**Diseño, desarrollo y programación de una solución tecnológica para la gestión del proceso de inscripción, participación y envío de artículos científicos a congresos académicos**

**Trabajo escrito**

**Jahir de Jesús Castillo Ochoa**

10 de diciembre de 2024

“Lis de Veracruz: Arte, Ciencia, Luz”

Índice

[Introducción 3](#_Toc152261351)

[Justificación 4](#_Toc152261352)

[Objetivos y metas 5](#_Toc152261353)

[Análisis, diseño y propuesta de entorno de programación 6](#_Toc152261354)

[Metodología de desarrollo 6](#_Toc152261355)

[Diseño 8](#_Toc152261356)

[Arquitectura MVC 8](#_Toc152261357)

[Propuesta de entorno de programación 11](#_Toc152261358)

[Tecnologías 11](#_Toc152261359)

[Cronograma 12](#_Toc152261360)

[Bibliografía 13](#_Toc152261361)

Índice de figuras

[Figura 1 El ciclo de vida en AUP 7](#_Toc152261362)

[Figura 2 Tipos de liberaciones en AUP 8](#_Toc152261363)

[Figura 3 Arquitectura del sistema web 9](#_Toc152261364)

[Figura 4 Casos de uso 10](#_Toc152261365)

# Introducción

En la era digital, la organización de eventos como congresos, conferencias y simposios ha evolucionado significativamente gracias al desarrollo de sistemas informáticos especializados. Estos sistemas permiten la automatización de procesos complejos, como el registro de participantes, la gestión de ponencias y la emisión de certificados, optimizando el tiempo y los recursos de los organizadores (Chen et al., 2019). Según Laudon y Laudon (2020), los sistemas de información juegan un papel crítico en la integración de datos y procesos, facilitando la toma de decisiones y mejorando la eficiencia operativa.

El diseño y desarrollo de un sistema de gestión de congresos requiere un enfoque interdisciplinario que combine principios de ingeniería de software, experiencia de usuario y tecnologías emergentes. Como señala Pressman (2021), un diseño centrado en el usuario es fundamental para garantizar la adopción y satisfacción de los usuarios finales. Además, la programación de estos sistemas debe incluir características como la seguridad de los datos y la escalabilidad para manejar diferentes volúmenes de usuarios y eventos (Somerville, 2019).

En este contexto, el presente documento describe el proceso de diseño, desarrollo y programación de un sistema de gestión de congresos, abordando los desafíos técnicos y metodológicos involucrados, y proponiendo soluciones basadas en buenas prácticas de la industria.

# Justificación

La creciente demanda de herramientas digitales para la organización de eventos académicos y profesionales ha impulsado la necesidad de sistemas especializados que gestionen de manera eficiente los procesos asociados a congresos. Según Laudon y Laudon (2020), la implementación de sistemas de información en organizaciones permite mejorar la coordinación, reducir costos y optimizar recursos, elementos clave para garantizar el éxito de eventos de esta naturaleza.

Actualmente, muchas instituciones enfrentan desafíos en la gestión manual de congresos, como la duplicidad de datos, errores en el registro de participantes y la falta de herramientas para analizar el impacto del evento (Pressman, 2021). La automatización de estos procesos a través de un sistema de gestión integral no solo facilita las operaciones, sino que también mejora la experiencia de los usuarios al ofrecer funciones como registro en línea, generación automática de certificados y comunicación efectiva entre organizadores y asistentes (Somerville, 2019).

Por ello, el desarrollo de este sistema se justifica en la necesidad de proporcionar una solución tecnológica que incremente la eficiencia, reduzca los errores operativos y fomente la innovación en la organización de eventos académicos. Además, esta propuesta responde a las tendencias actuales de digitalización en el ámbito educativo y profesional, que buscan transformar los procesos tradicionales mediante el uso de tecnologías avanzadas.

# Alcance del producto

El sistema de gestión de congresos está diseñado para facilitar y automatizar los procesos clave en la organización y desarrollo de eventos académicos.

## Funcionalidades principales

**Módulo de gestión de convocatorias**

* Permite a los organizadores crear, publicar y administrar convocatorias para congresos. Este módulo incluye la definición de fechas importantes, requisitos para los participantes y la publicación de información relevante en un portal accesible para los interesados.

**Módulo de registro de ponentes y ponencias**

* Registro de ponentes: Los usuarios pueden registrarse como ponentes, proporcionando sus datos personales de manera estructurada.
* Registro de ponencias: Los ponentes pueden cargar sus trabajos en formatos específicos, asociándolos con áreas temáticas definidas por los organizadores.

**Módulo de gestión de usuarios (revisores)**

* Proporciona herramientas para registrar y gestionar revisores, asignándolos a áreas temáticas específicas.

**Módulo de revisión de ponencias**

* Los revisores asignados pueden evaluar y calificar las ponencias siguiendo criterios predefinidos por los organizadores. Este módulo permite:
* Asignación automática o manual de ponencias a revisores.
* Registro de observaciones y resultados de las revisiones.
* Retroalimentación directa a los ponentes sobre el estado de sus trabajos (aceptado, rechazado o en revisión).

