FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

CRUD WITH VUEJS & REST API

LẬP TRÌNH JAVA 6

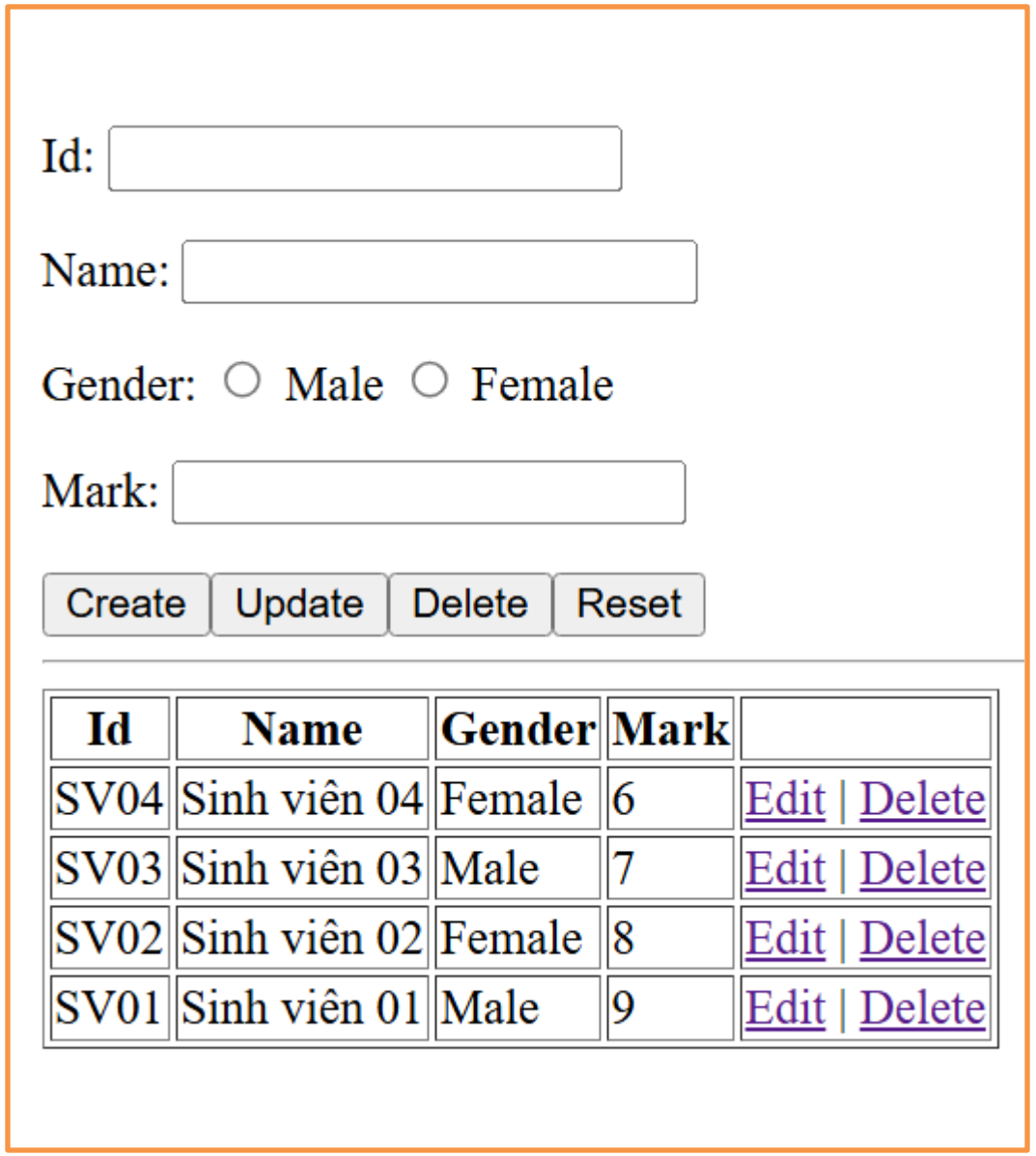
www.poly.edu.vn

NỘI DUNG

- Giới thiệu ứng dụng CRUD
- Tổ chức dự án CRUD
- Phân nhóm và nêu rõ các thành phần cần thiết
- Xây dựng Client (VueJS & Axios)
- Xây dựng Server (RestController & Service)
- Xây dựng DAO và tích hợp vào ứng dụng

