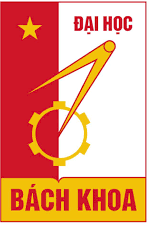
ĐẠI HỌC BÁCH KHOA HÀ NỘI



**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**

——————–o0o——————–

**BÁO CÁO THỰC TẬP KĨ THUẬT**

**LẬP TRÌNH CORE GIAO DỊCH CHỨNG KHOÁN VỚI DATABASE ORACLE, .NET**

**(CORE DATABASE)**

**VŨ VĂN THÀNH**

[thanh.vv216885@sis.hust.edu.vn](mailto:Chi.btl195952@sis.hust.edu.vn)

**Ngành Toán tin**

Người hướng dẫn: **Hoàng Duy Anh - Project Manager**

Bộ môn: **Toán tin**

Khoa: **Toán ứng dụng và Tin học**

Hà Nội, 05/2025

ĐẠI HỌC BÁCH KHOA HÀ NỘI

**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**

——————–o0o——————–

**BÁO CÁO THỰC TẬP KĨ THUẬT**

**LẬP TRÌNH CORE GIAO DỊCH CHỨNG KHOÁN VỚI DATABASE ORACLE, .NET**

**(CORE DATABASE)**

**VŨ VĂN THÀNH**

[thanh.vv216885@sis.hust.edu.vn](mailto:Chi.btl195952@sis.hust.edu.vn)

**Ngành Toán tin**

Người hướng dẫn: **Hoàng Duy Anh – Project Manager**

Bộ môn: **Toán tin**

Khoa: **Toán ứng dụng và Tin học**

Hà Nội, 05/2025

|  |  |
| --- | --- |
| **ĐẠI HỌC BÁCH KHOA HÀ NỘI**  **KHOA TOÁN – TIN** | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập - Tự do - Hạnh phúc** |
|  |  |

**PHIẾU ĐÁNH GIÁ KẾT QUẢ THỰC TẬP**

Sinh viên : Vũ Văn Thành

Mã số sinh viên : 20216885 Lớp : Toán tin 01 – K66

Số điện thoại : 0919434401 Email : thanh.vv216885@sis.hust.edu.vn

Địa điểm thực tập : Tầng 7, Số 315 Trường Chinh, Phường Khương Mai, Quận Thanh Xuân, TP. Hà Nội

Thời gian thực tập : 10/02/2025 – 26/05/2025 Hình thức : Full time

Cán bộ hướng dẫn tại cơ sở : Hoàng Duy Anh – Project Manager

1. **Nhận xét của cơ sở thực tập**
2. Nhận xét kết quả thực hiện đợt thực tập

* Về Chuyên môn, nghiệp vụ: …………………………………………………………………………………………………………………………………………………………………………
* Về Kỹ năng (Testing/Nghiên cứu/Viết báo cáo/Thuyết trình, ....):

……………………………………………………………………………………

……………………………………………………………………………………

* Về Ứng xử doanh nghiệp:

………………………………………………………………………………………………………………………………………………………………………………………

1. Ý thức của sinh viên

Sinh viên đã hoàn thành đợt thực tập với ý thức : Tốt / Khá / Kém

1. Kết quả đạt được

Điểm: ………………………

|  |  |
| --- | --- |
| **Xác nhận của cán bộ hướng dẫn** *(nếu có)* | **Xác nhận của cơ sở thực tập** |

**Lời cảm ơn**

Trong suốt quá trình thực tập kỹ thuật, em muốn gửi lời cảm ơn sâu sắc tới Công ty Giải pháp Phần mềm tài chính FSS, tập thể lãnh đạo, anh chị, các bạn đồng nghiệp và đặc biệt là các thầy cô viện Toán ứng dụng và Tin học, Đại học Bách khoa Hà Nội đã luôn hướng dẫn tận tình và tạo điều kiện giúp đỡ em hoàn thành mọi nhiện vụ, công việc được giao. Với điều kiện thời gian cũng như kiến thức của bản thân còn nhiều hạn chế dẫn đến báo cáo thực tập của em không tránh khỏi những thiếu sót. Vì vậy, em mong sẽ nhận được sự đóng góp từ các thầy cô để em có thêm nhiều bài học và kinh nghiệm áp dụng cho công việc sau này.

Em xin chân thành cảm ơn!

*Hà Nội, ngày 27 tháng 05 năm 2025*

Sinh viên

**Vũ Văn Thành – 20216885**

**LỜI MỞ ĐẦU**

Thực tập tốt nghiệp là một phần quan trọng trong chương trình đào tạo, giúp sinh viên áp dụng kiến thức đã học vào thực tế, đồng thời làm quen với môi trường làm việc chuyên nghiệp tại doanh nghiệp. Trong thời gian từ tháng 2 đến tháng 5 năm 2025, em đã có cơ hội thực tập tại Công ty Cổ phần Công nghệ Tài chính FSS (Financial Software Solutions) – một đơn vị chuyên cung cấp các giải pháp phần mềm trong lĩnh vực tài chính và chứng khoán.

Tại FSS, em được phân công vào nhóm phát triển và bảo trì hệ thống Core giao dịch chứng khoán Flex – một hệ thống phần mềm quan trọng đang được triển khai cho nhiều công ty chứng khoán lớn tại Việt Nam. Trong quá trình thực tập, em đã được tiếp xúc với các công cụ và công nghệ như .NET Framework, Oracle Database (PL/SQL), Visual Studio, Crystal Report, TortoiseGit, Docker Desktop, và IntelliJ IDEA. Ngoài ra, em còn được tìm hiểu quy trình làm việc thực tế tại công ty, bao gồm cách giao task, kiểm thử, review code, triển khai và quản lý mã nguồn.

Việc tham gia vào các đầu việc thực tế như viết báo cáo bằng Crystal Report, xử lý truy vấn với PL/SQL, sửa lỗi và bổ sung tính năng nhỏ trong hệ thống đã giúp em hiểu rõ hơn về cách một hệ thống phần mềm tài chính vận hành, cũng như cách phối hợp giữa các thành viên trong team để hoàn thành công việc đúng tiến độ.

Báo cáo này được thực hiện nhằm tổng hợp lại toàn bộ quá trình thực tập, từ môi trường làm việc, công việc đã thực hiện, đến những kiến thức và kỹ năng mà em đã tiếp thu. Qua đó, em rút ra được nhiều bài học bổ ích, làm nền tảng cho công việc sau này.

**Mục lục**

[Công ty Cổ phần Giải pháp phần mềm Tài chính FSS 6](#_Toc199275196)

[1.1. Giới thiệu chung 6](#_Toc199275197)

[1.2. Lịch sử hình thành 7](#_Toc199275198)

[1.3. Sản phẩm và dịch vụ 7](#_Toc199275199)

[Quá trình thực tập 9](#_Toc199275200)

[2.1. Tổng quan .NET và Oracle Database 9](#_Toc199275201)

[2.1.1. Lịch sử ra đời của .NET và Oracle Database 9](#_Toc199275202)

[2.1.2. Khái niệm, đặc điểm của .NET và Oracle Database 11](#_Toc199275203)

[2.2. Quy trình phát triển tại FSS 13](#_Toc199275204)

[2.2.1. Tổng quan các phương pháp và kĩ thuật phát triển phần mềm tại FSS 13](#_Toc199275205)

[2.2.2. Quy trình giao task 20](#_Toc199275206)

[2.2.3. Quy trình kiểm thử 21](#_Toc199275207)

[2.2.4. Quy trình review code 23](#_Toc199275208)

[2.2.5. Quy trình triển khai (Deployment) 24](#_Toc199275209)

[**2.3. Thực hành** 25](#_Toc199275210)

[2.3.1. Tìm hiểu hệ thống 25](#_Toc199275211)

[2.3.2. Công việc đã làm 27](#_Toc199275212)

[2.3.3. Công nghệ sử dụng 28](#_Toc199275213)

[**2.4. Môi trường làm việc tại FSS** 35](#_Toc199275214)

[**2.5. Đánh giá và đề xuất** 35](#_Toc199275215)

[Kinh nghiệm và học hỏi 37](#_Toc199275216)

[Kết luận 40](#_Toc199275217)

[Tài liệu tham khảo 41](#_Toc199275218)

**Danh sách hình ảnh**

[Hình 1: .Net Framework 10](#_Toc199273971)

[Hình 2 : Oracle Database 11](#_Toc199273972)

[Hình 3 : Giai đoạn phát triển phần mềm 12](#_Toc199273973)

[Hình 4 : Mô hình Agile 13](#_Toc199273974)

[Hình 5 : Vòng đời phát triển phần mềm 14](#_Toc199273975)

[Hình 6 : Docker Architecture 15](#_Toc199273976)

[Hình 7: Harbor - Quản lý image Docker 15](#_Toc199273977)

[Hình 8 : Jenkins Workflow 16](#_Toc199273978)

[Hình 9 : Jenskin - Tự động build image từ Harbor 16](#_Toc199273979)

[Hình 10: Gitlab - Quản lý mã nguồn 17](#_Toc199273980)

[Hình 11 : Postman - Kiểm thử API 18](#_Toc199273981)

[Hình 12 : Jira - Quản lý công việc 19](#_Toc199273982)

[Hình 13: Jira - Tạo mới Isssue 20](#_Toc199273983)

[Hình 14: Jira - WorkFlow Kiểm thử 22](#_Toc199273984)

[Hình 15: Quy trình CI/CD 24](#_Toc199273985)

[Hình 16 : CI/CD Workflow 24](#_Toc199273986)

[Hình 17 : Flex Example 26](#_Toc199273987)

[Hình 18 : PL/SQL - Lập trình database 27](#_Toc199273988)

[Hình 19 : Package giao dịch trong PL/SQL 28](#_Toc199273989)

[Hình 20 : Visual Studio - Lập trình C# 29](#_Toc199273990)

[Hình 21 : Crystal Report - Thiết kế báo cáo trong .NET 30](#_Toc199273991)

[Hình 22 : TortoiseGit - Git UI 31](#_Toc199273992)

[Hình 23 : Docker Desktop - Quản lý container 32](#_Toc199273993)

[Hình 24 : IntelliJ IDE - Lập trình Java 33](#_Toc199273994)

Hình 25 : SonarQube Architecture ……………………………………………….33

**Chương 1**

# Công ty Cổ phần Giải pháp phần mềm Tài chính FSS

## 1.1. Giới thiệu chung

Công ty cổ phần giải pháp phần mềm tài chính (FSS) được thành lập ngày 18/03/2008 bởi các chuyên gia đã có nhiều năm kinh nghiệm trong việc phát triển và triển khai giải pháp công nghệ thông tin cho các đơn vị hoạt động trong lĩnh vực tài chính, ngân hàng, chứng khoán tại Việt Nam và khu vực.  
Sau 16 năm hoạt động, với một đội ngũ hơn 400 chuyên gia làm việc trong lĩnh vực phần mềm, FSS đã trở thành một trong những công ty phần mềm tài chính – ngân hàng hàng đầu tại Việt Nam và được nhiều khách hàng tin tưởng, lựa chọn. Khách hàng của FSS bao gồm các ngân hàng và các công ty chứng khoán lớn như BIDV, TechcomBank, VPBank, MBBank, VNDirect, VCBS,BSC,BVSC,…  
Công ty cổ phần giải pháp phần mềm tài chính FSS mong muốn và không ngừng nỗ lực để trở thành công ty hàng đầu ở Việt Nam và có tên tuổi trong khu vực trong lĩnh vực cung cấp sản phẩm và dịch vụ phần mềm tài chính – ngân hàng; thông qua các sản phẩm, dịch vụ của mình mang lại sự thành công cho khách hàng, cổ đông, nhân viên công ty và đóng góp cho cộng đồn.g

## 1.2. Lịch sử hình thành

Công ty cổ phần Giải pháp phần mềm Tài chính

Tên tiếng anh: Financial Software Solutions

Địa chỉ: Tầng 7, Số 315 Trường Chinh, Phường Khương Mai, Quận Thanh Xuân, TP. Hà Nội

Tel: +1 804-314-5903

* 2008: FSS chính thức ra đời với mục tiêu cung cấp các giải pháp phần mềm chuyên biệt cho ngành tài chính – ngân hàng.
* Giai đoạn 2008–2024: FSS tập trung phát triển các sản phẩm phần mềm cốt lõi như hệ thống giao dịch chứng khoán, hệ thống quản lý quỹ, và các giải pháp kho dữ liệu (Data Warehouse) cùng phân tích kinh doanh (Business Intelligence). Công ty đã triển khai thành công nhiều dự án lớn cho các ngân hàng và công ty chứng khoán hàng đầu tại Việt Nam như BIDV, Techcombank, VPBank, MBBank, VNDirect, VCBS, BSC, BVSC…
* Tính đến năm 2024: FSS đã có hơn 400 chuyên gia trong lĩnh vực phần mềm, trở thành một trong những công ty phần mềm tài chính – ngân hàng hàng đầu tại Việt Nam.

