KHOA KỸ THUẬT VÀ CÔNG NGHỆ

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO KẾT THÚC MÔN

CÔNG NGHỆ PHẦN MỀM

ĐỀ TÀI:

PHÁT TRIỂN ỨNG DỤNG BÁN HÀNG TRỰC TUYẾN DỰA TRÊN NODEJS THEO PHƯƠNG PHÁP AGILE

Giáo viên hướng dẫn:

TS. Nguyễn Bảo Ân

Sinh viên thực hiện:

Họ tên: Nguyễn Hoàng Khang

MSSV: 110121035

Lớp: DA21TTB

Họ tên: Dương Thành Tân

MSSV: 110121097

Lớp: DA21TTB

Trà Vinh, tháng 06 năm 2024

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày ..... tháng …… năm ……

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày ..... tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trước hết, nhóm chúng em xin phép được bày tỏ lòng biết ơn sâu sắc nhất đến thầy Nguyễn Bảo Ân, người đã hỗ trợ và giúp đỡ nhóm chúng em trong suốt quá trình nghiên cứu và hoàn thiện bài báo cáo "Phát Triển Ứng Dụng Bán Hàng Trực Tuyến Dựa Trên Nodejs Theo Phương Pháp Agile". Sự hướng dẫn tận tình và những kiến thức quý báu mà thầy đã chia sẻ không chỉ giúp em hoàn thành báo cáo này mà còn là nguồn cảm hứng và động viên lớn lao cho em trong học tập và nghiên cứu.

Với những bài giảng của thầy về các phương pháp phát triển phần mềm linh hoạt, quản lý dự án Agile, và kỹ thuật lập trình hiện đại. Những kiến thức này không chỉ cung cấp cho chúng em nền tảng lý thuyết vững chắc mà còn là công cụ thực tiễn giúp chúng em phân tích và giải quyết các vấn đề phức tạp trong dự án. Sự kết hợp của các kỹ thuật quản lý dự án Agile, cùng với việc áp dụng các phương pháp phát triển phần mềm linh hoạt và kỹ thuật lập trình tiên tiến, đã mở ra những cánh cửa mới trong việc tiếp cận và xử lý các yêu cầu phức tạp, cho phép nhóm chúng em đạt được những kết quả đáng khích lệ trong quá trình phát triển ứng dụng của mình.

Đồng thời, nhóm chúng em nhận thức được rằng báo cáo của mình vẫn còn nhiều hạn chế và thiếu sót. Nên, nhóm chúng em mong muốn nhận được sự đóng góp ý kiến quý báu từ thầy, để em có thể tiếp tục học hỏi, nâng cao kỹ năng và kiến thức của mình. Nhóm chúng em tin rằng, với sự hỗ trợ và chỉ dẫn của thầy, nhóm sẽ hoàn thiện hơn trong những công trình nghiên cứu sắp tới.

Nhóm em xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

MỞ ĐẦU

1. Lý do chọn đề tài

Trong bối cảnh thương mại điện tử ngày càng phát triển, việc xây dựng các ứng dụng bán hàng trực tuyến hiệu quả và linh hoạt là rất quan trọng. NodeJS, với kiến trúc hướng sự kiện và khả năng xử lý bất đồng bộ, đã nổi lên như một nền tảng mạnh mẽ để xây dựng các ứng dụng web thời gian thực và có khả năng mở rộng cao. Đồng thời, phương pháp Agile, với sự tập trung vào sự linh hoạt, cộng tác và phản hồi liên tục, đã chứng minh tính hiệu quả trong việc quản lý các dự án phần mềm phức tạp.

Đề tài "Phát triển ứng dụng bán hàng trực tuyến dựa trên NodeJS theo phương pháp Agile" được lựa chọn nhằm mục đích kết hợp sức mạnh của NodeJS và phương pháp Agile để xây dựng một ứng dụng bán hàng trực tuyến đáp ứng được các yêu cầu của thị trường hiện đại. Đề tài này không chỉ mang tính thực tiễn cao mà còn có giá trị nghiên cứu, góp phần vào sự phát triển của lĩnh vực thương mại điện tử và công nghệ phần mềm.

2. Mục tiêu nghiên cứu

- **Nghiên cứu và áp dụng NodeJS:** Tìm hiểu kiến trúc, tính năng và các thư viện của NodeJS để xây dựng một ứng dụng web thương mại điện tử hiệu quả.
- **Áp dụng phương pháp Agile:** Sử dụng các nguyên tắc và thực hành của Agile, đặc biệt là khung làm việc Scrum, để quản lý dự án và đảm bảo tiến độ cũng như chất lượng sản phẩm.
- **Xây dựng ứng dụng bán hàng trực tuyến:** Phát triển một ứng dụng web hoàn chỉnh, bao gồm các chức năng quản lý sản phẩm, hiển thị sản phẩm, tìm kiếm, đặt hàng và thanh toán.
- **Đánh giá hiệu quả:** Đánh giá hiệu suất, tính khả dụng, bảo mật và khả năng mở rộng của ứng dụng.

3. Phương pháp nghiên cứu

- **Nghiên cứu lý thuyết:** Tìm hiểu các kiến thức liên quan đến NodeJS, ExpressJS, MongoDB, Agile và Scrum thông qua các tài liệu, sách, bài báo và các nguồn tài liệu khác.
- **Phát triển ứng dụng:** Áp dụng các kiến thức đã học để xây dựng ứng dụng bán hàng trực tuyến, sử dụng các công cụ và thư viện phù hợp.

- **Thực nghiệm và đánh giá:** Tiến hành các thử nghiệm để đánh giá hiệu suất, tính khả dụng, bảo mật và khả năng mở rộng của ứng dụng.

4.Đối tượng nghiên cứu

- **NodeJS:** Một nền tảng để xây dựng các ứng dụng mạng và máy chủ nhanh chóng và có khả năng mở rộng.
- **ExpressJS:** Một framework web cho NodeJS, giúp đơn giản hóa việc xây dựng các ứng dụng web và API.
- **MongoDB:** Một hệ quản trị cơ sở dữ liệu NoSQL linh hoạt và có khả năng mở rộng cao.
- **Agile:** Một phương pháp phát triển phần mềm linh hoạt, tập trung vào sự cộng tác, phản hồi và thích ứng với thay đổi.
- **Scrum:** Một khung làm việc Agile phổ biến, được sử dụng để quản lý các dự án phức tạp.

5. Phạm vi nghiên cứu

Đề tài tập trung vào việc xây dựng một ứng dụng bán hàng trực tuyến với các chức năng cơ bản như quản lý sản phẩm, hiển thị sản phẩm, tìm kiếm, đặt hàng và thanh toán.

CHƯƠNG 1: GIỚI THIỆU

1.1. Đặc tả ứng dụng

Đề tài "Phát Triển Ứng Dụng Bán Hàng Trực Tuyến Dựa Trên Nodejs Theo Phương Pháp Agile" nhằm mục đích nghiên cứu và áp dụng các nguyên tắc của phương pháp Agile vào quá trình phát triển. Cụ thể, đề tài tập trung vào việc khám phá khả năng và hiệu quả của NodeJS trong việc xây dựng các ứng dụng web, đồng thời đánh giá tính linh hoạt và khả năng thích ứng của nền tảng này với các yêu cầu thay đổi liên tục từ người dùng.

Nghiên cứu đã tiến hành phân tích các yêu cầu chức năng và phi chức năng của một hệ thống bán hàng trực tuyến, bao gồm việc thiết kế cơ sở dữ liệu và giao diện người dùng. Việc lựa chọn và sử dụng các thư viện phù hợp như Express, Mongoose, và các middleware khác đã được thực hiện để hỗ trợ quá trình phát triển, cũng như việc sử dụng Nodemon để cải thiện quá trình phát triển ứng dụng.

Kết quả của đề tài là việc triển khai thành công một trang web bán hàng trực tuyến với các chức năng cơ bản, bao gồm hiển thị sản phẩm, tìm kiếm, chỉnh sửa và đặt mua sản phẩm. Đề tài cũng đã chứng minh được khả năng mở rộng và tích hợp thêm các tính năng nâng cao trong tương lai, phản ánh sự linh hoạt và hiệu quả của việc áp dụng Agile trong phát triển phần mềm.

Đề tài này không chỉ cung cấp một mô hình thực tiễn cho việc áp dụng Agile trong phát triển phần mềm, mà còn góp phần vào việc nâng cao hiểu biết về NodeJS và các công nghệ liên quan, qua đó mở rộng khả năng áp dụng của nền tảng này trong các dự án phát triển phần mềm khác.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về Agile

2.1.1. Giới thiệu về Agile

Agile (phát triển phần mềm linh hoạt) không phải là một phương pháp hay quy trình cụ thể, mà là một tập hợp các giá trị và nguyên tắc hướng dẫn cách tiếp cận công việc và giải quyết vấn đề. Agile nhấn mạnh vào sự linh hoạt, thích ứng, cộng tác và tập trung vào giá trị để đạt được mục tiêu.

Agile ra đời vào năm 2001 khi 17 chuyên gia phát triển phần mềm đã cùng nhau soạn thảo và ký kết bản Tuyên ngôn Agile (Agile Manifesto). Tuyên ngôn này đã đặt nền móng cho sự phát triển của Agile và trở thành kim chỉ nam cho các phương pháp, quy trình Agile sau này.

Bốn giá trị cốt lõi của Agile:

- Con người và tương tác hơn quy trình và công cụ.
- Phần mềm hoạt động hơn tài liệu đầy đủ.
- Hợp tác với khách hàng hơn đàm phán hợp đồng.
- Phản ứng với thay đổi hơn tuân theo kế hoạch.

Những giá trị này được hỗ trợ bởi 12 nguyên tắc hướng dẫn quy trình Agile, đảm bảo cải tiến liên tục và khả năng thích ứng trong suốt vòng đời của dự án.

2.1.2. Các nguyên tắc của Agile

Agile không chỉ là một tập hợp các phương pháp, mà còn là một triết lý làm việc dựa trên 12 nguyên tắc cốt lõi. Các nguyên tắc này hướng dẫn cách các nhóm Agile tương tác, làm việc và cung cấp giá trị. Dưới đây là các nguyên tắc tương ứng:

Ưu tiên cao nhất là làm hài lòng khách hàng thông qua việc chuyển giao phần mềm có giá trị sớm và liên tục: Agile tập trung vào việc cung cấp giá trị cho khách hàng một cách nhanh chóng và liên tục, thay vì chờ đợi đến khi dự án hoàn thành. Điều này giúp khách hàng nhận được lợi ích sớm và có thể đưa ra phản hồi để cải thiện sản phẩm.

Chấp nhận thay đổi yêu cầu, ngay cả ở giai đoạn cuối của dự án: Agile hiểu rằng yêu cầu có thể thay đổi trong suốt quá trình phát triển dự án. Thay vì chống lại sự thay đổi, Agile khuyến khích các nhóm đón nhận và thích ứng với những thay đổi đó để đảm bảo sản phẩm cuối cùng đáp ứng được nhu cầu thực tế của khách hàng.

Chuyển giao phần mềm hoạt động thường xuyên (từ vài tuần đến vài tháng): Agile chia dự án thành các giai đoạn nhỏ hơn và chuyển giao các phần mềm hoạt động thường xuyên. Điều này giúp khách hàng có thể sử dụng và đánh giá sản phẩm sớm, đồng thời giúp nhóm phát triển nhận được phản hồi và cải thiện sản phẩm trong quá trình phát triển.

Nhà kinh doanh và nhà phát triển phải làm việc cùng nhau hàng ngày trong suốt dự án: Agile khuyến khích sự hợp tác chặt chẽ giữa các bên liên quan, bao gồm cả nhà kinh doanh (người hiểu rõ yêu cầu của khách hàng) và nhà phát triển (người xây dựng sản phẩm). Sự hợp tác này giúp đảm bảo rằng sản phẩm được phát triển đúng hướng và đáp ứng được nhu cầu của khách hàng.

Xây dựng dự án xung quanh những cá nhân có động lực: Agile tin rằng những cá nhân có động lực và được trao quyền sẽ làm việc hiệu quả hơn. Các nhóm Agile thường được tự quản lý và có quyền tự quyết định về cách thức làm việc của mình.

Đối thoại trực tiếp là phương thức giao tiếp hiệu quả nhất: Agile khuyến khích giao tiếp trực tiếp giữa các thành viên trong nhóm và với khách hàng. Điều này giúp tránh hiểu lầm và đảm bảo thông tin được truyền đạt một cách chính xác và hiệu quả.

Phần mềm hoạt động là thước đo chính của tiến độ: Agile không chỉ đo lường tiến độ dựa trên các tài liệu hoặc kế hoạch, mà còn dựa trên phần mềm hoạt động thực tế. Điều này giúp đảm bảo rằng dự án đang đi đúng hướng và sản phẩm đang được phát triển đáp ứng được yêu cầu của khách hàng.

Các quy trình Agile thúc đẩy phát triển bền vững: Agile khuyến khích các nhóm làm việc với tốc độ ổn định và bền vững. Điều này giúp tránh tình trạng kiệt sức và đảm bảo rằng nhóm có thể tiếp tục cung cấp giá trị trong thời gian dài.

Chú trọng kỹ thuật và thiết kế tốt giúp tăng cường sự nhanh nhạy: Agile không chỉ tập trung vào việc chuyển giao sản phẩm nhanh chóng, mà còn quan tâm đến chất lượng kỹ thuật và thiết kế. Một sản phẩm có thiết kế tốt sẽ dễ dàng bảo trì và mở rộng, từ đó giúp nhóm phát triển nhanh chóng thích ứng với những thay đổi trong tương lai.

Sự đơn giản – nghệ thuật tối đa hóa lượng công việc không cần làm – là điều cần thiết: Agile khuyến khích các nhóm tập trung vào những công việc quan trọng nhất và loại bỏ những công việc không cần thiết. Điều này giúp tiết kiệm thời gian và tài nguyên, đồng thời giúp nhóm làm việc hiệu quả hơn.

Các nhóm tự tổ chức tạo ra các kiến trúc, yêu cầu và thiết kế tốt nhất: Agile tin rằng các nhóm tự tổ chức có khả năng tự đưa ra quyết định và giải quyết vấn đề tốt hơn so với các nhóm được quản lý chặt chẽ. Các nhóm tự tổ chức thường có sự sáng tạo và đổi mới cao hơn.

Nhóm thường xuyên phản tư về cách làm việc hiệu quả hơn và điều chỉnh hành vi cho phù hợp: Agile khuyến khích các nhóm thường xuyên đánh giá và cải tiến cách làm việc của mình. Điều này giúp nhóm học hỏi từ những sai lầm và liên tục cải thiện hiệu suất làm việc.

2.1.3. Lợi ích và thách thức khi áp dụng Agile

Agile mang lại nhiều lợi ích đáng kể cho các tổ chức và dự án, nhưng cũng đi kèm với những thách thức cần được xem xét và giải quyết.

Lợi ích khi áp dụng Agile:

- **Tăng khả năng thích ứng với thay đổi:** Agile cho phép các nhóm phản ứng nhanh chóng và linh hoạt với những thay đổi trong yêu cầu, công nghệ hoặc thị trường. Điều này giúp dự án luôn đi đúng hướng và đáp ứng được nhu cầu thực tế của khách hàng.
- **Cải thiện chất lượng sản phẩm:** Agile tập trung vào việc chuyển giao các phần mềm hoạt động thường xuyên, giúp phát hiện và sửa lỗi sớm, từ đó nâng cao chất lượng sản phẩm cuối cùng.
- **Tăng sự hài lòng của khách hàng:** Agile khuyến khích sự tham gia tích cực của khách hàng trong suốt quá trình phát triển dự án, giúp đảm bảo rằng sản phẩm đáp ứng được mong đợi và nhu cầu của họ.
- **Tăng cường sự cộng tác và tinh thần làm việc nhóm:** Agile thúc đẩy sự giao tiếp mở và thường xuyên giữa các thành viên trong nhóm, tạo ra một môi trường làm việc tích cực và hiệu quả.
- **Giảm thiểu rủi ro:** Agile giúp giảm thiểu rủi ro bằng cách chia nhỏ dự án thành các giai đoạn nhỏ hơn, dễ quản lý và kiểm soát hơn.
- **Tăng tốc độ đưa sản phẩm ra thị trường:** Agile giúp rút ngắn thời gian phát triển sản phẩm bằng cách tập trung vào các tính năng quan trọng nhất và chuyển giao chúng sớm.

Thách thức khi áp dụng Agile:

- **Thay đổi văn hóa tổ chức:** Agile đòi hỏi sự thay đổi trong cách suy nghĩ và làm việc của các thành viên trong tổ chức. Điều này có thể gặp phải sự kháng cự từ những người quen với cách làm việc truyền thống.
- **Yêu cầu kỹ năng và kinh nghiệm:** Agile đòi hỏi các thành viên trong nhóm có kỹ năng và kinh nghiệm nhất định để làm việc hiệu quả. Việc đào tạo và huấn luyện là cần thiết để đảm bảo mọi người có thể thích ứng với Agile.
- **Khó khăn trong việc đo lường tiến độ:** Agile không sử dụng các phương pháp đo lường tiến độ truyền thống như biểu đồ Gantt. Điều này có thể gây khó khăn cho việc theo dõi và báo cáo tiến độ dự án.
- **Quản lý sự kỳ vọng của khách hàng:** Agile có thể khiến khách hàng khó hình dung được sản phẩm cuối cùng sẽ như thế nào vì yêu cầu có thể thay đổi trong quá trình phát triển. Việc quản lý sự kỳ vọng của khách hàng là rất quan trọng để tránh những hiểu lầm và xung đột.
- **Áp dụng Agile một cách máy móc:** Agile không phải là một công thức cứng nhắc. Việc áp dụng Agile một cách máy móc mà không hiểu rõ nguyên tắc và giá trị cốt lõi của nó có thể dẫn đến thất bại.

