LAB 3
DEVELOP JAVA SPRING BOOT WEB APP (P3)

❖ CONTENT

- Setup relationship between entities (tables)
- Implement extra features: Filter, Search, Sort

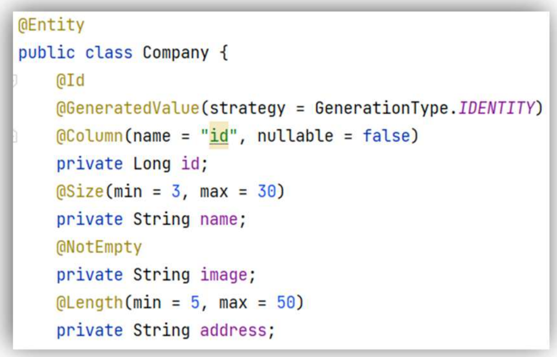
❖ INSTRUCTION

1. Open the previous Spring Boot project to continue coding

   File ⇨ **Open** ⇨ Select the project location

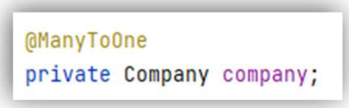
2. Create new entity **Company** then add an attribute to entity **Employee** as foreign key to represent for entity relationship

   <u>Note:</u> *Employee – Company :* ManyToOne



```java
@Entity
public class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;
    @Size(min = 3, max = 30)
    private String name;
    @NotEmpty
    private String image;
    @Length(min = 5, max = 50)
    private String address;
```

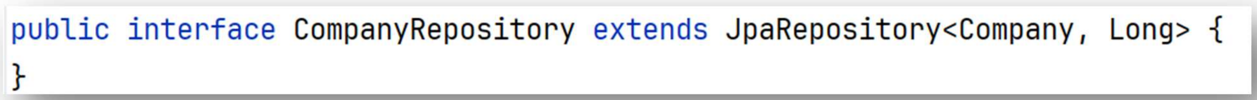
*Figure 1 - **Company.java***



```java
@ManyToOne
private Company company;
```

*Figure 2 - **Employee.java***

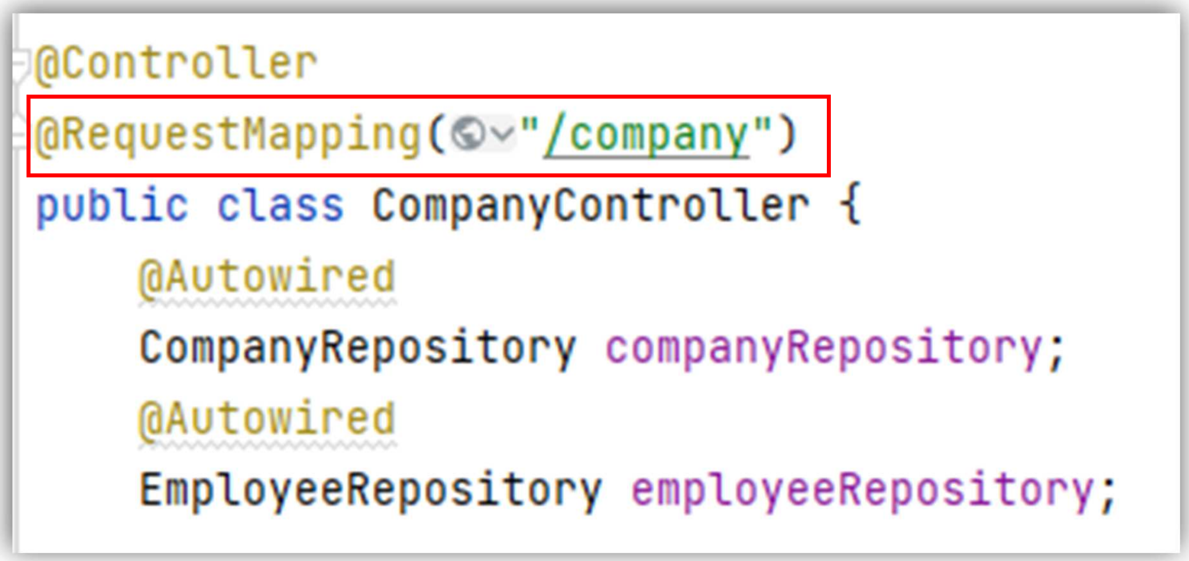
3. Create the *Repository* for **Company**



```
public interface CompanyRepository extends JpaRepository<Company, Long> {
}
```

