

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC UEH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH



BÁO CÁO KHÓA LUẬN TỐT NGHIỆP

(Theo hình thức: Học Kỳ Doanh Nghiệp)

Thực tập tại Công Ty: Công Ty TNHH Kỹ thuật số TYME

Từ ngày: 08/08/2022 đến ngày: 04/11/2022

Sinh Viên: TRẦN VĂN HOÀI

Chuyên Ngành: CÔNG NGHỆ PHẦN MỀM

Khóa: K45

Hướng dẫn từ UEH: TS. Đặng Ngọc Hoàng Thành

Hướng dẫn từ Doanh Nghiệp: Võ Hoàng Thông

TP. Hồ Chí Minh, Ngày 25 tháng 10 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC UEH – TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH
CHUYÊN NGÀNH: KỸ THUẬT PHẦN MỀM

CÔNG TY TNHH KỸ THUẬT SỐ TYME
Tầng 19 & 24, Tòa nhà HIU, 215 Điện Biên Phủ,
P.15, Q.Bình Thạnh, TP.HCM
career-vietnam@tyme.com



Khóa Luận Tốt Nghiệp – Học Kỳ Doanh Nghiệp

BÁO CÁO HỌC KÌ DOANH NGHIỆP

Họ tên sinh viên: Trần Văn Hoài

Mã sinh viên: 31191024554

Lớp: ST001

Khóa: 45

Họ tên GVHD: TS. Đặng Ngọc Hoàng Thành

Họ tên Mentor: Võ Hoàng Thông

Niên khóa 2019 – 2022

TP. Hồ Chí Minh, ngày 25 tháng 10 năm 2022

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với các thầy cô của trường Đại học Kinh tế TP.HCM, đặc biệt là thầy cô khoa Công nghệ thông tin kinh doanh của trường đã tận tâm giảng dạy và truyền đạt những kiến thức, kinh nghiệm quý giá trong suốt quá trình em học tập tại đây.

Em xin chân thành gửi lời cảm ơn thầy TS Đặng Ngọc Hoàng Thành, thầy đã dành thời gian quý báu của mình để tận tình giúp đỡ, hướng dẫn em trong suốt thời gian em thực tập tại công ty và hoàn thành bài báo cáo này

Em cũng xin bày tỏ lòng biết ơn sâu sắc đến Ban lãnh đạo Công ty TNHH Kỹ thuật số TYME và các anh chị trong công ty đã tạo điều kiện tốt nhất cho em trong thời gian thực tập tại công ty để em hoàn thiện hơn kiến thức, trang bị được kỹ năng và được trải nghiệm môi trường làm việc thực tế.

Không thể không nhắc đến sự hướng dẫn nhiệt tình của anh Võ Hoàng Thông, người đã truyền đạt cho em những kiến thức thực tiễn, chia sẻ kinh nghiệm trong nghề và hướng dẫn trong suốt quá trình thực tập tại công ty.

Do điều kiện, thời gian và kinh nghiệm còn hạn chế đối với một thực tập sinh, nên bài báo cáo này không thể tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp và sự giúp đỡ của quý thầy cô và cũng như các anh chị trong công ty.

Em xin chân thành cảm ơn!

CHƯƠNG 1. TỔNG QUAN

1.1. Về doanh nghiệp

Tyme là nhóm ngân hàng kỹ thuật số phát triển nhanh nhất thế giới, xây dựng các ngân hàng công nghệ cao và cảm xúc cao.

Công ty TNHH Kỹ thuật số Tyme là chi nhánh quốc tế của Tập đoàn Tyme với các hoạt động tại Hồng Kông, Việt Nam, Philippines và Nam Phi. Sở hữu nền tảng công nghệ và tài sản trí tuệ trên toàn cầu và hợp tác với các đối tác trong nước để ra mắt ngân hàng kỹ thuật số mới.

Tyme Việt Nam là sự kết hợp lý tưởng của các doanh nhân, kỹ sư và chuyên gia ngân hàng. Công ty là những người xây dựng ngân hàng nổi tiếp, những người hình dung lại tương lai kỹ thuật số của ngân hàng thông qua công nghệ.

Triết lý “công nghệ cao, cảm xúc cao” của công ty có nguồn gốc sâu xa trong văn hóa của công ty. Là một tổ chức được hỗ trợ bởi công nghệ, Tyme tận dụng các công nghệ mới nổi và các công cụ cộng tác để tạo ra một nơi làm việc kỹ thuật số và di động nhằm củng cố sự nhanh nhẹn, sáng tạo và làm việc theo nhóm.

Nhưng công ty không quên quan tâm đến cảm xúc - sức mạnh kết nối mọi người trong tập thể Tyme. Công ty cố gắng hết sức để tìm ra sự cân bằng phù hợp giữa thế giới kỹ thuật số và thế giới vật lý; sự kết hợp lành mạnh giúp tăng năng suất và nhân viên phúc lợi. Tyme Việt Nam đang xây dựng một môi trường làm việc tích cực, nhịp độ nhanh, mang lại sự kích thích trí tuệ và cảm hứng cho những tài năng xuất sắc nhất. Ở đây tại Tyme, môi trường làm việc phân chia thành các đội nhỏ nhưng tuyệt vời và tìm thấy những cơ hội thú vị với những cuộc phiêu lưu tuyệt vời.

1.2. Vắn tắt công việc đã làm tại doanh nghiệp

Trong quá trình thực tập tại Tyme, bản thân em đã tham gia vào rất nhiều công việc, từ cá nhân đến nhóm chủ yếu xoay quanh vai trò Java API Back end.

- Tham gia các buổi training do công ty tổ chức.
 - Training về văn hóa của công ty.
 - Training về cách làm việc của những nhóm trong công ty.
 - Training về bảo mật thông tin.
 - Training về kiến thức chuyên môn của vai trò thực tập sinh Java API.
- Tự học, tự tìm hiểu những công nghệ đang được ứng dụng tại công ty, thông qua đó vận dụng các kiến thức để tiến hành xây dựng những dự án cá nhân.
 - Tự học về Java core.
 - Tự học về Spring Boot.
 - Tự học về MySQL, Spring JPA.
 - Tự tìm hiểu về Microservices.
 - Tự tìm hiểu về Kafka.
 - Tự tìm hiểu về SQS.
 - Làm việc với Docker.

- Tham gia vào các hoạt động của nhóm, làm những công việc được người hướng dẫn giao phó (liên quan đến dự án của nhóm).

1.3. Mong muốn từ phía doanh nghiệp

Tyme mong muốn tạo ra được cơ hội cho các tài năng trẻ, còn thiếu sót về kinh nghiệm cũng nhưng kiến thức chuyên môn. Qua đó, tìm kiếm và thúc đẩy những cá nhân có khả năng phù hợp với công ty.

Xây dựng, phát triển và định hình hướng đi đúng đắn nhất cho thực tập sinh. Tyme cũng mong muốn những cá nhân sau quá trình thực tập ở công ty, có thể học được nhiều nhất có thể về những kiến thức chuyên môn, quy trình làm việc giữa các nhóm, tạo dựng các mối quan hệ và phát triển được các cộng đồng trong công ty.

Nâng cao mức độ tiếp cận thương hiệu của Tyme với tư cách là môi trường làm việc thuộc top đầu về công nghệ.

1.4. Mong muốn từ phía sinh viên

Bản thân em rất háo hức khi được tham gia vào vai trò thực tập sinh Java API tại công ty Tyme. Ngoài những kiến thức chuyên môn để đáp ứng được với vai trò này, em còn mong muốn mình còn có thể học hỏi được những công nghệ mới đang được ứng dụng tại Tyme.

Bên cạnh đó, còn có những kiến thức về văn hóa, môi trường làm việc ở đây – thứ mà bản thân em có thể học hỏi được qua quá trình làm việc và xây dựng các mối quan hệ tại đây.

Và tất nhiên, bản thân em cũng định hướng rõ ràng sẽ trở thành nhân viên chính thức tại Tyme với vai trò là một lập trình viên Fresher Java API sau quá trình thực tập.

CHƯƠNG 2. NHẬT KÝ CÔNG VIỆC

Tuần 1 (từ 08/08/2022 đến 12/08/2022)			
Ngày		Nội dung thực tập	Kết quả đạt được
Thứ 2 08/08/2022	10h30 - 11h15	Tham gia buổi chào mừng Onboard và chuẩn bị các thiết bị để bắt đầu quá trình làm việc.	- Nắm rõ được những gì sẽ diễn ra trong các ngày sắp tới. - Được cung cấp và cài đặt thiết bị laptop để phục vụ quá trình làm việc.
	11h15 - 12h	Tham gia vào buổi phổ biến hợp đồng lao động và các chính sách của công ty.	- Biết được những quy định trong hợp đồng lao động. - Biết được các chính sách phúc lợi của công ty.

	13h30 - 17h	<ul style="list-style-type: none"> - Tham gia buổi giới thiệu chi tiết hơn về công ty Tyme, về giá trị cốt lõi của công ty. - Chia sẻ cơ hội nghề nghiệp tại Tyme. 	<ul style="list-style-type: none"> - Hiểu hơn về công ty, văn hóa, con người và môi trường làm việc. - Nắm được các giá trị cốt lõi mà công ty hướng đến và cách để đạt được điều đó. - Biết được lộ trình thăng tiến ở Tyme. - Đánh giá quá trình dựa trên Growth Framework
Thứ 3 09/08/2022	9h - 12h	Tham gia buổi training về Engineering Culture	<ul style="list-style-type: none"> - Biết được về các cộng đồng Engineering trong công ty. Hiểu được vai trò, nhiệm vụ và giá trị mang lại của từng cộng đồng trong công ty. - Hiểu về văn hóa Engineering ở Tyme.
	13h - 18h	Làm quen với nhóm, quy trình làm việc của nhóm.	<ul style="list-style-type: none"> - Làm quen với mọi người trong nhóm Cypher. - Nhóm đang áp dụng mô hình Agile trong quy trình làm việc. - Cypher chịu trách nhiệm về Core Payment cho hệ thống ngân hàng điện tử GoTyme. - Thị trường của GoTyme là Philippines.
Thứ 4 10/08/2022	9h - 12h	Tự học về bảo mật thông tin thông qua website KnowBe4.	<ul style="list-style-type: none"> - Biết về đạo luật bảo vệ thông tin (POPIA). - Nâng cao nhận thức về an ninh thông tin. - Nắm bắt được những mối đe dọa tiềm tàng. - Hoàn thành các bài tập cuối mỗi bài giảng.
	14h - 18h	Tham gia buổi training về Cyber Security	<ul style="list-style-type: none"> - Biết được các cách bảo mật thông tin. - Biết sử dụng những tài sản được chấp thuận sử dụng. - Nắm bắt được những mẹo bảo mật. - Nhận biết và báo cáo những sự cố liên quan đến bảo mật. - Biết được quy trình testing SDLC, pen test.