**Módulo de itinerarios**

Facilita la planificación y publicación de las actividades del congreso, incluyendo:

* Creación de horarios detallados para presentaciones y eventos.

**Alcance general**

El sistema está dirigido a instituciones académicas y organizaciones encargadas de eventos académicos y profesionales, brindándoles una solución integral para la gestión de sus procesos operativos. Este producto no solo reduce la carga administrativa, sino que también mejora la experiencia de los usuarios mediante una plataforma intuitiva y funcional.

El alcance abarca desde la conceptualización y validación de las ponencias hasta la organización del congreso, integrando todos los módulos desarrollados en un flujo de trabajo lógico y eficiente. En el futuro, el sistema podrá escalarse para incluir módulos adicionales, como gestión de asistentes y pagos.

# Objetivos y metas

## Objetivo general

Diseñar, desarrollar e implementar un sistema de gestión de congresos que facilite y automatice los procesos relacionados con la gestión de convocatorias, registro y validación de ponencias, asignación y revisión de trabajos, gestión de usuarios (revisores) y planificación de itinerarios.

## Metas

**Automatización de procesos clave**

Eliminar la dependencia de métodos manuales mediante la implementación de módulos que gestionen de manera eficiente convocatorias, registros y revisiones de ponencias.

**Gestión efectiva de usuarios (revisores)**

Proporcionar herramientas para registrar, organizar y asignar revisores según áreas temáticas, asegurando una revisión más ordenada y rápida de los trabajos.

**Planificación eficiente de itinerarios**

Crear un módulo de itinerarios que permita diseñar y gestionar horarios detallados para las ponencias del congreso de manera sencilla.

**Accesibilidad y usabilidad**

Asegurar que el sistema sea intuitivo y accesible para los usuarios, tanto organizadores como participantes, a través de interfaces claras y funcionales.

**Escalabilidad**

Diseñar el sistema con una arquitectura que permita futuras expansiones, como la integración de módulos adicionales (gestión de participantes, generación de reportes avanzados, etc.).

**Cumplimiento de estándares de seguridad**

Proteger los datos sensibles de usuarios y participantes, garantizando confidencialidad, integridad y disponibilidad de la información.

# Desarrollo de la metodología

## Justificación de la metodología

El Proceso Unificado Ágil (AUP por sus siglas en inglés), es un trabajo realizado por Scott W. Amblera mediados de 1999, y publicado posteriormente en su libro *Agile Modeling*(Ambler, 2002). Este trabajo presenta una versión simplificada del Proceso Unificado de Rational (RUP por sus siglas en inglés), en esta versión se introducen algunas prácticas de las metodologías ágiles durante el ciclo de vida (Beck, 2003), el modelado y diseño ágil, y la refactorización constante.

AUP sigue los principios del manifiesto ágil, promueve la comunicación entre los miembros del equipo de trabajo, la documentación sencilla y concisa, el uso de las herramientas que mejor funcionen para el equipo de acuerdo a sus habilidades y conocimientos, y la atención a las actividades reales e importantes del proyecto (Ambysoft Inc., 2005).

La Figura 1 muestra el ciclo de vida del desarrollo de una aplicación con AUP. En esta figura puede apreciarse como el proceso se desarrolla en dos dimensiones. La primera de estas dimensiones (eje horizontal) representa la línea temporal, y está dividida en 4 fases consecutivas: **Inicio**, donde se determina el alcance del proyecto, requerimientos iniciales, se evalúan posibles riesgos, y se determina una posible arquitectura; **Diseño Técnico o** **Elaboración**, donde se prueba si la arquitectura definida en la etapa de inicio funcionará bien para la aplicación; **Construcción o Codificación**, donde se construye propiamente el software en una forma incremental; y la fase de **Transición**, donde se valida que el software construido cumpla los requerimientos y se libera una versión de la aplicación (Li & Wang, 2010).

Gráfico

Descripción generada automáticamente

Figura 1 El ciclo de vida en AUP

La segunda dimensión (eje vertical) señala una serie de disciplinas que se ejecutan iterativamente en cada una de las fases, estas disciplinas definen una serie de actividades que habrán de realizarse para construir, validar y finalmente liberar software funcional que cumpla con los requerimientos. Estas disciplinas incluyen:

**Modelado**, que trata acerca de entender las reglas del negocio y el dominio del problema, así como identificar soluciones factibles para éste.

**Implementación**, que se trata de transformar los modelos en código ejecutable.

**Pruebas**, que se trata de efectuar una evaluación objetiva de la calidad del producto, así como identificar defectos y validar que se cumplen los requerimientos.

**Despliegue**, que se trata de planificar la entrega final al usuario.

**Administración de la configuración**, que trata sobre la administración tanto de los artefactos, como de su disponibilidad, así como las diferentes versiones de los mismos.

**Administración del proyecto**, que trata sobre la administración de las actividades, personal y sistemas involucrados, así como prevención de riesgos.

