



FPT POLYTECHNIC



Conceive Design Implement Operate

ADVANCED RESTFUL API

GIẢNG VIÊN: NGUYỄN NGHIỆM

www.poly.edu.vn



- Spring Boot REST API



- REST API with JpaRepository



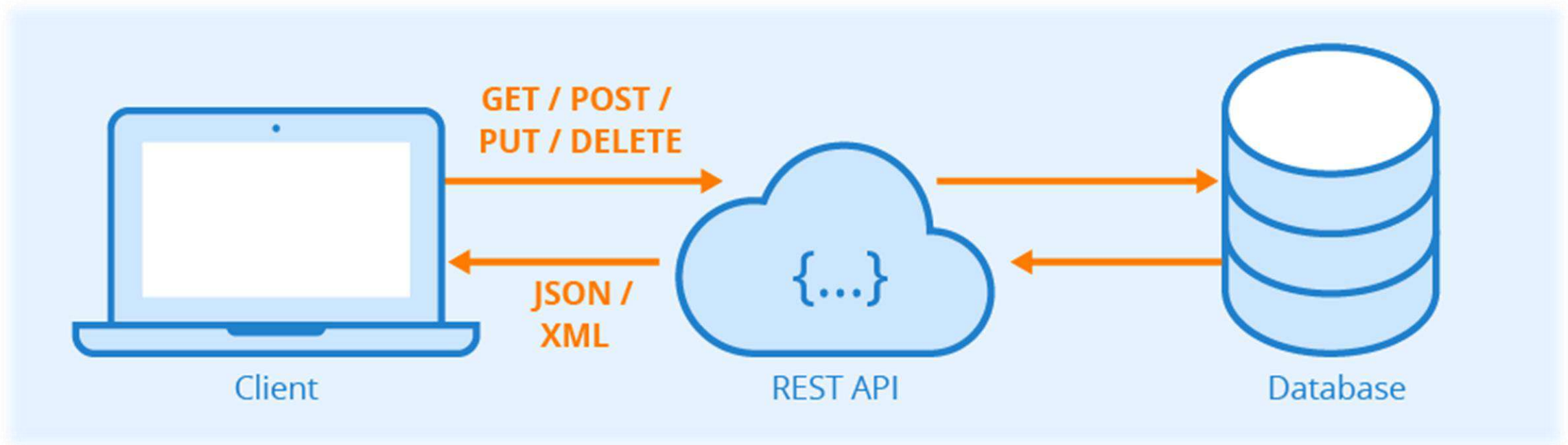
- Import data Excel using REST API



- Upload/Download File REST API



REST API COMMUNICATION MODEL



❑ Operations

- ❖ GET: (url) => response
- ❖ POST: (url, data) => response
- ❖ PUT: (url, data) => response
- ❖ DELETE: (url) => response (null)

❑ Transfer Data

- ❖ Dữ liệu trao đổi giữa Client và REST API là **JSON/XML**



RESTCONTROLLER

STUDENTRESTCONTROLLER

@RestController

public class StudentRestController {

Map<String, Student> map = new HashMap<>();

@GetMapping("/api/students")

public Collection<Student> get(){...}

@GetMapping("/api/students/{id}")

public Student get(@PathVariable("id") String id){...}

@PostMapping("/api/students")

public Student post(@RequestBody Student data){...}

@PutMapping("/api/students/{id}")

public Student put(@PathVariable("id") String id, @RequestBody Student data){...}

@DeleteMapping("/api/students/{id}")

public void delete(@PathVariable("id") String id){...}

}

GET(url): Collection<Student>

GET(url): Student

POST(url, data): Student

PUT(url, data): Student

DELETE(url)

id: String

name: String

email: String

STUDENTRESTCONTROLLER IMPLEMENTATION

```
public Collection<Student> get(){
    return map.values();
}
```

GET(url): Collection<Student>

```
public Student get(@PathVariable("id") String id){
    return map.get(id);
}
```

GET(url): Student

```
public Student post(@RequestBody Student data){
    map.put(data.getId(), data);
    return data;
}
```

POST(url, data): Student

```
public Student put(@PathVariable("id") String id, @RequestBody Student data){
    if(map.containsKey(data.getId())) {
        map.put(data.getId(), data);
        return data;
    }
    return null;
}
```

PUT(url, data): Student

```
public void delete(@PathVariable("id") String id){
    map.remove(id);
}
```

DELETE(url)

```
@Data
public class Student {
    String id;
    String name;
    Double marks;
}
```


