**Shape

Description automatically generated**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙞🙜🕮🙞🙜

**Logo

Description automatically generated**

**BÁO CÁO THỰC NGHIỆM**

**CÔNG CỤ PHÁT TRIỂN PHẦN MỀM**

**Giáo viên hướng dẫn: Hà Mạnh Đào**

**Nhóm : 14**

**Lớp : 20222IT6003001**

**Sinh viên thực hiện : Đỗ Duy Bảo**

**Hoàng Văn Tuyến**

**Trần Huy Thắng**

**Trần Duy Toàn**

**Hà Nội, 2023**

# MỤC LỤC

[MỤC LỤC 2](#_Toc136740046)

[LỜI CẢM ƠN 1](#_Toc136740047)

[CHƯƠNG 1: TỔNG QUAN 1](#_Toc136740048)

[1. Tổng quan về công cụ phát triển phần mềm 1](#_Toc136740049)

[2. Lý do chọn đề tài 1](#_Toc136740050)

[3. Ứng dụng của Git và GitHub 2](#_Toc136740051)

[CHƯƠNG 2. GIT VÀ GITHUB 3](#_Toc136740052)

[1. Tổng quan về Git 3](#_Toc136740053)

[1.1. Git là gì? 3](#_Toc136740054)

[1.2. Hoạt động của Git 4](#_Toc136740055)

[1.3. Lợi ích của Git 5](#_Toc136740056)

[1.4. Tập lệnh của Git 6](#_Toc136740057)

[1.5. Cài đặt Git và thiết lập môi trường Git 6](#_Toc136740058)

[1.6. Kỹ thuật sử dụng Git 7](#_Toc136740059)

[2. Tổng quan về GitHub 10](#_Toc136740060)

[2.1 GitHub là gì ? 10](#_Toc136740061)

[2.2 So sánh Git với GitHub 11](#_Toc136740062)

[2.3. Đặc điểm, tính năng của GitHub 11](#_Toc136740063)

[2.4. Cài đặt và thiết lập tham số cho GitHub 12](#_Toc136740064)

[2.5 Kỹ thuật sử dụng GitHub 14](#_Toc136740065)

[2.6. Một số GitHub phổ biến 17](#_Toc136740066)

[CHƯƠNG 3: KẾT QUẢ NGHIÊN CỨU 17](#_Toc136740067)

[KẾT LUẬN 17](#_Toc136740068)

[TÀI LIỆU THAM KHẢO 18](#_Toc136740069)

# LỜI CẢM ƠN

Hiện nay, việc sử dụng Git và GitHub là rất phổ biến trong các dự án phần mềm. Đây là những công cụ không thể thiếu trong môi trường phát triển phần mềm hiện đại. Quản lý phiên bản của phiên bản mã nguồn, đồng bộ hóa các sửa đổi và kiểm tra các lỗi là rất quan trọng trong quá trình phát triển phần mềm. Vì vậy, việc sử dụng Git và GitHub để quản lý phiên bản mã nguồn và phát triển dự án là cần thiết để đảm bảo chất lượng và hiệu quả trong công việc của đội ngũ phát triển.

Ngoài ra, sử dụng Git và GitHub còn giúp cho các nhà phát triển phối hợp với nhau một cách dễ dàng. Họ có thể tiếp cận và chia sẻ mã nguồn của nhau, đồng thời theo dõi lịch sử thay đổi của dự án. Qua đó, các nhà phát triển có thể phát hiện và sửa đổi lỗi nhanh chóng, đồng thời cũng giúp đẩy nhanh quá trình phát triển của dự án.

Với sự phát triển của mô hình phân tán trong quản lý dự án phần mềm, Git và GitHub là những công cụ không thể thiếu trong việc quản lý và phát triển các dự án phần mềm. Thông qua việc nghiên cứu, hiểu rõ và sử dụng được các tính năng của Git và GitHub, các nhà phát triển có thể nâng cao hiệu quả làm việc và đảm bảo chất lượng phát triển dự án.

Đề tài "Tìm hiểu công cụ Git và GitHub" nhắm đến mục tiêu giúp người học hiểu rõ hơn về các khái niệm về Git và GitHub, cách sử dụng công cụ này để quản lý và phát triển các dự án phần mềm. Cụ thể, đề tài này sẽ được giới thiệu các tính năng của Git và GitHub, cách thức sử dụng Git để quản lý phiên bản mã nguồn và GitHub để lưu trữ và chia sẻ mã nguồn và dự án phần mềm. Ngoài ra, đề tài cũng sẽ giải đáp các thắc mắc và giới thiệu các thủ thuật khi sử dụng công cụ này để đảm bảo hiệu quả và đồng bộ hóa tốt nhất cho đội ngũ phát triển phần mềm.Đề tài được thực hiện bởi nhóm 14 .

# CHƯƠNG 1: TỔNG QUAN

## Tổng quan về công cụ phát triển phần mềm

Công cụ phát triển phần mềm là bất kỳ phần mềm hoặc ứng dụng nào được sử dụng cho quá trình phát triển phần mềm, bao gồm quản lý dự án, quản lý phiên bản, lập trình, kiểm thử và triển khai.

Trong quá trình phát triển phần mềm, các công cụ phát triển phần mềm phổ biến bao gồm các IDE (Integrated Development Environment) như Eclipse, Visual Studio và IntelliJ IDEA cho phép lập trình viên viết code phù hợp với các ngôn ngữ lập trình khác nhau và cho phép kiểm tra cú pháp, lỗi và định dạng.

Các công cụ quản lý phiên bản phát triển phần mềm phổ biến như Git và GitHub cho phép các nhà phát triển phần mềm quản lý các phiên bản của mã nguồn trong các dự án phát triển và kiểm soát các thay đổi trong quá trình làm việc nhóm.

