

# Proyecto Individual

Memoria del Plan del Proyecto de Modelado Software

Autor: Luis Jaramillo Pulido

# *ÍNDICE*

1. Introducción
2. Objetivos
3. Software utilizado en el desarrollo de la app.
4. Capas de Presentación y Negocio.
5. Diseño
6. Vistas

# **PRESENTACIÓN DEL PROYECTO**

## **1. INTRODUCCIÓN**

La idea principal es desarrollar una herramienta que permita la gestión y administración de empleados y departamentos, entre sus principales funcionalidades están las funciones de alta, baja y modificación de los departamentos, salario de los empleados de un departamento y en empleados alta y baja.

## **2. OBJETIVOS:**

El objetivo de la aplicación es facilitar y simplificar la administración de empleados y departamentos a través de una interfaz gráfica amigable, que permita el acceso a las funciones de gestión de los empleados, donde se registrarán el ID de cada empleado, junto a su salario según el tipo de empleado que sea empleado a Tiempo Completo o Parcial. Adicionalmente, otros usuarios podrán realizar acciones de gestión limitada como consultas en departamentos (para saber el salario total de los empleados en un departamento).

### 3. SOFTWARE UTILIZADO EN EL DESARROLLO DE LA APP:

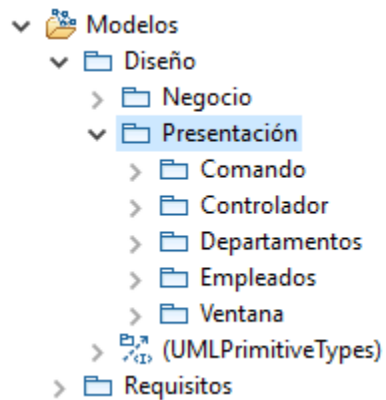
Los servicios y aplicaciones utilizadas para el desarrollo de la app son los siguientes:

- IBM RATIONAL SOFTWARE ARCHITECT FOR WEBSPHERE SOFTWARE v.9.0.
- NETBEANS
- SQLITE
- WORD
- Eclipse

### 4. CAPAS DE PRESENTACIÓN Y NEGOCIO











#### DIAGRAMA DE PRESENTACION

En el modelo UML en el apartado paquetes diseño, Presentación está ubicado el diagrama de presentación.
























#### Diagrama de Negocio

En el modelo UML en el apartado paquetes diseño, Negocio está ubicado el diagrama de negocio.

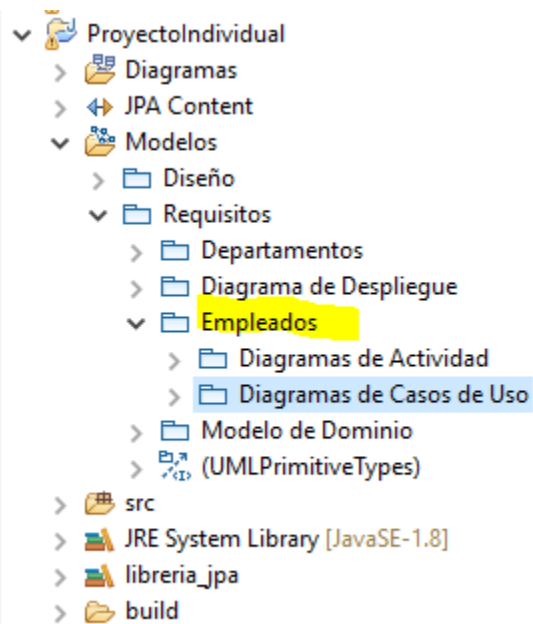
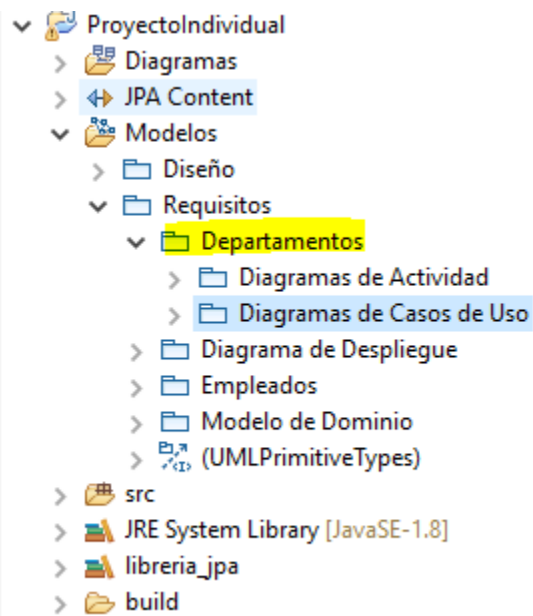
- ▼  Modelos
  - ▼  Diseño
    - ▼  Negocio
      - >  Departamentos
      - >  Empleados
      - >  FactoriaSA
      -  DiagramaEntidadesJPA
      - >  Presentación
      - >  (UMLPrimitiveTypes)
    - >  Requisitos

## Modelo de Dominio
















En el modelo UML en el apartado paquetes requisitos.

