Desarrollar Software a Partir de la Integración de sus Módulos Componentes

GA8-220501096-AA1-EV01

Presentado por: Alejandra Jhoana Gironza Bolaños

Análisis y Desarrollo de Software

Presentado a: Milton Ivan Barbosa Gaona

Centro de la Tecnología del Diseño y la Productividad Empresarial

SENA (Servicio Nacional de Aprendizaje)

Girardot - Cundinamarca

Octubre 29 del 2025

Tabla de Contenido

Tabla de Contenido 2

Tabla de Ilustraciones 6

Justificación 9

Objetivos 10

Objetivo General 10

Objetivos específicos 10

Introducción 12

1. Especificación de Requerimientos del Sistema 13

1.1. Requerimientos Funcionales (RF) 13

1.2. Requerimientos No Funcionales (RNF) 18

1.3. Requerimientos Legales y de Cumplimiento 22

2. Casos de Uso 23

2.1. Casos de Prueba del Módulo de Cronograma de Actividades 23

2.2. Casos de Prueba del Módulo de Usuarios, Autenticación y Roles 27

2.3. Casos de Prueba del Módulo de Seguridad y Control de Acceso 30

2.4. Casos de Prueba del Módulo de Comunicación y Notificaciones. 32

2.5. Casos de Prueba de Integración, Rendimiento y Mantenimiento. 36

3. Épicas e Historias de Usuario 38

3.1. Épica 1: Gestión de Usuarios y Autenticación 38

3.2. Épica 2: Gestión de Eventos Comunitarios 39

3.3. Épica 3: Aprobación de Eventos..... 40

3.4. Épica 4: Participación Comunitaria..... 41

3.5. Épica 5: Administración de Espacios y Finanzas 43

4. Informe sobre el IDE NetBeans 44

5. Diagramas de Cronograma de Actividades Comunitarias 45

5.1. Diagrama de Casos de Uso 45

5.2. Diagrama de Despliegue 46

5.3. Diagrama de Paquetes..... 47

5.4. Diagrama de Clases..... 48

5.5. Diagrama Entidad Relación..... 48

6. Mecanismos de seguridad que requiere la aplicación 49

7. Metodología SCRUM..... 51

8. Capas del sistema según la metodología SCRUM..... 52

9. Mapa de Navegación..... 53

10. Codificación Módulos en el Lenguaje Seleccionado 54

10.1. Módulo de Autenticación y Registro..... 54

10.2. Módulo de Ingreso 55

10.3. Módulo de Administración General 56

10.4. Módulo de Gestión de Usuarios..... 57

10.5. Módulo de Gestión de Eventos 58

10.6. Módulo de Comunicación / Notificaciones de Administrador 59

10.7. Módulo de Salida / Cierre de Sesión 60

11. Repositorio de Control de Versiones (GitHub) 60

12. Librerías Necesarias en las Capas de la Aplicación 61

12.1. Módulo de Autenticación y Registro..... 61

12.2. Módulo de Ingreso 61

12.3. Módulo de Consulta de Administrador 62

12.4. Módulo de Agregar Usuarios 62

12.5. Módulo de Modificar Eventos..... 63

12.6. Módulo de Salida / Cierre de Sesión 63

13. Determinación de Frameworks en cada Capa de la Aplicación 64

14. División del módulo en componentes reutilizables..... 65

15. Codificación con Buenas Prácticas de Escritura de Código 67

16. Código Fuente de cada Componente que Corresponde a un Módulo Dividido en Paquetes 69

17. Patrones de Diseño de Acuerdo con la Arquitectura del Software por Componente 71

18. Desarrollo de las Pruebas Unitarias de cada Módulo 73

18.1. Endpoint principal del proyecto: http://localhost:8080/ cronograma1/ 73

18.2. Endpoint de registro: http://localhost:8080/ registro.jsp/ 74

18.3. Endpoint de ingreso a la plataforma: http://localhost:8080/cronograma1/login.jsp..... 74

18.4. Endpoint de recuperación de contraseña: http://localhost:8080/cronograma1/recuperar.jsp 75

18.5. Endpoint de panel administrador, gestión de usuarios: http://localhost:8080/cronograma1/mod_admin_gestion_usuarios.jsp 75

18.6. Endpoint de panel administrador, gestión de eventos: http://localhost:8080/cronograma1/mod_admin_gestion_eventos.jsp 76

18.7. Endpoint de panel administrador, crear publicaciones: http://localhost:8080/cronograma1/muroAdmin.jsp..... 76

19. Configuraciones de Servidores y de Base de Datos..... 77

20. Documentación de Ambientes de Desarrollo y Pruebas 80

Conclusiones 81

Conclusión General..... 81

Conclusiones específicas..... 81

Bibliografía..... 82

Tabla de Ilustraciones

Tabla 1 Requerimientos Funcionales (RF)..... 13

Tabla 2 Requerimientos No Funcionales (RNF) 18

Tabla 3 Requerimientos Legales y de Cumplimiento 22

Tabla 4 Casos de Prueba del Módulo de Cronograma de Actividades 23

Tabla 5 Casos de Prueba del Módulo de Usuarios, Autenticación y Roles. 27

Tabla 6 Casos de Prueba del Módulo de Seguridad y Control de Acceso .. 30

Tabla 7 Casos de Prueba del Módulo de Comunicación y Notificaciones .. 32

Tabla 8 Casos de Prueba de Integración, Rendimiento y Mantenimiento .. 36

Tabla 9 Épica 1: Gestión de Usuarios y Autenticación..... 38

Tabla 10 Épica 2: Gestión de Eventos Comunitarios 39

Tabla 11 Épica 3: Aprobación de Eventos..... 40

Tabla 12 Épica 4: Participación Comunitaria 41

Tabla 13 Épica 5: Administración de Espacios y Finanzas 43

Tabla 14 Patrones de Diseño de Acuerdo con la Arquitectura del Software por Componente..... 71

Diagrama 1 de Casos de Uso..... 45

Diagrama 2 de Despliegue 46

Diagrama 3 de Paquetes 47

Diagrama 4 de Clases 48

Diagrama 5 Entidad Relación 48

Imagen 1 API de Mecanismos de seguridad 50
Imagen 2 Mapa de Navegación 53
Imagen 3 Código del Módulo de Autenticación y Registro 54
Imagen 4 Código del Módulo de Ingreso 55
Imagen 5 Código del Módulo de Administración General 56
Imagen 6 Código del Módulo de Gestión de Usuarios 57
Imagen 7 Código del Módulo de Gestión de Eventos 58
Imagen 8 Código del Módulo de Comunicación / Notificaciones de Administrador 59
Imagen 9 Código del Módulo de Salida / Cierre de Sesión 60
Imagen 10 Librerías del Módulo de Autenticación y Registro 61
Imagen 11 Librerías del Módulo de Ingreso 61
Imagen 12 Librerías del Módulo de Consulta de Administrador 62
Imagen 13 Librerías del Módulo de Agregar Usuarios 62
Imagen 14 Librerías del Módulo de Modificar Eventos 63
Imagen 15 Librerías del Módulo de Salida / Cierre de Sesión 63
Imagen 16 Componente Reutilizable de Conexión 66
Imagen 17 Componente Reutilizable de Estilos 66
Imagen 18 Buenas Prácticas de Escritura de Código en Styles.css 67
Imagen 19 Buenas Prácticas de Escritura de Código en LoginUser.java ... 68
Imagen 20 Buenas Prácticas de Escritura de Código en AgregarPresidente.java 68
Imagen 21 Paquete Web Pages 69
Imagen 22 Paquete Source Packages 70

Imagen 23 Endpoint principal del proyecto 73
Imagen 24 Endpoint de registro 74
Imagen 25 Endpoint de ingreso 74
Imagen 26 Endpoint de recuperación de contraseña 75
Imagen 27 Endpoint de panel administrador, gestión de usuarios 75
Imagen 28 Endpoint de panel administrador, gestión de eventos 76
Imagen 29 Endpoint de panel administrador, crear publicaciones 76
Imagen 30 Conexión a la Base de Datos 77
Imagen 31 Base de Datos MySQL 77
Imagen 32 Base de Datos MySQL 78
Imagen 33 Base de Datos MySQL 78
Imagen 34 Base de Datos MySQL 79
Imagen 35 Base de Datos MySQL 79

Justificación

El proyecto “Cronograma de Actividades Comunitarias” fue desarrollado con el propósito de optimizar la organización, gestión y comunicación de las actividades dentro de una comunidad, mediante una plataforma web moderna, segura y de fácil acceso. Tradicionalmente, las juntas de acción comunal y organizaciones locales enfrentan dificultades para difundir información, coordinar eventos y mantener la participación activa de sus habitantes.

Ante esta problemática, se plantea una solución tecnológica que centraliza la información, permite la interacción entre los distintos roles (administrador, presidente, organizador y habitante), y garantiza un flujo de comunicación claro y confiable. El sistema integra múltiples módulos —como autenticación, cronograma de eventos, gestión de usuarios y notificaciones— desarrollados bajo una arquitectura modular y principios de buenas prácticas de codificación.

Objetivos

Objetivo General

Desarrollar una aplicación web modular que permita administrar de manera eficiente el cronograma de actividades comunitarias, facilitando la interacción entre los diferentes roles de usuario, la organización de eventos, el control de asistencia y la comunicación mediante notificaciones, bajo estándares de calidad, seguridad y usabilidad.

Objetivos específicos

- Analizar los requerimientos funcionales, no funcionales y legales del sistema, garantizando que el software cumpla con las necesidades de la comunidad y las normativas vigentes sobre protección de datos personales.
- Diseñar la arquitectura del sistema aplicando el modelo MVC (Modelo–Vista–Controlador) y estructurando los componentes por capas de presentación, lógica de negocio y datos, para facilitar la escalabilidad y el mantenimiento.
- Implementar los módulos principales del sistema (autenticación, gestión de usuarios, eventos, comunicación y cierre de sesión) utilizando el lenguaje Java, Servlets, JSP y MySQL como gestor de base de datos.
- Incorporar patrones de diseño y buenas prácticas de programación como DAO, Singleton, DTO y Control de Acceso

Basado en Rol (RBAC) para garantizar la reutilización, seguridad y consistencia del código.

- Aplicar la metodología ágil SCRUM en la planificación y desarrollo de los sprints, asegurando entregas funcionales y control de versiones mediante GitHub.
- Diseñar y documentar los diagramas UML (de casos de uso, clases, despliegue, entidad-relación y navegación) para representar la estructura, comportamiento y flujo de la aplicación.
- Implementar y validar mecanismos de seguridad como el cifrado de contraseñas, el uso del protocolo HTTPS, la gestión de sesiones y el control de accesos según roles.
- Realizar pruebas unitarias, de integración, rendimiento y mantenimiento, garantizando la disponibilidad, estabilidad y buen desempeño del sistema en distintos entornos de ejecución.
- Documentar los resultados, configuraciones de servidor y ambientes de desarrollo, evidenciando el proceso técnico y metodológico que condujo a la integración completa del sistema.

Introducción

El documento presenta el proceso de desarrollo del sistema “Cronograma de Actividades Comunitarias”, una aplicación web diseñada para gestionar, difundir y controlar las actividades realizadas en una comunidad. Esta herramienta busca facilitar la participación ciudadana mediante una interfaz accesible que permite visualizar eventos, confirmar asistencia, recibir notificaciones y mantener una comunicación fluida entre los usuarios y administradores.

La solución fue desarrollada utilizando tecnologías como Java, Servlets, JSP y MySQL, bajo una arquitectura multicapa que separa la lógica del negocio, la presentación y la persistencia de datos. Además, se implementó la metodología ágil SCRUM, lo que permitió organizar el desarrollo en sprints, definir historias de usuario, generar pruebas continuas y documentar avances de forma iterativa.

El proyecto integra distintos componentes que abarcan desde el diseño de requerimientos y casos de uso hasta la codificación modular, las pruebas unitarias y la documentación técnica. Así, se refleja un proceso completo de desarrollo de software enfocado en la calidad, la colaboración y la aplicabilidad social.

1. Especificación de Requerimientos del Sistema

Versión: 2.0

Plataforma: Web (Java – Servlets/JSP – MySQL)

Despliegue: Nube (requiere conexión a Internet)

Nombre del proyecto: *Cronograma de Actividades Comunitarias*

1.1. Requerimientos Funcionales (RF)

Tabla 1 Requerimientos Funcionales (RF)

ID	Nombre del Requerimiento	Descripción	Prioridad	Tipo
RF-01	Visualización del cronograma	Los habitantes pueden visualizar el cronograma de actividades comunitarias con información de fecha, hora, título, descripción y ubicación.	Alta	Funcional
RF-02	Filtrado de actividades	Permite filtrar las actividades por tipo, fecha o ubicación.	Media	Funcional
RF-03	Creación de eventos	Los organizadores y administradores pueden crear nuevos eventos con todos los datos requeridos (fecha, hora,	Alta	Funcional

ID	Nombre del Requerimiento	Descripción	Prioridad	Tipo
		descripción, ubicación, tipo y responsable).		
RF-04	Edición y eliminación de eventos	Los administradores pueden editar o eliminar eventos; los organizadores solo sus propios eventos.	Alta	Funcional
RF-05	Solicitud de recursos y espacios	Los organizadores pueden solicitar reservas de espacios o recursos para los eventos creados.	Media	Funcional
RF-06	Aprobación de eventos	El presidente de la junta y los administradores pueden aprobar, rechazar o devolver eventos en revisión.	Alta	Funcional
RF-07	Gestión de usuarios	Permite crear cuentas de usuario para los diferentes roles (habitante, organizador,	Alta	Funcional

ID	Nombre del Requerimiento	Descripción	Prioridad	Tipo
		presidente, administrador, soporte técnico).		
RF-08	Autenticación y login	Los usuarios deben iniciar sesión con credenciales válidas y poder recuperar su contraseña.	Alta	Funcional
RF-09	Registro de usuario	Permite el registro de nuevos usuarios mediante un formulario con validación de correo electrónico.	Alta	Funcional
RF-10	Perfil de usuario	Cada usuario tendrá un perfil con información personal editable y permisos específicos.	Media	Funcional
RF-11	Control de roles y permisos	El sistema gestionará los permisos de acuerdo con el tipo de usuario (habitante, organizador, presidente, administrador, soporte técnico).	Alta	Funcional

ID	Nombre del Requerimiento	Descripción	Prioridad	Tipo
RF-12	Confirmación de asistencia	Los habitantes podrán confirmar o cancelar su asistencia a los eventos publicados.	Alta	Funcional
RF-13	Lista de asistentes	Los organizadores y administradores podrán visualizar la lista de asistentes confirmados.	Media	Funcional
RF-14	Calificación de eventos	Los habitantes podrán evaluar y calificar los eventos en los que participaron.	Media	Funcional
RF-15	Retroalimentación y comentarios	Los usuarios podrán dejar comentarios o sugerencias sobre los eventos.	Media	Funcional
RF-16	Visualización pública	Los visitantes no registrados pueden ver solo información básica de eventos pasados.	Baja	Funcional
RF-17	Gestión de notificaciones	El sistema enviará notificaciones a los usuarios sobre eventos,	Alta	Funcional

ID	Nombre del Requerimiento	Descripción	Prioridad	Tipo
		cambios, confirmaciones y recordatorios.		
RF-18	Personalización de notificaciones	Los usuarios podrán configurar sus preferencias de notificación.	Media	Funcional
RF-19	Estadísticas e informes	El sistema permitirá generar estadísticas e informes de participación y evaluación de eventos.	Media	Funcional
RF-20	Encuestas comunitarias	Se podrán crear y responder encuestas sobre las actividades o necesidades de la comunidad.	Baja	Funcional
RF-21	Administración técnica	El rol de soporte técnico podrá realizar mantenimiento, actualizaciones y revisión del rendimiento del sistema.	Alta	Funcional
RF-22	Integración con plataformas externas	El sistema podrá integrarse con redes sociales o herramientas de	Media	Funcional

ID	Nombre del Requerimiento	Descripción	Prioridad	Tipo
		mensajería para difusión y coordinación.		