- ❑ Mặc định chỉ có các Rest Consumer cùng domain được cho phép consume các REST API (khác domain sẽ không được phép).
- ❑ **@CrossOrigin()** được sử dụng để khai báo cho phép các nguồn địa chỉ Rest Consumer đáng tin cậy.

```
@CrossOrigin(origins = {"http://localhost:8080", "http://127.0.0.1:8080"})  
@RestController  
public class StudentRestController {...}
```

- ❑ Cấu hình này cho phép các trang web đặt tại các host: localhost:8080 và 127.0.0.1:8080 được phép truy cập.
- ❑ Sử dụng **origins="*"** để cho phép mọi host



DEMO


```
@GetMapping("/api/students")  
public Collection<Student> get(){  
    return map.values();  
}
```

```
@GetMapping("/api/students")  
public ResponseEntity<Collection<Student>> get(){  
    return ResponseEntity.ok(map.values());  
}
```

- ❑ Các cách viết mã (1) và (2) là tương đương. Tuy nhiên cách viết (2) cho chúng ta **mở rộng mã để điều khiển các lỗi** một cách chính xác (xem slide sau).

```
@GetMapping("/api/students")  
public Collection<Student> get(){  
    return map.values();  
}
```

```
@GetMapping("/api/students")  
public ResponseEntity<Collection<Student>> get(){  
    if(map.values().isEmpty()) {  
        return ResponseEntity.noContent().build();  
    }  
    return ResponseEntity.ok(map.values());  
}
```

Rest Consumer sẽ nhận được trạng thái với mã 204. Từ đó có thể đưa ra các xử lý, thông báo phù hợp

- ❑ Void là class đại diện cho void, được khai báo cho các phương thức không trả về kết quả

```
@DeleteMapping("/api/students/{id}")  
public ResponseEntity<Void> delete(@PathVariable("id") String id){  
    map.remove(id);  
    return ResponseEntity.ok().build();  
}
```

- ❑ ResponseEntity.**badRequest**().build()
 - ❖ **400** Bad Request: Địa chỉ tồi
- ❑ ResponseEntity.**noContent**().build()
 - ❖ **204** No Content: Không có nội dung
- ❑ ResponseEntity.**notFound**().build()
 - ❖ **404** Not Found: Không tìm thấy
- ❑ ResponseEntity.**ok**(body)
 - ❖ **200** OK: Thành công
- ❑ ResponseEntity.**status**(**HttpStatus**).build()
 - ❖ Status Code: Chứa trạng thái tùy chọn

```
@CrossOrigin(origins = "*")
```

```
@RestController
```

```
public class StudentApiController {
```

```
    public ResponseEntity<Collection<Student>> get(){...}
```

```
    public ResponseEntity<Student> get(@PathVariable("id") String id){...}
```

```
    public ResponseEntity<Student> post(@RequestBody Student data){...}
```

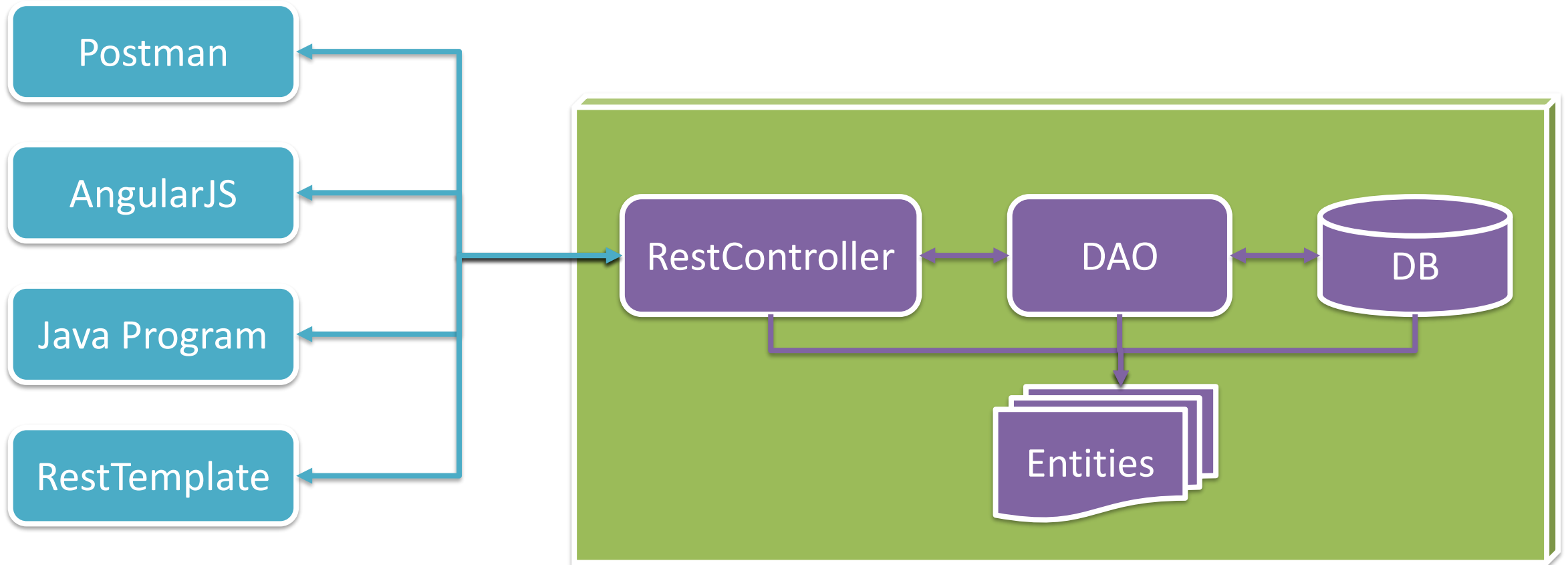
```
    public ResponseEntity<Student> put(@PathVariable("id") String id,  
                                       @RequestBody Student data){...}
```

```
    public ResponseEntity<Void> delete(@PathVariable("id") String id){...}
```

```
}
```



