UNJu | Universidad Nacional de Jujuy

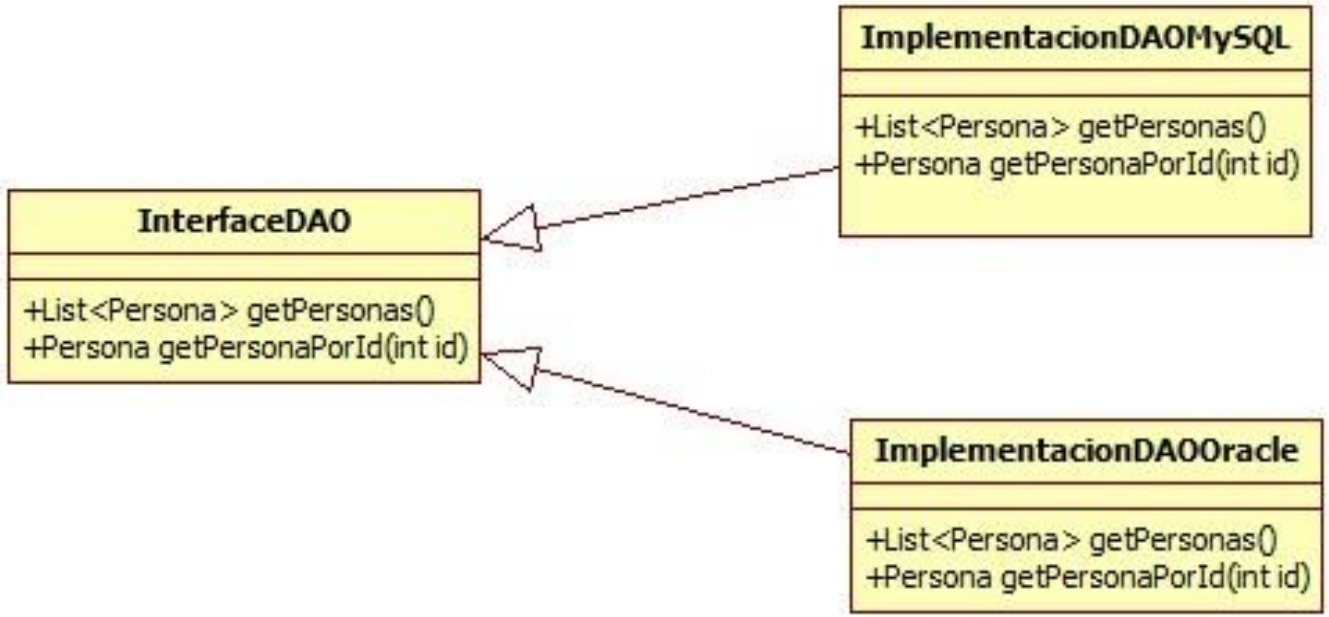
Argentina programa 4.0

Patron de diseño DAO

Patron de objetos de acceso a datos | DAO

```
public interface InterfaceDAO {
   public List<Persona> getPersonas();
   public Persona getPersonaPorNombre (String nombre);
   ...
   public void salvaPersona (Persona persona);
   public void modificaPersona (Persona persona);
   ...
   public void borraPersonaPorNombre (String nombre);
   ...
}
```

Interface | DAO



Participantes en el patrón DAO

Interfaz de objeto de acceso a datos
: Define las operaciones estándar que serializarán en un objeto modelo.

