MANUAL TÉCNICO

SITAPI

(Sistema Integrado de Tutorización de Alumnos de Primer Ingreso)

Autores (Equipo 307):
-Antonio Serrano Maldonado
-Adrián Sánchez Prieto
-Manuel Téllez Domínguez

Documento: Manual Técnico SITAPI Versión: 1.0 (Final) Fecha de emisión: 19 de diciembre de 2025 Estado: Aprobado para entrega.

Índice de contenidos

1. Introducción

***Nombre del sistema*:** Sistema Integrado de Tutorización de Alumnos de Primer Ingreso (SITAPI)
***Cliente*:** Vicerrectorado de la Universidad de Córdoba

1.2 Situación actual y problema identificado

El proceso de tutorización de estudiantes de primer ingreso en la UCO es manual y fragmentado: asignaciones por correo electrónico o reuniones, seguimiento con hojas de cálculo y comunicación informal que dificulta la trazabilidad.

Los tutores no tienen una vista consolidada de sus tutelados, los coordinadores carecen de métricas fiables y la detección de estudiantes con riesgo de fracaso académico es reactiva.
Esto provoca retrasos en intervenciones, sobrecarga administrativa y pérdida de oportunidades para acompañar a estudiantes en su adaptación académica.

1.3 Objetivo del sistema

- El sistema de tutorización digitalizará y centralizará la gestión de la tutoría para estudiantes de primer ingreso, facilitando la asignación (manual y automática) de tutores, el canal directo de comunicación tutor-estudiante, el registro estructurado de sesiones y la detección temprana de riesgos académicos.

- Su finalidad es mejorar la eficacia del acompañamiento, reducir la carga administrativa y proporcionar métricas accionables al vicerrectorado y coordinadores.

2. Metodología Scrum

Para la realización del proyecto Software se ha optado por la metodología Scrum con sprints de duración de entre 1-2 semanas, la plataforma de apoyo ha sido Youtrack, con dicho enlace a continuación, ***https://antonioserrano.youtrack.cloud/***.

La lista de Sprints es la siguiente:

Sprints de Análisis:
- Análisis 1 (1 - 7 oct) (Product Owner : Antonio Serrano)
- Análisis 2 (14 - 21 oct) (Product Owner : Adrián Sánchez)
- Análisis 3 (21 oct - 4 nov) (Product Owner : Manuel Téllez)

Sprints de Diseño

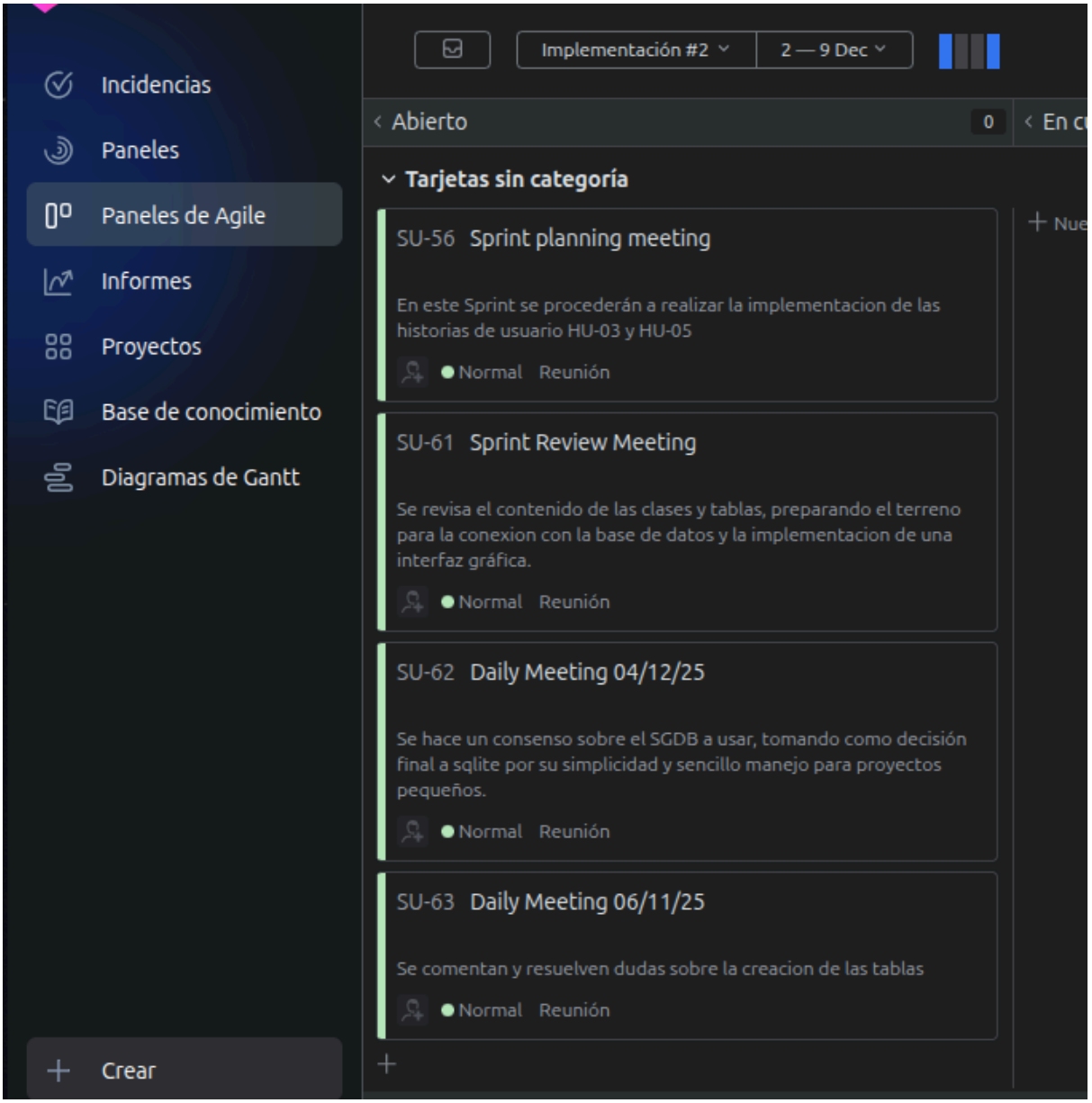
- Diseño (11 - 18 nov) (Product Owner : Antonio Serrano)

Sprint de Implementación:

- Implementación 1 (24 nov - 2 dic) (Product Owner : Antonio Serrano)
- Implementación 2 (2 - 9 dic) (Product Owner : Adrián Sánchez)
- Implementación 3 (9 - 16 dic) (Product Owner : Manuel Téllez)
- Implementación 4 (16 - 21 dic) (Product Owner : Antonio Serrano)

El tablero kanban principal se encuentra en el apartado de “Paneles de Agile”, ahí se encontrarán la lista de Sprints, las historias de usuario se encuentran en la pestaña de “Trabajo Pendiente” usada únicamente con fines de organización. Dentro del Tablero Kanban las reuniones del Sprint se encuentran en la columna de “Abierto”. A continuación se añade una captura de pantalla para complementar la explicación:

figura 1. Panel de Agile de Youtrack donde se pueden apreciar reuniones y el Sprint asociado



3. Especificación de Requisitos

A continuación se presenta una lista de requisitos preliminar, la cual fue la lista original, que luego mediante métodos débiles de análisis se corrigieron dando lugar a la lista de requisitos definitiva la cual se encuentra debajo.

