



**LỜI CẢM ƠN**

Để thực hiện được đề tài này, nhóm em xin cám ơn đến giáo viên hướng dẫn TS. Nguyễn Bảo Ân đã giúp đỡ chúng em suốt thời gian học. Trong bài báo cáo này có rất nhiều kiến thức mới vì vậy, cần đòi kiến thức học được từ trước do thầy giảng dạy đã giúp chúng em có đủ kiến thức nền tảng để xây dựng nên đề tài

**MỤC LỤC**

[**CHƯƠNG I: TÍNH CẦN THIẾT CỦA PHẦN MỀM PHÂN TÁN,MICROSERVICES,CLOUD** 1](#_Toc170653560)

[**1.1.** **Đặc tả nhu cầu của ứng dụng** 1](#_Toc170653561)

[**1.2.** **Phần mềm phân tán** 1](#_Toc170653562)

[**1.2.1.** **Phần mềm phân tán là gì?** 1](#_Toc170653563)

[**1.2.2.** **Phần mềm phân tán dùng làm gì?** 2](#_Toc170653564)

[**1.2.3.** **Ứng Dụng của Phần Mềm Phân Tán** 3](#_Toc170653565)

[**1.3.** **Microservices** 4](#_Toc170653566)

[**1.3.1.** **Microservices là gì?** 4](#_Toc170653567)

[**1.3.2.** **Chức Năng và Ứng Dụng của Microservices** 4](#_Toc170653568)

[**1.3.3.** **Ứng Dụng Của Microservices** 5](#_Toc170653569)

[**1.4.** **Cloud là gì?** 6](#_Toc170653570)

[**CHƯƠNG II: KẾ HOẠCH PHÁT TRIỂN PHẦN MỀM** 7](#_Toc170653571)

[**2.1.** **Độ khó của đề tài** 7](#_Toc170653572)

[**2.2.** **Lên kế hoạch trên Jira** 7](#_Toc170653573)

[**2.3.** **Phân công nhiệm vụ** 7](#_Toc170653574)

[**2.4.** **Kế hoạch triển khai kế hoạch trên Jira** 7](#_Toc170653575)

[**2.4.1.** **Giới thiệu các chức năng của Jira** 7](#_Toc170653576)

[**2.4.2.** **Triển khai kế hoạch trên Jira** 8](#_Toc170653577)

[**CHƯƠNG III: THIẾT KẾ GIAO DIỆN BẰNG FIGMA** 11](#_Toc170653578)

[**3.1.** **Giới thiệu phần mềm Figma** 11](#_Toc170653579)

[**3.1.1.** **Figma là gì?** 11](#_Toc170653580)

[**3.1.2.** **Mối quan hệ phụ thuộc Figma với Microservices** 11](#_Toc170653581)

[**3.1.3.** **Những ưu điểm nổi bật của Figma** 11](#_Toc170653582)

[**3.2.** **Giới thiệu chức năng** 13](#_Toc170653583)

[**3.3.** **Figma dành cho ai?** 14](#_Toc170653584)

[**3.4.** **Xây dựng hệ thống bán thiết bị điện tử** 15](#_Toc170653585)

[**CHƯƠNG VI: THIẾT KẾ KIẾN TRÚC PHẦN MỀM** 17](#_Toc170653586)

[**4.1. Kiến trúc của Microservices** 17](#_Toc170653587)

[**4.2. Quy trình, kiến trúc của Figma tích hợp các dịch vụ Microservices** 18](#_Toc170653588)

[**4.3. Quy trình hoạt động của Figma với Microservices** 18](#_Toc170653589)

[**CHƯƠNG VII: TRIỂN KHAI GITHUB VÀ DOCKER** 19](#_Toc170653590)

[**6.1. Giới thiệu về Docker** 19](#_Toc170653591)

[**6.2. Quy trình làm việc của Docker** 19](#_Toc170653592)

[**6.4. Thực hành với Docker** 20](#_Toc170653593)

[**6.4.1.** **Tại sao bạn nên sử dụng Docker?** 20](#_Toc170653601)

[**6.4.2.** **Kiến trúc của Docker** 21](#_Toc170653602)

[**6.4.3.** **Docker images** 21](#_Toc170653603)

[**6.4.4.** **Docker Container** 21](#_Toc170653604)

[**6.4.5.** **Quy trình thực thi của một hệ thống sử dụng Docker** 22](#_Toc170653605)

[**6.4.6.** **Các bước để Build Website lên Docker Desktop** 23](#_Toc170653606)

[**6.5.** **Thực hành với GitHub** 24](#_Toc170653607)

[**TÀI LIỆU THAM KHẢO** 27](#_Toc170653608)

# **CHƯƠNG I: TÍNH CẦN THIẾT CỦA PHẦN MỀM PHÂN TÁN,MICROSERVICES,CLOUD**

* 1. **Đặc tả nhu cầu của ứng dụng**

Trong bối cảnh thương mại điện tử phát triển mạnh mẽ, nhu cầu xây dựng một hệ thống bán thiết bị điện tử trực tuyến với giao diện người dùng thân thiện, tối ưu hóa trải nghiệm người dùng trở nên vô cùng cấp thiết. Ứng dụng cần có khả năng quản lý danh mục sản phẩm đa dạng, từ điện thoại di động, máy tính xách tay, máy ảnh, đến các phụ kiện điện tử. Ngoài ra, hệ thống cần tích hợp các chức năng thanh toán an toàn, giỏ hàng tiện dụng, và cơ chế theo dõi đơn hàng hiệu quả. Để đáp ứng các nhu cầu này, Figma được chọn làm công cụ thiết kế chính, nhờ khả năng tạo mẫu nhanh, dễ dàng chia sẻ và cộng tác trong thời gian thực. Bằng cách sử dụng Figma, nhóm phát triển có thể tạo ra các mô hình giao diện chi tiết, từ trang chủ, trang sản phẩm, đến trang thanh toán, giúp minh họa rõ ràng các yêu cầu chức năng và phi chức năng của hệ thống. Điều này không chỉ giúp giảm thiểu sai sót trong quá trình phát triển mà còn tăng cường khả năng tương tác giữa các thành viên trong nhóm, đảm bảo rằng sản phẩm cuối cùng sẽ đáp ứng tối đa kỳ vọng của người dùng.

* 1. **Phần mềm phân tán**
     1. **Phần mềm phân tán là gì?**

Phần mềm phân tán là các ứng dụng phần mềm được lưu trữ chủ yếu trên nền tảng điện toán đám mây và chạy trên nhiều hệ thống cùng một lúc. Các hệ thống chạy trên cùng một mạng và liên hệ với nhau trong nỗ lực hoàn thành một nhiệm vụ hoặc lệnh cụ thể.

