Student demo

Contents:

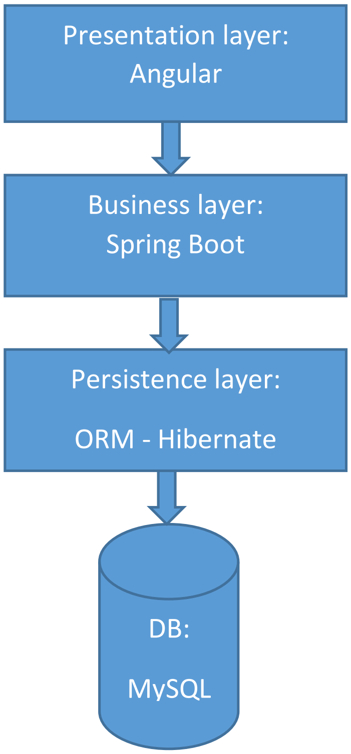
System design:

This is a simple single page web application.

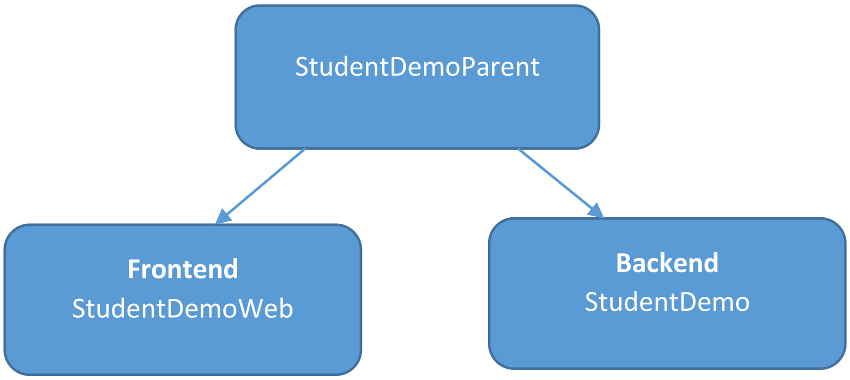
For backend Spring Framework is used.

For backend test, Junit is used.

For frontend Angular 6 framework is used.



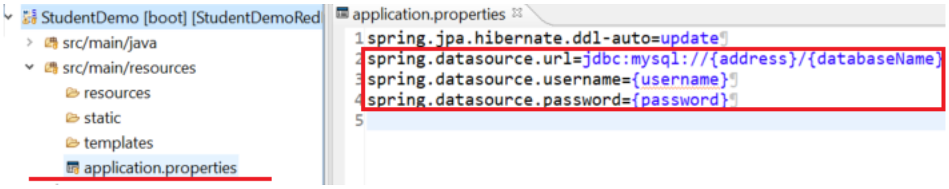
Maven module structure:



Development environment:

1. Install the following technologies.
    - JDK 8
    - Spring Boot2
    - Apache maven 3.3.9
    - Apache tomcat 9
    - MySQL8
    - NodeJS v10.5.0
    - Npm 6.1.0
    - GitHub

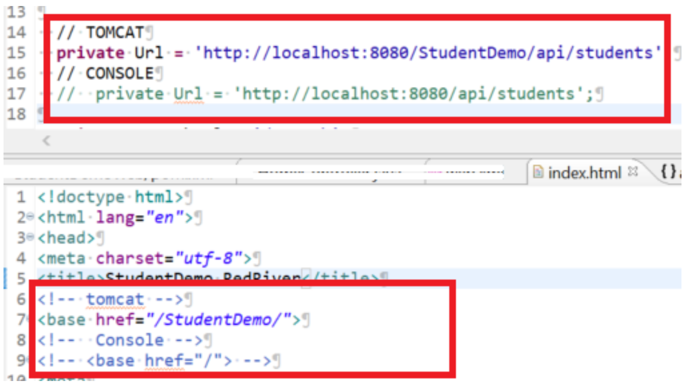
2. Download the source code from GIT Repository.
3. Create a new database in the database.
4. Configure database configuration in the application.properties.
   NOTE: *The application will create 'Student' table automatically, if it doesn't exist in the configured database.*



application.properties

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://{address}/{databaseName}
spring.datasource.username={username}
spring.datasource.password={password}
```

5. Choose base URL of application. It depends on how we run the application.
   If the application is deployed in External tomcat server, enable the lines commented 'TOMCAT'.
   If the application is deployed by console on the editor, enable the lines commented 'Console'



```
// TOMCAT
private Url = 'http://localhost:8080/StudentDemo/api/students'
// CONSOLE
//  private Url = 'http://localhost:8080/api/students';
```

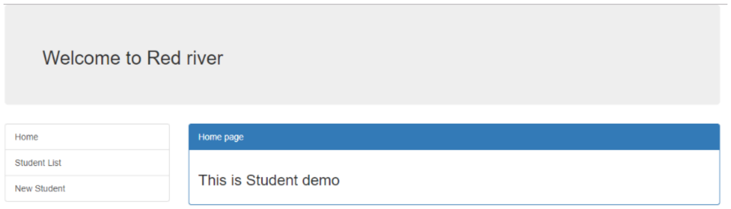
index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>StudentDemo RedRiver</title>
<!-- tomcat -->
<base href="/StudentDemo/">
<!--  Console -->
<!-- <base href="/"> -->
```

UI design:

The following pictures are main pages of the application.