3.1 Lista de requisitos preliminar

3.1.1 Requisitos Funcionales (preliminares)

ID	Descripción
RF-1	El sistema debe autenticar el rol de los usuarios.
RF-2	El sistema debe asignar automáticamente a cada estudiante un tutor desde el primer día.
RF-3	El sistema debe permitir la gestión de roles de los usuarios.
RF-4	El sistema debe permitir la asignación manual de tutores en casos especiales.
RF-5	Los coordinadores deben poder gestionar los roles de los usuarios.
RF-6	El sistema debe ofrecer un canal de mensajería instantánea entre tutor y estudiante.
RF-7	El tutor debe poder registrar las sesiones de tutoría.
RF-8	El sistema debe generar alertas automáticas si se detecta un estudiante con dificultades.
RF-9	Los tutores deben poder acceder fácilmente a la información relevante de sus tutorados.
RF-10	El sistema debe mostrar información sobre los estudiantes a los tutores.
RF-11	El sistema debe disponer de un módulo de asignación inteligente que empareje estudiantes con tutores en base a características.
RF-12	Los coordinadores deben poder generar informes estadísticos.
RF-13	Los reportes deben incluir métricas numéricas claras.

3.1.2 Requisitos No Funcionales (preliminares)

ID	Descripción
RNF-1	La interfaz debe ser intuitiva, fácil de usar y accesible.
RNF-2	El sistema debe ser multidispositivo.
RNF-3	El tiempo de respuesta debe ser rápido.
RNF-4	El sistema debe poder gestionar simultáneamente multitud de usuarios activos.
RNF-5	Los datos personales y académicos de los estudiantes deben almacenarse cifrados.
RNF-6	El sistema debe cumplir con las políticas de privacidad y las normativas vigentes sobre protección de datos personales.
RNF-7	El sistema ha de ser rápido indistintamente del dispositivo en el que se use.
RNF-8	El acceso a la información debe estar restringido según el rol del usuario.
RNF-9	El sistema debe integrarse fácilmente con otros sistemas informáticos de la UCO.

3.2 Métodos débiles de análisis

Para obtener la lista de requisitos definitiva hemos optado por el uso de una matriz de interacción que puede encontrar en el siguiente enlace:
Matriz de interacción

3.3 Lista de requisitos definitiva

3.3.1 Requisitos Funcionales (definitivos)

ID	Descripción
RF-1	El sistema debe autenticar y gestionar el rol de los usuarios.
RF-2	El sistema debe asignar automáticamente a cada estudiante un tutor desde el primer día, usando un módulo de asignación inteligente que empareje estudiantes con tutores en base a características.
RF-3	El sistema debe permitir la asignación manual de tutores en casos especiales.
RF-4	Los coordinadores deben poder gestionar los roles de los usuarios.
RF-5	El sistema debe ofrecer un canal de mensajería instantánea entre tutor y estudiante.
RF-6	El tutor debe poder registrar las sesiones de tutoría.
RF-7	El sistema debe generar alertas automáticas si se detecta un estudiante con dificultades.
RF-8	Los tutores deben poder acceder fácilmente a la información relevante de sus tutorados.
RF-9	Los coordinadores deben poder generar informes estadísticos.
RF-10	Los reportes deben incluir métricas numéricas claras.

3.3.2 Requisitos No Funcionales (definitivos)

del producto

ID	Descripción
RNF-1	La interfaz debe ser intuitiva, fácil de usar y accesible.
RNF-2	El sistema debe ser multidispositivo.
RNF-3	El tiempo de respuesta debe ser rápido en cualquier dispositivo.
RNF-4	El sistema debe poder gestionar simultáneamente multitud de usuarios activos.
RNF-6	El acceso a la información debe estar restringido según el rol del usuario.

Externos

ID	Descripción
RNF-5	El sistema debe cumplir con las políticas de privacidad y las normativas vigentes sobre protección de datos personales.

Organizacionales

ID	Descripción
RNF-7	El sistema debe integrarse fácilmente con otros sistemas informáticos de la UCO.

3.3.3 Requisitos de Información

ID	Entidad	Información gestionada
RI-1	**Estudiante**	Nombre, información académica, datos de alojamiento, tutor asignado, historial de tutorías, alertas registradas, métricas de progreso.
RI-2	**Tutor**	Datos personales y de contacto, rol, áreas de conocimiento, disponibilidad horaria, historial de sesiones realizadas, alertas generadas.
RI-3	**Coordinador**	Datos personales, facultad asociada, responsabilidades de gestión, informes generados, configuraciones de criterios de asignación.
RI-4	**Asignación de tutores**	Registros de emparejamientos estudiante-tutor, criterios utilizados, fecha de asignación y modificaciones.
RI-5	**Tutorías**	Detalles de cada sesión, nivel de satisfacción y alertas derivadas.
RI-6	**Mensajería**	Mensajes enviados entre tutores y estudiantes.
RI-7	**Alertas y avisos**	Registros de alertas tempranas asociadas a un estudiante y su tutor.
RI-8	**Informes y métricas**	Datos estadísticos.

4.Casos de Uso

4.1 Casos de Uso: Descripciones

Campo	Descripción
Identificación	CU-01
Nombre	Identificación de rol
Objetivo	Permite a un usuario registrado acceder al sistema proporcionando sus credenciales para verificar su rol.
Contexto	El usuario debe estar registrado previamente en el sistema.
Actor principal	Estudiante, Tutor, Coordinador
Escenario principal	El usuario accede a la página de inicio del sistema SITAPI. El sistema muestra un formulario para acreditar el rol. El usuario introduce su ID y pulsa el botón de acceder. El sistema valida el rol del usuario. El sistema concede el acceso y redirige al usuario a una página personalizada según su rol.

Campo	Descripción
Identificación	CU-02
Nombre	Registrar sesión de tutoría
Objetivo	El tutor registra los detalles de una sesión de tutoría realizada con un estudiante para mantener un historial.
Contexto	El tutor ha iniciado sesión con su rol, tiene estudiante asignado y ya ha realizado una sesión de tutoría.
Actor principal	Tutor
Escenario principal	El tutor accede a su panel y selecciona al estudiante. El tutor selecciona la opción “Registrar nueva tutoría”. El sistema presenta un formulario con campos como fecha, temas tratados, observaciones y satisfacción del estudiante. El tutor completa la información y guarda el registro. El sistema almacena la información en la base de datos y la asocia al historial del estudiante.
Escenario alternativo	Si un campo obligatorio no se rellena, el sistema muestra un aviso y no permite guardar hasta completar la información.

