**ĐẠI HỌC QUỐC GIA TP.HCM**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**



**BÁO CÁO**

**THỰC TẬP NGOÀI TRƯỜNG**

**HỌC KỲ: 3 - NĂM HỌC: 2022-2023**

* **NGÀNH:** Kỹ thuật máy tính
* **CHƯƠNG TRÌNH ĐÀO TẠO:** Chính quy
* **ĐƠN VỊ/ DOANH NGHIỆP NHẬN THỰC TẬP:**

Công ty TNHH Phần mềm FPT Hồ Chí Minh.

* **CÁN BỘ HƯỚNG DẪN CHUYÊN MÔN TRỰC TIẾP CỦA ĐƠN VỊ/ DOANH NGHIỆP:**

Nguyễn Tấn Thành (Mentor).

Nguyễn Đoàn Diệp Ngân (Admin).

* **CÁN BỘ HƯỚNG DẪN/ GIÁM SÁT/ CHẤM ĐIỂM BÁO CÁO CỦA KHOA (CBHD/CBGS/CĐBC):**

**……………………………………………………………...**

* **SINH VIÊN THỰC HIỆN (SVTH):**

**HỌ VÀ TÊN:** Nguyễn Phước Thịnh – **MSSV:** 1915317

TP. HỒ CHÍ MINH, NGÀY 10 THÁNG 8 NĂM 2023

**MỤC LỤC**

[CHƯƠNG 1: GIỚI THIỆU VỀ CÔNG TY FPT-SOFTWARE 3](#_Toc142519940)

[CHƯƠNG 2: TỔNG QUAN VỀ SPRING TRONG JAVA 4](#_Toc142519941)

[1. Spring MVC 4](#_Toc142519942)

[2. Spring Boot 4](#_Toc142519943)

[3. Spring JPA 4](#_Toc142519944)

[4. Spring Security 5](#_Toc142519945)

[5. JWT 5](#_Toc142519946)

[6. Rest Controller 5](#_Toc142519947)

[7. Rest Template 6](#_Toc142519948)

[CHƯƠNG 3: YÊU CẦU CỦA DỰ ÁN 7](#_Toc142519949)

[CHƯƠNG 4: QUÁ TRÌNH LÀM VIỆC CỦA CÁ NHÂN 8](#_Toc142519950)

[1. Học cách sử dụng Git Lab 8](#_Toc142519951)

[2. Vẽ ERD 8](#_Toc142519952)

[3. Xây dựng các thực thể (Entity) 8](#_Toc142519953)

[4. Xây dựng Repository 10](#_Toc142519954)

[5. Xây dựng Service và ServiceImpl 10](#_Toc142519955)

[6. Xây dựng Controller 12](#_Toc142519956)

[7. Merge code, Fix bug và hoàn thiện project 14](#_Toc142519957)

[CHƯƠNG 5: TỔNG KẾT 14](#_Toc142519958)

# CHƯƠNG 1: GIỚI THIỆU VỀ CÔNG TY FPT-SOFTWARE

FPT Software là tên gọi khác của công ty TNHH Phần Mềm FPT với nhiệm vụ chính là gia công phần mềm tại Việt Nam và nước ngoài.

Thành lập từ năm 1988 đến nay với 3 Trụ sở chính FPT Software đặt tại Việt Nam và một số nước trên toàn thế giới như Hoa Kỳ, Nhật Bản, Malaysia, Đức, Úc, Singapore, Malaysia, Thái Lan và Philipines hiện FPT Software đáp ứng nhu cầu gia công phần mềm lớn cho hơn 150 công ty hàng đầu tại 20 quốc gia lớn nhất trên thế giới hiện nay với các hợp đồng lớn có khi đạt cả 1 triệu USD dành cho một số đối tác lớn như Hitachi, NEOPOST, Petronas, Deutsche Bank, và Unilever với hơn 4000 nhân viên trên toàn thế giới.

FPT Software theo đuổi mục tiêu gia công phần mềm để đáp ứng cho nhu cầu phát triển CNTT của các hãng phần mềm trong nước, các công ty lớn trong nước và tham vọng hơn là xuất khẩu phần mềm trên toàn thế giới cho các công ty nước ngoài biết đến tập đoàn FPT, mục đích chính là vươn đến tầm cao mới thông qua công nghệ nhằm nâng cao năng suất lao động.

