

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC – TỰ NHIÊN

KHOA ĐIỆN TỬ - VIỄN THÔNG



BÁO CÁO ĐO ÁN

MÔN: LẬP TRÌNH HƯỚNG ĐÓI TƯỢNG

Đề tài: Xây dựng chương trình quản lý học sinh bằng lập trình hướng đối tượng

GVHD : Lê Đức Trị

SVTH : Nguyễn Anh Tuấn – 20200400

Nguyễn Gia Phụng – 20200313

Nguyễn Hữu Luật – 20200256

NHÓM 5 – LỚP 2

Thành phố Hồ Chí Minh, ngày 15 tháng 03 năm 2023

MỤC LỤC

I. Phân tích chức năng:	3
II. Triển khai lập trình:	7
2.1. File StudentManagementApplication.java:	7
2.2. File Istudent.java:	7
2.3. File StudentRepository.java: triển khai từ Istudent	8
2.4. File StudentService.java triển khai từ IStudent:	11
2.5. File IndexMapping.java:	12
2.6. File Person.java:	13
2.7. File Student.java kế thừa lớp cha Person:	15
2.8. File Response.java	18
2.9. File StudentResponse.java:	19
2.10. File Req.java và DoubleReq.java hỗ trợ các object truyền đi cho request	21
III. Minh họa trực quan	24
3.1. Giao diện:	24
3.2. Thao tác thêm học sinh:	25
3.3. Xem danh sách học sinh:	26
3.4. Cập nhật điểm 4.5 cho học sinh có id là 2:	27
IV. Tổng kết	28
4.1. Các tính chất OOP được sử dụng:	28
4.2. Công nghệ áp dụng:	28

**Ngôn ngữ sử dụng: Java, JavaScript

I. Phân tích chức năng:

Song song với việc phát triển mô hình quản lý sinh viên, không chỉ quản lý sinh viên nói riêng mà còn quản lý con người nói chung trong cả trường học với mục đích mở rộng hơn ứng dụng thì chúng ta có class Person với các thuộc tính name, citizenId, number. Các thuộc tính của đối tượng Student cần thiết như ngày sinh, điểm trung bình, địa chỉ, id học sinh, từ đó khởi tạo bảng thuộc tính với đối tượng một cách trực quan:

Collection	Field	Type	Remark
student	studentId	long	mã số học sinh
	className	string	tên lớp
	avgPoint	double	điểm trung bình
	born	string	ngày sinh
	address	string	địa chỉ

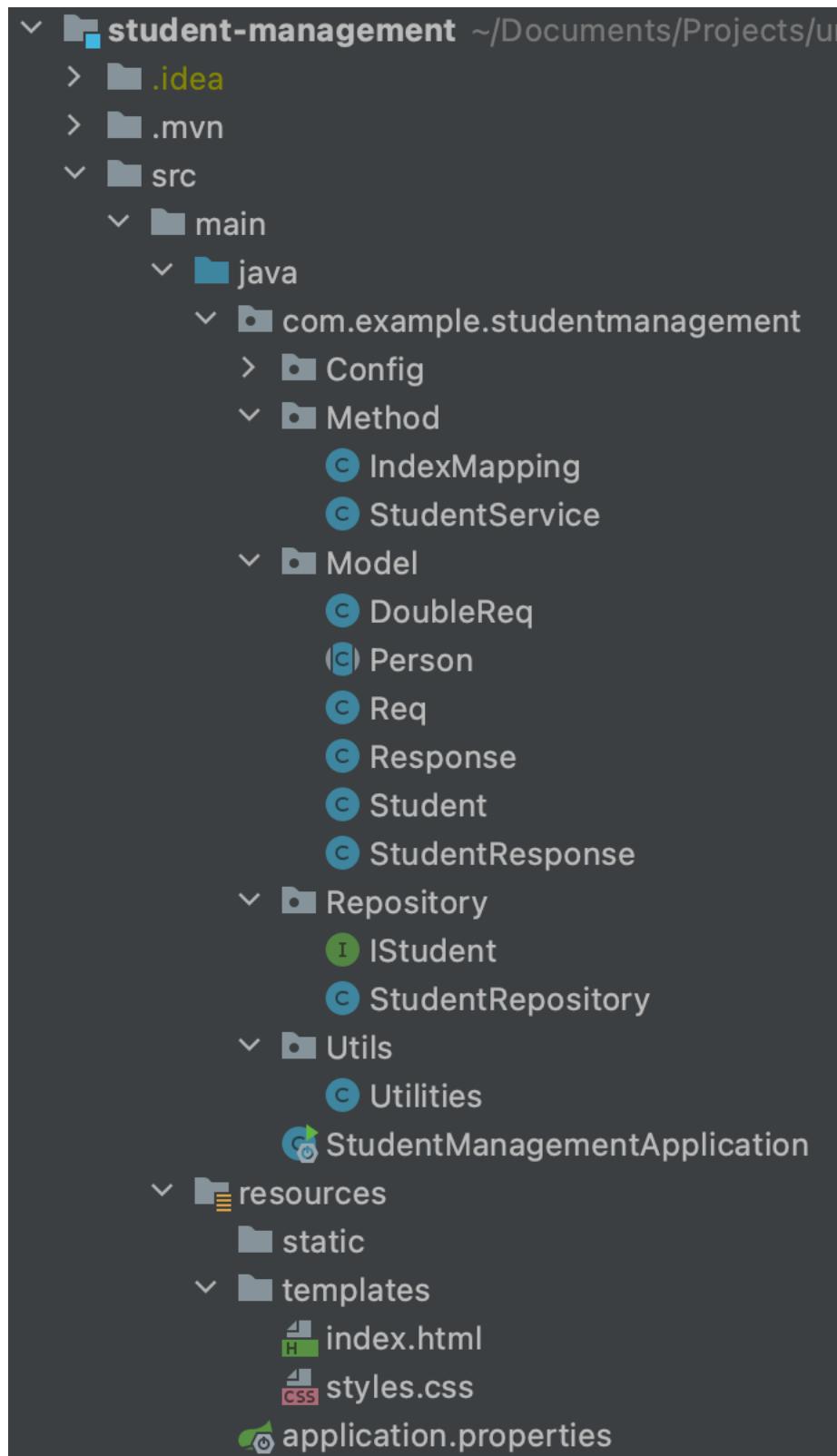
Collection	Field	Type	Remark
person	name	string	họ tên
	citizenId	long	số căn cước / cmnd
	number	string	số điện thoại

Lúc này, đối tượng Student sẽ kế thừa các thuộc tính của Person để nhận về các thuộc tính của Person và các hàm, câu nối thông qua khoá super(name,citizenId,number) khi khai báo phương thức khởi tạo Student.

Sau đó các actions được triển khai để quản lý danh sách sinh viên như CRUD (create, read, update, delete a.k.a thêm bớt xoá sửa):

Name of Method	Params	Remark
postStudent	Student student	thêm học sinh vào danh sách
deleteStudentByStudentId	Long studentId	xoá học sinh bằng id
getAllStudents		lấy danh sách tất cả hs
getStudentsFromClass	String classRoom	lấy danh sách tất cả hs chung lớp
getStudentFromStudentId	Long studentId	lấy thông tin học sinh bằng id
getStudentsLowPoint		lấy danh sách các học sinh điểm < 5
updateStudentClass	Long studentId, String className	cập nhật thông tin lớp học cho học sinh bằng id
updateStudentPoint	Long studentId, Double avgPoint	cập nhật điểm trung bình cho hs bằng id
updateStudentAddress	Long studentId, String address	cập nhật địa chỉ hs bằng id
updateStudentNumber	Long studentId, String number	cập nhật sdt hs bằng id

Cùng với ngôn ngữ Java, SpringBoot phát triển và hỗ trợ Java một cách mạnh mẽ và để phát triển báo cáo một cách trực quan nhất có thể, việc xây dựng một server-side với Java SpringBoot và một client-side với JavaScript là rất hữu ích và thôi thúc tinh thần tìm hiểu. Bước đầu tiếp cận với mô hình design pattern MVC là ổn định và cơ bản để nắm bắt. Cấu trúc dự án:



Trong đó:

- Package method là nơi định nghĩa các phương thức với endpoint API ví dụ như *abc.xyz/post-student* thì post-student là endpoint API đại diện

cho một phần tài nguyên trên máy chủ, cụ thể là server-side này thì với endpoint đó được gọi sẽ xử lý phương thức tương ứng được định nghĩa.

- Model là nơi chứa các class Object như Student, Person và các Response trả về client-side để xử lý dữ liệu.
- Repository là nơi chứa các triển khai của các hàm, phương thức để nối tới package method như trên, trong này sẽ là nơi xử lý dữ liệu chính. Class Istudent là class interface để triển khai cho StudentRepository và cả StudentService tránh tình trạng nhầm lẫn các method.
- Utils nơi chứa các xử lý logic kiểm tra vật.
- StudentManagementApplication là một lớp chứa phương thức main() để khởi chạy ứng dụng này theo trình tự :
 - Tìm kiếm tất cả các phụ thuộc của ứng dụng và xác định các cấu hình mặc định.
 - Tự động cấu hình ứng dụng dựa trên các cấu hình mặc định và các cấu hình được xác định bởi người dùng trong các tệp cấu hình (config files).
 - Tạo ra các Bean của Spring phù hợp để xử lý các yêu cầu từ người dùng, bao gồm các Bean để xử lý các yêu cầu HTTP, Bean để xử lý các yêu cầu cơ sở dữ liệu, và các Bean khác để xử lý các tác vụ khác trong ứng dụng.
 - Kết nối các Bean lại với nhau để tạo ra một ứng dụng hoàn chỉnh.
 - Chạy ứng dụng và lắng nghe các yêu cầu từ người dùng để xử lý chúng.

II. Triển khai lập trình:

2.1. File StudentManagementApplication.java:

```
package com.example.studentmanagement;

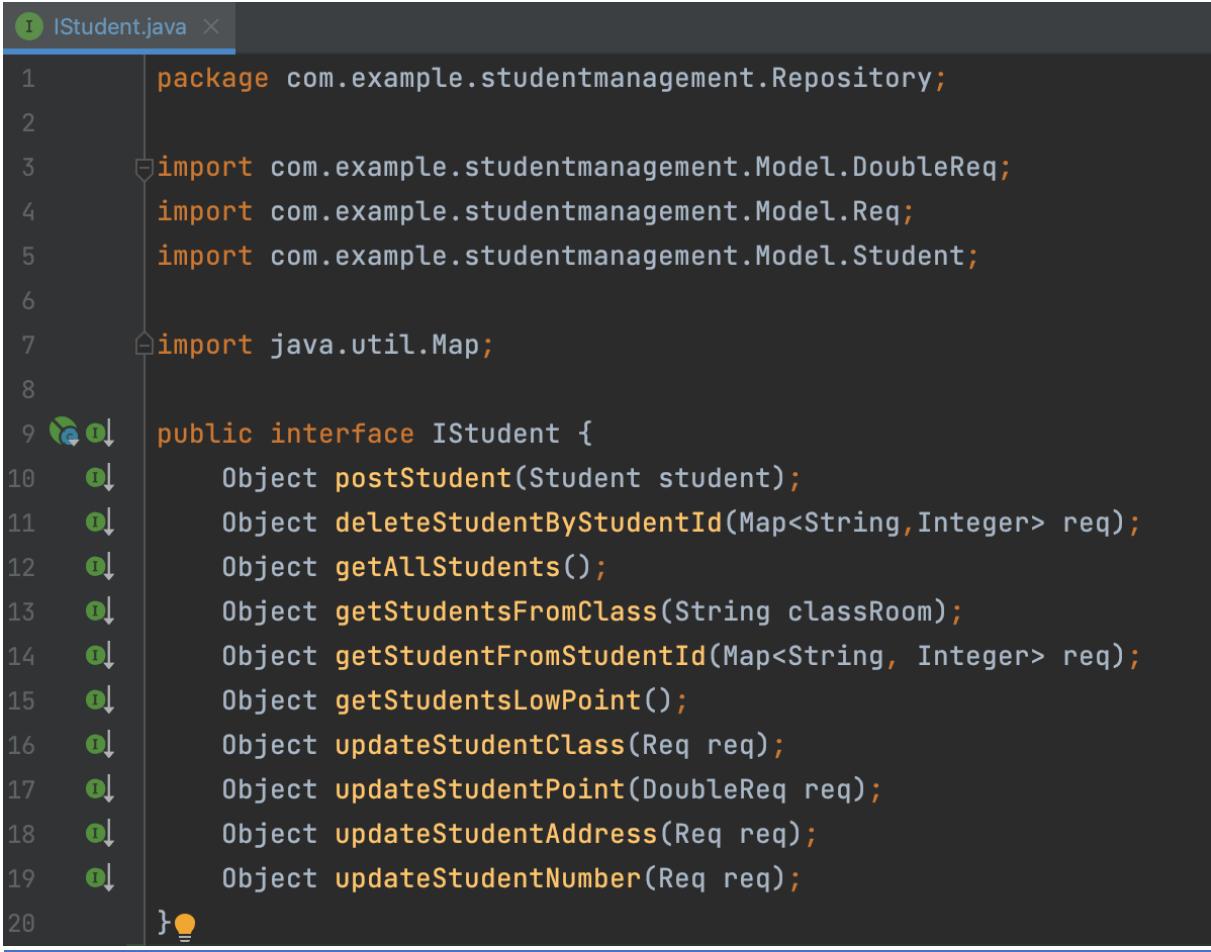
import io.swagger.v3.oas.annotations.OpenAPIDefinition;
import io.swagger.v3.oas.annotations.info.Info;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@SpringBootApplication(exclude = {SecurityAutoConfiguration.class})
@EnableWebMvc
@OpenAPIDefinition(info = @Info(title = "Student Management API", version = "1.0"))
public class StudentManagementApplication {

    public static void main(String[] args) {
        SpringApplication.run(StudentManagementApplication.class, args);
    }

}
```

2.2. File Istudent.java:



```
IStudent.java
1 package com.example.studentmanagement.Repository;
2
3 import com.example.studentmanagement.Model.DoubleReq;
4 import com.example.studentmanagement.Model.Req;
5 import com.example.studentmanagement.Model.Student;
6
7 import java.util.Map;
8
9 public interface IStudent {
10     Object postStudent(Student student);
11     Object deleteStudentByStudentId(Map<String, Integer> req);
12     Object getAllStudents();
13     Object getStudentsFromClass(String classRoom);
14     Object getStudentFromStudentId(Map<String, Integer> req);
15     Object getStudentsLowPoint();
16     Object updateStudentClass(Req req);
17     Object updateStudentPoint(DoubleReq req);
18     Object updateStudentAddress(Req req);
19     Object updateStudentNumber(Req req);
20 }
```

2.3. File StudentRepository.java: triển khai từ Istudent là controller quản lý thực thi các hành động service



```

1 package com.example.studentmanagement.Repository;
2
3 import com.example.studentmanagement.Model.*;
4 import com.example.studentmanagement.Utils.Utilities;
5
6 import java.util.ArrayList;
7 import java.util.List;
8 import java.util.Map;
9
10 public class StudentRepository implements IStudent{
11
12     private static StudentRepository studentRepository = new StudentRepository();
13
14     private static List<Student> studentList = new ArrayList<>();
15
16     /**
17      * Singleton design pattern
18      * @return always instance of StudentRepository
19      */
20     public static StudentRepository getStudentRepository(){
21         if (studentRepository == null){
22             studentRepository = new StudentRepository();
23             studentList = new ArrayList<>();
24         }
25
26         return studentRepository;
27     }
28
29     @Override
30     public Object postStudent(Student student) {
31         if (student.getName().equals("") ||
32             student.getNumber().equals("") ||
33             student.getStudentId() == null ||
34             student.getClassName().equals("") ||
35             student.getCitizenId() == null){
36             return new Response( status: false , info: "Fill all required cells");
37         } else {
38             Long id = student.getStudentId();
39             Long citizenID = student.getCitizenId();
40             for ( Student i : studentList){
41                 if (i.getStudentId().equals(id) || i.getCitizenId().equals(citizenID)){
42                     return new Response( status: false, info: "Duplicate student in id or citizenId");
43                 }
44             }
45             studentList.add(student);
46         }
47         return new Response( status: true, info: "Add successfully");
48     }
49 }

```

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

```
49     @Override
50     public Object deleteStudentByStudentId(Map<String, Integer> req) {
51         int n = studentList.size();
52         if (n == 0) return new Response( status: false, info: "There is nothing to delete");
53         long studentId = req.get("studentId");
54         for (int i = 0 ; i < n ; i++) {
55             Student currStudent = studentList.get(i);
56             if (currStudent.getStudentId() == studentId) {
57                 studentList.remove(currStudent);
58
59                 return new Response( status: true, info: "Deleted successfully student with id: "+ studentId);
60             }
61         }
62         return new Response( status: false, info: "There is no student with id: "+studentId);
63     }
64
65     @Override
66     public Object getAllStudents() {
67         return studentList;
68     }
69
70     @Override
71     public Object getStudentsFromClass(String classRoom) {
72         List<Student> res = new ArrayList<>();
73         for (Student student : studentList) {
74             if (student.getClassName().equals(classRoom)) res.add(student);
75         }
76         return res;
77     }
78
79     @Override
80     public Object getStudentFromStudentId(Map<String, Integer> req) {
81         long studentId = Integer.parseInt(String.valueOf(req.get("studentId")));
82         for (Student student : studentList) {
83             if (student.getStudentId() == studentId) return new StudentResponse( status: true, student);
84         }
85         return new StudentResponse().getStudentResponseByType(Utilities.DEFAULT);
86     }
}
```

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

```
88     @Override
89     public Object getStudentsLowPoint() {
90         List<Student> res = new ArrayList<>();
91         for (Student student : studentList) {
92             if (student.getAvgPoint() >= 0 && student.getAvgPoint() <= 5) res.add(student);
93         }
94         return res;
95     }
96
97     @Override
98     public Object updateStudentClass(Req req) {
99         int n = studentList.size();
100        if (n == 0) return new Response(status: false, info: "Please add one more student");
101        long studentId = req.getStudentId();
102        for (Student currStudent : studentList) {
103            if (currStudent.getStudentId() == studentId) {
104                currStudent.setClassName(req.getNewValue());
105                return new Response(status: true, info: "Update classroom for " + studentId + " successfully");
106            }
107        }
108        return new Response(status: false, info: "No student with id " + studentId + " in list");
109    }
110
111    @Override
112    public Object updateStudentPoint(DoubleReq req) {
113        int n = studentList.size();
114        if (n == 0) return new Response(status: false, info: "Please add one more student");
115        long studentId = req.getStudentId();
116        double avgPoint = req.getAvgPoint();
117        for (Student currStudent : studentList) {
118            if (currStudent.getStudentId() == studentId) {
119                currStudent.setAvgPoint(avgPoint);
120                return new Response(status: true, info: "Update average point for " + studentId + " successfully");
121            }
122        }
123        return new Response(status: false, info: "No student with id " + studentId + " in list");
124    }
```

```
126     @Override
127     public Object updateStudentAddress(Req req) {
128         int n = studentList.size();
129         if (n == 0) return new Response(status: false, info: "Please add one more student");
130         long studentId = req.getStudentId();
131         for (Student currStudent : studentList) {
132             if (currStudent.getStudentId() == studentId) {
133                 currStudent.setAddress(req.getNewValue());
134                 return new Response(status: true, info: "Update address for " + studentId + " successfully");
135             }
136         }
137         return new Response(status: false, info: "No student with id " + studentId + " in list");
138     }
139
140     @Override
141     public Object updateStudentNumber(Req req) {
142         int n = studentList.size();
143         if (n == 0) return new Response(status: false, info: "Please add one more student");
144         long studentId = req.getStudentId();
145         for (Student currStudent : studentList) {
146             if (currStudent.getStudentId() == studentId) {
147                 currStudent.setNumber(req.getNewValue());
148                 return new Response(status: true, info: "Update number for " + studentId + " successfully");
149             }
150         }
151         return new Response(status: false, info: "No student with id " + studentId + " in list");
152     }
```

2.4. File StudentService.java triển khai từ IStudent:

```

  C StudentService.java ×
1 package com.example.studentmanagement.Method;
2
3 import com.example.studentmanagement.Model.DoubleReq;
4 import com.example.studentmanagement.Model.Req;
5 import com.example.studentmanagement.Model.Student;
6 import com.example.studentmanagement.Repository.IStudent;
7 import com.example.studentmanagement.Repository.StudentRepository;
8 import org.springframework.web.bind.annotation.*;
9
10 import java.util.Map;
11
12 @RestController
13 @CrossOrigin(origins = "*", allowedHeaders = "*")
14 @RequestMapping(value = "/")
15 public class StudentService implements IStudent {
16     private final StudentRepository studentRepository = StudentRepository.getStudentRepository();
17
18     @Override
19     @PostMapping(value = "create")
20     public Object postStudent(@RequestBody Student student) {
21         return studentRepository.postStudent(student);
22     }
23
24     @Override
25     @DeleteMapping(value = "delete-by-id")
26     public Object deleteStudentByStudentId(@RequestBody Map<String, Integer> req) {
27         return studentRepository.deleteStudentByStudentId(req);
28     }
29
30     @Override
31     @GetMapping(value = "get-all")
32     public Object getAllStudents() {
33         return studentRepository.getAllStudents();
34     }
35
36     @Override
37     @GetMapping(value = "get-class")
38     public Object getStudentsFromClass(@RequestParam String classRoom) {
39         return studentRepository.getStudentsFromClass(classRoom);
40     }
41
42     @Override
43     @GetMapping(value = "get-student-id")
44     public Object getStudentFromStudentId(@RequestParam Map<String, Integer> req) {
45         return studentRepository.getStudentFromStudentId(req);
46     }

```

```

48     @Override
49     @GetMapping(value=@Value("get-low-point"))
50     public Object getStudentsLowPoint(){
51         return studentRepository.getStudentsLowPoint();
52     }
53
54     @Override
55     @PutMapping(value=@Value("update-class"))
56     public Object updateStudentClass(@RequestBody Req req){
57         return studentRepository.updateStudentClass(req);
58     }
59
60     @Override
61     @PutMapping(value=@Value("update-address"))
62     public Object updateStudentAddress(@RequestBody Req req) {
63         return studentRepository.updateStudentAddress(req);
64     }
65
66     @Override
67     @PutMapping(value=@Value("update-point"))
68     public Object updateStudentPoint(@RequestBody DoubleReq req) {
69         return studentRepository.updateStudentPoint(req);
70     }
71
72     @Override
73     @PutMapping(value=@Value("update-number"))
74     public Object updateStudentNumber(@RequestBody Req req) {
75         return studentRepository.updateStudentNumber(req);
76     }
77 }
```

2.5. File IndexMapping.java:

```

IndexMapping.java ×
1 package com.example.studentmanagement.Method;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.servlet.ModelAndView;
6
7 @Controller
8 public class IndexMapping {
9
10    @RequestMapping("/")
11    public ModelAndView index () {
12        ModelAndView modelAndView = new ModelAndView();
13        modelAndView.setViewName("index");
14        return modelAndView;
15    }
16 }
```

2.6. File Person.java:

```

1 package com.example.studentmanagement.Model;
2
3 import com.google.gson.annotations.Expose;
4 import com.google.gson.annotations.SerializedName;
5
6 /**
7  * Abstract object class
8 */
9
10 public abstract class Person {
11     @SerializedName("name")
12     @Expose
13     public String name;
14     @SerializedName("number")
15     @Expose
16     public String number;
17     @SerializedName("citizenId")
18     @Expose
19     public Long citizenId;
20
21     /**
22      * No args constructor for use in serialization
23      *
24      */
25     public Person() {
26     }
27
28     /**
29      *
30      * @param number
31      * @param name
32      * @param citizenId
33      */
34     public Person(String name, String number, Long citizenId) {
35         super();
36         this.name = name;
37         this.number = number;
38         this.citizenId = citizenId;
39     }
40
41     public String getName() {
42         return name;
43     }
44 }

```

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

```
45     public void setName(String name) {
46         this.name = name;
47     }
48
49     public String getNumber() {
50         return number;
51     }
52
53     public void setNumber(String number) {
54         this.number = number;
55     }
56
57     public Long getCitizenId() {
58         return citizenId;
59     }
60
61     public void setCitizenId(Long citizenId) {
62         this.citizenId = citizenId;
63     }
64
65     @Override
66     public String toString() {
67         StringBuilder sb = new StringBuilder();
68         sb.append(Person.class.getName()).append('@').append(Integer.toHexString(System.identityHashCode(this))).append('[');
69         sb.append("name");
70         sb.append('=');
71         sb.append((this.name == null)?<null>:this.name);
72         sb.append(',');
73         sb.append("number");
74         sb.append('=');
75         sb.append((this.number == null)?<null>:this.number);
76         sb.append(',');
77         sb.append("citizenId");
78         sb.append('=');
79         sb.append((this.citizenId == null)?<null>:this.citizenId);
80         sb.append(',');
81         if (sb.charAt((sb.length()- 1)) == ',') {
82             sb.setCharAt((sb.length()- 1), ch: ']');
83         } else {
84             sb.append(']');
85         }
86         return sb.toString();
87     }
88 }
```

2.7. File Student.java kế thừa lớp cha Person:

```
1  package com.example.studentmanagement.Model;
2
3  import com.google.gson.annotations.Expose;
4  import com.google.gson.annotations.SerializedName;
5
6  /**
7   * Encapsulation with datafields and getters/setters
8   * Polymorphism with initialize objects by constructor
9   * Inheritance from Person
10 */
11 public class Student extends Person {
12     @SerializedName("studentId")
13     @Expose
14     private Long studentId;
15     @SerializedName("avgPoint")
16     @Expose
17     private Double avgPoint;
18     @SerializedName("className")
19     @Expose
20     private String className;
21     @SerializedName("born")
22     @Expose
23     private String born;
24     @SerializedName("address")
25     @Expose
26     private String address;
27
28     /**
29      * No args constructor for use in serialization
30      *
31      */
32     public Student() {
33 }
```

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

```
35      /**
36       *
37       * @param studentId
38       * @param number
39       * @param address
40       * @param born
41       * @param name
42       * @param className
43       * @param citizenId
44       */
45     public Student(Long studentId, String name, String className, Long citizenId, Double avgPoint, String number, String born, String address) {
46       super(name, number, citizenId);
47       this.studentId = studentId;
48       this.className = className;
49       this.avgPoint = avgPoint;
50       this.born = born;
51       this.address = address;
52     }
53
54     public Long getStudentId() { return studentId; }
55
56     public void setStudentId(Long studentId) { this.studentId = studentId; }
57
58     public String getClassName() { return className; }
59
60     public void setClassName(String className) { this.className = className; }
61
62     public Double getAvgPoint() { return avgPoint; }
63
64     public void setAvgPoint(Double avgPoint) { this.avgPoint = avgPoint; }
65
66     public String getBorn() { return born; }
67
68     public void setBorn(String born) { this.born = born; }
69
70     public String getAddress() { return address; }
71
72     public void setAddress(String address) { this.address = address; }
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
```

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

```
94     @Override
95     public String toString() {
96         StringBuilder sb = new StringBuilder();
97         sb.append("MHS");
98         sb.append(':');
99         sb.append(((this.studentId == null)?<null>:this.studentId));
100        sb.append(',');
101        sb.append("Ho Ten");
102        sb.append(':');
103        sb.append(((this.name == null)?<null>:this.name));
104        sb.append(',');
105        sb.append("Diem trung binh");
106        sb.append(':');
107        sb.append(((this.avgPoint == null)?<null>:this.avgPoint));
108        sb.append(',');
109        sb.append("Sdt");
110        sb.append(':');
111        sb.append(((this.number == null)?<null>:this.number));
112        sb.append(',');
113        sb.append("CCCD");
114        sb.append(':');
115        sb.append(((this.citizenId == null)?<null>:this.citizenId));
116        sb.append(',');
117        sb.append("Ten lop");
118        sb.append(':');
119        sb.append(((this.className == null)?<null>:this.className));
120        sb.append(',');
121        sb.append("Ngay sinh");
122        sb.append(':');
123        sb.append(((this.born == null)?<null>:this.born));
124        sb.append(',');
125        sb.append("Dia chi");
126        sb.append(':');
127        sb.append(((this.address == null)?<null>:this.address));
128        sb.append(',');
129        if (sb.charAt((sb.length()- 1)) == ',') {
130            sb.setCharAt((sb.length()- 1), ch: ']');
131        } else {
132            sb.append(']');
133        }
134        return sb.toString();
135    }
136}
```

2.8. File Response.java

```

  C Response.java ×
1  package com.example.studentmanagement.Model;
2
3  import com.google.gson.annotations.Expose;
4  import com.google.gson.annotations.SerializedName;
5
6  public class Response {
7      @SerializedName("status")
8      @Expose
9      private Boolean status;
10     @SerializedName("info")
11     @Expose
12     private String info;
13
14     /**
15      * No args constructor for use in serialization
16      *
17      */
18     public Response() {
19     }
20
21     /**
22      *
23      * @param status
24      * @param info
25      */
26     public Response(Boolean status, String info) {
27         super();
28         this.status = status;
29         this.info = info;
30     }
31
32     public Boolean getStatus() { return status; }
33
34     public void setStatus(Boolean status) { this.status = status; }
35
36     public String getInfo() { return info; }
37
38     public void setInfo(String info) { this.info = info; }
39
40
41
42
43
44
45
46
47
48     @Override
49     public String toString() {
50         StringBuilder sb = new StringBuilder();
51         sb.append(Response.class.getName()).append('@').append(Integer.toHexString(System.identityHashCode(this))).append('[');
52         sb.append("status");
53         sb.append('=');
54         sb.append((this.status == null)?<null>:this.status);
55         sb.append(',');
56         sb.append("info");
57         sb.append('=');
58         sb.append((this.info == null)?<null>:this.info);
59         sb.append(',');
60         if (sb.charAt((sb.length()- 1)) == ',') {
61             sb.setCharAt((sb.length()- 1), ']');
62         } else {
63             sb.append(']');
64         }
65         return sb.toString();
66     }
67 }
```

2.9. File StudentResponse.java:

```

1  package com.example.studentmanagement.Model;
2
3  import com.example.studentmanagement.Utils.Utilities;
4  import com.google.gson.annotations.Expose;
5  import com.google.gson.annotations.SerializedName;
6
7  public class StudentResponse {
8      @SerializedName("status")
9      @Expose
10     private Boolean status;
11
12     @SerializedName("student")
13     @Expose
14     private Student student;
15
16     /**
17      * No args constructor for use in serialization
18      *
19      */
20
21     @
22     public StudentResponse getStudentResponseByType(String type) {
23         switch (type) {
24             case Utilities.DEFAULT: {
25                 return new StudentResponse( status: false,
26                     new Student((long) -1, name: "", className: "",(long) -1, (double) -1, number: "", born: "", address: ""));
27             }
28             default: {
29                 return new StudentResponse();
30             }
31         }
32     }
33
34     /**
35      *
36      * @param student
37      * @param status
38      */
39     public StudentResponse(Boolean status, Student student) {
40         super();
41         this.status = status;
42         this.student = student;
43     }
44

```

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

```
45     public Boolean getStatus() { return status; }
46
47     public void setStatus(Boolean status) { this.status = status; }
48
49     public Student getStudent() { return student; }
50
51     public void setStudent(Student student) { this.student = student; }
52
53
54     @Override
55     public String toString() {
56         StringBuilder sb = new StringBuilder();
57         sb.append(StudentResponse.class.getName()).append('@').append(Integer.toHexString(System.identityHashCode( this))).append('[');
58         sb.append("status");
59         sb.append('=');
60         sb.append((this.status == null)?<null>:this.status);
61         sb.append(',');
62         sb.append("student");
63         sb.append('=');
64         sb.append((this.student == null)?<null>:this.student);
65         sb.append(',');
66         if (sb.charAt((sb.length()- 1)) == ',') {
67             sb.setCharAt((sb.length()- 1), ch: ']');
68         } else {
69             sb.append(']');
70         }
71         return sb.toString();
72     }
73
74 }
```

2.10. File Req.java và DoubleReq.java hỗ trợ các object truyền đi cho request

```

1 package com.example.studentmanagement.Model;
2
3 import com.google.gson.annotations.Expose;
4 import com.google.gson.annotations.SerializedName;
5
6 public class Req {
7     @SerializedName("studentId")
8     @Expose
9     private Integer studentId;
10    @SerializedName("newValue")
11    @Expose
12    private String newValue;
13
14    /**
15     * No args constructor for use in serialization
16     *
17     */
18    public Req() {
19    }
20
21    /**
22     *
23     * @param studentId
24     * @param newValue
25     */
26    public Req(Integer studentId, String newValue) {
27        super();
28        this.studentId = studentId;
29        this.newValue = newValue;
30    }
31
32    public Integer getStudentId() { return studentId; }
33
34    public void setStudentId(Integer studentId) { this.studentId = studentId; }
35
36    public String getnewValue() { return newValue; }
37
38    public void setnewValue(String newValue) { this.newValue = newValue; }
39
40
41
42
43
44

```

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

```
48     @Override
49     public String toString() {
50         StringBuilder sb = new StringBuilder();
51         sb.append(Req.class.getName()).append('@').append(Integer.toHexString(System.identityHashCode(this))).append('[');
52         sb.append("studentId");
53         sb.append('=');
54         sb.append((this.studentId == null)?<null>:this.studentId);
55         sb.append(',');
56         sb.append("newValue");
57         sb.append('=');
58         sb.append((this.newValue == null)?<null>:this.newValue);
59         sb.append(',');
60         if (sb.charAt((sb.length()- 1)) == ',') {
61             sb.setCharAt((sb.length()- 1), ch: ']');
62         } else {
63             sb.append(']');
64         }
65         return sb.toString();
66     }
67 }
68 }
```

```
DoubleReq.java ×
1 package com.example.studentmanagement.Model;
2
3 import com.google.gson.annotations.Expose;
4 import com.google.gson.annotations.SerializedName;
5
6 public class DoubleReq {
7
8     @SerializedName("studentId")
9     @Expose
10    private Long studentId;
11
12     @SerializedName("avgPoint")
13     @Expose
14    private Double avgPoint;
15
16     /**
17      * No args constructor for use in serialization
18      *
19      */
20    public DoubleReq() {
21
22    /**
23     *
24     * @param studentId
25     * @param avgPoint
26     */
27    public DoubleReq(Long studentId, Double avgPoint) {
28        super();
29        this.studentId = studentId;
30        this.avgPoint = avgPoint;
31    }
32
33    public Long getStudentId() { return studentId; }
34
35    public void setStudentId(Long studentId) { this.studentId = studentId; }
36
37    public Double getAvgPoint() { return avgPoint; }
38
39    public void setAvgPoint(Double avgPoint) { this.avgPoint = avgPoint; }
40
41 }
```

```

49     @Override
50     public String toString() {
51         StringBuilder sb = new StringBuilder();
52         sb.append(DoubleReq.class.getName()).append('@').append(Integer.toHexString(System.identityHashCode(this))).append('[');
53         sb.append("studentId");
54         sb.append('=');
55         sb.append(((this.studentId == null)?<null>:this.studentId));
56         sb.append(',');
57         sb.append("avgPoint");
58         sb.append('=');
59         sb.append(((this.avgPoint == null)?<null>:this.avgPoint));
60         sb.append(',');
61         if (sb.charAt((sb.length()- 1)) == ',') {
62             sb.setCharAt((sb.length()- 1), ']');
63         } else {
64             sb.append(']');
65         }
66         return sb.toString();
67     }
68 }

```

Về phần client-side, phần này không áp dụng gì liên quan đến OOP vì vậy em xin phép không đề cập code trong này vì khá dài, nội dung chủ yếu là tạo các nút và thao tác để giao tiếp với server-side này và hiển thị kết quả trả về, cụ thể code lấy data như sau:

```

src > data.js > [e] deleteStudent > [e] response > ↵ body
You last week - init prj for heroku deploy
1 const createStudent = async (student) => {
2     try {
3         const response = await fetch('https://student-management-server.herokuapp.com/create', {
4             method: 'POST',
5             headers: {
6                 'Content-Type': 'application/json'
7             },
8             body: JSON.stringify(student)
9         });
10        const data = await response.json();
11        if (typeof data.status === 'number') {
12            return [status: false, info: "Something's wrong"];
13        }
14        return data;
15    } catch (error) {
16        console.error(error);
17        return [status: false, info: "Something's wrong"];
18    }
19 }
20 const deleteStudent = async (studentId) => {
21     try {
22         const response = await fetch('https://student-management-server.herokuapp.com/delete-by-id', {
23             method: 'DELETE',
24             headers: {
25                 'Content-Type': 'application/json'
26             },
27             body: JSON.stringify(studentId)
28         });
29        const data = await response.json();
30        if (typeof data.status === 'number') {
31            return [status: false, info: "Something's wrong"];
32        }
33        return data;
34    } catch (error) {
35        console.error(error);
36        return [status: false, info: "Something's wrong"];
37    }
38 }
39 const getAllStudent = async () => {
40     try {
41         const response = await fetch('https://student-management-server.herokuapp.com/get-all', {
42             method: 'GET',
43             headers: {
44                 'Content-Type': 'application/json'
45             }
46         });
47        const dataList = await response.json();
48        return dataList;
49    } catch (error) {
50        console.error(error);
51        return []
52    }
53 }
54 const getStudentFromClass = async (classRoom) => {
55     try {
56         const response = await fetch('https://student-management-server.herokuapp.com/get-class?classRoom=' + classRoom, {
57             method: 'GET',
58             headers: {
59                 'Content-Type': 'application/json'
60             },
61         });
62        const data = await response.json();
63    } catch (error) {
64        console.error(error);
65    }
66 }

```

```

src > data.js > [e] updateAvgPoint > [e] response > ↵ body
You last week - init prj for heroku deploy
54 const getStudentFromClass = async (classRoom) => {
55     try {
56         const response = await fetch('https://student-management-server.herokuapp.com/get-class?classRoom=' + classRoom, {
57             method: 'GET',
58             headers: {
59                 'Content-Type': 'application/json'
60             }
61         });
62        const data = await response.json();
63        return data;
64    } catch (error) {
65        console.error(error);
66    }
67 }
68 const getStudentFromStudentId = async (studentId) => {
69     try {
70         const response = await fetch('https://student-management-server.herokuapp.com/get-student-id?studentId=' + studentId, {
71             method: 'GET',
72             headers: {
73                 'Content-Type': 'application/json'
74             }
75         });
76        const data = await response.json();
77        return data;
78    } catch (error) {
79        console.error(error);
80        return [status: false, info: "Something's wrong"];
81    }
82 }
83 const getStudentAvgPoint = async () => {
84     try {
85         const response = await fetch('https://student-management-server.herokuapp.com/get-low-point', {
86             method: 'GET',
87             headers: {
88                 'Content-Type': 'application/json'
89             }
90         });
91        const data = await response.json();
92        return data;
93    } catch (error) {
94        console.error(error);
95        return [status: false, info: "Something's wrong"];
96    }
97 }
98 const updateClass = async (studentId, classRoom) => {
99     try {
100         const response = await fetch('https://student-management-server.herokuapp.com/update-class', {
101             method: 'PUT',
102             headers: {
103                 'Content-Type': 'application/json'
104             },
105             body: JSON.stringify({studentId, newValue: classRoom})
106         });
107        const data = await response.json();
108        if (typeof data.status === 'number') {
109            if (data.status === 404) {
110                return [status: false, info: "Something's wrong"];
111            }
112            return data;
113        } catch (error) {
114            console.error(error);
115            return [status: false, info: "Something's wrong"];
116        }
117 }

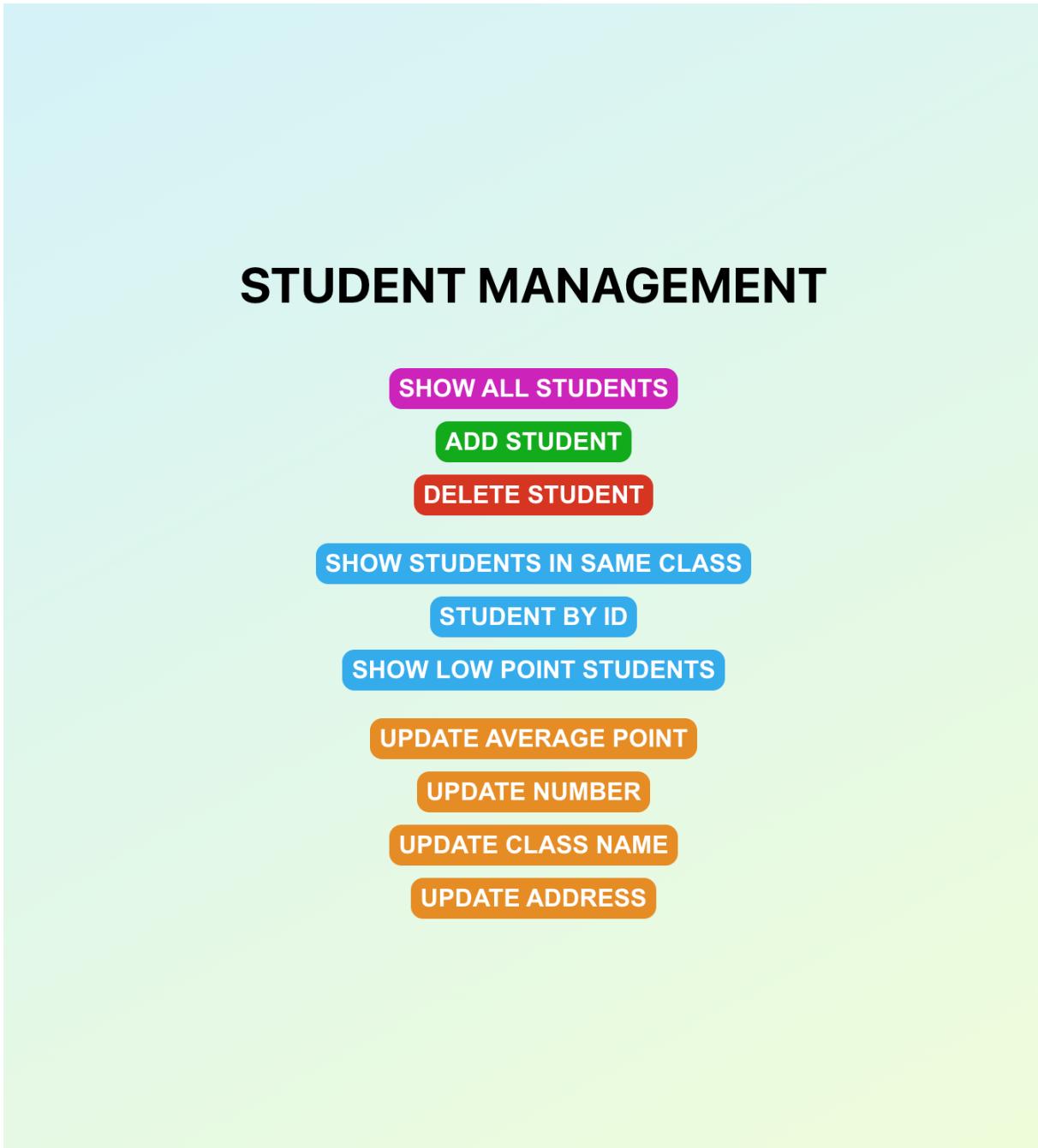
```

Sau khi deploy bằng heroku thì ta có một trang web cho *server-side* như sau: <https://student-management-server.herokuapp.com/>

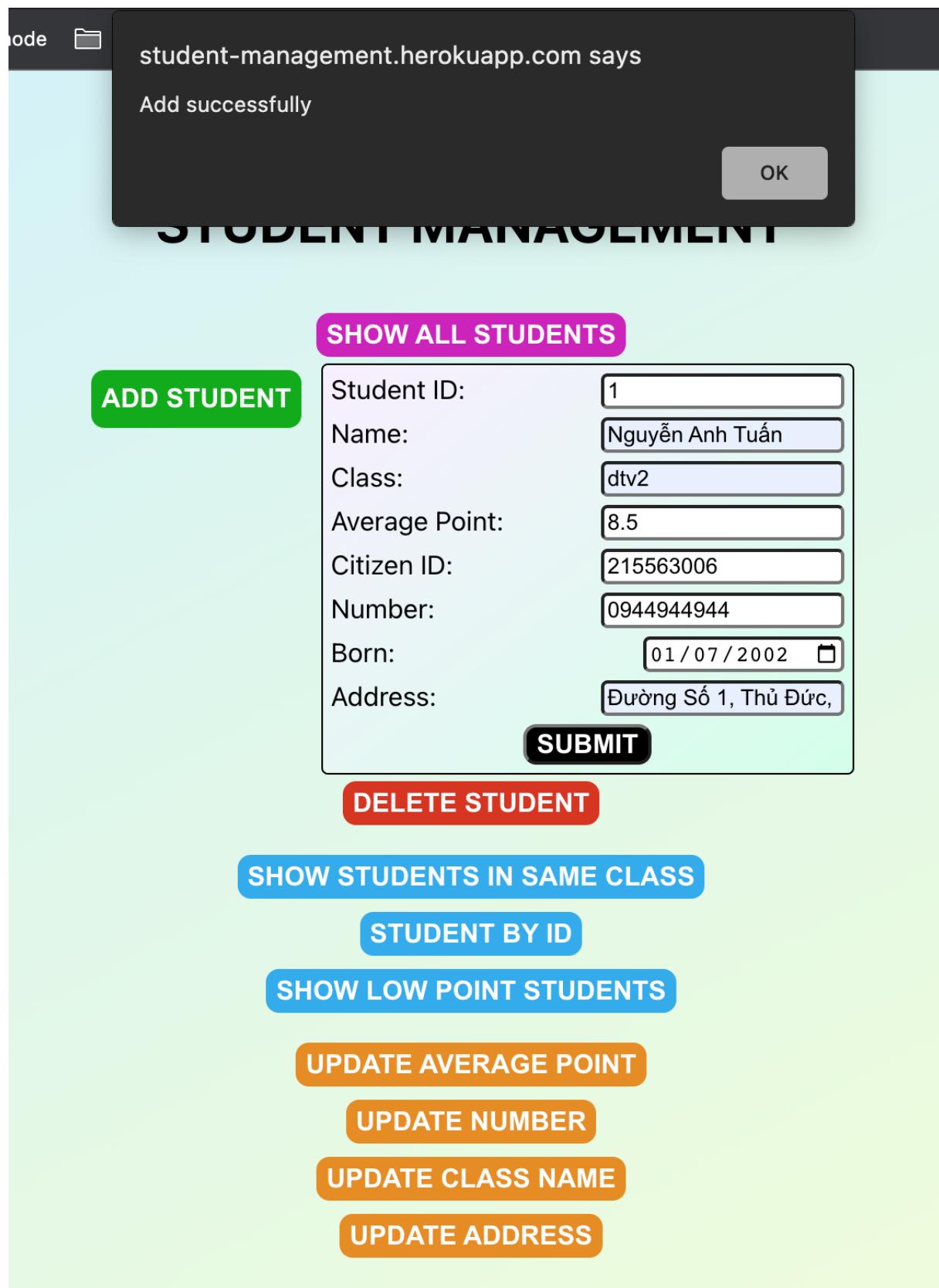
III. Minh họa trực quan

Link sản phẩm *client-side*: <https://student-management.herokuapp.com/> để có trải nghiệm thực tế tốt nhất.

3.1. Giao diện:



3.2. Thao tác thêm học sinh:



3.3. Xem danh sách học sinh:

STUDENT MANAGEMENT

Student ID	Name	Class	Average Point	Citizen ID	Contact Number	Date of Birth	Address
1	Nguyễn Anh Tuấn	dvt2	8.5	215563006	0944944944	2002-07-01	Đường Số 1, Thủ Đức, Hồ Chí Minh, 71317, Vietnam (Viet Nam)
2	Nguyễn Hữu Luật	dvt2	8	215563062	0999837625	2001-12-13	KTX khu B
3	Nguyễn Gia Phung	dvt2	8	215233062	0999827625	2002-12-11	Đồng Nai
4	Nguyễn Lê Minh Quang	dvt2	3	215925062	0989227625	2002-07-05	Vĩnh Long city

[SHOW ALL STUDENTS](#)

[ADD STUDENT](#)

[DELETE STUDENT](#)

[SHOW STUDENTS IN SAME CLASS](#)

[STUDENT BY ID](#)

[SHOW LOW POINT STUDENTS](#)

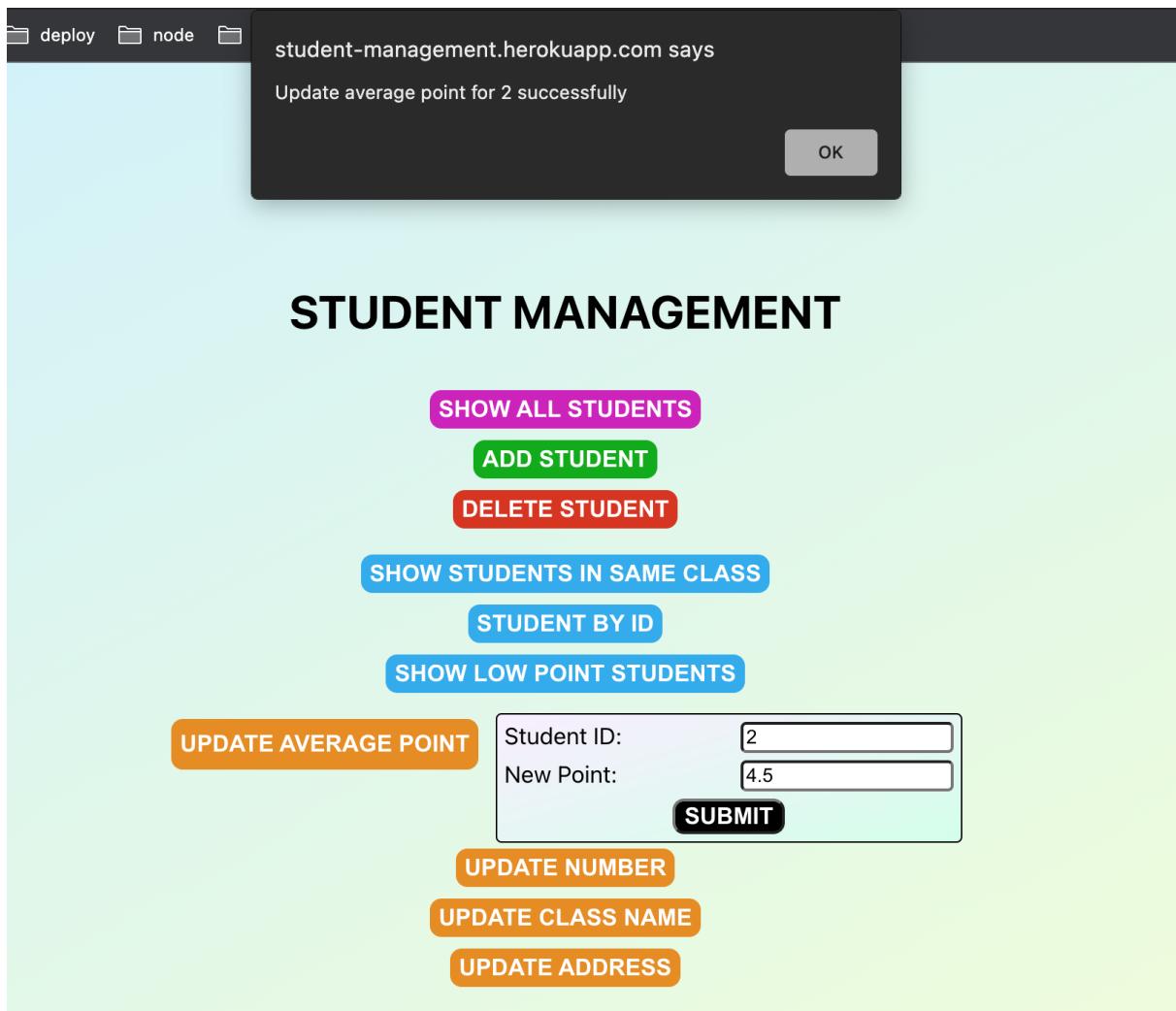
[UPDATE AVERAGE POINT](#)

[UPDATE NUMBER](#)

[UPDATE CLASS NAME](#)

[UPDATE ADDRESS](#)

3.4. Cập nhật điểm 4.5 cho học sinh có id là 2:



STUDENT MANAGEMENT							
Student ID	Name	Class	Average Point	Citizen ID	Contact Number	Date of Birth	Address
1	Nguyễn Anh Tuấn	dvt2	8.5	215563006	0944944944	2002-07-01	Đường Số 1, Thủ Đức, Hồ Chí Minh, 71317, Vietnam (Viet Nam)
2	Nguyễn Hữu Luật	dvt2	4.5	215563062	0999837625	2001-12-13	KTX khu B
3	Nguyễn Gia Phùng	dvt2	8	215233062	0999827625	2002-12-11	Đồng Nai
4	Nguyễn Lê Minh Quang	dvt2	3	215925062	0989227625	2002-07-05	Vĩnh Long city

Below the table is a copy of the "STUDENT MANAGEMENT" interface, identical to the one above but without the modal window.

IV. Tổng kết

4.1. Các tính chất OOP được sử dụng:

- Đóng gói: để bảo vệ dữ liệu thuộc tính và cung cấp các getter, setter để truy cập và chỉnh sửa chúng một cách độc lập, điều này giúp đảm bảo truy cập ổn định.
- Đa hình: các phương thức khởi tạo học sinh, người, response... với các thuộc tính truyền vào.
- Kế thừa: lớp student kế thừa các thuộc tính của person như name, citizenid, address và các method đóng gói khởi tạo liên quan.
- Trùu tượng: dùng interface mô tả các phương thức trừu tượng triển khai ở repository và service

Cách vận hành: trong cách hình thành mảng học sinh, mảng này dùng collection ArrayList với pattern Singleton sẽ luôn khởi chạy khi server còn chạy không reset, khi reload vẫn không bị mất dữ liệu không cần apply database như SQL,...

4.2. Công nghệ áp dụng:

- Ngôn ngữ: Java, Javascript
- Framework: Spring, Reactjs
- Hosting: Heroku

Họ tên	Công việc phân công	Mức độ hoàn thành
Nguyễn Anh Tuấn	Phân bổ công việc, đưa ra ý tưởng và code chính	10/10
Nguyễn Gia Phụng	Hỗ trợ và phát triển các tính năng giao diện	9/10

Nguyễn Hữu Luật	Xây dựng và đóng góp ý tưởng xây dựng giao diện + word	9/10
------------------------	---	------

Nhận xét giáo viên	
---------------------------	--

Đồ án này không tham khảo ở bất kỳ nguồn nào.