Campo	Descripción
Identificación	CU-03
Nombre	Enviar y recibir mensajes
Objetivo	Facilitar la comunicación directa entre un tutor y su estudiante asignado mediante un sistema de mensajería propio.
Contexto	Existe una asignación válida entre tutor y estudiante.
Actor principal	Estudiante, Tutor
Escenario principal	El usuario emisor accede a la sección de mensajería. Selecciona al destinatario. Escribe el mensaje y lo envía. El sistema almacena el mensaje y notifica al destinatario. El usuario receptor recibe la notificación y visualiza el mensaje. El mensaje se guarda en el historial de conversaciones.

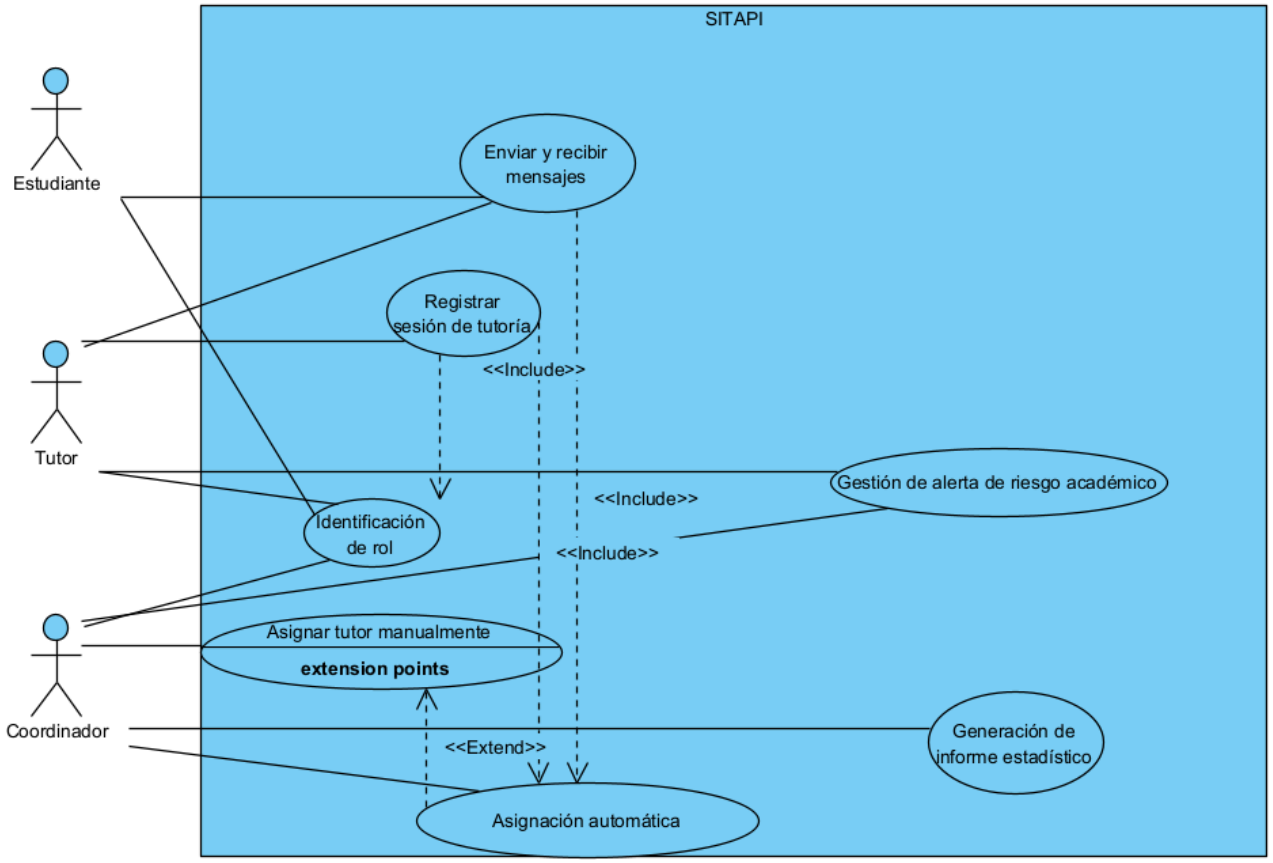
Campo	Descripción
Identificación	CU-04
Nombre	Asignar tutor manualmente
Objetivo	Permite al coordinador asignar un tutor específico a un estudiante por necesidad especial o para corregir una asignación automática.
Contexto	El estudiante y el tutor existen en la base de datos.
Actor principal	Coordinador
Escenario principal	El coordinador accede al módulo de gestión de asignaciones. Busca y selecciona al estudiante. El sistema muestra una lista de tutores indicando su disponibilidad. El coordinador selecciona al tutor deseado. El coordinador confirma la asignación.6. El sistema actualiza el registro realizando la vinculación.

Campo	Descripción
Identificación	CU-05
Nombre	Asignación automática
Objetivo	Permite la asignación tutor-estudiante sin intervención directa del coordinador mediante el uso de algoritmos.
Contexto	Los roles de los usuarios han de estar definidos previamente.
Actor principal	Coordinador
Escenario principal	El coordinador accede al módulo de gestión de asignaciones. Selecciona la opción de asignación automática. El sistema calcula, guarda y establece las asignaciones.

Campo	Descripción
Identificación	CU-06
Nombre	Gestión de alerta de riesgo académico
Objetivo	Notificar y gestionar alertas automáticas sobre estudiantes que presentan indicadores de riesgo académico.
Contexto	El tutor ha iniciado sesión y el sistema ha generado una alerta automática.
Actor principal	Tutor, Coordinador
Escenario principal	El sistema muestra una notificación de nueva alerta en el panel del tutor. El tutor accede a la sección de alertas y selecciona la alerta. El sistema muestra los detalles de la alerta. El tutor cambia el estado a “Vista”, “En seguimiento” o “Resuelta”. El tutor añade comentarios si es necesario. El sistema guarda el estado actualizado.

Campo	Descripción
Identificación	CU-07
Nombre	Generación de informe estadístico
Objetivo	Generar informes con datos agregados y métricas sobre la tutorización para evaluar su efectividad y tomar decisiones.
Contexto	Existen datos de tutorías, asignaciones y alertas en el sistema.
Actor principal	Coordinador
Escenario principal	El coordinador accede al módulo de informes y estadísticas. Selecciona el tipo de informe. Aplica filtros si lo deseas. Ejecuta la generación del informe. El sistema procesa los datos y muestra métricas y gráficos. El sistema permite exportar el informe.

4.2 Diagrama de casos de uso



5. Historias de Usuario

Campo	Contenido
Identificador	HU-01
Nombre	Identificación de rol
Responsable	Usuarios
Descripción	Como usuario quiero acceder al sistema con credenciales para identificar mi rol “Me gustaría poder entrar al sistema de forma segura para que el sistema identifique automáticamente si soy estudiante, tutor o coordinador y así solo ver las opciones y funcionalidades que me corresponden según mi rol, evitando accesos incorrectos o información que no necesito.”
Prioridad	Crítica
Estimación en días	3 días
Sprint	3
Criterios de validación	El sistema debe permitir el acceso mediante credenciales válidas. El sistema debe identificar correctamente el rol del usuario tras el inicio de sesión. El usuario debe ser redirigido a las funcionalidades correspondientes a su rol.. El acceso debe ser denegado cuando las credenciales sean incorrectas. El sistema debe mostrar un mensaje de error claro en caso de fallo de autenticación. Los roles y permisos deben estar definidos y consensuados previamente con el cliente.