Agile không phải là một giải pháp thần kỳ, nhưng nếu được áp dụng đúng cách, nó có thể mang lại những lợi ích to lớn cho các tổ chức và dự án. Bằng cách hiểu rõ lợi ích và thách thức của Agile, các tổ chức có thể đưa ra quyết định sáng suốt và xây dựng một kế hoạch áp dụng Agile hiệu quả.

2.1.4. Các phương pháp Agile phổ biến

Có rất nhiều phương pháp Agile khác nhau, mỗi phương pháp có những đặc điểm và cách tiếp cận riêng. Dưới đây là một số phương pháp Agile phổ biến nhất:

- **Scrum:** Scrum là một trong những phương pháp Agile phổ biến nhất, tập trung vào việc quản lý dự án và phát triển sản phẩm theo từng giai đoạn ngắn gọi là Sprint. Scrum nhấn mạnh vào sự tự quản lý của nhóm, giao tiếp thường xuyên và phản hồi liên tục để thích ứng với thay đổi.
- **Kanban:** Kanban là một phương pháp trực quan hóa công việc, sử dụng bảng Kanban để theo dõi tiến độ và quản lý luồng công việc. Kanban giúp các nhóm xác định các nút thắt cổ chai, tối ưu hóa quy trình và cải thiện hiệu suất.

- **Extreme Programming (XP):** XP là một phương pháp Agile tập trung vào kỹ thuật và chất lượng mã nguồn. XP khuyến khích các thực hành như lập trình cặp, tích hợp liên tục, kiểm thử tự động và tái cấu trúc mã để đảm bảo chất lượng sản phẩm.
- **Lean Software Development:** Lean Software Development là một phương pháp Agile dựa trên triết lý Lean, tập trung vào việc loại bỏ lãng phí và tối ưu hóa giá trị. Lean Software Development khuyến khích các thực hành như phát triển lặp lại, tích hợp liên tục, kiểm thử tự động và phản hồi nhanh chóng để giảm thiểu rủi ro và tăng tốc độ đưa sản phẩm ra thị trường.
- **Crystal:** Crystal là một họ các phương pháp Agile linh hoạt, tập trung vào con người và tương tác. Crystal khuyến khích các nhóm tự điều chỉnh và lựa chọn các thực hành phù hợp với đặc thù của dự án và môi trường làm việc.
- **Dynamic Systems Development Method (DSDM):** DSDM là một phương pháp Agile tập trung vào việc quản lý dự án và phát triển sản phẩm theo từng giai đoạn, với sự nhấn mạnh vào việc đáp ứng các ràng buộc về thời gian, chi phí và chất lượng.
- **Feature-Driven Development (FDD):** FDD là một phương pháp Agile tập trung vào việc phát triển sản phẩm theo từng tính năng. FDD chia dự án thành các mô hình tính năng nhỏ hơn, dễ quản lý và phát triển hơn.

Các yếu tố để lựa chọn phương pháp Agile phù hợp:

- **Đặc thù của dự án:** Kích thước, độ phức tạp, yêu cầu và mục tiêu của dự án.
- **Môi trường làm việc:** Văn hóa tổ chức, quy trình làm việc và kỹ năng của nhóm.
- **Kỳ vọng của khách hàng:** Mức độ tham gia, phản hồi và yêu cầu từ khách hàng.

Việc lựa chọn phương pháp Agile phù hợp là một bước quan trọng để đảm bảo thành công của dự án. Bằng cách tìm hiểu và đánh giá các phương pháp Agile khác nhau, các tổ chức có thể đưa ra quyết định sáng suốt và lựa chọn phương pháp phù hợp nhất với nhu cầu của mình.

2.2. Tổng quan về Scrum

2.2.1. Giới thiệu về Scrum

Scrum là một trong những phương pháp Agile phổ biến và được áp dụng rộng rãi nhất hiện nay. Nó được thiết kế để quản lý các dự án phức tạp, đặc biệt là trong lĩnh vực phát triển phần mềm, nhưng cũng có thể áp dụng cho nhiều lĩnh vực khác. Scrum tập

trung vào việc chia nhỏ công việc thành các giai đoạn ngắn gọi là Sprint, thường kéo dài từ 1 đến 4 tuần.

Đặc điểm chính của Scrum:

- **Tính linh hoạt và thích ứng:** Scrum cho phép các nhóm phản ứng nhanh chóng với những thay đổi trong yêu cầu, công nghệ hoặc thị trường.
- **Tập trung vào giá trị:** Scrum ưu tiên việc chuyển giao các sản phẩm có giá trị cho khách hàng sớm và thường xuyên.
- **Sự minh bạch:** Scrum khuyến khích sự minh bạch trong công việc và tiến độ, giúp các thành viên trong nhóm và các bên liên quan hiểu rõ tình hình dự án.
- **Sự tự quản lý:** Scrum trao quyền cho các nhóm tự tổ chức và tự quản lý công việc của mình.
- **Cải tiến liên tục:** Scrum khuyến khích các nhóm thường xuyên xem xét và cải tiến quy trình làm việc để nâng cao hiệu quả.

2.2.2. Các thành phần của Scrum

Scrum bao gồm ba thành phần chính: **Sự kiện, Vai trò và Công cụ**. Các thành phần này hoạt động cùng nhau để tạo nên một khung làm việc linh hoạt và hiệu quả cho các dự án.

Sự kiện (Events):

- **Sprint:** Khoảng thời gian lặp lại cố định (thường từ 1 đến 4 tuần) để hoàn thành một tập hợp công việc đã được xác định trước. Mỗi Sprint bao gồm các sự kiện khác như Sprint Planning, Daily Scrum, Sprint Review và Sprint Retrospective.
- **Sprint Planning:** Cuộc họp đầu tiên của mỗi Sprint, nơi nhóm xác định mục tiêu Sprint và lên kế hoạch chi tiết cho công việc cần hoàn thành.
- **Daily Scrum:** Cuộc họp ngắn hàng ngày (15 phút) để các thành viên trong nhóm cập nhật tiến độ, chia sẻ khó khăn và lên kế hoạch cho ngày tiếp theo.
- **Sprint Review:** Cuộc họp cuối mỗi Sprint để nhóm trình bày sản phẩm đã hoàn thành và nhận phản hồi từ các bên liên quan.
- **Sprint Retrospective:** Cuộc họp cuối mỗi Sprint để nhóm đánh giá quá trình làm việc và xác định các cải tiến cho Sprint tiếp theo.

Vai trò (Roles):

- **Product Owner:** Chịu trách nhiệm về tầm nhìn sản phẩm, xác định và ưu tiên các yêu cầu, đảm bảo sản phẩm đáp ứng nhu cầu của khách hàng.

- **Scrum Master:** Hỗ trợ nhóm áp dụng Scrum, loại bỏ các trở ngại, tạo điều kiện thuận lợi cho việc giao tiếp và hợp tác trong nhóm.
- **Development Team:** Nhóm các chuyên gia chịu trách nhiệm thực hiện công việc và chuyển giao sản phẩm.

Công cụ (Artifacts):

- **Product Backlog:** Danh sách tất cả các tính năng, yêu cầu và mục tiêu của sản phẩm, được sắp xếp theo thứ tự ưu tiên. Product Owner chịu trách nhiệm quản lý Product Backlog.
- **Sprint Backlog:** Danh sách các công việc được chọn từ Product Backlog để hoàn thành trong một Sprint. Nhóm phát triển chịu trách nhiệm quản lý Sprint Backlog.
- **Increment:** Kết quả của mỗi Sprint, là một phiên bản phần mềm có thể hoạt động được. Increment là một bước tiến tới mục tiêu cuối cùng của sản phẩm.

Ngoài ba thành phần chính trên, Scrum còn có các khái niệm quan trọng khác như:

- **Definition of Done:** Tiêu chuẩn để xác định khi nào một công việc được coi là hoàn thành.
- **Sprint Goal:** Mục tiêu của Sprint, là một kết quả cụ thể mà nhóm cam kết sẽ đạt được trong Sprint.
- **Velocity:** Tốc độ làm việc trung bình của nhóm trong một Sprint, được đo bằng số điểm Story Point hoàn thành.

Các thành phần của Scrum hoạt động cùng nhau để tạo ra một quy trình lặp đi lặp lại, giúp các nhóm liên tục cải tiến và cung cấp giá trị cho khách hàng. Sự kiện giúp nhóm lập kế hoạch, theo dõi tiến độ và cải tiến quy trình. Vai trò đảm bảo rằng mọi người có trách nhiệm rõ ràng và làm việc hiệu quả. Công cụ giúp nhóm quản lý công việc và theo dõi tiến độ.

2.2.3. Vai trò trong Scrum

Scrum xác định ba vai trò chính, mỗi vai trò có trách nhiệm và quyền hạn riêng biệt để đảm bảo dự án diễn ra suôn sẻ và đạt được mục tiêu.

Product Owner (Chủ sản phẩm):

Trách nhiệm:

- Xác định tầm nhìn sản phẩm và mục tiêu kinh doanh.

- Tạo và quản lý Product Backlog, đảm bảo các mục trong Backlog được sắp xếp theo thứ tự ưu tiên và rõ ràng.
- Đảm bảo rằng nhóm phát triển hiểu rõ các yêu cầu và mục tiêu.
- Chấp nhận hoặc từ chối kết quả công việc của nhóm phát triển.
- Đại diện cho lợi ích của các bên liên quan và khách hàng.

Kỹ năng cần thiết:

- Hiểu biết sâu sắc về thị trường và khách hàng.
- Khả năng giao tiếp và đàm phán tốt.
- Khả năng ra quyết định và giải quyết vấn đề.
- Tư duy chiến lược và tầm nhìn xa.

Scrum Master (Quản lý Scrum):

Trách nhiệm:

- Hỗ trợ nhóm phát triển hiểu và áp dụng Scrum.
- Loại bỏ các trở ngại và rào cản trong quá trình làm việc của nhóm.
- Tạo điều kiện thuận lợi cho các sự kiện Scrum (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective).
- Bảo vệ nhóm khỏi các yếu tố gây xao nhãng và ảnh hưởng đến tiến độ.
- Thúc đẩy sự tự quản lý và cải tiến liên tục của nhóm.

Kỹ năng cần thiết:

- Hiểu biết sâu sắc về Scrum và Agile.
- Khả năng lãnh đạo, huấn luyện và tạo động lực cho nhóm.
- Khả năng giải quyết xung đột và xây dựng mối quan hệ.
- Kỹ năng giao tiếp và thuyết trình tốt.

Development Team (Nhóm phát triển):

Trách nhiệm:

- Tự tổ chức và quản lý công việc để hoàn thành các mục tiêu của Sprint.
- Chịu trách nhiệm về chất lượng và tiến độ công việc.
- Ước lượng công sức và cam kết hoàn thành các mục tiêu của Sprint.
- Cộng tác chặt chẽ với Product Owner và Scrum Master.

Kỹ năng cần thiết:

- Kỹ năng chuyên môn trong lĩnh vực của mình

- Khả năng làm việc nhóm và tự quản lý.
- Kỹ năng giao tiếp và giải quyết vấn đề.
- Tinh thần trách nhiệm và cam kết cao.

2.2.4. Ưu điểm và nhược điểm của Scrum

Scrum là một phương pháp Agile mạnh mẽ và được sử dụng rộng rãi, nhưng không phải là không có những hạn chế. Việc hiểu rõ cả ưu điểm và nhược điểm của Scrum sẽ giúp bạn đánh giá xem phương pháp này có phù hợp với dự án và tổ chức của mình hay không.

Ưu điểm của Scrum:

- **Tính linh hoạt và thích ứng cao:** Scrum cho phép các nhóm phản ứng nhanh chóng với những thay đổi trong yêu cầu, công nghệ hoặc thị trường. Điều này đặc biệt quan trọng trong các dự án phức tạp và không chắc chắn.
- **Tập trung vào giá trị:** Scrum ưu tiên việc chuyển giao các sản phẩm có giá trị cho khách hàng sớm và thường xuyên. Điều này giúp khách hàng nhận được lợi ích sớm và có thể đưa ra phản hồi để cải thiện sản phẩm.
- **Tăng cường sự minh bạch:** Scrum khuyến khích sự minh bạch trong công việc và tiến độ, giúp các thành viên trong nhóm và các bên liên quan hiểu rõ tình hình dự án và đưa ra quyết định dựa trên thông tin chính xác.
- **Thúc đẩy sự tự quản lý:** Scrum trao quyền cho các nhóm tự tổ chức và tự quản lý công việc của mình. Điều này giúp tăng cường sự chủ động, sáng tạo và trách nhiệm của các thành viên trong nhóm.
- **Cải tiến liên tục:** Scrum khuyến khích các nhóm thường xuyên xem xét và cải tiến quy trình làm việc để nâng cao hiệu quả. Điều này giúp nhóm học hỏi từ những sai lầm và liên tục phát triển.

Nhược điểm của Scrum:

- **Phụ thuộc vào sự cam kết của nhóm:** Scrum đòi hỏi sự cam kết và hợp tác cao từ tất cả các thành viên trong nhóm. Nếu một thành viên không hoàn thành nhiệm vụ của mình, có thể ảnh hưởng đến tiến độ và kết quả của cả Sprint.
- **Khó khăn trong việc ước lượng và lập kế hoạch:** Scrum yêu cầu các nhóm ước lượng công việc và lập kế hoạch cho mỗi Sprint. Điều này có thể khó khăn, đặc biệt là đối với các dự án mới và phức tạp.

- **Khó áp dụng cho các dự án lớn và phức tạp:** Scrum có thể khó áp dụng cho các dự án lớn và phức tạp với nhiều nhóm phát triển và nhiều bên liên quan.
- **Tốn thời gian cho các cuộc họp:** Scrum yêu cầu các nhóm tổ chức nhiều cuộc họp, như Daily Scrum, Sprint Planning, Sprint Review và Sprint Retrospective. Điều này có thể tốn nhiều thời gian và ảnh hưởng đến tiến độ công việc.
- **Khó khăn trong việc đo lường hiệu quả:** Scrum không có các chỉ số đo lường hiệu quả rõ ràng như các phương pháp quản lý dự án truyền thống. Điều này có thể gây khó khăn cho việc đánh giá hiệu quả của dự án và đưa ra quyết định cải tiến.

Scrum là một phương pháp Agile mạnh mẽ và có thể mang lại nhiều lợi ích cho các dự án. Tuy nhiên, để áp dụng Scrum thành công, các tổ chức cần phải hiểu rõ cả ưu điểm và nhược điểm của nó, đồng thời có sự chuẩn bị và cam kết từ tất cả các thành viên trong nhóm.

2.2.5. So sánh Scrum với các phương pháp Agile khác

Scrum không phải là phương pháp Agile duy nhất. Có nhiều phương pháp khác nhau, mỗi phương pháp có những đặc điểm và cách tiếp cận riêng. Dưới đây là bảng so sánh Scrum với một số phương pháp Agile phổ biến khác:

Scrum tập trung vào việc quản lý dự án và phát triển sản phẩm theo từng giai đoạn ngắn gọi là Sprint. Scrum có cấu trúc chặt chẽ với các vai trò, sự kiện và công cụ cụ thể. Phương pháp này phù hợp với các dự án phức tạp, yêu cầu sự linh hoạt và thích ứng cao. Tuy nhiên, Scrum có thể trở nên cứng nhắc và tốn thời gian cho các dự án nhỏ hơn hoặc các nhóm chưa quen với Agile.

Kanban lại tập trung vào việc quản lý luồng công việc và tối ưu hóa quy trình. Kanban sử dụng bảng Kanban để trực quan hóa công việc và giới hạn số lượng công việc đang thực hiện (WIP). Phương pháp này rất linh hoạt và dễ áp dụng, đặc biệt phù hợp với các nhóm muốn cải thiện quy trình làm việc hiện tại. Tuy nhiên, Kanban không có khung thời gian cụ thể và có thể khó áp dụng cho các dự án lớn hoặc phức tạp.

Extreme Programming (XP) tập trung vào kỹ thuật và chất lượng mã nguồn. XP khuyến khích các thực hành như lập trình cặp, tích hợp liên tục, kiểm thử tự động và tái cấu trúc mã. Phương pháp này phù hợp với các dự án yêu cầu chất lượng phần mềm cao và có đội ngũ kỹ thuật mạnh.

Lean Software Development dựa trên triết lý Lean, tập trung vào việc loại bỏ lãng phí và tối ưu hóa giá trị. Lean Software Development khuyến khích các thực hành như phát triển lặp lại, tích hợp liên tục, kiểm thử tự động và phản hồi nhanh chóng. Phương pháp này phù hợp với các dự án cần tối ưu hóa quy trình và giảm thiểu rủi ro. Tuy nhiên, Lean Software Development đòi hỏi sự thay đổi tư duy và văn hóa làm việc và có thể khó đo lường hiệu quả.

Tóm lại, việc lựa chọn phương pháp Agile phù hợp phụ thuộc vào đặc thù của dự án, môi trường làm việc và kỳ vọng của khách hàng. Scrum phù hợp với các dự án phức tạp và yêu cầu sự linh hoạt, Kanban phù hợp với các dự án cần cải thiện quy trình làm việc, XP phù hợp với các dự án yêu cầu chất lượng phần mềm cao, và Lean Software Development phù hợp với các dự án cần tối ưu hóa giá trị và giảm thiểu lãng phí.

2.3. Tổng quan về NodeJS

2.3.1. Giới thiệu NodeJS

NodeJS, hay còn được gọi là Node.js, là một môi trường chạy (runtime environment) mã nguồn mở và đa nền tảng, cho phép thực thi mã JavaScript bên ngoài trình duyệt web. Được xây dựng trên nền tảng V8 JavaScript engine của Google Chrome, NodeJS mang đến khả năng xây dựng các ứng dụng mạng và máy chủ hiệu suất cao, có khả năng mở rộng.

