Documentación Técnica del Proyecto SICUE

Ingeniería del Software

Pablo Valencia
Marcos Cáceres García

Índice

1. Introducción

1.1. Propósito del proyecto

El propósito principal del proyecto SICUE es crear un sistema de gestión automatizado que permita optimizar y digitalizar los procesos asociados al programa de intercambio académico SICUE (Sistema de Intercambio entre Centros Universitarios Españoles). Este programa facilita a los estudiantes la posibilidad de cursar asignaturas en universidades diferentes a la suya, garantizando el reconocimiento de los créditos obtenidos.

La implementación del sistema tiene como objetivo abordar los siguientes problemas clave detectados en los procesos actuales:

- **Automatización de la gestión**: El sistema sustituirá los métodos manuales y basados en papel, reduciendo errores humanos, evitando la pérdida de documentos y agilizando la tramitación de inscripciones, anulaciones y gestión de convenios.
- **Optimización del manejo de información**: Facilitará la búsqueda, registro y actualización de datos relacionados con estudiantes, profesores y convenios interuniversitarios, minimizando los tiempos de respuesta y aumentando la eficiencia operativa.
- **Seguridad de los datos**: Debido a la naturaleza sensible de la información personal gestionada (datos de estudiantes y profesores), el sistema garantizará altos estándares de seguridad, incluyendo cifrado de datos y control de acceso basado en roles.
- **Cumplimiento normativo**: El sistema está diseñado para cumplir con las normativas legales vigentes en cuanto a protección de datos, como el Reglamento General de Protección de Datos (RGPD), asegurando la privacidad y confidencialidad de los usuarios.

En términos de resultados, el sistema se plantea como una solución integral que:

1. Mejora la experiencia del usuario final (estudiantes, profesores y administradores) al ofrecer una interfaz intuitiva y funcionalidades claramente definidas.
2. Aumenta la productividad de los administradores al centralizar la información y automatizar procesos rutinarios.
3. Establece un marco sólido y escalable para la integración futura con plataformas web alojadas en servidores institucionales, asegurando su adaptabilidad y sostenibilidad a largo plazo.

El sistema se implementará inicialmente en un entorno local, con planes futuros para su despliegue en los servidores de la Universidad de Córdoba (UCO), lo que permitirá una mayor accesibilidad y uso distribuido por parte de las universidades asociadas.

1.2. Alcance

El proyecto SICUE está diseñado para abordar las necesidades de gestión y automatización del programa de intercambio académico SICUE. El sistema cubrirá los siguientes aspectos:

Alcance funcional

1. **Gestión de inscripciones y anulaciones**:

- Permitir a los estudiantes registrarse en planes de convalidación de materias y anular sus inscripciones dentro de los periodos establecidos.
- Garantizar que los estudiantes no puedan inscribirse nuevamente si ya participaron en el programa.

2. **Roles y permisos**:
   - Asignar roles específicos (administrador, profesor, estudiante) con permisos adaptados a sus funciones.
   - Asegurar que cada usuario tenga acceso solo a la información y funcionalidades necesarias.
3. **Gestión de convenios y convalidaciones**:
   - Administrar los acuerdos entre universidades, permitiendo modificar los términos y gestionar las asignaturas convalidadas.
   - Cruzar datos de estudiantes y profesores para facilitar la coordinación interuniversitaria.
4. **Copias de seguridad y seguridad de datos**:
   - Implementar un sistema automático de copias de seguridad para prevenir pérdida de información.
   - Cifrar y proteger los datos personales de estudiantes y profesores según el RGPD.
5. **Generación de reportes**:
   - Producir informes detallados sobre estudiantes inscritos, planes de convalidación y participación, exportables en formatos como PDF o Excel.
6. **Configuración de periodos**:
   - Establecer rangos de fechas para la inscripción y anulación, asegurando que las operaciones estén disponibles solo en los periodos definidos.

Alcance no funcional

1. **Seguridad**:
   - Garantizar el almacenamiento cifrado de todos los datos sensibles.
   - Implementar autenticación de usuarios basada en credenciales únicas.
2. **Compatibilidad y escalabilidad**:
   - Diseñar el sistema para operar inicialmente en local y garantizar su migración futura a servidores web institucionales.
3. **Facilidad de uso**:
   - Proveer una interfaz gráfica simple e intuitiva para facilitar la interacción de los usuarios.

4. **Normativa**:
    - Cumplir con las regulaciones de protección de datos personales y confidencialidad establecidas en el RGPD.

Exclusiones

1. **Integración con sistemas externos**:
    - El sistema no incluye en esta fase la integración con plataformas externas distintas de las universidades asociadas al programa SICUE.
2. **Soporte multilenguaje**:
    - El sistema está diseñado inicialmente para operar en un único idioma (español).
3. **Funcionalidades avanzadas de análisis de datos**:
    - Aunque se generan reportes detallados, no se contempla la integración de herramientas avanzadas de análisis o visualización de datos.

El alcance definido establece una base sólida para que el sistema cumpla con los requisitos fundamentales del programa SICUE, garantizando la posibilidad de futuras expansiones según las necesidades del cliente o las instituciones asociadas.

1.3. Equipos de trabajo

El desarrollo del proyecto SICUE fue realizado por un equipo compuesto por dos integrantes que trabajaron de forma coordinada a lo largo de todas las fases del ciclo de vida del proyecto. Su enfoque colaborativo y su alternancia en roles clave garantizaron un desarrollo fluido y organizado, maximizando la eficiencia y la calidad de los entregables.

Miembros del equipo

1. **Marcos Cáceres García**
    - **Rol principal:** Desarrollador
      Marcos asumió la responsabilidad de implementar funcionalidades clave del sistema, trabajando en el código fuente y asegurando que las especificaciones funcionales se tradujeran en componentes sólidos y eficientes. Además, participó en la configuración de la arquitectura general del proyecto.
    - **Rol adicional:** Scrum Master (en semanas alternas)
      Durante las semanas en que fungió como Scrum Master, Marcos se encargó de liderar las ceremonias de Scrum (planificaciones, retrospectivas y revisiones de Sprint), facilitando las reuniones diarias y priorizando las tareas del backlog. También resolvió los bloqueos técnicos y ayudó a mantener el ritmo de trabajo del equipo.
2. **Pablo Valencia Palomino**
    - **Rol principal:** Diseñador y desarrollador
      Pablo lideró el diseño del sistema, generando diagramas de clases y secuencia, definiendo la estructura de la base de datos y estableciendo las relaciones entre

los diferentes componentes. Adicionalmente, trabajó en el desarrollo del sistema, asegurando que las funcionalidades implementadas cumplieran con los requisitos definidos.

- **Rol adicional:** Scrum Master (en semanas alternas)
  Como Scrum Master, Pablo organizó las tareas semanales y realizó un seguimiento exhaustivo de los objetivos planteados. Supervisó el cumplimiento de los tiempos y priorizó la comunicación efectiva dentro del equipo para asegurar que las metas se alcanzaran sin problemas.

Alternancia de roles de Scrum Master

El equipo adoptó una estrategia de alternancia semanal en el rol de Scrum Master para promover una distribución equitativa de las responsabilidades de liderazgo. Esto permitió que ambos miembros:

- Adquirieran experiencia en la gestión de proyectos y coordinación de equipos.
- Tuvieran la oportunidad de mejorar sus habilidades organizativas y de planificación.
- Evitaran la concentración de decisiones en una sola persona, lo que fomentó la transparencia y el balance de poder dentro del equipo.

