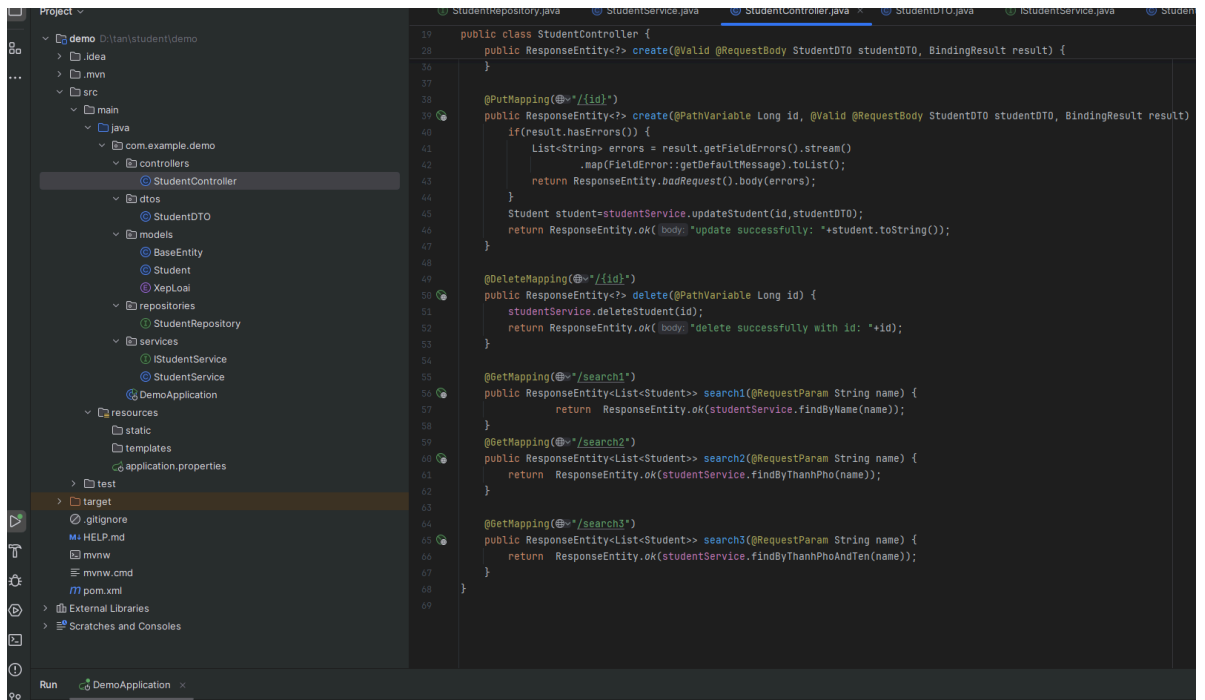
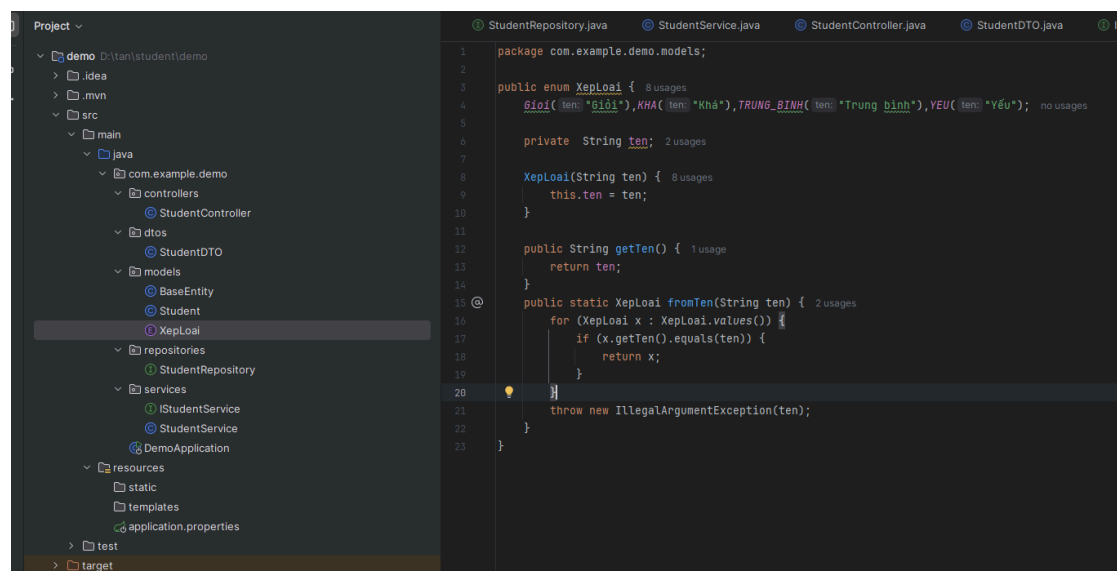