Ứng dụng phân tán được thiết kế cho phép người dùng mạng cộng tác và chia sẻ ý tưởng, điều phối các nhiệm vụ, truy cập thông tin và trao đổi ứng dụng thông qua máy chủ. Ứng dụng phân tán chủ yếu được sử dụng trên các mạng máy khách - máy chủ nơi máy tính của người dùng truy cập thông tin từ máy chủ hoặc máy chủ điện toán đám mây.

Các hệ thống máy tính khác nhau đã được phân phối trên mạng thường được giao nhiệm vụ với các mục tiêu tương tự hoặc khác nhau. Ví dụ: trong một nền tảng thương mại điện tử, mỗi máy tính có thể chịu trách nhiệm cho các tác vụ cụ thể như gửi và nhận email về các ưu đãi đặc biệt cho khách hàng hiện tại; tổng hợp danh sách khách hàng và lịch sử mua hàng của họ để nhắm mục tiêu sản phẩm tốt hơn đến họ; v.v...

* + 1. **Phần mềm phân tán dùng làm gì?**

**Xử Lý Phân Tán:**

Phân chia công việc thành nhiều phần nhỏ và xử lý chúng đồng thời trên nhiều máy tính để tăng hiệu suất và giảm thời gian xử lý.

**Lưu Trữ Phân Tán:**

Lưu trữ dữ liệu trên nhiều máy tính hoặc server để đảm bảo an toàn và sẵn sàng cao. Nếu một node bị hỏng, dữ liệu vẫn có thể truy cập được từ các node khác.

**Tính Sẵn Sàng Cao và Khả Năng Chịu Lỗi:**

Hệ thống phân tán có khả năng tiếp tục hoạt động ngay cả khi một hoặc nhiều node gặp sự cố, giúp giảm thiểu thời gian chết và tăng độ tin cậy.

**Mở Rộng Dễ Dàng:**

Dễ dàng thêm hoặc bớt tài nguyên (máy tính, server) mà không ảnh hưởng đến hoạt động của toàn hệ thống, giúp hệ thống dễ dàng mở rộng khi cần.

**Cộng Tác Thời Gian Thực:**

Cho phép nhiều người dùng làm việc đồng thời trên cùng một tài liệu hoặc dự án, cập nhật và đồng bộ hóa thay đổi ngay lập tức.

**Cung Cấp Dịch Vụ:**

Cung cấp các dịch vụ như web services, API, cơ sở dữ liệu, và nhiều dịch vụ khác mà người dùng và các ứng dụng khác có thể truy cập từ xa.

**Phân Tán Tải:**

Phân phối tải công việc đều đặn giữa các node để tránh tình trạng quá tải và đảm bảo hiệu suất ổn định.

* + 1. **Ứng Dụng của Phần Mềm Phân Tán**

Phần mềm phân tán được sử dụng rộng rãi trong nhiều lĩnh vực, bao gồm:

**Máy Chủ Web và Dịch Vụ Trực Tuyến:**

Các hệ thống phân tán cung cấp khả năng phục vụ hàng triệu yêu cầu từ người dùng trên toàn thế giới, ví dụ như Google, Amazon, Facebook.

**Tài Chính và Ngân Hàng:**

Xử lý giao dịch, quản lý tài khoản và dữ liệu khách hàng trên các hệ thống phân tán để đảm bảo độ tin cậy và bảo mật cao.

**Truyền Thông và Giải Trí:**

Cung cấp dịch vụ streaming, chơi game trực tuyến và các ứng dụng truyền thông khác với hiệu suất cao và độ trễ thấp.

**Khoa Học và Nghiên Cứu:**

Hỗ trợ các dự án nghiên cứu khoa học yêu cầu xử lý dữ liệu lớn và phức tạp, ví dụ như mô phỏng khí hậu, phân tích gene, nghiên cứu không gian.

**Hệ Thống Quản Lý Doanh Nghiệp:**

Quản lý tài nguyên, quy trình sản xuất, và các hoạt động kinh doanh khác trên một hệ thống phân tán để tăng hiệu quả và giảm chi phí.

**IoT (Internet of Things):**

Kết nối và quản lý các thiết bị thông minh, cảm biến và hệ thống điều khiển trên toàn cầu.

* 1. **Microservices**
     1. **Microservices là gì?**

Microservices là các module trong hệ thống được chia thành nhiều services nhỏ. Mỗi service sẽ thực hiện các chức năng chuyên biệt, như quản lý đơn hàng hoặc quản lý khách hàng,… và được đặt tại một server riêng, cho phép nâng cấp chỉnh sửa một cách độc lập.

Việc áp dụng kiến trúc microservices cho phép chia nhỏ chức năng của ứng dụng thành các dịch vụ nhỏ, tối ưu hóa trải nghiệm và tốc độ cho từng người dùng. Thiết kế giao diện dựa trên từng đối tượng giúp cải thiện tương thích và tốc độ, đồng thời giảm thiểu các chức năng không cần thiết.

* + 1. **Chức Năng và Ứng Dụng của Microservices**

Microservices được sử dụng rộng rãi trong phát triển phần mềm hiện đại vì nhiều lý do, bao gồm:

**Phát Triển Độc Lập:**

Các nhóm phát triển có thể làm việc đồng thời trên các dịch vụ khác nhau mà không ảnh hưởng đến nhau. Điều này giúp tăng tốc độ phát triển và cải thiện hiệu quả.

**Triển Khai và Cập Nhật Độc Lập:**

Mỗi dịch vụ có thể được triển khai và cập nhật độc lập mà không gây gián đoạn cho toàn bộ hệ thống. Điều này giảm thiểu rủi ro và tăng khả năng linh hoạt.

**Khả Năng Mở Rộng:**

Các dịch vụ có thể được mở rộng riêng lẻ theo nhu cầu. Nếu một dịch vụ cụ thể cần thêm tài nguyên, chỉ dịch vụ đó cần được mở rộng thay vì toàn bộ ứng dụng.

**Khả Năng Chịu Lỗi:**

Khi một dịch vụ gặp sự cố, các dịch vụ khác vẫn có thể tiếp tục hoạt động bình thường. Điều này giúp hệ thống tổng thể có khả năng chịu lỗi cao hơn.

**Tái Sử Dụng:**

Các dịch vụ có thể được tái sử dụng trong nhiều dự án khác nhau, giảm thiểu công sức phát triển lại từ đầu.

* + 1. **Ứng Dụng Của Microservices**

Microservices được áp dụng trong nhiều lĩnh vực và tình huống khác nhau:

**Dịch Vụ Web và Ứng Dụng Trực Tuyến:**

Các ứng dụng lớn như Amazon, Netflix, và Spotify sử dụng microservices để quản lý các chức năng như quản lý người dùng, thanh toán, phát video, và tìm kiếm.