Nota: Tabla realizada por Jhoana Gironza.

1.2. Requerimientos No Funcionales (RNF)

Tabla 2 Requerimientos No Funcionales (RNF)

ID	Categoría	Descripción	Prioridad	Tipo
RNF-01	Rendimiento	El sistema debe manejar múltiples usuarios y eventos simultáneamente sin pérdida de rendimiento.	Alta	No Funcional
RNF-02	Rendimiento	Las páginas deben cargarse en menos de 3 segundos bajo condiciones normales.	Alta	No Funcional
RNF-03	Seguridad	Debe implementarse autenticación segura (por ejemplo, contraseñas encriptadas y sesiones protegidas).	Alta	No Funcional
RNF-04	Seguridad	El sistema debe ser resistente a ataques de inyección SQL, XSS y CSRF.	Alta	No Funcional

ID	Categoría	Descripción	Prioridad	Tipo
RNF-05	Usabilidad	La interfaz debe ser intuitiva, accesible y adecuada para usuarios con diferentes niveles técnicos.	Alta	No Funcional
RNF-06	Usabilidad	Debe incluir ayuda contextual y mensajes de error comprensibles.	Media	No Funcional
RNF-07	Escalabilidad	El sistema debe poder ampliarse fácilmente para manejar más usuarios, eventos o módulos.	Media	No Funcional
RNF-08	Compatibilidad	Compatible con los principales navegadores (Chrome, Firefox, Edge) y dispositivos móviles.	Alta	No Funcional
RNF-09	Accesibilidad	Cumplimiento con las pautas de accesibilidad WCAG 2.1 (contraste, texto alternativo, navegación por teclado).	Alta	No Funcional
RNF-10	Integración	Debe permitir integración con redes sociales y sistemas externos mediante APIs.	Media	No Funcional

ID	Categoría	Descripción	Prioridad	Tipo
RNF-11	Privacidad y datos	Cumplimiento con la Ley 1581 de 2012 y normas de protección de datos personales en Colombia.	Alta	No Funcional
RNF-12	Privacidad y datos	Los usuarios deben poder decidir qué información personal comparten.	Alta	No Funcional
RNF-13	Disponibilidad	El sistema debe garantizar una disponibilidad mínima del 99% mensual.	Alta	No Funcional
RNF-14	Mantenimiento	El código debe estar documentado para facilitar el mantenimiento por el equipo técnico.	Media	No Funcional
RNF-15	Portabilidad	La arquitectura debe permitir la futura migración a frameworks como React sin rediseñar el backend.	Media	No Funcional
RNF-16	Backup	Se deben realizar copias de seguridad automáticas de la base de datos diariamente.	Alta	No Funcional

ID	Categoría	Descripción	Prioridad	Tipo
RNF-17	Soporte técnico	Debe existir un módulo de monitoreo y reporte de fallos para soporte técnico.	Media	No Funcional
RNF-18	Localización	El sistema debe estar disponible en idioma español y permitir adaptación a otros idiomas en el futuro.	Baja	No Funcional

Nota: Tabla realizada por Jhoana Gironza.

1.3. Requerimientos Legales y de Cumplimiento

Tabla 3 Requerimientos Legales y de Cumplimiento

ID	Nombre del Requerimiento	Descripción	Prioridad	Tipo
RL-01	Protección de datos personales	Cumplir con la Ley 1581 de 2012 sobre protección de datos personales en Colombia.	Alta	Legal
RL-02	Consentimiento informado	Los usuarios deben aceptar términos y políticas antes de registrarse o compartir datos.	Alta	Legal
RL-03	Derecho de eliminación de datos	Los usuarios pueden solicitar la eliminación de su cuenta y datos personales.	Media	Legal
RL-04	Propiedad intelectual	El software y los contenidos generados deben respetar derechos de autor y licencias.	Media	Legal
RL-05	Transparencia en el uso de información	Debe informarse claramente cómo se usan los datos personales recolectados.	Alta	Legal

Nota: Tabla realizada por Jhoana Gironza.

2. Casos de Uso

2.1. Casos de Prueba del Módulo de Cronograma de Actividades

Tabla 4 Casos de Prueba del Módulo de Cronograma de Actividades

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-01	Visualización del Cronograma de Actividades	Verificar que los usuarios autenticados puedan visualizar correctamente las actividades programadas.	El usuario ha iniciado sesión como usuario registrado. Existen actividades registradas en el sistema.	1. Iniciar sesión. 2. Ir al módulo “Cronograma de Actividades”. 3. Verificar que se muestren las actividades con fecha, hora, título, descripción y ubicación.	Se muestran todas las actividades correctamente con su información completa y legible.
CP-02	Filtrado del Cronograma	Validar que los usuarios puedan aplicar filtros por tipo de actividad, fecha o ubicación.	Usuario autenticado. Existen actividades con distintos tipos, fechas y ubicaciones.	1. Iniciar sesión. 2. Acceder al cronograma. 3. Aplicar un filtro por tipo, fecha o ubicación.	El sistema muestra únicamente las actividades que cumplen con los criterios seleccionados.

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
				4. Observar resultados.	
CP-03	Solicitud de Creación de Evento por Organizador	Verificar que los organizadores puedan solicitar la creación de un evento, el cual requiere aprobación antes de publicarse.	Usuario autenticado como organizador.	1. Iniciar sesión como organizador. 2. Ir a “Solicitar nuevo evento”. 3. Completar formulario con datos del evento (fecha, hora, título, descripción, ubicación, tipo, recursos, responsables). 4. Enviar solicitud.	El sistema registra la solicitud y la deja en estado Pendiente de aprobación. Se notifica al presidente o administrador.
CP-04	Aprobación de Evento por Presidente o Administrador	Verificar que el presidente o administrador pueda revisar solicitudes de creación y	Existen solicitudes pendientes enviadas por organizadores. Usuario autenticado como presidente o administrador.	1. Iniciar sesión. 2. Acceder al módulo de solicitudes. 3. Revisar la información del evento. 4. Aprobar o	Si se aprueba, el evento se publica en el cronograma. Si se rechaza, se notifica al

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
		aprobarlas o rechazarlas.		rechazar la solicitud.	organizador con observaciones.
CP-05	Creación Directa de Evento por Administrador	Verificar que el administrador pueda crear y publicar directamente eventos sin requerir aprobación.	Usuario autenticado como administrador.	1. Iniciar sesión como administrador. 2. Acceder al módulo “Crear evento”. 3. Ingresar los datos del evento. 4. Guardar.	El evento se publica inmediatamente en el cronograma y es visible para todos los usuarios.
CP-06	Edición de Eventos	Verificar que los administradores y presidentes puedan editar eventos existentes.	Usuario autenticado como administrador o presidente. Existen eventos en el sistema.	1. Iniciar sesión. 2. Ir al cronograma. 3. Seleccionar un evento. 4. Modificar información (fecha, hora, descripción, etc.). 5. Guardar cambios.	El sistema actualiza la información y muestra los datos modificados correctamente.

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-07	Eliminación de Eventos	Verificar que los administradores puedan eliminar eventos existentes.	Usuario autenticado como administrador.	1. Iniciar sesión. 2. Ir al cronograma. 3. Seleccionar evento. 4. Oprimir “Eliminar”. 5. Confirmar acción.	El evento se elimina del sistema y deja de mostrarse en el cronograma. Se registra el historial de eliminación.
CP-08	Notificación de Cambios	Verificar que los usuarios inscritos a un evento reciban notificación cuando el evento es modificado o cancelado.	Existen usuarios inscritos a un evento.	1. Iniciar sesión como administrador. 2. Editar o eliminar un evento. 3. Observar notificaciones enviadas a usuarios registrados.	Los usuarios reciben notificación de modificación o cancelación vía sistema o correo (según configuración).

Nota: Tabla realizada por Jhoana Gironza.

2.2. Casos de Prueba del Módulo de Usuarios, Autenticación y Roles

Tabla 5 Casos de Prueba del Módulo de Usuarios, Autenticación y Roles

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-09	Registro de Nuevo Usuario	Verificar que los nuevos usuarios puedan registrarse correctamente en el sistema.	El usuario no tiene cuenta registrada.	1. Acceder a la página principal. 2. Seleccionar “Registrarse”. 3. Ingresar nombre, correo, contraseña y demás información. 4. Confirmar registro.	El sistema crea la cuenta, muestra mensaje de éxito y permite iniciar sesión.
CP-10	Inicio de Sesión	Validar que los usuarios registrados puedan autenticarse con sus credenciales válidas.	El usuario está previamente registrado.	1. Acceder a la página de inicio de sesión. 2. Ingresar correo y contraseña. 3. Presionar “Iniciar sesión”.	El sistema valida las credenciales y redirige al panel correspondiente según el rol.

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-11	Inicio de Sesión Inválido	Verificar que el sistema muestre mensajes de error cuando las credenciales son incorrectas.	Usuario no autenticado.	1. Acceder a inicio de sesión. 2. Ingresar correo o contraseña incorrecta. 3. Presionar “Iniciar sesión”.	Se muestra mensaje: “Usuario o contraseña incorrectos”. El acceso no se permite.
CP-12	Cierre de Sesión	Verificar que los usuarios puedan cerrar sesión correctamente.	Usuario autenticado.	1. Iniciar sesión. 2. Seleccionar la opción “Cerrar sesión”.	El sistema finaliza la sesión y redirige a la página principal o de login.
CP-13	Recuperación de Contraseña	Validar que los usuarios puedan recuperar su contraseña mediante correo registrado.	Usuario con cuenta activa y correo válido.	1. Acceder a “¿Olvidaste tu contraseña?”. 2. Ingresar correo registrado. 3. Revisar correo y seguir enlace de recuperación. 4. Ingresar nueva contraseña.	El sistema actualiza la contraseña y notifica la recuperación exitosa.

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-14	Asignación de Roles	Verificar que el administrador pueda asignar o modificar roles de usuario (por ejemplo, de organizador a presidente).	Usuario autenticado como administrador. Existen usuarios en el sistema.	1. Iniciar sesión como administrador. 2. Ir a “Gestión de usuarios”. 3. Seleccionar usuario. 4. Modificar rol. 5. Guardar cambios.	El rol se actualiza correctamente y el usuario obtiene nuevos permisos al iniciar sesión.
CP-15	Restricción de Accesos según Rol	Verificar que cada rol (usuario, organizador, presidente, administrador) solo acceda a las secciones permitidas.	Existen usuarios de diferentes roles.	1. Iniciar sesión con cada rol. 2. Intentar acceder a módulos restringidos (por ejemplo, edición de eventos desde un usuario normal).	El sistema bloquea el acceso no autorizado y muestra mensaje de “Acceso denegado”.
CP-16	Validación de Sesión Expirada	Comprobar que el sistema redirija al login si la	Usuario autenticado. Tiempo de sesión	1. Iniciar sesión. 2. No interactuar durante el tiempo límite. 3. Intentar	El sistema redirige automáticamente al login e informa

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
		sesión expira por inactividad.	configurado (por ejemplo, 15 min).	navegar a otra página.	que la sesión expiró.
CP-17	Auditoría de Inicios de Sesión	Verificar que el sistema registre fecha, hora y dirección IP en cada inicio de sesión.	Sistema configurado con módulo de auditoría.	1. Iniciar sesión con usuario válido. 2. Acceder al registro de auditoría (solo visible para administrador).	El sistema muestra el registro del inicio con la información correspondiente.

Nota: Tabla realizada por Jhoana Gironza.

2.3. Casos de Prueba del Módulo de Seguridad y Control de Acceso

Tabla 6 Casos de Prueba del Módulo de Seguridad y Control de Acceso

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-18	Validación de Contraseña Segura	Verificar que el sistema exija contraseñas seguras (mínimo 8	Usuario intentando registrarse o cambiar contraseña.	1. Acceder a registro o cambio de contraseña.	El sistema muestra advertencia y no permite continuar hasta cumplir los

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
		caracteres, mayúscula, número y símbolo).		2. Ingresar una contraseña débil (ej. “12345”). 3. Guardar.	requisitos de seguridad.
CP-19	Cifrado de Contraseñas	Validar que las contraseñas se almacenen cifradas en la base de datos.	Base de datos accesible por administrador técnico.	1. Crear usuario. 2. Consultar tabla de usuarios en la base de datos.	La contraseña no se almacena en texto plano (se observa cifrada o en hash).
CP-20	Control de Sesiones Simultáneas	Comprobar que un usuario no pueda iniciar sesión simultáneamente desde varios dispositivos.	Usuario con cuenta activa.	1. Iniciar sesión en un dispositivo A. 2. Intentar iniciar sesión en dispositivo B con la misma cuenta.	El sistema bloquea la segunda sesión o cierra la primera, según configuración.
CP-21	Acceso HTTPS Seguro	Verificar que la aplicación web solo sea accesible	Servidor desplegado en	1. Intentar acceder con “http://”.	El acceso HTTP es redirigido

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
		mediante protocolo HTTPS.	entorno de pruebas o nube.	2. Intentar acceder con “https://”.	automáticamente a HTTPS.

Nota: Tabla realizada por Jhoana Gironza.

2.4. Casos de Prueba del Módulo de Comunicación y Notificaciones

Tabla 7 Casos de Prueba del Módulo de Comunicación y Notificaciones

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-22	Envío de Notificación de Nuevo Evento	Verificar que el sistema notifique automáticamente a los usuarios cuando se publica un nuevo evento aprobado.	Existen usuarios registrados con notificaciones activas. Un evento ha sido aprobado o creado.	1. Iniciar sesión como administrador o presidente. 2. Aprobar o crear un evento. 3. Observar notificación enviada a los usuarios.	Los usuarios reciben una notificación (en la bandeja del sistema o correo) con los detalles del nuevo evento.

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-23	Notificación de Actualización de Evento	Validar que los usuarios inscritos sean notificados cuando se modifica un evento.	Existen usuarios inscritos a un evento.	1. Iniciar sesión como administrador o presidente. 2. Modificar la información del evento. 3.Guardar cambios	El sistema envía una notificación automática con la información actualizada.
CP-24	Notificación de Cancelación de Evento	Verificar que los usuarios inscritos reciban aviso cuando un evento se cancela.	Evento existente con usuarios inscritos.	1. Iniciar sesión como administrador. 2. Eliminar o marcar el evento como cancelado. 3. Confirmar acción	Los usuarios reciben una notificación indicando la cancelación, con posibles motivos.
CP-25	Bandeja de Mensajes Internos	Validar que los usuarios puedan consultar mensajes internos enviados por administradores u organizadores.	Usuario autenticado con mensajes en la bandeja.	1. Iniciar sesión. 2. Ir a “Mensajes” o “Notificaciones”. 3. Visualizar mensajes recibidos	Se muestran los mensajes con fecha, remitente y contenido correctamente.