Các công cụ kiểm thử phần mềm bao gồm các phần mềm kiểm thử tự động như Selenium, Appium, JUnit, Cucumber và có chất lượng sữa ổn định giúp đảm bảo rằng các phần mềm được phát triển đáp ứng đầy đủ yêu cầu

## Lý do chọn đề tài

Hiện nay, Git và GitHub là hai công cụ quản lý phiên bản phổ biến nhất trong lĩnh vực phát triển phần mềm, với sự phát triển không ngừng của công nghệ và môi trường phát triển. Các công cụ khác như SVN hay Mercurial đã bị thay thế bởi Git và GitHub do cách tiếp cận phát triển phần mềm phù hợp hơn.

Việc chọn đề tài tìm hiểu về Git và GitHub là vì hai công cụ này được sử dụng rộng rãi trong lĩnh vực phát triển phần mềm và đóng vai trò quan trọng trong việc quản lý phiên bản của mã nguồn. Sử dụng các công cụ này sẽ giúp cho quá trình phát triển phần mềm trở nên suôn sẻ và không gặp phải những trở ngại không đáng có trong quá trình làm việc nhóm. Hiểu rõ về Git và GitHub còn giúp cho nhà phát triển phần mềm có thể tối ưu hóa quá trình quản lý phiên bản và phát triển phần mềm một cách hiệu quả nhất.

Ngoài ra, GitHub còn cung cấp các tính năng xã hội và cho phép người dùng cộng tác và phối hợp trong việc phát triển phần mềm. GitHub cũng có API để tương tác với dữ liệu và tích hợp với các công cụ và ứng dụng khác giúp đơn giản hoá quá trình phát triển phần mềm.

Tuy nhiên, mặc dù Git và GitHub là hai công cụ quản lý phiên bản phổ biến và được sử dụng nhất, những công cụ VCS khác như SVN và Mercurial vẫn được sử dụng trong một số dự án phát triển phần mềm, tuỳ vào yêu cầu cụ thể của dự án.

Tóm lại, sự phổ biến của Git và GitHub trong lĩnh vực phát triển phần mềm đang ngày càng tăng, cung cấp các tính năng tiên tiến, đáp ứng các nhu cầu cụ thể của người dùng để quản lý các phiên bản và mã nguồn của dự án phát triển phần mềm.

## Ứng dụng của Git và GitHub

Git và GitHub được sử dụng rộng rãi trong quá trình phát triển phần mềm và có các ứng dụng quan trọng sau đây:

Quản lý phiên bản: Git là công cụ quản lý phiên bản phổ biến nhất trong lĩnh vực phát triển phần mềm. Nó giúp các nhà phát triển phần mềm quản lý mã nguồn, lịch sử và phiên bản của dự án phát triển phần mềm một cách hiệu quả. Điều này giúp đảm bảo rằng tất cả các thành viên trong nhóm có thể làm việc trên cùng một mã nguồn và chia sẻ các thay đổi một cách dễ dàng.

Phát triển phần mềm nhóm: GitHub cung cấp các tính năng xã hội và cho phép người dùng cộng tác và phối hợp trong việc phát triển phần mềm một cách linh hoạt. Các nhà phát triển có thể làm việc cùng nhau trên cùng một dự án và tạo ra các phiên bản mới cũng như giải quyết các vấn đề bảo mật.

Lưu trữ và chia sẻ mã nguồn: GitHub cung cấp một kho lưu trữ mã nguồn công cộng hoặc riêng tư cho người dùng, giúp các nhà phát triển phần mềm tiện lợi hơn trong việc chia sẻ và lưu trữ mã nguồn. Người sử dụng có thể chia sẻ code của họ để được đóng góp và được sửa đổi, hoặc có thể giữ mã nguồn riêng tư và chỉ chia sẻ với những người họ tin tưởng.

Kiểm tra và xây dựng liên tục (Continuous Integration/Continuous Deployment - CI/CD): GitHub cũng có các tính năng xây dựng liên tục để đảm bảo rằng mã nguồn được xây dựng và kiểm tra tự động mỗi khi có thay đổi, giúp giảm thiểu thời gian kiểm tra chất lượng và đảm bảo rằng phần mềm được phát triển một cách liền mạch.

Tích hợp với các công cụ khác: GitHub cũng có API để tương tác với dữ liệu và tích hợp với các công cụ và ứng dụng khác như các công cụ kiểm tra lỗi, dịch vụ điện toán đám mây và các công cụ phát triển phần mềm khác, giúp đơn giản hoá quá trình phát triển phần mềm.

# CHƯƠNG 2. GIT VÀ GITHUB

## Tổng quan về Git

### 1.1. Git là gì?

Git là một hệ thống kiểm soát phiên bản phân tán mã nguồn mở . Nó được thiết kế để xử lý các dự án nhỏ đến lớn với tốc độ và hiệu quả cao. Nó được phát triển để phối hợp công việc giữa các nhà phát triển. Kiểm soát phiên bản cho phép chúng tôi theo dõi và làm việc cùng với các thành viên trong nhóm của chúng tôi tại cùng một không gian làm việc.



Git là nền tảng của nhiều dịch vụ như GitHub và GitLab , nhưng chúng ta có thể sử dụng Git mà không cần sử dụng bất kỳ dịch vụ Git nào khác. Git có thể được sử dụng riêng tư và công khai .

Git được Linus Torvalds tạo ra vào năm 2005 để phát triển Linux Kernel. Nó cũng được sử dụng như một công cụ kiểm soát phiên bản phân tán quan trọng cho DevOps .

Git rất dễ học và có hiệu suất nhanh. Nó vượt trội so với các công cụ SCM khác như Subversion, CVS, Perforce và ClearCase.