**Hệ Thống Thương Mại Điện Tử:**

Microservices giúp quản lý các thành phần như giỏ hàng, thanh toán, quản lý sản phẩm và tồn kho một cách hiệu quả và độc lập.

**Ứng Dụng Tài Chính:**

Các hệ thống ngân hàng và tài chính sử dụng microservices để xử lý giao dịch, quản lý tài khoản, và phân tích dữ liệu theo thời gian thực.

**Truyền Thông và Giải Trí:**

Các nền tảng như YouTube sử dụng microservices để quản lý nội dung, quảng cáo, và tương tác người dùng.

**Chăm Sóc Sức Khỏe:**

Các ứng dụng y tế sử dụng microservices để quản lý hồ sơ bệnh nhân, đặt lịch hẹn, và theo dõi sức khỏe.

**Internet of Things (IoT):**

Microservices giúp quản lý và điều phối các thiết bị IoT, xử lý dữ liệu từ cảm biến, và điều khiển thiết bị từ xa.

* 1. **Cloud là gì?**

Cloud (Điện toán đám mây) là mô hình điện toán sử dụng các công nghệ máy tính và phát triển dựa vào mạng Internet. Thuật ngữ "đám mây" ở đây là lối nói ẩn dụ chỉ mạng Internet (dựa vào cách được bố trí của nó trong sơ đồ mạng máy tính) và như một liên tưởng về độ phức tạp của các cơ sở hạ tầng chứa trong nó. Ở mô hình điện toán này, mọi khả năng liên quan đến công nghệ thông tin đều được cung cấp dưới dạng các "dịch vụ", cho phép người sử dụng truy cập các dịch vụ công nghệ từ một nhà cung cấp nào đó "trong đám mây" mà không cần phải có các kiến thức, kinh nghiệm về công nghệ đó, cũng như không cần quan tâm đến các cơ sở hạ tầng phục vụ công nghệ đó.

**CHƯƠNG II: KẾ HOẠCH PHÁT TRIỂN PHẦN MỀM**

* 1. **Độ khó của đề tài**

Báo cáo này tập trung vào việc phát triển một phần mềm phân tán sử dụng kiến trúc microservices và các dịch vụ cloud, với độ khó cao do sự phức tạp và đa dạng của các công nghệ liên quan. Các thành phần chính của báo cáo bao gồm việc xác định nhu cầu và tính cần thiết của ứng dụng, lập kế hoạch phát triển chi tiết với product backlog và phân chia sprint hợp lý. Thiết kế kiến trúc phần mềm bao gồm việc mô tả các chức năng, kiến trúc microservices và tích hợp với các dịch vụ đám mây, yêu cầu hiểu biết sâu về cách các dịch vụ tương tác và quản lý. Sử dụng các công cụ như Git/GitHub để quản lý mã nguồn và Docker để triển khai ứng dụng đòi hỏi kỹ năng và kinh nghiệm trong việc cấu hình và vận hành các công cụ này.

* 1. **Lên kế hoạch trên Jira**

Để thực hiện tốt việc triển khai ứng dụng, cần có kế hoạch như sau:

* Thêm thành viên
* Phân công
* Lập kế hoạch sprint
* Theo dõi tiến độ
* Quản lý backlog
  1. **Phân công nhiệm vụ**

Phân công nhiệm vụ rõ ràng và cụ thể là yếu tố quan trọng để đảm bảo thành công của dự án. Hy vọng đoạn văn bản này sẽ giúp bạn phân công nhiệm vụ hiệu quả cho các thành viên trong nhóm và hoàn thành dự án đúng tiến độ.

* 1. **Kế hoạch triển khai kế hoạch trên Jira**
     1. **Giới thiệu các chức năng của Jira**

**Backlog - Kho Lưu Trữ Công Việc:** Backlog là danh sách tập hợp tất cả các công việc cần làm của một dự án phần mềm. Đây là nơi lưu trữ các yêu cầu tính năng, lỗi cần sửa, và các nhiệm vụ khác. Backlog giúp nhóm phát triển tổ chức và ưu tiên các công việc cần hoàn thành để đảm bảo tiến độ dự án.

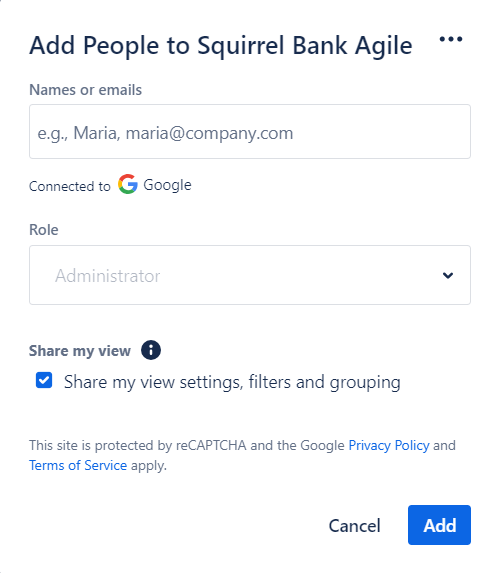
**Scrum và Kanban - Quy Trình Phát Triển Linh Hoạt:** Scrum và Kanban là hai phương pháp quản lý dự án linh hoạt được sử dụng rộng rãi trong phát triển phần mềm. Scrum tập trung vào các chu kỳ phát triển ngắn gọi là Sprint, trong khi Kanban chú trọng vào việc quản lý công việc liên tục thông qua bảng Kanban. Cả hai phương pháp đều giúp tối ưu hóa quy trình làm việc và cải thiện hiệu quả của nhóm.

**Sprint - Chu Kỳ Phát Triển Trong Scrum:** Sprint là một chu kỳ phát triển ngắn hạn trong phương pháp Scrum, thường kéo dài từ 1 đến 4 tuần. Trong mỗi Sprint, nhóm phát triển sẽ hoàn thành một tập hợp các công việc cụ thể từ Backlog. Kết quả của mỗi Sprint là một phần mềm hoạt động được, giúp theo dõi tiến độ và phản hồi nhanh chóng từ người dùng.

**Roadmap - Kế Hoạch Chi Tiết:** Roadmap là bản kế hoạch chi tiết mô tả các mục tiêu và lộ trình phát triển của dự án phần mềm theo thời gian. Nó bao gồm các mốc quan trọng, các tính năng cần phát triển và các giai đoạn triển khai. Roadmap giúp nhóm phát triển và các bên liên quan hiểu rõ hướng đi của dự án và đảm bảo rằng mọi người đều có cùng mục tiêu.