Se distinguen dos categorías de liberaciones de productos en AUP: en primer lugar, está la liberación de desarrollo, que abarca liberaciones de menor escala que podrían eventualmente convertirse en liberaciones de producción, pero que aún no han superado las pruebas requeridas. En segundo lugar, se encuentra la liberación de producción, que, como su nombre sugiere, representa liberaciones que han sido implementadas en producción y constituyen una versión oficial del producto. Estas liberaciones se llevan a cabo de manera iterativa a lo largo de la línea de tiempo del desarrollo, según se ilustra en el contexto de la metodología (Figueroa, 2023).

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Figura 2 Tipos de liberaciones en AUP

## Artefactos resultantes por cada etapa

En la metodología ágil AUP (Agile Unified Process), los artefactos generados en cada etapa del desarrollo están alineados con los objetivos de sus fases. AUP es una versión ligera y ágil del Proceso Unificado de Desarrollo (RUP), que se centra en iteraciones cortas, documentación mínima y entregables funcionales. A continuación, se describen las fases de AUP y los principales artefactos asociados:

**1. Inicio**

**Artefactos generados:**

* Declaración de visión del proyecto: Documento que describe los objetivos, alcance y justificación del proyecto.
* Lista de requisitos iniciales.
* Modelo de casos de uso inicial.
* Plan inicial del proyecto: Incluye cronograma.
* Prototipo inicial (opcional): Representación básica de la idea para validar requisitos críticos.

**2. Elaboración**

**Artefactos generados:**

* Lista priorizada y detallada de funcionalidades.
* Modelo de dominio: Diagramas de clases iniciales que representan el modelo conceptual.
* Diseño de arquitectura: Esquema de la arquitectura general del sistema, incluyendo patrones tecnológicos.
* Casos de uso detallados.
* Prototipo funcional: Versión temprana que muestra las funcionalidades más importantes para mitigar riesgos.
* Plan de iteraciones.

**3. Construcción**

**Artefactos generados**

* Código fuente: Entregas de software funcional y ejecutable al final de cada iteración.
* Pruebas unitarias y de integración: Scripts y reportes de pruebas realizados durante cada iteración.
* Prototipos de iteración: Versiones funcionales entregables que implementan características específicas.
* Actualización del backlog: Revisión y ajuste continuo de los requisitos según la retroalimentación recibida.

**4. Transición**

**Artefactos generados:**

* Versión final del producto: Software completo, probado y documentado.
* Manuales de usuario: Documentación orientada a los usuarios finales del sistema.
* Reportes de pruebas: Registro de pruebas finales, incluyendo pruebas de aceptación.
* Documentación del despliegue: Guías para implementar el sistema en el entorno de producción.

**5. Soporte**

**Artefactos generados:**

* Reportes de incidencias: Registro de problemas detectados y su resolución.
* Mejoras: Lista de funcionalidades futuras o ajustes derivados de la retroalimentación post-despliegue.
* Documentación de mantenimiento.

## Justificación del stack tecnológico

En el desarrollo del sistema de gestión de congresos, se seleccionaron CodeIgniter 4 (CI4), Bootstrap 5, jQuery y MariaDB como las tecnologías principales. Esta combinación fue elegida por su capacidad para proporcionar un entorno de desarrollo ágil, eficiente y escalable, ajustándose a las necesidades y objetivos del proyecto. A continuación, se detalla la justificación de estas elecciones y por qué no se optó por otras alternativas.

**CodeIgniter 4 (CI4)**

Ventajas principales:

* **Ligero y eficiente:** CI4 es conocido por su bajo consumo de recursos, lo que lo hace ideal para sistemas que no requieren infraestructura compleja pero que deben ser rápidos y efectivos.
* **Fácil curva de aprendizaje**: Su diseño minimalista permite que los desarrolladores trabajen rápidamente sin requerir una inversión de tiempo significativa en aprendizaje, a diferencia de frameworks más robustos como Laravel o Symfony.
* **Soporte para MVC:** Facilita la separación de la lógica de negocio, la presentación y los datos, promoviendo un diseño limpio y mantenible.
* **Adaptación a proyectos de tamaño medio:** Este framework es ideal para sistemas como el de gestión de congresos, donde la escalabilidad es moderada y no se requiere una estructura altamente compleja.

Razones para no usar otras tecnologías:

* **Laravel:** Aunque es más completo, Laravel tiene una curva de aprendizaje más pronunciada y puede ser excesivo para proyectos pequeños o medianos, aumentando innecesariamente la complejidad y los requerimientos de recursos.
* **Node.js:** Aunque es ideal para aplicaciones en tiempo real, no era una prioridad para este sistema basado en procesos definidos y controlados, sin alta demanda de concurrencia.
* **Symfony:** Este framework es robusto, pero su configuración y estructura son más adecuadas para proyectos a gran escala, donde se necesita modularidad extrema.

