Ruth, Raúl, Mario e Ivan Fu

DAM1C  Programación

proyecto web

Contenido

[Diagrama casos de uso 2](#_Toc106118891)

[Arquitectura de la aplicación web 3](#_Toc106118892)

[Modelo 3](#_Toc106118893)

[DAO. FactoryDAO 3](#_Toc106118894)

[FactoryDAO 5](#_Toc106118895)

[SQL FactoryDAO 5](#_Toc106118896)

[Interfaces DAO 7](#_Toc106118897)

[Repository 8](#_Toc106118898)

[Service 9](#_Toc106118899)

[Servlet 9](#_Toc106118900)

[DBConnection 10](#_Toc106118901)

## Diagrama casos de uso

Imagen que contiene Texto

Descripción generada automáticamente

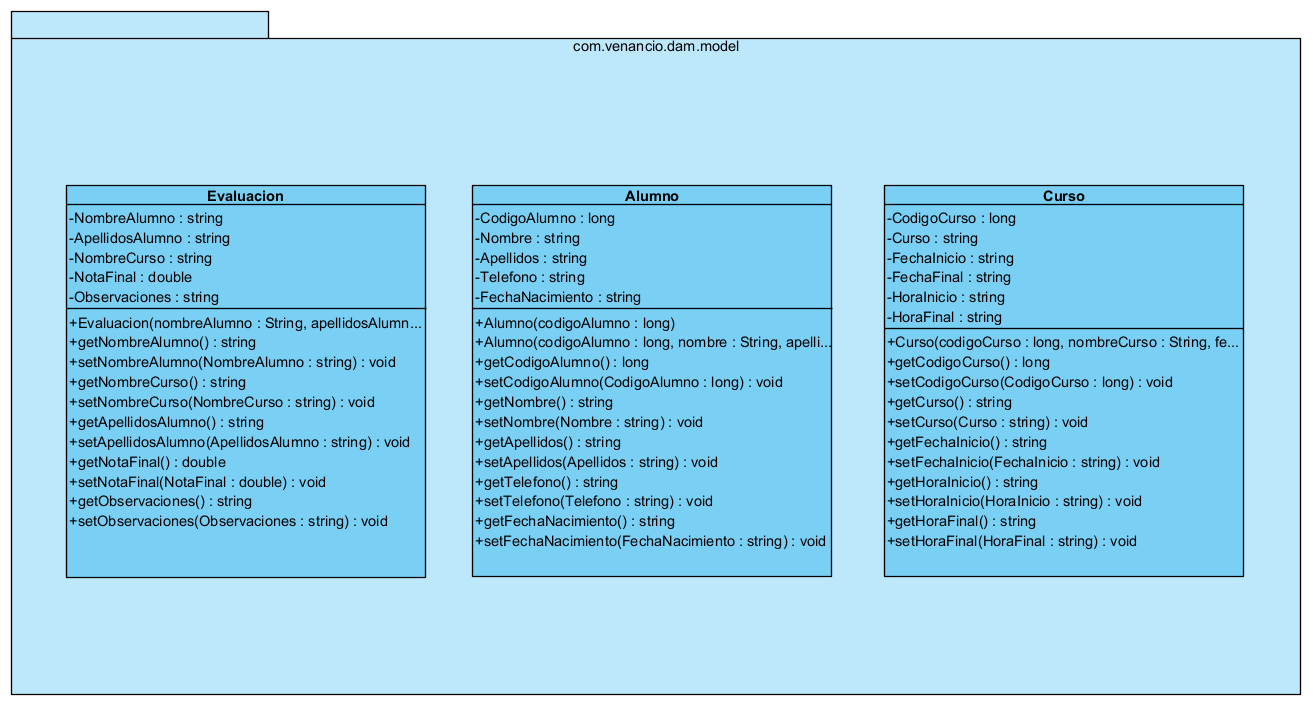
Las funcionalidades consisten en las siguientes partes:

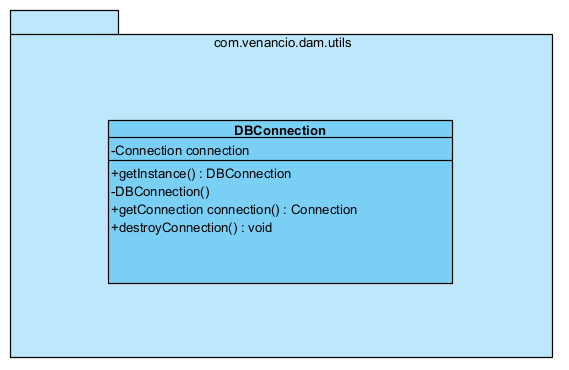
A partir de un usuario se podrán llevar a cabo distintas funciones como visualizar las notas del alumno, se podrá acceder en caso de ser alumno o tutor. También podemos encontrar la función de visualizar el listado de los alumnos pertenecientes al centro.

