Sistema de tutorización manual UCO

Manual de Usuario

Versión: 0100

Fecha: 15/12/2025

Versión 1.0

Grupo 201

David Llagas Rodríguez
Julián Lara Peso
Víctor Jesús Castañeda Morales

Como expertos ingenieros en desarrollo de software, el rectorado de la Universidad de Córdoba nos ha solicitado la elaboración de un programa informático que sea capaz de asignar automáticamente tutores a los alumnos de la universidad y que estos puedan ser modificables según ciertos requisitos y preferencias.

Se nos exige un software fácil de utilizar y que permita la interacción fácil y sencilla entre el alumno y el tutor. Aparte este sistema debe de almacenar de manera segura y fiable los datos, solicitudes y peticiones de los alumnos a los profesores.

1. ESPECIFICACIÓN DE REQUISITOS

1.1. Requisitos funcionales.

Funcionalidades que debe permitir el software al usuario:

RF1- El sistema debe asignar a un alumno con un tutor desde el primer día
RF2- El sistema debe asignar tutores de forma manual o automática, de forma que a un tutor se le asignará automáticamente un tutor de su nacionalidad y/o nacionalidad.
RF3- El sistema debe proporcionar un chat, independiente al email, para el contacto entre tutor y alumno.
RF4- El sistema debe tener una lista con un registro de las reuniones.
RF5- El sistema debe aportar un sistema de avisos al estudiantado.
RF6- Tutor debe poder ver la información de sus tutorizados.
RF7- El sistema debe tener un registro sobre las reuniones, donde se generarán informes sobre lo acordado en estas.
RF8- El coordinador debe ser capaz de crear y visualizar registros de las

actividades y de los alumnos.

RF9- El sistema debe de mandar los datos de inicio de sesión a la base de datos de la UCO

RF10- Los alumnos de tercer y cuarto año deber tener la opción de opositar como tutor.

RF11- El sistema deberá llevar un registro detallado sobre el alumno.

RF12 - El alumno podrá pedir una reunión con su respectivo tutor para tratar temas de importancia.

1.2. Requisitos no funcionales.

Condiciones de calidad que debe cumplir el software:

RNF1- Software único integrado con todas las funcionalidades en un mismo espacio.

RNF2- Debe de ser un software amigable con el usuario y sencillo de utilizar.

RNF3- El modelo debe de ser responsive (adaptado a móviles, tabletas y PCs).

RNF4- El software debe de funcionar durante las 24 horas del día sin interrupciones todos los días del año.

RNF5- Si el usuario entra al software 5 minutos después de su último inicio de sesión, deberá introducir de nuevo sus credenciales (por seguridad).

RNF6- El tiempo de respuesta del sistema ante cualquier evento debe de ser MÁXIMO 2 segundos.

RNF7- EL sistema debe de soportar un gran flujo de estudiantes (aproximadamente 1,5 veces los usuarios de la UCO para asegurar su correcto funcionamiento ante cualquier número de estudiantes).

RNF8- El sistema debe de realizar copias de seguridad del sistema cada 24 horas.

RNF9- Debe de haber un borrado de los chats de los estudiantes que han terminado la carrera.

1.3. Requisitos de información.

Datos que el sistema debe almacenar y gestionar de forma estructurada:

RI1-Datos de usuario como su rol (tutor/tutorado), correo electrónico, contraseña.
RI2-Datos personales de los tutores-tutorados de la BBDD de la UCO, que sean privados y únicamente puedan acceder a ellos los administradores.
RI3-Registro de chats entre tutores y tutorados.
RI4-Registro de incidencias y avisos
RI5-Informes generales sobre el número de tutores-tutorados, numero de incidencias, avisos, alertas.

2. HISTORIAS DE USUARIO

En este documento nos centraremos en una descripción básica de las historias de usuario que se nos requieren en nuestro software para la Universidad de Córdoba

Usuario UCO:

H1. COMO usuario/a de la Universidad de Córdoba, QUIERO poder iniciar sesión con mis credenciales de la UCO, PARA que nadie pueda reemplazar mi identidad.
El proceso de inicio de sesión o de creación de cuenta, se llevará siempre a cabo con las credenciales de la UCO de cada usuario. Esto evita en gran medida que se olviden usuarios o contraseñas y, en caso de que ocurra, se

pueden obtener fácilmente en la secretaría. Todos los usuarios, dependientemente de que sean alumnos, tutores o el coordinador, tienen sus propias credenciales de la UCO. Estas credenciales no cambian durante todo el tiempo que este usuario este en la UCO y la cuenta se acabará eliminando una vez el usuario deje de pertenecer a la Universidad de Córdoba.
Responsable: Víctor Castañeda
Criterios de validación: • El sistema debe conectarse con la base de datos de la UCO. • Si el usuario introduce un “login” erróneo, recibirá un aviso de este error. • Al introducir los datos correctos, te llevara a la página principal del sistema.
Prioridad: Alta
Estimación en días: 2 días
Sprint: 2

H2. COMO usuario/a de la Universidad de Córdoba, QUIERO poder crear un chat con otro usuario de mi interés, PARA tener una comunicación fluida en caso de que sea requerida.

Los usuarios tendrán la posibilidad de crear un chat directo dentro del propio programa con otra persona de su interés (el tutor solo podrá con un alumno y viceversa), exento de la utilización del correo electrónico u otro medio de comunicación externo al software, por el que podrán contactar con el otro usuario de forma rápida y sencilla.

El chat y su contenido se mantendrá guardado en el chat en caso de que se necesite consultar algo de este. Por ello, los mensajes enviados anteriormente, se mantendrán en el chat y podrán ser visualizados.

Responsable: Julián Lara

Prioridad: Media

Criterios de validación:

- El sistema devolverá error si el chat no es accesible.
- Cada mensaje estará capado a 500 caracteres, para una mejor gestión en la base de datos.
- El sistema deberá devolver un error si el mensaje no ha podido ser mandado

Estimación en días: 7 días
Sprint: 4

Coordinador UCO:

H3. COMO coordinador/a de la Universidad de Córdoba, QUIERO acceder a las estadísticas, PARA poder hacer revisiones y ver el funcionamiento del sistema.
El vicerrector debe de poseer de una opción en la que se le muestre por pantalla las estadísticas más relevantes del sistema. Entre estas se debe de encontrar el número de tutores (con opciones para ver la especialización de cada tutor, si este es un profesor o un alumno...), el número de alumnos y sus grados, el número de alumnos por tutor, reuniones registradas, promedio de reuniones por alumno...
Responsable: David Llagas
Criterios de validación: • El sistema debe mostrar toda la información de todos los tutores y alumnos.

- Si no existen alumnos o errores, debe mostrar un error correspondiente al fallo generado
- Si el coordinador quiere hacer una revisión, podrá modificar datos y hacer consultas en el sistema.

Prioridad: Media

Estimación en días: 7 días

Sprint: 3

H4. COMO coordinador/a de la Universidad de Córdoba, QUIERO poder generar informes periódicos, PARA verificar el uso del software y analizar tendencias.

Se añadirá la posibilidad de generar informes o archivos descargables en los que se incluyan las estadísticas descritas anteriormente y otras como grados en los que más tutorías se piden o grados con más alumnos tutorizados, para efectuar cambios y tomar decisiones estratégicas. 2

Dicho informe se podrá personalizar de manera que el vicerrector podrá elegir el periodo de tiempo sobre el cuál quiere recibir dicho informe y las estadísticas que aparecen en este.

Responsable: Julián Lara
Criterios de validación: • El sistema debe mostrar toda la información relacionada con el sistema. • Si el profesor desea, debe ser capaz de descargarse la información en un archivo CSV con toda la información seleccionada. • Si el coordinador selecciona una fecha sin informes o incoherente, el sistema devolverá un error
Prioridad: Baja
Estimación en días: 3 días
Sprint: 5

Alumno UCO:

H5. COMO alumno/a de la Universidad de Córdoba, QUIERO tener la posibilidad de ver quién es mi tutor, PARA poder mantenerme en contacto.