Thứ 5 11/08/2022	9h - 12h	Tự học về GIT thông qua khóa học online công ty cung cấp.	<ul style="list-style-type: none"> - Hiểu rõ về GIT và ứng dụng của nó. - Tiến hành cài đặt GIT trên máy tính. - Nắm bắt được các câu lệnh trong GIT.
	14h - 17h	Tham gia buổi training về GIT cùng với người hướng dẫn.	<ul style="list-style-type: none"> - Biết được quy trình phát triển một dự án sử dụng GIT. - Biết và phân biệt được Trunk based và Gitflow. - Luyện tập GIT: + clone/pull/push/commit code + pull request/resolve conflict/rebase
Thứ 6 12/08/2022	9h30 - 12h, 13h - 15h	Tự học Java core và Dependency management (Maven) qua khóa học online công ty cung cấp.	<ul style="list-style-type: none"> - Nắm được các khái niệm trong lập trình hướng đối tượng. - Nắm được toán tử và kiểu dữ liệu trong Java 8. - Biết cách sử dụng Interface, lớp Abstract. - Nắm được các Collection và Generics trong Java 8. - MultiThreading và Concurrency trong Java.
	15h - 18h	Tham gia buổi training về Java core và Dependency management với người hướng dẫn.	<ul style="list-style-type: none"> - Setup JDK, Maven, IntelliJ. - Hiểu được cách hoạt động và sử dụng Multitasking/MultiThreading. - Nhận được những chia sẻ về Java 8 của các anh hướng dẫn. - Hiểu rõ về Stream API trong Java 8.
Tuần 2 (từ 15/08/2022 đến 19/08/2022)			
Ngày		Nội dung thực tập	Kết quả đạt được

<p>Thứ 2 15/08/2022</p>	<p>9h30 - 12h, 13h - 18h</p>	<p>Tự học Spring Boot thông qua khóa học online mà công ty cung cấp (phần 1).</p>	<ul style="list-style-type: none"> - Setup một dự án Spring Boot bằng Spring Boot Initializr. - Hiểu Dependency Injection trong Spring Boot. + Khai báo một bean bằng annotation @Bean, @Component, @Service, @Repository. + Sử dụng annotation @Inject, @Autowired để inject một bean. + Ngoài ra, có thể sử dụng annotation @RequiredArgsConstructor từ thư viện Lombok để tự động tạo ra constructor chứa các trường cần inject. - Thử tạo một api đơn giản và sử dụng dependency injection. - Hiểu và ứng dụng viết unit test bằng thư viện JUnit 5. + Biết được cách để viết một test case unit test bằng thư viện JUnit 5. + Sử dụng thư viện Mockito để giả lập dữ liệu, ghi hình lại dữ liệu. - Nắm bắt và hiểu về RESTful API, cách để tạo một RESTful API trong Spring Boot. + @RequestMapping. + @GetMapping, @PostMapping, @PutMapping, @DeleteMapping. + Khai báo đường dẫn url cho controller.
-----------------------------	----------------------------------	---	---

Thứ 3 16/08/2022	9h30 - 12h, 13h - 18h	Tham gia buổi training về Spring Boot với người hướng dẫn.	<ul style="list-style-type: none"> - Hiểu được cấu trúc N-tier và ứng dụng vào trong dự án Spring Boot. + Xây dựng dự án với 3 tầng: Controller, Service, Repository. + Giao tiếp giữa tầng Controller và Service thông qua DTO. + Giao tiếp giữa tầng Service và Repository thông qua Entity. - Hiểu rõ hơn về Dependency Injection trong Spring Boot. - Nắm bắt được cách sử dụng thư viện Lombok để viết code nhanh hơn. - Thực hành tạo một dự án Spring Boot đơn giản để quản lý nhân viên. + Khởi tạo dự án bằng Spring Boot Initializr (thêm thư viện Lombok). + Xây dựng cấu trúc 3 tầng Controller, Service và Repository. + Tạo API để quản lý nhân viên. + Tạo Service để xử lý CRUD nhân viên. + Tạo Repository tương ứng cho nhân viên. - Nắm bắt được cách để kiểm tra dữ liệu. + Các annotation để định nghĩa cho một trường dữ liệu: @Null, @NotNull, @Size, @Length, ... + Áp dụng kiểm tra dữ liệu cho Controller bằng annotation @Validated và @Valid. - Biết được cách sử dụng Postman để test các API. - Biết được sử dụng Swagger của thư viện OpenAPI 3 để mô tả RESTful API.
---------------------	--------------------------	--	---

Thứ 4 17/08/2022	9h - 12h	Tự học Spring Boot thông qua khóa học online mà công ty cung cấp (phần 2).	<ul style="list-style-type: none"> - Nắm bắt được cách truyền dữ liệu qua controller thông qua các annotation như @RequestParam, @RequestBody, @PathVariable, ... trong Spring Boot. - Biết được cách cấu hình nhận và trả về dữ liệu dưới dạng gì bằng cách sử dụng annotation @Produces và @Consumes. - Biết được thêm các annotation khác như @JsonProperty, @JsonIgnore, ... để cấu hình cho dữ liệu json. - Biết được cách để gọi đến những API khác thông qua RESTEasy, RESTEasy Client hay Jersey.
	13h30 - 18h	Tham gia buổi training về Spring Data JPA với người hướng dẫn.	<ul style="list-style-type: none"> - Phân biệt được SQL và NoSQL. - Nắm bắt được các loại command trong SQL: <ul style="list-style-type: none"> + DDL: CREATE, ALTER, DROP, TRUNCATE. + DML: INSERT, UPDATE, DELETE. + DCL: GRANT, REVOK. - Hiểu về Spring JPA (là một ORM Framework). + Biết cách sử dụng Connector (JDBC) hoặc Hibernate (thực thi Interface JPA) để kết nối dữ liệu đến MySQL Server. + Biết được các annotation để định danh dữ liệu entity như @Table, @Column, @Entity, ... + Biết cách viết câu truy vấn thuần trong Spring JPA. + Nắm bắt được cách xây dựng ứng dụng dựa trên Code-First hoặc Database-First. + Biết được thư viện Flyway sử dụng cho việc xây dựng Database-First. + Cấu hình các mối quan hệ

			giữa các entity với nhau thông qua annotation @OneToOne, @ManyToOne, @OneToMany, @ManyToMany. + Hiểu về cơ chế Blocking.
Thứ 5 18/08/2022	9h - 12h, 13h - 18h	Tự xây dựng một dự án Spring Boot sử dụng Spring JPA để quản lý Nhân viên và Tòa nhà.	<ul style="list-style-type: none"> - Xây dựng thành công dự án sử dụng Spring Boot kết hợp cùng Spring JPA để quản lý nhân viên. - Hiểu rõ hơn cách để xây dựng và cấu hình các đoạn code một cách đúng đắn nhất.
Thứ 6 19/08/2022	10h - 12h	Tham gia buổi training về kiến trúc Microservices với người hướng dẫn.	<ul style="list-style-type: none"> - Biết thêm về kiến trúc Microservice và kiến trúc Monolith. - Nắm bắt được những khái niệm chính trong Microservice: <ul style="list-style-type: none"> + Khả năng triển khai độc lập. + Mô hình hóa dựa trên Business Domain. + Tự sở hữu những thứ thuộc về nó. + Tính linh hoạt (mỗi service có thể phát triển trên một ngôn ngữ khác nhau). + Có thể giao tiếp với các services nhưng không biết sự tồn tại lẫn nhau (giảm sự phụ thuộc). - Nắm bắt được cách để phân chia các services (dựa trên mẫu thiết kế Domain-Driven), mỗi service sẽ đảm nhiệm một business domain. - Nắm bắt được các cách để giao tiếp giữa các services. <ul style="list-style-type: none"> + Giao tiếp đồng bộ và bất đồng bộ. + Giao tiếp thông qua message queue qua cơ chế publish, subscribe. + Giao tiếp thông qua Kafka với cơ chế topic, subscribe. - Biết được về cách để deploy các services.

	13h - 16h	Tham gia buổi training về Unit test với người hướng dẫn.	<ul style="list-style-type: none"> - Hiểu rõ hơn về Unit test. - Biết được khái niệm về Code Coverage, Branch Coverage. - Biết được các best practices về Unit test thông qua ví dụ của người hướng dẫn. - Nắm bắt cách sử dụng thư viện Mockito.
Tuần 3 (từ 22/08/2022 đến 26/08/2022)			
	Ngày	Nội dung thực tập	Kết quả đạt được
Thứ 2 22/08/2022	9h30 - 12h	<ul style="list-style-type: none"> - Tự tìm hiểu về Docker. - Setup MySQL trên docker. - Triển khai dự án Spring Boot lên Docker. 	<ul style="list-style-type: none"> - Hiểu rõ hơn về Docker và cách hoạt động của nó. + Biết được về Docker Image, Docker Container. + Cách để pull image về và tạo container cho nó. - Setup thành công MySQL trên Docker. - Triển khai thành công dự án Spring Boot lên Docker. + Viết Dockerfile để tạo Docker image. + Chạy dự án trên Docker thông qua Docker container.
	12h - 15h30	<ul style="list-style-type: none"> - Tự tìm hiểu và nghiên cứu về Microservices trong Spring Boot. - Xây dựng microservices để xử lý: Khi một thanh toán được thực hiện thì: + Tiền được cộng vào tài khoản. + Một tin nhắn sms sẽ được gửi đến số điện thoại của người nhận. + Một push notification sẽ được gửi đến điện thoại của người nhận. 	<ul style="list-style-type: none"> - Nắm được cách xây dựng Microservices trong Spring Boot. - Tiến hành xây dựng dự án Spring Boot Microservice, phân chia các chức năng nhỏ thành một dự án Spring Boot riêng và thực thi những chức năng riêng của chúng.

	15h30 - 17h	<p>Tham gia meeting để bàn luận về công việc của nhóm:</p> <p>Thêm các giá trị cho phần Tags khi Finalize một giao dịch.</p>	<ul style="list-style-type: none"> - Thảo luận về những thứ cần thêm vào phần Tags khi finalize một payment. - Biết được giải pháp để thực hiện việc đó.
Thứ 3 23/08/2022	9h30 - 10h	<p>Tham gia meeting hằng ngày để catch up với nhóm.</p>	<ul style="list-style-type: none"> - Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì. - Thảo luận thêm về vấn đề có thể gặp phải đối với việc sử dụng kiểu dữ liệu LocalDate (khác nhau về mặt múi giờ giữa các nước). - Biết được rằng nhóm đang áp dụng Pair Programming để nâng cao hiệu suất làm việc.
	10h - 12h, 13h - 18h	<ul style="list-style-type: none"> - Tiếp tục xây dựng microservices để xử lý: Khi một thanh toán được thực hiện thì: + Tiền được cộng vào tài khoản. + Một tin nhắn sms sẽ được gửi đến số điện thoại của người nhận. + Một push notification sẽ được gửi đến điện thoại của người nhận. - Tìm hiểu cách giao tiếp giữa các services. 	<ul style="list-style-type: none"> - Hoàn thành các service riêng lẻ, sử dụng Spring JPA và Flyway để làm việc với database. - Biết cách sử dụng Eureka Server để quản lý và đặt tên cho các service. - Biết cách sử dụng Spring Cloud Gateway để tạo ra các cổng kết nối đến các service đã được đăng ký với Eureka. - Hoàn thành dự án microservice đơn giản.
Thứ 4 24/08/2022	9h30 - 10h	<p>Tham gia meeting hằng ngày để catch up với nhóm.</p>	<ul style="list-style-type: none"> - Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì. - Nắm bắt được kết quả của việc áp dụng Pair Programming vào trong quá trình làm việc.

	10h30 - 11h30	Tham gia meeting: Engineering practices biweekly.	<ul style="list-style-type: none"> - Biết thêm về Trunk based trong quy trình phát triển phần mềm, các nhóm đang áp dụng Trunk based. - Nắm bắt được những lợi ích và ảnh hưởng khi áp dụng Trunk based vào trong quy trình phát triển phần mềm. - Đưa ra được những blocker nếu áp dụng Trunk based.
	13h - 18h	<ul style="list-style-type: none"> - Tự tìm hiểu và nghiên cứu về cách giao tiếp giữa các services trong microservices. - Nghiên cứu về Kafka và ứng dụng. 	<ul style="list-style-type: none"> - Hiểu rõ hơn về event-driven trong microservices. - Nắm được cách để tránh khỏi blocking khi giao tiếp giữa các services. - Biết được cách sử dụng Kafka để giao tiếp. + Setup kafka trên máy và trong dự án Spring Boot. + Tạo topic. + Subscribe đến topic. + Áp dụng vào dự án Microservices đã làm hôm trước.
Thứ 5 25/08/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h, 13h - 14h	Tham gia buổi training về Docker.	<ul style="list-style-type: none"> - Hiểu rõ hơn các khái niệm trong Docker: + Container. + Image. + Docker engine. + Docker hub. - Hiểu được Docker compose và cách hoạt động của nó. - Nắm bắt và vận dụng để triển khai ứng dụng lên Docker thông qua việc viết Docker compose.