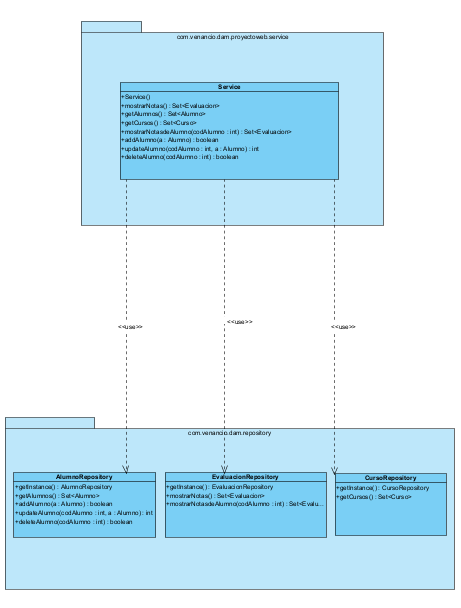
Otra función que podremos llevar a cabo será visualizar los cursos los cuales formarán parte del desarrollo de la distribución de los alumnos. Se podrán observar las notas por trimestre de un alumno especifico, las cuales solo podrán tener uso el propio alumno o el tutor legal del mismo.

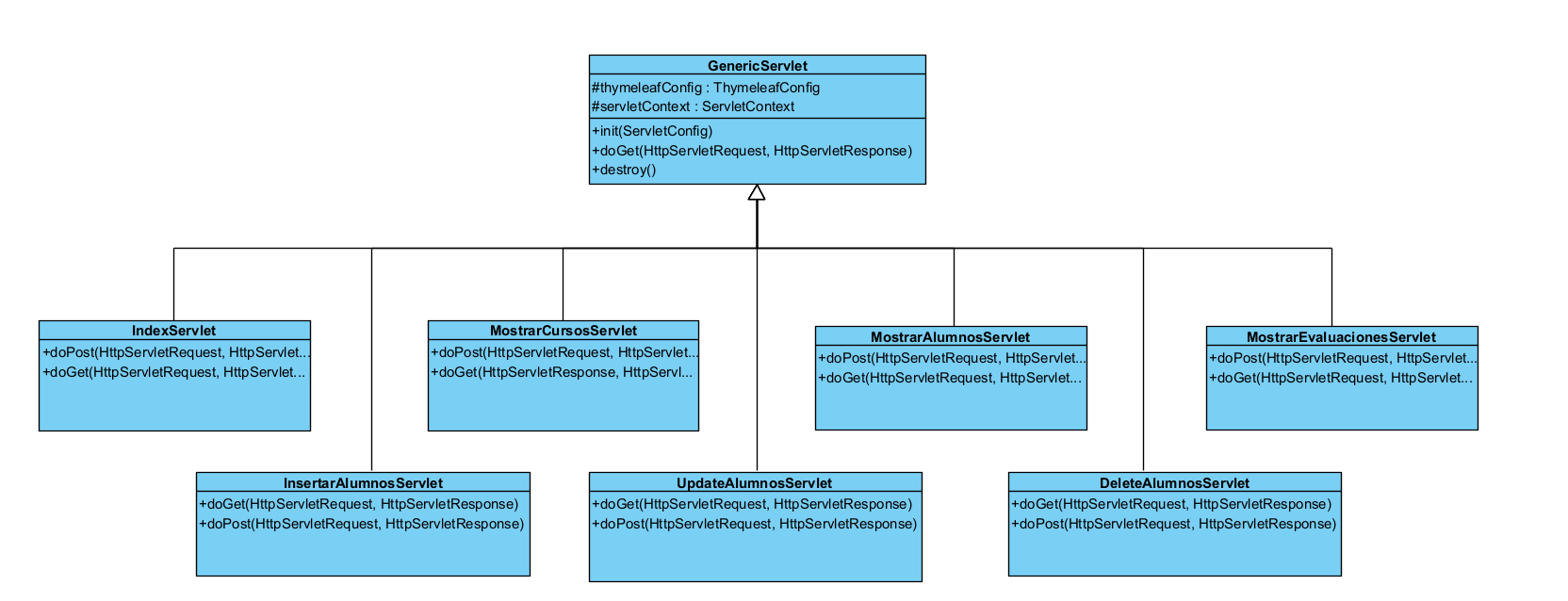
Posteriormente estas funcionalidades conllevan un gran papel como puede ser, tanto añadir alumnos al centro, como puede ser actualizar dicha cuenta a lo largo del curso o finalmente se podría llevar a cabo la eliminación de ese alumno.