## 1.3. Sản phẩm và dịch vụ

a. Giải pháp cho công ty chứng khoán và giao dịch tài chính

* FLEX: Hệ thống giao dịch chứng khoán.
* FDS: Hệ thống giao dịch chứng khoán phái sinh.
* HFT: Hệ thống giao dịch tần suất cao (High-Frequency Trading).
* VSD.STP: Hệ thống kết nối giao dịch với Trung tâm Lưu ký Chứng khoán Việt Nam.
* FundServe: Hệ thống quản lý dịch vụ đại lý chuyển nhượng và quản trị quỹ cho quỹ mở và quỹ hưu trí tự nguyện.
* FundTrack: Hệ thống quản lý và giao dịch chứng chỉ quỹ mở.

b. Giải pháp cho ngân hàng và tổ chức tài chính

* CBOND: Hệ thống giao dịch trái phiếu doanh nghiệp.
* iFTP: Hệ thống tính giá vốn nội bộ và phân bổ chi phí.
* ODS: Hệ thống dữ liệu vận hành (Operational Data Store) hỗ trợ phân tích và báo cáo.
* Data Warehouse & BI: Giải pháp kho dữ liệu và phân tích kinh doanh.

c. Hệ thống ngân hàng lưu ký và quản trị quỹ

* Giải pháp phần mềm hỗ trợ ngân hàng lưu ký trong việc giám sát và quản trị quỹ đầu tư.

d. Dịch vụ

* Triển khai phần mềm theo yêu cầu: Tư vấn, phát triển và triển khai phần mềm tùy chỉnh cho các tổ chức tài chính.
* Dịch vụ triển khai hệ thống ODS: Hỗ trợ xây dựng và triển khai hệ thống dữ liệu vận hành.
* Dịch vụ triển khai Data Warehouse & BI: Tư vấn và triển khai kho dữ liệu và giải pháp phân tích kinh doanh.
* Quản trị ứng dụng: Dịch vụ quản lý và vận hành các ứng dụng phần mềm.
* Hỗ trợ vận hành phần mềm ứng dụng: Dịch vụ hỗ trợ kỹ thuật và vận hành cho các ứng dụng phần mềm.
* Hỗ trợ vận hành hạ tầng hệ thống CNTT: Dịch vụ hỗ trợ kỹ thuật và vận hành cho hạ tầng công nghệ thông tin.
* Hỗ trợ sử dụng dịch vụ điện tử: Dịch vụ hỗ trợ người dùng trong việc sử dụng các dịch vụ điện tử.

**Chương 2**

# Quá trình thực tập

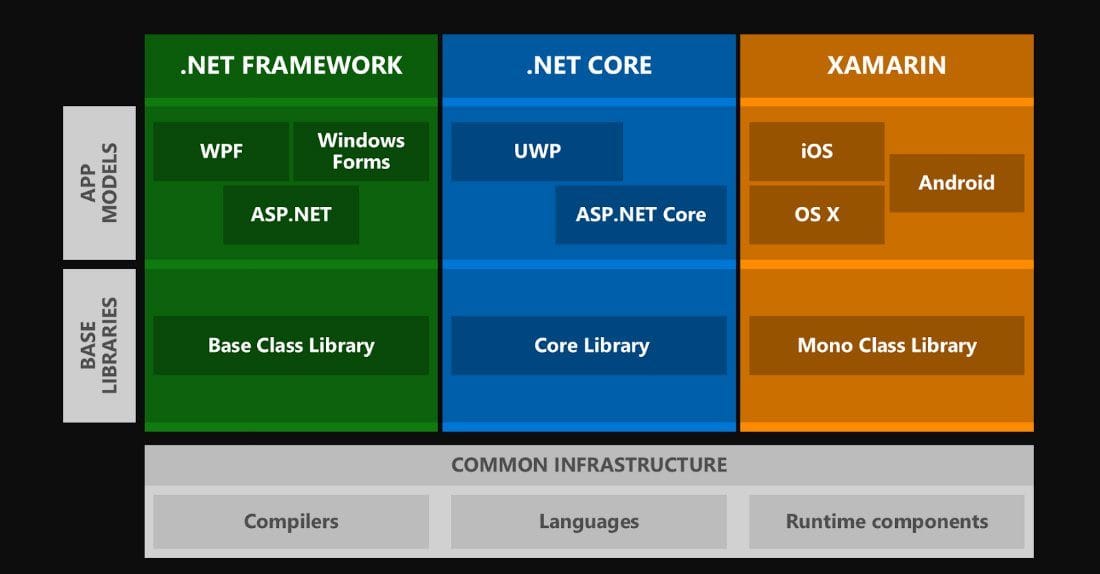
Được nhận vào FSS từ tháng 02/2025, em đã được giảng dạy và tiếp xúc với công nghệ .NET, Oracle Database, quy trình phát triển phần mềm, quy trình làm việc và các công cụ sử dụng trong quá trình làm việc. Dưới đây là những lý thuyết em đã học được và kết quả thực hành của mình.

## 2.1. Tổng quan .NET và Oracle Database

### 2.1.1. Lịch sử ra đời của .NET và Oracle Database

a. NET framework  
- Năm 2000, Microsoft giới thiệu .NET Framework như một nền tảng mới để phát triển phần mềm trên hệ điều hành Windows.  
- Mục tiêu: Đơn giản hóa phát triển phần mềm, hỗ trợ đa ngôn ngữ, cung cấp môi trường chạy an toàn và hiệu quả.  
- Giai đoạn phát triển :

* 2002 – Ra mắt .NET Framework 1.0 cùng với ngôn ngữ C#, nhanh chóng trở thành công cụ chính cho lập trình viên Windows.
* 2016 – Microsoft phát hành .NET Core (mã nguồn mở, đa nền tảng – Windows, Linux, macOS).
* 2020 – Microsoft hợp nhất tất cả các phiên bản lại thành .NET 5, đánh dấu bước chuyển mình lớn, định hướng phát triển thống nhất trên mọi nền tảng (Web, Desktop, Mobile, Cloud).
* 2021-nay – Các phiên bản mới như .NET 6, .NET 7, .NET 8 tiếp tục cải tiến về hiệu suất, bảo mật và hỗ trợ công nghệ hiện đại như microservices, cloud-native apps.



Hình 1: .Net Framework

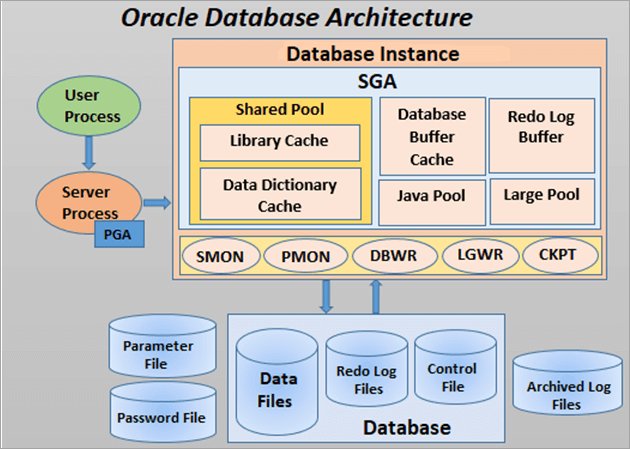
b. Oracle Database

- 1977, Larry Ellison, cùng với Bob Miner và Ed Oates thành lập Software Development Laboratories (SDL) – tiền thân của Oracle Corporation.

-Dựa trên lý thuyết "cơ sở dữ liệu quan hệ" của Edgar F. Codd (IBM), họ tạo ra phần mềm cơ sở dữ liệu quan hệ đầu tiên trên thế giới.

- Giai đoạn phát triển :

* 1979 – Ra mắt sản phẩm đầu tiên Oracle v2, trở thành hệ quản trị CSDL   
  thương mại đầu tiên sử dụng SQL.
* 1982 – Công ty đổi tên thành Oracle Corporation.
* 1990s–2000s – Oracle phát triển mạnh mẽ, dẫn đầu thị trường với Oracle 8i, 9i, 10g, và 11g – hỗ trợ Internet, điện toán lưới (grid computing).
* Hiện nay, Oracle là một trong những hệ quản trị cơ sở dữ liệu hàng đầu thế giới, phục vụ hàng triệu hệ thống doanh nghiệp toàn cầu, đồng thời mở rộng sang cloud services, AI, analytics...



Hình 2 : Oracle Database

### 2.1.2. Khái niệm, đặc điểm của .NET và Oracle Database

a. NET framework

- Khái niệm : .NET là một nền tảng phát triển phần mềm được Microsoft xây dựng, cho phép lập trình viên phát triển nhiều loại ứng dụng như web, desktop, mobile, cloud, IoT,... bằng nhiều ngôn ngữ như C#, VB.NET, F#.

- Đặc điểm nổi bật :

* Đa nền tảng : Hỗ trợ phát triển và chạy ứng dụng trên Windows, Linux, macOS.
* Hỗ trợ đa ngôn ngữ : Cho phép viết code bằng C#, F#, VB.NET,...
* Mã nguồn mở : .NET Core và .NET 5+ là open source, với sự đóng góp lớn từ cộng đồng.
* Hiệu năng cao : Cải tiến qua từng phiên bản, tối ưu hóa tốc độ thực thi và bộ nhớ.
* Hệ sinh thái phong phú : Có thư viện và framework đa dạng như ASP.NET, Entity Framework, Blazor,...
* Hỗ trợ cloud & container : Tích hợp tốt với Docker, Kubernetes và các nền tảng đám mây như Azure.
* Bảo mật : Cung cấp cơ chế mã hóa, xác thực (JWT, OAuth2), kiểm soát truy cập,...

- Ứng dụng của .NET framework :

* Ứng dụng Web: Sử dụng ASP.NET Core để xây dựng các ứng dụng web hiện đại, hiệu suất cao, hỗ trợ cả RESTful API và MVC.
* Ứng dụng Desktop: Với Windows Forms, WPF hoặc MAUI, .NET hỗ trợ phát triển phần mềm giao diện người dùng trên Windows hoặc đa nền tảng.
* Ứng dụng Mobile: Xamarin và .NET MAUI cho phép phát triển ứng dụng mobile đa nền tảng (Android, iOS) chỉ với một base code.
* Ứng dụng Cloud-native: Kết hợp với Docker, Kubernetes và Azure để xây dựng các ứng dụng microservices, serverless hoặc cloud-native.
* Trí tuệ nhân tạo (AI) và học máy (ML): Có thể tích hợp ML.NET hoặc gọi đến các dịch vụ AI của Azure để phân tích dữ liệu, xử lý ảnh, ngôn ngữ,...
* IoT và thiết bị nhúng: .NET hỗ trợ phát triển ứng dụng IoT chạy trên các thiết bị nhỏ như Raspberry Pi.
* Hệ thống doanh nghiệp: Thường dùng trong các phần mềm ERP, CRM, HRM, kế toán,... nhờ tính ổn định và bảo trì dễ dàng.

b. Oracle Database

- Khái niệm : Oracle Database là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) do Oracle Corporation phát triển, dùng để lưu trữ và xử lý khối lượng lớn dữ liệu một cách an toàn, hiệu quả và có khả năng mở rộng cao, thường được sử dụng nhiều trong các hệ thống doanh nghiệp lớn như ngân hang, tài chính, thương mại điện tử,..

- Đặc điểm nổi bật :

* Hiệu suất cao : Xử lý dữ liệu nhanh, tối ưu hóa truy vấn hiệu quả.
* Tính ổn định : Hỗ trợ clustering, replication, failover,…
* Bảo mật mạnh mẽ : Mã hóa dữ liệu, kiểm soát truy cập, audit, bảo vệ khỏi SQL Injection,..
* Hỗ trợ ngôn ngữ PL/SQL : Cho phép lập trình logic nghiệp vụ phức tạp ngay trong database.
* Khả năng mở rộng linh hoạt : Phù hợp với hệ thống từ vừa đến rất lớn.
* Tích hợp tốt : Kết nối dễ dàng với .NET, Java, Python, ERP, hệ thống BI,...
* Hỗ trợ công nghệ mới : Oracle 19c/21c hỗ trợ AI/ML, blockchain, JSON, in-memory DB,...