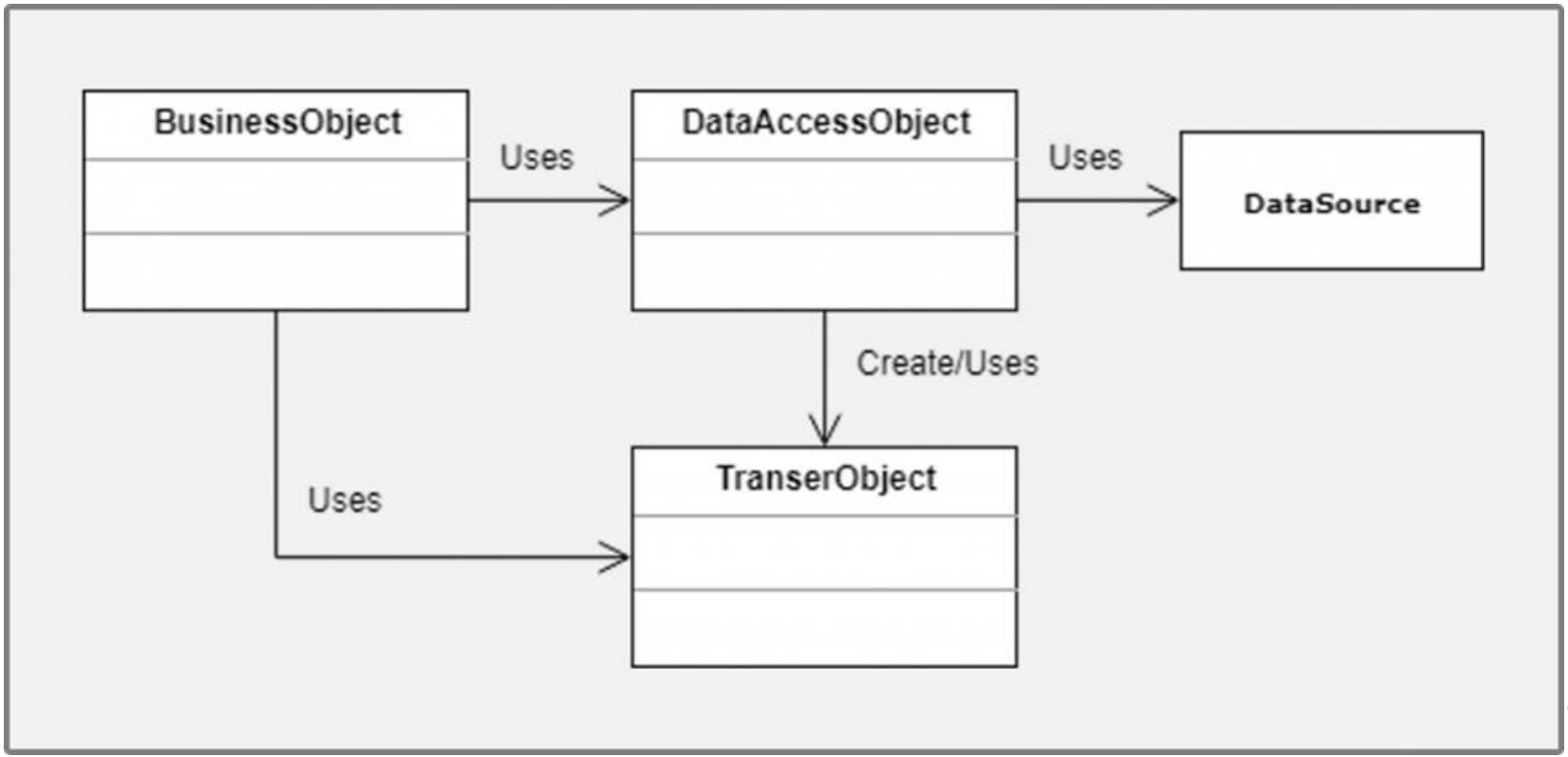
Clase concreta de objeto de acceso a datos
: Implementa la interfaz anterior.
Esta clase es responsable de obtener datos de una fuente de datos que puede ser una base de datos/xml o cualquier otro mecanismo de almacenamiento.

Participantes en el patrón DAO

Objeto modelo u objeto de valor

: Es un POJO (Plain Old Java Object o antiguo objeto plano de java) simple que contiene métodos get/set para almacenar datos recuperados mediante la clase DAO