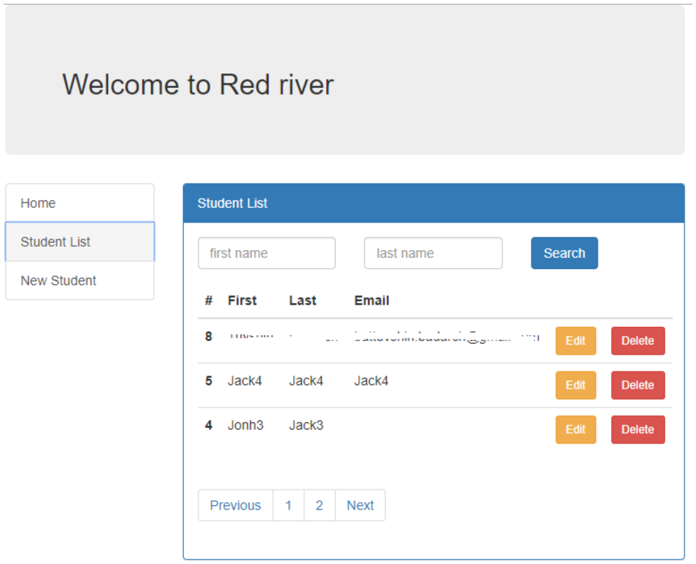
- Home page



Welcome to Red river

This is Student demo

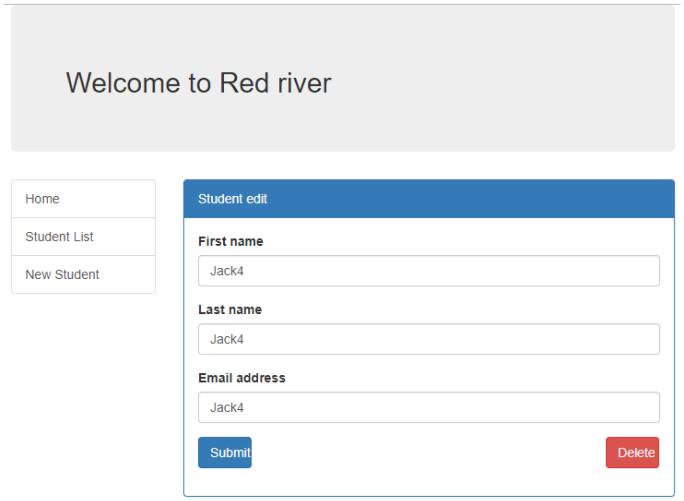
- Student list page



Welcome to Red river

#	First	Last	Email		
8	[illegible]	[illegible]	[illegible]	Edit	Delete
5	Jack4	Jack4	Jack4	Edit	Delete
4	Jonh3	Jack3		Edit	Delete

- Student detail page



Welcome to Red river

First name: Jack4

Last name: Jack4

Email address: Jack4

REST API

1. Retrieve student list (*IT IS NOT USED IN WEB*)	
URL	http://localhost:8080/api/students
HTTP METHOD	GET
REQUEST BODY	

RESPONSE BODY	[
	{
	"id": 1,
	"firstName": "Jonh",
	"lastName": "Jack",
	"email": null
	}
	]

2. Find student list by criteria	
URL	http://localhost:8080/api/students/search ?**currentPage**=0&**pageSize**=5&**firstName**=&**lastName**=
QUERY PARAM	
HTTP METHOD	GET
REQUEST BODY	

RESPONSE BODY	{
	"students": [
	{
	"id": 1,
	"firstName": "Jonh",
	"lastName": "Jack",
	"email": null
	}
	],
	"totalElement": 1,
	"totalPage": 1,
	"currentPage": 0,
	"pageSize": 2
	}

3. Get a student	
URL	http://localhost:8080/api/students/1
HTTP METHOD	GET
REQUEST BODY	
RESPONSE BODY	{ "id": 1, "firstName": "Jonh", "lastName": "Jack", "email": null

	}

4. Create a student	
URL	http://localhost:8080/api/students
HTTP METHOD	POST

REQUEST BODY	{
	"firstName": "Jonh",
	"lastName": "Jack",
	"email": "jonh.jack@redriver.com"
	}

RESPONSE BODY	{
	"id": 15,
	"firstName": "Jonh",
	"lastName": "Jack",
	"email": "jonh.jack@redriver.com"
	}

5. Update a student	
URL	http://localhost:8080/api/students/15
HTTP METHOD	PUT

REQUEST BODY	{
	"id": 15,
	"firstName": "Jonhson",
	"lastName": "Jack",
	"email": "jonh.jack@redriver.com"
	}

RESPONSE BODY	{
	"id": 15,
	"firstName": "Jonhson",
	"lastName": "Jack",
	"email": "jonh.jack@redriver.com"
	}

6. Delete a student	
URL	http://localhost:8080/api/students/15
HTTP METHOD	DELETE
REQUEST BODY	
RESPONSE BODY	true