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-26	Envío de Mensajes a Participantes	Verificar que los organizadores y administradores puedan enviar mensajes directos a los participantes de un evento.	Evento existente con participantes.	1. Iniciar sesión como organizador o administrador. 2. Seleccionar evento. 3. Elegir opción “Enviar mensaje a participantes”. 4. Escribir y enviar.	Los participantes reciben el mensaje en su bandeja interna o vía correo según configuración.
CP-27	Confirmación de Lectura de Mensajes	Verificar que el sistema registre cuándo un usuario visualiza un mensaje o notificación.	Usuario autenticado con mensajes sin leer.	1. Iniciar sesión. 2. Abrir un mensaje recibido. 3. Consultar registro o estado del mensaje.	El sistema marca el mensaje como “Leído” y actualiza el estado en la base de datos.
CP-28	Envío de Recordatorios Automáticos	Validar que el sistema envíe recordatorios automáticos antes de la fecha de un evento.	Existen eventos próximos con usuarios inscritos.	1. Simular fecha cercana al evento (1 o 2 días antes). 2. Revisar notificaciones automáticas.	El sistema envía recordatorio con fecha, hora y lugar del evento.

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-29	Comunicación entre Roles	Verificar que los usuarios puedan comunicarse según jerarquía (por ejemplo, organizador → presidente → administrador).	Módulo de mensajería activo. Usuarios con distintos roles.	1. Iniciar sesión como organizador. 2. Enviar mensaje al presidente solicitando aprobación de evento. 3. Iniciar sesión como presidente y revisar mensajes.	El presidente visualiza la solicitud y puede responder o aprobar directamente.
CP-30	Gestión de Notificaciones por Preferencias	Comprobar que los usuarios puedan activar o desactivar ciertos tipos de notificaciones (correo, sistema, móvil).	Usuario autenticado.	1. Iniciar sesión. 2. Ir a “Configuración de notificaciones”. 3. Activar/desactivar opciones. 4.Guardar cambios	Las preferencias se actualizan correctamente y el sistema respeta dichas configuraciones.

Nota: Tabla realizada por Jhoana Gironza.

2.5. Casos de Prueba de Integración, Rendimiento y Mantenimiento

Tabla 8 Casos de Prueba de Integración, Rendimiento y Mantenimiento

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-31	Rendimiento en Carga del Cronograma	Validar que el cronograma cargue en menos de 3 segundos con hasta 100 actividades.	Base de datos con múltiples actividades registradas.	1. Iniciar sesión. 2. Acceder al cronograma. 3. Medir tiempo de carga.	El cronograma carga completamente en menos de 3 segundos.
CP-32	Respaldo Automático de Datos	Comprobar que se realicen copias automáticas de seguridad de la base de datos.	Sistema configurado con tareas de respaldo.	1. Revisar logs o carpeta de respaldos. 2. Validar fecha y hora del último backup.	Se genera una copia automática en el intervalo configurado.
CP-33	Recuperación ante Fallos	Verificar que el sistema pueda restaurarse correctamente a partir de un respaldo reciente.	Backup disponible.	1. Simular pérdida parcial de datos. 2. Ejecutar proceso de restauración.	Los datos se recuperan completamente sin pérdida de integridad.

ID	Nombre del Caso de Prueba	Descripción	Precondiciones	Pasos	Resultado Esperado
CP-34	Disponibilidad Web	Comprobar que la aplicación esté accesible en la nube y responda a solicitudes HTTP/HTTPS.	Aplicación desplegada en servidor remoto.	1. Ingresar la URL del sistema desde diferentes navegadores o dispositivos. 2. Probar conexión	El sistema es accesible vía HTTPS desde internet y responde correctamente.
CP-35	Escalabilidad y Concurrencia	Validar que múltiples usuarios puedan acceder simultáneamente sin fallos ni lentitud.	Sistema operativo y servidor activos.	1. Simular 20+ usuarios accediendo al cronograma y a la creación de eventos. 2. Observar comportamiento.	El sistema responde sin caídas ni bloqueos, manteniendo buen rendimiento.

Nota: Tabla realizada por Jhoana Gironza.

3. Épicas e Historias de Usuario

3.1. Épica 1: Gestión de Usuarios y Autenticación

Rol principal: Usuario y Administrador

Objetivo: Permitir la creación, autenticación y administración de usuarios para garantizar un acceso controlado al sistema.

Tabla 9 Épica 1: Gestión de Usuarios y Autenticación

ID	Historia de Usuario	Prioridad	Criterios de Aceptación
HU-01	Como *Usuario* quiero registrarme en el sistema para poder participar en los eventos comunitarios.	Alta	Validar campos obligatorios (correo y cédula únicos). Confirmar registro exitoso.
HU-02	Como *Usuario* quiero iniciar sesión en el sistema para acceder a mis eventos de forma segura.	Alta	Validar correo y contraseña. Mostrar error si son incorrectos. Acceso solo a usuarios autenticados.
HU-16	Como *Administrador* quiero gestionar usuarios del sistema (crear, actualizar, eliminar) para mantener actualizado el registro.	Alta	CRUD sobre la tabla Usuario. Validar datos únicos.

Nota: Tabla realizada por Jhoana Gironza.

3.2. Épica 2: Gestión de Eventos Comunitarios

Rol principal: Organizador

Objetivo: Permitir a los organizadores registrar, modificar y solicitar aprobación de eventos comunitarios.

Tabla 10 Épica 2: Gestión de Eventos Comunitarios

ID	Historia de Usuario	Prioridad	Criterios de Aceptación
HU-09	Como *Organizador* quiero crear eventos con título, descripción, fecha y espacio para difundir actividades comunitarias.	Alta	Registrar nuevo evento con estado “Borrador”. Validar organizador y espacio asignado.
HU-10	Como *Organizador* quiero modificar eventos creados por mí para ajustar cambios de fecha, espacio o estado.	Alta	Solo el organizador asignado puede editar. Registrar fecha de última modificación.
HU-11	Como *Organizador* quiero enviar solicitudes de aprobación de eventos al presidente de la junta para su publicación oficial.	Alta	Crear registro en Solicitud_Eventos. Cambiar estado del evento a “Revisión”. Notificar al presidente.

ID	Historia de Usuario	Prioridad	Criterios de Aceptación
HU-12	Como *Organizador* quiero asignar un espacio a cada evento para asegurar disponibilidad del lugar.	Media	Mostrar lista de espacios. Validar disponibilidad por fecha. Guardar relación en Evento.id_espacio.

Nota: Tabla realizada por Jhoana Gironza.

3.3. Épica 3: Aprobación de Eventos

Rol principal: Presidente de Junta

Objetivo: Permitir al presidente revisar, aprobar o rechazar las solicitudes de eventos enviadas por los organizadores.

Tabla 11 Épica 3: Aprobación de Eventos

ID	Historia de Usuario	Prioridad	Criterios de Aceptación
HU-13	Como *Presidente de Junta* quiero revisar solicitudes de eventos para verificar que cumplan con los requisitos comunitarios.	Alta	Mostrar solicitudes pendientes. Relacionar con Solicitud_Eventos. Permitir aprobar o rechazar.
HU-14	Como *Presidente de Junta* quiero aprobar eventos para	Alta	Cambiar solicitud a “Aceptada”. Estado del evento a “Aprobado”. Notificar organizador.

ID	Historia de Usuario	Prioridad	Criterios de Aceptación
	que puedan realizarse oficialmente.		
HU-15	Como *Presidente de Junta* quiero rechazar eventos para evitar actividades que no cumplan los criterios establecidos.	Alta	Cambiar solicitud a “Rechazada”. Estado del evento a “Rechazado”. Notificar organizador.

Nota: Tabla realizada por Jhoana Gironza.

3.4. Épica 4: Participación Comunitaria

Rol principal: Usuario

Objetivo: Permitir a los usuarios visualizar, confirmar asistencia y recibir certificados de participación en los eventos.

Tabla 12 Épica 4: Participación Comunitaria

ID	Historia de Usuario	Prioridad	Criterios de Aceptación
HU-03	Como *Usuario* quiero consultar la lista de eventos disponibles para decidir en cuáles participar.	Alta	Mostrar lista con título, fecha y estado. Permitir filtrar.

ID	Historia de Usuario	Prioridad	Criterios de Aceptación
HU-04	Como *Usuario* quiero confirmar mi asistencia a un evento.	Alta	Registrar asistencia con confirmo = TRUE. Mostrar mensaje de confirmación.
HU-05	Como *Usuario* quiero registrar mi asistencia real al evento para obtener certificado.	Alta	Asistencia con asistio = TRUE. Generar certificado asociado.
HU-06	Como *Usuario* quiero visualizar detalles de los eventos (fecha, lugar, organizador, estado).	Media	Mostrar datos del evento y espacio.
HU-07	Como *Usuario* quiero recibir notificaciones cuando un evento que confirmé cambie de fecha o estado.	Media	Detectar cambios, enviar notificación y registrar historial.
HU-08	Como *Usuario* quiero descargar mi certificado de asistencia.	Media	Generar PDF de Certificado_Eventos. Solo si asistio = TRUE.

Nota: Tabla realizada por Jhoana Gironza.

3.5. Épica 5: Administración de Espacios y Finanzas

Rol principal: Administrador

Objetivo: Permitir al administrador controlar los espacios, pagos y reportes del sistema.

Tabla 13 Épica 5: Administración de Espacios y Finanzas

ID	Historia de Usuario	Prioridad	Criterios de Aceptación
HU-17	Como *Administrador* quiero registrar pagos de uso de espacios para controlar la financiación.	Media	CRUD sobre Espacio_Pago. Registrar fecha y costo.
HU-18	Como *Administrador* quiero gestionar los espacios disponibles.	Media	CRUD sobre Espacio. Validar que no haya eventos asociados antes de eliminar.
HU-19	Como *Administrador* quiero generar reportes de asistencia a eventos.	Baja	Consultar Asistencia. Mostrar totales por evento y usuario. Exportar a PDF/Excel.

Nota: Tabla realizada por Jhoana Gironza.

4. Informe sobre el IDE NetBeans

NetBeans es un entorno de desarrollo integrado (IDE) muy utilizado para crear aplicaciones en diferentes lenguajes de programación, especialmente en Java, aunque también permite trabajar con PHP, HTML, JavaScript, C++, entre otros. Es una herramienta gratuita y de código abierto, lo que significa que cualquier persona puede descargarla, usarla y adaptarla a sus necesidades. Su principal objetivo es facilitar el trabajo de los desarrolladores al ofrecer un espacio completo donde se puede escribir, depurar, compilar y ejecutar código dentro de una misma plataforma.

Entre sus características más importantes se destacan su interfaz gráfica intuitiva, el resaltado de sintaxis, el autocompletado de código, la integración con bases de datos y la posibilidad de crear interfaces gráficas con solo arrastrar y soltar componentes (lo cual es muy útil para proyectos con ventanas o formularios). Además, NetBeans permite administrar proyectos grandes mediante su sistema de organización por módulos, y cuenta con herramientas que ayudan a detectar errores o mejorar el rendimiento del código.

Entre sus ventajas se encuentra que es un IDE completo, gratuito y fácil de usar, ideal tanto para principiantes como para desarrolladores con más experiencia. También ofrece compatibilidad con servidores y frameworks comunes, lo que facilita el desarrollo de aplicaciones web o de escritorio. Como desventaja, puede ser más pesado y lento al iniciar en comparación con otros IDE más ligeros, y algunas de sus funciones avanzadas requieren equipos con buena capacidad de memoria. Aun así, NetBeans sigue siendo una excelente opción para proyectos

educativos y profesionales, como el sistema de cronograma de actividades comunitarias, donde su entorno visual y sus herramientas integradas permiten desarrollar aplicaciones de manera más ordenada y eficiente.

5. Diagramas de Cronograma de Actividades Comunitarias

5.1. Diagrama de Casos de Uso

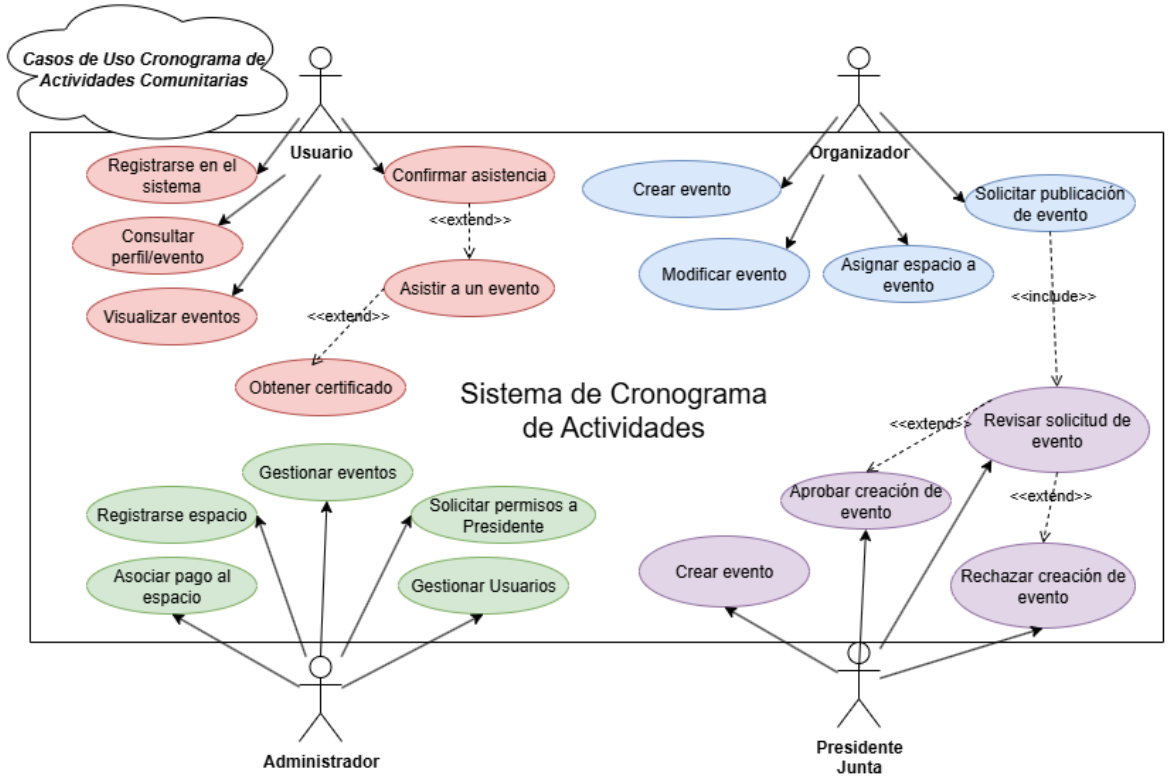


Diagrama 1 de Casos de Uso

Nota: Diagrama realizado por Jhoana Gironza.