	14h - 18h	Thực hành viết Docker compose để triển khai ứng dụng Microservices đã xây dựng trước đó.	<ul style="list-style-type: none"> - Biết cách viết Docker file để tự động build ra file JAR phục vụ tạo Docker image. - Biết cách viết một file Docker compose hoàn chỉnh. - Chạy thành công Docker compose file và triển khai thành công dự án lên Docker.
Thứ 6 26/08/2022	10h - 11h	Tham gia meeting: Sprint refinement	<ul style="list-style-type: none"> - Nắm bắt được những công việc đã hoàn thành và chưa hoàn thành của nhóm trong sprint hiện tại. - Thảo luận mức độ ưu tiên và những công việc cần phải làm cho sprint tiếp theo.
	11h - 12h	Tham gia meeting: HR catch up	<ul style="list-style-type: none"> - Catch up với HR về quãng thời gian đã qua khi bắt đầu thực tập tại Tyme. - Nắm bắt được kế hoạch sắp tới của công ty đối với thực tập sinh.
	13h - 18h	Annual Leave	
Tuần 4 (từ 29/08/2022 đến 02/09/2022)			
Ngày		Nội dung thực tập	Kết quả đạt được
Thứ 2 29/08/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	<ul style="list-style-type: none"> - Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì. - Thảo luận thêm về vấn đề có thể gặp phải đối với việc sử dụng kiểu dữ liệu LocalDate (khác nhau về mặt múi giờ giữa các nước). - Biết được rằng nhóm đang áp dụng Pair Programming để nâng cao hiệu suất làm việc.
	10h - 12h	Tham gia meeting với mentor để catch up và giới thiệu chi tiết hơn về những gì nhóm Cypher đang làm.	<ul style="list-style-type: none"> - Nắm bắt được những services mà nhóm Cypher đang phát triển. - Hiểu được luồng đi và cách giao tiếp giữa các services.

	13h30 - 17h	Tự tìm hiểu Spring Cloud OpenFeign.	<ul style="list-style-type: none"> - Biết được Feign là một REST Client dùng cho ứng dụng Spring Boot. - Nắm được cách setup dependency, cấu hình và tạo một Feign Client (annotation <code>@FeignClient</code>, <code>@EnableFeignClients</code>, ...). - Áp dụng vào dự án Microservices đã xây dựng trước đó thay thế cho RestTemplate để gọi đến các services.
Thứ 3 30/08/2022	9h - 9h30	Tham gia meeting: Discuss about Activity History for Taliad	Nắm bắt được những gì cần phải làm cho chức năng lịch sử giao dịch hỗ trợ cho bộ phận chăm sóc khách hàng của app GoTyme.
	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	<ul style="list-style-type: none"> - Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì. - Thảo luận thêm về vấn đề có thể gặp phải đối với việc sử dụng kiểu dữ liệu <code>LocalDate</code> (khác nhau về mặt múi giờ giữa các nước). - Biết được rằng nhóm đang áp dụng Pair Programming để nâng cao hiệu suất làm việc.
	10h30 - 11h30	Tham gia meeting: Sprint retrospective.	Nhìn lại sprint vừa qua đã làm được những gì, những gì tốt, những gì chưa tốt và cách khắc phục.

	13h30 - 18h	Tham gia buổi training về Message Broker với người hướng dẫn.	<ul style="list-style-type: none"> - Biết thêm về Distributed Computing. - Hiểu rõ hơn về Blocking và Non-Blocking trong giao tiếp giữa các microservices. - Biết về Message Broker. + Giúp xây dựng một cơ chế tích hợp chung để hỗ trợ các kiến trúc Cloud Native, microservices-based, serverless, hybrid. + Message broker là một phần của phần mềm, cho phép services và application giao tiếp với nhau thông qua messages. - Biết được vì sao nên sử dụng message để giao tiếp thay thế cho việc gọi trực tiếp qua api. + Giảm sự phụ thuộc giữa các services. + Giao tiếp bất đồng bộ. + Có thể sử dụng cho nhiều ngôn ngữ, đa dạng hệ thống. + Khả năng chịu lỗi khi bất kì một service nào không hoạt động thì các services khác vẫn hoạt động bình thường. + Hỗ trợ mở rộng theo chiều ngang. - Hiểu rõ hơn về Kafka và lý do nên dùng Kafka để giao tiếp trong các services. - Hiểu rõ hơn về AWS SQS, cách thức hoạt động. - Được xem demo về một dự án Microservices sử dụng Kafka để giao tiếp.
Thứ 4 31/08/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.

	10h30 - 12h, 13h - 18h	Tự tìm hiểu và nghiên cứu sử dụng Kafka thay thế cho cách giao tiếp bằng OpenFeign đến các services trong dự án Microservices đã xây dựng trước đó.	<ul style="list-style-type: none"> - Biết được cách setup Kafka trên máy tính và trên Docker. - Hiểu cách hoạt động của Kafka: Thông qua topic và consumer. - Nắm rõ cách để áp dụng Kafka vào một dự án Spring Boot. + Cách để publish payload đến topic. + Cách để consumer data từ topic và xử lý.
Thứ 5 01/09/2022		Nghỉ lễ 2/9.	
Thứ 6 02/09/2022		Nghỉ lễ 2/9.	
Tuần 5 (từ 05/09/2022 đến 09/09/2022)			
	Ngày	Nội dung thực tập	Kết quả đạt được
Thứ 2 05/09/2022	11h - 12h	Tham gia meeting: Planning for sprint #37.	<ul style="list-style-type: none"> - Nắm bắt lại những công việc còn dang dở trong sprint 36. - Biết được những công việc sẽ làm cho sprint 37. - Phân chia nhiệm vụ cho từng người.
	16h - 18h	Tham gia meeting: Breakdown story for sprint #37 (Activity History)	<ul style="list-style-type: none"> - Nắm bắt những gì cần phải làm để hoàn thành chức năng lịch sử giao dịch cho bộ phận chăm sóc khách hàng GoTyme. + Cần consume và publish những event nào. + Phân nhỏ công việc thành các công việc nhỏ hơn để chia đều cho từng thành viên.
Thứ 3 06/09/2022	9h - 9h30	Tham gia meeting: Example mapping sprint #37.	<ul style="list-style-type: none"> - Biết được những rules cần có cho chức năng lịch sử giao dịch cho bộ phận chăm sóc khách hàng app GoTyme. - Tất cả các thành viên trong nhóm Cypher đều tham gia để đóng góp ý kiến cho các rules.
	10h - 12h	Tự viết BDD dựa trên những rules đã được chỉ ra.	<ul style="list-style-type: none"> - Biết cách để viết một BDD test. + Cấu trúc: Given - When - Then. + Mỗi rules là một scenario.

Thứ 4 07/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h30 - 11h30	Tham gia meeting: Engineering practices biweekly.	
	13h30 - 17h	Thực hiện Pair Programming với anh Mentor để khởi tạo service mới phục vụ cho chức năng lịch sử giao dịch của bộ phận chăm sóc khách hàng app GoTyme (Taliad Service).	<ul style="list-style-type: none"> - Biết cách sử dụng TymePipeline để khởi tạo một service (gồm repository, infrastructure). - Biết được Pair Programming như nào (gồm 2 roles: Driver và Navigator). - Tạo thành công repository cho service và repository cho infrastructure.
Thứ 5 08/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h, 15h - 17h	Khởi tạo dự án Spring Boot cho repository của services đã tạo trước đó.	<ul style="list-style-type: none"> - Biết được cách để tạo dự án Spring Boot và đưa code lên repository. - Nắm được cấu trúc các tầng của dự án. - Biết được cách đánh version cho dự án. - Hiểu hơn cách CI/CD chạy, sử dụng Sonar Cloud để kiểm tra cách lỗi về dependency.
Thứ 6 09/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	14h - 16h	Tham gia meeting: Deep Dive into AWS IAM	<ul style="list-style-type: none"> - Hiểu hơn các khái niệm về IAM core trên AWS. - Nắm rõ cách để cấu hình các loại Policy cho IAM. - Biết được các bestpractices của IAM áp dụng cho GoTyme và TymeBank.
Tuần 6 (từ 12/09/2022 đến 16/09/2022)			
Ngày		Nội dung thực tập	Kết quả đạt được

Thứ 2 12/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h	Làm Task #1 do mentor giao: Update Taliad PUC to use new topic: manual-entry-order-events	<ul style="list-style-type: none"> - Biết cách để thay đổi topic dùng cho việc publish event (Kafka) bằng payment resource. - Hiểu hơn về luồng đi của code khi publish và subscribe từ topic.
	14h - 15h30	Tham gia meeting: ARB	Biết về việc sử dụng Egress Private API và Private Intergration để kết nối đến bên thứ 3 thông qua một kết nối riêng tư.
Thứ 3 13/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h	Deploy Task #1 lên môi trường SIT.	<ul style="list-style-type: none"> - Biết cách để deploy một service lên môi trường cao hơn phục vụ testing. - Nắm bắt được các môi trường đang có trong hệ thống (DEV -> SIT -> UAT -> PROD).
	14h - 15h	Tham gia buổi training: AWS Enterprise Onboarding.	<ul style="list-style-type: none"> - Biết về hệ thống hỗ trợ của AWS. - Biết cách thức liên hệ hỗ trợ khi gặp sự cố.
Thứ 4 14/09/2022	9h30 - 10h30	Tham gia meeting: Philippines Roadmap review biweek.	<ul style="list-style-type: none"> - Nắm bắt được những công việc đã hoàn thành trong sprint vừa qua của các feature team. - Biết được những công việc mà các team sắp tới phải hoàn thành.
	10h30 - 11h30	Tham gia meeting: Engineering practices biweekly.	<ul style="list-style-type: none"> - Biết được vấn đề sẽ gặp phải khi áp dụng Trunk based vào trong quy trình phát triển sản phẩm. - Đưa ra được những giải pháp có thể giải quyết những vấn đề trên.

	13h30 - 17h	Deploy Task #1 lên môi trường UAT.	<ul style="list-style-type: none"> - Viết CAB để xin deploy task #1 từ môi trường SIT lên môi trường UAT. - Biết được quy trình xin CAB, cần phải có 2 sự đồng ý đến từ UAT-Reviewers. - Biết được cách deploy lên môi trường UAT trên hệ thống AWS.
	17h - 18h	Tham gia meeting: TYMEX - TOWN HALL.	<ul style="list-style-type: none"> - Nắm được các hoạt động của toàn bộ Tyme Group. - Kế hoạch sắp tới của TymeX.
Thứ 5 15/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	13h - 18h	Tự học về thư viện Lombok trong Spring Boot.	<ul style="list-style-type: none"> - Biết cách setup dependency vào trong dự án Spring Boot. - Nắm rõ hơn các annotation hỗ trợ trong việc viết code nhanh hơn: <ul style="list-style-type: none"> + @Getter/@Setter. + @ToString + @EqualsAndHashCode + @NoArgsConstructor, @RequiredArgsConstructor and @AllArgsConstructor + @Builder
Thứ 6 16/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.

	10h - 12h, 13h - 18h	Tự nghiên cứu về các services hiện có của team Cypher.	<ul style="list-style-type: none"> - Biết được các services mà team Cypher đang phát triển và luồng đi dữ liệu của chúng. + Payment Auth: Chịu trách nhiệm xử lý authorize các giao dịch - Core banking. + Payment Processing: Chịu trách nhiệm nhận kết quả từ Payment Auth và đồng bộ lên Mambu. + Bank Account: Chịu trách nhiệm tạo tài khoản cho các service. - Hiểu được cách làm thế nào để một transaction có thể được authorize trong Payment Auth. - Nắm rõ cái được gọi là PUC - thứ để dựa trên đó để authorize payment.
Tuần 7 (từ 19/09/2022 đến 23/09/2022)			
	Ngày	Nội dung thực tập	Kết quả đạt được
Thứ 2 19/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h	Thực hiện Pair Programming với anh Mentor để xử lý Task #2: [Taliad Service] Consume profile event and store profile.	<ul style="list-style-type: none"> - Biết cách configuration cho các consumer để consume từ topic (Kafka). - Hiểu được profile event contract gồm những gì. - Biết được cách để consumer event và parse sang Object sử dụng ObjectMapper.
	13h30 - 17h	Thực hiện Task #3: [Taliad Service] Add DB migration script for Profile.	<ul style="list-style-type: none"> - Biết cách sử dụng Flyway để làm database first. - Viết được các câu query để tạo bảng Profile - nơi chứa thông tin các Profile của GoTyme app. - Tạo entity Profile và định nghĩa các cột và định dạng dữ liệu cho từng cột.
Thứ 3 20/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.