Kinh doanh của FPT Software xuất phát từ Công nghệ thông tin và Viễn Thông FPT là cốt lõi để đáp ứng cho khách hàng trên 63 tỉnh thành trên toàn lãnh thổ Việt Nam và mở rộng đến khách hàng toàn cầu. Từ khi được thành lập cho đến nay, FPT Software đã là công ty phần mềm số 1 tại Việt Nam trong lĩnh vực tích hợp hệ thống, phân phối và bán lẻ, dịch vụ Công Nghệ Thông tin, xuất khẩu gia công phần mềm, bán lẻ sản phẩm CNTT.

Ở lĩnh vực Viễn Thông FPT là một trong 3 nhà cung cấp dịch vụ internet cáp quang FPT hàng đầu tại Việt Nam. Về lĩnh vực quảng cáo, FPT là đơn vị quảng cáo trực tuyến số 1 tại Việt Nam thông qua trang báo điện tử VNexpress với hơn 42 triệu lượt truy cập mỗi ngày.

# CHƯƠNG 2: TỔNG QUAN VỀ SPRING TRONG JAVA

## Spring MVC

Spring Web model-view-controller framework (MVC) được thiết kế xung quanh một DispatcherServlet để gửi các yêu cầu tới các trình xử lý, với ánh xạ trình xử lý có thể định cấu hình, độ phân giải chế độ xem, ngôn ngữ và độ phân giải chủ đề cũng như hỗ trợ tải tệp lên. Trình xử lý mặc định dựa trên các chú thích(annotation) @Controller và @RequestMapping, cung cấp nhiều phương thức xử lý linh hoạt. Với sự ra đời của Spring 3.0, cơ chế @Controller cũng cho phép bạn tạo các trang Web và ứng dụng RESTful, thông qua chú thích @PathVariable và các tính năng khác.

Trong Spring Web MVC, bạn có thể sử dụng bất kỳ đối tượng nào làm đối tượng lệnh hoặc đối tượng hỗ trợ biểu mẫu; bạn không cần triển khai giao diện hoặc lớp cơ sở dành riêng cho khung. Liên kết dữ liệu của Spring rất linh hoạt: ví dụ: nó coi các kiểu không khớp là lỗi xác thực mà ứng dụng có thể đánh giá chứ không phải là lỗi hệ thống. Do đó, bạn không cần sao chép các thuộc tính của đối tượng nghiệp vụ của mình dưới dạng các chuỗi đơn giản, chưa được nhập trong các đối tượng biểu mẫu của bạn chỉ để xử lý các lần gửi không hợp lệ hoặc để chuyển đổi các Chuỗi đúng cách. Thay vào đó, tốt hơn là liên kết trực tiếp với các đối tượng kinh doanh của bạn.

Spring có nhiều công cụ cực kỳ linh hoạt. Bộ điều khiển thường chịu trách nhiệm chuẩn bị Bản đồ mô hình với dữ liệu và chọn tên chế độ xem nhưng nó cũng có thể ghi trực tiếp vào luồng phản hồi và hoàn thành yêu cầu. Độ phân giải tên của chế độ xem có thể định cấu hình cao thông qua phần mở rộng tệp hoặc Chấp nhận thương lượng loại nội dung tiêu đề, thông qua tên bean, tệp thuộc tính hoặc thậm chí triển khai ViewResolver tùy chỉnh. Mô hình (M trong MVC) là giao diện Bản đồ, cho phép trừu tượng hóa hoàn toàn công nghệ xem. Bạn có thể tích hợp trực tiếp với các công nghệ kết xuất dựa trên mẫu như JSP, Velocity và Freemarker hoặc trực tiếp tạo XML, JSON, Atom và nhiều loại nội dung khác. Bản đồ mô hình đơn giản được chuyển thành một định dạng phù hợp, chẳng hạn như các thuộc tính yêu cầu JSP, một mô hình mẫu Vận tốc.