- Kiến trúc tổng quan của Oracle Database : Oracle DB có kiến trúc Client – Server, chia làm 2 phần chính:

* Oracle Instance:
  + Gồm các tiến trình nền (Background Processes) và vùng nhớ (SGA – System Global Area).
  + Chịu trách nhiệm quản lý truy vấn, truy cập dữ liệu, cache, redo, undo...
* Oracle Database:
  + Chứa dữ liệu thực tế: Tables, Indexes, Views, Stored Procedures,...
  + Được lưu trong các file vật lý: datafiles, redo log files, control files,...

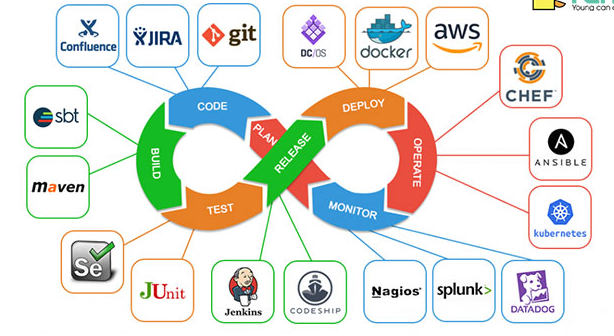
- Ứng dụng thực tế của Oracle Database

Oracle Database thường được sử dụng trong các **hệ thống lớn yêu cầu độ ổn định cao** và khả năng xử lý dữ liệu phức tạp như:

* Ngành tài chính – ngân hàng: Quản lý tài khoản, giao dịch, phân tích tín dụng, chống gian lận,...
* Thương mại điện tử: Lưu trữ thông tin sản phẩm, đơn hàng, thanh toán,...
* Doanh nghiệp lớn (ERP, CRM): Dùng làm nền tảng dữ liệu cho các hệ thống quản trị nguồn lực doanh nghiệp như Oracle E-Business Suite, SAP,...
* Logistics – chuỗi cung ứng: Lưu trữ và đồng bộ dữ liệu kho, vận chuyển, dự báo hàng tồn,...
* Y tế – giáo dục – chính phủ: Lưu trữ hồ sơ bệnh án, kết quả học tập, dữ liệu dân cư quốc gia,...

## 2.2. Quy trình phát triển tại FSS

### 2.2.1. Tổng quan các phương pháp và kĩ thuật phát triển phần mềm tại FSS



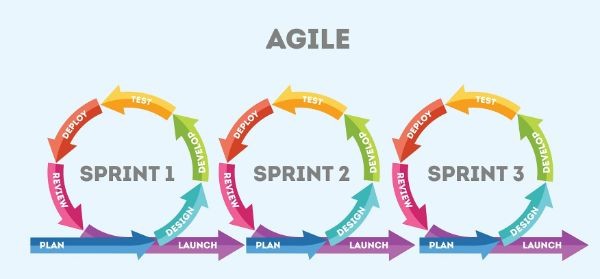
Hình 3 : Giai đoạn phát triển phần mềm

a. Phương pháp phát triển phần mềm Agile

FSS sử dụng quy trình phát triển phần mềm Agile :

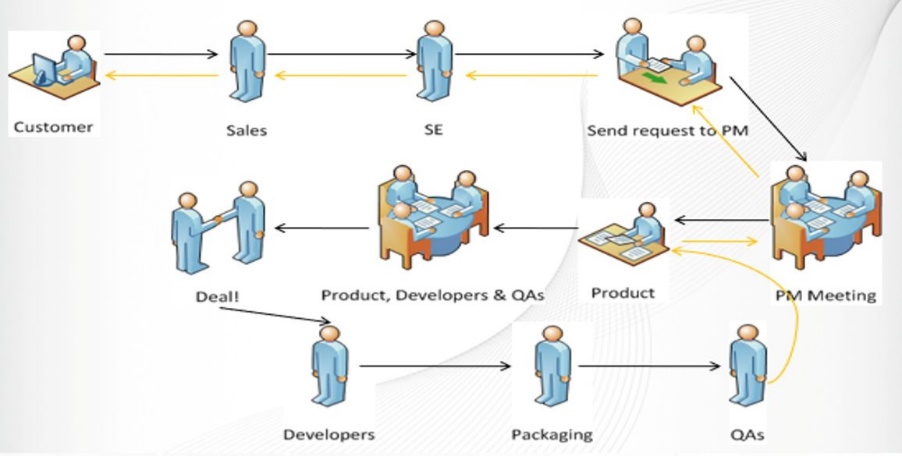
Quy trình bao gồm:

* + - Sprint Planning: Lập kế hoạch cho chu kỳ phát triển (thường 1–4 tuần); xác định backlog cần thực hiện.
    - Phân chia công việc: Các yêu cầu của khách hàng được phân chia thành các công việc nhỏ và cụ thể hơn, gọi là user story.
    - Scrum Daily Standup: Mỗi ngày, các thành viên trong nhóm có cuộc họp ngắn gọi là Daily Standup để báo cáo tiến độ và trao đổi về các khó khăn và hỗ trợ cần thiết.
    - Sprint Review: Cuối mỗi Sprint, nhóm tổ chức cuộc họp đánh giá kết quả và xác định công việc cho Sprint tiếp theo.
    - Kiểm thử và Tích hợp liên tục: Phần mềm được kiểm thử thường xuyên và tích hợp liên tục để đảm bảo chất lượng và tính ổn định của sản phẩm.
    - Demo sản phẩm: Cuối mỗi Sprint, sản phẩm được demo cho khách hàng và các bên liên quan để đánh giá và thu thập phản hồi.



Hình 4 : Mô hình Agile

Các bước xây dựng vòng đời yêu cầu trong phát triển phần mềm tại



Hình 5 : Vòng đời phát triển phần mềm

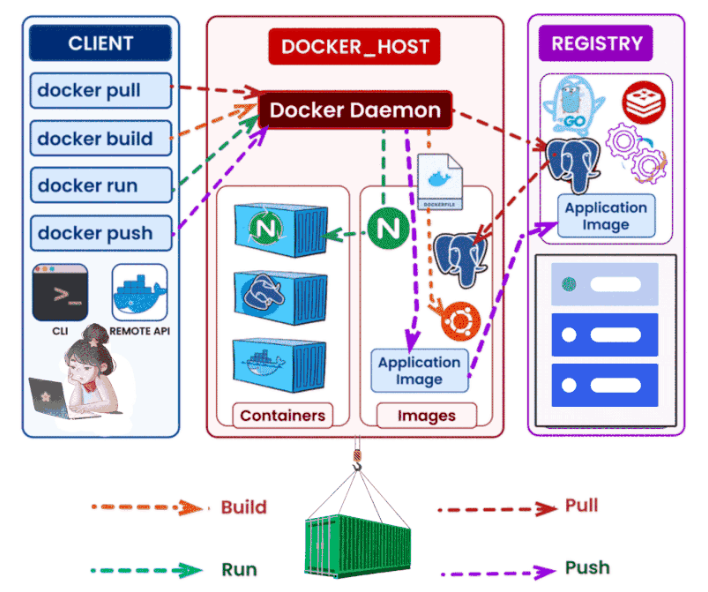
b. Công cụ và môi trường phát triển phần mềm

- IDE & Code Editor : Visual Studio, VS Code, IntelliJ IDEA là các công cụ lập trình chính cho C#, Java, JS,... trong FSS

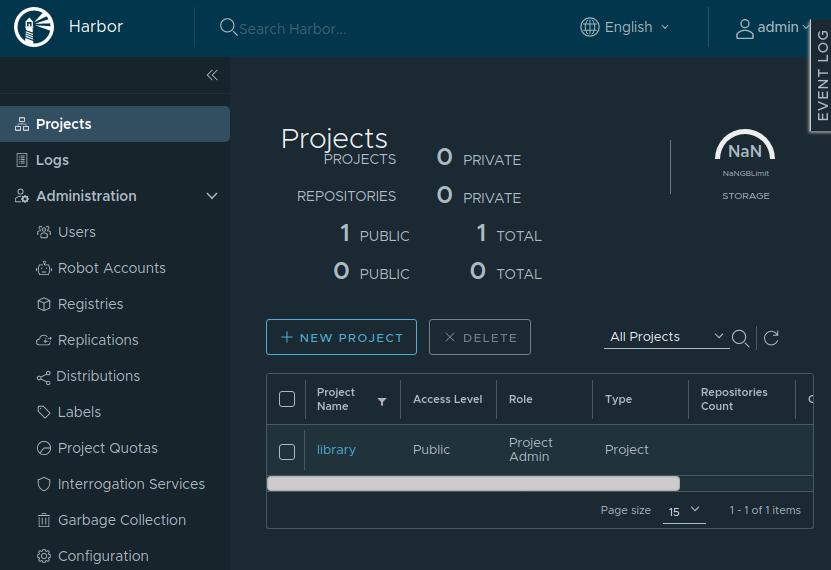
* **Visual Studio** : Là IDE (Integrated Development Environment) mạnh mẽ của Microsoft, chuyên dùng cho lập trình .NET (C#, VB.NET). Visual Studio hỗ trợ đầy đủ các tính năng như: viết mã, debug, quản lý project, tích hợp Git, test, và triển khai. Đây là công cụ chủ lực của các lập trình viên C# trong FSS.
* **Visual Studio Code (VS Code)** : Là trình soạn thảo mã nhẹ, đa nền tảng, rất phổ biến nhờ khả năng tùy biến cao thông qua các extension. VS Code thường được sử dụng cho các dự án front-end (HTML, CSS, JavaScript, TypeScript, Angular, React,...) hoặc back-end nhẹ (Node.js, Python,...). Ngoài ra, nó còn được dùng để viết tài liệu markdown, cấu hình YAML, Dockerfile, v.v.
* **IntelliJ IDEA** : Là IDE hàng đầu dành cho phát triển ứng dụng Java và các framework như Spring Boot. Với tính năng hỗ trợ thông minh như code completion, refactoring, phân tích mã nguồn và tích hợp với Maven/Gradle, IntelliJ là công cụ không thể thiếu cho các dự án Java của FSS.

- Devops : Đóng gói, lưu trữ image và triển khai dịch vụ trên các môi trường (dev/test/prod) : Docker, Harbor

* Docker là nền tảng container hóa giúp đóng gói và chạy ứng dụng cùng tất cả phụ thuộc trong một môi trường nhẹ, tách biệt (container), đảm bảo chạy nhất quán trên mọi môi trường.Docker gồm các thành phần :
  + Docker Engine: chạy và quản lý container.
  + Docker Image: mẫu (template) chứa app và môi trường.
  + Docker Container: instance chạy thực tế từ image.
  + Docker Hub: kho lưu trữ image dùng chung (ở đây là Harbor).
* Tính năng nổi bật :
  + Container nhẹ, khởi động nhanh vì không cần hệ điều hành.
  + Tách biệt môi trường, tránh xung đột vì mỗi container chạy độc lập, có hệ thống file , thư viện, cấu hình riêng biệt.
  + Dễ đóng gói, di chuyển và mở rộng.
  + Quản lý version qua Dockerfile.
  + Hỗ trợ tự động hóa và tích hợp CI/CD.



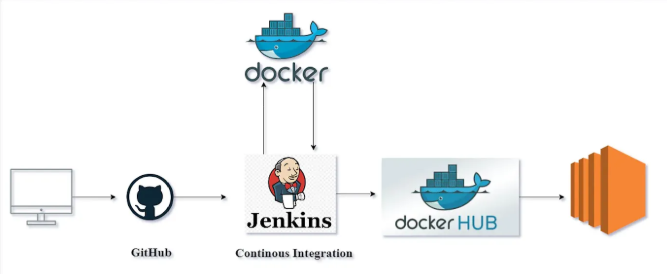
Hình 6 : Docker Architecture



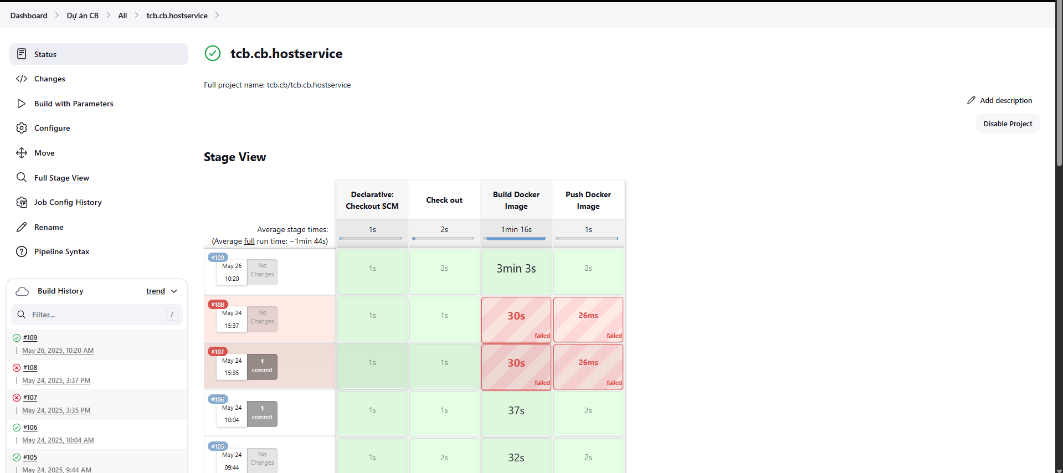
Hình 7: Harbor - Quản lý image Docker

- CI/CD (Jenkins, GitLab CI) : Tự động hóa build, kiểm thử và triển khai liên tục, hỗ trợ quy trình devops.