*Figure 3 - **CompanyRepository.java***

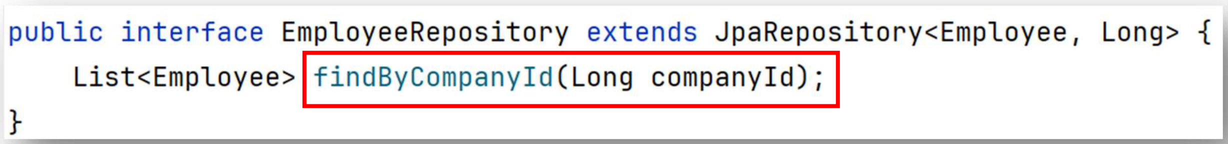
4. Create the *Controller* for **Company**



```
@Controller
@RequestMapping("/company")
public class CompanyController {
    @Autowired
    CompanyRepository companyRepository;
    @Autowired
    EmployeeRepository employeeRepository;
```

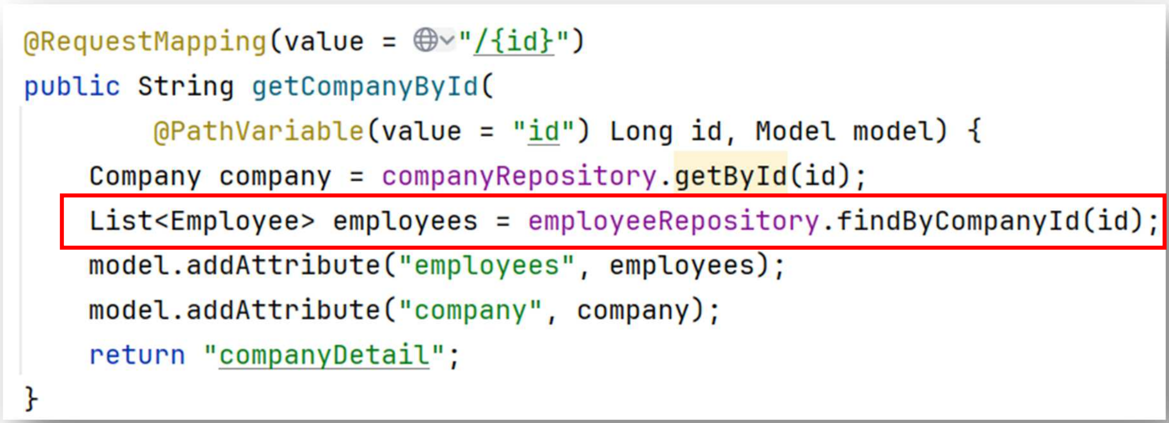
*Figure 4 - **CompanyController.java** (1)*

5. Implement *Filter* feature for **Company**



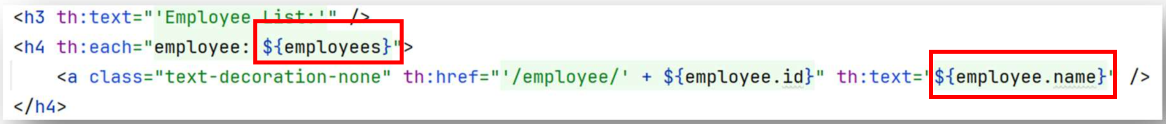
```
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
    List<Employee> findByCompanyId(Long companyId);
}
```

*Figure 5 - **EmployeeRepositor.java***



```java
@RequestMapping(value = "/{id}")
public String getCompanyById(
        @PathVariable(value = "id") Long id, Model model) {
    Company company = companyRepository.getById(id);
    List<Employee> employees = employeeRepository.findByCompanyId(id);
    model.addAttribute("employees", employees);
    model.addAttribute("company", company);
    return "companyDetail";
}
```