El alumno debe de tener alguna manera de poder saber quién es su tutor para poder ponerse en contacto con él, aparte de poder contactar con él en persona si fuera necesario. Debe de poder acceder a su información personal, aparte de su área de especialización y dirección de oficina física por si fuera necesario ir en persona.
Responsable: David Llagas
Prioridad: Media
Criterios de validación: • El sistema debe mostrar la información de el tutor asociado al alumno.
Estimación en días: 1
Sprint: 3

H6. COMO alumno/a de la Universidad de Córdoba, QUIERO poder cerrar sesión o que la sesión se cierre a los 5 minutos de inactividad, PARA proteger mi información personal.

El software debe de estar diseñado para que, a los 5 minutos de inactividad, la sesión se cierre automáticamente. Esto se hace para evitar que, en caso de que el usuario deje su cuenta abierta en un dispositivo que no sea suyo, otro usuario no pueda obtener datos personales o encontrar información alumno-tutor. Pese a esto, el usuario tiene un botón de cerrar sesión en el momento en el que finalice su consulta dentro del programa.
Responsable: Víctor Castañeda
Prioridad: Baja
Criterios de validación: • El sistema devolverá un mensaje de que se va a desconectar al usuario y cuando este sea desconectado
Estimación en días: 1 día
Sprint: 5

H7. COMO alumno/a de la Universidad de Córdoba, QUIERO que se me asigne automáticamente un tutor de mi misma

nacionalidad y ámbito de estudio , PARA facilitar el proceso de tutorización.
El sistema le asignara al alumno un tutor automáticamente, el criterio para dicha asignación se basara en la nacionalidad del alumno, además de también tomar en cuenta los estudios que cursa dicho alumno. Esto se hace para facilitar el proceso de tutorización.
Responsable: David Llagas
Prioridad: Media
Criterios de validación: • El sistema devolverá un aviso si no hay ningún tutor con esas características y le asignará uno de forma aleatoria. • Si el tutor elegido automáticamente está muy demandado frente a otro (2 estudiantes de diferencia) con similitudes, se le asignará a este otro para proporcionar un buen reparto del trabajo
Estimación en días: 5 días
Sprint: 2

Tutor UCO:

H8. COMO tutor/a de la Universidad de Córdoba, QUIERO poder enviar avisos sobre problemas académicos a mis tutorados, PARA informarles de situaciones importantes que requieran su atención.

El sistema permitirá al tutor enviar notificaciones o alertas (p.ej., bajo rendimiento, faltas, plazos importantes) que queden registradas para el alumno.

Con esto se busca mantener informados a los alumnos en cualquier tipo de situación.

Responsable: David Llagas

Prioridad: Baja

Criterios de validación:

- El sistema permitirá enviar alertas sobre incidencias de parte de los tutores a los alumnos tutorados.
- En caso de error el sistema mostrara que no se pudo enviar el aviso y dará la posibilidad de reenviarlo.

Estimación en días: 3

Sprint: 4

H9. COMO tutor/a de la Universidad de Córdoba, QUIERO poder acceder a la información de mis alumnos tutorados, PARA dar un mejor apoyo.
El software debe de estar diseñado para que los tutores puedan acceder a la información de los alumnos tutorados, esto se hace para tener la posibilidad de dar una ayuda más exhaustiva, además de ayudar al tutor a acercarse y conocer más sobre su alumno tutorado.
Responsable: Julián Lara
Prioridad: Baja
Criterios de validación: • Si el tutor no tiene ningún alumno asignado, el sistema devolverá un error al usuario.
Estimación en días: 3

Sprint: 3

H10. COMO tutor/a de la Universidad de Córdoba, QUIERO registrar las reuniones con mis alumnos, PARA llevar un seguimiento detallado de su progreso y de los temas tratados.

El tutor podrá crear un registro por cada reunión mantenida con un alumno.

Este registro permitirá anotar la fecha, los temas tratados y los acuerdos alcanzados, y quedará guardado en el historial del alumno para futuras consultas.

Responsable: David Llagas

Prioridad: Media

Criterios de validación:

- El sistema deberá permitir anotar los puntos tratados en la reunión, además de la fecha en la que esta tuvo lugar.
- En caso de fallo al guardar el registro de la reunión se mostrara un mensaje de error que dará la posibilidad de volver a guardar la misma información o sobrescribirla.

Estimación en días: 3
Sprint: 2

H11. COMO alumno de la Universidad de Córdoba, QUIERO poder solicitar tutorías con mi tutor PARA resolver situaciones o problemas académicos
El programa debe de tener una opcionalidad para solicitar tutorías de forma sencilla entre el alumno y su tutor en caso de que sean necesarias. Para ejecutar estas solicitudes, se deberá de indicar la fecha y hora ideal para el alumno y el motivo de dicha tutoría.
Responsable: Víctor Jesús Castañeda Morales
Prioridad: Alta
Criterios de validación: • La fecha de la tutoría debe de ser mínimo un día posterior al momento en el que esta es solicitada, para que el tutor tenga tiempo de cancelar, aceptar o modificar la tutoría.
Estimación en días: 2

Sprint: 4

3. CASOS DE USO

A continuación, se detallan los casos de uso principales del sistema, describiendo la interacción paso a paso entre los actores y el software para alcanzar un objetivo específico.

Nombre: Iniciar sesión en el sistema
Identificador: CU-01.
Objetivo: El usuario (Alumno, Profesor o Coordinador) quiere acceder al sistema con sus credenciales oficiales.
Contexto: El usuario se encuentra en la pantalla de acceso al sistema y dispone de credenciales válidas de la UCO.
Actor principal: Usuario UCO (Alumno, Tutor, Coordinador).
Escenario principal:

1. El usuario introduce su nombre de usuario (credencial UCO) y contraseña anteriormente proporcionadas por la Universidad de Córdoba.

2. El sistema valida las credenciales contra el servicio de autenticación de la UCO, mediante una conezión externa a la base de datos de la UCO.

3. El sistema concede el acceso y muestra la interfaz principal correspondiente al rol.

Extensiones:

5a. (Credenciales incorrectas) El sistema rechaza el acceso y muestra un mensaje de error: "Usuario o contraseña incorrectos".

Nombre: Solicitar Reunión con Tutor

Identificador: CU-02.

Objetivo: El alumno quiere solicitar y agendar una reunión con su tutor asignado.

Contexto: El alumno ha iniciado sesión en el sistema (Ver CU-01) y tiene un tutor asignado visible en la plataforma.

Actor principal: Alumno.

Escenario principal:

1. El alumno pulsa la opción "Solicitar reunión". Cuando la pulsa, le aparece la disponibilidad de su tutor y también tiene campos de texto en los que pondrá el motivo de la reunión.

2.Elije una hora y fecha para pedir la reunión y rellena el motivo de esta. Pulsa “Solicitar Reunión” y la reunión se almacena en la base de datos y el tutor es notificado de que uno de sus alumnos le ha solicitado la reunión.

Extensiones:

2a. (Fecha no disponible): El sistema muestra una notificación cuando el tutor no tiene horas ni fechas disponibles. El sistema muestra el mensaje “El tutor no tiene fechas disponibles”.

2b.(Modificar reunión): El alumno tiene la posibilidad de modificar una reunión ya creada. El sistema mostrará las horas y fechas disponibles por el profesor y el alumno podrá elegir una de las fechas disponibles y pulsar el botón “Cambiar reunión” y la reunión cambia automáticamente su fecha y hora y el tutor es notificado de este cambio.

Nombre: Registrar Datos de la Reunión

Identificador: CU-03.

Objetivo: El profesor quiere guardar un registro de la reunión mantenida con uno de sus alumnos tutorados.
Contexto: El profesor ha iniciado sesión en el sistema (Ver CU-01).
Actor principal: Tutor
Escenario principal: 1. Tras tener una reunión con un alumno que el profesor tutorice, selecciona la opción "Registrar nueva reunión". 2. En esta aparece un desplegable que le permite registrar todos los datos de la reunión (alumno, fecha, hora, motivo de la reunión...). El profesor rellena toda la información necesaria y pulsa “Guardar” 3. El registro se almacena en la base de datos, clasificado según la fecha y hora, el alumno y el tutor que realiza esta tutoría.