* CI – Continuous Integration (Tích hợp liên tục):
  + Mỗi khi lập trình viên đẩy code (push) lên Git (thường là GitHub, GitLab...):
    - Hệ thống tự động build code.
    - Tự động chạy test (unit test, integration test...).
* CD – Continuous Delivery / Deployment (Triển khai liên tục):
  + Continuous Delivery:
    - Sau bước CI, hệ thống sẵn sàng triển khai ứng dụng lên môi trường staging/production khi có yêu cầu (manual trigger).
  + Continuous Deployment:
    - Sau khi test xong, hệ thống sẽ tự động triển khai ứng dụng luôn mà không cần chờ người bấm nút.



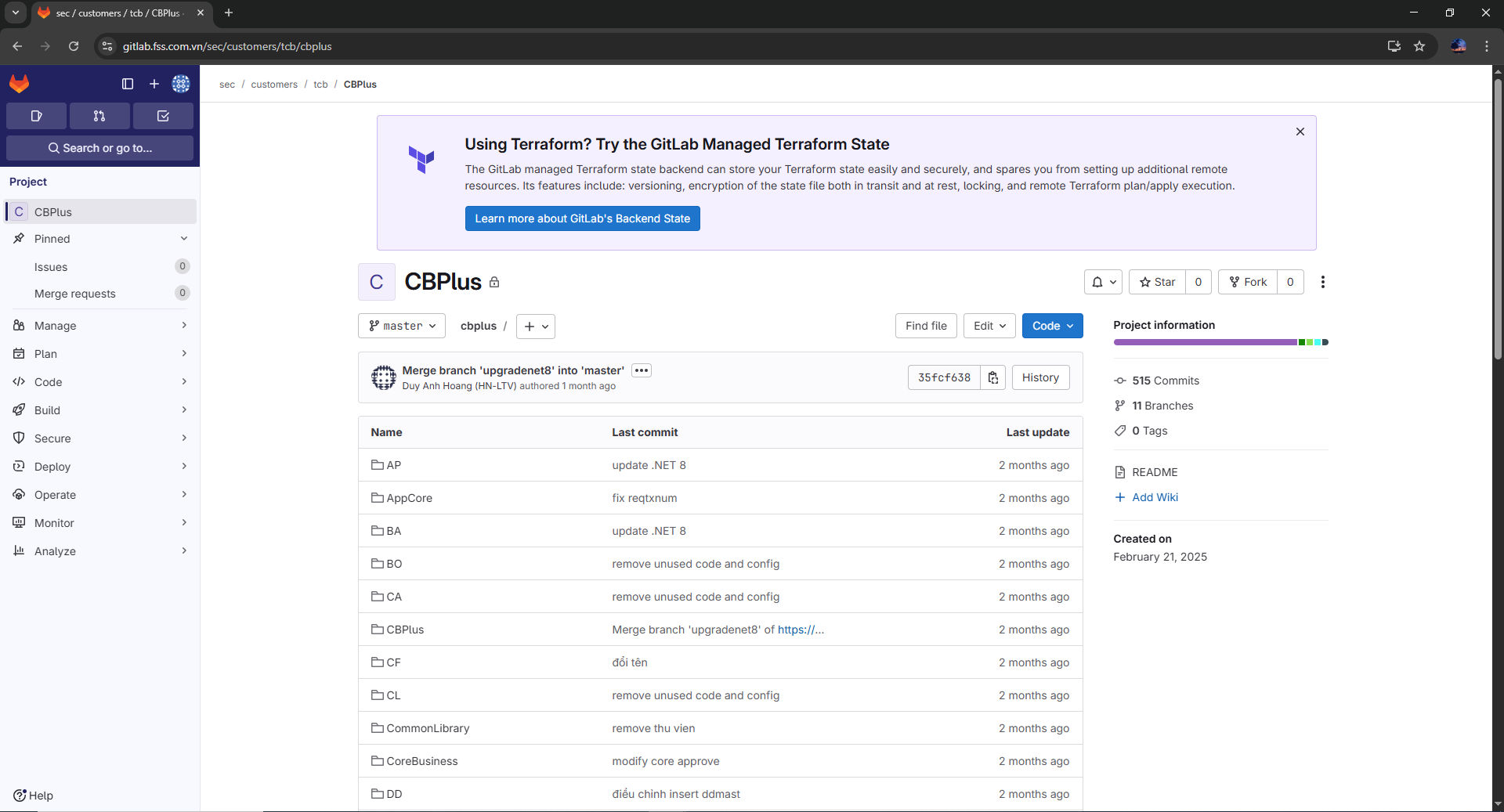
Hình 8 : Jenkins Workflow



Hình 9 : Jenskin - Tự động build image từ Harbor

- Quản lý mã nguồn : Git, GitHub, GitLab

* GitLab là một nền tảng DevOps mã nguồn mở cung cấp đầy đủ các công cụ để quản lý vòng đời phát triển phần mềm – từ lập kế hoạch, viết mã, kiểm thử, CI/CD, đến triển khai và giám sát. Nó được xây dựng dựa trên Git, tương tự như GitHub và Bitbucket, nhưng nổi bật ở khả năng self-hosted và tích hợp sâu các công cụ CI/CD.
* Các thành phần chính của GitLab :
  + Version Control System : GitLab sử dụng Git để quản lý mã nguồn, có thể tạo nhánh, commit, merge, pull request (trong GitLab gọi là Merge Request).
  + CI/CD (Continuous Integration / Continuous Deployment):  
    GitLab có tích hợp sẵn GitLab CI/CD – một hệ thống build/test/deploy tự động. Không cần cài thêm Jenkins hay Travis CI.
  + Issue Tracking và Project Management:  
    Tích hợp hệ thống quản lý công việc, giống như Trello/Jira (Board, Issue, Milestone...).
  + Merge Request + Code Review: Hỗ trợ xem diff, comment code, assign reviewer,...
  + Container Registry và Kubernetes Integration: Có thể deploy trực tiếp lên Kubernetes hoặc lưu image Docker trong GitLab Container Registry.
  + DevSecOps : GitLab hỗ trợ kiểm tra bảo mật (SAST, DAST, Dependency Scanning...) ngay trong pipeline.



Hình 10: Gitlab - Quản lý mã nguồn

- Kiểm thử : Postman, xUnit : Tích hợp test automation trong pipeline build.

* Postman là công cụ mạnh mẽ để kiểm thử các API. Bạn có thể tạo các bộ sưu tập (collection) chứa nhiều request API cùng với các kịch bản kiểm tra (tests) để đảm bảo API hoạt động đúng như kỳ vọng.
* Tích hợp vào pipeline build:
* Sử dụng Newman, giao diện dòng lệnh của Postman, để chạy các collection test tự động.
* Trong pipeline có thể gọi lệnh Newman sau khi build xong ứng dụng, đảm bảo các API đều hoạt động đúng.

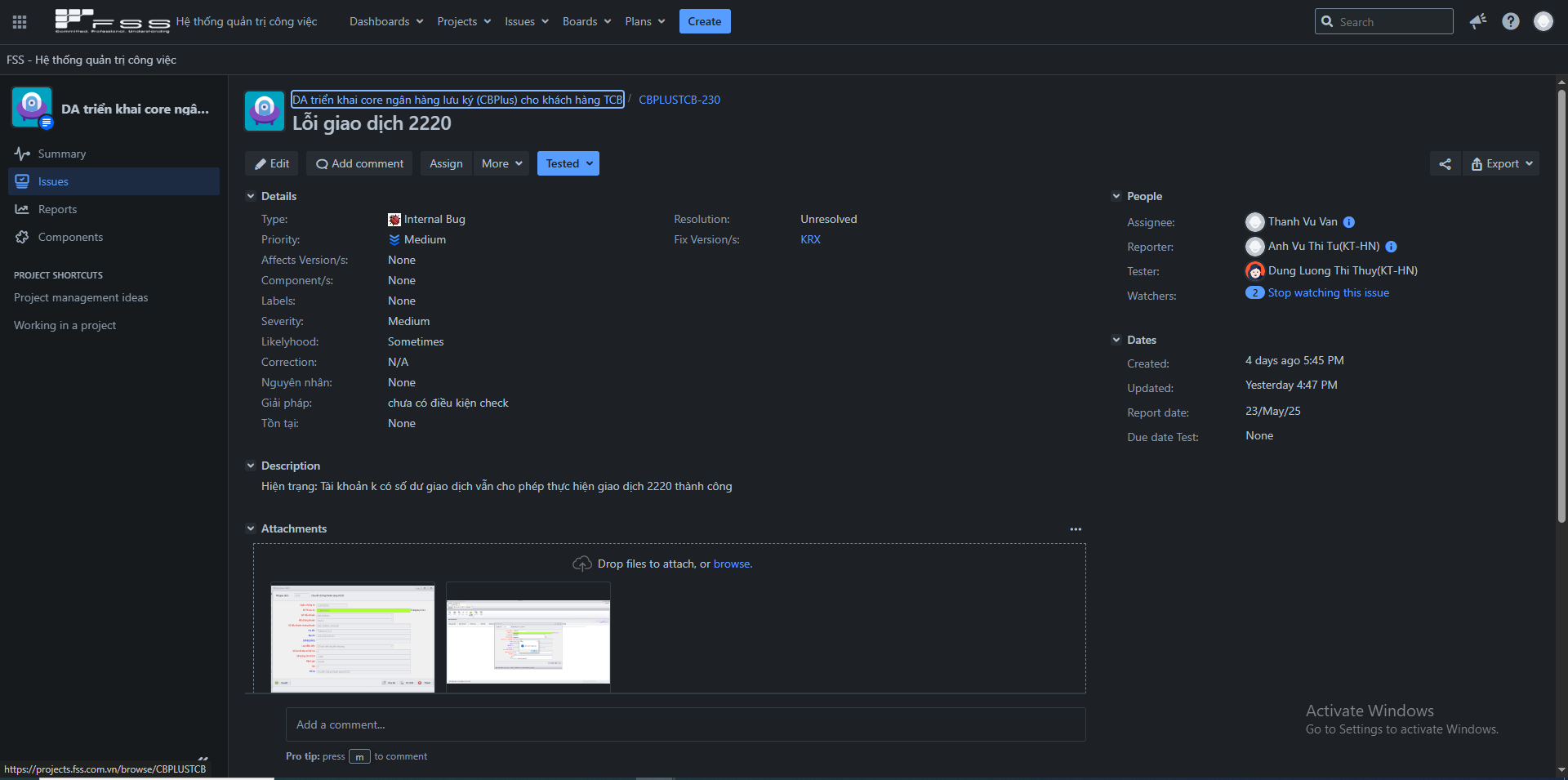
A screenshot of a computer

AI-generated content may be incorrect.

Hình 11 : Postman - Kiểm thử API

- Issue Tracking (Jira) : Quản lý tiến độ, backlog, bug và công việc nhóm phát triển.

* Jira là công cụ quản lý công việc, theo dõi tiến độ phát triển phần mềm, thường được dùng trong mô hình Agile/Scrum, các chức năng chính của Jira bao gồm :
  + Issue Tracking : Theo dõi từng task, bug, story, hoặc việc cần làm của team dev.
  + Backlog Management : Lưu trữ các yêu cầu chưa thực hiện, hỗ trợ lên kế hoạch sprint.
  + Sprint Planning : Hỗ trợ lập kế hoạch cho các đợt phát triển (sprint) và phân công task.
  + Bug Tracking : Ghi nhận, phân loại và theo dõi lỗi phần mềm, từ khi phát hiện đến khi fix xong.
  + Workflow tùy chỉnh : Thiết lập các bước xử lý riêng cho từng loại công việc: To Do → In Progress → Done.
  + Báo cáo & biểu đồ : Sinh ra các biểu đồ như Burndown, Velocity Chart để theo dõi tiến độ và năng suất nhóm.