5.2. Diagrama de Despliegue

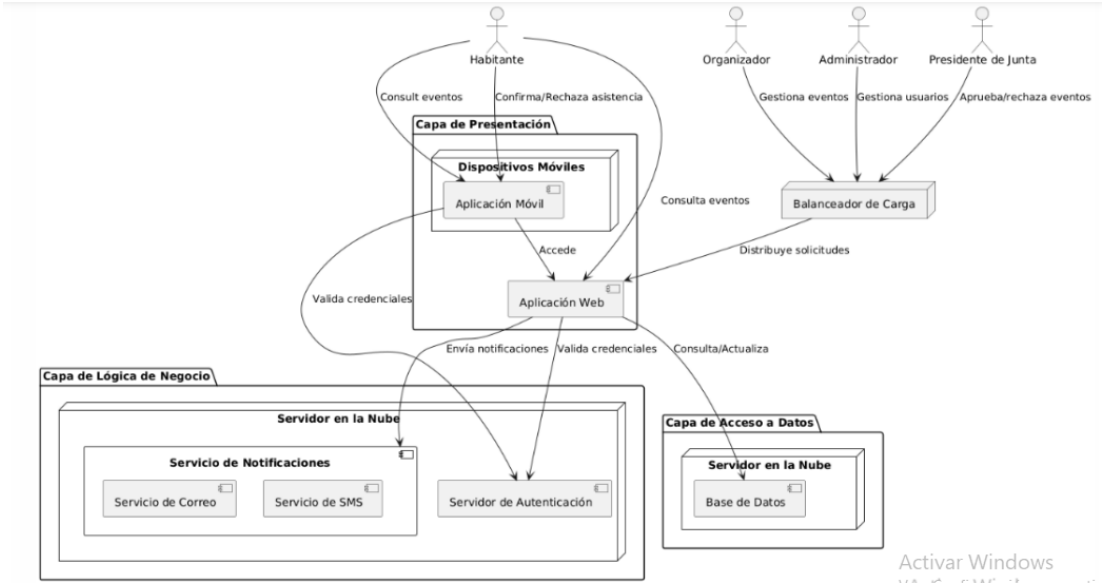


Diagrama 2 de Despliegue

Nota: Diagrama realizado por Jhoana Gironza.

5.3. Diagrama de Paquetes

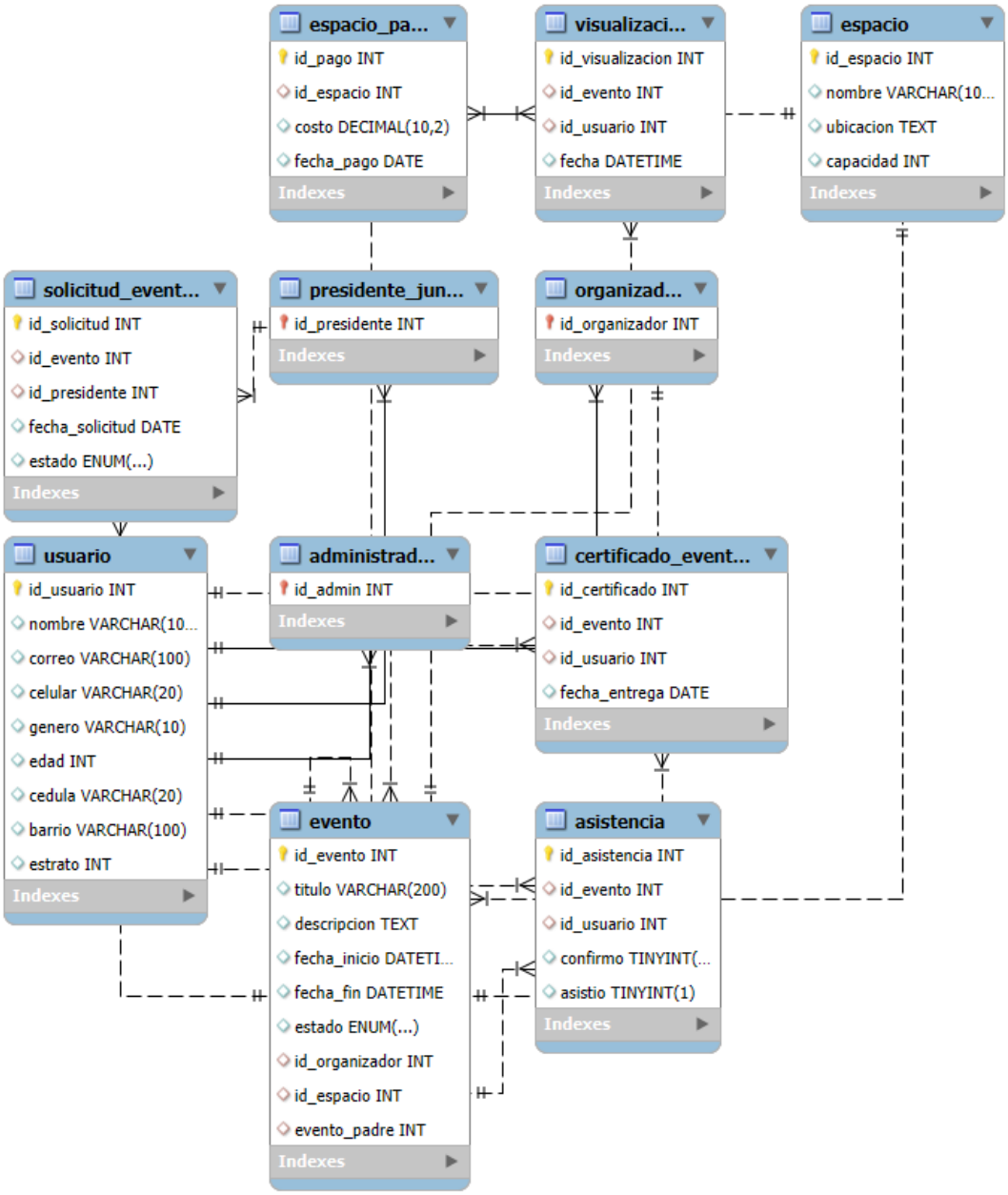


Diagrama 3 de Paquetes

Nota: Diagrama realizado por Jhoana Gironza.

5.4. Diagrama de Clases

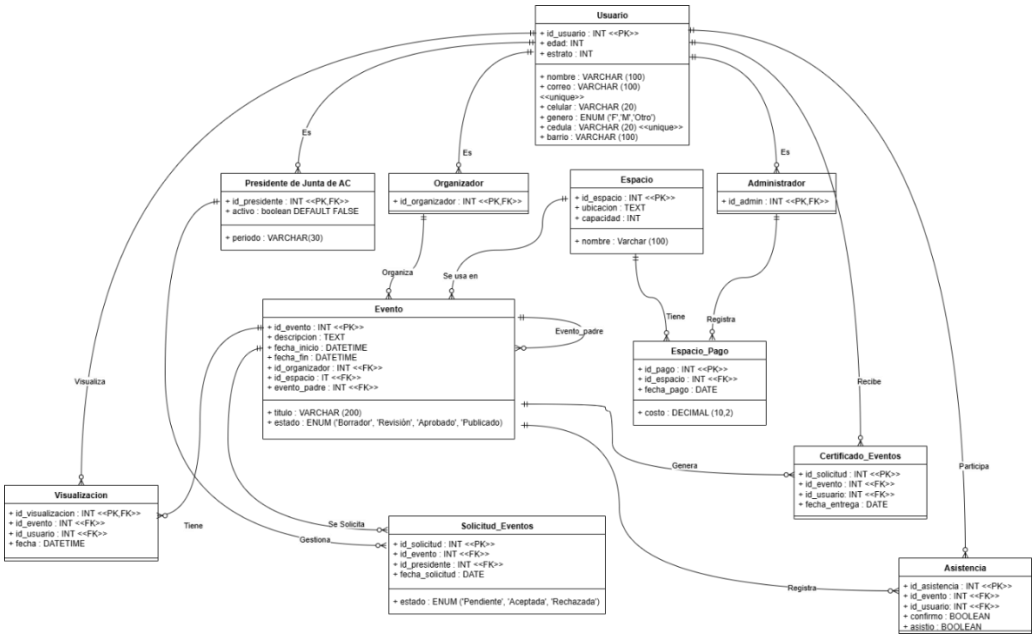


Diagrama 4 de Clases

Nota: Diagrama realizado por Jhoana Gironza.

5.5. Diagrama Entidad Relación

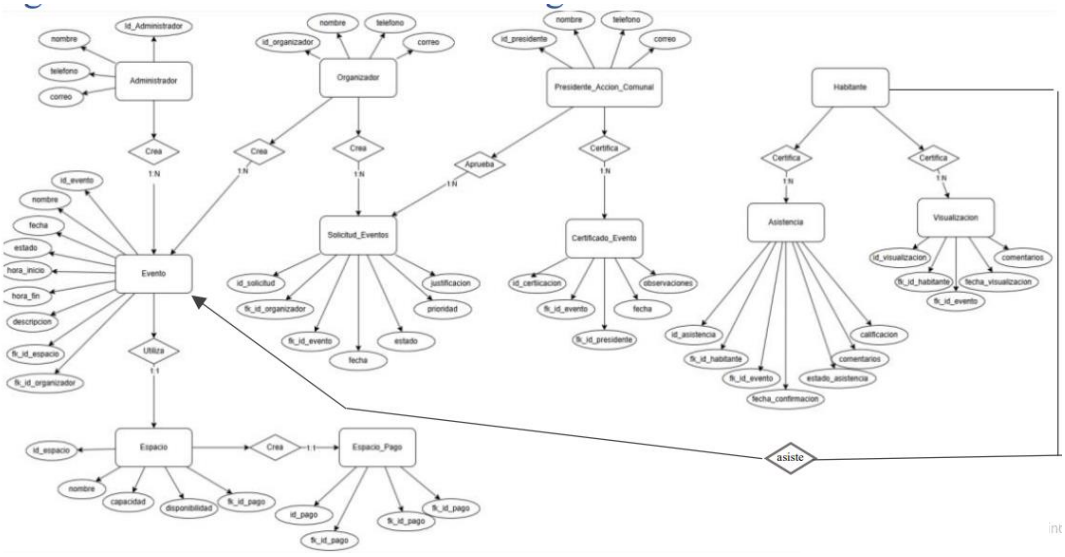


Diagrama 5 Entidad Relación

Nota: Diagrama realizado por Jhoana Gironza.

6. Mecanismos de seguridad que requiere la aplicación

Actualmente, la aplicación cuenta con un mecanismo básico de seguridad implementado a través de la API desarrollada, la cual controla el acceso al sistema mediante un proceso de autenticación de usuario y contraseña. Este procedimiento verifica que las credenciales ingresadas correspondan a registros válidos en la base de datos, permitiendo así que solo los usuarios autorizados puedan ingresar al aplicativo. Este control inicial es fundamental para proteger la información y evitar accesos no permitidos.

Sin embargo, es importante seguir fortaleciendo la seguridad del sistema. Entre los mecanismos de seguridad adicionales recomendados, se incluyen el cifrado de contraseñas en la base de datos, el uso de protocolos HTTPS para proteger la comunicación entre el cliente y el servidor, y la gestión de sesiones seguras para evitar que usuarios no autorizados puedan acceder sin volver a autenticarse. También se pueden implementar roles y permisos diferenciados, de modo que cada tipo de usuario (administrador, presidente, organizador o habitante) solo tenga acceso a las funciones que le correspondan.

En el anexo fotográfico se presenta la interfaz de la API donde se evidencia el proceso de autenticación, mostrando cómo se valida el usuario y la contraseña registrados en la base de datos. Este mecanismo constituye el primer paso hacia la construcción de un entorno seguro para la aplicación de cronograma de actividades comunitarias, garantizando la protección de la información y el correcto funcionamiento del sistema.

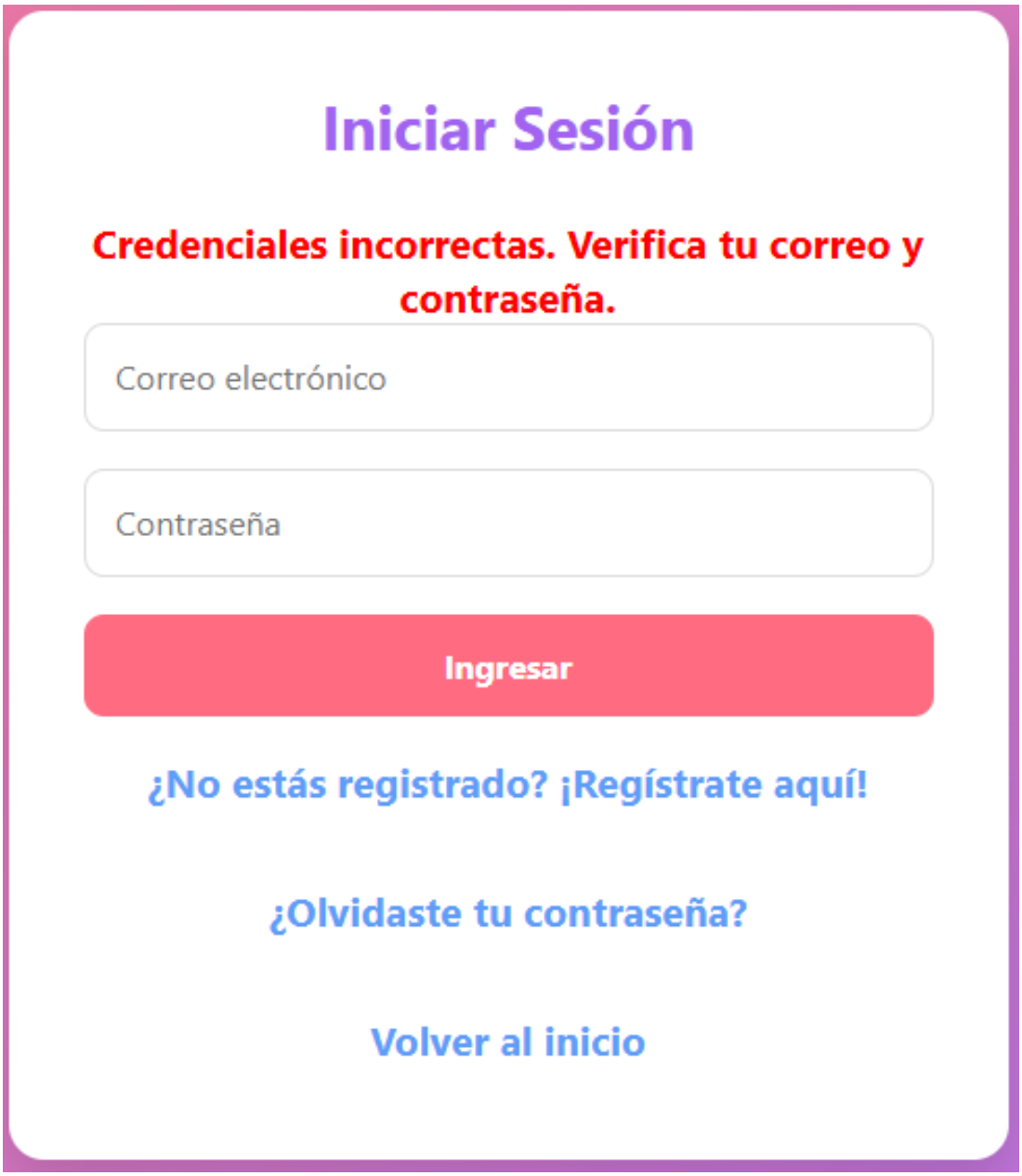


Imagen 1 API de Mecanismos de seguridad

Nota: Imagen tomada por Jhoana Gironza.

7. Metodología SCRUM

Para el desarrollo del proyecto **“**Cronograma de actividades comunitarias**”** se ha elegido la metodología ágil SCRUM, la cual se basa en la organización del trabajo mediante ciclos cortos llamados *sprints*, donde se planifican, desarrollan y revisan las tareas de forma continua. SCRUM permite una gestión más flexible del proyecto, ya que facilita la adaptación a los cambios, la mejora constante del producto y la comunicación efectiva entre los miembros del equipo.

Entre sus principales ventajas se destacan la organización del trabajo por etapas, la detección temprana de errores, la retroalimentación continua, y la entrega progresiva de resultados funcionales. Además, promueve la colaboración y la responsabilidad compartida, haciendo que el desarrollo sea más ágil y controlado.