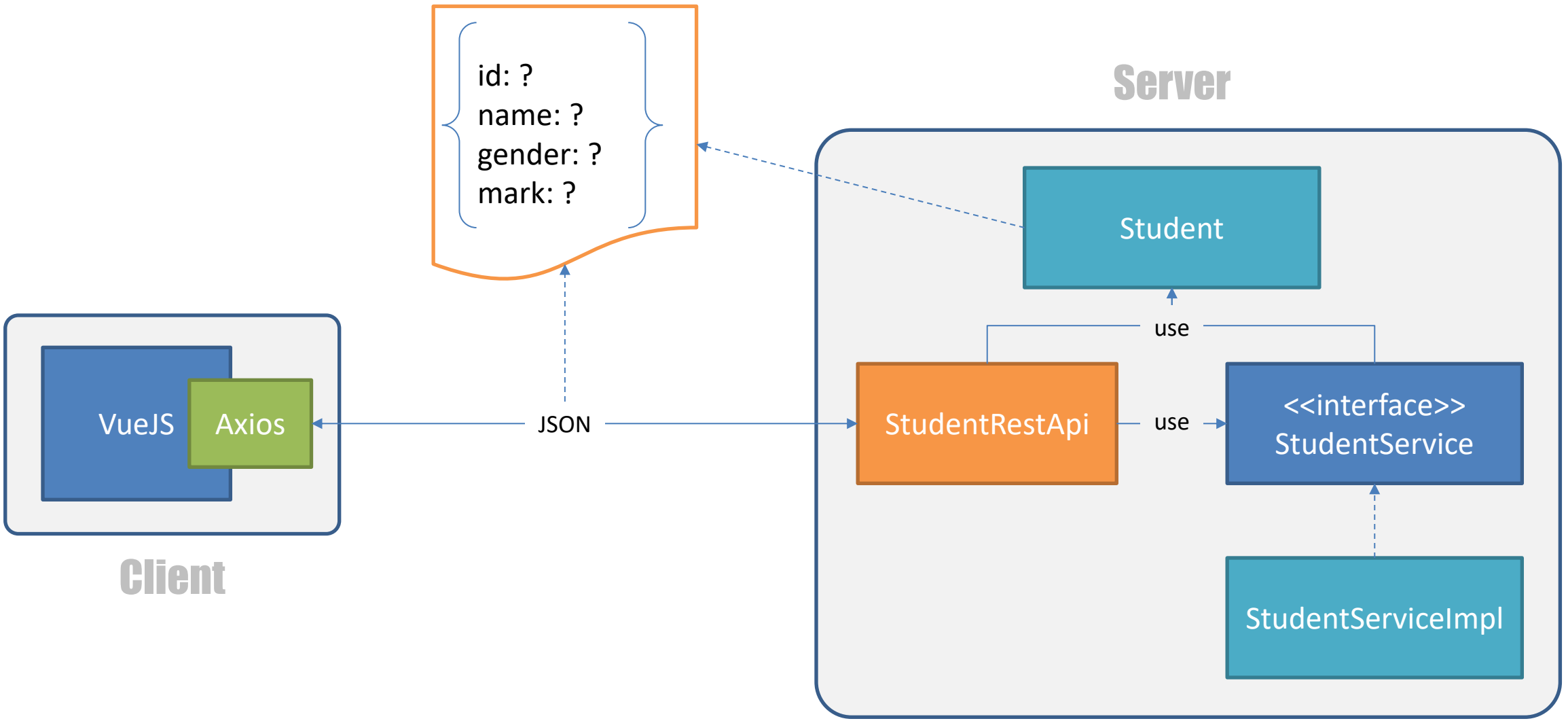
CRUD ĐƠN GIẢN

GIỚI THIỆU



Tương tác	Xử lý tương tác
Page.Open	• Hiển thị bảng • Xóa trắng form
Create.Click	• Thêm mới • Hiển thị lại bảng • Xóa trắng form
Update.Click	• Cập nhật • Hiển thị lại bảng • Giữ nguyên dữ liệu trên form
Delete.Click	• Xóa mục theo id trên form • Hiển thị lại bảng • Xóa trắng form
Reset.Click	• Xóa trắng form • Hiển thị lại bảng
Edit.Click	• Hiển thị mục chọn lên form • Hiển thị lại bảng