Nombre: Uso de chat de mensajes
Identificador: CU-04.

Objetivo: El usuario quiere enviar un mensaje de texto o un archivo a su tutor o alumno tutorado.
Contexto: El usuario ha iniciado sesión (Ver CU-01) y tiene una conversación iniciada con el destinatario.
Actor principal: Tutor, Alumno.
Escenario principal: 1. El usuario accede al chat creado. En caso de que se hayan enviado mensajes entre ambos usuarios previamente, al entrar aparecen todos los mensajes enviados anteriormente. 2. El usuario escribe el mensaje que quiere enviar al destinatario, y pulsa el botón de “Enviar”. Con esto, el mensaje se registra en la base de datos, almacenando junto a él la fecha, hora, destinatario… 3. El mensaje enviado aparece en la pantalla del destinatario si está dentro de la conversación y tiene la posibilidad de responder. Estos mensajes se guardan en el chat entre ambos usuarios en caso de que en un futuro vuelvan a entrar a la conversación.

Nombre: Generar registros de estadísticas del programa

Identificador: CU-05.
Objetivo: El coordinador quiere consultar las estadísticas principales sobre el uso del programa de tutorías y generar informes descargables de estas.
Contexto: El coordinador ha iniciado sesión en el sistema (Ver CU-01).
Actor principal: Coordinador UCO
Escenario principal: 1.El coordinador entra en una pestaña en la que pueden visualizar todas las estadísticas más importantes del uso del programa de forma organizada. 2. Puede filtrar estos datos según el grado de estudio, edad de los estudiantes y otros datos significativos. Aparecen los datos de manera general, pero se pueden ver individualmente por estudiante. 3.El usuario puede generar un informe descargable de los datos que le interesen y los puede filtrar por los mismos filtros explicados anteriormente.
Extensiones: 3a. Se descarga un informe con las estadísticas solicitadas.

Nombre: Mostrar información asociada a un alumno

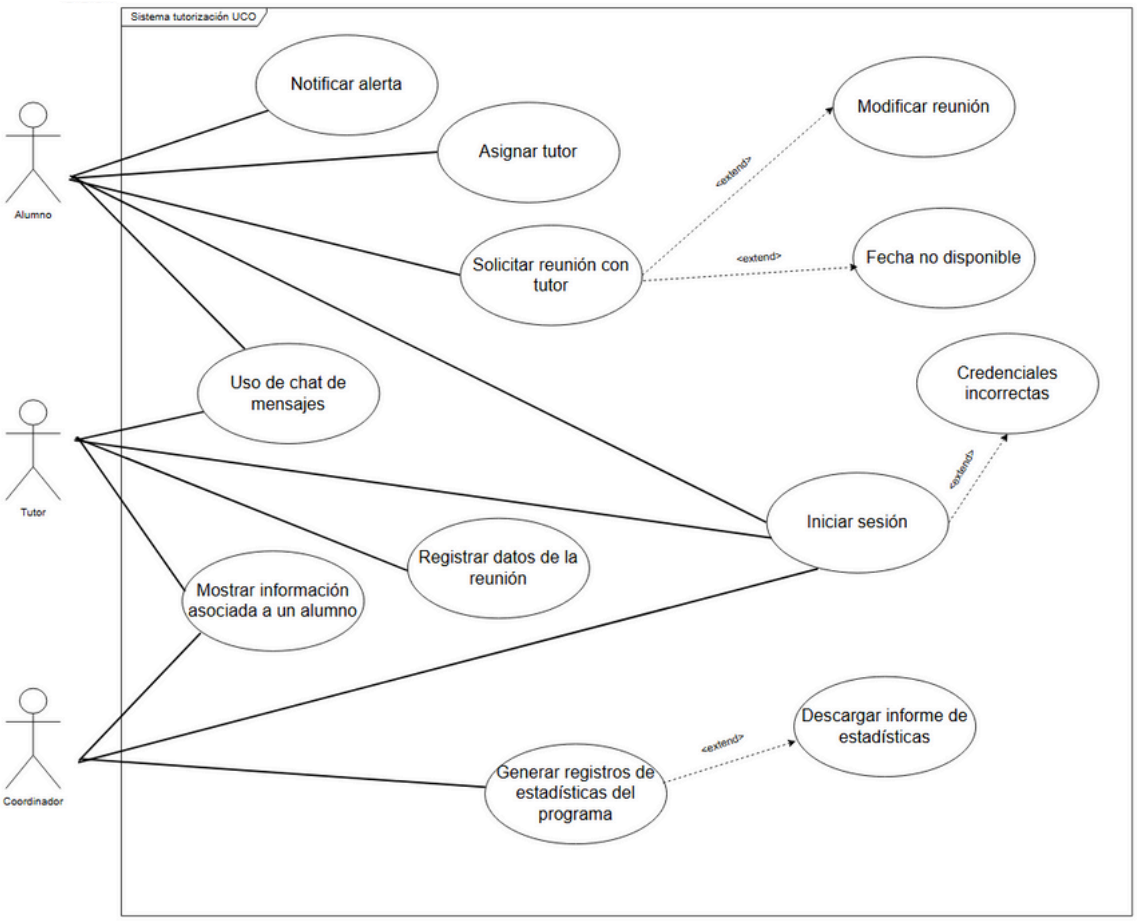
Identificador: CU-06.
Objetivo: Mostrar la información solicitada sobre un alumno.
Contexto: Se desea ver la información de un alumno.
Actor principal: Coordinador, tutor.
Escenario principal: 1. El usuario selecciona un alumno específico del que esté autorizado para ver su información. Con dicho alumno seleccionado, se pulsa “Ver información”. 2.Aparece en pantalla información relevante del alumno. Comienza identificando al alumno (credencial de usuario, fecha nacimiento, grado que cursa, asignaturas…) y posteriormente aparece la información de las reuniones en las que ha participado (cantidad, informe de cada reunión…)

Nombre: Asignación tutor
Identificador: CU-07.
Objetivo: Asignar un tutor disponible a un alumno que lo haya solicitado basándose en unos criterios previos.