	10h - 12h, 13h30 - 18h	Thực hiện Pair Programming với anh Mentor để tiếp tục xử lý Task #2: [Taliad Service] Consume profile event and store profile.	<ul style="list-style-type: none"> - Viết các BDD test, Unit test cho phần consume profile event. - Biết cách để chạy một BDD test thông qua Cucumber.
Thứ 4 21/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h30 - 11h30	Tham gia meeting: Engineering practices biweekly.	
	13h30 - 17h	Thực hiện Pair Programming với thành viên trong nhóm để xử lý Task #4: [Taliad Service] Write step definition for BDD testing.	Thực hiện viết BDD test cho phần consume general-entry-payment-order-events event (Kafka) và publish đến topic taliad-activity-events (Kafka).
Thứ 5 22/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	13h30 - 17h	Thực hiện Pair Programming với anh Mentor để xử lý Task #5: [Taliad Service] Fix consume profile when accountNo = null.	<ul style="list-style-type: none"> - Tiến thành testing việc consume profile event. - Hiểu được vì sao gặp lỗi xuất hiện. - Nắm được cách để fix lỗi bằng cách thay đổi ràng buộc dữ liệu cho trường accountNo thành có thể bị Null.
Thứ 6 23/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	14h30 - 15h30	Tham gia buổi training với người hướng dẫn: Professionalism in the Workplace with TYME.	<ul style="list-style-type: none"> - Catch up với HR về quãng thời gian đã qua. - Biết được cách để làm việc một cách chuyên nghiệp và hiệu quả. - Lắng nghe các thực tập sinh

			khác chia sẻ về làm việc chuyên nghiệp.
	16h - 17h	Tham gia buổi All-Hands	<ul style="list-style-type: none"> - Biết được các thông tin về Business mới nhất. - Lắng nghe các Kudos của từng thành viên trao cho nhau. - Biết được về tình hình nhân sự.
Tuần 8 (từ 26/09/2022 đến 30/09/2022)			
	Ngày	Nội dung thực tập	Kết quả đạt được
Thứ 2 26/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h	Thực hiện Pair Programming với anh Mentor để xử lý Task #6: [Taliad Service] Publish Notification event.	<ul style="list-style-type: none"> - Biết cách cung cấp template push notification cho notification service. - Publish notification event theo event contract.
	13h30 - 17h	Thực hiện smoke test cho phần lịch sử giao dịch và push notification (Taliad Service).	<ul style="list-style-type: none"> - Biết cách để install app trên điện thoại qua TestFlight. - Biết cách tạo ra một giao dịch bằng tay (thông qua UI của bên Taliad). - Kiểm thử việc hoạt động của Taliad Service có chính xác hay không.
Thứ 3 27/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	14h - 16h	Tham gia meeting: Philippines sprint review.	<ul style="list-style-type: none"> - Biết được những gì mà các feature team đã hoàn thành trong sprint vừa qua. - Xem demo chức năng đã hoàn thành của các feature team.

Thứ 4 28/09/2022	9h30 - 10h30	Tham gia meeting: Philippines Roadmap review biweek.	<ul style="list-style-type: none"> - Nắm bắt được những công việc đã hoàn thành trong sprint vừa qua của các feature team. - Biết được những công việc mà các team sắp tới phải hoàn thành.
	10h30 - 11h30	Tham gia meeting: Engineering practices biweekly.	<ul style="list-style-type: none"> - Biết được các chức năng đã hoàn thành cho GoTyme app. - Review lại một lần kỹ lưỡng từng chức năng để chuẩn bị cho đợt release sắp tới.
	15h30 - 16h30	Tham gia meeting: Sprint Planning Sprint #39	<ul style="list-style-type: none"> - Nắm bắt lại những công việc còn dang dở trong sprint 38 - Biết được những công việc sẽ làm cho sprint 39. - Phân chia nhiệm vụ cho từng người.
Thứ 5 29/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	13h30 - 15h	Tham gia meeting: Example mapping and Breakdown story sprint #39 for Payment <-> Mambu Recon (Payment Recon Service).	<ul style="list-style-type: none"> - Biết được những rules cần có cho chức năng đối soát giao dịch sau mỗi lần xử lý và gửi đến service khác. - Tất cả các thành viên trong nhóm Cypher đều tham gia để đóng góp ý kiến cho các rules. - Phân nhỏ task lớn thành từng task con để chia đều cho các thành viên.
Thứ 6 30/09/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h	Viết BDD test cho phần Example mapping Sprint #39.	Dựa trên những rules đã được thảo luận hôm qua viết ra BDD test cho các rules.
	16h - 17h	Tham gia buổi All-Hands	<ul style="list-style-type: none"> - Biết được các thông tin về Business mới nhất. - Lắng nghe các Kudos của từng thành viên trao cho nhau. - Biết được về tình hình nhân sự.
Tuần 9 (từ 03/10/2022 đến 07/10/2022)			

Ngày		Nội dung thực tập	Kết quả đạt được
Thứ 2 03/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h	Thực hiện Task #7: [Confluence page] Define transaction posted event.	- Viết document cho Transaction Posted Event. - Hiểu được những thông tin cần có để follow theo event contract chung.
	14h30 - 15h	Tham gia meeting: Get aligned on the works need to be done to resolve the Interest issue.	- Biết được vấn đề đang gặp phải với tài khoản saving và tiền lời. - Đưa ra giải pháp để giải quyết vấn đề trên (chưa chốt phương án cuối cùng, cần thông tin thêm).
	15h - 16h	Tham gia meeting: Special TymeBank and GoTyme joint All Hands	
Thứ 3 04/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h	Thực hiện Task #8: [Confluence page] transaction recon database.	- Viết document cho Payment Recon Service. - Biết được các định dạng của từng trường dữ liệu.
	13h30 - 18h	Thực hiện Task #9: [Payment Recon] Create migration script payment recon.	- Viết script để khởi tạo các database schema cho Payment Recon Service. - Tạo các entity và repository cho từng bảng dữ liệu.
Thứ 4 05/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h30 - 11h30	Tham gia meeting: Engineering practices biweekly.	- Hiểu về cấu hình kết nối đến database thông qua các công RDS endpoint và RDS proxy. - Nắm được vì sao nên chuyển sang sử dụng RDS proxy thay vì thông qua endpoint.

	13h - 18h	Thực hiện Task #10: [Payment Recon] Validate and recon transaction base on check point.	<p>Tiến hành code phần validate cho transaction.</p> <ul style="list-style-type: none"> + Idea chung là sẽ tạo một lớp service để xử lý phần validate giữa 2 transaction có giống nhau hay không. + Sử dụng thêm một danh sách những event cần có để xác định xem quá trình validate đó đã kết thúc hay vẫn còn thiếu event cần đợi.
Thứ 5 06/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h, 13h30 - 18h	Tiếp tục thực hiện Task #10: [Payment Recon] Validate and recon transaction base on check point.	<ul style="list-style-type: none"> - Refactor lại code, tối ưu logic. - Suy nghĩ kĩ các trường hợp có thể xảy ra, qua đó thay đổi code một cách hợp lý. - Tạo lớp Recon Service để phục vụ cho việc chuẩn bị những tham số đầu vào cho Validate Service thực hiện kiểm tra. - Kết nối lớp Recon Service và lớp Validate Service.
Thứ 6 07/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h, 13h30 - 16h	Tiếp tục thực hiện Task #10: [Payment Recon] Validate and recon transaction base on check point.	<ul style="list-style-type: none"> - Viết Unit test cho các method của lớp Validate Service và lớp Recon Service. - Chỉnh sửa lại 2 lớp trên để tối ưu, tìm ra một flow cho việc chuẩn bị và validate diễn ra hợp lý. - Rút gọn code để dễ hiểu và rõ ràng.
	16h - 17h	Tham gia buổi All-Hands	<ul style="list-style-type: none"> - Biết được các thông tin về Business mới nhất. - Lắng nghe các Kudos của từng thành viên trao cho nhau. - Biết được về tình hình nhân sự.
Tuần 10 (từ 10/10/2022 đến 14/10/2022)			

Ngày		Nội dung thực tập	Kết quả đạt được
Thứ 2 10/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h, 13h30 - 18h	Tiếp tục thực hiện Task #10: [Payment Recon] Validate and recon transaction base on check point.	<ul style="list-style-type: none"> - Setup Cucumber vào dự án. - Tiến hành viết BDD test cho phần Validate và Recon Service.
Thứ 3 11/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	11h - 12h	Tham gia meeting: Discuss solution for the Interest synchronization issue.	Biết được giải pháp cho vấn đề với tiền lời của tài khoản Saving.
	14h - 16h	Tham gia meeting: Philippines sprint review.	<ul style="list-style-type: none"> - Biết được những gì mà các feature team đã hoàn thành trong sprint vừa qua. - Xem demo chức năng đã hoàn thành của các feature team.
	13h - 14h, 16h - 18h	Tiếp tục thực hiện Task #10: [Payment Recon] Validate and recon transaction base on check point.	<ul style="list-style-type: none"> - Tiến hành fix các lỗi và bug của BDD test. - Commit code và nhờ các thành viên trong team giúp chạy BDD test (do không được cung cấp AWS account để pull các docker image về nên không chạy được).
Thứ 4 12/10/2022	9h30 - 10h30	Tham gia meeting: Philippines Roadmap review biweek.	<ul style="list-style-type: none"> - Nắm bắt được những công việc đã hoàn thành trong sprint vừa qua của các feature team. - Biết được những công việc mà các team sắp tới phải hoàn thành.
	10h30 - 11h30	Tham gia meeting: Engineering practices biweekly.	Các feature team chia sẻ về cách họ monitor service của mình trên DataDog để trực quan hóa và theo dõi được sự hoạt động của các service.

	13h - 14h	Tham gia meeting: Sprint #40 Planning.	<ul style="list-style-type: none"> - Nắm bắt lại những công việc còn dang dở trong sprint 39 - Biết được những công việc sẽ làm cho sprint 40. - Phân chia nhiệm vụ cho từng người.
	17h - 18h	Tham gia meeting: TYMEX - TOWN HALL.	<ul style="list-style-type: none"> - Nắm được các hoạt động của toàn bộ Tyme Group. - Kế hoạch sắp tới của TymeX.
Thứ 5 06/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	10h - 12h, 16h - 18h	Thực hiện Task #11: [Payment Recon] Subscribe message from SNS to slack	<ul style="list-style-type: none"> - Tìm hiểu về Slack API, Slack Webhook. - Tạo app trên Slack Webhook và tiến hành test thử gửi tin nhắn đến slack channel. - Tiến hành code Slack Service phục vụ cho việc gửi tin nhắn đến Slack. - Viết Unit test cho phần trên.
	14h - 15h	Tham gia meeting: Batch Payment Intergration With Taliad Discussion	<ul style="list-style-type: none"> - Thảo luận về việc Taliad (chăm sóc khách hàng) sử dụng Batch Payment như thế nào. - Đưa ra giải pháp cho bên Taliad áp dụng.
	15h - 16h	Tham gia meeting: Catch up with HR: Ship IT presentation information.	<ul style="list-style-type: none"> - Nắm bắt tình hình của thực tập sinh trong thời gian vừa qua. - Nắm bắt được thông tin cần chuẩn bị cho bài present cuối khóa thực tập. - Biết được cách chấm điểm cho bài thuyết trình. - Biết được quy trình các vòng trong đợt tổng kết cuối khóa.
Thứ 6 07/10/2022	9h30 - 10h	Tham gia meeting hằng ngày để catch up với nhóm.	Biết được mọi người trong nhóm đã làm những công việc gì hôm qua và dự định hôm nay sẽ làm gì.
	14h30 - 15h30	Tham gia buổi sharing: [Friday Tech Talks] Getting Started with TymePipeline.	<ul style="list-style-type: none"> - Hiểu cách mà TymePipeline đang vận hành. - Cách để provision một service.

Bảng 1. Nhật ký công việc

CHƯƠNG 3. CÔNG VIỆC

3.1. Khởi tạo database cho profile

3.1.1. Mục tiêu công việc

Tạo migration script và entity cho Profile.

3.1.2. Đầu vào công việc

- Kiến thức về Java core.
- Kiến thức về Spring Boot.
- Kiến thức về Spring Data JPA.
- Kiến thức về thư viện Flyway.
- Kiến thức về SQL.