TỔ CHỨC ỨNG DỤNG



CÁC THÀNH PHẦN CẦN THỰC HIỆN

- Client
  - CrudView.vue
  - Axios libarary
- Server
  - Data Transfer Object (Student)
  - Rest API (StudentRestApi)
  - Spring Bean
    - StudentService
    - StudentServiceImpl

- Bước 1: Tạo dự án VueJS
- Bước 2: Cài đặt axios online
  - Nhúng thư viện axios vào index.html
    - <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
- Bước 3: Tạo SFC StudentView.vue trong thư mục components
- Bước 4: Nhúng SFC vào App.vue

CLIENT - APP.VUE

- Sửa đổi App.vue với nội dung đơn giản như sau. Trong đó SFC StudentView được import từ /components/StudentView.vue

```
<script setup>
        import StudentView from './components/StudentView.vue'
</script>
<template>
  <StudentView />
</template>
<style scoped></style>
```

CLIENT – STUDENTVIEW.VUE

- Cấu trúc StudentView.vue
  - <script setup></script>
    - Mã JavaScript xử lý tương tác người dung và hiển thị dữ liệu lên giao diện
  - <template></template>
    - Giao diện và kết nối dữ liệu lên giao diện
  - <style scoped></style>
    - Định dạng giao diện người dùng

```
<script setup>
    ...
</script>

<template>
  <!--FORM-->
  <div>...</div>
  <hr>
  <!--BẢNG-->
  <table border="1">...</table>
</template>

<style scoped></style>
```

CLIENT – STUDENTVIEW.VUE - SCRIPT

- Tổ chức mã phần <script> như hình bên.
- Mã các phương thức sẽ được cài đặt cụ thể sau khi hoàn thành StudentRestApi

```
import { ref } from "vue";
const form = ref({}) // dữ liệu buộc với form
const list = ref([]) // dữ liệu hiển thị lên bảng
const host = "http://localhost:8080"; // REST host
const ctrl = { // đối tượng điều hành
  init() {}, // khởi đầu trang web
  reset() {}, // xóa trắng form
  load() {}, // tải dữ liệu lên bảng
  edit(entity) {}, // tải dữ liệu lên form
  create() {}, // tạo thực thể mới từ dữ liệu form
  update() {}, // cập nhật thực thể từ dữ liệu form
  delete(entity) {} // xóa thực thể
}
ctrl.init();
```

CLIENT – STUDENTVIEW.VUE - FORM

- Buộc các thuộc tính dữ liệu vào các điều khiển của form và liên kết các phương thức xử lý với các nút để xử lý tương tác

```
<p>Id: <input v-model="form.id"></p>
<p>Name: <input v-model="form.name"></p>
<p>Gender:
  <input v-model="form.gender" type="radio" value="true"> Male
  <input v-model="form.gender" type="radio" value="false"> Female
</p>
<p>Mark: <input v-model="form.mark"></p>
<button @click="ctrl.create()">Create</button>
<button @click="ctrl.update()">Update</button>
<button @click="ctrl.delete(form)">Delete</button>
<button @click="ctrl.reset()">Reset</button>
```

