Objetivo principal

El objetivo de este proyecto es leer archivos de varios formatos, en este caso implementamos la lectura de archivos Json, txt y csv, luego de leer el archivo la información de este se almacenarán en una Jtable y luego todos los datos de la Jtable se almacenarán en una base de datos y se mostrará en pantalla en un Jtable.

Funcionalidades

Leer archivos: Se tiene la funcionalidad de leer archivos de formato Json, CSV y TXT, primero se abrirá el gestor de archivos de tu computadora y se podrá elegir el archivo de su preferencia.

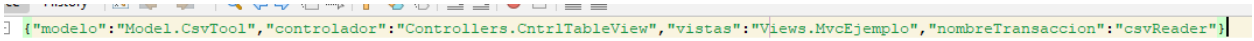
Insertar los datos leidos en JTables: Al elegir el archivo de su preferencia se leerá la información de este y luego los datos se insertarán en una JTable.

Subir los datos a una tabla en una base de datos: Los datos se leen de los archivos se suben a una base de datos automáticamente.

Mostrarlos en pantalla: Los datos leídos se insertarán en una Jtable que se mostrará en la interfaz del framework.

Ejemplo de uso

Primero se configura el mvc mediante un archivo json



Luego se llama a la clase initializer para inicializar el controlador

```
Initializer initializer = new Initializer();
initializer.initClass();
```

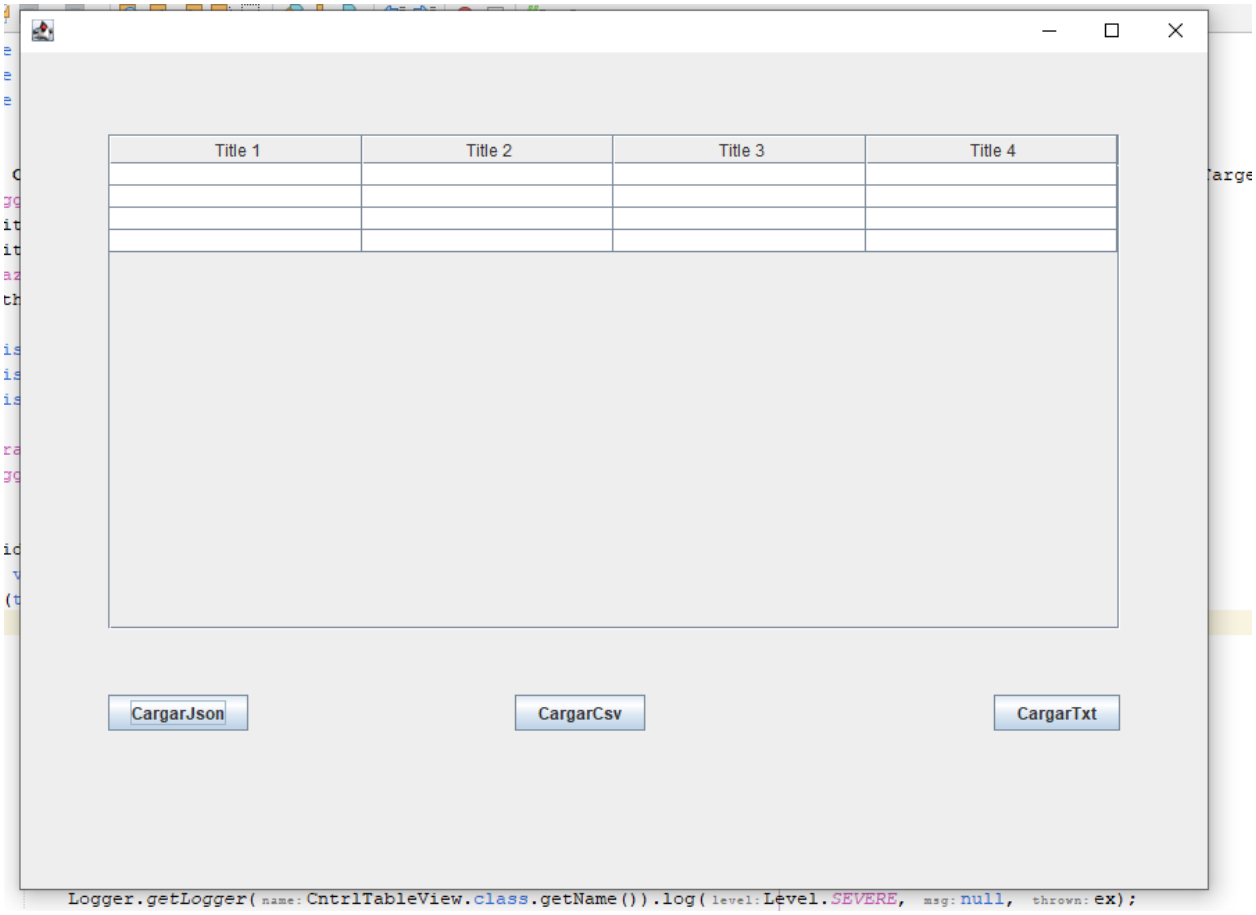
En el controlador asignado llamamos a los métodos initModel e initView para inicializar el modelo y la vista, dichos componentes se leen automáticamente desde el Json de configuración para su inicialización mediante reflect

```
initModel();
initView();
```