Một số thuật ngữ sử dụng trong Git:

* Repository: Kho mã nguồn, tức là nơi lưu trữ mã nguồn của dự án phát triển phần mềm.
* Branch: Nhánh, là một phiên bản riêng lẻ của kho mã nguồn, cho phép các nhà phát triển làm việc trên cùng một kho mã nguồn đồng thời, mà không ảnh hưởng đến các lần sửa đổi của nhau.
* Commit: Lưu trữ một phiên bản của code vào kho mã nguồn. Hành động này được thực hiện bằng cách đánh dấu các thay đổi và lưu chúng vào kho.
* Merge: Hợp nhất, tức là quá trình kết hợp những thay đổi từ một nhánh trong kho mã nguồn vào nhánh khác.
* Pull request: Yêu cầu kéo, là một yêu cầu gửi cho chủ sở hữu kho mã nguồn về việc hợp nhất các thay đổi được thực hiện trong một nhánh riêng lẽ.
* Fork: Sao chép, là hành động tạo ra một bản sao của một kho mã nguồn của ai đó vào kho của mình.
* Tag: Nhãn, là một cách để đánh dấu một phiên bản hoặc một điểm dabgm của kho mã nguồn, giúp nhà phát triển dễ dàng truy cập và kiểm soát các phiên bản của dự án phần mềm.
* Pull: Kéo, là quá trình tải về các sửa đổi mới nhất từ kho mã nguồn.
* Push: Đẩy, là quá trình đưa các phiên bản được thực hiện từ cục bộ lên kho mã nguồn trong Git.

### 1.2. Hoạt động của Git

Git là một phần mềm quản lý phiên bản mã nguồn, được phát triển nhằm giúp các nhà phát triển phần mềm quản lý các thay đổi của mã nguồn, quản lý phiên bản của mã nguồn và phối hợp làm việc một cách hiệu quả.

Để sử dụng Git, các nhà phát triển phải khởi tạo một Repository, một kho chứa trong đó các tệp và thư mục chứa mã nguồn được lưu trữ. Các nhà phát triển sẽ phải thực hiện các thao tác sau đây:

Checkout checkout: Lấy dữ liệu từ kho lưu trữ và đưa nó vào trong thư mục làm việc cục bộ trên máy tính của người dùng.

Add: Tạo các danh sách thay đổi mới, các file có sự thay đổi hoặc được thêm mới và đưa chúng vào một danh sách chuẩn bị cho việc commit.

Commit: Lưu trữ các thay đổi được thực hiện vào bản ghi trong repo. Các bản ghi này sẽ chứa một thông điệp giải thích sự thay đổi được thực hiện và đưa nó vào kho lưu trữ.

Push: Tải lên các bản ghi được commit từ bản sao cục bộ của người dùng lên kho lưu trữ chính để làm update cho kho lưu trữ đó.

Pull: Lấy về các thay đổi mới nhất từ kho lưu trữ và đưa chúng vào kho lưu trữ cục bộ để cập nhật các thay đổi được thực hiện mới nhất.

Merge: Kết hợp sự thay đổi giữa các nhánh khác nhau trong kho lưu trữ và giải quyết xung đột sự thay đổi giữa các nhánh.

Nhờ vào các thao tác trên, Git đã trở thành một công cụ quản lý phiên bản phần mềm phổ biến và được sử dụng rộng rãi trong các dự án phát triển phần mềm của các công ty lớn và nhỏ trên toàn thế giới.

### 1.3. Lợi ích của Git



* Tiết kiệm thời gian:

Git là công nghệ nhanh như chớp. Mỗi lệnh chỉ mất vài giây để thực thi nên chúng ta tiết kiệm được rất nhiều thời gian so với việc đăng nhập vào tài khoản GitHub và tìm hiểu các tính năng của nó.

* Làm việc ngoại tuyến:

Một trong những lợi ích quan trọng nhất của Git là nó hỗ trợ làm việc ngoại tuyến . Nếu chúng tôi đang gặp sự cố kết nối internet, điều đó sẽ không ảnh hưởng đến công việc của chúng tôi. Trong Git, chúng ta có thể làm hầu hết mọi thứ cục bộ. Một cách tương đối, các CVS ​​khác như SVN bị hạn chế và thích kết nối với kho lưu trữ trung tâm hơn.

* Hoàn tác sai lầm:

Một lợi ích bổ sung của Git là chúng ta có thể hoàn tác sai lầm. Đôi khi hoàn tác có thể là một lựa chọn cứu tinh cho chúng ta. Git cung cấp tùy chọn hoàn tác cho hầu hết mọi thứ.

* Theo dõi các thay đổi:

Git hỗ trợ một số tính năng thú vị như Diff, Log và Status , cho phép chúng tôi theo dõi các thay đổi để chúng tôi có thể kiểm tra trạng thái, so sánh các tệp hoặc nhánh của mình.

### 1.4. Tập lệnh của Git

Dưới đây là một số lệnh cơ bản trong Git:

* git init: Khởi tạo kho chứa (repo) vào thư mục hiện tại.
* git clone: Sao chép (clone) kho chứa từ một kho trên internet hoặc từ một kho trên máy tính cục bộ.
* git add: Thêm (add) các thay đổi hoặc tệp mới vào danh sách chuẩn bị để commit.
* git commit: Lưu trữ sự thay đổi đã thực hiện vào kho lưu trữ tại phiên bản hiện tại.
* git push: Tải lên các thay đổi mới từ phiên bản hiện tại của kho lưu trữ lên kho lưu trữ trên internet.
* git pull: Lấy các thay đổi tại phiên bản mới nhất của kho lưu trữ xuống để cập nhật phiên bản hiện tại.
* git status: Xem trạng thái của kho lưu trữ hiện tại, bao gồm danh sách các tệp thay đổi hoặc chưa được thêm.
* git log: Hiển thị lịch sử các sự thay đổi đã thực hiện trong kho lưu trữ.
* git branch: Xem hoặc tạo ra mới một nhánh (branch) của code.
* git merge: Hợp nhất (merge) nhánh khác với nhánh hiện tại trong kho lưu trữ.
* git diff: Xem sự khác biệt giữa hai phiên bản hoặc giữa hai nhánh.
* git remote: Liên kết kho lưu trữ cục bộ với một kho lưu trữ từ xa.
* git reset: Đặt lại phiên bản trước đó của kho lưu trữ.
* git config: Cấu hình Git, bao gồm tên đăng nhập và mật khẩu.