Campo	Contenido
Identificador	HU-02
Nombre	Registrar sesión de tutoría
Responsable	Tutor
Descripción	Como tutor/a quiero disponer de un registro de tutoría con el estudiante para mantener un historial centralizado y trazable de las sesiones. “Me gustaría poder guardar la información de cada tutoría que realizo para no depender de notas externas, poder recordar fácilmente lo que se ha tratado en sesiones anteriores y llevar un seguimiento claro y ordenado del progreso del estudiante”.
Prioridad	Mayor
Estimación en días	4 días
Sprint	3
Criterios de validación	El sistema debe permitir crear un registro por cada sesión de tutoría. El registro debe incluir fecha, tutor, estudiante y observaciones. Los registros deben almacenarse de forma centralizada. El tutor debe poder consultar el historial de sesiones de un estudiante. La información registrada debe ser trazable y no editable sin permisos.

Campo	Contenido
Identificador	HU-03
Nombre	Enviar y recibir mensajes
Responsable	Tutor / Estudiante
Descripción	Como tutor/a o estudiante quiero poder enviar y recibir mensajes dentro del sistema para facilitar la comunicación directa y asíncrona con mi tutor/a o estudiante asignado. “Me gustaría poder comunicarme directamente con mi tutor o estudiante desde el propio sistema para resolver dudas, coordinar tutorías y mantener una comunicación continua sin necesidad de usar aplicaciones externas, pudiendo consultar los mensajes cuando lo necesite.”
Prioridad	Crítica
Estimación en días	5 días
Sprint	3
Criterios de validación	El sistema debe permitir enviar mensajes entre tutor y estudiante asignados. Los mensajes deben poder recibirse de forma asíncrona. Las conversaciones deben almacenarse en el sistema. El usuario debe poder consultar el historial de mensajes. Solo los usuarios autorizados deben acceder a las conversaciones.

Campo	Contenido
Identificador	HU-04
Nombre	Asignar tutor manualmente
Responsable	Coordinador
Descripción	Como coordinador/a quiero poder asignar específicamente un tutor a un alumno de manera manual. “Me gustaría poder elegir manualmente qué tutor se asigna a un estudiante cuando la asignación automática no es adecuada, para asegurar que el alumno reciba el acompañamiento más adecuado según sus necesidades específicas”
Prioridad	Mayor
Estimación en días	4 días
Sprint	3
Criterios de validación	El coordinador debe poder seleccionar manualmente un tutor y un estudiante. La asignación debe quedar registrada en el sistema. El sistema debe permitir modificar la asignación si es necesario. Esta funcionalidad solo debe activarse cuando la asignación automática sea insuficiente, según criterio consensuado con el cliente.

Campo	Contenido
Identificador	HU-05
Nombre	Asignación automática
Responsable	Coordinador
Descripción	Como coordinador/a quiero que el sistema sea capaz de realizar una asignación automática tutor-estudiante. “Me gustaría que el sistema realice automáticamente la asignación de tutores a los estudiantes para ahorrar tiempo, evitar hacerlo de forma manual uno a uno y asegurar que las asignaciones se realicen siguiendo criterios previamente definidos.”
Prioridad	Crítica
Estimación en días	6 días
Sprint	3
Criterios de validación	El sistema debe asignar automáticamente tutores a estudiantes. La asignación debe basarse en criterios definidos previamente. El proceso debe ejecutarse sin intervención manual. Las asignaciones deben quedar registradas y ser consultables. El coordinador debe poder revisar el resultado de la asignación.

Campo	Contenido
Identificador	HU-06
Nombre	Gestión de alerta de riesgo académico
Responsable	Tutor / Coordinador
Descripción	Como tutor/a o coordinador/a quiero poder gestionar alertas académicas estudiantiles para llevar a cabo un seguimiento oportuno de casos de bajo rendimiento o riesgo académico. “Me gustaría poder detectar y gestionar alertas de riesgo académico para identificar a tiempo a los estudiantes con posibles dificultades y poder realizar un seguimiento adecuado, permitiendo intervenir de forma temprana y mejorar su rendimiento académico”
Prioridad	Mayor
Estimación en días	5 días
Sprint	3
Criterios de validación	El sistema debe permitir crear alertas de riesgo académico. Las alertas deben asociarse a un estudiante concreto. Tutor y coordinador deben poder consultar las alertas. Las alertas deben permitir registrar acciones de seguimiento. El historial de alertas debe mantenerse actualizado y trazable.

Campo	Contenido
Identificador	HU-07
Nombre	Generación de informe estadístico
Responsable	Coordinador
Descripción	Como coordinador/a quiero poder generar informes estadísticos con datos y métricas sobre la tutorización para evaluar su efectividad y tomar decisiones. “Me gustaría poder consultar resúmenes e informes estadísticos del sistema para analizar cómo está funcionando la tutorización, tomar decisiones basadas en datos reales y evaluar si las medidas aplicadas están siendo efectivas”.
Prioridad	Normal
Estimación en días	3 días
Sprint	3
Criterios de validación	El sistema debe generar informes estadísticos automáticamente. Los informes deben basarse en datos reales del sistema. El coordinador debe poder consultar y descargar los informes. Los datos deben presentarse de forma clara y comprensible. Las métricas mostradas deben estar consensuadas con el cliente.

6. Especificación de Clases

Clase Usuario <<abstract>>
Descripción: Clase base que representa a cualquier usuario del sistema

visibilidad	Atributos	Tipo	Descripción
-	ID	String	Identificador único del usuario
-	nombre	String	Nombre completo del usuario
-	email	String	Correo electrónico

Operaciones

	Operación	Tipo retorno	Descripción
+	Autenticar	String	Verifica las credenciales del usuario

Clase Estudiante
Descripción: Representa a un estudiante de primer ingreso

visibilidad	Atributos	Tipo	Descripción
-	información académica	String	Datos académicos del estudiante
-	datos alojamiento	String	Información sobre su lugar de residencia
-	id_tutor	String	Tutor asignado al estudiante
-	progreso	String	Indicadores de progreso académico

Operaciones

	Operación	Tipo retorno	Descripción
+	EnviarMensaje	Mensaje	Envía mensaje a tutor

Clase Tutor
Descripción: Representa a un tutor

visibilidad	Atributos	Tipo	Descripción
-	área conocimiento	String	Áreas de especialización
-	disponibilidad horaria	String	Horarios disponibles para tutorías
-	Facultad	String	Facultad a la que pertenece

Operaciones

	Operación	Tipo retorno	Descripción
+	Registrar Sesión	Tutoria	Registra una nueva sesión de tutoría
+	EnviarMensaje	Mensaje	Envía un mensaje a un estudiante
+	GestionarAlerta	void	Actualiza el estado de una alerta académica

Clase Coordinador

Descripción: Representa a un coordinador que tiene permisos de gestor en el sistema

visibilidad	Atributos	Tipo	Descripción
-	facultad	String	Facultad a la que pertenece
-	responsabilidades gestión	String	Responsabilidades asignadas