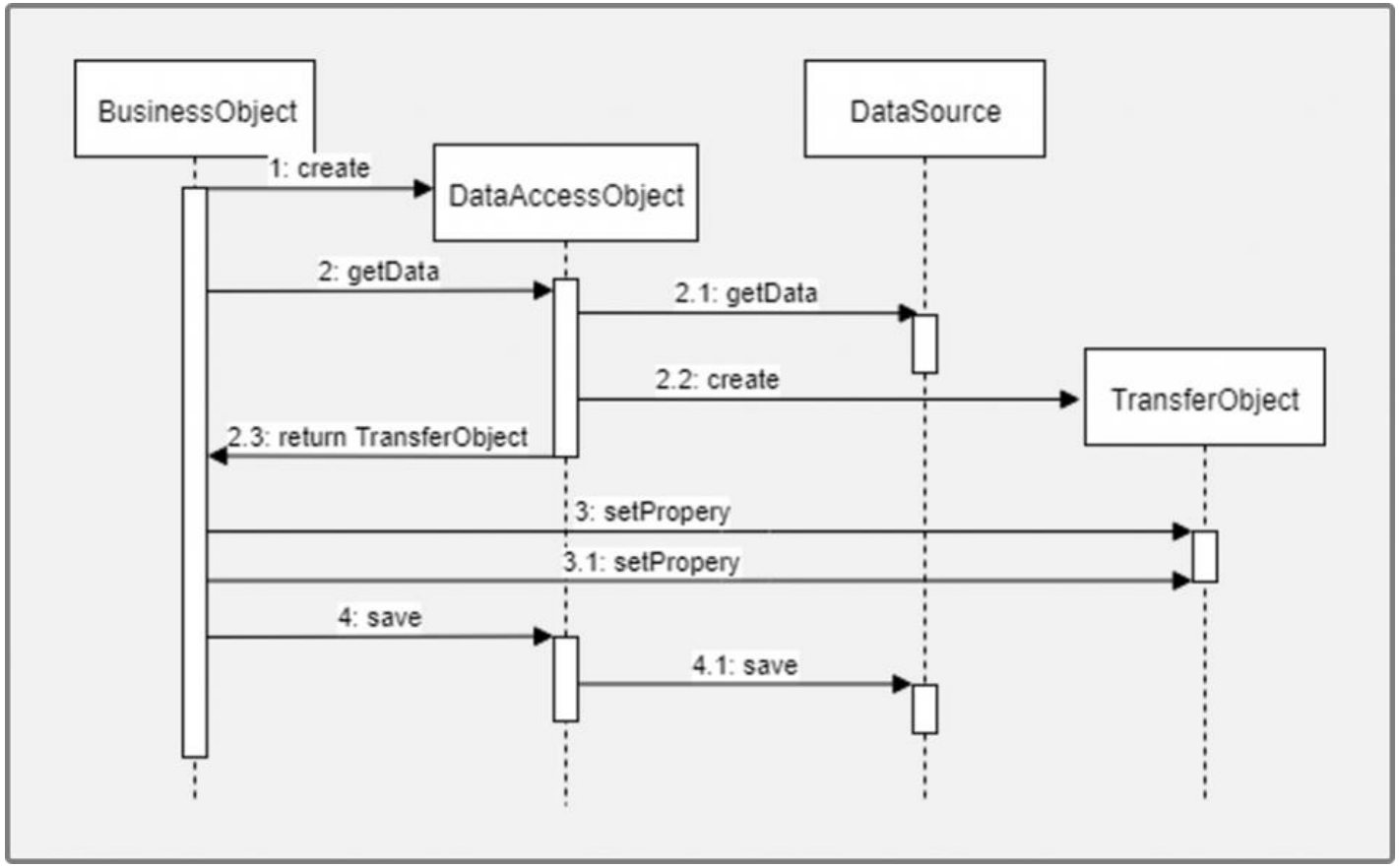
Componentes del patrón DAO



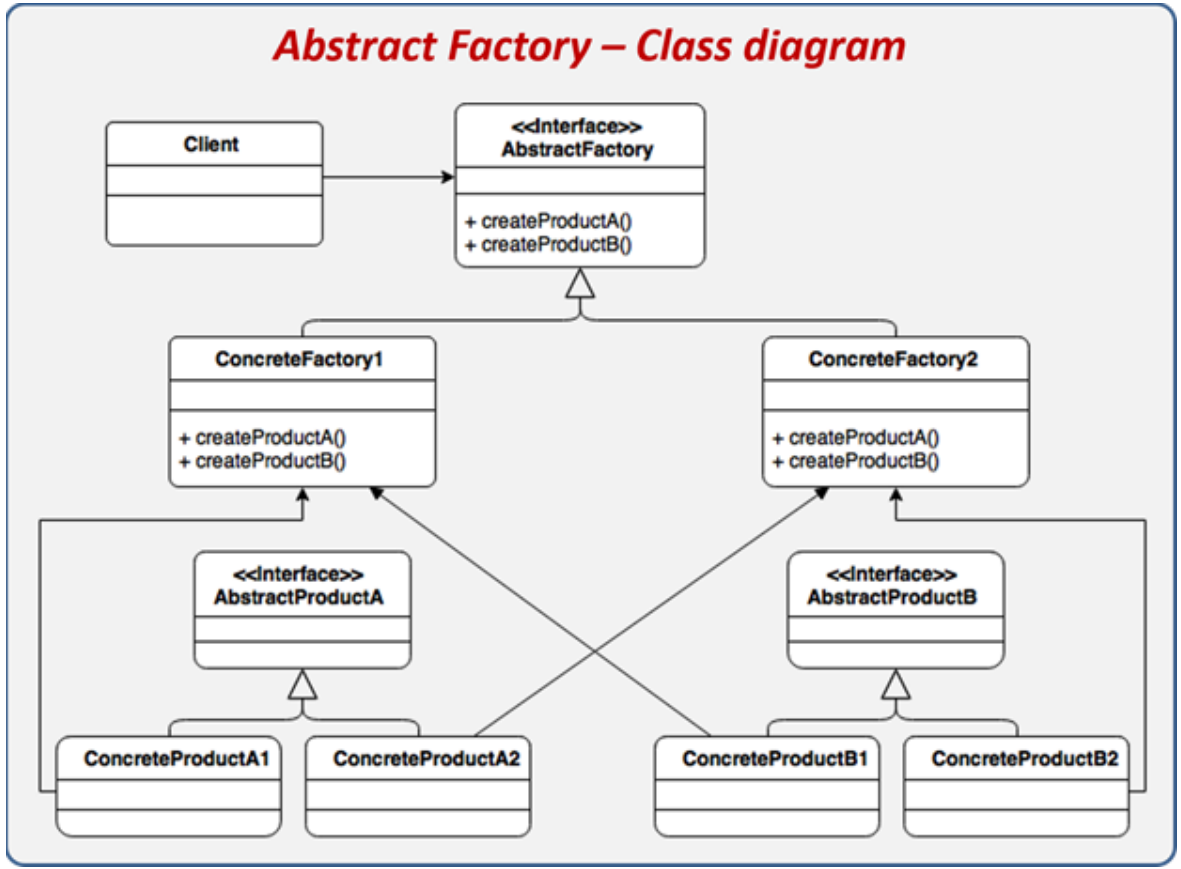
Componentes del patrón DAO

- **BusinessObject:** Representa un objeto con la lógica de negocio.
- **DataAccessObject:** Representa una capa de acceso a datos que oculta la fuente y los detalles técnicos para recuperar los datos.
- **TransferObject:** Este es un objeto plano que implementa el patrón Data Transfer Object (DTO), el cual sirve para transmitir la información entre el DAO y el Business Service.
- **DataSource:** Representa de forma abstracta la fuente de datos, la cual puede ser una base de datos, Webservices, LDAP, archivos de texto, etc