Đặc điểm nổi bật của NodeJS:

- **Đơn luồng (Single-threaded):** NodeJS hoạt động trên một luồng duy nhất, sử dụng mô hình non-blocking I/O để xử lý nhiều yêu cầu đồng thời mà không bị tắc nghẽn.
- **Phi đồng bộ (Asynchronous):** Các hoạt động trong NodeJS được thực hiện không theo thứ tự, cho phép xử lý nhiều tác vụ cùng lúc mà không cần chờ đợi một tác vụ hoàn thành trước khi bắt đầu tác vụ khác.
- **Hướng sự kiện (Event-driven):** NodeJS sử dụng kiến trúc hướng sự kiện, trong đó các sự kiện (như yêu cầu HTTP, kết nối cơ sở dữ liệu) được xử lý bởi các hàm callback.
- **Nhanh chóng và hiệu quả:** Nhờ vào V8 JavaScript engine và kiến trúc non-blocking I/O, NodeJS có khả năng xử lý một lượng lớn yêu cầu đồng thời với hiệu suất cao.

- **Cộng đồng lớn và hệ sinh thái phong phú:** NodeJS có một cộng đồng phát triển lớn và tích cực, cung cấp nhiều thư viện và công cụ hỗ trợ cho việc xây dựng ứng dụng.

2.3.2. Cách hoạt động của NodeJS

NodeJS hoạt động dựa trên một số khái niệm và cơ chế chính:

V8 JavaScript Engine: NodeJS sử dụng V8, một công cụ JavaScript cực kỳ nhanh được phát triển bởi Google cho trình duyệt Chrome. V8 biên dịch mã JavaScript thành mã máy, giúp NodeJS thực thi JavaScript với tốc độ rất cao.

Libuv: Libuv là một thư viện cung cấp cơ chế vòng lặp sự kiện (event loop) và xử lý I/O không đồng bộ (asynchronous I/O) cho NodeJS. Nó giúp NodeJS xử lý nhiều yêu cầu cùng lúc mà không bị chặn.

Mô hình đơn luồng (Single-threaded) và non-blocking I/O: NodeJS hoạt động trên một luồng duy nhất, nhưng có thể xử lý nhiều yêu cầu cùng lúc nhờ vào mô hình non-blocking I/O. Khi một yêu cầu I/O (như đọc file, truy vấn cơ sở dữ liệu) được gửi đi, NodeJS không chờ đợi kết quả mà tiếp tục xử lý các yêu cầu khác. Khi yêu cầu I/O hoàn thành, một sự kiện (event) sẽ được tạo ra và thêm vào hàng đợi sự kiện (event queue).

Vòng lặp sự kiện (Event Loop): Vòng lặp sự kiện là một cơ chế liên tục kiểm tra hàng đợi sự kiện. Khi có sự kiện trong hàng đợi, vòng lặp sự kiện sẽ lấy ra và xử lý sự kiện đó. Vòng lặp sự kiện đảm bảo rằng các sự kiện được xử lý theo thứ tự và không có sự kiện nào bị bỏ qua.

Callback: Callback là một hàm được truyền vào một hàm khác như một tham số và sẽ được gọi lại khi hàm đó hoàn thành. NodeJS sử dụng callback để xử lý các sự kiện I/O không đồng bộ. Khi một yêu cầu I/O hoàn thành, callback tương ứng sẽ được gọi để xử lý kết quả.

Quy trình hoạt động của NodeJS:

- **Bước 1:** Khi một yêu cầu đến, NodeJS sẽ nhận và xử lý yêu cầu đó trên luồng chính.
- **Bước 2:** Nếu yêu cầu liên quan đến I/O (như đọc file, truy vấn cơ sở dữ liệu), NodeJS sẽ gửi yêu cầu I/O đến hệ điều hành và tiếp tục xử lý các yêu cầu khác.
- **Bước 3:** Khi yêu cầu I/O hoàn thành, hệ điều hành sẽ gửi một tín hiệu đến NodeJS.

- **Bước 4:** NodeJS nhận được tín hiệu và tạo ra một sự kiện tương ứng, thêm vào hàng đợi sự kiện.
- **Bước 5:** Vòng lặp sự kiện liên tục kiểm tra hàng đợi sự kiện. Khi có sự kiện, vòng lặp sự kiện sẽ lấy ra và gọi callback tương ứng để xử lý sự kiện.

2.3.3. Ưu điểm và nhược điểm của NodeJS

NodeJS là một công nghệ mạnh mẽ và linh hoạt, nhưng cũng có những ưu điểm và nhược điểm riêng. Hiểu rõ những điểm này sẽ giúp bạn đưa ra quyết định sáng suốt khi lựa chọn NodeJS cho dự án của mình.

Ưu điểm của NodeJS:

- **Hiệu suất cao:** Nhờ sử dụng kiến trúc non-blocking I/O và V8 JavaScript engine, NodeJS có thể xử lý một lượng lớn yêu cầu đồng thời với hiệu suất cao. Điều này đặc biệt hữu ích cho các ứng dụng web thời gian thực và các ứng dụng cần xử lý nhiều kết nối đồng thời.
- **Khả năng mở rộng:** NodeJS dễ dàng mở rộng theo chiều ngang bằng cách thêm nhiều máy chủ vào hệ thống. Điều này giúp ứng dụng có thể đáp ứng được lượng truy cập lớn và tăng trưởng nhanh chóng.
- **Dễ học và sử dụng:** NodeJS sử dụng JavaScript, một ngôn ngữ lập trình phổ biến và dễ học. Điều này giúp các nhà phát triển web dễ dàng chuyển sang NodeJS và bắt đầu xây dựng ứng dụng.
- **Cộng đồng lớn và hệ sinh thái phong phú:** NodeJS có một cộng đồng phát triển lớn và tích cực, cung cấp nhiều thư viện, framework và công cụ hỗ trợ cho việc xây dựng ứng dụng.
- **Phát triển full-stack:** NodeJS cho phép sử dụng JavaScript cho cả frontend và backend, giúp giảm thiểu sự khác biệt giữa các công nghệ và tăng tốc độ phát triển.

Nhược điểm của NodeJS:

- **Không phù hợp với các tác vụ tính toán nặng:** Do sử dụng mô hình đơn luồng, NodeJS không phù hợp với các tác vụ tính toán nặng có thể chặn vòng lặp sự kiện (event loop).
- **Callback hell:** Việc sử dụng quá nhiều callback lồng nhau có thể dẫn đến mã nguồn khó đọc và khó bảo trì.

- **Quản lý lỗi:** Việc xử lý lỗi trong NodeJS có thể phức tạp do tính chất không đồng bộ của nó. Cần phải cẩn thận xử lý các lỗi có thể xảy ra trong quá trình thực thi để đảm bảo ứng dụng hoạt động ổn định.
- **Tính ổn định của API:** API của NodeJS vẫn đang trong quá trình phát triển và có thể thay đổi giữa các phiên bản. Điều này có thể gây khó khăn cho việc bảo trì và nâng cấp ứng dụng.

NodeJS là một công nghệ mạnh mẽ và linh hoạt với nhiều ưu điểm vượt trội. Tuy nhiên, nó cũng có những hạn chế nhất định. Việc hiểu rõ cả ưu điểm và nhược điểm của NodeJS sẽ giúp bạn đánh giá xem nó có phù hợp với dự án của mình hay không và đưa ra quyết định đúng đắn trong quá trình phát triển ứng dụng.

2.3.4. Các ứng dụng phổ biến của NodeJS

NodeJS được sử dụng rộng rãi trong nhiều lĩnh vực và ngành công nghiệp khác nhau, nhờ vào khả năng xử lý hiệu quả các tác vụ I/O, xây dựng ứng dụng thời gian thực và khả năng mở rộng cao. Dưới đây là một số ứng dụng phổ biến của NodeJS:

Ứng dụng web:

- **Ứng dụng web thời gian thực (Real-time web applications):** NodeJS rất phù hợp để xây dựng các ứng dụng web thời gian thực như ứng dụng chat, ứng dụng cộng tác, bảng tin trực tuyến, ứng dụng theo dõi trực tiếp, trò chơi trực tuyến, v.v.
- **Ứng dụng một trang (Single-page applications - SPA):** NodeJS thường được sử dụng để xây dựng các SPA, nơi toàn bộ ứng dụng được tải trong một trang duy nhất và nội dung được cập nhật động.
- **Microservices:** NodeJS là một lựa chọn phổ biến để xây dựng các kiến trúc microservices, nơi ứng dụng được chia thành các dịch vụ nhỏ, độc lập và giao tiếp với nhau thông qua API.

API (Application Programming Interface):

- **RESTful API:** NodeJS là một công cụ mạnh mẽ để xây dựng các API RESTful, cung cấp dữ liệu và chức năng cho các ứng dụng web, ứng dụng di động và các hệ thống khác.
- **GraphQL API:** NodeJS cũng có thể được sử dụng để xây dựng các API GraphQL, cung cấp một cách linh hoạt và hiệu quả hơn để truy vấn dữ liệu so với REST.

Ứng dụng dòng lệnh (Command-line applications):

- **Công cụ dòng lệnh:** NodeJS có thể được sử dụng để xây dựng các công cụ dòng lệnh đơn giản hoặc phức tạp, phục vụ cho các tác vụ tự động hóa, quản lý hệ thống, xử lý dữ liệu, v.v.

Ứng dụng desktop:

- **Electron:** Electron là một framework phổ biến sử dụng NodeJS và Chromium để xây dựng các ứng dụng desktop đa nền tảng, chạy trên Windows, macOS và Linux. Các ứng dụng nổi tiếng như Visual Studio Code, Slack và Discord được xây dựng bằng Electron.

Ứng dụng di động:

- **React Native:** React Native là một framework sử dụng NodeJS để xây dựng các ứng dụng di động đa nền tảng, chạy trên iOS và Android.

Ứng dụng IoT (Internet of Things):

- **Node-RED:** Node-RED là một công cụ lập trình trực quan dựa trên NodeJS, cho phép kết nối các thiết bị IoT và xây dựng các luồng công việc để xử lý dữ liệu IoT.

Ứng dụng xử lý dữ liệu lớn (Big data):

- **Apache Kafka:** NodeJS có thể được sử dụng để xây dựng các ứng dụng xử lý dữ liệu lớn với Apache Kafka, một nền tảng xử lý luồng dữ liệu phân tán.

Ứng dụng machine learning:

- **TensorFlow.js:** TensorFlow.js là một thư viện JavaScript cho phép chạy các mô hình machine learning trong trình duyệt hoặc trên NodeJS.

2.4. Tổng quan về Express Framework

2.4.1. Giới thiệu Express Framework

Express.js, thường được gọi tắt là Express, là một web framework tối giản và linh hoạt cho Node.js. Nó cung cấp một tập hợp các tính năng mạnh mẽ để xây dựng các ứng dụng web và API một cách nhanh chóng và dễ dàng. Express được xem là một trong những web framework phổ biến nhất cho Node.js và được sử dụng rộng rãi trong cộng đồng phát triển.

Đặc điểm nổi bật của Express:

- **Template engines:** Express hỗ trợ nhiều template engine khác nhau như Pug, EJS và Handlebars, cho phép bạn tạo ra các trang web động một cách dễ dàng.

- **Tối giản:** Express được thiết kế với triết lý "tối giản", chỉ cung cấp các tính năng cốt lõi cần thiết để xây dựng ứng dụng web. Điều này giúp Express nhẹ nhàng, nhanh chóng và dễ học.
- **Linh hoạt:** Express cho phép bạn tùy chỉnh và mở rộng dễ dàng để đáp ứng các yêu cầu cụ thể của dự án. Bạn có thể sử dụng các middleware (phần mềm trung gian) để thêm các chức năng như xác thực, phân tích cú pháp body, xử lý session, v.v.
- **Routing:** Express cung cấp một hệ thống routing mạnh mẽ để định nghĩa các tuyến đường (routes) cho ứng dụng của bạn. Bạn có thể dễ dàng ánh xạ các URL đến các hàm xử lý (handlers) cụ thể.
- **Middleware:** Middleware là các hàm được thực thi trong quá trình xử lý yêu cầu. Bạn có thể sử dụng middleware để thực hiện các tác vụ như ghi log, xác thực, phân tích cú pháp body, v.v.

2.4.2. Ứng dụng của Express Framework trong phát triển web

Express.js là một công cụ mạnh mẽ và linh hoạt, được sử dụng rộng rãi trong việc xây dựng các ứng dụng web hiện đại. Nhờ vào tính đơn giản, khả năng mở rộng và hệ sinh thái phong phú, Express.js đã trở thành lựa chọn hàng đầu cho nhiều nhà phát triển. Dưới đây là một số ứng dụng phổ biến của Express.js trong phát triển web:

Xây dựng API (Application Programming Interface): Express.js cung cấp các công cụ và tính năng cần thiết để xây dựng các API RESTful một cách nhanh chóng và dễ dàng. Với Express.js, bạn có thể định nghĩa các tuyến đường (routes), xử lý các yêu cầu HTTP, xác thực người dùng, truy cập cơ sở dữ liệu và trả về dữ liệu ở định dạng JSON hoặc XML.

Xây dựng ứng dụng web một trang (Single-Page Applications - SPA): Express.js có thể được sử dụng để xây dựng phần backend của các ứng dụng SPA, cung cấp API cho phần frontend (thường được xây dựng bằng các framework như React, Angular hoặc Vue.js) để tương tác và lấy dữ liệu.

Xây dựng ứng dụng web đa trang (Multi-Page Applications - MPA): Express.js cũng hỗ trợ xây dựng các ứng dụng web đa trang truyền thống, sử dụng các template engine như Pug, EJS hoặc Handlebars để tạo ra các trang HTML động.

Xây dựng các ứng dụng web thời gian thực (Real-time web applications): Kết hợp với các thư viện như Socket.IO, Express.js cho phép xây dựng các ứng dụng web thời gian thực, nơi dữ liệu được cập nhật và hiển thị cho người dùng ngay lập tức mà không cần phải tải lại trang.

Xây dựng các ứng dụng web server-side rendering (SSR): Express.js có thể được sử dụng để xây dựng các ứng dụng web SSR, nơi nội dung HTML được tạo ra trên máy chủ và gửi đến trình duyệt của người dùng. Điều này giúp cải thiện hiệu suất và SEO của trang web.

Tạo các middleware tùy chỉnh: Express.js cho phép bạn tạo các middleware tùy chỉnh để thực hiện các tác vụ cụ thể như ghi log, xác thực, phân tích cú pháp body, xử lý session, v.v. Các middleware này có thể được sử dụng lại trong nhiều dự án khác nhau, giúp tiết kiệm thời gian và công sức phát triển.

Tích hợp với các công nghệ khác: Express.js có thể dễ dàng tích hợp với các công nghệ khác như cơ sở dữ liệu (MongoDB, PostgreSQL, MySQL), hệ thống xác thực (Passport.js), công cụ ORM (Sequelize, Mongoose), v.v.

2.4.3. Các tính năng chính của Express Framework

Express.js cung cấp một loạt các tính năng mạnh mẽ giúp việc xây dựng ứng dụng web trở nên đơn giản và hiệu quả hơn:

Routing: Định nghĩa các tuyến đường (routes) cho ứng dụng, ánh xạ các URL tới các hàm xử lý (handlers) cụ thể. Hỗ trợ các phương thức HTTP (GET, POST, PUT, DELETE, etc.) và các tham số động trong URL. Cho phép xây dựng cấu trúc ứng dụng rõ ràng và dễ quản lý.

Middleware: Các hàm được thực thi trong quá trình xử lý yêu cầu, cho phép thực hiện các tác vụ như: Xác thực và ủy quyền người dung, phân tích cú pháp body của yêu cầu, ghi log, xử lý lỗi, nén dữ liệu, … Tạo ra một chuỗi xử lý yêu cầu linh hoạt và có thể mở rộng.

Template Engines:

- Hỗ trợ nhiều template engine phổ biến như Pug, EJS, Handlebars, etc.
- Cho phép tạo ra các trang HTML động bằng cách kết hợp dữ liệu với các mẫu (templates).
- Giúp tách biệt phần logic và phần hiển thị của ứng dụng.

Xử lý yêu cầu và phản hồi:

- Cung cấp các đối tượng req (request) và res (response) để tương tác với yêu cầu và phản hồi HTTP.
- Cho phép đọc dữ liệu từ yêu cầu, gửi dữ liệu về phía client, thiết lập header, cookie, v.v.
- Static File Serving:
- Dễ dàng phục vụ các tệp tĩnh (như hình ảnh, CSS, JavaScript) từ một thư mục cụ thể.

Error Handling: Cung cấp cơ chế xử lý lỗi tập trung, giúp dễ dàng bắt và xử lý các lỗi xảy ra trong quá trình thực thi ứng dụng.

Debugging: Hỗ trợ các công cụ gỡ lỗi để giúp bạn tìm và sửa lỗi trong ứng dụng của mình.

Hệ sinh thái phong phú: Có một cộng đồng lớn và tích cực, cung cấp nhiều thư viện và mô-đun mở rộng cho Express.js. Dễ dàng tích hợp với các công nghệ khác như cơ sở dữ liệu, hệ thống xác thực, v.v.

Hiệu suất: Express.js được thiết kế để tối ưu hóa hiệu suất, giúp ứng dụng của bạn chạy nhanh và mượt mà.

2.4.4. Ưu điểm và nhược điểm của Express Framework

Express.js là một web framework mạnh mẽ và phổ biến cho Node.js, tuy nhiên nó cũng có những ưu điểm và nhược điểm cần được xem xét trước khi lựa chọn sử dụng.

Ưu điểm:

- **Đơn giản và dễ sử dụng:** Express.js có cú pháp đơn giản, dễ hiểu và dễ học, đặc biệt là đối với những người đã quen thuộc với JavaScript. Nó cung cấp các tính năng cốt lõi cần thiết để xây dựng ứng dụng web mà không gây quá tải cho người mới bắt đầu.
- **Linh hoạt và có thể mở rộng:** Express.js cho phép bạn tùy chỉnh và mở rộng ứng dụng một cách dễ dàng thông qua việc sử dụng middleware. Bạn có thể thêm các chức năng mới, tích hợp với các thư viện và công cụ khác một cách linh hoạt để đáp ứng các yêu cầu cụ thể của dự án.
- **Hiệu suất cao:** Express.js được xây dựng trên nền tảng Node.js, tận dụng kiến trúc non-blocking I/O để xử lý nhiều yêu cầu đồng thời một cách hiệu quả. Điều

này giúp ứng dụng Express.js có thể xử lý một lượng lớn lưu lượng truy cập mà không bị giảm hiệu suất.