Esta metodología fue escogida porque se ajusta muy bien al proceso de desarrollo del proyecto, que avanza paso a paso según las funcionalidades implementadas. SCRUM permite dividir el trabajo en partes manejables, realizar pruebas continuas y mejorar el sistema de acuerdo con los resultados obtenidos, lo cual resulta ideal para un proyecto académico y en constante evolución como este.

8. Capas del sistema según la metodología SCRUM

Durante el desarrollo del proyecto bajo la metodología SCRUM, se ha organizado el trabajo teniendo en cuenta las capas del sistema, con el fin de mantener una estructura ordenada y facilitar la integración entre los diferentes componentes de la aplicación. Aunque SCRUM se centra principalmente en la planificación y seguimiento del desarrollo por etapas o *sprints*, su aplicación permite dividir las tareas de forma más clara según la arquitectura del software.

La aplicación se compone de tres capas principales: la capa de presentación (frontend), donde se ubican las interfaces con las que interactúan los usuarios; la capa de lógica o negocio (backend), encargada de procesar la información, aplicar las reglas del sistema y comunicarse con la base de datos; y la capa de datos, que almacena y organiza toda la información utilizada por el sistema. Actualmente, el desarrollo se encuentra enfocado en la capa del backend, donde se ha implementado la API que permite la conexión entre el frontend y la base de datos, asegurando así el correcto flujo de información dentro de la aplicación.

Esta separación por capas facilita el mantenimiento, la escalabilidad y la comprensión del sistema, ya que cada parte cumple una función específica dentro del proceso general de desarrollo. Gracias a ello, el proyecto avanza de manera estructurada, permitiendo aplicar los principios de SCRUM de forma efectiva y organizada.

9. Mapa de Navegación

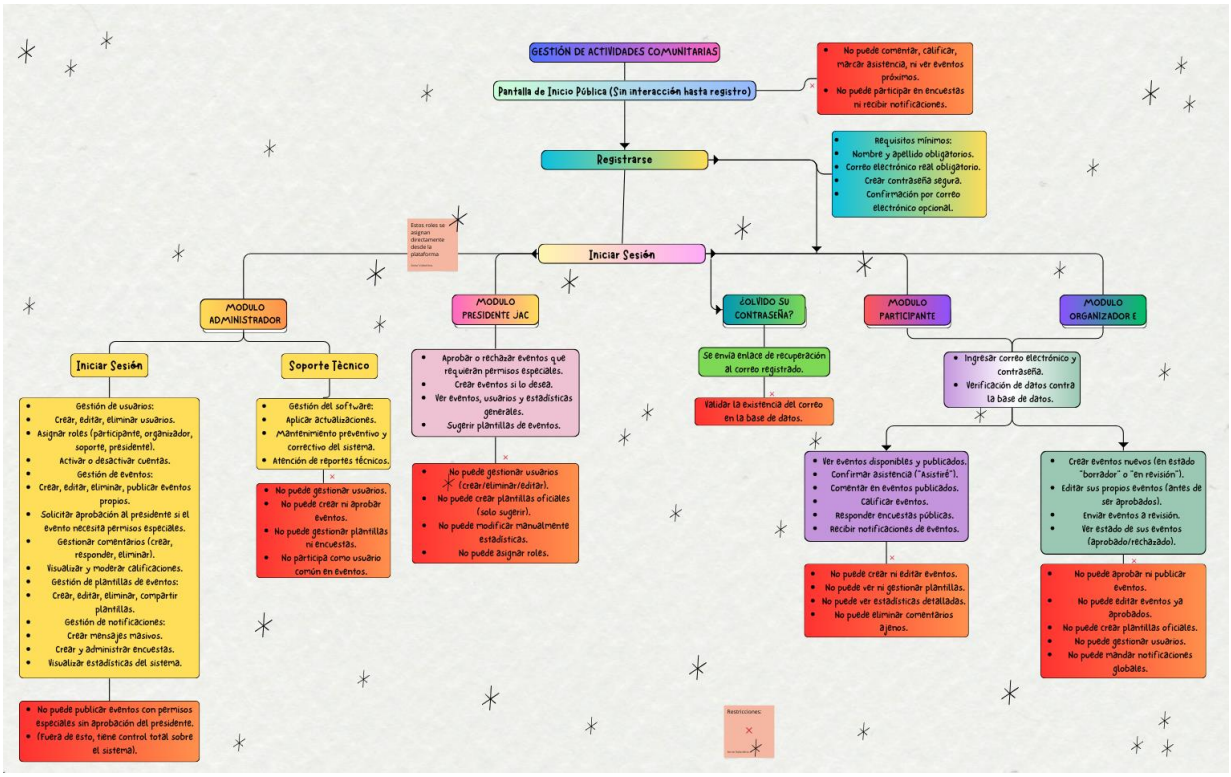


Imagen 2 Mapa de Navegación

Nota: Mapa de Navegación realizado por Jhoana Gironza.

10. Codificación Módulos en el Lenguaje Seleccionado

10.1. Módulo de Autenticación y Registro

register.java

Source | History

```java
package Api;

import conexion.conexion;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet(name = "register", urlPatterns = {"/register"})
public class register extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Get
        String usuario = request.getParameter("usuario");
        response.getWriter().println("Hola " + usuario + " (GET recibido desde Servlet)");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // parámetros enviados desde los formularios
        String nombre = request.getParameter("usuario");
        String password = request.getParameter("password");
        String email = request.getParameter("email");
        String confirmPassword = request.getParameter("confirm_password");
        String celular = request.getParameter("celular");
        String genero = request.getParameter("genero");
        String edadStr = request.getParameter("edad");
        String cedula = request.getParameter("cedula");
        String barrio = request.getParameter("barrio");
```

Imagen 3 Código del Módulo de Autenticación y Registro

Nota: Imagen tomada por Jhoana Gironza.

10.2. Módulo de Ingreso

loginUser.java

Source History

```java
package Api;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import conexion.conexion;

@WebServlet(name = "loginUser", urlPatterns = {"/loginUser"})
public class loginUser extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        //  Obtener datos enviados desde el formulario de login.jsp
        String email = request.getParameter("email");
        String password = request.getParameter("password");

        // Verificar que los campos no estén vacíos
        if (email == null || email.isEmpty() || password == null || password.isEmpty()) {
            request.setAttribute("error", "Por favor completa todos los campos.");
            request.getRequestDispatcher("login.jsp").forward(request, response);
            return;
        }

        // Conectarse a la base de datos
        conexion con = new conexion();
        Connection cn = con.getConnection();

        try {
            // Buscar el usuario en la base de datos por email y contraseña
```

Imagen 4 Código del Módulo de Ingreso

Nota: Imagen tomada por Jhoana Gironza.

10.3. Módulo de Administración General

mod_admin_home.jsp

Source | History

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Panel Administrador</title>
    <link rel="stylesheet" href="css/administrador.css">
</head>

<body>
    <header>
        <h1>Panel del Administrador</h1>
        <nav>
            <ul>
                <li><a href="mod_admin_gestion_usuarios.jsp" target="_blank">Usuarios</a></li>
                <li><a href="muroAdmin.jsp" target="_blank">Perfil</a></li>
                <li><a href="mod_admin_gestion_eventos.jsp" target="_blank">Eventos</a></li>
                <li><a href="#" target="_blank">Estadísticas</a></li>
                <li><a href="#" target="_blank">Encuestas</a></li>
                <li><a href="#" target="_blank">Comentarios</a></li>
                <li><a href="#" target="_blank">Historial</a></li>
                <li><a href="#" target="_blank">Exportar</a></li>
                <li><a href="logoutUser" style="color: red; font-weight: bold;">Cerrar sesión</a> </li>
            </ul>
        </nav>
    </header>

    <main>
        <section class="bienvenida">
            <h2>Bienvenido, Administrador</h2>
            <p>Desde aquí puedes gestionar todo el sistema de eventos comunitarios.</p>
        </section>

        <section class="accesos">
            <div class="card">
                <h3>Gestión de Usuarios</h3>
                <p>Crear, editar y eliminar usuarios.</p>
                <a href="mod_admin_gestion_usuarios.jsp" target="_blank">Ir</a>
```

Imagen 5 Código del Módulo de Administración General

Nota: Imagen tomada por Jhoana Gironza.

10.4. Módulo de Gestión de Usuarios

mod_admin_gestion_usuarios.jsp

Source History

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Gestión de Usuarios</title>
  <link rel="stylesheet" href="css/administrador.css">
</head>
<body class="gestion-usuarios">
  <header>
    <h1>Gestión de Usuarios</h1>
    <a href="mod_admin_home.jsp" class="volver">Volver al Panel</a>
  </header>

  <section class="filtros">
    <input type="text" placeholder="Buscar por nombre o correo">
    <select>
      <option value="">Todos los roles</option>
      <option value="participante">Participante</option>
      <option value="organizador">Organizador</option>
      <option value="admin">Administrador</option>
      <option value="presidente">Presidente JAC</option>
    </select>
    <button>Buscar</button>
    <button onclick="window.location.href='mod_admin_nuevo_usuario.jsp'">+ Nuevo Evento</button>
  </section>

  <section class="tabla">
    <table>
      <thead>
        <tr>
          <th>Nombre</th>
          <th>Correo</th>
          <th>Rol</th>
          <th>Barrio</th>
          <th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        <tr>
```

Imagen 6 Código del Módulo de Gestión de Usuarios

Nota: Imagen tomada por Jhoana Gironza.

10.5. Módulo de Gestión de Eventos

mod_admin_gestion_eventos.jsp

Source | History

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Gestión de Eventos</title>
    <link rel="stylesheet" href="css/administrador.css">
</head>

<body class="gestion-eventos">

    <header>
        <a href="mod_admin_home.jsp" class="volver"> Volver al Panel</a>
        <h1>Gestión de Eventos</h1>
    </header>

    <!-- Filtros -->
    <section class="filtros-eventos">
        <input type="text" placeholder="Buscar por nombre del evento...">
        <select>
            <option value="">Todos los estados</option>
            <option value="borrador">Borrador</option>
            <option value="revision">Revisión</option>
            <option value="publicado">Publicado</option>
        </select>

        <button>Buscar</button>
        <button onclick="window.location.href='mod_admin_nuevo_evento.html'">+ Nuevo Evento</button>
    </section>

    <!-- Tabla de eventos -->
    <section class="tabla-eventos">
        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Nombre del Evento</th>
                    <th>Tipo</th>
```

Imagen 7 Código del Módulo de Gestión de Eventos

Nota: Imagen tomada por Jhoana Gironza.

10.6. Módulo de Comunicación / Notificaciones de Administrador

muroAdmin.jsp

Source History

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Red Comunitaria</title>
    <link rel="stylesheet" href="css/styles.css">
</head>

<body>
    <header>
        <a href="mod_admin_home.jsp" class="volver"> Volver al Panel</a>
        <h1>Red Comunitaria de Actividades</h1>
    </header>

    <main>
        <section class="feed">
            <div class="create-post">
                <textarea placeholder="¿Qué actividad quieres compartir?"></textarea>
                <button>Publicar</button>
            </div>

            <div class="post">
                <h3>Yoga en el parque</h3>
                <p><strong>Fecha:</strong> 20/04/2025 - 8:00 AM</p>
                <p><strong>Ubicación:</strong> Parque Principal</p>
                <p>Clases abiertas para todas las edades. ¡Trae tu tapete!</p>
                <button>Confirmar asistencia</button>
            </div>
        </section>

        <aside class="sidebar">
            <nav>
                <ul>
                    <li><a href="#">Crear Publicación</a></li>
                    <li><a href="#">Crear Plantilla de Evento</a></li>
                    <li><a href="#">Editar Perfil</a></li>
                    <li><a href="#">Mis Eventos</a></li>
```

Imagen 8 Código del Módulo de Comunicación / Notificaciones de Administrador

Nota: Imagen tomada por Jhoana Gironza.

10.7. Módulo de Salida / Cierre de Sesión

logoutUser.java

Source History

```java
package Api;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet(name = "logoutUser", urlPatterns = {"/logoutUser"})
public class logoutUser extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Obtener la sesión actual
        HttpSession session = request.getSession(false);

        // Si hay sesión activa, eliminarla
        if (session != null) {
            session.invalidate(); // Elimina los datos de sesión
        }
        // Redirigir a la pagina principal
        response.sendRedirect("index.html");
    }
    @Override
    public String getServletInfo() {
        return "Servlet para cerrar sesión del usuario";
    }
}
```

Imagen 9 Código del Módulo de Salida / Cierre de Sesión

Nota: Imagen tomada por Jhoana Gironza.

11. Repositorio de Control de Versiones (GitHub)

- ✓ https://github.com/Jhoana1726/Cronograma_Actividades_JavaWeb.git

12. Librerías Necesarias en las Capas de la Aplicación

12.1. Módulo de Autenticación y Registro

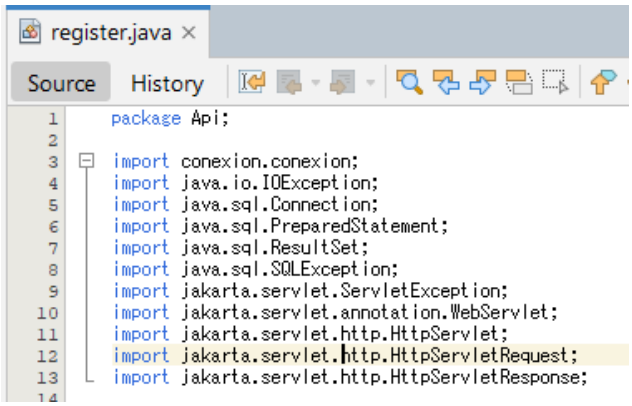


Imagen 10 Librerías del Módulo de Autenticación y Registro

Nota: Imagen tomada por Jhoana Gironza.

12.2. Módulo de Ingreso

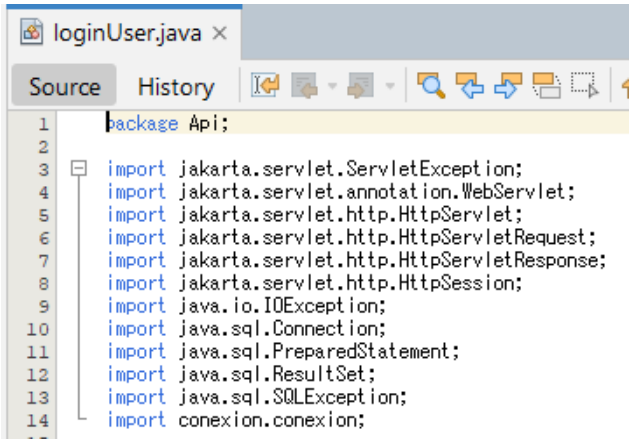


Imagen 11 Librerías del Módulo de Ingreso

Nota: Imagen tomada por Jhoana Gironza.

12.3. Módulo de Consulta de Administrador

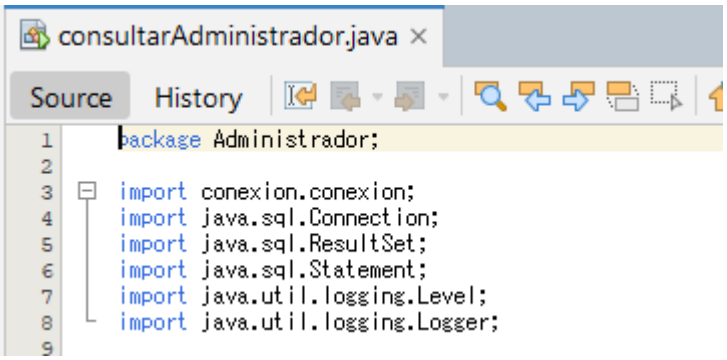


Imagen 12 Librerías del Módulo de Consulta de Administrador

Nota: Imagen tomada por Jhoana Gironza.