Đó chỉ là một số lệnh cơ bản trong Git, và còn nhiều lệnh khác chứa nhiều tính năng và lựa chọn phức tạp hơn để giúp các nhà phát triển làm việc hiệu quả hơn.

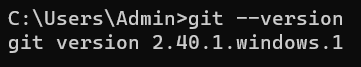
### 1.5. Cài đặt Git và thiết lập môi trường Git

#### 1.5.1. Cách cài đặt trên Windows

Tải xuống bản cài đặt Git tương ứng với hệ điều hành của bạn tại địa chỉ: <https://git-scm.com/downloads>

Chạy tệp cài đặt đã tải xuống và làm theo các hướng dẫn trên màn hình.

Mở cửa sổ terminal hoặc command prompt và xác minh Git đã được cài đặt thành công bằng cách nhập lệnh: git --version. Nếu Git được cài đặt thành công, phiên bản của nó sẽ được hiển thị trên terminal.



#### 1.5.2. Thiết lập môi trường Git

Thiết lập tên người dùng Git: Sử dụng lệnh dưới đây để thiết lập tên người dùng cho các commit:

*git config --global user.name "Tên của bạn"*

Thiết lập email người dùng Git: Sử dụng lệnh dưới đây để thiết lập địa chỉ email cho các commit:

*git config --global user.email "youremail@example.com"*

Thiết lập trình chỉnh sửa Git (Editor): Git sử dụng trình chỉnh sửa của máy tính của bạn để nhập thông tin commit của bạn và xử lý các thao tác khác. Bạn có thể sử dụng trình chỉnh sửa mặc định của Git hoặc thiết lập trình chỉnh sửa tùy chỉnh bằng cách sử dụng lệnh:

*git config --global core.editor "tên trình chỉnh sửa"*

Thiết lập cache thông tin đăng nhập: Nếu bạn không muốn nhập thông tin đăng nhập của mình mỗi khi bạn push hoặc pull dữ liệu từ kho lưu trữ, bạn có thể sử dụng lệnh sau để lưu trữ thông tin đăng nhập của bạn trên một cache:

*git config --global credential.helper cache*

Những thiết lập này sẽ giúp bạn bắt đầu sử dụng Git trên máy tính của mình và cải thiện trải nghiệm của bạn khi làm việc với Git.

### 1.6. Kỹ thuật sử dụng Git

* Tạo và sử dụng nhánh (branch): Điều này giúp đảm bảo rằng bạn có thể làm việc trên code một cách độc lập và tránh xảy ra xung đột khi merge (hợp nhất) các thay đổi.
* Tạo commit nhỏ và trực tiếp: Hãy tạo ra nhiều commit nhỏ trong quá trình làm việc thay vì làm một commit lớn với nhiều thay đổi khác nhau. Ngoài ra, bạn nên đảm bảo rằng tất cả các commit đều có thông tin trực quan, để người khác có thể hiểu được các thay đổi của bạn.
* Sử dụng lệnh git status thường xuyên: Git status đem lại thông tin chi tiết về trạng thái của kho lưu trữ của bạn, bao gồm danh sách các tệp đã được thay đổi hoặc chưa được đưa vào stage.
* Sử dụng kỹ thuật stash để lưu các thay đổi tạm thời: Kỹ thuật stash cho phép bạn tạm dừng việc làm việc trên một nhánh và lưu các thay đổi tạm thời, để tránh xung đột khi bạn chuyển sang một nhánh khác.
* Xem lại nội dung của commit trước khi push lên kho lưu trữ: Trước khi push lên kho lưu trữ, bạn nên kiểm tra lại nội dung của các commit để đảm bảo rằng không có thay đổi nào không cần thiết. Bạn có thể xem nội dung của các commit bằng lệnh git log hoặc bằng các công cụ quản lý kho lưu trữ.
* Tạo pull request và yêu cầu review: Nếu bạn đang làm việc trên một kho lưu trữ chung với nhiều người khác, bạn nên tạo một pull request để yêu cầu review code của người khác. Review code giúp đảm bảo rằng code của bạn đáp ứng đầy đủ các yêu cầu và điều kiện của dự án, tránh các lỗi đáng tiếc trước khi merge (hợp nhất) code vào nhánh chính.

##### Ví dụ:

Để minh họa cách sử dụng Git, hãy xem xét ví dụ sau:

Giả sử bạn đang làm việc trên một dự án phần mềm đơn giản với hai thành viên là A và B. Bắt đầu từ một kho lưu trữ trên GitHub, sau đó mỗi thành viên sẽ sao chép (clone) kho lưu trữ này và bắt đầu phát triển trên máy của riêng họ.

