

# Buổi 1: Tổng quan, Môi trường & Git Workflow

---

## Ôn lại buổi trước

Buổi 1 là điểm xuất phát nên chưa có nội dung nào trước đó. Bạn chỉ cần:

- Đảm bảo máy tính ổn định, có quyền cài phần mềm và đủ dung lượng đĩa.
- Chuẩn bị sẵn tài khoản GitHub, Gmail (dùng cho các dịch vụ tích hợp sau này).
- Xem trước tổng quan về nghề Backend để hiểu lý do chúng ta học Spring Boot.

## Kiến thức

### 1. Backend Architecture: Client - Server - Database

**Giải thích:** Khi bạn mở ứng dụng đặt xe trên điện thoại và nhấn "Đặt xe", điều gì xảy ra?

1. **Client (Ứng dụng trên điện thoại):** Gửi yêu cầu "Tôi muốn đặt xe từ điểm A đến điểm B"
2. **Server (Backend - nơi chúng ta sẽ xây dựng):** Nhận yêu cầu, xử lý logic (tính giá, lưu thông tin đặt xe)
3. **Database (Cơ sở dữ liệu):** Lưu trữ thông tin (thông tin khách hàng, lịch sử đặt xe)

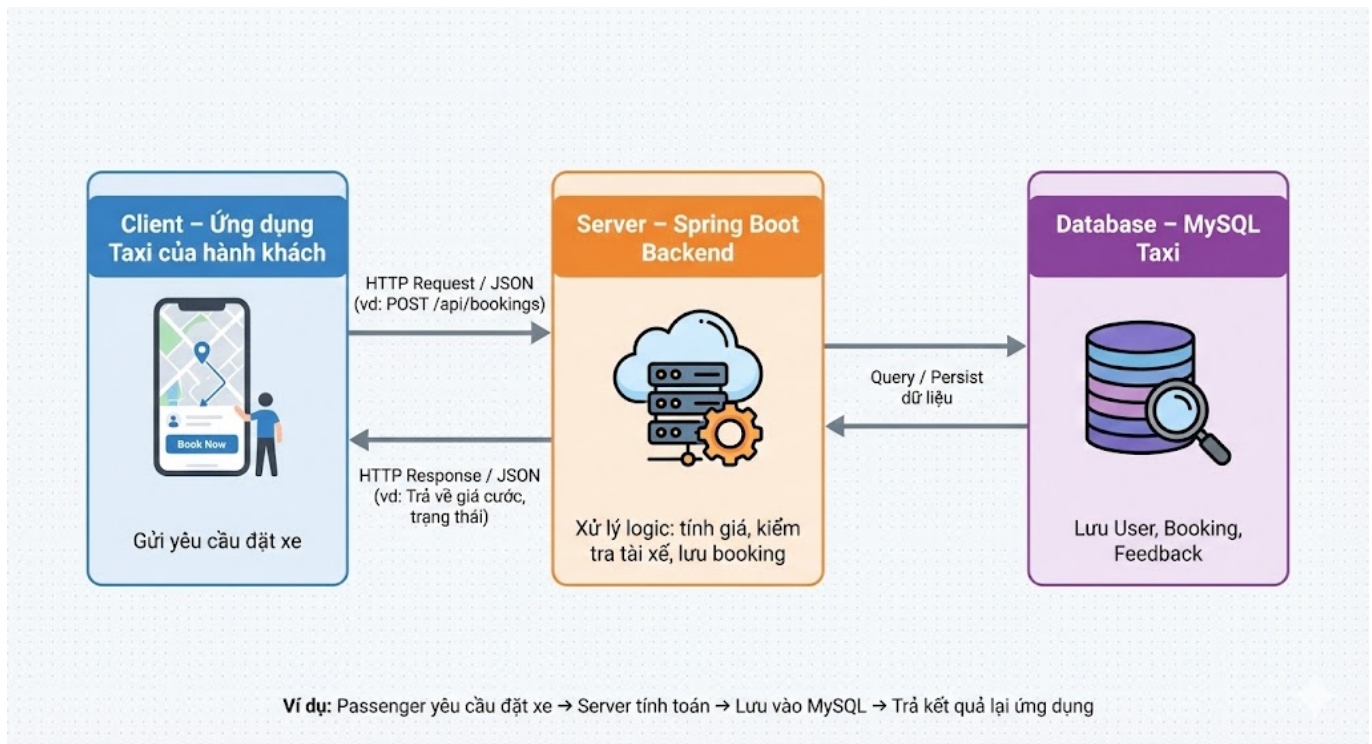
**Ví dụ thực tế:**

- Bạn vào nhà hàng (Client)
- Bạn gọi món với nhân viên (Server nhận yêu cầu)
- Nhân viên ghi vào sổ (Database lưu thông tin)
- Nhân viên xử lý và mang món ra (Server trả kết quả)

**Trong dự án Taxi:**

- **Client:** Ứng dụng đặt xe (có thể là web app hoặc mobile app)
- **Server:** Spring Boot Backend (chúng ta sẽ xây dựng)
- **Database:** MySQL (lưu thông tin user, booking, driver)