## Spring Boot

Spring Boot là một bộ khung dựa trên Java mã nguồn mở được sử dụng để tạo một Dịch vụ vi mô. Nó được phát triển bởi Pivotal Team và được sử dụng để xây dựng các ứng dụng Spring sẵn sàng sản xuất và độc lập. Chương này sẽ giới thiệu về Spring Boot và giúp bạn làm quen với các khái niệm cơ bản của nó.

- Thuận lợi:

* Triển khai dễ dàng
* Khả năng mở rộng đơn giản
* Tương thích với Container
* Cấu hình tối thiểu
* Thời gian xây dựng thấp.

## Spring JPA

Spring Data JPA, một phần của họ Spring Data lớn hơn, giúp dễ dàng triển khai các kho lưu trữ dựa trên JPA. Mô-đun này liên quan đến hỗ trợ nâng cao cho các lớp truy cập dữ liệu dựa trên JPA. Nó giúp việc xây dựng các ứng dụng dựa trên Spring sử dụng các công nghệ truy cập dữ liệu trở nên dễ dàng hơn.

Việc triển khai lớp truy cập dữ liệu của một ứng dụng đã khá cồng kềnh trong một thời gian dài. Quá nhiều mã soạn sẵn phải được viết để thực hiện các truy vấn đơn giản cũng như thực hiện phân trang và kiểm tra. Spring Data JPA nhằm mục đích cải thiện đáng kể việc triển khai các lớp truy cập dữ liệu bằng cách giảm nỗ lực xuống mức thực sự cần thiết. Với tư cách là nhà phát triển, bạn viết các giao diện kho lưu trữ của mình, bao gồm các phương thức tìm kiếm tùy chỉnh và Spring sẽ tự động cung cấp triển khai.

- Các đặc trưng của JPA:

* Hỗ trợ tinh vi để xây dựng kho lưu trữ dựa trên Spring và JPA
* Kiểm toán minh bạch của lớp miền
* Hỗ trợ phân trang, thực thi truy vấn động, khả năng tích hợp mã truy cập dữ liệu tùy chỉnh
* Xác thực các truy vấn có chú thích @Query tại thời điểm khởi động
* Hỗ trợ ánh xạ thực thể dựa trên XML
* Cấu hình kho lưu trữ dựa trên JavaConfig bằng cách giới thiệu @EnableJpaRepositories.

## Spring Security

Spring Security là một khung của hệ thống Spring, dành riêng cho việc thiết lập bảo mật ứng dụng bao gồm xác thực và ủy quyền.

- Trong đó có 2 loại:

* Xác thực (Authentication): là cách chúng tôi xác minh danh tính của người đang cố truy cập vào một tài nguyên cụ thể.
* Ủy quyền (Authorization): là xác định ai được phép truy cập vào một tài nguyên cụ thể.

## JWT

JWT là một chuỗi giá trị được mã hóa dưới dạng Base64url, bao gồm 3 phần chính: Header, Payload và Signature.

- Trong dó:

* Header: chứa 2 thông tin chính là loại token (JWT) và thuật toán sử dụng, thường là HS256, SHA256 hoặc RSA.
* Payload: chứa thông tin chính về người dùng (hoặc bất kỳ thông tin nào bạn cho là cần thiết) và các thông tin khác như thời gian hết hạn.
* Signature: để tạo chữ ký cần có phần header được mã hóa, payload được mã hóa, khóa bí mật và thuật toán đã nêu trong phần Header.

## Rest Controller

RestController được sử dụng để tạo các dịch vụ web tĩnh với sự trợ giúp của annotation @RestController. Annotaiton này được sử dụng ở cấp lớp và cho phép lớp xử lý các yêu cầu do máy khách đưa ra. RestController cho phép xử lý tất cả các RestAPI như các yêu cầu GET, POST, DELETE, PUT và nó được kết hợp bởi @Controller và @ResponseBody giúp cho việc xây dựng các RESTful API được dễ dàng hơn, đơn giản hơn.



## Rest Template

RestTemplate được sử dụng để tạo các ứng dụng sử dụng Dịch vụ Web RESTful. Bạn có thể sử dụng phương thức exchange() để sử dụng các dịch vụ web cho tất cả các phương thức HTTP.