- **Cộng đồng lớn và hệ sinh thái phong phú:** Express.js có một cộng đồng phát triển lớn và tích cực, cung cấp nhiều tài liệu, hướng dẫn và hỗ trợ. Ngoài ra, có rất nhiều thư viện và mô-đun có sẵn để mở rộng chức năng của Express.js, giúp bạn tiết kiệm thời gian và công sức phát triển.
- **Phổ biến và được sử dụng rộng rãi:** Express.js là một trong những web framework phổ biến nhất cho Node.js và được sử dụng bởi nhiều công ty và dự án lớn. Điều này đảm bảo rằng bạn sẽ dễ dàng tìm thấy sự hỗ trợ và tài nguyên cần thiết khi làm việc với Express.js.

Nhược điểm:

- **Thiếu cấu trúc rõ ràng:** Express.js không áp đặt một cấu trúc cụ thể cho ứng dụng của bạn. Điều này có thể là một lợi thế về tính linh hoạt, nhưng cũng có thể gây khó khăn cho việc tổ chức và quản lý mã nguồn, đặc biệt là đối với các dự án lớn và phức tạp.
- **Khó khăn trong việc xử lý các tác vụ phức tạp:** Express.js có thể không phải là lựa chọn tốt nhất cho các ứng dụng yêu cầu xử lý các tác vụ tính toán phức tạp hoặc các tác vụ đòi hỏi nhiều tài nguyên.
- **Cần hiểu rõ về JavaScript và Node.js:** Để sử dụng Express.js một cách hiệu quả, bạn cần có kiến thức vững chắc về JavaScript và Node.js. Điều này có thể là một trở ngại đối với những người mới bắt đầu.
- **Vấn đề bảo mật:** Giống như bất kỳ web framework nào khác, Express.js cũng có thể gặp phải các vấn đề bảo mật nếu không được cấu hình và sử dụng đúng cách. Bạn cần phải chú ý đến việc bảo mật ứng dụng của mình bằng cách sử dụng các biện pháp bảo mật phù hợp.

Express.js là một web framework mạnh mẽ và linh hoạt cho Node.js, mang lại nhiều lợi ích cho việc phát triển ứng dụng web. Tuy nhiên, nó cũng có những hạn chế nhất định. Việc cân nhắc kỹ lưỡng giữa ưu điểm và nhược điểm của Express.js sẽ giúp bạn đưa ra quyết định đúng đắn khi lựa chọn công cụ phát triển cho dự án của mình.

2.5. Tổng quan về MongoDB

2.5.1. Giới thiệu MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, linh hoạt và có khả năng mở rộng cao. Nó được thiết kế để lưu trữ dữ liệu phi cấu trúc hoặc bán cấu trúc dưới dạng các tài liệu JSON (JavaScript Object Notation). MongoDB thuộc loại cơ sở dữ liệu hướng tài liệu (document-oriented database), khác biệt với các cơ sở dữ liệu quan hệ truyền thống (như MySQL, PostgreSQL) sử dụng bảng và lược đồ cố định.

Đặc điểm nổi bật của MongoDB:

Hướng tài liệu (Document-oriented): Dữ liệu được lưu trữ dưới dạng các tài liệu JSON linh hoạt, không yêu cầu lược đồ cố định. Điều này cho phép dễ dàng lưu trữ và truy vấn các dữ liệu phức tạp và thay đổi thường xuyên.

Khả năng mở rộng (Scalability): MongoDB được thiết kế để dễ dàng mở rộng theo chiều ngang bằng cách thêm nhiều máy chủ vào hệ thống. Điều này giúp đáp ứng nhu cầu lưu trữ và truy vấn dữ liệu lớn.

Hiệu suất cao (High performance): MongoDB sử dụng kỹ thuật lưu trữ dữ liệu trên bộ nhớ và chỉ mục (indexing) hiệu quả, giúp tăng tốc độ truy vấn và xử lý dữ liệu.

Linh hoạt (Flexibility): MongoDB cho phép bạn dễ dàng thay đổi cấu trúc dữ liệu mà không cần phải thay đổi ứng dụng.

Cộng đồng lớn và hệ sinh thái phong phú: MongoDB có một cộng đồng người dùng và nhà phát triển lớn, cung cấp nhiều tài liệu, hướng dẫn và hỗ trợ. Ngoài ra, có rất nhiều thư viện và công cụ tích hợp với MongoDB, giúp bạn dễ dàng xây dựng và triển khai ứng dụng.

MongoDB là một lựa chọn phổ biến cho các ứng dụng hiện đại, đặc biệt là các ứng dụng web, ứng dụng di động và các hệ thống phân tích dữ liệu lớn. Với những tính năng mạnh mẽ và linh hoạt, MongoDB giúp bạn dễ dàng xây dựng và triển khai các ứng dụng có khả năng mở rộng cao và đáp ứng được nhu cầu lưu trữ và truy vấn dữ liệu ngày càng tăng cao.

2.5.2. Các tính năng chính của MongoDB

MongoDB cung cấp một loạt các tính năng mạnh mẽ hỗ trợ việc lưu trữ, truy vấn và quản lý dữ liệu hiệu quả:

Hướng tài liệu (Document-Oriented):

- Lưu trữ dữ liệu dưới dạng các tài liệu JSON linh hoạt, không cần lược đồ cố định.

- Cho phép lưu trữ dữ liệu có cấu trúc phức tạp, lồng nhau và thay đổi theo thời gian.
- Dễ dàng thêm, sửa, xóa các trường trong tài liệu mà không ảnh hưởng đến toàn bộ cấu trúc dữ liệu.

Index (Chỉ mục):

- Tạo chỉ mục trên các trường trong tài liệu để tăng tốc độ truy vấn.
- Hỗ trợ nhiều loại chỉ mục khác nhau như chỉ mục đơn, chỉ mục ghép, chỉ mục không gian địa lý, chỉ mục văn bản đầy đủ, v.v.
- Cho phép tùy chỉnh chỉ mục để tối ưu hóa hiệu suất truy vấn cho từng ứng dụng cụ thể.

Aggregation Framework:

- Cung cấp một khung mạnh mẽ để thực hiện các truy vấn và phân tích dữ liệu phức tạp.
- Hỗ trợ các phép toán như lọc, nhóm, sắp xếp, tính toán, kết hợp, v.v.
- Cho phép xây dựng các báo cáo và phân tích dữ liệu một cách linh hoạt và hiệu quả.

Replication (Sao chép):

- Tạo các bản sao của dữ liệu trên nhiều máy chủ để đảm bảo tính sẵn sàng cao và khả năng chịu lỗi.
- Hỗ trợ nhiều cấu hình sao chép khác nhau như replica set, master-slave, sharding, v.v.
- Giúp bảo vệ dữ liệu và đảm bảo tính liên tục của ứng dụng trong trường hợp xảy ra sự cố.

Sharding (Phân mảnh):

- Phân chia dữ liệu thành các phần nhỏ hơn và lưu trữ trên nhiều máy chủ khác nhau.
- Giúp mở rộng khả năng lưu trữ và truy vấn dữ liệu khi lượng dữ liệu tăng lên.
- Đảm bảo hiệu suất và khả năng mở rộng của ứng dụng.

Transactions (Giao dịch):

- Hỗ trợ các giao dịch đa tài liệu (multi-document transactions) để đảm bảo tính toàn vẹn dữ liệu.

- Cho phép thực hiện các thao tác trên nhiều tài liệu một cách nhất quán (consistency), cô lập (isolation) và bền vững (durability).

Security (Bảo mật):

- Cung cấp nhiều cơ chế bảo mật như xác thực, ủy quyền, mã hóa dữ liệu, kiểm soát truy cập, v.v.
- Giúp bảo vệ dữ liệu khỏi các truy cập trái phép và đảm bảo tính an toàn của ứng dụng.

Monitoring and Management (Giám sát và quản lý):

- Cung cấp các công cụ và tính năng để giám sát hiệu suất, tình trạng và hoạt động của hệ thống MongoDB.

2.5.3. Ưu điểm và nhược điểm của MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mạnh mẽ và linh hoạt, nhưng cũng có những ưu điểm và nhược điểm cần được cân nhắc khi lựa chọn sử dụng.

Ưu điểm của MongoDB:

- **Tính linh hoạt:** MongoDB không yêu cầu lược đồ (schema) cố định, cho phép bạn lưu trữ dữ liệu có cấu trúc linh hoạt và dễ dàng thay đổi. Điều này đặc biệt hữu ích cho các ứng dụng có yêu cầu thay đổi thường xuyên và dữ liệu không đồng nhất.
- **Khả năng mở rộng:** MongoDB dễ dàng mở rộng theo chiều ngang bằng cách thêm nhiều máy chủ vào hệ thống. Điều này giúp đáp ứng nhu cầu lưu trữ và truy vấn dữ liệu lớn một cách hiệu quả.
- **Hiệu suất cao:** Nhờ sử dụng kỹ thuật lưu trữ dữ liệu trên bộ nhớ và chỉ mục (indexing) hiệu quả, MongoDB có thể xử lý các truy vấn nhanh chóng và hiệu quả, đặc biệt là đối với các truy vấn đơn giản và các truy vấn tìm kiếm theo chỉ mục.
- **Cộng đồng lớn và hệ sinh thái phong phú:** MongoDB có một cộng đồng người dùng và nhà phát triển lớn, cung cấp nhiều tài liệu, hướng dẫn và hỗ trợ. Ngoài ra, có rất nhiều thư viện và công cụ tích hợp với MongoDB, giúp bạn dễ dàng xây dựng và triển khai ứng dụng.

Nhược điểm của MongoDB:

- **Không hỗ trợ ACID đầy đủ:** MongoDB không hỗ trợ đầy đủ các tính chất ACID (Atomicity, Consistency, Isolation, Durability) như các hệ quản trị cơ sở dữ liệu quan hệ truyền thống. Điều này có thể gây ra các vấn đề về tính toàn vẹn dữ liệu trong một số trường hợp.
- **Giới hạn về dung lượng tài liệu:** MongoDB giới hạn kích thước tối đa của một tài liệu là 16MB. Điều này có thể gây khó khăn cho việc lưu trữ các tài liệu lớn hoặc các tệp nhị phân.
- **Tốn nhiều bộ nhớ:** MongoDB sử dụng nhiều bộ nhớ hơn so với các hệ quản trị cơ sở dữ liệu quan hệ truyền thống. Điều này có thể gây ra vấn đề về hiệu suất nếu không được cấu hình và quản lý đúng cách.
- **Khó khăn trong việc thực hiện các truy vấn phức tạp:** MongoDB không hỗ trợ các truy vấn JOIN như các hệ quản trị cơ sở dữ liệu quan hệ truyền thống. Điều này có thể gây khó khăn cho việc thực hiện các truy vấn phức tạp liên quan đến nhiều collection.

MongoDB là một lựa chọn tuyệt vời cho các ứng dụng cần lưu trữ dữ liệu phi cấu trúc hoặc bán cấu trúc, yêu cầu khả năng mở rộng cao và hiệu suất truy vấn nhanh. Tuy nhiên, bạn cần cân nhắc kỹ lưỡng các ưu điểm và nhược điểm của MongoDB trước khi quyết định sử dụng nó cho dự án của mình.

2.6. Mô hình MVC

2.6.1. Giới thiệu về Mô hình MVC

Mô hình MVC (Model-View-Controller) là một mẫu kiến trúc phần mềm phổ biến, được sử dụng để thiết kế và phát triển các ứng dụng web và giao diện người dùng. MVC chia ứng dụng thành ba thành phần chính:

Model (Mô hình):

- Chịu trách nhiệm quản lý dữ liệu của ứng dụng, bao gồm việc truy xuất, cập nhật và xác thực dữ liệu.
- Đại diện cho cấu trúc và logic nghiệp vụ của ứng dụng.
- Không phụ thuộc vào các thành phần View và Controller.

View (Giao diện):

- Chịu trách nhiệm hiển thị dữ liệu cho người dùng.
- Thường là các tệp HTML, CSS và JavaScript để tạo ra giao diện người dùng.

- Nhận dữ liệu từ Model và hiển thị nó theo một định dạng cụ thể.

Controller (Bộ điều khiển):

- Chịu trách nhiệm xử lý các yêu cầu từ người dùng.
- Nhận đầu vào từ người dùng, tương tác với Model để lấy dữ liệu, và sau đó cập nhật View để hiển thị kết quả.
- Điều khiển luồng dữ liệu giữa Model và View.

2.6.2. Cách hoạt động của mô hình MVC

Mô hình MVC hoạt động dựa trên sự phối hợp chặt chẽ giữa ba thành phần Model, View và Controller, tạo nên một quy trình xử lý yêu cầu và phản hồi rõ ràng, giúp ứng dụng hoạt động hiệu quả và dễ bảo trì.

Quy trình hoạt động:

- **Bước 1:** Nhận yêu cầu (Request), người dùng tương tác với giao diện (View) bằng cách nhấp chuột, nhập dữ liệu hoặc thực hiện các hành động khác. Yêu cầu này được gửi đến Controller.
- **Bước 2:** Xử lý yêu cầu (Controller), nhận yêu cầu và phân tích nó. Dựa trên yêu cầu, Controller sẽ tương tác với Model để lấy hoặc cập nhật dữ liệu cần thiết.
- **Bước 3:** Truy xuất/Cập nhật dữ liệu (Model), thực hiện các thao tác trên cơ sở dữ liệu (hoặc nguồn dữ liệu khác) để lấy hoặc cập nhật dữ liệu theo yêu cầu của Controller.
- **Bước 4:** Chuẩn bị dữ liệu (Model), chuẩn bị dữ liệu ở định dạng phù hợp để View có thể hiển thị.
- **Bước 5:** Trả về dữ liệu (Model), trả về dữ liệu đã chuẩn bị cho Controller.
- **Bước 6:** Cập nhật giao diện (Controller), nhận dữ liệu từ Model và chuyển nó đến View để hiển thị.
- **Bước 7:** Hiển thị kết quả (View), nhận dữ liệu từ Controller và hiển thị nó cho người dùng dưới dạng giao diện trực quan.

Mô hình MVC cung cấp một cách tiếp cận có cấu trúc và hiệu quả để xây dựng các ứng dụng phức tạp. Bằng cách phân tách rõ ràng các thành phần, MVC giúp cải thiện khả năng bảo trì, tái sử dụng mã và khả năng kiểm thử của ứng dụng.

2.6.3. Ưu điểm và nhược điểm của mô hình MVC

Mô hình MVC (Model-View-Controller) là một mẫu thiết kế phổ biến trong phát triển phần mềm, mang lại nhiều lợi ích nhưng cũng không tránh khỏi một số hạn chế.

Ưu điểm:

- **Phân tách rõ ràng:** MVC tách biệt ứng dụng thành ba thành phần độc lập (Model, View, Controller), giúp tổ chức mã nguồn tốt hơn, dễ đọc, dễ bảo trì và dễ kiểm tra hơn.
- **Tái sử dụng mã:** Các thành phần trong MVC có thể được tái sử dụng ở nhiều nơi trong ứng dụng hoặc trong các ứng dụng khác. Ví dụ, cùng một Model có thể được sử dụng bởi nhiều View khác nhau.
- **Dễ dàng bảo trì:** Khi có thay đổi trong một thành phần, bạn chỉ cần cập nhật thành phần đó mà không ảnh hưởng đến các thành phần khác. Điều này giúp việc bảo trì và nâng cấp ứng dụng trở nên dễ dàng hơn.
- **Phù hợp với phát triển nhóm:** MVC cho phép các thành viên trong nhóm làm việc song song trên các thành phần khác nhau mà không gây xung đột. Điều này giúp tăng tốc độ phát triển và giảm thiểu rủi ro.
- **Hỗ trợ SEO:** MVC giúp tạo ra các URL thân thiện với công cụ tìm kiếm, giúp cải thiện khả năng SEO của trang web.

Nhược điểm:

- **Độ phức tạp tăng lên:** Đối với các ứng dụng nhỏ, MVC có thể làm tăng độ phức tạp không cần thiết. Việc chia nhỏ ứng dụng thành nhiều thành phần có thể khiến việc tìm hiểu và làm quen với dự án trở nên khó khăn hơn.
- **Khó khăn trong việc gỡ lỗi:** Khi ứng dụng gặp lỗi, việc xác định nguyên nhân và vị trí của lỗi có thể trở nên khó khăn hơn do sự tương tác phức tạp giữa các thành phần.
- **Tốn thời gian phát triển ban đầu:** Việc thiết lập cấu trúc MVC ban đầu có thể tốn nhiều thời gian và công sức hơn so với các phương pháp khác.
- **Không phù hợp với mọi loại ứng dụng:** MVC không phải là giải pháp tối ưu cho tất cả các loại ứng dụng. Đối với các ứng dụng đơn giản hoặc các ứng dụng thời gian thực, MVC có thể không phải là lựa chọn tốt nhất.

Mô hình MVC mang lại nhiều lợi ích về tính tổ chức, tái sử dụng mã và khả năng bảo trì, nhưng cũng có thể làm tăng độ phức tạp và khó khăn trong việc gỡ lỗi. Việc lựa chọn sử dụng MVC hay không phụ thuộc vào quy mô và đặc thù của dự án. Đối với các dự án lớn và phức tạp, MVC có thể là một lựa chọn tốt để giúp tổ chức mã nguồn và dễ dàng bảo trì.

2.7. Tổng quan về RESTful API

2.7.1. Giới thiệu về RESTful API

RESTful API (Representational State Transfer Application Programming Interface) là một kiểu kiến trúc phần mềm được sử dụng để thiết kế các dịch vụ web. RESTful API sử dụng các nguyên tắc của giao thức HTTP để tạo ra các giao diện đơn giản và dễ sử dụng, cho phép các ứng dụng khác nhau giao tiếp và trao đổi dữ liệu với nhau qua mạng.