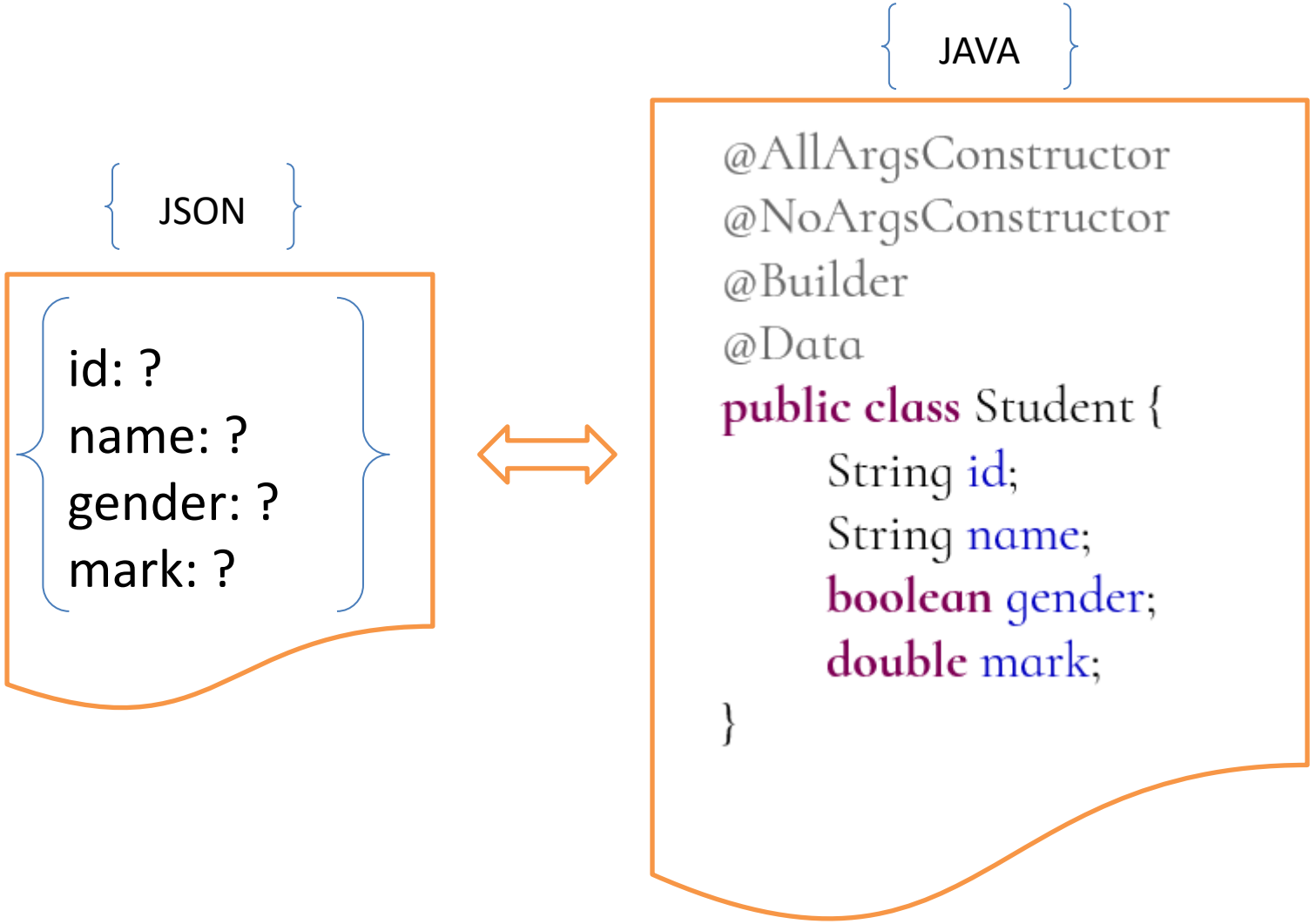
CLIENT – STUDENTVIEW.VUE – BẢNG

```
<table border="1">
  <thead>
    <tr><th>Id</th><th>Name</th><th>Gender</th><th>Mark</th><th></th></tr>
  </thead>
  <tbody>
    <tr v-for="e in list" :key="e.id">
      <td>{{ e.id }}</td>
      <td>{{ e.name }}</td>
      <td>{{ e.gender ? 'Male' : 'Female' }}</td>
      <td>{{ e.mark }}</td>
      <td>
        <a @click.stop.prevent="ctrl.edit(e)" href="">Edit</a> |
        <a @click.stop.prevent="ctrl.delete(e)" href="">Delete</a>
      </td>
    </tr>
  </tbody>
</table>
```