# CHƯƠNG 3: YÊU CẦU CỦA DỰ ÁN

- Mục tiêu là xây dựng hệ thống cho tuyển dụng gồm các chức năng chính:

* Với RECcer:
* Quản lý thông tin vị trí tuyển dụng, tuyển sinh.
* Quản lý các hồ sơ ứng tuyển (candidate), histoty của các hồ sơ này, blacklist các kiểu.
* Quản lý thông tin ứng viên (là data collect từ các hoạt động chứ không phải chỉ từ ứng viên nộp CV).
* Suggestion các hồ sơ phù hợp vị trí ứng tuyển dựa vào kỹ năng của ứng viên với các thông tin đang có.
* Quản lý thông tin các interviewer ( người có thể hỗ trợ phỏng vấn).
* Chức năng hỗ trợ và theo dõi tracking các kết quả phỏng vấn, history ứng viên qua các lần phỏng vấn....  ( từ settup MR gởi PV, lưu trữ kết quả PV, ....).
* Các báo cáo report hỗ trợ phân tích.
* Quản lý thông tin tuyển sinh.
* Quản lý việc liên lạc và kết quả với các ứng viên tuyển sinh
* Quản lý thông tin các event và các candidate săn được từ event hoặc các chương trình liên kết.
* Với Candidate ( tham khảo chức năng của vietnamwork, linkedlin)
	+ Có thể xem các vị trí tuyển dụng.
	+ Có thể nôp CV ứng tuyển vào vị trí tuyển dụng.
	+ Có thể tạo CV trên hệ thống và sử dụng để apply vào vị trí tuyển dụng. ==> giống như 1 trang profile của ứng viên. ( tham khảo linkedin).
* Vớt Interviewer
	+ Chấm điểm ứng viên
	+ Các câu hỏi suggestion cho buổi phỏng vấn theo từng skill, từng level.
	+ Đọc được các CV, các thông tin của ứng viên được assigned phỏng vấn.
	+ Hỗ trợ kết nối với các tool phỏng vấn online như team, zoom(optional ).
* Với Admin
	+ Toàn quyền kiểm soát hệ thống và quản lý người dùng.
	+ Admin có được phần lớn các quyền về CRUD, nhưng vẫn sẽ không được can thiệp trực tiếp vào thông tin người dùng đưa lên như CV.

# CHƯƠNG 4: QUÁ TRÌNH LÀM VIỆC CỦA CÁ NHÂN

- Sau khi định hướng được dự án Tuyển dụng, em được phân công về hiện thực quá trình xây dựng hồ sơ người dùng và cập nhật các thông tin của người dùng như bằng cấp, trình độ học vấn,…

## Học cách sử dụng Git Lab

GitLab là một trang web dựa trên DevOps mã nguồn mở và là một phần mềm có chức năng và nhiệm vụ quản lý phiên bản mã nguồn, cung cấp một trình quản lý Git-repository. Trình quản lý này có các tiện ích như wiki, theo dõi sự cố và tích hợp liên tục. Bên cạnh đó là khả năng triển khai các tính năng pipeline và sử dụng license mã nguồn mở được phát triển bởi GitLab Inc.

Vì chúng ta đang làm một dự án gồm nhiều thành viên nên chúng sự dụng git lab để lưu project. Công ty cũng hướng dẫn em sử dụng câu lệnh đơn giản như “git pull”, “git push”, “git checkout –b <branch-name>”, … để phục vụ cho việc lưu project và quản lý các phần được phân công của các bạn trong nhóm.

## **Vẽ ERD**

Đây là bước quan trọng nhất trong dự án vì chúng ta phải hình dung dự án cần những gì, các bảng, trường dữ liệu sẽ định nghĩa các thông tin ra sao và sự liên kết giữa các bảng dữ liệu với nhau diễn ra như thế nào.



## Xây dựng các thực thể (Entity)

- Nhiệm vụ của em làm Profile và các tính năng liên quan đến Profile người dùng.