Después se trabaja libremente en la clase controlador creada

```
public void actionPerformed(ActionEvent e){
    if(this.button.equals( obj: e.getSource())){
        try {
```

Luego como resultado se abrirá la vista del framework



En este caso, el unico boton con funcionalidad es el de cargar una información a partir del CSV

Descripción de componentes

Nombre: MVC

Descripción

Este componente brinda al framework las bases para la creación de un modelo vista controlador sencillo y configurable a partir de un Json, esto con el objetivo de facilitar al usuario la configuración y relación entre los modelos y las clases, haciendo este proceso más accesible y dinámico a la hora de trabajarlo.

Dependencias con otros componentes

Este componente emplea a la bitácora para elaborar los logs de los errores que podrían surgir al momento de manejar dicho componente y para notificar con logs el éxito de las operaciones realizadas.

- Función que emplea: getTheLogger()

Interfaces de salida

- ***GlobalController:*** Esta interfaz es la base de todos los controladores que el usuario pretenda crear, este inicializa el modelo y la vista con las respectivas funciones initModel() e initView(), los controladores que implementen esta clase van a tener todo el código del mvc.
- ***Initializer:*** Esta clase se emplea para inicializar el controlador que el usuario haya configurado en el json de configuración del mvc, esto lo hace mediante al método initClass().

Interfaces de entrada

- ***Reader***: Esta es la interfaz base de todos los modelos que el usuario pretenda crear, dicha interfaz contiene únicamente el método abstracto documentReader(), el cual se usa para implementar clases que tengan la función de leer y devolver un objeto.

Artefactos

- Documento de configuración en formato Json

```
{"modelo":"Model.CsvTool","controlador":"Controllers.CntrlTableView","vistas":"Views.MvcEjemplo","nombreTransaccion":"csvReader"}
```

- json.jar

Nombre: Bitácora

Descripción
El componente hace que el framework tenga la capacidad de generar logs en archivos de texto (.txt), estos logs que crea son implementados en el código del framework en la función llamada getTheLogger() sin embargo, solo se crea una instancia de este mismo componente y fue implementada con el propósito de notificar al usuario sobre si las funcionalidades han sido ejecutadas exitosa o erróneamente

Dependencia con otros componentes
Este componente no tiene dependencias con otro componente implementado en el framework

Interfaces de salida

Este componente no necesita el uso de interfaces de salida

Interfaces de entrada

Este componente no necesita el uso de interfaces de entrada

Nombre: Archivo de configuración de la base de datos

Descripción
Este componente contiene los datos necesarios para configurar la conexión a una base de datos, a si mismo, lleva un número limitante que configura la cantidad de conexiones que pueden estar dentro de la pool de conexiones, este archivo de configuración lleva un formato json para que la obtención de los datos sea más sencilla.

Dependencia con otros componentes
Este componente no tiene dependencias con otro componente implementado en el framework puesto que puede existir sin la necesidad de ellos

Interfaces de salida

Este componente no necesita el uso de interfaces de salida

Interfaces de entrada

Este componente no necesita el uso de interfaces de entrada

Nombre: Pool de conexiones

Descripción
Este componente se emplea dentro del framework para llevar un mejor control sobre la cantidad de conexiones que se hacen a la base de datos, dicha pool es configurable gracias al archivo de configuración a la base de datos y a su vez, también obtiene los datos necesarios de dicho archivo para realizar la conexión con el método getConnection() .

Dependencia con otros componentes
Este componente solo depende del archivo de configuración a bases de datos, puesto que es de ahí donde saca la información necesaria para hacer la conexión y para configurarse a sí misma.