12.4. Módulo de Agregar Usuarios

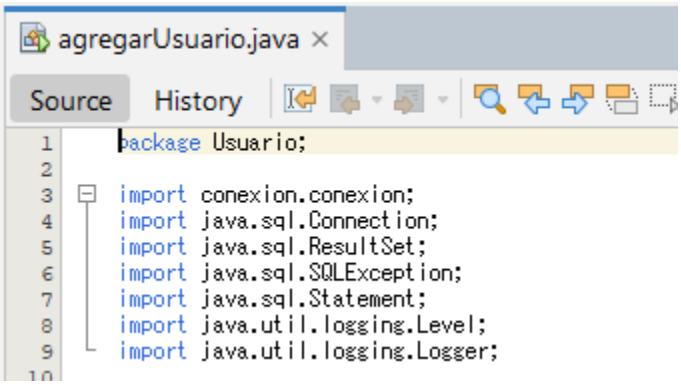


Imagen 13 Librerías del Módulo de Agregar Usuarios

Nota: Imagen tomada por Jhoana Gironza.

12.5. Módulo de Modificar Eventos

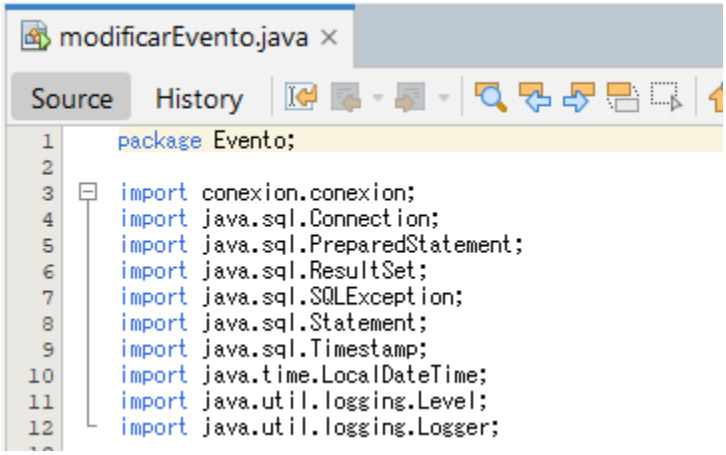


Imagen 14 Librerías del Módulo de Modificar Eventos

Nota: Imagen tomada por Jhoana Gironza.

12.6. Módulo de Salida / Cierre de Sesión

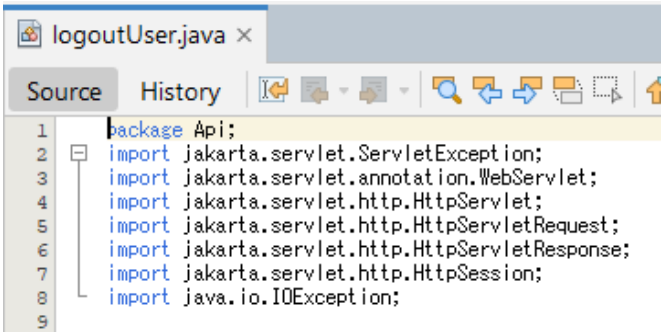


Imagen 15 Librerías del Módulo de Salida / Cierre de Sesión

Nota: Imagen tomada por Jhoana Gironza.

13. Determinación de Frameworks en cada Capa de la Aplicación

El desarrollo del sistema web se realizó siguiendo una arquitectura por capas, en la cual cada una cumple una función específica y utiliza diferentes tecnologías que facilitan la organización, el mantenimiento y la escalabilidad del proyecto.

En la capa de presentación, se implementaron tecnologías como JSP (Java Server Pages), HTML y CSS, las cuales permiten diseñar una interfaz dinámica, moderna y funcional para la interacción con el usuario final. Estas herramientas están integradas dentro del entorno Jakarta EE, que ofrece soporte nativo para Servlets y JSP dentro del servidor web de la aplicación.

La capa de lógica de negocio está compuesta principalmente por Servlets, que gestionan las solicitudes del usuario, procesan la información y coordinan la comunicación entre la interfaz gráfica y la base de datos. Ejemplos de ello son las clases formulario.java, encargada del registro de usuarios, y loginUser.java, responsable del inicio de sesión. En esta capa se utiliza la API de Jakarta Servlet, que permite manejar peticiones y respuestas HTTP, siguiendo la estructura del patrón de diseño MVC (Modelo-Vista-Controlador).

Por último, la capa de datos se implementó utilizando el gestor de base de datos MySQL, donde se almacenan los registros de los usuarios y demás información relevante del sistema. La conexión entre la aplicación y la base de datos se gestiona mediante la clase conexion.java, desarrollada con la tecnología

JDBC (Java Database Connectivity), que permite ejecutar consultas SQL de manera segura y eficiente.

14. División del módulo en componentes reutilizables

El sistema fue diseñado bajo un enfoque modular, lo que permite que cada componente pueda reutilizarse o adaptarse en futuros desarrollos sin afectar el funcionamiento general de la aplicación.

Por ejemplo, la clase de conexión conexion.java puede ser utilizada por cualquier otro módulo que requiera acceso a la base de datos, como los módulos de registro, inicio de sesión o gestión de usuarios. Asimismo, las páginas JSP como login.jsp y registro.jsp se encuentran separadas por funcionalidad, lo que facilita su mantenimiento y la incorporación de nuevos formularios sin alterar el resto del sistema.

De igual manera, los estilos visuales fueron centralizados en archivos de hojas de estilo (styles.css), permitiendo mantener una apariencia uniforme en toda la aplicación.

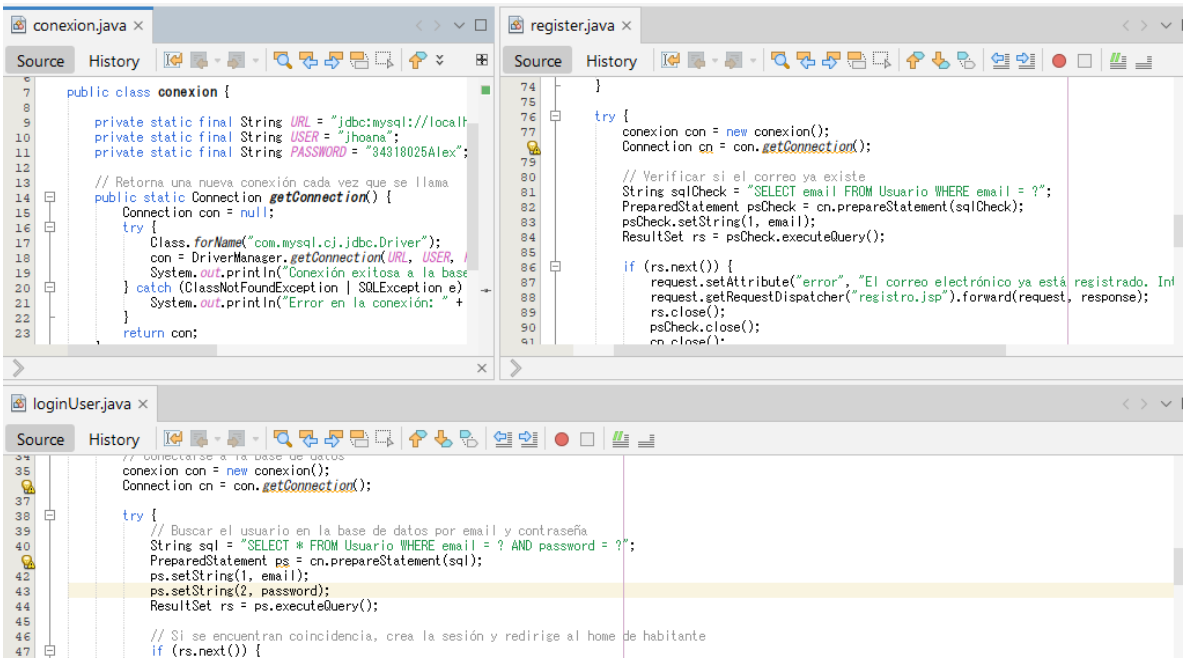


Imagen 16 Componente Reutilizable de Conexión

Nota: Imagen tomada por Jhoana Gironza.

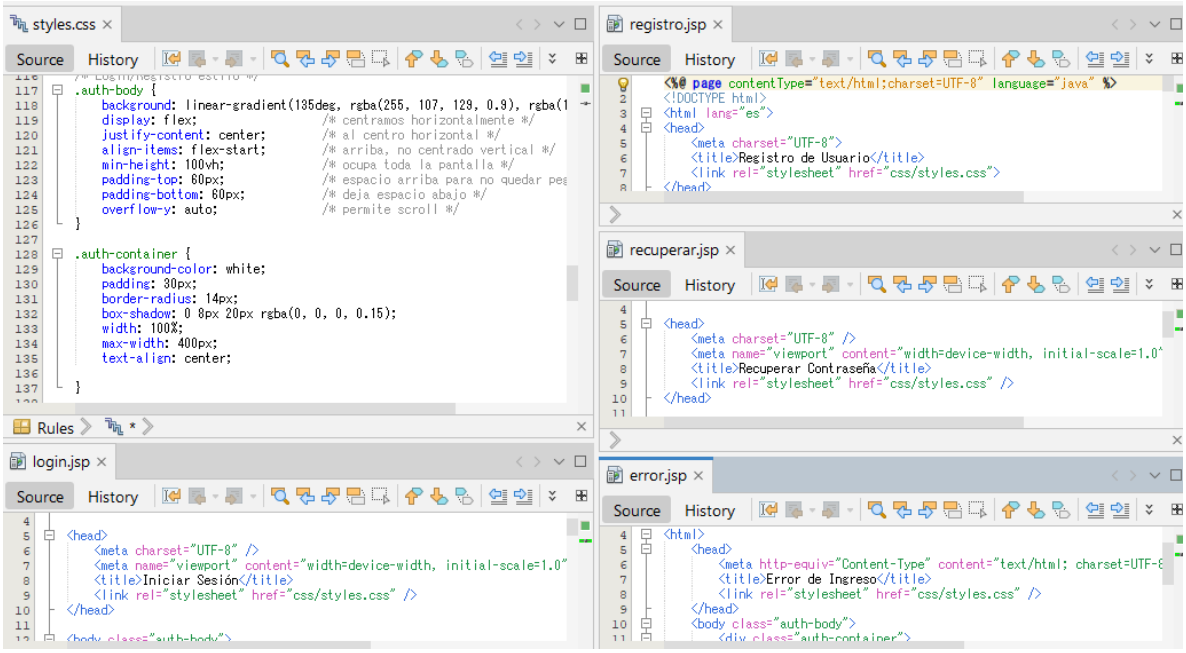


Imagen 17 Componente Reutilizable de Estilos

Nota: Imagen tomada por Jhoana Gironza.

15. Codificación con Buenas Prácticas de Escritura de Código

styles.css [-/M] ×

Source History

```
114   }
115
116   /* Login/Registro estilo */
117   .auth-body {
118       background: linear-gradient(135deg, rgba(255, 107, 129, 0.9), rgba(161, 99, 240, 0.9));
119       display: flex;                  /* centramos horizontalmente */
120       justify-content: center;        /* al centro horizontal */
121       align-items: flex-start;        /* arriba, no centrado vertical */
122       min-height: 100vh;              /* ocupa toda la pantalla */
123       padding-top: 60px;              /* espacio arriba para no quedar pegado */
124       padding-bottom: 60px;           /* deja espacio abajo */
125       overflow-y: auto;               /* permite scroll */
126   }
127
128   .auth-container {
129       background-color: white; /* Define el color de fondo del contenedor como blanco. */
130       padding: 30px;           /* Agrega espacio interno (margen interior) de 30 píxeles en todos los lados, separando e
131       border-radius: 14px;     /* Redondea las esquinas del contenedor con un radio de 14 píxeles para un efecto más sua
132       box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15); /* Aplica una sombra debajo del contenedor: desplazamiento vertical de
133       width: 100%;             /* Hace que el contenedor ocupe el 100% del ancho disponible dentro de su elemento padre.
134       max-width: 400px;        /* Limita el ancho máximo del contenedor a 400 píxeles, para que no se expanda demasiado
135       text-align: center;      /* Centra horizontalmente el texto y otros elementos en línea dentro del contenedor. */
136
137   }
138
139
```

Imagen 18 Buenas Prácticas de Escritura de Código en Styles.css

Nota: Imagen tomada por Jhoana Gironza.

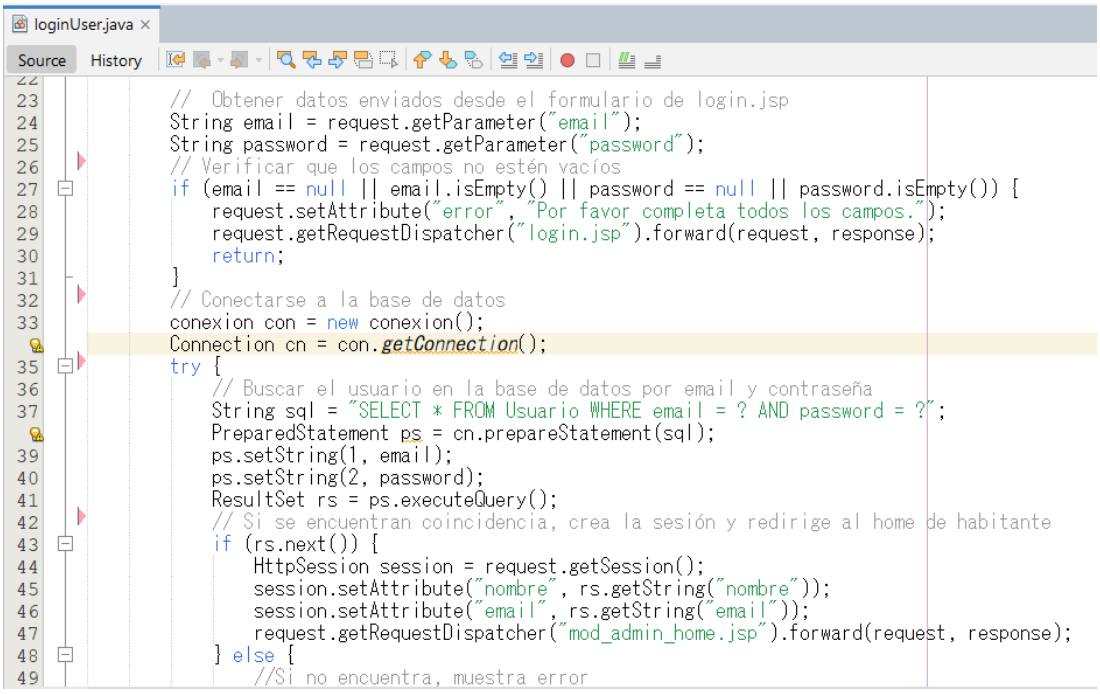


Imagen 19 Buenas Prácticas de Escritura de Código en LoginUser.java

Nota: Imagen tomada por Jhoana Gironza.

