Paquetes del integrador

Paquetes:

- **edu.unam**: clase Sistema *(Inicia el programa)*.
- **edu.unam.vistas**: clases correspondientes a las vistas.
- **edu.unam.servicios**: clases correspondientes a los servicios *(Puede ser una clase al ser un proyecto simple)*.
- **edu.unam.modelo**: clases del modelo *(de esto deben armar el diagrama de clases)*.
- **edu.unam.repositorio**: clases que interactua con la parte de persistencia.

Consideraciones

Los puristas de la POO consideran que una capa de servicio es un antipatrón (porque produce un modelo de dominio anémico), pero es útil en aplicaciones de grandes empresas porque ofrece un modelo más fácil de trabajar.

La capa de Servicio encapsula la lógica de negocios de la aplicación. La palabra Servicio se usa para enfatizar el punto de que una capa de lógica de negocios modelada usando principios de diseño orientados al Servicio puede ser utilizada por diferentes consumidores, como una capa que representa una aplicación de escritorio, una capa de presentación web, una capa de integración API, clientes móviles remotos, etc. Esto permite que la lógica de negocios se escriba una vez y se consuma en múltiples lugares a través de diferentes tecnologías, ubicaciones y aplicaciones.

La capa de persistencia es responsable de ofrecer operaciones de acceso a datos a la capa de servicio. Para cumplir con el principio de bajo acoplamiento, la capa de servicio no debe preocuparse por cómo y dónde se almacenan los datos, simplemente debe acceder a los datos requeridos cuando sea necesario. Mediante esto el código de acceso a datos esté separado del código de lógica empresarial.

El repositorio es un patrón de diseño comúnmente utilizado para implementar la capa de persistencia. Entre muchas otras cosas, permite que los datos de la aplicación se modelen y administren como un modelo de dominio. Esto permite el uso del diseño impulsado por el dominio (Domain-driven design) al garantizar que los usuarios técnicos y no técnicos compartan una terminología común y que los artefactos técnicos imiten a sus contrapartes del mundo real lo más fielmente posible. El patrón de repositorio también es útil porque permite que la capa de servicio acceda a todas las fuentes de datos (o al menos a la mayoría) a través de una interfaz consistente, la mayoría de los marcos que ofrecen una capa de persistencia basada

en repositorio proporcionan los métodos de alta, baja, modificación y consulta básicos en todos los repositorios.

En resumen vamos a tener lo siguiente:

Aplicación <--> Servicio <--> Repositorio (Persistencia) <--> Modelo (o entidades)