Índice

Presentación

Instituto Tecnologico De Informatica.

Proyecto Final Aplicación de gestión de faltas y licencias medicas.

Grupo: Los Programadores.

Integrantes: Matías Santillán, Román Rodríguez y Fabricio Álvez.

Docentes: Marcela Mederos, Leandro López y Eduardo Mazzetti.

Fecha de entrega: lunes 10 de noviembre, entrega final.

Programación, Base de datos y Redes informáticas.

Descripción general

El sistema desarrollado es una aplicación de escritorio destinada al Instituto Tecnológico de Informática (ITI) para la gestión de faltas e inasistencias docentes, incluyendo las licencias médicas. Está diseñada para facilitar el acceso, registro y control de información dentro del centro educativo, garantizando la organización y actualización constante de los datos almacenados.

La aplicación cuenta con dos tipos de acceso principales: administrativo y público. El perfil administrativo permite iniciar sesión mediante cédula y contraseña, pudiendo registrar, modificar o eliminar licencias médicas o faltas docentes. El perfil público, dirigido a alumnos y docentes, permite consultar las inasistencias del día sin necesidad de autenticarse, mostrando el nombre del docente ausente, el grupo afectado, el turno y la fecha de finalización de la licencia.

Cada usuario interactúa con una interfaz intuitiva y estructurada por menús, desarrollada en Java (NetBeans 19) con conexión a una base de datos MySQL. Los botones y formularios fueron diseñados para que cada operación pueda completarse en un tiempo promedio de entre tres y cinco segundos, asegurando un uso ágil y comprensible sin requerir conocimientos técnicos previos.

El sistema almacena toda la información en una base de datos local, garantizando la integridad de los registros mediante el uso de claves primarias y foráneas. No requiere conexión a Internet, ya que funciona dentro de la red interna del instituto, lo que asegura estabilidad y confidencialidad de los datos.

En su funcionamiento general, la aplicación permite registrar nuevas licencias médicas o faltas, modificar o eliminar registros existentes, consultar las inasistencias activas del día, y cerrar sesión de forma segura para proteger la información.

Esta herramienta tiene como propósito reducir los errores administrativos, mejorar

la comunicación interna y optimizar la gestión académica dentro del ITI, brindando un sistema confiable, accesible y adaptado a las necesidades reales de la institución.

Descripción de la realidad

En la actualidad, el Instituto Tecnológico de Informática (ITI) presenta dificultades en el manejo manual de las faltas y licencias médicas de los docentes. El registro de estas ausencias se realiza mediante planillas físicas o documentos digitales sin una base de datos unificada, lo que genera errores frecuentes, pérdida de información y demoras en la comunicación entre los distintos actores institucionales.

Estas demoras afectan la organización de las clases y la planificación académica, ya que los estudiantes y administrativos no siempre cuentan con información actualizada sobre qué docentes se encuentran ausentes y por cuánto tiempo. Además, el proceso de búsqueda y verificación de datos puede tardar varios minutos o incluso horas, dependiendo del volumen de información y del personal disponible.

Ante esta situación, se plantea la necesidad de implementar una aplicación informática que permita registrar, consultar y gestionar las licencias médicas y las inasistencias docentes de forma centralizada y ordenada. El sistema busca reducir los tiempos de registro y consulta a pocos segundos por operación, garantizando la disponibilidad constante de los datos dentro de la red local del centro.

De esta manera, se pretende mejorar la comunicación interna, optimizar la organización de horarios y asegurar que toda la información relacionada con las ausencias docentes esté actualizada, accesible y resguardada en una base de datos confiable.

Propuesta funcional

La aplicación se desarrollará en **Java (NetBeans v19)** y utilizará **MySQL** como gestor de base de datos.

Su objetivo principal es permitir que el administrador registre y gestione las inasistencias docentes (ya sean faltas comunes o licencias médicas) dentro de una

misma interfaz.

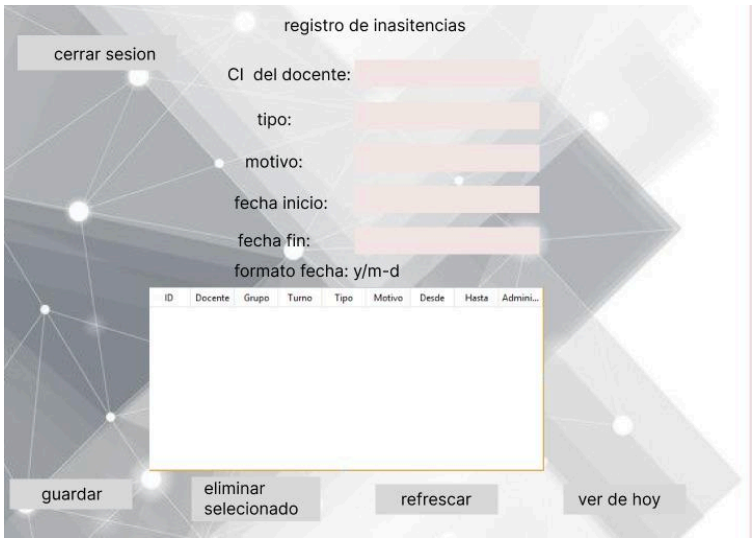
Por motivos de eficiencia y simplicidad, el proceso de registro de faltas y registro de licencias se unifica en un único formulario que utiliza los mismos campos: cédula del docente*, grupo, turno, fechas y motivo*.

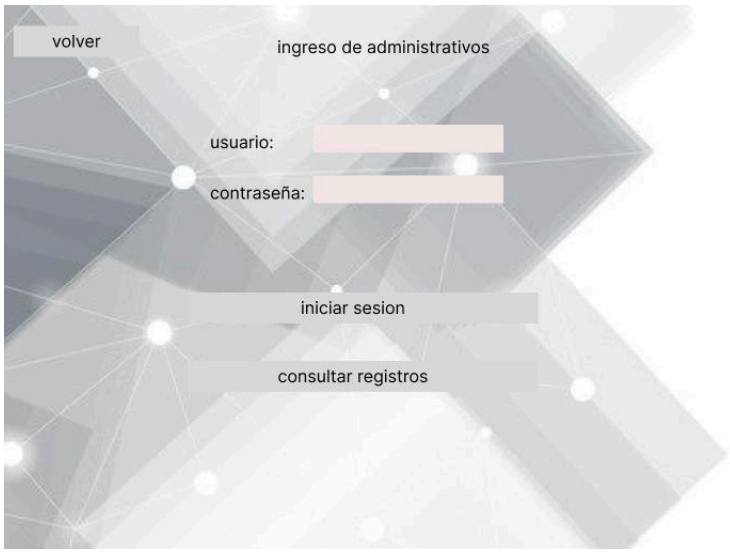
El motivo será el elemento diferenciador entre una falta y una licencia médica, permitiendo registrar ambas situaciones en la misma pantalla sin duplicar funciones.

Los alumnos y docentes podrán consultar las inasistencias del día sin iniciar sesión, simplemente presionando el botón “Consultar”, que mostrará la lista de docentes ausentes, grupos afectados y fechas de finalización.

