Death Note

DEATH NOTE DOCUMENTATION

- This document is a documentation about the project 2DV512 Web 2.0

- It represented the third assignment of this class

- It was realized by the Blue Team :

  - Maxime Gasnier / mg222yz
    Backend Responsible

  - Hugo Herrmann / hh222rg
    Frontend Responsible

TABLE OF CONTENT

TECHNOLOGIES OF THE PROJECT

To fill the requirements of this project, we had to use different technologies. As a website has two sides, a front side which is the visual part in client-side, and a back side where there is all programming logical, we needed to use a lot of different technologies.

Front Side Technologies

In this project we needed to create a dynamic timeline. As all website, languages for the front side are:

- **HTML**, for the structure of pages
- **CSS**, to design pages and animate some elements
- **JavaScript**, to make the website dynamic
- **AJAX**, to send requests

But because we had to draw a timeline, we decided to use another XML based language:

- **SVG**, to draw visual elements in HTML page

Thanks to SVG, we could create visuals elements which are directly connected to our data and on which we could use JavaScript functionalities.

We didn't use any framework for these languages because these technologies were enough evolved to create all what we wanted.
Of course, we know that some of our design choices could be realized with some frameworks of these languages, but we preferred to realized it without to have a smaller project with just what we needed.

Back Side Technologies

In this project, we had to use Java on the server side. To increase our productivity, we decided to use:

- Spring MVC
- JSP (Java Server Page)
- Hibernate
- Maven
- MySQL Database

Spring MVC is a framework based on Model – View – Controller padding, which is used to create a web project with Java. It made easier the development of our website.

JSP is a technology for controlling the content or appearance of webpages. It's the technology that we used to bind data between Web pages on the client side, and database on the server side. It's the *View part* of our MVC Project, the link between the front side, and the back side.

Hibernate is an object-relational tool for Java. It's a framework for mapping object-oriented domain to a relational database. Thanks to this tool, we could improve our databinding between our MySQL database, and our Java classes.

Maven is a build automation tool used for Java projects. It describes how software is built, and its dependencies.

MySQL Database is a relational database management system, used to store data, and manage it. It's the *Model Part* of our MVC Project.

RATIONALE OF CHOICES

Front Side Choices

If we decided to didn't use Frameworks as examples Angular, JQuery, Bootstrap, LESS, or SASS, it's because of some reasons.

Firstly, we are developers and we love to develop our own work. We know that to use library of others is a very good practice, because we use something that it already exists, so we can stay focus on what we have to create. But after lost time to find how we could create our timeline with someone's library, we finally decide to create our own Timeline Creator: we will be sure that the timeline will answer to our requirements.

Secondly, HTML5 - CSS3 – JavaScript ES6 are very easy to use. Create its own components are sometimes a time saving than to find and understand components of others.

Finally, we wanted to have a small project. Use a lot of frameworks increases the weight of a web project.

Back Side Choices

As we mentioned it in previous page, all our choices for the back side were taken to simplify the development of our Website. Spring MVC is a complex and powerful framework, so we wanted to use it to have a good project. But sometimes, it was a little bit too complex, that's why we used several tools as Hibernate or Maven to make it easier.

TECHNOLOGICAL DOCUMENTATION

Timeline-Creator

Our timeline is a SVG element directly inserted in our HTML code. But this element can't be created before access to the server: indeed, it needs to check data to create a timeline ordered with good events.

Once we have all events as Java objects, we use a reader function to convert all this data in **NodeObject** (a JavaScript Object) which will have all information of the event and fields for SVG :

```
function NodeObject(eventModel) {

    // [FIELDS]
    this.onNewBranch = false;   // boolean to check if the node is on a intersection of two lines
    this.previous = null;       // previous NodeObject
    this.borderLeft = null;     // border left of the Node in pixel
    this.positionX = null;      // position X of the Node in pixel
    this.positionY = null;      // position Y of the Node in pixel
    this.borderRight = null;    // border rieght of the Node in pixel
    this.next = null;           // next NodeObject
```

After, when we are drawing all NodeObjects created, we add each to the **LineObject** which is corresponding to its category. A LineObject is basically an array of NodeObjects. We group all NodeObjects in different LineObjects, because our timeline has several lines.

We use after a **SVGDrawerObject** which creates all SVG elements thanks to NodeObject and LineObject. Its functions allow us to draw timeline:

- **drawNode()**, to draw a node;
- **goToMaster()**, to draw a line to reach the main line
- **createBranchFor()**, to draw a new line from main line

Finally, we just have to use a global program to determinate when should we use each function on SVGDrawerObject.