**Sơ đồ Client ↔ Server ↔ Database trong dự án Taxi:**



### Giải thích sơ đồ:

- **Client:** Ứng dụng của hành khách gửi HTTP Request dạng JSON, ví dụ `POST /api/bookings`, để yêu cầu đặt xe.
- **Server:** Spring Boot Backend nhận request, tính giá, tìm tài xế phù hợp và trả HTTP Response chứa kết quả (giá cước, trạng thái).
- **Database:** MySQL Taxi lưu thông tin User/Booking/Feedback. Server đọc/ghi dữ liệu liên tục để đảm bảo lịch sử chuyến đi không bị mất.
- Hai mũi tên giữa Client ↔ Server biểu thị JSON qua lại; mũi tên giữa Server ↔ Database biểu thị câu lệnh SQL truy vấn và lưu trữ.

## 2. Cài đặt môi trường

### 2.1. JDK 17+ (Java Development Kit)

**Giải thích:** JDK giống như bộ công cụ để xây nhà. Bạn cần có búa, cưa, đinh... thì mới xây được nhà. JDK là bộ công cụ để viết và chạy chương trình Java.

#### Cách kiểm tra:

```
java -version
```

#### Kết quả mong đợi:

```
openjdk version "17.0.x" ...
```

## Nếu chưa có JDK:

- **macOS:** Cài qua Homebrew: `brew install openjdk@17`
  - **Windows:** Tải từ [Oracle JDK](#) hoặc [OpenJDK](#)
- 

## 2.2. VS Code với các Extensions

**Giải thích:** VS Code giống như một chiếc bút thông minh. Nó không chỉ viết code, mà còn gợi ý cho bạn, tìm lỗi, và giúp bạn làm việc nhanh hơn. Extensions giống như các plugin để bút thông minh này làm được nhiều việc hơn.

### Cài đặt VS Code:

- Tải từ: <https://code.visualstudio.com/>
- Cài đặt như bình thường

### Các Extensions cần cài (theo thứ tự ưu tiên):

#### 1. Extension Pack for Java (Microsoft)

- Tại sao cần: Giúp VS Code hiểu code Java, gợi ý code, tìm lỗi
- Cách cài: Mở VS Code → Nhấn `Cmd+Shift+X` (macOS) hoặc `Ctrl+Shift+X` (Windows) → Tìm "Extension Pack for Java" → Install

#### 2. Spring Boot Extension Pack (VMware)

- Tại sao cần: Hỗ trợ đặc biệt cho Spring Boot (tạo project, chạy app, debug)
- Cách cài: Tìm "Spring Boot Extension Pack" → Install

#### 3. Spring Boot Tools (VMware)

- Tại sao cần: Công cụ chuyên biệt cho Spring Boot
- Lưu ý: Thường đã được bao gồm trong Spring Boot Extension Pack

#### 4. Spring Initializr Java Support (Microsoft)

- Tại sao cần: Tạo Spring Boot project trực tiếp trong VS Code
- Lưu ý: Thường đã được bao gồm trong Spring Boot Extension Pack

#### 5. Maven for Java (Microsoft)

- Tại sao cần: Quản lý thư viện (dependencies) cho project
- Lưu ý: Thường đã được bao gồm trong Extension Pack for Java

#### 6. GitLens (GitKraken) - Tùy chọn nhưng rất hữu ích

- Tại sao cần: Xem lịch sử code, ai sửa gì, khi nào
- Cách cài: Tìm "GitLens" → Install

### Cách kiểm tra extensions đã cài:

- Mở VS Code → Nhấn `Cmd+Shift+X` → Xem danh sách "Installed"
-

---

## 2.3. Postman hoặc Thunder Client

**Giải thích:** Khi bạn viết API (ví dụ: API đặt xe), bạn cần cách để test xem API có hoạt động không. Postman/Thunder Client giống như một công cụ để "gọi điện" đến API và xem kết quả.

### Lựa chọn 1: Postman (Tool bên ngoài)

- Tải từ: <https://www.postman.com/downloads/>
- Ưu điểm: Mạnh mẽ, nhiều tính năng

### Lựa chọn 2: Thunder Client (Extension trong VS Code) - Khuyến dùng

- Tại sao: Tích hợp sẵn trong VS Code, không cần mở app riêng
  - Cách cài: Tìm "Thunder Client" trong Extensions → Install
  - Cách dùng: Sau khi cài, bạn sẽ thấy icon Thunder Client ở thanh bên trái VS Code
- 

## 2.4. Git

**Giải thích:** Git giống như một cuốn sổ ghi chép tự động. Mỗi khi bạn thay đổi code, Git ghi lại. Nếu bạn làm hỏng code, bạn có thể "quay ngược thời gian" về phiên bản trước đó.