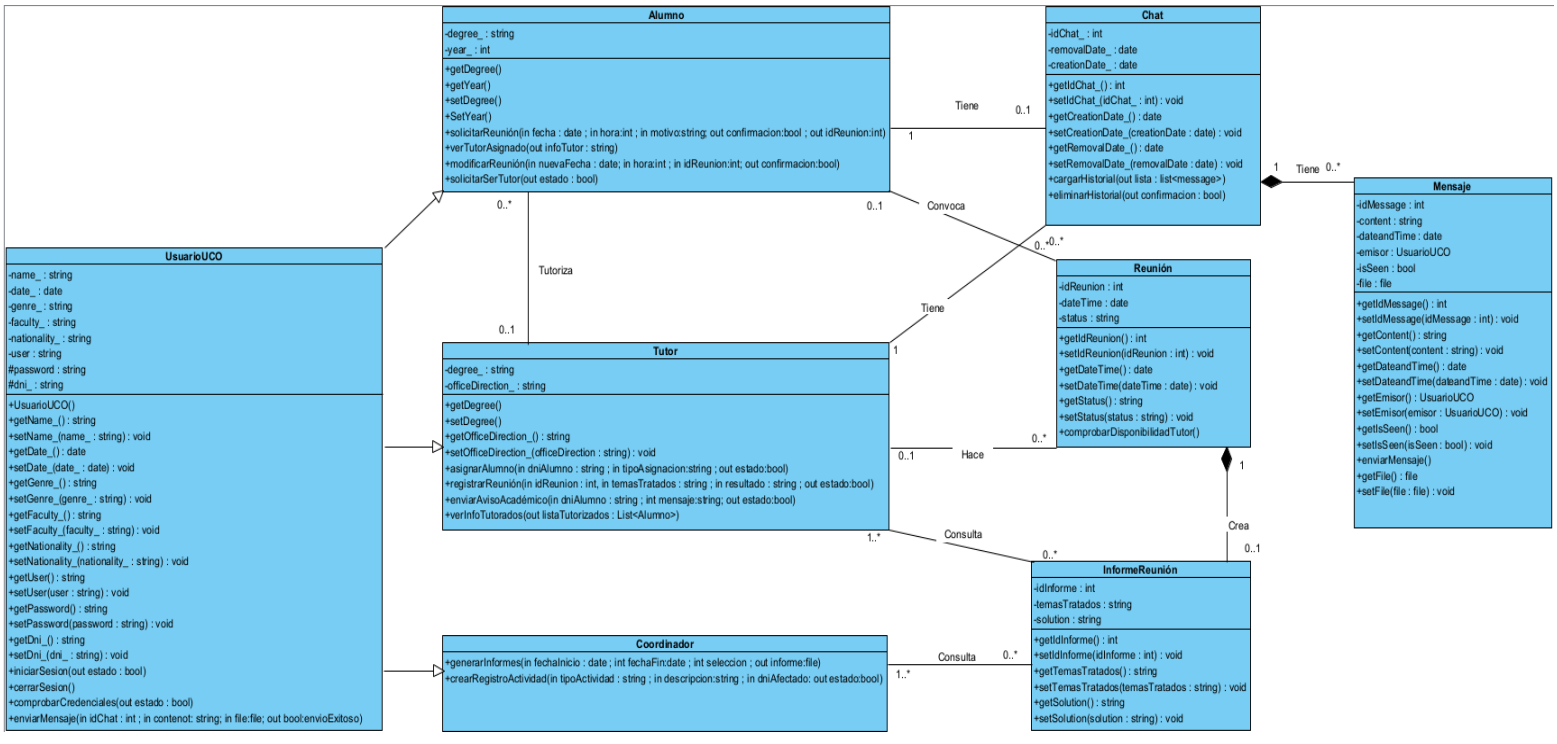
Contexto: El alumno desea solicitar un tutor.
Actor principal: Alumno.
Escenario principal: 1. El sistema aplica un tutor automáticamente al alumno. Este se asigna teniendo en cuenta el grado del alumno, idioma que hable y más aspectos para que el tutor sea lo más acorde posible con el alumno. Se verifica la disponibilidad del tutor, si uno coincide mas, pero tiene 30 alumnos y otro coincide menos pero tiene 0 alumnos, se asignará el segundo. 2. El alumno puede solicitar el cambio de tutor en caso de que se aporte un motivo de peso que se debe de indicar. A la misma vez si el mismo tutor quiere que un alumno tutorizado por él mismo tenga otro tutor, lo puede solicitar también. Todo esto se llevará a cabo mediante una opción en nuestro programa que permita solicitar el cambio y explicar el motivo.

Nombre: Notificar alerta
Identificador: CU-08.
Objetivo: Notificar al alumno en casos de problemas académicos o incidencias.

Contexto: El sistema notifica al alumno en caso de que ocurra algún problema.
Actor principal: Alumno.
Escenario principal: 1. El usuario elige si desea recibir notificaciones de incidencias de parte del sistema o enviadas por su tutor. 2. El sistema detecta una incidencia o el tutor hace una alerta para el estudiante tutorado debido a los datos recogidos por la aplicación y los registros de los informes. 3. El sistema notifica al usuario enviándole una alerta a su dispositivo móvil personalizada según el tipo de aviso realizado por el tutor.


Sistema tutorización UCO
Notificar alerta
Asignar tutor
Modificar reunión
Solicitar reunión con tutor
Fecha no disponible
Uso de chat de mensajes
Credenciales incorrectas
Iniciar sesión
Registrar datos de la reunión
Mostrar información asociada a un alumno
Descargar informe de estadísticas
Generar registros de estadísticas del programa
<extend>
Alumno
Tutor
Coordinador

4.CLASES


Alumno
-degree_ : string
-year_ : int
+getDegree()
+getYear()
+setDegree()
+SetYear()
+solicitarReunión(in fecha : date ; in hora:int ; in motivo:string; out confirmacion:bool ; out idReunion:int)
+verTutorAsignado(out infoTutor : string)
+modificarReunión(in nuevaFecha : date; in hora:int ; in idReunion:int; out confirmacion:bool)
+solicitarSerTutor(out estado : bool)
Chat
-idChat_ : int
-removalDate_ : date
-creationDate_ : date
+getIdChat_() : int
+setIdChat_(idChat_ : int) : void
+getCreationDate_() : date
+setCreationDate_(creationDate : date) : void
+getRemovalDate_() : date
+setRemovalDate_(removalDate : date) : void
+cargarHistorial(out lista : list<message>)
+eliminarHistorial(out confirmacion : bool)
Mensaje
-idMessage : int
-content : string
-dateandTime : date
-emisor : UsuarioUCO
-isSeen : bool
-file : file
+getIdMessage() : int
+setIdMessage(idMessage : int) : void
+getContent() : string
+setContent(content : string) : void
+getDateandTime() : date
+setDateandTime(dateandTime : date) : void
+getEmisor() : UsuarioUCO
+setEmisor(emisor : UsuarioUCO) : void
+getIsSeen() : bool
+setIsSeen(isSeen : bool) : void
+enviarMensaje()
+getFile() : file
+setFile(file : file) : void
Tiene
Convoca
Tutoriza
Reunión
-idReunion : int
-dateTime : date
-status : string
+getIdReunion() : int
+setIdReunion(idReunion : int) : void
+getDateTime() : date
+setDateTime(dateTime : date) : void
+getStatus() : string
+setStatus(status : string) : void
+comprobarDisponibilidadTutor()
UsuarioUCO
-name_ : string
-date_ : date
-genre_ : string
-faculty_ : string
-nationality_ : string
-user : string
#password : string
#dni_ : string
+UsuarioUCO()
+getName_() : string
+setName_(name_ : string) : void
+getDate_() : date
+setDate_(date_ : date) : void
+getGenre_() : string
+setGenre_(genre_ : string) : void
+getFaculty_() : string
+setFaculty_(faculty_ : string) : void
+getNationality_() : string
+setNationality_(nationality_ : string) : void
+getUser() : string
+setUser(user : string) : void
+getPassword() : string
+setPassword(password : string) : void
+getDni_() : string
+setDni_(dni_ : string) : void
+iniciarSesion(out estado : bool)
+cerrarSesion()
+comprobarCredenciales(out estado : bool)
+enviarMensaje(in idChat : int ; in contenot: string; in file:file; out bool:envioExitoso)
Tutor
-degree_ : string
-officeDirection_ : string
+getDegree()
+setDegree()
+getOfficeDirection_() : string
+setOfficeDirection_(dfficeDirection : string) : void
+asignarAlumno(in dniAlumno : string ; in tipoAsignacion:string ; out estado:bool)
+registrarReunión(in idReunion : int, in temasTratados : string ; in resultado : string ; out estado:bool)
+enviarAvisoAcadémico(in dniAlumno : string ; int mensaje:string; out estado:bool)
+verInfoTutorados(out listaTutorizados : List<Alumno>)
Hace
Consulta
Crea
InformeReunión
-idInforme : int
-temasTratados : string
-solution : string
+getIdInforme() : int
+setIdInforme(idInforme : int) : void
+getTemasTratados() : string
+setTemasTratados(temasTratados : string) : void
+getSolution() : string
+setSolution(solution : string) : void
Coordinador
+generarInformes(in fechaInicio : date ; int fechaFin:date ; int seleccion ; out informe:file)
+crearRegistroActividad(in tipoActividad : string ; in descripcion:string ; in dniAfectado: out estado:bool)
Consulta

Clase: UsuarioUCO		
Clase padre en la que se reunen todos los posibles usuarios de la aplicación		
Atributos		
· + name_	String	Nombre del usuario
· + date_	Date	Fecha de nacimiento del usuario
· + genre_	String	Género del usuario
· + faculty_	String	Campo de conocimiento
· + nationality_	String	Nacionalidad del usuario
· + user_	String	Credencial de nombre de usuario