- ▼  ProyectoIndividual
  - >  Diagramas
  - >  JPA Content
  - ▼  Modelos
    - >  Diseño
    - ▼  Requisitos
      - >  Departamentos
      - >  Diagrama de Despliegue
      - >  Empleados
      - ▼  Modelo de Dominio
        - >  Asociaciones
        -  Modelo de Dominio
        - >  A Tiempo Completo
        - >  A Tiempo Parcial
        - >  Departamentos
        - >  Empleados
      - >  (UMLPrimitiveTypes)
    - >  src
    - >  JRE System Library [JavaSE-1.8]
    - >  libreria\_jpa
    - >  build

## Diagramas de Actividad y Casos de Uso



## Diagrama de Despliegue

- ▼  ProyectoIndividual
  - >  Diagramas
  - >  JPA Content
  - ▼  Modelos
    - >  Diseño
    - ▼  Requisitos
      - >  Departamentos
      - >  Diagrama de Despliegue
      - >  Empleados
      - >  Modelo de Dominio
      - >  (UMLPrimitiveTypes)
  - >  src
  - >  JRE System Library [JavaSE-1.8]
  - >  libreria\_jpa
  - >  build



## **5. DISEÑO**

Se ha usado una arquitectura multicapa, dentro de la arquitectura multicapa los principales patrones aplicados son:

### **Patrones Generales usados tanto en presentación como en negocio**

**Patron Factoria Abstracta:** Utilizada para la generación de los DAO y los servicios de aplicación Singleton.

**PATRON DE DISEÑO SINGLETON** Garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella. Es la encargada de que una clase solo pueda instanciar un objeto, su constructor es privado, y se instancia a través de un método publico estático.

### **CAPA DE PRESENTACION**

**Patron Command:** Patrón que indica las acciones posibles en el sistema.

**DISPATCHER VIEW** Es el responsable del control de la vista y la navegación, controlando la elección de la siguiente vista a mostrar y proporcionando el mecanismo para dirigir el control a este recurso.

**APPLICATION CONTROLLER** Centraliza las peticiones de comandos y vistas.

**VIEWHELPER** Es el responsable de ayudar a la vista o al controlador a completar su procesamiento incluyendo la obtención de los datos requeridos por la vista y su almacenamiento en el modelo intermedio.

**Patron Service to Worker** Se desea llevar a cabo el manejo de peticiones y la invocación de lógica de negocio antes de pasar el control a la vista

### **CAPA DE NEGOCIO**

**Servicio de Aplicación** Se usa un servicio de aplicación para centralizar y agregar un comportamiento para proporcionar una capa de servicio.

**Transfer Object** se usa para encapsular los datos. Se usa para enviar y recuperar el Transfer Object. Construye un nuevo Transfer Object basado en los requerimientos de la aplicación cuando el cliente solicita un Transfer Object.

**Entity ManagerFactory** Es único y es con el que se gestionan todas las entidades, este será capaz de construir un objeto de tipo EntityManager que como su nombre indica gestiona un conjunto de entidades u objetos. En principio estas entidades son objetos POJO normales.

**Entity Manager** Es el encargado de salvar los datos a la base de datos, eliminarlos de la base de datos o etc. Para ello define otro concepto adicional "PersistenceContext". Este concepto hace referencia a los objetos que han sido manipulados por el EntityManager y se encuentran controlados por él. Para conseguir que algunos de nuestros objetos pasen a ubicarse dentro del PersistenceContext bastará con invocar a alguno de los métodos típicos del EntityManager.

**Patrón Objeto de Negocio.** Separa los datos de negocio y la lógica usando un modelo de objetos, estos objetos de negocio encapsulan y manejan datos del negocio, comportamiento y persistencia

**Almacen de Dominio**

## 6. VISTAS

### EMPLEADOS

Proyecto Individual

Empleados Departamentos

EMPLEADO

ID: 7

DNI: 1234

Nombre: Luis

Apellidos: Jaramillo

Departamento: 10

☐ Tiempo Completo ☒ Tiempo Parcial

Impuesto:  Horas: 5

Salario:  Salario por hora: 10

Nómina

50,00€

Alta Baja

## Departamentos

Proyecto Individual

Empleados

Departamentos

DEPARTAMENTO

ID: 10

Cargar

Nombre: Recursos humanos

EMPLEADOS

7: Luis Jaramillo DNI: 1234

NÓMINA DEPARTAMENTO

50,00€

Alta

Baja

Modificar

DEPARTAMENTOS

9: Marketing

10: Recursos humanos

11: Contabilidad