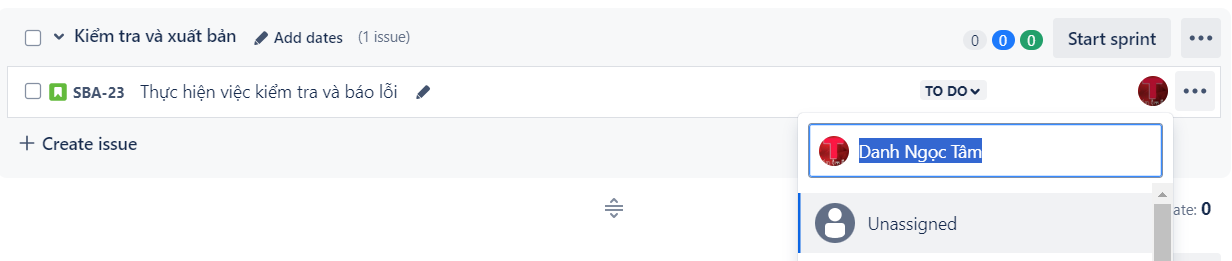
**Bitbucket và Confluence - Tích Hợp Hệ Thống:** Bitbucket là một hệ thống quản lý mã nguồn và điều phối công việc phát triển phần mềm, trong khi Confluence là một nền tảng tài liệu và cộng tác nhóm. Khi tích hợp với nhau, Bitbucket và Confluence tạo ra một môi trường phát triển hợp nhất, giúp nhóm dễ dàng quản lý mã nguồn, theo dõi tiến độ dự án, và chia sẻ thông tin một cách hiệu quả.

* + 1. **Triển khai kế hoạch trên Jira**
* Thêm thành viên:



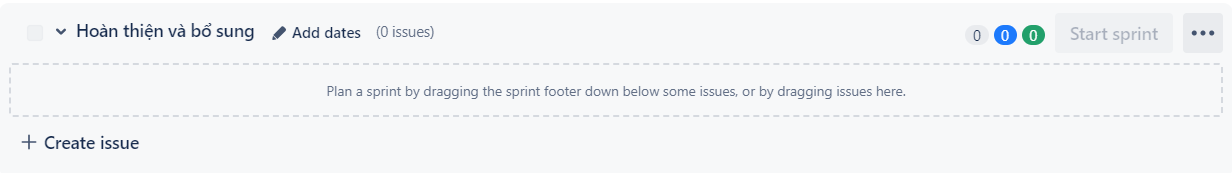
*Hình 1.1. Thêm thành viên*

* Phân công



*Hình 1.2. Phân công cho thành viên*

* Lập kế hoạch sprint



*Hình 1.3. Tạo Sprint*

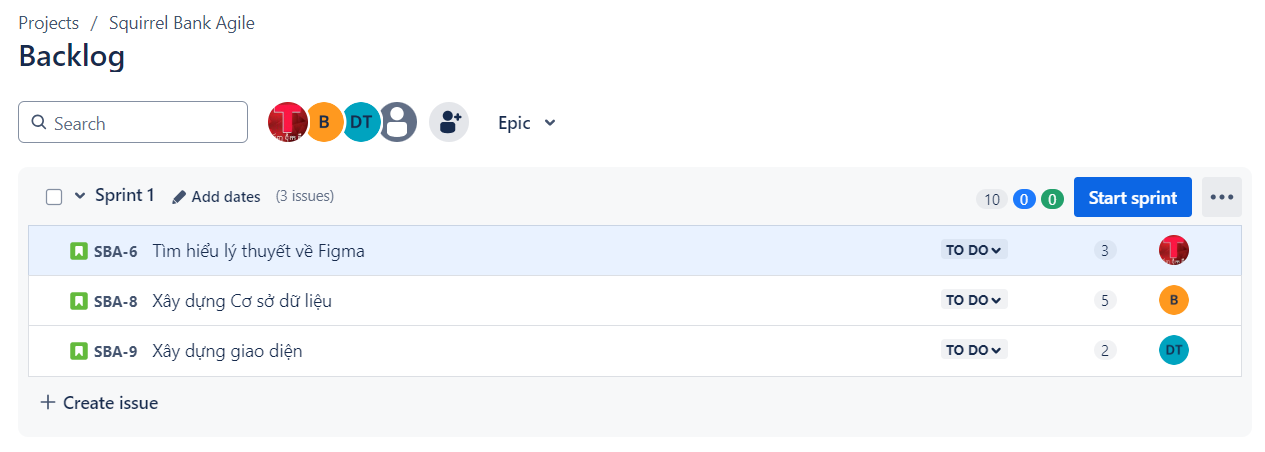
* Theo dõi tiến độ

A screenshot of a chat

Description automatically generated

*Hình 1.4. Trạng thái của công việc*

* Quản lý backlog



*Hình 1.5. Giao diện Backlog*

**CHƯƠNG III: THIẾT KẾ GIAO DIỆN BẰNG FIGMA**

* 1. **Giới thiệu phần mềm Figma**
     1. **Figma là gì?**



*Hình 1.6. Logo Figma*

Figma giúp các nhà thiết kế tạo ra các giao diện và trải nghiệm người dùng này bằng cách cung cấp các công cụ mạnh mẽ và trực quan. Người dùng có thể vẽ và chỉnh sửa các thiết kế, kiểm tra chúng qua các bản prototype (mô hình mẫu), và dễ dàng chia sẻ công việc với các thành viên khác trong nhóm để cùng hợp tác và hoàn thiện sản phẩm. Figma hoạt động hoàn toàn trên nền tảng đám mây, do đó các thành viên trong nhóm có thể làm việc cùng nhau từ bất kỳ đâu chỉ cần có kết nối internet.

* + 1. **Mối quan hệ phụ thuộc Figma với Microservices**

Trong một quy trình phát triển sản phẩm hoàn chỉnh, thiết kế giao diện người dùng và trải nghiệm người dùng được tạo ra bởi Figma sẽ cần phải được tích hợp và tương tác với các dịch vụ microservices.

* + 1. **Những ưu điểm nổi bật của Figma**
* Khả năng tương thích cao

Hiện nay, với nhiều hệ điều hành phổ biến khác nhau như Windows, Linux, MacOS,… Figma đều có khả năng truy cập và hoạt động bình thường, không gặp bất cứ trở ngại nào.

Người dùng có thể truy cập đồng thời các dự án của mình. Bắt đầu thiết kế từ bất kỳ máy tính hoặc nền tảng nào. Không cần phải mua nhiều giấy phép hoặc cài đặt phần mềm.