### Cách kiểm tra:

```
git --version
```

### Kết quả mong đợi:

```
git version 2.x.x
```

### Nếu chưa có Git:

- **macOS:** Thường đã có sẵn, hoặc cài qua Homebrew: `brew install git`
  - **Windows:** Tải từ <https://git-scm.com/download/win>
- 

## 3. VS Code cơ bản

### 3.1. Mở folder project

#### Cách làm:

1. Mở VS Code
2. File → Open Folder... (hoặc `Cmd+O` / `Ctrl+O`)
3. Chọn folder chứa project của bạn

**Ví dụ:** Nếu bạn có folder `taxi-booking-backend` trên Desktop, bạn mở folder đó.

---

### 3.2. Terminal tích hợp

**Giải thích:** Terminal giống như cửa sổ dòng lệnh. Thay vì click chuột, bạn gõ lệnh để làm việc.

**Cách mở Terminal trong VS Code:**

- Cách 1: **View** → **Terminal** (hoặc **Ctrl+`** - dấu backtick)
- Cách 2: **Terminal** → **New Terminal**
- Cách 3: Nhấn **Ctrl+Shift+(Windows/Linux)** hoặc **Cmd+** (macOS)

**Ví dụ sử dụng Terminal:**

```
# Kiểm tra Java version
java -version

# Kiểm tra Git version
git --version

# Xem các file trong folder hiện tại
ls # macOS/Linux
dir # Windows
```

---

### 3.3. Command Palette

**Giải thích:** Command Palette giống như một menu tìm kiếm. Bạn gõ tên lệnh, VS Code sẽ tìm và thực hiện.

**Cách mở:**

- **Cmd+Shift+P** (macOS) hoặc **Ctrl+Shift+P** (Windows/Linux)

**Ví dụ sử dụng:**

- Gõ "Java: Clean Java Language Server Workspace" → Enter (để reset Java extension nếu có lỗi)
- Gõ "Spring Initializr: Create a Maven Project" → Enter (để tạo Spring Boot project mới)

---

### 3.4. Run và Debug Spring Boot app từ VS Code

**Giải thích:** Khi bạn viết code xong, bạn cần chạy nó để xem có hoạt động không. VS Code có thể chạy và debug (tìm lỗi) Spring Boot app ngay trong editor.

**Cách Run:**

1. Tìm file có **@SpringBootApplication** (thường là file **TaxiBookingBackendApplication.java**)
2. Click vào nút "Run" bên cạnh method **main**
3. Hoặc nhấn **F5** (sẽ hỏi bạn chọn "Java" để debug)

### Cách Debug:

1. Click vào số dòng bên trái để đặt breakpoint (điểm dừng)
2. Nhấn **F5** để chạy ở chế độ debug
3. Khi code chạy đến breakpoint, nó sẽ dừng lại
4. Bạn có thể xem giá trị các biến, từng bước chạy code

### Ví dụ trong dự án Taxi:

- Đặt breakpoint trong Controller khi nhận request đặt xe
- Chạy debug → Gọi API từ Postman
- Xem từng bước: Controller nhận request → Service xử lý → Repository lưu vào DB