Khởi tạo kho lưu trữ

Trên GitHub, người tạo dự án sẽ tạo một kho lưu trữ mới. Cả hai thành viên A và B sẽ sao chép repo này bằng cách sử dụng lệnh sau trên terminal:

git clone <url của kho lưu trữ trên GitHub>

Tạo và làm việc trên nhánh

Mỗi thành viên sẽ làm việc trên một nhánh (branch) riêng biệt để tránh xung đột khi commit. Ví dụ, A sẽ tạo một nhánh mới và chuyển đổi sang nhánh đó:

git checkout -b feature-A

Khi đã chuyển sang nhánh mới, A sẽ thực hiện các thay đổi và commit vào code trên nhánh đó:

git add .

git commit -m "implement feature A"

Tương tự, B sẽ tạo và chuyển sang một nhánh mới khác để làm việc trên nhánh đó:

git checkout -b feature-B

Sau đó, B thực hiện các thay đổi và commit của mình trên nhánh đó:

git add .

git commit -m "implement feature B"

Kết hợp nhánh

Khi tất cả công việc đã được hoàn thành trên các nhánh riêng, nhánh chính (master branch) sẽ được kết hợp lại với nhánh feature-A và feature-B để tạo thành một phiên bản hoàn chỉnh của code.

Để kết hợp nhánh, A sẽ checkout về nhánh chính và pull (lấy xuống) những thay đổi mới nhất từ repo trên GitHub:

git checkout master

git pull

Sau đó, A tiến hành merge (hợp nhất) nhánh feature-A vào nhánh chính:

git merge feature-A

Lệnh merge sẽ tự động kết hợp các commit của feature-A vào nhánh chính và tạo ra một commit mới. Tương tự như vậy, B sẽ merge nhánh feature-B của mình vào nhánh chính.

Push lên repo trên GitHub

Sau khi kết hợp các nhánh thành công và đảm bảo code hoạt động đúng như mong muốn, A và B sẽ push (tải lên) các commit mới lên repo trên GitHub:

git push origin master

Nếu có xảy ra xung đột thì Git sẽ tạo ra một merge conflict mà người dùng cần phải giải quyết thủ công trước khi push lên repo.

Với việc sử dụng Git như vậy, hai thành viên A và B có thể làm việc độc lập trên các tính năng khác nhau của dự án mà không làm lỗi code của nhau, và sau đó họ có thể kết hợp lại thành một phiên bản hoàn chỉnh để tạo thành sản phẩm cuối cùng.

## Tổng quan về GitHub

### 2.1 GitHub là gì ?

GitHub là một dịch vụ lưu trữ kho lưu trữ Git. GitHub cũng tạo điều kiện thuận lợi với nhiều tính năng của nó, chẳng hạn như kiểm soát truy cập và cộng tác. Nó cung cấp một giao diện đồ họa dựa trên Web.

Nó cung cấp cả chức năng kiểm soát phiên bản phân tán và quản lý mã nguồn (SCM) của Git. Nó cũng hỗ trợ một số tính năng cộng tác như theo dõi lỗi, yêu cầu tính năng, quản lý tác vụ cho mọi dự án.



Các tính năng của GitHub

* Sự hợp tác
* Tích hợp vấn đề và theo dõi lỗi
* Biểu diễn đồ họa của các chi nhánh
* Lưu trữ kho lưu trữ Git
* Quản lý dự án
* quản lý nhóm
* lưu trữ mã
* Theo dõi và giao nhiệm vụ
* cuộc trò chuyện
* Wikisc

Lợi ích của GitHub

* Thật dễ dàng để đóng góp cho các dự án mã nguồn mở thông qua GitHub.
* Nó giúp tạo ra một tài liệu tuyệt vời.
* Bạn có thể thu hút nhà tuyển dụng bằng cách thể hiện công việc của bạn. Nếu bạn có hồ sơ trên GitHub, bạn sẽ có cơ hội được tuyển dụng cao hơn.
* Nó cho phép công việc của bạn xuất hiện trước công chúng.
* Bạn có thể theo dõi các thay đổi trong mã của mình qua các phiên bản.

### 2.2 So sánh Git với GitHub

GitHub và Git là hai khái niệm liên quan đến quản lý mã nguồn nhưng có một số điểm khác biệt cơ bản như sau:

* Git là một hệ thống quản lý mã nguồn phân tán (distributed version control system) được phát triển bởi Linus Torvalds. Git được sử dụng để quản lý phiên bản (versioning) và kiểm soát mã nguồn một cách dễ dàng và linh hoạt.
* GitHub là một nền tảng dịch vụ lưu trữ mã nguồn, cung cấp một cách tiếp cận thân thiện và dễ dàng cho các nhà phát triển khi quản lý mã nguồn.
* Git được cài đặt trên máy tính của người dùng và sử dụng dòng lệnh (command line) để thực hiện các thao tác quản lý thư viện mã nguồn.
* GitHub là dịch vụ lưu trữ mã nguồn được cung cấp trên đám mây (cloud) và có thể được truy cập từ bất kỳ máy tính nào có kết nối Internet thông qua trình duyệt web.
* Git cung cấp các tính năng cơ bản như quản lý phiên bản, đồng bộ hóa và đối tượng (object) de-duplication.
* GitHub cung cấp các tính năng bổ sung như quản lý issue, pull request, collaboration tools và các tính năng tích hợp của các ứng dụng khác.

Tóm lại, Git là một công cụ quản lý mã nguồn phổ biến, trong khi GitHub là một dịch vụ lưu trữ mã nguồn được xây dựng trên nền tảng Git. GitHub cung cấp nhiều tính năng bổ sung cho quản lý mã nguồn, thực hiện công việc của các nhà phát triển và làm cho quá trình các nhà phát triển làm việc trên cùng một dự án trở nên dễ dàng và hiệu quả hơn.