**Bootstrap 5**

Ventajas principales:

* **Diseño responsivo:** Bootstrap 5 permite que las interfaces sean completamente adaptables a diferentes dispositivos (escritorio, tablet y móvil) sin esfuerzo adicional.
* **Flexbox y Grid System:** Facilita la creación de diseños flexibles y modernos con menor dependencia de CSS personalizado.
* **Comunidad amplia y recursos:** Proporciona una base sólida para resolver problemas y encontrar plantillas.

Razones para no usar otras tecnologías:

* **Tailwind CSS:** Aunque más flexible, habría requerido más tiempo de configuración y diseño detallado desde cero.
* **Material UI:** Enfocado en frameworks como React, lo que no encaja con CodeIgniter.
* **Frameworks personalizados**: Desarrollar un framework propio habría incrementado significativamente los tiempos de entrega.

**jQuery**

Ventajas principales:

* **Compatibilidad:** Ideal para manejar dinámicamente el DOM, eventos y llamadas AJAX de forma simple y funcional.
* **Rapidez en el desarrollo:** Permite implementar características dinámicas de manera eficiente, como validación de formularios, modales y carga dinámica de contenido.
* **Amplia comunidad:** Garantiza soluciones rápidas para problemas comunes.

Razones para no usar otras tecnologías:

* **JavaScript puro (Vanilla JS):** Aunque es una alternativa viable, habría incrementado el tiempo requerido para escribir código para funciones comunes.
* **Frameworks modernos (React, Angular, Vue.js):** Estos frameworks son más adecuados para aplicaciones de una sola página (SPA) y proyectos más complejos, y no justifican su implementación en un sistema de este tipo debido a su curva de aprendizaje y configuración.

**MariaDB**

Ventajas principales:

**Compatibilidad con MySQL:** MariaDB ofrece una transición fluida desde MySQL, manteniendo compatibilidad con la mayoría de los comandos y herramientas.

**Rendimiento mejorado:** Ofrece optimizaciones de rendimiento en consultas complejas, lo que lo hace ideal para sistemas que manejan datos relacionados con usuarios, ponencias y revisiones.

**Licencia abierta:** A diferencia de MySQL (controlado por Oracle), MariaDB es completamente de código abierto, lo que garantiza mayor transparencia y flexibilidad.

**Escalabilidad:** Permite manejar bases de datos de tamaño mediano y grande sin comprometer el rendimiento.

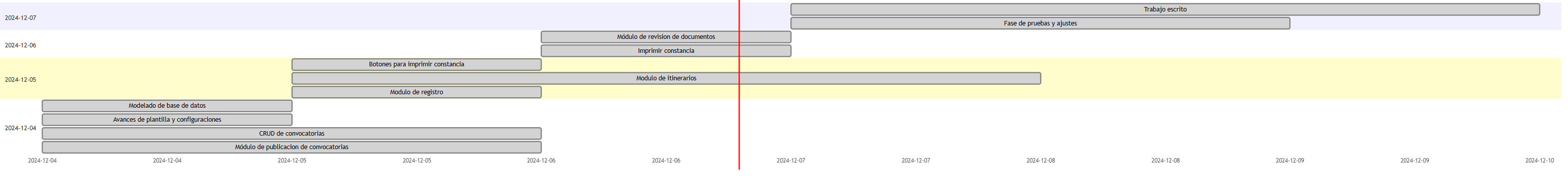
**Seguridad avanzada:** Ofrece características mejoradas de cifrado y autenticación para proteger los datos sensibles del sistema.

Razones para no usar otras tecnologías:

* **MySQL:** Aunque similar, el control comercial de Oracle puede generar restricciones futuras, mientras que MariaDB mantiene su filosofía de software libre.
* **PostgreSQL:** Si bien es potente y altamente escalable, sus capacidades avanzadas no eran necesarias para este sistema y habría incrementado la curva de aprendizaje.
* **SQLite:** Aunque es ligero, no es adecuado para aplicaciones con múltiples usuarios concurrentes ni para un entorno de producción robusto.

# Gestión del proyecto

## Planeación de tiempos



URL: <https://shorturl.at/26ocP>

## Stackeholders involucrados en el desarrollo

El desarrollo de un sistema de gestión de congresos involucra la participación de diversos stakeholders que interactúan con el sistema de manera directa, los cuales se describen a continuación:

**Ponentes:** Son los usuarios que se registran en el sistema, registran sus ponencias, y suben los documentos asociados a estas para su evaluación.

**Administrador:** Es el organizador del congreso, responsable de gestionar las convocatorias y administrar los usuarios (revisores) del sistema.

**Revisores:** Usuarios encargados de evaluar las ponencias registradas por los ponentes, proporcionando retroalimentación y validando el contenido para su inclusión en el congreso.

**Equipo de Desarrollo:** El encargado de diseñar, desarrollar y mantener el sistema, asegurando que cumpla con los requisitos funcionales y técnicos del congreso.