Các đặc điểm chính của RESTful API:

- **Client-server:** RESTful API tuân theo mô hình client-server, trong đó client gửi yêu cầu đến server và server trả về phản hồi.
- **Stateless:** Mỗi yêu cầu từ client đến server phải chứa tất cả thông tin cần thiết để server hiểu và xử lý yêu cầu đó. Server không lưu trữ bất kỳ trạng thái nào của client giữa các yêu cầu.
- **Cacheable:** Các phản hồi từ server có thể được lưu trữ tạm thời (cache) để cải thiện hiệu suất.
- **Layered system:** RESTful API có thể được xây dựng thành nhiều lớp, mỗi lớp có chức năng riêng biệt. Điều này giúp cải thiện khả năng mở rộng và bảo trì của hệ thống.
- **Code on demand (optional):** Server có thể mở rộng chức năng của client bằng cách gửi mã thực thi (code) cho client. Tuy nhiên, tính năng này là tùy chọn và không phải lúc nào cũng được sử dụng.
- **Uniform interface:** RESTful API sử dụng một giao diện thống nhất để giao tiếp giữa client và server. Điều này giúp đơn giản hóa việc phát triển và sử dụng API.

Các thành phần của RESTful API:

Tài nguyên (Resource): Bất kỳ thông tin nào có thể được đặt tên và được truy cập thông qua một URI (Uniform Resource Identifier). Ví dụ: một bài viết trên blog, một sản phẩm trong cửa hàng trực tuyến, hoặc một người dùng trong hệ thống.

Phương thức HTTP (HTTP Method): Các hành động mà client có thể thực hiện trên tài nguyên. Các phương thức HTTP phổ biến bao gồm:

- **GET:** Lấy thông tin về tài nguyên.
- **POST:** Tạo một tài nguyên mới.
- **PUT:** Cập nhật toàn bộ tài nguyên.
- **PATCH:** Cập nhật một phần của tài nguyên.

- **DELETE:** Xóa tài nguyên.

Định dạng dữ liệu (Data Format): Định dạng dữ liệu được sử dụng để trao đổi thông tin giữa client và server. Các định dạng dữ liệu phổ biến bao gồm JSON và XML.

Mã trạng thái HTTP (HTTP Status Code): Các mã số cho biết trạng thái của yêu cầu và phản hồi. Ví dụ:

- **200 OK:** Yêu cầu thành công.
- **404 Not Found:** Không tìm thấy tài nguyên.
- **500 Internal Server Error:** Lỗi máy chủ nội bộ.

RESTful API được sử dụng rộng rãi trong các ứng dụng web và di động hiện đại. Chúng cho phép các ứng dụng khác nhau giao tiếp và chia sẻ dữ liệu một cách dễ dàng và hiệu quả.

2.7.2. Các nguyên tắc của RESTful API

RESTful API tuân thủ một tập hợp các nguyên tắc thiết kế giúp đảm bảo tính nhất quán, khả năng mở rộng, đơn giản và dễ sử dụng. Dưới đây là các nguyên tắc chính của RESTful API:

Client-Server:

- RESTful API tuân theo mô hình kiến trúc client-server, trong đó client và server hoạt động độc lập.
- Client chịu trách nhiệm gửi yêu cầu đến server và nhận phản hồi, trong khi server chịu trách nhiệm xử lý yêu cầu và trả về kết quả.
- Sự phân tách này giúp cải thiện khả năng quản lý, mở rộng và bảo trì của hệ thống.

Stateless (Không trạng thái):

- Mỗi yêu cầu từ client đến server phải chứa đầy đủ thông tin cần thiết để server có thể hiểu và xử lý yêu cầu đó.
- Server không lưu trữ bất kỳ trạng thái nào của client giữa các yêu cầu.
- Điều này giúp đơn giản hóa việc triển khai server và cải thiện khả năng mở rộng của hệ thống.

Cacheable (Có thể lưu trữ):

- Các phản hồi từ server có thể được lưu trữ tạm thời (cache) ở phía client hoặc trên các máy chủ trung gian.
- Việc sử dụng cache giúp cải thiện hiệu suất và giảm tải cho server.

Layered System (Hệ thống phân lớp):

- RESTful API có thể được xây dựng thành nhiều lớp, mỗi lớp có chức năng riêng biệt.
- Các lớp này có thể bao gồm lớp trình bày (presentation layer), lớp nghiệp vụ (business logic layer) và lớp truy cập dữ liệu (data access layer).
- Kiến trúc phân lớp giúp cải thiện khả năng bảo trì, mở rộng và bảo mật của hệ thống.

Code on Demand (Tùy chọn):

- Server có thể mở rộng hoặc tùy chỉnh chức năng của client bằng cách gửi mã thực thi (code) cho client.
- Tuy nhiên, tính năng này là tùy chọn và không phải lúc nào cũng được sử dụng.
- Uniform Interface (Giao diện thống nhất):
- RESTful API sử dụng một giao diện thống nhất để giao tiếp giữa client và server.

Việc tuân thủ các nguyên tắc này giúp đảm bảo rằng RESTful API của bạn là nhất quán, dễ sử dụng, có khả năng mở rộng và bảo trì.

2.7.3. Ưu điểm và nhược điểm của RESTful API

RESTful API là một trong những phương pháp phổ biến nhất để xây dựng các dịch vụ web, nhưng cũng giống như mọi công nghệ khác, nó có cả ưu điểm và nhược điểm.

Ưu điểm:

Khả năng mở rộng (Scalability): RESTful API dễ dàng mở rộng để đáp ứng nhu cầu của các ứng dụng lớn và phức tạp. Do tính chất không trạng thái (stateless), việc thêm nhiều máy chủ để xử lý lưu lượng truy cập tăng cao là tương đối đơn giản.

Tính linh hoạt (Flexibility) và khả năng tương thích (Portability): RESTful API có thể được sử dụng với nhiều loại ngôn ngữ lập trình và nền tảng khác nhau. Điều này giúp các nhà phát triển dễ dàng tích hợp API vào các ứng dụng của họ.

Tính đơn giản (Simplicity) và dễ sử dụng (Ease of use): RESTful API sử dụng các phương thức HTTP tiêu chuẩn và định dạng dữ liệu phổ biến như JSON, giúp cho việc học và sử dụng trở nên dễ dàng hơn so với các giao thức khác như SOAP.

Hiệu suất (Performance): RESTful API thường có hiệu suất tốt hơn so với SOAP do sử dụng định dạng dữ liệu nhẹ hơn và không yêu cầu xử lý XML phức tạp.

Khả năng lưu trữ (Cacheability): Các phản hồi từ RESTful API có thể được lưu trữ tạm thời (cache) để cải thiện hiệu suất và giảm tải cho server.

Sự độc lập (Independence): Client và server hoạt động độc lập với nhau, cho phép chúng được phát triển và bảo trì riêng biệt. Điều này giúp tăng tính linh hoạt và khả năng mở rộng của hệ thống.

Nhược điểm:

Thiếu tiêu chuẩn chặt chẽ: Không có một tiêu chuẩn chính thức nào cho RESTful API, dẫn đến việc các API khác nhau có thể có cách triển khai khác nhau. Điều này có thể gây khó khăn cho việc tích hợp các API từ các nhà cung cấp khác nhau.

Vấn đề bảo mật: Do tính chất không trạng thái, RESTful API có thể gặp phải các vấn đề bảo mật nếu không được thiết kế và triển khai đúng cách. Các biện pháp bảo mật như xác thực, ủy quyền và mã hóa cần được xem xét kỹ lưỡng.

Không phù hợp với các trường hợp yêu cầu giao tiếp hai chiều: RESTful API hoạt động theo mô hình request-response, không phù hợp với các trường hợp yêu cầu giao tiếp hai chiều liên tục giữa client và server, như các ứng dụng chat hoặc ứng dụng thời gian thực.

RESTful API là một lựa chọn tuyệt vời cho việc xây dựng các dịch vụ web linh hoạt, có khả năng mở rộng và dễ sử dụng. Tuy nhiên, cần phải xem xét kỹ lưỡng các nhược điểm của nó và áp dụng các biện pháp bảo mật phù hợp để đảm bảo tính an toàn và hiệu quả của hệ thống.

2.7.4. Tầm quan trọng của RESTful API trong kiến trúc microservices

RESTful API đóng một vai trò quan trọng trong kiến trúc microservices, là "keo kết dính" giúp các dịch vụ độc lập giao tiếp và trao đổi dữ liệu với nhau. Trong kiến trúc microservices, ứng dụng được chia thành các dịch vụ nhỏ, độc lập, mỗi dịch vụ có thể được phát triển, triển khai và mở rộng một cách riêng biệt. RESTful API cung cấp một giao diện thống nhất và linh hoạt để các dịch vụ này có thể tương tác với nhau.

Dưới đây là một số lý do tại sao RESTful API lại quan trọng trong kiến trúc microservices:

Giao tiếp giữa các dịch vụ: RESTful API cho phép các microservices giao tiếp với nhau một cách hiệu quả và an toàn thông qua các yêu cầu HTTP tiêu chuẩn. Điều này giúp giảm sự phụ thuộc giữa các dịch vụ và tăng tính linh hoạt của hệ thống.

Tính độc lập của các dịch vụ: Nhờ vào việc sử dụng RESTful API, các microservices có thể được phát triển, triển khai và mở rộng một cách độc lập với nhau. Điều này giúp tăng tốc độ phát triển, giảm thiểu rủi ro và dễ dàng bảo trì hệ thống.

Khả năng mở rộng: RESTful API cho phép các microservices mở rộng một cách linh hoạt để đáp ứng nhu cầu của ứng dụng. Bạn có thể dễ dàng thêm hoặc bớt các microservices mà không ảnh hưởng đến hoạt động của toàn bộ hệ thống.

Tính linh hoạt: RESTful API hỗ trợ nhiều định dạng dữ liệu khác nhau như JSON và XML, giúp các microservices có thể trao đổi dữ liệu một cách linh hoạt.

Tiêu chuẩn hóa: RESTful API tuân theo các nguyên tắc và tiêu chuẩn chung, giúp đơn giản hóa việc tích hợp các microservices từ các nhà cung cấp khác nhau.

Khả năng phát hiện: RESTful API có thể tự mô tả và cung cấp thông tin về cấu trúc và chức năng của chúng thông qua siêu dữ liệu (metadata). Điều này giúp các nhà phát triển dễ dàng khám phá và sử dụng API.

2.8. Tổng quan về Jira

2.8.1. Giới thiệu về Jira

Jira là một công cụ theo dõi vấn đề và quản lý dự án được phát triển bởi Atlassian. Ban đầu, Jira được thiết kế để theo dõi lỗi và các vấn đề trong quá trình phát triển phần mềm, nhưng hiện nay nó đã được mở rộng để phục vụ cho nhiều mục đích khác nhau, bao gồm cả quản lý dự án Agile, theo dõi tác vụ, quản lý quy trình làm việc và hỗ trợ khách hàng.

Các tính năng chính của Jira:

- **Theo dõi vấn đề (Issue Tracking):** Jira cho phép bạn tạo, gán, theo dõi và quản lý các vấn đề (issues) trong dự án. Các vấn đề có thể là lỗi phần mềm, tác vụ, yêu cầu tính năng, hoặc bất kỳ công việc nào cần được theo dõi và giải quyết.
- **Quản lý dự án Agile:** Jira cung cấp các công cụ hỗ trợ cho các phương pháp Agile như Scrum và Kanban, bao gồm bảng Kanban, biểu đồ burn-down, và báo cáo tiến độ.
- **Tùy chỉnh quy trình làm việc (Workflow Customization):** Bạn có thể tùy chỉnh quy trình làm việc trong Jira để phù hợp với quy trình làm việc của tổ chức hoặc dự án của bạn.
- **Báo cáo và phân tích (Reporting and Analytics):** Jira cung cấp các báo cáo và công cụ phân tích để giúp bạn theo dõi tiến độ dự án, hiệu suất nhóm và các chỉ số quan trọng khác.
- **Tích hợp (Integrations):** Jira có thể tích hợp với nhiều công cụ và dịch vụ khác nhau như Bitbucket, Confluence, Slack, GitHub, và nhiều công cụ khác.

Các phiên bản của Jira:

- **Jira Software:** Phiên bản dành cho các nhóm phát triển phần mềm, hỗ trợ quản lý dự án Agile và theo dõi vấn đề.
- **Jira Service Management:** Phiên bản dành cho các nhóm hỗ trợ khách hàng và dịch vụ, hỗ trợ quản lý yêu cầu dịch vụ và theo dõi sự cố.
- **Jira Core:** Phiên bản cơ bản của Jira, cung cấp các tính năng theo dõi vấn đề và quản lý dự án chung.

2.8.2. Ứng dụng Jira trong quản lý Agile và Scrum

Jira là một công cụ mạnh mẽ hỗ trợ các nhóm phát triển phần mềm áp dụng và quản lý các phương pháp Agile, đặc biệt là Scrum, một cách hiệu quả. Dưới đây là một số ứng dụng cụ thể của Jira trong quản lý Agile và Scrum:

Quản lý Product Backlog:

- Jira cho phép Product Owner tạo và quản lý Product Backlog, bao gồm việc thêm, chỉnh sửa, xóa và sắp xếp thứ tự ưu tiên các User Story (câu chuyện người dùng) hoặc các mục công việc khác.
- Product Owner có thể sử dụng các tính năng như Epic (chủ đề lớn) và Story Point (điểm câu chuyện) để tổ chức và ước lượng công việc trong Backlog.
- Jira cung cấp các bộ lọc và chế độ xem khác nhau để Product Owner dễ dàng theo dõi và quản lý Backlog.

Lập kế hoạch Sprint:

- Trong Jira, nhóm phát triển có thể sử dụng Sprint Planning Board để lập kế hoạch cho Sprint tiếp theo.
- Nhóm có thể kéo các User Story từ Product Backlog vào Sprint Backlog và ước lượng công việc cần thiết để hoàn thành chúng.
- Jira giúp nhóm theo dõi tiến độ Sprint thông qua biểu đồ burn-down chart, hiển thị lượng công việc còn lại theo thời gian.

Theo dõi tiến độ Sprint:

- Jira cung cấp các bảng Scrum và Kanban để nhóm phát triển theo dõi tiến độ của từng User Story trong Sprint.
- Các thành viên trong nhóm có thể cập nhật trạng thái của các User Story, ghi nhận thời gian làm việc và thêm các bình luận.

- Jira tự động tính toán và hiển thị các chỉ số quan trọng như Velocity (tốc độ làm việc của nhóm) và Remaining Estimate (ước lượng công việc còn lại).

Tổ chức các sự kiện Scrum:

- Jira hỗ trợ việc tổ chức các sự kiện Scrum như Daily Scrum, Sprint Review và Sprint Retrospective.
- Nhóm có thể sử dụng Jira để lên lịch các cuộc họp, ghi chú và theo dõi các hành động cần thực hiện.

s

- Jira cung cấp các báo cáo và công cụ phân tích để giúp nhóm đánh giá hiệu suất và cải tiến quy trình làm việc.
- Các báo cáo phổ biến bao gồm Velocity Chart (biểu đồ tốc độ), Control Chart (biểu đồ kiểm soát), Cumulative Flow Diagram (biểu đồ luồng tích lũy) và Burndown Chart (biểu đồ đốt cháy).

Tích hợp với các công cụ khác: Jira có thể tích hợp với nhiều công cụ khác như Bitbucket, Confluence, Jenkins, Slack, v.v. để tạo ra một môi trường làm việc liền mạch và hiệu quả.

Jira là một công cụ không thể thiếu cho các nhóm áp dụng Scrum và Agile. Nó cung cấp một nền tảng mạnh mẽ và linh hoạt để quản lý dự án, theo dõi tiến độ và cải tiến quy trình làm việc.

2.8.3. Tính năng và lợi ích của Jira

Jira không chỉ là một công cụ theo dõi lỗi đơn thuần mà còn cung cấp nhiều tính năng mạnh mẽ, mang lại nhiều lợi ích cho việc quản lý dự án và quy trình làm việc. Dưới đây là một số tính năng và lợi ích nổi bật của Jira:

Tính năng:

- Theo dõi vấn đề (Issue Tracking):
- Tạo, gán, ưu tiên, theo dõi và quản lý các vấn đề (issues) trong dự án.
- Tùy chỉnh các trường (fields) và loại vấn đề (issue types) để phù hợp với quy trình làm việc của bạn.
- Liên kết các vấn đề với nhau để theo dõi các mối quan hệ phụ thuộc.
- Thêm các bình luận, tệp đính kèm và liên kết đến các vấn đề.

Quản lý dự án Agile:

- Hỗ trợ các phương pháp Agile như Scrum và Kanban với các bảng Scrum và Kanban trực quan.
- Lập kế hoạch Sprint, theo dõi tiến độ và quản lý Backlog.
- Tạo các báo cáo và biểu đồ để theo dõi hiệu suất dự án.

Tùy chỉnh quy trình làm việc (Workflow Customization):

- Tạo các quy trình làm việc tùy chỉnh để phản ánh quy trình làm việc thực tế của bạn.
- Tự động hóa các tác vụ và chuyển đổi trạng thái của vấn đề.

Báo cáo và phân tích (Reporting and Analytics):

- Tạo các báo cáo tùy chỉnh để theo dõi tiến độ dự án, hiệu suất nhóm và các chỉ số quan trọng khác.
- Sử dụng các biểu đồ và bảng để trực quan hóa dữ liệu.

Tích hợp (Integrations):

- Tích hợp với hàng trăm công cụ và dịch vụ khác nhau như Bitbucket, Confluence, Slack, GitHub, v.v.
- Mở rộng chức năng của Jira bằng các plugin và add-on từ Atlassian Marketplace.

Lợi ích:

Tăng cường khả năng hiển thị và minh bạch: Jira giúp bạn có cái nhìn tổng quan về tiến độ dự án, các vấn đề đang gặp phải và hiệu suất của nhóm.