* Hỗ trợ làm việc nhóm

Những thành viên trong nhóm có thể tương tác với nhau một cách nhanh chóng và dễ dàng. Tất cả thành viên có trong file sẽ được hiển thị avatar. Những thay đổi trong file cũng được cập nhật để người dùng thuận tiện theo dõi quá trình thiết kế.

Nhờ vào tính năng design và prototyping, nhận xét, bình luận hay phản hồi của các bên liên quan sẽ được thông báo qua email hoặc slack (Phòng chat). Do đó, người dùng có thể chỉnh sửa và hoàn thiện sản phẩm kịp thời và nhanh chóng.

* Kho plugin cực kỳ mạnh, đa dạng

Kho puglin của Figma đáp ứng hầu hết nhu cầu của người dùng hiện nay. Thậm chí, kho plugin của Figma còn tương đương với Sketch và ít lỗi hơn hẳn.

Người dùng có thể tạo và lưu trữ 3 dự án đang hoạt động cùng một lúc trên Figma.

* Lưu trữ trên hệ thống điện toán đám mây

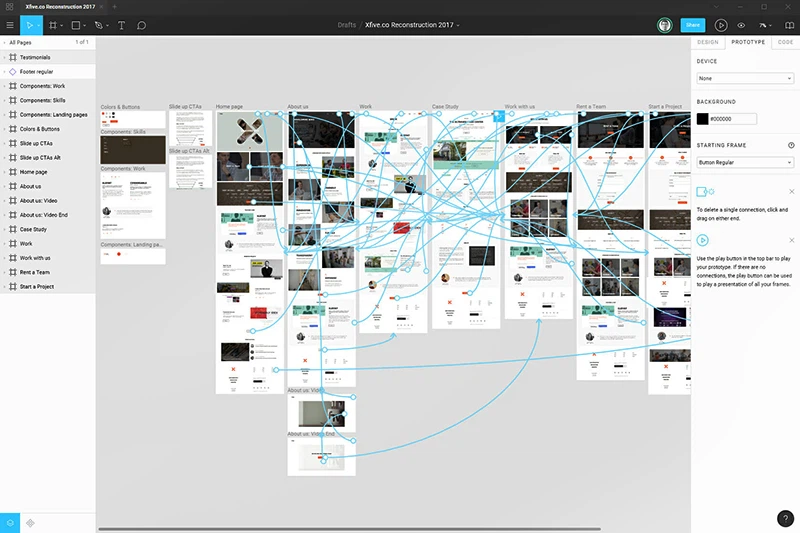
Bạn có đang lo sợ việc đang thiết kế thì máy tính bị ngắt mạng và mất hết dữ liệu? Nếu vậy, với Figma, bạn không còn phải lo lắng về việc đó. Sử dụng hệ thống lưu trữ điện toán đám mây, các thiết kế đều có thể khôi phục chỉ với vài thao tác đơn giản. Ngoài ra, người dùng cũng có thể chia thành các mục lưu trữ. Bạn hoàn toàn có thể vừa có thể quản lý, kiểm soát dễ dàng vừa tránh được các rủi ro không mong muốn.

Ngoài những ưu điểm trên, Figma còn vô vàn những điểm hay mà người dùng có thể tận hưởng khi trải nghiệm ứng dụng trong công việc.

* 1. **Giới thiệu chức năng**

Một lý do khác khiến các Designer yêu thích ứng dụng này là Figma cung cấp một gói miễn phí phong phú với đầy đủ các tính năng, bạn có thể tạo và lưu trữ 3 dự án đang hoạt động cùng một lúc. Nó là quá đủ để bạn học hỏi, thử nghiệm và làm việc trong các dự án nhỏ. Dưới đây là các chức năng chính của Figma:

**Thiết kế nhiều layout trong một sản phẩm:** Figma tích hợp tính năng quản lý nhiều artboard đồng thời, cho phép bạn tạo nhiều bố cục với kích thước khác nhau trên cùng trên một sản phẩm mà không cần phải tạo thêm bất cứ tệp mới nào.

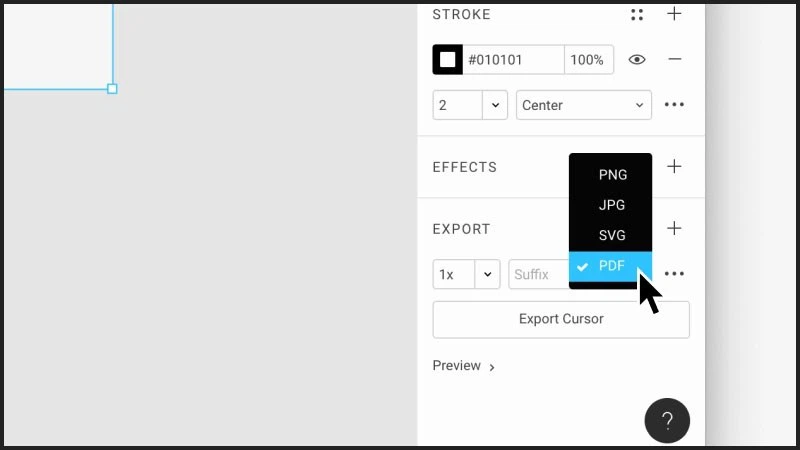


*Hình 1.7. Thiết kế nhiều layout trong một sản phẩm*

**Xuất được đa dạng file ảnh cực sắc nét**

Figma, sử dụng định dạng vector trong quá trình thiết kế, từ đó đảm bảo chất lượng của hình ảnh không giảm và rõ nét nhất khi bị thay đổi kích thước.

Khả năng xuất ra hình ảnh với độ sắc nét được giữ nguyên, bạn có thể lưu dưới nhiều định dạng khác nhau như SVG, PDF, PNG, JPG. Điều này giúp linh hoạt trong việc sử dụng hình ảnh và phục vụ đa dạng trong các tình huống khác nhau.



*Hình 1.8. Đa dạng file ảnh*

**Hỗ trợ lưu trữ đám mây** Figma tận dụng dịch vụ đám mây để thuận tiện cho việc lưu trữ và chỉnh sửa dữ liệu, mô hình này tương đồng với việc chỉnh sửa nội dung trực tuyến trên Google Docs. Điều này giúp loại bỏ lo ngại về việc ổ đĩa bị đầy hoặc quên sao lưu, bởi Figma sẽ thực hiện tự động các công đoạn này.

* 1. **Figma dành cho ai?**

Với những ưu điểm nổi bật được nêu trên, có thể khẳng định rằng Figma là một công cụ hoàn hảo cho các nhà thiết kế. Nó không chỉ cung cấp đầy đủ các công cụ hỗ trợ và tính năng mạnh mẽ nhất, giúp nhà thiết kế có khả năng sáng tạo và quản lý dự án một cách hiệu quả, mà còn đặc biệt bởi việc có sẵn phiên bản miễn phí dành cho tất cả người dùng.