### 2.3. Đặc điểm, tính năng của GitHub

* Hosted Git repository: GitHub cung cấp một kho lưu trữ Git trên đám mây, cho phép lưu trữ mã nguồn của dự án trực tiếp trên GitHub. GitHub hỗ trợ một số nền tảng hệ điều hành khác nhau, đảm bảo inter-operability và tự động đồng bộ phiên bản
* Collaboration: GitHub là một nền tảng tích hợp các tính năng cộng tác (Collaboration) cho phép các thành viên của dự án tương tác với nhau một cách dễ dàng và hiệu quả. Các nhà phát triển có thể bình luận và thảo luận về mã nguồn của dự án, tạo ra các đánh giá, chỉnh sửa, cải thiện code và giải quyết các sự cố (issue) phát sinh trong quá trình phát triển.
* Pull requests: GitHub cung cấp tính năng Pull requests cho phép các nhà phát triển yêu cầu và thực hiện các sửa đổi vào mã nguồn của dự án. Pull request cho phép các thay đổi được kiểm tra và xem xét trước khi chúng được tích hợp vào kho chính.
* Tìm kiếm: GitHub cung cấp tính năng tìm kiếm tệp và nội dung trong kho quản lý của người dùng. Đây là cách nhanh nhất để tìm kiếm mã nguồn, tìm kiếm lỗi hoặc một số lỗi phát sinh trong quá trình xây dựng dự án phần mềm.
* Documentation: GitHub cung cấp cơ chế để tạo ra các tài liệu cho dự án phần mềm. Các tài liệu có thể được lưu trữ trong hệ thống và có thể được truy cập từ cùng một nền tảng dịch vụ.
* Integration: GitHub có thể tích hợp với nhiều tính năng khác nhau như CI/CD (Continuous Integration and Deployment) và các công cụ quản lý công việc như Trello và Asana. Điều này giúp đơn giản hóa dòng công việc cho quản lý mã nguồn và đồng thời tăng độ chuyên nghiệp và hiệu quả cho toàn bộ quá trình phát triển phần mềm.

### 2.4. Cài đặt và thiết lập tham số cho GitHub

#### 2.4.1. Cài đặt GitHub cho Windows

1. Tải và cài đặt Git cho Windows: Đầu tiên, bạn cần tải và cài đặt Git cho Windows theo hướng dẫn tại https://git-scm.com/download/win. Sau khi tải xuống, mở tệp cài đặt và thực hiện theo hướng dẫn để cài đặt Git lên máy tính của bạn.
2. Đăng ký tài khoản GitHub: Bạn cần đăng ký một tài khoản GitHub nếu bạn chưa có một tài khoản. Bạn có thể đăng ký tại https://github.com/signup.
3. Tạo kho lưu trữ (repository): Sau khi đăng nhập vào tài khoản GitHub, bạn có thể tạo một kho lưu trữ cho dự án của mình bằng cách nhấp vào nút "Create a new repository".
4. Clone kho lưu trữ: Bạn cần sao chép kho lưu trữ từ GitHub về máy tính của bạn để có thể làm việc với dự án của mình. Để sao chép kho lưu trữ, mở Git Bash và chạy lệnh sau:

*git clone https://github.com/your-username/your-repository.git*

Thay your-username và your-repository bằng tên người dùng và tên kho lưu trữ của bạn trên GitHub.

1. Thiết lập thông tin người dùng: Để xác định ai đang thực hiện thay đổi trên dự án phần mềm, bạn cần thiết lập tên và email của mình cho Git bằng cách chạy các lệnh sau trong Git Bash:

*git config --global user.name "Tên của bạn"*

*git config --global user.email "Địa chỉ email của bạn"*

1. Tạo và thực hiện các thay đổi: Bây giờ bạn đã có thể thực hiện các thay đổi trên mã nguồn của dự án và đẩy chúng lên kho lưu trữ GitHub. Để thực hiện các thay đổi, hãy sử dụng các lệnh như "git add", "git commit" và "git push".

#### 2.4.2. Thiết lập tham số cho GitHub

Để thiết lập các tham số cho GitHub, bạn có thể sử dụng các lệnh Git để cấu hình các tùy chọn và thông tin như sau:

1. Thiết lập thông tin người dùng: Để xác định ai đang thực hiện thay đổi trên dự án phần mềm, bạn có thể sử dụng các lệnh Git sau để thiết lập tên và email người dùng:

*git config --global user.name "Tên của bạn"*

*git config --global user.email "Địa chỉ email của bạn"*

1. Cấu hình mặc định cho trình soạn thảo: Nếu bạn sử dụng một trình soạn thảo khác với trình soạn thảo mặc định, bạn cần cấu hình Git để sử dụng trình soạn thảo đó. Ví dụ, để sử dụng trình soạn thảo Visual Studio Code trên Windows, bạn có thể sử dụng lệnh sau:

*git config --global core.editor "code --wait"*

1. Thiết lập mặc định cho hành động sau khi thực hiện merge: Khi sử dụng tính năng merge của Git, bạn có thể sử dụng các lệnh sau để cấu hình hành động mặc định sau khi thực hiện merge:

*git config --global merge.conflictstyle diff3*

*git config --global merge.tool vimdiff*

1. Cấu hình thông tin proxy: Nếu bạn sử dụng một máy chủ proxy để truy cập Internet, bạn có thể sử dụng các lệnh sau để cấu hình thông tin proxy cho Git:

*git config --global http.proxy http://proxy-server-url:proxy-port*

*git config --global https.proxy http://proxy-server-url:proxy-port*

1. Cấu hình SSH key: Nếu bạn muốn sử dụng phương thức xác thực SSH khi truy cập GitHub thay vì phương thức HTTP, bạn cần cấu hình khóa SSH trên kho lưu trữ của mình. Bạn có thể sử dụng các lệnh sau để thiết lập khóa SSH của bạn:

*ssh-keygen -t rsa -b 4096 -C "youremail@example.com"*

*eval "$(ssh-agent -s)"*

*ssh-add ~/.ssh/id\_rsa*

Sau đó, bạn có thể sao chép khóa công khai SSH và thêm nó vào tài khoản GitHub của mình để sử dụng phương thức xác thực SSH khi truy cập kho lưu trữ.