Cải thiện sự cộng tác và giao tiếp: Jira tạo ra một nền tảng chung để các thành viên trong nhóm và các bên liên quan có thể giao tiếp, chia sẻ thông tin và làm việc cùng nhau.

Tăng hiệu quả và năng suất: Jira giúp tự động hóa các tác vụ, giảm thiểu các công việc thủ công và tối ưu hóa quy trình làm việc.

Đáp ứng nhanh chóng với thay đổi: Jira giúp bạn dễ dàng thích ứng với các thay đổi trong yêu cầu hoặc kế hoạch dự án.

Quản lý rủi ro: Jira giúp bạn xác định và quản lý các rủi ro tiềm ẩn trong dự án.

Cải thiện chất lượng sản phẩm: Jira giúp bạn theo dõi và giải quyết các vấn đề một cách kịp thời, từ đó nâng cao chất lượng sản phẩm cuối cùng.

Jira là một công cụ mạnh mẽ và linh hoạt, mang lại nhiều lợi ích cho việc quản lý dự án và quy trình làm việc. Với khả năng tùy biến cao, tích hợp đa dạng và cộng đồng hỗ trợ lớn, Jira là một lựa chọn tuyệt vời cho các tổ chức và doanh nghiệp thuộc mọi quy mô.

CHƯƠNG 3: XÁC ĐỊNH NHU CẦU

3.1. Đặc tả mục tiêu dự án

Mục tiêu chính của dự án là xây dựng một ứng dụng bán hàng trực tuyến hoàn chỉnh, đáp ứng các yêu cầu sau:

Về mặt chức năng:

- **Quản lý sản phẩm:** Cho phép người bán hàng thêm sản phẩm mới, chỉnh sửa thông tin sản phẩm (tên, mô tả, giá, hình ảnh...), và xóa sản phẩm.
- **Hiển thị sản phẩm:** Trình bày danh sách sản phẩm một cách trực quan, bao gồm hình ảnh, tên sản phẩm, giá cả, và các thông tin liên quan khác.
- **Tìm kiếm sản phẩm:** Cung cấp tính năng tìm kiếm sản phẩm theo từ khóa, danh mục, hoặc các tiêu chí khác để người dùng dễ dàng tìm thấy sản phẩm.
- **Xem chi tiết sản phẩm:** Hiển thị thông tin chi tiết về sản phẩm, bao gồm mô tả đầy đủ, hình ảnh, đánh giá của khách hàng, và các thông tin khác liên quan.
- **Đặt hàng:** Thu thập thông tin giao hàng và thanh toán của người dùng, tạo đơn hàng và xác nhận đơn hàng cho người dùng.

Việc xác định rõ ràng các mục tiêu này sẽ giúp nhóm phát triển tập trung vào các ưu tiên chính và đảm bảo rằng sản phẩm cuối cùng đáp ứng được nhu cầu của người dùng và mang lại giá trị kinh doanh cho doanh nghiệp.

3.2. Xác định personas

Để hiểu rõ hơn về người dùng mục tiêu, tiến hành xác định các personas đại diện cho các nhóm người dùng khác nhau:

Persona 1: Người mua hàng trực tuyến

- **Nhu cầu:** Tìm kiếm sản phẩm dễ dàng, so sánh giá cả, thanh toán an toàn, giao hàng nhanh chóng.
- **Mong đợi:** Giao diện trực quan, thông tin sản phẩm đầy đủ, hỗ trợ khách hàng tốt.

Persona 2: Người bán hàng

- **Nhu cầu:** Quản lý sản phẩm hiệu quả, theo dõi đơn hàng, tương tác với khách hàng.
- **Mong đợi:** Công cụ quản lý dễ sử dụng, báo cáo chi tiết, tiếp cận khách hàng tiềm năng.

3.3. Xác định các user stories

Dựa trên các personas, có thể xác định các user stories để mô tả các tính năng và tương tác của người dùng với ứng dụng như sau:

Người mua hàng:

- Có thể tìm kiếm sản phẩm theo từ khóa.
- Có thể xem chi tiết sản phẩm để biết thêm thông tin trước khi quyết định mua.
- Có thể đặt hàng.

Người bán hàng:

- Thêm sản phẩm mới vào cửa hàng của mình để khách hàng có thể mua.
- Cập nhật thông tin sản phẩm để đảm bảo thông tin chính xác.
- Xóa sản phẩm khi hết hàng.

3.4. Xác định các product backlog

Dựa trên các user stories đã xác định trong mục **3.3**, có thể xác định các product backlog cho dự án phát triển ứng dụng bán hàng trực tuyến như sau:

- Cài đặt Node JS và Express JS
- Cấu hình Nodemon và inspector
- Cấu hình Template Handlebars
- Thiết lập cấu hình các file tĩnh và css
- Cấu hình thư viện Bootstrap 4
- Xây dựng Controller cơ bản cho trang chủ, trang tìm kiếm và login
- Xây dựng View cơ bản cho trang chủ, trang tìm kiếm và login
- Cấu hình kết nối với cơ sở dữ liệu MongoDB
- Xây dựng thành phần Model Product
- Xây dựng logic hiển thị sản phẩm từ DB
- Xây dựng Controller xem chi tiết sản phẩm
- Xây dựng giao diện trang chi tiết sản phẩm
- Xây dựng Controller thêm mới sản phẩm
- Xây dựng giao diện trang thêm mới sản phẩm
- Xây dựng Controller tìm kiếm sản phẩm
- Xây dựng giao diện trang tìm kiếm sản phẩm
- Chỉnh sửa một số lỗi trong dự án

- Xây dựng Model Order
- Xây dựng Controller mua sản phẩm
- Xây dựng giao diện trang mua sản phẩm
- Xây dựng Model Login
- Xử lý logic login trang đăng nhập
- Xây dựng giao diện trang đăng nhập, thêm chức năng sửa, xóa sản phẩm
- Sửa lỗi upload hình ảnh, triển khai ứng dụng trên Docker và format code

3.5. Xác định các nhu cầu phi tính năng

Sau khi xác định các product backlog, tiến hành xét các yêu cầu phi chức năng và đánh giá dựa trên các tiêu chuẩn đã đề ra:

- **Hiệu suất:** Ứng dụng phải hoạt động nhanh chóng, thời gian tải trang ngắn, và đáp ứng tốt với lượng truy cập lớn.
- **Bảo mật:** Đảm bảo an toàn thông tin người dùng, thông tin sản phẩm, và các giao dịch thanh toán.
- **Tính khả dụng:** Ứng dụng phải dễ dàng sử dụng, giao diện trực quan và thân thiện với người dùng.
- **Khả năng mở rộng:** Thiết kế ứng dụng linh hoạt, dễ dàng mở rộng và thêm các tính năng mới trong tương lai.
- **Tương thích:** Ứng dụng phải hoạt động tốt trên các trình duyệt và thiết bị khác nhau.

CHƯƠNG 4: LẬP KẾ HOẠCH SCRUM

4.1 Sắp xếp thứ tự ưu tiên các product backlog

Việc sắp xếp thứ tự ưu tiên các product backlog được thực hiện dựa trên mức độ quan trọng và cấp thiết của từng tính năng đối với người dùng và hệ thống. Dưới đây là thứ tự ưu tiên các product backlog của dự án:

- **IS1:** Cài đặt Node JS và Express JS
- **IS2:** Cấu hình Nodemon và inspector
- **IS3:** Cấu hình Template Handlebars
- **IS4:** Thiết lập cấu hình các file tĩnh và css
- **IS5:** Cấu hình thư viện Bootstrap 4
- **IS6:** Xây dựng Controller cơ bản cho trang chủ, trang tìm kiếm và login
- **IS7:** Xây dựng View cơ bản cho trang chủ, trang tìm kiếm và login
- **IS8:** Cấu hình kết nối với cơ sở dữ liệu MongoDB
- **IS9:** Xây dựng thành phần Model Product
- **IS10:** Xây dựng logic hiển thị sản phẩm từ DB
- **IS11:** Xây dựng Controller xem chi tiết sản phẩm
- **IS12:** Xây dựng giao diện trang chi tiết sản phẩm
- **IS13:** Xây dựng Controller thêm mới sản phẩm
- **IS14:** Xây dựng giao diện trang thêm mới sản phẩm
- **IS15:** Xây dựng Controller tìm kiếm sản phẩm
- **IS16:** Xây dựng giao diện trang tìm kiếm sản phẩm
- **IS17:** Chỉnh sửa một số lỗi trong dự án
- **IS18:** Xây dựng Model Order
- **IS19:** Xây dựng Controller mua sản phẩm
- **IS20:** Xây dựng giao diện trang mua sản phẩm
- **IS21:** Xây dựng Model Login
- **IS22:** Xử lý logic login trang đăng nhập
- **IS23:** Xây dựng giao diện trang đăng nhập, thêm chức năng sửa, xóa sản phẩm
- **IS24:** Sửa lỗi upload hình ảnh, triển khai ứng dụng trên Docker và format code

Thứ tự ưu tiên này được xác định dựa trên các yếu tố sau:

- **Tính thiết yếu:** Các tính năng cốt lõi như quản lý sản phẩm, đặt hàng, đăng nhập được ưu tiên hàng đầu vì chúng là những chức năng cơ bản của một ứng dụng bán hàng trực tuyến.
- **Giá trị kinh doanh:** Các tính năng mang lại giá trị kinh doanh cao như tìm kiếm sản phẩm, xem chi tiết sản phẩm, quản lý đơn hàng được ưu tiên hơn các tính năng khác.
- **Mức độ phức tạp:** Các tính năng đơn giản và dễ thực hiện được ưu tiên hơn các tính năng phức tạp để đảm bảo tiến độ dự án.

Việc sắp xếp thứ tự ưu tiên các product backlog giúp nhóm phát triển tập trung vào các công việc quan trọng nhất, đảm bảo rằng sản phẩm được phát triển theo đúng tiến độ và đáp ứng được nhu cầu của người dùng.

4.2 Xác định các sprint và chọn products backlogs

Dự án được chia thành 4 sprint, mỗi sprint kéo dài 1 tuần (từ thứ Hai đến Chủ Nhật) và được chọn products backlogs để đưa vào sprint như sau:

- **Sprint 1:** Cài đặt môi trường, xây dựng cấu trúc cơ bản của ứng dụng (IS1, IS2, IS3, IS4, IS5, IS6).
- **Sprint 2:** Xây dựng các chức năng hiển thị và tìm kiếm sản phẩm (IS7, IS8, IS9, IS10, IS11, IS12).
- **Sprint 3:** Xây dựng các chức năng quản lý sản phẩm và một số xử lý lỗi cơ bản (IS13, IS14, IS15, IS16, IS17, IS18).
- **Sprint 4:** Xây dựng chức năng đăng nhập và hoàn thiện ứng dụng (IS19, IS20, IS21, IS22, IS23, IS24).

Việc chia thành các sprint nhỏ giúp nhóm dễ dàng quản lý tiến độ và tập trung vào việc hoàn thành từng phần của dự án. Các product backlogs được sắp xếp vào các sprint dựa trên mức độ ưu tiên và sự phụ thuộc lẫn nhau giữa các tính năng.

Các sprint được xác định cụ thể như sau:

ID	Issue	Person	Points	Sprint	Start	End
IS1	Cài đặt Node JS và Express JS	Thành Tân	1	Sprint 1	01/05/2024	02/05/2024
IS2	Cấu hình Nodemon và inspector	Hoàng Thanh	2	Sprint 1	02/05/2024	03/05/2024

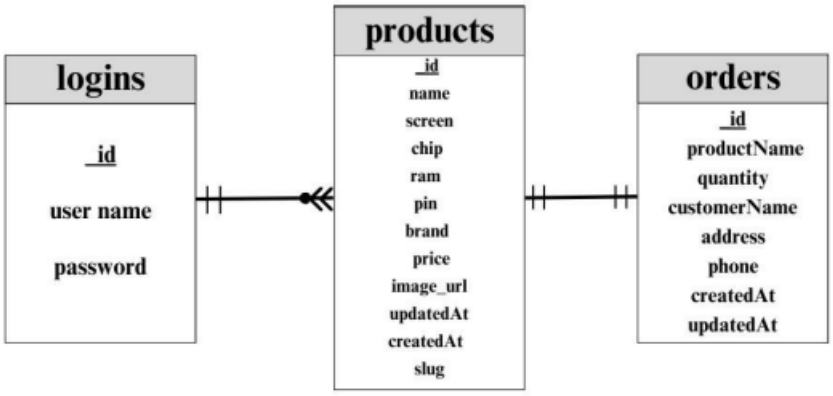
IS3	Cấu hình Template Handlebars	Hoàng Khang	1	Sprint 1	03/05/2024	04/05/2024
IS4	Thiết lập cấu hình các file tĩnh và css	Thành Tân	3	Sprint 1	04/05/2024	05/05/2024
IS5	Cấu hình thư viện Bootstrap 4	Hoàng Thanh	2	Sprint 1	05/05/2024	06/05/2024
IS6	Xây dựng Controller cơ bản cho trang chủ, trang tìm kiếm và login	Hoàng Khang	3	Sprint 1	06/05/2024	07/05/2024
IS7	Xây dựng View cơ bản cho trang chủ, trang tìm kiếm và login	Thành Tân	2	Sprint 2	08/05/2024	09/05/2024
IS8	Cấu hình kết nối với cơ sở dữ liệu MongoDB	Hoàng Thanh	2	Sprint 2	09/05/2024	10/05/2024
IS9	Xây dựng thành phần Model Product	Hoàng Khang	1	Sprint 2	10/05/2024	11/05/2024
IS10	Xây dựng logic hiển thị sản phẩm từ DB	Thành Tân	2	Sprint 2	11/05/2024	12/05/2024
IS11	Xây dựng Controller xem chi tiết sản phẩm	Quốc Trọng	2	Sprint 2	12/05/2024	13/05/2024
IS12	Xây dựng giao diện trang chi tiết sản phẩm	Hoàng Khang	3	Sprint 2	13/05/2024	14/05/2024
IS13	Xây dựng Controller thêm mới sản phẩm	Hoàng Tân	2	Sprint 3	15/05/2024	16/05/2024
IS14	Xây dựng giao diện trang thêm mới sản phẩm	Hoàng Thanh	2	Sprint 3	16/05/2024	17/05/2024
IS15	Xây dựng Controller tìm kiếm sản phẩm	Hoàng Khang	2	Sprint 3	17/05/2024	18/05/2024
IS16	Xây dựng giao diện trang tìm kiếm sản phẩm	Thành Tân	2	Sprint 3	18/05/2024	19/05/2024
IS17	Chỉnh sửa một số lỗi trong dự án	Quốc Trọng	2	Sprint 3	19/05/2024	20/05/2024
IS18	Xây dựng Model Order	Hoàng Khang	2	Sprint 3	20/05/2024	21/05/2024

IS19	Xây dựng Controller mua sản phẩm	Thành Tân	3	Sprint 4	22/05/2024	23/05/2024
IS20	Xây dựng giao diện trang mua sản phẩm	Hoàng Thanh	3	Sprint 4	23/05/2024	24/05/2024
IS21	Xây dựng Model Login	Hoàng Khang	1	Sprint 4	24/05/2024	25/05/2024
IS22	Xử lý logic login trang đăng nhập	Thành Tân	1	Sprint 4	25/05/2024	26/05/2024
IS23	Xây dựng giao diện trang đăng nhập, thêm chức năng sửa, xóa sản phẩm	Hoàng Thanh	1	Sprint 4	26/05/2024 Đến 27/05/2024	
IS24	Sửa lỗi upload hình ảnh, triển khai ứng dụng trên Docker và format code	Hoàng Khang	3	Sprint 4		

Bảng 4. 1. Bảng phân nhiệm vụ cho các thành viên

CHƯƠNG 5: HIỆN THỰC HÓA KẾ HOẠCH

5.1. Mô hình dữ liệu



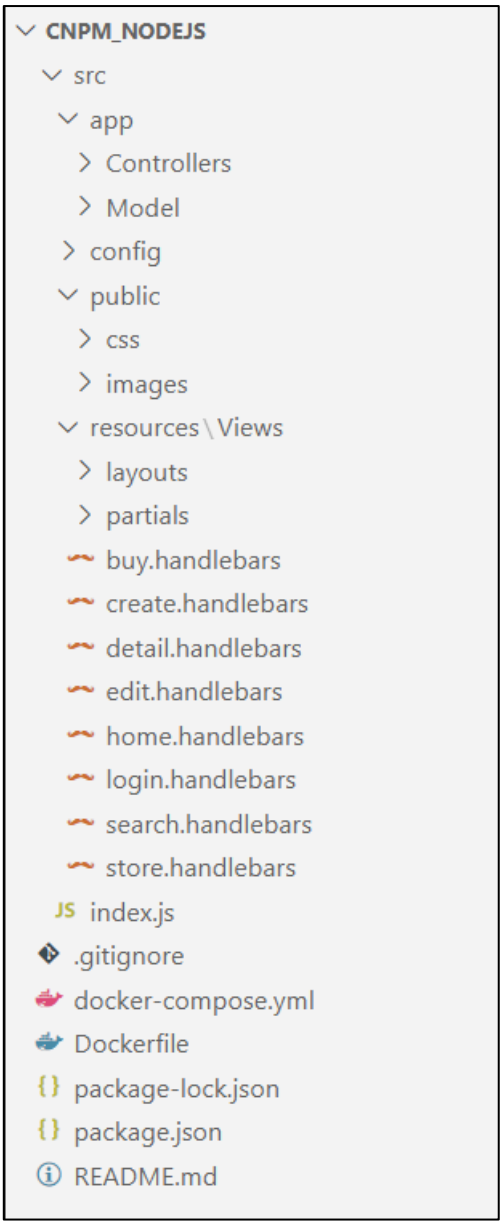
Hình 5. 1. Minh họa mô hình dữ liệu

Collection "logins": chứa thông tin người dùng bao gồm tên đăng nhập và mật khẩu, mỗi document trong bảng đại diện cho một người dùng.