- Bước 1: Student
  - Mô tả cấu trúc dữ liệu JSON
- Bước 2: StudentRestApi
  - Các Endpoint các thiết cho client
- Bước 3: StudentService
  - Các phương thức CRUD phục cho StudentRestApi
- Bước 4: StudentServiceImpl
  - Cài đặt mã cho StudentService
- Bước 5: Hoàn thiện
  - Kết nối StudentService vào StudentRestApi

SERVER - STUDENT

- Student là lớp mô tả cấu trúc dữ liệu JSON để thực hiện chuyển đổi JSON nhận từ Client thành đối tượng Java và ngược lại chuyển đổi đối tượng Java thành JSON để gửi về Client



SERVER - STUDENTRESTAPI

```
@CrossOrigin("*")
@RestController
public class StudentRestApi {
    @GetMapping("students") //=> [student1, student2,...]
    public Collection<Student> findAll(){...}

    @GetMapping("students/{id}") //=> student
    public Student findById(@PathVariable("id") String id){...}

    @PostMapping("students") //=> student
    public Student create(@RequestBody Student student){...}

    @PutMapping("students/{id}") //=> student
    public Student update(@PathVariable("id") String id, @RequestBody Student student){...}

    @DeleteMapping("students/{id}") //=> nothing
    public void delete(@PathVariable("id") String id){...}
}
```

- *Endpoints*
  - *GET (http://.../students)*
  - *GET (http://.../students/{id})*
  - *POST (http://.../students, JSON)*
  - *PUT (http://.../students/{id}, JSON)*
  - *DELETE (http://.../students/{id})*
- *@RequestBody được sử dụng để yêu cầu Spring Boot chuyển đổi JSON gửi từ Client thành đối tượng Java*
- *Kết quả trả về của các phương thức điều khiển trong RestController sẽ được chuyển đổi thành JSON để gửi về cho Client*
- *Mã của các phương thức điều khiển sẽ được cài đặt sau*

SERVER – STUDENTSERVICE

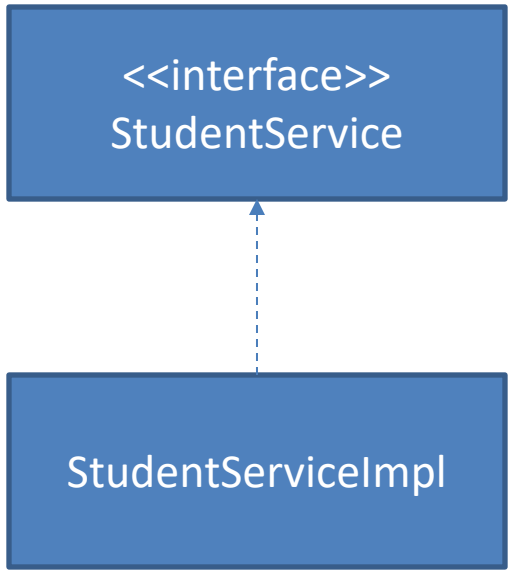
```
public interface StudentService {
        // Truy vấn tất cả
        Collection<Student> findAll();

        // Truy vấn theo khóa chính
        Student findById(String id);

        // Tạo mới
        Student create(Student student);

        // Cập nhật
        Student update(Student student);

        // Xóa theo khóa chính
        void deleteById(String id);
}
```



SERVER – STUDENTSERVICEIMPL

```
@Service
public class StudentServiceImpl implements StudentService{
    Map<String, Student> db = new HashMap<>(Map.of(
            "SV01", new Student("SV01", "Sinh viên 01", true, 9),
            "SV02", new Student("SV02", "Sinh viên 02", false, 8),
            "SV03", new Student("SV03", "Sinh viên 03", true, 7),
            "SV04", new Student("SV04", "Sinh viên 04", false, 6)
    ));
    @Override public Collection<Student> findAll() {...}
    @Override public Student findById(String id) {...}
    @Override public Student create(Student student) {...}
    @Override public Student update(Student student) {...}
    @Override public void deleteById(String id) {...}
}
```