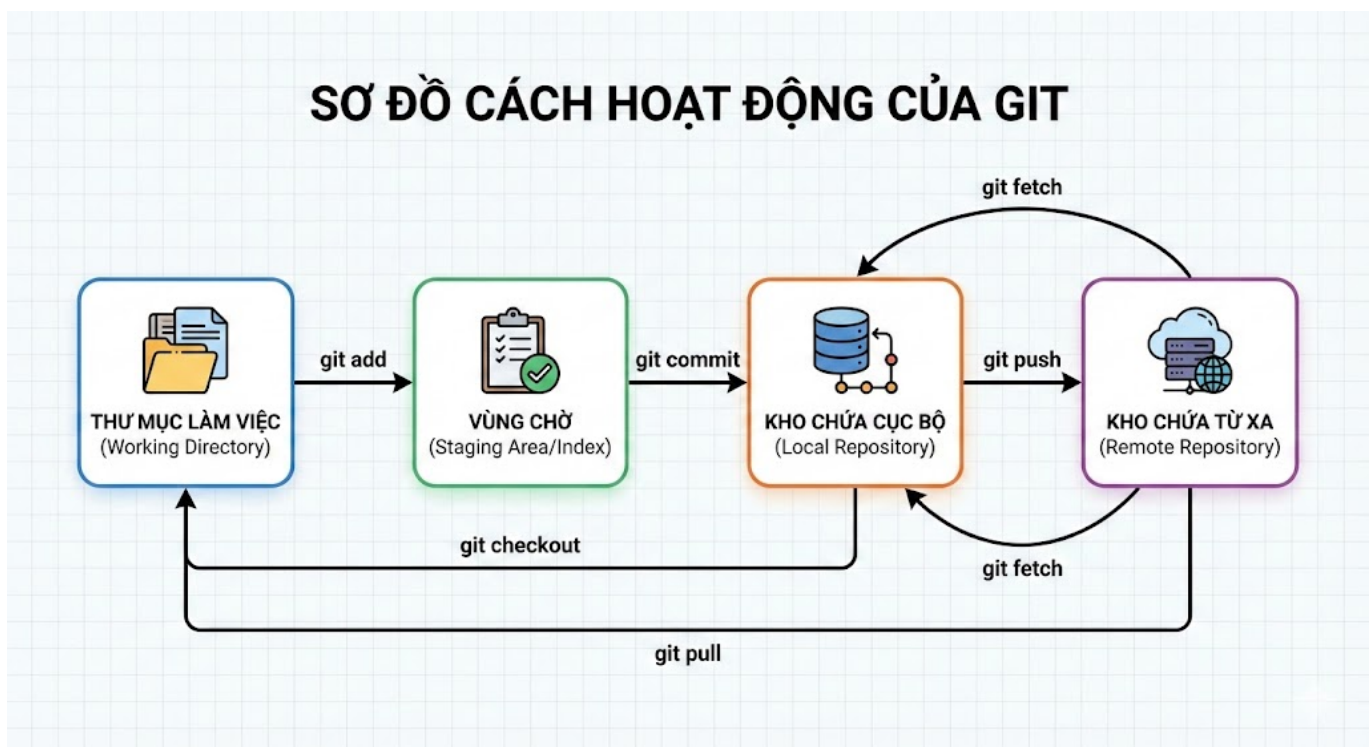
## 4. Git cơ bản

### 4.1. Các lệnh Git cơ bản

**Giải thích:** Git hoạt động theo 3 giai đoạn:

1. **Working Directory:** Nơi bạn đang viết code
2. **Staging Area:** Nơi bạn "chuẩn bị" code để lưu
3. **Repository:** Nơi Git lưu trữ code đã commit

**Sơ đồ tổng quan:** Hình dưới đây mô tả dòng chảy **git add** → **git commit** → **git push** và cách **git fetch**, **git pull**, **git checkout** đưa code đi giữa các khu vực. Khi học, hãy mở sơ đồ này trong VS Code (Explorer → [images/buoi1/git-follow.png](#)) để vừa đọc vừa đối chiếu các lệnh.



**Giải thích nhanh sơ đồ:**

- **Working Directory (Thư mục làm việc):** Bạn đang viết code Taxi Booking tại đây. Lệnh **git status** cho biết những file nào vừa sửa, giống như kiểm tra còn món nào chưa rửa trong bếp.

- **Staging Area (Vùng chờ):** Khi chạy `git add BookingController.java`, bạn nói với Git “món này đã sẵn sàng để chụp ảnh”. Các file trong vùng này sẽ có mặt trong lần commit kế tiếp.
- **Local Repository (Kho cục bộ):** `git commit -m "..."` chụp lại toàn bộ thay đổi đã stage. Đây là lịch sử nằm ngay trên máy bạn, có thể xem lại bằng `git log`.
- **Remote Repository (Kho từ xa):** `git push` gửi lịch sử ở máy bạn lên GitHub để cả nhóm taxi-backend thấy được. Nếu đồng đội cập nhật trước, hãy `git fetch` hoặc `git pull` để kéo code mới về rồi tiếp tục làm việc.
- **git checkout:** Cho phép bạn quay ngược về một commit khác hoặc chuyển sang branch mới, tương tự việc mở một bản nháp khác để thử ý tưởng.

#### Ví dụ đơn giản:

- Bạn viết code (Working Directory)
- Bạn "chọn" những file muốn lưu: `git add` (Staging Area)
- Bạn "chụp ảnh" trạng thái hiện tại: `git commit` (Repository)
- Bạn "gửi ảnh" lên GitHub: `git push`

#### Các lệnh cơ bản:

```
# Clone (tải về) repository từ GitHub
git clone <URL-của-GitHub-repo>
# Ví dụ: git clone https://github.com/Hau-ThucTap/taxi-booking-backend.git

# Khởi tạo Git repository trong folder hiện tại (nếu tạo project mới)
git init

# Xem trạng thái các file (file nào đã thay đổi, file nào chưa được add)
git status

# Xem danh sách các branch
git branch

# Tạo branch mới và chuyển sang branch đó
git checkout -b tên-branch-mới

# Chuyển sang branch khác
git checkout tên-branch

# Thêm file vào staging area (chuẩn bị để commit)
git add <tên-file>
# Hoặc thêm tất cả file: git add .

# Commit (lưu trạng thái hiện tại với một thông điệp)
git commit -m "Thông điệp mô tả thay đổi"

# Xem lịch sử commit
git log

# Kết nối với GitHub repository (nếu tạo project mới)
git remote add origin <URL-của-GitHub-repo>
```

```
# Đẩy code lên GitHub
git push -u origin tên-branch
# Ví dụ: git push -u origin nguyen-van-a
```

### Ví dụ thực tế - Clone repository và tạo branch:

```
# Bước 1: Clone repository từ GitHub
git clone https://github.com/Hau-ThucTap/taxi-booking-backend.git

# Bước 2: Di chuyển vào folder project
cd taxi-booking-backend

# Bước 3: Tạo branch mới với tên của bạn
git checkout -b nguyen-van-a

# Bước 4: Tạo thay đổi (ví dụ: sửa README.md)
# ... sửa file ...

# Bước 5: Thêm file vào staging
git add .

# Bước 6: Commit với thông điệp
git commit -m "Initial commit: Branch của Nguyễn Văn A"

# Bước 7: Đẩy branch lên GitHub
git push -u origin nguyen-van-a
```