Collection "products": bảng chứa thông tin của sản phẩm như tên sản phẩm, thông tin chi tiết của sản phẩm, giá bán, hình ảnh và ngày tạo, ngày cập nhật sản phẩm. Mỗi document trong bảng đại diện cho một loại sản phẩm.

Collection "orders": Bảng này liên kết thông tin từ bảng "products" cho biết sản phẩm này cho đơn hàng nào. Bảng chứa các thông tin bao gồm tên sản phẩm, số lượng, thông tin chi tiết người mua hàng, ngày tạo và ngày cập nhật. Mỗi người dùng có thể đặt nhiều đơn hàng, nhưng mỗi đơn hàng chỉ thuộc về một người dùng. Mỗi sản phẩm có thể xuất hiện trong nhiều đơn hàng nhưng mỗi đơn hàng chỉ chứa một sản phẩm.

5.2. Cấu trúc dự án



Hình 5. 2. Cấu trúc dự án

Thư mục public\images: nơi lưu trữ hình ảnh được sử dụng trong dự án.

Thư mục src\app: chứa các thư mục con như:

- **Model:** định nghĩa câu trúc dữ liệu. config: kết nối với cơ sở dữ liệu MongoDB.
- **Controller:** Điều khiển sự tương tác giữa hai thành phần là Model và Views.

Thư mục resoures\Views: hiển thị dữ liệu, giao diện cho người dùng.

File index.js: Là nơi định nghĩa các route, khởi tạo các thư viện cần thiết cho dự án, cấu hình middleware, kết nối cơ sở dữ liệu.

File docker-compose.yml: Tệp cấu hình để chạy ứng dụng trong môi trường Docker Compose.

File Dockerfile: Tệp cấu hình để xây dựng một image Docker cho ứng dụng.

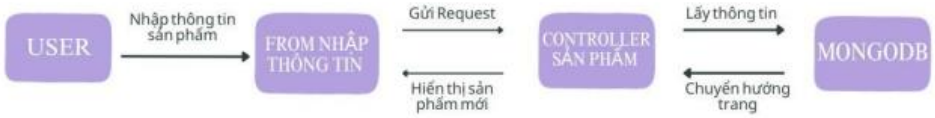
File package.json và package-lock.json: Các tệp quản lý các dependencies có trong dự án.

5.3. Sơ đồ mô tả chức năng tìm kiếm sản phẩm

Hình 5. 3. Sơ đồ mô tả chức năng tìm kiếm sản phẩm

Người dùng sẽ nhập từ khóa vào form tìm kiếm, sau đó form sẽ gửi yêu cầu đến Controller tìm kiếm tương ứng bằng phương thức GET. Controller tiếp nhận thông tin sử lý và truy xuất thông tin tương ứng từ cơ sở dữ liệu MongoDB. Cơ sở dữ liệu sẽ trả về chuỗi JSON thông tin sản phẩm tìm kiếm và Controller tiếp nhận sẽ chuyển người dùng đến trang tìm kiếm sản phẩm và hiển thị nội dung người dùng cần tìm kiếm

5.4. Sơ đồ mô tả chức năng thêm mới sản phẩm



Hình 5. 4. Sơ đồ mô tả chức năng thêm mới sản phẩm

Người dùng nhập thông tin cần thêm sản phẩm mới bao gồm tên, mô tả sản phẩm, hình ảnh, giá bán vào form nhập thông tin sản phẩm. Sau đó form sẽ gửi yêu cầu đến Controller tương ứng với phương thức là POST để đưa chuỗi JSON từ form đến cơ sở dữ liệu. MongoDB sẽ tiếp nhận thông tin và lưu vào các documents của collection tương ứng. Sau khi lưu vào cơ sở dữ liệu người dùng sẽ được chuyển hướng đến trang hiển thị giao diện của sản phẩm.

5.5. Sơ đồ mô tả chức năng cập nhật sản phẩm

Hình 5. 5. Sơ đồ mô tả chức năng cập nhật sản phẩm

Giao diện cập nhập thông tin sản phẩm sẽ hiển thị toàn bộ danh sách các sản phẩm, người dùng sẽ chọn sản phẩm cần cập nhật sau đó sẽ cập nhật lại thông tin của sản phẩm bao gồm tên, mô tả sản phẩm, hình ảnh, giá bán. Khi người dùng ấn Cập Nhật thì form sẽ gửi yêu cầu đến Controller cập nhật với phương thức là UPDATE để đưa chuỗi JSON từ form đến cơ sở dữ liệu. Sau đó người dùng sẽ chuyển đến trang cập nhật sản phẩm để tiếp tục cập nhật sản phẩm.

5.6. Sơ đồ mô tả chức năng xóa sản phẩm

Hình 5. 6. Sơ đồ mô tả chứng năng xóa sản phẩm

Giao diện xóa sản phẩm sẽ hiện thị danh sách các sản phẩm. Người dùng lựa chọn sản phẩm cần xóa. Sau khi người dùng xác nhận xóa dữ liệu trong form sẽ được gửi yêu cầu đến Controller xóa sản phẩm với phương thức là DELETE để thực hiện việc xóa sản phẩm trong cơ sở dữ liệu. Sau khi xóa dữ liệu thành công, trang danh sách sản phẩm sẽ được cập nhật lại các sản phẩm hiện tại.

5.7. Sprint

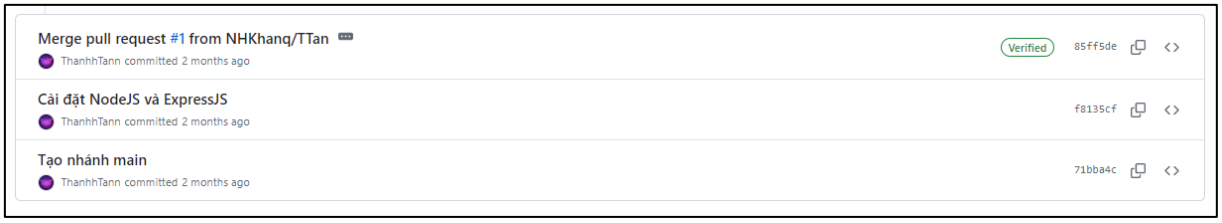
5.7.1. Sprint 1

Trong sprint này, nhóm tập trung vào việc cài đặt môi trường phát triển và xây dựng các thành phần cơ bản của ứng dụng như:

- Cài đặt và cấu hình NodeJs, ExpressJS
- Cấu hình Nodemon, Inspector
- Cấu hình thư viện Bootstrap 4
- Xây dựng Controller cho trang chủ, tìm kiếm, đăng nhập

Hình 5. 7. Minh họa Spint 1

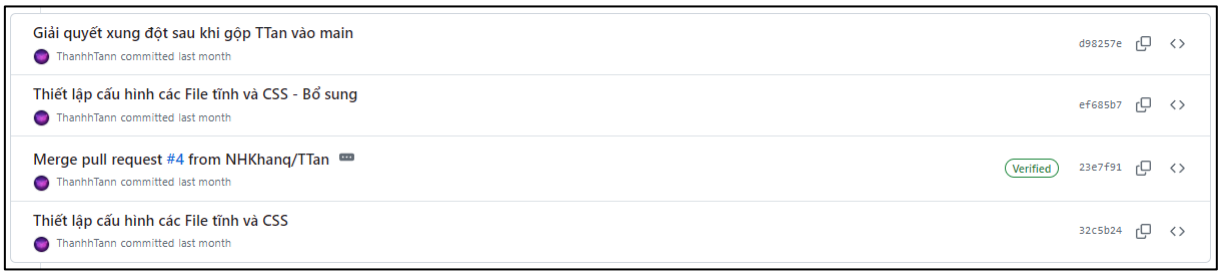
Tiếp theo là các commit của các thành viên theo sự phân công được thiết lập sẵn trên Jira:



Hình 5. 8. Minh họa commit cài đặt Node JS và Express JS

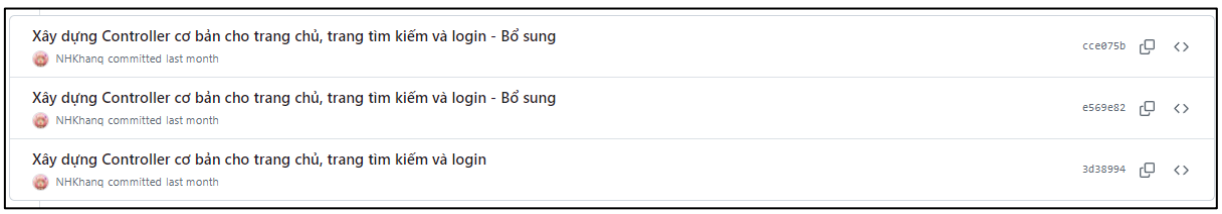
Hình 5. 9. Minh họa commit cài đặt Nodemon va Inspector

Hình 5. 10. Minh họa commit cấu hình Template Handlebars

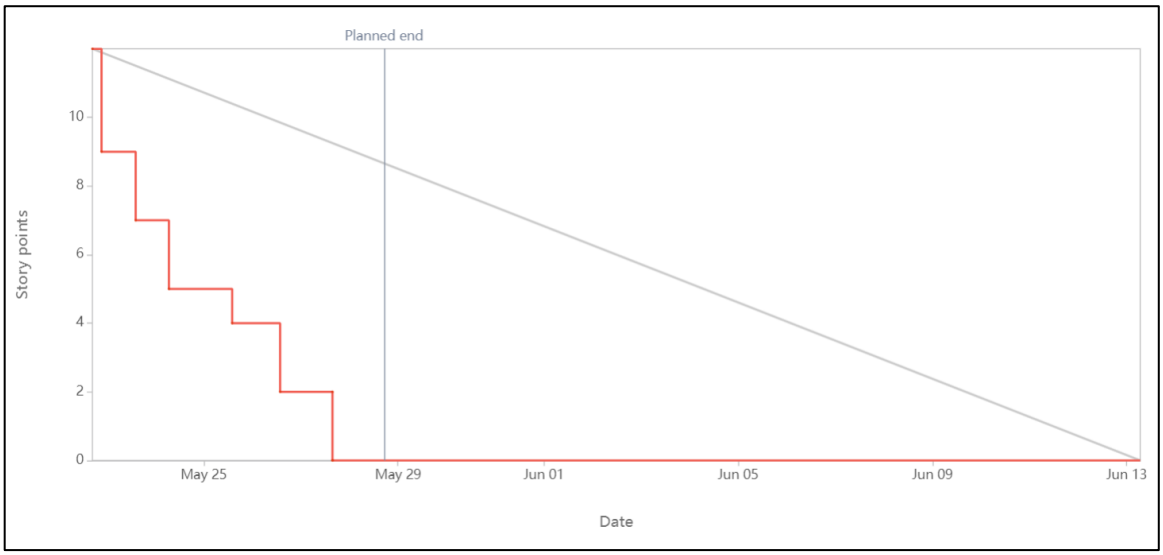


Hình 5. 11. Minh họa commit thiết lập cấu hình các File tĩnh và CSS

Hình 5. 12. Minh họa commit cấu hình sử dụng thư viện Bootstrap 4



Hình 5. 13. Minh họa commit xây dựng Controller cơ bản cho các trang



Hình 5. 14. Minh họa burndown chart sprint 1

Kết thúc Sprint 1, nhóm đã hoàn thành việc cài đặt môi trường phát triển NodeJS, ExpressJS, cấu hình Nodemon, inspector, Template Handlebars, Bootstrap 4 và các file tĩnh, cũng như xây dựng các controller và view cơ bản cho trang chủ, trang tìm kiếm và login.

5.7.2. Sprint 2

Trong sprint này, nhóm đã hoàn thành tất cả các công việc được giao, bao gồm:

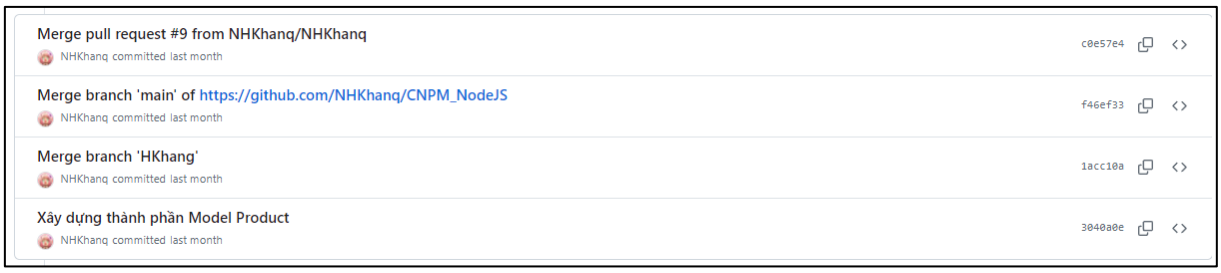
- Xây dựng View cơ bản cho trang chủ, trang tìm kiếm và đăng nhập
- Cấu hình kết nối với cơ sở dữ liệu MongoDB
- Xây dựng thành phần Model Product
- Xây dựng logic hiển thị sản phẩm từ DB
- Xây dựng Controller xem chi tiết sản phẩm
- Xây dựng giao diện trang chi tiết sản phẩm

Hình 5. 15. Minh họa Sprint 2

Tiếp theo là các commit của các thành viên theo sự phân công được thiết lập sẵn trên Jira:

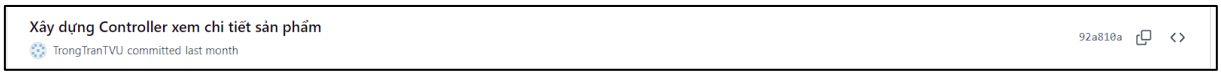
Hình 5. 16. Minh họa commit xây dựng View cơ bản cho các trang

Hình 5. 17 Minh họa commit cấu hình kết nối với cơ sở dữ liệu MongoDB



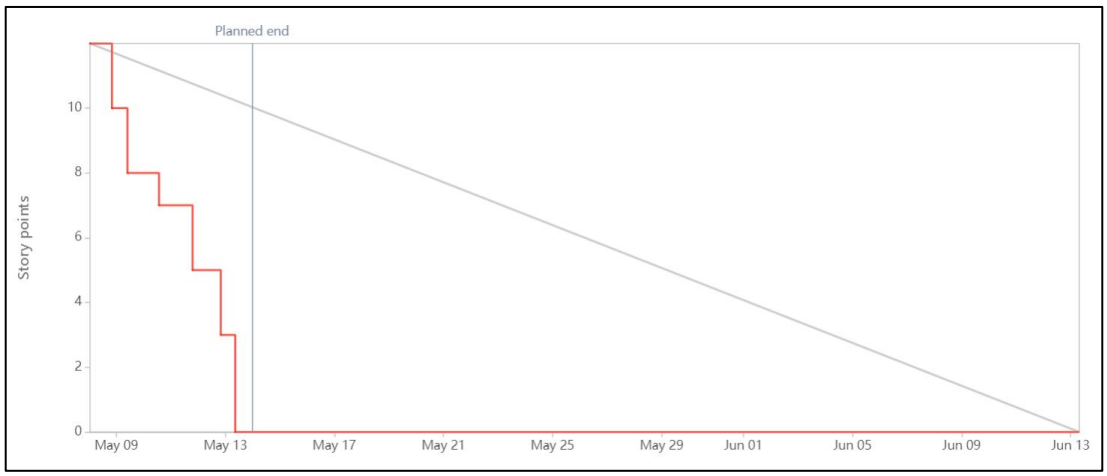
Hình 5. 18. Minh họa commit xây dựng thành phần Model Product

Hình 5. 19. Minh họa commit xây dựng logic hiển thị sản phẩm từ DB



Hình 5. 20. Minh họa commit xây dựng Controller xem chi tiết sản phẩm

Hình 5. 21. Minh họa commit xây dựng giao diện trang chi tiết sản phẩm



Hình 5. 22. Minh họa burndown chart sprint 2

Hình 5. 23. Minh họa giao diện thực thi Sprint 2

Kết thúc sprint 2, trang web đã có thể hiển thị danh sách sản phẩm, cho phép người dùng xem chi tiết sản phẩm và tìm kiếm sản phẩm. Nhóm đã hoàn thành tốt các công việc được giao và đảm bảo tiến độ của sprint.

Mặc dù đã hoàn thành tất cả các công việc, nhóm vẫn sẽ tiếp tục tối ưu hóa hiệu suất truy vấn cơ sở dữ liệu và cải thiện giao diện người dùng trong các sprint tiếp theo.