3.1.3. Nội dung công việc

- Viết câu lệnh SQL để tạo bảng Profile gồm các cột id, profile_id, username, first_name, middle_name, last_name, status, email, account_no, version.
- Sử dụng thư viện Flyway để tạo migration script.
- Sử dụng Spring Data JPA để tạo lớp thực thể Profile và cấu hình tương ứng.

3.1.4. Chi tiết cụ thể và phương pháp làm

Đầu tiên, cần viết câu lệnh SQL để tạo bảng Profile và các cột dữ liệu như mục 3.1.3 đã nêu.

```
1. CREATE TABLE `profile` (  
2.   `id` BIGINT(20) NOT NULL AUTO_INCREMENT PRIMARY KEY,  
3.   `profile_id` VARCHAR(50) NOT NULL,  
4.   `username` VARCHAR(50) NOT NULL,  
5.   `first_name` VARCHAR(50) NOT NULL,  
6.   `last_name` VARCHAR(50) NOT NULL,  
7.   `middle_name` VARCHAR(50) NULL,  
8.   `status` VARCHAR(50) NOT NULL,  
9.   `type` VARCHAR(50) NOT NULL,  
10.  `email` VARCHAR(255) NULL,  
11.  `account_no` VARCHAR(50) NOT NULL,  
12.  `version` BIGINT(50) NULL,  
13.  `created_date` TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP(3),  
14.  `modified_date` TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP(3),  
15.  UNIQUE INDEX `profile_id_UNIQUE` (`profile_id` ASC)  
16. ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE = utf8mb4_unicode_ci;
```

Code 1. Câu lệnh SQL tạo bảng Profile

Tiếp đến, sử dụng Flyway để thực thi migration script. Tạo file đặt tên là V3__add_profile_table.sql (quy ước đặt tên Flyway: [Version]__[Any_thing].sql) nằm trong thư mục resources/db/migration.

Sau cùng, tạo lớp thực thể Profile và cấu hình tên, kiểu dữ liệu, ... tương ứng với bảng Profile đã tạo.

```
1. @Data  
2. @Entity  
3. @Builder  
4. @Table(name = "profile")  
5. @EqualsAndHashCode(callSuper = false)
```

```

6. public class Profile extends Auditable {
7.
8.     @Id
9.     @GeneratedValue(strategy = GenerationType.IDENTITY)
10.    private Long id;
11.
12.    @Column(name = "profile_id", nullable = false)
13.    private String profileId;
14.
15.    @Column(name = "username")
16.    private String username;
17.
18.    @Column(name = "first_name")
19.    private String firstName;
20.
21.    @Column(name = "last_name")
22.    private String lastName;
23.
24.    @Column(name = "middle_name")
25.    private String middleName;
26.
27.    @Column(name = "status")
28.    @Enumerated(EnumType.STRING)
29.    private ProfileStatus status;
30.
31.    @Column(name = "type")
32.    @Enumerated(EnumType.STRING)
33.    private ProfileType type;
34.
35.    @Column(name = "email")
36.    private String email;
37.
38.    @Column(name = "account_no")
39.    private String accountNo;
40.
41.    @Column(name = "version")
42.    @Version
43.    private Long version;
44.
45. }

```

Code 2. Lớp entity Profile

3.1.5. Kiến thức chuyên ngành vận dụng

- Kiến thức về cơ sở lập trình.
- Kiến thức về lập trình hướng đối tượng.
- Kiến thức về cơ sở dữ liệu.

3.1.6. Kết quả có được

Thêm thành công migration script sử dụng thư viện Flyway và lớp thực thể Profile. Khi bắt đầu chạy ứng dụng, Flyway tự động sử dụng migration script kiểm tra và tạo mới bảng Profile cùng với các cột tương ứng trong database.

3.2. Tạo lớp service xử lý kiểm tra giao dịch

3.2.1. Mục tiêu công việc

Kiểm tra tính toàn vẹn dữ liệu của giao dịch trong quá trình xử lý và đi qua các microservices khác nhau.

3.2.2. Đầu vào công việc

- Kiến thức về Java core.
- Kiến thức về Spring Boot.
- Kiến thức về Spring Data JPA.
- Kiến thức về OOP.
- Kiến thức về cấu trúc dữ liệu và giải thuật.
- Kiến thức về Event-Driven Programming.

3.2.3. Nội dung công việc

Từng service sẽ xử lý yêu cầu giao dịch và bắn ra event chứa thông tin của giao dịch đó sau khi xử lý. Sau khi nhận được event, tiến hành kiểm tra và xác nhận sự toàn vẹn của dữ liệu. Dựa trên đó, tiến hành xây dựng các lớp để kiểm tra và xác nhận sự thay đổi dữ liệu của giao dịch và lưu xuống database.

Với những giao dịch toàn vẹn dữ liệu giữa tất cả các service, status của giao dịch đó sẽ được đánh là MATCHED. Ngược lại, status của giao dịch sẽ được đánh là MISMATCHED. Trong trường hợp khi chưa nhận đủ giao dịch của các service cần kiểm tra, status của giao dịch sẽ được đánh là PENDING.

3.2.4. Chi tiết cụ thể và phương pháp làm

Trước tiên, cần xây dựng một lớp service đặt tên là ValidationService với chức năng là kiểm tra và trả về kết quả kiểm tra gọi là ValidationResult (bao gồm tình trạng và lý do cho tình trạng đó).

```
1. @Service
2. @RequiredArgsConstructor
3. public class ValidationServiceImpl implements ValidationService {
4.
5.     @MethodLog(level = LogLevel.INFO)
6.     public ValidationResult validate(
7.         TransactionLoaded transactionLoadedOriginal, TransactionLoaded
8.         transactionLoadedGiven,
9.         List<MessageType> messageTypesRequired, List<MessageType>
10.         messageTypesOriginal) {
11.
12.         if (!compareMessageType(messageTypesOriginal, messageTypesRequired)) {
13.             return buildValidateResult(ReconStatus.MISMATCHED,
14.                 ValidateReason.EVENT_TYPE_NOT_MATCH);
15.         }
16.
17.         if (!compareTransactionLoaded(transactionLoadedGiven,
18.             transactionLoadedOriginal)) {
19.             return buildValidateResult(ReconStatus.MISMATCHED,
20.                 ValidateReason.FIELD_NOT_MATCH);
21.         }
22.
23.         if (messageTypesRequired.size() != messageTypesOriginal.size()) {
24.             return buildValidateResult(ReconStatus.PENDING,
25.                 ValidateReason.EVENT_TYPE_TOTAL_INVALID);
26.         }
27.
28.         return buildValidateResult(ReconStatus.MATCHED,
29.             ValidateReason.ALL_FIELD_MATCH);
30.     }
31. }
```



```

25. private boolean compareTransactionLoaded(TransactionLoaded
    transactionLoadedGiven,
26.     TransactionLoaded transactionLoadedOriginal) {
27.
28.     return transactionLoadedGiven.equals(transactionLoadedOriginal);
29. }
30.
31. private boolean compareMessageType(List<MessageType> messageTypesOriginal,
32.     List<MessageType> messageTypesRequired) {
33.
34.     return messageTypesOriginal.stream()
35.         .allMatch(messageType ->
    messageTypesRequired.stream().anyMatch(messageType::equals));
36. }
37.
38. private ValidateResult buildValidateResult(ReconStatus status, ValidateReason
    reason) {
39.
40.     return ValidateResult.builder()
41.         .reason(reason)
42.         .status(status)
43.         .build();
44. }
45. }

```

Code 3. Lớp kiểm tra giao dịch ValidationService

Tiếp theo, xây dựng một lớp xử lý đặt tên là ReconService, trong đó có gọi đến phương thức validate của lớp ValidationService để kiểm tra. Sau cùng, lưu kết quả xuống database.

```

1. @Override
2. @MethodLog(level = LogLevel.INFO)
3. public void process(TransactionLoaded transactionLoadedGiven) {
4.
5.     var transaction = transactionService.getTransaction(transactionLoadedGiven);
6.     var checkpoint =
    checkpointService.saveCheckpoint(buildCheckpoint(transactionLoadedGiven));
7.
8.     var messageTypesOriginal = checkpointService.getCheckpoints(
9.         transactionLoadedGiven.getTransactionId())
10.        .stream()
11.        .map(Checkpoint::getEventType)
12.        .collect(Collectors.toList());
13.
14.     var validateResult =
    validationService.validate(buildTransactionLoaded(transaction),
15.         transactionLoadedGiven, messageTypesRequired, messageTypesOriginal);
16.
17.     transaction.setStatus(validateResult.getStatus());
18.     transaction.setReason(validateResult.getReason().getMessage());
19.
20.     if (MessageType.TRANSACTION_POSTED == transactionLoadedGiven.getEventType()) {
21.         transaction.setBankingLedgerTxnId(transactionLoadedGiven.getBankingLedgerTxnI
    d());
22.     }
23.
24.     checkpoint.setIsMatched(!ReconStatus.MISMATCHED.equals(validateResult.getStatus
    ()));
25.
26.     transactionRepository.save(transaction);
27.     checkpointService.saveCheckpoint(checkpoint);
28. }

```

3.2.5. Kiến thức chuyên ngành vận dụng

- Kiến thức về cơ sở lập trình.
- Kiến thức về lập trình hướng đối tượng.
- Kiến thức về cơ sở dữ liệu.
- Kiến thức về cấu trúc dữ liệu và giải thuật.

3.2.6. Kết quả có được

Kiểm tra được sự toàn vẹn của giao dịch khi đi qua các microservices, xác định được những giao dịch nào gặp phải mất mát hoặc không đi qua hết các services khác từ đó có thể tìm cách giải quyết.

3.3. Viết unit test cho lớp kiểm tra và xử lý giao dịch

3.3.1. Mục tiêu công việc

Kiểm thử chức năng kiểm tra và xử lý giao dịch.

3.3.2. Đầu vào công việc

- Kiến thức về Java core.
- Kiến thức về Spring Boot.
- Kiến thức về unit test trong Spring Boot.
- Kiến thức về thư viện Mockito.

3.3.3. Nội dung công việc

Viết unit test cho lớp ValidateService.

3.3.4. Chi tiết cụ thể và phương pháp làm

Trước tiên, cần thêm thư viện Mockito vào trong file build.gradle để sử dụng trong việc viết test và cần mock data.

```
1. testImplementation 'org.mockito:mockito-inline:4.1.0'
```

Code 5. Thêm dependency Mockito vào dự án

Tiếp đến, tạo lớp unit test đặt tên ValidationServiceImplTest (theo quy ước). Tiến hành InjectMock service vào để thực hiện viết unit test.

```
1. @ExtendWith(MockitoExtension.class)
2. public class ValidationServiceImplTest {
3.
4.     @InjectMocks
5.     private ValidationServiceImpl validationService;
```

Code 6. Inject Mock cho ValidationService

Trường hợp #1: Giá trị ValidateResult trả về nên là MISMATCHED và lý do là có sự mất mát dữ liệu giao dịch.

```
1. @Test
```

```

2.    void
    shouldReturnStatusMismatchedAndReasonEventTypeNotMatch_whenValidateWithTwoDifferentEventTypes() {
3.
4.        var txnId = 113L;
5.        var orderLoadedOriginal = buildTransactionLoaded(txnId);
6.        var orderLoadedGiven = buildTransactionLoaded(txnId);
7.        var messageTypesRequired = List.of(MessageType.TRANSACTION_POSTED);
8.        var messageTypesOriginal = List.of(MessageType.PAYMENT_ORDER_AUTHORIZED);
9.
10.       var validateResult = validationService.validate(orderLoadedOriginal,
11.               orderLoadedGiven,
12.               messageTypesRequired, messageTypesOriginal);
13.       assertThat(validateResult.getStatus(), equalTo(ReconStatus.MISMATCHED));
14.       assertThat(validateResult.getReason(),
15.               equalTo(ValidateReason.EVENT_TYPE_NOT_MATCH));
15.    }

```

Code 7. Test case #1 ValidateResult trả về MISMATCHED và lý do có mất mát dữ liệu

Trường hợp #2: Giá trị ValidateResult trả về nên là MISMATCHED và lý do là có sự mất mát dữ liệu giao dịch.

```

1.    @Test
2.    void
    shouldReturnStatusMismatchedAndReasonFieldNotMatch_whenValidateWithTwoDifferentTransactionLoaded() {
3.
4.        var txnId = 113L;
5.        var orderLoadedOriginal = buildTransactionLoaded(txnId);
6.        var orderLoadedGiven = buildTransactionLoaded(txnId + 2L);
7.        var messageTypesRequired = List.of(MessageType.TRANSACTION_POSTED,
8.               MessageType.PAYMENT_ORDER_AUTHORIZED);
9.        var messageTypesOriginal = List.of(MessageType.PAYMENT_ORDER_AUTHORIZED,
10.               MessageType.TRANSACTION_POSTED);
11.
12.       var validateResult = validationService.validate(orderLoadedOriginal,
13.               orderLoadedGiven,
14.               messageTypesRequired, messageTypesOriginal);
15.       assertThat(validateResult.getStatus(), equalTo(ReconStatus.MISMATCHED));
16.       assertThat(validateResult.getReason(),
17.               equalTo(ValidateReason.FIELD_NOT_MATCH));
17.    }

```

Code 8. Test case #2 ValidateResult trả về MISMATCHED và lý do có mất mát dữ liệu

Trường hợp #3: Giá trị ValidateResult trả về nên là PENDING và lý do là chưa nhận đủ giao dịch từ các services.

```

1.    @Test
2.    void
    shouldReturnStatusPendingAndReasonEventTypeTotalInvalid_whenValidateWithMissingEventType() {
3.
4.        var txnId = 113L;
5.        var orderLoadedOriginal = buildTransactionLoaded(txnId);
6.        var orderLoadedGiven = buildTransactionLoaded(txnId);
7.        var messageTypesRequired = List.of(MessageType.TRANSACTION_POSTED,
8.               MessageType.PAYMENT_ORDER_AUTHORIZED);
9.        var messageTypesOriginal = List.of(MessageType.PAYMENT_ORDER_AUTHORIZED);
10.

