación que representa las principales entidades del sistema y sus relaciones:

Entidad "Usuario": Representa a los usuarios del sistema, tanto los administradores como los usuarios comunes. Los campos clave incluyen:

- UsuarioID: Identificador único.
- Nombre: Nombre del usuario.
- Correo: Correo electrónico para autenticación.
- Rol: Rol del usuario (administrador, usuario estándar).

Entidad "Evento": Contiene los detalles de los eventos creados en la aplicación. Los campos clave incluyen:

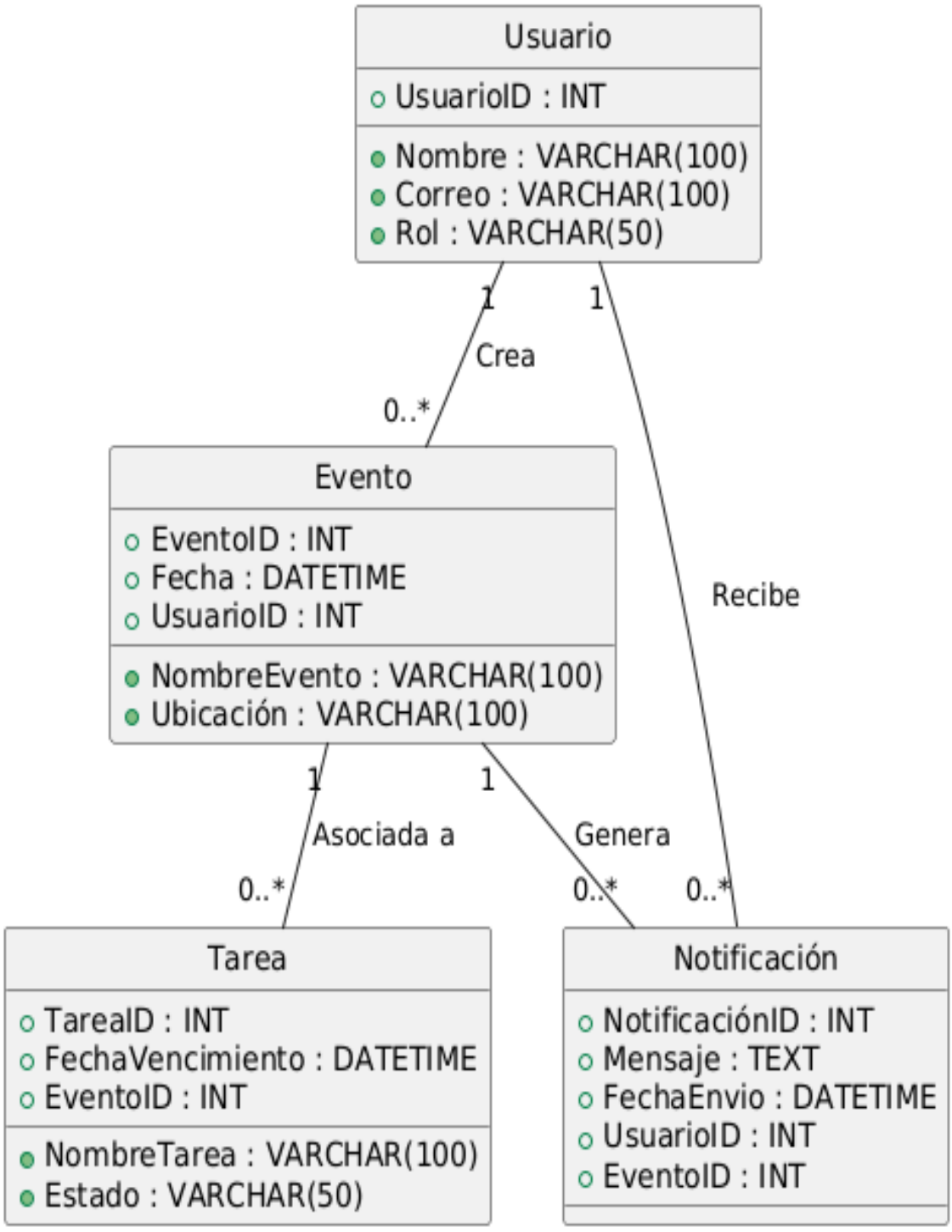
- EventoID: Identificador único del evento.
- NombreEvento: Nombre del evento.
- Fecha: Fecha del evento.
- Ubicación: Lugar donde se llevará a cabo el evento.
- UsuarioID: Llave foránea que conecta con la entidad Usuario.

Entidad "Tarea": Se encarga de gestionar las tareas relacionadas con los eventos. Los campos clave incluyen:

- TareaID: Identificador único de la tarea.
- NombreTarea: Descripción de la tarea.
- Estado: Estado de la tarea (pendiente, en progreso, completada).
- FechaVencimiento: Fecha límite de la tarea.
- EventoID: Llave foránea que conecta con la entidad Evento.