El sistema busca reducir errores**,** simplificar tareas administrativas y mejorar la comunicación interna dentro del Instituto Tecnológico de Informática, funcionando exclusivamente en la red local del centro.

Figma





Propósito del sistema

Este sistema busca resolver el problema de los errores frecuentes y la confusión generada por el sistema actual de registro de faltas y licencias médicas de los docentes. Estos problemas afectan el correcto acceso a la información por parte de los estudiantes y la administración, provocando desorganización en los horarios de clase.

El sistema atiende la necesidad de contar con un registro preciso y eficiente de las ausencias docentes y sus licencias médicas, con el fin de reducir los errores y garantizar una correcta distribución de la información, mejorando la comunicación y la organización dentro de la institución.

Definiciones, acrónimos y abreviaturas

- APNB**:** *Apache NetBeans.* Entorno de desarrollo integrado (IDE) utilizado para programar la aplicación en lenguaje Java.
- XAMPP**:** Paquete de software que incluye Apache, MySQL, PHP y PHPMyAdmin. Se utiliza para ejecutar de forma local el servidor y la base de datos MySQL en los equipos del ITI.
- FIGMA**:** Herramienta de diseño utilizada para crear y visualizar la interfaz gráfica de la aplicación, incluyendo pantallas, botones y formularios.
- MySQL**:** Sistema de gestión de bases de datos relacional empleado para almacenar y administrar la información sobre docentes, licencias y faltas.
- JDBC**:** *Java Database Connectivity.* Conjunto de clases y métodos de Java que permiten conectar la aplicación con la base de datos MySQL.
- CI**:** Cédula de Identidad. Dato único de identificación utilizado por los usuarios dentro del sistema para iniciar sesión o registrar información.
- CRUD**:** *Create, Read, Update, Delete.* Operaciones básicas realizadas sobre la base de datos: crear, leer, modificar y eliminar registros.
- ITI**:** *Instituto Tecnológico de Informática.* Centro educativo donde se desarrolla y utiliza el sistema.
- IDE**:** *Integrated Development Environment.* Programa que permite escribir, compilar y ejecutar código de manera integrada, como NetBeans.
- SQL**:** *Structured Query Language.* Lenguaje estructurado utilizado para crear, modificar y consultar los datos de la base de datos.
- FODA: Fortalezas, Oportunidades, Debilidades, Amenazas.

Alcance del producto

Abarca el desarrollo de una aplicación de escritorio destinada al Instituto Tecnológico de Informática (ITI). Su función principal será registrar, consultar y gestionar las licencias médicas de los docentes. Permitirá que los administrativos carguen e

actualicen las licencias, que los estudiantes consulten las faltas diarias, y que los docentes puedan verificar su información registrada.

La aplicación mostrará datos como el nombre del docente, el turno, el grupo afectado y la fecha de finalización de la licencia médica. Está desarrollada en Java utilizando NetBeans como entorno de programación y MySQL como sistema de gestor de base de datos.

El sistema funcionará en computadoras del ITI conectadas a la red interna del centro por lo que no requerirá conexión a internet. No incluye funciones externas como envío de correos electrónicos, versiones móviles ni acceso desde fuera de la institución.

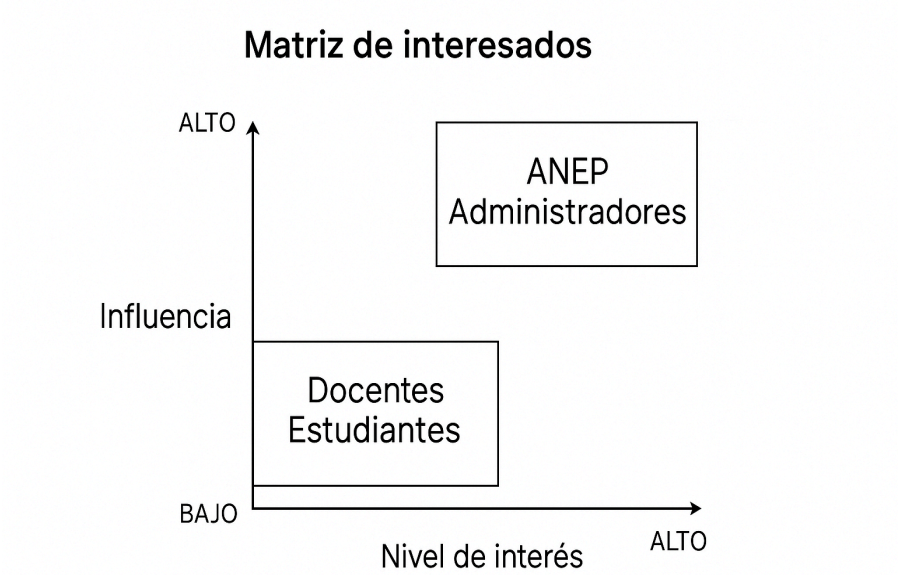
El alcance se limita al uso interno de la aplicación dentro del ITI, buscando mejorar la comunicación, la organización y la eficiencia.

Análisis de los interesados (stakeholder)

- **ANEP**: son los principales interesados ya que en ellos está la responsabilidad de que nuestra empresa (**Los Programadores**) se encargue del desarrollo correcto de la aplicación.
- **Docentes**: informarán sobre su asistencia o inasistencia a la institución.
- **Administradores:** se encargarian de cargar licencias médicas y/o inasistencias, además de mantener la empresa en orden.

- **Estudiantes:** son los más interesados ya que necesitan tener el registro gráfico de las inasistencias.

Matriz de interesados



Organización del equipo

El equipo de desarrollo está conformado por tres integrantes: **Román Rodríguez**, **Fabricio Álvez** y **Matías Santillán**.

Todos desempeñan el rol principal de **programadores**, participando de manera conjunta en el desarrollo, prueba y documentación del sistema.

Con el fin de optimizar el trabajo grupal, se asignaron tareas específicas según las

fortalezas de cada integrante:

- **Román Rodríguez – Programador y Diseñador Técnico:**

  Encargado principal de la elaboración de los diagramas técnicos del proyecto (UML, DER, RNE y otros), así como de la configuración y conexión del sistema con la base de datos MySQL. También colaboró en la redacción general del documento y en la revisión de aspectos técnicos.

- **Fabricio Álvez – Programador y Documentador Principal:**

  Responsable de la redacción y estructura general del documento del proyecto, incluyendo el desarrollo de los casos de uso, la normalización de tablas y el diseño de la documentación formal. Además, participó activamente en la programación junto al resto del equipo.

- **Matías Santillán – Programador y Analista:**

  Participó en el desarrollo del sistema y se encargó del análisis general del proyecto, revisando el funcionamiento del código, la coherencia de la documentación y el cumplimiento de los objetivos establecidos.

Esta distribución equilibrada permitió que todos los integrantes participaran en la

programación, asegurando al mismo tiempo una documentación completa y una validación constante de la calidad del sistema.