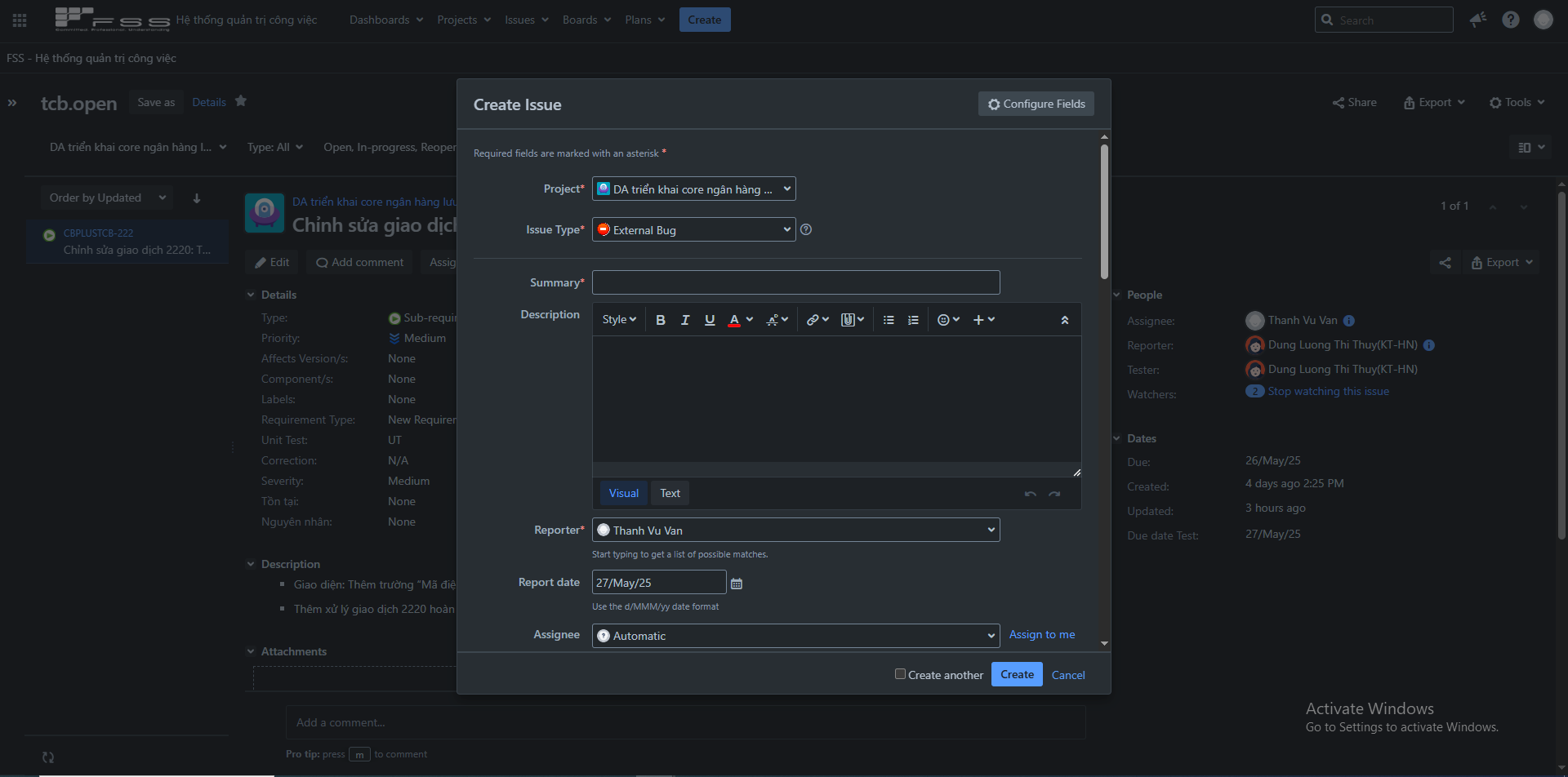


Hình 12 : Jira - Quản lý công việc

2.2.2. Quy trình giao task

Trong môi trường phát triển phần mềm theo mô hình Agile, việc phân chia và quản lý công việc theo từng Sprint là yếu tố then chốt để đảm bảo tiến độ và chất lượng dự án. Tại AvePoint, quy trình giao task được thực hiện chặt chẽ và có tổ chức như sau:

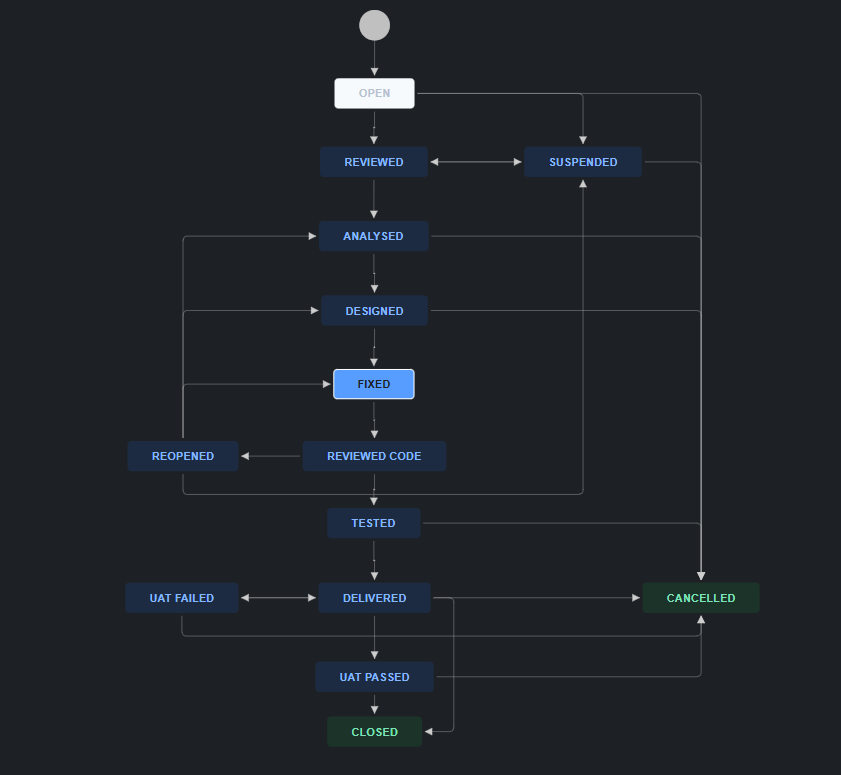
* Planning Meeting (Họp lập kế hoạch Sprint):  
  Trước khi bắt đầu mỗi Sprint (thường kéo dài 1-2 tuần), team sẽ tổ chức một buổi họp lên kế hoạch với sự tham gia đầy đủ của các thành viên bao gồm Developer, Tester, Scrum Master, và Product Owner. Trong buổi họp này, các tính năng (user stories) sẽ được phân tích, đánh giá độ phức tạp (thường dùng story points), xác định phạm vi thực hiện và thống nhất mức độ ưu tiên.
* Tạo task trên hệ thống quản lý (Jira):  
  Sau buổi họp, các task sẽ được tạo tương ứng với các user story đã thống nhất. Mỗi task bao gồm:
  + Tiêu đề rõ ràng, mô tả chi tiết: giải thích mục tiêu công việc, kết quả mong đợi.
  + Tiêu chí hoàn thành (Acceptance Criteria): giúp đảm bảo task hoàn tất đúng yêu cầu.
  + Ưu tiên (Priority): để sắp xếp mức độ cần thiết của công việc.
  + Thời hạn (Deadline) và thời lượng ước tính (Estimated Time).
  + Người phụ trách (Assignee): được phân công dựa theo kỹ năng và khối lượng công việc hiện tại.
* Phân công công việc : Các thành viên trong team sẽ thảo luận và nhận task phù hợp với năng lực và kinh nghiệm của mình. Trong trường hợp có task vượt quá khả năng hoặc mới mẻ, các lập trình viên sẽ được leader hoặc các thành viên có kinh nghiệm hơn hướng dẫn trực tiếp hoặc pair programming.
* Loại task được phân chia:
  + Task chính (feature/bug/task): phát triển tính năng mới, sửa lỗi hoặc cải tiến hệ thống.
  + Task hỗ trợ: bao gồm refactor code, tối ưu hiệu suất, viết tài liệu kỹ thuật, cập nhật hướng dẫn sử dụng, viết unit test, hỗ trợ kiểm thử…
  + Spike: các task nghiên cứu kỹ thuật hoặc công nghệ để chuẩn bị cho các task phát triển phức tạp hơn.
* Theo dõi tiến độ:  
  Mỗi task đều được cập nhật trạng thái theo tiến độ thực hiện : *To Do → In Progress → In Review → Done*. Các thành viên cập nhật liên tục để Scrum Master và cả team nắm được tình hình và điều phối hợp lý.
* Review & retrospective : Sau mỗi Sprint, nhóm sẽ họp Sprint Review để trình bày kết quả thực hiện, và Sprint Retrospective để rút kinh nghiệm, chia sẻ khó khăn, từ đó cải tiến quy trình cho các Sprint sau.



Hình 13: Jira - Tạo mới Isssue

2.2.3. Quy trình kiểm thử

* **Kiểm thử bởi lập trình viên (Developer Self-testing):**Sau khi hoàn thiện chức năng, lập trình viên chịu trách nhiệm thực hiện kiểm thử ban đầu để đảm bảo tính đúng đắn của mã nguồn:
* Unit Test (Kiểm thử đơn vị): được viết nhằm kiểm tra logic từng hàm, từng module độc lập.
* Test thủ công: lập trình viên sẽ chạy ứng dụng và kiểm thử trực tiếp các luồng nghiệp vụ cơ bản, dữ liệu đầu vào và các trường hợp ngoại lệ để phát hiện lỗi sớm.
* **Chuyển sang bộ phận QA/Tester:** Khi lập trình viên xác nhận chức năng hoạt động ổn định ở mức cơ bản, chức năng sẽ được đẩy sang môi trường kiểm thử (Staging hoặc Dev Test) để QA thực hiện kiểm thử chuyên sâu hơn:
* Kiểm thử chức năng (Functional Testing): xác minh tính năng theo đúng yêu cầu.
* Kiểm thử giao diện (UI Testing): kiểm tra bố cục, tương tác và trải nghiệm người dùng.
* Kiểm thử biên (Boundary Testing): kiểm tra các trường hợp đầu vào đặc biệt.
* Kiểm thử hồi quy (Regression Testing): đảm bảo các thay đổi không ảnh hưởng đến chức năng cũ.
* **Ghi nhận và xử lý lỗi:** Nếu phát hiện lỗi, QA sẽ ghi nhận lỗi chi tiết trên hệ thống quản lý như **Jira**, bao gồm:
* Mô tả lỗi (Steps to Reproduce).
* Mức độ ưu tiên (Priority).
* Môi trường xảy ra lỗi (Environment).
* Ảnh chụp màn hình hoặc video nếu cần.
* Sau đó, lỗi sẽ được gán lại cho lập trình viên để phân tích và khắc phục. Quy trình này có thể lặp lại cho đến khi lỗi được xử lý hoàn toàn.
* **Áp dụng kiểm thử tự động:**  
  Với những module quan trọng hoặc có tính chất lặp lại cao, nhóm có thể viết:
* Test tự động bằng Cypress, Selenium, v.v. để kiểm tra UI/Workflow.
* Integration test: đảm bảo tính toàn vẹn dữ liệu và luồng giữa các hệ thống liên quan.
* Performance test: nếu cần kiểm tra khả năng chịu tải.
* **Theo dõi và quản lý quy trình kiểm thử:**Toàn bộ tiến trình test, báo cáo lỗi và theo dõi tiến độ xử lý được thực hiện thông qua hệ thống quản lý như **Jira**, giúp đảm bảo tính minh bạch, đồng bộ và dễ truy vết khi cần thiết.
* **Kiểm thử chấp nhận (UAT - User Acceptance Testing):** Trong một số trường hợp, sau khi QA hoàn tất, sản phẩm sẽ được đội ngũ BA hoặc khách hàng nội bộ kiểm thử theo kịch bản thực tế để xác nhận đã đáp ứng đúng yêu cầu người dùng.