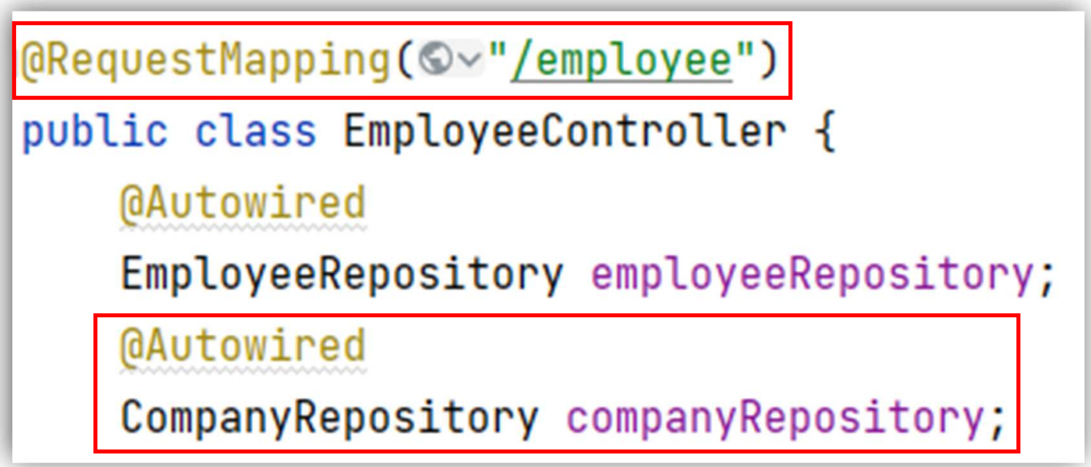
*Figure 6 - **CompanyController.java** (2)*



```html
<h3 th:text="'Employee List:'" />
<h4 th:each="employee: ${employees}">
    <a class="text-decoration-none" th:href="'/employee/' + ${employee.id}" th:text="${employee.name}" />
</h4>
```

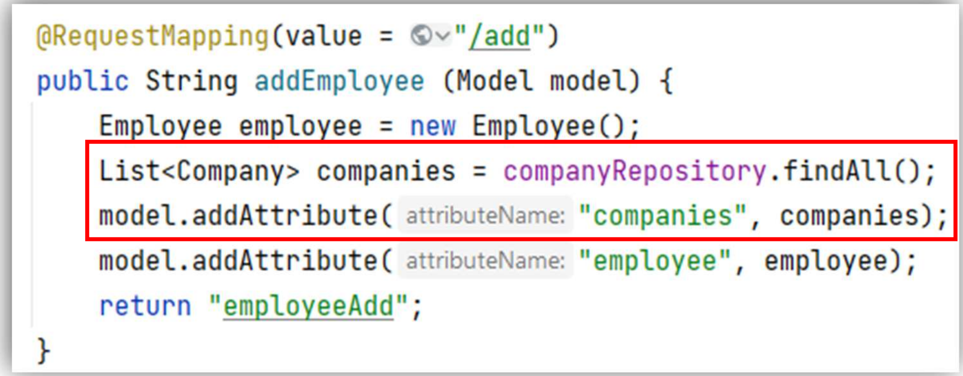
*Figure 7 - **CompanyDetail.html***

6. Update *Controller* for **Employee** (update links & add *CompanyRepository*)



```java
@RequestMapping("/employee")
public class EmployeeController {
    @Autowired
    EmployeeRepository employeeRepository;
    @Autowired
    CompanyRepository companyRepository;
```

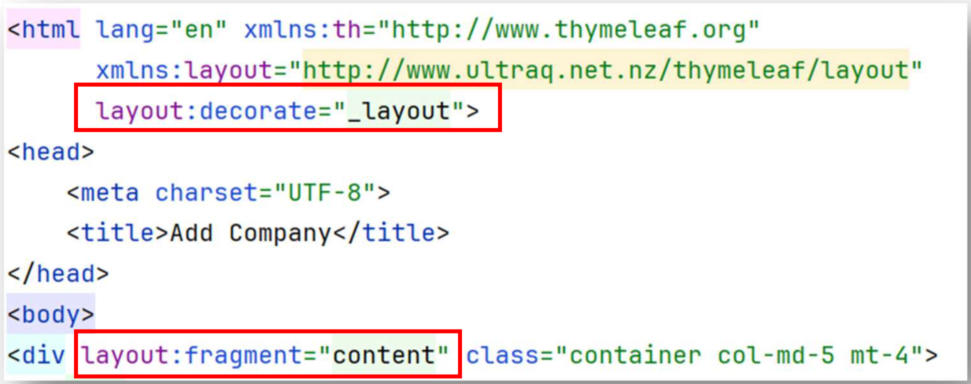
*Figure 8 - **EmployeeController.java** (1)*



```java
@RequestMapping(value = "/add")
public String addEmployee (Model model) {
    Employee employee = new Employee();
    List<Company> companies = companyRepository.findAll();
    model.addAttribute( attributeName: "companies", companies);
    model.addAttribute( attributeName: "employee", employee);
    return "employeeAdd";
}
```