Interfaces de salida

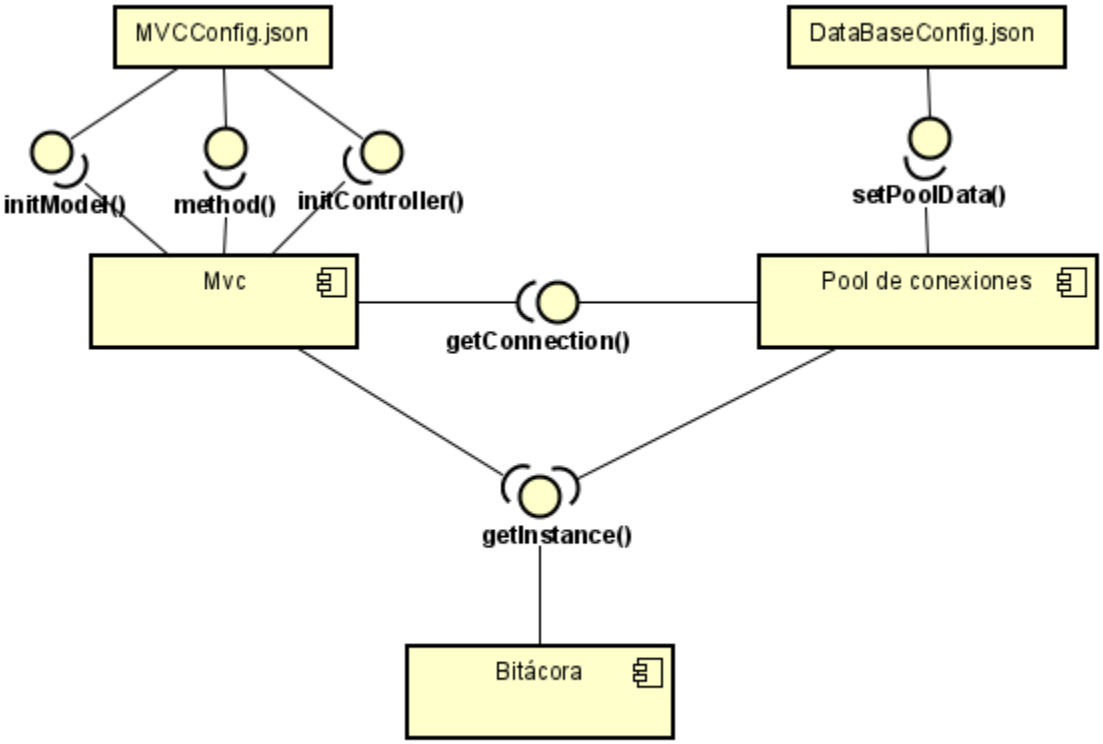
Este componente no necesita el uso de interfaces de salida

Interfaces de entrada

Este componente no necesita el uso de interfaces de entrada

Artefactos
- Commons-dbcp2-2.6.0.jar
- Commons-loggin-1.2.jar
- Commons-pool2-2.6.2.jr
- Mysql-connector-java-8.0.16.jar
- DataBaseConfig.json

Diagrama de componentes



Descripción de Clases

Nombre de clase: GlobalController

Descripción: Interfaz base de los controladores que inicializa el modelo y la vista puestas en el archivo de configuración.

Dependencias de otras clases: Ninguna.

Atributos:

- String modelName
- String viewName
- Object model
- Object view

Funciones:

- initModel()
- initView()
- method()

Diagrama de clases:

Nombre de clase: Reader

Descripción: Interfaz base de los modelos, sirve para darle un método en concreto a los modelos para que los usuarios trabajen.

Dependencias de otras clases: Initializer, puesto que ahí se inicializa esta clase..

Atributos:

- ninguno

Funciones:

- documentReader()

Nombre de clase: ConnectionPool

Descripción: Clase que establece la pool de conexiones a bases de datos y la gestión de las mismas..

Dependencias de otras clases: Ninguna.

Atributos:
- Final String PATH
- String db
- String url
- String user
- String pass
- LoggerManager logger
- ConnectionPool dataSource
- BasicDataSource basicDataSource

Funciones:
- ConnectionPool()
- getInstance()
- getConnection()
- closeConnection()
- setPoolData()

Nombre de clase: Initializer

Descripción: Clase empleada para inicializar el controlador con el que el usuario pretende trabajar

Dependencias de otras clases: Ninguna.

Atributos:
- Object Cntrl
- String CntrlName

Funciones:
- initClass()

Nombre de clase: LoggerManager

Descripción: Clase creada para la bitácora a partir de la generación de logs..

Dependencias de otras clases: Ninguna.