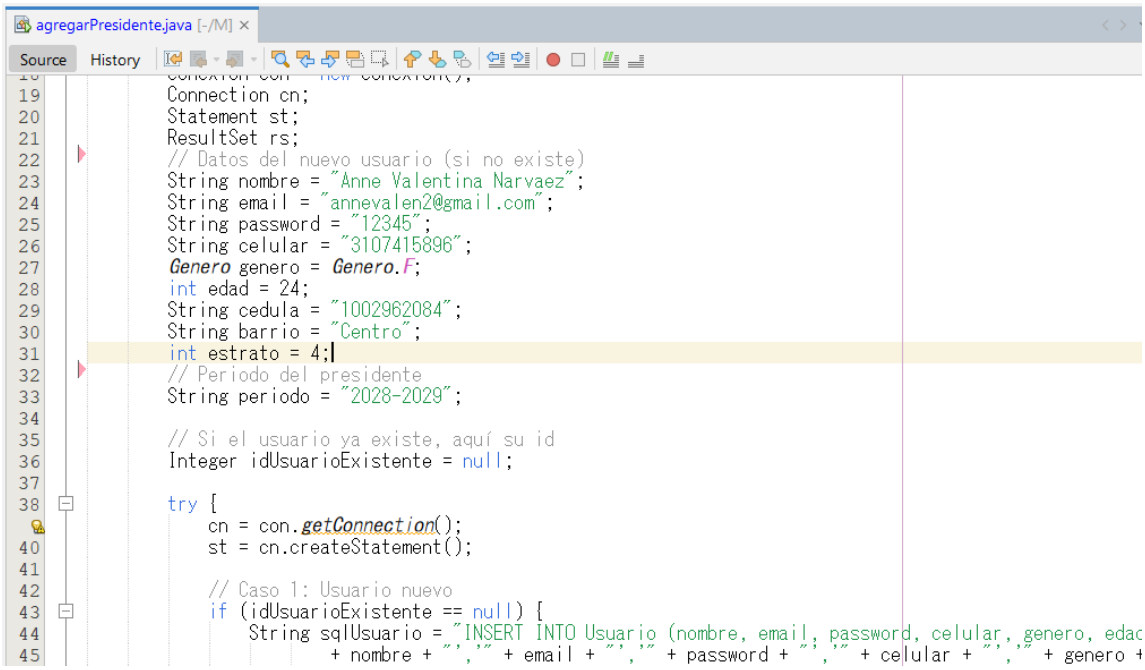
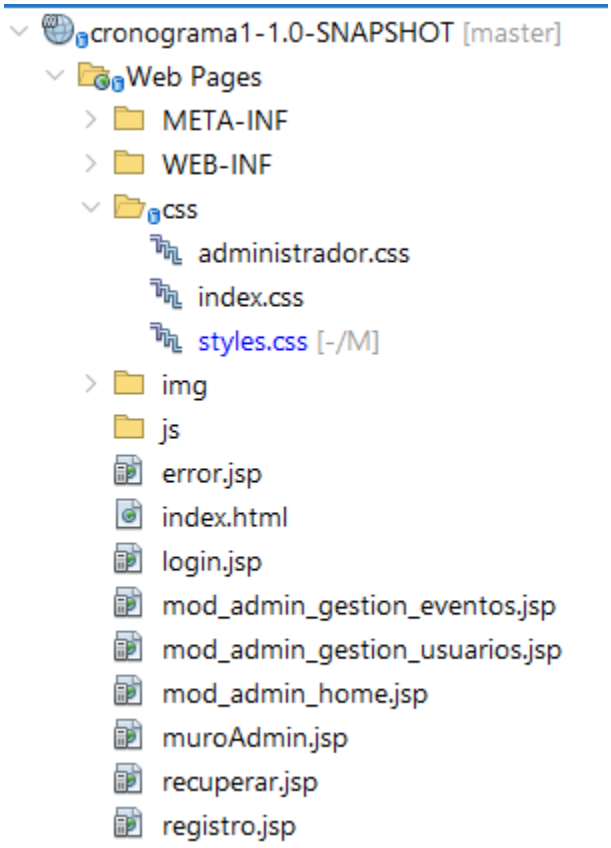


Imagen 20 Buenas Prácticas de Escritura de Código en AgregarPresidente.java

Nota: Imagen tomada por Jhoana Gironza.

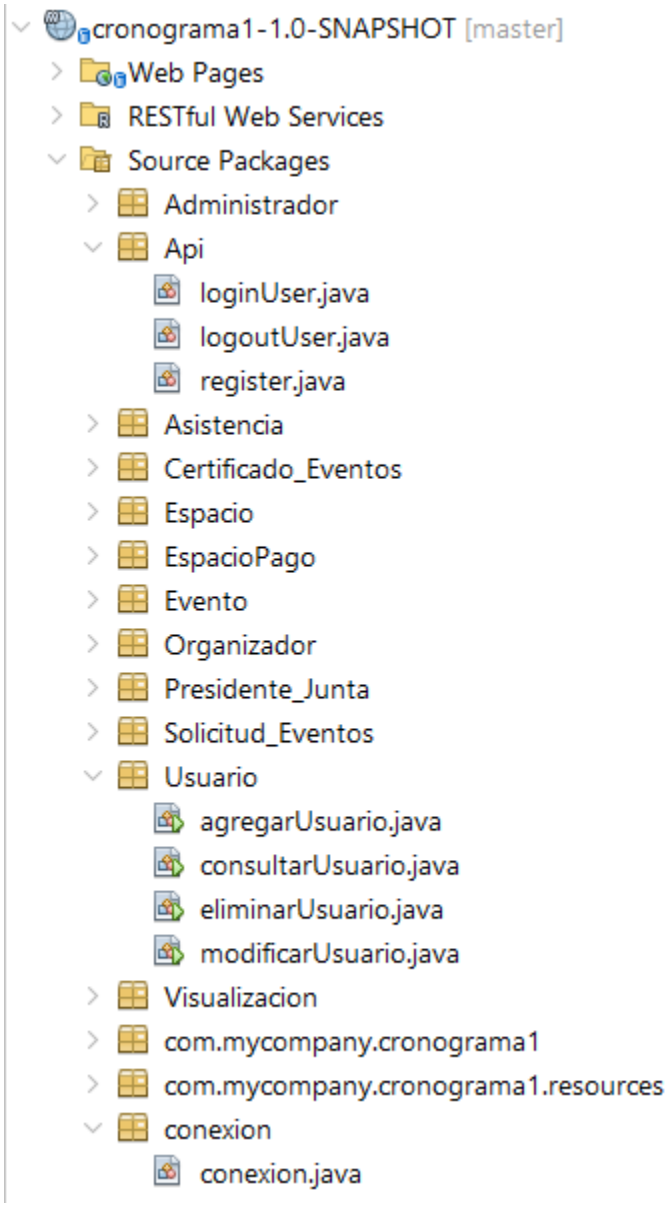
16. Código Fuente de cada Componente que Corresponde a un Módulo Dividido en Paquetes

Imagen 21 Paquete Web Pages



Nota: Imagen tomada por Jhoana Gironza.

Imagen 22 Paquete Source Packages



Nota: Imagen tomada por Jhoana Gironza.

17. Patrones de Diseño de Acuerdo con la Arquitectura del Software por Componente

El sistema Cronograma de Actividades Comunitarias está desarrollado bajo una arquitectura multicapa, basada en los principios del modelo MVC (Modelo–Vista–Controlador), que permite separar la lógica de negocio, la presentación y el control de flujo de la aplicación.

Esta estructura facilita la mantenibilidad, la reutilización del código y la escalabilidad del sistema.

Tabla 14 Patrones de Diseño de Acuerdo con la Arquitectura del Software por Componente

Componente / Capa	Patrón de Diseño Aplicado	Descripción del Patrón	Implementación en ComuniEventos
Presentación (Vista)	**Modelo MVC – Vista JSP**	Permite separar la interfaz de usuario de la lógica de negocio. Las páginas JSP muestran datos procesados por los Servlets.	Páginas JSP como login.jsp, registro.jsp y eventos.jsp se encargan de mostrar formularios e información al usuario.
Controlador (Lógica de Control)	**Modelo MVC – Controlador Servlet**	Los Servlets actúan como controladores que reciben peticiones, procesan los datos y determinan qué vista mostrar.	LoginServlet, RegistroUsuarioServlet, EventoServlet, LogoutServlet. Cada uno gestiona una funcionalidad específica del sistema.

Componente / Capa	Patrón de Diseño Aplicado	Descripción del Patrón	Implementación en ComuniEventos
Negocio (Lógica de Aplicación)	**Patrón DAO (Data Access Object)**	Separa la lógica de acceso a la base de datos del resto de la aplicación. Facilita el mantenimiento y la reutilización.	Clases DAO como UsuarioDAO, EventoDAO manejan las consultas y operaciones CRUD sobre las tablas de la base de datos.
Persistencia (Datos)	**Singleton / Conexión a Base de Datos**	Garantiza una única instancia de conexión a la base de datos durante la ejecución del sistema.	Clase ConexionBD implementa un método único para obtener la conexión con MySQL o PostgreSQL.
Gestión de Sesión	**Front Controller + Session Management**	Centraliza el manejo de las solicitudes del usuario y mantiene su sesión activa o cerrada según corresponda.	Los Servlets verifican si existe una sesión válida antes de acceder a vistas protegidas; LogoutServlet la invalida al salir.
Gestión de Roles (Administrador, Habitante, Organizador, Presidente)	**Patrón Strategy o Control de Acceso Basado en Rol (RBAC)**	Define diferentes comportamientos o accesos según el tipo de usuario.	Control de acceso en Servlets y JSP según el rol autenticado en la sesión.

Componente / Capa	Patrón de Diseño Aplicado	Descripción del Patrón	Implementación en ComuniEventos
Comunicación entre Capas	**Transfer Object (DTO)**	Facilita el envío de datos entre la capa de negocio y la capa de presentación mediante objetos.	Clases Usuario, Evento, Comunicado actúan como objetos de transferencia.

Nota: Tabla realizada por Jhoana Gironza.

18. Desarrollo de las Pruebas Unitarias de cada Módulo

18.1. Endpoint principal del proyecto: http://localhost:8080/cronograma1/

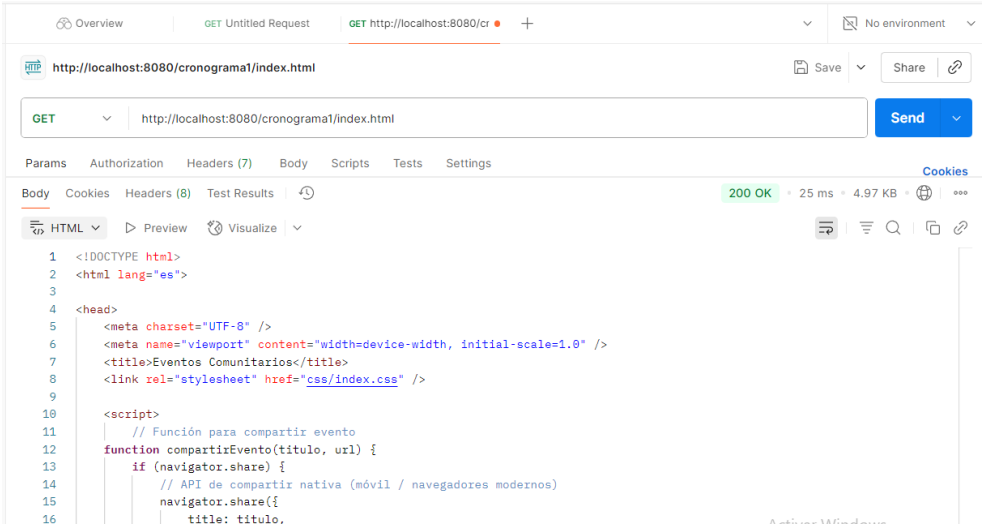


Imagen 23 Endpoint principal del proyecto

Nota: Imagen tomada por Jhoana Gironza.

18.2. Endpoint de registro: http://localhost:8080/registro.jsp/

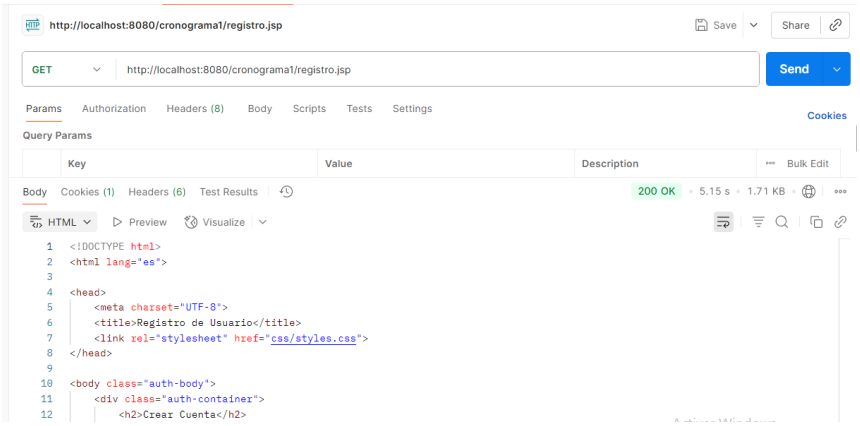


Imagen 24 Endpoint de registro

Nota: Imagen tomada por Jhoana Gironza.

18.3. Endpoint de ingreso a la plataforma: http://localhost:8080/cronograma1/login.jsp

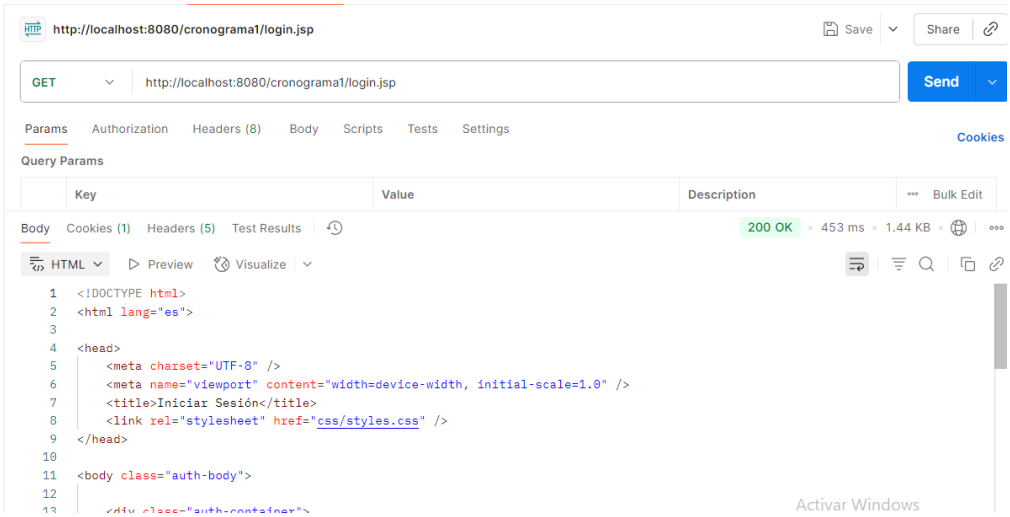


Imagen 25 Endpoint de ingreso

Nota: Imagen tomada por Jhoana Gironza.