## Características del desarrollo (facilidad de mantenimiento)

El desarrollo del sistema se ha diseñado siguiendo el patrón MVC (Modelo-Vista-Controlador) proporcionado por CodeIgniter 4, lo que facilita la separación de responsabilidades y mejora la organización del código. Esto permite que las modificaciones en la interfaz de usuario, la lógica de negocio y la interacción con la base de datos se realicen de forma independiente, simplificando el mantenimiento y la escalabilidad del sistema. Además, se ha adoptado el estándar de codificación PSR-12, lo que asegura un estilo de código coherente y legible. Estas prácticas, junto con una arquitectura modular, garantizan que el sistema sea fácil de mantener y actualizar a lo largo del tiempo.

## Riesgos y estrategias de mitigación

**Retrasos en el desarrollo:** Para evitar retrasos, se establecerá un cronograma claro con revisiones periódicas y se utilizará la metodología ágil para adaptarse a cambios rápidamente.

**Errores en la implementación:** Se realizarán pruebas continuas y retroalimentación constante para detectar y solucionar problemas a tiempo.

**Falta de adopción por parte de los usuarios:** Se proporcionará formación y soporte adecuado, y se diseñará una interfaz intuitiva.

**Fallos en la seguridad:** Se implementarán medidas de seguridad como encriptación de datos y auditorías periódicas.

**Desempeño deficiente bajo alta carga:** Se realizarán pruebas de carga y optimización del sistema para garantizar su rendimiento en picos de tráfico.

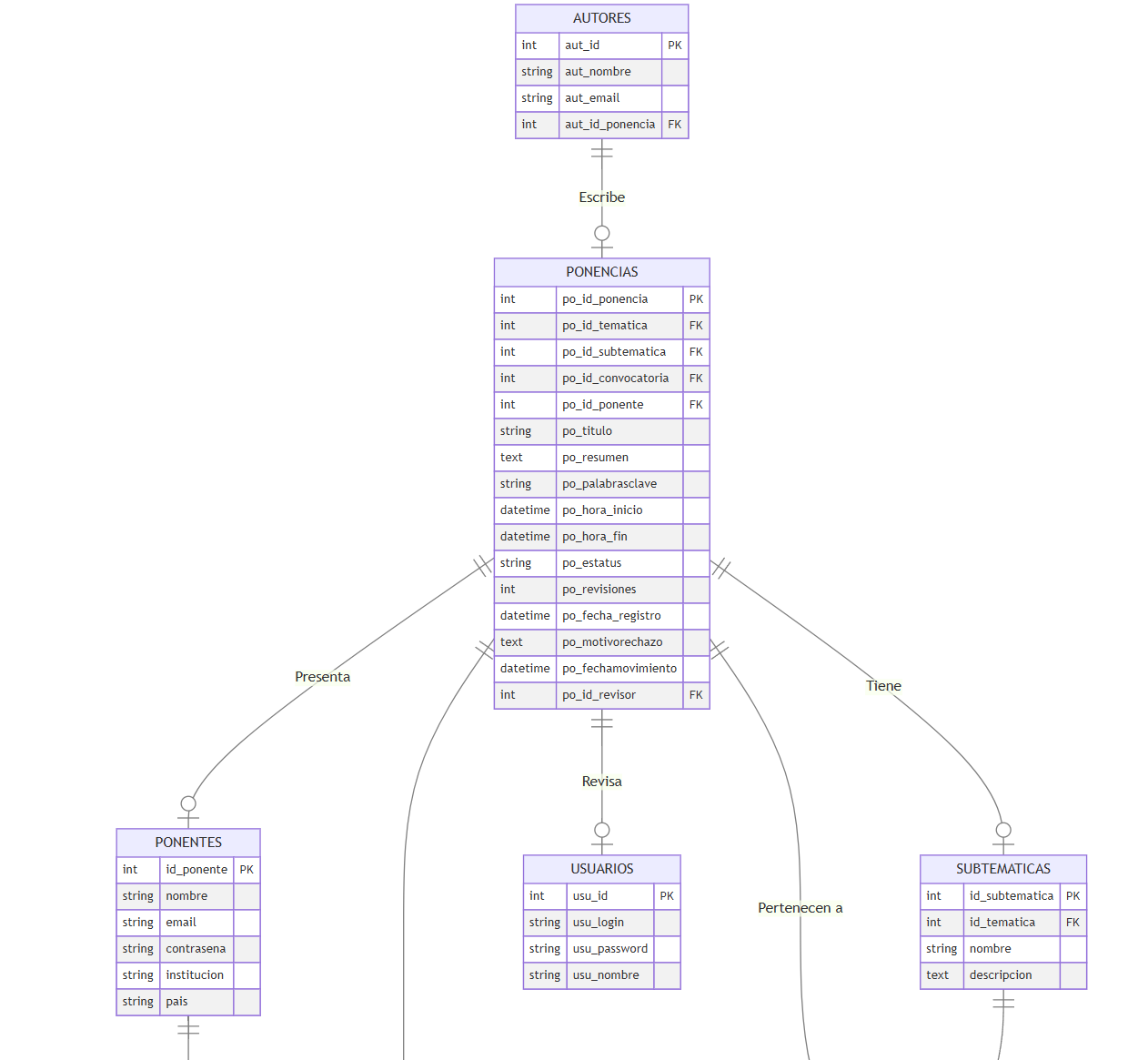
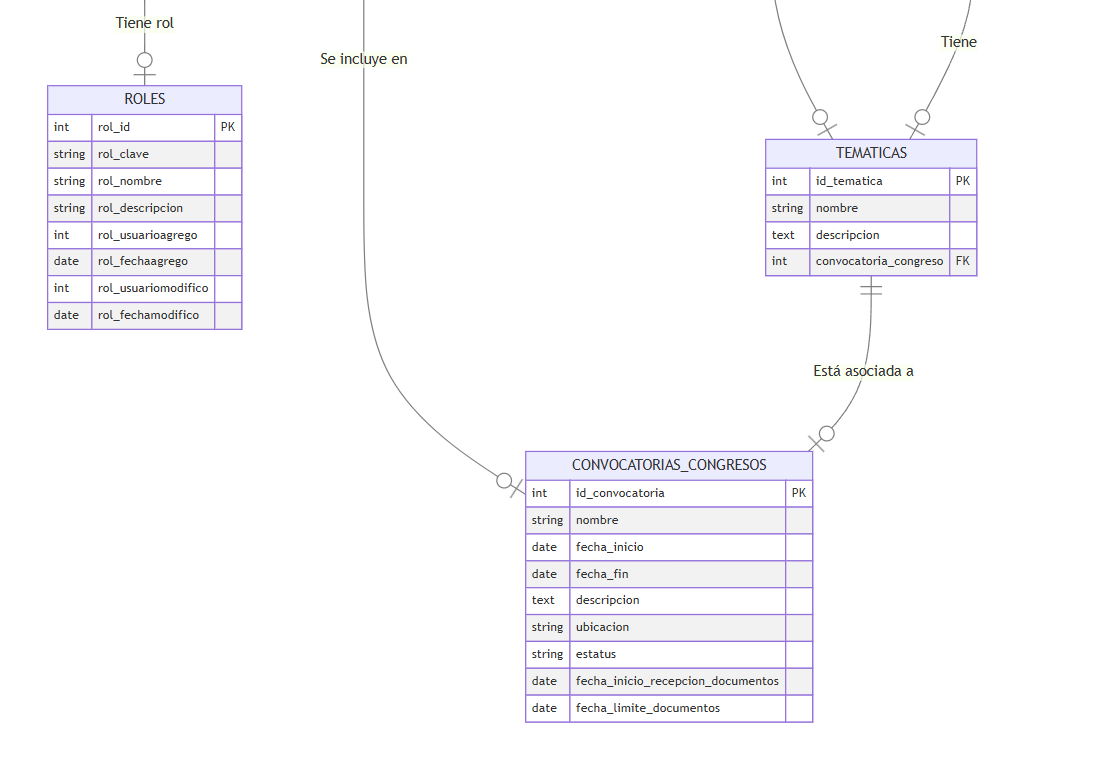
**Desalineación con los requisitos del cliente:** Se mantendrá una comunicación constante con los stakeholders para asegurar que el sistema cumpla con sus expectativas.