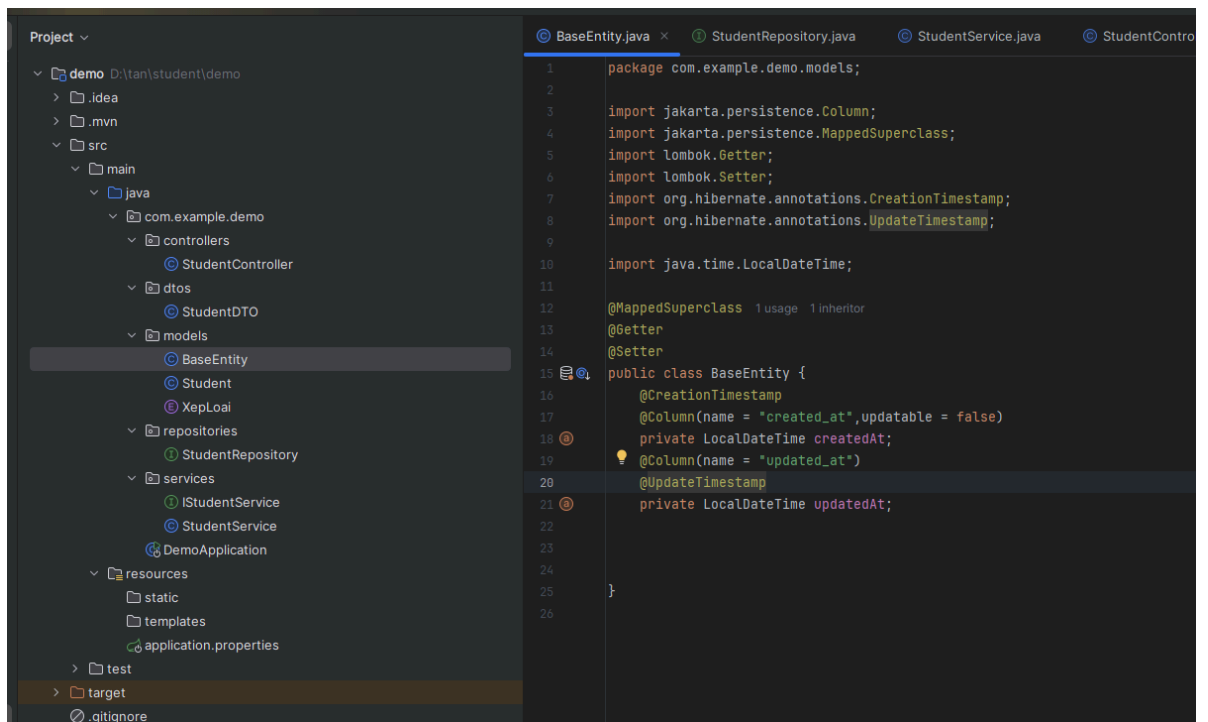
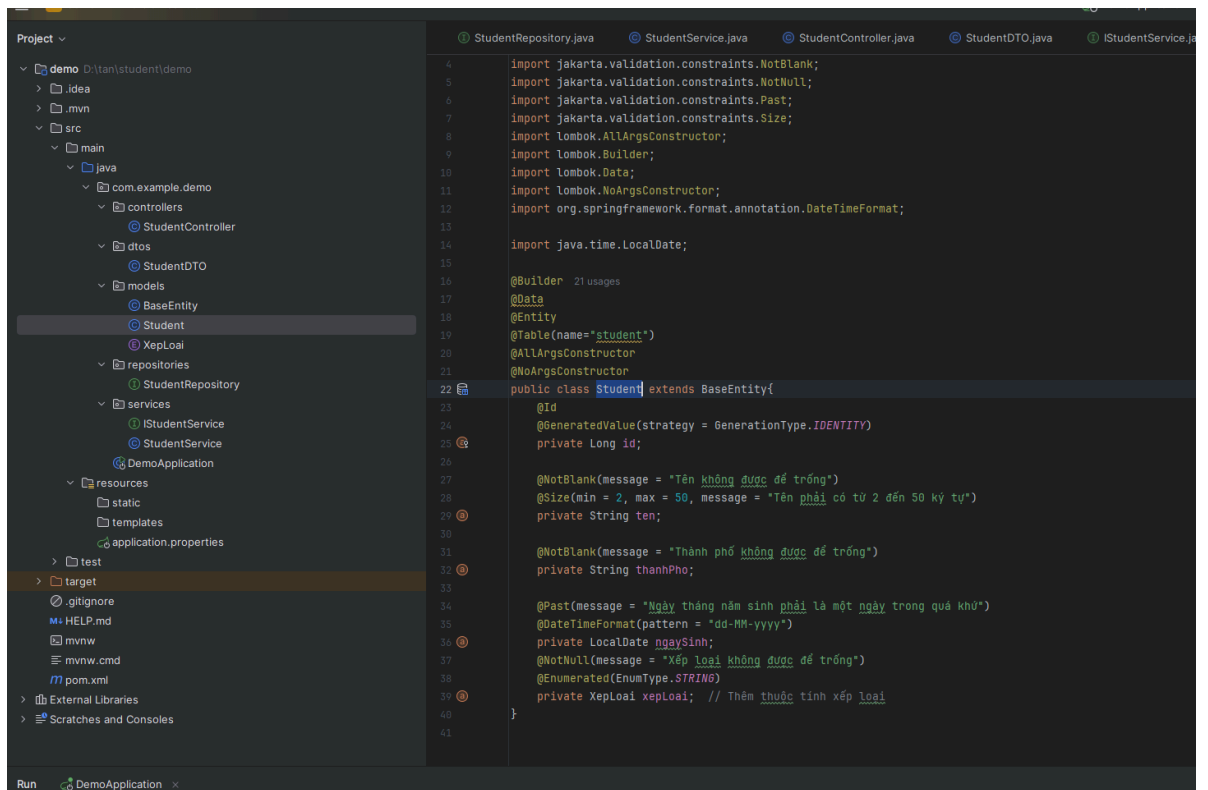
# Controller

```
StudentRepository.java StudentService.java StudentController.java x StudentDTO.java IStudentService.java S
1 package com.example.demo.controllers;
2
3 import com.example.demo.dtos.StudentDTO;
4 import com.example.demo.models.Student;
5 import com.example.demo.services.StudentService;
6 import jakarta.validation.Valid;
7 import lombok.RequiredArgsConstructor;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.validation.BindingResult;
10 import org.springframework.validation.FieldError;
11 import org.springframework.web.bind.annotation.*;
12
13 import java.util.ArrayList;
14 import java.util.List;
15
16 @RestController
17 @RequestMapping("/api/v1/admin/student")
18 @RequiredArgsConstructor
19 public class StudentController {
20     private final StudentService studentService;
21     @GetMapping("")
22     public ResponseEntity<List<Student>> index() {
23
24         return ResponseEntity.ok().body(studentService.getStudents());
25     }
26
27     @PostMapping()
28     public ResponseEntity<?> create(@Valid @RequestBody StudentDTO studentDTO, BindingResult result) {
29         if(result.hasErrors()) {
30             List<String> errors = result.getFieldErrors().stream()
31                 .map(FieldError::getDefaultMessage).toList();
32             return ResponseEntity.badRequest().body(errors);
33         }
34         Student student=studentService.saveStudent(studentDTO);
35         return ResponseEntity.ok( body: "Insert successfully: "+student.toString());
36     }
}
```

