Bartoluche, Matías Jesús

Software para la administración de personal de la construcción

2024

Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía

Carrera de ingeniería en informática

Práctica profesional supervisada
Informe final

Software para la administración de personal de la construcción

Autor: Bartoluche Matías Jesús

Florencio Varela, diciembre 2024

Estudiante

Apellido y nombres: Bartoluche Matías Jesús

Correo electrónico: bartoluche.matias@gmail.com

Organización donde se realiza la práctica profesional supervisada

KYDMA S.R.L

Tutor organizacional

Apellido y nombres: Alegre Ariel Andrés

Correo electrónico: kydmasrl.construcciones@outlook.com

Docente supervisor

Apellido y nombres: Dr. Conde, Sergio Daniel

Correo electrónico: sconde@unaj.edu.ar

Docente tutor del taller de apoyo para la producción de textos académicos

Apellido y nombres: Leone, Nelson

Correo electrónico: nelsonleone2012@gmail.com

Coordinador de la carrera de ingeniería en informática

Apellido y nombres: Dr. Morales, Martín

Correo electrónico: martin.morales@unaj.edu.ar

Resumen

Este trabajo presenta el informe final de la práctica profesional supervisada realizada en el área de sistemas de una empresa del sector privado. El principal objetivo es analizar el flujo de trabajo y la infraestructura informática, identificar las necesidades de la organización y desarrollar un sistema de gestión administrativo que cumpla con las expectativas y requerimientos de la empresa tales como facilitar el almacenamiento de información y agilizar las tareas administrativas.

El informe describe conceptos como el ciclo de vida del software, la gestión de un proyecto y la aplicación de estos en un ambiente laboral real. Abarca temas como planeación, documentación de requerimientos, alcance, evaluación de riesgos, diseño, arquitectura de software, desarrollo, despliegue, pruebas y soporte técnico.

En base a los requerimientos del cliente y su infraestructura informática se evaluaron diversas tecnologías tales como lenguajes de programación, frameworks y bases de datos para el desarrollo de un proyecto que pueda satisfacer las necesidades del cliente de forma eficiente y que se pueda adaptar a su infraestructura informática.

Palabras clave: ciclo de vida del software, gestión de proyecto, aplicación web, servlet.

Abstract

This work presents the final report of the supervised professional practice carried out in the systems area of a private sector company. The main objective is to analyze the workflow and computer infrastructure, identify the needs of the organization and develop an administrative management system that meets the expectations and requirements of the company such as facilitating the storage of information and streamlining administrative tasks. The report describes concepts such as the software life cycle, project management and application of these in a real work environment. It covers topics such as planning, requirements documentation, scope, risk assessment, design, software architecture, development, deployment, testing, and technical support. Based on the client's requirements and its computer infrastructure, various technologies such as programming languages, frameworks and databases were evaluated for the development of a project that can satisfy the client's needs efficiently and that can be adapted to its infrastructure. computing.

Dedicatorias y agradecimientos

En primer lugar debo agradecer a mi madre, alguien que estuvo en todo momento de mi vida. Me ha enseñado a ser una persona íntegra y me ha alentado para que no baje los brazos a lo largo de la carrera.

En segundo lugar, agradezco a mi familia, padre y hermanos. Sin duda alguna ha sido un apoyo todo este tiempo.

También agradezco a la familia Tello por recibirme y apoyarme, su hogar es mi segundo hogar y mi segunda oficina de trabajo en la que he pasado varias noches en vela completando actividades de la universidad.

Agradezco a los docentes de la UNAJ con los que aprendí y compañeros con los que compartí a lo largo de este camino.

Contenido

1. Introducción

En el ámbito de la empresa KYDMA S.R.L se ha llevado a cabo esta práctica profesional supervisada que da como resultado este informe final.

Este documento presenta el ciclo de vida del desarrollo de software solicitado por la empresa en cuestión, se presentan las etapas del proyecto desde su fase inicial hasta la implementación y utilización del producto final. Su objetivo es presentar una metodología de trabajo ordenada para la gestión de un proyecto, que abarca la captura de requisitos de la empresa, diseño, desarrollo e implementación de un software personalizado y adaptado a las exigencias de un cliente determinado.

El propósito del software final apunta a la agilización y automatización de las diferentes tareas a realizar dentro de la organización y la posterior evaluación de satisfacción de los usuarios del sistema.

El informe abarca las etapas de la creación de un software personalizado y el análisis de la misma, desde la captura de requisitos, casos de uso, definición del alcance, diagramas de actividades, prototipos, evaluación de frameworks a utilizar, modelados de bases de datos, diagramas UML, desarrollo, pruebas, instalación y capacitación del personal que utilizara el sistema.

La finalidad de este informe intenta demostrar la importancia de los conocimientos teóricos en programación y gestión de proyectos, llevándolos a la práctica en un caso real de una empresa. Definir un flujo de trabajo es de vital importancia para poder ordenar las diferentes actividades a realizar en un proyecto

2. Objetivos

El objetivo principal es poder cubrir las necesidades informáticas de una empresa que posee un flujo de trabajo administrativo precario. Se procederá a analizar las distintas actividades de la organización con el fin de poder identificar las necesidades de los empleados, y brindar una solución informática que permita la agilización de los procesos, reducción de documentos en papel y digitalización de la información.

Para lograr estos objetivos, se debe tener en cuenta los factores de éxito de un proyecto.

- Planeación y comunicación: son fundamentales para el éxito del proyecto. Evita que ocurran problemas y/o reducen su impacto al mínimo en el logro del objetivo del proyecto.

- Desarrollar un plan bien elaborado: tomarse el tiempo adecuado para la planificación es crucial para el logro exitoso del proyecto.
- Objetivos claros: definido en términos de producto final o entregable, programa y presupuesto, y aceptado por el cliente.
- Satisfacción del cliente: requiere una comunicación continua para mantenerlo informado y determinar si las expectativas han cambiado.
- Control eficaz del proyecto: medir el avance real y compararlo con el avance planeado y aplicar acciones correctivas de ser necesario.
- Mejora continua: evaluar la realización para identificar mejoras, se debe tener retroalimentación del cliente

3. Ciclo de vida del software

El ciclo de vida del software describe los pasos necesarios para la correcta creación y despliegue de una aplicación de software, reduciendo los costos, optimizando recursos y aumentando la eficiencia.

Dicho ciclo está formado por ocho fases:

3.1. Identificación de problemas

Por medio de entrevistas a los stakeholders, se recaba información relacionada a las necesidades y los requerimientos a cubrir en la empresa.

3.2. Planeación

Los responsables del proyecto evalúan la información obtenida con el relevamiento de requerimientos, con el fin de poder elaborar un cronograma con objetivos.

La planificación también abarca un plan de gestión de recursos, costos y alcance del proyecto.

Es necesario definir con claridad el alcance del proyecto, el propósito de la aplicación, las restricciones necesarias para evitar que el proyecto cambie o se salga del alcance establecido.

3.3. Definición de los requerimientos

Consiste en definir y documentar los requerimientos, estos requerimientos deben estar aprobados por parte de los stakeholders. Se define qué es lo que hará la aplicación y las características del mismo. También es necesario identificar los recursos e incorporarlos al proyecto para definir los requisitos.

3.4. Diseño y creación de prototipos

Incluye una fase de diseño para modelar el funcionamiento de la aplicación y los aspectos de diseño, como por ejemplo:

- User interface (UI): cómo interactúan los usuarios con el sistema y como el sistema responde ante determinadas entradas.
- Programación: el lenguaje de programación que se utilizará, así como la forma en que el software solucionara los problemas y realizará las tareas.
- Seguridad: las medidas que se tomarán para asegurar que el sistema esté protegido. Esto incluye cifrado, protección por contraseña y el almacenamiento seguro de los datos.
- Comunicaciones: define cómo se comunicará el sistema con otros activos de la empresa, como por ejemplo un servidor central.
- Arquitectura: plantillas, patrones de diseño y lenguajes de programación específicos.
- Plataformas: describe el sistema operativo que aloja a la aplicación, por ejemplo, Windows, Linux, Apple, etc.;

Luego de definir el diseño, se puede crear prototipos para visualizar una versión simple del software para hacerse una idea básica de los aspectos básicos de la aplicación, como responderá y que capacidades tiene. En esta fase los programadores reciben comentarios de parte de los stakeholders para aprobar o rechazar el prototipo.

3.5. Desarrollo de software

En esta fase se escribe el programa en sí. Durante el proceso de desarrollo de software, se puede anticipar a problemas que podrían retrasar la producción.

3.6. Evaluación

Las aplicaciones se deben probar continuamente para garantizar su funcionamiento en conjunto, ya que el desarrollo de software suele dividirse en proyectos más pequeños realizados por personas o equipos diferentes. Asegurarse de que cada funcionalidad se ejecute como debería y de que cada parte de la aplicación interactúe correctamente con el resto. La evaluación continua reduce la cantidad de errores que los usuarios pueden encontrar y aumenta la tasa de uso y la satisfacción del cliente.

3.7. Despliegue

Una vez concluida la fase de pruebas, la aplicación se pone a disposición de los usuarios. Se procederá a instalar la aplicación en los equipos de computación de la empresa cliente.

3.8. Mantenimiento

Una vez que la aplicación se haya desplegado y esté en uso, en la fase final se detectan aquellos errores que se han pasado por alto en la fase de desarrollo y se procede a resolverlos, lo que puede iniciar un proceso iterativo.

4. Administración de proyecto

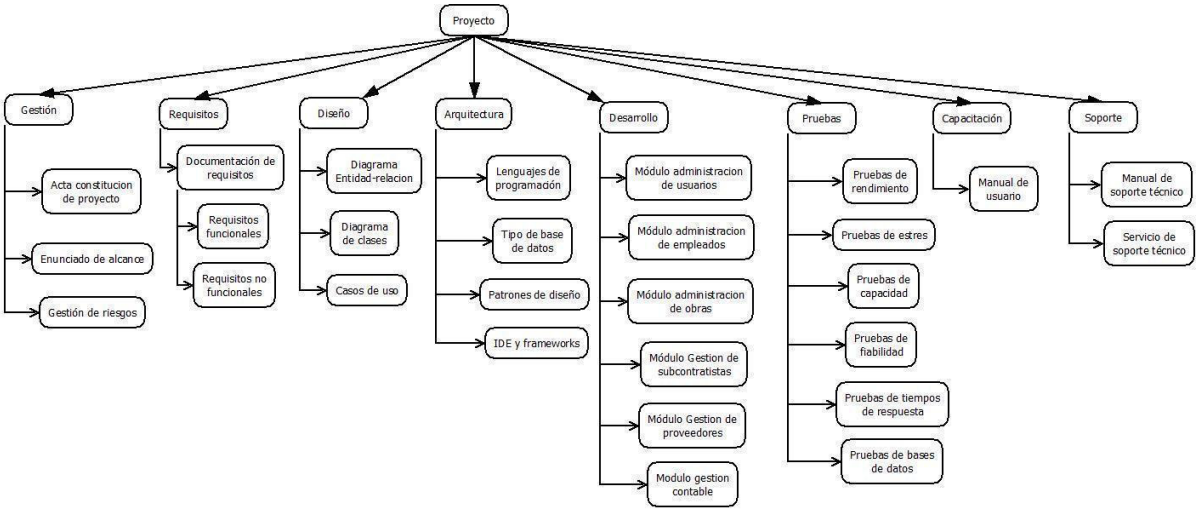
La administración de proyectos es la planeación, organización, coordinación, dirección y control de los recursos para lograr el objetivo del proyecto.

La administración de un proyecto está respaldada por una serie de documentación que reflejan los aspectos importantes, tales como planeación, requerimientos, cronogramas, riesgos, costos, alcance, etc.

En la siguiente figura se puede observar las diferentes fases de la administración de proyectos y sus diferentes documentaciones.

Figura 1

Desglose de la administración de proyecto



A continuación se enumerará las fases del proyecto y un breve enunciado de cada documentación interviniente en dicha fase

.

4.1. Gestión

- Acta de constitución de proyecto: el acta de constitución del proyecto documenta las necesidades del área de negocio que dieron la iniciativa a la solicitud del cliente, las premisas, restricciones, y los requisitos que el proyecto debe proporcionar.
- Enunciado de alcance: la gestión del alcance del proyecto incluye los procesos necesarios para garantizar que el proyecto incluya todo el trabajo requerido y únicamente el trabajo para completar el proyecto con éxito.

  El alcance del proyecto se enfoca primordialmente en definir y controlar que se incluye en el proyecto y que no.
- Gestión de riesgos: el riesgo de un proyecto es un evento o condición incierta que, de producirse, tendrá un efecto positivo o negativo en uno o más de los objetivos del proyecto, tales como el cronograma, el alcance, el costo, etc. La gestión de riesgos del proyecto incluye los procesos para llevar a cabo la planificación de la gestión de los riesgos, así como la identificación, análisis, planificación de respuestas y control de los riesgos del proyecto.

4.2. Requerimientos

- Documentación de requerimientos: documentar las necesidades, deseos y expectativas de los stakeholders. Los requerimientos deben estar redactados de forma sencilla para facilitar la comprensión de todos los sectores interesados del proyecto.

4.3. Diseño

- Diagramas de entidad-relación: es un diagrama de flujo que demuestra cómo las entidades (personas, objetos, conceptos) se relacionan entre sí en un sistema. Los diagramas de entidad-relación se usan para diseñar bases de datos relacionales.
- Diagramas de clases: es un tipo de diagrama de estructura estática que describe los componentes de un sistema, mostrando las clases de un sistema, sus atributos, métodos e interacción con otras clases.
- Diagramas de casos de uso: proporciona uno o más escenarios que indican cómo debe interactuar el sistema con los usuarios u otros sistemas para conseguir un objetivo específico.

4.4. Desarrollo

La fase de desarrollo es en donde se escribe el código fuente del sistema, esta fase está respaldada por toda la documentación enumerada anteriormente, de esta forma se asegura que lo que se está desarrollando sea de forma ordenada, optimizando los recursos y minimizando riesgos.

4.5. Pruebas

- Pruebas de rendimiento: estas pruebas verifican como el sistema responde ante una cantidad de usuarios concurrentes
- Prueba de estrés: verifica la cantidad máxima de usuarios soportados por el sistema antes de fallar.
- Pruebas de capacidad: verifica cual es la cantidad máxima de usuarios que pueden manejar el sistema, manteniendo un rendimiento aceptable.
- Prueba de fiabilidad: verifica si el sistema se mantiene disponible después de largos periodos de tiempo.

- Pruebas de tiempo de respuesta: verifica cuánto tiempo tarda en procesar peticiones específicas.
- Pruebas en bases de datos: verifica los tiempos de ejecución de consultas complejas o masivas.

4.6. Capacitación

- Redacción de un manual de usuario: documentación que especifica las características del sistema, instalación, configuración y utilización.

4.7. Soporte

- Manual de soporte técnico: documentación que detalla las especificaciones, arquitectura del sistema, procesos y procedimientos del sistema. También cuenta con una sección de solución de problemas comunes y diagramas.
- Soporte técnico al cliente: luego del desarrollo e instalación del sistema, el cliente contará con un servicio de soporte técnico por un tiempo determinado, para atender las necesidades y problemas que puedan surgir y que no hayan sido detectadas en la etapa de desarrollo y pruebas.

5. Caso de estudio

El ámbito de esta práctica profesional supervisada se llevará a cabo dentro del área de sistemas de una empresa privada, dedicada a la construcción.

Dentro de esta empresa se llevará a cabo el relevamiento de requisitos y se seguirán las pautas del ciclo de desarrollo de software mencionados anteriormente para el desarrollo de una aplicación que sea funcional al modelo de negocios, teniendo en cuenta la situación actual, infraestructura organizacional, organigrama, futuros usuarios del sistema y equipos de computación con los que cuenta la empresa.

5.1. Infraestructura

La empresa cuenta con el alquiler de una oficina, en la que se realizan tareas administrativas y se guardan documentaciones. Los socios de la organización realizan sus

reuniones y llevan sus computadoras portátiles al lugar. La oficina también cuenta con una computadora de escritorio.

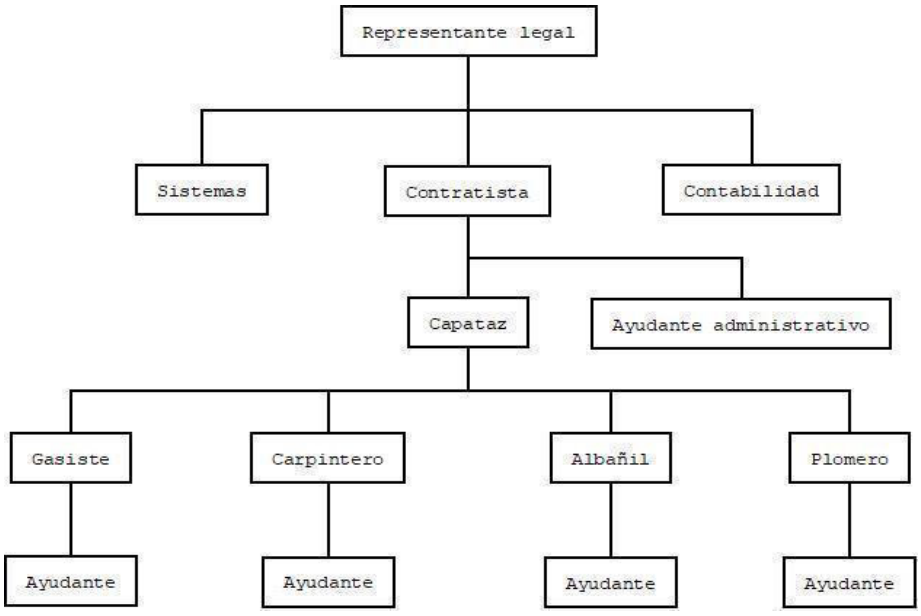
Debido a las dimensiones del lugar, por el momento no es necesario desarrollar un sistema con conectividad a internet.

5.2. Organigrama

Un organigrama es un diagrama que muestra la estructura interna de una organización. Representa las jerarquías y las relaciones entre los distintos departamentos, roles y responsabilidades de los empleados. La siguiente figura muestra un organigrama de la empresa en la que se desarrolla ésta práctica profesional supervisada.

Figura 2

Organigrama de la empresa



- Representante legal: es quien actúa en nombre de la empresa y que es reconocido por la ley. Puede ser uno o varios socios y se encarga de la representación y toma de decisiones de la empresa.

- Sistemas: encargado de asistir a la empresa con la adquisición e instalación de equipos de computación, desarrollo de soluciones informáticas que se ajusten a las necesidades de la empresa, brindar capacitación y soporte técnico a los usuarios
- Contratista: encargado de organizar y dirigir la mano de obra, responsable de proporcionar los materiales y herramientas necesarias para llevar a cabo la construcción del proyecto
- Contabilidad: departamento de la organización que lleva registro de las finanzas de la organización
- Capataz: encargado de dirigir un grupo de trabajadores, organiza las tareas a realizar y los materiales necesarios para llevar a cabo dicha tarea. Además controla las medidas de seguridad e informa al contratista del estado y avances de la construcción.
- Ayudante administrativo: asistente del contratista, encargado de recorrer las obras, tomar asistencias, encargarse de las planillas de entrega de equipos de protección personal, informar al contratista de las inspecciones
- Albañil: encargado de las tareas de excavación, limpieza de terreno, preparación de materiales para la construcción, levantar paredes, colocación de aberturas, cerámicos y accesorios
- Carpintero: encargado del armado de encofrados para posteriormente ser llenados de hormigón por los albañiles
- Plomero/Gasista: encargado de la instalación de tuberías para proporcionar a la obra de los servicios de agua y/o gas.

5.3. Situación actual de la empresa

Actualmente la empresa no cuenta con una gran infraestructura informática, los empleados que llevan a cabo tareas administrativas cuentan con computadoras portátiles personales, utilizan editores de texto y hojas de cálculo de programas de ofimática. La gran mayoría de la información relevante de la empresa se encuentra en papel y tanto estos como los documentos digitales no se encuentran centralizados, por lo tanto, no todos los empleados de la empresa tienen acceso directo e inmediato a estos.

La información no cuenta con ningún sistema almacenamiento de copia de resguardo y la seguridad de estos datos es mínima, únicamente están protegidos por el usuario y

contraseña personal de cada empleado y no todos poseen un antivirus con protección de ataques de red.

6. Relevamiento de requisitos

Teniendo en cuenta la situación actual de la empresa, se procede al relevamiento de requisitos. En esta etapa se definirá las necesidades de la empresa y los futuros usuarios del sistema, dichos requerimientos han sido relevados mediante una serie de entrevistas con el cliente, analizando el modelo de negocio, el flujo de datos y la protección de los datos potencialmente sensibles para la organización.

El proyecto consistirá en el desarrollo de un software que permita el acceso mediante un login de usuario y contraseña, a cada usuario se le asignará un rol específico y tendrá acceso a una parte de la información dependiendo de su rol.

El sistema proporcionará, en términos generales, las siguientes funcionalidades:

- Registro y administración de datos de usuarios
- Registro y administración de datos de empleados
- Registro y administración de datos de sociedades
- Registro y administración de aseguradoras de riesgo de trabajo (ART)
- Liquidación de sueldos de empleados
- Registro y administración de datos de subcontratistas
- Registro y administración de datos de proveedores
- Registro de gastos de obra

6.1. Usuarios del sistema

Según los relevamientos de requisitos, se detectó cuatro roles de usuarios:

- Administrador de sistemas: con acceso total al sistema, encargado de administrar los permisos de los demás usuarios
- Administrativo/contratista: con acceso a gran parte del sistema, encargado de administrar la información referente a las sociedades, ART, obras, empleados, subcontratistas y proveedores.

- Ayudante administrativo: con acceso restringido al sistema, encargado de la carga de asistencias de los empleados, inspecciones de cada obra y entrega de equipos de protección personal.
- Contabilidad: con acceso restringido al sistema, encargado de administrar las liquidaciones de sueldos y gastos de obras

En cuanto a las características de los futuros usuarios, cuentan con conocimientos de software de ofimática, aun así, requerirán de una breve capacitación en la utilización de un software específico como el que se busca desarrollar en este proyecto.

6.2. Restricciones del sistema

Debido a la pequeña escala de la empresa y la cantidad de usuarios finales, el sistema se ejecutará en una red privada en una oficina, y podrá ser usada en cualquier tipo de sistema operativo.

La seguridad del sistema contará con un acceso mediante usuario y contraseña, las restricciones de la información estarán determinados por los roles de cada usuario.

6.3. Requerimientos del sistema

A continuación se presentarán los requerimientos del sistema que fueron identificados en las entrevistas con el cliente. El cumplimiento de dichos requerimientos determinará la satisfacción del cliente respecto al software.

Estos requerimientos se pueden clasificar en requerimientos funcionales (que es lo que el sistema hará) y requerimientos no funcionales (características de seguridad, rendimiento, disponibilidad, etc.)

6.4. Requerimientos funcionales

Esta sub sección específica los requerimientos funcionales, estos definen la expectativa del cliente ante el sistema. Los mismos han sido descritos a un nivel esencial sin detalles de los mecanismos informáticos requeridos para resolverlos.

6.4.1. Acceso al sistema

Tabla 1

Ingreso al sistema

Código	Req-1
Descripción	El sistema deberá permitir el acceso a usuarios registrados. El ingreso al sistema por parte de un administrador o empleado se habilita mediante un usuario y contraseña.
Prioridad	Alta

Tabla 2

Ingreso al sistema por roles de usuario

Código	Req-2
Descripción	El sistema permitirá al usuario visualizar parte de las funciones e información según el rol de usuario asignado
Prioridad	Alta

6.4.2. Módulo sociedades

Tabla 3

Registro de nuevas sociedades

Código	Req-3
Descripción	El sistema permitirá guardar datos de las sociedades (S.R.L, S.A, autónomo, etc.)
Prioridad	Media

Tabla 4

Actualización de datos de sociedades

Código	Req-4
Descripción	El sistema permitirá modificar parcial o totalmente los datos de las sociedades
Prioridad	Media

Tabla 5

Eliminación de registros de sociedades

Código	Req-5
Descripción	El sistema permitirá eliminar parcial o totalmente los datos de las sociedades
Prioridad	Media

Tabla 6

Visualización de datos de sociedades

Código	Req-6
Descripción	El sistema permitirá visualizar los datos de las sociedades
Prioridad	Media

Tabla 7

Contratación de pólizas de seguros

Código	Req-7
Descripción	El sistema permitirá guardar los datos de la ART de la sociedad
Prioridad	Alta

Tabla 8

Notificaciones de vencimientos de póliza

Código	Req-8
Descripción	El sistema permitirá visualizar un recordatorio de vencimiento próximo de la ART con una anticipación de 60 días.
Prioridad	Alta

Tabla 9

Actualización de datos de pólizas

Código	Req-9
Descripción	El sistema permitirá modificar parcial o totalmente los datos de la ART de la sociedad También deberá registrarse el nuevo contrato de ART (por año)
Prioridad	Alta

Tabla 10

Visualización de datos de pólizas

Código	Req-10
Descripción	El sistema permitirá visualizar los datos de la ART de la sociedad
Prioridad	Alta

6.4.3. Módulo empleados

Tabla 11

Registro de nuevos empleados

Código	Req-11
Descripción	El sistema permitirá guardar datos de los empleados ya existentes y nuevos empleados contratados
Prioridad	Alta

Tabla 12

Actualización de datos de empleados

Código	Req-12
Descripción	El sistema permitirá modificar parcial o totalmente los datos de cada empleado
Prioridad	Alta

Tabla 13

Visualización de datos de Empleados

Código	Req-13
Descripción	El sistema permitirá visualizar los datos personales de cada empleado
Prioridad	Alta

Tabla 14

Eliminación de registros de empleados

Código	Req-14

Descripción	El sistema permitirá eliminar parcial o totalmente los datos de cada empleado
Prioridad	Alta

Tabla 15

Despidos y renuncias de empleados

Código	Req-15
Descripción	El sistema permitirá registrar el despido y/o renuncia de obreros
Prioridad	Alta

Tabla 16

Creación de cargos jerárquicos

Código	Req-16
Descripción	El sistema permitirá la creación de un cargo jerárquico que se le asignará a cada empleado
Prioridad	Media

Tabla 17

Actualización de datos de cargos jerárquicos

Código	Req-17
Descripción	El sistema permitirá modificar parcial o totalmente los datos de los cargos jerárquicos.
Prioridad	Media

Tabla 18

Eliminación de cargos jerárquicos

Código	Req-18
Descripción	El sistema permitirá eliminar los datos de un cargo jerárquico
Prioridad	Media

Tabla 19

Creación de grupos de trabajo

Código	Req-19
Descripción	El sistema permitirá la creación de Grupos de empleados, compuestos por un capataz y sus empleados
Prioridad	Alta

Tabla 20

Actualización de datos de grupos de trabajo

Código	Req-20
Descripción	El sistema permitirá editar parcial o totalmente los datos de los grupos de trabajo
Prioridad	Alta

Tabla 21

Disolución de grupos de trabajo

Código	Req-21
Descripción	El sistema permitirá eliminar parcial o totalmente los datos de los grupos de trabajo.
Prioridad	Alta

Tabla 22

Visualización de datos de grupos de trabajo

Código	Req-22
Descripción	El sistema permitirá visualizar los grupos de trabajo existentes.
Prioridad	Alta

Tabla 23

Reportes de empleados

Código	Req-23
Descripción	El sistema permitirá generar reportes de empleados según filtros aplicados (datos personales, contratado, despedido, etc.)
Prioridad	Alta

Tabla 24

Registro de asistencias de empleados

Código	Req-24
Descripción	El sistema permitirá cargar la asistencia de cada empleado
Prioridad	Alta

Tabla 25

Visualización de Asistencias de empleados

Código	Req-25
Descripción	El sistema generará reportes que permita ver las asistencias de los empleados
Prioridad	Alta

Tabla 26

Registro de accidentes de trabajo

Código	Req-26
Descripción	El sistema deberá registrar los accidentes que involucren a los empleados, guardando datos del empleado, fecha, número de siniestro, detalles y estado actual del empleado
Prioridad	Alta

Tabla 27

Visualización de empleados disponibles

Código	Req-27
Descripción	El sistema deberá visualizar el estado de actividad de cada empleado, si se encuentra trabajando, de vacaciones, o inactivo por accidente.
Prioridad	Alta

6.4.4. Módulo subcontratistas

Tabla 28

Registro de nuevos subcontratistas

Código	Req-28
Descripción	El sistema permitirá guardar datos de subcontratistas, tipo de actividad, empleados del subcontratista, salario, etc.
Prioridad	Alta

Tabla 29

Actualización de datos de subcontratistas

Código	Req-29
Descripción	El sistema permitirá modificar parcial o totalmente los datos de subcontratistas.
Prioridad	Alta

Tabla 30

Visualización de datos de subcontratistas

Código	Req-30
Descripción	El sistema permitirá visualizar los datos de subcontratistas.
Prioridad	Alta

Tabla 31

Eliminación de registros de subcontratistas

Código	Req-31

Descripción	El sistema permitirá eliminar parcial o totalmente los datos de subcontratistas.
Prioridad	Alta

Tabla 32

Visualización de datos de subcontratistas

Código	Req-32
Descripción	El sistema permitirá ver la lista de facturas asociadas a un subcontratista
Prioridad	Alta

6.4.5. Módulo obras

Tabla 33

Registro de Nuevas obras

Código	Req-33
Descripción	El sistema permitirá guardar datos de nuevas obras
Prioridad	Alta

Tabla 34

Actualización de datos de obras

Código	Req-34
Descripción	El sistema permitirá modificar parcial o totalmente los datos referentes a cada obra

Prioridad	Alta

Tabla 35

Visualización de datos de obras

Código	Req-35
Descripción	El sistema permitirá visualizar los datos de cada obra
Prioridad	Alta

Tabla 36

Actualización del estado de las obras

Código	Req-36
Descripción	El sistema permitirá registrar el estado actual de cada obra (Iniciado, pausado, cancelado, finalizado, clausurado)
Prioridad	Media

Tabla 37

Reportes de obras

Código	Req-37
Descripción	El sistema permitirá generar reportes de obras según filtros aplicados (provincia. Localidad, estado de obra)
Prioridad	Alta

Tabla 38

Reportes de empleados en obras

Código	Req-38
Descripción	El sistema permitirá generar reportes de los empleados que actualmente están trabajando en cada obra según filtros (empleado activo, despedido, etc.)
Prioridad	Alta

Tabla 39

Asignación de empleados a obras

Código	Req-39
Descripción	El sistema permitirá asignar un capataz y/o subcontratista y sus empleados a la obra, si algún empleado ya se encuentra trabajando en otra obra, el sistema deberá realizar el traspaso de obras, quitando al empleado de la obra actual y asignándole a la nueva obra
Prioridad	Alta

Tabla 40

Registro de entregas de equipos de protección personal (EPP)

Código	Req-40
Descripción	El sistema registrará la entrega de EPP a cada empleado, permitiendo visualizar datos personales del empleado, fecha y tipo de EPP (ropa, botines, etc.).
Prioridad	Alta

Tabla 41

Visualización de entregas de equipos de protección personal (EPP)

Código	Req-41
Descripción	El sistema permitirá visualizar los registros de entregas de EPP
Prioridad	Alta

Tabla 42

Registros de gastos por obra

Código	Req-42
Descripción	El sistema permitirá registrar los gastos de cada obra, en base a materiales y herramientas de proveedores, sueldos de empleados, gastos por daños, reparaciones, etc.
Prioridad	Alta

Tabla 43

Cotización de obras

Código	Req-43
Descripción	El sistema permitirá realizar una cotización estimada de obra, en base a los datos de obras anteriores, metros cuadrados, cantidad necesaria de hormigón, etc.
Prioridad	Media

6.4.6. Módulo proveedores

Tabla 44

Registro de nuevos proveedores

Código	Req-44
Descripción	El sistema permitirá guardar datos de nuevos proveedores
Prioridad	Media

Tabla 45

Actualización de datos de proveedores

Código	Req-45
Descripción	El sistema permitirá modificar parcial o totalmente los datos de los proveedores
Prioridad	Media

Tabla 46

Visualización de datos de proveedores

Código	Req-46
Descripción	El sistema permitirá visualizar los datos de los proveedores
Prioridad	Media

Tabla 47

Eliminación de registros de proveedores

Código	Req-47
Descripción	El sistema permitirá borrar parcial o totalmente los datos de los proveedores
Prioridad	Media

Tabla 48

Registro de facturas de proveedores

Código	Req-48
Descripción	El sistema permitirá Visualizar la lista de facturas asociadas a un proveedor
Prioridad	Alta

6.4.7. Módulo contabilidad

Tabla 49

Liquidación de sueldos

Código	Req-49
Descripción	El sistema permitirá calcular el sueldo en base al monto asignado, asistencias, antigüedad, cargas sociales y beneficios
Prioridad	Alta

Tabla 50

Reportes de sueldos

Código	Req-50
Descripción	El sistema permitirá generar reportes de sueldos según filtros aplicados (fecha, obra, empleados, etc.)
Prioridad	Alta

Tabla 51

Generación de recibos de sueldo

Código	Req-51
Descripción	El sistema contará con un generador de recibos, dicho documento se generará en formato PDF.
Prioridad	Alta

Tabla 52

Registros en el libro diario

Código	Req-52
Descripción	El sistema contará con una sección de libro diario, que registra los activos y pasivos de la empresa
Prioridad	Medio

Tabla 53

Registros en el libro mayor

Código	Req-53
Descripción	El sistema contará con una sección de libro mayor, que controlará los registros del libro diario
Prioridad	Medio

Tabla 54

Liquidación de sueldos

Código	Req-54

Descripción	El sistema contará con una sección de libro de sumas y saldos, que controlará los registros del libro mayor.
Prioridad	Medio

Tabla 55

Reportes de libros contables

Código	Req-55
Descripción	El sistema permitirá generar reportes de los libros contables según filtros aplicados
Prioridad	Medio

6.4.8. Modulo Administrador

Tabla 56

Alta de perfiles de usuarios

Código	Req-56
Descripción	El sistema permitirá crear un perfil de usuario, dependiendo de su rol, tendrá predefinido un set básico de acceso y funcionalidades del sistema. Por ejemplo, rol contabilidad, tendrá acceso al módulo contable
Prioridad	Alto

Tabla 57

Registro de nuevos usuarios

Código	Req-57

Descripción	El sistema permitirá crear un usuario, que podrá acceder al sistema mediante usuario y password
Prioridad	Alta

Tabla 58

Gestión de permisos de usuario

Código	Req-58
Descripción	El sistema permitirá asignar un rol a cada usuario
Prioridad	Alta

Tabla 59

Actualización de permisos de usuarios

Código	Req-59
Nombre	Eliminar usuarios
Descripción	El sistema permitirá modificar los permisos de acceso a cada usuario
Prioridad	Alta

Tabla 60

Eliminación de usuarios

Código	Req-60
Descripción	El sistema permitirá eliminar usuarios para restringir el acceso al sistema
Prioridad	Alta

6.5. Requerimientos no funcionales

En esta sub sección se especifican los requerimientos no funcionales, características que pueden usarse para evaluar la operación y rendimiento de un sistema.