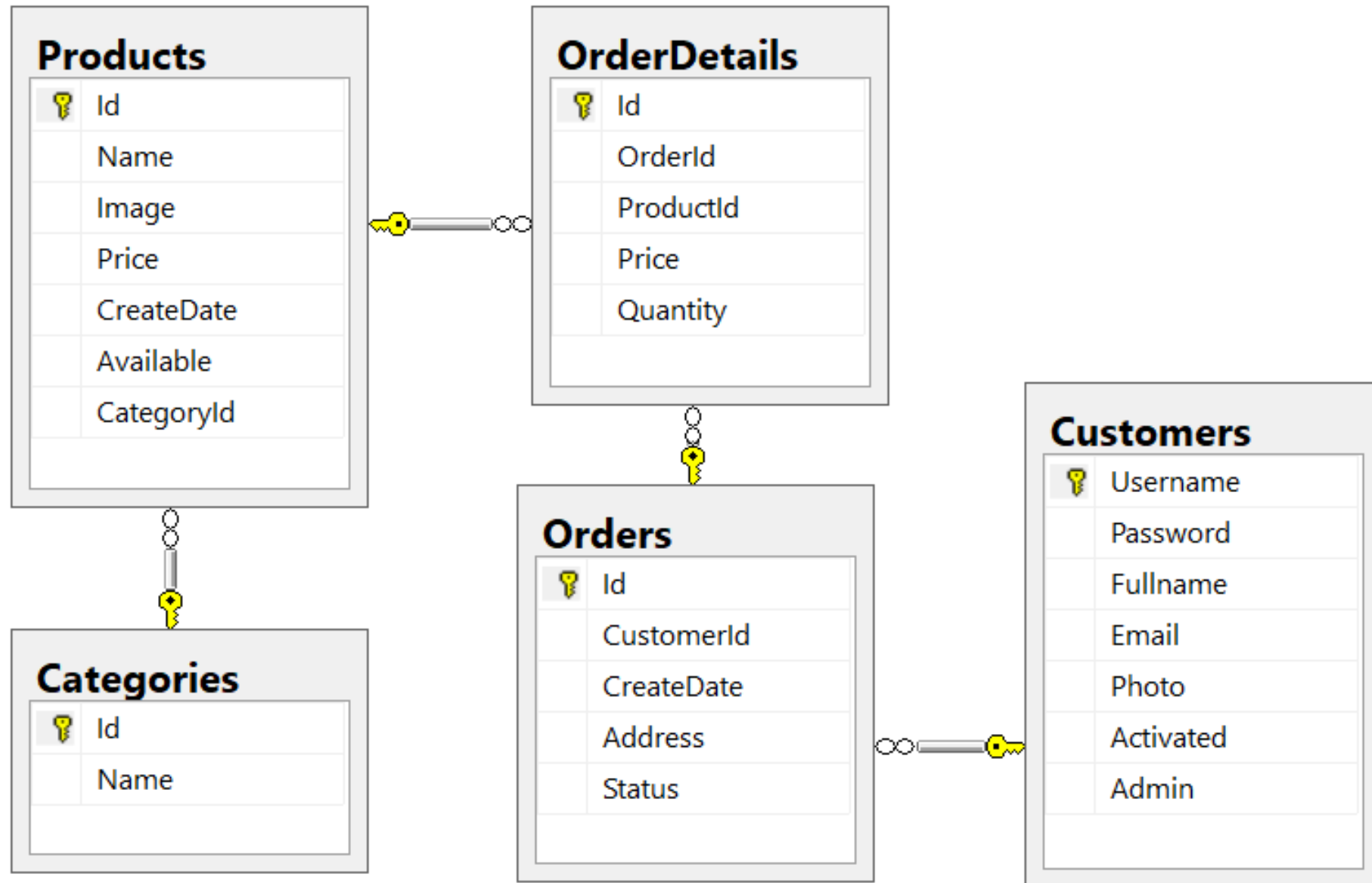

REST API WITH JPARepository



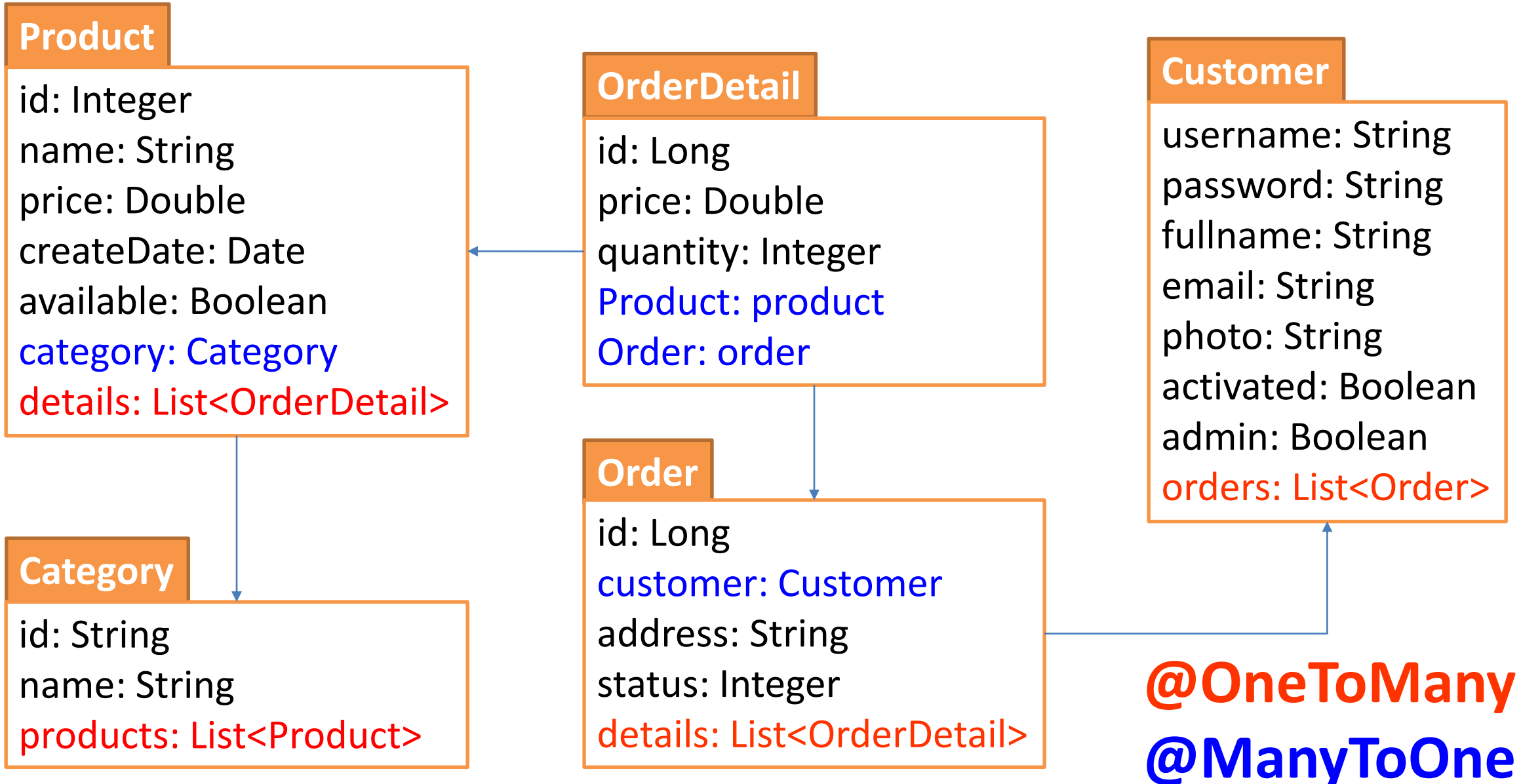
REST Consumers

REST API

DATABASE RELATIONSHIP DIAGRAM




ERD – ENTITY RELATIONAL DIAGRAM



ASSOCIATION ENTITY CLASS MAPPING

Categories

	Column Name	Condensed Type	Nullable	Identity
	Id	char(4)	No	<input type="checkbox"/>
	Name	nvarchar(50)	No	<input type="checkbox"/>
				<input type="checkbox"/>