## Diagrama de clases





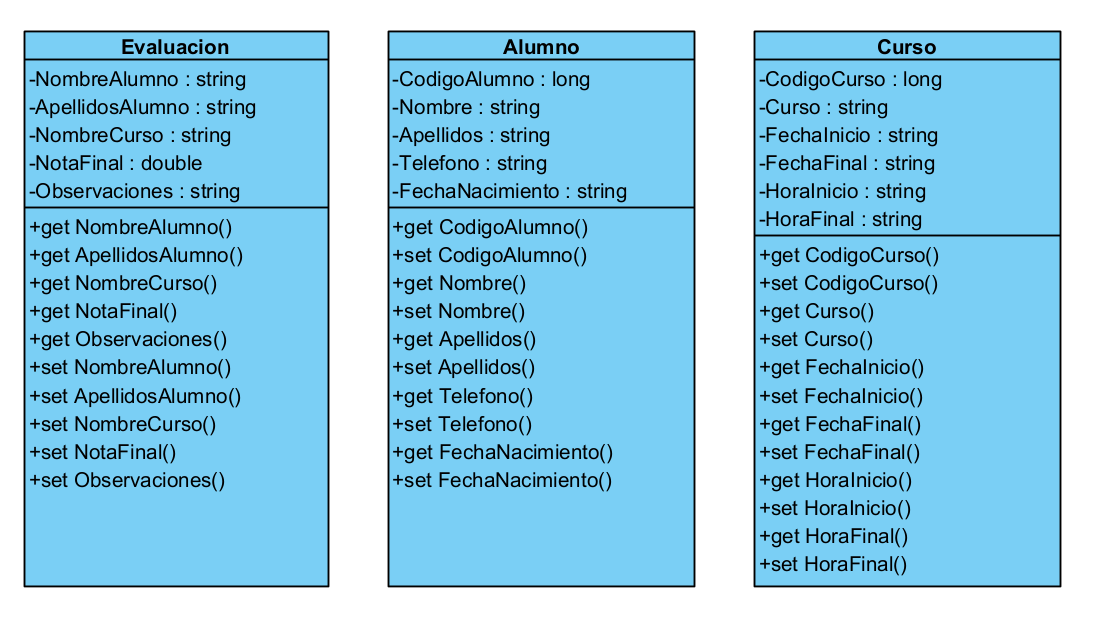




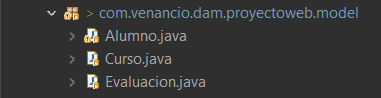
# Arquitectura de la aplicación web

## Modelo

Las clases que componen el modelo serán una representación de los datos que maneja el sistema. En nuestra aplicación, las clases que utilizaremos serán Alumno, Evaluación y Curso, análogas a las tablas que componen la base de datos del sistema, los atributos de las clases no son una representación directa de las tuplas de las tablas de la base de datos, sino que responden a las necesidades funcionales de la aplicación.