Patrón de objetos de acceso a datos | DAO



Creación de interface | DAO

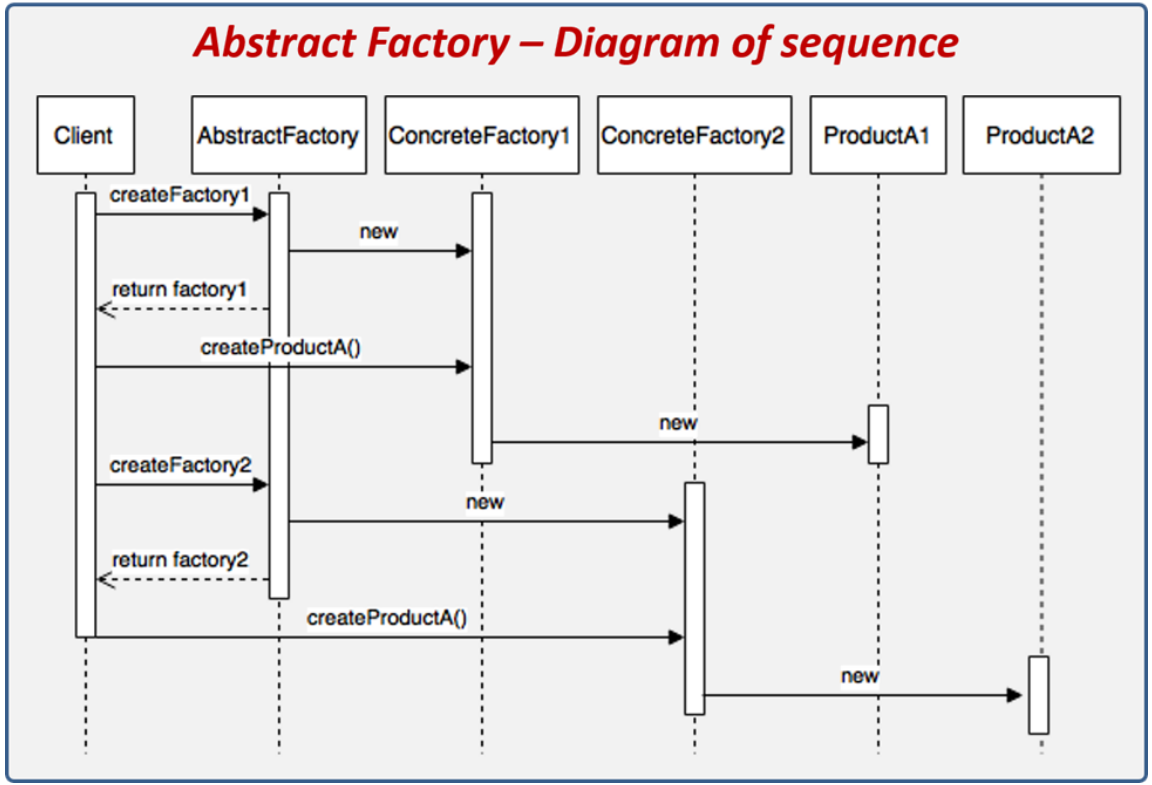


Estructura del Patrón de diseño AbstactFactory

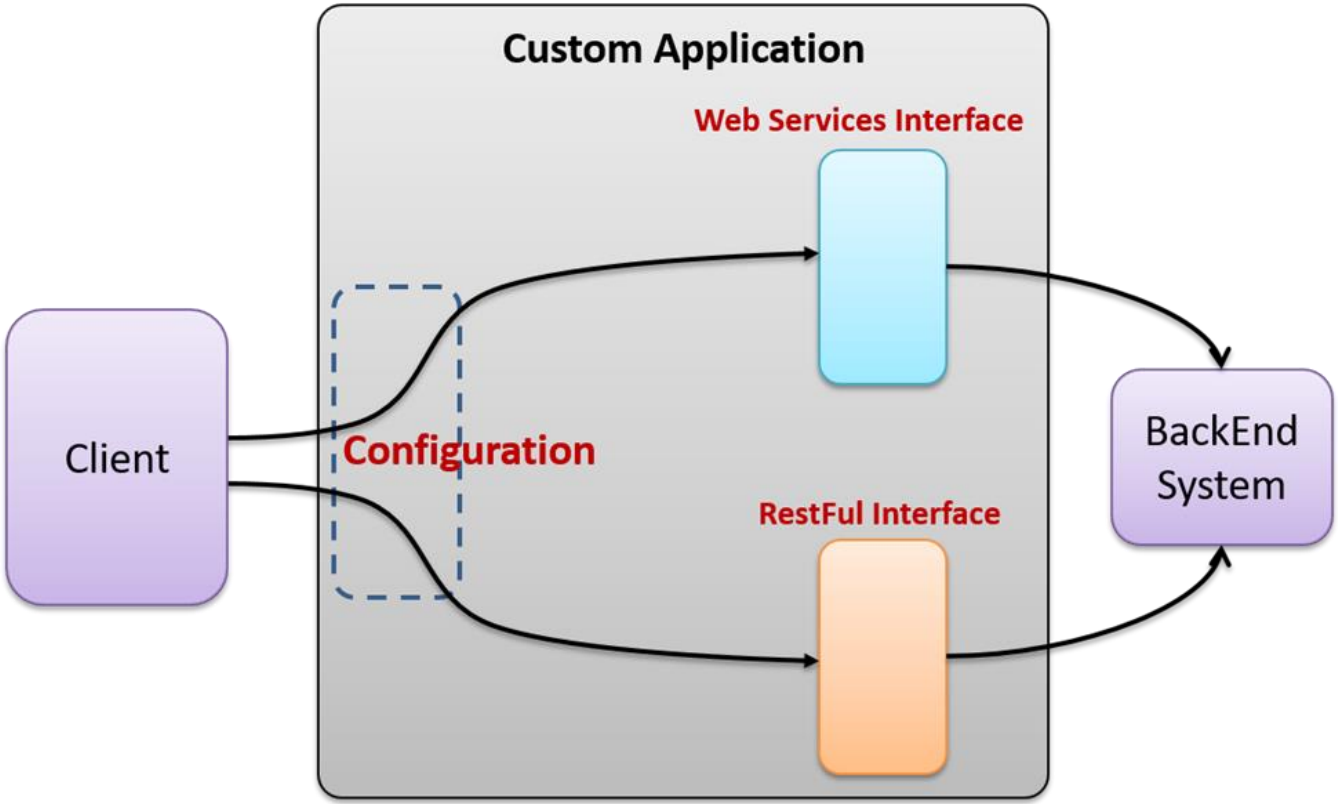
Componentes:

- **Client: Representa la persona o evento que dispara la ejecución del patrón.**
- **AbstractProduct (A, B): Interfaces que definen la estructura de los objetos para crear familias.**
- **ConcreteProduct (A, B): Clases que heredan de AbstractProduct con el fin de implementar familias de objetos concretos.**
- **ConcreteFactory: Representan las fábricas concretas que servirán para crear las instancias de todas las clases de la familia. En esta clase debe existir un método para crear cada una de las clases de la familia.**
- **AbstractFactory: Define la estructura de las fábricas y deben proporcionar un método para cada clase de la familia.**

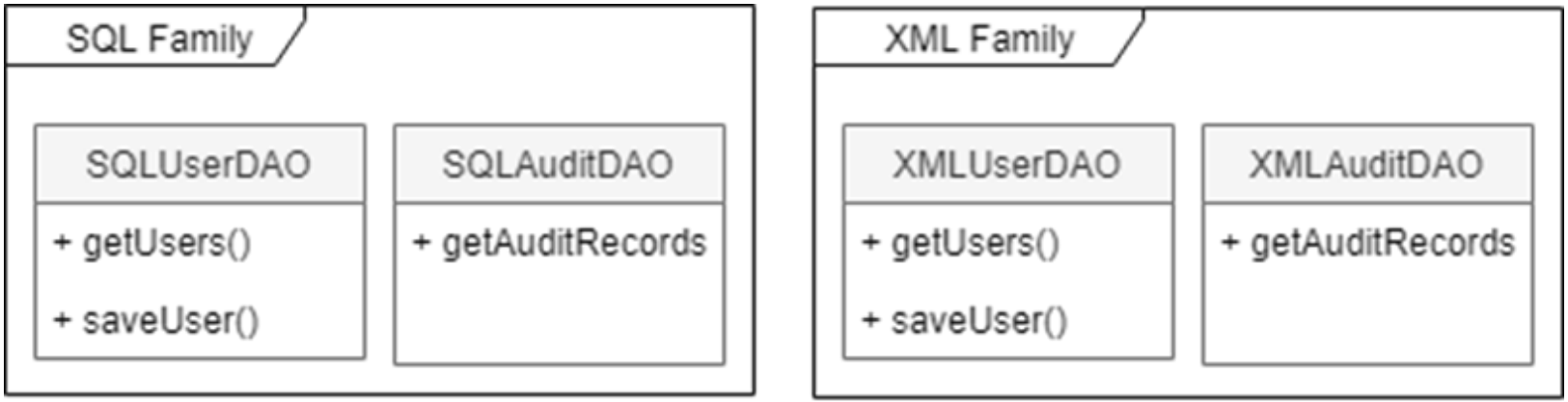