```

```

11.     var validateResult = validationService.validate(orderLoadedOriginal,
12.         orderLoadedGiven,
13.         messageTypesRequired, messageTypesOriginal);
14.     assertThat(validateResult.getStatus(), equalTo(ReconStatus.PENDING));
15.     assertThat(validateResult.getReason(),
16.         equalTo(ValidateReason.EVENT_TYPE_TOTAL_INVALID));
17. }

```

Code 9. Test case #3 ValidateResult trả về PENDING và lý chưa nhận đủ giao dịch

Trường hợp #4: Giá trị ValidateResult trả về nên là MATCHED.

```

1.  @Test
2.  void
3.  shouldReturnStatusMatchedAndReasonAllFieldMatch_whenValidateEqualTransactionLoadedAndEnoughEventRequired() {
4.      var txnId = 113L;
5.      var orderLoadedOriginal = buildTransactionLoaded(txnId);
6.      var orderLoadedGiven = buildTransactionLoaded(txnId);
7.      var messageTypesRequired = List.of(MessageType.TRANSACTION_POSTED,
8.          MessageType.PAYMENT_ORDER_AUTHORIZED);
9.      var messageTypesOriginal = List.of(MessageType.PAYMENT_ORDER_AUTHORIZED,
10.          MessageType.TRANSACTION_POSTED);
11.
12.      var validateResult = validationService.validate(orderLoadedOriginal,
13.          orderLoadedGiven,
14.          messageTypesRequired, messageTypesOriginal);
15.      assertThat(validateResult.getStatus(), equalTo(ReconStatus.MATCHED));
16.      assertThat(validateResult.getReason(),
17.          equalTo(ValidateReason.ALL_FIELD_MATCH));
18. }

```

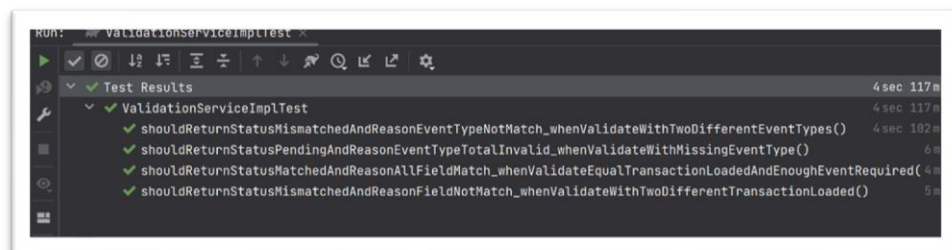
Code 10. Test case #4 ValidateResult trả về MATCHED

3.3.5. Kiến thức chuyên ngành vận dụng

- Kiến thức về cơ sở lập trình.
- Kiến thức về lập trình hướng đối tượng.
- Kiến thức về đảm bảo và kiểm soát chất lượng phần mềm.

3.3.6. Kết quả có được

Viết thành công unit test cho chức năng kiểm tra sự toàn vẹn dữ liệu của giao dịch. Chạy unit test và được kết quả như sau:



Hình 1. Chạy unit test cho ValidationService

3.4. Phần mở rộng

3.4.1. Về công việc

Tạo cơ sở dữ liệu đơn giản chứa danh sách quyền sách. Sử dụng để tạo một API cho phép truy cập dữ liệu với từ khoá nhập vào và tạo một trang web đơn giản (trang tìm kiếm và hiển thị sản phẩm) sử dụng API đã tạo.

3.4.2. Các công nghệ sử dụng

3.4.2.1. Cơ sở dữ liệu

Sử dụng hệ quản trị cơ sở dữ liệu mã nguồn mở MySQL.

Phiên bản sử dụng: MySQL Server 8.0.30.

3.4.2.2. API

Sử dụng Spring Boot (phiên bản 2.7.3) để xây dựng API cho phép tìm kiếm và hiển thị sách.

Spring Boot là phiên bản mở rộng của Spring Framework, giúp cho việc phát triển ứng dụng trở nên đơn giản hơn khi Spring Boot hỗ trợ việc tự động hóa các cấu hình các phụ thuộc hay các thư viện thứ ba.

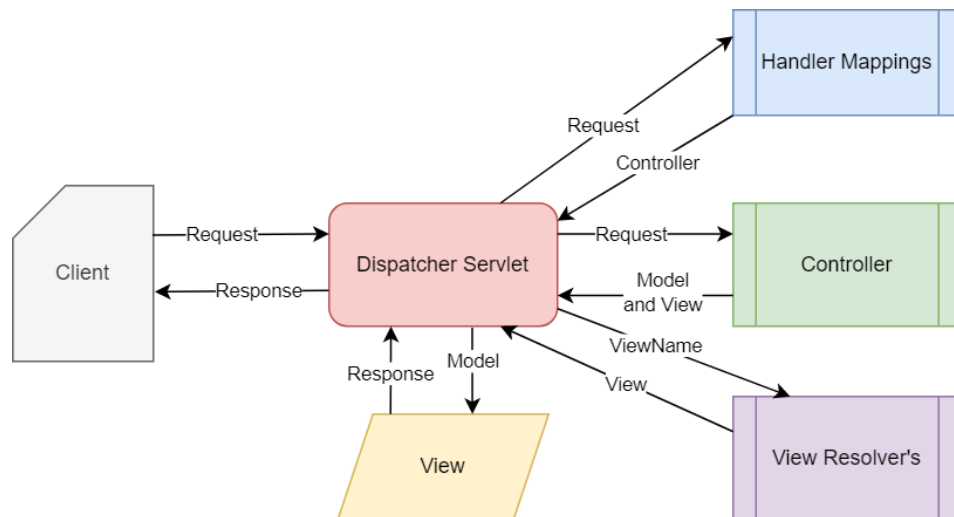
Spring Boot giúp các nhà phát triển tạo các ứng dụng chỉ cần chạy. Cụ thể, nó cho phép bạn tạo các ứng dụng độc lập tự chạy mà không cần phụ thuộc vào máy chủ web bên ngoài, bằng cách nhúng máy chủ web như Tomcat hoặc Netty vào ứng dụng của bạn trong quá trình khởi tạo. Do đó, có thể khởi chạy ứng dụng của mình trên bất kỳ nền tảng nào bằng cách nhấn lệnh Run.

3.4.2.3. Front-end

Sử dụng Spring MVC để xây dựng trang web cho phép tìm kiếm và hiển thị sách.

Spring MVC là một Java framework được sử dụng để xây dựng các ứng dụng web. Nó tuân theo mô hình thiết kế Model-View-Controller và các component có thể được sử dụng để phát triển các ứng dụng web linh hoạt, giảm thiểu sự phụ thuộc. Mô hình MVC dẫn đến việc phân tách các khía cạnh khác nhau của ứng dụng (logic đầu vào, logic nghiệp vụ và logic giao diện người dùng), đồng thời hỗ trợ giảm thiểu sự phụ thuộc giữa các phần tử này.

Sơ đồ luồng đi của Spring MVC:



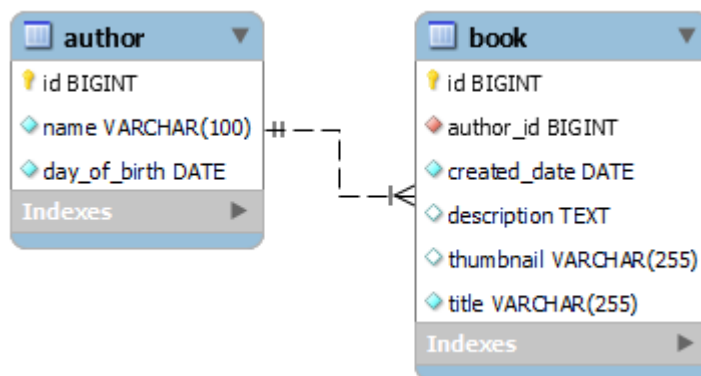
Hình 2. Sơ đồ luồng đi của Spring MVC.

3.4.3. Xây dựng API

3.4.3.1. Tầng Repository

Xây dựng cơ sở dữ liệu gồm 2 bảng *book* và *author*. Trong đó, mối quan hệ giữa hai bảng là mối quan hệ *1 – nhiều* với ràng buộc khóa ngoại là *author_id* trong bảng *book* ứng với *id* trong bảng *author*:

- Một cuốn sách chỉ thuộc về một tác giả.
- Một tác giả có thể có nhiều cuốn sách.



Hình 3. Database schema

Cấu hình kết nối database qua file *application.yml* nằm trong thư mục *resources* với url là connector đến database *book_db*.

```

1. spring:
2.   jpa:
3.     database: MYSQL
4.     show-sql: true
5.   datasource:
6.     driver-class-name: com.mysql.cj.jdbc.Driver
7.     url: jdbc:mysql://localhost:3306/book_db
8.     username: root
9.     password: root
  
```

Sử dụng thư viện Flyway và Spring JPA để tiến hành khởi tạo các thực thể cũng như migration database cho dự án. Để sử dụng, ta cần tiến hành thêm các dependencies vào file pom.xml (maven).

```
1. <!-- Spring JPA -->
2. <dependency>
3.     <groupId>org.springframework.boot</groupId>
4.     <artifactId>spring-boot-starter-data-jpa</artifactId>
5. </dependency>
6.
7. <!-- MySQL Connector -->
8. <dependency>
9.     <groupId>mysql</groupId>
10.    <artifactId>mysql-connector-java</artifactId>
11. </dependency>
12.
13. <!-- Flyway -->
14. <dependency>
15.     <groupId>org.flywaydb</groupId>
16.     <artifactId>flyway-mysql</artifactId>
17. </dependency>
```

Tạo migration script vào trong thư mục resources.db.migration đặt tên là *V1__Init_DB.sql* (tuân theo quy tắc đặt tên của Flyway: [Version]__[Any_thing].sql) với nội dung:

```
1. CREATE TABLE IF NOT EXISTS `author` (
2.     `id` BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
3.     `name` VARCHAR(100) NOT NULL,
4.     `day_of_birth` DATE NOT NULL
5. ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 collate=utf8mb4_unicode_ci;
6.
7. CREATE TABLE IF NOT EXISTS `book` (
8.     `id` BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
9.     `author_id` BIGINT NOT NULL,
10.    `created_date` DATE NOT NULL,
11.    `description` TEXT NULL,
12.    `thumbnail` VARCHAR(255) NULL,
13.    `title` VARCHAR(255) NOT NULL,
14.    CONSTRAINT FK_BookAuthor FOREIGN KEY (`author_id`)
15.    REFERENCES author(`id`)
16. ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 collate=utf8mb4_unicode_ci;
```