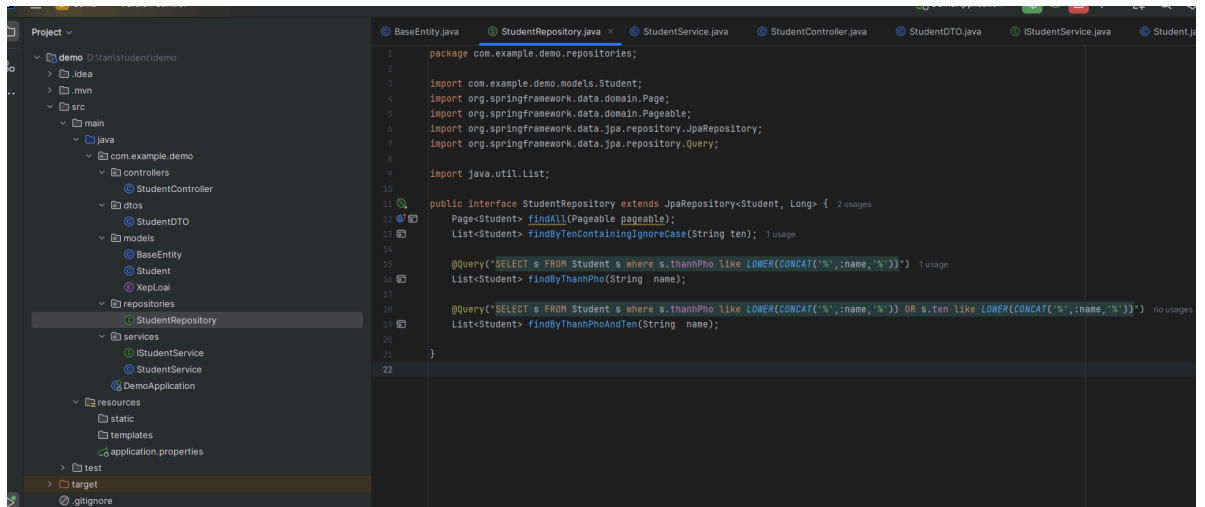


# Model

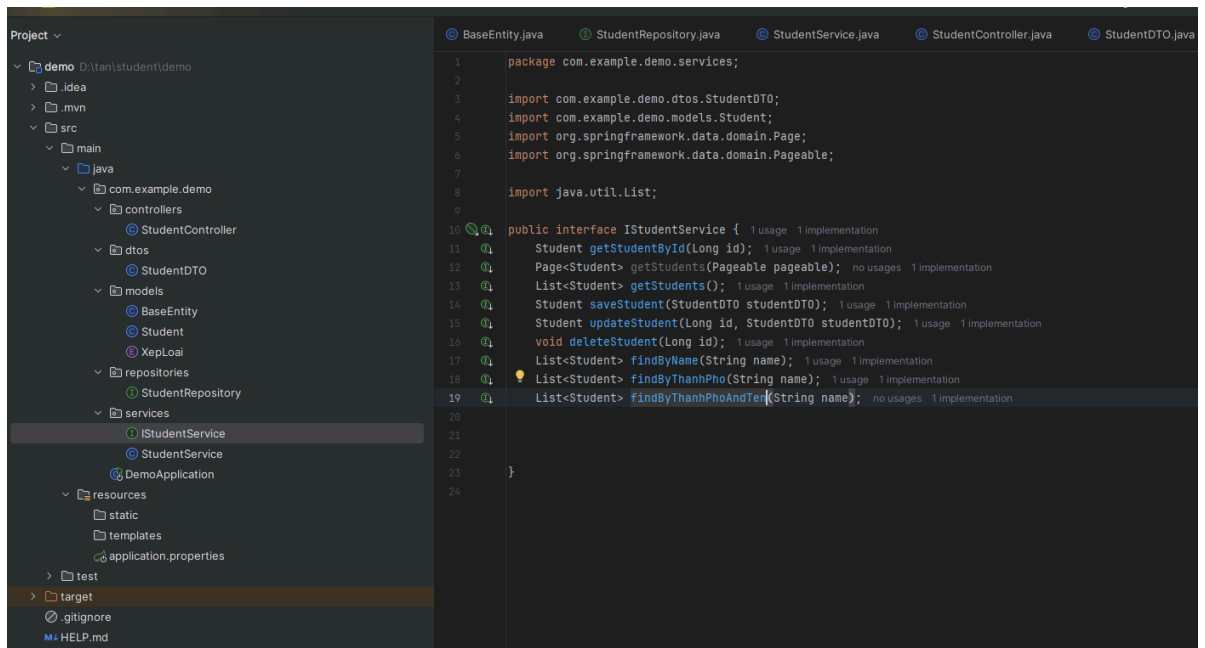


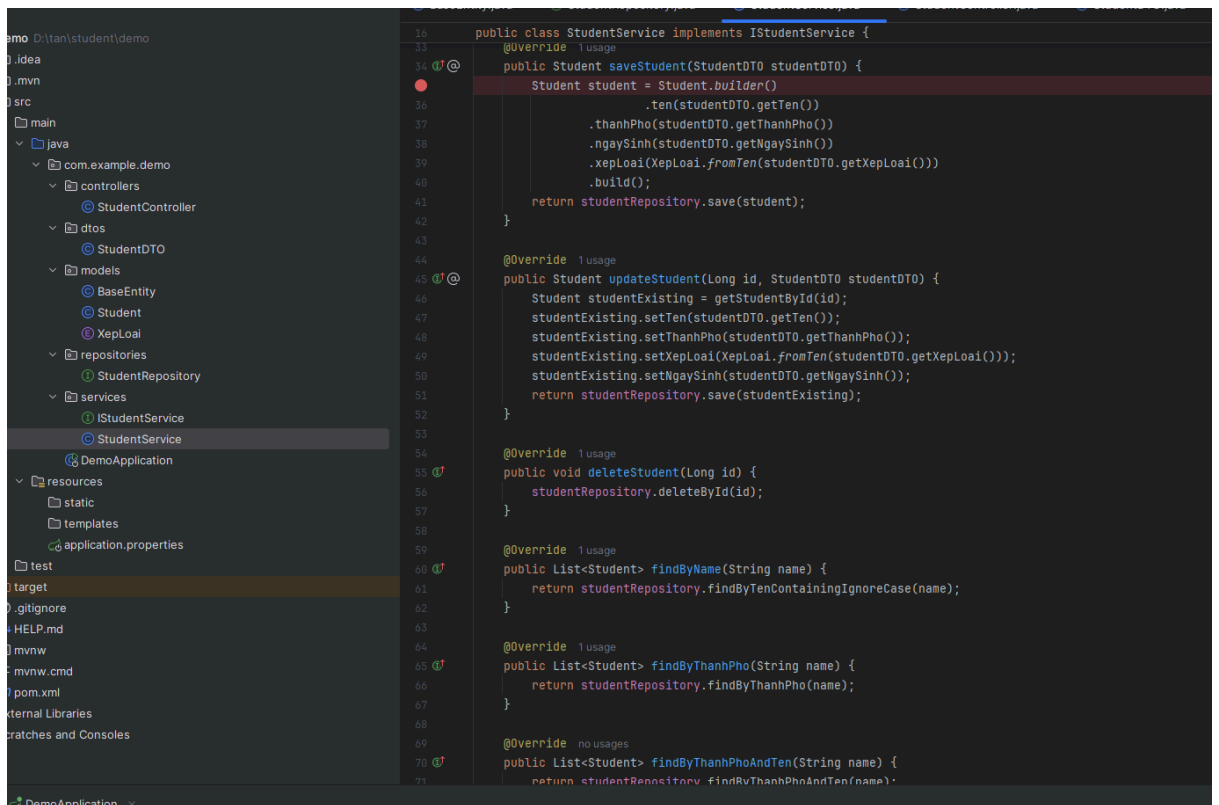
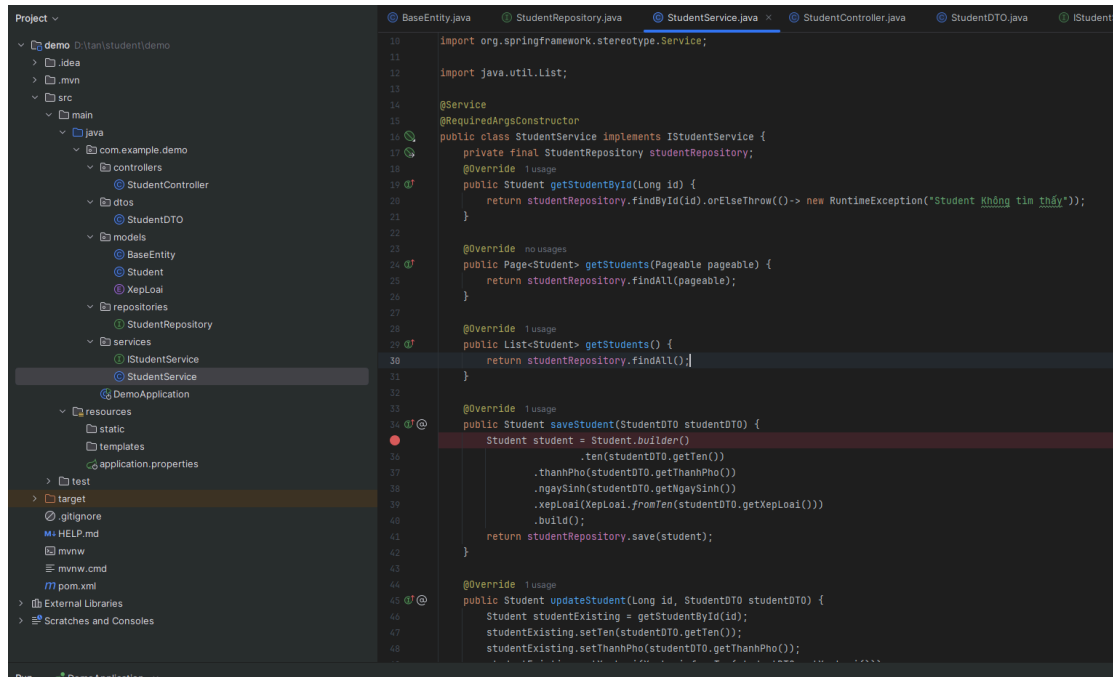