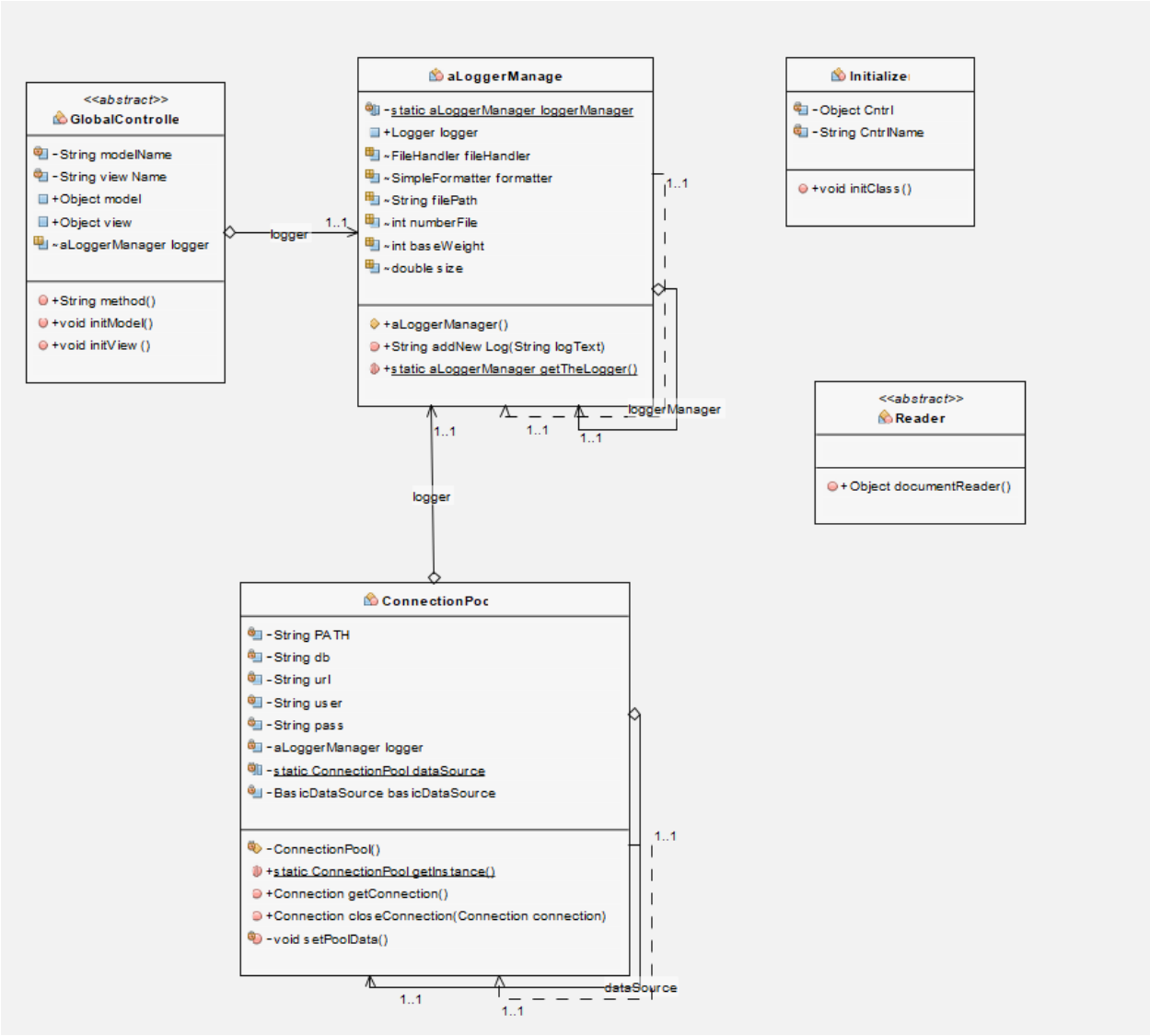
Atributos:

- LoggerManager loggerManager
- Logger logger;
- SimpleFormatter formatter
- String filePath
- Int numberFile

Funciones:

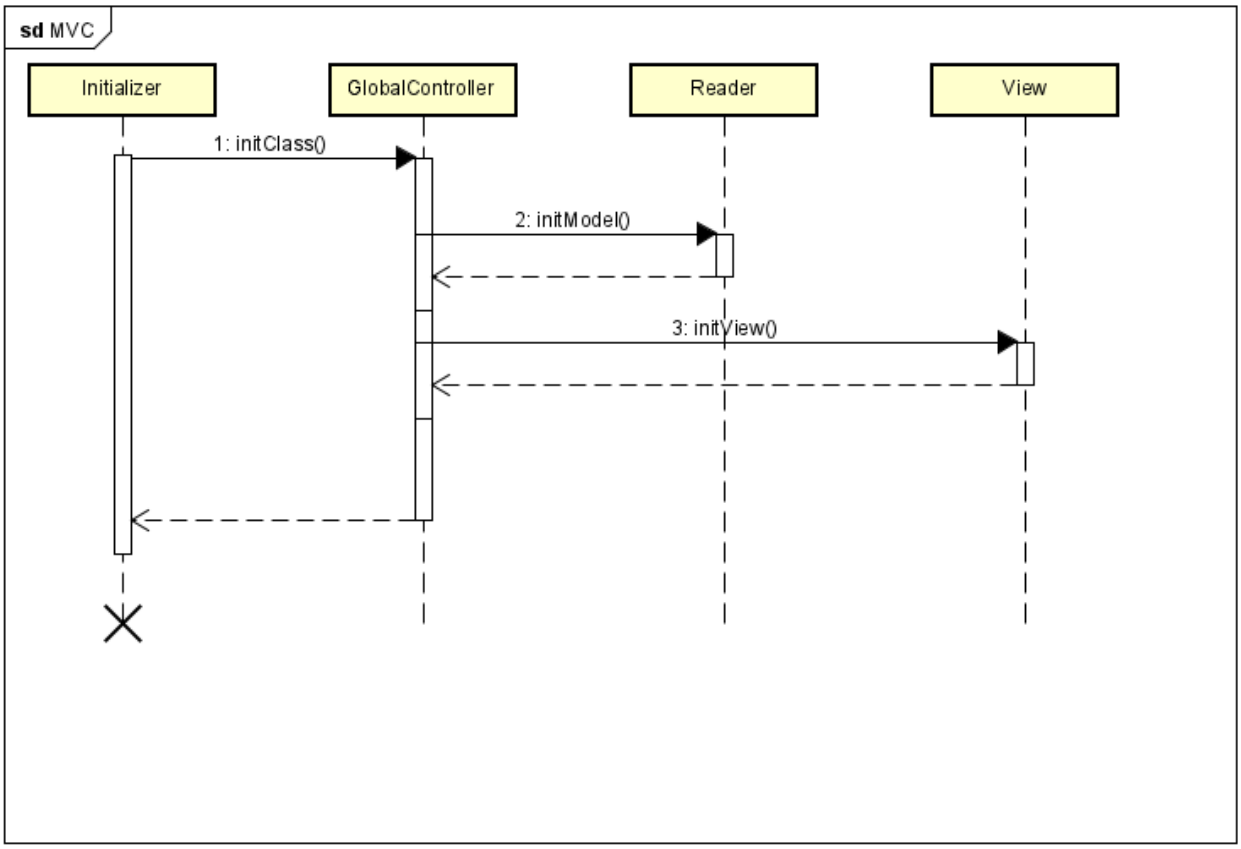
- LoggerManager()
- addNewLog()
- getTheLogger()