El cambio regular de Scrum Master también garantizó que las tareas y objetivos se revisaran desde perspectivas distintas, favoreciendo la identificación de mejoras y manteniendo al equipo alineado con la visión del proyecto.

Metodología y herramientas utilizadas

Para garantizar un desarrollo eficiente y una comunicación constante, el equipo utilizó herramientas y metodologías adaptadas a las necesidades del proyecto:

1. **Metodología de trabajo: Scrum**
   - Sprints de una semana para permitir iteraciones cortas y controladas.
   - Reuniones diarias para reportar avances, obstáculos y planificar acciones inmediatas.
   - Revisión y retrospectiva de Sprint al final de cada semana para analizar resultados y optimizar procesos.
2. **Herramientas empleadas**:
   - **Trello**: Se utilizó para gestionar el backlog de tareas, asignar prioridades, monitorizar el progreso y mantener la trazabilidad de las actividades.
   - **Documentación compartida**: Archivos en formato PDF y diagramas visuales para representar requisitos, casos de uso y flujos de trabajo.
   - **Control de versiones**: Git fue utilizado para gestionar el código fuente, facilitando la colaboración simultánea y evitando conflictos entre desarrolladores.
3. **Comunicación interna**:

- Reuniones diarias rápidas (stand-ups) de 10-15 minutos para sincronizar los esfuerzos del equipo.
- Resolución inmediata de problemas mediante canales directos de comunicación (por ejemplo, mensajería instantánea o llamadas).

Contribución de cada miembro

Ambos integrantes del equipo desempeñaron roles complementarios, y aunque la alternancia como Scrum Master fue clave, cada uno aportó de manera única al éxito del proyecto:

- **Marcos Cáceres García**:
  - Lideró la implementación de funcionalidades críticas como el sistema de inscripción y anulación.
  - Aseguró la integridad del código y participó en pruebas funcionales exhaustivas.
  - Durante sus semanas como Scrum Master, promovió un enfoque técnico centrado en resolver bloqueos y mejorar la arquitectura del sistema.
- **Pablo Valencia Palomino**:
  - Diseñó los componentes base del sistema, como diagramas UML y estructuras de base de datos.
  - Supervisó que los desarrollos cumplieran con los requisitos funcionales y normativos.
  - Como Scrum Master, se enfocó en la organización del trabajo y la priorización de tareas para cumplir con los plazos establecidos.

2. Especificación de requisitos

2.1. Requisitos funcionales

Los requisitos funcionales del sistema SICUE describen las funcionalidades específicas necesarias para satisfacer las necesidades del programa de intercambio académico. Estos requisitos fueron definidos a partir de reuniones con los interesados y del análisis detallado de los flujos de trabajo actuales.

1. Inscripción de estudiantes en planes de convalidación

- **Descripción**: El sistema debe permitir a los estudiantes registrar sus datos en los planes de intercambio académico, asegurando el reconocimiento de las asignaturas seleccionadas en sus universidades de origen.
- **Características específicas**:
  - Validar que el estudiante esté matriculado en segundo o tercer año y no haya participado previamente en el programa SICUE.
  - Permitir la inscripción solo dentro del rango de fechas establecido por el administrador.

- Registrar las asignaturas pendientes de cursos anteriores, permitiendo su inclusión en el plan de intercambio.

2. Anulación de inscripción

- **Descripción**: Los estudiantes deben tener la opción de anular su inscripción dentro del período permitido, sin eliminar registros del sistema.
- **Características específicas**:
  - Verificar que la solicitud de anulación esté dentro del período de inscripción activo.
  - Marcar las inscripciones como "anuladas" en lugar de eliminarlas, manteniendo el historial para auditorías.
  - Evitar que un estudiante pueda anular su inscripción fuera del periodo establecido o cuando el intercambio ya haya comenzado.

3. Gestión de roles y permisos

- **Descripción**: El sistema debe implementar un control de acceso basado en roles para garantizar que los usuarios solo puedan acceder a las funcionalidades y datos correspondientes a su función.
- **Roles definidos**:
  - **Administrador**:
    - Permisos completos sobre el sistema.
    - Gestión de usuarios, configuración de fechas y supervisión de inscripciones.
  - **Profesor**:
    - Validación y gestión de planes de convalidación relacionados con sus asignaturas.
    - Acceso limitado a datos relevantes para su participación.
  - **Estudiante**:
    - Acceso a su perfil, inscripciones y planes de convalidación.
    - Restricción para visualizar o modificar datos de otros usuarios.

4. Copias de seguridad automáticas

- **Descripción**: El sistema debe realizar copias de seguridad periódicas para prevenir la pérdida de información.
- **Características específicas**:
  - Programar intervalos regulares para la creación de respaldos.
  - Almacenar las copias de seguridad en un entorno seguro, con posibilidad de guardarlas en local y en servidores institucionales en el futuro.

- Notificar al administrador en caso de fallos durante el proceso de respaldo.

5. Gestión de convenios y convalidaciones

- **Descripción**: El administrador debe poder crear, modificar y supervisar convenios entre universidades.
- **Características específicas**:
  - Configuración de convenios para uno o dos semestres académicos.
  - Cruzar datos de estudiantes y profesores de manera automática y segura.
  - Restringir la modificación de convenios una vez iniciado el intercambio.

6. Generación de reportes

- **Descripción**: El sistema debe permitir la creación de reportes detallados para análisis y toma de decisiones.
- **Características específicas**:
  - Incluir información sobre estudiantes inscritos, asignaturas convalidadas y estadísticas de participación.
  - Exportar reportes en formatos PDF y Excel.
  - Limitar el acceso a los reportes a administradores y personal autorizado.

7. Configuración de periodos

- **Descripción**: Los administradores deben poder establecer y gestionar los periodos de inscripción y anulación.
- **Características específicas**:
  - Habilitar y deshabilitar funcionalidades automáticamente según las fechas configuradas.
  - Enviar notificaciones a usuarios relevantes cuando los periodos estén activos.

8. Seguridad de datos

- **Descripción**: Garantizar la protección de datos personales y académicos de los usuarios.
- **Características específicas**:
  - Almacenamiento cifrado de toda la información sensible.
  - Implementar autenticación de usuarios mediante credenciales únicas y seguras.
  - Cumplir estrictamente con las normativas del RGPD.

9. Compatibilidad académica

- **Descripción**: Asegurar que los estudiantes de cursos superiores puedan gestionar asignaturas de cursos inferiores en sus planes de convalidación.
- **Características específicas**:

- Validar inscripciones en asignaturas pendientes de cursos previos.
- Prevenir conflictos entre requisitos académicos y normativas institucionales.

2.2. Requisitos no funcionales

Los requisitos no funcionales del sistema SICUE definen las características de calidad y los criterios técnicos necesarios para garantizar la funcionalidad, seguridad y usabilidad del sistema. Estos requisitos complementan las especificaciones funcionales, asegurando que el sistema sea eficiente, seguro y escalable.

1. Seguridad

- **Descripción**: Garantizar la confidencialidad, integridad y disponibilidad de los datos.
- **Características específicas**:
  - Implementación de cifrado avanzado para los datos almacenados en la base de datos.
  - Control de acceso estricto basado en roles, permitiendo que los usuarios accedan solo a la información y funcionalidades correspondientes a su perfil.
  - Cumplimiento del Reglamento General de Protección de Datos (RGPD), incluyendo procesos para anonimizar datos si es necesario.