· # password_	String	Contraseña del usuario
Operaciones		
· +UsuarioUCO()	Void	Constructor usuario UCO
· +getName_()	String	Devuelve el nombre del usuario
· +setName_(name : string)	Void	Actualiza el nombre del usuario
· +name	String	Nombre a introducir
· +getDate_()	Date	Devuelve la fecha de nacimiento del usuario
· +setDate_(date : string)	Void	Actualiza la fecha de nacimiento del usuario
· +date	Date	Fecha de nacimiento a introducir

· +getGenre_()	String	Devuelve el género del usuario
· +setGenre_(genre : string)	Void	Actualiza el género del usuario
· +genre	String	Género a introducir
· +getFaculty_()	String	Devuelve el campo de conocimiento del usuario
· +setFaculty_(faculty : string)	Void	Actualiza el campo de conocimiento del usuario
· +faculty	String	Campo de conocimiento a introducir
· +getNationality_()	String	Devuelve la nacionalidad del usuario

· +setNationality_(nationality : string)	Void	Actualiza la nacionalidad del usuario
· +nationality	String	Campo de conocimiento a introducir
· +getUser_()	String	Devuelve la credencial del usuario
· +setUser_(user : string)	Void	Actualiza la credencial de usuario
· +user	String	Credencial de usuario a introducir
· +getPassword_()	String	Devuelve la contraseña del usuario
· +setPassord_(password : string)	Void	Actualiza la contraseña de usuario
· +password	String	Contraseña a introducir

· +getDni_()	String	Devuelve el DNI del usuario
· +setDni_(dni : string)	Void	Actualiza el DNI del usuario
· +dni	String	DNI de usuario a introducir
· +iniciarSesion(out estado : bool)	Bool	Devuelve si se ha iniciado sesión o no
· +estado	Bool	Confirma si ha sido posible
· +cerrarSesion()	Void	Cierra la cuenta del usuario
· +comprobarCredenciales(out estado : bool)	Bool	Devuelve si las credenciales son correctas o no
· +estado	Bool	Confirma si ha sido posible

Clase: Alumno		
Clase hija de "UsuarioUCO" en la que se encierran todos los alumnos de la Universidad de Córdoba		
Atributos		
· + degree_	String	Grado que estudia el alumno
· + year_	Date	Curso en el que está el alumno
Operaciones		
· +getDegree_()	String	Devuelve el grado que estudia el alumno
· +setDegree_(degree : string)	Void	Actualiza el grado que estudia el alumno

· +degree	String	Grado a introducir
· +getYear_()	Date	Devuelve el curso en el que está el alumno
· +setYear_(year : string)	Void	Actualiza el curso en el que está el alumno
· +year	Date	Curso a introducir
· +solicitarReunion(in fecha : date; in hora : int; in motivo : string; out confirmacion : bool; out idReunion : int)	Bool	Con esta función un alumno solicita una reunión
· +fecha	Date	Fecha solicitada para la reunión
· +hora	Int	Hora solicitada para la reunión
· +motivo	String	Por qué se solicita la reunión

· +confirmacion	Bool	Devuelve si se acepta o rechaza la reunión
· +idReunion	Int	Id de la reunión
· +verTutorAsignado(out infoTutor : string)	String	El alumno puede ver la información de su tutor
· +infoTutor	String	Información del tutor
· +modificarReunion(in nuevaFecha : date ; in hora ; in idReunion ; out confirmacion : bool)	Bool	El alumno solicita modificar una reunión ya existente
· +nuevaFecha	Date	Fecha por la que se quiere cambiar la existente
· +hora	Int	Hora a la que se quiere poner la nueva reunión
· +idReunion	Int	Id de la reunión

· +confirmacion	Bool	Devuelve si se acepta o rechaza el cambio
Clase: Tutor		
Clase hija de "UsuarioUCO" en la que se encierran todos los tutores de la Universidad de Córdoba		
Atributos		
· + degree_	String	Grado que imparte el tutor o estudia el alumno
· + officeDirection_	String	Dirección de la oficina del tutor
Operaciones		

· +getDegree_()	String	Devuelve el grado que estudia el alumno o imparte el tutor
· +setDegree_(degree : string)	Void	Actualiza el grado que estudia el alumno o imparte el tutor
· +degree	String	Grado a introducir
· +getOfficeDirection_()	String	Devuelve la dirección de la oficina del tutor
· +setOfficeDirection_(officedirection : string)	Void	Actualiza la dirección de la oficina del tutor
· +officedirection	String	Dirección a introducir
· +asignarAlumno(in dniAlumno : string, in tipoAsignacion : string, out estado : bool)	Bool	Con esta función se asigna un alumno a su tutor

· +dniAlumno	String	DNI del alumno a tutorizar
· +tipoAsignacion	String	Tipo de tutorización sobre el alumno
· +estado	Bool	Devuelve si se ha tutorizado correctamente
· +verTutorAsignado(out infoTutor : string)	String	El alumno puede ver la información de su tutor
· +infoTutor	String	Información del tutor
· +registrarReunión(in idReunion : int, in temasTratados : string, in resultado : string, out estado : bool)	Bool	Para registrar la reunión en la base de datos
· +idReunion	Int	ID correspondiente a la reunión
· +temasTratados	String	Contenido de la reunión

· +estado	Bool	Devuelve si se acepta o rechaza el registro
· +enviarAvisoAcadémico (in dniAlumno : string, in mensaje : string, out estado : bool)	Bool	El tutor puede mandar avisos
· +dniAlumno	String	DNI del alumno al que mandar el aviso
· +mensaje	String	Contenido del mensaje
· +estado	Bool	Devuelve si se envía o no el aviso
· +verInfoTutorados(out listaTutorizados : List<Alumno>)	List<Alumno>	Posibilita ver los alumnos tutorizados
· +listaTutorizados	List<Alumno>	Lista devuelta con alumnos y su información

Clase: Coordinador		
Clase hija de "UsuarioUCO" para el vicerrector de la Universidad de Córdoba		
Atributos		
Operaciones		
· +generarInformes(in fechaInicio : date, in fechaFin : date, in seleccion : int, out informe : file)	File	Generar informes sobre los datos almacenados
· +fechaInicio	Date	Fecha sobre la que comienza el informe
· +fechaFin	Date	Fecha sobre la que acaba el informe
· +selección	Int	

· +informe	File	Informe generado con toda la información
· +crearRegistroActividad (in tipoActividad : string, in descripcion : string, in dniAfectado : string, out estado : bool)	Bool	Esta función crea un registro de las actividades
· +tipoActividad	String	Indica la clase de actividad que hay que reflejar
· +descripcion	String	Texto explicativo sobre la actividad
· +dniAfectado	String	DNI del alumno afectado
· +estado	Bool	Certifica si se ha creado la actividad o no

Clase: Reunión		
Clase que representa una reunión programada, incluyendo su estado y fecha.		
Atributos		
· - idReunion_	Int	Identificador único de la reunión
· - dateTime_	Date	Fecha y hora de la reunión
· - status_	String	Estado actual de la reunión ("Finalizada", "Pendiente"...)
Operaciones		
· +getIdReunion_()	Int	Devuelve el ID de una reunión determinada
· +setIdReunion(idreunion : int)	Void	Actualiza el ID de una reunión determinada

· +idreunion	Int	ID de la reunión a introducir
· +getDateTime_()	Date	Devuelve la fecha de una reunión determinada
· +setDateTime(datetime : date)	Void	Actualiza la fecha de una reunión determinada
· +datetime	Date	Fecha de la reunión a introducir
· +getStatus_()	String	Devuelve el estado actual de la reunión
· +setStatus(status : string)	Void	Actualiza el estado actual de la reunión
· +status	String	Estado de la reunión que se quiere poner

· +comprobarDisponibilidadTutor()	Bool	Devuelve si el tutor esta disponible o no
Clase: Chat		
Clase que gestiona un chat individual, incluyendo su historial y las fechas de creación y eliminación.		
Atributos		
· - idChat_	Int	Identificador único del chat
· - removalDate_	Date	Fecha de eliminación del chat
· - creationDate_	Date	Fecha de creación del chat
Operaciones		