Project Organization

The project is organized with a basic Spring MVC architecture. We have linked the database with Hibernate, this allows us to play easily with data (Execute some CRUD).

In our project we use MAVEN, to simplify the uses of dependencies.

So we have this organizations:

We use the patern MVC to build the project. So in this section we will speak about the Back-End side of the project.

We have chosen to use Spring MVC with Hibernate because datas are easy data-binded. Indeed by simply http calls we can make some CRUD to the database and control the datas thanks to some Services implemented.

HOW TO INSTALL IT

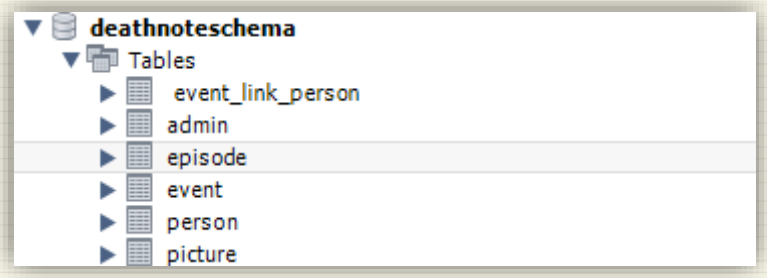
First Step: DOWNLOAD

- Download Eclipse Neon.1 and Java EE with it. (don't forget Maven)
- Download Wildfly or Tomcat 7.0 (version of the server used on our project)
- Download the project through github/get the project in another way.
- Download Spring (if you don't have spring Jars are maybe already included in the project)
- Download MySQL Workbench.

Second Step : START ENVIRONMENT

Once you have the project, download the database attached to the project.

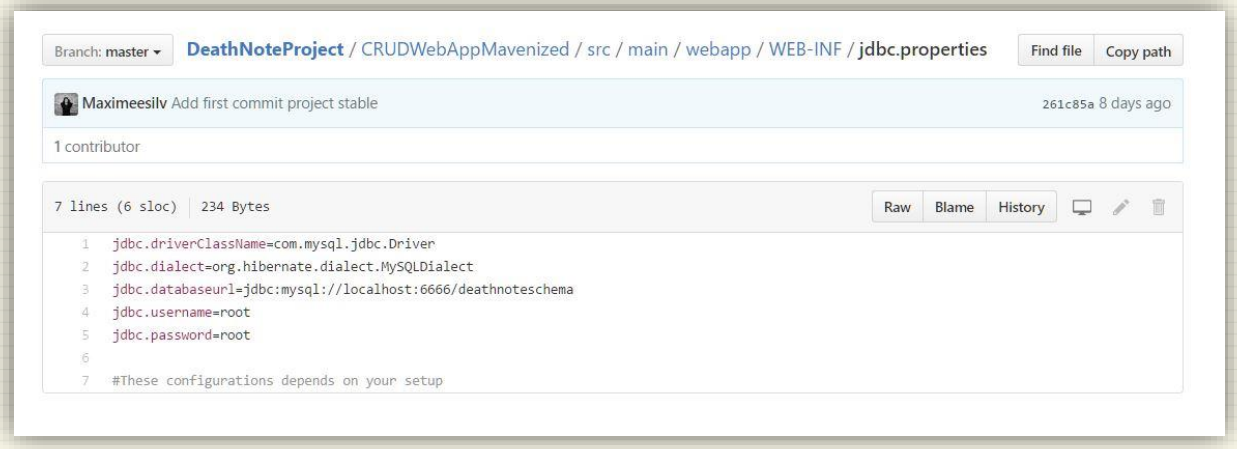
Once you have download it, IMPORT it to your Workbench and make sure that you have the following tables:



Start your server with workbench.

- Next, open Eclipse, and click on import existing project. Select the project downloaded (in which the pom.xml is located)

Once this is done, clean the project once, and go into your build path configuration to take a Maven project dependencies. Add all JARS.



- Now you have to configure the access to the database into the file:

You just have to enter the information of your configuration server (Workbench).

Third Step

Once all steps are done make sure all dependencies are well downloaded, make sure that you have an internet connection, and launch the app.