*Figure 9 - **EmployeeController.java** (2)*

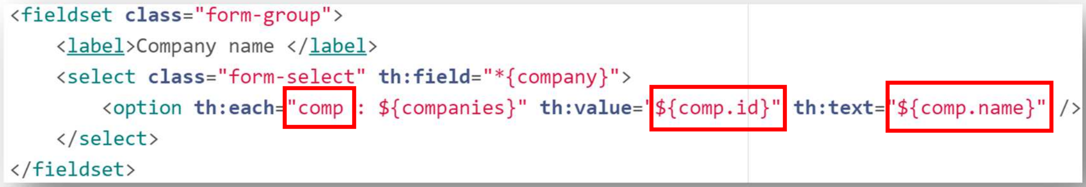
7. Create *Views* for **Company** which extends web layout



```html
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="_layout">
<head>
    <meta charset="UTF-8">
    <title>Add Company</title>
</head>
<body>
<div layout:fragment="content" class="container col-md-5 mt-4">
```

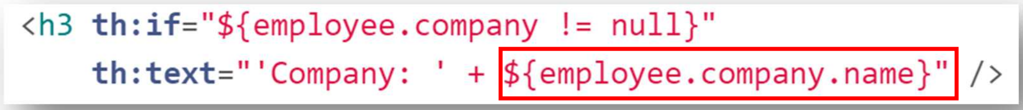
*Figure 10 - **companyAdd.html***

8. Update *Views* for **Employee**



```html
<fieldset class="form-group">
    <label>Company name </label>
    <select class="form-select" th:field="*{company}">
        <option th:each="comp : ${companies}" th:value="${comp.id}" th:text="${comp.name}" />
    </select>
</fieldset>
```

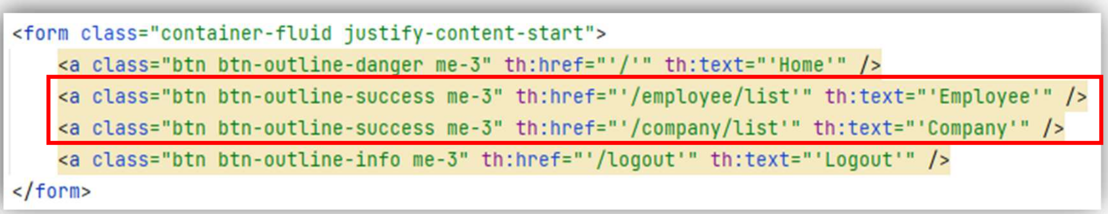
*Figure 11 - **employeeAdd.html***



```
<h3 th:if="${employee.company != null}"
    th:text="'Company: ' + ${employee.company.name}" />
```

*Figure 12 - **employeeDetail.html***

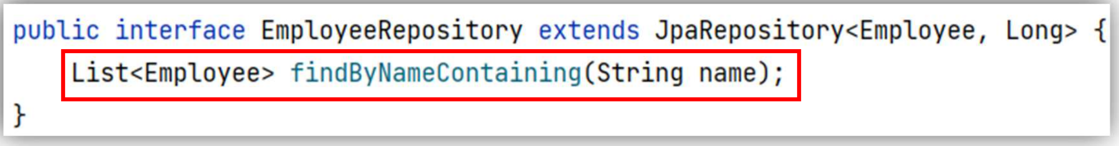
9. Update navigation path in web layout



```
<form class="container-fluid justify-content-start">
    <a class="btn btn-outline-danger me-3" th:href="'/'" th:text="'Home'" />
    <a class="btn btn-outline-success me-3" th:href="'/employee/list'" th:text="'Employee'" />
    <a class="btn btn-outline-success me-3" th:href="'/company/list'" th:text="'Company'" />
    <a class="btn btn-outline-info me-3" th:href="'/logout'" th:text="'Logout'" />
</form>
```