· +getIdChat_()	Int	Devuelve el ID de un chat determinada
· +setIdChat(chat : int)	Void	Actualiza el ID de un chat determinada
· +chat	Int	ID del chat a introducir
· +getCreationDate_()	Date	Devuelve la fecha de comienzo de un chat
· +setCreationDate(creationdate : date)	Void	Actualiza la fecha de comienzo de un chat
· +creationdate	Date	Fecha a introducir
· +getRemovalDate_()	Date	Devuelve la fecha de eliminación de un chat
· +setRemovalDate(removaldate : date)	Void	Actualiza la fecha de eliminación de un chat
· +removaldate	Date	Fecha a introducir

· +cargarHistorial(out lista : list<message>)	List<message>	Devuelve el historial del chat
· +eliminarHistorial(out confirmacion : bool)	Bool	Elimina el historial de un chat determinado
· +confirmacion	Bool	Determina si se ha eliminado correctamente o no
Clase: Mensaje		
Clase que representa un mensaje enviado en un chat, incluyendo su contenido, emisor y estado.		
Atributos		
· - idMessage_	Int	Identificador del mensaje
· - content_	String	Contenido del mensaje

· - emisor_	UsuarioUCO	Usuario que envia ese mensaje
· - dateandtime_	Date	Hora y fecha del mensaje enviado
· - isSeen_	Bool	Certifica si el mensaje ha sido visto por el receptor o no
Operaciones		
· +geIdMessage_()	Int	Devuelve el identificador del mensaje
· +setIdMessage(idMessage : int)	Void	Actualiza el identificador del mensaje
· +idMessage	Int	Identificador del mensaje a introducir
· +getContent_()	String	Devuelve el contenido del mensaje enviado

· +setContent(content : string)	Void	Actualiza el contenido del mensaje enviado
· +content	String	Contenido del mensaje a introducir
· +getEmisor_()	UsuarioUCO	Devuelve el usuario de la UCO que envía el mensaje
· +setEmisor(emisor : UsuarioUCO)	Void	Actualiza el usuario de la UCO que envía el mensaje
· +emisor	UsuarioUCO	Usuario a introducir
· +getDateAndTime_()	Date	Devuelve la fecha y hora en la que se envió el mensaje
· +setDateAndTime(dateandtime : date)	Void	Actualiza la fecha y hora en la que se envió el mensaje

· +dateandtime	Date	Fecha y hora a introducir
· +getIsSeen_()	Bool	Devuelve si el receptor ha leido el mensaje
· +setIsSeen(isSeen : Bool)	Void	Actualiza si el receptor ha leído el mensaje
· +isSeen	Bool	Variable bool a introducir
· +getFile_()	File	Devuelve el archivo que se ha enviado en el mensaje
· +setFile(file : file)	Void	Actualiza el archivo que se ha enviado en el mensaje
· +file	File	Archivo a introducir

Clase: InformeReunion		
Clase que almacena la información resultante de una reunión, como los temas tratados y las soluciones acordadas.		
Atributos		
· - idInforme_	Int	Identificador del informe
· - temasTratados_	String	Descripción de los temas tratados durante la reunión
· - solution_	String	Solución determinada en la reunión
Operaciones		
· +geIdInforme_()	Int	Devuelve el identificador del informe

· +setIdInforme(idInforme : int)	Void	Actualiza el identificador del informe
· +idInforme	Int	Identificador del informe a introducir
· +getTemasTratados_()	String	Devuelve la descripción de los temas tratados
· +setTemasTratados(temasTratados : string)	Void	Actualiza la descripción de los temas tratados
· +temasTratados	String	Descripción de los temas tratados que se quiere introducir
· +getSolution_()	String	Devuelve la solución tratada durante la reunión
· +setSolution(solution : string)	Void	Actualiza la solución tratada durante la reunión

· +solution	String	Solución a introducir

5.MATRICES DE TRAZABILIDAD

	CU-1	CU-2	CU-3	CU-4	CU-5	CU-6	CU-7	CU-8
RF1							X	
RF2							X	

RF3				X				
RF4			X		X			
RF5								X
RF6						X		
RF7			X					
RF8					X			
RF9	X							

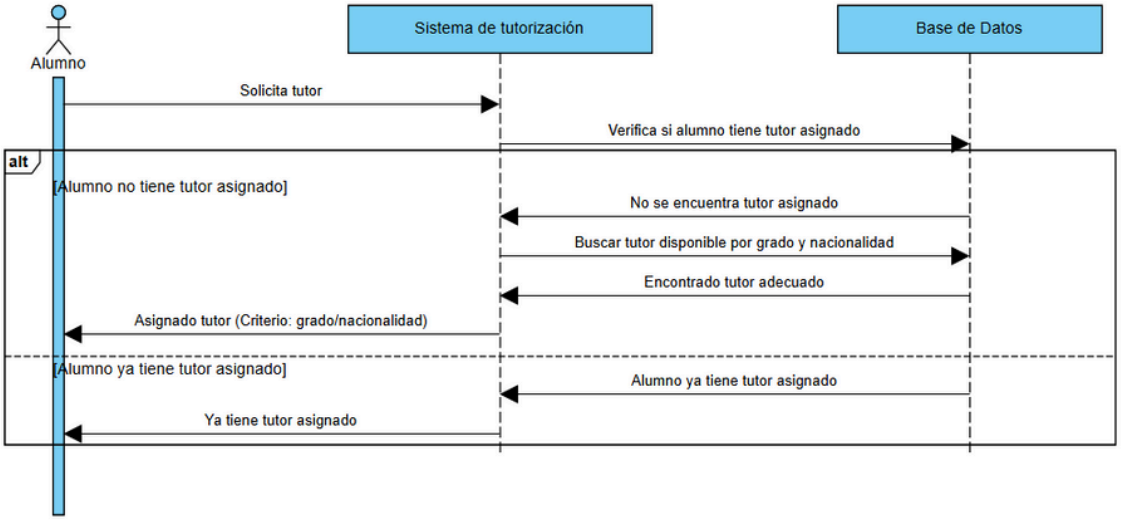
RF10							X	
RF11						X		
RF12		X						

	Usuario UCO	Alumno	Tutor	Coordinador	Chat	Reunión	Mensaje	Informe Reunión
CU-01	**X**	**X**	**X**	**X**				
CU-02		**X**				**X**		
CU-03			**X**					**X**

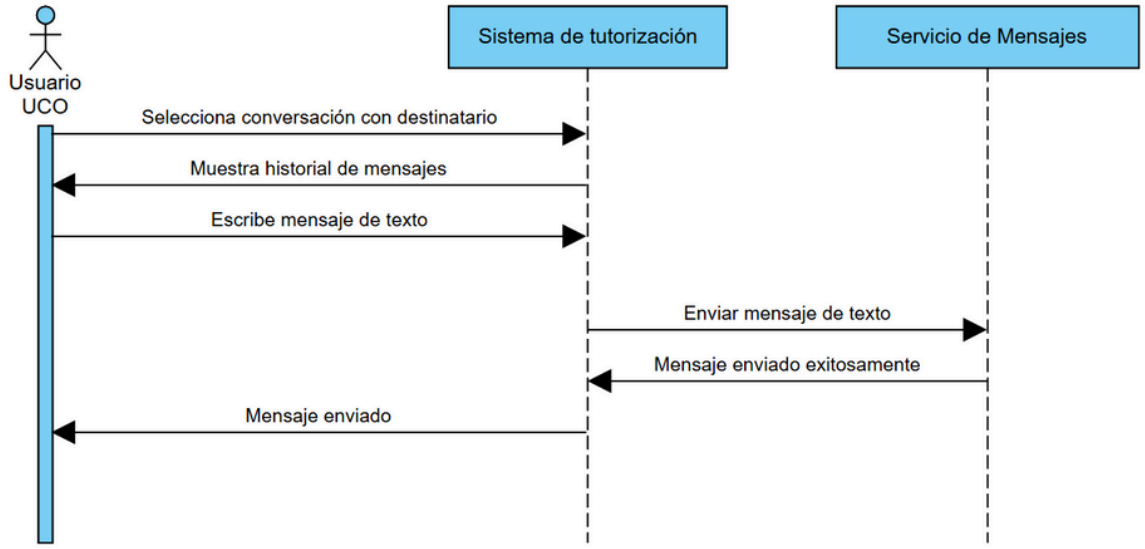
CU-04		X	X		X		X	
CU-05				X				
CU-06			X	X				
CU-07		X	X					
CU-08		X					X	