- To launch it click on "Run configurations" and right in the Goals input: tomcat7:run.
- Or to launch it you can directly try to download a Tomcat Server 7.0 and launch the app with it. (Apache TomCat v7.0)

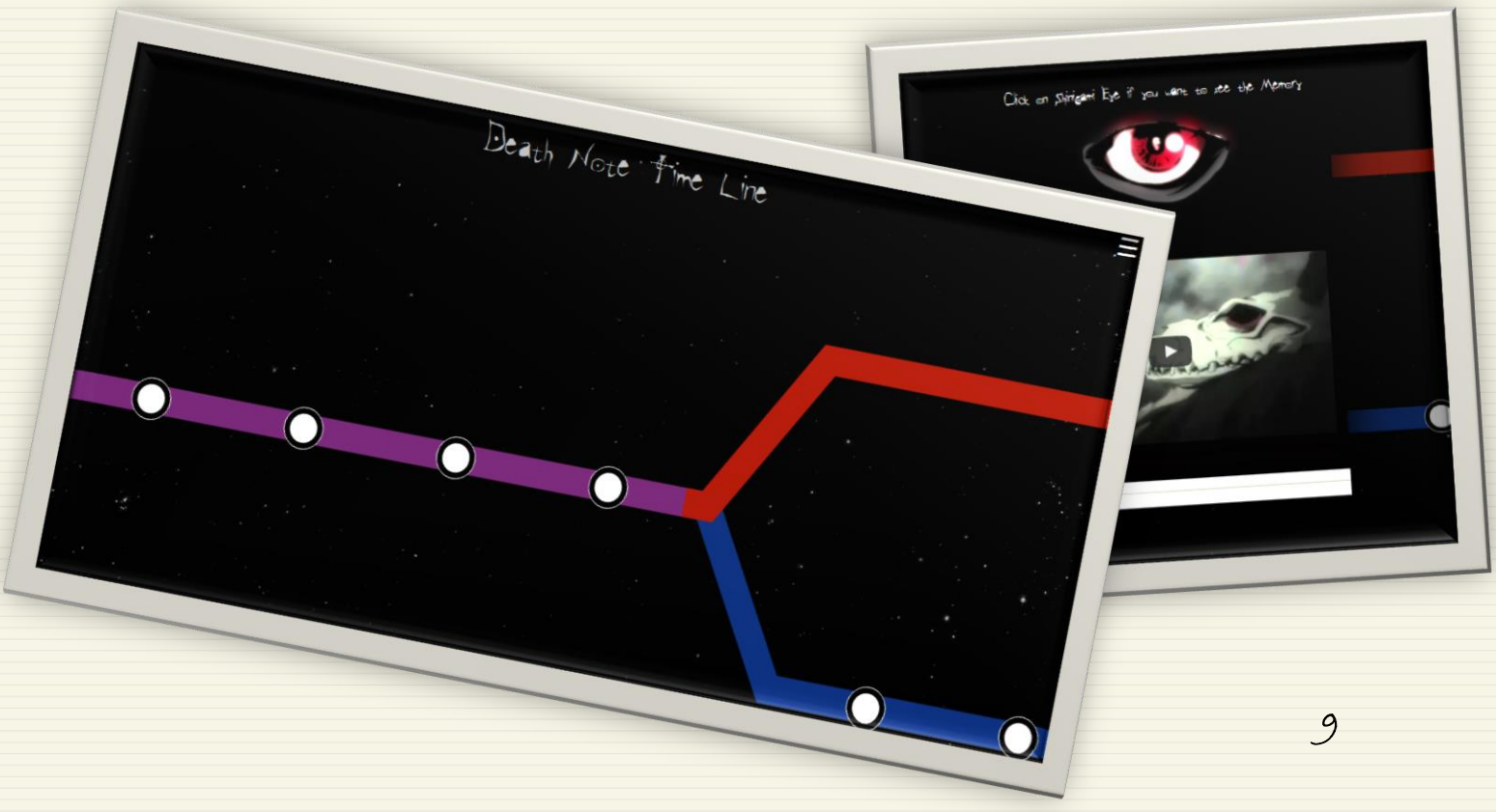
Open Chrome (because the project has been especially made with it) and just indicate http://localhost:8080/CRUDWebAppMavenized/loadEvents