# Repository

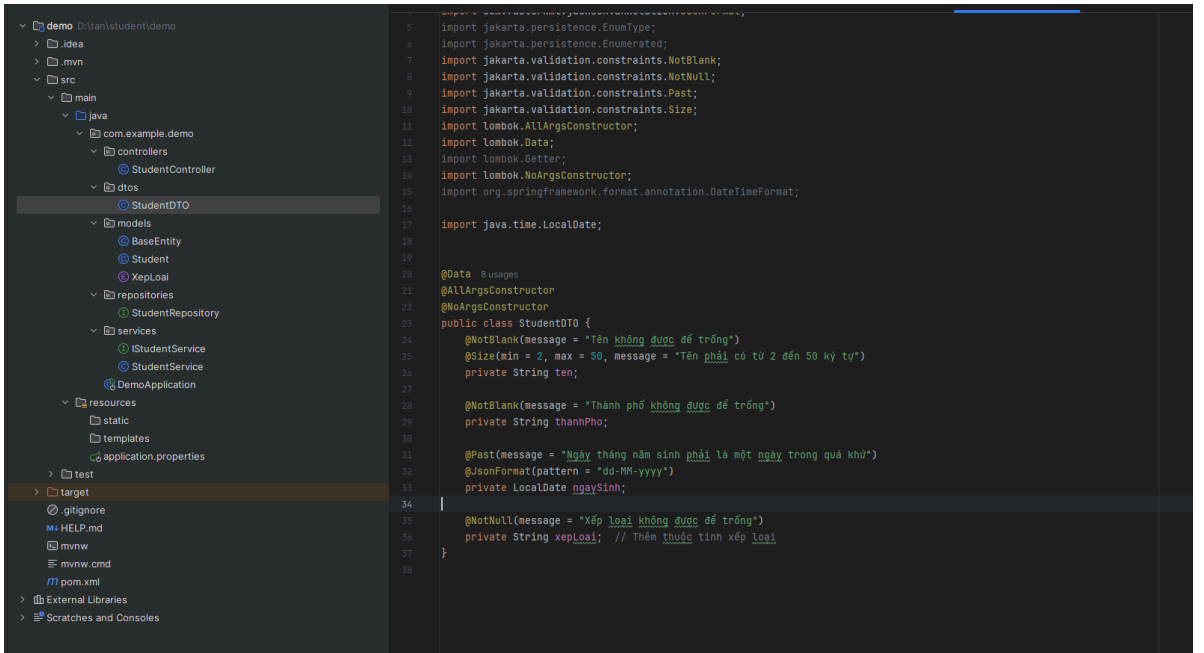


# service





# DTO



## api

ent / search city and name

▼

{{baseUrl}}/{{api\_prefix}}/admin/student/search3?name=ho

Authorization   Headers (7)   Body   Scripts   Settings

ams	
key	Value
name	ho
key	Value

PUT

▼

{{baseUrl}}/{{api\_prefix}}/admin/student/5

Params   Authorization   Headers (9)   Body ●   Scripts   Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON ▼

1

2

3

4

5

6

7

{

"ten": "string",

"thanhPho": "Hoa phượng đỏ",

"xepLoai": "Khá",

"ngaySinh": "08-08-2023"

}

1. q 회사명 : CÔNG TY CỔ PHẦN ASSURETEE VIỆT NAM

2. 사업분야

\* 보험 관련 스타트업 (거래중개, 금융 주선)

\* 전기오토바이 수입

\* IT개발 및 프로그래밍, 컨서팅

(자세한 업종은 아래 신고업종 확인) ; (

3. 웹사이트 : [www.assuretee.co.kr](http://www.assuretee.co.kr)

4. 주소

- HANOI HEAD OFFICE

Số nhà Sh02, khu đô thị mới Hạ Đình, Tân Triều, Thanh Trì, Hà Nội

- HCMC OFFICE

Tầng 2, 69 Đ. Võ Nguyên Giá



```
13 public class GlobalExceptionHandler {
14     @ExceptionHandler({})
15     public ResponseEntity<ApiResponse> handleGeneralException(Exception ex, HttpServletRequest request) {
16         ApiResponse response = ApiResponse.builder()
17             .status(HttpStatus.INTERNAL_SERVER_ERROR.value())
18             .message("An unexpected error occurred: " + ex.getMessage())
19             .data(null)
20             .build();
21         return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(response);
22     }
23
24     @ExceptionHandler({MethodArgumentNotValidException.class})
25     public ResponseEntity<ApiResponse> handleValidationException(MethodArgumentNotValidException ex) {
26         StringBuilder errorMessage = new StringBuilder();
27         ex.getBindingResult().getAllErrors().forEach((error) -> {
28             errorMessage.append(error.getDefaultMessage()).append("; ");
29         });
30
31         ApiResponse response = ApiResponse.builder()
32             .status(HttpStatus.BAD_REQUEST.value())
33             .message("Validation failed: " + errorMessage.toString())
34             .data(null)
35             .build();
36         return ResponseEntity.badRequest().body(response);
37     }
38
39     @ExceptionHandler({ResourceNotFoundException.class})
40     public ResponseEntity<ApiResponse> handleResourceNotFoundException(ResourceNotFoundException ex) {
41         ApiResponse response = ApiResponse.builder()
42             .status(HttpStatus.NOT_FOUND.value())
43             .message("Resource not found: " + ex.getMessage())
44             .data(null)
45             .build();
46         return ResponseEntity.status(HttpStatus.NOT_FOUND).body(response);
47     }
48 }
```

```
1 package com.example.demo.exceptions;
2
3 public class ResourceNotFoundException extends RuntimeException { 2 usages
4     public ResourceNotFoundException(String message) { no usages
5         super(message);
6     }
7 }
```