2. Rendimiento

- **Descripción**: El sistema debe responder eficientemente a las solicitudes de los usuarios, incluso bajo carga moderada.
- **Características específicas**:
  - Tiempos de respuesta inferiores a 2 segundos para las operaciones comunes, como inscripción, anulación y generación de reportes.
  - Optimización de consultas SQL para minimizar el tiempo de procesamiento de datos.
  - Capacidad para manejar al menos 500 usuarios simultáneos en un entorno local inicial, con posibilidades de ampliación futura.

3. Escalabilidad

- **Descripción**: Diseñar el sistema de manera que pueda adaptarse a mayores demandas sin necesidad de una reestructuración completa.
- **Características específicas**:
  - Estructura modular del código que permita agregar nuevas funcionalidades sin afectar las existentes.
  - Compatibilidad para migrar el sistema de un entorno local a servidores web institucionales (por ejemplo, en la Universidad de Córdoba).

4. Compatibilidad

- **Descripción**: Garantizar que el sistema sea compatible con diferentes entornos y dispositivos.
- **Características específicas**:
    - Soporte inicial para ejecución en sistemas operativos Windows, con vistas a ampliarlo a otros sistemas en fases posteriores.
    - Interfaz adaptable para accesibilidad básica, optimizada para resolución estándar de monitores de escritorio.

5. Mantenimiento

- **Descripción**: Facilitar el mantenimiento y las actualizaciones del sistema.
- **Características específicas**:
    - Documentación clara del código, incluyendo comentarios detallados y manuales de usuario y administrador.
    - Estructura de archivos organizada que permita a los desarrolladores identificar y modificar módulos específicos de manera rápida.

6. Usabilidad

- **Descripción**: Garantizar que el sistema sea fácil de usar para todos los usuarios, incluidos estudiantes, profesores y administradores.
- **Características específicas**:
    - Interfaz gráfica simple e intuitiva, con etiquetas claras y pasos guiados para tareas críticas como inscripciones y anulación.
    - Provisión de mensajes de error descriptivos y orientados a la solución, como "El periodo de inscripción ha terminado" en lugar de mensajes genéricos.

7. Fiabilidad

- **Descripción**: El sistema debe ser robusto y capaz de manejar errores o fallos sin comprometer la integridad de los datos.
- **Características específicas**:
    - Recuperación automática de operaciones fallidas, como reintento de copia de seguridad en caso de error.
    - Pruebas de estrés para identificar posibles puntos de fallo y solucionarlos antes del despliegue.

8. Portabilidad

- **Descripción**: Facilitar la instalación y ejecución del sistema en distintos entornos.
- **Características específicas**:
    - Provisión de un instalador único que contenga todas las dependencias necesarias para su funcionamiento.

- Instrucciones claras para la migración de bases de datos entre entornos locales y en la nube.

9. Normativas y estándares

- **Descripción**: Cumplir con normativas legales y estándares técnicos aplicables.
- **Características específicas**:
  - Cumplimiento con el RGPD para el manejo de datos personales.
  - Adaptación a estándares de desarrollo seguro de software, como OWASP.

3. Análisis de Requisitos

El análisis de requisitos del proyecto SICUE incluyó la identificación, clasificación y modelado de las necesidades funcionales y no funcionales del sistema. Este análisis se basó en reuniones con los interesados, revisiones del flujo de trabajo actual y documentación normativa. El enfoque incluyó historias de usuario, casos de uso y diagramas de casos de uso para representar de manera clara y estructurada las interacciones clave del sistema.

3.1. Historias de usuario

HU01: Inscripción en un plan de convalidación

- **Como** estudiante SICUE,
- **quiero** inscribirme en un plan de convalidación,
- **para** participar en el intercambio y asegurar que mis asignaturas sean reconocidas en mi universidad de origen.
- **Criterios de aceptación**:
  1. Validar que el estudiante esté matriculado en segundo o tercer año.
  2. Verificar que el estudiante no haya participado previamente en el programa SICUE.
  3. Permitir la inscripción dentro del rango de fechas establecido.
  4. Incluir asignaturas pendientes de cursos anteriores en el plan de convalidación.

HU02: Anulación de inscripción

- **Como** estudiante SICUE,
- **quiero** anular mi inscripción en el programa,
- **para** evitar participar si cambian mis circunstancias.
- **Criterios de aceptación**:
  1. La anulación solo puede realizarse dentro del periodo permitido.
  2. Marcar la inscripción como "anulada" en lugar de eliminarla.
  3. Prevenir que un estudiante anule su inscripción fuera de plazo.

4. El sistema debe notificar al estudiante tras completar la anulación.

HU03: Validación de planes de convalidación

- **Como** profesor participante en SICUE,
- **quiero** validar los planes de convalidación de los estudiantes,
- **para** asegurar que cumplen con los requisitos académicos y normativas del programa.
- **Criterios de aceptación**:
    1. El profesor debe autenticarse en el sistema con credenciales válidas.
    2. Seleccionar el plan de convalidación correspondiente a su asignatura.
    3. Aprobar o rechazar el plan según los criterios establecidos.

HU04: Gestión de convenios interuniversitarios

- **Como** administrador,
- **quiero** gestionar los convenios SICUE entre universidades,
- **para** garantizar que los acuerdos sean consistentes y estén actualizados.
- **Criterios de aceptación**:
    1. El administrador debe poder crear y modificar convenios.
    2. Configurar términos específicos, como duración de un semestre o un año académico.
    3. Restringir la edición de convenios a administradores autorizados.

HU05: Control de acceso y permisos

- **Como** administrador del sistema,
- **quiero** gestionar los roles y permisos de los usuarios,
- **para** asegurar que cada usuario acceda solo a la información y funcionalidades necesarias.
- **Criterios de aceptación**:
    1. Asignar roles predefinidos (administrador, profesor, estudiante).
    2. Limitar las acciones disponibles según el rol asignado.
    3. Prevenir accesos no autorizados mediante autenticación segura.

HU06: Copias de seguridad automáticas

- **Como** administrador,
- **quiero** que el sistema realice copias de seguridad de los datos periódicamente,
- **para** proteger la información en caso de fallos o pérdidas.
- **Criterios de aceptación**:

1. Configurar la frecuencia de las copias de seguridad.
2. Almacenar los respaldos en un entorno seguro y cifrado.
3. Notificar al administrador si ocurre un error durante el respaldo.

HU07: Generación de reportes

- **Como** administrador,
- **quiero** generar reportes detallados de estudiantes inscritos y sus planes de convalidación,
- **para** analizar y mejorar el programa SICUE.
- **Criterios de aceptación**:
    1. Seleccionar los parámetros del reporte (por ejemplo, periodo académico).
    2. Incluir información detallada de estudiantes, asignaturas y universidades involucradas.
    3. Exportar el reporte en formatos PDF y Excel.

HU08: Gestión de periodos

- **Como** administrador,
- **quiero** establecer rangos de fechas para inscripción y anulación,
- **para** controlar las operaciones del sistema según las fases del programa.
- **Criterios de aceptación**:
    1. Configurar fechas de inicio y fin para cada funcionalidad.
    2. Habilitar y deshabilitar las operaciones automáticamente según las fechas configuradas.
    3. Notificar a los usuarios cuando el periodo esté próximo a finalizar.