### 2.5 Kỹ thuật sử dụng GitHub

Dưới đây là một số kỹ thuật cơ bản để sử dụng GitHub hiệu quả:

* Tạo và quản lý kho lưu trữ (repository): Sau khi đăng nhập vào tài khoản GitHub, bạn có thể tạo một kho lưu trữ mới bằng cách nhấp vào nút "New repository". Bạn cần cung cấp tên kho lưu trữ, mô tả và các tùy chọn khác. Bạn có thể quản lý các kho lưu trữ của mình bằng các tính năng như fork (nhánh), clone (sao chép) và pull request (yêu cầu kéo).
* Tạo và quản lý các nhánh (branches): Trong Git và GitHub, các nhánh được sử dụng để phát triển và quản lý các tính năng, sửa lỗi, thử nghiệm và phát hành mã nguồn. Bạn có thể tạo các nhánh mới trên kho lưu trữ của mình bằng cách sử dụng các lệnh Git hoặc trên giao diện GitHub.
* Thực hiện các thay đổi trên nhiều nhánh: Khi làm việc trên một dự án, bạn có thể đặt các thay đổi của mình trên các nhánh khác nhau để giữ cho mã nguồn ổn định trên nhánh chính. Bạn có thể sử dụng các lệnh Git để chuyển đổi giữa các nhánh và thực hiện các thay đổi cần thiết.
* Sử dụng Pull Request: Khi bạn muốn đưa các thay đổi của mình vào kho lưu trữ chính, bạn có thể sử dụng tính năng pull request để yêu cầu có ai đó duyệt và chấp nhận các thay đổi của bạn. Một pull request cũng cho phép bạn thảo luận với các thành viên khác của dự án và tìm kiếm sự hỗ trợ.
* Sử dụng issue tracker: Nếu bạn muốn báo cáo các lỗi hoặc đề xuất tính năng mới cho dự án, bạn có thể sử dụng tính năng issue tracker của GitHub. Điều này giúp cho việc tìm kiếm, theo dõi và giải quyết các vấn đề liên quan đến dự án trở nên dễ dàng hơn.
* Collaborate (hợp tác) để tạo ra những sản phẩm tốt hơn: Một trong những điều tốt nhất về GitHub là bạn có thể hợp tác với các nhà phát triển khác trên cùng một dự án. Bạn có thể đóng góp vào các dự án open source cũng như làm việc với các nhóm phát triển khác nhau để tạo ra các sản phẩm tốt hơn.

##### Ví dụ:

Ví dụ: Giả sử bạn đang tham gia vào một dự án open source trên GitHub có tên Example Project. Dưới đây là một số kỹ thuật sử dụng GitHub để làm việc trên dự án này:

Fork (nhánh) dự án: Đầu tiên, bạn cần nhấp vào nút "Fork" để tạo một bản sao của chính dự án trên tài khoản GitHub của bạn. Điều này cho phép bạn làm việc trên dự án mà không ảnh hưởng đến bản gốc.

Clone (sao chép) kho lưu trữ (repository): Tiếp theo, bạn cần sao chép kho lưu trữ của dự án bằng cách sử dụng lệnh Git sau:

git clone https://github.com/your-username/example-project.git

Thay your-username bằng tên người dùng của bạn trên GitHub.

Tạo và chuyển đổi giữa các nhánh (branches): Bạn có thể tạo một nhánh mới để làm việc với các tính năng, sửa lỗi hoặc thử nghiệm. Để tạo một nhánh mới, hãy sử dụng lệnh Git sau:

git checkout -b feature/new-feature

Sau đó, bạn có thể thực hiện các thay đổi trên nhánh mới này và commit (xác nhận) chúng.

Thực hiện pull request: Sau khi đã thực hiện các thay đổi trên nhánh, bạn có thể yêu cầu duyệt và chấp nhận các thay đổi bằng cách sử dụng tính năng pull request. Điều này cho phép các thành viên khác trong dự án xem, đánh giá và cho ý kiến ​​về các thay đổi của bạn. Để tạo một pull request, hãy làm theo các bước sau:

Điều hướng đến thông tin kho lưu trữ của dự án và nhấp vào nút "New pull request".

Chọn nhánh mà bạn muốn chuyển đến.

Cung cấp các thông tin và mô tả về các thay đổi của bạn.

Nhấp vào nút "Create pull request".

Sử dụng issue tracker: Nếu bạn muốn báo cáo các lỗi hoặc đề xuất tính năng mới cho dự án, bạn có thể sử dụng tính năng issue tracker của GitHub để tạo các vấn đề mới. Điều này giúp các thành viên của dự án liên lạc và giải quyết các vấn đề nhanh chóng. Bạn có thể tạo một vấn đề mới bằng cách nhấp vào nút "New issue" và cung cấp thông tin và mô tả chi tiết về vấn đề của bạn.

Tóm lại, bằng cách sử dụng các kỹ thuật như fork dự án, clone kho lưu trữ, tạo và chuyển đổi giữa các nhánh, thực hiện pull request và sử dụng issue tracker, bạn có thể làm việc hiệu quả trên các dự án GitHub, hợp tác với những người khác và tạo ra các sản phẩm tốt hơn.