Nguồn dữ liệu giả lập (cuối bài học sẽ áp dụng với CSDL)

Cài đặt mã cho các phương thức truy xuất dữ liệu ở slide sau sẽ hiện thực hóa hoạt động

SERVER – STUDENTSERVICEIMPL

```
@Override public Collection<Student> findAll() {
        return db.values();
}
@Override public Student findById(String id) {
        return db.get(id);
}
@Override public Student create(Student student) {
        return db.put(student.getId(), student);
}
@Override public Student update(Student student) {
        return db.put(student.getId(), student);
}
@Override public void deleteById(String id) {
        db.remove(id);
}
```

SERVER – HOÀN THIỆN

- Kết nối StudentService vào StudentRestApi để hoàn thiện mã phía server.
  - Tiêm (inject) StudentService vào StudentRestApi
    - @Autowired StudentService studentService;
  - Trong các phương thức điều khiển của StudentRestApi gọi các phương thức truy xuất dữ liệu của StudentService
    - findAll(): return studentService.findAll();
    - findById(): return studentService.findById(id);
    - create(): return studentService.create(student);
    - update(): return studentService.update(student);
    - delete(): return studentService.deleteById(id);

HOÀN THIỆN CLIENT

```
import { ref } from "vue";
const form = ref({}) // dữ liệu buộc với form
const list = ref([]) // dữ liệu hiển thị lên bảng
const host = "http://localhost:8080"; // REST host
const ctrl = { // đối tượng điều hành
  init() {}, // khởi đầu trang web
  reset() {}, // xóa trắng form
  load() {}, // tải dữ liệu lên bảng
  edit(entity) {}, // tải dữ liệu lên form
  create() {}, // tạo thực thể mới từ dữ liệu form
  update() {}, // cập nhật thực thể từ dữ liệu form
  delete(entity) {} // xóa thực thể
}
ctrl.init();
```

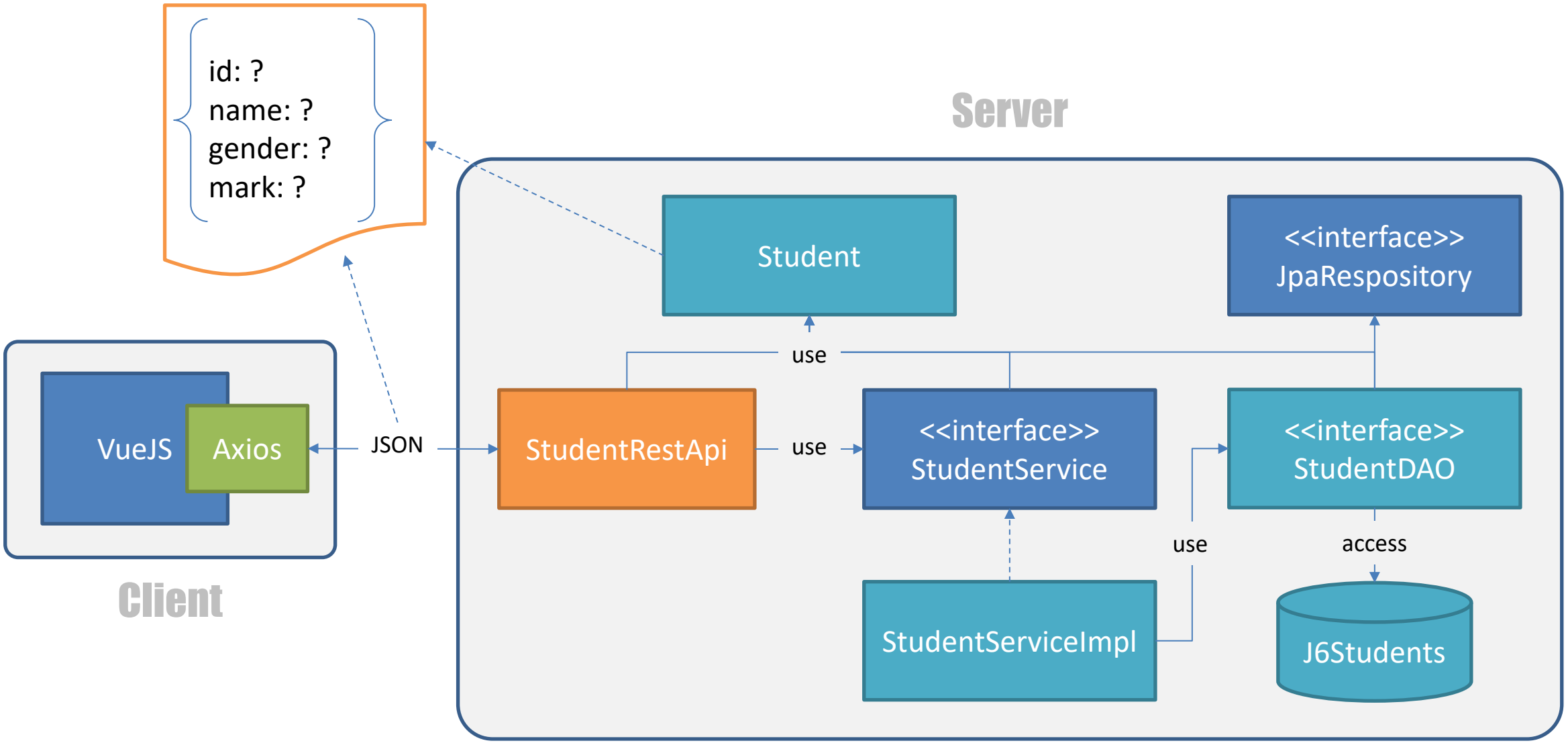
- Load()
  - url=`${host}/students`
  - **GET**(url)
- Edit()
  - url=`${host}/students/${id}`
  - **GET**(url)
- Create()
  - url=`${host}/students/${id}`
  - **POST**(url, form)
- Update()
  - url=`${host}/students/${id}`
  - **PUT**(url, form)
- Delete()
  - url=`${host}/students/${id}`
  - **DELETE**(url

HOÀN THIỆN CLIENT

```
init() { // khởi đầu trang web
        this.reset();
        this.load();
},
reset() { // xóa trắng form
        form.value = {id: "", name: "", gender: true, mark: 5}
},
load() { // tải dữ liệu lên bảng
        var url = `${host}/students`;
        axios.get(url).then(resp => {
                list.value = resp.data;
        })
},
edit(entity) { // tải dữ liệu lên form
        var url = `${host}/students/${entity.id}`;
        axios.get(url).then(resp => {
                form.value = resp.data;
        })
},
create() { // tạo thực thể mới từ dữ liệu form
        var entity = Object.assign({}, form.value);
        var url = `${host}/students`;
        axios.post(url, entity).then(resp => {
                this.load();
                this.reset();
        })
},
update() { // cập nhật thực thể từ dữ liệu form
        var entity = Object.assign({}, form.value);
        var url = `${host}/students/${entity.id}`;
        axios.put(url, entity).then(resp => {
                this.load();
        })
},
delete(entity) { // xóa thực thể
        var url = `${host}/students/${entity.id}`;
        axios.delete(url).then(resp => {
                this.load();
                this.reset();
        })
}
```

CRUD KẾT NỐI CSDL

TỔ CHỨC ỨNG DỤNG



CÁC THÀNH PHẦN CẦN THỰC HIỆN

- Client
  - CrudView.vue
  - Axios libarary
- Server
  - Rest API (StudentRestApi)
  - ***Data Transfer Object (Student)***
  - Spring Bean
    - StudentService
    - ***StudentServiceImpl***
- DAO
  - **Dependency (pom.xml)**
  - **Data Source (application.properties**
  - **Table (J6Students)**
  - **Entity (Student)**
  - **Data Access Object (StudentDAO)**