@Data

@Entity

@Table(name = "Categories")

public class Category {

@Id

String id;

String name;

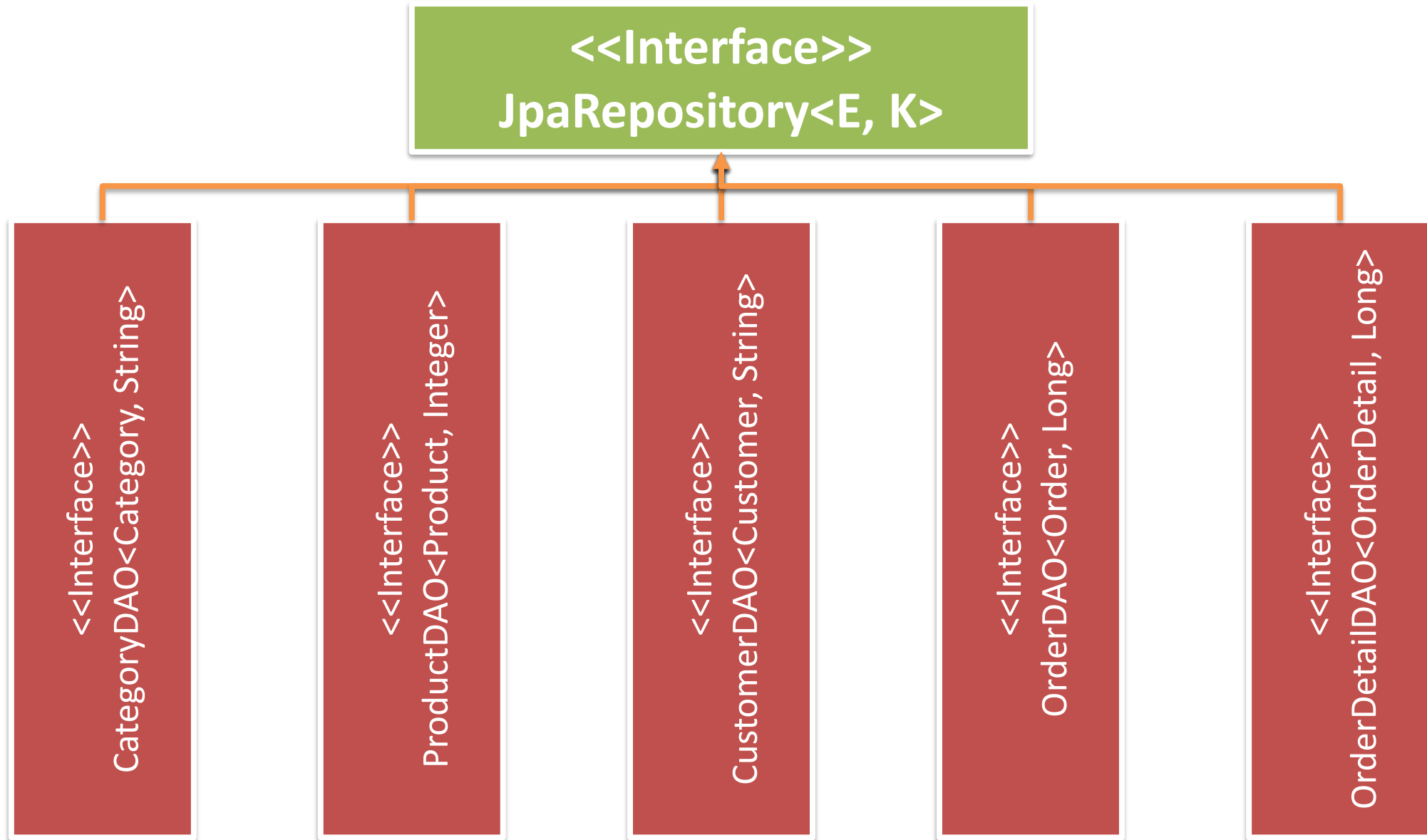
@JsonIgnore

@OneToMany(mappedBy = "category")

List<Product> products;

}

*Thêm **@JsonIgnore** vào các kết hợp **@OneToMany** để loại bỏ thuộc tính này khi chuyển đổi từ Java Object sang JSON.*