HU09: Almacenamiento seguro de datos

- **Como** administrador del sistema,
- **quiero** que los datos personales de estudiantes y profesores se almacenen cifrados,
- **para** cumplir con las normativas de protección de datos y evitar accesos no autorizados.
- **Criterios de aceptación**:
    1. Los datos personales deben cifrarse al ser almacenados en la base de datos.
    2. Solo usuarios autorizados deben tener acceso a datos sensibles.
    3. Registrar todas las actividades relacionadas con el acceso a datos sensibles.

3.2. Casos de uso

CU01: Inscripción en un Plan de Convalidación

- **Actor principal**: Estudiante.
- **Descripción**: Permitir a los estudiantes registrarse en un plan de convalidación de materias para participar en el intercambio académico.
- **Flujo principal**:
    1. El estudiante accede al sistema utilizando sus credenciales.
    2. Selecciona un plan de convalidación disponible en su área de estudio.
    3. Introduce las asignaturas que desea incluir en el plan.
    4. El sistema valida que:
        - El estudiante esté en segundo o tercer año.
        - Las asignaturas ingresadas son válidas y reconocidas por la universidad de destino.
    5. El sistema registra la inscripción y notifica al estudiante.
- **Extensiones**:
    - Si el estudiante no cumple los requisitos académicos, se muestra un mensaje de error y no se completa la inscripción.
    - Si el periodo de inscripción ha finalizado, el sistema bloquea el acceso a la funcionalidad.

CU02: Anulación de Inscripción

- **Actor principal**: Estudiante.
- **Descripción**: Proporcionar a los estudiantes la posibilidad de anular su inscripción dentro del periodo permitido.
- **Flujo principal**:
    1. El estudiante accede al sistema y selecciona la opción de anulación de inscripción.
    2. El sistema verifica que la solicitud esté dentro del rango de fechas configurado.
    3. La inscripción se marca como "anulada" y el estudiante recibe una notificación.
- **Extensiones**:
    - Si el periodo de anulación ha expirado, el sistema informa al estudiante que no puede anular la inscripción.
    - Si el intercambio ya ha comenzado, la funcionalidad de anulación queda deshabilitada.

CU03: Gestión de Convenios

- **Actor principal**: Administrador.
- **Descripción**: Administrar los convenios entre universidades participantes en el programa SICUE.
- **Flujo principal**:
    1. El administrador accede al módulo de gestión de convenios.
    2. Introduce o edita los detalles del convenio, como las universidades involucradas, la duración (cuatrimestre o año académico) y las asignaturas asociadas.
    3. El sistema valida y guarda la información del convenio.
- **Extensiones**:
    - Si el administrador intenta modificar un convenio activo con estudiantes inscritos, el sistema genera una alerta para evitar cambios.

CU04: Validación de Planes de Convalidación

- **Actor principal**: Profesor.
- **Descripción**: Permitir a los profesores validar los planes de convalidación de los estudiantes en base a las asignaturas propuestas.
- **Flujo principal**:
    1. El profesor accede al sistema y revisa las solicitudes de convalidación.
    2. Valida que las asignaturas propuestas por el estudiante sean equivalentes a las ofrecidas por la universidad de destino.
    3. Aprueba o rechaza la solicitud y el sistema notifica al estudiante.
- **Extensiones**:
    - Si las asignaturas no son equivalentes, el profesor puede sugerir modificaciones al plan.

CU05: Generación de Reportes

- **Actor principal**: Administrador.
- **Descripción**: Generar reportes detallados sobre los estudiantes inscritos y sus planes de convalidación.
- **Flujo principal**:
    1. El administrador selecciona los parámetros del reporte, como el rango de fechas, tipo de intercambio (cuatrimestre o anual) y área de estudio.
    2. El sistema genera un informe en formato PDF o Excel.
    3. El administrador descarga o visualiza el reporte.
- **Extensiones**:

- Si no hay datos disponibles para los parámetros seleccionados, el sistema genera un mensaje indicando que no se encontraron registros.

CU06: Configuración de Periodos

- **Actor principal**: Administrador.
- **Descripción**: Configurar los rangos de fechas en los que se permiten las inscripciones y anulaciones.
- **Flujo principal**:
  1. El administrador accede al módulo de configuración y establece las fechas de inicio y fin para cada periodo.
  2. El sistema actualiza automáticamente la disponibilidad de las funcionalidades según las fechas configuradas.
- **Extensiones**:
  - Si un rango de fechas se solapa con otro existente, el sistema alerta al administrador y no permite la configuración.

CU07: Seguridad de Datos

- **Actor principal**: Sistema (automático).
- **Descripción**: Garantizar la protección y el cifrado de datos personales.
- **Flujo principal**:
  1. Los datos personales se cifran automáticamente antes de almacenarse en la base de datos.
  2. El sistema valida las credenciales de los usuarios antes de permitir el acceso a los datos.
  3. Las actividades críticas, como cambios de datos o accesos, se registran en el log de auditoría.
- **Extensiones**:
  - Si se detecta un acceso no autorizado, el sistema bloquea la cuenta del usuario y notifica al administrador.

CU08: Copias de Seguridad

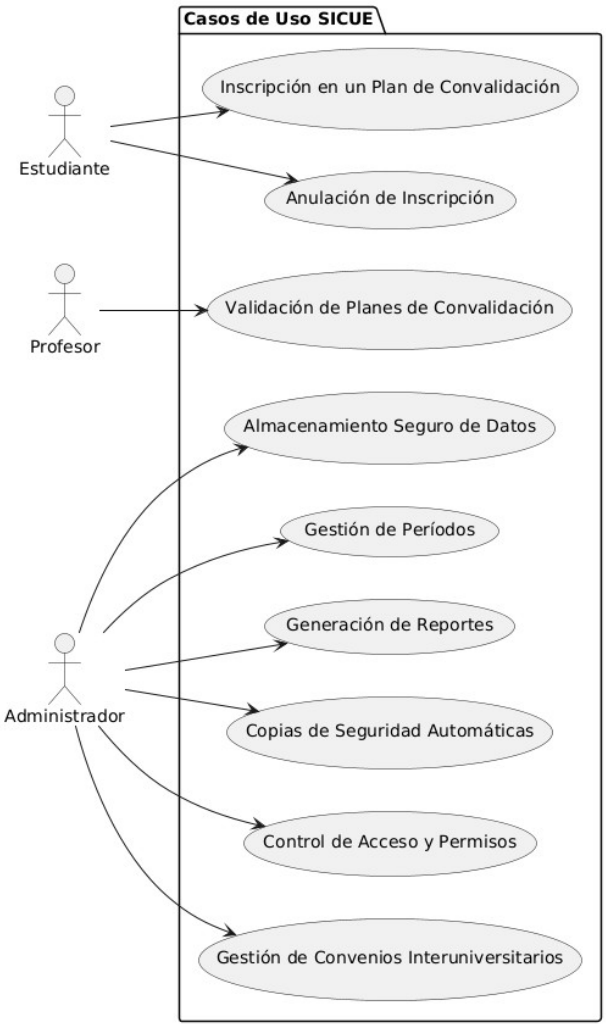
- **Actor principal**: Administrador y sistema (automático).
- **Descripción**: Realizar copias de seguridad regulares para garantizar la recuperación de datos en caso de fallo.
- **Flujo principal**:
  1. El administrador configura la frecuencia y el destino de las copias de seguridad.
  2. El sistema genera respaldos automáticos en los intervalos definidos.
  3. En caso de fallo en el proceso, se notifica al administrador.