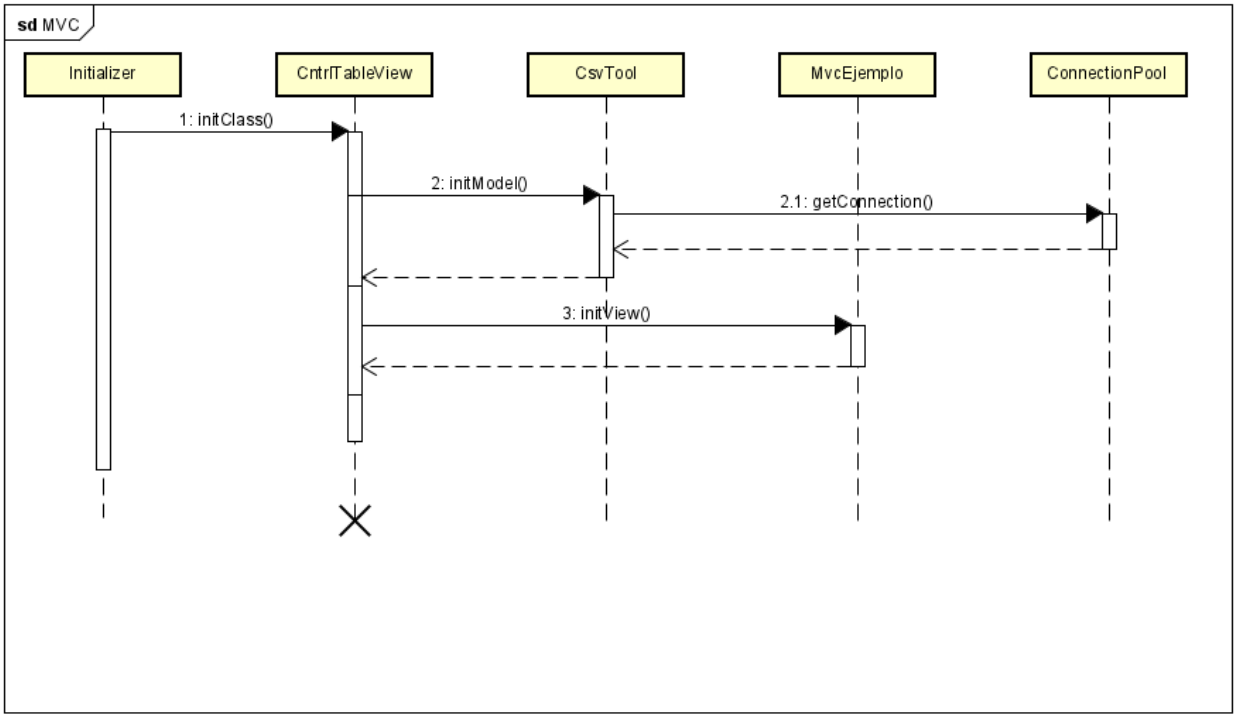
Diagrama de clases



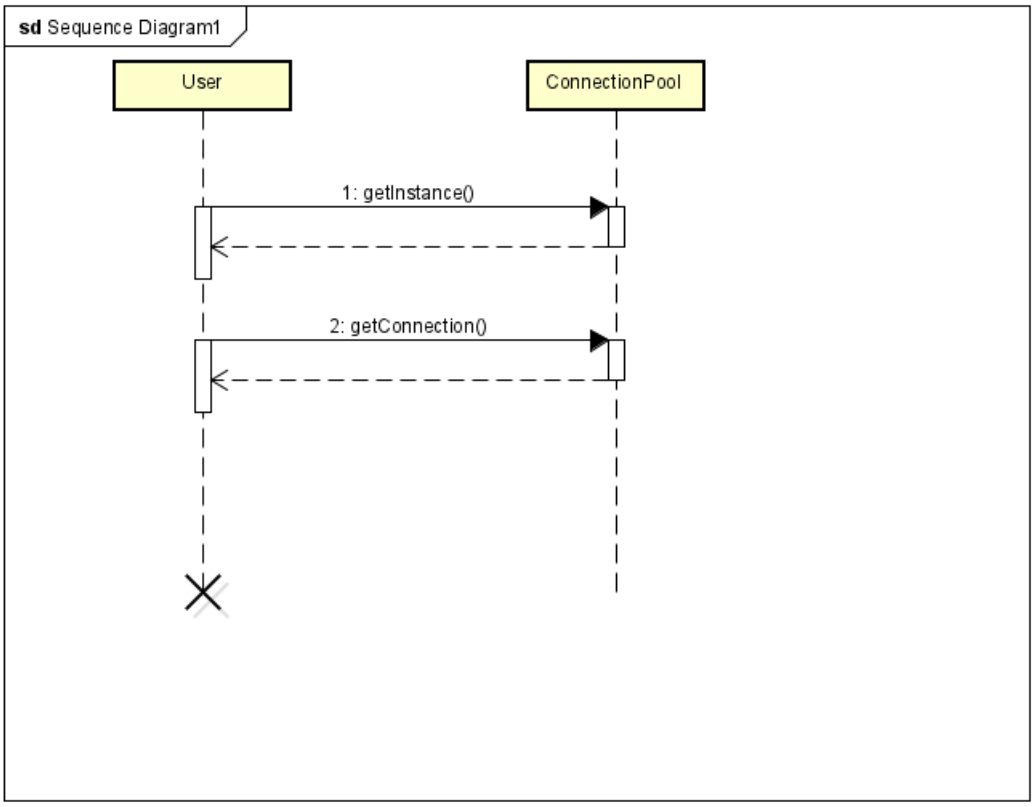