<<Interface>>

CrudRepository<T, ID>

<S extends T> S **save**(S entity)
 void **delete**(T entity)
 Optional<T> **findById**(ID id)
 T **getOne**(ID id)
 Iterable<T> **findAll**()
 Long **count**()
 boolean **exists**(ID id)

<<Interface>>

PagingAndSortingRepository<T, ID>

Iterable<T> **findAll**(Sort sort)
 Page<T> **findAll**(Pageable pageable)

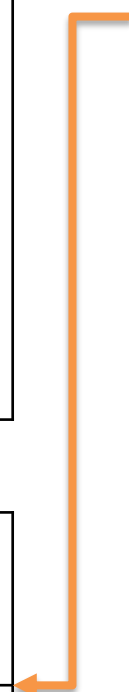
<<Interface>>

JpaRepository<T, ID>

List<T> **findAll**()
 List<T> **findAll**(Sort sort)
 List<T> **save**(Iterable<? extends T> entities)
 void **flush**()
 T **saveAndFlush**(T entity)
 void **deleteInBatch**(Iterable<T> entities)

<<Interface>>

CategoryDAO<Category, String>





DEMO

- ❑ **GET:** /api/categories
 - ❖ *() => List<Category>*
- ❑ **GET:** /api/categories/{id}
 - ❖ *(id) => Category*
- ❑ **POST:** /api/categories & **Category**
 - ❖ *(Category) => Category*
- ❑ **PUT:** /api/categories/{id} & **Category**
 - ❖ *(id, Category) => Category*
- ❑ **DELETE:** /api/categories/{id}
 - ❖ *(id) => Void*

```
@RestController
@CrossOrigin(origins = "*")
@RequestMapping("/api/categories")
public class CategoryController {
    @GetMapping
    public ResponseEntity<List<Category>> findAll() {...}
    @GetMapping("/{id}")
    public ResponseEntity<Category> findById(@PathVariable("id") String id) {...}
    @PostMapping
    public ResponseEntity<Category> post(@RequestBody Category category) {...}
    @PutMapping("/{id}")
    public ResponseEntity<Category> put(@PathVariable("id") String id,
                                       @RequestBody Category category) {...}
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> delete(@PathVariable("id") String id) {...}
}
```


CATEGORYRESTCONTROLLER IMPLEMENTATION

@Autowired

CategoryDAO cdao;

findAll()

return ResponseEntity.ok(cdao.findAll());

findById(String id)

```
Optional<Category> optional = cdao.findById(id);  
if(!optional.isPresent()) {  
    return ResponseEntity.notFound().build();  
}  
return ResponseEntity.ok(optional.get());
```

CATEGORYRESTCONTROLLER IMPLEMENTATION

```
@Autowired
CategoryDAO cdao;
```

post(Category category)

```
if(cdao.existsById(category.getId())) {
    return ResponseEntity.badRequest().build();
}
cdao.save(category);
return ResponseEntity.ok(category);
```

**put(String id,
Category category)**

```
if(!cdao.existsById(id)) {
    return ResponseEntity.notFound().build();
}
cdao.save(category);
return ResponseEntity.ok(category);
```

delete(String id)

```
if(!cdao.existsById(id)) {
    return ResponseEntity.notFound().build();
}
cdao.deleteById(id);
return ResponseEntity.ok().build();
```



DEMO



IMPORT EXCEL USING REST API

Test.xlsx - Excel

File Home Insert Page Layout Formulas Data Review View Tell me what you want to do... Sign in Share

B2 X ✓ fx Nguyễn Thị Phương Trang

	A	B	C	D
1	MSSV	FULLNAME	EMAIL	PHONE
2	PS10956	Nguyễn Thị Phương Trang	trangntpps10956@fpt.edu.vn	0944499177
3	PS10835	Ngọc Châu	chaulnnps10835@fpt.edu.vn	0903040074
4	PS10970	Lê Vương Minh Suốt	suotlvmps10970@fpt.edu.vn	0914696915
5	PS10930	Nguyễn Vũ Đạt	datnvps10930@fpt.edu.vn	0969048779
6	PS10858	Nguyễn Sơn	sonntns10858@fnt.edu.vn	0976272392

Row Number

Cell Value

Sheet Name

DSSV Subjects

Ready

160%

❑ Mục tiêu:

- ❖ Đọc dữ liệu từ file excel, mỗi row chuyển thành một đối tượng JSON và gửi lên server để lưu vào CSDL

❑ API và thư viện cần thiết

- ❖ **FileReader** API: đọc file từ trường file
- ❖ Thư viện **ExcelJS**: Xử lý dữ liệu excel của file

❑ Các thành phần trong **ExcelJS**

- ❖ **Workbook**: gồm tất cả các sheet trong file excel
- ❖ **Worksheet**: gồm nhiều hàng (row)
- ❖ **Row**: gồm nhiều ô (cell) theo hướng ngang
- ❖ **Column**: gồm nhiều ô (cell) theo hướng đứng
- ❖ **Cell**: chứa dữ liệu

// ĐỌC DỮ LIỆU TỪ FILE

```
var reader = new FileReader();
```

```
reader.onloadend = async () => {
```

// DỮ LIỆU ĐỌC ĐƯỢC CHỨA TRONG READER.RESULT

```
var data = reader.result;
```

```
};
```

```
reader.readAsArrayBuffer(file);
```

<input type="file">


- ❑ FileReader được JS cung cấp sẵn, cho phép đọc dữ liệu file từ trường file để có thể xử lý ngay trên trình duyệt

- ❑ ExcelJS là thư viện JS cho phép xử lý dữ liệu từ file excel
- ❑ <https://cdnjs.cloudflare.com/ajax/libs/exceljs/4.2.0/exceljs.min.js>

★ **// TẠO WORKBOOK CHỨA DỮ LIỆU EXCEL**
var workbook = **new** ExcelJS.Workbook();

★ **// ĐỌC DỮ LIỆU FILE VÀO WORKBOOK**
await workbook.xlsx.load(**reader.result**);

★ **// LẤY WORKSHEET THÔNG QUA TÊN SHEET**
const worksheet = workbook.getWorksheet('DSSV');

★ **// DUYỆT CÁC HÀNG CỦA WORKSHEET**
worksheet.eachRow((row, index) => {
 
});

Xử lý dữ liệu mỗi row

READ DATA FROM CELLS OF A ROW

```
worksheet.eachRow((row, index) => {  
  if(index > 1){  
    let student = {  
      id: row.getCell(1).value,  
      name: row.getCell(2).value,  
      email: row.getCell(3).value,  
      phone: row.getCell(4).value  
    }  
  }  
})
```

Loại bỏ hàng đầu tiên

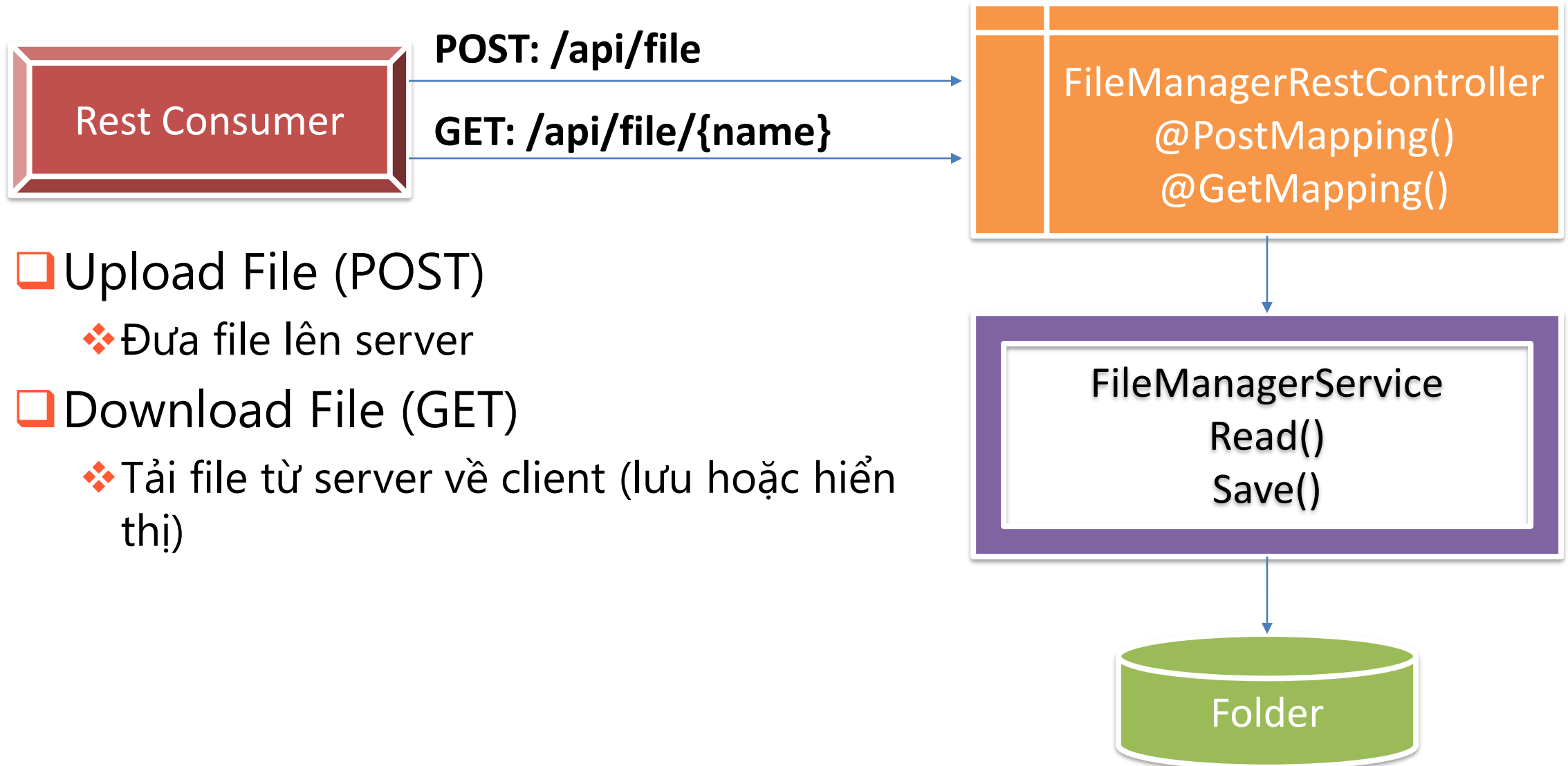
	A	B	C	D
1	MSSV	FULLNAME	EMAIL	PHONE
2	PS10956	Nguyễn Thị Phương Trang	trangntpps10956@fpt.edu.vn	0944499177
3	PS10835	Lê Nguyễn Ngọc Châu	chaulnnps10835@fpt.edu.vn	0903040074