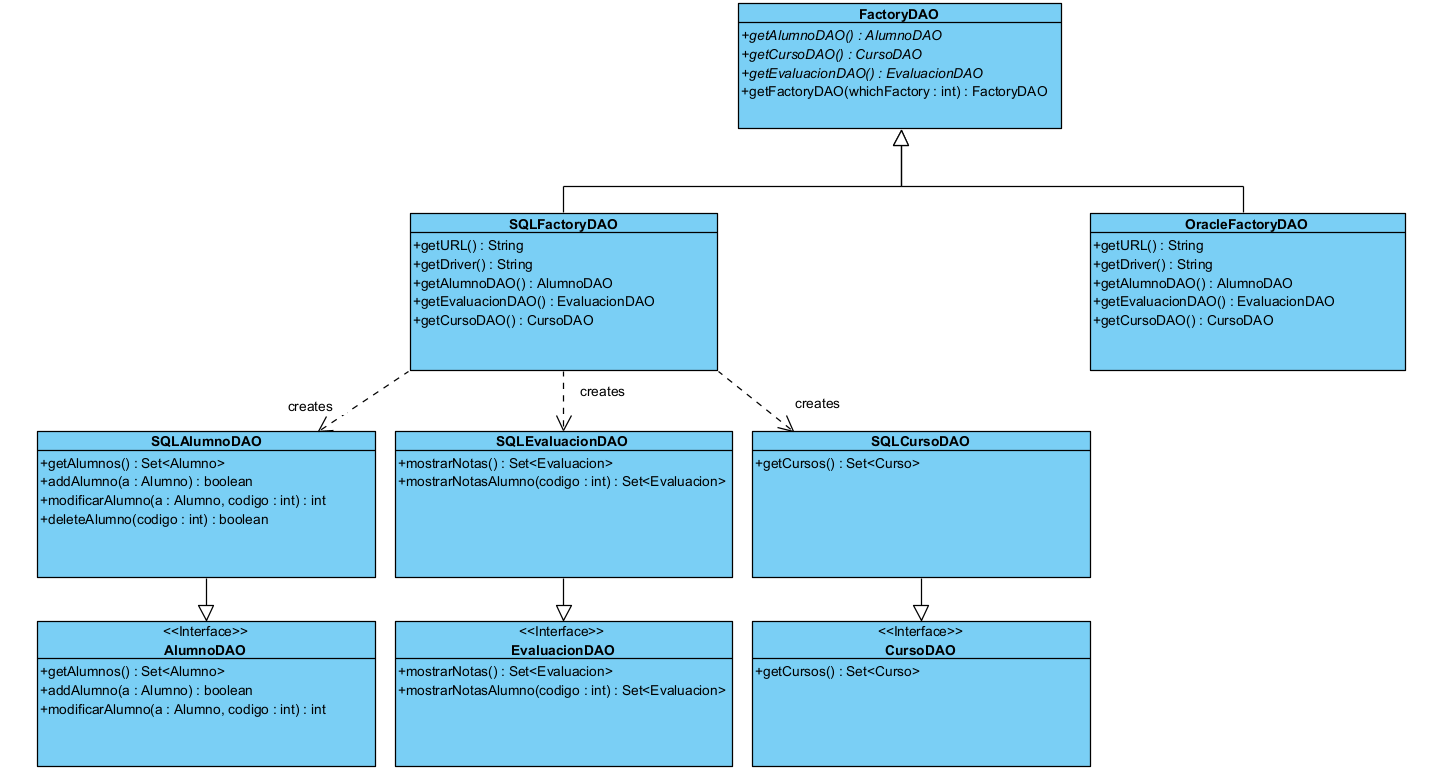


A nivel de programación, generaremos las clases representadas en el diagrama en el paquete modelo.



## DAO. FactoryDAO

Los DAO o Data Access Object gestionan la conexión con la fuente de los datos para obtener y almacenar la información. Con el objetivo de desacoplar el código y facilitar una futura ampliación del programa, se ha implementado el esquema del FactoryDAO.



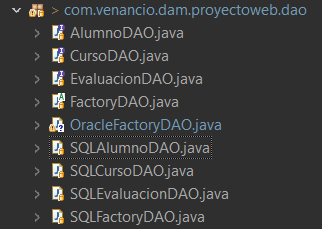
Este esquema está compuesto por una clase abstracta, FactoryDAO, que se encargará de facilitar el DAO apropiado al ser invocada.

Se generará además una clase por cada sistema gestor de base de datos o fuente de datos que se emplee en el programa; estas clases heredarán de la clase abstracta FactoryDAO, heredando la obligación de generar métodos que devuelvan los DAO apropiados, esta vez, los específicos de su sistema gestor. En nuestro programa tendremos dos clases, SQLFactoryDAO y OracleFactoryDAO, si bien solo la primera está operativa y es funcional para el programa.