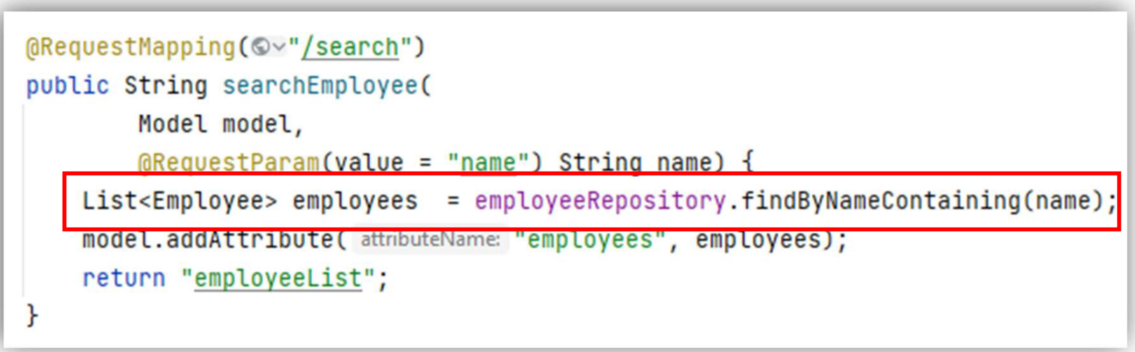
*Figure 13 - **_layout.html***

10.Implement *Search* feature for **Employee**



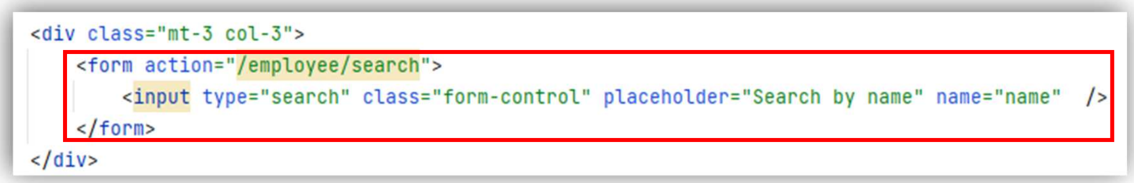
```
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
    List<Employee> findByNameContaining(String name);
}
```

*Figure 14 – **EmployeeRepository.java***



```
@RequestMapping("/search")
public String searchEmployee(
        Model model,
        @RequestParam(value = "name") String name) {
    List<Employee> employees  = employeeRepository.findByNameContaining(name);
    model.addAttribute( attributeName: "employees", employees);
    return "employeeList";
}
```

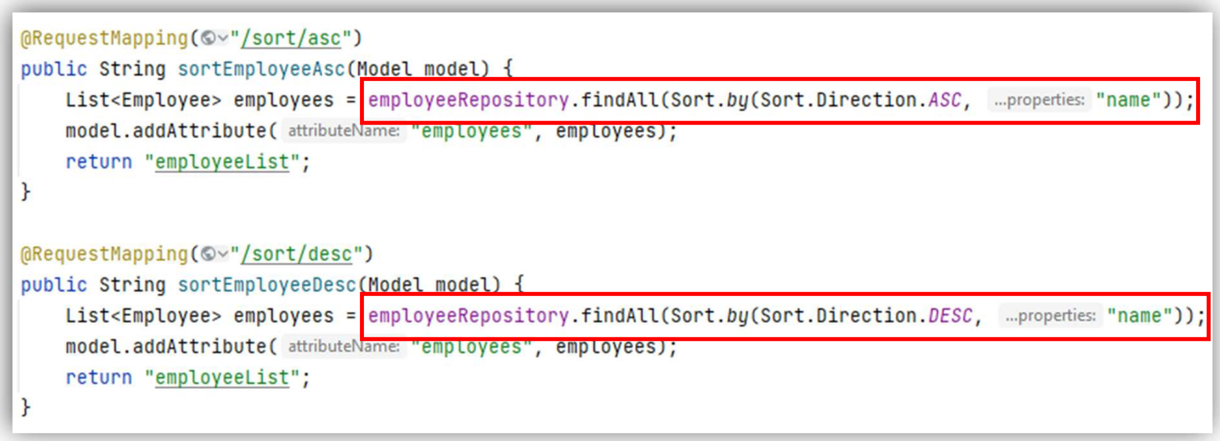
*Figure 15 – **EmployeeController.java** (3)*



```html
<div class="mt-3 col-3">
    <form action="/employee/search">
        <input type="search" class="form-control" placeholder="Search by name" name="name"  />
    </form>
</div>
```