Trong bối cảnh dịch bệnh diễn biến phức tạp, việc làm việc từ xa (remote work) trở nên ngày càng phổ biến và Figma trở thành một giải pháp linh hoạt và hiệu quả. Tính năng cộng tác theo thời gian thực của Figma đặc biệt nổi bật cho phép nhóm làm việc từ xa có thể dễ dàng cùng nhau làm việc trên cùng một dự án, đồng thời thực hiện các tác vụ như đưa ra phản hồi, kiểm tra tiến độ công việc và trao đổi về thiết kế một cách linh hoạt.

Cho đến hiện nay, khó có thể tìm ra một công cụ nào khác có thể vượt qua Figma trong việc kết hợp hiệu quả giữa công nghệ thiết kế và khả năng cộng tác từ xa, làm nổi bật vị thế của nó là một trong những lựa chọn hàng đầu cho các đội ngũ thiết kế làm việc theo mô hình làm việc từ xa.

Ngoài ra, các vị trí như nhà phát triển ứng dụng/ website (front-end/ back-end developer), hay ngay cả những vị trí cấp cao như giám đốc công nghệ (CTO) cũng thường xuyên ứng dụng Figma trong công việc để dễ dàng kiểm soát và trao đổi về tiến độ làm việc.

* 1. **Xây dựng hệ thống bán thiết bị điện tử**
* **Tạo dự án mớiA screenshot of a computer

  Description automatically generated**

*Hình 1.9. Bắt đầu xây dựng dự án mới*

**A screenshot of a login form

Description automatically generated**

*Hình 1.10. Xây dựng trang giao diện*

**CHƯƠNG VI: THIẾT KẾ KIẾN TRÚC PHẦN MỀM**

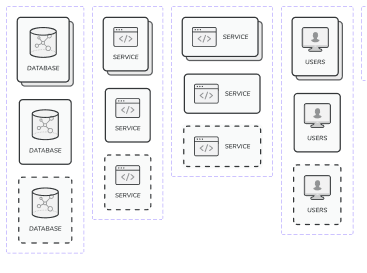
## **4.1. Kiến trúc của Microservices**

****

*Hình 1.11. Kiến trúc của Microservices*

* **Client**: Đây là người dùng của ứng dụn. Nó giao tiếp với các dịch vụ thông qua API Gateway.
* **API Gateway:** Đây là điểm truy cập duy nhất cho ứng dụng. Nó định tuyến các yêu cầu từ khách hàng đến các dịch vụ thích hợp.
* **Service**: Đây là các khối xây dựng cơ bản của ứng dụng microservice. Mỗi dịch vụ có trách nhiệm cho một chức năng cụ thể và có thể triển khai và quản lý độc lập.
* **Management/Orchestration:** Đây là thành phần chịu trách nhiệm quản lý vòng đời của các dịch vụ microservice. Nó có thể bao gồm các công cụ như Kubernetes, Docker.

## **4.2. Quy trình, kiến trúc của Figma tích hợp các dịch vụ Microservices**



*Hình 1.12. Kiến trúc của Microservices*

## **4.3. Quy trình hoạt động của Figma với Microservices**

1. Người dùng tạo ra các thiết kế giao diện người dùng trong Figma.
2. Figma lưu trữ các thiết kế này trong cơ sở dữ liệu.
3. Các dịch vụ microservices truy cập dữ liệu thiết kế từ cơ sở dữ liệu để thực hiện các chức năng cụ thể, chẳng hạn như quản lý tài nguyên thiết kế, quản lý quy trình làm việc, v.v.
4. Figma có thể giao tiếp với các dịch vụ microservices thông qua API để truy cập các chức năng bổ sung này.
5. Các dịch vụ microservices có thể giao tiếp với nhau thông qua API để chia sẻ dữ liệu và phối hợp các tác vụ.

**CHƯƠNG VII: TRIỂN KHAI GITHUB VÀ DOCKER**

## **6.1. Giới thiệu về Docker**

Docker là nền tảng cung cấp cho các công cụ, service để các developers, adminsystems có thể phát triển, thực thi, chạy các ứng dụng với containers. Hay nói một cách khác nó là một nền tảng để cung cấp cách để building, deploy và run các ứng dụng một cách dễ dàng trên nền tảng ảo hóa - "Build once, run anywhere". Hay nói một cách dễ hiểu như sau: Khi chúng ta muốn chạy app thì chúng ta phải thiết lập môi trường chạy cho nó. Thay vì chúng ta sẽ đi cài môi trường chạy cho nó thì chúng ta sẽ chạy docker.

Ứng dụng Docker chạy trong vùng chứa (container) có thể được sử dụng trên bất kỳ hệ thống nào: máy tính xách tay của nhà phát triển, hệ thống trên cơ sở hoặc trong hệ thống đám mây. Và là một công cụ tạo môi trường được "đóng gói" (còn gọi là Container) trên máy tính mà không làm tác động tới môi trường hiện tại của máy, môi trường trong Docker sẽ chạy độc lập.

## **6.2. Quy trình làm việc của Docker**

* Xây dựng Ứng dụng trong Môi trường Cơ sở: Đầu tiên, các nhà phát triển xây dựng ứng dụng của họ trong một môi trường cơ sở. Điều này có thể là một môi trường phát triển cục bộ trên máy tính cá nhân của họ.
* Đóng gói Ứng dụng bằng Dockerfile: Người phát triển tạo một tệp Dockerfile, một tài liệu văn bản định nghĩa cách Docker nên xây dựng hình ảnh cho ứng dụng. Dockerfile chứa các chỉ thị để cài đặt và cấu hình môi trường cho ứng dụng.
* Xây dựng Hình ảnh Docker: Sau khi có Dockerfile, người phát triển sử dụng lệnh docker build để tạo một hình ảnh Docker cho ứng dụng của họ. Quá trình này sẽ thực hiện các bước được chỉ định trong Dockerfile và tạo ra một hình ảnh Docker chứa tất cả các thành phần cần thiết cho ứng dụng.
* Chạy và Kiểm tra Hình ảnh Docker: Người phát triển sử dụng lệnh docker run để chạy hình ảnh Docker mới tạo. Họ có thể kiểm tra ứng dụng để đảm bảo rằng nó hoạt động đúng cách trong môi trường container.
* Lưu trữ Hình ảnh Docker trên Registry: Khi hình ảnh Docker đã được kiểm tra và sẵn sàng, người phát triển có thể lưu trữ nó trên một Docker Registry như Docker Hub hoặc một Registry riêng tư để chia sẻ với nhóm hoặc cộng đồng.
* Triển khai và Vận hành trong Môi trường Sản xuất: Cuối cùng, hình ảnh Docker có thể được triển khai trong môi trường sản xuất. Các công cụ quản lý Docker như Docker Compose hoặc Kubernetes có thể được sử dụng để tự động hóa quy trình triển khai và vận hành.