- **Extensiones**:
    - Si el almacenamiento configurado está lleno, el sistema suspende el respaldo hasta que se libere espacio.

CU09: Gestión de Roles de Usuario

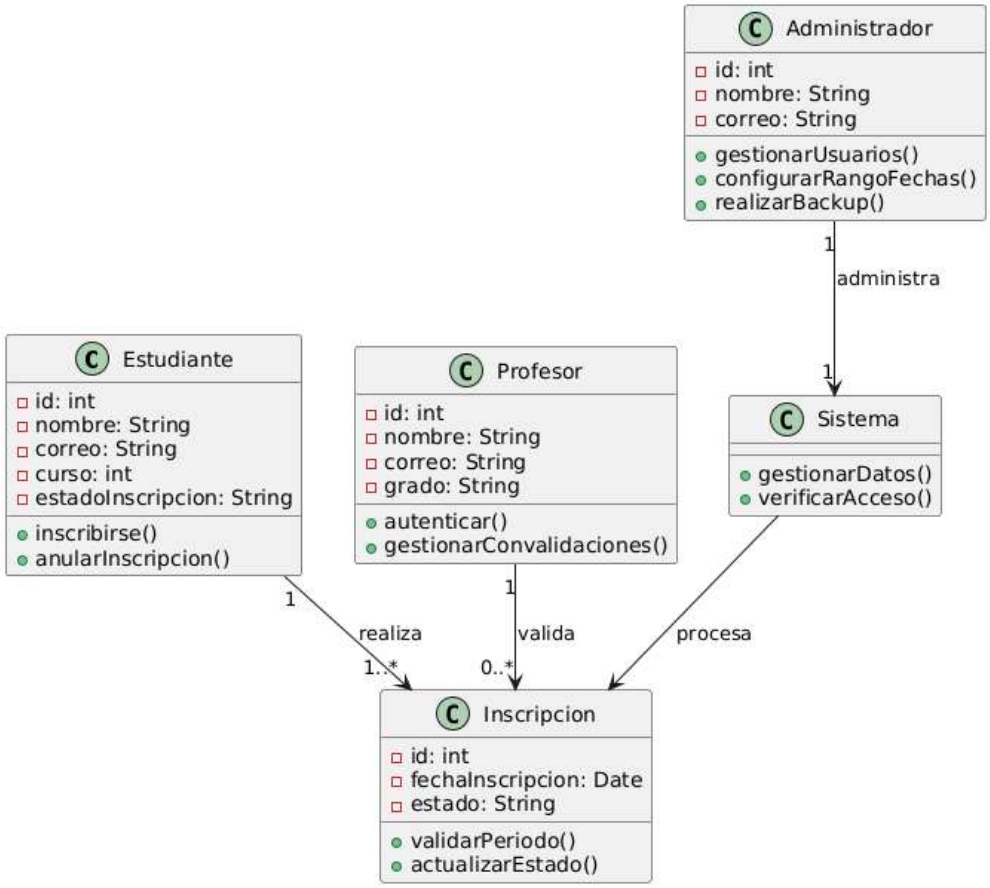
- **Actor principal**: Administrador.
- **Descripción**: Asignar y gestionar roles para estudiantes, profesores y administradores.
- **Flujo principal**:
    1. El administrador crea un nuevo usuario o edita uno existente.
    2. Asigna el rol correspondiente y configura los permisos asociados.
    3. El sistema registra los cambios y notifica al usuario.
- **Extensiones**:
    - Si se intenta asignar un rol inválido, el sistema alerta al administrador.

3.3. Diagrama de casos de uso



4. Diseño

4.1. Diagrama de clases



Clases principales

1. **Estudiante**
    - **Atributos**:
        - id: Identificador único del estudiante.
        - nombre: Nombre del estudiante.
        - correo: Dirección de correo electrónico.
        - curso: Curso académico del estudiante.
        - estadoInscripcion: Estado actual de la inscripción (por ejemplo, "activa", "anulada").
    - **Métodos**:
        - inscribirse(): Permite al estudiante realizar una inscripción en el programa.
        - anularInscripcion(): Permite anular una inscripción existente.
2. **Profesor**

- **Atributos**:
  - id: Identificador único del profesor.
  - nombre: Nombre del profesor.
  - correo: Dirección de correo electrónico.
  - grado: Grado académico o asignaturas impartidas por el profesor.
- **Métodos**:
  - autenticar(): Verifica la identidad del profesor para acceder al sistema.
  - gestionarConvalidaciones(): Permite gestionar las solicitudes de convalidación de asignaturas.

3. **Administrador**
   - **Atributos**:
     - id: Identificador único del administrador.
     - nombre: Nombre del administrador.
     - correo: Dirección de correo electrónico.
   - **Métodos**:
     - gestionarUsuarios(): Permite agregar, editar o eliminar usuarios.
     - configurarRangoFechas(): Define los periodos para inscripciones y anulaciones.
     - realizarBackup(): Realiza una copia de seguridad de los datos del sistema.
4. **Sistema**
   - **Atributos**: No se especifican.
   - **Métodos**:
     - gestionarDatos(): Procesa los datos relacionados con inscripciones, usuarios y permisos.
     - verificarAcceso(): Valida los permisos y roles de los usuarios.
5. **Inscripción**
   - **Atributos**:
     - id: Identificador único de la inscripción.
     - fechaInscripcion: Fecha en la que se realizó la inscripción.
     - estado: Estado actual de la inscripción (por ejemplo, "activa", "anulada").
   - **Métodos**:

- validarPeriodo(): Comprueba si la inscripción está dentro del periodo permitido.
- actualizarEstado(): Cambia el estado de la inscripción (por ejemplo, de "activa" a "anulada").