- Dựa vào ERD ở trên ta viết mã nguồn model cho Profile Ẻntity:

|  |
| --- |
| @Entity@Table(name = "profile")@NoArgsConstructor@AllArgsConstructor@Getter@Setter@Builderpublic class Profile implements Serializable { @Id @GeneratedValue(strategy = GenerationType.IDENTITY) private Long id; @Column(name = "first\_name", nullable = false) private String firstName; @Column(name = "last\_name", nullable = false) private String lastName; @Column(name = "date\_of\_birth") private LocalDate dob; @Column(name = "contact\_number") private String contactNumber; @Column(name = "image\_url") private String userImageURL; @CreationTimestamp private LocalDateTime createdDate; @UpdateTimestamp private LocalDateTime updatedDate; private boolean enabled = true; private String street; private String district; private String city; @Enumerated(EnumType.STRING) @Column(nullable = false) private Gender gender; @ManyToMany(fetch = FetchType.LAZY, targetEntity = CandidateSkill.class) @JoinTable( name = "profile\_skill", joinColumns = @JoinColumn(name = "profile\_id"), inverseJoinColumns = @JoinColumn(name = "candidate\_skill\_id") ) private List<CandidateSkill> candidateSkills; @ManyToMany(fetch = FetchType.LAZY, targetEntity = EducationDetail.class) @JoinTable( name = "profile\_education", joinColumns = @JoinColumn(name = "profile\_id"), inverseJoinColumns = @JoinColumn(name = "education\_detail\_id") ) private List<EducationDetail> educationDetails; @ManyToMany(fetch = FetchType.LAZY, targetEntity = ExperienceDetail.class) @JoinTable( name = "profile\_experience\_detail", joinColumns = @JoinColumn(name = "profile\_id"), inverseJoinColumns = @JoinColumn(name = "experience\_detail\_id") ) private List<ExperienceDetail> experienceDetails; @OneToMany(fetch = FetchType.LAZY, targetEntity = Resume.class) @JoinTable( name = "profile\_resume", joinColumns = @JoinColumn(name = "profile\_id"), inverseJoinColumns = @JoinColumn(name = "resume\_id") ) private List<Resume> resumes; @Override public int hashCode() { return Objects.hash(id, firstName, lastName, dob, contactNumber, userImageURL, createdDate, updatedDate, enabled, street, district, city, gender, candidateSkills, educationDetails, experienceDetails, resumes); } public String toString() { return "Profile{ " + "id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", dob=" + dob + ", contactNumber=" + contactNumber + ", userImageURL=" + userImageURL + ", createdDate=" + createdDate + ", updatedDate=" + updatedDate + ", enabled=" + enabled + ", street=" + street + ", district=" + district + ", city=" + city + ", gender=" + gender + ", candidateSkills=" + candidateSkills + ", educationDetails=" + educationDetails + ", experienceDetails=" + experienceDetails + ", resumes=" + resumes + "}"; }} |

##  Xây dựng Repository

|  |
| --- |
| import org.springframework.data.jpa.repository.JpaRepository;import org.springframework.stereotype.Repository;@Repositorypublic interface ProfileRepository extends JpaRepository<Profile, Long> {} |

##  Xây dựng Service và ServiceImpl

|  |
| --- |
| import java.util.List;import org.springframework.web.multipart.MultipartFile;public interface ProfileService { ResponseTemplate<ProfileDto> getProfileById(Long id); List<ProfileDto> getAllProfiles(); ResponseTemplate<ProfileDto> createProfile(ProfileDto profileDto); ResponseTemplate<ProfileDto> updateProfile(Long id, ProfileDto profileDto); boolean deleteProfile(Long id);} |