Operaciones

	Operación	Tipo retorno	Descripción
+	AsignarTutorManual	List<Asignacion>	Asigna manualmente un tutor a un estudiante
+	EjecutarAsignación Automática	List<Asignacion>	Ejecuta el algoritmo de asignación automática
+	GenerarInforme	Informe	Genera un informe estadístico

Clase Tutoría
Descripción: Representa una sesión de tutoría entre un tutor y un estudiante

visibilidad	atributos	tipo	descripción
-	id	int	Identificador único de la tutoría
-	id_estudiante	String	Identificador del estudiante que recibe la tutoría
-	id_tutor	String	Identificador del tutor que da la tutoría.
-	fecha	Date	Fecha de la sesión
-	temas tratados	String	Temas discutidos en la tutoría
-	observaciones	String	Observaciones adicionales
-	satisfacción	int	Nivel de satisfacción del estudiante
-	alertas derivadas	List<Alerta>	lista de alertas generadas a partir de la sesión

Operaciones

	Operación	Tipo retorno	Descripción
+	GenerarTutoría	bool	Genera una tutoría
+	ConsultarTutoría	List<Tutoría>	Muestra las tutorías asociadas a un alumno
+	ModificarTutoría	bool	Modifica los valores de una tutoría

Clase Mensaje
Descripción: Representa un mensaje entre tutor y estudiante

visibilidad	Atributos	Tipo	Descripción
-	id_mensaje	int	Identificador del mensaje
-	id_chat	int	Identificador del chat al que pertenece
-	contenido	String	Contenido del mensaje
-	fecha de envío	Date	Fecha y hora de envío
-	remitente	String	Usuario que realiza el envío

Clase Alerta
Descripción: Representa una alerta de riesgo académico asociada a un estudiante.

visibilidad	Atributos	Tipo	Descripción
-	id	int	Identificador único de la alerta
-	id_estudiante	String	Estudiante asociado a la alerta
-	motivo	String	Razón de la alerta
-	fecha	Date	Fecha de generación
-	estado	String	Estado: "Pendiente", "En seguimiento", "Resuelta"
-	comentarios	String	Comentarios adicionales

Operaciones

	Operación	Tipo retorno	Descripción
+	ConsultarTodasLasAlertas	List<Alerta>	Devuelve todas las alertas del sistema
+	ConsultarAlertaPorEstudiante	List<Alerta>	Devuelve todas las alertas asociadas a un estudiante
+	GenerarAlerta	bool	Genera una alerta
+	ActualizarEstadoAlerta	bool	Actualiza una alerta

Clase Informe
Descripción: Representa un informe estadístico creado por un coordinador

visibilidad	atributos	tipo	descripción
-	tipo	String	Tipo de informe (ej. efectividad, asignaciones)
-	métricas	String	Métricas numéricas incluidas
-	gráficos	File	Gráficos generados
-	fecha generación	Date	Fecha de creación del informe

Operaciones

	Operación	Tipo retorno	Descripción
+	Exportar	void	Exporta el informe en formato descargable

Clase Asignación
Descripción: Representa una asignación tutor-estudiante

	Atributos	Tipo	Descripción
-	id	int	Identificador único de la asignación
-	id_estudiante	String	Estudiante asignado
-	id_tutor	String	Tutor asignado
-	criterios	String	Criterios aplicados en la asignación
-	fecha Asignacion	Date	Fecha en que se realizó la asignación
-	modificaciones	String	Historial de modificaciones

Operaciones

	Operación	Tipo retorno	Descripción
+	ConsultarTodasLasAsignaciones	list<Asignaciones>	Devuelve todas las asignaciones de la base de datos
+	GetTutorDeEstudiante	String	Devuelve el tutor asociado a un estudiante
+	VerHistorialDeEstudiante	list<Asignacion>	Devuelve el historial de asignaciones que ha tenido un alumno
+	GetAlumnosDeTutor	list<Estudiante>	Devuelve los alumnos asociados a un tutor

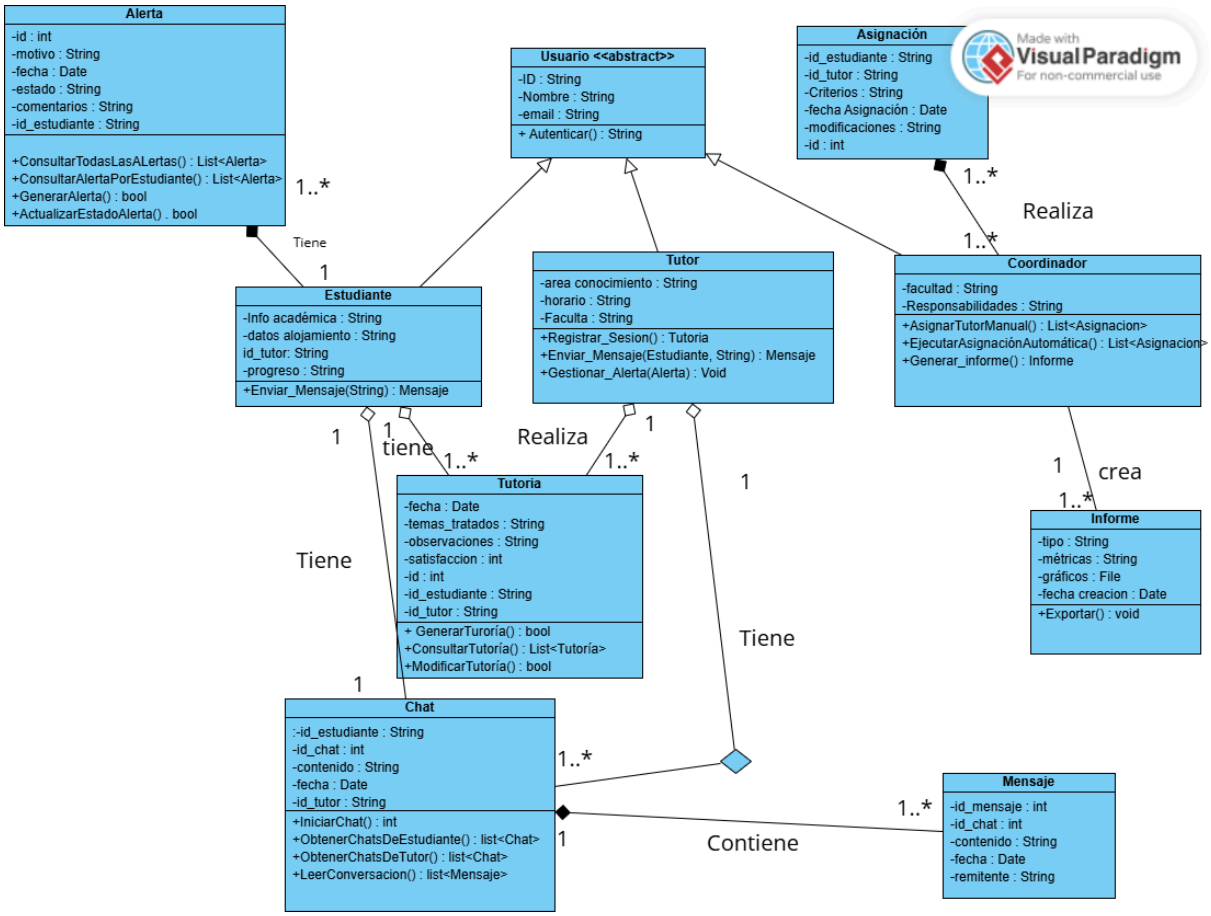
Clase Chat
Descripción: Representa un chat con mensajes entre alumno-tutor