7. DIAGRAMAS DE SECUENCIA

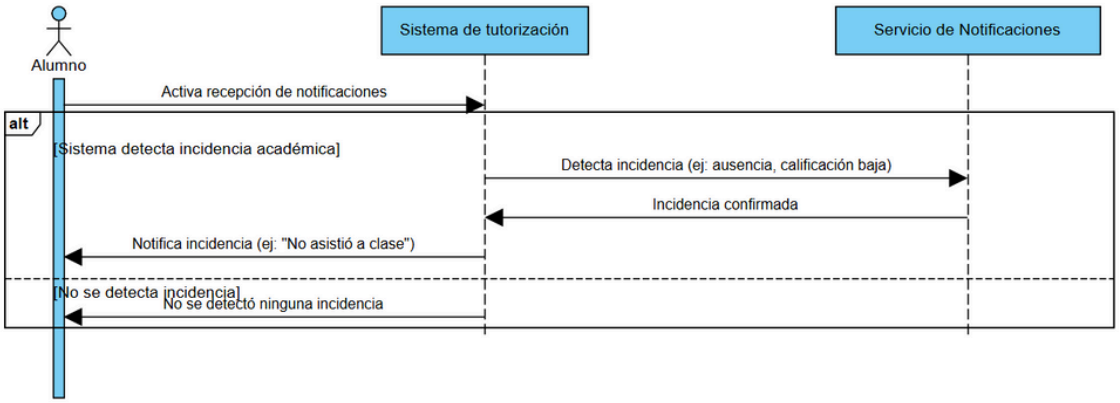
1) Asignar tutor a alumno CU-15



2) Enviar mensaje por Chat CU-04



3) Notificar alerta CU-16



8. IMPLEMENTACIÓN DEL SISTEMA

8.1 SPRINT 1

La primera semana de trabajo, comenzamos definiendo las bases de nuestro proyecto. No trabajamos con ninguna historia de usuario ni nos centramos directamente en empezar a programar, si no que construimos los pilares fundamentales de nuestro programa:

-Creación del diagrama de clases: Comenzamos definiendo todas las clases que iban a conformar nuestro programa. Estuvimos 2 días de trabajo con ello, ya que tuvimos errores y tuvimos que modificar y eliminar algunas de ellas.

- Creación de matrices de trazabilidad: Al día siguiente, una vez definidos todos los casos de uso e historias de usuario, creamos las matrices de trazabilidad expuestas anteriormente en este mismo documento. En este proceso tardamos un solo día.

- Creación diagrama de secuencia: Con todo lo anterior definido, y teniendo claro el flujo y la funcionalidad de nuestro programa, terminamos la semana

trabajando dos días diseñando los diagramas de secuencia que definen y detallan los procesos básicos de nuestro software.

Días de trabajo: 5 días

8.2 SPRINT 2

Esta segunda semana de trabajo, comenzamos eligiendo el lenguaje de programación. El backend iba a ser desarrollado con C++ y el frontend lo creamos usando Crow, que es un estándar bastante común para este lenguaje. Con las bases del proyecto ya definidas, pudimos empezar a programar de manera óptima con todo claro:

- H1. Inicio de sesión: El primer objetivo que nos pusimos fue la creación del login o inicio de sesión. Conseguimos desarrollarlo en 2 días. Esto es la base del programa. Nos propusimos crear un login sencillo que identificara al usuario y según su "rol" dentro de la universidad, pudiésemos clasificar al usuario en tutores o alumnos, para así, en los siguientes "pasos", poder dar unas funciones diferentes a ambos. Dependiendo del rol del usuario, creamos dos interfaces vacías idénticas, pero diferentes para modificar cada una de manera distinta.

- H7. Asignación automática de tutor: Tras el inicio de sesión, implementamos la asignación automática de tutor según ciertos parámetros como relación entre el ámbito académico tutor/alumno, idioma hablado por ambos y el número de alumnos que tenía el tutor. Siguiendo estas pautas, el programa calcula quién es el mejor tutor disponible para el alumno y lo asigna automáticamente. Este proceso tardamos 3 días en crearlo.

Días de trabajo: 5 días

8.3 SPRINT 3

Una vez teníamos las funcionalidades básicas de la aplicación con dos interfaces diferenciadas según el tipo de usuario, comenzamos a darle forma a la aplicación:

- H5. Mostrar nombre tutor: Para que el alumno sepa quién es su tutor y la información necesaria sobre él, modificamos la aplicación para añadir el nombre del tutor, para así poder saber quién es la persona que tutoriza al alumno

- H9. Mostrar alumnos tutorizados: El tutor podrá visualizar los alumnos tutorizados que tiene. El sistema ofrece la posibilidad de mostrar todos los alumnos que el tutor tiene a su cargo para tener un control del número, campo de conocimiento, idioma...

Días de trabajo: 4 días

8.4 SPRINT 4

Esta semana iba a ser la semana en la que más avanzáramos y en la que más funciones íbamos a implementar:

- H2. Chat de usuario: Comenzamos a trabajar en el chat entre usuarios, con un tiempo estimado de trabajo de 3 días, permitiendo únicamente la comunicación entre alumno y tutor o viceversa. Empezamos la semana con este objetivo ya que es el corazón del programa y la mejor forma de mejorar la comunicación alumno - tutor. Creamos una primera pestaña en la interfaz llamada “Chat” y los usuarios pueden visualizar los chats ahí. Los mensajes se almacenan en la base de datos y se siguen mostrando en el tiempo, para almacenar la información.

- H8. Envío de avisos: Diseñamos el sistema mediante el cuál el tutor puede enviar alertas o avisos personalizados a sus alumnos. Este sistema fue diseñado en 1 solo día ya que teníamos que hacerlo rápido para poder continuar con el siguiente rápidamente y acabarlo a tiempo con el sprint. Decidimos crear una pestaña adicional junto a la de “Chat” llamada “Avisos” en la que el tutor podía enviarlos y el estudiante los recibe y marca como leídos.

- H11. Sistema de tutorías: Este fue el sistema más complejo de desarrollar, tardamos 3 días a pleno pulmón. Esta función permite al alumno solicitar una tutoría con su tutor a una fecha y hora y escribir el motivo por el que solicita dicha tutoría. Creamos una última pestaña llamada “Tutorías” en la que el alumno puede solicitar una tutoría y el tutor puede ver, aceptar, rechazar y modificar dicha tutoría. Decidimos crearla de forma que el alumno verá un registro de las solicitudes de las tutorías que ha solicitado en una pestaña en la parte inferior y el tutor verá un registro de las solicitudes de todos sus alumnos en la parte lateral.

Días de trabajo: 6 días

8.5 SPRINT 5

Este es el último sprint que hicimos y nos centramos en pulir el programa para que la experiencia del usuario fuera la correcta

- H6. Cerrar sesión: Por último implementamos el cierre de sesión del programa en caso de que el alumno lo solicite o, mediante un contador, cuando el sistema detecte 5 minutos de inactividad del usuario. Este proceso tardamos 1 día en ejecutarlo.

- Pulir errores: Los siguientes 4 días de este sprint los dedicamos a pulir errores y fallos que había en el sistema y ejecutar y hacer pruebas. Por ejemplo, añadimos ciertos usuarios ficticios (tutores y alumnos) y estuvimos probando

todas las funcionalidades, la asignación automática del tutor y el correcto funcionamiento del chat.