*Figure 16 - **employeeList.html***

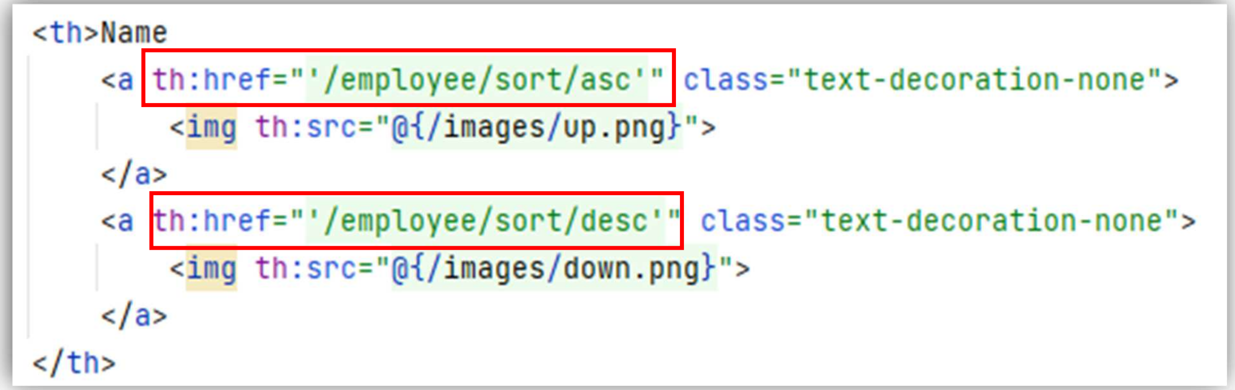
11.Implement *Sort* feature for **Employee**



```java
@RequestMapping("/sort/asc")
public String sortEmployeeAsc(Model model) {
    List<Employee> employees = employeeRepository.findAll(Sort.by(Sort.Direction.ASC,  ...properties: "name"));
    model.addAttribute( attributeName: "employees", employees);
    return "employeeList";
}

@RequestMapping("/sort/desc")
public String sortEmployeeDesc(Model model) {
    List<Employee> employees = employeeRepository.findAll(Sort.by(Sort.Direction.DESC,  ...properties: "name"));
    model.addAttribute( attributeName: "employees", employees);
    return "employeeList";
}
```

*Figure 17 - **EmployeeController.java** (4)*



```html
<th>Name
    <a th:href="'/employee/sort/asc'" class="text-decoration-none">
        <img th:src="@{/images/up.png}">
    </a>
    <a th:href="'/employee/sort/desc'" class="text-decoration-none">
        <img th:src="@{/images/down.png}">
    </a>
</th>
```

*Figure 18 - **employeeList.java** (2)*

12.Test the web application

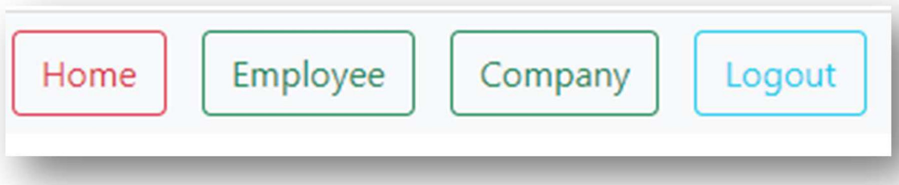
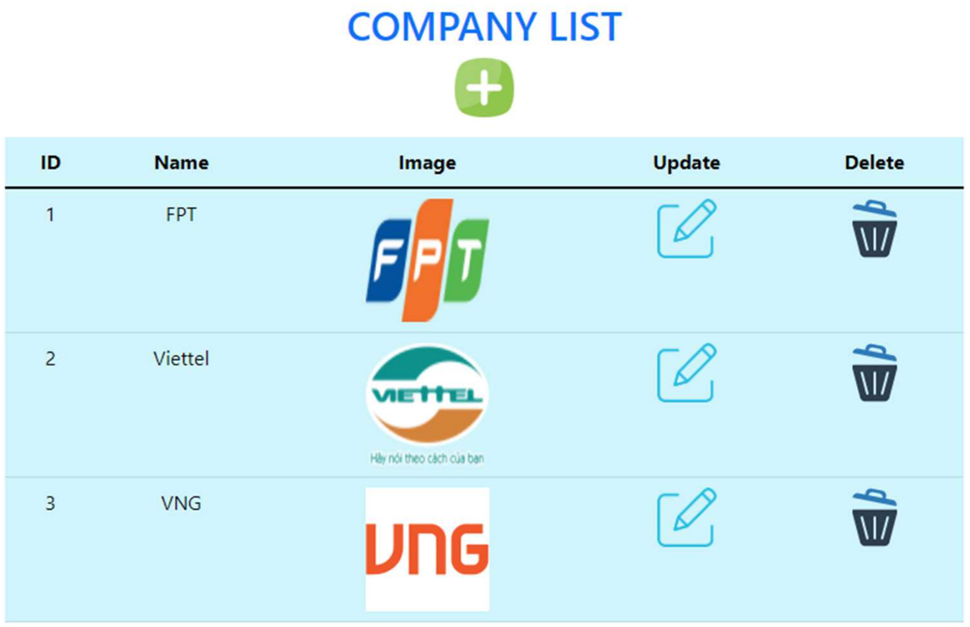