	Atributos	Tipo	Descripción
-	id_chat	int	Identificador único del chat
-	id_estudiante	String	Estudiante del chat
-	id_tutor	String	Tutor del chat
-	fecha_creacion	Date	Fecha en que se inició el chat

Operaciones

	Operación	Tipo retorno	Descripción
+	IniciarChat	int	Devuelve todas las asignaciones de la base de datos
+	ObtenerChatsDeEstudiante	list<Chat>	Devuelve el tutor asociado a un estudiante
+	ObtenerChatsDeTutor	list<Chat>	Devuelve el historial de asignaciones que ha tenido un alumno
+	LeerConversacion	list<Mensaje>	Devuelve los alumnos asociados a un tutor

6.1 Diagrama de clases



7. Validación del Sistema

Matriz de requisitos frente a casos de uso

Identificador	CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07
RF-1	X						
RF-2					X		
RF-3				X			
RF-4				X		X	
RF-5			X				
RF-6		X					
RF-7						X	
RF-8		X				X	
RF-9							X
RF-10							X

Matriz de clases frente a casos de uso

CASO DE USO	Usuario	Estudiante	Tutor	Coordinador	Tutoría	Mensaje	Alerta	Informe	Asignación	Chat
CU-01	X	X	X	X						
CU-02		X	X		X					
CU-03		X	X			X				X
CU-04		X	X	X					X	
CU-05		X	X	X					X	
CU-06		X	X	X			X			
CU-07				X	X		X	X	X	