## Sistema de control de versiones

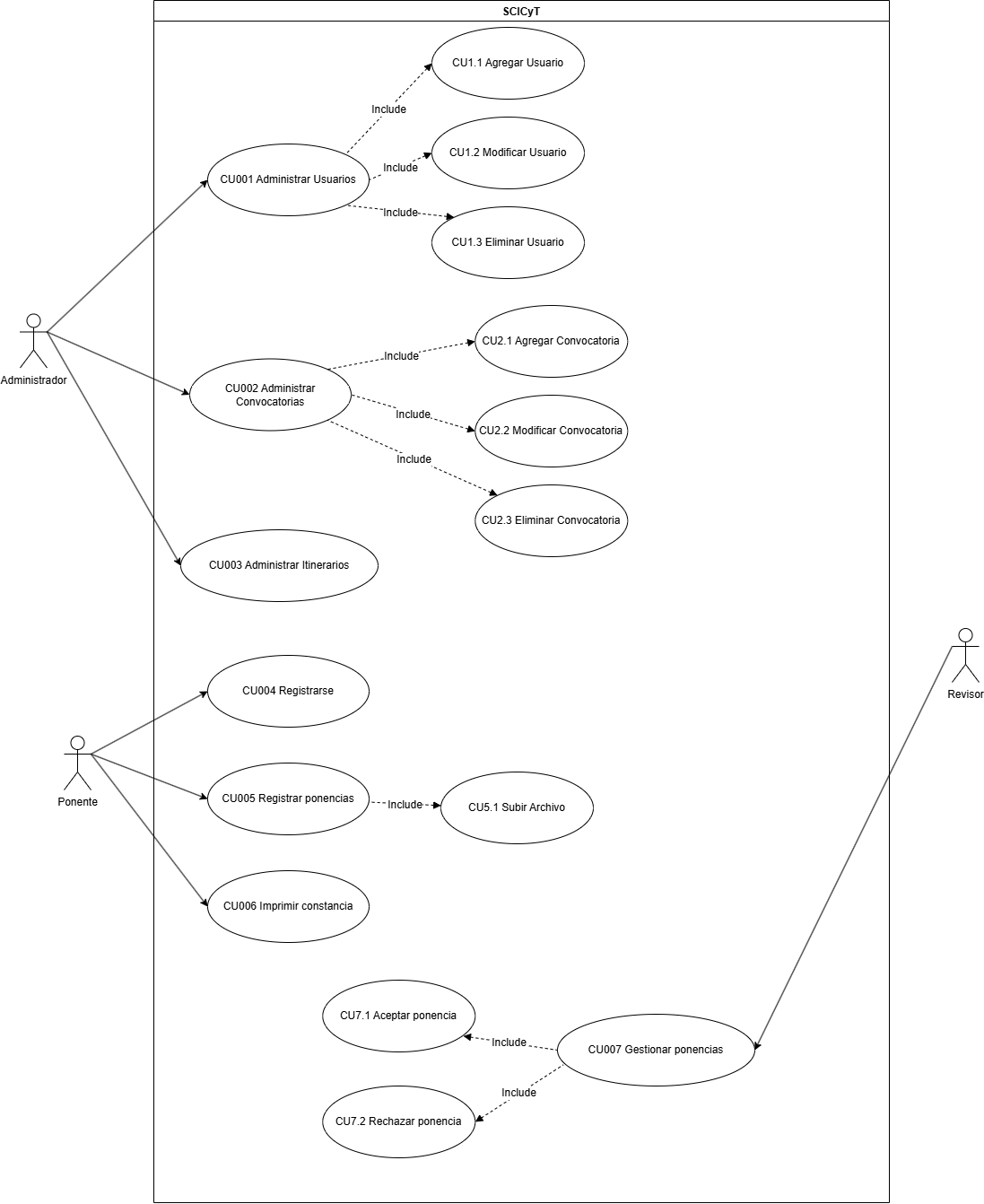
El desarrollo del sistema se gestionó utilizando Git como herramienta de control de versiones, lo que permitió llevar un registro detallado de todos los cambios realizados en el código a lo largo del proyecto. Esto facilitó la colaboración y la gestión de versiones, asegurando que cualquier modificación se pudiera rastrear y revertir si fuera necesario. El repositorio del proyecto se encuentra alojado en GitHub [<https://github.com/JahirCastillo/congreso>] , lo que proporcionó una plataforma centralizada para almacenar el código. Así mismo el proyecto se encuentra alojado en al dirección <http://148.226.1.32>