2.6. API của GitHub

GitHub cung cấp một loạt các API cho phép các nhà phát triển tương tác với dữ liệu của GitHub, bao gồm cả thông tin người dùng, kho lưu trữ, trạng thái yêu cầu kéo (PRs), vấn đề và nhiều loại dữ liệu khác. Dưới đây là một số API phổ biến của GitHub:

* REST API: Đây là API cơ bản và phổ biến nhất của GitHub. Nó cho phép các nhà phát triển tương tác với dữ liệu GitHub thông qua các yêu cầu HTTP đơn giản như GET, POST, PUT, DELETE. REST API cung cấp quyền truy cập vào hầu hết các dữ liệu được hiển thị trên giao diện GitHub.com.
* GraphQL API: Đây là một API mới và được phát triển bởi GitHub. GraphQL được thiết kế để có thể truy vấn và lấy dữ liệu theo nhu cầu của một ứng dụng cụ thể. GraphQL API được cung cấp nhằm giải quyết những vấn đề mà REST API gặp phải, như mất khả năng hiệu suất, quản lý phiên nhiều lần yêu cầu và khó khăn trong việc tôi ẩn tùy chọn đối tượng.
* OAuth API: Đây là một phiên bản của giao thức OAuth đang được sử dụng rộng rãi trong việc xác thực bên thứ ba. Nó cho phép ứng dụng truy cập vào tài khoản GitHub của người dùng, nhưng không bảo mật mật khẩu của người dùng. Thay vào đó, OAuth tạo ra một token truy cập để đại diện cho người dùng và được sử dụng trong các yêu cầu API.
* Webhooks API: Đây là API cho phép các ứng dụng đăng ký lắng nghe các sự kiện xảy ra trên dịch vụ GitHub để có thể phản hồi một cách tự động. Khi một sự kiện xảy ra trên GitHub, Webhooks sẽ gửi một thông báo đến ứng dụng đăng ký để ứng dụng có thể thực hiện hành động phù hợp.
* GitHub Apps API: Đây là một loại ứng dụng được phát triển bởi GitHub, dành riêng cho công việc liên quan đến việc tương tác với các kho lưu trữ của dịch vụ. GitHub Apps API hỗ trợ nhiều chức năng, bao gồm truy cập dữ liệu nen tảng, xác thực độc lập và giải quyết vấn đề bảo mật.

Các API của GitHub cho phép các nhà phát triển truy cập và tương tác với dữ liệu GitHub của mình một cách linh hoạt và tiện lợi, bao gồm cả thông tin người dùng, kho lưu trữ, trạng thái yêu cầu kéo (PRs), vấn đề và nhiều loại dữ liệu khác.

### 2.6. Một số GitHub phổ biến

GitHub là nền tảng quản lý dự án và phiên bản phân tán được sử dụng phổ biến bởi các nhà phát triển trên toàn thế giới. Dưới đây là một số GitHub phổ biến:

* Node.js: Node.js là một nền tảng phát triển ứng dụng máy chủ được xây dựng trên JavaScript. Nó được sử dụng rộng rãi bởi các nhà phát triển để xây dựng các ứng dụng web, ứng dụng di động và ứng dụng máy chủ.
* Angular: Angular là một khung làm việc (framework) phát triển ứng dụng web được sử dụng rộng rãi bởi các nhà phát triển. Nó được phát triển bởi Google và cung cấp một số tính năng tiện ích để phát triển ứng dụng web phức tạp.
* React: React là một thư viện JavaScript được sử dụng để phát triển các giao diện người dùng (UI) bằng cách sử dụng các thành phần độc lập. Nó được phát triển bởi Facebook và được sử dụng rộng rãi bởi các nhà phát triển để phát triển các ứng dụng web phức tạp.
* Python: Python là một ngôn ngữ lập trình phổ biến được sử dụng để phát triển nhiều loại ứng dụng, từ các ứng dụng web đơn giản đến các hệ thống máy tính phức tạp.
* TensorFlow: TensorFlow là một thư viện phổ biến cho phát triển các mô hình học máy và trí tuệ nhân tạo. Nó được sử dụng để phát triển các ứng dụng trong các lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên và ứng dụng trí tuệ nhân tạo khác.
* VS Code: VS Code là một trình biên tập mã nguồn mở miễn phí được phát triển bởi Microsoft. Nó được sử dụng rộng rãi bởi các nhà phát triển để phát triển các ứng dụng web và phần mềm.
* Bootstrap: Bootstrap là một khung làm việc (framework) CSS bao gồm một số định dạng và phong cách được sử dụng rộng rãi để phát triển các giao diện người dùng web đẹp và linh hoạt.

# CHƯƠNG 3: KẾT QUẢ NGHIÊN CỨU

“ Cần ảnh cho bài làm chung của nhóm”

# KẾT LUẬN

Với sự hướng dẫn của giảng viên Hà Mạnh Đào về đề tài ***“Tìm hiểu Git và GitHub”*** đã giúp nhóm em hoàn thành được đề tài với những kết quả đã nêu ra. Tìm hiểu tổng quancông cụ phát triển phần mềm. Tìm hiểu về công cụ Git và GitHub trong việc phát triển phần mềm

So sánh được sự khác nhau giữa Git và Github với một số công cụ khác. Nắm bắt được những lệnh hay dùng trên Git và

Lợi ích mà Git và Github mang lại cho các Dev về mặt quản lý phiên bản là không thể phủ nhận. Tuy nhiên, do Git và Github chỉ là một công cụ để quản lý phiên bản cho nên nó không thể bao quát được nội dung của phát triển phần mềm.

Quy trình thực hiện trên thực tế có được sự tham gia đóng góp của nhiều người với những vai trò khác nhau và việc phát triển phần mềm cũng vậy. Do điều kiện chưa được tiếp xúc với môi trường phát triển phần mềm thực tế nên không thể làm rõ được vai trò quyền hạn của từng người trong quá trình thực hiện quản lý.

Git và Github còn hỗ trợ nhiều công cụ khác trong quá trình phát triển phần mềm và kết hợp với việc phát triển phần mềm để tạo ra quy trình phát triển phần mềm chuyên nghiệp như: Quản lý nhóm, quản lý tiến độ công việc,..

Trong qua trình thực hiện báo cáo tài liệu củ nhóm còn nhiều thiếu sót mang thầy góp ý để báo cáo của chúng em hoàn thiện hơn. Chúng em xin chân thành cảm ơn !

# TÀI LIỆU THAM KHẢO

<https://www.javatpoint.com/git>

https://www.javatpoint.com/github