

List of students

ID	Prénom	Nom	Email	Actions	
1	Lucas	Becker	lucas.becker@edu.devinci.fr	Edit	Delete
2	Thomas	Ngo	thomas.ngo@edu.devinci.fr	Edit	Delete
3	Alain	Delon	alain.delon@gmail.com	Edit	Delete
4	Jean	Dujardin	jean.dujardin@gmail.com	Edit	Delete
5	Gérard	Depardieu	gerard.depardieu@gmail.com	Edit	Delete
6	Marion	Cotillard	marion.cotillard@gmail.com	Edit	Delete
7	Guillaume	Canet	guillaume.canet@gmail.com	Edit	Delete

Add

DYNAMIC WEB APPLICATION

WEB STUDENT BOOK

BECKER LUCAS & NGO THOMAS
IOS 1 – PROMO 2022

DESCRIPTION

In this project, we want to develop the « WebStudentBook » application using frameworks (JSF for JavaEE and EclipseLink for JPA).

The application allows us to:

- Display students from a MySQL student database in a browser.
- Add a new student to the database.
- Edit a displayed student.
- Delete a displayed student.
-

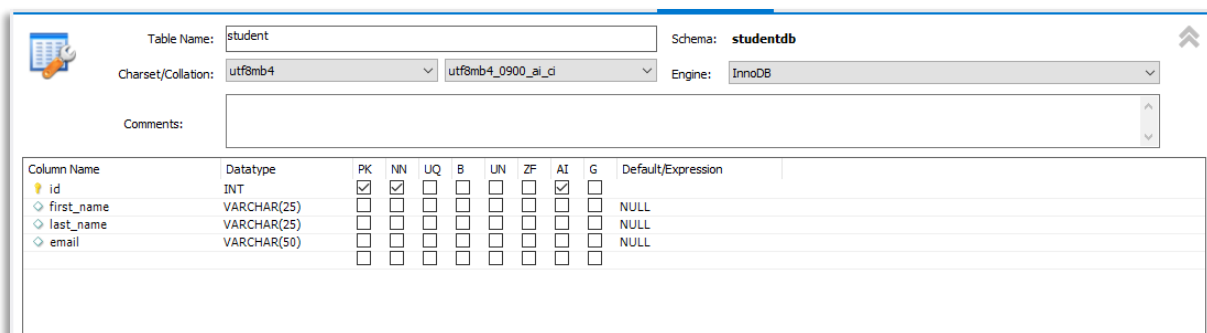
To make simple, a student is characterized by his:

- ID
- First name
- Last name
- Email

THE DATABASE

In the MySQL server, the database “studentdb” was created and initialized by a table:

- student: receive all the username and password for the login system



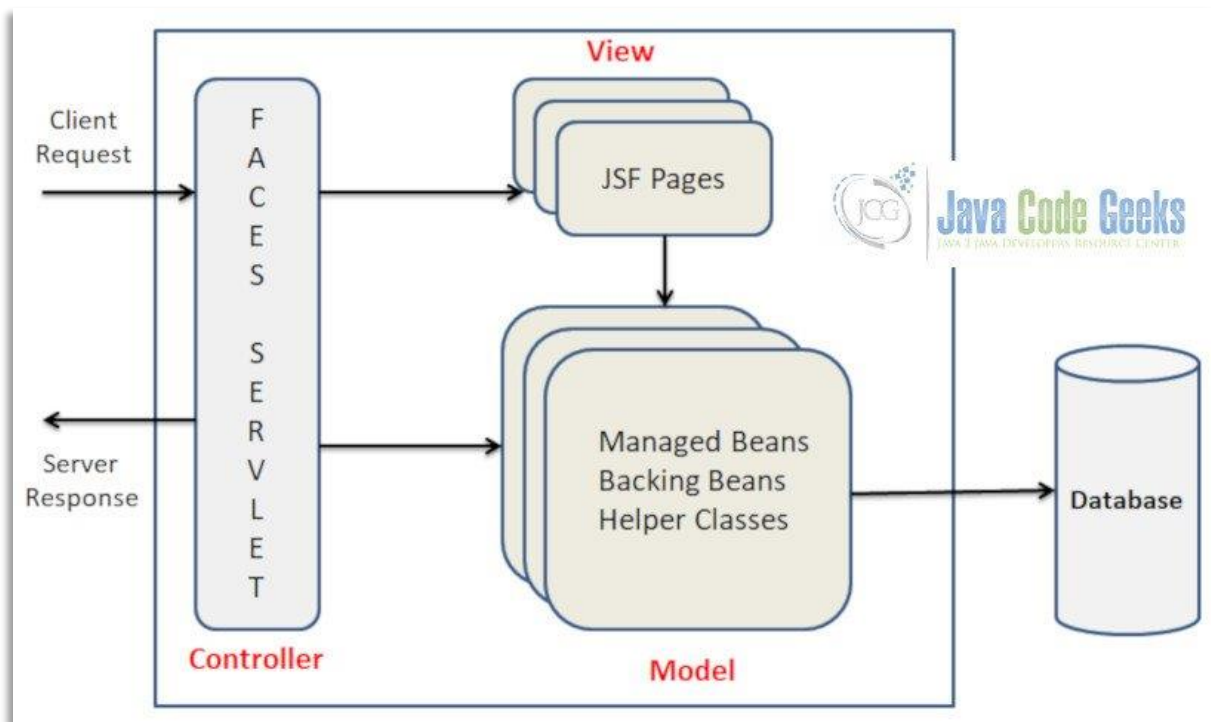
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
first_name	VARCHAR(25)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
last_name	VARCHAR(25)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
email	VARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