Días de trabajo: 5 días

En ese enlace se encuentra el programa completo con el código correspondiente:

GitHub - vicctoor18/proyecto-is-201: Grupo 201 - IS - Proyecto Tutorización UCO

9.TESTS DEL SISTEMA

9.1. Diseño de pruebas unitarias

ID	**PRUEBA 01**	**Fecha**	2/12/2025
Historia de usuario	H1. Inicio de sesión		
Título de la prueba	UserAuthentication		
Responsable de la prueba	Víctor Jesús Castañeda Morales		
Descripción			
Se verifica que el sistema permita el acceso a usuarios registrados con contraseña correcta y deniegue el acceso a contraseñas erróneas o usuarios inexistentes.			
Prerrequisitos			
La base de datos debe estar inicializada y contener al usuario "alberto" con contraseña "1234".			
Entradas			
1. Usuario: "alberto", Pass: "1234" 2. Usuario: "alberto", Pass: "wrongpass" 3. Usuario: "ghost", Pass: "1234"			
Acciones a realizar			
Invocar al método DatabaseManager::instance().login(...) con los pares de credenciales indicados en Entradas.			
Salida esperada			
1. Retorna true, rol obtenido: "tutor". 2. Retorna false. 3. Retorna false.			
Salida obtenida			
true (Rol: tutor). 2. false. 3. false.			
Observaciones			
Prueba ejecutada correctamente. El bloqueo de seguridad funciona.			

ID	**PRUEBA 02**	**Fecha**	5/12/2025
Historia de usuario	H7. Asignación automática de tutor		
Título de la prueba	TutorAssignment		
Responsable de la prueba	David Llagas Rodríguez		
Descripción			
Verificar que al registrar un alumno nuevo, el sistema le asigna un tutor compatible automáticamente.			
Prerrequisitos			
Existencia de tutores en la BD con especialidad "Informatica" o nacionalidad "Espana".			
Entradas			
Nuevo registro: User "new_student_test", Grado "Informatica", País "Espana".			
Acciones a realizar			
1. registerUser(...) con los datos de entrada. 2. Login con el nuevo usuario. 3. getAssignedTutor(studentId).			
Salida esperada			
El ID del tutor devuelto debe ser distinto de -1. El tutor debe coincidir en Grado o Nacionalidad.			
Salida obtenida			
Tutor ID válido asignado. El perfil del tutor coincide con los criterios del alumno.			
Observaciones			
El algoritmo de emparejamiento funciona correctamente al crear el usuario.			

ID	**PRUEBA 03**	**Fecha**	8/12/2025
Historia de usuario	H11. Sistema de tutorías		
Título de la prueba	AppointmentRequest		
Responsable de la prueba	Julián Lara Peso		
Descripción			
Comprobar que un alumno puede crear una solicitud de cita y esta se guarda con el estado inicial correcto.			
Prerrequisitos			
Alumno "javier" logueado y con tutor asignado.			
Entradas			
Fecha: "2025-12-25", Hora: "10:00", Motivo: "Duda practica".			
Acciones a realizar			
1. requestAppointment(...). 2. Recuperar citas con getAppointments. 3. Buscar coincidencia de datos.			
Salida esperada			
La función devuelve true. La cita existe en la lista con estado "REQUESTED".			
Salida obtenida			
Cita insertada correctamente en la BD y recuperada con los datos exactos.			
Observaciones			
La inserción en la tabla de citas respeta la integridad referencial (Student ID y Tutor ID).			

ID	**PRUEBA 04**	**Fecha**	12/12/2025
Historia de usuario	H11. Sistema de tutorías		
Título de la prueba	AppointmentStatusUpdate		
Responsable de la prueba	Julián Lara Peso		
Descripción			
Verificar la capacidad del sistema para modificar el estado de una cita existente (de solicitada a programada).			
Prerrequisitos			
Una cita creada previamente con motivo "Revision".			
Entradas			
ID de la cita creada, Nuevo Estado: "SCHEDULED".			
Acciones a realizar			
1. Obtener ID de la cita. 2. Llamar a updateAppointmentStatus(id, "SCHEDULED"). 3. Verificar cambio.			
Salida esperada			
La actualización devuelve true. Al consultar la cita de nuevo, el estado es "SCHEDULED".			
Salida obtenida			
Estado actualizado correctamente en la persistencia.			
Observaciones			
El cambio de estado es persistente tras cerrar y abrir consultas.			

ID	PRUEBA 05	Fecha	16/12/2025
Historia de usuario	H2. Chat de usuario		
Título de la prueba	MessagingSystem		
Responsable de la prueba	Víctor Jesús Castañeda Morales		
Descripción			
Verificar el envío y recepción de mensajes de texto entre alumno y tutor.			
Prerrequisitos			
Alumno y tutor válidos vinculados entre sí.			
Entradas			
Contenido del mensaje: "Hola tutor".			
Acciones a realizar			
1. sendMessage(studentId, tutorId, msgContent). 2. getMessages(studentId, tutorId).			
Salida esperada			
El historial de mensajes no está vacío. El último mensaje coincide en contenido y remitente.			
Salida obtenida			
Mensaje almacenado y recuperado sin corrupción de datos.			
Observaciones			
El timestamp se genera automáticamente al insertar.			

ID	**PRUEBA 06**	**Fecha**	19/12/2025
Historia de usuario	H8. Envío de avisos y alertas		
Título de la prueba	AlertSystem		
Responsable de la prueba	David Llagas Rodríguez		
Descripción			
Comprobar el ciclo de vida de una alerta: creación, recepción y marcado como leída.			
Prerrequisitos			
Tutor "alberto" y Alumno "javier" logueados.			
Entradas			
Texto alerta: "Alerta de prueba".			
Acciones a realizar			
1. sendAlert(...). 2. Verificar isRead es false. 3. markAlertAsRead(...). 4. Verificar isRead es true.			
Salida esperada			
La alerta debe cambiar su booleano isRead de 0 a 1 tras la acción de marcar como leída.			
Salida obtenida			
La alerta se actualiza correctamente en la base de datos.			
Observaciones			
El sistema filtra correctamente las alertas por ID de estudiante.			

9.2.Informe de errores de pruebas unitarias

ID Prueba	Acción	Error	Solución
PRUEBA-01	Inicializar entorno y Login	**Fallo de Drivers SQL.** Al ejecutar el test de autenticación, la conexión fallaba con "Driver not loaded" porque QSqlDatabase necesita una instancia de aplicación Qt activa.	Se añadió la instanciación de QApplication en el main.cpp para cargar los plugins SQL antes de los tests.
PRUEBA-02	Asignación automática de tutor	**Fallo en coincidencia (Match).** El sistema devolvía -1 (sin tutor) incluso habiendo tutores compatibles, debido a que la consulta SQL era sensible a mayúsculas/minúsculas.	Se normalizó la consulta SQL usando UPPER() tanto en la columna de la base de datos como en la entrada del usuario.
PRUEBA-03	Insertar fecha de cita (requestAppointment)	**Error de formato de fecha.** La base de datos no reconocía el formato de fecha por defecto de Qt, guardando valores nulos o corruptos.	Se forzó el formato de cadena estándar ISO 8601 ("yyyy-MM-dd") antes de pasar la fecha a la consulta INSERT.

PRUEBA-04	Actualizar estado (UPDATE)	**Persistencia fallida.** La función retornaba true, pero al consultar de nuevo la cita, el estado seguía siendo el antiguo debido a un error de sintaxis en el WHERE.	Se corrigió la sentencia SQL UPDATE asegurando que el ID de la cita se pasaba correctamente como parámetro bind (:id).
PRUEBA-05	Recuperar historial de chat	**Desorden temporal.** Los mensajes se recuperaban en orden aleatorio de inserción, perdiendo la coherencia de la conversación.	Se añadió explícitamente la cláusula ORDER BY timestamp ASC en la consulta de recuperación de mensajes.
PRUEBA-06	Leer estado de alerta (Booleano)	**Error de tipo de dato.** SQLite guardaba el booleano como 0 o 1, pero C++ no lo convertía automáticamente a false/true al leerlo, fallando la validación.	Se implementó una conversión explícita (query.value(...).toInt() != 0) para mapear correctamente el entero de SQLite al booleano de C++.

10. ENLACES DE INTERÉS

-**Proyecto Youtrack:** proyectoingsoftware.youtrack.cloud

-**Diagramas de casos de uso:**

https://drive.google.com/file/d/19odQzO4N-9JlJfSGDvCyCLPKwF3PeCKG/view?usp=sharing

-**GitHub:** https://github.com/vicctoor18/proyecto-is-201