6.5.1. Requerimientos de eficiencia

Tabla 60

Tiempos de respuesta del sistema

Código	Req-61
Descripción	Las funcionalidades del sistema deberán responder al usuario en un tiempo menor a 30 segundos
Prioridad	Alta

Tabla 62

Multiplicidad de usuarios conectados al sistema

Código	Req-62
Descripción	El sistema deberá ser capaz de aceptar hasta 5 usuarios simultáneamente
Prioridad	Alta

6.5.2. Requerimientos de seguridad y datos

Tabla 63

Permisos de acceso al sistema

Código	Req-63

Descripción	Los permisos de acceso al sistema podrán ser cambiados únicamente por el administrador de sistemas
Prioridad	Media

Tabla 64

Respaldo del sistema

Código	Req-64
Descripción	El sistema deberá contar con un respaldo de datos que se actualizará cada 24 horas.
Prioridad	

Tabla 65

Mecanismos de seguridad

Código	Req-65
Descripción	El sistema deberá contar con medidas de seguridad que eviten el acceso al sistema (ataque de fuerza bruta, ataque de diccionario, etc.)
Prioridad	Alta

6.5.3. Requerimientos de disponibilidad

Tabla 66

Disponibilidad del sistema

Código	Req-66
Descripción	El sistema deberá tener una disponibilidad de 95% de las veces que el usuario intente acceder.

Prioridad	Media

Tabla 67

Tiempo de acceso al sistema

Código	Req-67
Descripción	El tiempo de acceso al sistema no deberá ser mayor a 30 segundos
Prioridad	Alta

6.5.4. Requerimientos de usabilidad

Tabla 68

Interfaz gráfica del sistema

Código	Req-68
Descripción	El sistema deberá contar con una interfaz de usuario sencilla de usar para el usuario
Prioridad	Alta

Tabla 69

Manual de usuario

Código	Req-69
Descripción	El sistema deberá contar con un manual de usuario
Prioridad	Alta

Tabla 70

Notificaciones del sistema

Código	Req-70
Descripción	El sistema deberá contar con mensajes de error informativos para el usuario (por ejemplo, ingresar texto en campos que solo acepta valores numéricos)
Prioridad	Alta

Tabla 71

Sistema multiplataforma

Código	Req-71
Descripción	El sistema deberá poder ejecutarse en versiones de Windows 8 o posteriores
Prioridad	Alta

7. Enunciado del alcance del proyecto

El enunciado de alcance del proyecto permite establecer los límites del proyecto y definir con precisión los objetivos, plazos y entregables del proyecto que se desean lograr. Al definir claramente el alcance se puede asegurar el logro de las metas y objetivos del proyecto sin sufrir demoras ni sobrecarga de trabajo.

El alcance del proyecto se debe definir en conjunto con los sectores interesados, clientes, inversores, empleados y usuarios, de esta manera quedará asentado los entregables, funciones y límites del producto final.

Mediante una serie de reuniones con el cliente, se estableció el siguiente alcance del proyecto

Tabla 72

Enunciado del alcance del proyecto

Descripción del alcance del proyecto	
Requisitos: condiciones o capacidades que desea poseer o satisfacer el producto para cumplir con contratos, normas, especificaciones u otros documentos formalmente impuestos	Características: propiedades físicas, energéticas, cualitativas, cuantitativas, descriptivas, que son distintivas del producto y/o que describen su singularidad.
1. Acceso al sistema mediante autenticación de usuarios e incorporación de roles	Software que permita el acceso a determinadas funciones del sistema según el tipo de usuario
2. Gestión de los datos personales y asistencias de los empleados	Software CRUD de empleados
3. Gestión de entrega de equipos de protección personal	Software de registro de entrega de EPP
4. Reportes de empleados según filtros	Software de consulta de datos según filtros aplicados
5. Gestión de cargos jerárquicos	Software CRUD de cargos jerárquicos
6. Gestión de los datos relevantes de las obras en construcción	Software CRUD de obras
7. Asignación de empleados a las obras en construcción activas	Software que permita visualizar una obra y empleados disponibles en la misma vista de usuario
8. Reportes de empleados por obra	Software de consulta según filtros aplicados
9. Liquidación de sueldos	Software que calcula el sueldo neto en base al sueldo bruto, beneficios y cargas sociales
10. Generar recibos de sueldo	Software de creación de recibo de sueldo en formato pdf

11. Reportes de sueldos	Software de consulta de sueldos según filtros aplicados

Tabla 73

Criterios de aceptación del proyecto

Criterios de aceptación: especificaciones o requisitos de rendimiento, funcionalidad, etc. Que deben cumplirse antes que se acepte el producto del proyecto	
Concepto	Criterio de aceptación
1. Técnicos	El software debe cumplir con el 60% de los requerimientos funcionales. El software debe cumplir con el n% de los requerimientos no funcionales El software debe poder instalarse en todos los equipos de la empresa
2. De calidad	El sistema deberá cumplir con el 70% de satisfacción del cliente. Además deberá cumplir con los factores y métricas establecidas como requisitos de usabilidad, disponibilidad, seguridad y datos
3. Administrativo	Todos los entregables serán aprobados por el grupo de trabajo y el cliente

Tabla 74

Entregables del proyecto

Entregables del proyecto: documentación, diagramas,, prototipos, productos tangibles o intangibles que se entregará al cliente durante el desarrollo del proyecto

Fase del proyecto	Productos entregables
1. Gestión	Acta de constitución del proyecto, enunciado de alcance, matriz de trazabilidad.
2. Requisitos	Documentación de requisitos
3. Diseño	Diagramas entidad-relación, diagrama de clases, diagramas de casos de uso
4. Desarrollo	Módulo gestión de personal, módulo gestión de obras, módulo gestión contable, módulo gestión de usuarios.
5. Pruebas	
6. Capacitación	Manual de usuario

Tabla 75

Exclusiones del proyecto

Exclusiones del proyecto. Entregables, procesos, áreas, procedimientos características, requisitos, funciones, especialidades, fases, etapas, espacios físicos, virtuales, regiones, etc., que son exclusiones conocidas y no serán abordadas por el proyecto, y que por lo tanto deben estar claramente establecidas para evitar incorrectas interpretaciones entre los stakeholders del proyecto
1. La integración con otros módulos y/o software que podrá tener la empresa
2. El sistema no estará disponible para internet
3. El sistema no generará gráficos
4. El sistema no permitirá personalizar la fuente y tamaño de letra
5. El sistema no contará con una versión para dispositivos móviles

Tabla 76

Restricciones del proyecto

Restricciones del proyecto: factores que limitan el rendimiento del proyecto, el rendimiento de procesos, o las opciones de planificación del proyecto. Pueden aplicar a los objetivos del proyecto o a los recursos empleados en él.	
Factores internos a la organización	Factores externos a la organización
La fase de desarrollo comenzará una vez aprobados los prototipos de interfaz de usuario.	
Una vez iniciado el desarrollo, el equipo y el cliente se reunirán una vez por semana para evaluar los avances del proyecto	
El diseño deberá ser flexible para permitir la incorporación de nuevas funcionalidades en el futuro	

Tabla 77

Supuestos del proyecto

Supuestos del proyecto: factores que, para la planificación del proyecto, se consideran verdaderos, reales o ciertos	
Internos a la organización	Ambientales o externos a la organización
Se cuenta con el personal necesario para el desarrollo del proyecto	
Se cuenta con los equipos de computación necesarios para el desarrollo del proyecto	

8. Gestión de riesgos

Una vez establecido el alcance y los requerimientos del proyecto, se procede a realizar la gestión de riesgos del proyecto, para cada aspecto del proyecto se plantean los casos desfavorables que podrían complicar de alguna manera el normal desarrollo de las actividades.

Cada riesgo identificado es calificado con un índice de probabilidad de ocurrencia y un índice de impacto al proyecto.

Tabla 78

Porcentajes de probabilidades e impactos

Probabilidad	Valor numérico	impacto	Valor numérico
Muy improbable	0.1	Muy bajo	0,05
Relativamente probable	0.3	Bajo	0,1
Probable	0.5	Moderado	0,2
Muy probable	0.7	Alto	0,4
Casi certeza	0.9	Muy alto	0,8

Luego de esta calificación, se procedió a realizar el cálculo de probabilidad*impacto y evaluar el tipo de riesgo en función al resultado

Tabla 79

Clasificación de riesgos según su probabilidad e impacto

Tipo de riesgo	Probabilidad x impacto
Muy alto	Mayor a 0,5
Alto	0,5 a 0,3
Medio	0,29 a 0,10
Bajo	0,09 a 0,05
Muy bajo	menor a 0,05

Se determinó que un resultado de probabilidad * impacto mayor a 0,5 se considera un riesgo muy alto.

Entre 0,3 y 0,5 se considera un riesgo alto.

Entre 0,10 y 0,29 se considera un riesgo medio

Entre un 0,05 y un 0,09 se considera un riesgo bajo

Menor a 0,05 se considera un riesgo muy bajo

Estableciendo estos valores previos, se procedió a realizar la intersección de probabilidades e impactos

Tabla 80

Matriz de probabilidad x impacto

Probabilidad * impacto		Muy bajo	Bajo	Moderado	Alto	Muy alto
		0,05	0,1	0,2	0,4	0,8
Muy bajo	0,1	0,005	0,01	0,02	0,04	0,08
Bajo	0,3	0,015	0,03	0,06	0,12	0,24
Moderado	0,5	0,025	0,05	0,1	0,2	0,4
Alto	0,7	0,035	0,07	0,14	0,28	0,56
Muy alto	0,9	0,045	0,09	0,18	0,36	0,72

Cualquier resultado resaltado en verde, es un riesgo aceptable.

Cualquier resultado resaltado en amarillo es un riesgo a mitigar.

Cualquier resultado resaltado en rojo es un riesgo a evitar.

Mediante estos valores establecidos previamente, se procedió a evaluar los distintos riesgos que se podrían presentar a lo largo del desarrollo del proyecto, calificando la probabilidad de ocurrencia y el impacto en los diferentes objetivos del proyecto.

Tabla 81

Análisis cualitativo y plan de riesgo, primera parte.

Código de referencia	Descripción	Causa raíz	Trigger	Entregables afectados
R-001	Falta de hardware necesario	Falta de comunicación con el	Falta de hardware necesario para realizar	Todo el proyecto

		cliente	el proyecto	
R-002	Incumplimiento del cronograma	Mala estimación de los plazos a cumplir de cada parte del proyecto	Incumplimiento de los hitos a lo largo del proyecto	Todo el proyecto
R-003	Errores inesperados en los módulos	Falta de pruebas en cada modulo	Detección de errores en los módulos	Módulos específicos del proyecto
R-004	Incumplimiento de las expectativas del proyecto	Incumplimiento de los objetivos	Resultados de encuestas de evaluación del cliente	Módulos específicos del proyecto
R-005	Falta de comunicación con los stakeholders	Falta de evaluación de características del proyecto	Falla de comunicación/pérdida de tiempo en acuerdos y negociaciones de requerimientos	Documentación del proyecto-módulos específicos
R-006	Cambios de requerimientos por parte del cliente	Agregar nuevas funcionalidades al proyecto	Cambios de los requerimientos iníciales	Documentación del proyecto-módulo específico del proyecto
R-007	Documentación a entregar poco clara	Mala estimación inicial, falta de control y coordinación	Captura de requerimientos poco clara y/o falta de detalle	Documentación del proyecto
R-008	Desaprobación de los informes periódicos	Mala estimación inicial, falta de control y coordinación	Captura de requerimientos poco clara y/o falta de detalle	Informes periódicos del proyecto
R-009	Enfermedad de alguno de los miembros del equipo de desarrollo	Riesgo inesperado	Disminución de objetivos cumplidos-atrasos en el cronograma	Módulos y documentación a desarrollar en ese momento

R-010	Falta de capacitación de los miembros del equipo de desarrollo	Falta de recursos destinados para la capacitación	Detección de incumplimiento y/o avances del proyecto	Proyecto completo
R-011	Fechas de entrega breves	Mala estimación de los plazos a cumplir de cada parte del proyecto	Incumplimiento de las fechas del cronograma	Documentación y/o módulos a desarrollar en ese momento
R-012	Falta de especificaciones	Mal relevamiento de requerimientos	Tareas a realizar poco claras	Documentación del proyecto-módulos del proyecto
R-013	Pérdida parcial/total del hardware	Causas naturales-eventos inesperados	Incendios-tormentas-cortes de luz	Proyecto completo

Tabla 82

Análisis cualitativo y plan de riesgo, segunda parte.

Código de referencia	Estimación de probabilidad	Objetivos afectados	Estimación de impacto	probabilidad x impacto	Tipo de riesgo
R-001	0,1	Alcance	0,05	0,005	Bajo
		Tiempo	0,4	0,04	
		Costo	0,2	0,02	
		Calidad			
		Total prob x impacto		0,065	
R-002	0,5	Alcance	0,2	0,1	Alto
		Tiempo	0,4	0,2	
		Costo	0,1	0,05	
		Calidad	0,2	0,1	
		Total prob x impacto		0,45	
R-003	0,3	Alcance	0,1	0,03	Medio
		Tiempo	0,1	0,03	
		Costo			
		Calidad	0,4	0,12	
		Total prob x impacto		0,18	

R-004	0,3	Alcance	0,4	0,12	Medio
		Tiempo			
		Costo			
		Calidad	0,4	0,12	
		Total prob x impacto		0,24	
R-005	0,1	Alcance	0,05	0,005	Muy bajo
		Tiempo	0,1	0,01	
		Costo			
		Calidad	0,1	0,01	
		Total prob x impacto		0,025	
R-006	0,1	Alcance	0,2	0,02	Bajo
		Tiempo	0,4	0,04	
		Costo			
		Calidad	0,2	0,02	
		Total prob x impacto		0,08	
R-007	0,1	Alcance			Muy bajo
		Tiempo	0,05	0,005	
		Costo	0,05	0,005	
		Calidad	0,05	0,005	
		Total prob x impacto		0,015	
R-008	0,1	Alcance			Medio
		Tiempo	0,4	0,04	
		Costo	0,2	0,02	
		Calidad	0,4	0,04	
		Total prob x impacto		0,1	
R-009	0,3	Alcance			Medio
		Tiempo	0,2	0,06	
		Costo	0,1	0,03	
		Calidad	0,2	0,06	
		Total prob x impacto		0,15	
R-010	0,1	Alcance	0,4	0,04	Medio
		Tiempo	0,8	0,08	
		Costo	0,4	0,04	
		Calidad	0,8	0,08	
		Total prob x impacto		0,24	
R-011	0,3	Alcance	0,4	0,12	Alto
		Tiempo	0,4	0,12	
		Costo	0,2	0,06	
		Calidad	0,4	0,12	
		Total prob x impacto		0,42	
R-012	0,1	Alcance	0,4	0,04	Medio
		Tiempo	0,8	0,08	
		Costo	0,2	0,02	
		Calidad	0,4	0,04	

		Total prob x impacto		0,18	
R-013	0,5	Alcance	0,2	0,1	Muy alto
		Tiempo	0,8	0,4	
		Costo	0,2	0,1	
		Calidad	0,8	0,4	
		Total prob x impacto		1	

Tabla 83

Análisis cualitativo y plan de riesgo de riesgos, tercera parte.

Código de referencia	Tipo de respuesta	Plan de respuesta	Plan de contingencia
R-001	Aceptar	Evaluación de arquitectura y lenguajes de programación a utilizar en el proyecto para determinar el hardware necesario	Adquirir hardware necesario antes de la etapa de desarrollo
R-002	Evitar	Control periódico de etapas del proyecto, verificación del cumplimiento de objetivos	Dedicar horas extra
R-003	Mitigar	Realizar pruebas al finalizar cada modulo	Control continuo y pruebas a cada funcionalidad integrada al proyecto
R-004	Mitigar	Comunicación y control del cliente, evaluaciones periódicas y al finalizar cada modulo	Analizar causas de insatisfacción, realizar medidas correctivas, control de resultados
R-005	Aceptar	Mantener comunicación constante con los stakeholders	Tomar acciones correctivas en el relevamiento de requerimientos y alcance del proyecto
R-006	Aceptar	Comunicación y control del cliente	Implementar solicitudes de cambios,

			evaluar la importancia de los cambios solicitados y el impacto en el proyecto
R-007	Aceptar	Evaluación y validación continua de cada documentación junto con el cliente	Tomar acciones correctivas
R-008	Mitigar	Programar con tiempo la elaboración de los informes, presentar borradores antes de presentar versión final	Analizar causas de insatisfacción, realizar medidas correctivas, control de resultados
R-009	Mitigar		Dedicar horas extra
R-010	Mitigar	Elaborar informe de las aéreas y/o herramientas en las que se requiere capacitación	Cursos de capacitación
R-011	Evitar	Revisar cronograma	Reasignar fechas de entrega, dedicar horas extra
R-012		Comunicación y control del cliente	Modificar la captura de requerimientos
R-013	Evitar	Verificar si se cuenta con repuestos de hardware	Contar con un stock de repuestos de hardware y/o equipos de computación extra. Backup de los avances del proyecto

9. Diagramas de casos de uso

El diagrama de caso de uso es un tipo de diagrama de comportamiento y se usa frecuentemente para analizar varios sistemas. Permite visualizar los distintos tipos de roles en un sistema y cómo estos interactúan con el sistema.

Los diagramas son de gran utilidad cuando se le presentan a los sectores interesados, se pueden destacar los papeles que interactúan con el sistema y la funcionalidad proporcionada por el sistema sin tener que profundizar en el funcionamiento interno.

Los siguientes diagramas muestran los distintos roles del sistema y el acceso a las funciones del sistema de cada uno. Dichos diagramas fueron evaluados y validados por el cliente.

Figura 3

Casos de uso de ingreso al sistema y gestión de usuarios

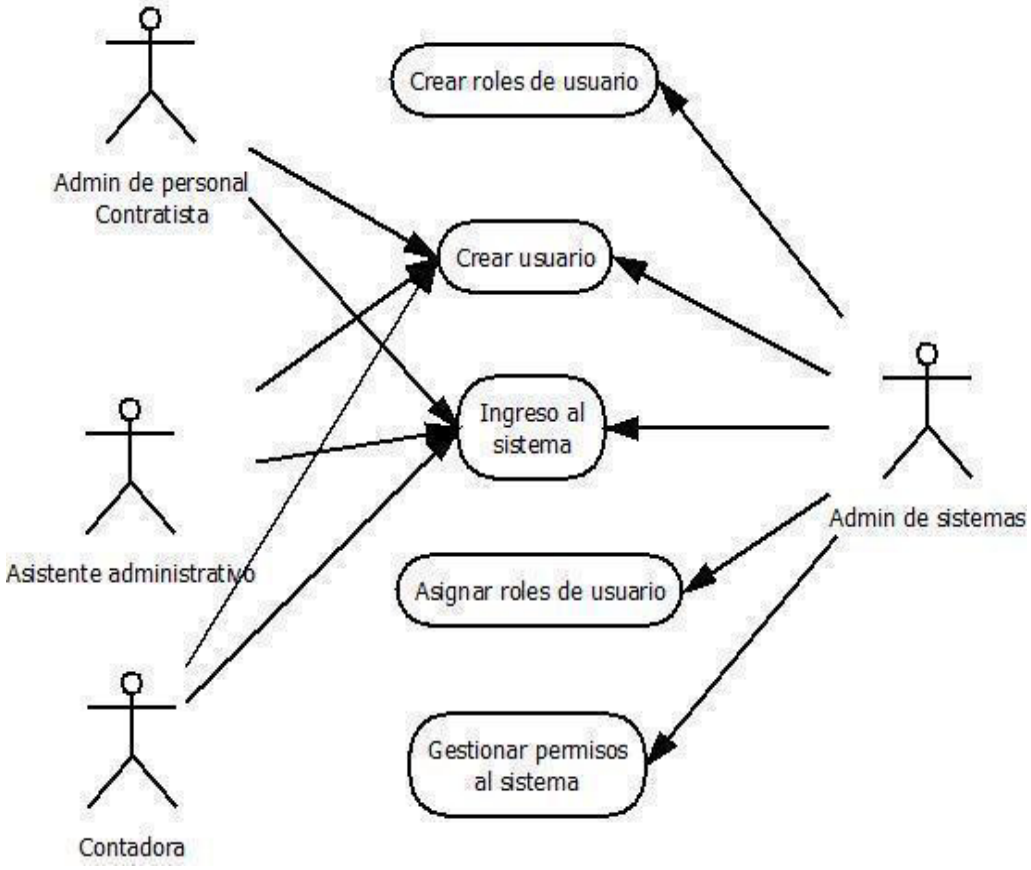


Figura 4

Acceso al módulo de sociedades

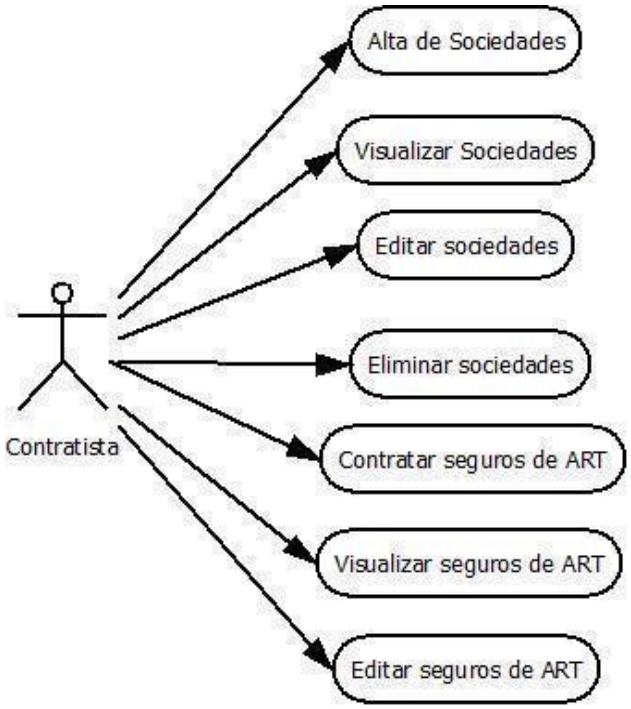


Figura 5

Acceso al módulo obras, primera parte

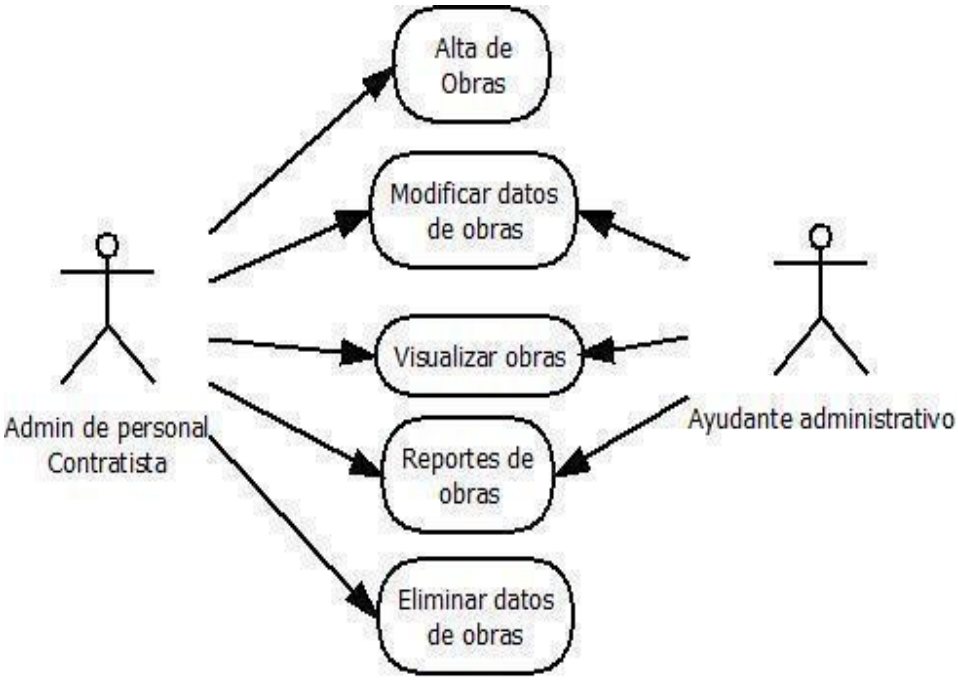


Figura 6

Acceso al módulo obras, segunda parte

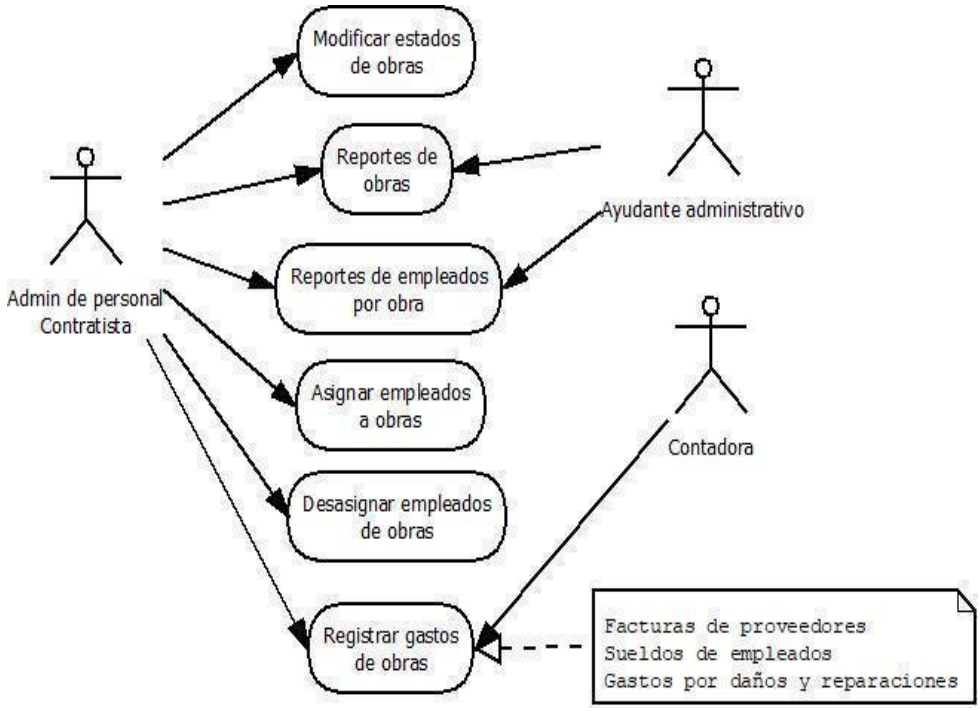


Figura 7

Acceso al módulo empleados, primera parte

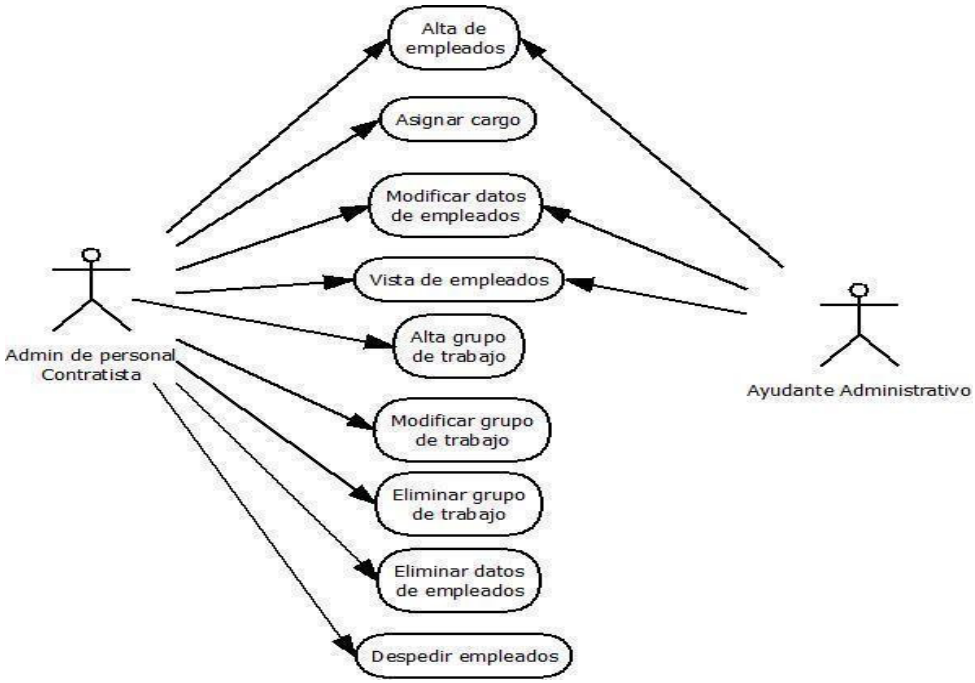


Figura 8

Acceso al módulo empleado, segunda parte

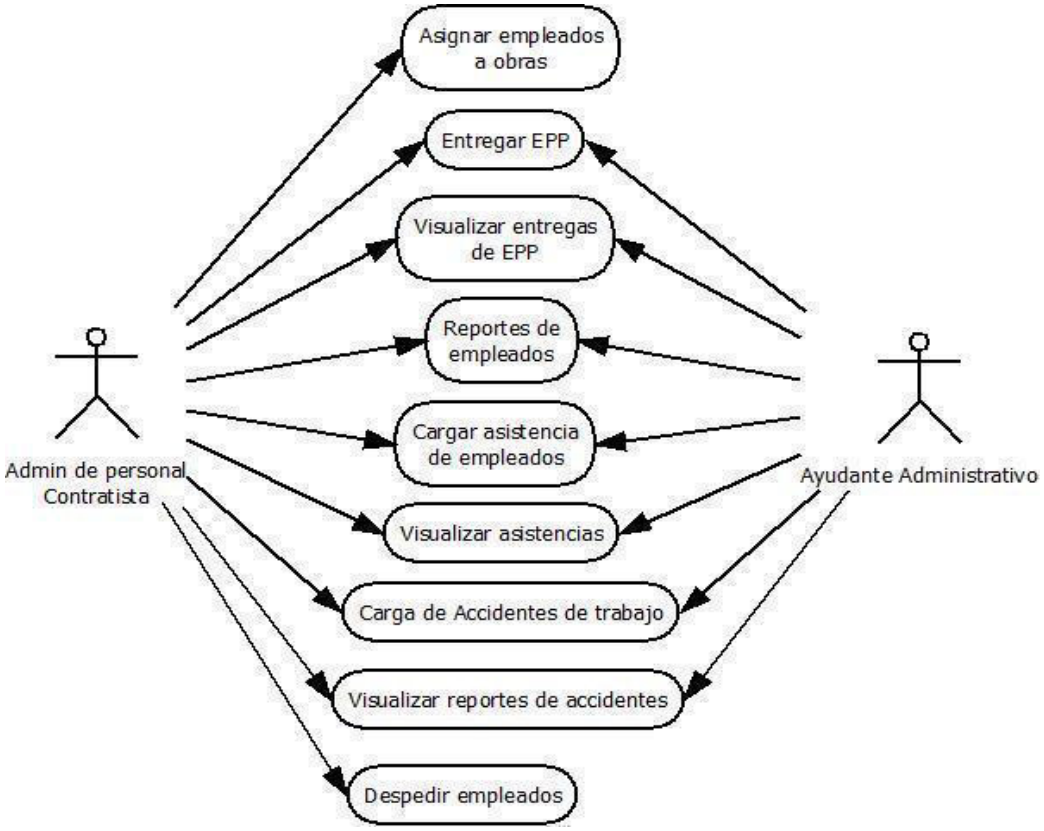


Figura 9

Acceso al módulo empleado, tercera parte.

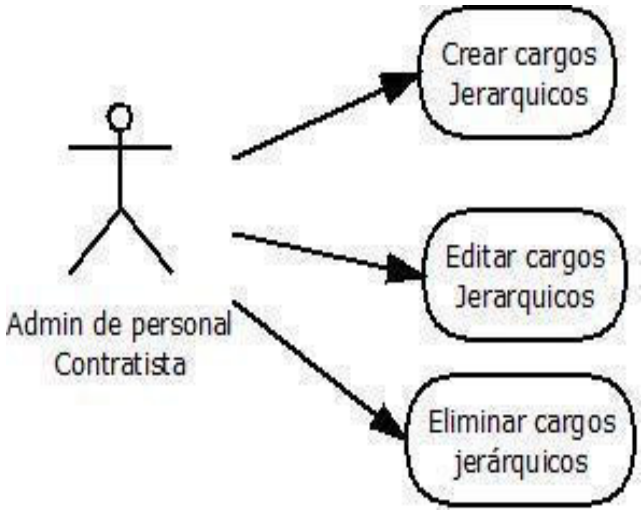


Figura 10

Acceso al módulo subcontratista.

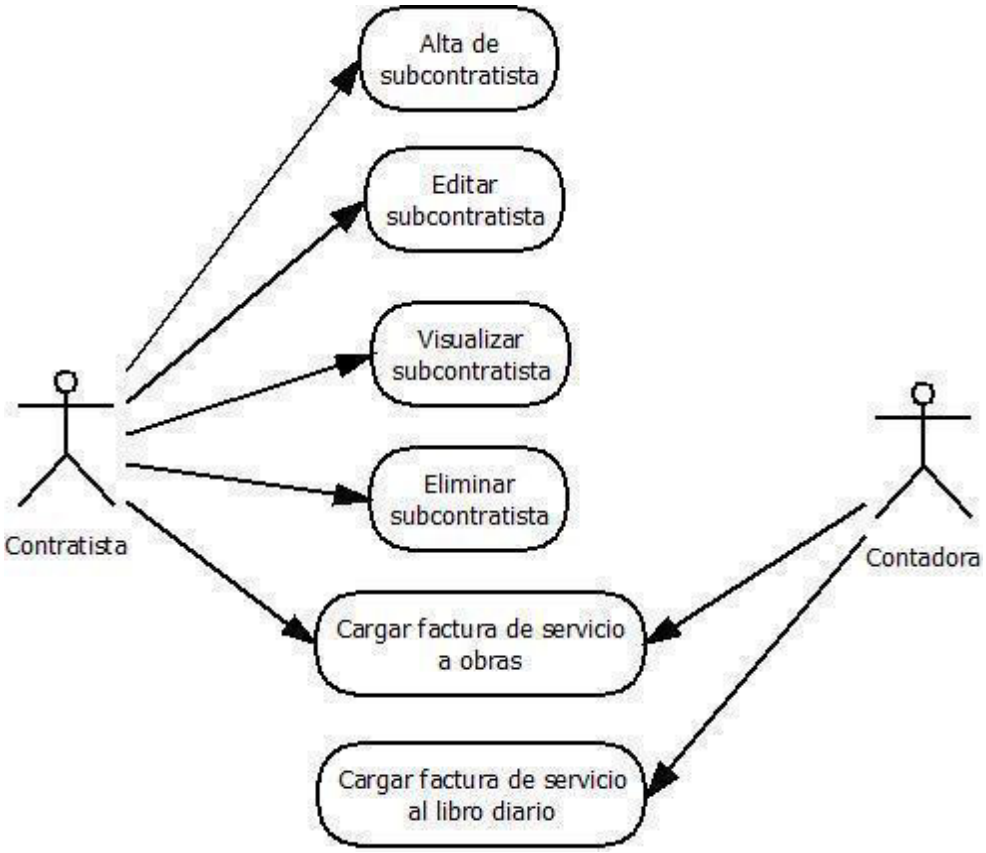


Figura 11

Acceso al módulo proveedores.