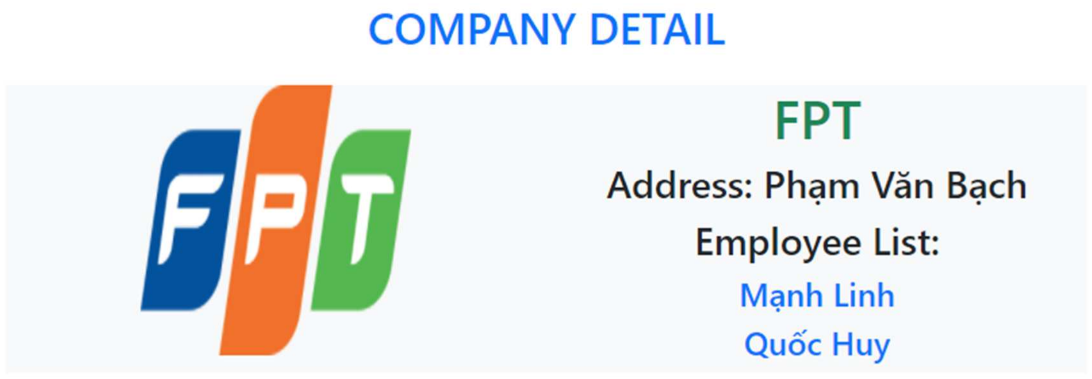
Figure 19 - Navigation bar (updated)