### Ví dụ thực tế - Tạo project mới (nếu cần):

```
# Bước 1: Khởi tạo Git
git init

# Bước 2: Thêm tất cả file vào staging
git add .

# Bước 3: Commit với thông điệp
git commit -m "Khởi tạo dự án Taxi Booking Backend"

# Bước 4: Kết nối với GitHub (nếu đã tạo repo trên GitHub)
git remote add origin https://github.com/username/taxi-booking-backend.git

# Bước 5: Đẩy code lên
git push -u origin main
```

---

## 4.2. File .gitignore

**Giải thích:** File **.gitignore** giống như một danh sách "không được mang theo". Bạn liệt kê những file/folder không muốn Git theo dõi (ví dụ: file tạm, file build, thông tin nhạy cảm).

**Ví dụ file **.gitignore** cho Spring Boot:**

```
# Compiled class files
*.class

# Log files
*.log

# Maven
target/
.mvn/

# IDE
.idea/
.vscode/
*.iml

# OS
.DS_Store
Thumbs.db

# Environment variables
.env
application-local.properties
```

**Cách tạo:**

1. Trong VS Code, tạo file mới tên **.gitignore**
2. Copy nội dung trên vào
3. Lưu file

**Tại sao cần:** Nếu không có **.gitignore**, bạn có thể vô tình commit file chứa mật khẩu database lên GitHub (rất nguy hiểm!)

---

### 4.3. Cách tạo Pull Request (PR)

**Giải thích:** Pull Request giống như việc bạn viết một bản nháp, gửi cho người khác xem và góp ý trước khi đưa vào bản chính thức.

**Quy trình:**

1. Tạo branch mới (nhánh mới) để làm tính năng mới
2. Code và commit trên branch đó
3. Push branch lên GitHub
4. Tạo Pull Request trên GitHub
5. Người khác review (xem xét) code

6. Nếu OK, merge (gộp) vào branch chính

### Ví dụ trong dự án Taxi:

```
# Bước 1: Tạo branch mới cho tính năng "Đăng ký tài khoản"
git checkout -b feature/user-registration

# Bước 2: Code và commit
git add .
git commit -m "Thêm API đăng ký tài khoản"

# Bước 3: Push branch lên GitHub
git push -u origin feature/user-registration

# Bước 4: Vào GitHub, bạn sẽ thấy nút "Compare & pull request"
# Click vào và tạo PR
```

---

## 5. Git Branching

### 5.1. Branch là gì?

**Giải thích:** Branch (nhánh) giống như một bản sao của code. Bạn có thể thử nghiệm trên nhánh này mà không ảnh hưởng đến nhánh chính.

#### Ví dụ thực tế:

- Bạn đang viết một cuốn sách (branch **main**)
- Bạn muốn thử viết một chương mới nhưng chưa chắc có hay không
- Bạn tạo một bản sao (branch **new-chapter**)
- Bạn viết trên bản sao đó
- Nếu hay, bạn gộp vào cuốn sách chính
- Nếu không hay, bạn xóa bản sao đó

---

### 5.2. Main vs Develop

#### Giải thích:

- **main (hoặc master):** Branch chính, code ở đây phải luôn ổn định, sẵn sàng để deploy lên production
- **develop:** Branch phát triển, nơi tích hợp các tính năng mới trước khi đưa vào **main**

#### Quy trình làm việc:

```
feature/user-registration (tính năng mới) → develop (tích hợp) → main (ổn định)
```

**Giải thích flow:**

- Tính năng mới được phát triển trên branch **feature/\***
- Khi hoàn thành, merge vào **develop** để tích hợp và test
- Khi **develop** ổn định, merge vào **main** để deploy production

**Ví dụ trong dự án Taxi:**

```
# Tạo branch develop
git checkout -b develop

# Tạo branch cho tính năng mới từ develop
git checkout develop
git checkout -b feature/booking-api

# Code xong, merge vào develop
git checkout develop
git merge feature/booking-api

# Khi develop ổn định, merge vào main
git checkout main
git merge develop
```

---

**5.3. Cách tạo và merge branch****Tạo branch mới:**

```
# Cách 1: Tạo và chuyển sang branch mới
git checkout -b feature/tên-tính-năng

# Cách 2: Tạo branch mới (chưa chuyển)
git branch feature/tên-tính-năng
# Sau đó chuyển sang: git checkout feature/tên-tính-năng
```

**Xem danh sách branch:**

```
git branch
# Branch hiện tại sẽ có dấu *
```

**Chuyển sang branch khác:**

```
git checkout tên-branch
```

## Merge branch:

```
# Bước 1: Chuyển sang branch muốn merge vào (thường là main hoặc develop)
git checkout main

# Bước 2: Merge branch khác vào
git merge feature/tên-tính-năng

# Bước 3: Xóa branch đã merge (tùy chọn)
git branch -d feature/tên-tính-năng
```