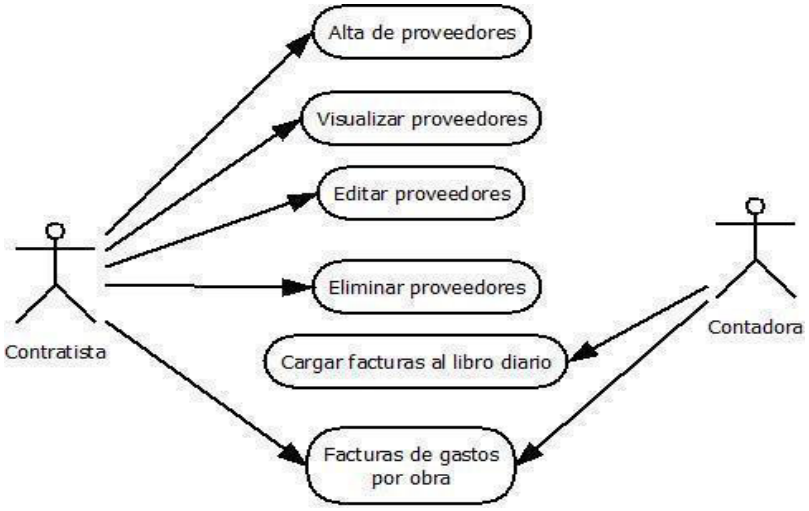
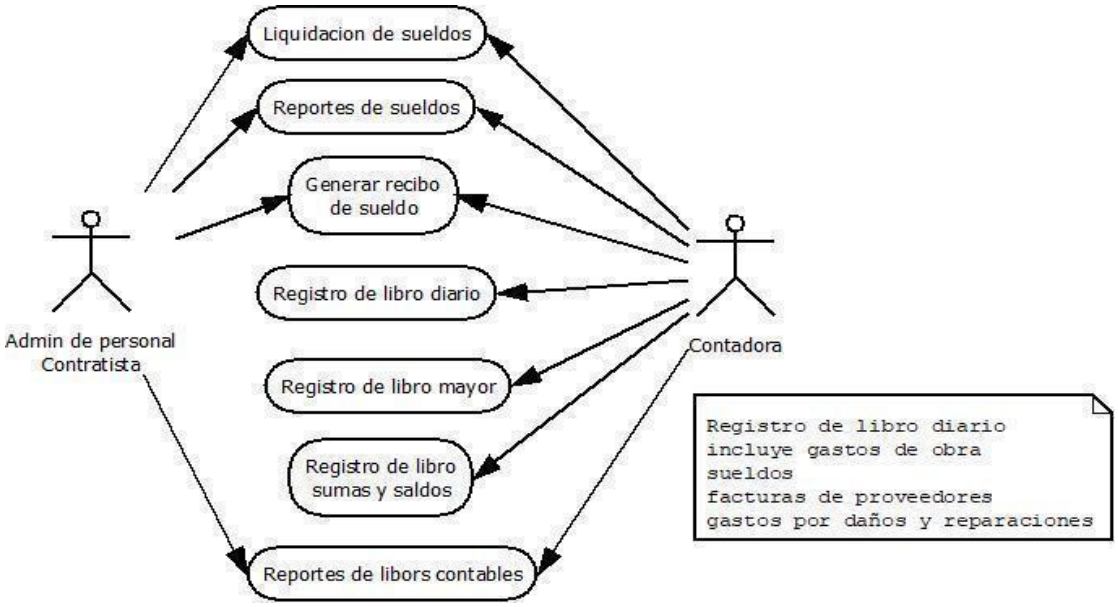


Figura 12

Acceso al módulo contabilidad



10. Diagrama entidad-relación

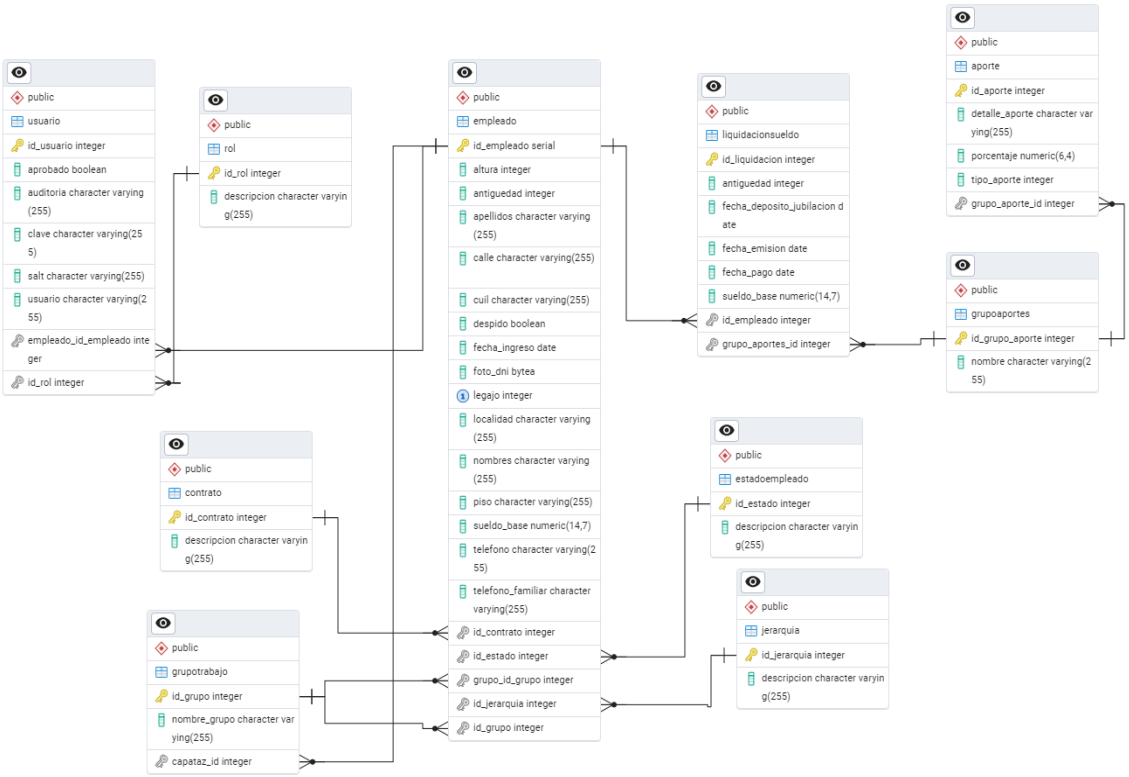
El diagrama entidad-relación refleja como las entidades (personas, objetos, conceptos) se relacionan entre sí en un sistema. Estos diagramas se usan para diseñar bases de datos relacionales y ayudan a los desarrolladores de software a visualizar las relaciones entre elementos claves del sistema.

Es un modelo lógico que muestra cómo fluye la información de una entidad a otra.

Las siguientes figuras muestran la representación del modelo de negocio plasmado en un diagrama entidad-relación.

Figura 13

Diagrama entidad-relación. E*mpleados*



Nota: la llave amarilla indica que el atributo es una Primary Key (clave primaria), este atributo es único e irrepetible, usado para diferenciar un registro de otro. Las llaves grises indican que el atributo es una Foreign Key (clave foránea) es una referencia a una Primary Key de otra tabla.

Figura 14

Diagrama entidad-relación. Obras

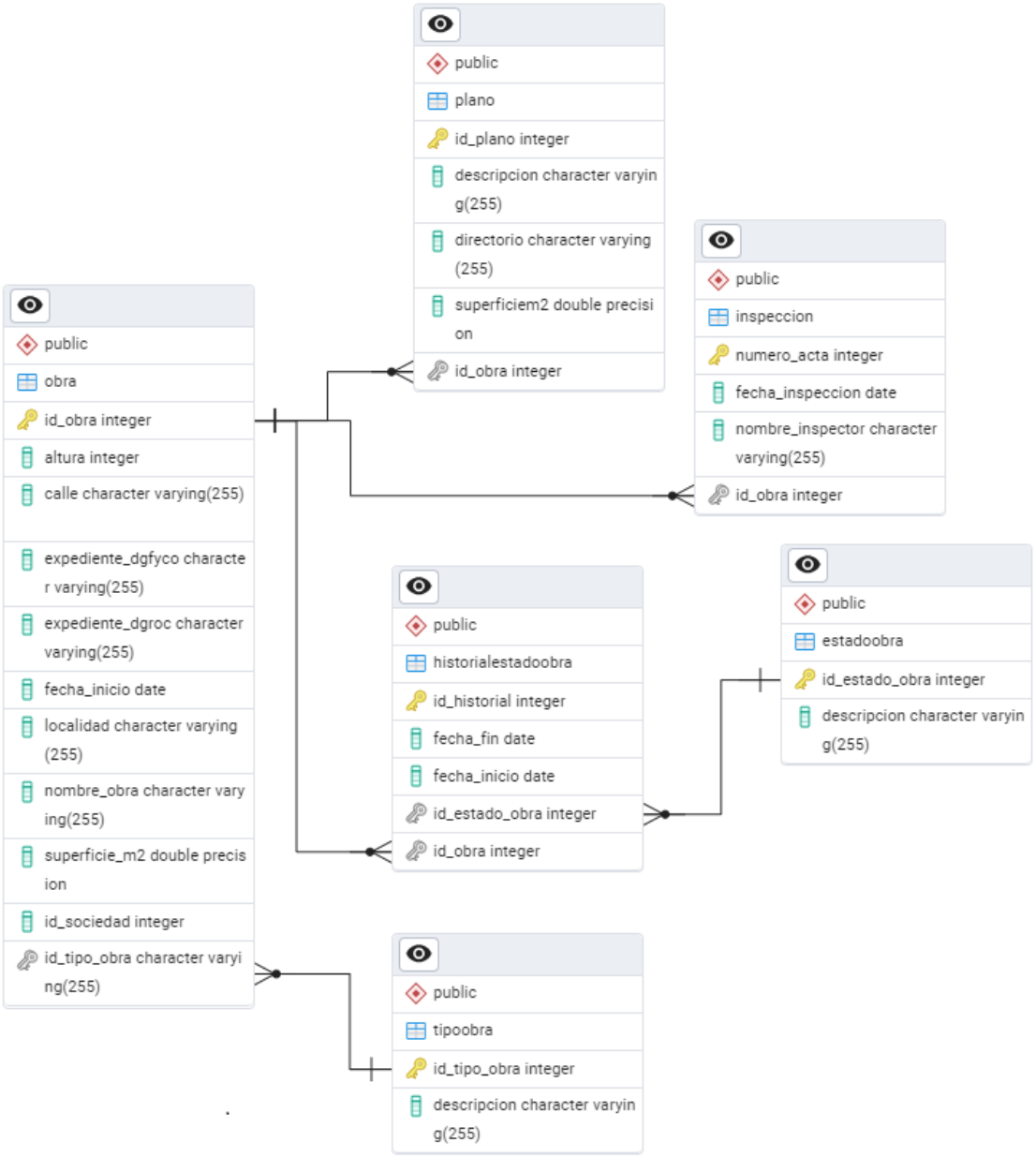

public
plano
id_plano integer
descripcion character varying(255)
directorio character varying(255)
superficiem2 double precision
id_obra integer
public
obra
id_obra integer
altura integer
calle character varying(255)
expediente_dgfyco character varying(255)
expediente_dgroc character varying(255)
fecha_inicio date
localidad character varying(255)
nombre_obra character varying(255)
superficie_m2 double precision
id_sociedad integer
id_tipo_obra character varying(255)
public
inspeccion
numero_acta integer
fecha_inspeccion date
nombre_inspector character varying(255)
id_obra integer
public
historialestadoobra
id_historial integer
fecha_fin date
fecha_inicio date
id_estado_obra integer
id_obra integer
public
estadoobra
id_estado_obra integer
descripcion character varying(255)
public
tipoobra
id_tipo_obra integer
descripcion character varying(255)

Figura 15

Diagrama entidad-relación. Obras, facturaciones y presupuestos

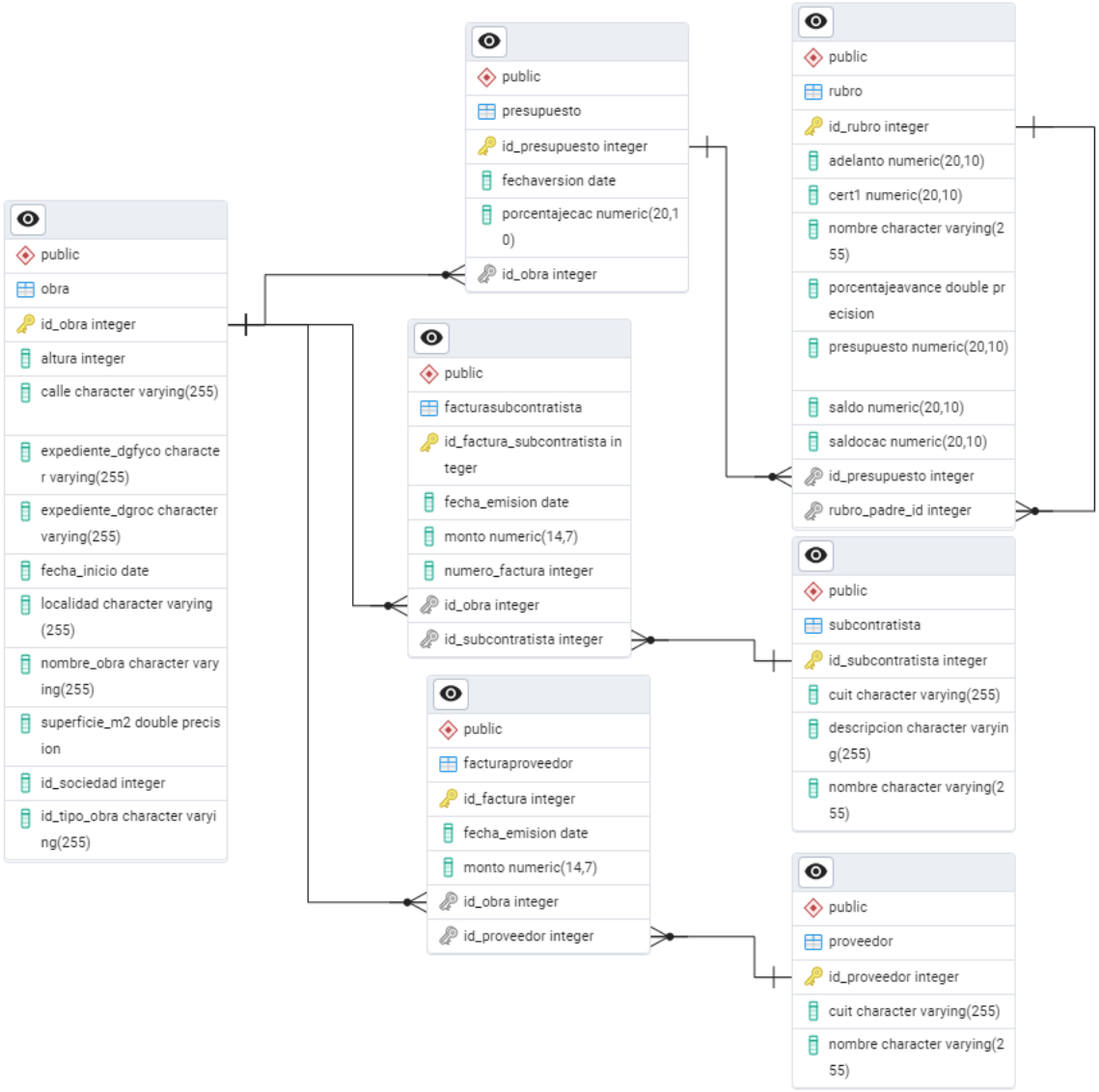


Figura 16

Diagrama entidad-relación. Obras y sociedades

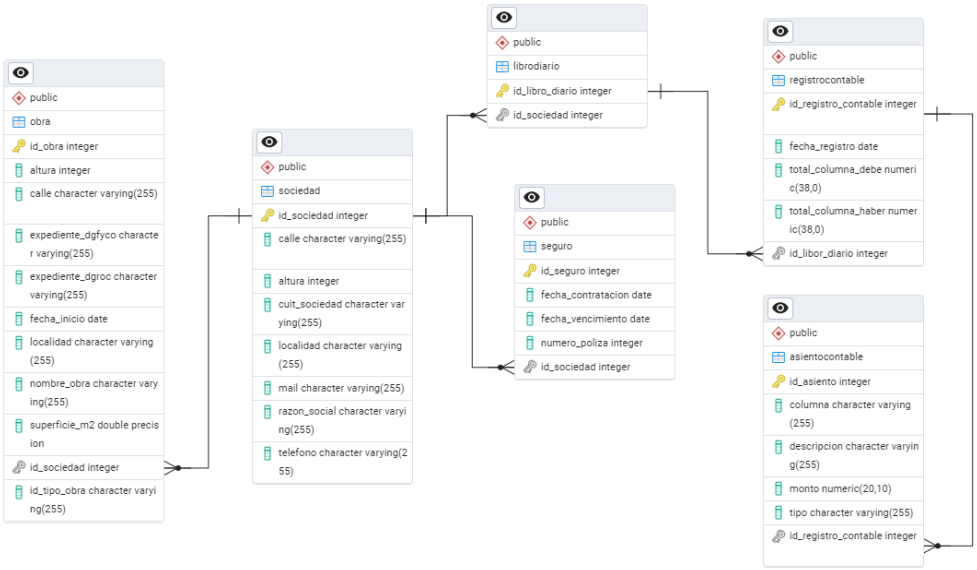
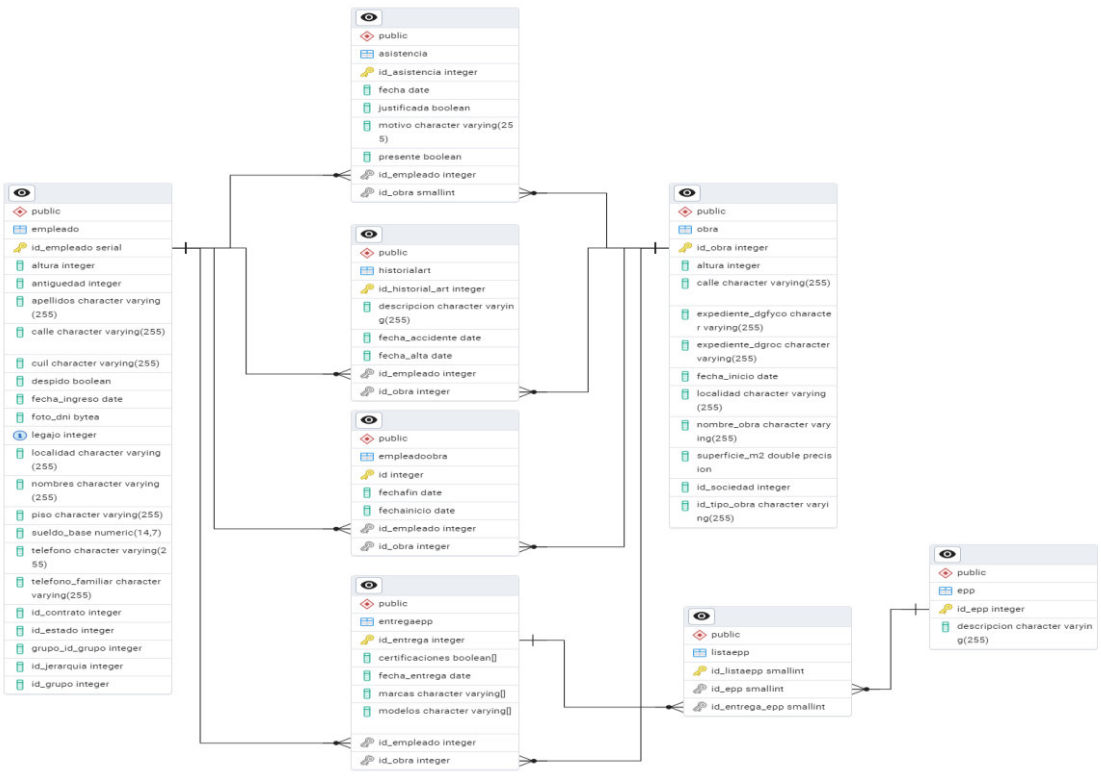


Figura 17

Diagrama entidad-relación. Empleados, obras y relaciones intermedias



11. Diagrama de clases

El diagrama de clases describe la estructura de un sistema, mostrando las clases, atributos y métodos y relaciones entre ellos.

Las clases en el diagrama están representadas por cuadros. Los cuadros están compuestos por tres partes:

- Sección superior: contiene el nombre de la clase
- Sección media: contiene los atributos de la clase. Cada atributo posee niveles de acceso y se representan por los símbolos “+” y “-”.
    - Símbolo +: el atributo es público, puede ser accedido y modificado por otras clases
    - Símbolo -: el atributo es privado, solo puede ser accedido por la propia clase.
- Sección inferior: incluye operadores de clases (métodos). Los operadores describen como una clase puede interactuar con los datos. Los operadores también poseen niveles de acceso representados por los símbolos “+” y “-”.

El diagrama también cuenta con diferentes tipos de flechas que reflejan el tipo de relación entre clases:

- Relación de agregación: representado por una flecha con punta rombo blanco que apunta desde la clase contenida hacia la clase contenedora. La clase contenida y la clase contenedora pueden existir de forma independiente.
- Relación de composición: representada por un rombo negro que apunta desde la clase contenida hacia la clase contenedora, la diferencia con la relación anterior radica en que la clase contenida no puede existir son la clase contenedora
- Dependencias: un cambio en la clase “A” provoca cambios en la clase “B”, por lo tanto la clase B depende de la clase A. Esta relación utiliza el objeto de otra clase como parámetro, el objeto que depende de otra clase no puede existir sin la clase a la que depende. Se representa esta relación con una flecha punteada.

La cardinalidad refleja el número de instancias de una clase que se relaciona con una instancia de otra clase. Consiste en una combinación de número y símbolos que indica el rango de cardinalidad permitido entre dos clases.

- 1: uno a uno, una instancia de la clase A se relaciona con una instancia de la clase B.
- *: cero o más, una instancia de la clase A puede estar relacionada con cero o más instancias de la clase B.
- 1..* : uno a muchos, una instancia de la clase A puede estar relacionada con una o más instancias de la clase B.
- 0 ... 1 : cero o uno, una instancia de la clase A puede estar relacionada con cero o una instancia de la clase B.
- 0..* : cero o más, una instancia de la clase A puede estar relacionada con cero o más instancias de la clase B.

El modelado del diagrama de clase basado en las especificaciones del cliente y en el modelado de diagrama entidad-relación se puede observar en las siguientes figuras.

Figura 18

Diagrama de clases, empleados.

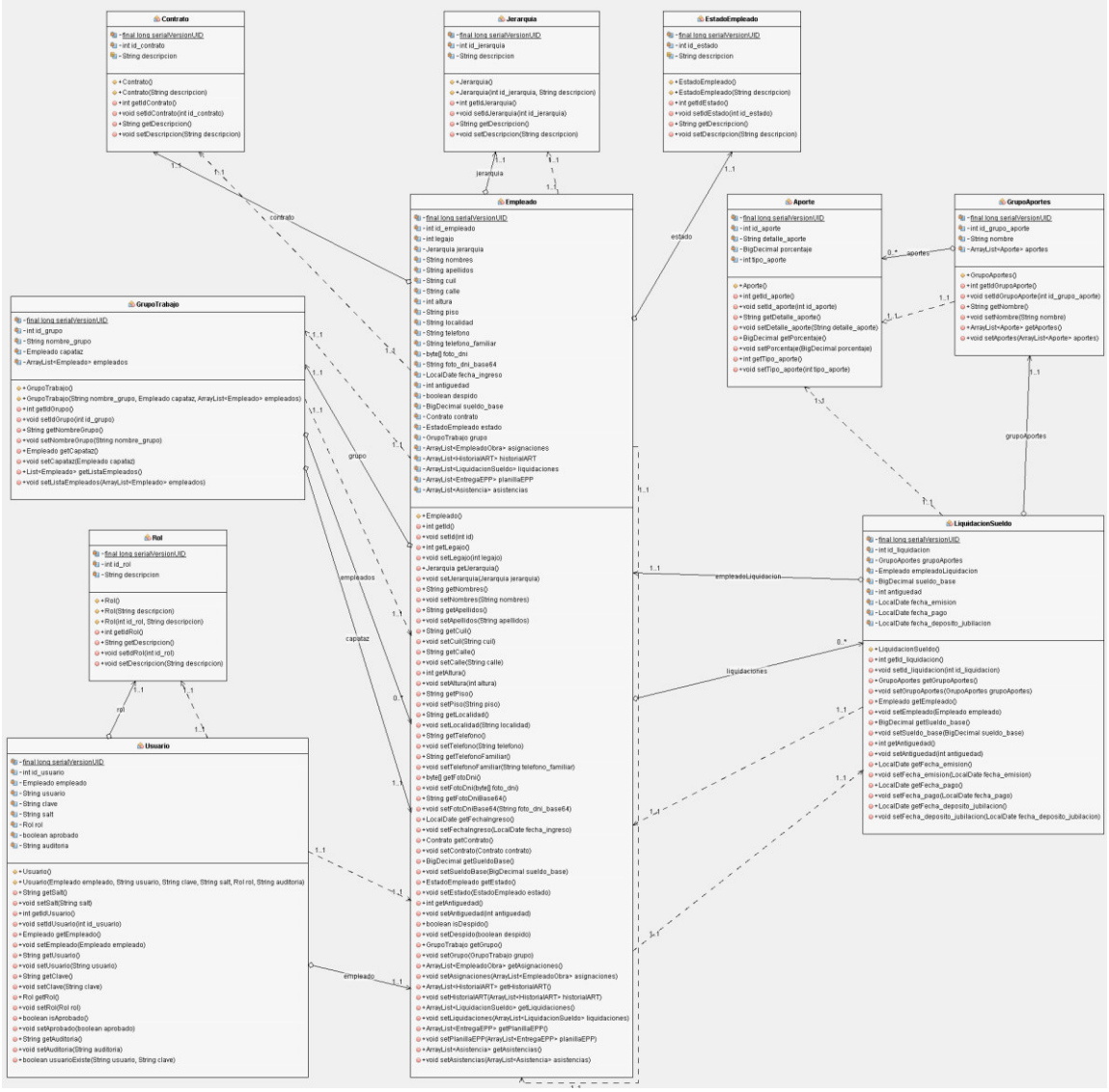


Figura 19

Diagrama de clases, Obras.

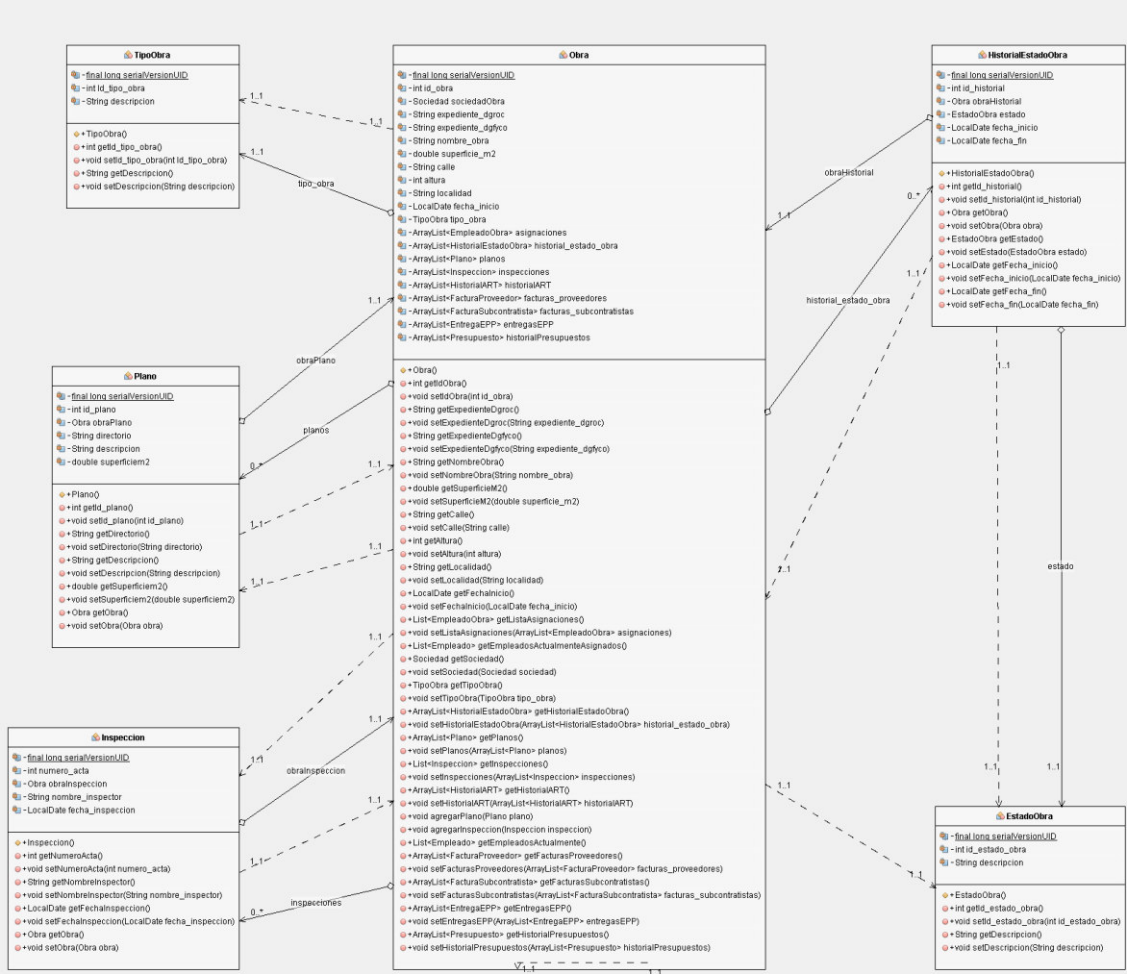


Figura 20

Diagrama de clases, Obras, facturaciones y presupuestos

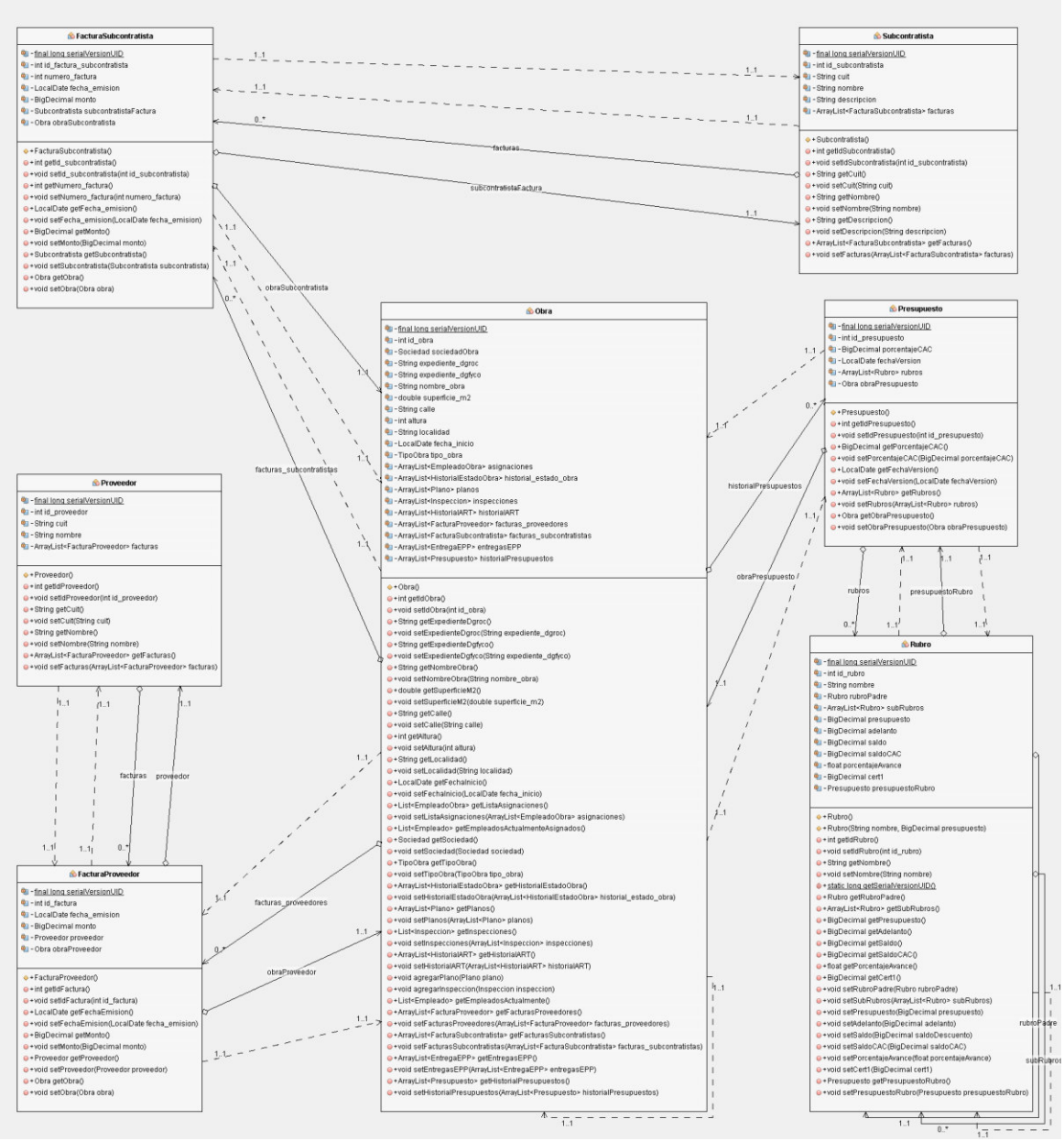


Figura 21

Diagrama de clases. Obras y sociedades.