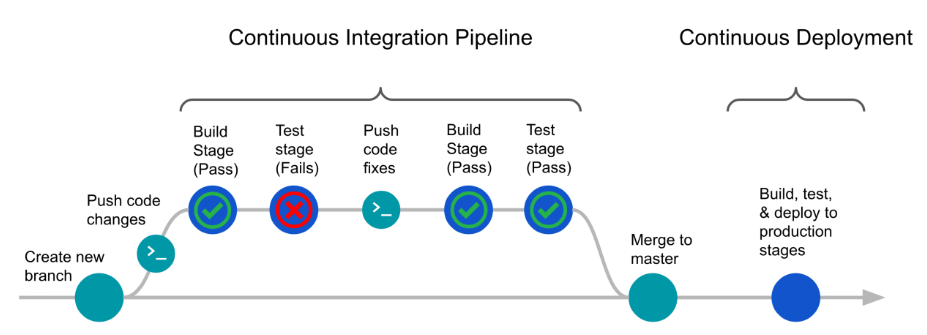
Hình 14: Jira - WorkFlow Kiểm thử

2.2.4. Quy trình review code

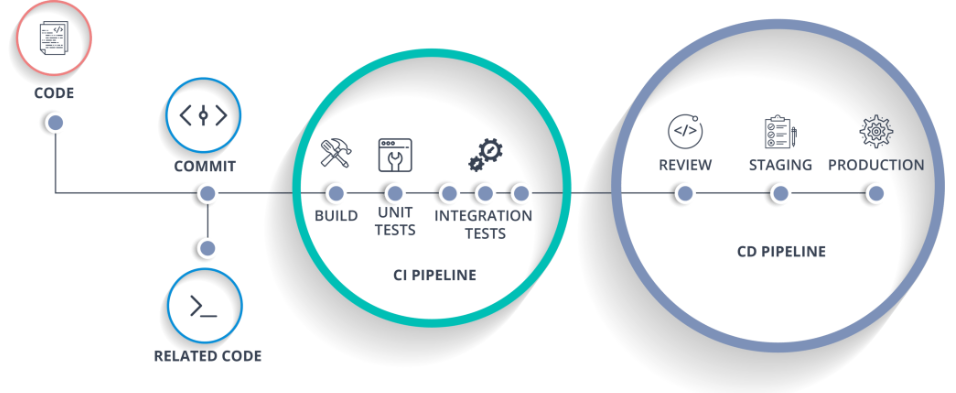
* **Tạo Pull Request (PR):** Sau khi hoàn thành một task hoặc tính năng, lập trình viên sẽ tạo pull request (PR) trên hệ thống quản lý mã nguồn như GitLab, GitHub, hoặc Bitbucket. PR sẽ được gắn mô tả chi tiết bao gồm:
* Mục đích của PR.
* Những thay đổi chính đã thực hiện.
* Task ID liên quan (nếu có).
* Hướng dẫn kiểm thử (nếu cần).
* **Review bởi thành viên có kinh nghiệm:** PR sẽ được assign cho **senior developer** hoặc **team leader** để thực hiện quá trình review. Người review sẽ đánh giá kỹ các khía cạnh sau:
* Quy chuẩn mã nguồn (Code Convention): Đảm bảo mã tuân thủ theo quy định coding style guide chung của dự án (naming, indentation, comments,...).
* Tối ưu hiệu suất: Kiểm tra các đoạn mã có khả năng gây ra hiệu năng kém, vòng lặp không cần thiết, truy vấn nặng,…
* Tính rõ ràng và dễ bảo trì: Mã phải dễ đọc, có phân tách rõ ràng các chức năng, sử dụng cấu trúc hợp lý, tránh hard-code.
* Bảo mật (Security): Kiểm tra các lỗ hổng phổ biến như SQL Injection, lộ thông tin nhạy cảm, xử lý dữ liệu không kiểm soát,…
* Ảnh hưởng tới hệ thống chung: Xem xét liệu PR có ảnh hưởng đến các module khác hay phá vỡ logic hệ thống không.
* **Comment & chỉnh sửa:** Nếu phát hiện điểm chưa hợp lý, người review sẽ comment trực tiếp vào dòng code trong PR để người tạo PR hiểu rõ vấn đề và sửa lại. Một PR có thể trải qua nhiều vòng chỉnh sửa trước khi được duyệt.
* **Phê duyệt và merge:** Khi mã đã đạt yêu cầu và được tối thiểu một reviewer approve, PR sẽ được merge vào nhánh chính (nhánh develop hoặc main/master) theo quy trình Git-flow.
* Một số dự án còn áp dụng rule: ít nhất 2 approve mới được merge.
* Trước khi merge, hệ thống CI sẽ tự động build và chạy test để đảm bảo không có lỗi phát sinh.
* **Kiểm soát quyền merge:** Chỉ những thành viên có quyền tương ứng (thường là leader hoặc người có vai trò maintainer) mới có thể thực hiện thao tác merge nhằm tránh rủi ro.
* **Ghi nhận lịch sử:** Tất cả các PR và hoạt động review đều được lưu lại trên hệ thống Git, giúp dễ dàng truy vết lại quá trình phát triển cũng như quyết định thiết kế trong tương lai.

2.2.5. Quy trình triển khai (Deployment)

* **Tự động kích hoạt sau khi merge:**  
  Sau khi một pull request (PR) được review, approved và merge vào nhánh chính (thường là develop, release hoặc main), hệ thống CI/CD sẽ tự động khởi động pipeline tương ứng.
* **Các bước trong pipeline CI/CD:**
  + Build ứng dụng: Mã nguồn sẽ được lấy từ Git và được tự động build để tạo ra các artifact (file thực thi, file deploy, container image,…). Bước này đảm bảo mã có thể biên dịch và hoạt động đúng.
  + Chạy kiểm thử tự động (nếu có): Các unit test, integration test hoặc API test sẽ được thực thi để đảm bảo không có lỗi logic hoặc lỗi nghiêm trọng lọt qua. Nếu bước này thất bại, pipeline sẽ dừng lại và báo lỗi cho developer.
  + **Triển khai lên môi trường staging:**  
    Sau khi build và test thành công, hệ thống sẽ tự động deploy lên môi trường staging/test, nơi nhóm QA sẽ tiến hành kiểm thử hệ thống lần cuối trong môi trường gần giống production.
* **Phê duyệt triển khai production:**  
  Sau khi QA xác nhận hệ thống hoạt động ổn định và không có lỗi nghiêm trọng:
  + Leader hoặc DevOps sẽ thực hiện manual approve để triển khai bản build lên môi trường production.
  + Một số dự án có thể áp dụng auto-deploy nếu pipeline vượt qua toàn bộ kiểm thử và không yêu cầu phê duyệt thủ công.



Hình 15: Quy trình CI/CD



Hình 16 : CI/CD Workflow

**2.3. Thực hành**

2.3.1. Tìm hiểu hệ thống

Trong những ngày đầu thực tập, em đã được anh/chị hướng dẫn tìm hiểu hệ thống phần mềm hiện tại mà công ty đang phát triển. Em đã nghiên cứu cấu trúc source code, cách tổ chức các module và công nghệ được sử dụng như NET Framework, Oracle Database,…

a. Tổng quan hệ thống Core giao dịch Chứng khoán Flex :

* **Mục tiêu và vai trò**

Hệ thống FLEX đóng vai trò như trung tâm xử lý nghiệp vụ giao dịch chứng khoán (core trading system), chịu trách nhiệm:

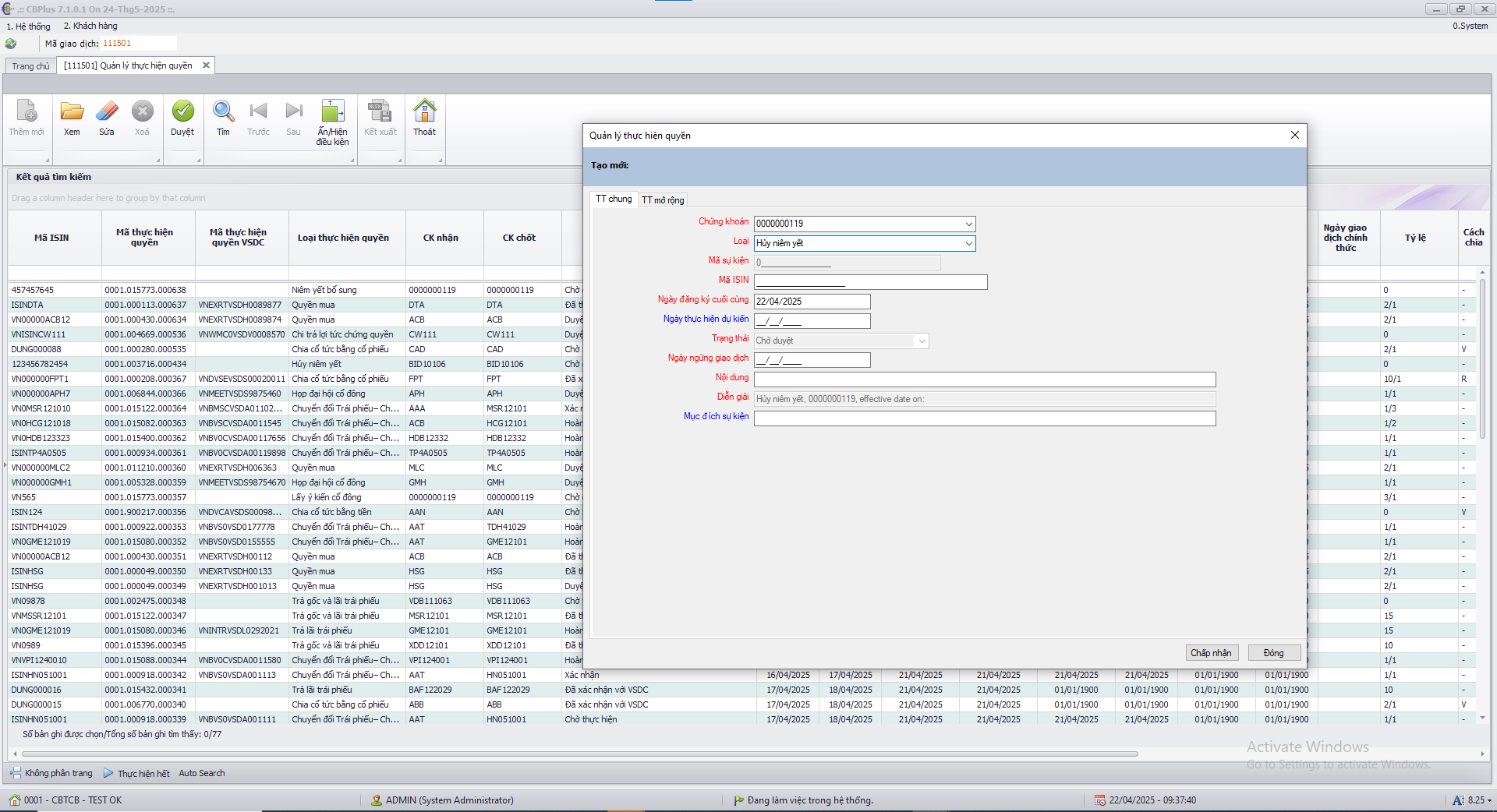
* Nhận và xử lý lệnh giao dịch từ các kênh frontend (web, mobile, broker,…)
* Kiểm tra điều kiện lệnh (sức mua, hạn mức, trạng thái tài khoản,…)
* Gửi lệnh lên sàn giao dịch HOSE, HNX, UPCOM qua gateway
* Nhận kết quả khớp lệnh, cập nhật danh mục đầu tư và trạng thái tài khoản
* Hỗ trợ thanh toán, lưu ký, ứng trước tiền bán, chuyển khoản chứng khoán,...
* **Các phân hệ chính**

Hệ thống FLEX bao gồm nhiều phân hệ nghiệp vụ:

* OD (Order Management): Quản lý đặt lệnh, khớp lệnh, trạng thái lệnh
* BL (Backlog Management): Xử lý các nghiệp vụ sau giao dịch (giao hàng, thanh toán)
* CA (Corporate Action): Quản lý quyền mua, chia cổ tức, phát hành thêm
* CC (Cash & Collateral): Quản lý tiền mặt, tài sản ký quỹ
* Margin: Quản lý giao dịch ký quỹ (cho vay, lãi suất, hạn mức,…)
* Risk Management: Quản lý rủi ro, sức mua, giới hạn lệnh
* Interface: Giao tiếp với sàn giao dịch (HOSE, HNX, VSD, VSDT,…)
* Report: Sinh báo cáo nội bộ, báo cáo theo yêu cầu UBCK/NHNN
* **Kiến trúc hệ thống**
* FLEX được xây dựng theo mô hình client-server kết hợp với kiến trúc modular (module hóa nghiệp vụ).
* Sử dụng Cơ sở dữ liệu Oracle với các thủ tục PL/SQL để xử lý nghiệp vụ.
* Tích hợp với các hệ thống bên ngoài qua SOAP/REST API,   
    
  message queue hoặc batch job.
* Phần frontend có thể là Web (.NET), Desktop (WinForms), hoặc tích hợp Mobile App (dành cho khách hàng cá nhân).
* **Tính năng nổi bật**
* Xử lý hàng trăm nghìn lệnh/ngày với độ trễ thấp.
* Khả năng cấu hình linh hoạt theo từng công ty chứng khoán (rules, hạn mức, biểu phí,…)
* Hệ thống báo cáo đa dạng, có thể xuất ra Excel, PDF hoặc tích hợp Crystal Reports.
* Bảo mật cao với cơ chế phân quyền, mã hóa dữ liệu, logging chi tiết.