|  |
| --- |
| @Primary@Service@RequiredArgsConstructorpublic class ProfileServiceImpl implements ProfileService { private final ProfileRepository profileRepository; private final ProfileMapper profileMapper; @Override public ResponseTemplate<ProfileDto> getProfileById(Long id) { Profile profile = profileRepository .findById(id) .orElseThrow(() -> new ResourceNotFoundException(String.format("Unknown id %s", id))); ProfileDto profileToDto = profileMapper.toProfileDto(profile); return ResponseTemplate.<ProfileDto>builder() .code(HttpStatus.CREATED.value()) .data(profileToDto) .build(); } @Override public List<ProfileDto> getAllProfiles() { List<Profile> profiles = new ArrayList<>(); profiles.addAll(profileRepository.findAll()); return profileMapper.toProfileDtoList(profiles); } @Override public ResponseTemplate<ProfileDto> createProfile(ProfileDto profileDto) { Profile profile = profileMapper.toProfileEntity(profileDto); profile.setFirstName(profileDto.getFirstName()); profile.setLastName(profileDto.getLastName()); profile.setDob(profileDto.getDob()); profile.setGender(Gender.valueOf(profileDto.getGender())); profile.setContactNumber(profileDto.getPhoneNumber()); profile.setUserImageURL(profileDto.getAvatarImageUrl()); Profile savedProfile = profileRepository.save(profile); ProfileDto profileToDto = profileMapper.toProfileDto(savedProfile); return ResponseTemplate.<ProfileDto>builder() .code(HttpStatus.CREATED.value()) .messages("Profile created successfully") .data(profileToDto) .build(); } @Override public ResponseTemplate<ProfileDto> updateProfile(Long id, ProfileDto profileDto) { Profile profile = profileRepository .findById(id) .orElseThrow(() -> new ResourceNotFoundException(String.format("Unknown id %s", id))); if (profileDto.getFirstName() != null) { profile.setFirstName(profileDto.getFirstName()); } if (profileDto.getLastName() != null) { profile.setLastName(profileDto.getLastName()); } if (profileDto.getDob() != null) { profile.setDob(profileDto.getDob()); } if (profileDto.getGender() != null) { profile.setGender(Gender.valueOf(profileDto.getGender())); } if (profileDto.getPhoneNumber() != null) { profile.setContactNumber(profile.getContactNumber()); } if (profileDto.getAvatarImageUrl() != null) { profile.setUserImageURL(profile.getUserImageURL()); } Profile updatedProfile = profileRepository.save(profile); return ResponseTemplate.<ProfileDto>builder() .code(HttpStatus.CREATED.value()) .messages("Profile updated successfully") .data(profileMapper.toProfileDto(updatedProfile)) .build(); } @Override public boolean deleteProfile(Long id) { Profile profile = profileRepository .findById(id) .orElseThrow(() -> new ResourceNotFoundException(String.format("Unknown id %s", id))); profileRepository.delete(profile); return true; }} |

##  Xây dựng Controller

|  |
| --- |
| @RestController@RequestMapping("/api/v1/profile")@RequiredArgsConstructorpublic class ProfileController { private final ProfileService profileService; private final AccountService accountService; @Operation( summary = "Create new profile", description = "Use this endpoint to create a new profile" ) @ApiResponse( responseCode = "201", description = "Successfully created" ) @PostMapping public ResponseEntity<ResponseTemplate<ProfileDto>> createProfile(@Valid @RequestBody ProfileDto profileDto) { return ResponseEntity.status(HttpStatus.CREATED) .body(profileService.createProfile(profileDto)); } @GetMapping("/{id}") public ResponseEntity<ResponseTemplate> getProfile(@PathVariable long id) { return ResponseEntity.ok(profileService.getProfileById(id)); } @PutMapping("/update/{id}") public ResponseEntity<ResponseTemplate<ProfileDto>> updateProfile( @PathVariable Long id, @RequestBody ProfileDto profileDto) { ResponseTemplate<ProfileDto> profile = profileService.updateProfile(id, profileDto); return ResponseEntity.status(HttpStatus.CREATED).body(profile); } @DeleteMapping("/{id}") public ResponseEntity<String> deleteProfile(@PathVariable Long id) { boolean isDeleted = profileService.deleteProfile(id); if (!isDeleted) { String errorMessage = "Profile could not be deleted with id: %s".formatted(id); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(errorMessage); } else { String successMessage = "Profile is deleted successfully!"; return ResponseEntity.ok(successMessage); } } @GetMapping public ResponseEntity<List<ProfileDto>> getAllProfile() { return ResponseEntity.ok(profileService.getAllProfiles()); } @PatchMapping public ResponseEntity<ResponseTemplate<ProfileDto>> editProfile( @Valid @RequestPart(value = "data") ProfileDto profileDto, @RequestPart(required = false, value = "avatar") MultipartFile avatarImage, Principal principal ) { var savedProfile = accountService.editProfile(principal.getName(), DtoUtil.filterNullValue(profileDto), avatarImage); var result = ResponseTemplate.<ProfileDto>builder() .code(HttpStatus.OK.value()) .data(savedProfile) .messages("Updated Successfully") .build(); return ResponseEntity.ok(result); } @GetMapping("/{id}") public ResponseEntity<ResponseTemplate<ProfileDto>> getProfileById(@PathVariable Long id) { ProfileDto profileDto = accountService.getProfileById(id); if (profileDto == null) { return ResponseEntity.notFound().build(); } var result = ResponseTemplate.<ProfileDto>builder() .code(HttpStatus.OK.value()) .data(profileDto) .messages("Profile retrieved successfully") .build(); return ResponseEntity.ok(result); }} |