Tiếp đến, tiến hành tạo các entity Book và Author đặt trong package repository.entity và cấu hình phù hợp với migration script đã tạo ở trên.

```
1. @Data
2. @Entity
3. @Table(name = "author")
4. public class Author {
5.
6.     @Id
7.     @Column(name = "id")
8.     @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```

9.     private Long id;
10.
11.     @Column(name = "name")
12.     private String name;
13.
14.     @Column(name = "day_of_birth")
15.     private LocalDate dayOfBirth;
16.
17.     @OneToMany(cascade = CascadeType.ALL, mappedBy = "author")
18.     private List<Book> employees;
19. }

```

Code 14. Entity Author tương ứng với bảng Author trong database

Cấu hình quan hệ một – nhiều cho bảng Author sử dụng annotation `@OneToMany` và truyền vào tham số `mappedBy` tương ứng với tên trường dữ liệu trong bảng Book dưới đây.

```

1. @Data
2. @Entity
3. @Table(name = "book")
4. public class Book {
5.
6.     @Column(name = "id")
7.     @GeneratedValue(strategy = GenerationType.IDENTITY)
8.     private @Id Long id;
9.
10.    @Column(name = "title")
11.    private String title;
12.
13.    @Column(name = "description")
14.    private String desc;
15.
16.    @Column(name = "thumbnail")
17.    private String thumb;
18.
19.    @Column(name = "created_date")
20.    private LocalDate createdAt;
21.
22.    @ManyToOne(cascade = CascadeType.ALL)
23.    @JoinColumn(name = "author_id")
24.    private Author author;
25. }

```

Code 15. Entity Book tương ứng với bảng Book trong database

Tiến hành tạo một interface *BookRepository* nằm trong package repository kế thừa từ *JpaRepository* để sử dụng các phương thức tương tác đến database.

```

1. @Repository
2. public interface BookRepository extends JpaRepository<Book, Long> {
3.     @Query(value = "SELECT book.*, author.name " +
4.         "FROM book " +
5.         "JOIN author ON book.author_id = author.id " +
6.         "WHERE book.title LIKE %:keyword% " +
7.         "OR author.name LIKE %:keyword%", nativeQuery = true)
8.     List<Book> findAllByKeyword(String keyword);
9. }

```

Code 16. Interface BookRepository chứa các phương thức tương tác đến database

Trong đó, khai báo phương thức *findAllByKeyword* với tham số truyền vào là *keyword* – tương ứng với từ khóa cần tìm.

Mặc định, Spring JPA đã cung cấp sẵn các phương thức CRUD cơ bản, để có thể tự viết truy vấn thuần trong Spring JPA, ta sử dụng annotation *@Query* với tham số *value* truyền vào tương ứng là câu truy vấn cần thực thi và tham số *nativeQuery* được set bằng true.

3.4.3.2. Tầng Service

Tạo một lớp service đặt tên là *BookService* nằm trong package *service* để chứa các logic xử lý dữ liệu. Tầng service sẽ kết nối đến database thông qua tầng repository đã được tạo trước đó (*BookRepository*).

```
1. @Service
2. @RequiredArgsConstructor
3. public class BookService {
4.
5.     private final BookRepository bookRepository;
6.
7.     public List<BookResponse> getAll() {
8.         return bookRepository.findAll()
9.             .stream()
10.            .map(BookResponse::fromEntity)
11.            .collect(Collectors.toList());
12.    }
13.
14.    public List<BookResponse> getAllByKeyword(String keyword) {
15.        return bookRepository.findAllByKeyword(keyword)
16.            .stream()
17.            .map(BookResponse::fromEntity)
18.            .collect(Collectors.toList());
19.    }
20.
21.    public BookResponse getBook(Long id) {
22.        var optBook = bookRepository.findById(id);
23.        return optBook.map(BookResponse::fromEntity).orElse(null);
24.    }
25. }
```

Code 17. Lớp *BookService* dùng để xử lý logic

Trong đó:

- Trường *bookRepository*: Là trường interface repository dùng để kết nối đến bảng *book* trong database. Ở đây, sử dụng annotation *@RequiredArgsConstructor* để tự động tạo ra các hàm khởi tạo chứa tất cả các trường trong lớp này và cụ thể ở đây là *BookRepository* để phục vụ cho việc nạp vào sự phụ thuộc.
- Phương thức *getAll*: Dùng để lấy ra tất cả các quyển sách hiện có và trả về một danh sách cuốn sách đã được chuyển đổi sang DTO.
- Phương thức *getAllByKeyword*: Tương tự như phương thức *getAll* nhưng ở đây cho phép tìm kiếm bằng từ khóa thông qua tham số truyền vào *keyword*.
- Phương thức *getBook*: Dùng để lấy ra quyển sách có id bằng với tham số truyền vào *id*, nếu không tìm thấy giá trị trả về sẽ là *null*.

3.4.3.3. Tầng Controller

Sử dụng port 8080 bằng cách cấu hình trong file application.yml nằm trong thư mục resources.

```
1. server:
2.   port: 8080
```

Code 18. Cấu hình port cho API

Để giao tiếp giữa Client với Controller, Controller với Service, ta sử dụng đến DTO – là các class đóng gói data, mục đích để giảm tải và kiểm soát được dữ liệu trả về và truyền đi, giúp tăng cường bảo mật (vì không ai biết thực tế có những trường nào trong entity).

Tiến hành tạo lớp *BookResponse* nằm trong package dto.

```
1. @Data
2. @Builder
3. public class BookResponse {
4.
5.     private Long id;
6.     private String title;
7.     private String desc;
8.     private String thumb;
9.     private LocalDate createdAt;
10.    private String author;
11.
12.    public static BookResponse fromEntity(Book book) {
13.        return BookResponse.builder()
14.            .id(book.getId())
15.            .title(book.getTitle())
16.            .desc(book.getDesc())
17.            .thumb(book.getThumb())
18.            .createdAt(book.getCreatedAt())
19.            .author(book.getAuthor().getName())
20.            .build();
21.    }
22. }
```

Code 19. Lớp DTO BookResponse

Trong đó, BookResponse chỉ chứa những dữ liệu cơ bản của sách như id, tiêu đề, mô tả, ảnh bìa, ngày đăng và tên tác giả. Ngoài ra, trong lớp BookResponse còn chứa phương thức fromEntity, phương thức này giúp chuyển đổi từ entity sang DTO một cách nhanh chóng.

Tạo lớp *BookController* nằm trong package controller chứa những request mapping và kết nối đến tầng service để trả về data qua DTO cho client.

```
1. @RestController
2. @RequestMapping(path = "/api/v1/books")
3. @RequiredArgsConstructor
4. public class BookController {
5.
6.     private final BookService bookService;
7.
8.     @GetMapping
```

```

9.     @ResponseStatus(HttpStatus.OK)
10.    public List<BookResponse> getAll() {
11.        return bookService.getAll();
12.    }
13.
14.    @GetMapping(path = "search")
15.    @ResponseStatus(HttpStatus.OK)
16.    public List<BookResponse> getAll(@RequestParam(name = "q") String keyword) {
17.        return bookService.getAllByKeyword(keyword);
18.    }
19.
20.    @GetMapping(path = "{bookId}")
21.    @ResponseStatus(HttpStatus.OK)
22.    public BookResponse getBookById(@PathVariable Long bookId) {
23.        return bookService.getBook(bookId);
24.    }
25. }

```

Code 20. Lớp controller BookController

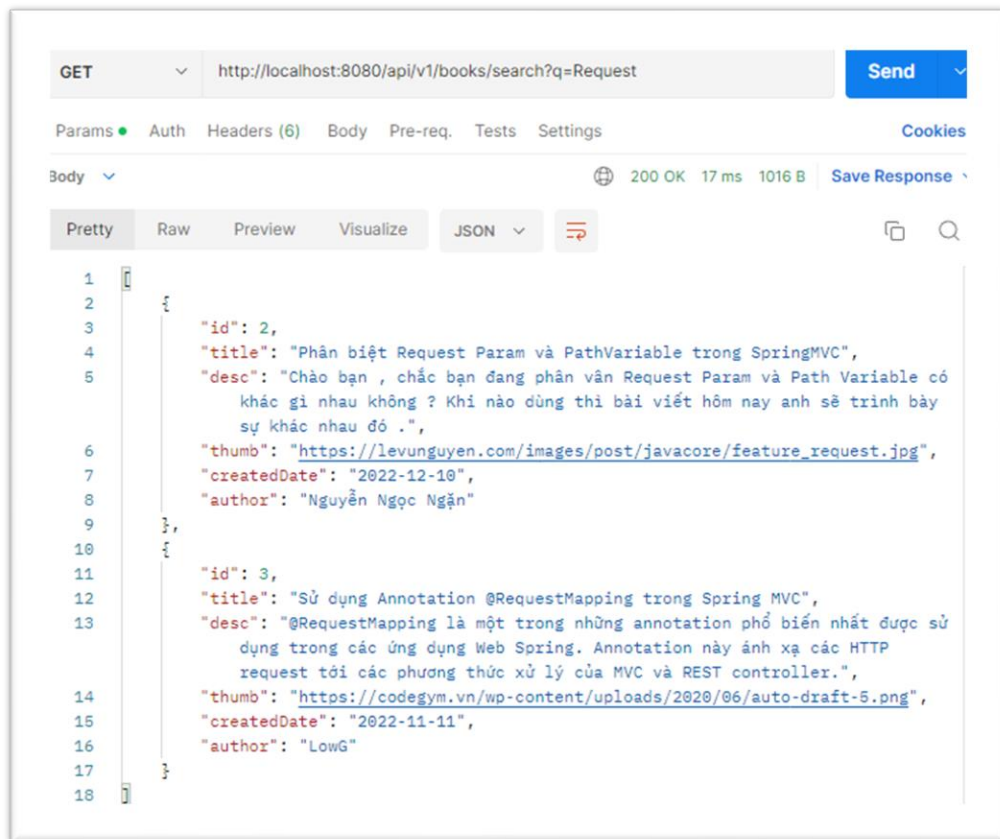
Trong đó:

- Annotation *@RestController*: Là sự kết hợp giữa bean component *@Controller* và annotation *@ResponseBody*, cho phép mapping request và trả về thẳng dữ liệu cho client mà không cần phải thông qua xử lý view để sinh ra HTML (Spring MVC).
- Annotation *@RequestMapping*: Là annotation sử dụng cho mục đích mapping request từ client đến những phương thức trong controller. Tham số path truyền vào với mục đích định nghĩa đường dẫn URL đến controller.
- Annotation *@ResponseCode*: Để chỉ định những response code có thể sẽ trả về cho client.
- Annotation *@RequestParam*: Lấy giá trị nhập vào thông qua tham số ở đường dẫn URL theo định dạng URL?name=value.
- Annotation *@PathVariable*: Lấy giá trị truyền vào thông qua đường dẫn URL theo định dạng URL/value.
- Trường BookService: Chứa những logic xử lý đã được đề cập ở phần 3.2.3.
- Các phương thức gọi đến service để xử lý, nhận và trả về data qua DTO cho client.

3.4.3.4. Sử dụng Postman kiểm thử

Sử dụng Postman để kiểm tra API, tiến hành mở Postman và tạo một GET request với đường dẫn <http://localhost:8080/api/v1/books> và thực hiện gửi request.

Kết quả nhận được:



Hình 4. Sử dụng Postman kiểm tra API

3.4.4. Xây dựng trang Web

3.4.4.1. Xây dựng model

Tạo lớp *BookResponse* – chứa những thông tin trả về khi gọi đến API đặt trong package model.

```
1. @Getter
2. @Setter
3. @NoArgsConstructor
4. @AllArgsConstructor
5. public class BookResponse {
6.
7.     private Long id;
8.     private String title;
9.     private String desc;
10.    private String thumb;
11.    private LocalDate createdDate;
12.    private String author;
13. }
```

Code 21. Lớp model *BookResponse*

3.4.4.2. Xây dựng view

Để xây dựng phần view (trong mô hình MVC), sử dụng Thymeleaf – một server-side Java template engine dành cho web hoặc những môi trường độc lập. Mục tiêu chính của Thymeleaf là cung cấp các template thân thiện với HTML, có thể được

hiển thị chính xác trong các trình duyệt và cũng hoạt động như các nguyên mẫu tĩnh, cho phép cộng tác mạnh mẽ hơn trong các nhóm phát triển.