Análisis FODA :

	FORTALEZAS	DEBILIDADES
Análisis interno	- Capacidad para **pensar y programar** eficazmente. - Buena base técnica y creatividad para resolver problemas.	- **Falta de constancia** en algunos momentos. - **Frustración o tensión** que puede generar lentitud en el trabajo. - Dificultad para mantener una comunicación constante.
	OPORTUNIDADES	AMENAZAS
Análisis externo	- Se puede **consultar a otros grupos o compañeros** para resolver dudas técnicas. - El **centro educativo cuenta con equipos informáticos y conexión a Internet**, permitiendo avanzar en clase. - El proyecto puede servir como **experiencia práctica o referencia futura** en el ámbito académico o laboral.	- **Fallas técnicas** en los equipos del laboratorio o en los programas necesarios. - **Sobrecarga académica** por tareas o exámenes de otras asignaturas que reducen el tiempo disponible. - **Interrupciones externas** como paros, feriados o suspensiones de clases.

Estrategias

Estrategias ofensivas (Fortalezas + Oportunidades)

El equipo aprovechará su buena base técnica y creatividad para sacar el máximo provecho de los recursos tecnológicos del centro, realizando pruebas frecuentes en los equipos informáticos disponibles.
Además, se fomentará la consulta entre grupos y compañeros para intercambiar soluciones y optimizar el código, fortaleciendo el aprendizaje colaborativo y aumentando la calidad del producto final.

Estrategias defensivas (Fortalezas + Amenazas)

Ante posibles fallas técnicas o interrupciones externas, el grupo utilizará sus conocimientos de programación y resolución de problemas para mantener copias de seguridad periódicas del proyecto y garantizar la continuidad del trabajo.
También se planificarán las entregas con anticipación para evitar retrasos causados por paros o sobrecarga académica.

Estrategias de adaptación (Debilidades + Oportunidades)

Para contrarrestar la falta de constancia y mejorar la comunicación, el equipo aprovechará las herramientas digitales disponibles (Discord, WhatsApp y Crea2) para coordinar tareas y mantener una comunicación continua.
Además, se organizarán reuniones semanales breves para evaluar el avance y mantener la motivación del grupo.

Estrategias de supervivencia (Debilidades + Amenazas)

En caso de enfrentar sobrecarga académica o fallas en los equipos, el grupo dividirá las tareas según la disponibilidad de cada integrante, priorizando las actividades más urgentes.
Se establecerán planes alternativos de trabajo, como el uso de copias locales o computadoras personales, para asegurar que el proyecto continúe sin interrupciones.

Reglamento interno

1- La comunicación principal se realizará por Discord y WhatsApp para coordinar

tareas, reuniones y avances.

2- Si un integrante no puede asistir y/o entregar su parte a tiempo, debe avisar con antelación de 48 horas y justificar el motivo.

3- Ante situaciones médicas o personales justificadas, el integrante afectado no será penalizado. El resto del grupo asumirá temporalmente sus responsabilidades para asegurar que el trabajo continúe sin interrupciones.

4- Las modificaciones finales en los documentos o programas sólo podrán realizarse con el consentimiento de todos los integrantes. Se considerará que una modificación es final cuando todos los miembros hayan revisado y aprobado los cambios. Las aprobaciones de entregas parciales o definitivas serán acordadas grupalmente antes de ser enviadas.

5- Se prioriza la colaboración, el respeto mutuo y la mejora continua durante todo el desarrollo del proyecto.

6- Si se le nota perdido o poco serio a un integrante, se charla en el grupo y se llega a soluciones posibles.

Este reglamento se basa en los principios de la norma ISO 9001, priorizando la mejora continua y la responsabilidad grupal.

Bitácora de reuniones

N°	Fecha	Integrantes presentes	Tema / Motivo de la reunión	Discusión / Comentarios	Acuerdos / Soluciones	Responsable de seguimiento
-	-	-	-	-	-	-
-	-	-	-	-	-	-

Relevamiento de necesidades

El plan de relevamiento de necesidades se basa en analizar las demandas de estudiantes,

docentes y administrativos mediante observación y entrevistas internas.

A partir de esto, se establecen los requerimientos funcionales y no funcionales que guían el desarrollo del sistema.

Requerimientos funcionales

R1. Inicio de sesión (solo administrativos) (Alta)

El sistema debe permitir que los usuarios administrativos inicien sesión utilizando su usuario y contraseña.

R2 – Validación de credenciales

El sistema debe verificar que el usuario y contraseña sean correctas, mostrando un mensaje de error si son inválidas.

R3 – Registro de licencias médicas (Alta)

El Administrador debe poder registrar nuevas licencias médicas, ingresando la cédula del docente, tipo, motivo, y fechas correspondientes.

R4 – Eliminación de licencias (Media)

El Administrador debe poder eliminar licencias médicas del sistema de manera definitiva.

R5. Consulta de faltas del día (acceso directo) (Alta)

El alumno/docente debe poder consultar las faltas del día sin necesidad de iniciar sesión, mediante un botón que muestra los docentes ausentes junto con su grupo, turno y fecha de finalización de la licencia.

R6 – Actualización de datos personales

El sistema debe permitir que los usuarios actualicen su información personal, como contraseña o datos.

R7 – Cierre de sesión (Alta)

El sistema debe permitir que cualquier usuario cierre su sesión de manera segura, volviendo a la pantalla de inicio.

R8 – Validación de campos obligatorios

El sistema debe verificar que todos los campos requeridos estén completos antes de permitir guardar o modificar una licencia médica.

R9. Visualización pública de información

El sistema mostrará la información visible a los alumnos/docentes en formato de lista con sus correspondientes datos (id, docente, grupo, turno, tipo, motivó, fecha de inicio, fecha de finalización de la licencia y el nombre del administrativo que llevó a cabo el registro), sin permitir modificaciones.

R10. Registro predefinido de usuario administrador

El sistema deberá incluir por defecto un usuario administrador con credenciales iniciales para el acceso al sistema.

Requerimientos no funcionales

RNF1 – Gestor de Base de Datos

El sistema utilizará **MySQL** como base de datos relacional, garantizando la integridad

y consistencia de los datos almacenados mediante claves primarias y foráneas.

RNF2 – Plataforma de Desarrollo

La aplicación estará desarrollada en **Java (NetBeans 19)** utilizando **Swing** para la interfaz gráfica y se ejecutará en computadoras con sistema operativo **Windows 10 o superior**.

RNF3 – Seguridad de Acceso

Solo el administrador podrá iniciar sesión mediante usuario y contraseña válidas.

Los alumnos y docentes tendrán acceso directo únicamente a la función de consulta pública, sin posibilidad de modificar datos.

RNF4 – Rendimiento

El tiempo máximo de respuesta para consultar las inasistencias del día no debe superar los 3 segundos desde que el usuario presiona el botón hasta que se muestra la información.

Las operaciones de registro o eliminación deben completarse en menos de 5 segundos bajo condiciones normales de red local.

RNF5 – Usabilidad

El sistema deberá permitir que un usuario sin formación técnica pueda realizar correctamente las operaciones básicas (consultar, registrar, o eliminar) en menos de 5 minutos de uso sin requerir asistencia externa.

Los textos de botones y mensajes deberán ser claros, evitando abreviaturas o términos técnicos.

RNF6 – Disponibilidad Local

El sistema deberá estar operativo durante el horario de funcionamiento del instituto (de 07:00 a 22:00 horas), sin requerir conexión a Internet.

Podrá ejecutarse en cualquier equipo de la red local del ITI con Java instalado.

RNF7 – Confiabilidad de Datos

Los datos ingresados deberán guardarse correctamente en la base de datos en un 100 % de los casos válidos.

El sistema no debe duplicar registros ni perder información durante el uso normal.

RNF8 – Mantenibilidad

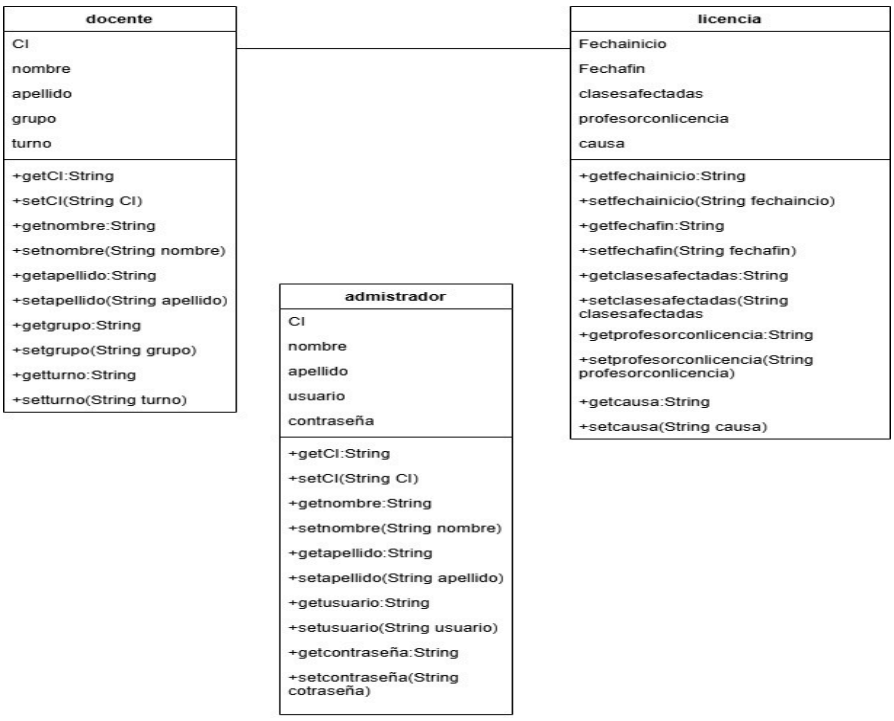
El sistema deberá permitir realizar cambios menores (textos, colores o validaciones) en menos de 30 minutos de edición, sin necesidad de reestructurar el código fuente ni la base de datos.

RNF9 – Estabilidad

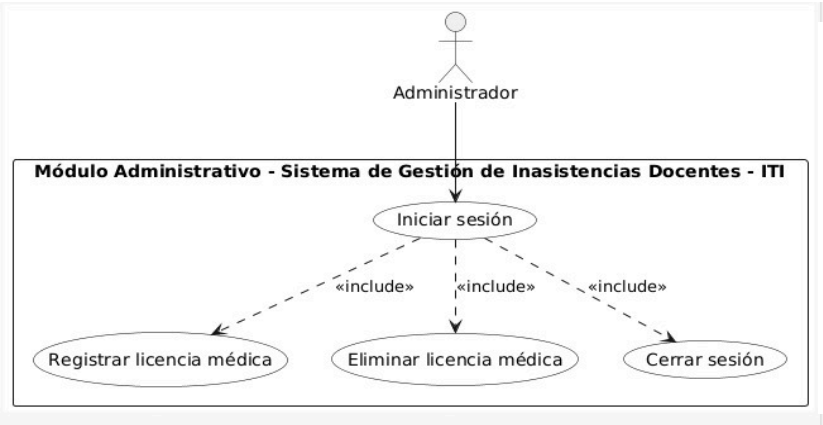
El sistema no deberá cerrarse inesperadamente más de una vez cada 20 ejecuciones.

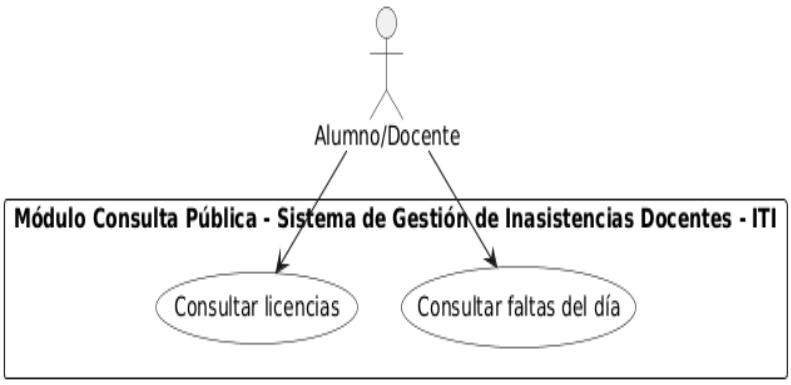
Ante un error, deberá mostrar un mensaje informativo sin afectar los datos almacenados.

Diagramas UML



Diagramas casos de uso.





Desarrollo de casos de uso.

Registro de inasistencias	
Nombre:	Registro de inasistencias
Actores:	Administrador
Prioridad:	Alta
Puntos de Extensión:	-
Puntos de Inclusión:	Iniciar sesión
Extiende a:	-
Incluye a:	Iniciar sesión
Precondiciones:	El administrador debe haber iniciado sesión correctamente en el sistema.
Poscondiciones:	**Escenario de éxito**: La falta o licencia médica del docente queda registrada correctamente en la base de datos, junto con el nombre, grupo, turno, motivo, tipo y fechas correspondientes. **Escenario de fracaso:** El sistema muestra un mensaje de error si los datos son incompletos o inválidos.
Flujo de Eventos:	1. El administrador accede al menú principal. 2. El sistema muestra un formulario con los campos requeridos. 3. El administrador completa los datos. 4. El sistema valida la información ingresada. 5. Si los datos son válidos, guarda el registro en la base de datos. 6. El sistema muestra el mensaje: *"Falta registrada correctamente"*, y vuelve al paso 3.
Flujo alternativo:	1. Si falta información, el sistema muestra: *"Complete todos los campos obligatorios"*. 2. Si se ingresa una fecha inválida, muestra: *"Fecha inválida"*.

Iniciar sesión	
Nombre:	Iniciar sesión
Actores:	Administrador
Prioridad:	Alta
Puntos de Extensión:	-
Puntos de Inclusión:	-
Extiende a:	-
Incluye a:	-
Precondiciones:	El usuario debe abrir la aplicación del sistema en una computadora del ITI
Poscondiciones:	Éxito: El usuario accede correctamente al menú de iniciar sesión.. Fracaso: El sistema muestra un mensaje de error indicando que el usuario o la contraseña son incorrectas.
Flujo de Eventos:	1. El sistema solicita el usuario y la contraseña. 2. El administrador ingresa sus credenciales. 3. El sistema valida la información. 4. Si las credenciales son correctas, el usuario accede al sistema. 5. Si no son válidas, se activa el flujo alternativo.
Flujo alternativo:	El sistema muestra el mensaje “Credenciales incorrectas” y vuelve al paso 1.

Eliminar licencia médica

Nombre:	Eliminar licencia médica
Actores:	Administrador
Prioridad:	Media
Puntos de Extensión:	-
Puntos de Inclusión:	Iniciar sesión
Extiende a:	-
Incluye a:	Iniciar sesión
Precondiciones:	Debe existir al menos una licencia cargada en el sistema.
Poscondiciones:	Éxito: La licencia seleccionada se elimina definitivamente. Fracaso: El sistema muestra un error y no elimina los datos.
Flujo de Eventos:	1. El administrador selecciona una licencia de la tabla. 2. Elige la opción **"Eliminar seleccionado"**. 3. El sistema verifica la solicitud en conexión con la base de datos. 4. La licencia se borra y se muestra: *"Registro eliminado correctamente"*.
Flujo alternativo:	Si el administrador cancela, no se elimina la licencia. Si la licencia no existe, se muestra: *"Error al eliminar registro"*.

Consultar licencias	
Nombre:	Consultar licencias
Actores:	Alumno/Docente
Prioridad:	Alta
Puntos de Extensión:	-
Puntos de Inclusión:	-

Extiende a:	-
Incluye a:	-
Precondiciones:	El usuario (Alumno o Docente) accede al sistema y selecciona el botón **“Consultar”**, sin necesidad de iniciar sesión.
Poscondiciones:	Éxito: El sistema muestra las faltas o licencias activas del día, incluyendo nombre del docente ausente, grupo afectado, turno, fecha de finalización y nombre del administrativo que registró la licencia. Fracaso: No existen faltas registradas para ese día y el sistema muestra un mensaje informativo.
Flujo de Eventos:	1. El usuario abre la aplicación desde una computadora del ITI. 2. El usuario presiona el botón **“Consultar”**. 3. El sistema consulta la base de datos y obtiene las licencias activas del día. 4. El sistema muestra en pantalla los docentes ausentes, los grupos afectados, el turno y la fecha de finalización de cada licencia.
Flujo alternativo:	Si no hay resultados, el sistema muestra el mensaje: *“No hay docentes ausentes hoy”*. Si ocurre un error de conexión o lectura de datos, el sistema muestra: *“No se encontraron inasistencias registradas”*.

Consultar faltas del dia	
Nombre:	Consultar faltas del dia
Actores:	Alumno/Docente
Prioridad:	Alta
Puntos de Extensión:	-
Puntos de Inclusión:	-
Extiende a:	-
Incluye a:	-

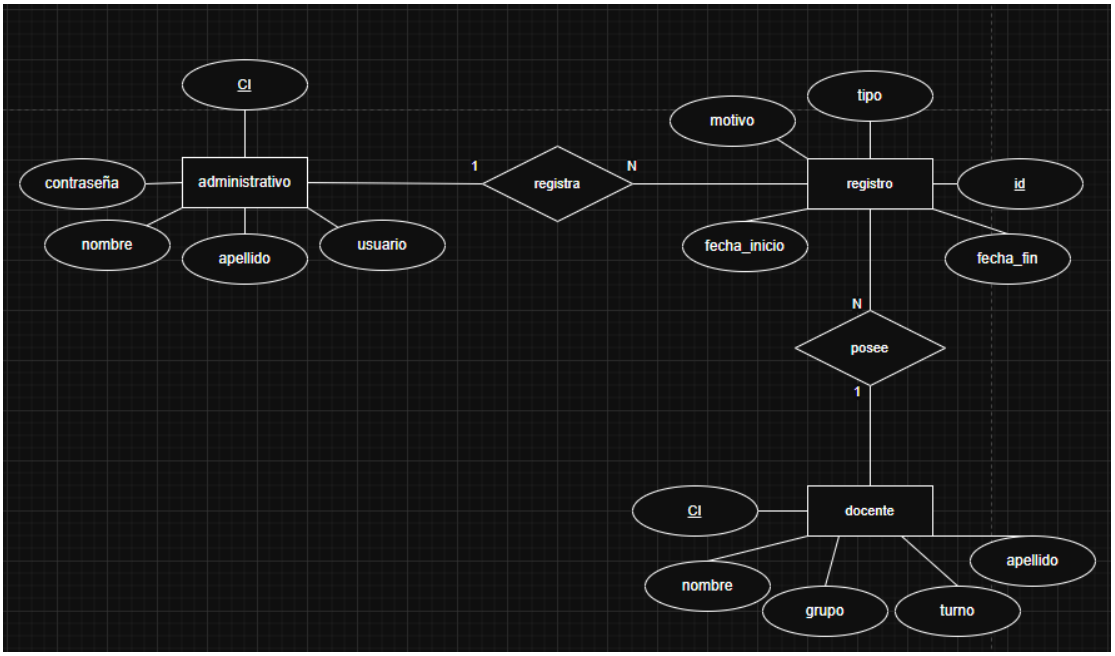
Precondiciones:	El Alumno/Docente accede al sistema mediante el botón **"Consultar"**, sin necesidad de iniciar sesión.
Poscondiciones:	Éxito: El sistema muestra los docentes ausentes del día con grupo, turno, tipo, motivo y fecha de correspondiente. Fracaso: No se encuentran registros y se muestra un mensaje informativo.
Flujo de Eventos:	1. El docente/alumno abre la aplicación y presiona el botón **"Consultar"**. 2. El sistema busca las licencias registradas asociadas a los docentes 3. Se muestran los resultados con docente, grupo, turno, tipo, motivó fechas correspondientes y administrativo que registró la licencia.
Flujo alternativo:	Si no hay registros, se muestra: *"No se encontraron inasistencias registradas"*. Si ocurre un error, se muestra: *"Error al cargar la información"*.

Cerrar Sesión	
Nombre:	Cerrar sesión
Actores:	Administrador
Prioridad:	Alta
Puntos de Extensión:	
Puntos de Inclusión:	Iniciar Sesión
Extiende a:	
Incluye a:	Iniciar Sesión
Precondiciones:	El administrador debe tener una sesión activa.
Poscondiciones:	Éxito: La sesión se cierra correctamente. Fracasó: Se muestra un mensaje de error si la sesión no puede cerrarse.
Flujo de Eventos:	1. El administrador selecciona la opción **"Cerrar**

	sesión”. 2. El sistema finaliza la sesión activa. 3. Se muestra la pantalla principal con las opciones de acceso.
Flujo alternativo:	Si ocurre un error, se muestra: *“No se pudo cerrar la sesión, inténtelo nuevamente”*.

Persistencia

Diagrama Entidad-Relación



RNE:

- **Cada registro de licencia o inasistencia debe ser creado por un único administrativo.**
  Un administrativo puede registrar varios registros, pero cada registro sólo puede ser ingresado por un administrativo.

- **Cada registro pertenece a un único docente.**
  Un docente puede tener varios registros, pero cada registro sólo puede estar asociado a un docente.

- **Las fechas ingresadas en un registro deben ser coherentes.**
  La fecha de inicio no puede ser posterior a la fecha de finalización.

- **No se pueden registrar licencias superpuestas para el mismo docente.**
  Es decir, no puede existir más de un registro activo para el mismo docente dentro del mismo rango de fechas.

- **El tipo y el motivo de la licencia son datos obligatorios.**
  No se puede registrar una licencia sin especificar ambos campos.

- **El usuario del administrativo debe ser único.**
  No puede haber dos administrativos con el mismo nombre de usuario en el sistema.

- **Solo los usuarios con rol de administrativo pueden crear, modificar o eliminar registros.**
  Los usuarios con rol de estudiante solo podrán realizar consultas.

Pasaje a tablas:

Entidades del DER pasadas a MR

ADMINISTRATIVO(ci, nombre, apellido, usuario, contraseña)

ci es **PK**

DOCENTE(ci, nombre, apellido, grupo, turno)

ci es **PK**

REGISTRO(id, tipo, motivo, fecha_inicio, fecha_fin, ci_docente, ci_administrativo)

id es **PK**

ci_docente es **FK** de **DOCENTE(ci)**

ci_administrativo es **FK** de **ADMINISTRATIVO(ci)**

Relaciones del DER pasadas a MR

registra (id, ci_administrativo)

`id` es FK de REGISTRO

`ci_administrativo` es **FK** de **ADMINISTRATIVO**

posee (id, ci_docente)

`id` es **FK** de **REGISTRO**

`ci_docente` es **FK** de **DOCENTE**

ADMINISTRATIVO(ci, nombre, apellido, usuario, contraseña)

Superclaves	Claves candidatas	Clave Primaria
{ci} {usuario} {ci, usuario} {ci, nombre} {ci, apellido}	{ci} {usuario}	{ci}

DOCENTE (ci, nombre, apellido, grupo, turno)

Superclaves	Claves candidatas	Clave Primaria
{ci} {ci, grupo}	{ci} {nombre, grupo, turno}	{ci}

{ci, turno} {nombre, grupo, turno}		

REGISTRO(id, tipo, motivo, fecha_inicio, fecha_fin, ci_docente, ci_administrativo)

Superclaves	Claves candidatas	Clave Primaria
{id} {id, ci_docente} {id, ci_administrativo} {ci_docente, fecha_inicio, fecha_fin}	{id} {ci_docente, fecha_inicio, fecha_fin}	{id}

registra(id, ci_administrativo)

Superclaves	Claves candidatas	Clave Primaria
{id, ci_administrativo} {id, ci_administrativo tipo}	{id, ci_administrativo}	{id, ci_administrativo}

posee(id, ci_docente)

Superclaves	Claves candidatas	Clave Primaria
{id, ci_docente} {id, ci_docente, tipo}	{id, ci_docente}	{id, ci_docente}

Normalización hasta 3FN

Relación ADMINISTRATIVO(ci, nombre, apellido, usuario, contraseña)

1FN:
Se encuentra en 1FN porque todos sus atributos son atómicos.

2FN:
Se encuentra en 2FN ya que la clave primaria está compuesta por un solo atributo (ci), y todos los demás dependen totalmente de él.

3FN:
Se encuentra en 3FN porque no hay dependencias transitivas entre atributos. El usuario es único, pero no depende de ningún otro atributo no clave.

Resultado:
ADMINISTRATIVO(ci, nombre, apellido, usuario, contraseña)
ci es PK.
usuario es atributo único (restricción adicional).

Relación DOCENTE(ci, nombre, apellido, grupo, turno)

1FN:
Se encuentra en 1FN porque todos los atributos son simples y atómicos.

2FN:
Cumple 2FN, ya que todos los atributos dependen completamente de la clave primaria ci.

3FN:
Cumple 3FN, porque no existen dependencias transitivas entre atributos no clave.

Resultado:
DOCENTE(ci, nombre, apellido, grupo, turno)
ci es PK.

Relación REGISTRO(id, tipo, motivo, fecha_inicio, fecha_fin, ci_docente, ci_administrativo)

1FN:
Se encuentra en 1FN porque todos los atributos son atómicos.

2FN:
Cumple 2FN ya que la clave primaria id identifica totalmente todos los atributos.

3FN:
Cumple 3FN porque no existen dependencias transitivas.

Los atributos ci_docente y ci_administrativo son claves foráneas y no generan redundancia.

Resultado:
REGISTRO(id, tipo, motivo, fecha_inicio, fecha_fin, ci_docente, ci_administrativo)
id es PK.
ci_docente es FK → DOCENTE(ci).
ci_administrativo es FK → ADMINISTRATIVO(ci).

Relación REGISTRA(id, ci_administrativo)

1FN:
Se encuentra en 1FN porque ambos atributos son atómicos.

2FN:
Cumple 2FN, ya que la clave primaria está compuesta por (id, ci_administrativo) y ambos atributos participan totalmente.

3FN:
Cumple 3FN porque no existen atributos no clave ni dependencias transitivas.

Resultado:
REGISTRA(id, ci_administrativo)

id es FK → REGISTRO(id).
ci_administrativo es FK → ADMINISTRATIVO(ci).

Relación POSEE(id, ci_docente)

1FN:
Se encuentra en 1FN porque los atributos son simples y atómicos.

2FN:
Cumple 2FN porque la clave primaria (id, ci_docente) determina completamente la relación.

3FN:
Cumple 3FN, ya que no hay dependencias entre atributos no clave.

Resultado:
POSEE(id, ci_docente)
id es FK → REGISTRO(id).
ci_docente es FK → DOCENTE(ci)

Creación de Tablas en SQL

A continuación, se presenta el código SQL utilizado para la creación de las tablas del sistema de Inasistencias Docentes, respetando las reglas de negocio específicas (RNE) y asegurando la integridad referencial entre los datos.

Tabla Administrativo:

```
CREATE TABLE ADMINISTRATIVO (
    CI          VARCHAR(20)  NOT NULL,
    nombre      VARCHAR(50)  NOT NULL,
    apellido    VARCHAR(50)  NOT NULL,
    usuario     VARCHAR(30)  NOT NULL,
    contrasena  VARCHAR(30)  NOT NULL,
    CONSTRAINT PK_ADMINISTRATIVO PRIMARY KEY (CI),
```

Tabla: Docente

```
CREATE TABLE DOCENTE (
  CI        VARCHAR(20)  NOT NULL,
  nombre    VARCHAR(50)  NOT NULL,
  apellido  VARCHAR(50)  NOT NULL,
  grupo     VARCHAR(30)  NULL,
  turno     VARCHAR(30)  NULL,
  CONSTRAINT PK_DOCENTE PRIMARY KEY (CI)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Tabla: Registro

```
CREATE TABLE REGISTRO (
  id                INT NOT NULL AUTO_INCREMENT,
  tipo              VARCHAR(30)   NOT NULL,
  motivo            VARCHAR(100)  NOT NULL,
  fecha_inicio      DATE          NOT NULL,
  fecha_fin         DATE          NOT NULL,
  CI_docente        VARCHAR(20)   NOT NULL,
  CI_administrativo VARCHAR(20)   NOT NULL,
  CONSTRAINT PK_REGISTRO PRIMARY KEY (id),
  CONSTRAINT FK_REGISTRO_DOCENTE
    FOREIGN KEY (CI_docente) REFERENCES DOCENTE(CI)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  CONSTRAINT FK_REGISTRO_ADMIN
    FOREIGN KEY (CI_administrativo) REFERENCES ADMINIST
    ON UPDATE CASCADE
    ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Se utilizó el motor InnoDB y codificación UTF8MB4 para permitir integridad referencial y compatibilidad con

caracteres especiales.