##  Merge code, Fix bug và hoàn thiện project

- Sau khi thực hiện hết các bước, Cả nhóm bắt đầu hợp nhất mã nguồn(Merge Code), trong quá trình merge code sẽ xuất hiện conflict -> yêu cầu các thành viên cần tìm hiểu về resolve conflict.

- Sau khi merge code, nhóm sẽ chạy thử project để xem trình tự hoạt động của các tính năng và phát hiện các lỗi phát sinh, khi đó nhóm sẽ cùng sửa lỗi.

- Thảo luận với Front-End về các lỗi gặp phải khi triển khai cơ sở dữ liệu(Deploy database) và giải thích sự di chuyển của các luồng dữ liệu khi gọi các method GET hoặc POST.

- Xử lý message lỗi và phối hợp với Front-End để kiểm tra lỗi xảy ra trong các trường hợp nào.

- Phân quyền cho từng API.

# CHƯƠNG 5: TỔNG KẾT

- Trong lúc thực tập tại FPT-Software, em đã tiếp thu nhiều kiến thức. Trong đó:

* Ưu điểm:
* Thấy và tiếp thu thái độ khi làm việc trên công ty
* Các anh chị Mentor và Admin tận tình hướng dẫn, giới thiệu công nghệ mới, quy trình làm việc.
* Các bạn trong nhóm hòa đồng, đoàn kết.
* Công ty còn tổ chức nhiều hoạt động cho sinh viên như Code War, Bóng đá và English Speaking Club.
* Hoàn thành Project đúng thời hạn.
* Nhược điểm:
* Bản thân có chút choáng ngợp trước quy mô của một dự án thực tế.
* Conflict source code khá nhiều vì sự khác biệt về góc độ tiếp cận dự án.
* Chưa tối ưu sức mạnh của các thư viện của Java Spring Boot, cá nhân em sử dụng mã nguồn tiêu chuẩn trong project còn cứng nhắc.

Qua 8 tuần làm quen với công việc, tuy chỉ là một thực tập sinh nhưng em đã có được cái nhìn tổng quan hơn về mảng Spring Boot, có cơ hội để làm quen được phần mềm và các công cụ đi kèm. Công ty đã hỗ trợ em hết mình, cả về vật chất lẫn tinh thần, tạo mọi điều kiện thuận lợi cho em có thể dễ tiếp thu và nắm bắt nguồn kiến thức mới. Sau kì thực tập này, em mong rằng mọi kiến thức thu thập được sẽ góp phần không nhỏ trên bước đường của em sau này, trong cuộc sống cũng như trong môi trường làm việc. Em xin cảm ơn nhà trường đã tạo điều kiện cho em thực tập trong một công ty như FPT này.

Tp. Hồ Chí Minh, ngày… tháng … năm …….

|  |  |  |
| --- | --- | --- |
| **Sinh viên báo cáo****(**Ký, ghi rõ họ và tên**)** | **Người đánh giá****(**Ký, ghi rõ họ và tên**)** | **Ban lãnh đạo công ty****(**Ký, ghi rõ họ và tên**)** |