# Diseño, desarrollo y programación

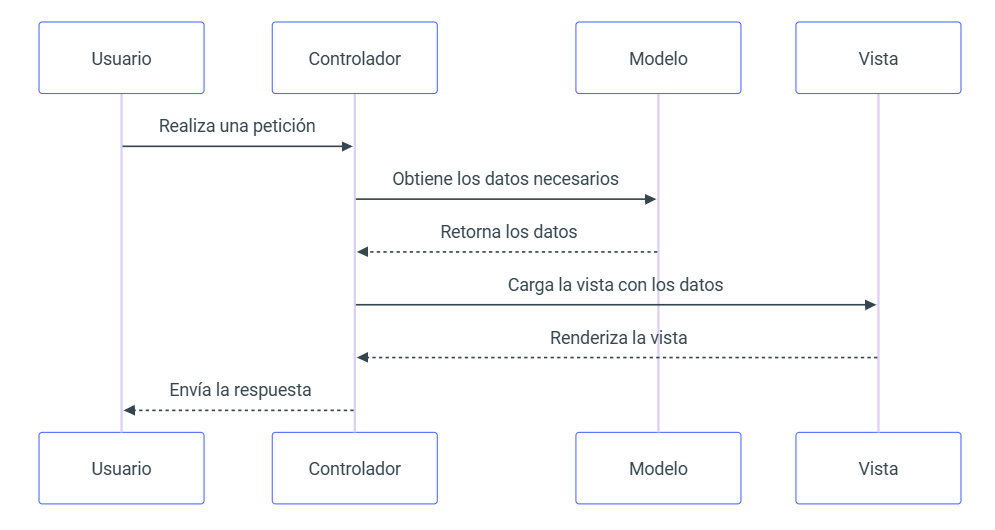
## Modelo de datos



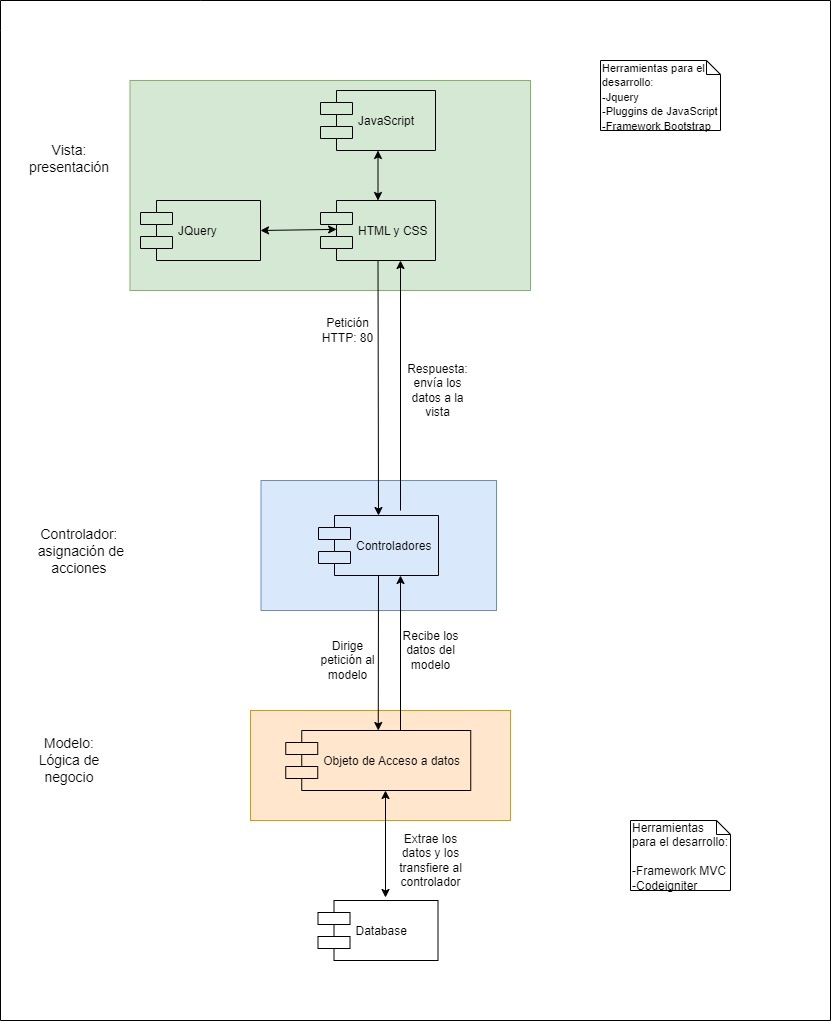
Diagramas de caso de uso



## Diagrama de secuencia



## Diagrama de arquitectura



## Requisitos de seguridad

Para proteger una aplicación web de vulnerabilidades comunes como inyección SQL, CSRF y XSS, CodeIgniter 4 ofrece varias medidas de seguridad:

Inyección SQL: CodeIgniter utiliza consultas preparadas que aseguran que los datos proporcionados por los usuarios sean tratados de manera segura en las consultas a la base de datos, evitando la ejecución de código malicioso.

CSRF: Se implementa un token CSRF con la función csrf\_field(), que verifica que las solicitudes provengan de fuentes legítimas, protegiendo la aplicación contra ataques de tipo CSRF.

XSS: La función esc() de CodeIgniter escapa los datos de usuario antes de mostrarlos en las vistas, evitando la ejecución de scripts maliciosos en el navegador.

Estas medidas ayudan a mitigar riesgos y a mantener la seguridad de la aplicación.

## Herramientas utilizadas para el desarrollo

**Visual Studio Code con la Extensión Copilot para Profesores**

Visual Studio Code (VSCode) es un editor de código fuente ligero y potente, ampliamente utilizado en el desarrollo de aplicaciones debido a su velocidad, personalización y soporte para múltiples lenguajes. Durante el desarrollo del sistema, se utilizó VSCode para escribir y editar el código, beneficiándose de su excelente integración con Git, su capacidad para depurar el código y su amplia gama de extensiones.

Una de las extensiones clave utilizadas fue GitHub Copilot para Profesores, que proporciona sugerencias inteligentes de código basadas en IA, ayudando a acelerar la escritura del código y mejorar la calidad del mismo. Copilot es especialmente útil para autocompletar bloques de código y sugerir implementaciones de funciones, lo que optimiza el proceso de desarrollo y reduce el tiempo de implementación.

**Navicat para la Gestión de Bases de Datos**

Navicat es una herramienta de administración de bases de datos que permite la gestión visual y eficiente de bases de datos en diversas plataformas, incluyendo MySQL y MariaDB, que fueron utilizadas en este proyecto. Navicat proporciona una interfaz gráfica intuitiva para crear y administrar bases de datos, diseñar estructuras, ejecutar consultas SQL y realizar copias de seguridad, lo cual facilita la administración y optimización de la base de datos del sistema de gestión de congresos.

Gracias a Navicat, fue posible gestionar y mantener la base de datos de manera eficaz, asegurando la integridad y consistencia de los datos a lo largo del ciclo de vida del proyecto.

# Bibliografías

Chen, M., Mao, S., & Liu, Y. (2019). Big data: A survey. Mobile Networks and Applications, 19(2), 171–209. https://doi.org/10.1007/s11036-019-1237-4

Laudon, K. C., & Laudon, J. P. (2020). Management information systems: Managing the digital firm (16th ed.). Pearson.

Pressman, R. S. (2021). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill Education.

Sommerville, I. (2019). Software engineering (10th ed.). Pearson.