Xây dựng trang hiển thị các quyển sách bằng kết quả tìm kiếm là từ khóa nhập vào, đặt tên file là index.html và đặt vào thư mục resources/templates (Spring MVC tự động quét những file nằm trong thư mục này và nhận diện là những view cho trang web).

```
1. <!DOCTYPE html>
2. <html lang="en" xmlns:th="http://www.thymeleaf.org">
3. <head>
4.     <title>Book Management System</title>
5.     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
6.     <link th:href="@{/css/bootstrap.min.css}" rel="stylesheet"/>
7. </head>
8. <body>
9.     <div class="container">
10.         <div class="row alert alert-info justify-content-between"
11.             th:fragment="navbar">
12.             <div class="col-auto">
13.                 <a class="btn btn-primary" th:href="@{/}">Back to home</a>
14.             </div>
15.             <form class="d-flex col-auto" th:action="@{/search}"
16.                 method="GET">
17.                 <input name="q" th:value="${keyword}" type="text"
18.                     class="form-control mr-2"
19.                     placeholder="Wanna search anything?">
20.                 <button class="btn btn-success">Search</button>
21.             </form>
22.         </div>
23.         <div class="row">
24.             <div class="col-12 alert alert-danger"
25.                 th:if="${books.isEmpty()}">There is no book for now,
26.                 <a th:href="@{/}">back to home</a></div>
27.             <table class="table table-striped table-bordered"
28.                 th:if="${!books.isEmpty()}">
29.                 <thead>
30.                     <tr>
31.                         <th scope="col">#</th>
32.                         <th scope="col">Author</th>
33.                         <th scope="col">Title</th>
34.                         <th scope="col">Description</th>
35.                         <th scope="col">Publish date</th>
36.                     </tr>
37.                 </thead>
38.                 <tbody>
39.                     <tr th:each="book : ${books}">
40.                         <th scope="row" th:text="${book.getId()}"></th>
41.                         <td th:text="${book.getAuthor()}"></td>
42.                         <td><a th:text="${book.getTitle()}"
43.                             th:href="@{/detail/{id}(id=${book.getId()})}"></a></td>
44.                         <td th:text="${book.getDesc()}"></td>
45.                         <td th:text="${book.getCreatedDate()}"></td>
46.                     </tr>
47.                 </tbody>
48.             </table>
49.         </div>
50.     </div>
51.
52. <script th:src="@{/js/bootstrap.js}"></script>
53. </body>
54. </html>
```

Code 22. Trang view hiển thị kết quả tìm kiếm

Xây dựng trang hiển thị chi tiết cuốn sách, đặt tên file là detail.html và đặt vào trong thư mục resources/templates.

```
1. <!DOCTYPE html>
2. <html lang="en" xmlns:th="http://www.thymeleaf.org">
3. <head>
4.     <title th:text="${book.title}"></title>
5.     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
6.     <link th:href="@{/css/bootstrap.min.css}" rel="stylesheet"/>
7. </head>
8. <body>
9.     <div class="container">
10.         <div th:replace="index.html::navbar"></div>
11.         <div class="row">
12.             <div class="col-12 card">
13.                 
14.                 <div class="card-body">
15.                     <h5 class="card-title" th:text="${book.getTitle()}"></h5>
16.                     <p class="card-text" th:text="${book.getAuthor()}"></p>
17.                     <hr/>
18.                     <p class="card-text" th:text="${book.getDesc()}"></p>
19.                 </div>
20.             </div>
21.         </div>
22. </div>
23. <script th:src="@{/js/bootstrap.js}"></script>
24. </body>
25. </html>
```

Code 23. Trang view hiển thị chi tiết cuốn sách

3.4.4.3. Xây dựng controller

Đối với dự án web Spring MVC, sử dụng port 8099 bằng cách cấu hình trong file application.yml nằm trong thư mục resources.

```
1. server:
2.     port: 8099
```

Code 24. Cấu hình port cho web Spring MVC

Để tạo các kết nối gọi đến API đã tạo ở mục 3.4.3, sử dụng thư viện Spring Cloud OpenFeign – sử dụng interface và các annotation để định nghĩa các phương thức request đến API.

Tiến hành thêm dependency cho Spring Cloud OpenFeign bằng cách thêm vào file pom.xml.

```
1. <dependency>
2.     <groupId>org.springframework.cloud</groupId>
3.     <artifactId>spring-cloud-starter-openfeign</artifactId>
4. </dependency>
```

Code 25. Thêm dependency Spring Cloud OpenFeign

Tiếp đến, cấu hình OpenFeign thông qua file application.yml nằm trong thư mục resources và khai báo url đến API (ở đây API chạy trên localhost).

```

1. feign:
2.   okhttp:
3.     enabled: true
4.   client:
5.     config:
6.       book-api:
7.         url: http://localhost:8080

```

Code 26. Cấu hình Spring Cloud OpenFeign

Tạo một cổng kết nối sử dụng OpenFeign, đặt tên là *BookApiConnector* nằm trong package connector và khai báo các request mapping tương ứng trong API.

```

1. @FeignClient(value = "book-api", url = "${feign.client.config.book-api.url}",
2.   path = "api/v1/books")
3. public interface BookApiConnector {
4.   @GetMapping
5.   List<BookResponse> getAllBooks();
6.
7.   @GetMapping(path = "search")
8.   List<BookResponse> getAllBooks(@RequestParam(name = "q") String keyword);
9.
10.  @GetMapping(path = "{bookId}")
11.  BookResponse getBook(@PathVariable Long bookId);
12. }

```

Code 27. Lớp feign BookApiConnector

Tạo lớp controller cho trang web và đặt tên là *BookController* nằm trong package controller. Chứa các request mapping để xử lý truy cập đến đường dẫn các trang và xử lý trả về view cho từng request.

```

1. @Controller
2. @Validated
3. @RequestMapping
4. @RequiredArgsConstructor
5. public class BookController {
6.
7.   private final BookApiConnector bookApiConnector;
8.
9.   @GetMapping
10.  public String viewIndex(Model model) {
11.    var books = bookApiConnector.getAllBooks();
12.    model.addAttribute("books", books);
13.    return "index";
14.  }
15.
16.  @GetMapping(path = "/search")
17.  public String viewSearch(Model model,
18.    @RequestParam(name = "q") String keyword) {
19.    var books = bookApiConnector.getAllBooks(keyword);
20.    model.addAttribute("books", books);
21.    model.addAttribute("keyword", keyword);
22.    return "index";
23.  }
24.
25.  @GetMapping(path = "/detail/{bookId}")
26.  public String viewDetail(Model model, @PathVariable Long bookId) {
27.    var book = bookApiConnector.getBook(bookId);
28.
29.    if (Objects.isNull(book)) {

```

```

30.     return "redirect:/404";
31. }
32.
33.     model.addAttribute("book", book);
34.     return "detail";
35. }
36.
37. @GetMapping(path = "/404")
38. public String viewError() {
39.     return "404";
40. }
41. }

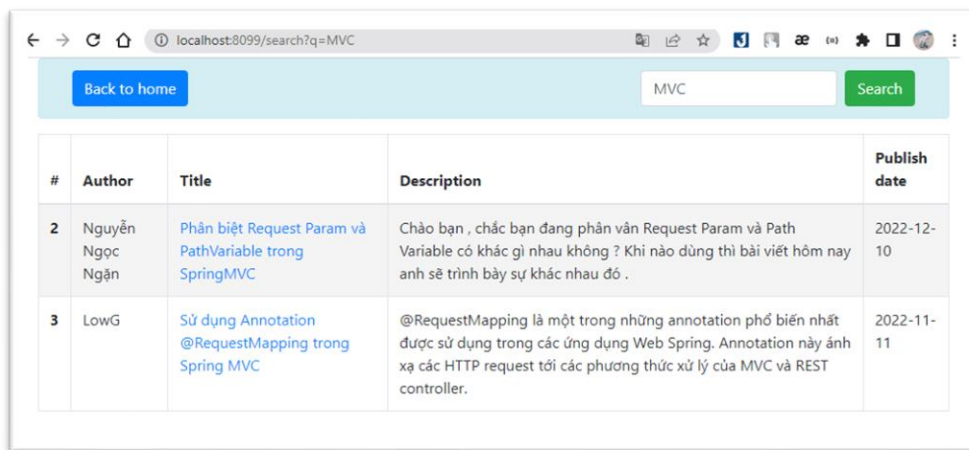
```

Code 28. Lớp BookController xử lý request mapping và trả về view

3.4.5. Kết quả đạt được

Sau khi xây dựng API và trang web sử dụng Spring Boot và Spring MVC, tiến hành khởi chạy cả hai dự án để kiểm tra kết quả đạt được.

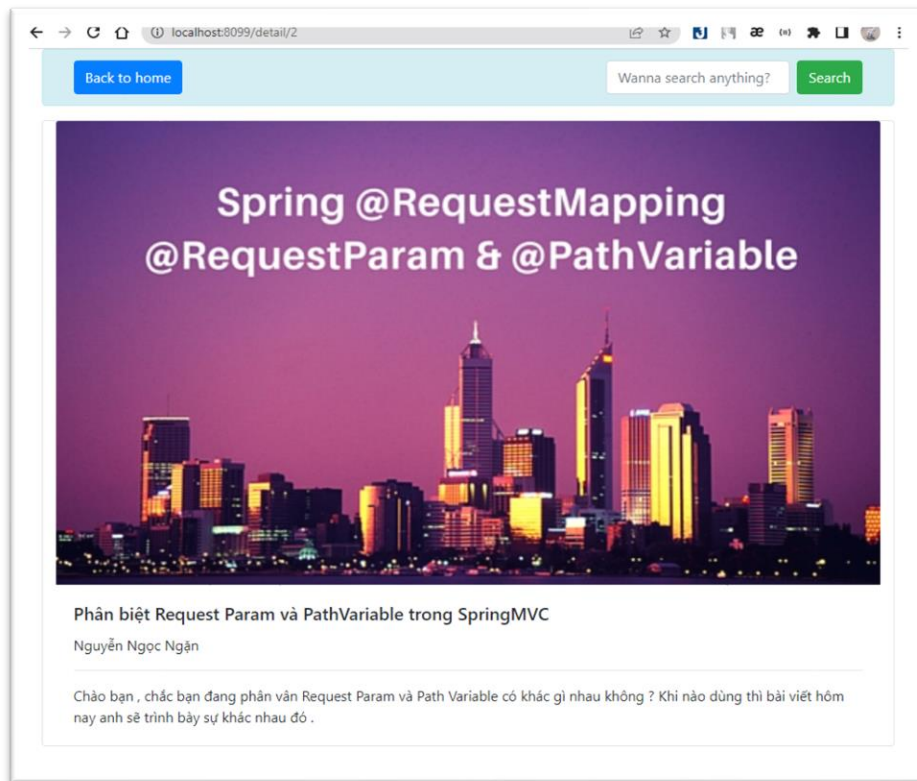
Đối với trang tìm kiếm bằng từ khóa, thử tìm kiếm với từ khóa “MVC” và kết quả trả về tương ứng là những cuốn sách có title chứa từ khóa.



#	Author	Title	Description	Publish date
2	Nguyễn Ngọc Ngạn	Phân biệt Request Param và PathVariable trong SpringMVC	Chào bạn , chắc bạn đang phân vân Request Param và Path Variable có khác gì nhau không ? Khi nào dùng thì bài viết hôm nay anh sẽ trình bày sự khác nhau đó .	2022-12-10
3	LowG	Sử dụng Annotation @RequestMapping trong Spring MVC	@RequestMapping là một trong những annotation phổ biến nhất được sử dụng trong các ứng dụng Web Spring. Annotation này ánh xạ các HTTP request tới các phương thức xử lý của MVC và REST controller.	2022-11-11

Hình 5. Trang tìm kiếm và trả về kết quả theo từ khóa nhập vào

Đối với trang hiển thị chi tiết cuốn sách, thử truy cập trang nội dung chi tiết của cuốn sách có id là 2 và title là “Phân biệt Request Param và PathVariable trong SpringMVC”.



Hình 6. Trang hiển thị chi tiết cuốn sách

CHƯƠNG 4. KẾT LUẬN VÀ ĐÁNH GIÁ