Relaciones entre las clases

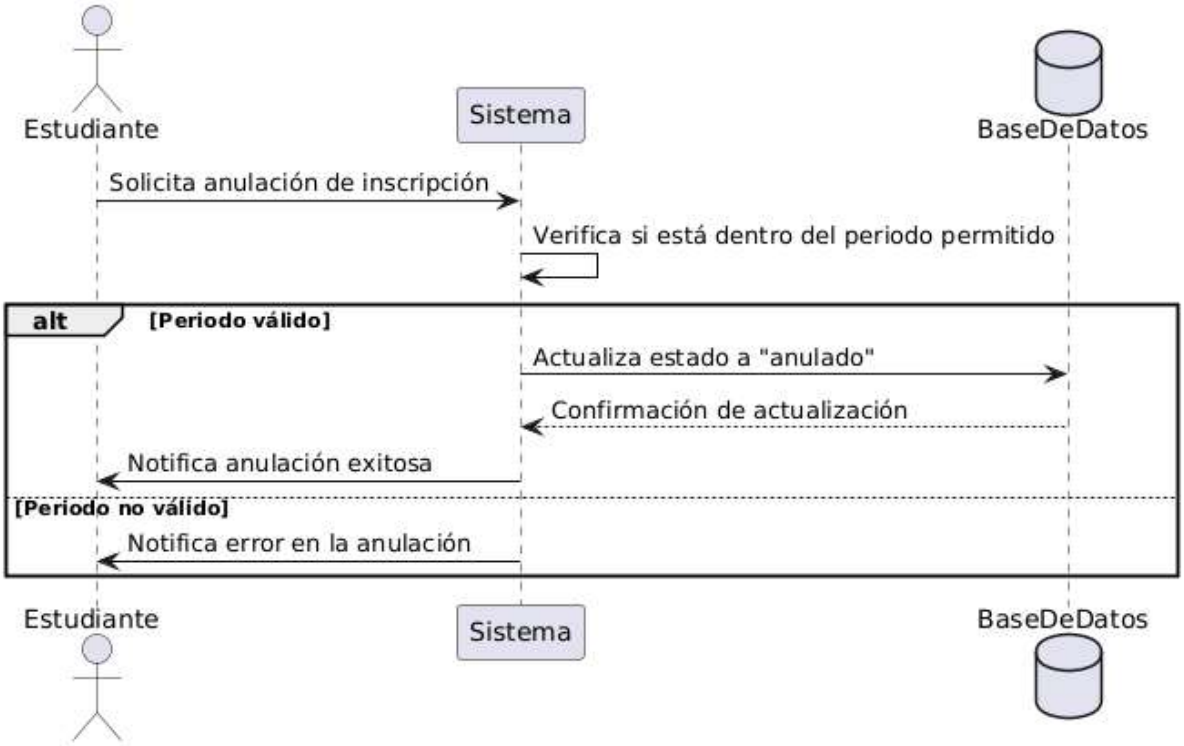
1. **Estudiante e Inscripción**:
   - Relación de "1 a muchos" (1..*): Un estudiante puede realizar varias inscripciones.
2. **Profesor e Inscripción**:
   - Relación de "0 a muchos" (0..*): Un profesor puede validar varias inscripciones, pero no es obligatorio.
3. **Administrador y Sistema**:
   - Relación de "1 a 1": Un administrador gestiona directamente el sistema.
4. **Sistema e Inscripción**:
   - Relación de procesamiento: El sistema procesa las solicitudes de inscripción y anulación.
5. **Profesor y Sistema**:
   - Relación indirecta: Los profesores interactúan con el sistema para validar inscripciones.

Interpretación general

Este diagrama detalla cómo se estructuran las clases dentro del sistema y cómo interactúan entre sí:

- Los **Estudiantes** realizan inscripciones, las cuales son validadas por **Profesores** y gestionadas por el **Sistema** bajo la supervisión de un **Administrador**.
- El **Administrador** tiene control total sobre el sistema, permitiéndole configurar periodos, realizar copias de seguridad y administrar usuarios.
- El **Sistema** actúa como núcleo, gestionando los datos y asegurando la seguridad y validez de las operaciones.

4.2. Diagramas de secuencia



Elementos del diagrama

1. **Actores y componentes**:
    - **Estudiante**: Usuario que solicita la anulación de la inscripción.
    - **Sistema**: Componente encargado de procesar la solicitud y realizar validaciones.
    - **BaseDeDatos**: Almacén donde se guardan los datos de las inscripciones, incluyendo su estado.
2. **Mensajes e interacciones**:
    - Las flechas representan mensajes enviados entre los actores y componentes.
    - Las líneas verticales son las **líneas de vida**, que indican la duración de cada objeto en el flujo.
3. **Bloque condicional (alt)**:
    - Representa una bifurcación en el flujo del proceso, dependiendo de si el período de anulación es válido o no.

Flujo de la secuencia

1. **Inicio del proceso**:
    - El **Estudiante** envía una solicitud de anulación de inscripción al **Sistema**.
2. **Validación del período**:
    - El **Sistema** verifica si la solicitud se realiza dentro del período permitido para anulaciones.

    - Este paso involucra una consulta interna o acceso a la **BaseDeDatos** para validar las fechas.

3. **Bloque condicional (alt)**:
    - **[Período válido]**:
        - El **Sistema** actualiza el estado de la inscripción en la **BaseDeDatos**, marcándola como "anulada".
        - La **BaseDeDatos** confirma que el estado fue actualizado correctamente.
        - El **Sistema** notifica al **Estudiante** que la anulación fue exitosa.
    - **[Período no válido]**:
        - Si el período ya no es válido, el **Sistema** notifica al **Estudiante** que hubo un error y que no es posible realizar la anulación.
4. **Fin del proceso**:
    - El flujo termina con la notificación al **Estudiante**, indicando el resultado de su solicitud.

Puntos clave del diseño

- **Validación estricta**: El sistema verifica las condiciones antes de modificar el estado en la base de datos, asegurando que las anulaciones se realicen solo dentro del período permitido.
- **Notificación clara**: El estudiante siempre recibe un mensaje que indica si la operación fue exitosa o no, mejorando la usabilidad.
- **Uso del bloque condicional (alt)**:
    - Proporciona una forma visual clara de mostrar las posibles rutas en el proceso.
    - Incluye tanto el caso exitoso como el caso de error.

Aplicación práctica

Este diagrama es esencial para comprender cómo el sistema maneja las solicitudes de anulación y asegura que las reglas del programa (como los períodos válidos) se respeten. Es especialmente útil para:

- Validar el diseño antes de implementar el código.
- Comunicar el flujo del proceso a los desarrolladores y al cliente.
- Identificar posibles puntos de fallo, como errores en la validación o actualizaciones fallidas.

5. Matriz de validación

Requisito funcional	**Historia de Usuario**	**Caso de Uso**
Inscripción de estudiantes	HU01: Inscripción en un plan de convalidación	CU01: Inscripción en un plan de convalidación
Anulación de inscripción	HU02: Anulación de inscripción	CU03: Anulación de inscripción
Validación de planes de convalidación	HU03: Validación de planes de convalidación	CU02: Validación de planes de convalidación
Gestión de convenios interuniversitarios	HU04: Gestión de convenios	CU04: Gestión de convenios interuniversitarios
Control de acceso y permisos	HU05: Control de acceso y permisos	CU05: Gestión de roles y permisos
Copias de seguridad automáticas	HU06: Copias de seguridad automáticas	CU06: Sistema de copias de seguridad
Generación de reportes	HU07: Generación de reportes	CU07: Generación de reportes
Configuración de periodos	HU08: Gestión de periodos	CU08: Configuración de periodos
Almacenamiento seguro de datos	HU09: Almacenamiento seguro de datos	CU09: Almacenamiento seguro y protección de datos

6. Implementación

6.1. Pruebas de rendimiento

Para garantizar la calidad del sistema SICUE, se diseñó un plan de pruebas que abarcó pruebas funcionales, de integración y de rendimiento. Estas pruebas permitieron verificar que el sistema cumpliera con los requisitos especificados y que las funcionalidades operaran correctamente bajo diversas condiciones.

1. Objetivos

- Verificar que las funcionalidades implementadas cumplan con los requisitos funcionales.
- Validar que los módulos individuales interactúen correctamente entre sí.
- Evaluar el rendimiento del sistema bajo condiciones normales y de alta carga.
- Asegurar la robustez del sistema frente a errores y entradas no válidas.