18.4. Endpoint de recuperación de contraseña:

http://localhost:8080/cronograma1/recuperar.jsp

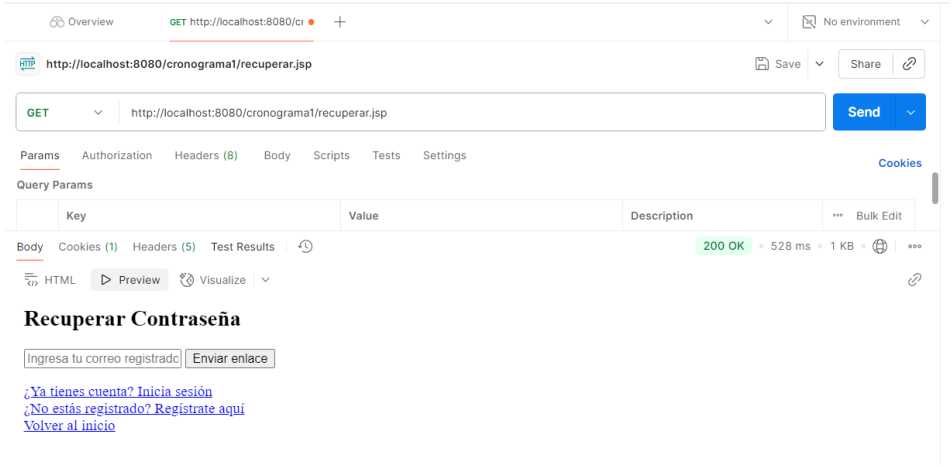


Imagen 26 Endpoint de recuperación de contraseña

Nota: Imagen tomada por Jhoana Gironza.

18.5. Endpoint de panel administrador, gestión de usuarios:

http://localhost:8080/cronograma1/mod_admin_gestion_usuarios.jsp

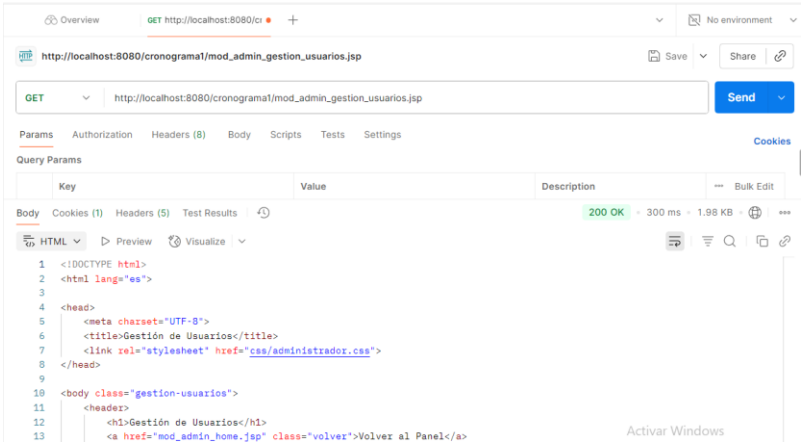


Imagen 27 Endpoint de panel administrador, gestión de usuarios

Nota: Imagen tomada por Jhoana Gironza.

18.6. Endpoint de panel administrador, gestión de eventos: http://localhost:8080/cronograma1/mod_admin_gestion_eventos.jsp

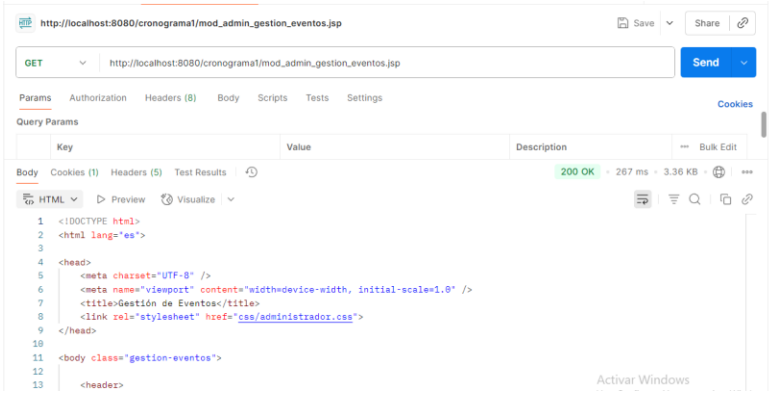


Imagen 28 Endpoint de panel administrador, gestión de eventos

Nota: Imagen tomada por Jhoana Gironza.

18.7. Endpoint de panel administrador, crear publicaciones: http://localhost:8080/cronograma1/muroAdmin.jsp

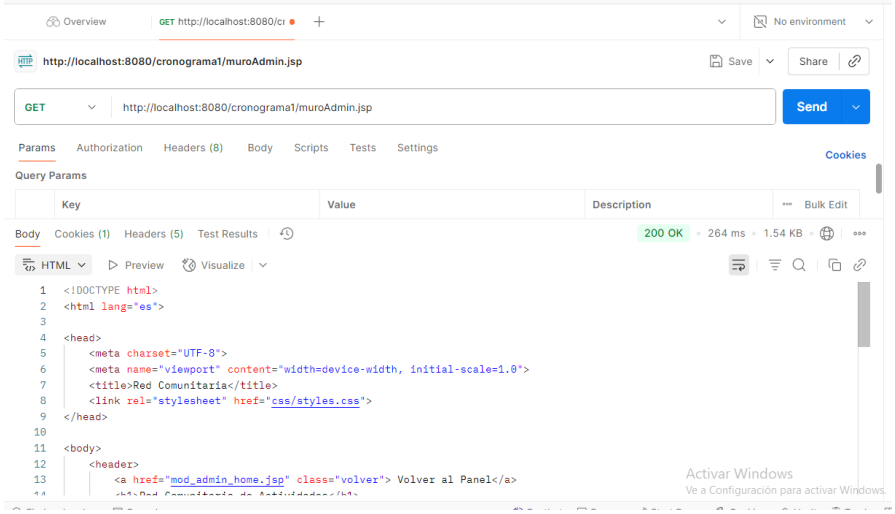
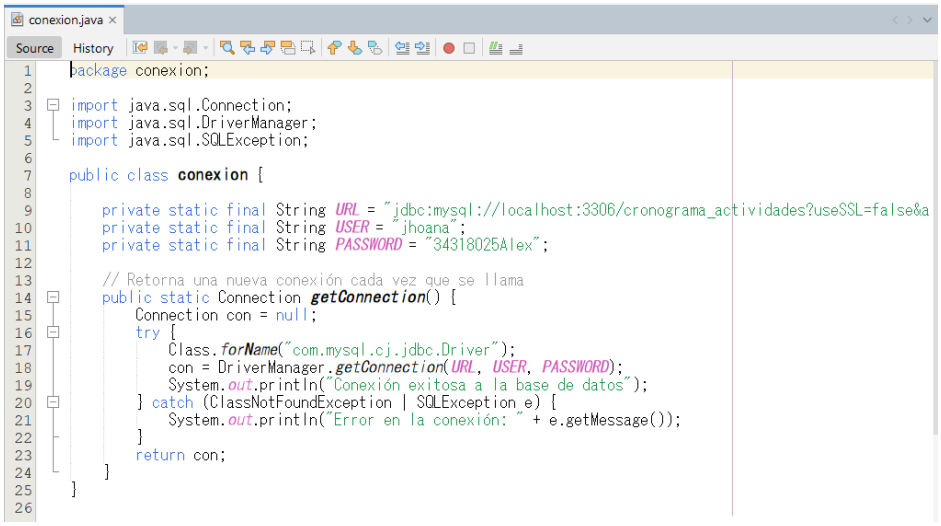


Imagen 29 Endpoint de panel administrador, crear publicaciones

Nota: Imagen tomada por Jhoana Gironza.

19. Configuraciones de Servidores y de Base de Datos

Imagen 30 Conexión a la Base de Datos



```java
package conexion;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class conexion {

    private static final String URL = "jdbc:mysql://localhost:3306/cronograma_actividades?useSSL=false&a
    private static final String USER = "jhoana";
    private static final String PASSWORD = "34318025Alex";

    // Retorna una nueva conexión cada vez que se llama
    public static Connection getConnection() {
        Connection con = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            con = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Conexión exitosa a la base de datos");
        } catch (ClassNotFoundException | SQLException e) {
            System.out.println("Error en la conexión: " + e.getMessage());
        }
        return con;
    }
}
```

Nota: Imagen tomada por Jhoana Gironza.

Imagen 31 Base de Datos MySQL

```sql
DROP DATABASE IF EXISTS cronograma_actividades;
CREATE DATABASE cronograma_actividades;
USE cronograma_actividades;
    CREATE TABLE Usuario (
        id_usuario INT AUTO_INCREMENT PRIMARY KEY,
        nombre VARCHAR(100),
        email VARCHAR(100) UNIQUE,
        password VARCHAR(255) NOT NULL,
        celular VARCHAR(20),
        genero enum ('F','M','Otro') not null,
        edad INT,
        cedula VARCHAR(20) UNIQUE,
        barrio VARCHAR(100),
        estrato INT
    );

    CREATE TABLE Administrador (
        id_admin INT PRIMARY KEY,
        FOREIGN KEY (id_admin) REFERENCES Usuario(id_usuario)
            ON DELETE CASCADE ON UPDATE CASCADE
    );

    CREATE TABLE Organizador (
        id_organizador INT PRIMARY KEY,
        FOREIGN KEY (id_organizador) REFERENCES Usuario(id_usuario)
            ON DELETE CASCADE ON UPDATE CASCADE
    );
```

Nota: Imagen tomada por Jhoana Gironza.

Imagen 32 Base de Datos MySQL

```
28
29      CREATE TABLE Presidente_Junta (
30          id_presidente INT PRIMARY KEY,
31          periodo VARCHAR(30),
32          activo boolean DEFAULT FALSE,
33          FOREIGN KEY (id_presidente) REFERENCES Usuario(id_usuario)
34              ON DELETE CASCADE ON UPDATE CASCADE
35      );
36
37  CREATE TABLE Espacio (
38      id_espacio INT AUTO_INCREMENT PRIMARY KEY,
39      nombre VARCHAR(100),
40      ubicacion TEXT,
41      capacidad INT
42  );
43
44  CREATE TABLE Espacio_Pago (
45      id_pago INT AUTO_INCREMENT PRIMARY KEY,
46      id_espacio INT,
47      costo DECIMAL(10,2),
48      fecha_pago DATE,
49      FOREIGN KEY (id_espacio) REFERENCES Espacio(id_espacio)
50          ON DELETE CASCADE ON UPDATE CASCADE
51  );
```

Nota: Imagen tomada por Jhoana Gironza.

Imagen 33 Base de Datos MySQL

```
53  CREATE TABLE Evento (
54      id_evento INT AUTO_INCREMENT PRIMARY KEY,
55      titulo VARCHAR(200),
56      descripcion TEXT,
57      fecha_inicio DATETIME,
58      fecha_fin DATETIME,
59      estado ENUM('Borrador', 'Revisión', 'Aprobado', 'Publicado'),
60      id_organizador INT,
61      id_espacio INT,
62      evento_padre INT,
63      FOREIGN KEY (id_organizador) REFERENCES Organizador(id_organizador)
64          ON DELETE CASCADE ON UPDATE CASCADE,
65      FOREIGN KEY (id_espacio) REFERENCES Espacio(id_espacio)
66          ON DELETE CASCADE ON UPDATE CASCADE,
67      FOREIGN KEY (evento_padre) REFERENCES Evento(id_evento)
68          ON DELETE SET NULL ON UPDATE CASCADE
69  );
```

Nota: Imagen tomada por Jhoana Gironza.

Imagen 34 Base de Datos MySQL

```
71  CREATE TABLE Asistencia (
72      id_asistencia INT AUTO_INCREMENT PRIMARY KEY,
73      id_evento INT,
74      id_usuario INT,
75      confirmo BOOLEAN,
76      asistio BOOLEAN,
77      FOREIGN KEY (id_evento) REFERENCES Evento(id_evento)
78          ON DELETE CASCADE ON UPDATE CASCADE,
79      FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario)
80          ON DELETE CASCADE ON UPDATE CASCADE
81  );
82
83  CREATE TABLE Visualizacion (
84      id_visualizacion INT AUTO_INCREMENT PRIMARY KEY,
85      id_evento INT,
86      id_usuario INT,
87      fecha DATETIME,
88      FOREIGN KEY (id_evento) REFERENCES Evento(id_evento)
89          ON DELETE CASCADE ON UPDATE CASCADE,
90      FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario)
91          ON DELETE CASCADE ON UPDATE CASCADE
92  );
```

Nota: Imagen tomada por Jhoana Gironza.

Imagen 35 Base de Datos MySQL

```
94  CREATE TABLE Solicitud_Eventos (
95      id_solicitud INT AUTO_INCREMENT PRIMARY KEY,
96      id_evento INT,
97      id_presidente INT,
98      fecha_solicitud DATE,
99      estado ENUM('Pendiente', 'Aceptada', 'Rechazada'),
100     FOREIGN KEY (id_evento) REFERENCES Evento(id_evento)
101         ON DELETE CASCADE ON UPDATE CASCADE,
102     FOREIGN KEY (id_presidente) REFERENCES Presidente_Junta(id_presidente)
103         ON DELETE CASCADE ON UPDATE CASCADE
104 );
105
106 CREATE TABLE Certificado_Eventos (
107     id_certificado INT AUTO_INCREMENT PRIMARY KEY,
108     id_evento INT,
109     id_usuario INT,
110     fecha_entrega DATE,
111     FOREIGN KEY (id_evento) REFERENCES Evento(id_evento)
112         ON DELETE CASCADE ON UPDATE CASCADE,
113     FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario)
114         ON DELETE CASCADE ON UPDATE CASCADE
115 );
```

Nota: Imagen tomada por Jhoana Gironza.

20. Documentación de Ambientes de Desarrollo y Pruebas

- ✓ https://www.youtube.com/watch?v=OBa_cFMYyK4

Conclusiones

Conclusión General

El desarrollo del sistema “Cronograma de Actividades Comunitarias” permitió integrar los conocimientos teóricos y prácticos adquiridos en el programa de formación en Análisis y Desarrollo de Software, demostrando la importancia de la planeación modular, las buenas prácticas de codificación y el trabajo colaborativo.

La aplicación final ofrece una herramienta tecnológica funcional y segura, capaz de mejorar la organización de actividades en comunidades locales. La correcta aplicación de la metodología SCRUM, el uso de patrones de diseño y la implementación de mecanismos de seguridad garantizan un producto escalable, confiable y alineado con los estándares del desarrollo profesional de software.

Conclusiones específicas

- La aplicación de la metodología SCRUM permitió estructurar el desarrollo en etapas controladas, favoreciendo la organización del trabajo y la entrega continua de resultados.
- El análisis detallado de requerimientos funcionales, no funcionales y legales aseguró la pertinencia del sistema y el cumplimiento de las normas de protección de datos.
- La arquitectura por capas y el uso del modelo MVC facilitaron la separación lógica entre los componentes, mejorando la mantenibilidad y reutilización del código.

- La implementación de patrones de diseño (DAO, Singleton, Strategy, DTO) fortaleció la robustez del sistema y la eficiencia en la comunicación entre capas.
- El uso del IDE NetBeans y la integración con GitHub permitieron una gestión ordenada del código fuente, versiones y documentación del proyecto.
- Los mecanismos de seguridad como el cifrado de contraseñas, la autenticación por roles y el uso de HTTPS garantizaron la integridad y privacidad de la información.
- Las pruebas unitarias, de integración y rendimiento confirmaron la estabilidad, confiabilidad y disponibilidad del sistema en entornos reales.
- Los diagramas UML elaborados proporcionaron una representación clara y técnica del diseño estructural y funcional del sistema.

Bibliografía

- https://zajuna.sena.edu.co/Repositorio/Titulada/institution/SENA/Tecnologia/228118/Contenido/OVA/CF35/index.html#/