Diagrama de secuencia AbstractFactory



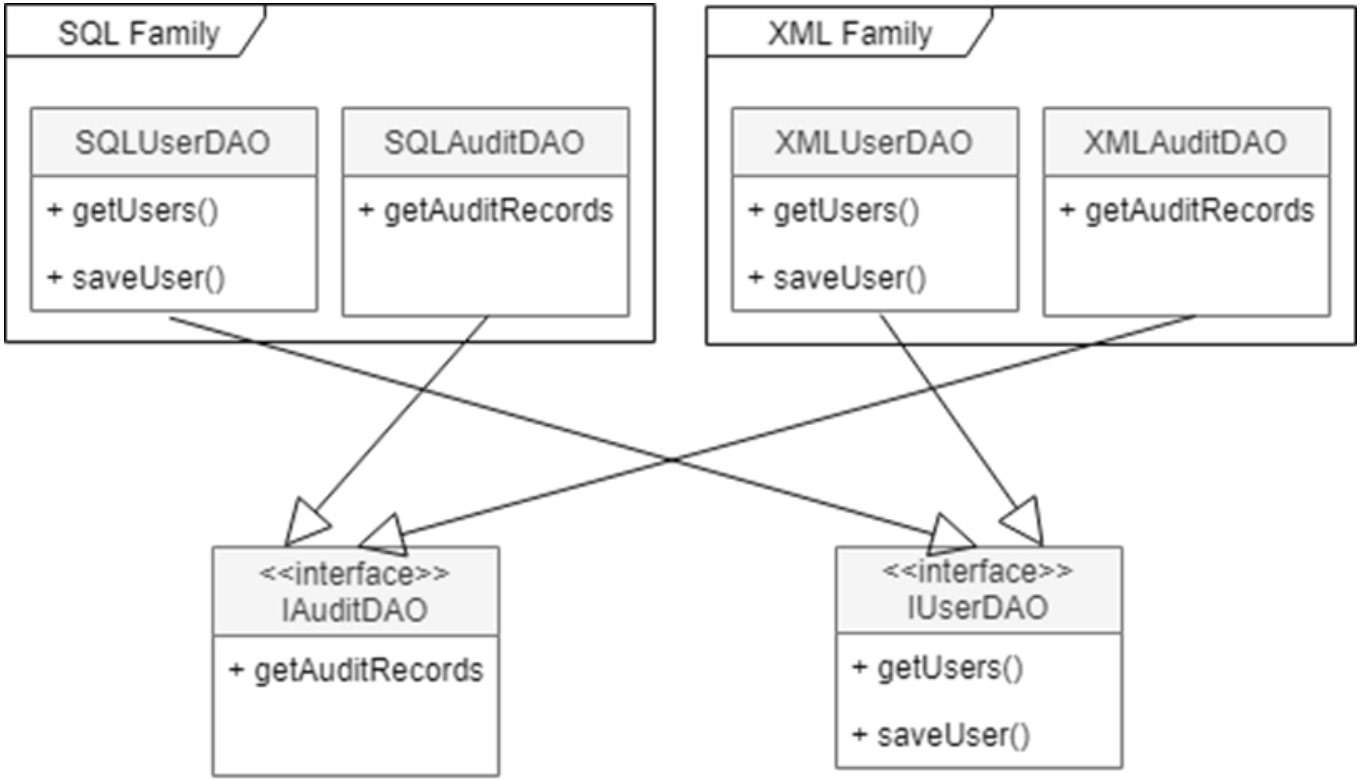
Ejemplo del mundo real | DAO



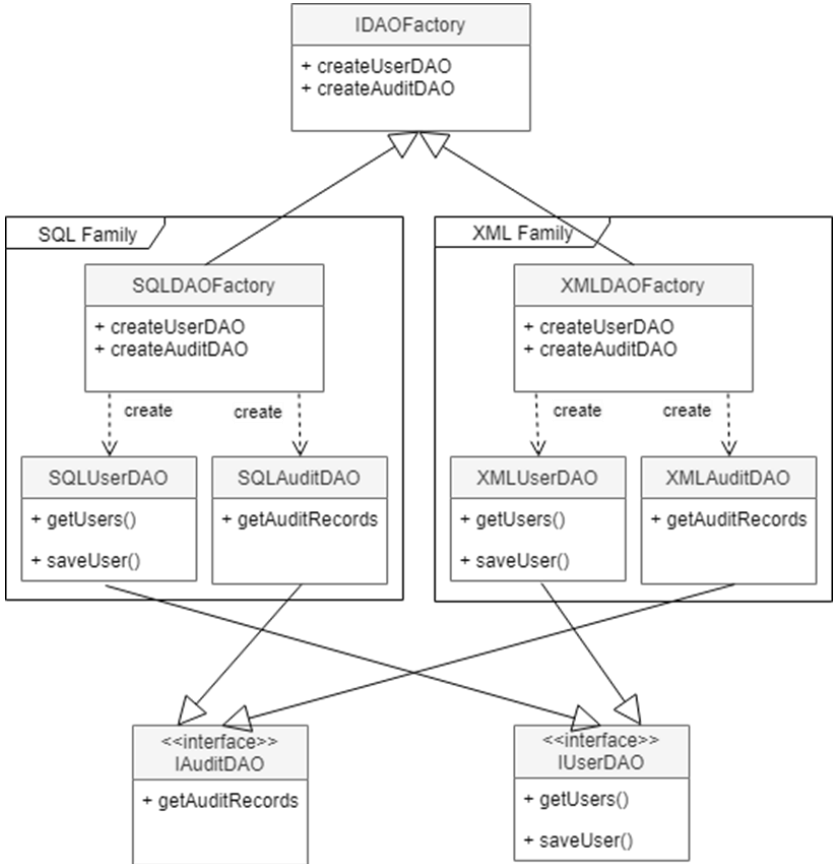
Ejemplo del mundo real | DAO



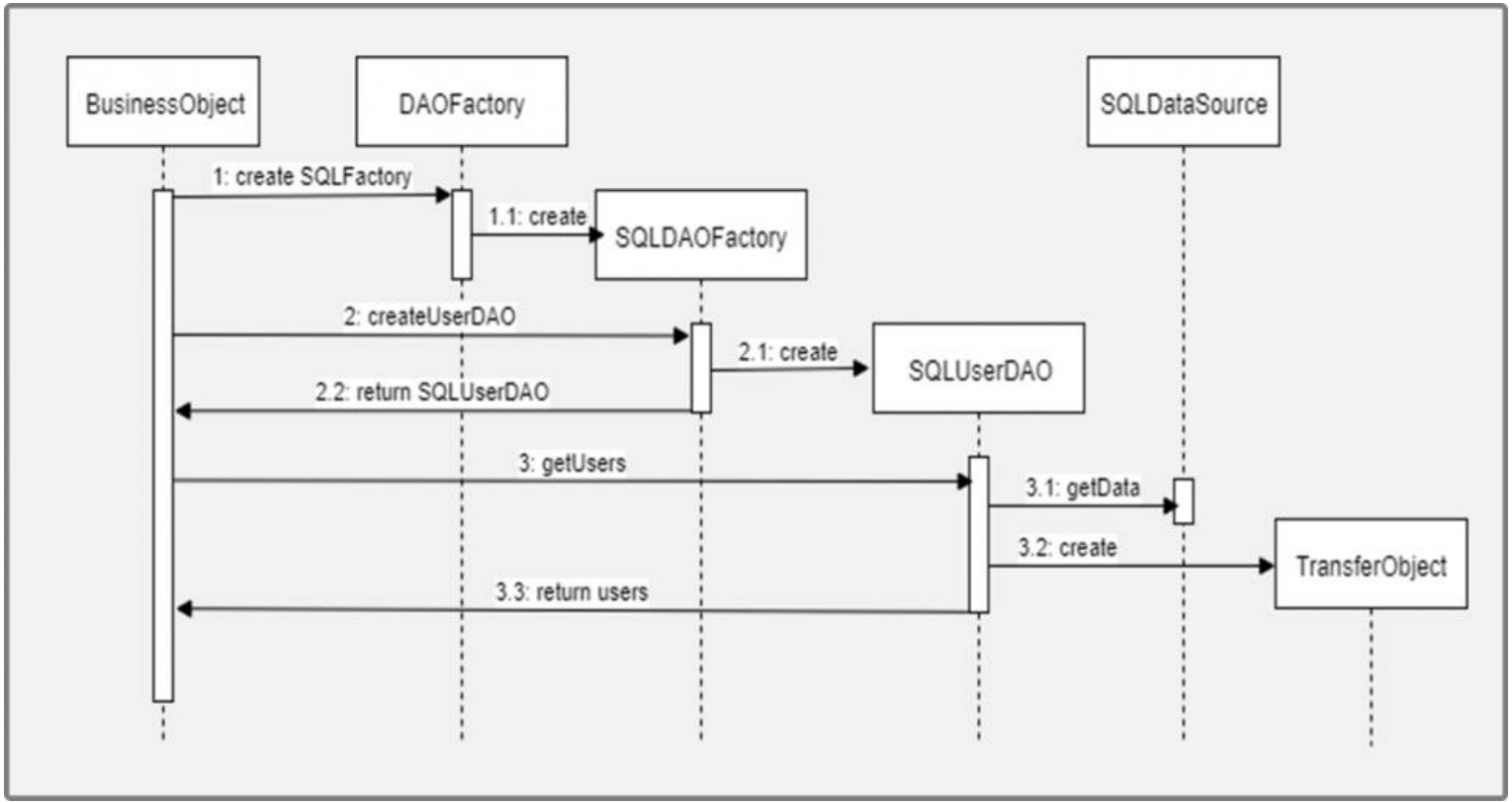
Ejemplo del mundo real | DAO



Ejemplo del mundo real | DAO



Secuencia de ejecución:



Implementación en Java | DAO