2. Tipos de pruebas

1. **Pruebas funcionales**:
    - **Objetivo**: Validar que cada funcionalidad del sistema opere de acuerdo con los requisitos especificados.
    - **Ejemplos de casos probados**:
        - Inscripción de estudiantes en planes de convalidación.
        - Anulación de inscripciones dentro del periodo permitido.
        - Configuración de periodos de inscripción y anulación por parte del administrador.
2. **Pruebas de integración**:
    - **Objetivo**: Asegurar que los módulos del sistema funcionen correctamente al interactuar entre ellos.
    - **Ejemplos de interacciones probadas**:
        - Comunicación entre el sistema y la base de datos al realizar inscripciones y actualizaciones.
        - Validación de permisos al acceder a funcionalidades restringidas según el rol de usuario.
        - Generación de reportes basados en datos almacenados en la base de datos.

3. Metodología

- **Preparación**:
    - Creación de un plan de pruebas detallado basado en los requisitos funcionales y casos de uso.

- Desarrollo de un conjunto de datos de prueba que incluyera casos válidos, inválidos y extremos.

- **Ejecución**:
  - Las pruebas se realizaron de forma iterativa al final de cada Sprint, comenzando con pruebas funcionales y avanzando hacia pruebas de integración y rendimiento.
  - Uso de herramientas automáticas para pruebas de carga y análisis de rendimiento.

7. Funcionamiento general

Funciones generales del sistema

1. **Inicio de sesión y gestión de usuarios** (login2.py):
   - **Iniciar sesión**: Permite a los usuarios acceder al sistema según su tipo (alumno, profesor, administrador).
   - **Registrar usuarios**: Permite registrar nuevos alumnos o profesores en la base de datos, validando que los datos no se dupliquen.
   - **Redirección a menús específicos**:
     - Administradores pueden gestionar planes, inscripciones y grados.
     - Profesores seleccionan asignaturas y consultan planes.
     - Alumnos realizan inscripciones, consultas y anulaciones.
2. **Menú principal** (menu.py):
   - Accesos directos para crear planes, realizar inscripciones y anular inscripciones.
   - Organización modular mediante scripts ejecutables.

Gestión de inscripciones

3. **Registrar inscripciones de alumnos** (inscripcion.py):
   - Validación del DNI y curso del alumno (solo alumnos de 2º o 3º año).
   - Inserción de inscripciones en la base de datos, asociándolas a un plan de convalidación.
   - Comprobación de duplicados antes de registrar una inscripción.
4. **Registrar inscripciones de profesores** (inscripciones_profesores2.py):
   - Selección de grados y asignaturas impartidas.
   - Registro en la base de datos para vincular profesores con asignaturas y periodos académicos.
5. **Anulación de inscripciones** (anular_inscripciones.py):

- Búsqueda y eliminación de inscripciones según el DNI del alumno.
- Muestra de inscripciones asociadas antes de anularlas, garantizando la confirmación.

6. **Consulta de inscripciones** (consulta_inscripciones.py):
   - Búsqueda de inscripciones por DNI del alumno.
   - Presentación de resultados en una tabla con detalles como universidades y fecha de inscripción.

Gestión de planes

7. **Crear planes de convalidación** (crear_plan.py):
   - Definición de universidades origen y destino, duración y asignaturas asociadas.
   - Inserción en la base de datos con validación de campos obligatorios.
8. **Consulta de planes de convalidación** (consulta_planes.py):
   - Recuperación y visualización de todos los planes disponibles en la base de datos.
   - Muestra información como universidades y asignaturas involucradas.

Gestión de grados y asignaturas

9. **Administrar grados y asignaturas** (administrar_asignaturas.py):
   - **Agregar o eliminar grados**: Gestión de programas educativos en la base de datos.
   - **Asignar o eliminar asignaturas de un grado**: Modificación dinámica de asignaturas asociadas a grados.
   - Actualización en tiempo real del listado de grados y asignaturas.

Características adicionales

- **Validación y seguridad**:
  - Validación de campos, como formato del DNI o requisitos específicos para inscripciones y anulaciones.
  - Protección contra duplicados en inscripciones, usuarios y planes.
- **Interfaz gráfica**:
  - Todas las funcionalidades están integradas en ventanas con interfaz gráfica basada en Tkinter.
  - Uso de componentes como TreeView, menús desplegables y botones para facilitar la interacción.
- **Modularidad**:
  - Cada funcionalidad está separada en scripts específicos, permitiendo mantenimiento y ampliación eficiente.

8. Pruebas

8.1. Pruebas de creación de planes de convalidación

Esta prueba valida la funcionalidad de crear un nuevo plan de convalidación en la base de datos.

```
@patch('crear_plan.conectar_db')
@patch('crear_plan.messagebox')
def test_crear_plan_convalidacion(mock_messagebox, mock_conectar_db):
    mock_conn = MagicMock()
    mock_cursor = MagicMock()
    mock_conectar_db.return_value.__enter__.return_value = mock_conn
    mock_conn.cursor.return_value = mock_cursor

    universidad_origen = "Universidad A"
    universidad_destino = "Universidad B"
    duracion = "6"
    asignaturas = "Asignatura 1, Asignatura 2"
    asignaturas_convalidadas = "Asignatura 3, Asignatura 4"

    with patch('crear_plan.entry_origen.get',
return_value=universidad_origen), \
         patch('crear_plan.entry_destino.get',
return_value=universidad_destino), \
         patch('crear_plan.entry_duracion.get', return_value=duracion), \
         patch('crear_plan.entry_asignaturas.get',
return_value=asignaturas), \
         patch('crear_plan.entry_asignaturas_convalidadas.get',
return_value=asignaturas_convalidadas):

        crear_plan_convalidacion()

        mock_cursor.execute.assert_called_once_with("""
            INSERT INTO planes_convalidacion (universidad_origen,
universidad_destino, duracion, asignaturas, asignaturas_convalidadas)
            VALUES (?, ?, ?, ?, ?)
        """, (universidad_origen, universidad_destino, duracion,
asignaturas, asignaturas_convalidadas))
        mock_conn.commit.assert_called_once()

        mock_messagebox.showinfo.assert_called_once_with("Éxito", "Plan
de convalidación creado con éxito.")

        crear_plan.entry_origen.delete.assert_called_once_with(0, tk.END)
        crear_plan.entry_destino.delete.assert_called_once_with(0,
tk.END)
        crear_plan.entry_duracion.delete.assert_called_once_with(0,
tk.END)
        crear_plan.entry_asignaturas.delete.assert_called_once_with(0,
tk.END)
```

```
        crear_plan.entry_asignaturas_convalidadas.delete.assert_called_on
ce_with(0, tk.END)
```

Explicación:

- Se valida que los datos ingresados se registren correctamente en la base de datos.
- Mocking asegura que los campos de entrada estén completos y que se muestre un mensaje de éxito si la operación es correcta.
- En caso de campos vacíos, se prueba que se emita un mensaje de error.