DEMO



UPLOAD AND DOWNLOAD FILE




```
@RestController
public class FileManagerRestCont
    @Autowired
    FileManagerService fman; }
```

```
@Service
public class FileManagerService {
    @Autowired
    ServletContext app;
    public byte[] read(String name) {...}
    public File save(MultipartFile file) {...}
}
```

```
@PostMapping("/api/file")
public List<String> upload(@RequestParam("file") MultipartFile file) {...}
```

```
@GetMapping("/api/file/{name}")
public byte[] download(@PathVariable("name") String name) {...}
```

```
}
```

```
try {  
    File file = new File(app.getRealPath("/files/" + name));  
    return Files.readAllBytes(file.toPath());  
} catch (Exception e) {  
    throw new RuntimeException(e);  
}
```

*ServletContext.**getRealPath()**
để chuyển đổi đường dẫn ảo
(đường dẫn tính từ gốc của
website) sang đường dẫn thực*

```
File dir = new File(app.getRealPath("/files/"));  
if(!dir.exists()) { // tạo mới nếu chưa tồn tại  
    dir.mkdirs();  
}  
try {  
    String filename = file.getOriginalFilename();  
    File newFile = this.getFile(dir, filename);  
    file.transferTo(newFile);  
    return newFile;  
} catch (Exception e) {  
    throw new RuntimeException(e);  
}
```

```
@PostMapping("/api/file")
public List<String> upload(@RequestPart("file") MultipartFile file) {
    File uploadFile = fman.save(file);
    return uploadFile.getName();
}

@GetMapping("/api/file/{name}")
public byte[] download(@PathVariable("name") String name,
    HttpServletResponse response){
    response.setHeader("Content-Disposition", "attachment;filename=" + name);
    return fman.read(name);
}
```

❑ HttpHeaders: **Content-Disposition**

❖ Thiết lập tên file lúc tải về

AngularJS Upload

```
var form = new FormData();  
form.append("files", files[0]);  
  
$http.post(url, form, {  
    transformRequest: angular.identity,  
    headers: {'Content-Type': undefined}  
}).then(resp => {  
    // upload thành công  
}).catch(error => {  
    // upload thất bại  
});
```

HTML Download

```
<a href="/api/file/abc.jpg">Tải về</a>
```

HTML Hiển thị ảnh

```

```



DEMO

- ✓ Spring Boot REST API
 - ✓ @RestController, @GetMapping...
- ✓ REST API with JpaRepository
 - ✓ @JsonIgnore với @OneToMany
- ✓ Import Excel File using REST API
 - ✓ FileReader, ExcelJS
- ✓ Upload File using REST API
 - ✓ FormData
- ✓ Download File using REST API
 - ✓ Header: **Content-Disposition**





Cảm ơn