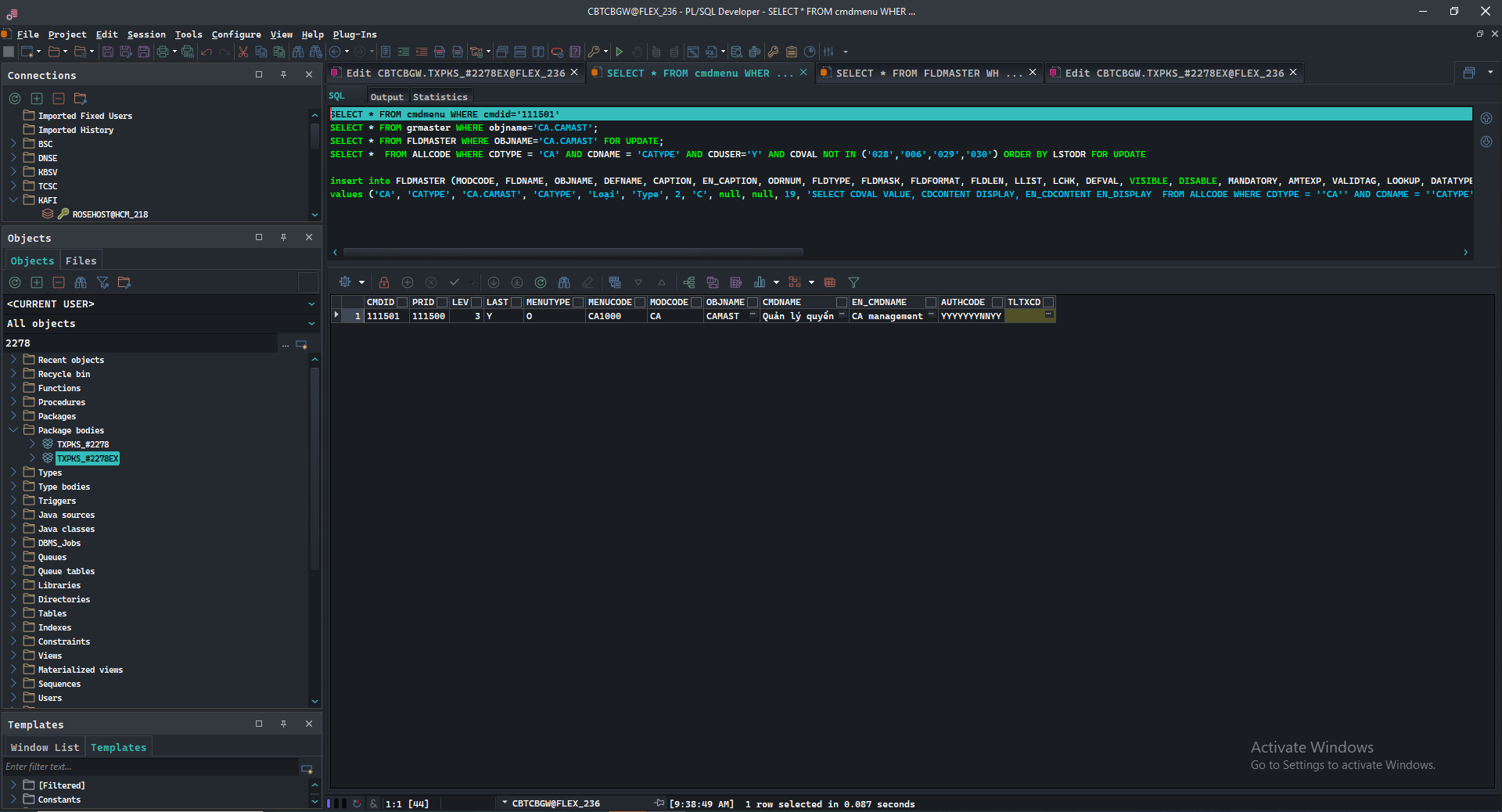
2.3.2. Công việc đã làm

Trong thời gian thực tập tại Công ty Cổ phần Công nghệ Tài chính FSS, em đã được tham gia vào các công việc cụ thể liên quan đến bảo trì và phát triển hệ thống Core giao dịch chứng khoán Flex. Các đầu việc chính bao gồm:

* Sửa và cấu hình Client để hiển thị giao diện động dựa trên dữ liệu được định nghĩa từ cơ sở dữ liệu. Giao diện được sinh tự động từ các bảng hệ thống lưu trong database như SEARCH, SEARCHFLD, FLDMASTER,...  
    
   

Hình 17 : Flex Example

* Sửa đổi và phát triển các package, procedure, function trong Oracle PL/SQL để xử lý các logic nghiệp vụ, chẳng hạn như:
  + Kiểm tra điều kiện khi tạo/sửa lệnh giao dịch;
  + Tự động tính toán thuế, phí giao dịch dựa trên loại chứng khoán;
  + Kiểm tra giới hạn đặt lệnh, hạn mức tài khoản.
  + Kiểm tra điều kiện đặt/sửa lệnh (kiểm tra trạng thái tài khoản, ngày giao dịch hợp lệ,…).
  + Tính thuế, phí giao dịch tự động (dựa theo loại tài sản: cổ phiếu, trái phiếu, ETF,…).
  + Kiểm tra hạn mức theo khách hàng, loại lệnh và giá trị giao dịch.



Hình 18 : PL/SQL - Lập trình database

* Tham gia chỉnh sửa các báo cáo Crystal Report để bổ sung chỉ tiêu, định dạng, điều kiện lọc theo yêu cầu của bộ phận nghiệp vụ.
  + Sử dụng phần mềm **SAP Crystal Reports** để mở các mẫu báo cáo .rpt, cập nhật các chỉ tiêu (field), thêm điều kiện lọc động.
  + Thêm biểu đồ, tổng hợp số liệu theo nhóm dữ liệu và kiểm tra hiển thị trên preview thực tế.

-> Tùy biến báo cáo theo yêu cầu nghiệp vụ, học cách thao tác với dữ liệu đầu ra từ PL/SQL.

* Fix bug giao diện hiển thị sai dữ liệu do lỗi ánh xạ không khớp giữa client và kết quả trả về từ stored procedure.
* Thêm mới hoặc chỉnh sửa các bảng cấu hình (ví dụ: bảng định nghĩa trường dữ liệu, loại lệnh, loại tài sản...) để đáp ứng các yêu cầu từ phía nghiệp vụ mà không cần sửa code.
* Kiểm tra và test chức năng mới sau khi sửa, sử dụng dữ liệu thực tế trên môi trường DEV/TEST để đảm bảo tính chính xác của hệ thống.
* Sử dụng TortoiseGit để commit code, quản lý version và phối hợp làm việc nhóm.
* Học cách sử dụng Docker Desktop để dựng nhanh môi trường chạy thử client kết nối với dịch vụ backend.
  + Học cách dùng Docker Desktop để chạy image chứa server backend test.
  + Kết nối Flex Client với Docker Container chạy service backend để kiểm tra chức năng end-to-end mà không cần cài đặt nhiều.
* Viết tài liệu mô tả chức năng hoặc flow xử lý để hỗ trợ đồng nghiệp trong nhóm nắm rõ luồng hoạt động của các chức năng đã chỉnh sửa.
  + Sau khi thực hiện chỉnh sửa hoặc bổ sung tính năng, viết tài liệu Markdown hoặc Word mô tả:
    - Mục đích chức năng, logic xử lý chính.
    - Thay đổi đã thực hiện.
    - Các luồng xử lý liên quan và ảnh hưởng.
  + Tài liệu được lưu trên GitLab hoặc chia sẻ nội bộ để hỗ trợ cho các thành viên khác review, bảo trì sau này.

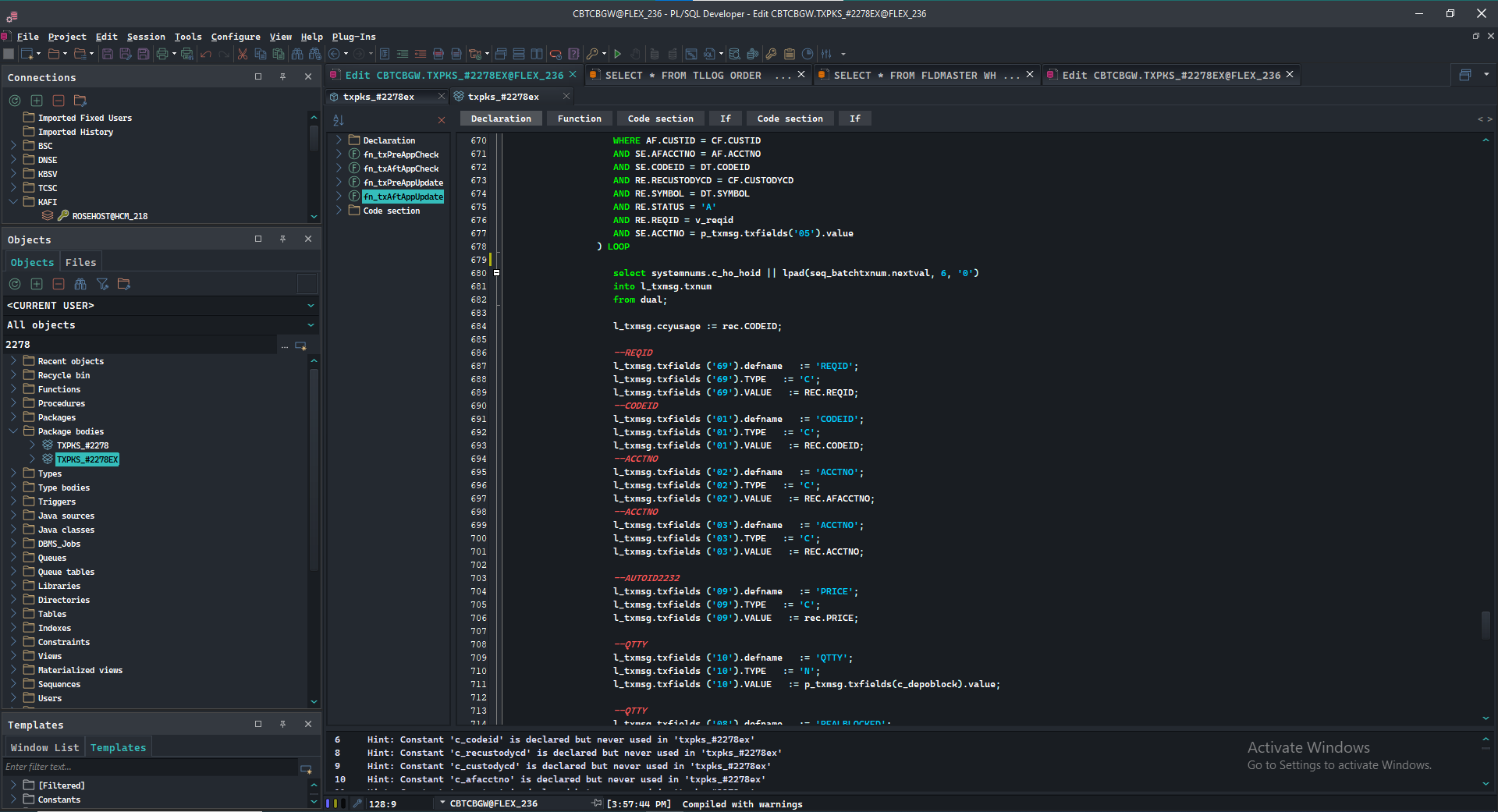
2.3.3. Công nghệ sử dụng

- PL/SQL : Lập trình database, phát triển logic nghiệp vụ phức tạp dưới database (chủ yếu là Oracle).