**6.3. Docker và GitHub dựa có dựa vào kiến trúc Microservices?**

* **Docker:** Docker là một nền tảng ảo hóa container, cho phép bạn đóng gói ứng dụng và tất cả các phụ thuộc của nó vào một container duy nhất. Mặc dù Docker không phải là một dịch vụ microservices, nhưng nó cung cấp một cách tiếp cận linh hoạt cho việc xây dựng và vận hành các microservices. Bằng cách sử dụng Docker, các microservices có thể được đóng gói và triển khai độc lập trong các container riêng biệt, giúp tăng cường tính cô lập, di động và quản lý của chúng.
* **GitHub:** GitHub là một nền tảng dựa trên web cho việc lưu trữ mã nguồn, quản lý phiên bản và hợp tác trong các dự án phát triển phần mềm. Mặc dù GitHub không phải là một dịch vụ microservices, nhưng nó cung cấp các tính năng và công cụ hỗ trợ cho việc phát triển các ứng dụng dựa trên kiến trúc microservices. Ví dụ, GitHub Actions cho phép bạn tự động hóa các quy trình phát triển, kiểm thử và triển khai, trong khi GitHub Packages cung cấp một cách để lưu trữ và quản lý các gói phần mềm.

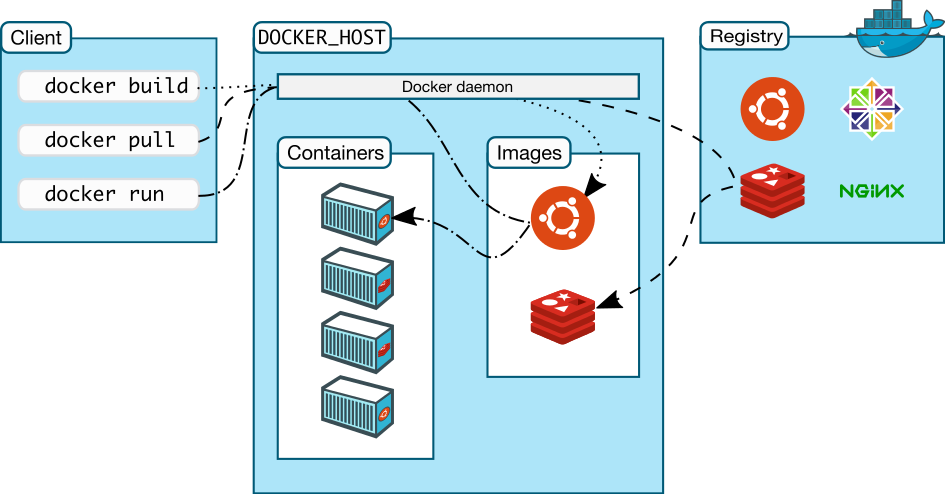
## **6.4. Thực hành với Docker**

3. 4. 1. **Tại sao bạn nên sử dụng Docker?**

Khi công việc kinh doanh cần tới nhiều ứng dụng web khác nhau, khi mà bây giờ frame work hay ngôn ngữ lập trình chỉ là công cụ. Các công ty không bị giới hạn và có thể sử dụng bất kỳ ngôn ngữ nào cần. Vì vậy chúng ta cần có một môi trường mà nhiều ứng dụng khác nhau có thể chạy cùng nhau trên đó.

* + 1. **Kiến trúc của Docker**

Docker sử dụng kiến trúc client-server. Docker client sẽ nói liên lạc với các Docker daemon, các Docker daemon sẽ thực hiện các tác vụ build, run và distribuing các Docker container.  Cả Docker client và Docker daemon có thể chạy trên cùng 1 máy, hoặc có thể kết nối theo kiểu Dockẻ client điều khiển các docker daemon như hình dưới. Docker client và daemon giao tiếp với nhau thông qua socker hoặc RESTful API.



*Hình 1.13. Kiến trúc hoạt động của Docker*

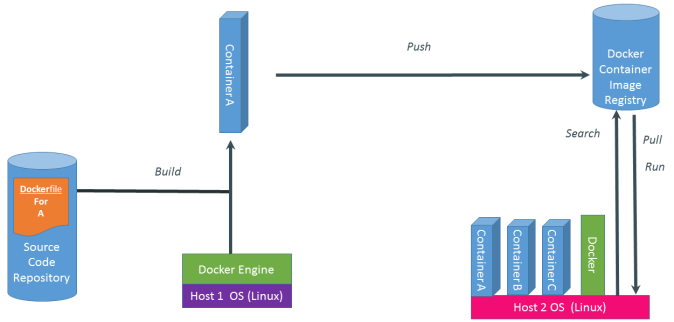
* + 1. **Docker images**

Là một template chỉ cho phép đọc, ví dụ một image có thể chứa hệ điều hành Ubuntu và web app. Images được dùng để tạo Docker container. Docker cho phép chúng ta build và cập nhật các image có sẵn một cách cơ bản nhất, hoặc bạn có thể download Docker images của người khác.

* + 1. **Docker Container**

Docker container có nét giống với các directory. Một Docker container giữ mọi thứ chúng ta cần để chạy một app. Mỗi container được tạo từ Docker image. Docker container có thể có các trạng thái run, started, stopped, moved và deleted.

* + 1. **Quy trình thực thi của một hệ thống sử dụng Docker**



*Hình 1.14. Quy trình thực thi của một hệ thống sử dụng Docker*

Như trong hình vẽ, một hệ thống Docker được thực thi với 3 bước chính :

Build 🡪 Push 🡪 Pull,Run

Lần lượt trình bày từng bước để chúng ta có thể hiểu rõ hơn về nguyên lí 3 bước này.

**a. Build**

Đầu tiên chúng ta sẽ tạo một dockerfile, trong dockerfile này chính là code của chúng ta.

Dockerfile này sẽ được Build tại một máy tính đã cài đặt Docker Engine.

Sau khi build ta sẽ thu được Container, trong Container này chứa bộ thư viện và ứng dụng của chúng ta.

**b. Push**

Sau khi có được Container, chúng ta thực hiện push Container này lên đám mây và lưu trữ ở đó.

Việc push này có thể thực hiện qua môi trường mạng Internet.