Chỉnh sửa để Student thực hiện cả 2 nhiệm vụ DTO và Entity

Chỉnh sửa StudentService để sử dụng StudentDAO làm việc với CSDL

Phần công việc mới

DAO - DEPENDENCY

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <scope>runtime</scope>
</dependency>
```

DAO – Table & Data Source

```
spring.datasource.url=jdbc:sqlserver://...;database=...;encrypt=true;trustServerCertificate=true;
spring.datasource.username=...
spring.datasource.password=...
spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver

spring.jpa.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect
spring.jpa.hibernate.ddl-auto = none
spring.jpa.show-sql=false
```

```sql
CREATE TABLE J6Students(
    Id VARCHAR(50) NOT NULL PRIMARY KEY,
    Name NVARCHAR(50) NOT NULL,
    Mark FLOAT NOT NULL,
    Gender BIT NOT NULL
)
```

DAO - ENTITY

```java
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Data
@Entity(name = "J6Students")
public class Student {
    @Id
    String id;
    String name;
    boolean gender;
    double mark;
}
```

Entity

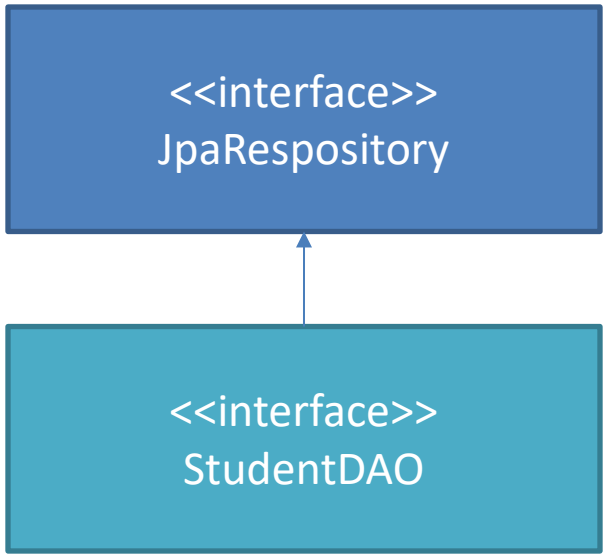
```sql
CREATE TABLE J6Students(
    Id VARCHAR(50) NOT NULL PRIMARY KEY,
    Name NVARCHAR(50) NOT NULL,
    Mark FLOAT NOT NULL,
    Gender BIT NOT NULL
)
```

DTO

JSON

```
id: ?
name: ?
gender: ?
mark: ?
```

DAO - StudentDAO




```
public interface StudentDAO
        extends JpaRepository<Student, String>{
}
```

JpaRepository<T, ID> (T là Entity và ID là kiểu của khóa chính) được cung cấp bởi Spring Boot. Các phương thức làm việc với CSDL đã được cài đặt sẵn, chúng ta chỉ thừa kế và cụ thể hóa thực thể T và thuộc tính khóa ID để truy xuất bảng dữ liệu cụ thể.

DAO – StudentServiceImpl

```
@Service public class StudentServiceImpl implements StudentService{
    @Autowired
    StudentDAO dao;
    @Override public List<Student> findAll() {
        return dao.findAll();
    }
    @Override public Student findById(String id) {
        return dao.findById(id).get();
    }
    @Override public Student create(Student student) {
        return dao.save(student);
    }
    @Override public Student update(Student student) {
        return dao.save(student);
    }
    @Override public void deleteById(String id) {
        dao.deleteById(id);
    }
}
```

- Tiêm DAO vào StudentServiceImpl
- Gọi các phương thức của dao để thực hiện các hoạt động truy xuất dữ liệu của dao.
- Các phương thực sau đây đã được cài đặt mã trong JpaRepository
  - findAll()
  - findById()
  - save()
  - deleteById()

SUMMARY

- ✓ Giới thiệu ứng dụng CRUD
- ✓ Tổ chức dự án CRUD
- ✓ Phân nhóm và nêu rõ các thành phần cần thiết
- ✓ Xây dựng Client (VueJS & Axios)
- ✓ Xây dựng Server (RestController & Service)
- ✓ Xây dựng DAO và tích hợp vào ứng dụng


TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
Cảm ơn