5.7.3. Sprint 3

Trong Sprint 3, nhóm đã hoàn thành tất cả 6 công việc được giao, bao gồm:

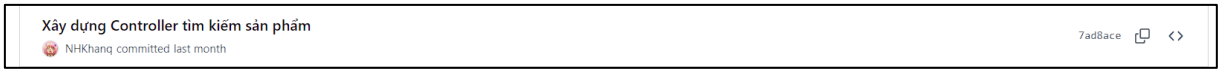
- Xây dựng Controller thêm mới sản phẩm
- Xây dựng giao diện trang thêm mới sản phẩm
- Xây dựng Controller tìm kiếm sản phẩm
- Xây dựng giao diện trang tìm kiếm sản phẩm
- Chỉnh sửa một số lỗi trong dự án
- Xây dựng Model Order

Hình 5. 24. Minh họa Sprint 3

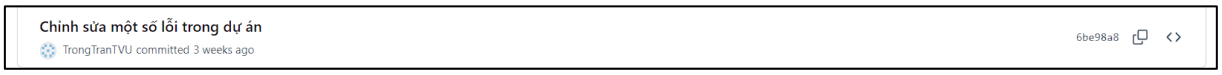
Tiếp theo là các commit của các thành viên theo sự phân công được thiết lập sẵn trên Jira:

Hình 5. 25. Minh họa commit xây dựng Controller thêm mới sản phẩm

Hình 5. 26. Minh họa commit xây dựng giao diện trang thêm mới sản phẩm

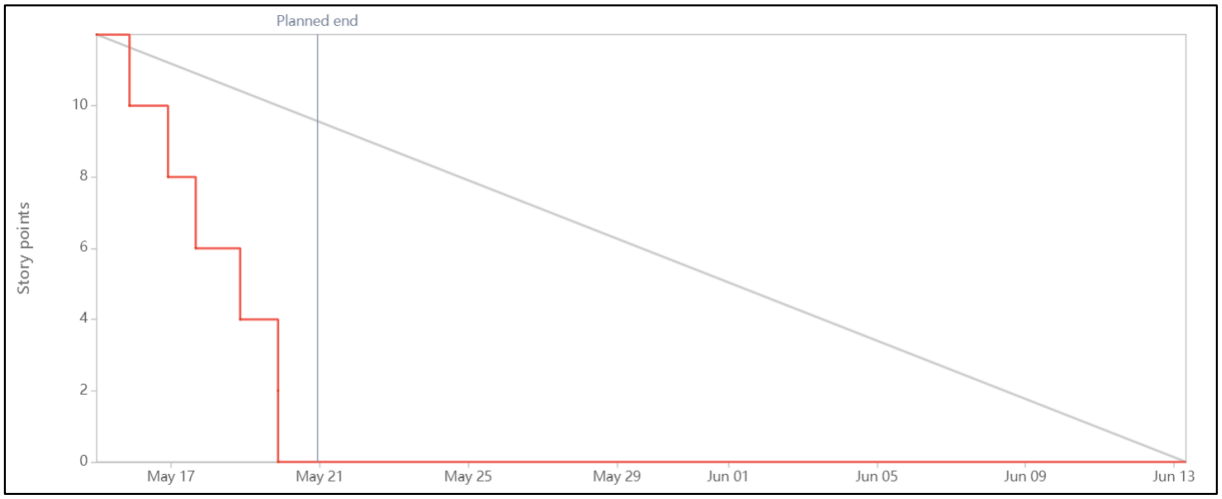


Hình 5. 27. Minh họa commit xây dựng Controller tìm kiếm sản phẩm



Hình 5. 28. Minh họa commit chỉnh sửa một số lỗi trong dự án

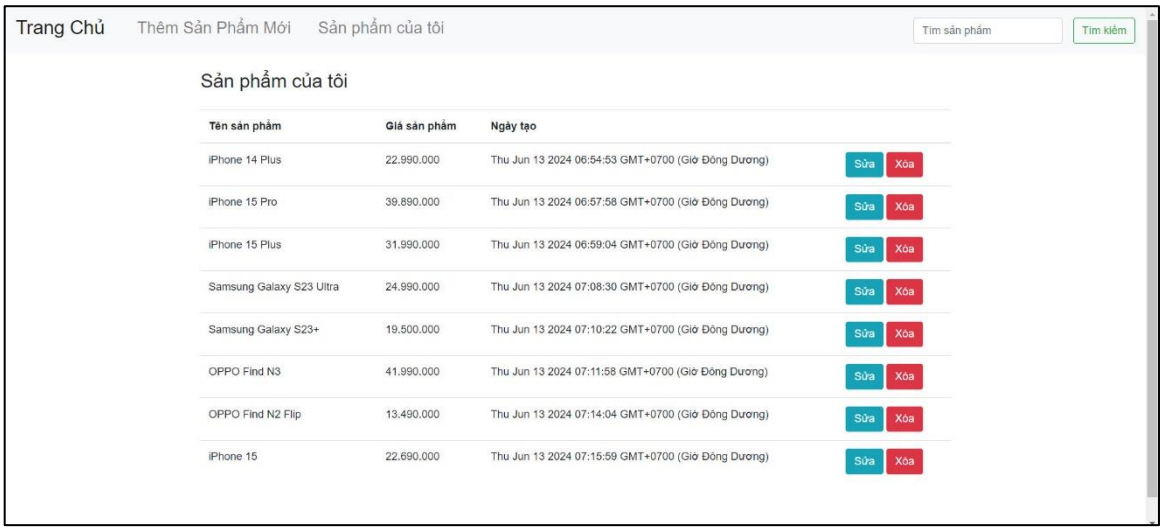
Hình 5. 29. Minh họa commit xây dựng Model Order



Hình 5. 30. Minh họa burndown chart sprint 3

Sprint 3 tập trung vào việc xây dựng các thành phần tìm kiếm và mua sản phẩm, đồng thời chỉnh sửa một số lỗi còn tồn đọng trong quá trình phát triển. Kết thúc sprint, nhóm đã hoàn thành tất cả các công việc được giao trong khoảng thời gian dự kiến và không có công việc nào bị trễ hạn.

Tuy nhiên, trong quá trình thực hiện sprint 3, nhóm gặp một số khó khăn trong việc tích hợp các thành phần mới với các thành phần đã có. Nhưng nhóm đã nỗ lực để giải quyết vấn đề và hoàn thành công việc đúng hạn.



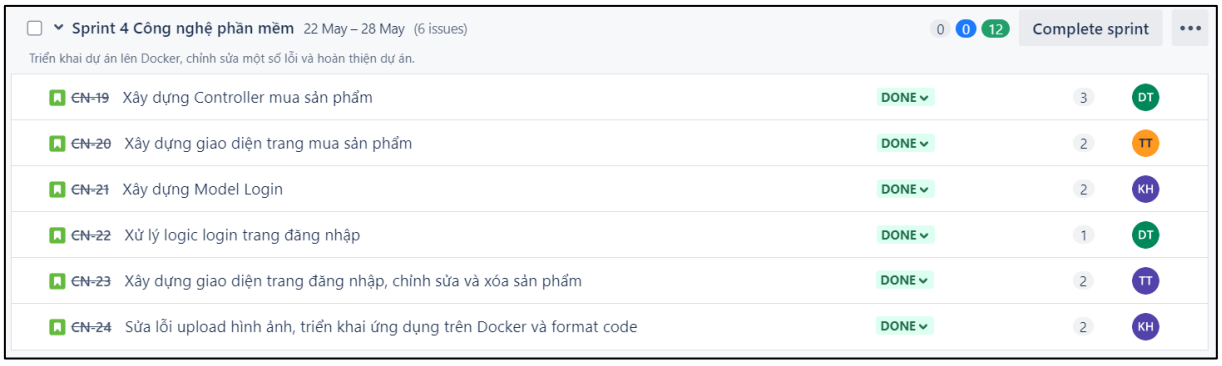
Hình 5. 31. Minh họa giao diện thực thi Sprint 3

Kết quả của sprint 3 là một trang web đã có thể cho phép người dùng tìm kiếm sản phẩm và người bán có thể thêm sản phẩm mới.

5.7.4. Sprint 4

Trong Sprint 4, nhóm đã hoàn thành tất cả 6 công việc được giao, bao gồm:

- Xây dựng Controller mua sản phẩm
- Xây dựng giao diện trang mua sản phẩm
- Xây dựng Model Login
- Xử lý logic login trang đăng nhập
- Xây dựng giao diện trang đăng nhập, chỉnh sửa và xóa sản phẩm
- Sửa lỗi upload hình ảnh, triển khai ứng dụng trên Docker và format code



Hình 5. 32. Minh họa Sprint 4

Tiếp theo là các commit của các thành viên theo sự phân công được thiết lập sẵn trên Jira:

Hình 5. 33. Minh họa commit xây dựng Controller mua sản phẩm

Hình 5. 34. Minh họa commit xây dựng giao diện mua sản phẩm

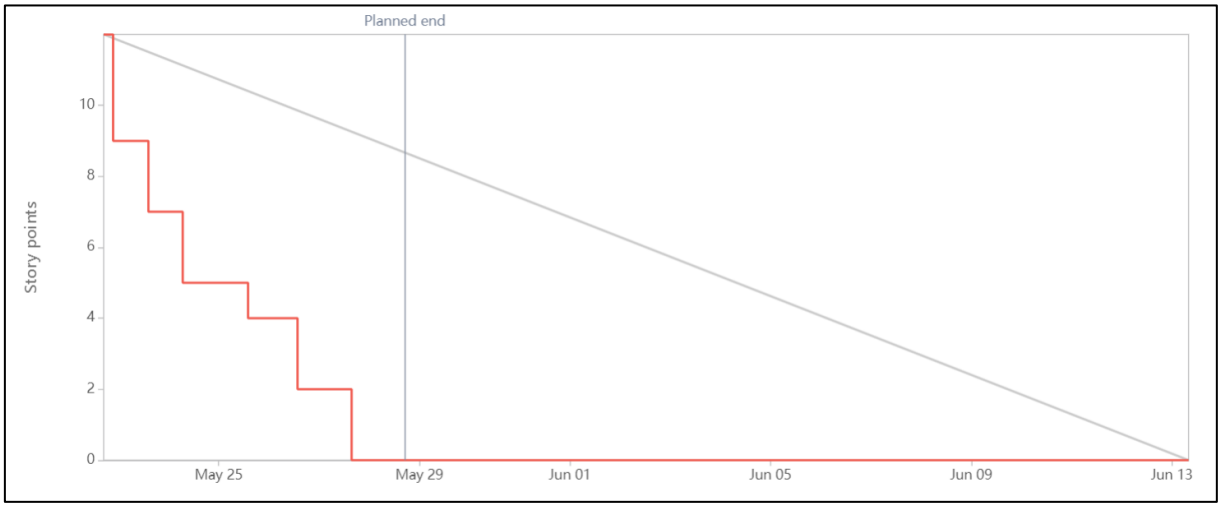
Hình 5. 35. Minh họa commit xây dựng Model Login

Hình 5. 36. Minh họa commit xử lý logic trang đăng nhập

Hình 5. 37. Minh họa commit xây dựng giao diện trang đăng nhập

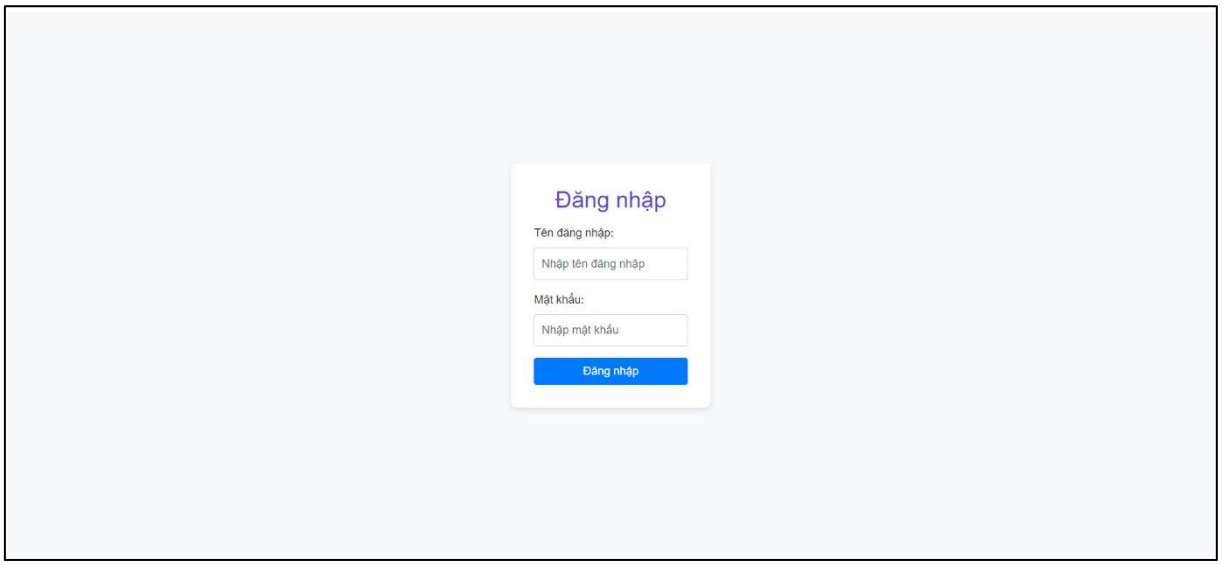
Hình 5. 38. Minh họa commit sửa lỗi và triển khai docker

Sprint 4 tập trung vào việc hoàn thiện các tính năng còn lại của ứng dụng, bao gồm chức năng mua hàng, đăng nhập, và triển khai ứng dụng. Nhóm đã hoàn thành tất cả các công việc đúng hạn và đạt chất lượng tốt.



Hình 5. 39. Minh họa burndown chart sprint 4

Mặc dù không có vấn đề lớn phát sinh trong quá trình thực hiện sprint 4, nhóm vẫn gặp một số khó khăn nhỏ trong việc triển khai ứng dụng trên Docker. Tuy nhiên, nhóm đã nhanh chóng tìm ra giải pháp và khắc phục vấn đề.



Hình 5. 40. Minh họa giao diện thực thi Sprint 4

Kết quả của sprint 4 là một trang web hoàn chỉnh, đã được triển khai trên Docker và sẵn sàng để đưa vào sử dụng. Nhóm đã hoàn thành dự án thành công và đạt được các mục tiêu đã đề ra.

CHƯƠNG 6: KẾT LUẬN

6.1. Kết luận

Nhóm đã đã thành công trong việc xây dựng một ứng dụng web thương mại điện tử hoàn chỉnh, đáp ứng các yêu cầu về chức năng và phi chức năng đã đề ra. Ứng dụng được xây dựng dựa trên kiến trúc MVC, sử dụng các công nghệ NodeJS, ExpressJS, MongoDB và Handlebars, mang lại hiệu suất tốt, khả năng mở rộng và bảo trì dễ dàng.

Trong quá trình phát triển, nhóm đã áp dụng phương pháp Agile và khung làm việc Scrum để quản lý dự án. Việc chia nhỏ dự án thành các sprint, lập kế hoạch chi tiết, theo dõi tiến độ và thường xuyên đánh giá đã giúp nhóm làm việc hiệu quả và đạt được các mục tiêu đề ra.

Ứng dụng đã được triển khai thành công trên môi trường Docker, đảm bảo tính ổn định và khả năng mở rộng. Giao diện người dùng được thiết kế thân thiện, dễ sử dụng, mang lại trải nghiệm tốt cho người dùng.

6.2. Hạn chế

Tuy đã đạt được các mục tiêu đề ra, dự án vẫn còn một số hạn chế cần được khắc phục và cải thiện trong tương lai:

- **Chức năng:** Ứng dụng mới chỉ bao gồm các tính năng cơ bản của một trang web thương mại điện tử. Cần phát triển thêm các tính năng nâng cao như giỏ hàng, thanh toán trực tuyến, đánh giá sản phẩm, khuyến mãi, v.v.
- **Hiệu suất:** Mặc dù ứng dụng đã được tối ưu hóa, nhưng vẫn cần tiếp tục cải thiện hiệu suất, đặc biệt là khi lượng truy cập tăng cao.
- **Bảo mật:** Cần nâng cao tính bảo mật của ứng dụng bằng cách áp dụng các biện pháp bảo mật mạnh mẽ hơn, như xác thực hai yếu tố, mã hóa dữ liệu nhạy cảm, và kiểm tra lỗ hổng bảo mật thường xuyên.
- **Kiểm thử:** Cần mở rộng phạm vi kiểm thử để đảm bảo chất lượng và độ tin cậy của ứng dụng.
- **Xác thực người dùng và phân quyền:** Chưa xây dựng được xác thực người dùng và phân quyền.
- **Chức năng giỏ hàng:** Chưa xây dựng được chức năng giỏ hang.

6.3. Hướng phát triển

Trong tương lai, nhóm dự định sẽ tiếp tục phát triển và hoàn thiện ứng dụng theo các hướng sau:

Bổ sung tính năng:

- Xây dựng hệ thống giỏ hàng và thanh toán trực tuyến.
- Phát triển tính năng đánh giá và xếp hạng sản phẩm.
- Thêm các chương trình khuyến mãi và giảm giá.
- Xây dựng hệ thống quản lý khách hàng thân thiết.

Cải thiện hiệu suất:

- Tối ưu hóa truy vấn cơ sở dữ liệu.
- Áp dụng các kỹ thuật caching để giảm tải cho máy chủ.
- Sử dụng CDN (Content Delivery Network) để tăng tốc độ tải trang.

Nâng cao bảo mật:

- Áp dụng các biện pháp bảo mật mạnh mẽ hơn như xác thực hai yếu tố, mã hóa dữ liệu nhạy cảm.
- Thực hiện kiểm tra lỗ hổng bảo mật thường xuyên.

Mở rộng nền tảng:

- Phát triển ứng dụng di động cho iOS và Android.
- Hỗ trợ đa ngôn ngữ và đa tiền tệ.

Tích hợp với các dịch vụ khác:

- Tích hợp với các mạng xã hội để tăng khả năng tiếp cận khách hàng.
- Tích hợp với các công cụ phân tích dữ liệu để theo dõi hiệu quả kinh doanh.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] N. C. Hau Nguyen, "Node.js Web Development," Packt Publishing, 2014.

[2] M. Fowler and J. Highsmith, "Agile Software Development: The Cooperative Game," Addison-Wesley Professional, 2001.

[3] T. Dinh-Trong and T. Nguyen-Duc, "Building an E-commerce Website Using Node.js and MongoDB," International Journal of Computer Science and Information Security, vol. 16, no. 4, pp. 1-6, 2018.

[4] A. Cockburn and J. Highsmith, "Agile Software Development: The Business of Innovation," IEEE Computer, vol. 34, no. 9, pp. 120-122, 2001.

[5] V. T. Nguyen and H. V. Pham, "Developing an Online Shopping Application Using Node.js and Agile Methodology," in Proceedings of the 2020 International Conference on Advanced Technologies for Communications (ATC), pp. 123-128, 2020.

[6] T. T. Nguyen and T. T. Nguyen, "Applying Agile Methods in E-commerce Development: A Case Study," in Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), pp. 567-572, 2019.

[7] N. T. Nguyen, "Developing an E-commerce Platform Using Node.js and Agile Methodology," Master's thesis, University of Information Technology, Ho Chi Minh City, Vietnam, 2022.

[8] Node.js Foundation, "Node.js Documentation," https://nodejs.org/en/docs/, accessed June 13, 2024.

[9] Agile Alliance, "Agile 101," https://www.agilealliance.org/agile101/, accessed June 13, 2024.

[10] MongoDB, Inc., "MongoDB Documentation," https://www.mongodb.com/docs/, accessed June 13, 2024.