controllers

```
demo D:\tan\student\demo
├── .idea
├── .mvn
├── src
├── main
│   ├── java
│   │   ├── com.example.demo
│   │   │   ├── controllers
│   │   │   │   └── StudentController
│   │   │   ├── dtos
│   │   │   ├── exceptions
│   │   │   │   ├── GlobalExceptionHandler
│   │   │   │   └── ResourceNotFoundException
│   │   │   ├── models
│   │   │   │   ├── BaseEntity
│   │   │   │   ├── Student
│   │   │   │   └── XepLoai
│   │   │   ├── repositories
│   │   │   │   └── StudentRepository
│   │   │   ├── responses
│   │   │   │   ├── ApiResponse
│   │   │   │   └── ApiResponse
│   │   │   ├── services
│   │   │   │   ├── IStudentService
│   │   │   │   ├── StudentService
│   │   │   │   └── DemoApplication
│   │   ├── resources
│   │   │   ├── static
│   │   │   ├── templates
│   │   │   └── application.properties
│   ├── test
│   └── target
├── .gitignore
├── HELP.md
├── mvnw
├── mvnw.cmd
├── pom.xml
└── External Libraries

1 package com.example.demo.controllers;
2
3 import com.example.demo.dtos.StudentDTO;
4 import com.example.demo.exceptions.ResourceNotFoundException;
5 import com.example.demo.models.Student;
6 import com.example.demo.responses.ApiResponse;
7 import com.example.demo.services.StudentService;
8 import jakarta.validation.Valid;
9 import lombok.RequiredArgsConstructor;
10 import org.springframework.http.HttpStatus;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.validation.BindingResult;
13 import org.springframework.validation.FieldError;
14 import org.springframework.web.bind.annotation.*;
15
16 import java.util.ArrayList;
17 import java.util.List;
18
19 @RestController
20 @RequestMapping("/api/v1/admin/student")
21 @RequiredArgsConstructor
22 public class StudentController {
23     private final StudentService studentService;
24     @GetMapping("")
25     public ResponseEntity<ApiResponse> index() {
26         ApiResponse apiResponse = ApiResponse.builder()
27             .data(studentService.getStudents())
28             .status(HttpStatus.OK.value())
29             .message("OK")
30             .build();
31         return ResponseEntity.ok().body(apiResponse);
32     }
33
34     @PostMapping("")
35     public ResponseEntity<ApiResponse> create(@Valid @RequestBody StudentDTO studentDTO, BindingResult result) {
36         if(result.hasErrors()) {
37             List<String> errors = result.getFieldErrors().stream()
38                 .map(FieldError::getDefaultMessage).toList();
39             ApiResponse apiResponse = ApiResponse.builder()
40                 .data(errors)
41                 .message("Validation failed")
42                 .status(HttpStatus.BAD_REQUEST.value())
43                 .build();
44             return ResponseEntity.badRequest().body(apiResponse);
45         }
46         Student student = studentService.saveStudent(studentDTO);
47         ApiResponse apiResponse = ApiResponse.builder()
48             .data(student)
49             .message("Insert successfully")
50             .status(HttpStatus.OK.value())
51             .build();
52         return ResponseEntity.ok(apiResponse);
53     }
54
55     @PutMapping("/{id}")
56     public ResponseEntity<ApiResponse> update(@PathVariable Long id, @Valid @RequestBody StudentDTO studentDTO, BindingResult result) {
57         if(result.hasErrors()) {
58             List<String> errors = result.getFieldErrors().stream()
59                 .map(FieldError::getDefaultMessage).toList();
60             ApiResponse apiResponse = ApiResponse.builder()
61                 .data(errors)
62                 .message("Validation failed")
63                 .status(HttpStatus.BAD_REQUEST.value())
64                 .build();
65             return ResponseEntity.badRequest().body(apiResponse);
66         }
67         Student student = studentService.updateStudent(id, studentDTO);
68         if (student == null) {
69             throw new ResourceNotFoundException("Student không tìm thấy với id: " + id);
70         }
71     }
72 }
```

```
demo D:\tan\student\demo
├── .idea
├── .mvn
├── src
├── main
│   ├── java
│   │   ├── com.example.demo
│   │   │   ├── controllers
│   │   │   │   └── StudentController
│   │   │   ├── dtos
│   │   │   ├── exceptions
│   │   │   │   ├── GlobalExceptionHandler
│   │   │   │   └── ResourceNotFoundException
│   │   │   ├── models
│   │   │   │   ├── BaseEntity
│   │   │   │   ├── Student
│   │   │   │   └── XepLoai
│   │   │   ├── repositories
│   │   │   │   └── StudentRepository
│   │   │   ├── responses
│   │   │   │   ├── ApiResponse
│   │   │   │   └── ApiResponse
│   │   │   ├── services
│   │   │   │   ├── IStudentService
│   │   │   │   ├── StudentService
│   │   │   │   └── DemoApplication
│   │   ├── resources
│   │   │   ├── static
│   │   │   ├── templates
│   │   │   └── application.properties
│   ├── test
│   └── target
├── .gitignore
├── HELP.md
├── mvnw
├── mvnw.cmd
├── pom.xml
└── External Libraries

22 public class StudentController {
23     }
24
25     @PostMapping("")
26     public ResponseEntity<ApiResponse> create(@Valid @RequestBody StudentDTO studentDTO, BindingResult result) {
27         if(result.hasErrors()) {
28             List<String> errors = result.getFieldErrors().stream()
29                 .map(FieldError::getDefaultMessage).toList();
30             ApiResponse apiResponse = ApiResponse.builder()
31                 .data(errors)
32                 .message("Validation failed")
33                 .status(HttpStatus.BAD_REQUEST.value())
34                 .build();
35             return ResponseEntity.badRequest().body(apiResponse);
36         }
37         Student student = studentService.saveStudent(studentDTO);
38         ApiResponse apiResponse = ApiResponse.builder()
39             .data(student)
40             .message("Insert successfully")
41             .status(HttpStatus.OK.value())
42             .build();
43         return ResponseEntity.ok(apiResponse);
44     }
45
46     @PutMapping("/{id}")
47     public ResponseEntity<ApiResponse> update(@PathVariable Long id, @Valid @RequestBody StudentDTO studentDTO, BindingResult result) {
48         if(result.hasErrors()) {
49             List<String> errors = result.getFieldErrors().stream()
50                 .map(FieldError::getDefaultMessage).toList();
51             ApiResponse apiResponse = ApiResponse.builder()
52                 .data(errors)
53                 .message("Validation failed")
54                 .status(HttpStatus.BAD_REQUEST.value())
55                 .build();
56             return ResponseEntity.badRequest().body(apiResponse);
57         }
58         Student student = studentService.updateStudent(id, studentDTO);
59         if (student == null) {
60             throw new ResourceNotFoundException("Student không tìm thấy với id: " + id);
61         }
62     }
63 }
```

```

@PutMapping("/{id}")
public ResponseEntity<ApiResponse> create(@PathVariable Long id, @Valid @RequestBody StudentDTO studentDTO, BindingResult result) {
    if(result.hasErrors()) {
        List<String> errors = result.getFieldErrors().stream()
            .map(FieldError::getDefaultMessage).toList();
        ApiResponse apiResponse = ApiResponse.builder()
            .data(errors)
            .message("Validation failed")
            .status(HttpStatus.BAD_REQUEST.value())
            .build();
        return ResponseEntity.badRequest().body(apiResponse);
    }
    Student student = studentService.updateStudent(id, studentDTO);
    if (student == null) {
        throw new ResourceNotFoundException("Student không tìm thấy với id: " + id);
    }
    ApiResponse apiResponse = ApiResponse.builder()
        .data(student)
        .message("Update successfully")
        .status(HttpStatus.OK.value())
        .build();
    return ResponseEntity.ok(apiResponse);
}

@DeleteMapping("/{id}")
public ResponseEntity<ApiResponse> delete(@PathVariable Long id) {
    Student student = studentService.getStudentById(id);
    if (student == null) {
        throw new ResourceNotFoundException("Student không tìm thấy với id : " + id);
    }
    studentService.deleteStudent(id);
    ApiResponse apiResponse = ApiResponse.builder()
        .data(null)
        .message("delete successfully")
        .status(HttpStatus.OK.value())
        .build();
    return ResponseEntity.ok(apiResponse);
}

```

```

@DeleteMapping(Ⓜ"/{id}")
public ResponseEntity<ApiResponse> delete(@PathVariable Long id) {
    Student student = studentService.getById(id);
    if (student == null) {
        throw new ResourceNotFoundException("Student không tìm thấy với id : " + id);
    }
    studentService.deleteStudent(id);
    ApiResponse apiResponse = ApiResponse.builder()
        .data(null)
        .message("delete successfully")
        .status(HttpStatus.OK.value())
        .build();
    return ResponseEntity.ok(apiResponse);
}

@GetMapping(Ⓜ"/search1")
public ResponseEntity<ApiResponse> search1(@RequestParam String name) {
    ApiResponse apiResponse = ApiResponse.builder()
        .data(studentService.findByName(name))
        .message("Search successfully")
        .status(HttpStatus.OK.value())
        .build();
    return ResponseEntity.ok(apiResponse);
}

@GetMapping(Ⓜ"/search2")
public ResponseEntity<ApiResponse> search2(@RequestParam String name) {
    ApiResponse apiResponse = ApiResponse.builder()
        .data(studentService.findByThanhPho(name))
        .message("Search successfully")
        .status(HttpStatus.OK.value())
        .build();
    return ResponseEntity.ok(apiResponse);
}

@GetMapping(Ⓜ"/search3")

```

# Phân trang

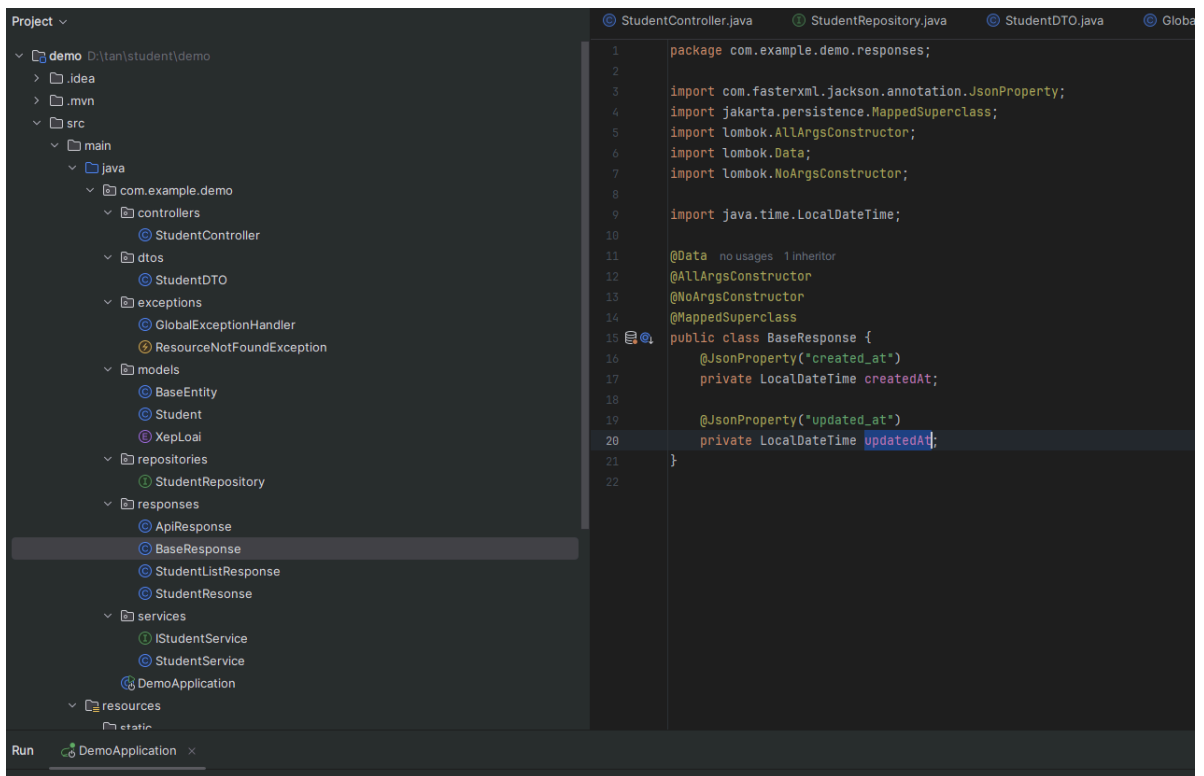
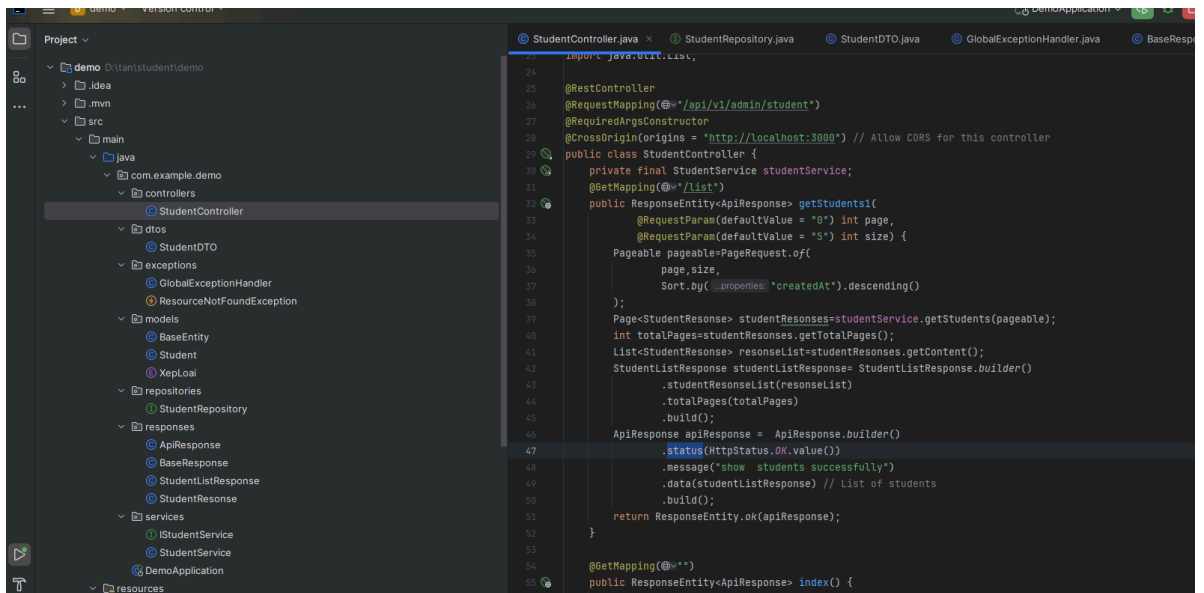
The screenshot shows an IDE with a project named 'demo' at 'D:\tan\student\demo'. The file explorer on the left shows the project structure, with 'StudentService' selected under 'services'. The main editor displays the implementation of 'StudentService' in 'StudentService.java'. The code includes imports for 'List' and 'Pageable', and implements methods for finding a student by ID, finding all students, and saving a student. The 'saveStudent' method uses a 'StudentBuilder' to construct a 'Student' object from a 'StudentDTO'.

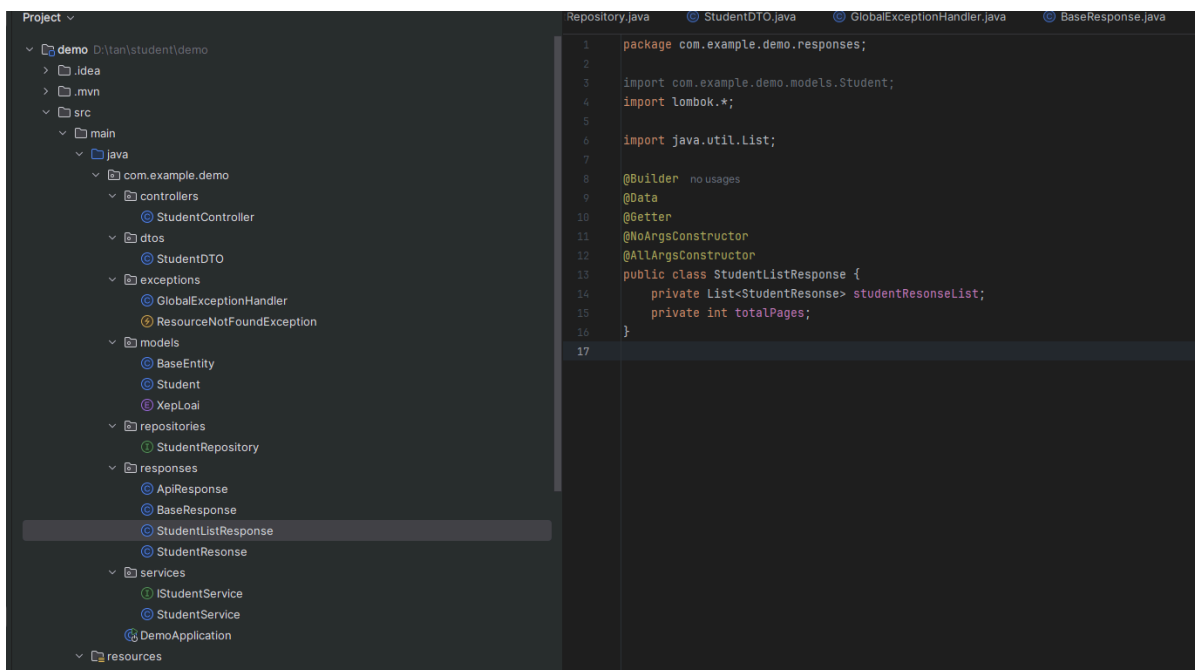
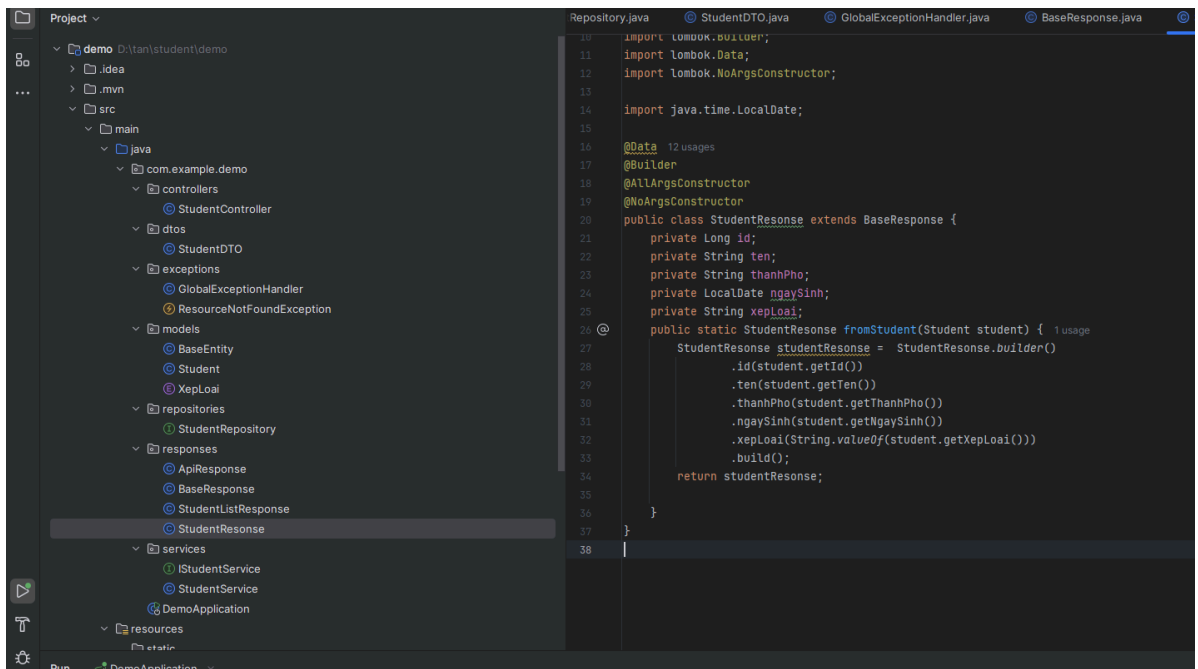
```
11
12 import java.util.List;
13
14 @Service
15 @RequiredArgsConstructor
16 public class StudentService implements IStudentService {
17     private final StudentRepository studentRepository;
18     @Override 1 usage
19     public Student getStudentById(Long id) {
20         return studentRepository.findById(id).get();
21     }
22
23     @Override no usages
24     public Page<Student> getStudents(Pageable pageable) {
25         return studentRepository.findAll(pageable);
26     }
27
28     @Override 1 usage
29     public List<Student> getStudents() {
30         return studentRepository.findAll();
31     }
32
33     @Override 1 usage
34     public Student saveStudent(StudentDTO studentDTO) {
35         Student student = Student.builder()
36             .ten(studentDTO.getTen())
37             .thanhPho(studentDTO.getThanhPho())
38             .ngaySinh(studentDTO.getNgaySinh())
39             .xepLoai(XepLoai.fromTen(studentDTO.getXepLoai()))
40             .build();
41         return studentRepository.save(student);
42     }
43
44     @Override 1 usage
```

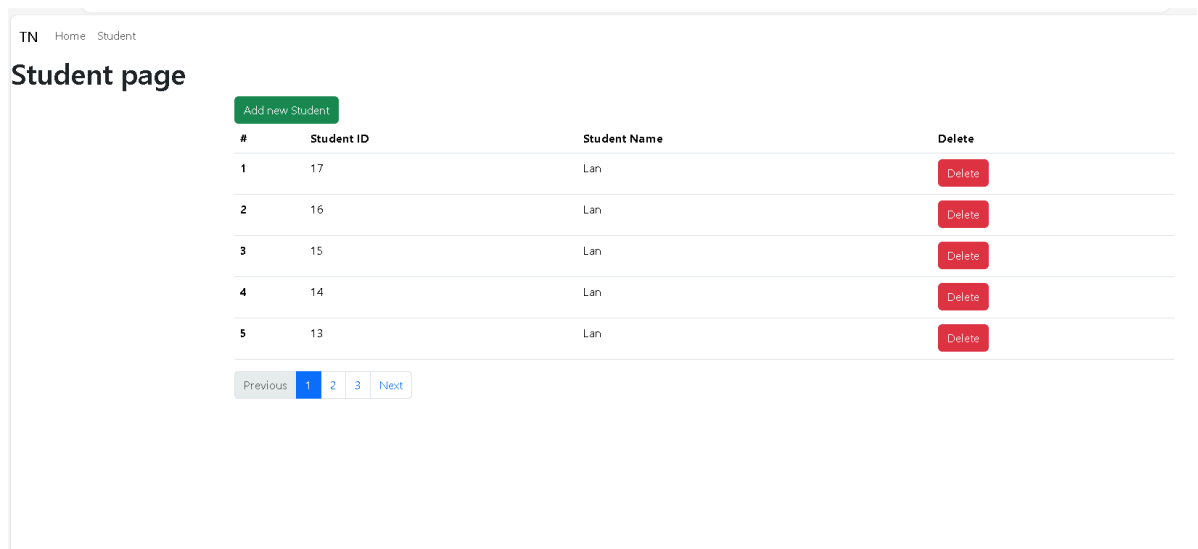
The screenshot shows the same IDE with the 'StudentController' selected in the file explorer. The main editor displays the implementation of 'StudentController' in 'StudentController.java'. The code includes imports for 'ArrayList' and 'List', and implements methods for getting students and an index page. The 'getStudents' method uses 'PageRequest' to create a 'Pageable' object and returns a 'ResponseEntity' containing the list of students. The 'index' method returns a 'ResponseEntity' with a status of 'OK' and a message.

```
19 import java.util.ArrayList;
20 import java.util.List;
21
22 @RestController
23 @RequestMapping("/api/v1/admin/student")
24 @RequiredArgsConstructor
25 public class StudentController {
26     private final StudentService studentService;
27     @GetMapping("/{list}")
28     public ResponseEntity<ApiResponse> getStudents(
29         @RequestParam(defaultValue = "0") int page,
30         @RequestParam(defaultValue = "5") int size) {
31         Pageable pageable = PageRequest.of(page, size);
32         Page<Student> studentPage = studentService.getStudents(pageable);
33
34         ApiResponse apiResponse = ApiResponse.builder()
35             .status(HttpStatus.OK.value())
36             .message("show students successfully")
37             .data(studentPage.getContent()) // List of students
38             .build();
39         return ResponseEntity.ok(apiResponse);
40     }
41
42     @GetMapping("/")
43     public ResponseEntity<ApiResponse> index() {
44         ApiResponse apiResponse = ApiResponse.builder()
45             .data(studentService.getStudents())
46             .status(HttpStatus.OK.value())
47             .message("OK")
48             .build();
49     }
50 }
```