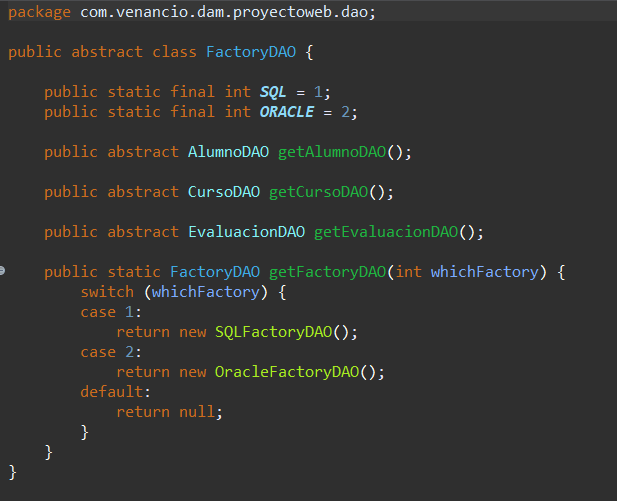
Esta clase además será la encargada de generar la conexión con la base de datos correspondiente, si bien en el caso de nuestro programa, la clase SQLFactoryDAO facilita los datos necesarios a la clase DBConnection para que sea esta última la que genere la conexión.

Cada una de las clases del modelo tendrá su clase de acceso a datos o DAO para cada uno de los sistemas gestores que se manejen en el programa, por ello para facilitar la generación de estos DAO, se implementará una interfaz para cada una de las clases del modelo con los métodos que lanzarán consultas a la BBDD.

A nivel de programación, generaremos las siguientes clases en el paquete DAO:

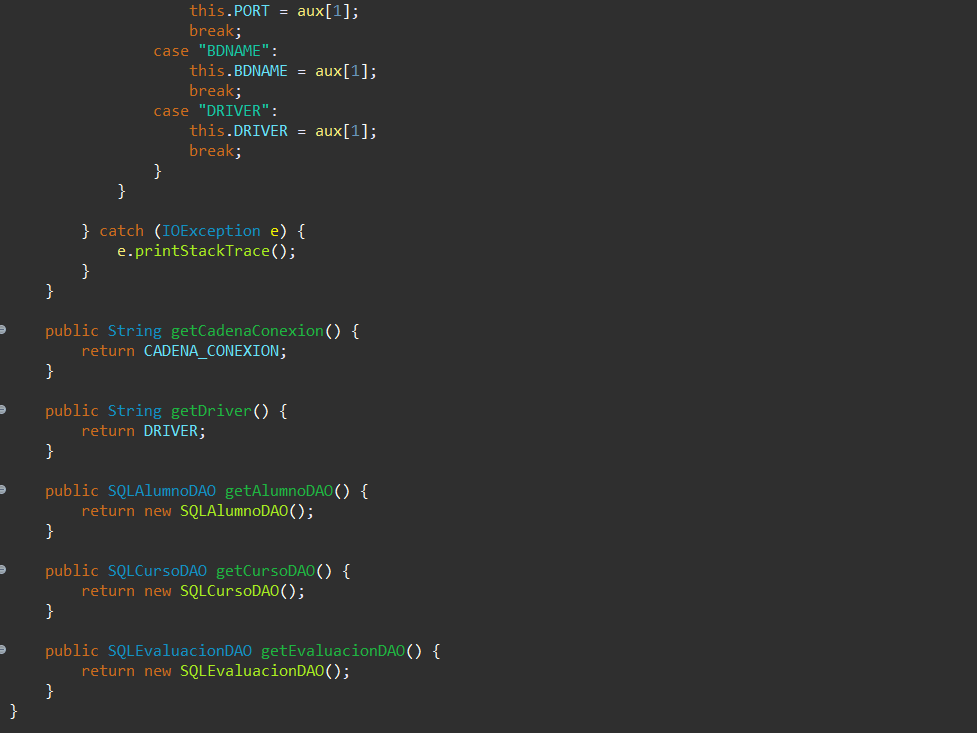
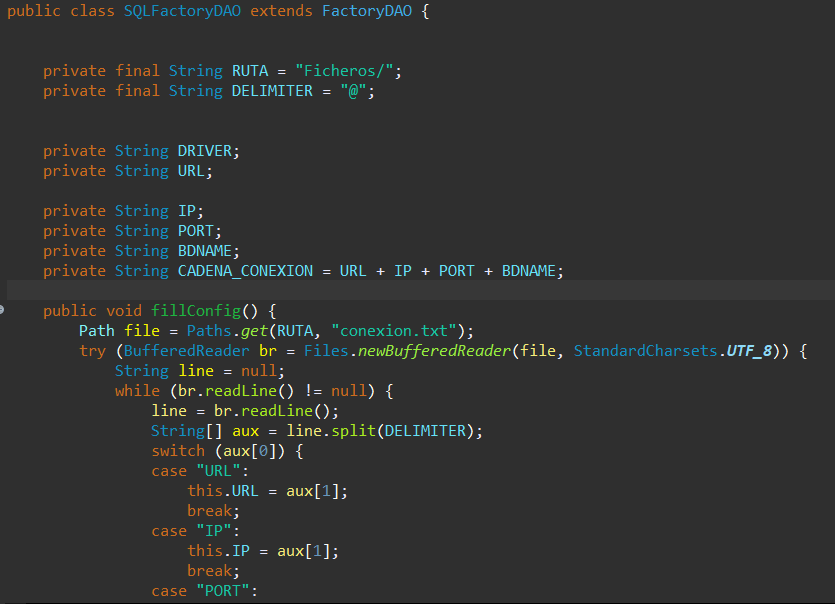


### FactoryDAO

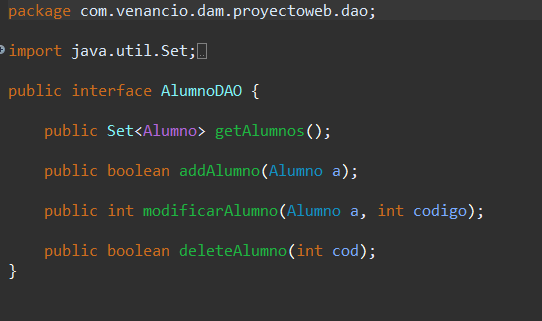


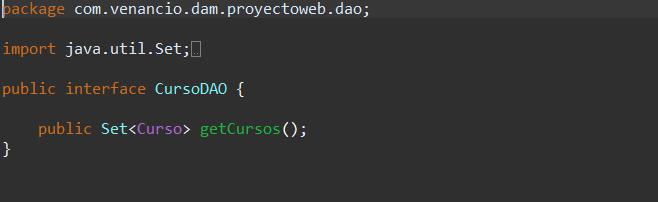
### SQL FactoryDAO

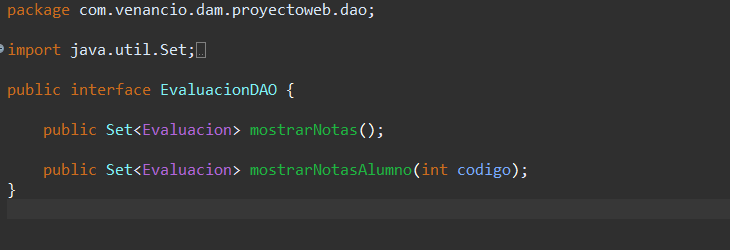
Obtendrá los datos necesarios para realizar la conexión con la base de datos de un fichero de conexión y contará con métodos que devuelvan dicha información al ser invocados (en la clase DBConnection), en adición a los métodos que devuelven los DAO requeridos.



### Interfaces DAO



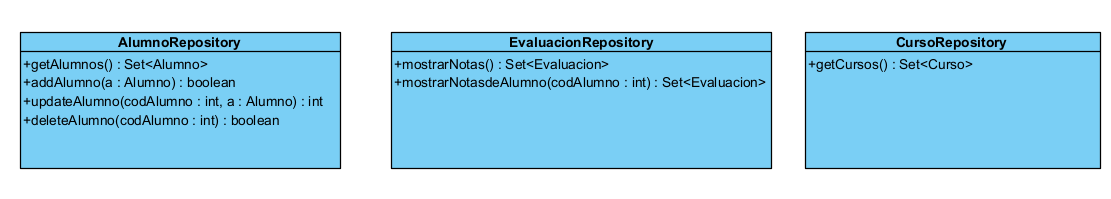




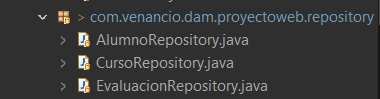
## Repository

El repository es la capa intermedia entre el objecto de acceso a datos o DAO y la capa de negocio. Desde la capa repository llamaremos al FactoryDAO indicando con qué sistema gestor de bases de datos queremos realizar la conexión de manera que este nos facilite el DAO apropiado.

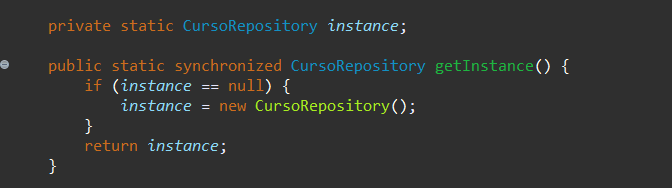
Así, en nuestro programa tendremos tres clases de repositorio correspondientes a las tres clases del modelo, Alumno, Evaluación y Curso.



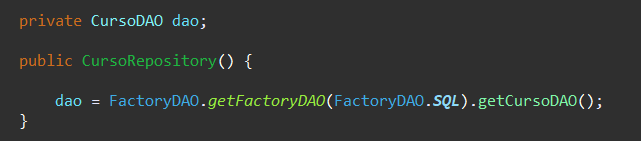
A nivel de programación, se generarán las clases ya explicitadas en el diagrama en el paquete repository:



Los repository tendrán un singleton para generar un único flujo de conexión.

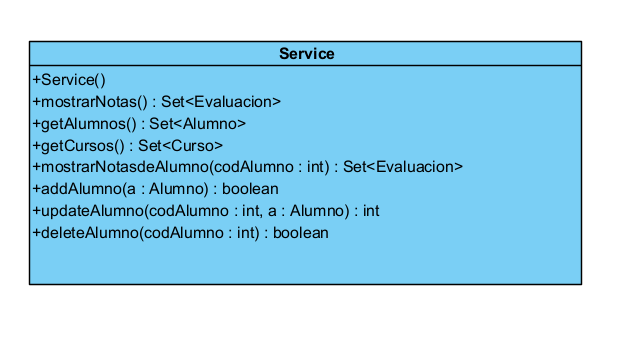


Las llamadas a los DAO, se realizarán a través del FactoryDAO, que proporcionará el DAO correcto.



## Service

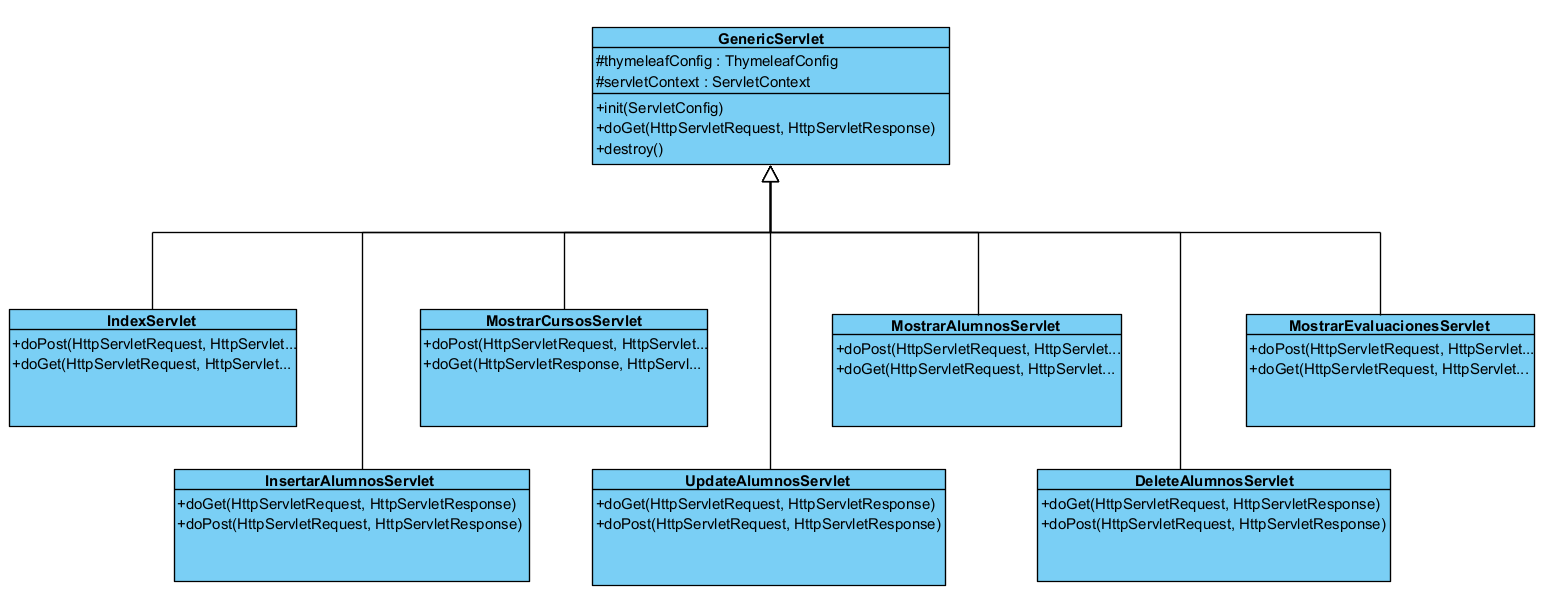
El service o capa de negocio, reunirá toda la lógica de programación. Se encargará de conectar con el servlet y de llamar a las capas intermedias en caso de ser necesario lanzar una consulta a la base de datos.