## Ví dụ thực tế:

```
# Tạo branch cho tính năng "Tính giá cước"
git checkout -b feature/calculate-price

# Code xong, commit
git add .
git commit -m "Thêm API tính giá cước"

# Chuyển về main và merge
git checkout main
git merge feature/calculate-price

# Xóa branch đã merge
git branch -d feature/calculate-price
```

## Tại sao quan trọng cho teamwork và CI/CD:

- **Teamwork:** Mỗi người làm trên branch riêng, không xung đột code
- **CI/CD:** GitHub Actions có thể tự động test và deploy khi code được merge vào **main**

---

## Thực hành

### Bài tập 1: Setup môi trường development

**Mục tiêu:** Đảm bảo máy tính của bạn đã sẵn sàng để code Spring Boot

#### Yêu cầu:

1. Kiểm tra JDK 17+ đã cài chưa

```
java -version
```

- Nếu chưa có, cài JDK 17

## 2. Kiểm tra Git đã cài chưa

```
git --version
```

- Nếu chưa có, cài Git

## 3. Cài đặt VS Code

- Tải và cài VS Code từ <https://code.visualstudio.com/>

## 4. Cài đặt các Extensions trong VS Code:

- Extension Pack for Java
- Spring Boot Extension Pack
- Thunder Client (hoặc Postman)
- GitLens (tùy chọn)

### Kết quả mong đợi:

- Chạy được lệnh `java -version` và thấy version 17+
- Chạy được lệnh `git --version`
- VS Code đã cài với các extensions cần thiết

---

## Bài tập 2: Cài đặt và cấu hình VS Code với Java extensions

**Mục tiêu:** Làm quen với VS Code và đảm bảo Java extension hoạt động

### Yêu cầu:

1. Mở VS Code
2. Mở Command Palette (`Cmd+Shift+P` / `Ctrl+Shift+P`)
3. Gõ "Java: Clean Java Language Server Workspace" → Enter (để reset nếu cần)
4. Tạo một file Java test:
  - File → New File
  - Lưu với tên `Test.java`
  - Gõ code:

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("Hello from VS Code!");  
    }  
}
```

5. Kiểm tra xem VS Code có gợi ý code (IntelliSense) không
6. Chạy file Java (nếu có nút "Run" xuất hiện)

### Kết quả mong đợi:

- VS Code hiểu code Java (không có lỗi đỏ)
- Có gợi ý code khi gõ
- Chạy được file Java và thấy output "Hello from VS Code!"

---

## Bài tập 3: Clone Repository và Mở Project trong VS Code

**Mục tiêu:** Clone project có sẵn từ GitHub và mở trong VS Code

**Giải thích:** Thay vì tạo project mới, chúng ta sẽ clone (tải về) project đã được chuẩn bị sẵn từ GitHub. Điều này giúp tất cả học viên bắt đầu với cùng một codebase.

**Yêu cầu:**

### 1. Mở Terminal trong VS Code:

- Cách 1: **View** → **Terminal** (hoặc **Ctrl+ / Cmd+ `** )
- Cách 2: **Terminal** → **New Terminal**

### 2. Chọn thư mục để clone project:

- Ví dụ: Desktop, Documents, hoặc một folder riêng cho projects
- Trong Terminal, di chuyển đến thư mục đó:

```
cd ~/Documents # macOS/Linux
# hoặc
cd C:\Users\YourName\Documents # Windows
```

### 3. Clone repository:

```
git clone https://github.com/Hau-ThucTap/taxi-booking-backend.git
```

**Giải thích:** Lệnh **git clone** sẽ tải toàn bộ code từ GitHub về máy tính của bạn.

### 4. Di chuyển vào folder project:

```
cd taxi-booking-backend
```

### 5. Mở project trong VS Code:

- Cách 1: Trong Terminal, gõ:

```
code .
```

(Lệnh **code .** sẽ mở folder hiện tại trong VS Code)

- Cách 2: Mở VS Code → File → Open Folder → Chọn folder **taxi-booking-backend**

## 6. Đợi VS Code load project:

- VS Code sẽ tự động detect đây là Java project
- Maven dependencies sẽ được tự động download (xem progress bar dưới cùng)
- Đợi cho đến khi không còn thông báo "Loading..." hoặc "Indexing..."

### Kết quả mong đợi:

- Có folder **taxi-booking-backend** với cấu trúc đầy đủ
- VS Code đã mở project
- Maven dependencies đã được load (không còn lỗi đỏ trong code)
- Có thể thấy các file Java trong Explorer panel bên trái

---

## Bài tập 4: Tạo Branch với Tên Học Viên và Push lên GitHub

**Mục tiêu:** Tạo branch riêng cho mỗi học viên và push lên GitHub

**Giải thích:** Mỗi học viên sẽ làm việc trên branch riêng của mình. Điều này giúp:

- Mỗi người có không gian làm việc riêng, không ảnh hưởng đến code của người khác
- Giảng viên có thể xem code của từng học viên dễ dàng
- Tránh xung đột code khi nhiều người cùng làm việc

### Yêu cầu:

#### 1. Kiểm tra bạn đang ở branch nào:

```
git branch
```

- Bạn sẽ thấy dấu **\*** bên cạnh branch hiện tại (thường là **main**)

#### 2. Tạo branch mới với tên của bạn:

```
git checkout -b ten-ho-ten-cua-ban
```

### Ví dụ:

- Nếu tên bạn là "Nguyễn Văn A" → **git checkout -b nguyen-van-a**
- Nếu tên bạn là "Trần Thị B" → **git checkout -b tran-thi-b**
- Nếu tên bạn là "Lê Minh C" → **git checkout -b le-minh-c**

### Lưu ý:

- Tên branch nên viết thường, không dấu, dùng dấu gạch ngang **-** để ngăn cách
- Tránh dùng ký tự đặc biệt, khoảng trắng
- Nếu tên bạn có dấu, bỏ dấu đi (ví dụ: "Nguyễn" → "nguyen")

### 3. Kiểm tra đã chuyển sang branch mới chưa:

```
git branch
```

- Bạn sẽ thấy dấu \* bên cạnh branch mới tạo

### 4. Tạo một thay đổi nhỏ để test (tùy chọn):

- Mở file **README.md** (nếu có) hoặc tạo file mới
- Thêm một dòng: **# Học viên: [Tên của bạn]**
- Lưu file

### 5. Kiểm tra thay đổi:

```
git status
```

- Bạn sẽ thấy file vừa thay đổi trong danh sách "Changes"

### 6. Thêm file vào staging area:

```
git add .
```

(Lệnh này thêm tất cả file đã thay đổi)

### 7. Commit thay đổi:

```
git commit -m "Initial commit: Branch của [Tên bạn]"
```

Ví dụ: `git commit -m "Initial commit: Branch của Nguyễn Văn A"`

### 8. Push branch lên GitHub:

```
git push -u origin ten-ho-ten-cua-ban
```

(Thay **ten-ho-ten-cua-ban** bằng tên branch bạn vừa tạo)

**Ví dụ:**

```
git push -u origin nguyen-van-a
```

### 9. Nếu Git hỏi username/password:

- Bạn cần tạo Personal Access Token trên GitHub (xem hướng dẫn bên dưới)
- Khi Git hỏi password, paste token vào (không phải password GitHub của bạn)

### Cách tạo Personal Access Token (nếu cần):

1. Vào GitHub → Click avatar (góc trên bên phải) → **Settings**
2. Scroll xuống → **Developer settings** (ở cuối menu bên trái)
3. Click **Personal access tokens** → **Tokens (classic)**
4. Click **Generate new token (classic)**
5. Đặt tên token: **VS Code Git** (hoặc tên bất kỳ)
6. Chọn quyền: Tích vào **repo** (full control of private repositories)
7. Scroll xuống → Click **Generate token**
8. **QUAN TRỌNG:** Copy token ngay (chỉ hiện 1 lần, không xem lại được!)
9. Khi Git hỏi password, paste token này vào

### Hoặc dùng VS Code Source Control panel:

1. Click icon **Source Control** ở thanh bên trái (hoặc **Ctrl+Shift+G** / **Cmd+Shift+G**)
2. Bạn sẽ thấy danh sách file đã thay đổi
3. Click dấu **+** bên cạnh file để stage (hoặc click **+** bên cạnh "Changes" để stage tất cả)
4. Nhập commit message ở ô trên cùng: **Initial commit: Branch của [Tên bạn]**
5. Click icon **✓** (checkmark) để commit
6. Click icon **...** (3 chấm) → **Push** → Chọn branch của bạn
7. Nếu lần đầu, VS Code sẽ hỏi bạn có muốn publish branch không → Chọn **OK**

### Kết quả mong đợi:

- Đã tạo branch mới với tên của bạn
- Đã commit và push branch lên GitHub
- Vào GitHub repository: <https://github.com/Hau-ThucTap/taxi-booking-backend>
- Click vào dropdown "Branch" (góc trên bên trái) → Bạn sẽ thấy branch của mình trong danh sách
- Click vào branch của bạn → Xem code đã được push lên

---

## Dự án Taxi

### Clone Repository và Setup Branch Cá Nhân

**Mục tiêu:** Clone project Taxi Booking Backend từ GitHub và tạo branch riêng cho mỗi học viên

#### Yêu cầu:

1. **Clone repository** (theo Bài tập 3)

```
git clone https://github.com/Hau-ThucTap/taxi-booking-backend.git
cd taxi-booking-backend
code .
```

2. **Mở project trong VS Code**

- VS Code sẽ tự động mở sau lệnh `code .`
- Hoặc: File → Open Folder → Chọn folder `taxi-booking-backend`
- Đợi VS Code load Maven dependencies (xem progress bar dưới cùng)
- Đợi cho đến khi không còn thông báo "Loading..." hoặc "Indexing..."

### 3. Kiểm tra Java extension hoạt động:

- Mở file `TaxiBookingBackendApplication.java` (hoặc file main application tương tự)
- Xem có lỗi đỏ không (không nên có)
- Hover chuột vào `@SpringBootApplication` → Xem có tooltip giải thích không
- Nhấn `F12` (Go to Definition) → Xem có mở file định nghĩa không

### 4. Tạo branch với tên của bạn (theo Bài tập 4)