**c. Pull, Run**

Giả sử một máy tính muốn sử dụng Container chúng ta đã push lên đám mây (máy đã cài Docker Engine) thì bắt buộc máy phải thực hiện việc Pull container này về máy. Sau đó thực hiện Run Container này.

Đó chính là quy trình 3 bước miêu tả hoạt động của một hệ thống sử dụng Docker. Rất đơn giản và rõ ràng.

* + 1. **Các bước để Build Website lên Docker Desktop**

1. **Đăng nhập vào Docker Hub**:

Sử dụng lệnh sau để đăng nhập vào Docker Hub

A close-up of a computer code

Description automatically generated

1. **Kiểm tra tài khoản Docker Hub**

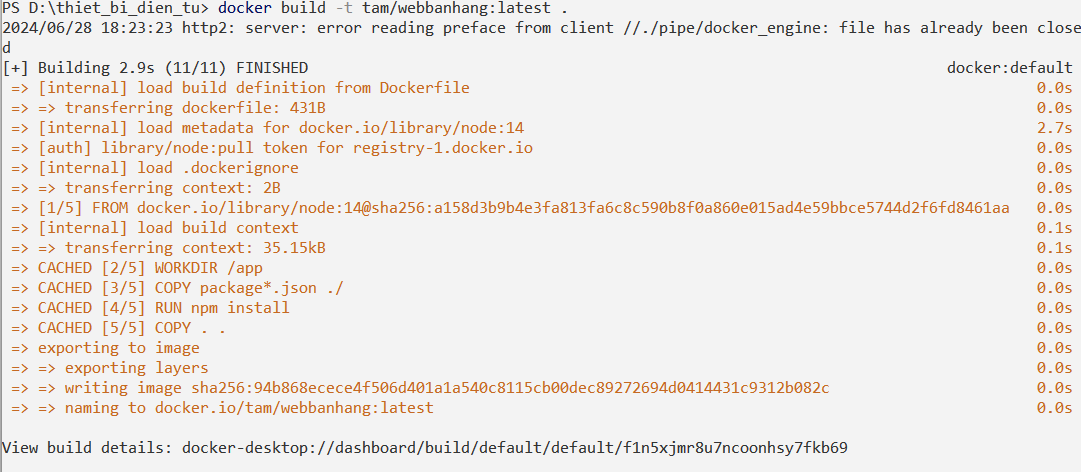
Sau khi đăng nhập, bạn có thể kiểm tra thông tin tài khoản của mình bằng cách xem các thông tin đăng nhập trên Docker Hub hoặc chạy lệnh sau để kiểm tra tên người dùng hiện tại (nếu bạn đã đăng nhập trước đó):

**A computer screen with a white background

Description automatically generated**

1. **Xây dựng Image Docker**

Sử dụng lệnh docker build để xây dựng Image

****

1. **Kiểm tra Image vừa được tạo**

**A black text on a white background

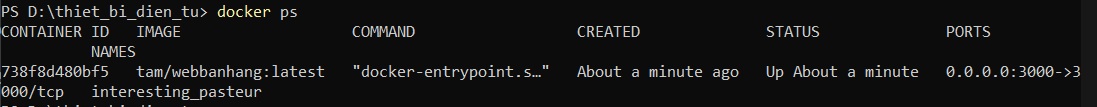
Description automatically generated**

Kết quả lệnh docker images sẽ hiển thị danh sách các Image Docker trên máy của bạn, bao gồm Image vừa được xây dựng với tên tài khoản Docker Hub.

1. **sử dụng lệnh docker run để tạo và chạy một container từ Image mà bạn đã xây dựng.**

****

1. **Kiểm tra Container đang chạy**

****

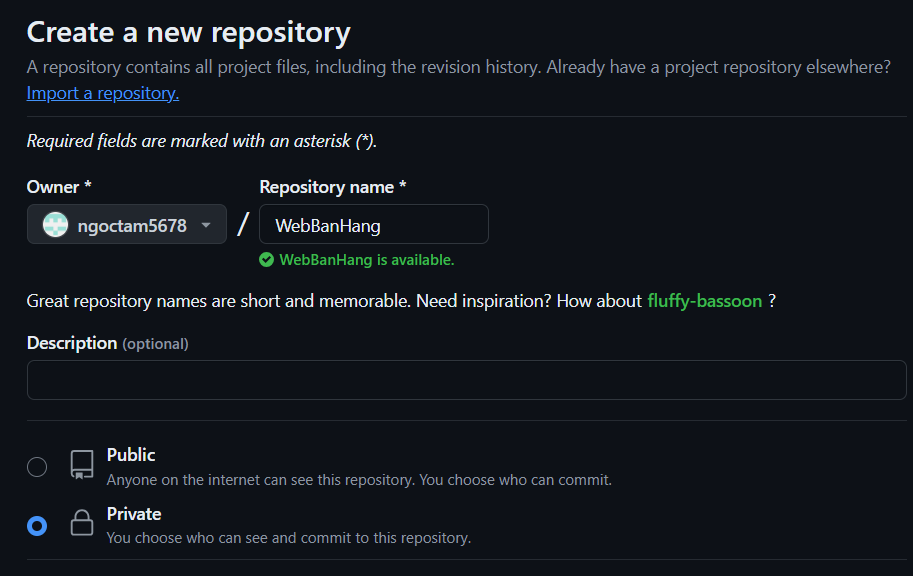
1. **Truy cập trang web:**

Mở trình duyệt và truy cập: http://localhost:3000

* 1. **Thực hành với GitHub**

1. Sau khi log in vào GitHub, chọn dấu + góc trên phải màn hình và chọn New Repository

2. Nhập tên Repository là WebBanHang và chọn thể loại là Private, tạo Repository đầu tiên

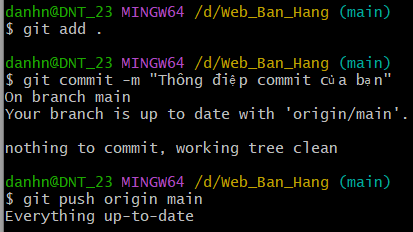
****

3. Di chuyển vào chính xác đường dẫn

**A black screen with colorful text

Description automatically generated**

1. Các bước bổ sung file lên github

****

# **TÀI LIỆU THAM KHẢO**

1. [Kiến trúc phần mềm phân tán](https://viblo.asia/p/tim-hieu-ve-he-phan-tan-phan-2-kien-truc-he-phan-tan-naQZRLyG5vx)
2. [Tìm hiểu về Cloud](https://viblo.asia/p/tim-hieu-ve-cloud-YWOZrEdwKQ0)
3. [Phong cách kiến ​​trúc Microservices](https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices)