SEMINARIO DE INFORMÁTICA

Trabajo Práctico 4

Módulo 4

Sistema de Gestión y Administración de Licencias

Maria Camila Caracciolo
35459191
29/06/2025
INF275-11145
Ana Carolina Ferreyra

Índice

Introducción.....3

Justificación.....3

Objetivo general del proyecto.....4

Objetivo general del sistema.....4

Elicitación y Conocimiento del negocio.....4

Propuesta de solución.....6

Características de la solución.....7

Beneficios esperados.....7

Requisitos Específicos.....7

Requerimientos funcionales.....7

Matriz de trazabilidad 10

Restricciones – Requerimientos No funcionales.....11

Requisitos futuros. 11

Suposiciones y dependencias. 11

Inicio de análisis – casos de uso.....11

Diagramas de casos de uso.....14

Diagrama de Transición de estados y descripción.....16

Diagrama de clases clases.....17

Descripción de diagrama de clases..... 17

Aplicación del proceso unificado de desarrollo.....18

Etapa de análisis – Diagrama de Secuencia.....19

Etapa de diseño – Diagrama de Clases de Diseño.....22

Etapa de implementación – Diagrama Despliegue.....22

Etapa de pruebas.....23

Definición de base de datos para el sistema. 26

Diagrama entidad-relación de la base de datos. 26

Creación de las tablas. 27

Inserción, consulta y borrado de registros. 27

Presentación de las consultas SQL. 28

Interfaz.....28

Definiciones de comunicación.....29

Explicación del desarrollo en Java. 30

Presentación del desarrollo en Java.....33

Bibliografía.....34

Introducción

En todos los ámbitos laborales, y en particular de acuerdo con la Ley de Trabajo Nacional N° 471, para la administración pública de la Ciudad de Buenos Aires, todos los empleados que se encuentren en blanco, tienen derecho a gozar de diferentes licencias a lo largo de su carrera profesional.

Cada una de estas licencias implica ciertas responsabilidades, como justificar la ausencia por el motivo que se haya solicitado la misma.

También cada una de las áreas que se responsabiliza de la gestión de las licencias de su personal, deberá seguir procedimientos y diferentes lineamientos impuestos por su ámbito laboral.

El presente proyecto tiene como objetivo desarrollar un sistema en el cual se puedan solicitar las licencias que el personal tiene disponible para usufructuar, gestionar las licencias, llevar un control de las mismas con diferentes tipos de tableros y reportes, y a su vez que se pueda realizar un control de ausentismo de manera dinámica.

En cuanto a los antecedentes, en mi ámbito laboral existen sistemas actualmente en uso, como SIAL Meta 4, SMLAM, SIGENO y SILOL. Si bien estos sistemas permiten la gestión y solicitud de licencias, presentan diversas limitaciones tanto para los empleados como para el personal administrativo. Estas problemáticas incluyen falta de integración, interfaces poco intuitivas y la ausencia de herramientas para el análisis y control de ausentismo.

Justificación

Aunque este proyecto se basa en soluciones preexistentes, busca superar las limitaciones actuales y ofrecer herramientas innovadoras para facilitar la gestión de licencias. Esto incluye la implementación de análisis de datos que apoyen la toma de decisiones estratégicas en la administración del personal.

Uno de los principales problemas detectados es la duplicación de licencias, dado que estas se cargan en diferentes sistemas, lo que permite que una misma persona figure con dos licencias en la misma fecha. Además, la segmentación de los sistemas según la relación laboral genera barreras de acceso para el personal administrativo, incluso cuando los empleados pertenecen al mismo departamento.

Por último, ninguno de los sistemas mencionados (SIAL Meta 4, SMLAM, SIGENO, SILOL) cuenta con tableros de control o reportes de ausentismo automatizados, lo que obliga a elaborar estos informes manualmente mediante herramientas como Excel. Sin embargo, manejar grandes volúmenes de datos con hardware limitado genera problemas de rendimiento, dificultando el análisis eficaz de la información. El sistema busca avanzar en la automatización del análisis de ausencias, implementando modelos escalables y optimizados para bases de datos grandes.

Objetivo general del proyecto

El objetivo general del proyecto es la descripción de un sistema más intuitivo, con mejor usabilidad y navegabilidad para el usuario. Diseñar, desarrollar e implementar un sistema centralizado de gestión de licencias laborales que permita optimizar procesos administrativos y garantizar la integridad de los datos.

Objetivo general del sistema

- Facilitar la solicitud y manejo de licencias dentro de la Institución
- Evitar la duplicidad de información y realización de tareas
- Recolectar y analizar datos para la gestión diaria y a futuro del personal
- Facilitar el control mediante tableros y reportes para el personal administrativo
- Extraer y descargar datos para poder realizar presentaciones
- Solicitud de licencias por parte de los usuarios.
- Validación y control por parte del sector RRHH y áreas administrativas.
- Almacenamiento seguro en base de datos MySQL.
- Acceso vía login con perfil de usuario y de administrador.
- Registro de acciones y auditoría de cambios.
- Confección de reportes, gráficos, en base al análisis de los datos, que sean descargables en diferentes formatos.
- Facilitar la solicitud, validación y control de licencias laborales mediante un sistema accesible y centralizado

Elicitación y Conocimiento del negocio

Para llevar a cabo este proyecto, se realizaron relevamientos de los sistemas preexistentes, entrevistas con el personal solicitante y administrativo, y al mismo tiempo, un análisis en base a mi experiencia personal al momento de manejar los sistemas nombrados anteriormente.

A cada uno de los actores, se le realizaron preguntas abiertas como las siguientes:

- Que categorización de usuarios se requieren.
- Como es el procedimiento habitual desde que se solicita la licencia.
- Cuáles son las acciones que deben quedar registradas en el historial, y deben ser auditadas.
- Que cantidad de usuarios pueden ingresar a los sistemas de manera simultánea.
- Que restricciones crees que existen, ya sea de hardware o software que puede afectar el rendimiento de los sistemas.
- Cuáles son los navegadores y herramientas de análisis que se usan con mayor frecuencia.

- Desde que dispositivos ingresan a los sistemas de licencias.
- Que otros tipos de datos se deben almacenar en el sistema.
- Específicamente al área de sistemas, se le consultó:
- Cuanto espacio de almacenamiento se requiere para toda la información que debe almacenarse, y con cuantos servidores se cuenta.
- El equipo de desarrollo tiene experiencia previa con HTML y Java, o se requiere capacitación adicional
- Existen preferencias por otras tecnologías que puedan complementar la arquitectura propuesta.
- Qué estándares de desarrollo deben seguirse.
- Qué nivel de seguridad debe implementarse en la comunicación cliente-servidor (por ejemplo, HTTPS)
- Qué criterios se deben cumplir para garantizar la escalabilidad del servidor
- Cómo se manejará la sincronización de datos entre el cliente y el servidor en caso de desconexión

Como respuesta a los interrogantes que se presentan, luego de realizadas las entrevistas y relevamientos, se concluye lo siguiente que será descripto en resumen:

- Hoy en día el personal que solicitara la licencia, debe dirigirse al área de administración vía telefónica o presencial, indicando el motivo de la ausencia y si es a futuro, en qué fecha se gozará de dicho derecho. Una vez realizada la solicitud, la administración carga la licencia en el sistema correspondiente detallado a continuación.
- Primero si es Planta permanente o transitoria.
- Luego se establece si es personal de Policía o Bomberos de la Ciudad, o si es personal de Ministerio.
- En tercer lugar, que tipo de licencia es.
- Finalmente se carga en el sistema indicado. Si es Personal de Ministerio se carga cualquier licencia en el sistema SIAL Meta4. SI es personal de Policía o Bomberos de la Ciudad, depende que licencia usufructuará. Si es médica en SMLAM, si no es médica en SILOL, si es cotidiana o fuera de horario administrativo en SIGENO.
- Una vez que se realiza la carga, el personal tendrá que justificar la licencia y la administración la otorgará o no. Si médica, será trabajo del área administrativa médica, audita la licencia. Sino, con el visto bueno del jefe de cada área, la administración la otorgará por sistema.

- Luego si se requiere un análisis del ausentismo o presentismo, se debe realizar manualmente en formato Excel. Solo se cuenta con un reporte con ciertas restricciones. No todo el personal cuenta con permiso para descargarlo, solo salen las licencias que se cargaron, sin importar fechas, sin filtros, y al no encontrarse las cargas de licencias, unificadas en un mismo sistema, necesito entrar a cada uno para descargar el informe correspondiente.

El informe de ausentismo, las dependencias deberían tenerlo a diario, y las dependencias que se encargan de la parte de auditoria, también deberán tenerlos de manera semanal, quincenal y mensual. Hoy en día todo esto se realiza, como dije anteriormente, en formato Excel y manual. Como respuesta a las consultas realizadas al área de sistemas obtuvimos que el sistema implementará un mecanismo de acceso mediante login con contraseñas encriptadas para proteger la información sensible, determinando si el usuario es administrativo, empleado, o auditor de RRHH (Recursos Humanos) o Médico, acompañado de un módulo de auditoría que registre todas las acciones realizadas por los usuarios y almacene un historial completo para garantizar la trazabilidad de eventos críticos y el cumplimiento de las políticas de seguridad. Se utilizarán bases de datos SQL optimizadas para manejar grandes volúmenes de datos, asegurando tiempos de respuesta inferiores a 5000 milisegundos para las operaciones más comunes y la capacidad de procesar al menos 3000 acciones por hora sin comprometer el rendimiento. Será compatible con sistemas operativos Windows y funcionará en navegadores modernos como Google Chrome, Firefox y Microsoft Edge, además de contar con un diseño responsivo para adaptarse a dispositivos con diferentes tamaños de pantalla y resoluciones, ofreciendo accesibilidad desde móviles y equipos de escritorio. El sistema requerirá un almacenamiento mínimo de 1 TB en el servidor, suficiente para gestionar datos de usuarios, licencias, documentos adjuntos, solo PDF, y registros de auditoría, asegurando una operación fluida. Se desarrollará utilizando HTML y Java, tecnologías seleccionadas por su robustez y escalabilidad, apoyadas en frameworks que optimicen su desarrollo y mantenimiento. La arquitectura será cliente-servidor, con una interfaz accesible para los usuarios y un servidor que gestione las operaciones críticas como la validación de licencias y la generación de reportes, garantizando la seguridad de la comunicación mediante protocolos como HTTPS.

Propuesta de solución

El sistema propuesto centraliza y optimiza el proceso de gestión de licencias laborales en una única plataforma desarrollada en Java con base de datos MySQL, desarrollarlo con lenguaje JAVA. El objetivo principal es reducir la fragmentación actual y evitar duplicidad de información y tareas.

Características de la solución:

- Unificación del proceso: Un solo sistema accesible para todo el personal dependiendo del tipo de usuario, para la carga, otorgamiento, validación de licencias y para la obtención de reportes.
- Interfaz intuitiva: Adaptada para empleados y administrativos, con acceso diferenciado según el perfil del usuario.
- Validación y auditoría interna: RRHH y médicos auditores podrán gestionar y validar las licencias mediante flujos de trabajo automatizados.
- Reportes automáticos: Tableros y gráficos con visualización de datos filtrables (por área, fechas, tipo de licencia).
- Módulo de auditoría: Registro detallado de cambios, accesos y operaciones realizadas por cada usuario.
- Exportación de datos: Posibilidad de exportar reportes en formatos como Excel o PDF para presentaciones administrativas. O gráficos y demás para presentaciones en Power Point.

Beneficios esperados:

- Ahorro de tiempo y recursos.
- Reducción de errores administrativos.
- Mejora en la disponibilidad y calidad de la información.
- Apoyo a la toma de decisiones con herramientas de análisis.
- Mayor transparencia en la gestión del personal.

Requisitos Específicos.

Requerimientos funcionales

Id del requerimiento	RF01
Nombre del requerimiento	Alta de Licencias
Descripción del requerimiento	El sistema debe permitir cargar los diferentes tipos de licencias, otorgándole a cada una un ID.
Autores de requerimiento	Empleado solicitante
Prioridad del requerimiento	ALTA

Id del requerimiento	RF02
Nombre del requerimiento	Validar licencias
Descripción del requerimiento	El sistema debe permitir otorgar, rechazar, suspender o anular licencias.
Autores de requerimiento	Personal auditor: RRHH o Médico
Prioridad del requerimiento	ALTA

Id del requerimiento	RF03
Nombre del requerimiento	Clasificar licencias
Descripción del requerimiento	El sistema debe permitir elegir qué tipo de licencia se solicitará. Y de ser necesario modificar la categoría.
Autores de requerimiento	Empleado solicitante, personal auditor, administrativo.
Prioridad del requerimiento	ALTA

Id del requerimiento	RF04
Nombre del requerimiento	Modificar licencia
Descripción del requerimiento	El sistema debe permitirle al usuario administrador modificar la licencia o los datos de la licencia
Autores de requerimiento	Personal auditor o administrativo.
Prioridad del requerimiento	MEDIA

Id del requerimiento	RF05
Nombre del requerimiento	Adjuntar documentación
Descripción del requerimiento	El sistema debe posibilitar la carga de documentación en cada licencia tanto para solicitarla, como para validarla.
Autores de requerimiento	Personal auditor o administrativo.
Prioridad del requerimiento	MEDIA

Id del requerimiento	RF06
Nombre del requerimiento	Listar licencias
Descripción del requerimiento	El sistema debe permitir listar las licencias cargadas.
Autores de requerimiento	Personal auditor o administrativo.
Prioridad del requerimiento	BAJA

Id del requerimiento	RF07
Nombre del requerimiento	Filtrar licencias
Descripción del requerimiento	El sistema debe permitir filtrar el listado de licencias por estado, área, tipo de licencia, tipo de persona, Id de la persona, Id de la licencia y fechas.
Autores de requerimiento	Personal auditor o administrativo.
Prioridad del requerimiento	MEDIA

Id del requerimiento	RF08
Nombre del requerimiento	Consulta de licencia
Descripción del requerimiento	El usuario debe poder consultar las licencias también con los filtros correspondientes.
Autores de requerimiento	Personal auditor o administrativo.
Prioridad del requerimiento	BAJA

Id del requerimiento	RF09
Nombre del requerimiento	Confeccionar reportes
Descripción del requerimiento	El usuario debe poder confeccionar los reportes y gráficos con los campos que elija.
Autores de requerimiento	Personal auditor
Prioridad del requerimiento	ALTA

Id del requerimiento	RF10
Nombre del requerimiento	Eliminar reportes
Descripción del requerimiento	El usuario debe poder eliminar los reportes ya utilizados.
Autores de requerimiento	Personal auditor
Prioridad del requerimiento	BAJA

Id del requerimiento	RF11
Nombre del requerimiento	Guardar reportes
Descripción del requerimiento	El usuario debe poder guardar los reportes.
Autores de requerimiento	Personal auditor o administrativo.
Prioridad del requerimiento	MEDIA

Id del requerimiento	RF12
Nombre del requerimiento	Consultar reportes
Descripción del requerimiento	El usuario debe poder consultar los reportes que ya confeccionó.
Autores de requerimiento	Personal auditor o administrativo.
Prioridad del requerimiento	BAJA

Id del requerimiento	RF13
Nombre del requerimiento	Descarga de reportes
Descripción del requerimiento	El usuario debe poder descargar los reportes en diferentes formatos.
Autores de requerimiento	Personal auditor o administrativo.
Prioridad del requerimiento	ALTA

Matriz de trazabilidad

ID Requerimiento	Requerimiento	Casos de uso relacionados
RF01	Alta de Licencias	Solicitar licencia
RF02	Validar licencias	Validar licencia
RF03	Clasificar licencias	Solicitar licencia/Modificar licencias
RF04	Modificar licencia	Validar licencias/ Modificar licencia
RF05	Adjuntar documentación	Adjuntar documentación
RF06	Listar licencias	Validar licencias/ Modificar licencia
RF07	Filtrar licencias	Validar licencias/ Modificar licencia
RF08	Consulta de licencia	Validar licencias/ Modificar licencia
RF09	Confeccionar reportes	Confeccionar reportes
RF10	Eliminar reportes	Confeccionar reportes
RF11	Guardar reportes	Confeccionar reportes
RF12	Consultar reportes	Confeccionar reportes / Gestionar reportes
RF13	Descargar reportes	Confeccionar reportes /Gestionar reportes

Restricciones – Requerimientos No funcionales

Requerimiento n°	Descripción
RFN01	Seguridad: Acceder mediante login y con contraseña encriptada, auditoria de las acciones de cada usuario, guardando su historial.
RNF02	Rendimiento: Utilización de bases de datos SQL ya que son grandes volúmenes de datos. Que las respuesta a los usuarios, no superen los 5000 milisegundos, y se puedan generar al menos 3000 acciones por hora. Reportes hasta 2000 registros en 5000 milisegundos,
RNF03	Almacenamiento: Se requiere disco rígido de al menos 1 TB.
RNF04	Compatibilidad: Funcionar en sistema operativo Windows, que se pueda trabajar en navegadores como Google Chrome, Firefox y Edge. Que el sistema sea responsive para poder navegar en pantallas de menor tamaño y resolución. Archivos PDF.
RNF05	Concurrencia: Se diseñara en un modelo cliente – servidor y múltiples descargas simultáneas.
RNF06	Lenguajes y tecnología en HTML y JAVA.

Requisitos futuros

Sistema escalable, y la posibilidad de implementar funciones nuevas a futuro.

Suposiciones y dependencias

El sistema debe poder migrar la información que ya se encuentra registrada.

Inicio de análisis – casos de uso.

A continuación se describen los análisis de caso de uso del Sistema de Gestión de Licencias

Actor	Empleado
Caso de uso	Solicitar licencia – CU01
Descripción	El usuario solicita la licencia a usufructuar por sistema, utilizando filtros para elegir la categoría, luego la que desea utilizar.
Requisitos	El sistema debe permitir la búsqueda las licencias vigentes mediante filtros, la selección y la solicitud de las mismas.
Precondición	El usuario debe estar registrado en el sistema de licencias.
Postcondición	La licencia cargada y en estado solicitado en el sistema.
Requerimiento funcional	RF01, RF03

Actor	Administrativo
Caso de uso	Modificar licencia – CU02
Descripción	El usuario filtra las licencias en estado solicitado, luego las lista y modifica el estado de cada una a Pendiente, luego de adjuntar documentación de ser necesario. Si el usuario que modifica es auditor, podrá modificar datos y estados de la misma.
Requisitos	El sistema debe permitir la búsqueda mediante filtros de las licencias solicitadas, listarlas, consultarlas y posteriormente modificar el estado y datos de la licencia en cuestión.
Precondición	El usuario debe estar registrado en el sistema de licencias.
Postcondición	La licencia pasa a estado Pendiente en el sistema.
Requerimiento funcional	RF03, RF04, RF05 ,RF06, RF07, RF08
Actor	Auditor RRHH/ Auditor Médico

Caso de uso	Validar licencias – CU03
Descripción	El usuario debe filtrar y listar todas las licencias en estado Pendiente. Si son médicas, deberán auditarlas el personal administrativo médico, y sino solo el administrativo de RRHH. Luego auditar la documentación y cambiar al estado que corresponda. Anulada /rechazada/suspendida /otorgada.
Requisitos	El sistema debe permitir la búsqueda mediante filtros de las licencias pendientes, otorgadas o anuladas, listarlas, consultarlas y posteriormente modificar el estado de ellas adjuntando también documentación de ser necesario.
Precondición	El usuario debe estar registrado en el sistema de licencias.
Postcondición	La licencia pasa a estado Anulada/rechazada/suspendida/otorgada en el sistema.
Requerimiento funcional	RF02, RF03, RF04, RF05 ,RF06, RF07, RF08

Actor	Auditor RRHH/ Auditor Médico
Caso de uso	Confeccionar reportes – CU04
Descripción	Los usuarios deberán poder elegir que información deben presentar para confeccionar los reportes, guardarlos y eliminarlos. Luego se envía a los administrativos para que tomen conocimiento.
Requisitos	El sistema debe permitir seleccionar los datos para confeccionar tablas y gráficos.
Precondición	Deberán estar cargadas las licencias las cuales impactarán para realizar los reportes.
Postcondición	Reportes confeccionados, guardados o eliminarlos, si lo requiere el usuario.
Requerimiento funcional	RF09, RF11

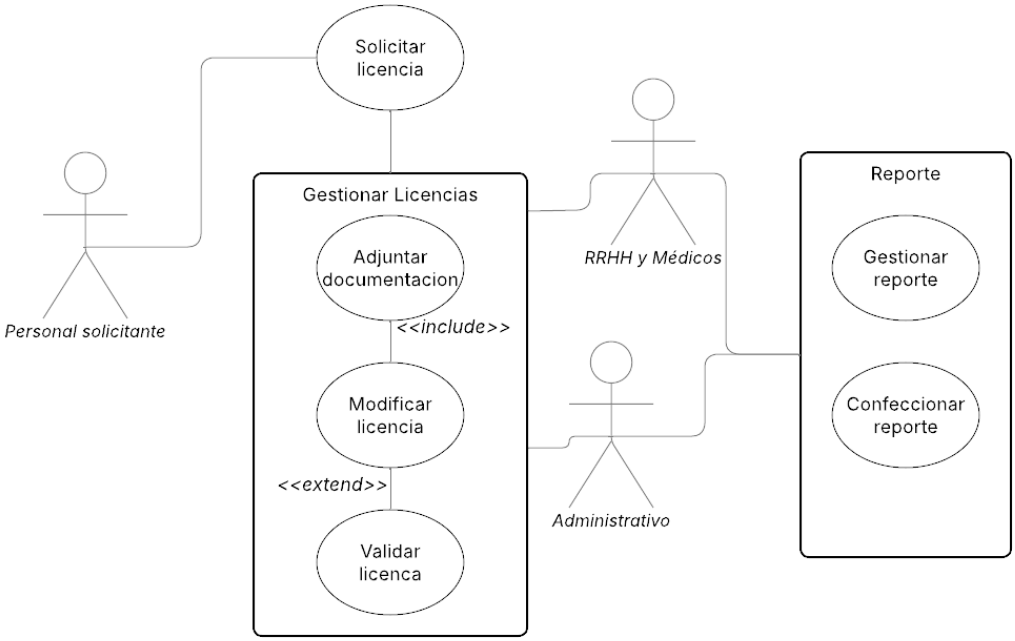
Actor	Administrativo , Auditor RRHH o Auditor Médico
Caso de uso	Gestionar reportes – CU05
Descripción	Los usuarios deben poder consultar los reportes confeccionados y descargarlos,
Requisitos	El sistema debe permitir ver los reportes confeccionados y guardados.
Precondición	Reportes confeccionados y guardados
Postcondición	Reportes confeccionados y descargados.
Requerimiento funcional	RF10, RF11, RF12, RF13

Actor	Administrativo , Auditor RRHH o Auditor Médico
Caso de uso	Adjuntar documentación CU06
Descripción	Los usuarios deben adjuntar documentación para pasar la licencia de estado solicitada a pendiente, por lo que es obligatoria en primera instancia, y si la licencia está pendiente, y es para validar, el auditor, debe poder adjuntar documentación de manera optativa.
Requisitos	El sistema debe permitir adjuntar documentación, en formato PDF, JPG, JPEG hasta 10 Kb.
Precondición	Licencia en estado Solicitado para administrativos, y pendiente para auditor.
Postcondición	Archivos adjuntos en las licencias.
Requerimiento funcional	RF05

Diagramas de casos de uso

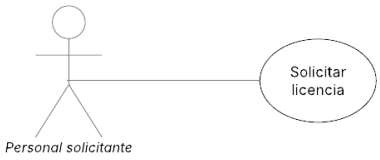
En estos diagramas se representan las principales acciones que los usuarios realizan. Estas son, por ejemplo, validar, solicitar o modificar licencias.

Diagrama general

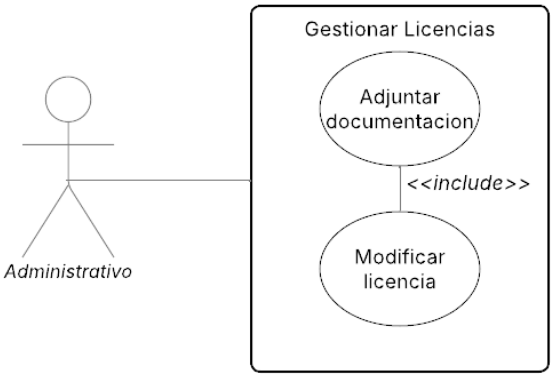


INDICE

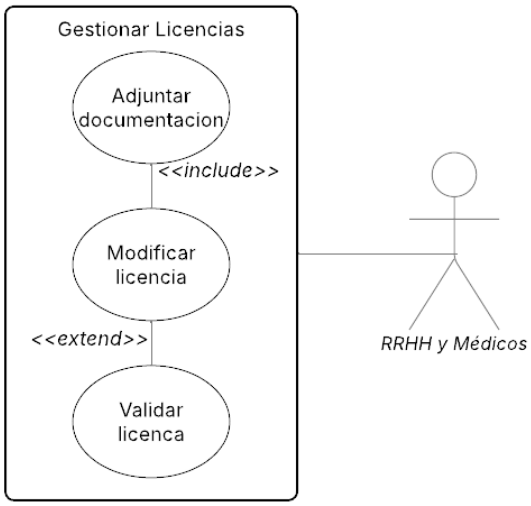
Caso de uso 1 “Solicitar Licencia”



Caso de uso 2 “Modificar Licencia” y Caso de uso 6 “Adjuntar documentación”



Caso de uso 3 “Validar licencia” y Caso de uso 6 “Adjuntar documentación”



INDICE

Caso de uso 4 “Confeccionar Reporte” y Caso de uso 5 “Gestionar Reporte”

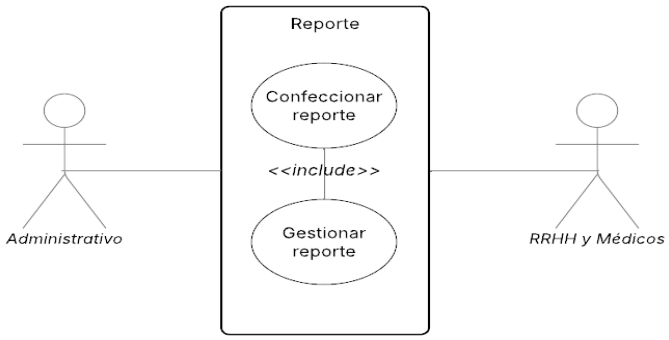
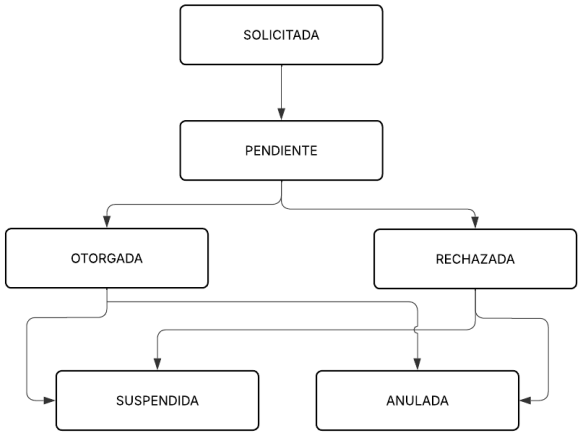


Diagrama de transición de estados



Descripción del diagrama de transición

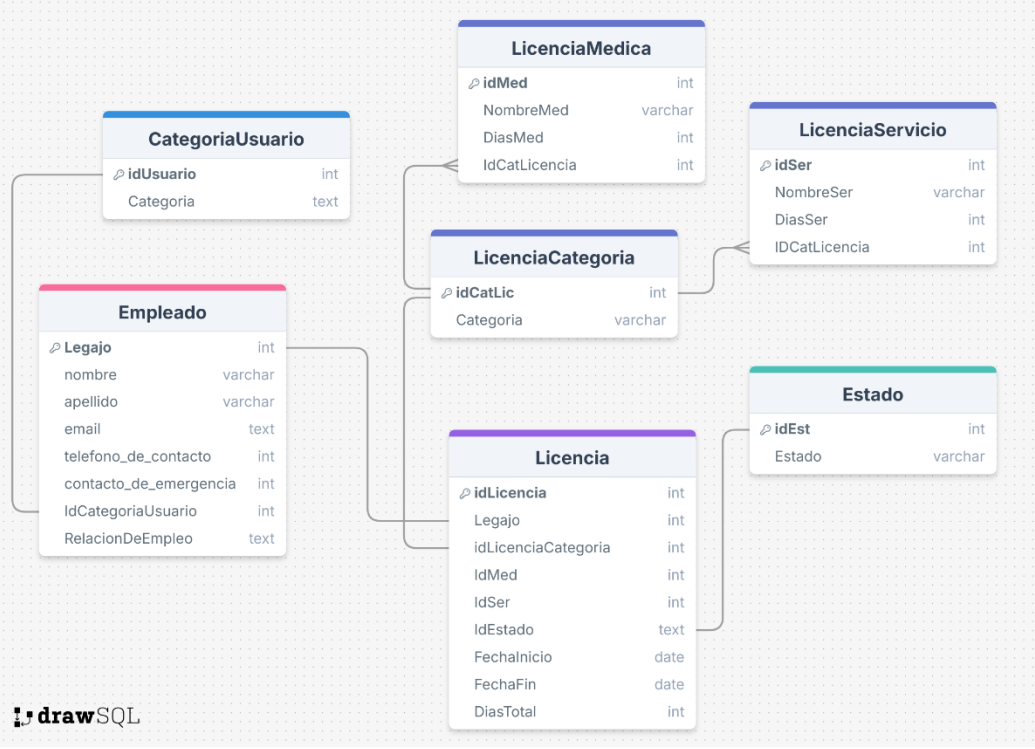
El empleado ingresa al sistema, elije la categoría de la licencia a solicitar, la cual puede ser médica o del servicio. Luego elije que licencia de las que están vigentes en la ley de contratos de trabajo y de la Institución, carga los datos y cuando confirma, la licencia queda en estado “solicitada”.

Luego presenta la documentación a los administradores, los cuales la adjuntarán, si está completa, y si no, les solicitan la documentación faltante. Una vez completa la documentación, la adjuntan y la licencia pasa a estado Pendiente.

Si la licencia en cuestión es médica, el médico auditor verifica la documentación, y otorga o rechaza la licencia. Lo mismo si la licencia es de servicio, el auditor será Recursos Humanos siguiendo el mismo procedimiento.

Las licencias otorgadas o rechazadas, pueden cambiar de estado a suspendidas o anuladas, por diferentes motivos. Esta acción solo podrá ser realizada por los auditores.

Diagrama de Clase



Descripción de diagrama de SQL

Empleado

- Legajo: Legajo único de la persona. Clave Primaria
- IdCategoriaUsuario: tipo de usuario (empleado, administrativo, auditoria RRHH, auditoria médico)
- RelacionDeEmpleo: Define si el empleado es de planta permanente o transitoria, de Ministerio, Policía, Bomberos de la ciudad.

CategoriaUsuario

- Define las categorías de usuario en base a sus permisos y accesos.

Licencia

- Se relaciona con el empleado por medio del legajo, y con las categorías de licencias.
- Atributos como el nombre y cantidad de días, varia en base a la licencia solicitada

LicenciaCategoria

- Define las categorías de las licencias que pueden ser, médicas o de servicio.

LicenciaMedica

- Tabla que contendrá toda la descripción de las licencias médicas.

LicenciaServicio

- Tabla con la descripción de cada licencia vigente de servicio.

Estado

- Define el estado en el cual puede estar la licencia. Solicitada/Rechazada/Anulada/Otorgada/Pendiente.

Aplicación del proceso unificado de desarrollo

Fase 1 – Inicio

- Relevamiento del problema actual.
- Identificación de actores (empleado, administrativo, RRHH, médico auditor).
- Análisis de riesgos: migración de datos, dependencia de múltiples sistemas, resistencia al cambio.
- Estimación inicial de tiempos y recursos disponibles.

Fase 2 – Elaboración

- Modelado del dominio: entidades como Empleado, Licencia, Estado, LicenciaCategoria.
- Diseño de arquitectura cliente-servidor (Java + MySQL).
- Definición de los casos de uso prioritarios: Solicitar licencia, Validar licencia, Generar reportes.
- Identificación de requerimientos funcionales y no funcionales.

Fase 3 – Construcción

- Desarrollo del prototipo operacional con módulos iniciales:
- Autenticación
- Solicitud y validación de licencias
- Generación de reportes
- Pruebas funcionales por iteración.
- Ajustes según feedback de usuarios administrativos.
-

INDICE

Fase 4 – Transición

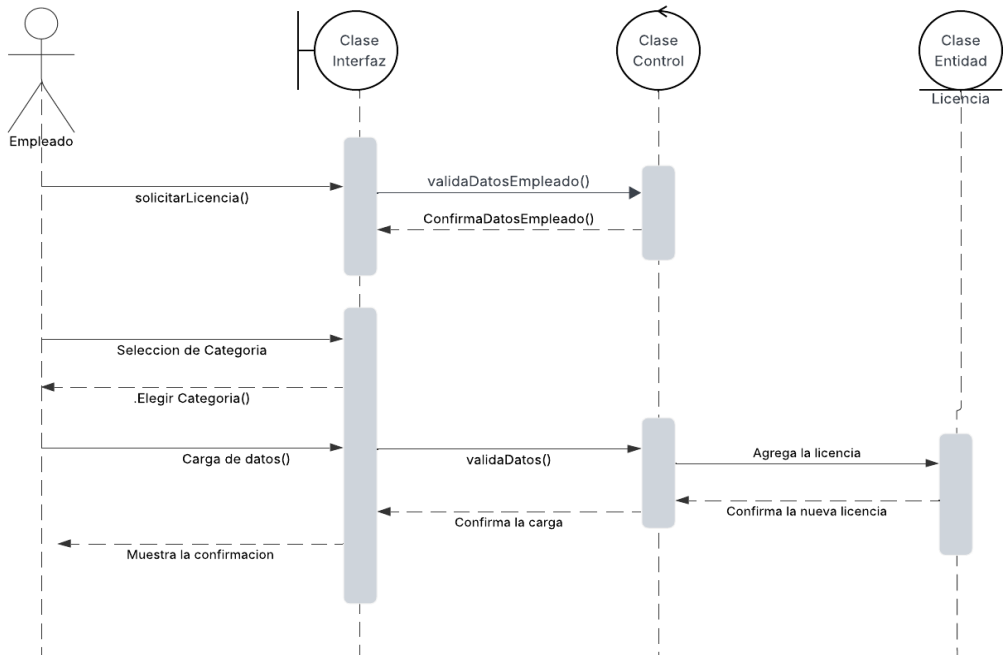
- Capacitación a usuarios (empleados y administrativos).
- Migración de datos de los sistemas actuales.
- Puesta en producción y seguimiento post-implementación.

Etapa de Análisis

Diagramas de secuencia

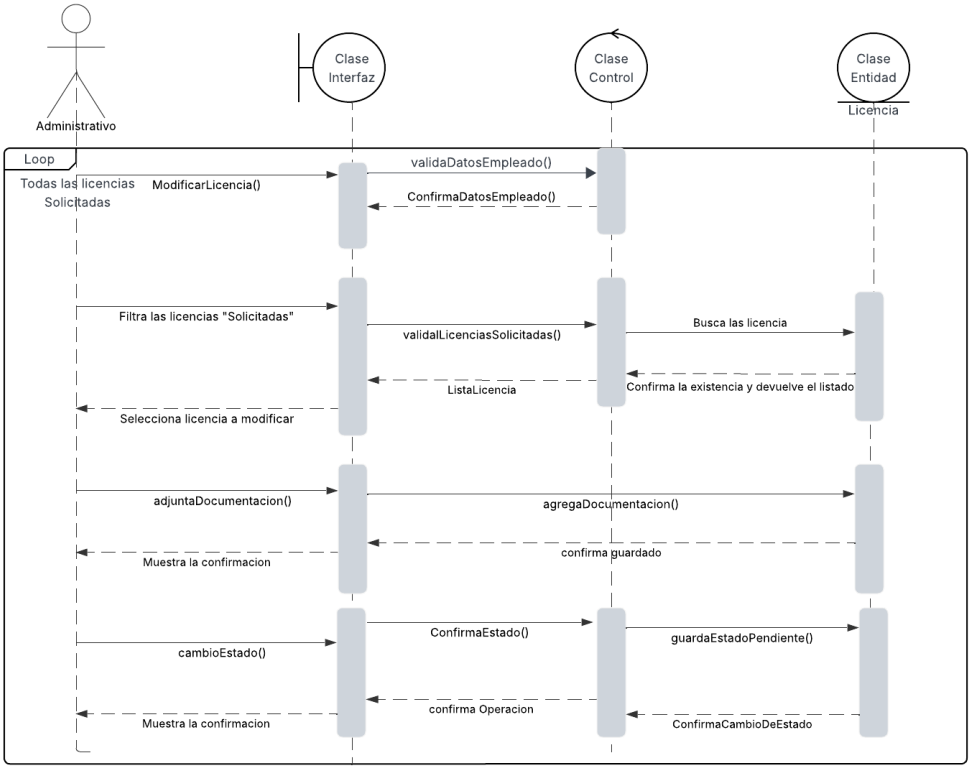
En estos diagramas represento la comunicación entre los objetos y el sistema para llevar a cabo funciones específicas. Permite que visualice la lógica interna del sistema y como fluye la información en el mismo.

CU01

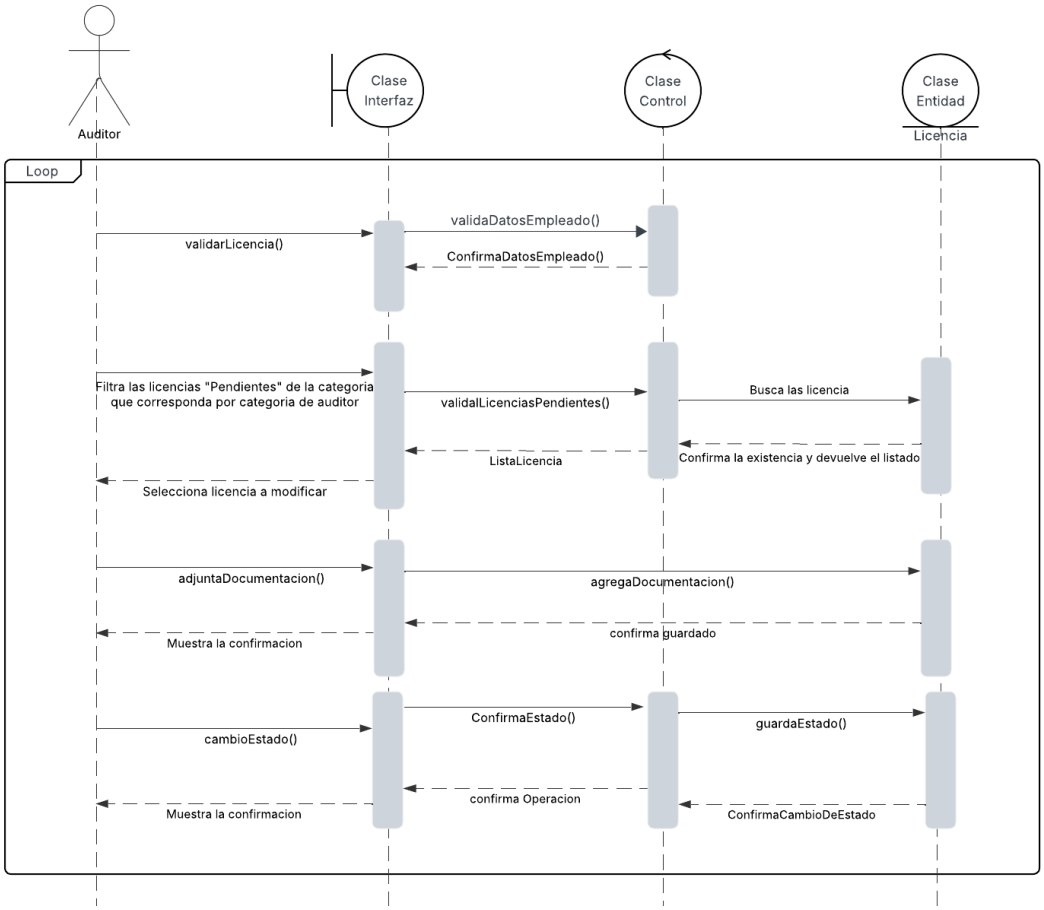


INDICE

CU02 Y CU06

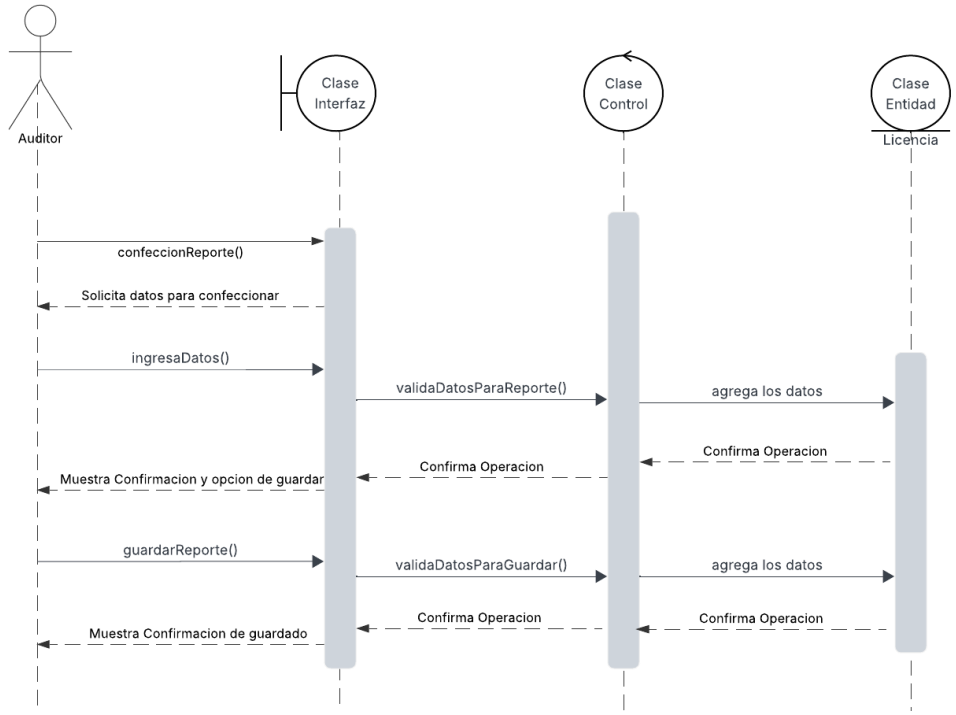


CU03 Y CU06

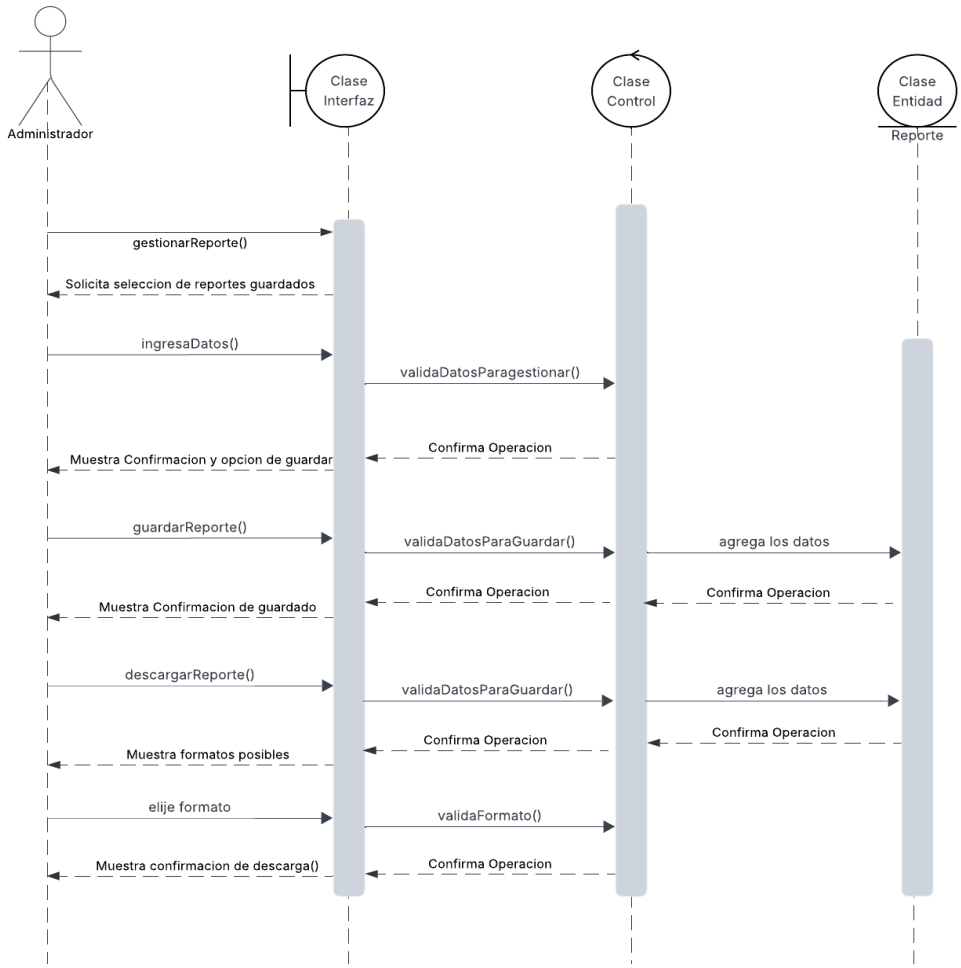


INDICE

CU04



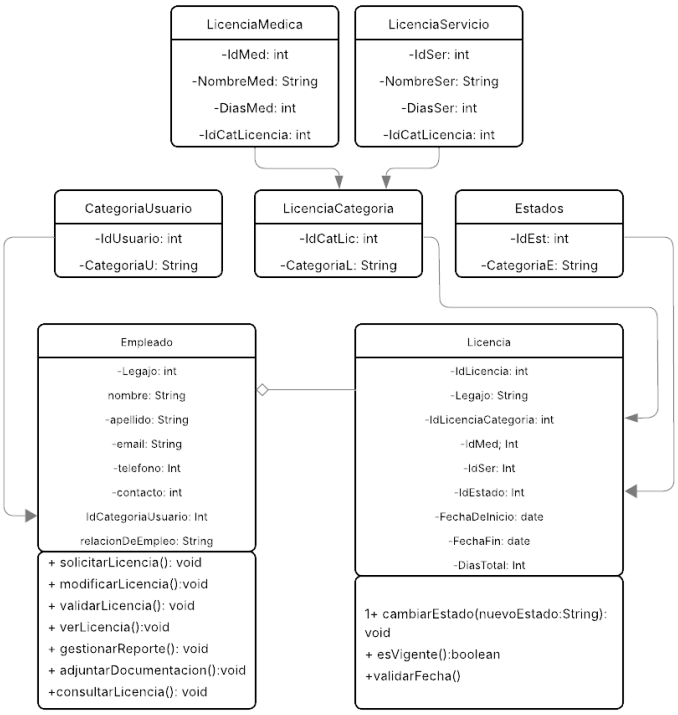
CU05



INDICE

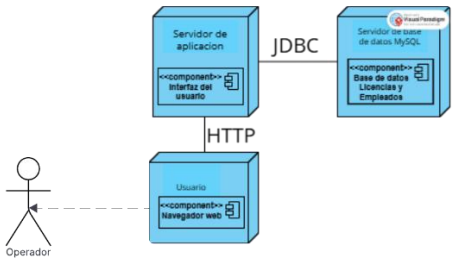
Etapa de Diseño

Diagrama de clase diseño: Modela las clases que componen la lógica del sistema, sus atributos y métodos. Refleja cómo se relacionan entre sí entidades como Empleado, Licencia, Estado o CategoriaUsuario, y qué operaciones puede realizar cada una. Sirve como base para la programación orientada a objetos.



Etapa de Implementación

Diagrama de despliegue: Describe cómo se distribuyen los componentes del sistema en la infraestructura física. En tu caso, representa la arquitectura cliente-servidor, mostrando los elementos como navegador web (cliente), servidor de aplicación y base de datos, y cómo se comunican entre ellos.



INDICE

Etapa de pruebas.

Plan de prueba

Para ejecutar un plan de pruebas completo, aunque sabemos que no es posible realizar pruebas exhaustivas, se realizan pruebas funcionales y no funcionales. Se llevan a cabo pruebas unitarias, de integración, de sistemas, de escalabilidad, seguridad y rendimiento entre otras. Se utilizan técnicas de caja negra y caja blanca, como también el uso de valores frontera, entre otros.

Caso de Uso	**N° Caso de prueba**	**Caso de prueba**	**Nivel de prueba**
CU01	CPC01	Solicitar licencia: validar fechas	Componente
CU01	CPC02	Solicitar licencia: Calcular días	Componente
CU02	CPS03	Modificar licencia: cambiar estado siendo administrador	Sistema
CU03	CPI04	Validar licencia: aplicar filtro por estado "Pendiente"	Integración
CU04	CPS05	Administrar reporte: guardar correctamente	Sistema
CU05	CPS06	Descargar reporte en Excel	Sistema
CU06	CPC07	Adjuntar archivo con formato no permitido	Componente
CU01	CPC08	Validar campo obligatorio no completado	Componente
CU01	CPC09	Solicitar licencia: validación con código (caja blanca)	Componente
CU05	CPS10	Intentar exportar más de 5000 registros	Sistema
CU04	CPS11	Generar reporte con hasta 2000 registros	Sistema
CU04	CPS12	Múltiples usuarios generan reportes simultáneamente	Sistema
CU05	CPS13	Descargar archivo sin conexión o con error de red	Sistema
CU04	CPS14	Generación de reporte en móviles	Sistema
CU01	CPA15	Solicitar licencia	Aceptación

N° Caso de prueba	Tipo de prueba	Técnica	Entrada de prueba	Resultado esperado
CPC01	Funcional	Caja negra - Validación de campos	Fecha inicio 01/04/2025 Fecha fin 01/03/2025	"Error: Fecha fin menor a la fecha de inicio"
CPC02	Funcional	Caja negra – Partición + Frontera	Fecha inicio 01/04/2025 Fecha fin 01/03/2025 Calculo -30	"Error en fechas cantidad de días inexistente"
CPS03	Funcional / Seguridad	Caja negra – Restricción por rol	Usuario administrador intenta cambiar a "Otorgada"	Acción bloqueada. Se muestra mensaje: "No tiene permiso para modificar estado".
CPI04	Funcional	Caja negra – Prueba de selección	Filtro aplicado = "Pendiente"	Solo se muestran licencias con estado = Pendiente.
CPS05	Funcional	Caja negra – Persistencia	Botón “Guardar reporte” presionado	Reporte se guarda correctamente en la base o como archivo.
CPS06	Funcional	Caja negra – Exportación	Solicitud de descarga en formato `.xlsx`	Descarga exitosa del archivo con los datos esperados.
CPC07	Funcional / Seguridad	Caja negra – Validación de extensión	Archivo `.docx` en vez de `.pdf`	Rechaza el archivo. Muestra mensaje: "Solo se permite formato .pdf".
CPC08	Funcional	Caja negra – Validación de campos	Formulario sin completar campo obligatorio	El sistema bloquea el envío y muestra advertencia de campo obligatorio.

CPC09	Funcional	Caja blanca – Cobertura de sentencias	Método calcularDiasTotales(fechaInicio, fechaFin) Inicio 01/07/2025 - fin 15/07/2025	Se prueban rutas de código para distintos valores, se valida cálculo correcto.
CPS10	Rendimiento / Límite	Caja negra – Límite	Solicitud de descarga con >5000 registros	Rechaza la acción. Muestra mensaje: "Solo se permite hasta 2000 registros."
CPS11	Rendimiento	Caja negra – Tiempo de respuesta	Solicitud con 2000 registros	Reporte generado en menos de 5 segundos, sin errores.
CPS12	Rendimiento / Concurrencia	Caja negra – Estrés / carga	10 usuarios generan reportes a la vez	El sistema no se bloquea, responde a todos sin error ni demora significativa.
CPS13	Robustez / Tolerancia	Caja negra – Simulación de fallo	Simulación de red desconectada al intentar descarga	Muestra mensaje de error: "Sin Conexión", permite reintentar.
CPS14	Usabilidad / Compatibilidad	Caja negra – Exploratoria	Usuario accede desde móvil y genera reporte	Interfaz adaptada, botones funcionales, descarga exitosa.
CPA15	Navegabilidad/ Usabilidad	Beta	El usuario ingresa los campos, y solicita la licencia mediante el botón de confirmar. Categoría de licencia: Servicio, Tipo de licencia: Vacaciones, Inicio 01/07/2025 - fin 15/07/2025	"Solicitud Exitosa"

INDICE

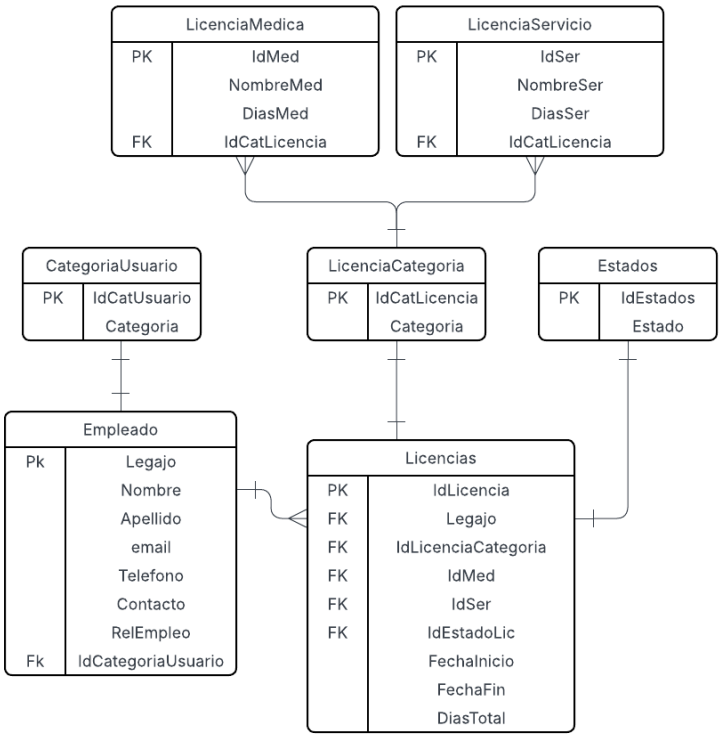
Definición de base de datos para el sistema.

Para este sistema se utiliza una base de datos relacional en MySQL, ya que es una opción práctica, segura y compatible con los requerimientos del proyecto. MySQL permite trabajar con grandes volúmenes de información y es ideal para garantizar el rendimiento del sistema.

La base de datos almacena la información más importante, como los datos de los empleados, las licencias solicitadas, los estados de cada trámite y las categorías de usuarios. Esto asegura que la información esté organizada, disponible y que se puedan generar reportes y estadísticas cuando sea necesario.

Además, como estamos trabajando con un prototipo y siguiendo el Proceso Unificado de Desarrollo, las clases que se diseñaron en las etapas anteriores dieron lugar a las tablas que forman esta base de datos, asegurando coherencia entre el modelo y la implementación. Las tablas serán “Licencia”, “Estados”, “Empleados”, “CategoriaLicencia”, ”CategoriaUsuario”, ”Estados”, “LicenciaMedica”, “LicenciaServicio”.

Diagrama Entidad Relación: Representa las tablas de la base de datos, con sus claves primarias (PK), foráneas (FK) y relaciones entre entidades. Permite visualizar cómo se estructura la información que maneja el sistema y cómo se conectan datos como empleados, tipos de licencia y estados.



INDICE

Creación de tablas

```
CREATE TABLE CategoriaUsuario (
  idUsuario INT PRIMARY KEY,
  CategoriaU VARCHAR(100)
);

CREATE TABLE Empleado (
  Legajo INT PRIMARY KEY,
  nombre VARCHAR(100),
  apellido VARCHAR(100),
  email TEXT,
  telefono_de_contacto INT,
  contacto_de_emergencia INT,
  RelacionDeEmpleo TEXT,
  IdCategoriaUsuario INT,
  FOREIGN KEY (IdCategoriaUsuario) REFERENCES CategoriaUsuario(idUsuario)
);
```

RESULTS

Tables_in_SeminarioInformatica
CategoriaUsuario
Empleado
Estado
Licencia
LicenciaCategoria
LicenciaMedica
LicenciaServicio

Resultados:

Inserción, borrado y consulta de registros

Insertar Datos de prueba

```
INSERT INTO LicenciaMedica (idMed, NombreMed, DiasMed, IdCatLicencia)
VALUES (501, 'Reposo por enfermedad común', 3, 10),
(502, 'Familiar', 2, 10),
(506, 'Internación breve', 5, 10),
(503, 'Consulta médica con reposo', 1, 10),
(504, 'Licencia por intervención quirúrgica menor', 7, 10);

INSERT INTO LicenciaServicio (idSer, NombreSer, DiasSer, IdCatLicencia)
VALUES
(601, 'Curso obligatorio de formación', 3, 11),
(602, 'Licencia por tareas extraordinarias', 2, 11),
(603, 'Permiso institucional', 1, 11);
```

Consulta de datos

```sql
SELECT
  e.nombre AS NombreEmpleado,
  e.apellido AS ApellidoEmpleado,
  lc.Categoria AS TipoLicencia,
  lm.NombreMed AS LicenciaMedica,
  ls.NombreSer AS LicenciaServicio,
  est.Estado AS EstadoActual,
  l.FechaInicio,
  l.FechaFin,
  l.DiasTotal
FROM Licencia l
JOIN Empleado e ON l.Legajo = e.Legajo
JOIN LicenciaCategoria lc ON l.idLicenciaCategoria = lc.idCatLic
LEFT JOIN LicenciaMedica lm ON l.IdMed = lm.idMed
LEFT JOIN LicenciaServicio ls ON l.IdSer = ls.idSer
JOIN Estado est ON l.IdEstado = est.idEst;

SELECT * from Licencia

```

Resultado

RESULTS

EXPORT

NombreEmpleado	ApellidoEmpleado	TipoLicencia	LicenciaMedica	LicenciaServicio	EstadoActual	FechaInicio	FechaFin	DiasTo
Camila	Caracciolo	Licencia ...	Reposo por ...		Solicitada	2025-05-...	2025-0...	3
Lucas	Pérez	Licencia d...		Curso obligat...	Solicitada	2025-05-...	2025-0...	3
Sofía	Gómez	Licencia d...		Licencia por t...	Solicitada	2025-05-...	2025-0...	2
Mateo	Ramírez	Licencia d...		Permiso insti...	Solicitada	2025-05-...	2025-0...	1

Borrado de registros

```sql
DELETE FROM Licencia
WHERE idLicencia = 9001;

```

Interfaz Gráfica para solicitar licencias

Presentación de las consultas SQL.

Repositorio de Github

Definición de comunicaciones

Para el desarrollo de este sistema se utiliza una arquitectura cliente-servidor, donde la interfaz será accesible desde navegadores modernos y estará diseñada en HTML y Java, permitiendo que los empleados y administrativos puedan ingresar según su perfil. La comunicación entre el cliente y el servidor se realiza a través del protocolo HTTPS para garantizar la seguridad de los datos. El servidor se encarga de procesar las solicitudes, validar licencias, generar reportes y conectarse con la base de datos MySQL, donde se almacena toda la información relevante. De esta forma, se asegura una interacción fluida, segura y ordenada entre todos los componentes del sistema.

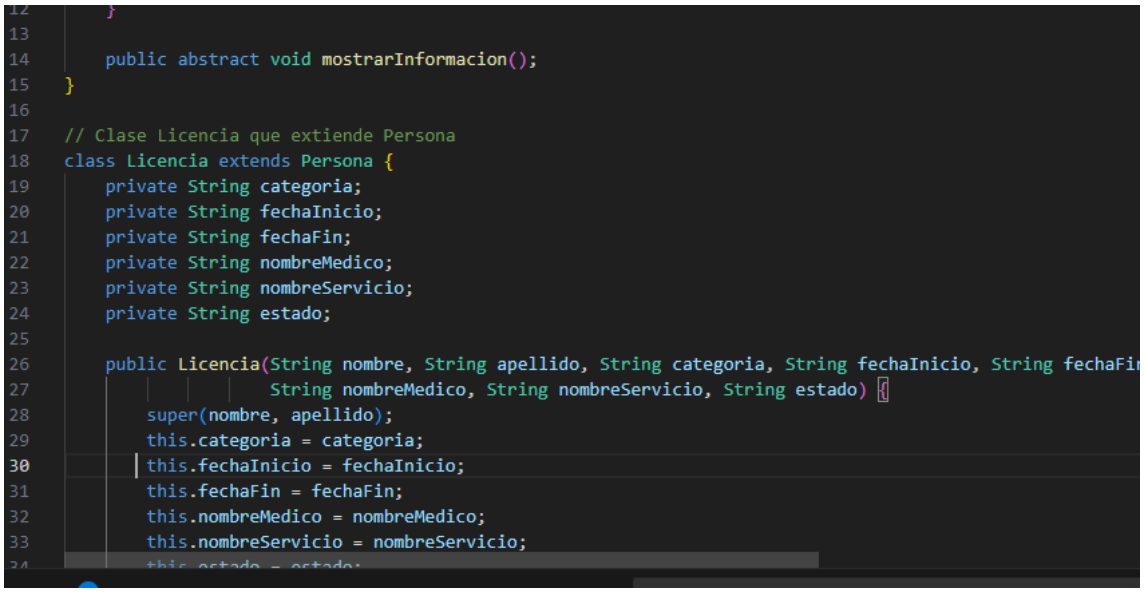
Explicación del desarrollo en Java

En el desarrollo del sistema de gestión de licencias laborales, utilicé diferentes funcionalidades que proporciona Java para cumplir con los requisitos del proyecto. Apliqué los principios de Programación Orientada a Objetos, como encapsulamiento, herencia, polimorfismo y abstracción, además de estructuras como ArrayList, manejo de excepciones, uso de constructores, conexión con base de datos mediante JDBC, y una organización del código en capas lógicas.

Comencé creando una clase abstracta llamada Persona, que define atributos comunes como nombre y apellido, y un método abstracto mostrarInformacion(). Esta clase no puede instanciarse directamente, y sirve como base para otras clases.

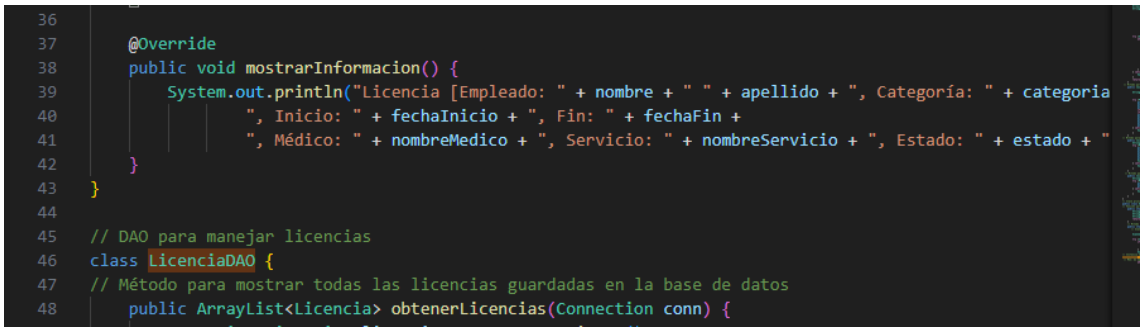
```
5   abstract class Persona {
6       protected String nombre;
7       protected String apellido;
8
9       public Persona(String nombre, String apellido) {
10          this.nombre = nombre;
11          this.apellido = apellido;
12      }
13
14      public abstract void mostrarInformacion();
15  }
16
```

Luego implementé la clase Licencia (línea 18), que hereda de Persona y agrega nuevos atributos como dniEmpleado, fechaInicio, fechaFin, diasLicencia, además de los datos del médico. En su constructor (línea 28) uso super() para inicializar los campos heredados.



También implementé encapsulamiento declarando los atributos como private o protected (líneas 18–24), y accediendo a ellos mediante métodos públicos get y set. Esto protege los datos internos del objeto.

Para aplicar polimorfismo, sobreescribí el método mostrarInformacion() en la clase Licencia (línea 46), permitiendo que cada clase muestre sus datos a su manera.



En cuanto a la persistencia de datos, utilicé JDBC para conectarme a una base de datos MySQL. En la clase LicenciaDAO (línea 55), usé Connection, DriverManager, PreparedStatement y ResultSet (líneas 58–73) para ejecutar una consulta SQL que une las tablas empleado, médico y licencia. Los resultados se recorren con un while (línea 62), y por cada fila se crea un objeto Licencia (línea 73), que se guarda en un ArrayList (línea 49). Esta estructura me permite almacenar muchos objetos de forma dinámica y sin tamaño fijo.

```java
class LicenciaDAO {
// Método para mostrar todas las licencias guardadas en la base de datos
    public ArrayList<Licencia> obtenerLicencias(Connection conn) {
        ArrayList<Licencia> licencias = new ArrayList<>();
        String query = "SELECT e.nombre, e.apellido, lc.Categoria, l.FechaInicio, l.FechaFin, " +
                "IFNULL(lm.NombreMed, 'Sin médico') AS Medico, " +
                "IFNULL(ls.NombreSer, 'Sin servicio') AS Servicio, " +
                "IFNULL(est.Estado, 'Sin estado') AS Estado " +
                "FROM licencia l " +
                "LEFT JOIN empleado e ON l.Legajo = e.Legajo " +
                "LEFT JOIN licenciacategoria lc ON l.idLicenciaCategoria = lc.idCatLic " +
                "LEFT JOIN licenciamedica lm ON l.IdMed = lm.idMed " +
                "LEFT JOIN licenciaservicio ls ON l.IdSer = ls.idSer " +
                "LEFT JOIN estado est ON l.IdEstado = est.idEst";

        try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
            while (rs.next()) {
                Licencia licencia = new Licencia(
                        rs.getString(columnLabel:"nombre"),
                        rs.getString(columnLabel:"apellido"),
                        rs.getString(columnLabel:"Categoria"),
                        rs.getString(columnLabel:"FechaInicio"),
                        rs.getString(columnLabel:"FechaFin"),
                        rs.getString(columnLabel:"Medico"),
                        rs.getString(columnLabel:"Servicio"),
                        rs.getString(columnLabel:"Estado")
                );
                licencias.add(licencia);
            }
        } catch (SQLException e) {
            System.err.println("Error al obtener licencias: " + e.getMessage());
```

La vista del sistema está representada por el método main, donde se implementa un menú por consola. Al seleccionar la opción “ver licencias”, se llama al método obtenerLicencias() del DAO, y luego se muestra la información llamando a mostrarInformacion() sobre cada objeto.

Además, implementé el manejo de excepciones con bloques try-catch para evitar que errores en la conexión o la consulta detengan el programa de forma inesperada. Finalmente, usé constructores para inicializar objetos correctamente, tanto en Persona como en Licencia.

```java
// Instancia de la clase DAO para acceder a la base de datos
        LicenciaDAO dao = new LicenciaDAO();
// Menú de opciones
        while (true) {
            System.out.println("\n--- MENÚ ---");
            System.out.println("1. Ver licencias");
            System.out.println("2. Registrar nueva licencia");
            System.out.println("3. Modificar licencia existente");
            System.out.println("4. Eliminar licencia");
            System.out.println("5. Salir");

            int opcion = sc.nextInt();
            sc.nextLine();

            switch (opcion) {
                case 1:
                    ArrayList<Licencia> licencias = dao.obtenerLicencias(conn);
                    if (licencias.isEmpty()) {
                        System.out.println("No hay licencias registradas.");
                    } else {
                        for (Licencia l : licencias) l.mostrarInformacion();
                    }
                    break;
                case 2:
                    try {
                        System.out.println("ID de licencia:");
                        int idLicencia = sc.nextInt();
                        System.out.println("Legajo del empleado:");
                        int legajo = sc.nextInt();
                        System.out.println("ID Categoría de licencia:");
```

```java
class LicenciaDAO {
    public void eliminarLicencia(Connection conn, int idLicencia) {
            System.err.println("Error al eliminar licencia: " + e.getMessage());
        }
    }
}
// Clase principal que maneja todo el sistema
public class SistemaLicencias_ModificadoFinal {
// Función para conectar con la base de datos.
    public static Connection conectarBD() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/licenciasdb";// URL de la base de datos
        String usuario = "root";// Usuario de la base.
        String contrasena = "Luca1905+";// Contraseña del usuario
        return DriverManager.getConnection(url, usuario, contrasena);// Se devuelve la conexión.
    }
    // Método principal
    Run | Debug
    public static void main(String[] args) {
        // Scanner para leer entrada del usuario

        Scanner sc = new Scanner(System.in);

        try (Connection conn = conectarBD()) {
            if (conn == null) {
                System.out.println("No se pudo conectar a la base de datos.");
                return;
            }
            System.out.println("Conexión exitosa.");
// Instancia de la clase DAO para acceder a la base de datos
            LicenciaDAO dao = new LicenciaDAO();
```

```java

        try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
            while (rs.next()) {
                Licencia licencia = new Licencia(
                        rs.getString("nombre"),
                        rs.getString("apellido"),
                        rs.getString("Categoria"),
                        rs.getString("FechaInicio"),
                        rs.getString("FechaFin"),
                        rs.getString("Medico"),
                        rs.getString("Servicio"),
                        rs.getString("Estado")
                );
                licencias.add(licencia);
            }
        } catch (SQLException e) {
            System.err.println("Error al obtener licencias: " + e.getMessage());
        }
        return licencias;
    }
// Método para insertar una nueva licencia en la base de datos.
    public void registrarLicencia(Connection conn, int idLicencia, int legajo, int idCatLic, Integer idMed, Integer idSer
```


INDICE

Presentación del desarrollo en Java

Repositorio de Github - "SistemaLicenciasCompleto.java

Bibliografía

- Kendall, K. E., & Kendall, J. E. (2011). Análisis y diseño de sistemas (8va ed.). Pearson Educación.
- Pressman, R. S. (2010). Ingeniería del software: un enfoque práctico. McGraw-Hill.
- Jacobson, I., Booch, G., & Rumbaugh, J. (1999). El proceso unificado de desarrollo. Addison-Wesley.
- Sommerville, I. (2011). Ingeniería de software (9na ed.). Pearson Educación.
- Módulo 1, 2, 3 y 4, "Prueba de Sistemas", Canvas Siglo XXI
- UTM, (s.f.) Usabilidad. http://www.utm.mx/usabilidad.html
- Ecured, (s.f.) Usabilidad. https://www.ecured.cu/Usabilidad
- Módulo 1, 2, 3 y 4, "Ingeniería de software", Canvas, Siglo XXI
- Módulo 1, 2 y 3, "Seminario de Práctica de Informática", Canvas, Siglo XXI