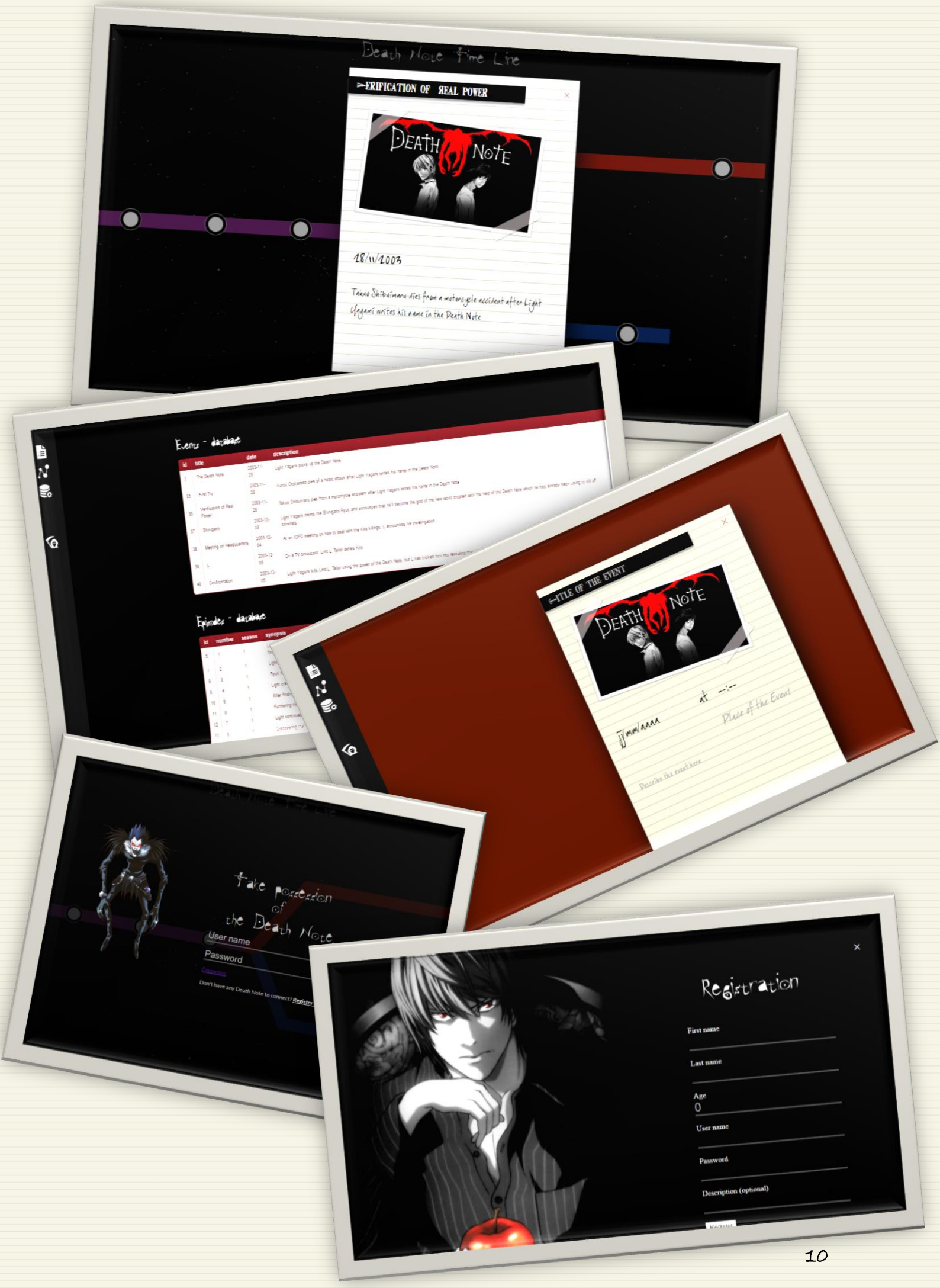
Here are all the available Request that you can execute:

- http://localhost:8080/CRUDWebAppMavenized/loadEvents - TIMELINE
- http://localhost:8080/CRUDWebAppMavenized/adminPanel - AdminPanel
- http://localhost:8080/CRUDWebAppMavenized/register - REGISTER

Enjoy the Death Note Timeline!

ATTACHMENTS




Death Note Time Line
ERIFICATION OF REAL POWER
DEATH NOTE
28/11/2003
Takao Shibuimaru dies from a motorcycle accident after Light Yagami writes his name in the Death Note
Events - database
Episodes - database
TITLE OF THE EVENT
DEATH NOTE
Place of the Event
Describe the event here
Take possession of the Death Note
User name
Password
Don't have any Death Note to connect? Register
Registration
First name
Last name
Age
0
User name
Password
Description (optional)