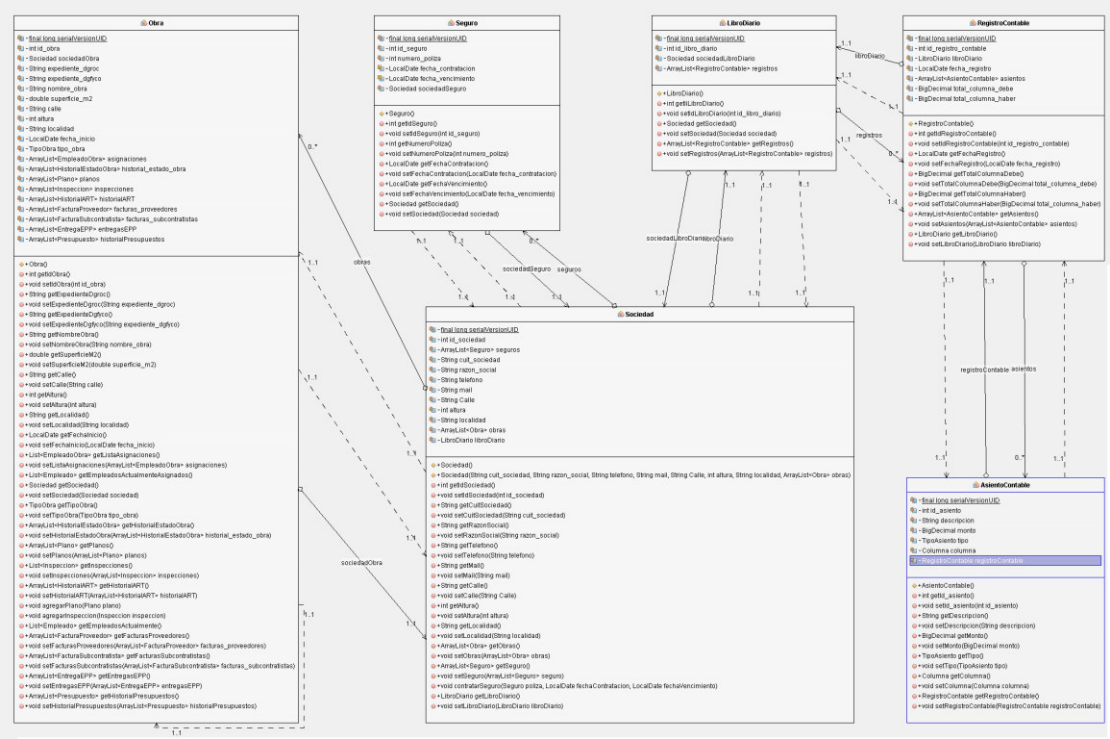
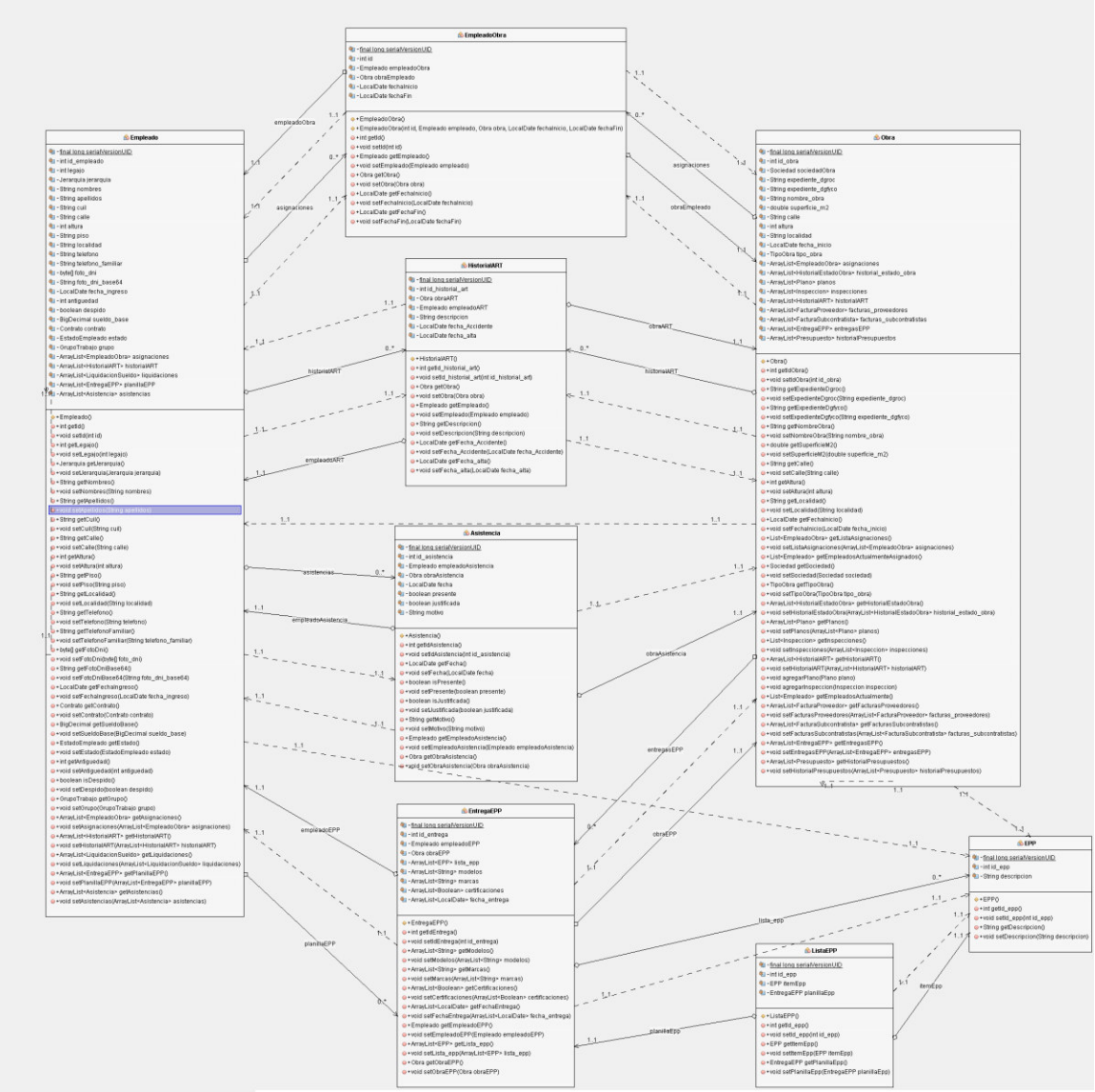


Figura 22

Diagrama de clases. Obras y empleados.



12. Arquitectura

El paso previo al desarrollo de la aplicación es la definición de la arquitectura. En esta etapa se define que tipo de aplicación se desarrollará en función de los lenguajes de programación a seleccionar, patrones de diseño, la base de datos a implementar, el entorno de desarrollo y los frameworks a utilizar

12.1. Tipos de aplicaciones

Existen dos tipos de aplicaciones a desarrollar, aplicaciones de escritorio y aplicaciones web. Cada una tiene sus características, puntos a favor y en contra.

A continuación, se evaluarán las características de cada una de las aplicaciones para poder definir cuál aplicación es más adecuada en función de las necesidades del cliente.

Tabla 84

Comparación entre aplicación de escritorio y aplicación web.

Aplicaciones de escritorio	Aplicaciones web
Se instalan en la computadora	No se instalan en la computadora
Se ejecutan en el sistema operativo y brinda mayor acceso a los recursos del sistema	Se ejecutan en un servidor y se accede mediante un navegador web
Tiempos de respuesta más rápidos	Tiempos de respuesta menos veloz en comparación a aplicaciones web
No requieren conexión de red	Requieren conexión de red
Generalmente se ejecutan para un sistema operativo específico	Se ejecutan en cualquier sistema operativo
Se requiere actualizaciones en cada computadora	Actualizaciones centralizadas, no requiere instalación en cada computadora

En función de los requerimientos del cliente, se necesita una aplicación que sea portable, segura y que pueda ser utilizada por múltiples usuarios. Las aplicaciones web cumplen con la mayoría de estos requerimientos

- Portabilidad: la aplicación web se accede mediante un navegador web, no dependen de un sistema operativo, no requieren instalación ni configuración por parte del usuario

- Rendimiento: generalmente, una aplicación de escritorio tiene tiempos de respuesta más rápido, pero con las nuevas tecnologías, las aplicaciones web están casi a la par.
- Seguridad: cualquiera de las dos aplicaciones se puede desarrollar y configurar de manera segura.
- Simultaneidad: la aplicación web puede soportar acceso de múltiples usuarios

En conclusión, se decidió por desarrollar una aplicación web.

12.2. Lenguajes de programación

Al optar por el desarrollo de una aplicación web, se requiere de un lenguaje de programación apto para el servidor. A continuación, se evaluaron las características de cuatro de los lenguajes de programación más utilizados frecuentemente para el desarrollo de aplicaciones web.

12.2.1. Java

Java es un lenguaje de programación ampliamente utilizado para codificar aplicaciones web. Ha sido una opción popular entre los desarrolladores durante más de dos décadas, con millones de aplicaciones Java en uso en la actualidad. Java es un lenguaje multiplataforma, orientado a objetos y centrado en la red que se puede utilizar como una plataforma en sí mismo. Es un lenguaje de programación rápido, seguro y confiable para codificarlo todo, desde aplicaciones móviles y software empresarial hasta aplicaciones de macro datos y tecnologías del servidor.

Las características principales de Java incluyen:

- Orientado a objetos: Java sigue la programación orientada a objetos para construir el diseño del programa en torno a objetos y clases. Esta propiedad de programación aumenta la reutilización y productividad de Java.
- Simple y seguro: es fácil dominar Java debido a su sintaxis simple y su naturaleza OOP. De manera similar, los enfoques de autenticación utilizan cifrado de clave pública para fortalecer la seguridad al utilizar Java. Además,

JVM actúa para proteger los datos contra manipulaciones y amenazas de virus.

- Multiproceso: esta característica permite que Java ejecute varias tareas al mismo tiempo. Menos uso de recursos del sistema, mejor capacidad de respuesta del servidor y comunicación son ventajas del subproceso múltiple.

Ventajas de Java:

- Java se considera una pila de tecnología altamente escalable que podría usarse en cualquier sistema operativo y dispositivo. También puede utilizarlo para múltiples sectores.
- El sistema API de Java está lleno de paquetes, clases, campos e interfaces. Por lo tanto, los equipos de desarrollo pueden utilizar Java para vincular una variedad de aplicaciones.

Limitaciones de Java:

- Java utiliza más recursos de procesamiento y memoria. Este inconveniente también aumenta los gastos de hardware.
- La ausencia de punteros limita el soporte de Java para la programación de bajo nivel.
- El control limitado sobre la recolección de basura también es una desventaja del uso de Java.

12.2.2. C#

C#, es una evolución que Microsoft realizó de este lenguaje, tomando lo mejor de los lenguajes C y C + +, y ha continuado añadiendo funcionalidades, tomando de otros lenguajes, como java, algo de su sintaxis evolucionada. Lo orientó a objetos para toda su plataforma NET (tanto Framework como Core), y con el tiempo adaptó las facilidades de la creación de código que tenía otro de sus lenguajes más populares, Visual Basic, haciéndolo tan polivalente y fácil de aprender como éste, sin perder ni un ápice de la potencia original de C.

En la versión de .NET Core, se ha reconstruido por completo su compilador, haciendo las aplicaciones un 600% más rápidas.

Las características principales de C# incluyen:

- Recolección de basura: C# administra la memoria de manera excelente con la ayuda de funciones de recolección de basura. Por lo tanto, esta función identifica y elimina automáticamente la memoria que no está en uso.
- Soporte IDE: este lenguaje aprovecha el uso de entornos de desarrollo integrados (IDE). C# cuenta con el soporte de JetBrains Rider, Visual Studio y Visual Studio Code. Los IDE son útiles para compilar código, probar marcos y depurar errores.
- Tipado fuerte: esta característica de C# permite a los desarrolladores escribir código más limpio sin fallos en el tiempo de ejecución ni en el tiempo de compilación. De manera similar, puede crear aplicaciones más seguras y organizadas con lenguajes tipográficos potentes.
- Interoperabilidad: permite que los lenguajes se comuniquen entre sí en el mismo programa. Con C#, la interoperabilidad es factible con código administrado y no administrado. Una mejor protección de datos, alta productividad y reducción de errores son ventajas clave de esta característica.

Ventajas de C#:

- C# es un producto de Microsoft, razón por la cual los equipos de desarrollo encuentran un apoyo comunitario masivo cuando codifican con C#. En este sentido, varios foros de discusión como ‘C# Developer Community’ y ‘C# Corner’ están disponibles para obtener respuestas a sus consultas.
- La funcionalidad de alto nivel, la sintaxis comprensible y una gran biblioteca garantizan un desarrollo rápido.
- C# pertenece a la familia de lenguajes C. De esta forma, tiene una gran compatibilidad con C + +, Java y C.

Limitaciones de C#:

- Si no utiliza .NET como su pila tecnológica principal, C# no es una opción adaptable para su proyecto.
- La curva de aprendizaje de C# es pronunciada, especialmente si la compara con lenguajes como Python y Java. Del mismo modo, a los desarrolladores les resulta difícil comprender las bibliotecas .NET.

- C# utiliza Common Language Runtime (CLR), lo que puede aumentar la sobrecarga de su tiempo de ejecución.

12.2.3. Python

Python es un lenguaje de programación de código abierto. Se trata de un lenguaje orientado a objetos, fácil de interpretar y con una sintaxis que permite leerlo de manera semejante a como se lee el inglés. Es un lenguaje interpretado, esto significa que el código de programación se convierte en bytecode y luego se ejecuta por el intérprete, que, en este caso, es la máquina virtual de Python.

Las características principales de Python incluyen:

- Curva de aprendizaje fluida: Python es un lenguaje de backend sencillo. Su sintaxis tiene similitudes con la del inglés, por lo que podrá escribirlo y leerlo sin esfuerzo. Muchas universidades incluso incluyen Python en sus cursos de nivel principiante.
- Bibliotecas: este lenguaje de backend posee bibliotecas masivas. Los desarrolladores pueden encontrar fácilmente la biblioteca estándar de Python y utilizar una variedad de constantes, variables, tipos y módulos integrados de forma gratuita. Algunas otras bibliotecas de Python de código abierto son Pandas, NumPy, TensorFlow y Scikit Learn.
- Portátil: Python permite a los desarrolladores utilizar el mismo código sin cambios en una variedad de máquinas. Sí escribe un código para Windows, Python le permite ejecutar el mismo código en Mac. Esta propiedad también reduce el tiempo de desarrollo.
- Orientado a procedimientos y orientado a objetos: este lenguaje sigue la naturaleza de la programación orientada a procedimientos y objetos al mismo tiempo. Por lo tanto, puede diseñar el software en torno a funciones y objetos según las necesidades de su proyecto.

Ventajas de Python:

- Python es un lenguaje de programación backend altamente flexible que las empresas pueden utilizar para industrias y aplicaciones modernas. De hecho, podría utilizarse para el desarrollo de aplicaciones de informática matemática, ciencia de datos, comercio, finanzas y SIG.

- Es robusto escribir el código con Python. Con líneas de código limitadas, puede realizar más funciones en comparación con otros lenguajes del lado del servidor.
- El soporte activo de la comunidad y la disponibilidad de módulos de terceros también lo convierten en un lenguaje atractivo.
- Limitaciones de Python:
- Python es lento en tiempo de ejecución, especialmente si lo comparamos con lenguajes como C + +, PHP, Java y Javascript. También consume más recursos de hardware.
- Esta tecnología de backend no se considera una opción confiable para crear aplicaciones móviles y de gran escala.
- El soporte insuficiente para subprocesos múltiples y acceso a bases de datos también son algunas desventajas de la programación con Python.

12.2.4. PHP

PHP es un lenguaje de programación destinado a desarrollar aplicaciones para la web y crear páginas web, favoreciendo la conexión entre los servidores y la interfaz de usuario. Entre los factores que hicieron que PHP se volviera tan popular, se destaca el hecho de que es de código abierto. Esto significa que cualquiera puede hacer cambios en su estructura.

Las características principales de PHP incluyen:

- Altamente flexible: PHP es flexible para integrarse con JS, XML y HTML. De igual forma, tiene una gran compatibilidad con diversas bases de datos, dispositivos y servidores.
- Simple: este lenguaje de backend es fácil de entender y aprender. Las funcionalidades predeterminadas y la naturaleza coherente de PHP mantienen la programación fluida tanto para desarrolladores experimentados como novatos.
- Sensible a las mayúsculas y minúsculas: es una tecnología del lado del servidor que distingue ligeramente entre mayúsculas y minúsculas. Sí, los nombres de las variables distinguen entre mayúsculas y minúsculas en PHP. Sin embargo, los nombres de clases y funciones no distinguen entre mayúsculas y minúsculas.

- Seguro: PHP es un lenguaje seguro. En este sentido, no sólo tiene funciones de cifrado de datos prediseñadas, sino que los desarrolladores también pueden emplear herramientas de terceros para obtener una mayor protección. Además, PHP
- utiliza algoritmos de seguridad para proporcionar un entorno seguro para las secuencias de comandos.
- Rápido: Es un lenguaje ultrarrápido. De hecho, PHP supera el tiempo de carga porque consume menos memoria durante la ejecución del código. Esta característica también ayuda a mejorar el rendimiento de una aplicación.

Ventajas de PHP:

- PHP viene con una curva de aprendizaje sencilla. Eso significa que no es difícil comprender y codificar este lenguaje de programación para desarrollar backend.
- Las pequeñas empresas prefieren adoptar PHP para sus empresas debido a su naturaleza escalable. Seguramente puede ampliar sus recursos virtuales y físicos con PHP en cualquier momento.
- Tiene una comunidad activa. Por lo tanto, puede beneficiarse de las experiencias de desarrolladores PHP experimentados.

Limitaciones de PHP:

- PHP confiere herramientas de segunda categoría para el manejo de errores. Del mismo modo, las herramientas de depuración insuficientes también son un inconveniente del uso de PHP.
- Este no es un lenguaje versátil, por lo que es difícil usar PHP para proyectos de IA, ML y big data.
- Los fallos de seguridad y la escasez de bibliotecas dedicadas también son desventajas importantes de la programación con PHP.

12.2.5. Conclusiones

El lenguaje de programación C# ofrece mayor rendimiento y velocidad, sin embargo, se descartó esta opción ya que su curva de aprendizaje es más pronunciada y el proyecto no cuenta con flexibilidad en cuanto al tiempo de finalización.

Los lenguajes de programación Python, Java y PHP poseen características similares. A continuación se mencionan algunas características que diferencian un lenguaje de programación del otro.

- Python es más lento en tiempo de ejecución en comparación con los lenguajes Java y PHP
- PHP es más veloz, sin embargo, no cuenta con muchas herramientas de depuración, lo que podría conllevar a retrasos en la fase de pruebas del proyecto
- Java requiere de más hardware para su ejecución, pero soporta la programación multihilo, lo que permite la ejecución de varios procesos simultáneamente.

El lenguaje C# es el más óptimo a utilizar, sin embargo, por el tiempo limitado del proyecto y su curva de aprendizaje, se optará por usar el lenguaje de programación Java.

12.3. Bases de datos

Una base de datos es un conjunto de datos estructurados que pertenecen a un mismo contexto, y se utiliza para administrar de forma electrónica grandes volúmenes de datos.

Existen varios tipos de bases de datos, en esta sección se analizarán las características de dos de las bases de datos más utilizadas, bases de datos SQL y NOSQL, con el fin de determinar cuál se ajusta más a los requerimientos del cliente y del proyecto.

12.3.1. Base de datos SQL

Las bases de datos SQL disponen de una serie de ventajas que las han convertido en el tipo de base de datos más utilizada. Las principales ventajas son:

- Dispone de herramientas que permiten evitar la duplicidad de registros, garantizando la integridad referencial (al eliminarse un registro, se eliminan todos los registros relacionados dependientes del mismo).
- Tienen un mayor soporte al llevar mucho tiempo en el mercado (mayor comunidad, aplicaciones y complementos).
- Atomicidad de la información. Al realizar cualquier operación en la base de datos, si surge algún problema, la operación no se realiza.

- Dispone de un sistema estándar bien definido (SQL) para las operaciones con la base de datos, como inserción, actualización o consultas. Este sistema es sencillo de comprender ya que se adapta al lenguaje común.

¿En qué casos utilizar SQL?

Cuando en un proyecto el volumen de los datos no tendrá un gran crecimiento, o este se realice de forma lenta, las bases de datos SQL.

En proyectos donde el pico de usuarios que accedan a la base de datos esté previsto, las bases de datos relacionales funcionan de manera óptima.

Si las necesidades de procesamiento de la base de datos requieren de un único servidor, se pueden utilizar bases de datos SQL.

12.3.2. Base de datos NOSQL

Las principales ventajas de las bases de datos no relacionales NOSQL son:

- Son bases de datos versátiles que permiten agregar información o hacer cambios en el sistema sin necesidad de agregar configuraciones extra
- Las bases de datos NOSQL open source no requieren del pago de licencia y no necesitan un hardware muy potente para poder ser ejecutadas.
- Soportan el crecimiento horizontal, es decir, al soportar estructuras distribuidas se pueden instalar nuevos nodos operativos que balancean la carga de trabajo. Es más fácil su expansión debido a este escalado horizontal.
- Permiten guardar datos de cualquier tipo, en cualquier momento, sin requerir una verificación.
- Realizan consultas utilizando JSON (*Javascript Object Notation*, formato sencillo de intercambio de texto).

¿En qué casos utilizar NOSQL?

Si el crecimiento de la base de datos se realiza de forma rápida, con grandes aumentos en poco tiempo, lo ideal es recurrir a bases de datos no relacionales NOSQL.

Si el acceso a la base de datos puede sufrir picos altos y en múltiples ocasiones, lo mejor es optar por No SQL.

Si las necesidades de procesamiento no se pueden prevenir, es mejor utilizar bases de datos NOSQL escalables (permiten expandirse).

12.3.3. Conclusiones

Una base de datos NOSQL requiere de poco hardware, es veloz, versátil y escalable, sin embargo, la consistencia de datos es baja en comparación a una base de datos SQL.

Por el modelo de negocio del cliente, se requiere una gran consistencia de datos, ya que la base de datos a utilizar almacenará datos sensibles útiles para gestionar la contabilidad, datos de empleados, obras.

Se optara por utilizar una base de datos SQL ya que su consistencia de datos es una característica importante para el desarrollo de este proyecto

12.4. Modelo cliente-servidor

Al optar por el desarrollo de una aplicación web, la arquitectura de la misma será la del modelo cliente-servidor.

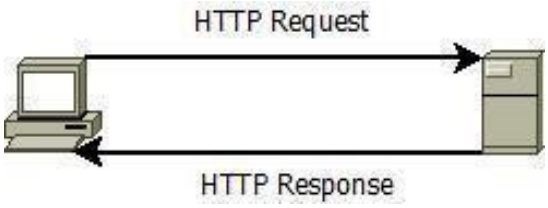
El concepto de cliente-servidor es un modelo que vincula a varios dispositivos informáticos a través de una red. Los dispositivos clientes realizan peticiones de servicio a los dispositivos servidores, que se encarga de satisfacer dichas peticiones.

Este tipo de modelo permite repartir la capacidad de procesamiento. El servidor puede ejecutarse sobre más de un equipo. Los clientes centralizan la aplicación y recursos en el servidor. El servidor, a su vez, se encarga de que estos recursos estén disponibles cada vez que el cliente los requiera.

La comunicación entre clientes y servidores utiliza el protocolo de comunicación HTTP (Hypertext Transfer Protocol)

Figura 23

Comunicación protocolo HTTP



12.4.1. ***Protocolo HTTP***

Es un protocolo el cual permite a un cliente, generalmente un navegador web, realizar peticiones (Request) de datos y recursos, como pueden ser documentos HTML (Hypertext Markup Language), hojas de estilo CSS (Cascade Style Sheet) textos, imágenes, videos, scripts, etc. Y el servidor en el que se almacena dichos datos responde al cliente (Response).

Clientes y servidores se comunican intercambiando mensajes individuales.

Cada petición individual se envía a un servidor el cual gestiona y responde. Entre cada petición y respuesta hay varios intermediarios, normalmente denominados Proxies, los cuales realizan distintas funciones como gateways o caché. Gracias a la arquitectura en capas de la red, estos intermediarios son transparentes al navegador y el servidor ya que HTTP se apoya en los protocolos de red y transporte.

Algunas de las peticiones que un cliente puede solicitar son los métodos GET y POST

- Método GET: solicita una representación de un recurso específico. Las peticiones que utilizan el método GET solo deben recuperar datos. A la petición GET puede añadirse información con la intención que el servidor web la procese. Esta información se añade a la URL de la siguiente forma

  1. La secuencia de petición se inicia con un signo de interrogación “?”.
  2. Todos los parámetros se componen de un nombre y un valor “nombre=valor”.
  3. Si se han adjuntado varios parámetros, estos se concatenan con el carácter “&”.

  Por ejemplo, si se quisiera consultar un listado de empleados con nombre y apellido, la URL resultante sería

  GET /search?nombre=Juan&apellido=Pérez

- Método POST: se utiliza cuando se tiene que enviar al servidor web paquetes grandes de datos o datos de formularios de carácter privado. Este método no escribe información en la URL, sino que lo adjunta en el encabezado HTTP, de esta forma la información enviada ya no es legible a simple vista.

12.5. Aplicación java web

La aplicación web a implementar se desarrollará con el lenguaje de programación Java. Para llevar esto a cabo, Java cuenta con JSP (Java Server Pages).

Es una tecnología que permite crear páginas web dinámicas basadas en HTML y XML.

JSP requiere de un servidor web que sea compatible con contenedores Servlet, como por ejemplo Apache Tomcat.

Cada JSP contendrá mayoritariamente etiquetas de HTML, sin embargo, podrá utilizar etiquetas especiales para poder insertar código Java en donde sea necesario.

Un Servlet es una clase de java que se usa como un punto intermedio entre una página JSP y el servidor web donde están alojados los recursos de la aplicación.

El Servlet se encarga de recibir una petición Request desde un cliente, tratarlas y analizar si es necesario realizar alguna solicitud en particular o brindar alguna respuesta Response al cliente.

Si bien los Servlet no son tecnologías novedosas, tampoco son obsoletas. Cuentan con gran cantidad de documentación, foros de programación y es una tecnología estable.

Los servlets tienen diferentes métodos que pueden ser utilizados dependiendo del tipo de solicitud que reciban por parte de un cliente, los dos más utilizados son:

- doGet (): implementado para recibir solicitudes Request GET del cliente y responder al cliente mediante Response.
- doPost (): implementado para recibir solicitudes Request POST del cliente y responder al cliente mediante Response.

12.6. Javascript

Javascript es un lenguaje de programación que se ejecuta del lado del cliente y permite ejecutar funciones complejas en una página web. Con Javascript se puede acceder al DOM (document Object model) de las páginas HTML, insertar contenido HTML y cambiar estilos de estos, generando páginas dinámicas.

JQuery es una biblioteca de Javascript que simplifica la manipulación de elementos HTML y CSS en una página web con menor cantidad de código.

JQuery será un nexo entre las páginas JSP y los servlets, capturando información de los formularios HTML, enviando la información mediante AJAX al Servlet, recibiendo la

respuesta del Servlet, procesando la información recibida e insertándose en las páginas JSP.

12.7. AJAX

AJAX (Asynchronous Javascript And XML) es un término que describe un modo de utilizar conjuntamente varias tecnologías existentes para crear aplicaciones web dinámicas.

Estas tecnologías incluyen:

- HTML o XHTML para la estructura de la página.
- CSS: hoja de estilo en cascada para el diseño visual.
- Javascript: lenguaje de programación para la lógica de negocio y la interacción con el usuario.
- DOM para acceder y manipular el contenido de la página.
- XMLHttpRequest: componente para la comunicación asíncrona con el servidor

AJAX permite a las aplicaciones web intercambiar información con el servidor de manera sencilla, sin recargar la página completa. Esto se logra mediante el uso del objeto XMLHttpRequest, que permite solicitar y recibir recursos en segundo plano, mientras el usuario sigue interactuando con la página.

JQuery ofrece varios métodos para realizar solicitudes AJAX, pero el más común y versátil es $.ajax (). Este método permite configurar la solicitud AJAX a través de un objeto que contiene propiedades como:

- url: la dirección url del servidor para la solicitud.
- data: los datos a enviar al servidor (pueden ser objetos, arrays o cadenas).
- type: el método HTTP para la solicitud (GET, POST).
- dataType: el tipo de dato esperado en la respuesta (JSON, HTML, XML, texto, etc.).
- Success: una función a ejecutar si la solicitud es exitosa.
- error: una función a ejecutar si la solicitud falla.
- Complete: una solicitud a ejecutar después de que la solicitud se complete, independientemente de su resultado.

Figura 24

Estructura de un AJAX

```
$.ajax({
  url: 'datos.php',
  data: { id: 123 },
  type: 'GET',
  dataType: 'json',
  success: function(data) {
    $('#resultados').html(JSON.stringify(data));
  },
  error: function(xhr, status) {
    alert('Error al cargar datos');
  }
});
```

12.8. JSON

JSON (Javascript Object Notation) es un formato de intercambio de datos, ligero y legible, basado en texto que se utiliza para representar información estructurada.

JSON se define como una serie de pares clave-valor donde las claves son cadenas de texto, y los valores pueden ser:

- cadenas de texto
- números enteros o decimales
- boolean (true, false)
- arreglos (array) de elementos JSON
- Objetos (object) JSON anidados

Los datos JSON se ordenan en una estructura jerárquica, utilizando llaves “{ }” para definir objetos, y corchetes “[ ]” para definir arreglos, los pares clave-valor están separados por comas.

Figura 25

Estructura de un JSON

12.9. Programación orientada a objetos

La programación orientada a objetos es un paradigma de la programación que parte del concepto de “objetos” como base, los cuales pueden contener información en forma de atributos, y código en forma de métodos.

Los objetos son capaces de interactuar y modificar los valores contenidos en sus atributos a través de sus métodos.

El modelo de negocio presentado en la ilustración 17 seguirá las pautas de desarrollo de programación orientada a objetos.

Cada instancia de un objeto estará asociada a una entidad de la base de datos por medio de una herramienta de java llamada JPA (Java persistence application).

12.10. Patrón de diseño MVC (Model, View, Controller)

El patrón MVC es una arquitectura comúnmente utilizada en el desarrollo de aplicaciones web y de escritorio. Su objetivo es separar la lógica de negocio, la presentación y la interacción con el usuario en tres capas diferentes, lo que facilita el desarrollo, la mantenibilidad y la escalabilidad de la aplicación.

- Modelo: representa la capa de la lógica de negocio. Se encarga de gestionar los datos y la lógica del sistema. El modelo puede interactuar con una base de datos, realizar cálculos o validar datos.
- Vista: es la capa de interfaz de usuario. Se encarga de mostrar los datos y la lógica del sistema a los usuarios.

- Controlador: es la capa de enlace entre el modelo y la vista. Se encarga de recibir solicitudes de la vista, interactuar con el modelo para obtener o actualizar datos, y luego pasar los datos a la vista para su presentación.

12.11. Java persistence application

La java persistence application es una API (Application programming interface) estándar para la persistencia de objetos en java, que permite a los desarrolladores interactuar con las bases de datos relacionales de manera transparente y eficiente.

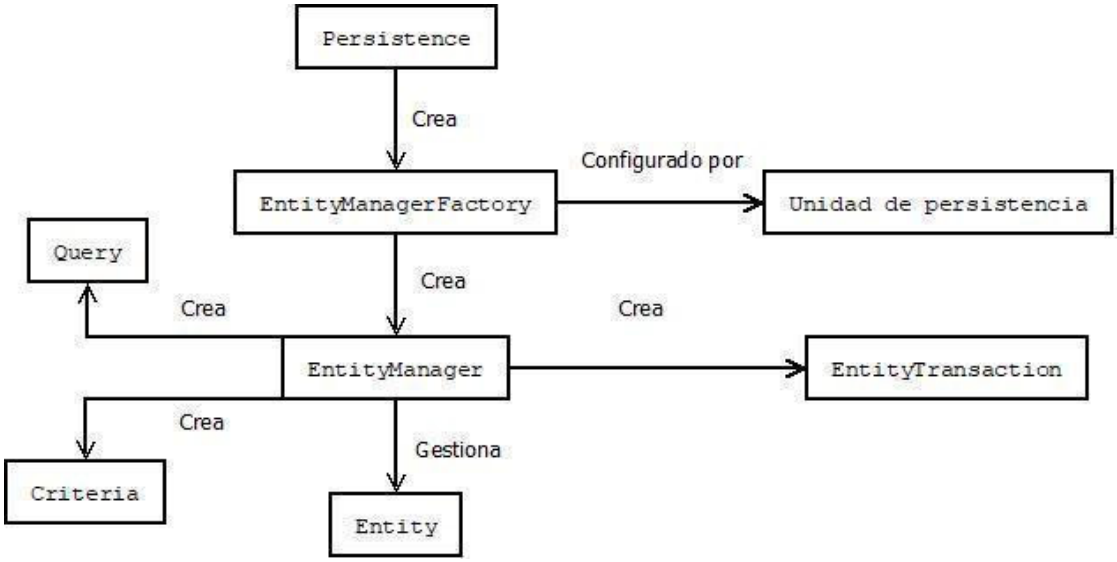
JPA proporciona las siguientes funcionalidades:

- Mapeo objeto-relacional: permite mapear objetos java a tablas y columnas en una base de datos relacional, de manera tal que los objetos java puedan persistir y recuperarse desde la base de datos.
- Persistencia: permite a los objetos java ser persistidos en una base de datos relacional y recuperados posteriormente
- Consultas: admite consultas definidas en un lenguaje de consulta específico, conocido como Java Persistence Query Language (JPQL) que se traduce a consultas SQL para interactuar con la base de datos

Dado que JPA es una especificación, no proporciona clases para poder trabajar con la información, lo que hace es proveer de una serie de interfaces que se puede utilizar para implementar la capa de persistencia de la aplicación, apoyándose en algunas implementaciones concretas de JPA.

Figura 26

Arquitectura de la API de JPA



12.12. EclipseLink

EclipseLink es una implementación de la especificación de Java Persistence API (JPA) que proporciona una capa de abstracción entre la aplicación Java y las bases de datos relacionales. Permite a los desarrolladores acceder y manipular datos en una base de datos utilizando un modelo de objetos Java, sin necesidad de escribir código SQL directo.

EclipseLink ofrece varias características como:

- Mapeo objeto-relacional
- Acceso a bases de datos: admite diferentes tipos de bases de datos, como Oracle, MySQL, PostgreSQL, Microsoft SQL server.
- Caching: permite almacenar objetos y resultados de consultas en caché para mejorar el rendimiento
- Web services: puede ser utilizado para generar servicios web que accedan a bases de datos

12.13. PostgreSQL

PostgreSQL es una base de datos de código abierto que se caracteriza por su fiabilidad, flexibilidad y soporte de estándares técnicos abiertos. A diferencia de otros sistemas gestores de bases de datos, PostgreSQL soporta tipos de datos relacionales y no relacionales, esto lo convierte en una de las bases de datos más compatibles, maduros y estables disponibles actualmente.

PostgreSQL es expandible y versátil, por lo que soporta rápidamente una variedad de casos de uso especializados con un poderoso ecosistema de extensión, que abarca desde tipos de datos de series de tiempo hasta análisis geoespaciales.

Creado como una base de datos de código abierto, PostgreSQL está libre de restricciones de licencia.

12.14. Resumen de la arquitectura

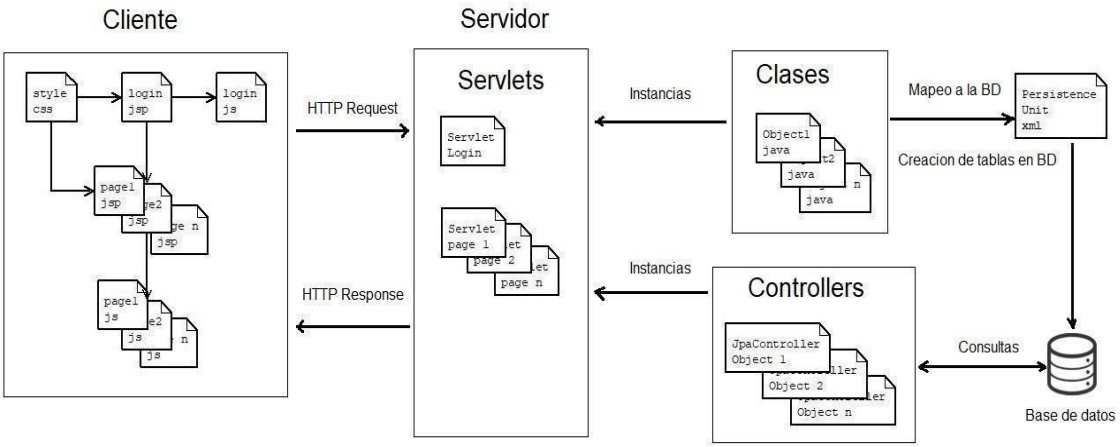
Finalmente, el software a desarrollar será una aplicación Java web, compuesta por páginas JSP (Java server pages) para la estructura de la vista, archivos CSS (Cascade Style Sheet) para el diseño y Javascript para crear páginas dinámicas, gestión de solicitudes GET/POST, captura y presentación de información de cada página del lado del cliente.

Del lado del servidor, las clases Servlet recibirán las solicitudes GET/POST del cliente, procesarán la información y devolverán una respuesta.

Estas clases están mapeadas a la base de datos por medio de la notación de EclipseLink, y cada clase contará con una clase JpaController que se encargará de realizar las consultas a la base de datos.

Figura 27

Arquitectura de una aplicación java web



12.15. Servidor Apache Tomcat

Apache Tomcat es un servidor web que se ejecuta en una máquina virtual de java (JVM)

Y está diseñado para ser ligero y fácil de configurar. Como contenedor de Servlets y JSP, su función principal es procesar solicitudes HTTP de los clientes y enviar respuestas.

Tomcat también cuenta con una arquitectura modular que permite la integración de componentes adicionales y la extensión de su funcionalidad. Por ejemplo, se puede agregar nuevas bibliotecas de servlets, conectores de bases de datos y componentes de autenticación y autorización.

Tomcat también ofrece una serie de características de seguridad, lo que lo hace ideal para el desarrollo y despliegue de aplicaciones web empresarial. Por ejemplo, puede integrarse con una variedad de herramientas de seguridad como firewalls y herramientas de análisis de vulnerabilidades, para proteger a la aplicación de potenciales amenazas.

13. Instalación de herramientas de desarrollo

Una vez finalizado el análisis de la arquitectura, comienza la etapa de desarrollo de software. Esta fase consiste en la instalación de entornos de desarrollo, instalación de bases de datos, instalación de servidores y codificación del software.

13.1. Instalación de PostgreSQL y creación de una base de datos

El primer paso es la instalación de la base de datos, desde la página oficial de PostgreSQL se descargó el instalador versión 17.2.1

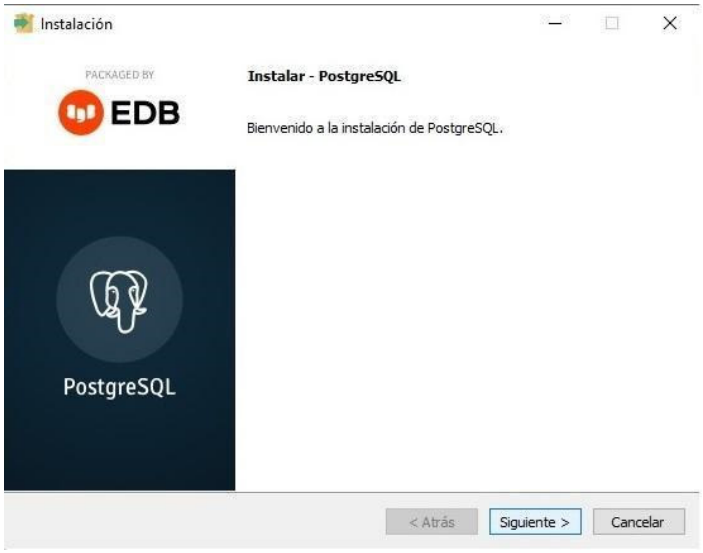
Entrar en la sección “Download” y seleccionar el tipo de sistema operativo que posee la computadora en la que se desarrollará la aplicación. A continuación se mostrará una lista informativa con las distintas versiones de PostgreSQL y su compatibilidad con las distintas distribuciones del sistema operativo.

En la parte superior de la pantalla se encuentra un link con la frase “Download the installer” que redirigirá a la página “Enterprisedb.com” en la que se podrá descargar el instalador de PostgreSQL correspondiente.

Una vez descargado, ejecuta el instalador e inicia la instalación.

Figura 28

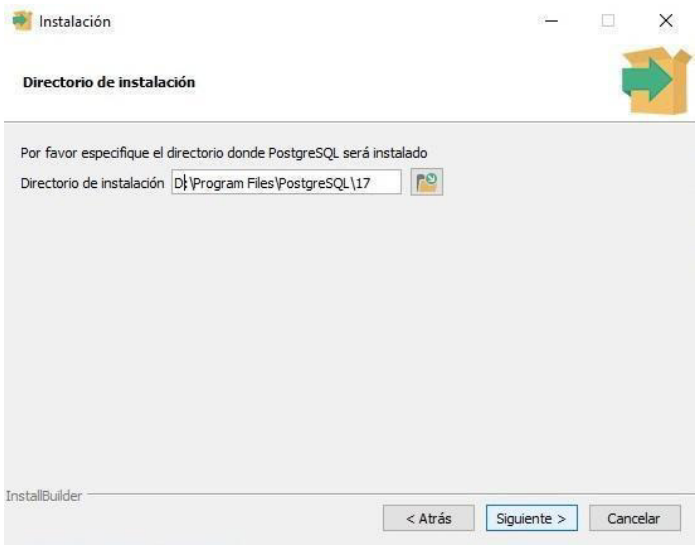
Instalación de PostgreSQL



El siguiente paso es determinar la ruta de instalación del programa

Figura 29

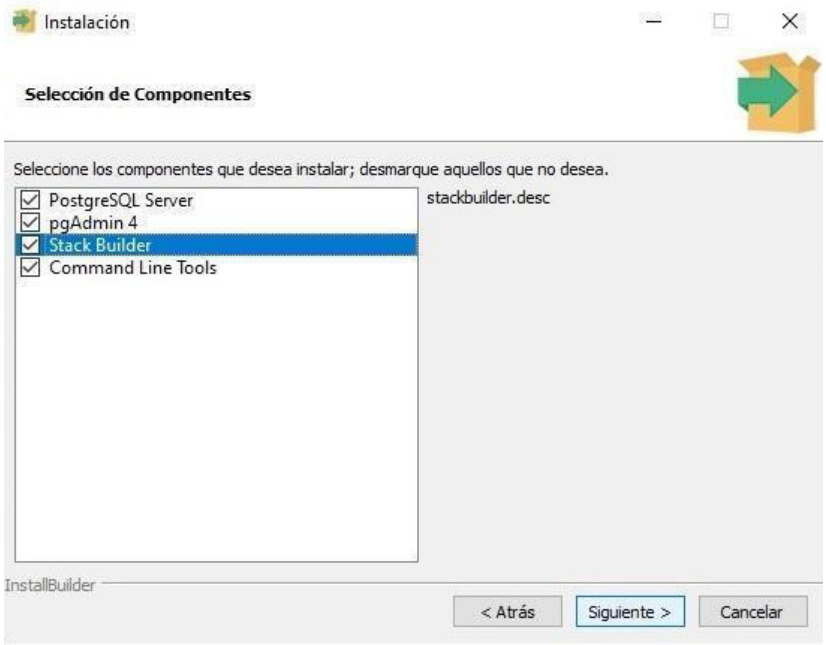
Ruta de instalación



A continuación se muestra una lista de componentes a instalar, esta lista no se modificará.

Figura 30

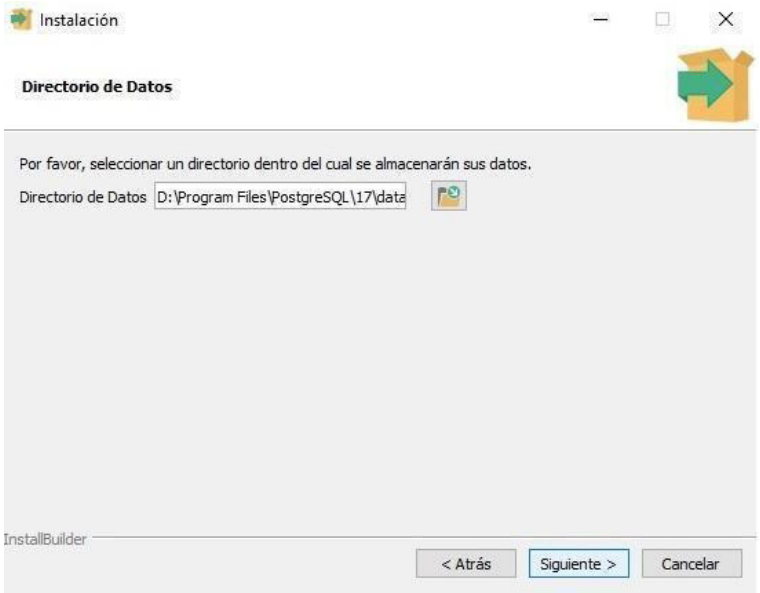
Selección de componentes



A continuación se deberá determinar la ubicación de almacenamiento de datos de PostgreSQL, en esta ruta se guardaran los datos de bases de datos, consultas, triggers, etc.

Figura 31

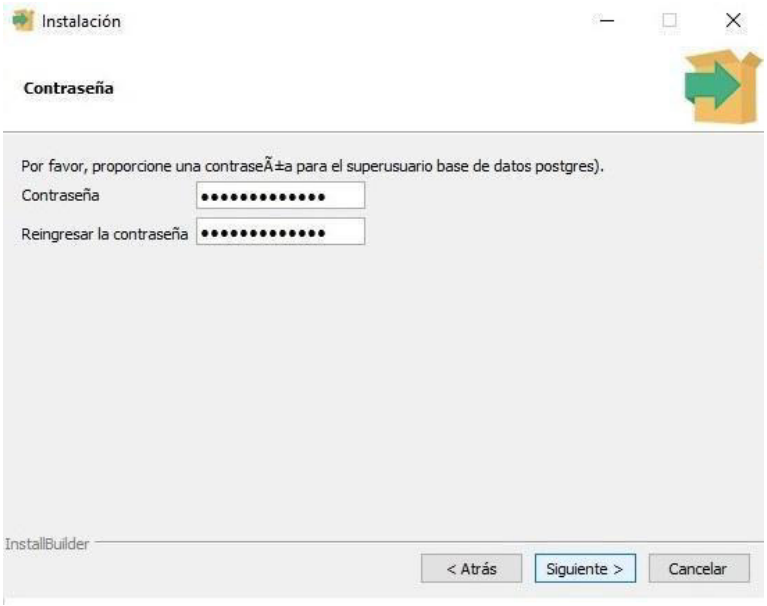
Ruta de almacenamiento de datos



Luego, se deberá ingresar una contraseña para acceder a PostgreSQL.

Figura 32

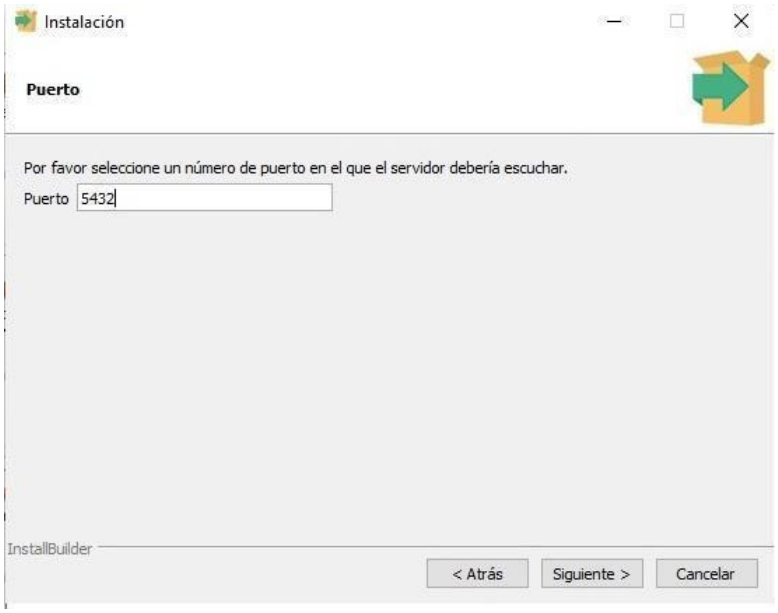
Contraseña de acceso a PostgreSQL



A continuación, se solicitará un número de puerto TCP/IP, este puerto se utilizará para establecer conexiones entre el servidor PostgreSQL y los clientes que desean conectarse a la base de datos.

Figura 33

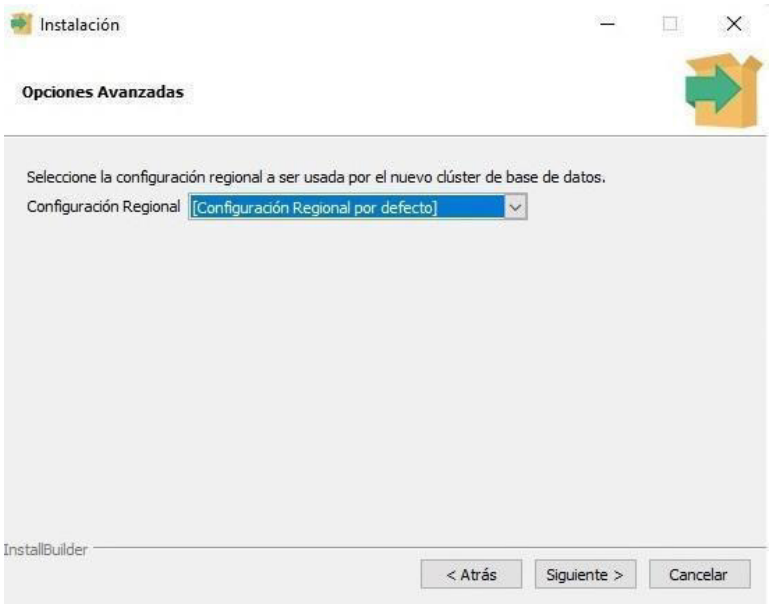
Configuración del puerto de conexión.



Luego, se seleccionará la configuración regional, seleccionar “Configuración regional por defecto”

Figura 34

Configuración regional de idioma.



En este momento se puede proceder con la instalación, una vez finalizado, aparecerá un último mensaje en el que se preguntara si se desea utilizar “Stack Builder”, esta herramienta no es necesaria, así que se puede desmarcar y finalizar la instalación

Figura 35

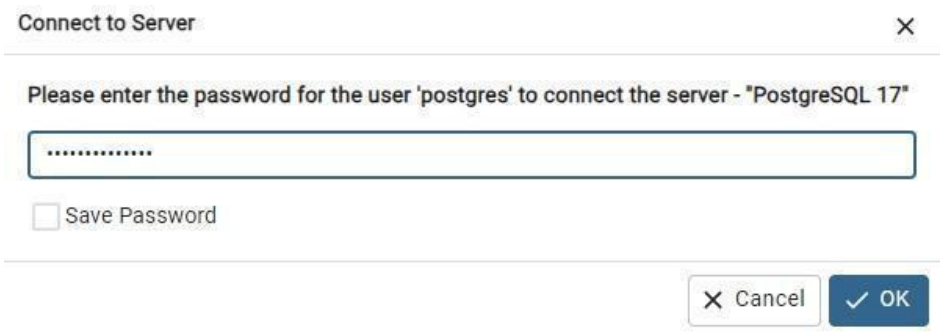
Finalización de la instalación.

El paso siguiente es crear una nueva base de datos, acceder al programa “pgAdmin 4”

Instalado recientemente e ingresar con la contraseña que se ha solicitado en el instalador

Figura 36

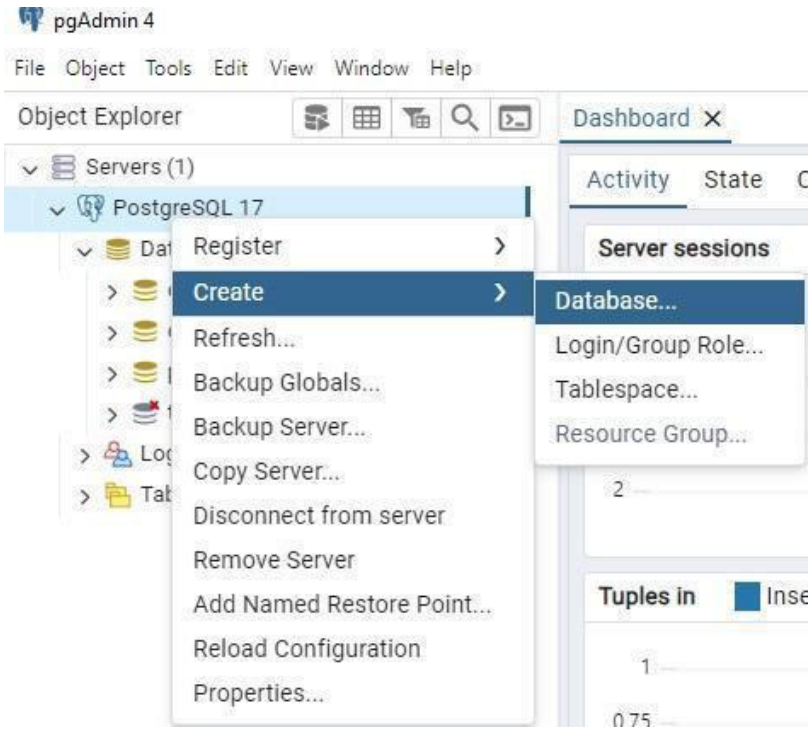
Acceso al programa pgAdmin 4



Al ingresar, aparecerá un menú desplegable en la parte superior izquierda de la pantalla. Sobre el icono de “Databases” hacer click derecho y seleccionar las opciones New -> Database

Figura 37

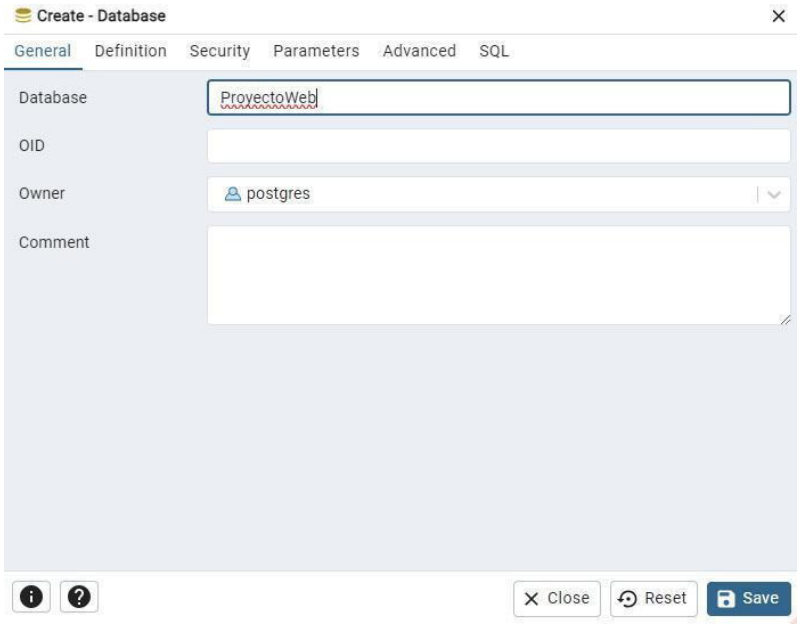
Creación de una base de datos, primera parte.



Se abrirá una ventana en la que se podrá modificar configuraciones iníciales de la base de datos a crear, por el momento no se necesita modificar nada, solo ingresar un nombre para la base de datos y dar click en el botón “Save”.

Figura 38

Creación de una base de datos, segunda parte.



La base de datos ha sido creada, no se requiere crear manualmente las tablas o escribir scripts ya que las tablas se generarán automáticamente cuando las clases Java sean mapeadas desde el entorno de desarrollo.

13.2. Instalación de Java Development Kit (JDK) versión 23

Ingresar a la página oficial de Java, en la sección de “JDK Development kit 23.0.1” seleccionar el tipo de sistema operativo que posee la computadora en la que se desarrollará la aplicación y hacer click sobre la opción de descarga de preferencia.

Ejecutar el instalador, se abrirá una ventana como la de la siguiente figura

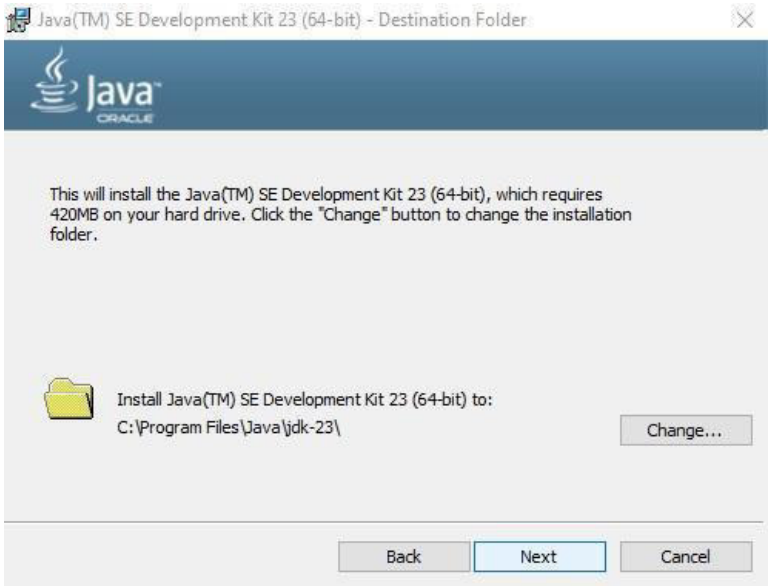
Figura 39

Instalación de JDK 23, primera parte.

Hacer click sobre el botón “Next” y seleccionar la ruta en la que se instalará el JDK.

Figura 40

Instalación de JDK 23, segunda parte.



Hacer click en el botón “Next” y esperar a que finalice la instalación.

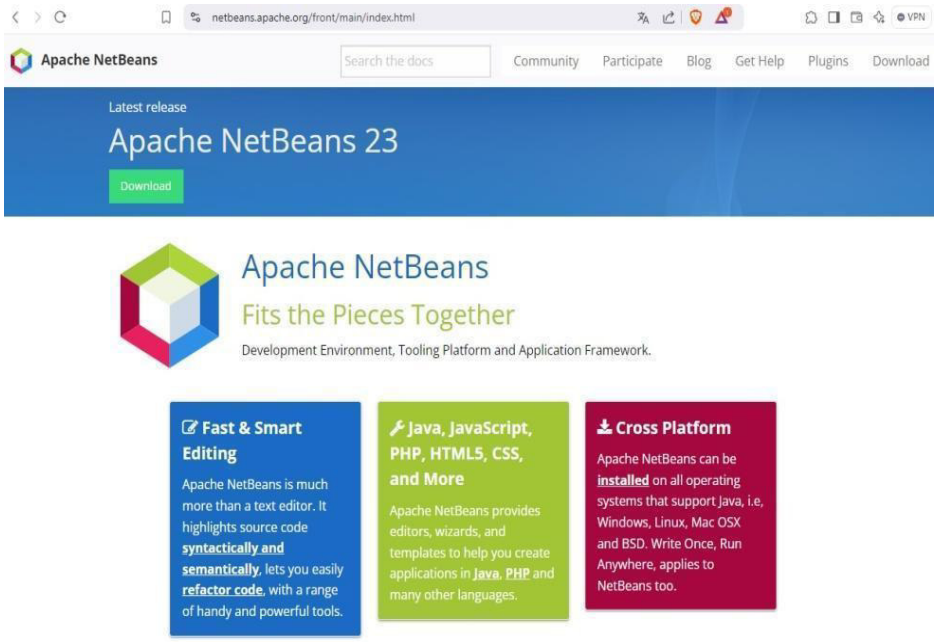
Una vez finalizada la instalación, hacer click en el botón “Finish”.

13.3. Instalación del entorno de desarrollo integrado (IDE) NetBeans 23

Ingresar a la página oficial “netbeans.apache.org” y hacer click en el botón “Download”.

Figura 42

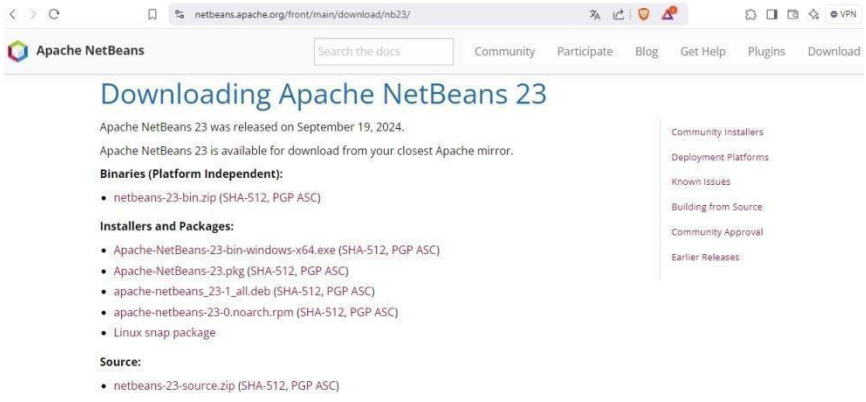
Página oficial de NetBeans



En la página siguiente, seleccionar la versión de NetBeans compatible con el sistema operativo en el que se desarrollará la aplicación.

Figura 43

Selección de versión de NetBeans

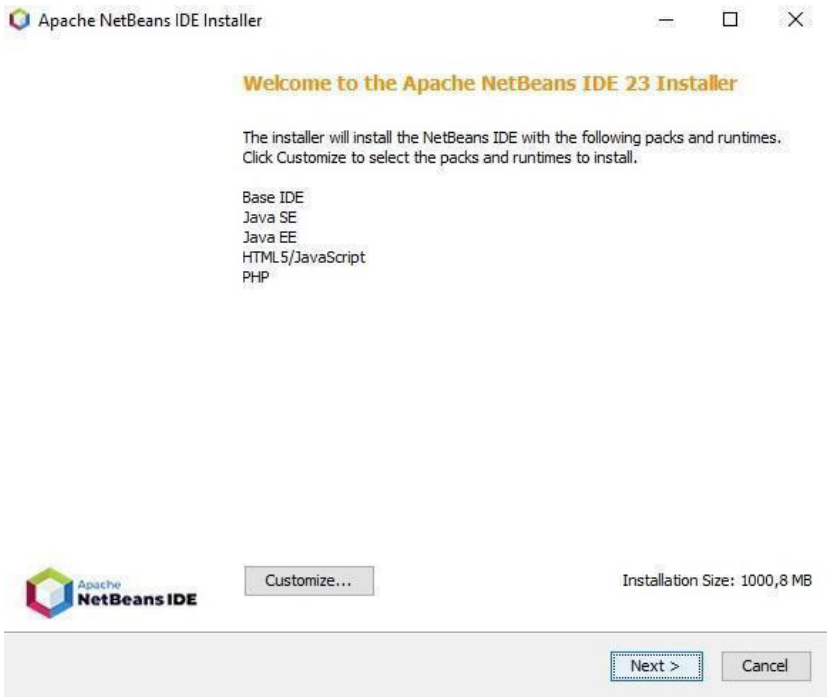


Selecciona la versión correspondiente y guarda el instalador.

Una vez terminada la descarga, ejecutar el instalador e iniciar el asistente de instalación de NetBeans

Figura 44

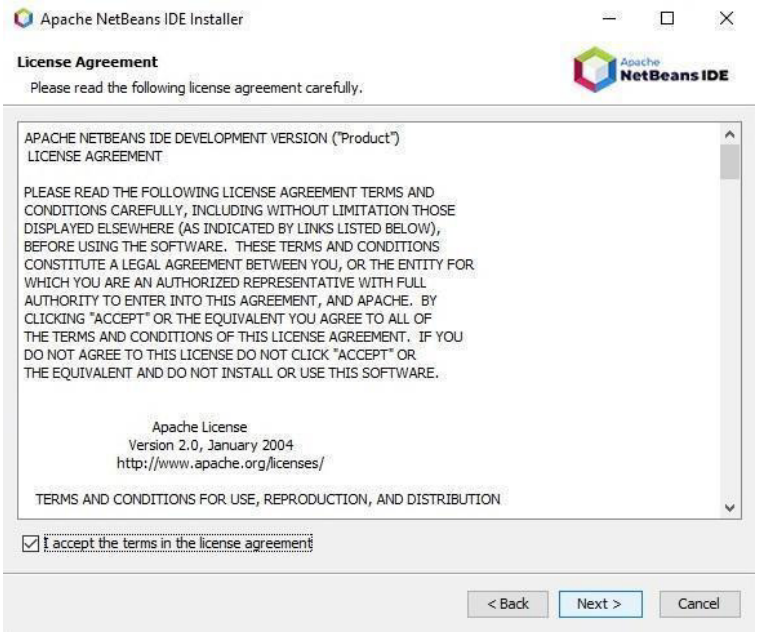
Instalación de NetBeans



En la ventana siguiente se debe aceptar los términos y condiciones de la licencia para continuar con la instalación.

Figura 45

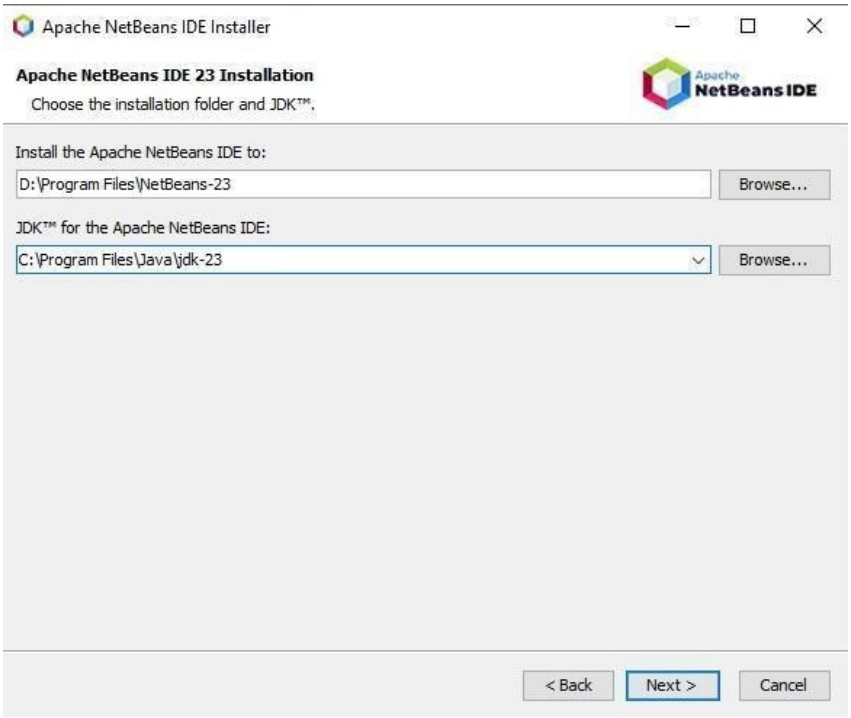
Aceptar términos y condiciones de NetBeans



A continuación, el instalador permitirá ingresar la ruta en la que se instalará NetBeans, también se debe indicar la ruta en la que se ha instalado el JDK

Figura 46

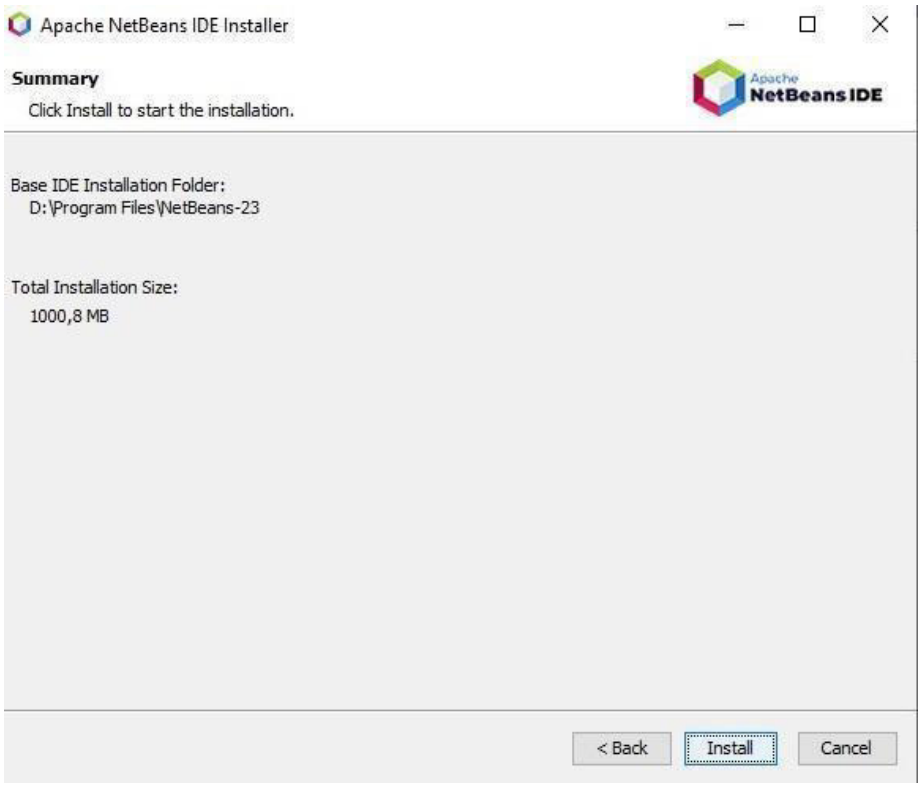
Ruta de instalación de NetBeans



En la ventana siguiente, el instalador mostrará la ruta en la que se va a instalar y el espacio necesario en el disco rígido. Si todo es correcto, se puede iniciar la instalación haciendo click en el botón “Install”

Figura 47

Confirmar la instalación de NetBeans



Luego de terminar el proceso de instalación, hacer click en el botón “Finish“.

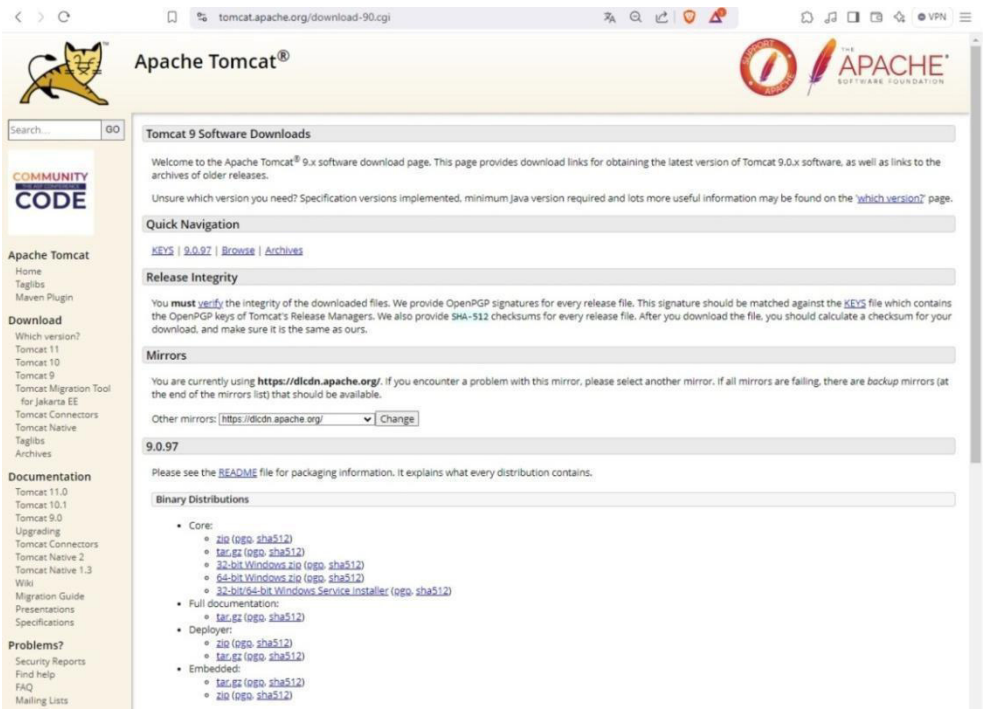
14. Creación de un nuevo proyecto en NetBeans e instalación de Apache Tomcat

El primer paso para iniciar un proyecto web es descargar Apache Tomcat.

Ingresar a la página oficial de Apache Tomcat “Tomcat.apache.org” y en la sección “Download” seleccionar “Tomcat 9”. Seleccionar la opción compatible con la distribución de sistema operativo que posee la computadora en la que se va a desarrollar la aplicación, descargar el archivo zip y descomprimirlo.

Figura 48

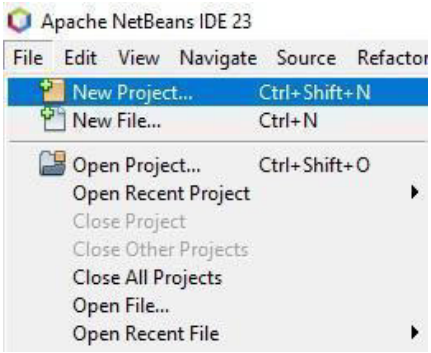
Descarga de Apache Tomcat



Para iniciar un nuevo proyecto en NetBeans, hacer click sobre la pestaña File > New proyect.

Figura 49

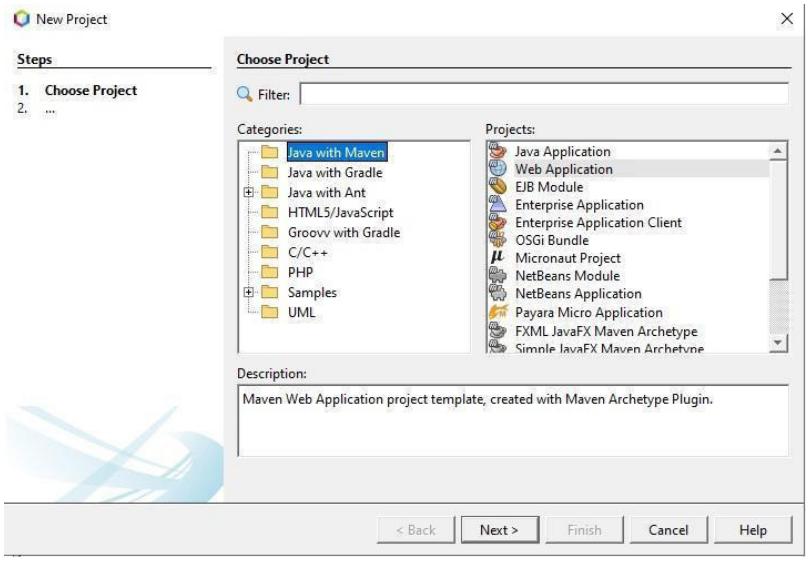
Nuevo proyecto NetBeans



En la siguiente ventana seleccionar las opciones “Java with Maven”, “Web application” y hacer click sobre el botón “Next”.

Figura 50

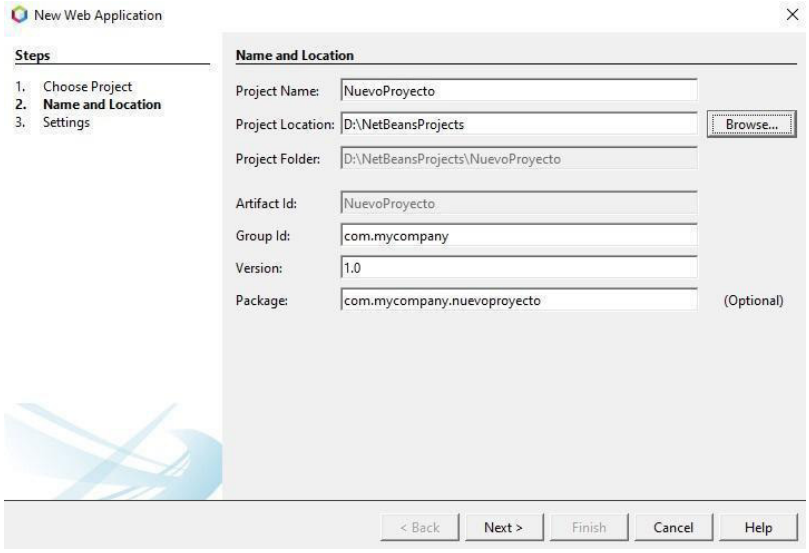
Selección del tipo de proyecto.



En la siguiente ventana se podrá nombrar al proyecto y seleccionar la ruta en la que se guardará. Una vez finalizado este paso, hacer click sobre el botón “Next”

Figura 51

Selección de nombre y ruta de almacenamiento.

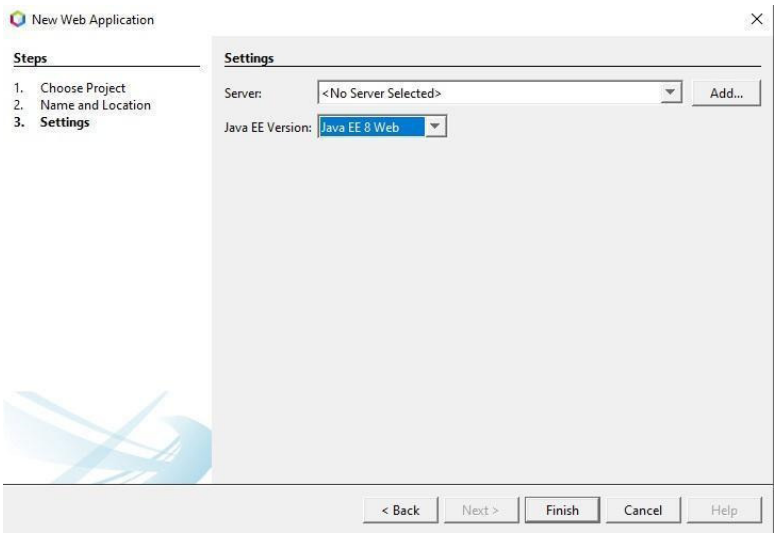


En la siguiente ventana se debe indicar qué clase de versión de java y que servidor se utilizará para este proyecto, en la sección de Java EE Version seleccionar “Java EE 8” y en la sección de Server seleccionar “Add…”.

En esta parte comienza la instalación y configuración del servidor Apache Tomcat.

Figura 52

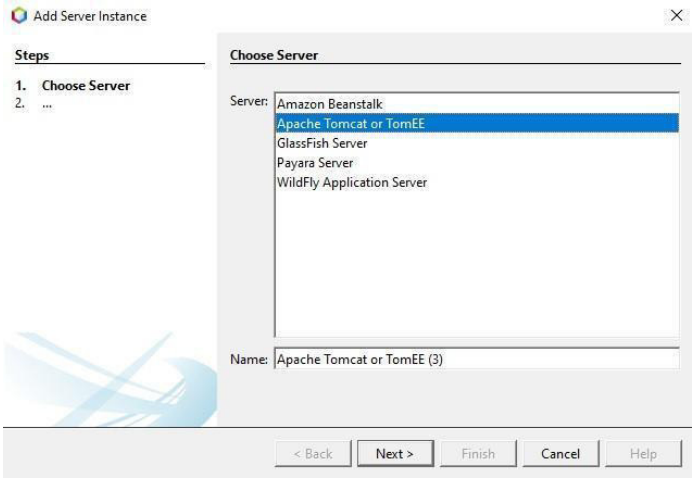
Selección de versiones Java y servidor web.



En la nueva ventana, seleccionar el tipo de servidor que se va a utilizar, en este caso Apache Tomcat.

Figura 53

Selección del servidor web.

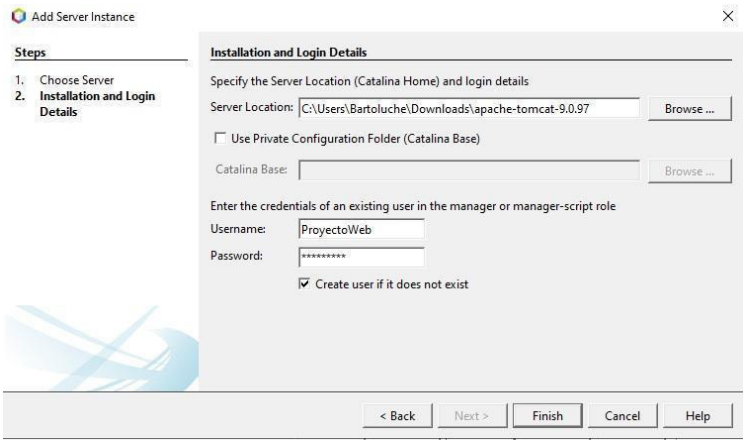


Al hacer click en “Browser…” se deberá localizar la carpeta en la que se encuentra el servidor Apache Tomcat que se descargó previamente.

Se debe ingresar un usuario y contraseña para Apache Tomcat y hacer click en “Finish”

Figura 54

Localización del servidor Apache Tomcat

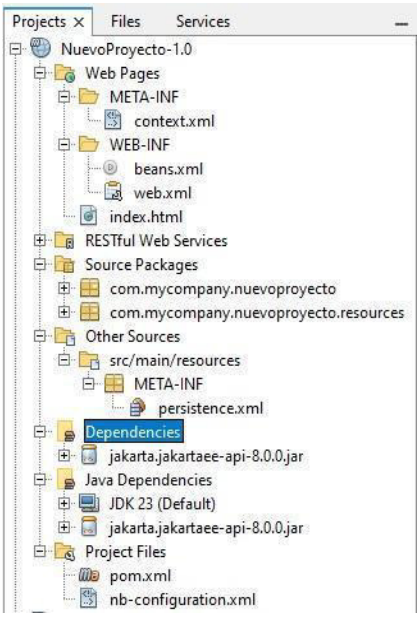


Una vez realizado estos pasos, se volverá a la ventana de la figura 52 y se terminará de crear el proyecto haciendo click en el botón “Finish”.

Al crear un proyecto nuevo, se creará por defecto una estructura de directorio con algunos archivos de configuración para el proyecto.

Figura 55

Estructura de un proyecto web.



- Carpeta “Web Pages”: Carpeta donde se guardan los recursos estáticos y dinámicos utilizados para la aplicación web. Estos recursos incluyen archivos HTML, JSP, CSS y Javascript para la interfaz de usuario
    - Carpeta “META-INF”: contiene archivos de metadatos y configuraciones para la aplicación. Estos archivos proporcionan información importante para el contenedor de servlets y otros componentes de la plataforma Java EE[1].
    - Carpeta “WEB-INF”: contiene archivos y configuraciones que no deben servir directamente al cliente, estos archivos son internos y necesarios para el funcionamiento de la aplicación web.
    - Archivo “web.xml”: define configuraciones de la aplicación web, como servlets, filtros, mapeos de url, etc.
    - Archivo “beans.xml”: contiene configuraciones utilizados en aplicaciones web que se utiliza para marcar la aplicación como una aplicación Web Beans[2].
- Carpeta “Source Packages”: Carpeta principal donde se almacenarán las clases, servlets y JpaController de la aplicación.
- Carpeta “Other sources”: almacena recursos adicionales que no son código fuente, como archivos de configuración, imágenes, sonidos, videos, etc.

[1] Java EE: plataforma de desarrollo de aplicaciones empresariales basadas en el lenguaje de programación Java. Conjunto de especificaciones y APIs que permiten desarrollar aplicaciones estables, seguras y robustas.
[2] Web Beans: clases de java que encapsulan la lógica de negocio y pueden ser utilizados para crear aplicaciones empresariales, como sistemas de gestión, aplicaciones de comercio electrónico, etc.

- Carpeta “Dependencies”: carpeta que almacena las librerías y dependencias externas necesarias para el funcionamiento del proyecto, como por ejemplo bibliotecas de Java JAR[3] y dependencias de Maven[4].
- Carpeta “Java Dependencies”: carpeta que utiliza Maven para descargar y almacenar las dependencias del proyecto.
- Carpeta “Project Files”: Carpeta que almacena archivos de configuración de Maven y configuración del proyecto.
    - Archivo “pom.xml”: archivo de configuraciones de un proyecto Maven, contiene información sobre el proyecto, como su nombre, versión, dependencias y compilación. Maven utiliza este archivo para empaquetar y desplegar el proyecto
    - Archivo “nb-configuration.xml”: archivo que gestiona configuraciones específicas del proyecto, como propiedades relacionadas con el compilador, servidor de aplicaciones, etc. Este archivo se genera automáticamente y es exclusivo de proyectos NetBeans.

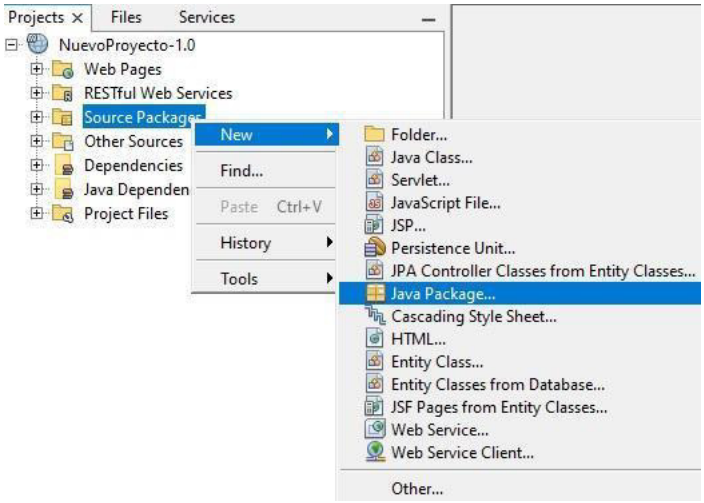
15. Codificación

Una vez creado el proyecto en NetBeans, el siguiente paso es la codificación de la aplicación. Para respetar el patrón MVC es necesario crear unos Java Packages en el proyecto, para esto hay que hacer click derecho sobre la carpeta “Source Packages” y seleccionar New -> Java Packages.

Figura 56

Creación de un Java Package, primer parte

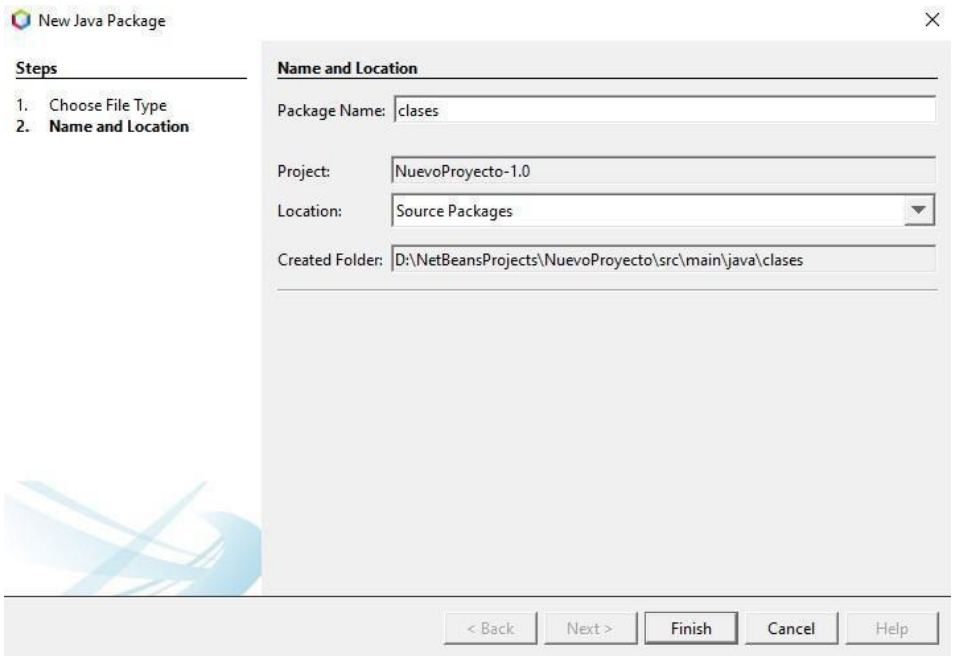
[3] JAR: archivo que combina múltiples archivos en uno solo, diseñado para almacenar y distribuir aplicaciones y bibliotecas escritas en Java.
[4] Maven: herramienta de gestión de proyectos de software diseñado para proyectos Java, simplifica y estandariza el proceso de construcción, compilación, prueba y distribución de aplicaciones



En la ventana siguiente, se debe nombrar el nuevo package.

Figura 57

Creación de un Java Package, segunda parte

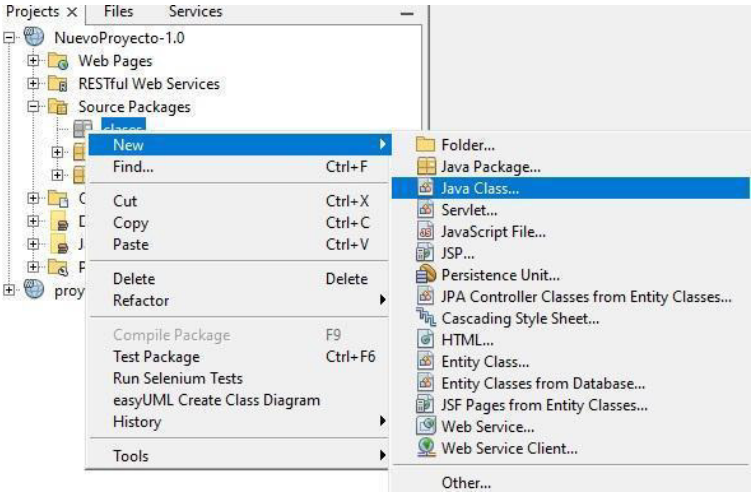


Se creará tres package, uno llamado “clases” para almacenar las clases de Java, otro llamado “Servlets” para almacenar los servlets, y otro llamado “persistencia” para almacenar los JpaController pertenecientes a cada clase.

Para crear las clases java, situar el cursor sobre el package “clases”, hacer click derecho y seleccionar New -> Java class

Figura 58

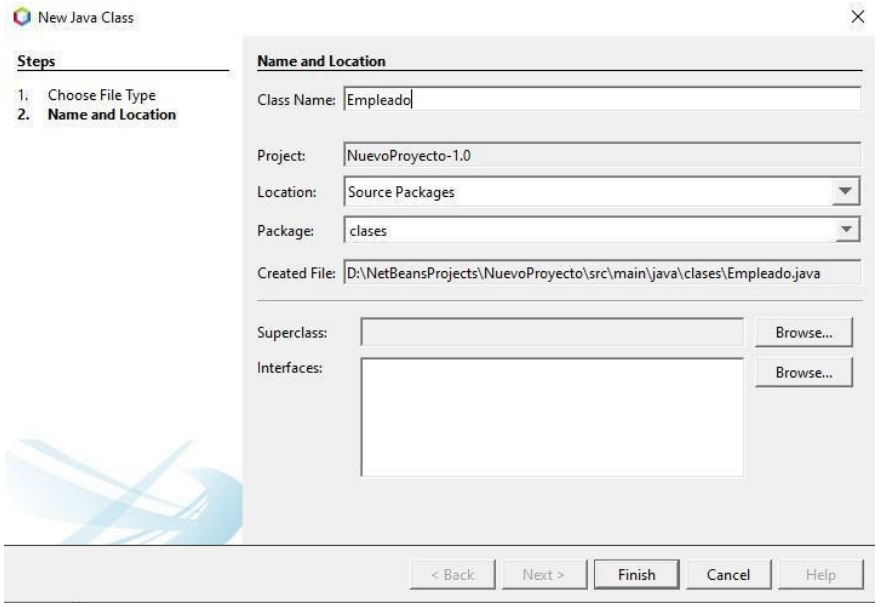
Creación de una clase Java, primera parte.



El siguiente paso es Nombrar la clase, según su representación en el diagrama de clases, por ejemplo Empleado.

Figura 59

Creación de una clase Java, segunda parte.



De esta forma se crearán todas las clases necesarias para el proyecto.

Dentro de la nueva clase Java, se codificará la clase “Empleado”, con sus atributos y métodos.

Figura 60

Clase “Empleado”.

```java
public class Empleado implements Serializable {
    private int id_empleado;
    private int legajo;
    private Jerarquia jerarquia;
    private String nombres;
    private String apellidos;
    private String cuil;
    private String calle;
    private int altura;
    private String piso;
    private String localidad;
    private String telefono;
    private String telefono_familiar;
    private byte[] foto_dni;
    private String foto_dni_base64;
    private LocalDate fecha_ingreso; // LocalDate fecha sin hora
    private int antiguedad;
    private boolean despido; // true = despido, false = empleado vigente
    private BigDecimal sueldo_base;
    private Contrato contrato;
    private EstadoEmpleado estado;
    private GrupoTrabajo grupo;
    private ArrayList<EmpleadoObra> asignaciones = new ArrayList<>();
    private ArrayList<HistorialART> historialART = new ArrayList<>();
    private ArrayList<LiquidacionSueldo> liquidaciones = new ArrayList<>();
    private ArrayList<EntregaEPP> planillaEPP = new ArrayList<>();
    private ArrayList<Asistencia> asistencias;

    public Empleado() {

    }
```

Los tipos de datos utilizados son:

- Int: almacena números entero.
- String: almacena cadenas de texto.
- byte[ ]: entero de 8 bits complemento a dos (rango de -128 a 127) usado para almacenar imágenes
- boolean: representa valores de verdadero (true) o falso (false)
- BigDecimal: almacena números decimales, usado para realizar cálculos precisos con números decimales, como por ejemplo, el sueldo.
- LocalDate: almacena fechas, sin hora.
- ArrayList<Object>: almacena listas, pueden ser listas de int, String, instancias de otros objetos, etc.
- Instancias de otras clases: por ejemplo, Jerarquía, Contrato; Estado Empleado; GrupoTrabajo, etc.

Estos tipos de datos serán utilizados para la codificación de las demás clases.

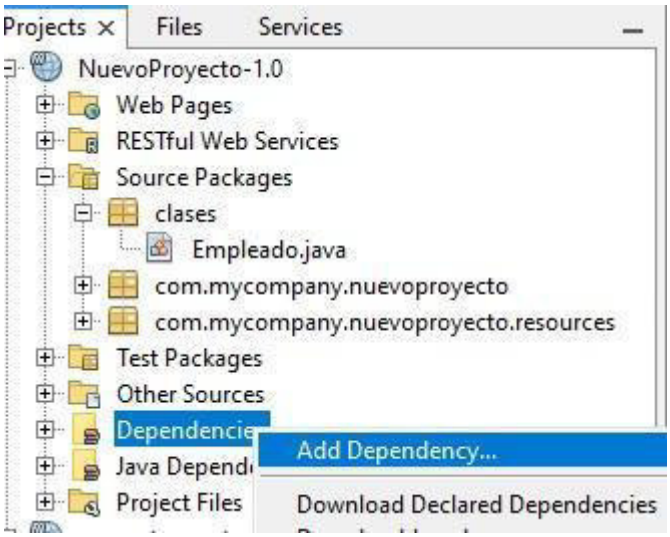
15.1. Conexión con la base de datos

El primer paso para conectar la aplicación a una base de datos es descargar el driver que permitirá realizar dicha conexión. Esta descarga se puede realizar mediante Maven.

En el proyecto, situar el cursor sobre la carpeta “Dependencies” y seleccionar “Add dependency”

Figura 61

Agregar dependencias, primera parte.

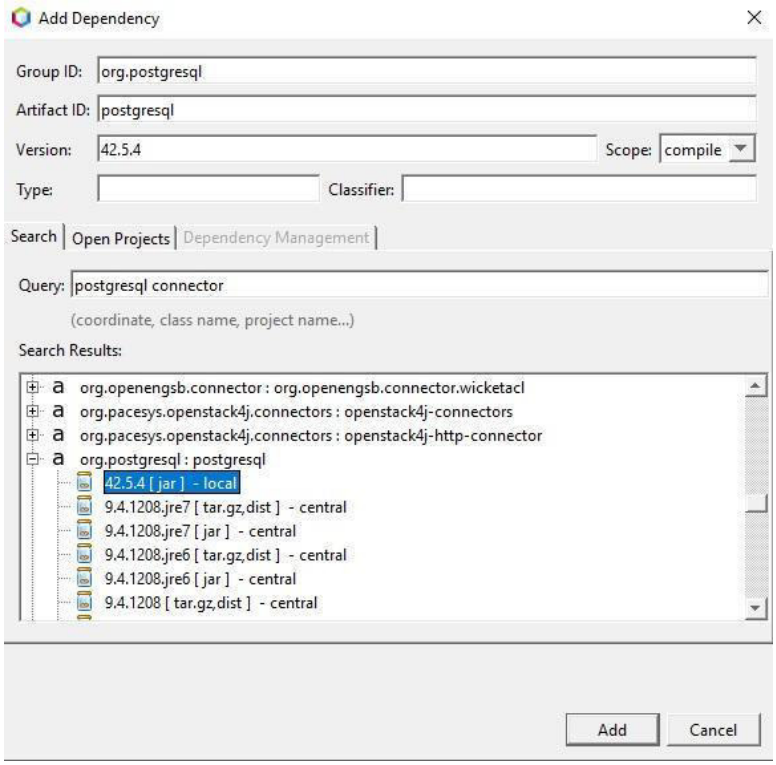


En el campo “query” del buscador de Maven, escribir “postgresql connector”, Maven buscará en su repositorio los conectores disponibles para PostgreSQL.

Entre los resultados, buscar la opción “org.postgresql: postgresql” y seleccionar el conector 42.5.4 y hacer clic sobre el botón “Add”

Figura 62

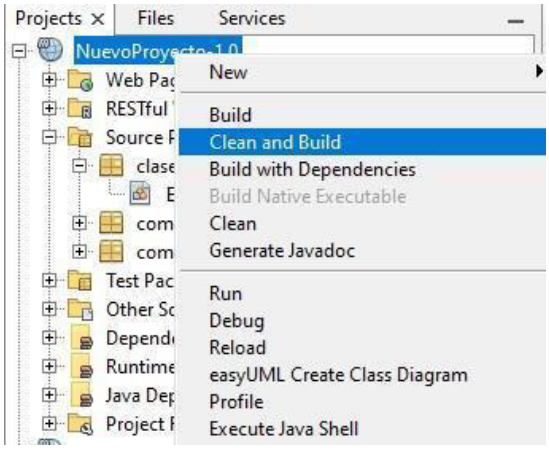
Agregar dependencias, segunda parte.



Cada vez que se agregue una dependencia se debe actualizar el proyecto, para esto, hacer click derecho sobre la raíz del proyecto y seleccionar “Clean and build”

Figura 63

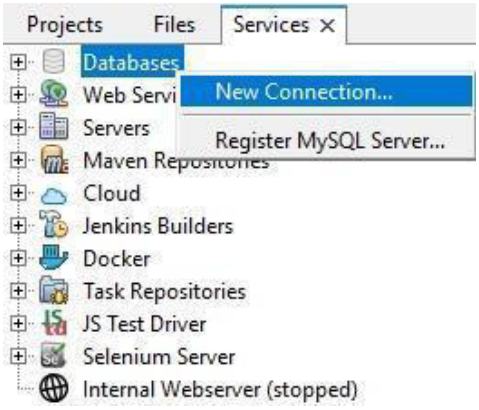
Agregar dependencias, tercera parte.



Para realizar una nueva conexión, seleccionar la pestaña “Services” en la parte superior izquierda del proyecto, click derecho sobre la opción “Databases” y seleccionar “New Connection”

Figura 64

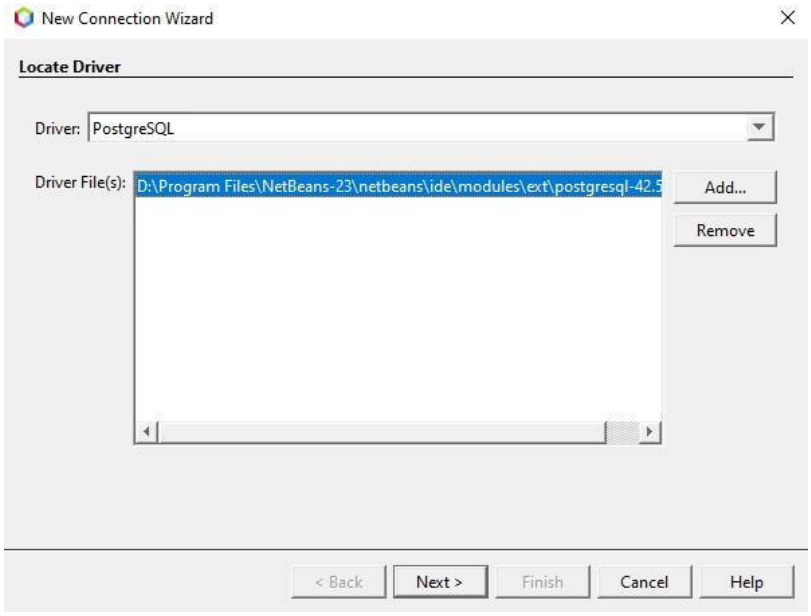
Conexión a la base de datos, primera parte



Seleccionar la opción “PostgreSQL” y seleccionar el conector que se ha descargado anteriormente con Maven

Figura 65

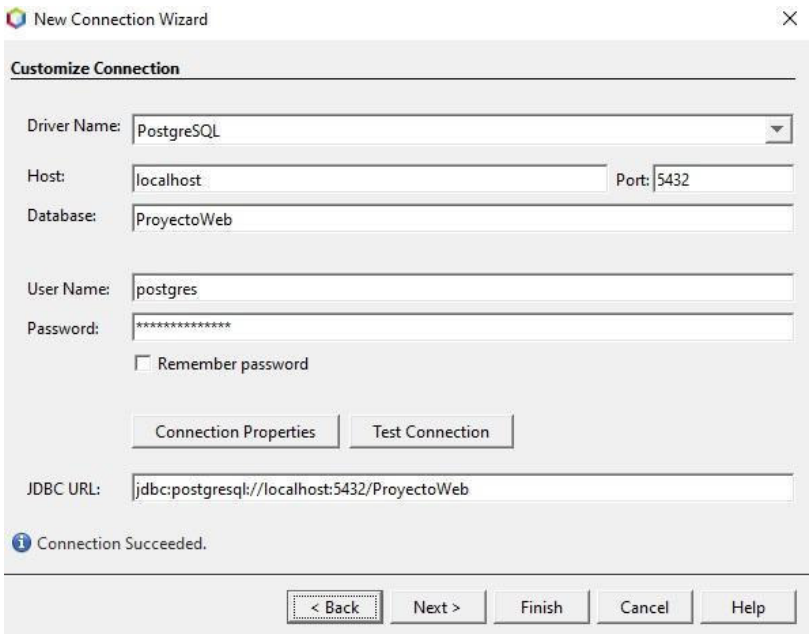
Conexión a la base de datos, segunda parte.



En la ventana siguiente se debe ingresar el nombre de la base de datos creada anteriormente “ProyectoWeb”, y la contraseña utilizada en la instalación de PostgreSQL. Se puede verificar la conexión con el botón “Test connection”

Figura 66

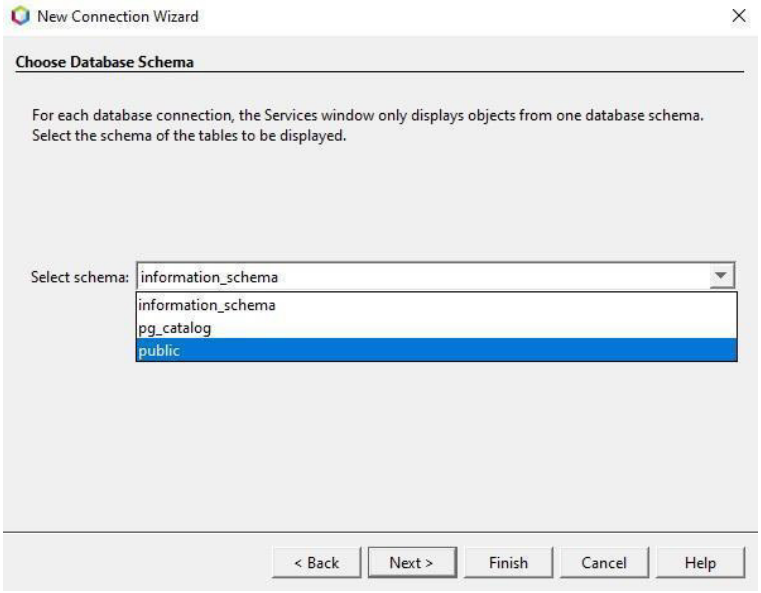
Conexión a la base de datos, tercera parte.



Seleccionar el esquema en que las tablas serán creadas en la base de datos y finalizar el asistente de conexiones.

Figura 67

Conexión a la base de datos, cuarta parte



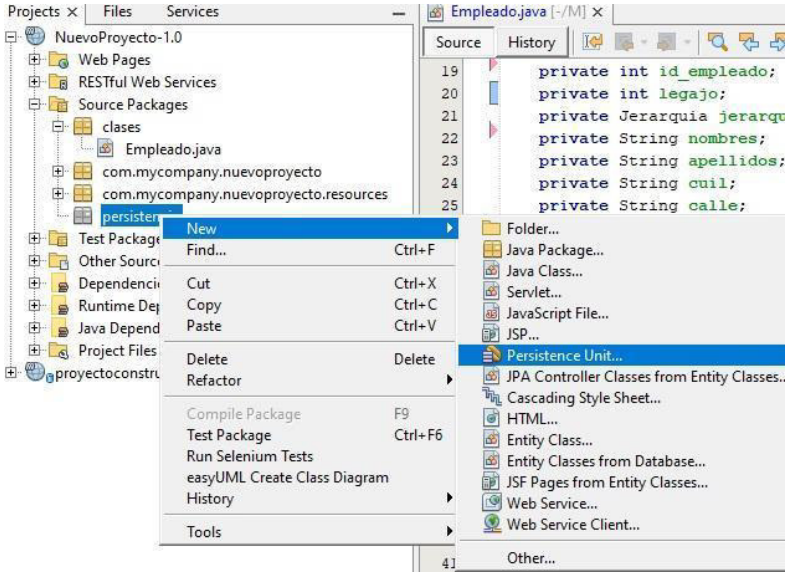
15.2. Creación de unidad de persistencia

Una unidad de persistencia (Persistence Unit) define los detalles y cómo se conectará la aplicación a una base de datos, las clases que van a persistir, y como se relacionan entre sí.

Para crear una nueva unidad de persistencia, hacer click derecho sobre el package “persistencia” creado anteriormente y seleccionar New -> Persistence unit.

Figura 68

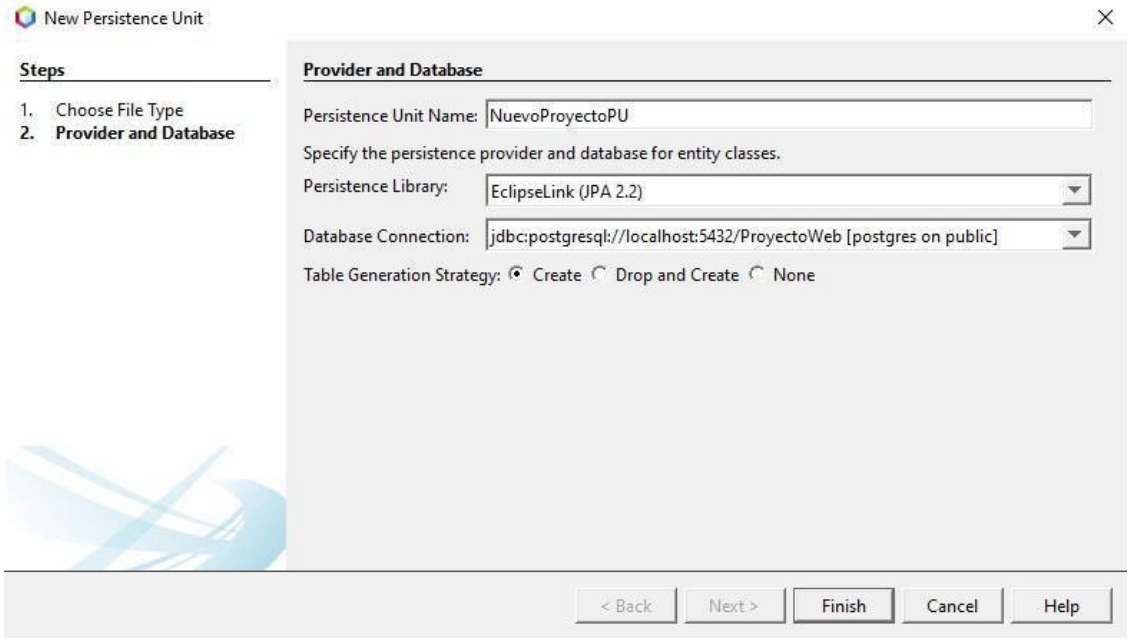
Creación de la unidad de persistencia, primera parte.



En la nueva ventana, se debe nombrar la unidad de persistencia, seleccionar la librería de persistencia (En este caso EclipseLink) y seleccionar la conexión a la base de datos creada anteriormente.

Figura 69

Creación de la unidad de persistencia, segunda parte.



15.3. Mapeo de clases a la base de datos con EclipseLink

Con EclipseLink se pueden mapear clases de un proyecto a una base de datos en pocos pasos y sin necesidad de escribir scripts en SQL.

Para mapear la clase “Empleado” creada anteriormente se usaron las siguientes notaciones:

- @Entity: se utiliza para marcar las clases de java, cada clase marcada de esta manera se transformará en una entidad en la base de datos
- @Id: se utiliza para marcar un atributo de la clase, este atributo será la clave primaria de la entidad en la base de datos
- @GeneratedValue(strategy=GenerationType.SEQUENCE): utilizado junto a la anotación @Id, se asigna un valor de forma secuencial al atributo al almacenarse en la base de datos
- @Transient: utilizado para evitar que un atributo sea mapeado a la base de datos
- @Column: se utiliza para el resto de los atributos, su utilización es opcional ya que en la mayoría de los casos EclipseLink mapea el tipo de dato de la clase con su correspondiente tipo de dato en la base de datos.
- @Column (unique = true): utilizado para verificar que el valor de un atributo sea único y no se repita con otras entidades.
- @Column (precisión=14, scale=7): utilizado para mapear el atributo “sueldo_base”, convierte el BigDecimal de la clase en “numeric” en la entidad de la base de datos, precisión y scale indican que se almacenará un número con hasta 7 dígitos enteros y hasta 7 dígitos decimales.
- @OneToOne: utilizado para mapear la relación uno a uno entre dos clases.
- CascadeType:utilizado estrategia de actualización de datos para clases relacionadas, estas estrategias pueden ser:
    - cascadeType.MERGE: propagación de la actualización de datos a las demás entidades relacionadas.
    - CascadeType.REMOVE: propaga la eliminación de datos a las demás entidades relacionadas.
    - CascadeType.PERSIST: propaga la persistencia de datos a las demás entidades relacionadas
    - CascadeType.ALL: aplica todas las estrategias anteriores

- @JoinColumn: utilizado para asociar el atributo identificador de una clase contenedora con el atributo identificador de una clase contenida, esta anotación cuenta con dos parámetros
    - name=”empleado_id_jerarquia”: al ser mapeada la clase contenedora a una entidad, se crea una columna extra que almacenará el id de la clase contenida.
    - referencedColumnName=”id_jerarquia”: especifica la columna correspondiente al id de la clase contenida
- @OneToMany: utilizado para mapear una relación de uno a muchos, esta anotación se puede utilizar con un parámetro.
    - mappedBy=”empleadoLiquidacion”: esta clase es una clase contenida, una instancia de la clase “Empleado” está relacionada con muchas instancias de la clase “LiquidacionSueldo”
- @ManyToOne: Utilizado para mapear una relación de muchos a muchos, esta relación no se puede mapear directamente, se necesita una clase intermedia que referencie a las dos clases principales. Por ejemplo, la clase “EmpleadoObra” tiene una relación muchos a uno con la clase “Empleado” y la clase “Obra”, la clase “EmpleadoObra” Contiene instancias de las clases principales y las mapea con la anotación @JoinColumn. Por ejemplo @JoinColumn (name=”id_empleado”)

Figura 70

Mapeo de Clase Empleado y atributos.

```java
@Entity
public class Empleado implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id_empleado;

    @Column(unique = true)
    private int legajo; // el legajo sera ingresado manualmente por el usua

    private String nombres;
    private String apellidos;
    private String cuil;
    private String calle;
    private int altura;
    private String piso;
    private String localidad;
    private String telefono;
    private String telefono_familiar;
    private LocalDate fecha_ingreso; // LocalDate fecha sin hora
    private int antiguedad;
    private boolean despido; // true = despido, false = empleado vigente
    private byte[] foto_dni;

    @Transient
    private String foto_dni_base64;

    @Column(precision = 14, scale = 7)
    private BigDecimal sueldo_base;
```

Figura 71

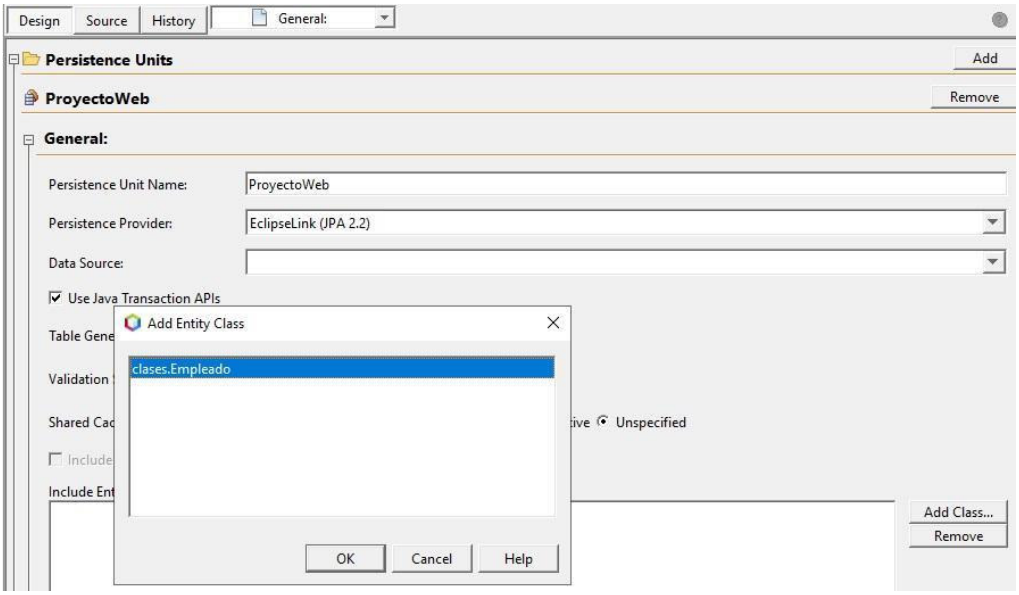
Mapeo de Clase Empleado y relaciones con otras clases

```
@OneToOne(cascade = {CascadeType.MERGE, CascadeType.REFRESH})
@JoinColumn(name = "id_jerarquia", referencedColumnName = "id_jerarquia")
private Jerarquia jerarquia;

@OneToOne(cascade = {CascadeType.MERGE, CascadeType.REFRESH})
@JoinColumn(name = "id_contrato", referencedColumnName = "id_contrato")
private Contrato contrato;

@OneToOne(cascade = {CascadeType.MERGE, CascadeType.REFRESH})
@JoinColumn(name = "id_estado", referencedColumnName = "id_estado")
private EstadoEmpleado estado;

@ManyToOne
@JoinColumn(name = "id_grupo", nullable = true)
private GrupoTrabajo grupo;

@OneToMany(mappedBy = "empleadoObra", cascade = {CascadeType.MERGE, CascadeType.REFRESH})
private ArrayList<EmpleadoObra> asignaciones = new ArrayList<>();

@OneToMany(mappedBy = "empleadoART", cascade = {CascadeType.MERGE, CascadeType.REFRESH})
private ArrayList<HistorialART> historialART = new ArrayList<>();

@OneToMany(mappedBy = "empleadoLiquidacion", cascade = {CascadeType.MERGE, CascadeType.REFRESH})
private ArrayList<LiquidacionSueldo> liquidaciones = new ArrayList<>();

@OneToMany(mappedBy = "empleadoEPP", cascade = {CascadeType.MERGE, CascadeType.REFRESH})
private ArrayList<EntregaEPP> planillaEPP = new ArrayList<>();

@OneToMany(mappedBy = "empleadoAsistencia", cascade = {CascadeType.MERGE, CascadeType.REFRESH})
private ArrayList<Asistencia> asistencias;
```

Al finalizar el mapeo de todas las clases, se debe agregar las clases a la unidad de persistencia creada anteriormente, haciendo click en el botón “Add class” la unidad de persistencia detecta automáticamente las clases mapeadas con EclipseLink.

Figura 72

Agregar clases a la unidad de persistencia.



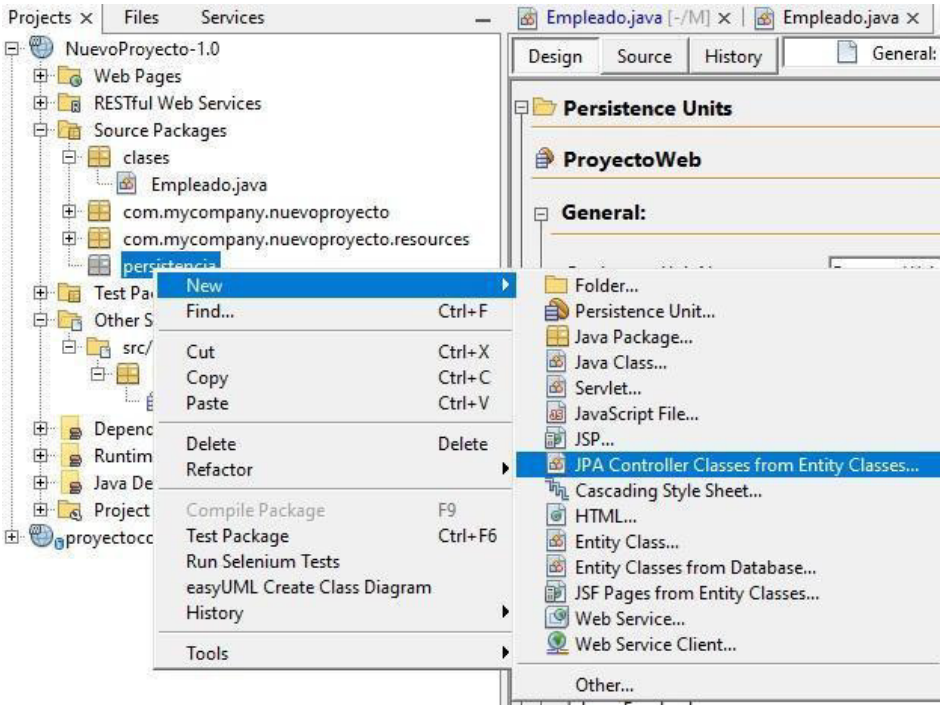
15.4. Creación de los JpaController

Al agregar las clases a la unidad de persistencia, el siguiente paso es crear los JpaController pertenecientes a cada clase. Las clases “JpaController” utilizan especificaciones Java Persistence Api para interactuar con la base de datos.

Sobre el package “persistencias” hacer click derecho y seleccionar New -> Jpa Controller.

Figura 73

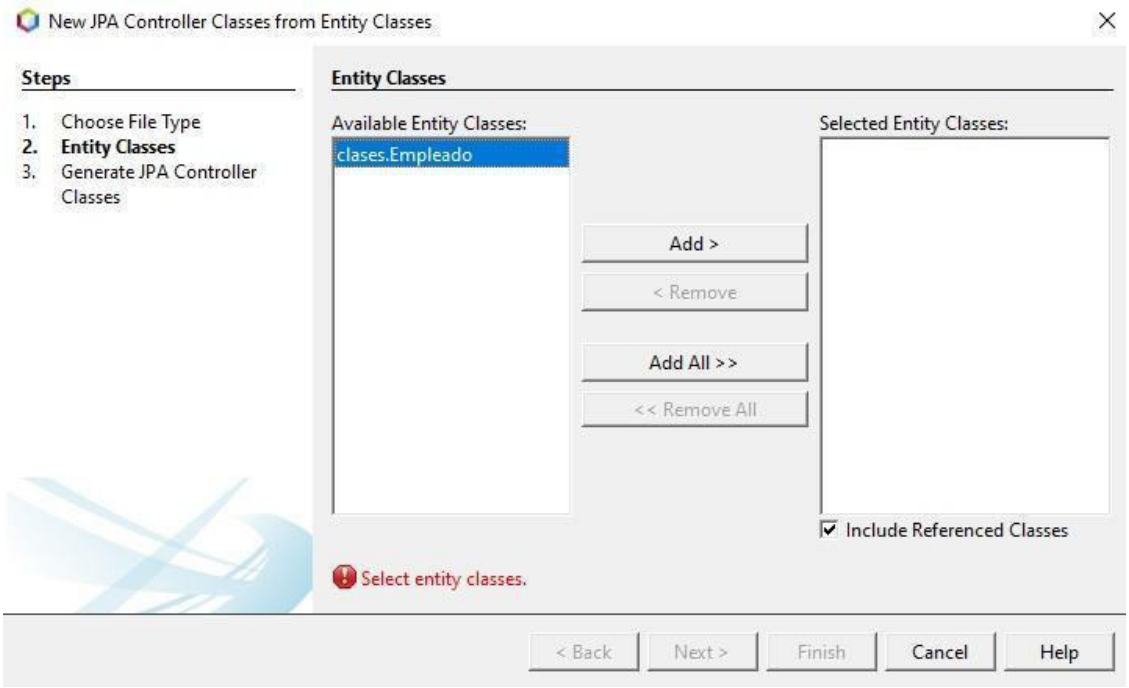
Creación de los Jpa Controller, primera parte.



En la siguiente ventana, del lado izquierdo aparecerá una lista con todas las clases que fueron agregadas a la unidad de persistencia. Seleccionar las clases a las que se quiera crear sus Jpa Controller y pasarlos a la parte derecha con el botón “Add”

Figura 74

Creación de los Jpa Controller, segunda parte.



Por último, se especifica en qué carpeta se guardarán los Jpa Controller, para mantener una estructura de proyecto ordenada, se guardarán en el package “persistencia”.

Figura 75

Creación de los Jpa Controller, tercera parte.

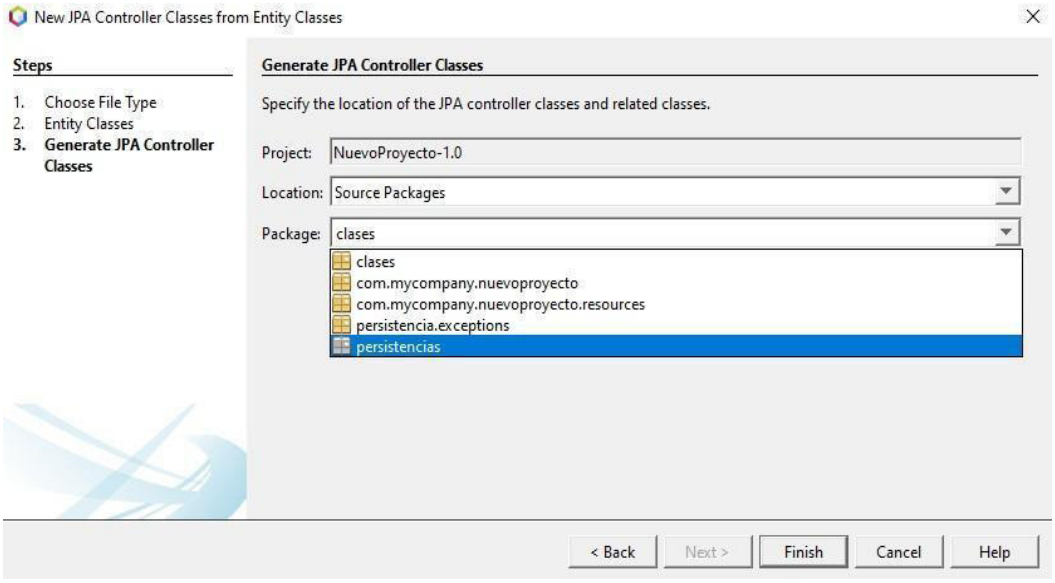


Figura 76

Contenido de "EmpleadoJpaController"

```
public class EmpleadoJpaController{
    private EntityManagerFactory emf = null;

    public EmpleadoJpaController(EntityManagerFactory emf) {
        this.emf = emf;
    }

    public EmpleadoJpaController(){
        emf = Persistence.createEntityManagerFactory("ProyectoWebPU");
    }

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    // Crear Empleado
    public void create(Empleado empleado) {...12 lines }

    // Editar Empleado
    public void edit(Empleado empleado) throws Exception {...18 lines }

    // Eliminar Empleado
    public void destroy(int id) throws EntityNotFoundException {...19 lines }

    // Buscar un empleado por id
    public Empleado findEmpleado(int id) {...8 lines }

    // Buscar la lista de todos los empleados
    public List<Empleado> findEmpleadoEntities(){...3 lines }

    private List<Empleado> findEmpleadoEntities(int maxResults, int firstResult){...3 lines }

    private List<Empleado> findEmpleadoEntities(boolean all, int maxResults, int firstResult)
```

La creación de los Jpa Controller autogenera un conjunto de métodos que realizan consultas a la base de datos sin necesidad de escribir consultas en lenguaje SQL. Estos métodos son:

- create (Empleado empleado): recibe una instancia de la clase "Empleado" y guarda sus datos en la base de datos.
- edit (Empleado empleado): recibe una instancia de la clase "Empleado", y si su atributo identificador coincide con un registro en la base de datos, actualiza los datos.
- destroy (int id): recibe un número como parámetro, y si ese número coincide con el atributo identificador de un registro en la base de datos, elimina ese registro.
- findEmpleado (int id): recibe un número como parámetro, y si ese número coincide con el atributo identificador de un registro en la base de datos, devuelve una instancia de la clase Empleado, en caso contrario, devuelve el valor "null".

- findEmpleadoEntities (): devuelve una lista con todos los empleados existentes.

El proceso de creación de los Jpa Controller es similar para todas las clases del proyecto.

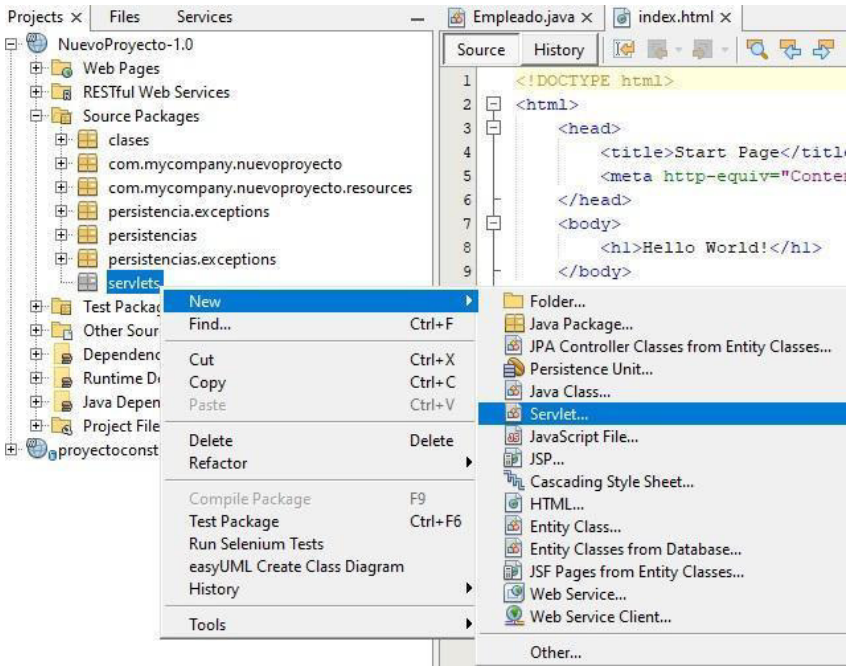
Al desplegar nuestra aplicación por primera vez, se abrirá una ventana del navegador web con un mensaje “Hello world!”, al mismo tiempo, el archivo “persistence.xml” habrá creado las entidades en la base de datos utilizando las clases mapeadas como modelo.

15.5. Creación de Servlet

Los servlets son los encargados de actuar como intermediarios entre la interfaz de usuario y la base de datos, para crear un nuevo Servlet, se debe hacer click derecho sobre el package “servlets” creado anteriormente y seleccionar New -> Servlet.

Figura 77

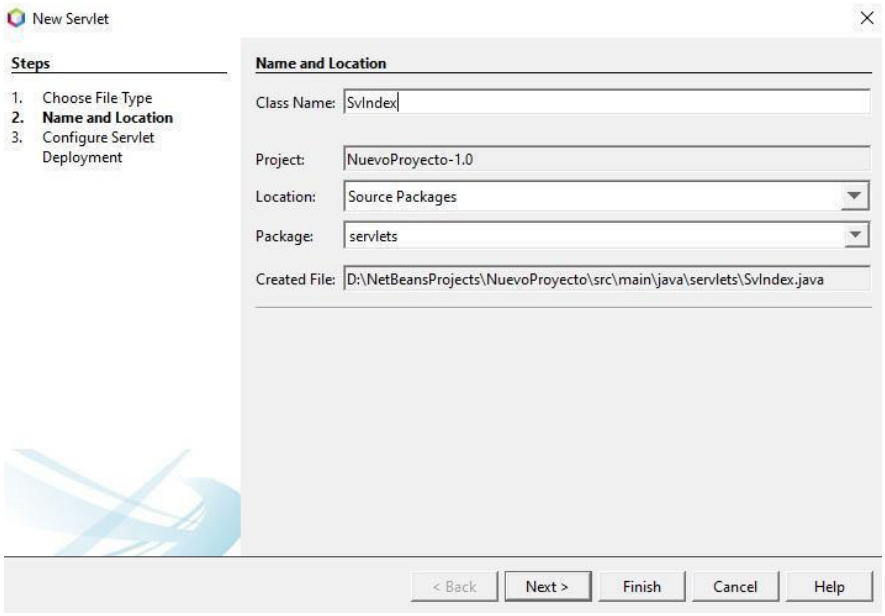
Creación de un Servlet, primera parte.



En la ventana siguiente, se debe ingresar el nombre del Servlet, se recomienda que el Servlet lleve como nombre el nombre de la página a la que va a atender sus solicitudes.

Figura 78

Creación de un Servlet, segunda parte.



Al crear un nuevo Servlet, este autogenera dos métodos vacios, doGet () y doPost (). Estos métodos vacios servirán para atender las solicitudes de las páginas y devolver respuestas.

Figura 79

Creación de Servlet, tercera parte.

```
package servlets;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "SvIndex", urlPatterns = {"/SvIndex"})
public class SvIndex extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

15.6. Creación de páginas JSP

El siguiente paso es la creación de páginas, con la utilización de HTML5 y CSS3 se ha modificado la estructura de la página autogenerada “index.jsp” para crear una ventana de log in de la siguiente forma

Figura 80

Página de ingreso al sistema

La página cuenta con validadores de campos, verifica a través de JQuery que ningún campo esté vacío y que ambos campos tengan un mínimo de 8 caracteres, también verifica que solo se ingresen caracteres alfabéticos o numéricos mediante expresiones regulares.

En caso de que los campos cumplan con los requisitos, se arma una solicitud AJAX y se envía el usuario y la contraseña ingresados al Servlet.

La contraseña ingresada es previamente encriptado con el algoritmo SHA256.

Figura 81

Solicitud AJAX

```
function login(){
    $('#ingresar').click(function () {
        var usuario = $('#usuario').val();
        var clave = $('#pass').val();
        var verificar = verificarLongitud(); //funcion que verifica la longitud de los campos
        var claveEncriptada = CryptoJS.SHA256(clave).toString(); // encriptar clave antes de enviar
        if(verificar === true){
            console.log('enviar datos al servidor');
            $.ajax({
                url: 'SvIndex', // sevlet al que envia la solicitud
                type: 'POST',
                //contentType: 'application/json; charset=utf-8', // especifico que es json
                data: {usuario: usuario, clave: claveEncriptada},
                dataType: 'json', // configurar el tipo de mensaje a enviar y recibir
                success: function (response) {
                    if(response.status === 'success'){
                        if(response.autorizado === true){
                            window.location.href = response.redirectUrl;
                        }
                        else{
                            console.log("denegado");
                            console.log(response);
                            window.location.href = "/proyectoconstruccion/vistas/noaprobado.jsp";
                        }
                    }
                    else{
                        $('#mensajeIncorrecto').empty();
                        $('#mensajeIncorrecto').text('El usuario y/o clave son incorrectos');
                        $('#mensajeIncorrecto').css('color', 'red');
                        console.log('usuario y/o clave incorrectos');
                        console.log(response);
                    }
                },
                error: function (xhr, status, error) {
                    console.error("Error:", error);
                    console.error("Status:", status);
                }
            });
        }
```

Figura 82

Petición AJAX y respuesta JSON del Servlet

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //capturo el usuario y clave ingresados
    String usuarioIngresado = request.getParameter("usuario");
    String claveIngresada = request.getParameter("clave");
    // consulto la lista de usuarios
    List<Usuario> listaUsuarios;
    listaUsuarios = controlador.buscarListaUsuarios();
    // armo un map para guardar informacion y convertirlo en json
    Map <String, Object> armarJson = new HashMap<>();
    // inicio el map con valores para usuario no encontrado
    armarJson.put("status", "error");
    armarJson.put("message", "Usuario y/o clave incorrectos");

    for(Usuario usuario : listaUsuarios){
        // variable que almacenara el resultado de la clave ingresada por el usuario
        // mas el salt de la clase "Usuario"
        String claveEncriptada = null;
        String saltGuardado = usuario.getSalt();
        try {
            // utilizo la clave ingresada y el salt del usuario iterado en este momento, para generar
            // la clave encriptada y luego comparar esta variable "claveEncriptada"
            // con la clave guardada de este usuario
            claveEncriptada = encriptarClave(claveIngresada, saltGuardado);
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(SvIndex.class.getName()).log(Level.SEVERE, null, ex);
        }
        if(usuario.usuarioExiste(usuarioIngresado, claveEncriptada)){
                // elimino los valores del map para usuario no encontrado
                armarJson.remove("status");
                armarJson.remove("message");
                // agrego valores para usuario encontrado
                armarJson.put("status", "success");
                armarJson.put("rol", usuario.getRol().getDescripcion());
                armarJson.put("autorizado", usuario.isAprobado());
            if(usuario.isAprobado()){
                HttpSession sesion = request.getSession();
                sesion.setAttribute("usuario", usuario);
                String rolUsuario = usuario.getRol().getDescripcion();
                sesion.setAttribute("rolUsuario", rolUsuario);
                String redirectUrl = obtenerUrl(rolUsuario);
                armarJson.put("redirectUrl", redirectUrl);
```

Cuando los datos llegan al Servlet, este realiza una consulta a la base de datos, obtiene una lista de usuarios y compara los campos recibidos con los usuarios y contraseñas de cada usuario, en caso de encontrar una coincidencia, verifica si el usuario está autorizado para ingresar al sistema, en caso de estar autorizado el Servlet arma una sesión de usuario y una respuesta en formato JSON, la envía a la página, otorgándole acceso al sistema. La solicitud JSON de usuario autorizado contiene una url que redirecciona al usuario a su correspondiente pagina dependiendo del rol de usuario, en caso de no contar con autorización, el JSON envía una url que lo redirecciona a una página “noautorizado.jsp”

En caso de no encontrar usuario, arma una respuesta de error y la envía a la página, esta página se encarga de mostrar el error al usuario como se puede observar en la figura 79.

Figura 83

Métodos de encriptación y obtención de url según rol de usuario

```
// utilizo la clave que recibo de index.jsp, el salt generado y hago una nueva encriptacion
private String encriptarClave(String claveRecibida, String salt) throws NoSuchAlgorithmException {
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    String saltedPassword = salt + claveRecibida;
    byte[] hashBytes = md.digest(saltedPassword.getBytes(StandardCharsets.UTF_8));
    return Base64.getEncoder().encodeToString(hashBytes);
}

private String obtenerUrl(String rolUsuario) {
    String url = null;

    if(null != rolUsuario)switch (rolUsuario) {
        case "Admin sistemas":
            url = "/proyectoconstruccion/vistas/sistemas/home.jsp";
            break;
        case "Administrativo":
            url = "/proyectoconstruccion/vistas/administrativo/home.jsp";
            break;
        case "Ayudante":
            url = "/proyectoconstruccion/vistas/ayudante/home.jsp";
            break;
        case "Contador":
            url = "/proyectoconstruccion/vistas/contador/home.jsp";
            break;
        default:
            break;
    }
    return url;
}
```

El método obtenerUrl () devuelve una url según el rol de usuario que intenta ingresar al sistema.

El método encriptarClave () recibe como parámetro la clave recibida de la página, y una cadena de texto llamada “salt”, concatena ambos String y lo encripta nuevamente en SHA256, si el resultado de esta operación es igual a la clave almacenada del usuario, significa que es la clave de usuario correcta.

La página principal también cuenta con una página de registro de nuevos usuarios. El empleado que quiera crear un usuario para ingresar al sistema deberá ingresar su número de legajo, esta implementación es para evitar que personas que no sean empleados no puedan acceder al sistema.

Se envía una solicitud AJAX a un Servlet encargado de verificar que el legajo ingresado no tenga un usuario asociado, en caso de no contar con usuario, envía una respuesta a la página de registro con información básica del empleado para verificar si sus datos son correctos. Luego el empleado puede registrar su nuevo usuario.

La pagina solicitará un nombre de usuario, contraseña, repetir contraseña y una pestaña desplegable para seleccionar el rol de usuario que tendrá en el sistema. Estos campos están verificados por JQuery y expresiones regulares, en caso que los campos cuenten con 8 caracteres o más, estos datos son enviados al Servlet que se encarga del registro de nuevos usuarios. La clave es encriptada con el algoritmo SHA256 antes de ser enviada al sevlet

Figura 84

Página Registrar.jsp

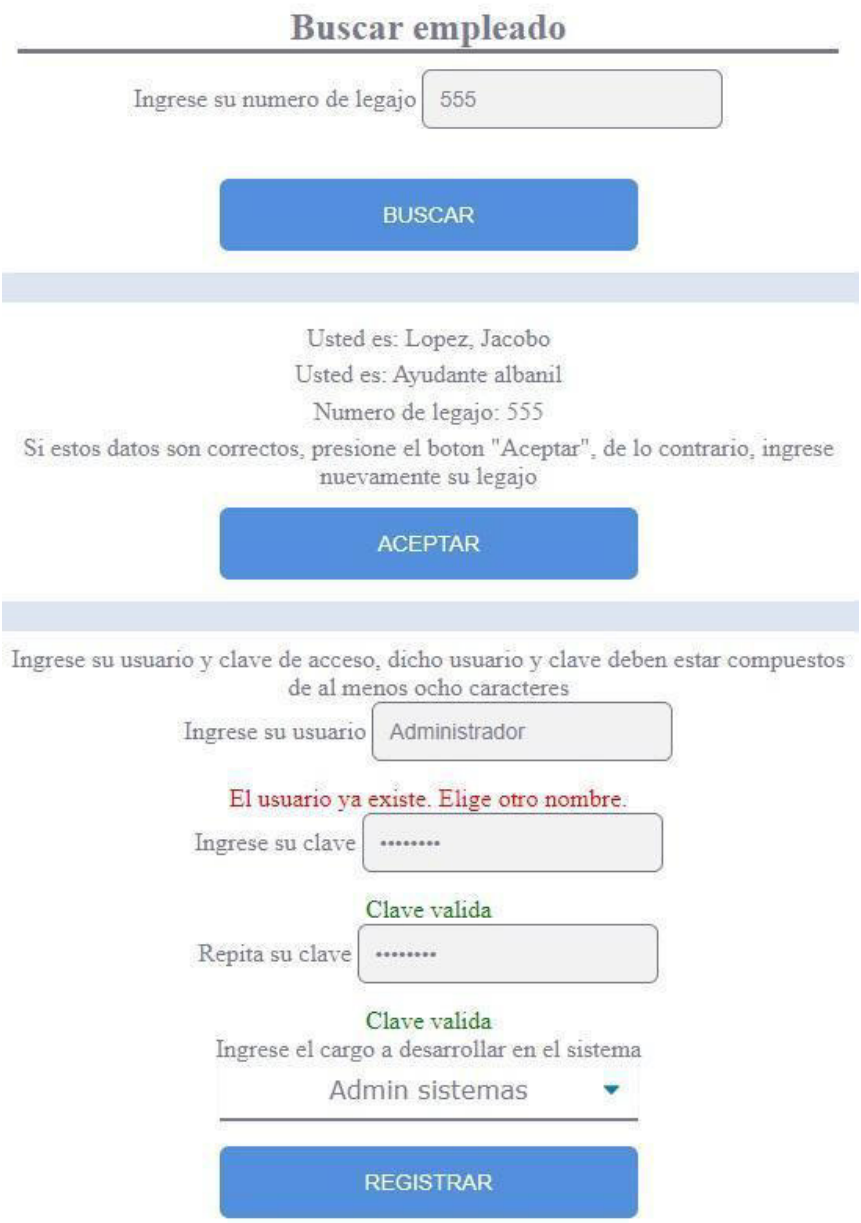


Figura 85

Solicitud AJAX para registro de nuevos usuarios

```
// recibo el objeto empleado para enviarlo al servlet con ajax
// para evitar una nueva consulta a la base de datos en el servlet

function crearUsuario(usuario){
    $.ajax({
        url: 'SvRegistrar',
        type: 'POST',
        dataType: 'json',
        contentType: 'application/json; charset=utf-8', // especifico que es jsc
        data: JSON.stringify(usuario),
        success: function (response) {
            redirigirIndex();
        },
        error: function (xhr, status, error) {
            console.error("Error:", error);
        }
    });
}

function redirigirIndex(){
    console.log('redirigir');

    $('#buscarEmpleado').css('display', 'none');
    $('#resultadoBusquedaEmpleado').css('display', 'none');
    $('#formularioRegistro').css('display', 'none');
    $('#mensajeCreacion').css('display', 'block');

    $('#btnUsuarioCreado').click(function () {
        window.location.href = "index.jsp";
    });
}
```

Al terminar de registrar el usuario, el sistema redirige a la página “index.jsp”.

La solicitud AJAX arma un objeto “Usuario” y lo envía al Servlet en formato JSON, el Servlet recibe este JSON y lo convierte en instancia de la clase “Usuario”. Mediante una instancia de Gson el mensaje JSO se puede convertir en una instancia de la clase Usuario, en este momento se genera de forma aleatoria el atributo “salt” que se utilizará para encriptar nuevamente la clave con el algoritmo SHA256. Este campo se almacenará junto con el nuevo usuario para ser utilizado posteriormente en la validación del usuario para el ingreso al sistema.

Todos los usuarios son almacenados con su atributo “autorizado” en false, de esta forma, el administrador del sistema podrá verificar los usuarios antes de otorgarles permisos para ingresar al sistema.

Figura 86

Servlet de registro de usuarios.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    // Leer el cuerpo de la solicitud JSON
    StringBuilder jsonBuffer = new StringBuilder();
    BufferedReader reader = request.getReader();
    String line;
    while ((line = reader.readLine()) != null) {
        jsonBuffer.append(line);
    }
    // almaceno el string obtenido del json
    String jsonData = jsonBuffer.toString();
    // instancia de Gson, incluyo el adapter en el builder para evitar problemas de parse a objetos LocalDate
    Gson gson = new GsonBuilder().registerTypeAdapter(LocalDate.class, new LocalDateAdapter()).create();
    // leer el string y parse a objeto Usuario
    Usuario usuario = gson.fromJson(jsonData, Usuario.class);
    // clave recibida
    String claveRecibida = usuario.getClave();
    // genero una cadena de caracteres aleatoria
    String salt = generarSalt();
    try {
        // encripto la clave recibida con el salt generado
        String claveEncriptada = encriptarClave(claveRecibida, salt);
        // guardo la nueva clave y el salt en el usuario
        usuario.setClave(claveEncriptada);
        usuario.setSalt(salt);
        control.crearUsuario(usuario);              // crear usuario
        // Enviar una respuesta al cliente
        response.getWriter().write("{\"mensaje\": true}");
    } catch (NoSuchAlgorithmException ex) {
        Logger.getLogger(SvRegistrar.class.getName()).log(Level.SEVERE, null, ex);
        response.getWriter().write("{\"mensaje\": false}");
    }
}

// generar un salt por cada nuevo usuario
private String generarSalt() {
    SecureRandom random = new SecureRandom();
    byte[] saltBytes = new byte[16];
    random.nextBytes(saltBytes);
    return Base64.getEncoder().encodeToString(saltBytes);
}
```

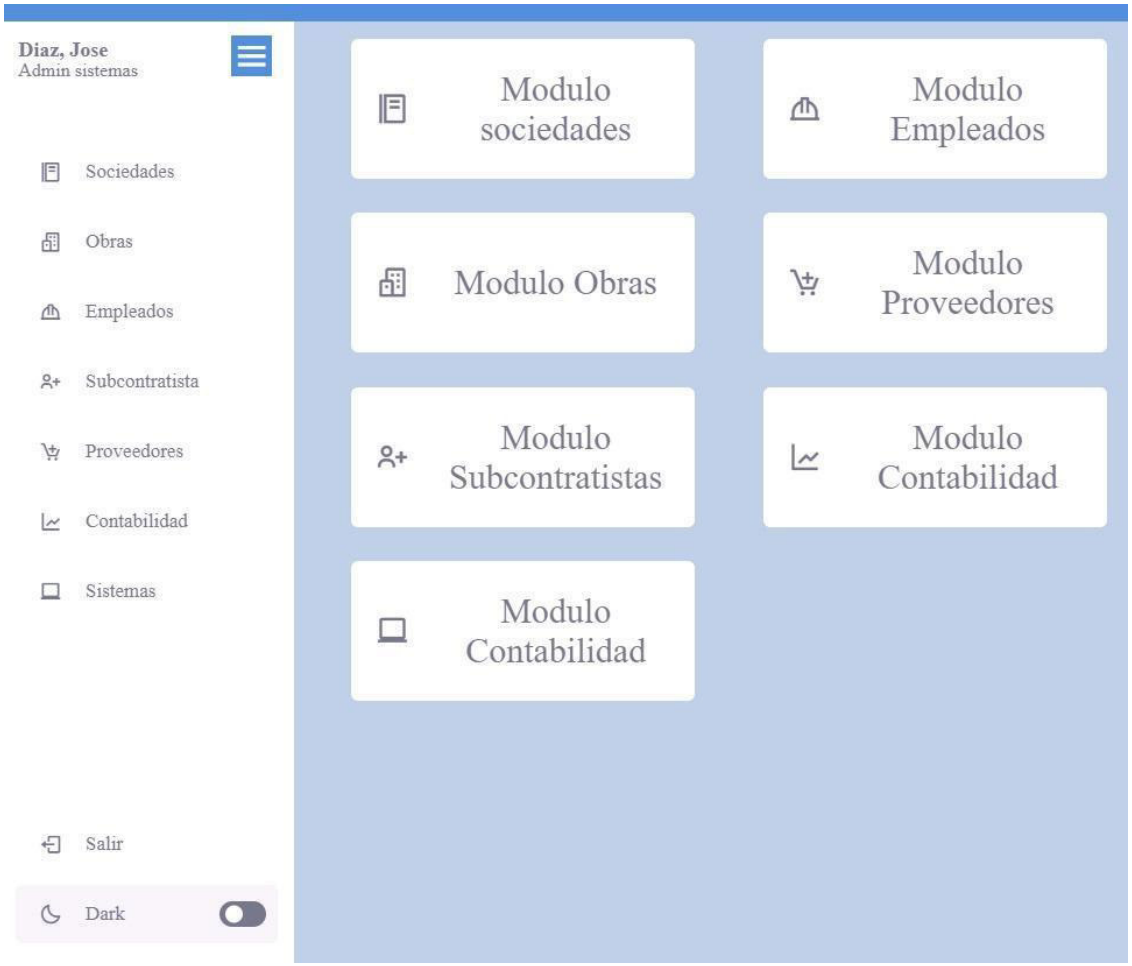
15.7. Creación de páginas para gestión de empleados

Una vez ingresado al sistema, el usuario podrá visualizar diferentes funcionalidades según su rol de usuario.

La mecánica de la gestión de los diferentes módulos es similar, creaciones, modificaciones, consultas y filtros se aplican a todos los aspectos del sistema. A modo de ejemplo se observará la mecánica de la gestión de empleados.

Figura 87

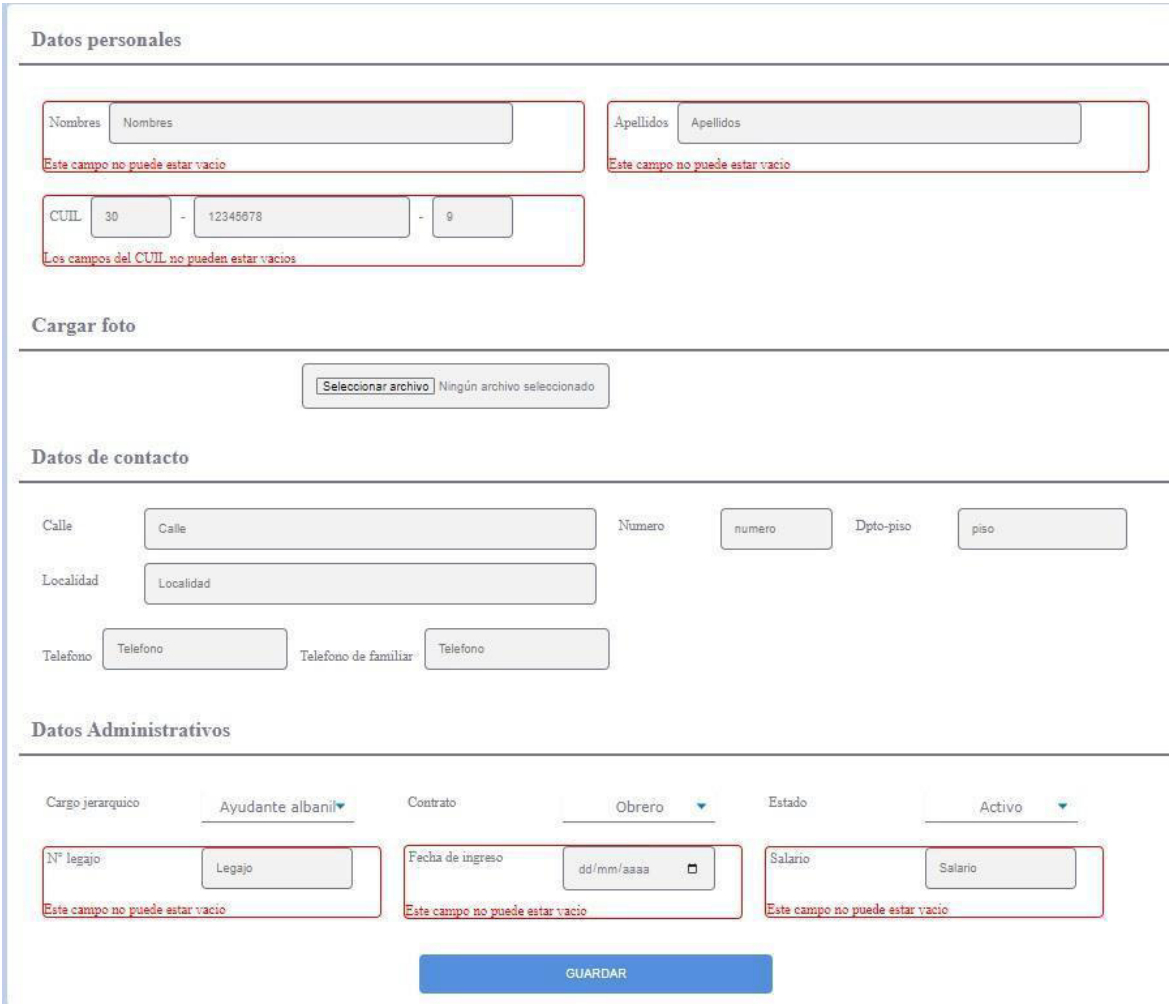
Página principal del usuario



El sistema está separado en módulos que engloba la gestión de la información por categorías. Dentro de cada categoría se puede acceder a la gestión de cada tema.

Figura 88

Registro de nuevos empleados



La página de registro de nuevos empleados cuenta con campos de ingreso de datos personales, datos de contacto y datos administrativos.

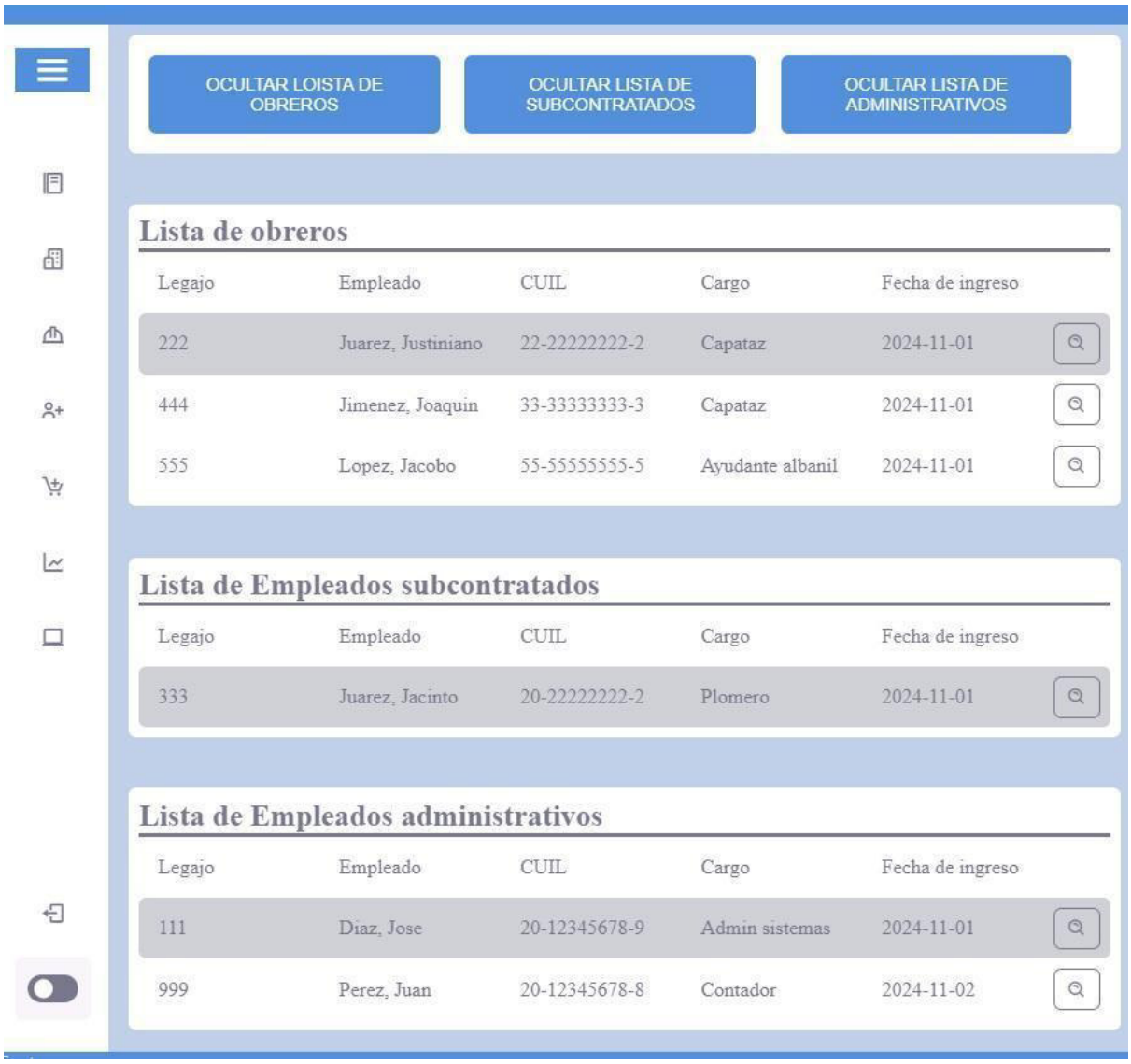
Los campos son capturados y verificados por JQuery, algunos de estos campos no pueden estar vacios, como por ejemplo, el campo sueldo. Aunque este campo presenta una particularidad, según el modelo de negocio, el campo “sueldo” no puede estar vacio, a menos que el empleado a registrar sea un empleado subcontratado, en este caso, el campo se vacía y se deshabilita.

En caso de que todos los campos sean verificados y sean correctos, la página envía una solicitud AJAX al Servlet correspondiente que se encarga del registro del empleado.

La siguiente página es la lista de empleados, al acceder a esta página, se envía una solicitud AJAX que devuelve una lista con todos los empleados. La pagina recibe esta lista, filtra los empleados por el tipo de contrato y los inserta en diferentes vistas en las que se puede visualizar información básica de cada empleado.

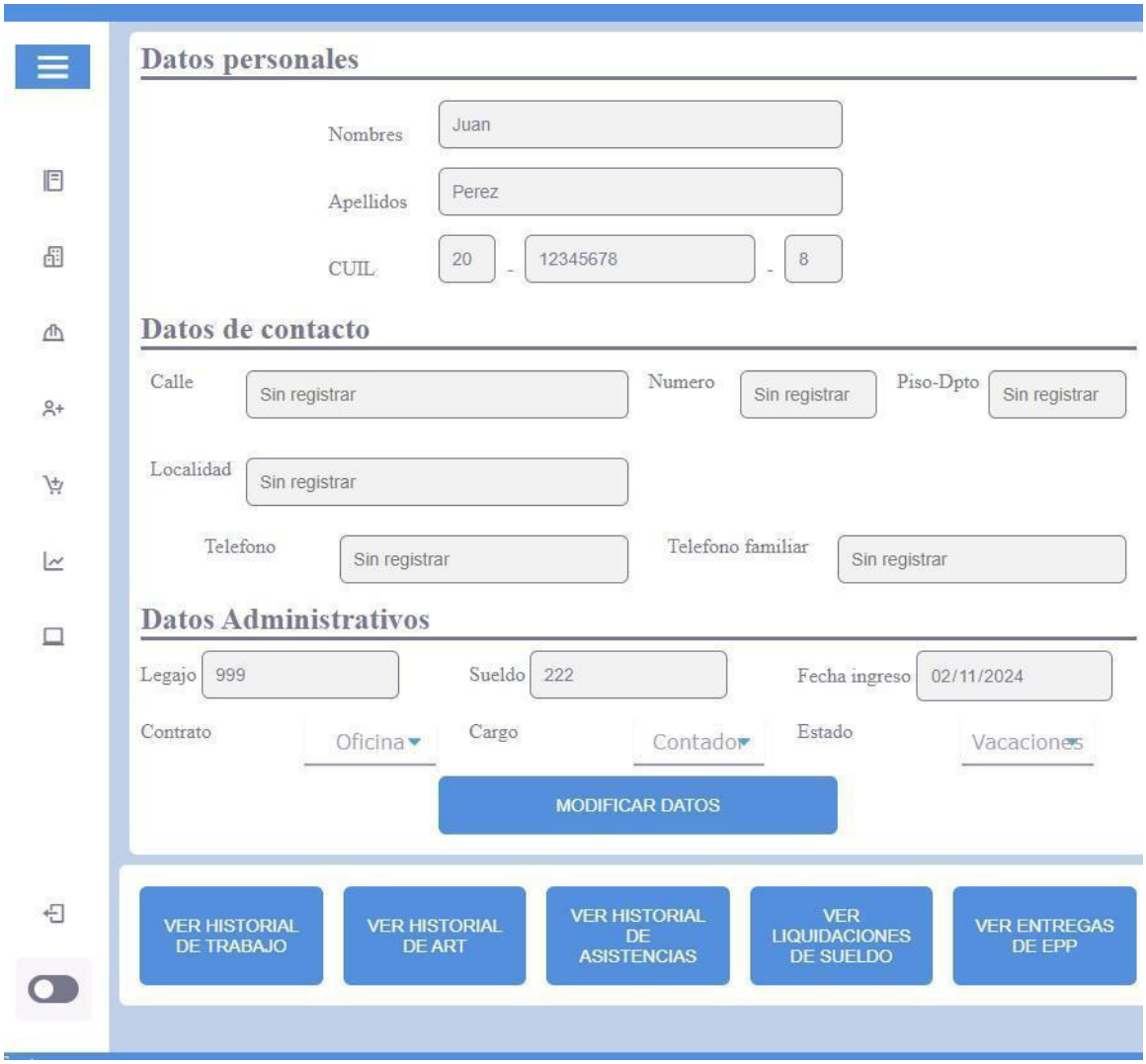
Figura 89

Listas de empleados



Cada empleado cuenta con un botón en forma de lupa, que sirve para redirigir a una página en la que se podrá ver la información del emplead en detalle.

Figura 90
Pagina de detalle de empleados



Esta página cumple una doble función, permite al usuario observar la información detallada de un empleado, y si así lo deseará, modificar los datos.

Para esta implementación, la información del empleado fue insertada en campos de texto que inicialmente están deshabilitados, si el usuario hace click sobre el botón “Modificar datos”, estos campos serán habilitados para su modificación. Si todos los campos son correctos, la página envía una solicitud AJAX con los nuevos datos del empleado para que el Servlet correspondiente actualice la información del empleado.

16. Problemas de desarrollo y soluciones

Durante el transcurso del desarrollo de la aplicación, han surgido algunos inconvenientes que han acarreado problemas al correcto funcionamiento de la aplicación. A continuación se presentan algunos de estos inconvenientes y sus respectivas soluciones.

16.1. GSON y conversión de datos de tipo LocalDate.

Cada vez que un servlet envía información de alguna instancia de clases, lo hace en formato JSON. En el proyecto se utiliza la biblioteca GSON.

Gson es una biblioteca desarrollada por Google de código abierto para Java y Android, que permite la conversión de objetos Java a su representación en JSON y vice-versa.

Esta librería fue agregada a través de Maven, de la misma forma en que se agrego el conector para base de datos PostgreSQL.

GSON se utiliza para serializar la información de la aplicación a JSON, sin embargo, GSON no puede serializar objetos de tipo LocalDate, como por ejemplo, la clase “Empleado” cuyo atributo “fecha_ingreso” almacena un objeto de tipo LocalDate.

Para solucionarlo, se debe registrar un adaptador personalizado que le indique a GSON como manejar datos de éste tipo.

Figura 91
Implementación de LocalDateAdapter.java para GSON

```java
public class LocalDateAdapter extends TypeAdapter<LocalDate> {

    private static final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

    @Override
    public void write(JsonWriter jsonWriter, LocalDate localDate) throws IOException {
        if(localDate == null){
            jsonWriter.nullValue();// manejo para LocalDate = null
        }
        else{
            jsonWriter.value(localDate.format(formatter));
        }
    }

    @Override
    public LocalDate read(JsonReader jsonReader) throws IOException {
        if(jsonReader.peek() == JsonToken.NULL){
            jsonReader.nextNull();
            return null;
        }
        String fecha = jsonReader.nextString();
        return fecha.isEmpty() ? null : LocalDate.parse(fecha, formatter);
    }
}
```

De esta forma, GSON traduce los datos de tipo LocalDate a String y luego convierte a JSON normalmente.

Para utilizar esta implementación de GSON, se debe inicializar una instancia de GSON, y a través del método “GsonBuilder ()” se debe agregar el LocalDateAdapter.

Figura 92
Deserializar un JSON a una instancia de la clase Empleado, utilizando GSON

```
response.setContentType("application/json");
response.setCharacterEncoding("UTF-8");

// request.getReader retorna un Reader que permite leer la peticion
BufferedReader recibirEmpleado = request.getReader();
//construyo el Gson con LocalDateAdapter
Gson gson = new GsonBuilder()
        .registerTypeAdapter(LocalDate.class, new LocalDateAdapter())
        .create();
// Convertir JSON a objeto Java y guardarlo en la variable de tipo Empleado
Empleado empleadoRecibido = gson.fromJson(recibirEmpleado, Empleado.class);
```

Al recibir un JSON que representa los datos de un empleado, se utiliza el método gson.fromJson (recibirEmpleado, Empleado.class), de esta forma, GSON convierte los datos a una instancia de la clase Empleado.

Figura 93
Serializar una instancia de la clase Empleado a JSON, utilizando GSON

```
// convierto el empleado encontrado a json
Gson gson = new GsonBuilder()
        .registerTypeAdapter(LocalDate.class, new LocalDateAdapter())
        .create();
String empleadoJson = gson.toJson(empleado);
respuestaJson = empleadoJson;
```

Para enviar datos de una instancia de la clase Empleado, se utiliza el método gson.toJson(empleado), de esta forma GSON lee los datos de la clase Empleado y convierte a JSON para luego ser enviado a la pagina que lo solicita.

16.2. GSON y la conversión de clases con relaciones cíclicas

Otro inconveniente que se presentó en el desarrollo de la aplicación es la conversión de clases con relaciones cíclicas.

Un ejemplo de esto son las clases “Empleado” y “GrupoTrabajo”, la relación en cuestión es que un empleado puede pertenecer a un grupo de trabajo, y un grupo de trabajo puede tener uno o muchos empleados.

Cuando se trata de convertir a formato JSON un empleado que pertenece a un grupo de trabajo, GSON comienza convirtiendo los datos de la clase “Empleado” como por ejemplo,

nombre, apellido, legajo, etc. La clase “Empleado” a su vez, esta clase contiene una instancia de la clase “GrupoTrabajo”. Gson convierte esta clase, pero esta a su vez, contiene una lista de instancias de la clase “Empleado” así que procede a convertir las instancias de “Empleado”. Pero estas a su vez contienen instancias de la clase “GrupoTrabajo”.

Esto genera un bucle sin fin que deriva en un error “stackOverflow”. Para solucionar este problema, se implemento clases DTO(Data Transfer Object).

Las clases DTO son clases que contienen los mismos atributos y métodos que las clases comunes, con la diferencia que estas no tienen relaciones cíclicas. Estas clases se utilizan para transcribir la información de otra clase.

Al transcribir la información a las clases DTO, son estas las que se utilizaran para ser convertidas a JSON con la librería GSON.

En este ejemplo se implementaron las clases “EmpleadoDTO” y “GrupoTtabajoDTO”. Esta implementación se aplicará a cualquier otra clase que tenga relación cíclica.

Figura 94

Clase EmpleadoDTO

```java
public class EmpleadoDTO {
    private int id_empleado;
    private int legajo;
    private Jerarquia jerarquia;
    private String nombres;
    private String apellidos;
    private String cuil;
    private String calle;
    private int altura;
    private String piso;
    private String localidad;
    private String telefono;
    private String telefono_familiar;
    private byte[] foto_dni;
    private String foto_dni_base64;
    private LocalDate fecha_ingreso; // LocalDate fecha sin hora
    private int antiguedad;
    private boolean despido; // true = despido, false = empleado vigente
    private BigDecimal sueldo_base;

    private Contrato contrato;
    private EstadoEmpleado estado;
    private GrupoTrabajoDTO grupoDTO;
    private ArrayList<EmpleadoObra> asignaciones = new ArrayList<>();
    private ArrayList<HistorialART> historialART = new ArrayList<>();
    private ArrayList<LiquidacionSueldo> liquidaciones = new ArrayList<>();
    private ArrayList<EntregaEPP> planillaEPP = new ArrayList<>();
    private ArrayList<Asistencia> asistencias;
```

Figura 95

Clase GrupoTtabajoDTO

```java
// Data transfer objects
public class GrupoTrabajoDTO {
    private int id_grupo;
    private String nombre_grupo;
    private EmpleadoDTO capataz;
    private List<EmpleadoDTO> empleados;
```

Estas clases no contienen relación cíclica entre ellas, por lo que no generaran un bucle infinito al ser convertidas a JSON con la biblioteca GSON.

También se implementaron métodos para la conversión de una clase a una clase DTO en los JPA Controller de las respectivas clases

Figura 96

Método convertirAEmpleadoDTO en EmpleadoJpaController

```
//convertir una instancia de Empleado a EmpleadoDto
public EmpleadoDTO convertirAEmpleadoDTO(Empleado empleado) {
    EmpleadoDTO empDTO = null;

    if (empleado == null) {
        empDTO = null;
    }
    else{
        empDTO = new EmpleadoDTO();
        empDTO.setIdEmpleado(empleado.getId());
        empDTO.setLegajo(empleado.getLegajo());
        empDTO.setNombres(empleado.getNombres());
        empDTO.setApellidos(empleado.getApellidos());
        empDTO.setCuil(empleado.getCuil());
        empDTO.setCalle(empleado.getCalle());
        empDTO.setAltura(empleado.getAltura());
        empDTO.setPiso(empleado.getPiso());
        empDTO.setLocalidad(empleado.getLocalidad());
        empDTO.setTelefono(empleado.getTelefono());
        empDTO.setTelefonoFamiliar(empleado.getTelefonoFamiliar());
        empDTO.setFotoDni(empleado.getFotoDni());
        empDTO.setFechaIngreso(empleado.getFechaIngreso());
        empDTO.setDespido(empleado.isDespido());
        empDTO.setSueldoBase(empleado.getSueldoBase());

        empDTO.setJerarquia(empleado.getJerarquia());
        empDTO.setContrato(empleado.getContrato());
        empDTO.setEstado(empleado.getEstado());
        empDTO.setAsignaciones(empleado.getAsignaciones());
        empDTO.setHistorialART(empleado.getHistorialART());
        empDTO.setLiquidaciones(empleado.getLiquidaciones());
        empDTO.setPlanillaEPP(empleado.getPlanillaEPP());
        empDTO.setAsistencias(empleado.getAsistencias());
    }

    return empDTO;
}
```

Figura 96

Método convertirGrupoTrabajoDTO en GrupoTrabajoJpaController

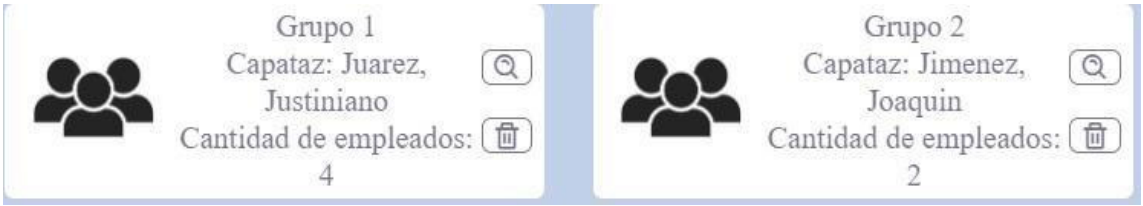
```
public GrupoTrabajoDTO convertirGrupoTrabajoDTO(GrupoTrabajo grupoTrabajo) {
    EmpleadoJpaController empJpa = new EmpleadoJpaController();
    if (grupoTrabajo == null) {
        return null;
    }
    EmpleadoDTO capataz = empJpa.convertirAEmpleadoDTO(grupoTrabajo.getCapataz());
    List<EmpleadoDTO> listaEmpleadosDTO = new ArrayList<>();
    // obtener lista de empleados
    List<Empleado> listaEmpleados = grupoTrabajo.getListaEmpleados();
    for(Empleado emp : listaEmpleados){
        // si el empleado no es capataz, agrego a la lista de empleadosDTO
        if(!"Capataz".equals(emp.getJerarquia().getDescripcion())){
            EmpleadoDTO empDTO = empJpa.convertirAEmpleadoDTO(emp);
            listaEmpleadosDTO.add(empDTO);
        }
    }
    // Crear y retornar el DTO
    return new GrupoTrabajoDTO(
        grupoTrabajo.getIdGrupo(),
        grupoTrabajo.getNombreGrupo(),
        capataz,
        listaEmpleadosDTO
    );
}
```

16.3. Interfaz grafica con datos desactualizados

Este problema ocurre cuando el navegador está mostrando una versión anterior de una página almacenada en cache. Un ejemplo de este caso ocurrió en la página “lista_grupo_trabajo.jsp” esta página muestra información básica de los grupos de trabajo existentes, tales como nombre del grupo, capataz del grupo y cantidad de empleados del grupo.

Figura 97

Pagina “lista_grupo_trabajo.jsp”



Cada grupo cuenta con un botón que redirige una página “detalle_grupo_trabajo.jsp” la cual muestra información más detallada del grupo.

Figura 98

Pagina “detalle_grupo_trabajo.jsp”

En esta nueva página se pueden realizar varias acciones, como cambiar el capataz, agregar y desvincular empleados del grupo. Estas acciones de cambios de datos se ven reflejadas en la base de datos, sin embargo, al retroceder y volver a la página “lista_grupo_trabajo.jsp” la nueva información del grupo no está cargado.

Una de las causas de este problema radica en el lado del cliente, al retroceder de página en el navegador, éste carga una versión anterior de la página “lista_grupo_trabajo.jsp” almacenada en cache. La solución a este inconveniente consiste en forzar la solicitud AJAX al volver a la página, desde el archivo Javascript que gestiona a la página en cuestión se utiliza el evento “pageshow” que se dispara al regresar a la pagina desde el historial del navegador.

Figura 99

Evento “pageshow” en Javascript

```
// Detectar cuando se carga la página desde el historial del navegador
window.addEventListener('pageshow', function (event) {
    if (event.persisted|| performance.getEntriesByType('navigation')[0].type === 'back_forward') {
        buscarGrupos(); // Volver a cargar los datos de grupos
    }
});
```

La otra causa de este problema es la persistencia de la memoria cache de la JPA, a pesar de que los datos fueron almacenados correctamente, la memoria cache de la JPA está devolviendo una versión anterior de los datos en lugar de una versión actualizada.

La solución consiste en forzar la sincronización de la JPA con la base de datos, mediante el método “refresh()” dentro del JPA Controller de la clase en cuestión, se asegura que cualquier cambio en la base de datos se vea reflejada en la entidad gestionada por la JPA.

Figura 100

Forzar sincronización de JPA

```
public GrupoTrabajo findGrupoTrabajo(int id) {
    EntityManager em = getEntityManager();
    try {
        GrupoTrabajo grupo = em.find(GrupoTrabajo.class, id);
        // metodo refresh forzar sincronizacion con la base de datos
        em.refresh(grupo);
        return grupo;
    } finally {
        em.close();
    }
}
```

16.4. Problemas al recuperar imágenes desde la base de datos

Desde el formulario de registro de nuevos empleados, el usuario tiene la opción de cargar una foto del DNI empleado. Del lado del cliente, esta foto es convertida a Base64 y enviada al servlet, al llegar al servlet, éste convierte el dato Base64 a byte y lo almacena en la base de datos en este nuevo formato.

Al intentar recuperar la imagen e insertarla en la interfaz grafica, en la página de detalle del empleado por ejemplo, esta imagen se carga cortada. Este problema puede presentarse debido a que el tamaño de la imagen cargada desde el formulario de registro de empleados

es muy grande, esto requiere un redimensionamiento o de lo contrario, el tamaño de la imagen puede ser incompatible con el diseño de la página.

Para solucionar este inconveniente, se implementó el método “redimensionarImagenBase64()” en el servlet encargado del registro de nuevos empleados. El método recibe como parámetros el dato “Base64” enviado desde la página, y unos parámetros que indican el alto y ancho de la nueva imagen a redimensionar, y devuelve la imagen en formato byte con nuevas dimensiones de alto y ancho.

Figura 101

Método para redimensionar imágenes

```
public byte[] redimensionarImagenBase64(String base64Image, int ancho, int alto) throws IOException {
    // Decodificar la imagen Base64 a bytes
    byte[] imageBytes = Base64.getDecoder().decode(base64Image);
    // Crear un InputStream a partir de la coonversion a byte
    ByteArrayInputStream inputStream = new ByteArrayInputStream(imageBytes);
    // Crear un ByteArrayOutputStream para almacenar la imagen redimensionada
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    // Usar Thumbnails para redimensionar
    Thumbnails.of(inputStream)
              .size(ancho, alto) // Dimensiones deseadas
              .outputFormat("jpg") // Formato de salida
              .toOutputStream(outputStream);
    // Retornar la imagen redimensionada como byte[]
    return outputStream.toByteArray();
}
```

De esta manera se asegura que cada imagen tendrá el mismo tamaño, independientemente del tamaño inicial que poseían al ser cargadas en el formulario. También reducen el espacio de almacenamiento en la base de datos y se adaptan de mejor forma al diseño de las páginas.

El presente informe muestra todas las actividades realizadas en la empresa KYDMA S.R.L. En esta actividad se trabajo principalmente en la evaluación de la infraestructura informática de la organización para una posterior captura de requerimientos, planeación y desarrollo de un software que cumpla con las expectativas de los stakeholders.

La comunicación constante con el cliente es fundamental para garantizar el éxito del proyecto ya que describe con detalle los procesos de la empresa y las necesidades a cubrir, gracias a esto se ha podido desarrollar una planeación clara y consistente para el desarrollo de una aplicación personalizada al modelo de negocios de esta organización.

La fase de diseño permitió representar gráficamente los actores y sus actividades dentro de la empresa, dando un vistazo inicial de cómo será el software.

La fase de desarrollo comenzó sin problemas mayores. Llegado a esta etapa ya se tenía en claro los requerimientos y funcionalidades del software gracias al relevamiento de requisitos en la fase inicial del proyecto. Aun así surgieron algunos problemas de desarrollo, específicos de los modelos planteados, frameworks y las tecnologías utilizados.

Los avances presentados al cliente han sido recibidos de forma positiva, la interfaz grafica es intuitiva y fácil de utilizar. Aunque las vistas del sistema son simples y contienen algunos aspectos a mejorar, logra cumplir con los requerimientos del cliente.

Respecto a los demás usuarios del sistema, no se ha podido evaluar el nivel de aceptación. Cada software nuevo posee una curva de aprendizaje de usuario, por lo que se requiere de más tiempo para poder evaluar la satisfacción de los usuarios.

Algunas de las mejoras que se pueden aplicar al sistema son la implementación de una interfaz de usuario más atractiva y que permita al usuario modificar aspectos como las paletas de colores o el tamaño del texto.

Implementación de búsquedas avanzadas mediante los métodos en los Jpa Controller que permita la búsqueda de resultados en base a una serie de parámetros. En el sistema se ha implementado métodos específicos para búsquedas especializadas, sin embargo es recomendable unificar estos métodos en un nuevo método que permita búsquedas mas genéricas.

Pruebas para verificar la robustez, disponibilidad, capacidad, vulnerabilidad del sistema, etc.

Redacción de un manual de usuario y manual de soporte técnico.

17. Bibliografía

Bourque P., Fairley Richard E. (2014) *Guía al cuerpo de conocimiento de la ingeniería en software.* IEEE Computer Society.

Pressman Roger S. (2003) *Ingeniería del software un enfoque práctico séptima edición*. McGraw Hill.

Appmaster (23 de septiembre de 2021) Aplicación de escritorio o aplicación web: pros y contras.
https://appmaster.io/es/blog/aplicacion-de-escritorio-o-aplicacion-web-pros-y-contras

Back4App (19 de junio de 2022) Los 10 mejores lenguajes de programación del lado del servidor.
https://blog.back4app.com/es/los-10-mejores-lenguajes-de-codificacion-del-lado-del-servidor/

DataScientest (30 de octubre de 2023) SQL vs NOSQL: diferencias, usos, ventajas e inconvenientes.
https://datascientest.com/es/sql-vs-nosql-diferencias-usos-ventajas-y-inconvenientes

HostDime (26 de marzo de 2020) ¿Qué es apache Tomcat?
https://www.hostdime.com.ar/blog/que-es-apache-tomcat/

Hostinger (22 de marzo de 2024) ¿Qué es AJAX? Ejemplos prácticos y funcionamiento.
https://www.hostinger.com.ar/tutoriales/que-es-ajax

IBM (30 de enero de 2024) Java Persistence Api (JPA)
https://www.ibm.com/docs/es/was-liberty/nd?topic=liberty-java-persistence-api-jpa

JavaBeat (21 de febrero de 2014) EclipseLink – JPA Annotations.
https://javabeat.net/eclipselink-jpa-annotations/

Openwebinars (24 de agosto de 2022) Arquitectura de software: que es y que tipos existen.
https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/

Oracle (4 de abril de 2024) ¿Qué es JSON?
https://www.oracle.com/ar/database/what-is-json/