COMPANY LIST

ID	Name	Image	Update	Delete
1	FPT			
2	Viettel			
3	VNG			

Figure 20 - Company List



COMPANY DETAIL

FPT

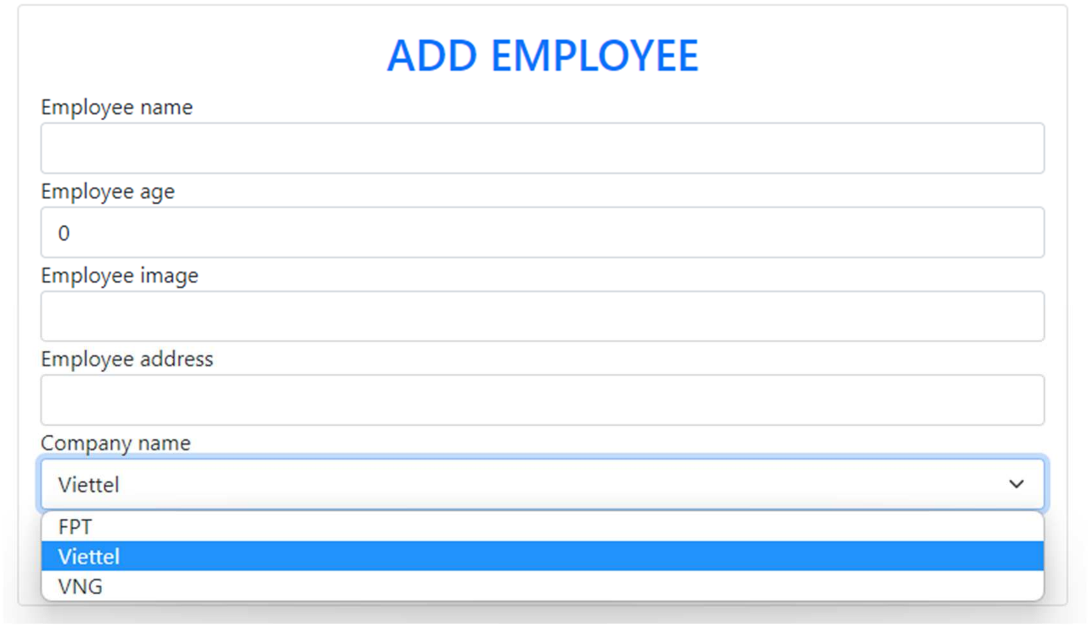
Address: Phạm Văn Bạch

Employee List:

Mạnh Linh

Quốc Huy

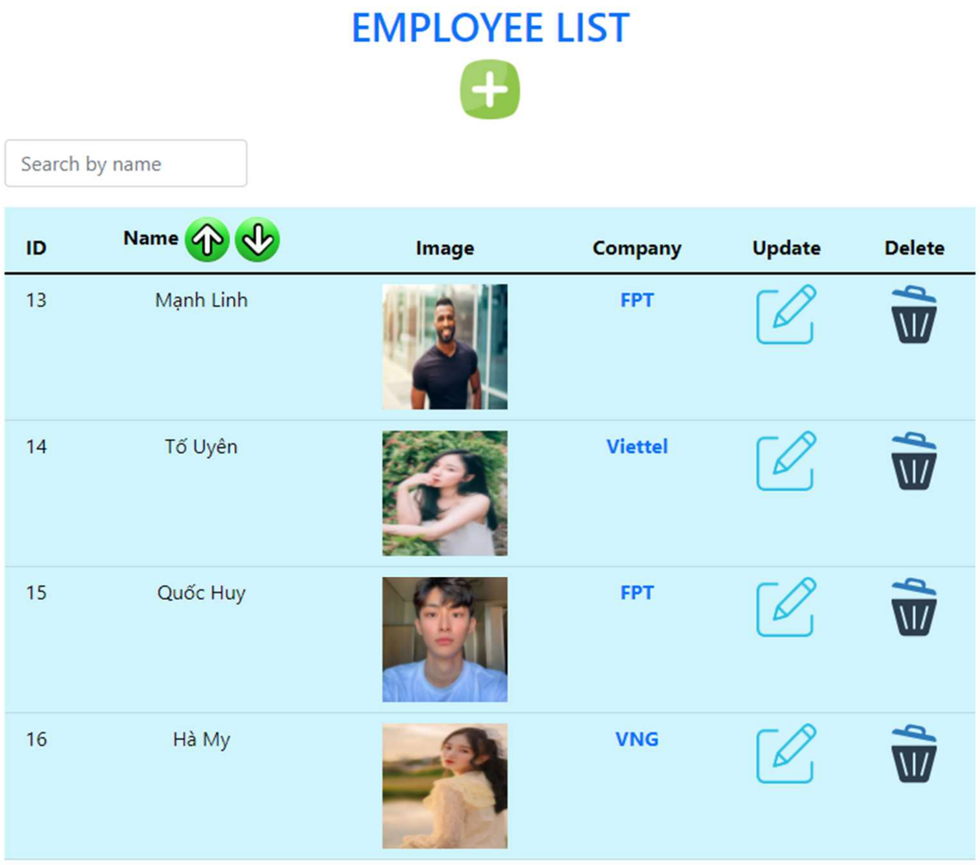
Figure 21 - Company Detail



ADD EMPLOYEE

Employee name	
Employee age	0
Employee image	
Employee address	
Company name	Viettel

Figure 22 - Add Employee (updated)



EMPLOYEE LIST

ID	Name	Image	Company	Update	Delete
13	Mạnh Linh		FPT		
14	Tố Uyên		Viettel		
15	Quốc Huy		FPT		
16	Hà My		VNG		

Figure 23 - Employee List (updated)

❖ TO-DO

- Complete the remained codes to run web application
- Implement search & sort features for *Company*
- Add new entity *Job* (*Employee – Job* : *ManyToMany*) then do similar with entity *Company*
- *Extra:* Implement the pagination feature (such as display 5 records/page)