It was initialized by MySQL Workbench

```
1 • CREATE DATABASE IF NOT EXISTS studentdb ;
2 • USE studentdb;
3
4 • CREATE TABLE `studentdb`.`student` (
5     `id` INT NOT NULL AUTO_INCREMENT,
6     `first_name` VARCHAR(45) NULL,
7     `last_name` VARCHAR(45) NULL,
8     `email` VARCHAR(45) NULL,
9     PRIMARY KEY (`id`));
10
11 • INSERT INTO student(first_name,last_name,email) VALUES ('Lucas','Becker','lucas.becker@edu.devinci.fr');
12 • INSERT INTO student(first_name,last_name,email) VALUES ('Thomas','Ngo','thomas.ngo@edu.devinci.fr');
13 • INSERT INTO student(first_name,last_name,email) VALUES ('Alain','Delon','alain.delon@gmail.com');
14 • INSERT INTO student(first_name,last_name,email) VALUES ('Jean','Dujardin','jean.dujardin@gmail.com');
15 • INSERT INTO student(first_name,last_name,email) VALUES ('Gérard','Depardieu','gerard.depardieu@gmail.com');
16 • INSERT INTO student(first_name,last_name,email) VALUES ('Marion','Cotillard','marion.cotillard@gmail.com');
17 • INSERT INTO student(first_name,last_name,email) VALUES ('Guillaume','Canet','guillaume.canet@gmail.com');
```

HOW IT WORKS?

Our program is based on MVC Design Pattern, it based on Managed Beans and JSF components.



FACE SERVLETS

We create several face servlets as shown in the picture below.

- Add-student.xhtml
- Edit-student.xhtml
- List-students.xhtml

List-student is the main face servlet, it displays the list of students and the several modifications that we can do to it.

Add-student is the face servlet which allow us to enter student new information and send it to the database.

New Student

LastName

FirstName

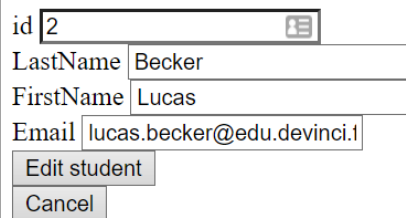
Email

Save

Cancel

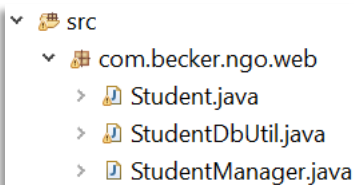
Edit-student, the face servlet which identify a student by his id and allow us to modify the information associate to this student

Edit Student



MANAGED BEANS

We create three managed beans.



- The **student.java** have all the attribute of the student.

```
1 package com.becker.ngo.web;
2
3 import javax.faces.bean.ManagedBean;
4
5 @ManagedBean
6 @RequestScoped
7 @Entity
8 public class Student {
9     @Id //primary key
10    @GeneratedValue(strategy=GenerationType.IDENTITY) //auto-incrementation
11    @Column(name="id")
12    private int Id;
13
14    @Column(name="first_name")
15    private String firstName;
16
17    @Column(name="last_name")
18    private String lastName;
19
20    private String email;
21
22    public Student() {
23        Id = 0;
24        firstName = null;
25        lastName = null;
26        email = null;
27    }
28
29    public Student(int IdArg, String firstNameArg, String lastNameArg, String emailArg) {
```

- **StudentDbUtil.java** is the model which managed the database thanks to JPA.

```

1 package com.becker.ngo.web;
2
3 import java.util.ArrayList;
4
5 @ManagedBean(eager=true)
6 //Si on met pas eager, il fait une lazy instanciation c'est à dire qu'il ne passe pas par le constructeur
7 @ApplicationScoped
8
9 public class StudentDbUtil {
10
11     private static final EntityManagerFactory factory = Persistence.createEntityManagerFactory("JSFJPA");
12     static EntityManager em = factory.createEntityManager();
13
14     //private EntityManagerFactory factory;
15     //private static final String PERSIS_NAME = "JSFJPA";
16     //factory = Persistence.createEntityManagerFactory(PERSIS_NAME);
17
18     public static List<Student> getStudents(){
19
20         List<Student> students;
21
22         students = em.createQuery("SELECT s FROM Student AS s").getResultList();
23
24         return students;
25     }
26
27     public static void addStudent(Student student) {

```

- **StudentManager.java** allows us to load, save, edit or even delete a student from the database.

```

1 package com.becker.ngo.web;
2
3 import java.util.ArrayList;
4
5 @ManagedBean
6 @SessionScoped
7
8 public class StudentManager {
9
10     List<Student> students;
11
12     public StudentManager() {
13         students = new ArrayList<Student>();
14     }
15
16     public List<Student> getStudents() {
17         return students;
18     }
19
20     public void setStudents(List<Student> students) {
21         this.students = students;
22     }
23
24     public void loadStudents() {
25         this.students = StudentDbUtil.getStudents();
26     }
27
28     public String addStudent(Student stu) {
29
30         StudentDbUtil.addStudent(stu);

```