8.2. Pruebas de registro de inscripciones

Esta prueba valida que los estudiantes puedan registrarse en un plan de convalidación bajo condiciones válidas.

```
def test_validar_dni():
    assert validar_dni("12345678A") == True
    assert validar_dni("12345678") == False
    assert validar_dni("1234567A") == False
    assert validar_dni("12345678AA") == False

@patch('logica_inscripcion.conectar_db')
def test_registrar_inscripcion(mock_conectar_db):
    # Crear un mock para la conexi□n y el cursor
    mock_conn = MagicMock()
    mock_cursor = MagicMock()
    mock_conectar_db.return_value.__enter__.return_value = mock_conn
    mock_conn.cursor.return_value = mock_cursor

    dni = "12345678A"
    nombre = "Nombre"
    curso = "2□"
    plan_id = 1

    estado, mensaje = registrar_inscripcion_logic(dni, nombre, curso,
plan_id, mock_conectar_db)

    mock_cursor.execute.assert_any_call("SELECT * FROM inscripciones
WHERE estudiante_id = ? AND plan_id = ?", (dni, plan_id))
    mock_cursor.execute.assert_any_call("SELECT * FROM estudiantes WHERE
dni=?", (dni,))
    mock_cursor.execute.assert_any_call("INSERT INTO estudiantes (dni,
nombre, curso) VALUES (?, ?, ?)", (dni, nombre, curso))
    mock_cursor.execute.assert_any_call("""
        INSERT INTO inscripciones (estudiante_id, plan_id,
fecha_inscripcion)
        VALUES (?, ?, datetime('now'))
    """, (dni, plan_id))
    mock_conn.commit.assert_called()
```

```
    assert estado == "Éxito"
    assert mensaje == "Inscripción realizada con éxito."
```

Explicación:

- Verifica que un estudiante puede inscribirse en un plan si cumple con las condiciones: curso válido (2º o 3º año) y DNI correcto.
- Prueba también errores comunes, como DNI duplicado o inscripción previa en el mismo plan.

8.3. Pruebas de anulación de inscripciones

Esta prueba valida la funcionalidad de anular inscripciones existentes en el sistema.

```
@patch('logica_anular_inscripciones.conectar_db')
def test_anular_inscripciones(mock_conectar_db):
    mock_conn = MagicMock()
    mock_cursor = MagicMock()
    mock_conectar_db.return_value.__enter__.return_value = mock_conn
    mock_conn.cursor.return_value = mock_cursor

    dni = "12345678A"

    inscripciones = [("Nombre", "Curso", "Universidad Origen",
"Universidad Destino", "Fecha Inscripcion")]
    mock_cursor.fetchall.return_value = inscripciones

    estado, mensaje, resultado = anular_inscripciones_logic(dni,
mock_conectar_db)

    mock_cursor.execute.assert_any_call("""
        SELECT e.nombre, e.curso, p.universidad_origen,
p.universidad_destino, i.fecha_inscripcion
        FROM inscripciones i
        JOIN estudiantes e ON e.dni = i.estudiante_id
        JOIN planes_convalidacion p ON p.id = i.plan_id
        WHERE e.dni = ?
    """, (dni,))
    mock_cursor.execute.assert_any_call("DELETE FROM inscripciones WHERE
estudiante_id = ?", (dni,))
    mock_conn.commit.assert_called_once()

    assert estado == "□xito"
    assert mensaje == "Inscripciones anuladas con □xito para el alumno
con DNI: " + dni
    assert resultado == inscripciones
```

Explicación:

- Valida que, al ingresar un DNI, se eliminen las inscripciones asociadas correctamente.
- Se prueba que se devuelva un mensaje de éxito junto con los datos de las inscripciones anuladas.

- También se verifica el manejo de errores, como DNI vacío o inexistente.

9. Conclusiones

Conclusiones sobre el proyecto SICUE: Oportunidades de mejora y expansión

El proyecto SICUE cumple con los objetivos principales establecidos al inicio, proporcionando un sistema funcional para la gestión del intercambio académico. Sin embargo, tras analizar las funcionalidades y pruebas implementadas, se identifican áreas en las que el sistema puede mejorar y expandirse para optimizar su desempeño y adaptarse a futuras necesidades.

1. Mejoras en la experiencia del usuario

- **Interfaz de usuario más intuitiva**:
    - Aunque el sistema cuenta con una interfaz gráfica funcional basada en Tkinter, esta puede ser limitada en términos de diseño moderno y usabilidad.
    - Implementar una interfaz web con tecnologías como Django o Flask mejoraría significativamente la experiencia del usuario, permitiendo un acceso más versátil desde distintos dispositivos.
- **Mensajes de error más descriptivos**:
    - Algunos mensajes de error actuales son genéricos y podrían ser más específicos para guiar mejor al usuario en caso de problemas.

2. Optimización del rendimiento

- **Consultas a la base de datos**:
    - Las consultas SQL actuales son funcionales, pero podrían optimizarse para manejar grandes volúmenes de datos, especialmente en casos de uso intensivo del sistema.
    - La implementación de índices en columnas clave, como DNI o plan_id, mejoraría significativamente el tiempo de respuesta.
- **Gestión de transacciones**:
    - En operaciones críticas como inscripciones y anulaciones, se puede mejorar el manejo de transacciones para garantizar la consistencia de los datos, especialmente en escenarios de alta concurrencia.

3. Escalabilidad

- **Soporte para múltiples instituciones**:
    - Actualmente, el sistema está diseñado para operar en un entorno local y gestionar un número limitado de universidades.
    - Escalar el sistema para soportar múltiples instituciones y sincronizar datos entre ellas sería un paso clave hacia una implementación a nivel nacional.
- **Migración a la nube**:

- Implementar el sistema en un entorno en la nube permitiría un acceso remoto más amplio, con mejor disponibilidad y rendimiento para usuarios de diferentes ubicaciones.

4. Expansión de funcionalidades

- **Automatización de notificaciones**:
  - Incorporar notificaciones por correo electrónico o SMS para informar a estudiantes y profesores sobre el estado de sus inscripciones, plazos de inscripción o cambios en los planes de convalidación.
- **Gestión avanzada de roles**:
  - Incluir más niveles de acceso y permisos, permitiendo una mayor personalización según las necesidades de cada universidad.
- **Seguimiento de progreso académico**:
  - Integrar funcionalidades para que los estudiantes puedan hacer seguimiento del progreso de sus convalidaciones y ver el impacto en su historial académico.
- **Generación de estadísticas avanzadas**:
  - Incorporar herramientas de análisis para generar informes detallados sobre la participación, el rendimiento del programa y las tendencias, lo que sería útil para la toma de decisiones estratégicas.

5. Cumplimiento normativo y seguridad

- **Auditoría avanzada de acciones**:
  - Aunque el sistema registra ciertas acciones, una funcionalidad de auditoría avanzada permitiría rastrear todas las operaciones realizadas por usuarios, mejorando la transparencia y el cumplimiento normativo.
- **Cifrado de extremo a extremo**:
  - Aunque los datos se almacenan cifrados, implementar cifrado en tránsito (por ejemplo, HTTPS) mejoraría la seguridad en despliegues en red.

6. Automatización de pruebas

- **Cobertura de pruebas más amplia**:
  - Ampliar la cobertura de pruebas, especialmente en escenarios de carga alta y pruebas de seguridad, asegurará que el sistema sea robusto incluso en condiciones adversas.
- **Integración continua**:
  - Incorporar herramientas de integración continua como Jenkins o GitHub Actions para ejecutar pruebas automáticamente en cada actualización del sistema, reduciendo el riesgo de introducir errores.

10. Enlaces

Enlace Trello

Enlace GitHub

11. Uso Fuentes Externas

En este proyecto se ha utilizado ChatGPT, de la compañía OpenAI con los siguientes prompts:

- Lístame módulos gráficos de Python
- ¿Qué librería utilizo para crear test para Python?

También se ha utilizado un manual de Tkinter para la interfaz gráfica del programa.