8. Diagramas de secuencias

Diagrama de gestión de alertas, autor: Adrián Sánchez Prieto

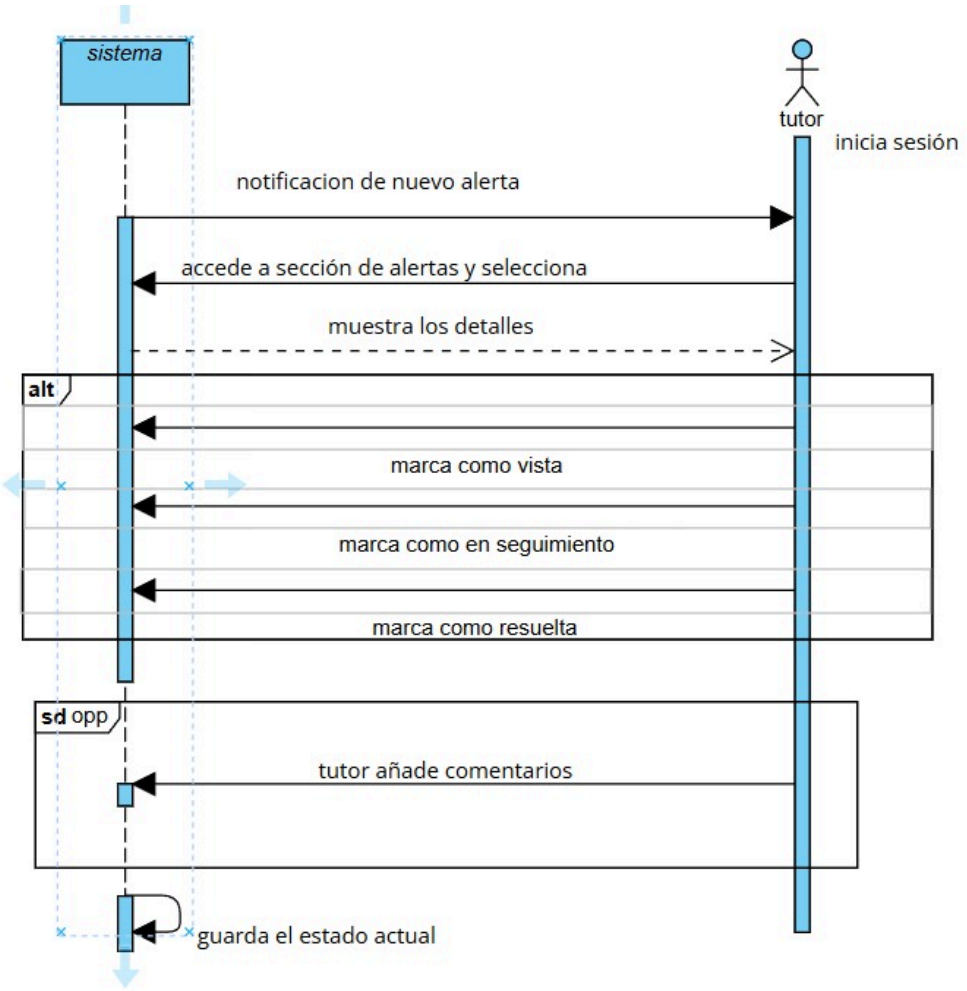


Diagrama de servicio de mensajería, autor: Manuel Téllez Domínguez

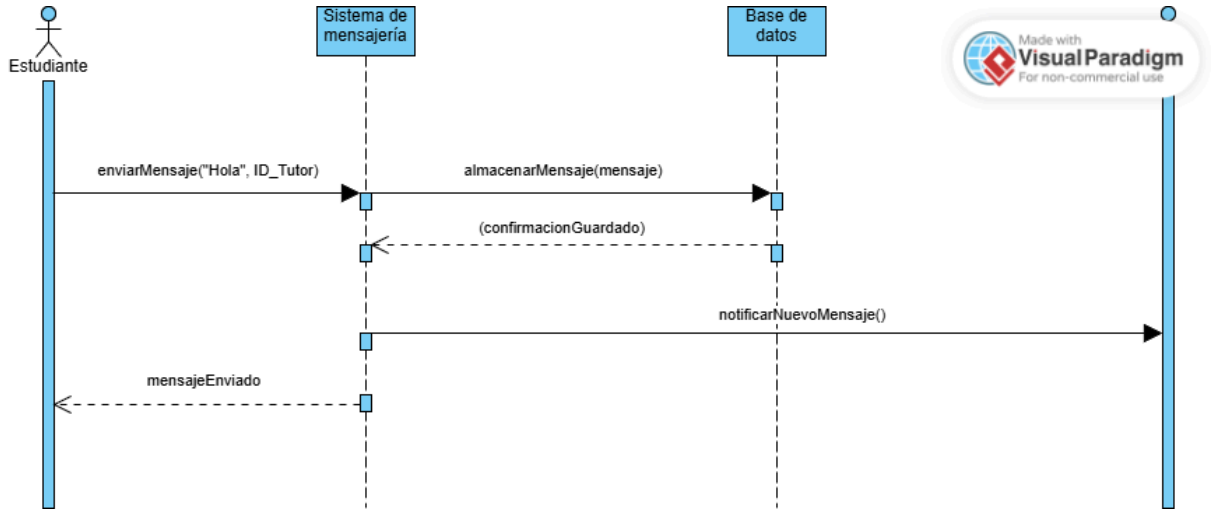
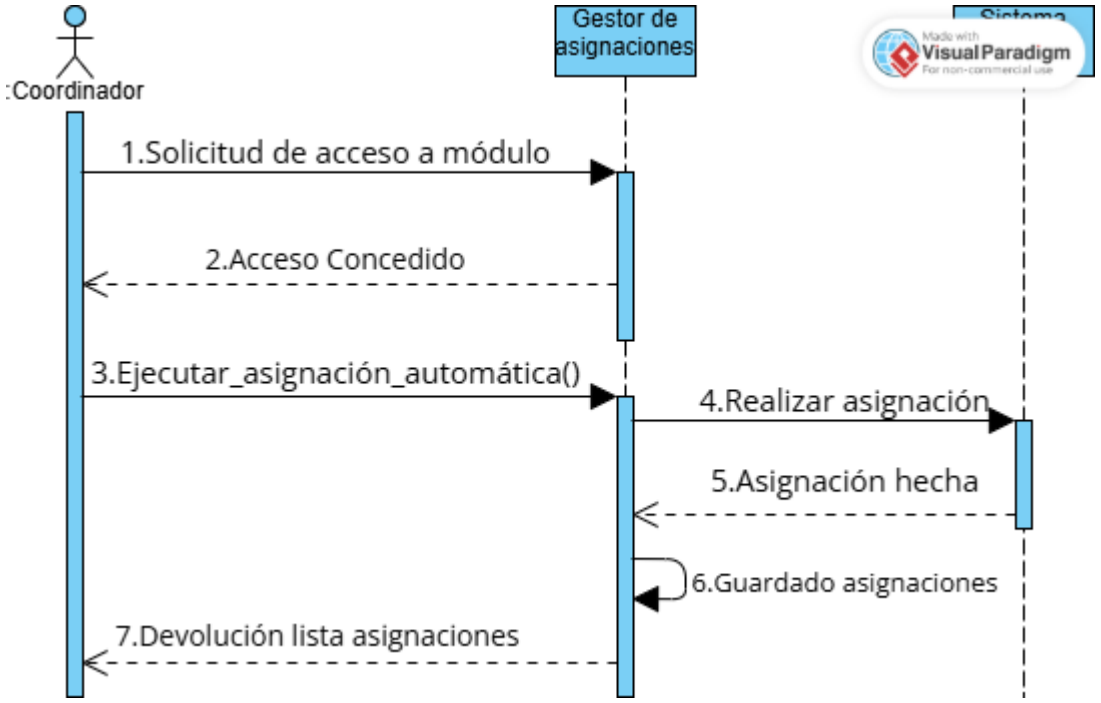


Diagrama de asignación automática, autor: Antonio Serrano Maldonado



9. Implementación del sistema

Entorno y herramientas de trabajo

IDE: Eclipse.
Lenguaje de Programación: C++
Repositorio GitHub:https://github.com/antonioserranom/SITAPI

Decisiones de Implementación

SGBD: SQLite
Interfaz Gráfica: Qt (C++ library)

Para la implementación del sistema se ha optado por una Arquitectura en Capas (Layered Architecture), lo que permite un desacoplamiento efectivo entre la lógica y los datos:

Capa de Repositorios: Es la encargada de la persistencia de datos y la comunicación directa con la base de datos.

Capa de Servicios: Contiene la lógica de negocio del sistema, actuando como intermediaria entre la interfaz y los datos.

Modelos de Dominio: Las clases modelo especificadas en el diseño actúan como representación en memoria de las entidades reales alojadas en la base de datos.

Finalmente, la interfaz gráfica presenta un diseño minimalista y funcional, comunicándose exclusivamente con la capa de servicios para garantizar un flujo de datos ordenado y un tiempo de respuesta óptimo.

El Algoritmo de Asignación Automática utilizado se asegura de que ningún tutor tenga más de 10 estudiantes asignados, repartiendo de forma equitativa los estudiantes entre los tutores, dando gran prioridad al hecho de que pertenezcan a la misma facultad. Asimismo, el algoritmo está perfectamente capacitado para convivir con bases de datos que ya contemplen asignaciones anteriores.

Metodología Scrum

El reparto de historias de usuario por sprint ha sido el siguiente.

Implementación 1 (HU-06)

Implementación 2 (HU-03 HU-05)

Implementación 3 (Front-end, conexión base de datos)

Implementación 4 (Pruebas y Documentación)

Para realizar las historias de usuario en los Sprint 1 y 2 se han codificado las clases modelo, repositorios y servicios con sus métodos. Luego a partir del Sprint 3 se codificó la interfaz gráfica y la configuración de la base de datos, para luego en el Sprint 4 pasar a las pruebas y documentación.

10. Pruebas del Sistema

Pruebas del módulo de mensajería

ID	UT-01	**Fecha**	16/12/25
Historia de usuario	HU-03		
Título de la prueba	Chat_EnviarMensajeCorrectamente		
Responsable de la prueba	Manuel Téllez Domínguez		
Descripción			
El objetivo es verificar que la función de envío de mensajes procesa correctamente la solicitud y persiste el nuevo mensaje en la base de datos sin errores de clave o formato.			
Prerrequisitos			
La base de datos de pruebas está inicializada y existe un registro de estudiante ('E01'), un registro de tutor ('T01'), y un registro de CHAT ya iniciado entre 'E01' y 'T01' con idChat = 1.			
Entradas			
Llamada al método de servicio: ChatService::EnviarMensaje(idChat: 1, idEmisor: 'E01', contenido: 'Hola Tutor, esto es un test')			
Acciones a realizar			
El servicio ChatService recibe los parámetros, luego, el ChatRepository ejecuta una sentencia sql insert en la tabla mensaje con el contenido y el emisor, verificando el éxito de la operación.			
Salida esperada			
El método debe retornar un valor true. La tabla MENSAJE debe contener una nueva fila con el contenido y el idEmisor correctos.			
Salida obtenida			
Aprobado (PASS)			
Observaciones			
se confirma la correcta interacción entre la capa de Servicio y la de Repositorio para la persistencia de datos.			

ID	UT-02	**Fecha**	16/12/25
Historia de usuario	HU-03		
Título de la prueba	Chat_RecuperarHistorial		
Responsable de la prueba	Manuel Téllez Domínguez		
Descripción			
El objetivo es verificar que la función de lectura de conversaciones recupera todos los mensajes asociados a un idChat específico y los devuelve ordenados cronológicamente.			
Prerrequisitos			
La base de datos de pruebas está inicializada y existe un chat activo que contiene al menos dos mensajes previamente guardados por por emisores distintos			
Entradas			
Llamada al método de servicio: ChatService::LeerConversacion(idChat: 1)			
Acciones a realizar			
El servicio ChatService solicita los mensajes al ChatRepository, el repositorio ejecuta una consulta select a la tabla mensaje filtrando por idChat y ordenador por fecha/ID ascendente, luego se mapean los resultados a objetos mensajes.			
Salida esperada			
El método debe retornar un std::vector<Mensaje> con una dimensión igual a 2. El primer mensaje debe ser el más antiguo.			
Salida obtenida			
Aprobado (PASS)			
Observaciones			
Se asegura que la funcionalidad de lectura de datos es correcta y que los mensajes se presentan en la interfaz de usuario en el orden lógico de la conversación.			