Resumen general de restricciones:

Tipo de restricción	Descripción	Tablas afectadas
PRIMARY KEY	Identifica cada registro de manera única	Administrativo, Docente, Registro
UNIQUE	Evita duplicados en usuarios administrativos	Administrativo
FOREIGN KEY	Mantiene integridad referencial entre tablas	Registro
ON UPDATE CASCADE	Actualiza automáticamente claves en relaciones	Registro
ON DELETE RESTRICT	Evita eliminaciones de entidades con registros asociados	Registro

Población de datos

Poblado de tabla administrativo

```
INSERT INTO ADMINISTRATIVO (CI, nombre, apellido, usuario, contrasena) VALUES
('4.321.654-8', 'Fabricio', 'Alvez', 'falvez', 'fabricio123'),
('5.982.334-5', 'Matias', 'Santillan', 'msantillan', 'santillan223'),
('5.742.915-4', 'Roman', 'Rodriguez', 'rrodriguez', 'ro1234');
```

Poblado de tabla docente

```
INSERT INTO DOCENTE (CI, nombre, apellido, grupo, turno) VALUES
('4.125.698-7', 'Leandro', 'Lopez', '2MC', 'Matutino'),
('5.698.321-4', 'Marcela', 'Mederos', '2MC', 'Matutino'),
('4.857.963-2', 'Sofia', 'Fernandez', '1BI', 'Vespertino'),
('5.632.874-9', 'Martin', 'Cabrera', '3MC', 'Nocturno'),
('5.487.213-1', 'Lucia', 'Pereyra', '4TI', 'Vespertino');
```

Población de tabla registro

```
INSERT INTO REGISTRO (tipo, motivo, fecha_inicio, fecha_fin, CI_docente, CI_administrativo) VALUES
('Enfermedad', 'Gripe estacional', '2025-10-15', '2025-10-18', '4.125.698-7', '4.321.654-8'),
('Concurso', 'Participación en concurso docente', '2025-09-22', '2025-09-24', '5.698.321-4', '5.982.334-5'),
('Compensada', 'Licencia por día trabajado', '2025-11-05', '2025-11-05', '4.857.963-2', '5.742.915-4');
```

Consultas necesarias para la aplicación:

1. Listar todas las licencias registradas

```
SELECT
    r.id,
    d.nombre AS Nombre_Docente,
    d.apellido AS Apellido_Docente,
    r.tipo,
    r.motivo,
    r.fecha_inicio,
    r.fecha_fin,
    a.nombre AS Nombre_Administrativo,
    a.apellido AS Apellido_Administrativo
FROM REGISTRO r
JOIN DOCENTE d ON r.CI_docente = d.CI
```

Descripción:
Muestra todas las licencias cargadas en el sistema con la información del docente y del administrativo que la registró, ordenadas desde la más reciente.

2. Mostrar licencias vigentes al día de hoy

```sql
SELECT
    d.nombre,
    d.apellido,
    d.grupo,
    d.turno,
    r.tipo,
    r.fecha_inicio,
    r.fecha_fin
FROM REGISTRO r
JOIN DOCENTE d ON r.CI_docente = d.CI
WHERE CURDATE() BETWEEN r.fecha_inicio AND r.fecha_fin;
```

Descripción:

Devuelve las licencias actualmente vigentes según la fecha del sistema.

Es útil para que los estudiantes consulten qué docentes no asistirán ese día.

3. Buscar licencias por docente

```
SELECT
    r.id,
    r.tipo,
    r.motivo,
    r.fecha_inicio,
    r.fecha_fin,
    a.nombre AS Administrativo
FROM REGISTRO r
JOIN ADMINISTRATIVO a ON r.CI_administrativo = a.CI
WHERE r.CI_docente = '4.125.698-7';
```

Descripción:
Permite obtener todos los registros asociados a un docente en particular (en este caso, el docente con CI 4.125.698-7).
El criterio puede ser modificado dinámicamente desde la interfaz de la aplicación.

4. Listar docentes con al menos una licencia registrada

```
SELECT
    d.CI,
    d.nombre,
    d.apellido,
    COUNT(r.id) AS total_licencias
FROM DOCENTE d
JOIN REGISTRO r ON d.CI = r.CI_docente
GROUP BY d.CI, d.nombre, d.apellido
HAVING COUNT(r.id) > 0;
```

Descripción:

Muestra los docentes que tienen al menos una licencia registrada, indicando la cantidad total.

Es útil para informes administrativos o listados generales.

5. Consultar licencias entre dos fechas determinadas

```
SELECT
    d.nombre,
    d.apellido,
    r.tipo,
    r.motivo,
    r.fecha_inicio,
    r.fecha_fin
FROM REGISTRO r
JOIN DOCENTE d ON r.CI_docente = d.CI
WHERE r.fecha_inicio BETWEEN '2025-09-01' AND '2025-09-30'
ORDER BY r.fecha_inicio;
```

Descripción:Permite visualizar las licencias ocurridas dentro de un período definido. Puede adaptarse fácilmente a filtros por mes o rango temporal desde la interfaz.

6. Consultar qué administrativo registró una licencia

```
SELECT
    r.id,
    r.tipo,
    r.motivo,
    r.fecha_inicio,
    r.fecha_fin,
    a.nombre,
    a.apellido
FROM REGISTRO r
JOIN ADMINISTRATIVO a ON r.CI_administrativo = a.CI
WHERE r.id = 1;
```

Descripción:
Dada una licencia específica (por ID), muestra los datos del administrativo responsable de su registro.
Ayuda a rastrear modificaciones o validar información.

7. Consultar licencias por tipo

```
SELECT
    tipo,
    COUNT(*) AS total
FROM REGISTRO
GROUP BY tipo
ORDER BY total DESC;
```

Descripción:
Cuenta cuántas licencias hay de cada tipo (Enfermedad, Concurso, Compensada, etc.).
Se usa para generar estadísticas generales del sistema.

8. Mostrar docentes sin licencias registradas

```
SELECT
    d.CI,
    d.nombre,
    d.apellido,
    d.grupo,
    d.turno
FROM DOCENTE d
LEFT JOIN REGISTRO r ON d.CI = r.CI_docente
WHERE r.id IS NULL;
```

Descripción:

Devuelve los docentes que aún no tienen ninguna licencia registrada.

Sirve para verificar docentes activos sin registros en el sistema.

9. Mostrar registros ordenados por fecha de fin

```sql
SELECT
    r.id,
    d.nombre,
    d.apellido,
    r.tipo,
    r.fecha_inicio,
    r.fecha_fin
FROM REGISTRO r
JOIN DOCENTE d ON r.CI_docente = d.CI
ORDER BY r.fecha_fin DESC;
```

Descripción:

Permite revisar rápidamente qué licencias finalizan antes o después, útil para reportes de control.