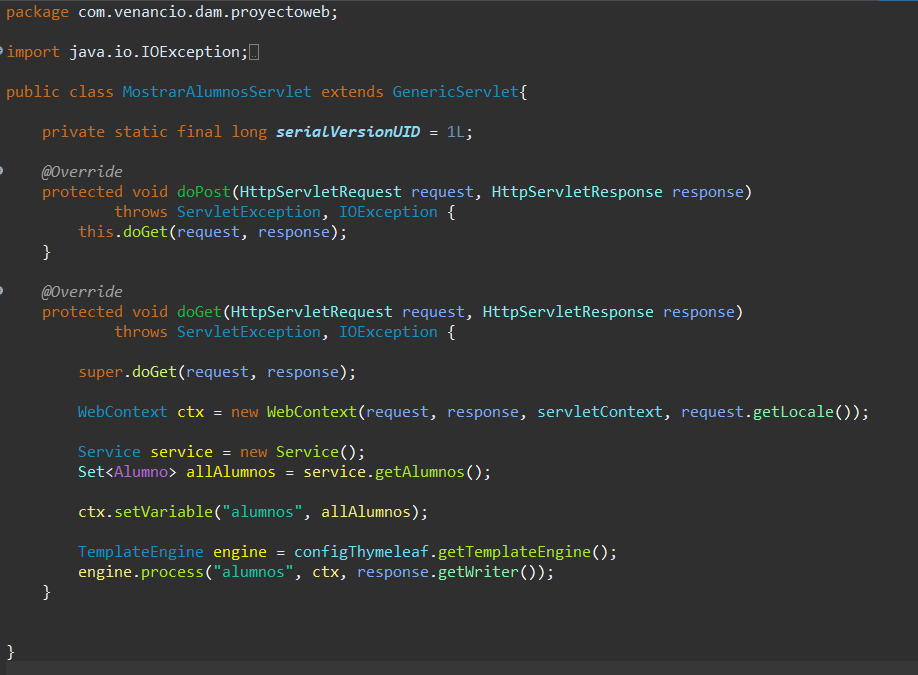
El service de nuestra aplicación recoge los métodos necesarios para llevar a cabo las siete funcionalidades de nuestro programa representadas en el diagrama de casos de uso.

## Servlet

El servlet conectará el service con los documentos HTML y CSS que componen el front-end de la aplicación. Contaremos con un servlet genérico del que heredarán el resto de servlets específicos que creemos para facilitar la generación del código. En nuestro programa, crearemos un servlet para cada una de las siete funcionalidades que ofrece.



A nivel de programación, se generarán las clases previamente mencionadas en el paquete correspondiente.



## DBConnection

Clase de utilidad que genera la conexión con la base de datos.