Descripción de las secuencias

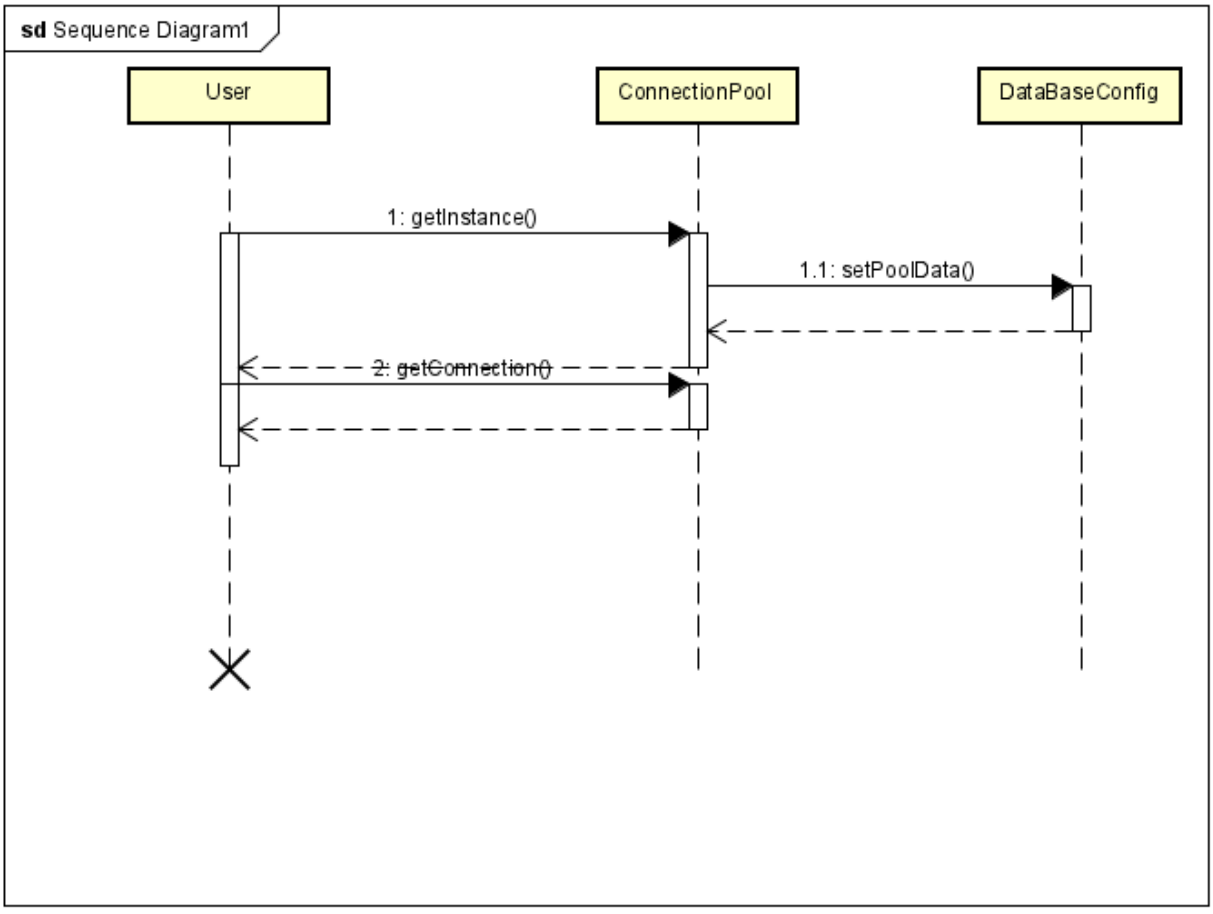
MVC:



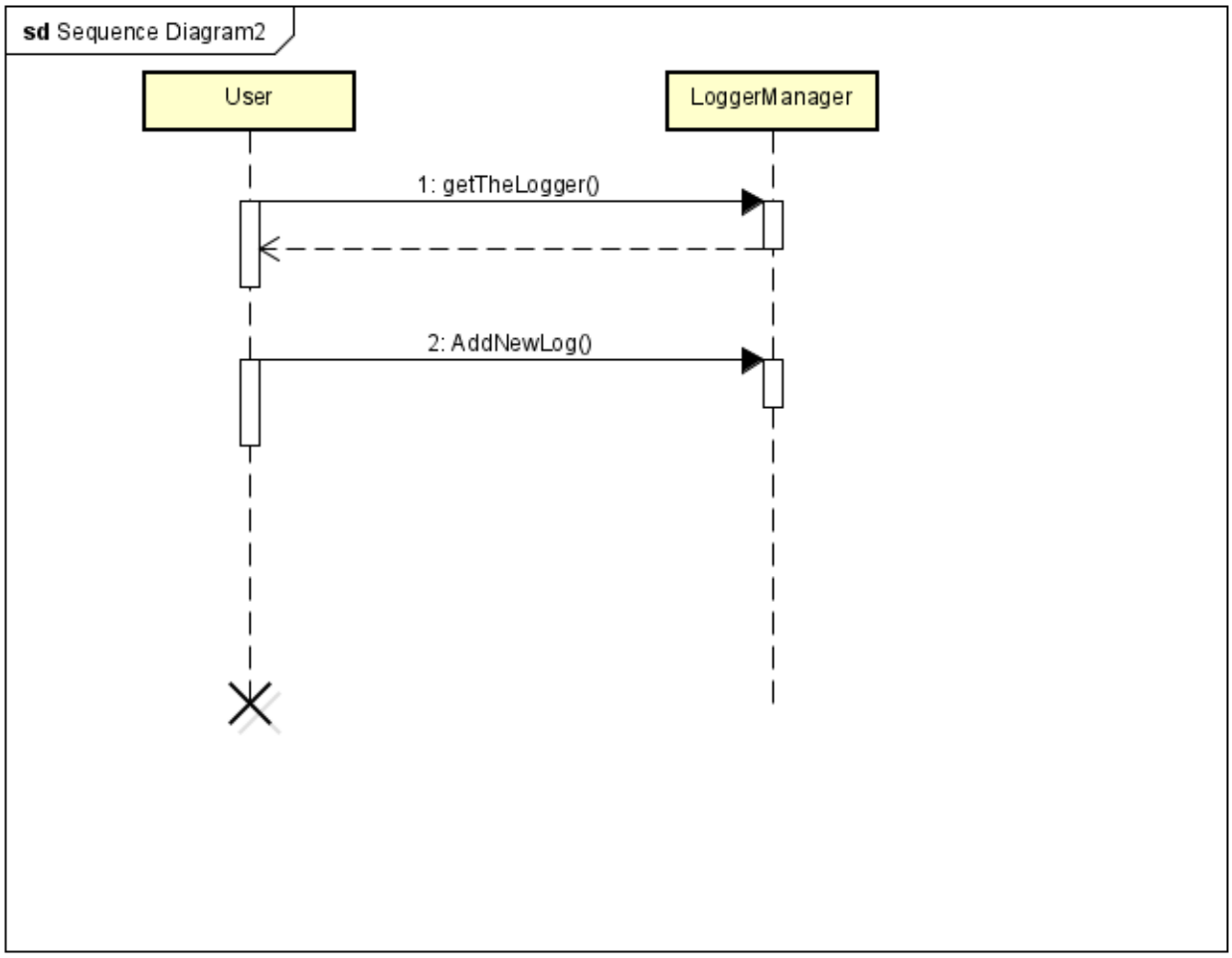


Pool de conexiones

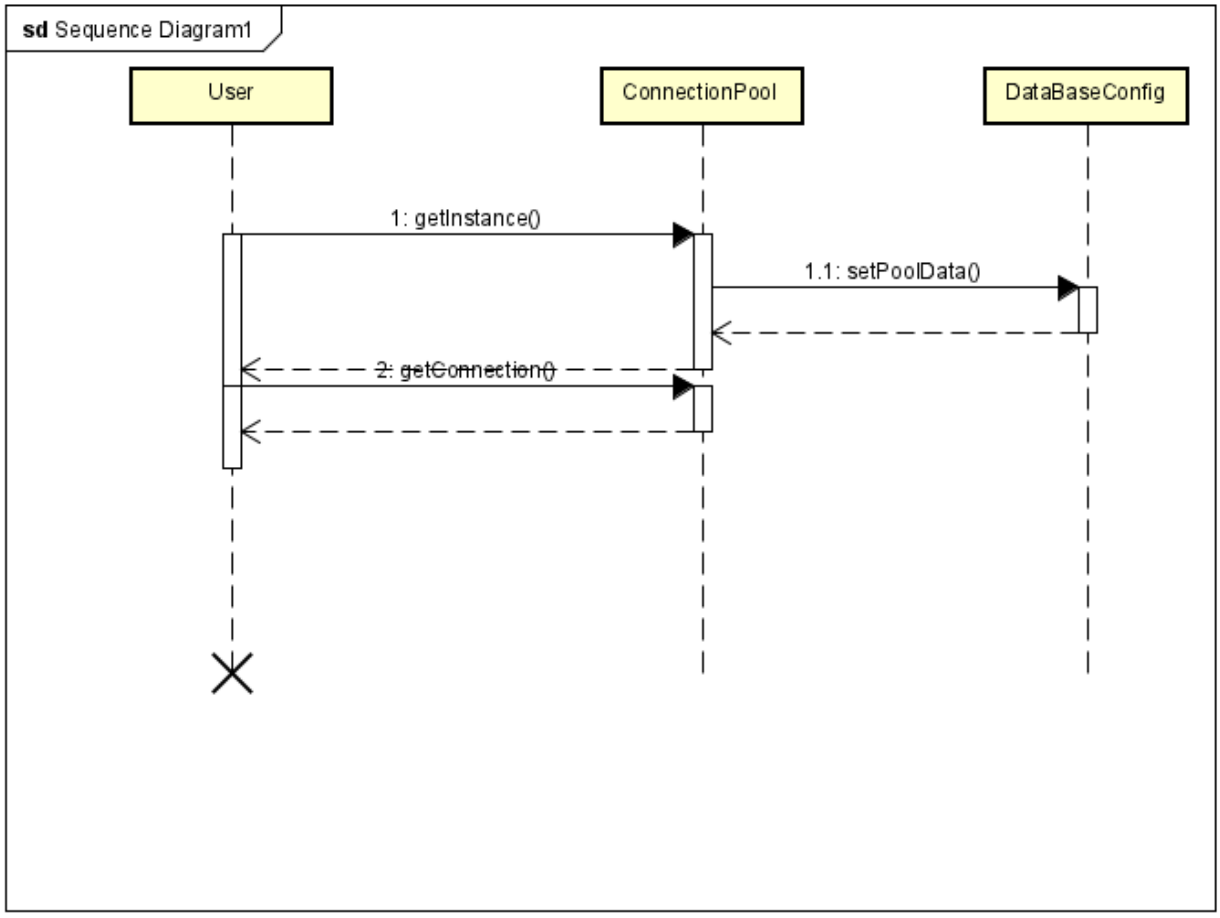




Bitácora


sd Sequence Diagram2
User
LoggerManager
1: getTheLogger()
2: AddNewLog()

Configuración de base de datos



Link del video

https://www.youtube.com/watch?v=9EHGSNyroFQ