```
git checkout -b ten-ho-ten-cua-ban
```

#### Ví dụ:

- `git checkout -b nguyen-van-a`
- `git checkout -b tran-thi-b`
- `git checkout -b le-minh-c`

### 5. Kiểm tra cấu trúc project:

- Xem trong Explorer panel (bên trái VS Code)
- Project nên có cấu trúc:

```
taxi-booking-backend/  
├── src/  
│   ├── main/  
│   │   ├── java/  
│   │   │   └── com/taxi/booking/  
│   │   └── resources/  
│   └── test/  
├── pom.xml  
├── .gitignore  
└── README.md
```

### 6. Kiểm tra project có chạy được không:

- Tìm file có `@SpringBootApplication` (thường là `TaxiBookingBackendApplication.java`)
- Click nút "Run" bên cạnh method `main`
- Hoặc nhấn `F5` → Chọn "Java"
- Xem Terminal, nếu thấy:

```
Started TaxiBookingBackendApplication in X.XXX seconds
```

→ Thành công!

- Nhấn **Ctrl+C** để dừng ứng dụng

## 7. Tạo commit và push branch lên GitHub:

- Tạo một thay đổi nhỏ (ví dụ: thêm comment trong README.md)
- Commit và push (theo Bài tập 4):

```
git add .  
git commit -m "Initial commit: Branch của [Tên bạn]"  
git push -u origin ten-ho-ten-cua-ban
```

## 8. Xác nhận trên GitHub:

- Vào: <https://github.com/Hau-ThucTap/taxi-booking-backend>
- Click dropdown "Branch" (góc trên bên trái)
- Tìm và click vào branch của bạn
- Xác nhận code đã được push lên

## Kết quả mong đợi:

- ☒ Đã clone repository thành công
- ☒ Project mở trong VS Code và Maven dependencies đã load
- ☒ Đã tạo branch riêng với tên của bạn
- ☒ Project có thể chạy được (run thành công)
- ☒ Branch đã được push lên GitHub và có thể thấy trên GitHub
- ☒ VS Code Java extension hoạt động tốt (có IntelliSense, không có lỗi)

## Lưu ý quan trọng:

- **Luôn làm việc trên branch của bạn**, không push trực tiếp lên **main**
- Mỗi học viên có branch riêng, không ảnh hưởng đến code của người khác
- Ở buổi này, chúng ta chỉ setup môi trường. Code thực tế sẽ được viết ở các buổi sau
- Đảm bảo project có thể chạy được (run thành công) trước khi kết thúc buổi học

## Troubleshooting:

- **Nếu Git hỏi username/password:** Tạo Personal Access Token (xem hướng dẫn trong Bài tập 4)
- **Nếu Maven dependencies không load:** Đợi thêm vài phút, hoặc mở Terminal và chạy: **./mvnw clean install**
- **Nếu có lỗi Java version:** Kiểm tra **java -version** phải là 17+

---

## Tổng kết buổi 1

### Những gì đã học:

1. ☒ Hiểu kiến trúc Client-Server-Database

2. ☒ Cài đặt môi trường development (JDK, VS Code, Git)
3. ☒ Làm quen với VS Code và các extensions
4. ☒ Hiểu Git cơ bản (clone, branch, add, commit, push)
5. ☒ Hiểu Git Branching (main, develop, feature branches)
6. ☒ Clone Spring Boot project từ GitHub
7. ☒ Tạo branch riêng và push lên GitHub

### Chuẩn bị cho buổi 2:

- ☒ Đã clone project **taxi-booking-backend** từ GitHub
- ☒ Đã tạo branch riêng với tên của bạn
- ☒ Project đã được mở trong VS Code và chạy thành công
- ☒ VS Code đã cài đặt và hoạt động tốt
- ☒ Git và GitHub đã được cấu hình
- ☒ Branch của bạn đã được push lên GitHub

### Kiểm tra lại trước buổi 2:

- ☐ Project có thể chạy được (run thành công, không có lỗi)
- ☐ Branch của bạn đã có trên GitHub (vào <https://github.com/Hau-ThucTap/taxi-booking-backend> và kiểm tra)
- ☐ VS Code Java extension hoạt động (có IntelliSense, không có lỗi đỏ)
- ☐ Đã biết cách mở Terminal trong VS Code
- ☐ Đã biết cách commit và push code

### Bài tập về nhà (tùy chọn):

- Thử tạo một file mới trong project, commit và push lên branch của bạn
- Đọc thêm về Spring Boot tại <https://spring.io/projects/spring-boot>
- Xem lại các lệnh Git cơ bản: **git status**, **git log**, **git branch**