1. Singleton

El patrón Singleton se aplica en la clase *Conexion* para garantizar que solo exista una instancia de *BasicDataSource*, la cual gestiona las conexiones a la base de datos. Esto optimiza el uso de recursos y asegura que la creación y administración de conexiones se haga de manera controlada y eficiente. Se encuentra en el método *getDataSource()* que devuelve la instancia única de *BasicDataSource*.

2. DAO (Data Access Object)

El patrón DAO se aplica en las clases *EmpleadoDAO* y *NominaDAO* para separar la lógica de acceso a la base de datos de la lógica de negocio. Este patrón centraliza las operaciones de consulta, inserción y modificación de datos, facilitando la reutilización y el mantenimiento del código. Se usa cuando los métodos como *obtenerEmpleados*() o *buscarEmpleados*() interactúan con la base de datos.

3. Front Controller

La clase *EmpresaController* actúa como un Front Controller, centralizando la gestión de las solicitudes entrantes y enrutándolas a los controladores adecuados (*EmpleadoController*, *NominaController*). Esto permite un punto de entrada único para el manejo de peticiones, mejorando la estructura y control del flujo de navegación en la aplicación.

4. MVC (Model-View-Controller)

El patrón MVC se integra en la arquitectura para separar la lógica de negocio (modelo), la presentación (vista) y el manejo de solicitudes (controlador). Los controladores como *EmpleadoController* y *NominaController* manejan las peticiones del usuario, los DAO actúan como modelos de acceso a los datos y las vistas son archivos JSP que presentan los datos al usuario. Este patrón organiza el sistema, facilitando el mantenimiento y la escalabilidad.