Phân trang - bản cuối







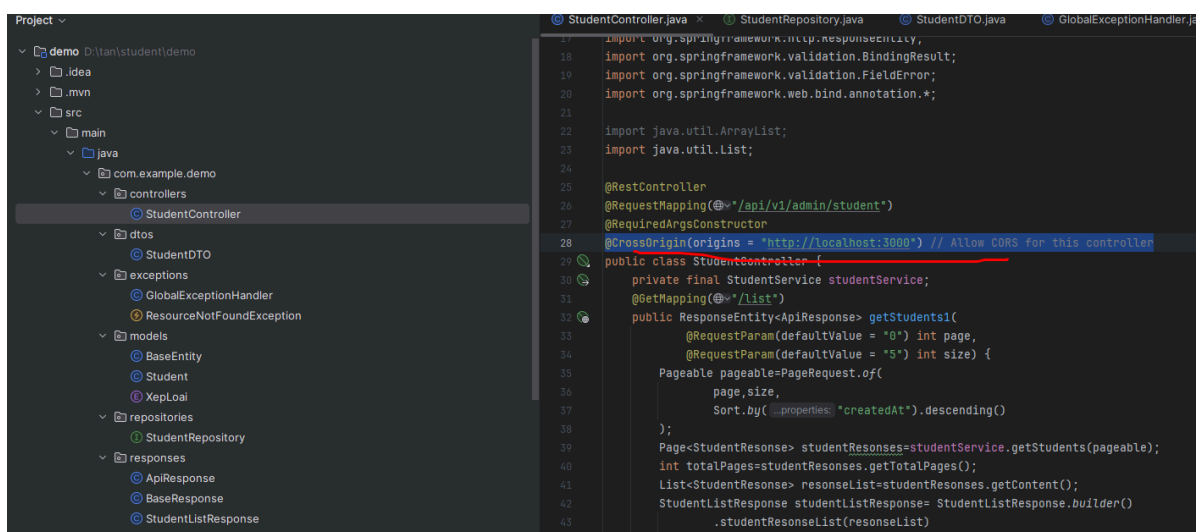
[https://github.com/ngoctan1234/BaitapStudent\\_Frontend.git](https://github.com/ngoctan1234/BaitapStudent_Frontend.git)

Cài: Reactstrap, axios, react-router-dom, redux toolkit, pagination

npm i reactstrap axios react-router-dom  
redux-toolkit react-paginate @reduxjs/toolkit  
cors



- **CORS** (Cross-Origin Resource Sharing) là một tính năng bảo mật được trình duyệt web triển khai nhằm ngăn chặn các ứng dụng web gửi yêu cầu tới một miền khác với miền đã phục vụ trang web. Chính sách này được thực thi bởi trình duyệt để bảo vệ người dùng khỏi các hành động có thể gây hại, như các cuộc tấn công giả mạo yêu cầu giữa các trang (CSRF).
- **Bảo Mật:** CORS rất quan trọng để ngăn chặn việc truy cập trái phép vào tài nguyên từ các nguồn khác, bảo vệ người dùng và máy chủ khỏi các cuộc tấn công giữa các trang.
- **Tính Linh Hoạt:** CORS được cấu hình đúng cho phép các ứng dụng web tương tác với các tài nguyên được lưu trữ trên các miền khác nhau trong khi vẫn duy trì được sự an toàn.



The screenshot shows an IDE with a project named 'demo' and a code file 'StudentController.java'. The project structure on the left includes folders for 'idea', '.mvn', 'src', 'main', 'java', 'com.example.demo', 'controllers', 'dtos', 'exceptions', 'models', 'repositories', and 'responses'. The 'StudentController.java' file is open, showing imports for Spring Framework and utility classes. The code includes a REST controller with a mapping to '/api/v1/admin/student' and a CORS configuration allowing requests from 'http://localhost:3000'. The controller has a method 'getStudents1' that returns a paginated list of students.

```
27 import org.springframework.http.ResponseEntity;
28 import org.springframework.validation.BindingResult;
29 import org.springframework.validation.FieldError;
30 import org.springframework.web.bind.annotation.*;
31
32 import java.util.ArrayList;
33 import java.util.List;
34
35 @RestController
36 @RequestMapping("/api/v1/admin/student")
37 @RequiredArgsConstructor
38 @CrossOrigin(origins = "http://localhost:3000") // Allow CORS for this controller
39 public class StudentController {
40     private final StudentService studentService;
41
42     @GetMapping("/list")
43     public ResponseEntity<ApiResponse> getStudents1(
44         @RequestParam(defaultValue = "0") int page,
45         @RequestParam(defaultValue = "5") int size) {
46         Pageable pageable = PageRequest.of(
47             page, size,
48             Sort.by( ...properties).descending()
49         );
50         Page<StudentResponse> studentResponses = studentService.getStudents(pageable);
51         int totalPages = studentResponses.getTotalPages();
52         List<StudentResponse> responseList = studentResponses.getContent();
53         StudentListResponse studentListResponse = StudentListResponse.builder()
54             .studentResponseList(responseList)
55             .totalPages(totalPages)
56             .build();
57         return ResponseEntity.ok(studentListResponse);
58     }
59 }
```

The screenshot shows a web browser window with two tabs: "TN" and "Student". The address bar displays "http://localhost:3000/". The main content area shows a "Student page" with a "Total: 10" label. Below this is a table with three columns: "#", "Tên sinh viên", and "Thành phố". The first row of the table has a blue highlight under the number "1". To the right of the table is a "Delete" button.

The bottom half of the browser window shows the developer console. It contains several error messages related to CORS policy violations. The first message states: "Access to XMLHttpRequest at 'http://localhost:3000/api/csls/student/{id?name=MsIcs}' from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource." This message is repeated twice. A second message indicates a 404 Not Found status: "GET http://localhost:3000/api/csls/admns/student/{id?name=MSIcs} net::ERR\_FAILED 200 (OK)". At the bottom of the console, there is a tip: "[NEW] Explain Console errors by using Copilot in Edge: click ⓘ to explain an error. Learn more".

Trịnh Bình Minh đã xóa file ngày xưa của th.