Pruebas del módulo de asignación automática

ID	UT-03	**Fecha**	17/12/25
Historia de usuario	HU-05		
Título de la prueba	Asignacion_Automatica_PrioridadAfinidad		
Responsable de la prueba	Antonio Serrano Maldonado		
Descripción			
Validar que el algoritmo identifica y prioriza al tutor cuya área de conocimiento coincide con la información académica del estudiante			
Prerrequisitos			
La base de datos de pruebas está inicializada y existen dos tutores, Tutor A (área: IA) y Tutor B (área: Redes) y un estudiante con información académica en “IA”.			
Entradas			
Llamada al método de servicio: AssignmentService::RealizarAsignacionAutomatica()			
Acciones a realizar			
El servicio recupera la lista de tutores y estudiantes y calcula la puntuación comparando el área del tutor con la info del alumno, el tutor A recibe 1000 puntos por coincidencia de intereses y se ejecuta la asignación			
Salida esperada			
El método debe retornar true y el estudiante debe quedar vinculado al ID del Tutor A			
Salida obtenida			
Aprobado (PASS)			
Observaciones			
La prueba confirma que el peso de la afinidad funciona correctamente sobre el resto de variables del algoritmo.			

ID	UT-04	**Fecha**	17/12/25
Historia de usuario	HU-05		
Título de la prueba	Asignacion_Automatica_BalanceoCarga		
Responsable de la prueba	Antonio Serrano Maldonado		
Descripción			
Verificar que, ante áreas de conocimiento genéricas o iguales, el sistema asigna al estudiante al tutor con menor número de alumnos asignados.			
Prerrequisitos			
La base de datos de pruebas está inicializada y existen dos tutores, Tutor A con 5 alumnos asignados y Tutor B con ninguno y un estudiante nuevo sin tutor.			
Entradas			
Llamada al método de servicio: AssignmentService::RealizarAsignacionAutomatica()			
Acciones a realizar			
El algoritmo calcula la carga actual de cada tutor, penaliza al tutor A con -50 alumnos y el tutor B mantiene una puntuación de 0, luego el estudiante se asigna al tutor con mayor puntuación.			
Salida esperada			
El método debe retornar true y el estudiante debe quedar vinculado al ID del Tutor B			
Salida obtenida			
Aprobado (PASS)			
Observaciones			
Se valida que el factor de balanceo de carga evita la saturación de tutores específicos cuando no hay una afinidad académica excluyente.			

pruebas del módulo de alertas

ID	UT-05	**Fecha**	18/12/25
Historia de usuario	HU-06		
Título de la prueba	Alerta_GenerarNueva		
Responsable de la prueba	Adrián Sánchez Prieto		
Descripción			
Verificar que el sistema es capaz de registrar una alerta de seguimiento para un estudiante específico, almacenando correctamente el motivo y asignando el estado inicial por defecto.			
Prerrequisitos			
La base de datos de pruebas está inicializada y hay un estudiante con Id “E01”,			
Entradas			
Llamada al método de servicio: AlertService::GenerarAlerta(idEstudiante: 'E01', motivo: 'Faltas de asistencia reiteradas')			
Acciones a realizar			
El servicio recibe el Id del alumno y el motivo, el repositorio, luego, inserta una nueva fila en la tabla alerta vinculada al estudiante y se asigna automáticamente el estado Pendiente si no se especifica otro.			
Salida esperada			
El método debe retornar true. Al consultar las alertas del estudiante 'E01', debe aparecer un registro con el motivo 'Faltas de asistencia reiteradas' y estado 'Pendiente'			
Salida obtenida			
Aprobado (PASS)			
Observaciones			
Se confirma que la comunicación entre la capa de servicio de alertas y la persistencia en base de datos funciona correctamente para la creación de registros.			

ID	UT-06	**Fecha**	18/12/25
Historia de usuario	HU-06		
Título de la prueba	Alerta_ActualizarEstado		
Responsable de la prueba	Adrián Sánchez Prieto		
Descripción			
Validar que un tutor puede cambiar el estado de una alerta existente (por ejemplo, de 'Pendiente' a 'Resuelta') y añadir comentarios de seguimiento.			
Prerrequisitos			
La base de datos de pruebas está inicializada y existe una alerta registrada con Id = 1, cuyo estado inicial es pendiente, asociada a un estudiante.			
Entradas			
Llamada al método de servicio: AlertService::ActualizarEstadoAlerta(idAlerta: 1, nuevoEstado: 'Resuelta', comentarios: 'Se ha mantenido una reunión presencial con el alumno')			
Acciones a realizar			
El servicio localiza la alerta por su Id y se ejecuta una sentencia sql update para modificar los campos estado y observaciones, luego el repositorio confirma la actualización de la fila.			
Salida esperada			
El método debe retornar true. Al recuperar la alerta con ID 1, el campo estado debe ser 'Resuelta' y el campo comentarios debe contener el texto introducido.			
Salida obtenida			
Aprobado (PASS)			
Observaciones			
Esta prueba asegura la integridad del flujo de gestión de alertas, permitiendo el cierre de incidencias y la trazabilidad del seguimiento realizado.			

informe de errores

ID Prueba	Acción	Error	Solución
UT-01	Enviar mensaje	GetContenido falla, pues el id estaba en el lugar donde debía estar el mensaje	Las asignaciones en la base de datos se hacían mal, corregir el repositorio
UT-02	Recuperar chat	GetContenido falla, pues el id estaba en el lugar donde debía estar el mensaje	Las asignaciones en la base de datos se hacían mal, corregir el repositorio
UT-03		Sin errores	
UT-04	Asignación automática entre alumno y tutor sin intereses mutuos	El estudiante se asocia al tutor con más tutorados en lugar de al mas libre.	Pequeño error tipográfico en el algoritmo.
UT-05		Sin errores	
UT-06		Sin errores	

11. Uso de IA Generativa

Manuales: Usada para la estructuración y maquetación de los documentos, así como para la corrección de posibles incongruencias en ciertos textos, pasando como prompt el documento a revisar.

Implementación: Usada para generar código de interfaz gráfica debido a nuestro desconocimiento de las técnicas apropiadas, además ha ayudado con la generación de código sql para inserción de datos de ejemplo y la creación de makefiles.

En ambos casos el chatbot de IA usado ha sido Gemini AI.

12. Bibliografía

ISO/IEC. (2023). *ISO/IEC 14882:2023 Programming languages — C++*. International Organization for Standardization.

Hipp, R., et al. (2024). *SQLite Documentation*. Recuperado de https://www.sqlite.org/docs.html