10. Consultar licencias por grupo o turno

```sql
SELECT
    d.grupo,
    d.turno,
    d.nombre,
    d.apellido,
    r.tipo,
    r.fecha_inicio,
    r.fecha_fin
FROM DOCENTE d
JOIN REGISTRO r ON d.CI = r.CI_docente
WHERE d.grupo = '2MC' AND d.turno = 'Matutino';
```

Descripción:

Filtra licencias de docentes pertenecientes a un grupo y turno determinados.

Es útil para que un grupo específico de estudiantes vea las inasistencias de sus docentes.

Resumen

N°	Consulta	Función principal
1	Listar todas las licencias	Vista general del sistema
2	Licencias vigentes	Consultas de estudiantes
3	Buscar por docente	Filtrado individual
4	Docentes con licencias	Estadísticas administrativas
5	Licencias por rango de fechas	Informes temporales
6	Administrativo que registró	Auditoría y control
7	Licencias por tipo	Estadísticas
8	Docentes sin licencias	Verificación
9	Ordenar por fecha fin	Seguimiento temporal
10	Por grupo y turno	Consulta por estudiantes

Sentencias SQL necesarias para la aplicación.

1. **Insertar un nuevo administrativo**

INSERT INTO administrativo (CI, nombre, apellido, usuario, contrasena)

VALUES (?, ?, ?, ?, ?);

Algebra: Π CI, nombre, apellido, usuario, contrasena $($ $\sigma_{usuario=:usuario}$ (ADMINISTRATIVO) $)$

2. Buscar administrativo por usuario (para login)

SELECT CI, nombre, apellido, usuario, contrasena FROM administrativo
WHERE usuario = ?;

Algebra: Π id, tipo, motivo, fecha_inicio, fecha_fin, CI_docente, CI_administrativo, .nombre_docente, apellido_docente, grupo, turno, nombre_admin, apellido_admin ((registro $\bowtie$(CI_docente = CI) docente) $\bowtie$(CI_administrativo = CI) administrativo)

3. Insertar una nueva inasistencia

```
INSERT INTO registro (tipo, motivo, fecha_inicio, fecha_fin,
CI_docente, CI_administrativo) VALUES (?, ?, ?, ?, ?, ?);
```

Algebra: Π id, tipo, motivo, fecha_inicio, fecha_fin, CI_docente, CI_administrativo, nombre_docente, apellido_docente, grupo, turno, nombre_admin, apellido_admin (σ (fecha_inicio <= HOY $\wedge$ fecha_fin >= HOY) ((registro $\bowtie$(CI_docente = CI) docente) $\bowtie$(CI_administrativo = CI) administrativo))

4. Actualizar un registro existente

```
UPDATE registro

SET tipo = ?, motivo = ?, fecha_inicio = ?, fecha_fin = ?,
CI_docente = ?, CI_administrativo = ?

WHERE id = ?;
```

5. Eliminar un registro por ID

```
DELETE FROM registro

WHERE id = ?;
```

6. Obtener todos los registros

```
SELECT r.id, r.tipo, r.motivo, r.fecha_inicio, r.fecha_fin,

    r.CI_docente, r.CI_administrativo,

    d.nombre AS nombre_docente, d.apellido AS
apellido_docente, d.grupo, d.turno,

    a.nombre AS nombre_admin, a.apellido AS
apellido_admin

FROM registro r
```

```
LEFT JOIN docente d ON r.CI_docente = d.CI

LEFT JOIN administrativo a ON r.CI_administrativo = a.CI

ORDER BY r.fecha_inicio DESC, r.id DESC;
```

7. **Obtener las inasistencias activas en la fecha actual**

```
SELECT r.id, r.tipo, r.motivo, r.fecha_inicio, r.fecha_fin,

    r.CI_docente, r.CI_administrativo,

    d.nombre AS nombre_docente, d.apellido AS
```

```
apellido_docente, d.grupo, d.turno,

      a.nombre AS nombre_admin, a.apellido AS
apellido_admin

FROM registro r

LEFT JOIN docente d ON r.CI_docente = d.CI

LEFT JOIN administrativo a ON r.CI_administrativo = a.CI

WHERE r.fecha_inicio <= CURDATE() AND r.fecha_fin >=
CURDATE()
```

```
ORDER BY r.fecha_inicio DESC, r.id DESC;
```

Webgrafía

OpenAI. (2025). *ChatGPT* [Plataforma en línea].

Disponible en: https://chat.openai.com

Herramienta utilizada para la generación de ideas, redacción de textos técnicos, apoyo en la elaboración de diagramas UML, desarrollo de requerimientos y resolución de dudas relacionadas con programación y bases de datos.

Plan Ceibal. (2025). *Crea2 – Plataforma educativa* [Sitio web].

Disponible en: https://crea.ceibal.edu.uy

Plataforma utilizada para acceder a los materiales y tareas proporcionadas por los docentes, especialmente de las asignaturas Programación Avanzada, Base de Datos y Redes Informáticas.

Apache Friends. (2024). *XAMPP* [Software de servidor local].

Disponible en: https://www.apachefriends.org/es/index.html

Aplicación usada para ejecutar el servidor local y la base de datos MySQL durante las pruebas del sistema.

Oracle. (2024). *MySQL* [Sistema de gestión de bases de datos].

Disponible en: https://www.mysql.com

Gestor de base de datos relacional utilizado para almacenar y administrar la información de las licencias y faltas docentes.

Apache Software Foundation. (2024). *Apache NetBeans IDE* [Entorno de desarrollo].

Disponible en: https://netbeans.apache.org

Entorno de desarrollo integrado utilizado para programar la aplicación en lenguaje Java y conectar con la base de datos mediante JDBC.

PlantUML. (2025). *PlantUML – Herramienta de diagramación UML* [Software en línea].

Disponible en: https://www.planttext.com/

Herramienta empleada para la creación del diagrama de caso de uso y la generación de representaciones UML del sistema.

Figma. (2025). *Figma – Diseño de interfaz de usuario* [Aplicación web].

Disponible en: https://www.figma.com

Herramienta usada para el diseño visual de la interfaz del sistema (pantalla de inicio, botones y menú principal).

Material docente – Prof. Marcela Mederos y Prof. Leandro López. (2025). *Apuntes y guías del curso Programación Avanzada, Redes Informáticas y Base de Datos.* Proporcionado a través de la plataforma Crea2 (Plan Ceibal).

Fuente de referencia para la estructura del documento, la metodología UML y la letra del proyecto.

Conclusión

Podemos cerrar de todo esto que hay que generar el diseño de una aplicación con distintos fundamentos, textos, diagramas y tablas para luego en una próxima instancia poder llevar acabo esta aplicación con los fines de poder comunicar dentro de la aplicación inasistencias docentes para todos los grupos pertenecientes a la UTU y las herramientas a usar para poder cometer este fin serán para la parte de programación NetBeans en el caso de la base de datos será racional y pertenece al lenguaje MySQL. Desde una parte más colectiva o introspectiva del

equipo tuvimos algunos problemas más que nada de comunicación, pero para próximas entregas se evaluará un trabajo mucho más estricto y mejor comunicado entre el grupo generando así un mejor desempeño y mejora de las habilidades blandas.

La unificación del registro de faltas y licencias médicas en un mismo formulario permitió simplificar el desarrollo del sistema, reduciendo tiempos de programación y mantenimiento sin afectar la funcionalidad ni los objetivos del proyecto.