Entidad "Notificación": Registra las notificaciones enviadas a los usuarios. Los campos clave incluyen:

- NotificaciónID: Identificador único de la notificación.
- Mensaje: Mensaje de la notificación.
- FechaEnvio: Fecha y hora del envío.
- UsuarioID: Llave foránea que conecta con la entidad Usuario.
- EventoID: Llave foránea que conecta con la entidad Evento (si aplica).


Usuario
UsuarioID : INT
Nombre : VARCHAR(100)
Correo : VARCHAR(100)
Rol : VARCHAR(50)
1
Crea
0..*
1
Recibe
Evento
EventoID : INT
Fecha : DATETIME
UsuarioID : INT
NombreEvento : VARCHAR(100)
Ubicación : VARCHAR(100)
1
Asociada a
0..*
1
Genera
0..*
0..*
Tarea
TareaID : INT
FechaVencimiento : DATETIME
EventoID : INT
NombreTarea : VARCHAR(100)
Estado : VARCHAR(50)
Notificación
NotificaciónID : INT
Mensaje : TEXT
FechaEnvio : DATETIME
UsuarioID : INT
EventoID : INT

Además del diagrama Entidad-Relación, se presenta un diccionario de datos donde se listan los campos de cada tabla de la base de datos:

Tabla	Campo	Tipo de Dato	Descripción
Usuario	UsuarioID	INT	Identificador único del usuario
Usuario	Nombre	VARCHAR(100)	Nombre completo del usuario
Usuario	Correo	VARCHAR(100)	Correo electrónico del usuario
Usuario	Rol	VARCHAR(50)	Rol del usuario (admin, estándar)
Evento	EventoID	INT	Identificador único del evento
Evento	NombreEvento	VARCHAR(100)	Nombre del evento
Evento	Fecha	DATETIME	Fecha y hora del evento
Evento	Ubicación	VARCHAR(100)	Ubicación del evento
Evento	UsuarioID	INT	Relación con la tabla Usuario
Tarea	TareaID	INT	Identificador único de la tarea
Tarea	NombreTarea	VARCHAR(100)	Descripción de la tarea
Tarea	Estado	VARCHAR(50)	Estado de la tarea
Tarea	FechaVencimiento	DATETIME	Fecha límite de la tarea
Tarea	EventoID	INT	Relación con la tabla Evento
Notificación	NotificaciónID	INT	Identificador único de la notificación
Notificación	Mensaje	TEXT	Mensaje enviado en la notificación
Notificación	FechaEnvio	DATETIME	Fecha y hora de envío
Notificación	UsuarioID	INT	Relación con la tabla Usuario
Notificación	EventoID	INT	Relación con la tabla Evento

Presupuesto de aplicación ASP.NET

Descripción	Horas estimadas	Tarifa por Hora (USD)	Costo Total (USD)	Descripción	Horas estimadas	Tarifa por Hora (USD)	Costo Total (USD)
1. Fase de Planificación y Análisis				**3. Fase de Pruebas y Validación**			
Reuniones iniciales, análisis de requisitos	15	600 / 160 h = $3.75/h	$56.25	Pruebas de seguridad (HTTPS, autenticación)	10	$3.75/h	$37.50
Diseño arquitectónico del API Gateway	10	$3.75/h	$37.50	Pruebas funcionales de las APIs	10	$3.75/h	$37.50
Diseño de seguridad y autenticación/autorización	10	$3.75/h	$37.50	Pruebas de integración (Gateway, APIs)	12	$3.75/h	$45.00
Subtotal Fase de Planificación	**35 horas**		**$131.25**	Revisión de documentación Swagger	5	$3.75/h	$18.75
				Subtotal Fase de Pruebas	**37 horas**		**$138.75**
2. Fase de Desarrollo				**4. Fase de Despliegue e Infraestructura**			
Configuración del proyecto ASP.NET Core	10	$3.75/h	$37.50				
Desarrollo del API Gateway	20	$3.75/h	$75.00	Configuración del servidor para HTTPS	5	$3.75/h	$18.75
Implementación de HTTPS	8	$3.75/h	$30.00	Despliegue de la aplicación en ambiente de producción	8	$3.75/h	$30.00
Implementación de autenticación (JWT/OAuth)	12	$3.75/h	$45.00	Documentación final y manuales de usuario	5	$3.75/h	$18.75
Implementación de autorización (Roles/Claims)	10	$3.75/h	$37.50	**Subtotal Fase de Despliegue**	**18 horas**		**$67.50**
Integración de Swagger para la documentación de API	8	$3.75/h	$30.00				
Subtotal Fase de Desarrollo	**68 horas**		**$255.00**	**Total General (Proyecto Completo)**	**158 horas**		**$592.50**

Notas Adicionales sobre el Salario de Colaboradores y Cálculos Estimados

1. **Salario de los Programadores:**
    - El salario mensual de los programadores se ha fijado en $600.00. Se ha asumido una jornada laboral estándar de 160 horas al mes, lo que da como resultado una tarifa por hora de:

      Salario por hora = 600.00 / 160 = 3.75 USD / Hora

    - Esta tarifa se ha utilizado para calcular los costos de las distintas tareas del proyecto en función de la cantidad de horas estimadas que cada programador invertirá en cada fase del desarrollo.
2. **Cálculos Estimados del Presupuesto:**
    - Para cada tarea o fase del proyecto, se ha estimado el número de horas requeridas. Estas estimaciones se basan en la experiencia promedio de proyectos similares que involucran ASP.NET, configuración de seguridad con HTTPS, integración de API Gateway, y la implementación de autenticación/autorización.
    - Los costos totales de cada fase se han calculado multiplicando las horas estimadas por la tarifa por hora de $3.75**.**

Ejemplo de cálculo:

- Si una tarea tiene una duración estimada de 10 horas, el costo se calcula como:

  Costo Total de la Tarea = 10 horas * 3.75 USD /hora = 37.50 USD

3. **Fases del Proyecto:**
    - El presupuesto se ha dividido en cuatro fases principales: planificación y análisis, desarrollo, pruebas y validación, y despliegue e infraestructura. Cada fase tiene tareas detalladas con horas asignadas y su costo correspondiente.
4. **Costos Indirectos:**
    - No se han incluido en este presupuesto costos adicionales como licencias, certificaciones HTTPS de pago, o infraestructura en la nube. Estos costos dependerán de las herramientas y servicios específicos que se decidan utilizar.

Otros Posibles Costos Adicionales

Además de los costos de desarrollo descritos, es importante considerar los siguientes posibles costos adicionales que podrían influir en el presupuesto final del proyecto:

1. **Certificado SSL para HTTPS:**
    - Certificados Gratuitos: Existen soluciones gratuitas como Let's Encrypt que pueden ser usadas para implementar HTTPS sin costo adicional. Sin embargo, estas opciones pueden requerir renovaciones periódicas y configuraciones manuales.
    - Certificados de Pago: Dependiendo de los requerimientos de seguridad o del tipo de cliente, puede ser necesario optar por un certificado SSL de pago que ofrezca características adicionales como validación extendida o mayor nivel de soporte. Los precios de estos certificados varían, pudiendo oscilar entre $50 y $300 USD al año.
2. **Infraestructura en la Nube o Servidores:**
    - Servidor en la Nube: Si se despliega la aplicación en una infraestructura de nube (por ejemplo, Azure, AWS, Google Cloud), se deberán considerar los costos de servicios como:
        - Máquinas virtuales (para ejecutar la aplicación),
        - Almacenamiento (para bases de datos),
        - Balanceadores de carga y otros servicios de red.
        - Los costos dependen del tamaño y uso de los recursos, pero un entorno promedio para una aplicación de este tipo podría costar entre $100 y $500 USD al mes.
    - Servidor Dedicado o Local: Si se opta por desplegar en un servidor físico dedicado o una infraestructura local, se deberán considerar los costos de hardware, mantenimiento, energía, y soporte técnico. Estos costos pueden ser significativos dependiendo de la escala del proyecto.
3. **Licencias de Software:**
    - Aunque ASP.NET Core es de código abierto y gratuito, otros componentes del entorno de desarrollo podrían requerir licencias. Algunas herramientas y software de terceros que podrían utilizarse incluyen:
        - Herramientas de monitoreo y análisis para el seguimiento del rendimiento de la aplicación.

        - Herramientas de seguridad para auditorías o pruebas de penetración (por ejemplo, Burp Suite, OWASP ZAP).
        - Sistemas de integración continua (CI) y automatización de despliegues que pueden tener costos asociados.
        - Dependiendo de las herramientas seleccionadas, los costos pueden variar entre $50 y $200 USD al mes.

4. **Pruebas de Seguridad y Auditorías:**
    - Para garantizar que el sistema cumpla con los estándares de seguridad, podría ser necesario realizar pruebas de seguridad avanzadas, como auditorías de seguridad y pruebas de penetración. Estas pruebas pueden ser realizadas por empresas externas especializadas o mediante la contratación de servicios profesionales. El costo de estas pruebas varía ampliamente dependiendo de la profundidad del análisis, pero puede oscilar entre $500 y $5,000 USD por auditoría.
5. **Mantenimiento y Actualizaciones:**
    - Después del lanzamiento de la aplicación, se debe considerar un presupuesto para mantenimiento continuo, que podría incluir actualizaciones de seguridad, nuevas características, soporte técnico, y corrección de errores. Este costo dependerá de la frecuencia y cantidad de cambios que se realicen, pero una estimación básica podría ser de $100 a $300 USD al mes.

Resumen de Posibles Costos Adicionales:

- Certificado SSL (HTTPS): $0 a $300 USD al año.
- Infraestructura en la Nube/Servidor: $100 a $500 USD al mes.
- Licencias de Software: $50 a $200 USD al mes.
- Pruebas de Seguridad/Auditorías: $500 a $5,000 USD (una vez o de forma periódica).
- Mantenimiento y Actualizaciones: $100 a $300 USD al mes.