Trong quá trình làm việc với hệ thống sử dụng cơ sở dữ liệu Oracle, em đã được làm quen và thực hành viết PL/SQL (Procedural Language/SQL) – một ngôn ngữ mở rộng của SQL được Oracle phát triển, cho phép lập trình thủ tục bên trong cơ sở dữ liệu.

Cụ thể, em đã thực hiện một số công việc liên quan đến PL/SQL như sau:

* Viết stored procedure để xử lý dữ liệu sau khi người dùng nhập từ giao diện. Ví dụ: tự động cập nhật trạng thái đơn hàng sau khi người duyệt thao tác.
* Tạo function PL/SQL để tính toán dữ liệu trung bình và định dạng dữ liệu đầu ra phục vụ báo cáo.
* Tối ưu hóa truy vấn SQL trong PL/SQL bằng cách sử dụng các biến nội bộ (DECLARE, BEGIN, END) và kiểm soát luồng bằng các cấu trúc IF, FOR, WHILE.
* Sử dụng CURSOR để duyệt qua từng dòng dữ liệu và xử lý theo điều kiện.
* Áp dụng EXCEPTION HANDLING trong PL/SQL để bắt lỗi và ghi log khi có vấn đề xảy ra.



Hình 19 : Package giao dịch trong PL/SQL

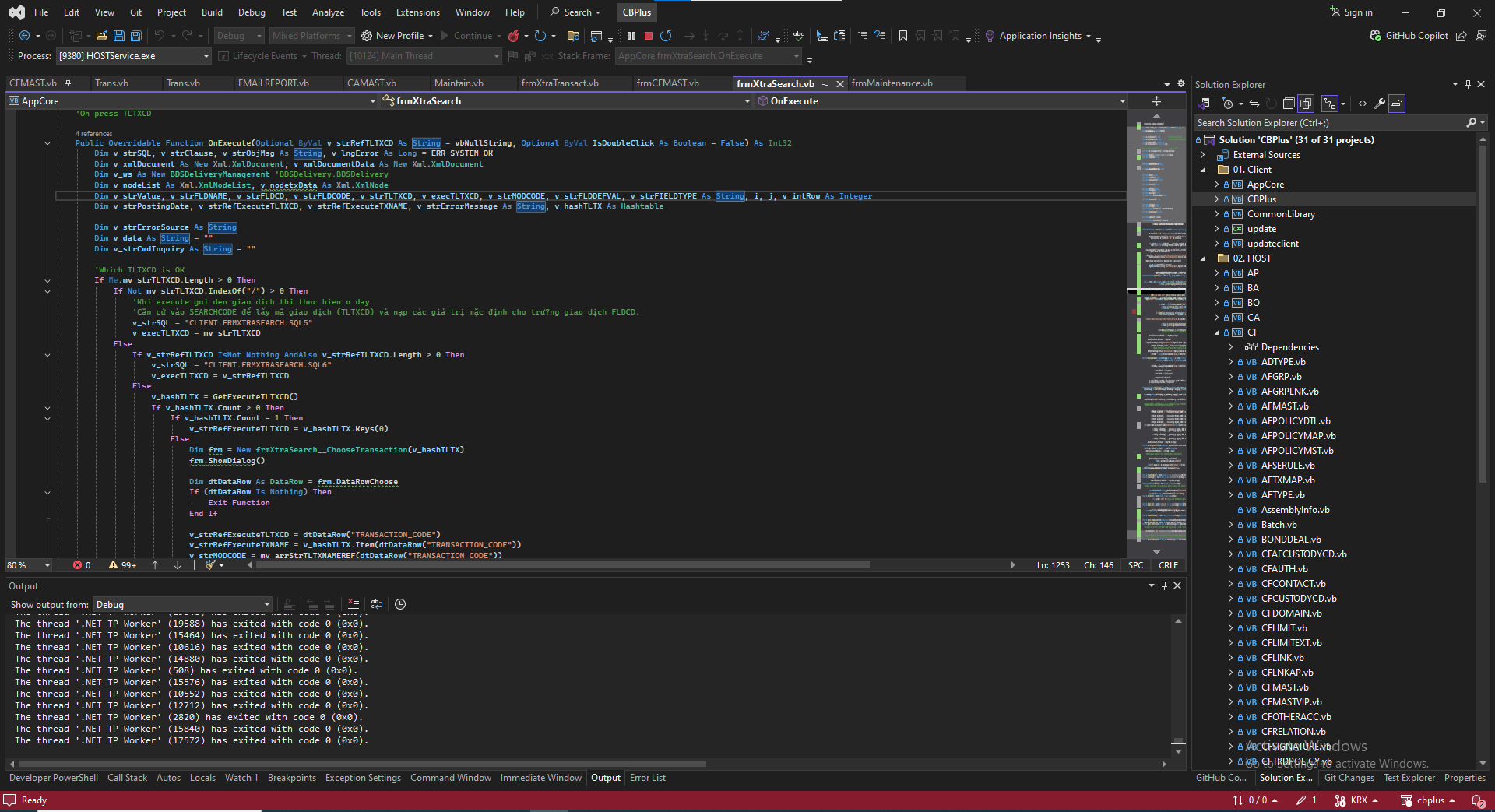
- Visual Studio : IDE phát triển ứng dụng C#, Visual Basic do Microsoft cung cấp.

Trong quá trình thực tập, em sử dụng Visual Studio 2022 làm công cụ chính để  
phát triển và kiểm thử ứng dụng web sử dụng nền tảng .NET/.NET Core.

Visual Studio là một môi trường phát triển tích hợp (IDE) mạnh mẽ, hỗ trợ đầy đủ các tính năng từ viết mã, gỡ lỗi, quản lý source control cho đến triển khai ứng dụng.

Cụ thể, em đã sử dụng Visual Studio để:

* Tạo và quản lý dự án ASP.NET Core: Em sử dụng mẫu Web API/MVC để phát triển các chức năng backend, kết nối với cơ sở dữ liệu Oracle thông qua Entity Framework hoặc ADO.NET.
* Tạo thêm các form mới trong ứng dụng window form để render giao diện nghiệp vụ.
* Viết và tổ chức mã nguồn: Visual Studio hỗ trợ IntelliSense giúp gợi ý cú pháp, tự động hoàn thành và phát hiện lỗi ngay trong quá trình viết mã.
* Gỡ lỗi (debug): Sử dụng tính năng đặt breakpoint và theo dõi giá trị biến theo từng dòng, giúp em nhanh chóng phát hiện và khắc phục lỗi logic.
* Kết nối Git: Em kết nối Visual Studio với GitLab để thực hiện các thao tác như pull/push, commit, tạo branch hoặc tạo pull request.
* Quản lý NuGet packages: Em cài đặt và cập nhật các thư viện như Serilog, Swashbuckle (Swagger), Oracle.ManagedDataAccess,… phục vụ cho quá   
  trình phát triển.



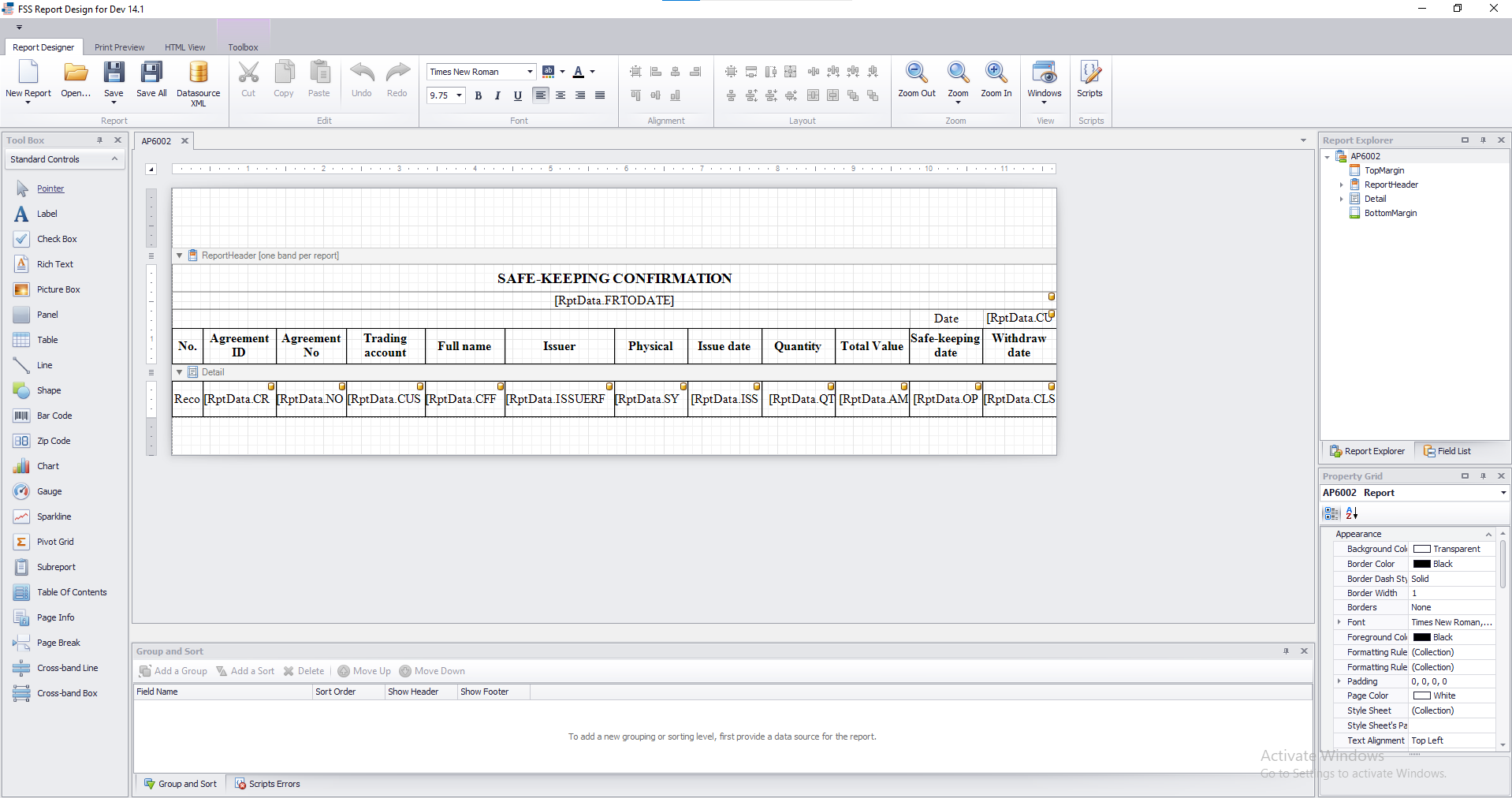
Hình 20 : Visual Studio - Lập trình C#

- Crystal Report

Trong thời gian thực tập tại công ty, em có cơ hội được tiếp cận và làm việc với Crystal Reports – một công cụ mạnh mẽ dùng để tạo và thiết kế các báo cáo động từ cơ sở dữ liệu.

Các công việc em đã thực hiện với Crystal Reports:

* Tạo báo cáo dựa trên dữ liệu Oracle: Em sử dụng Crystal Reports để tạo báo   
    
  cáo từ dữ liệu trong hệ quản trị Oracle, thông qua kết nối OLE DB hoặc trực   
  tiếp từ view/stored procedure.
* Thiết kế giao diện báo cáo: Em thực hiện kéo thả các trường dữ liệu, nhóm dữ liệu theo tiêu chí (ví dụ: theo phòng ban, ngày tạo,…), chèn biểu đồ, tổng cộng, định dạng số và ngày tháng.
* Tạo tham số lọc (parameter): Cho phép người dùng chọn ngày, mã nhân viên hoặc loại dữ liệu để in báo cáo theo nhu cầu.
* Tích hợp báo cáo vào ứng dụng .NET: Em sử dụng CrystalReportViewer trong ứng dụng WinForms hoặc WebForms để hiển thị báo cáo cho người dùng cuối.
* Tối ưu tốc độ tải báo cáo bằng cách cấu hình truy vấn chỉ lấy các cột, dòng cần thiết và giới hạn dữ liệu đầu vào thông qua tham số động.



Hình 21 : Crystal Report - Thiết kế báo cáo trong .NET

- TortoiseGit

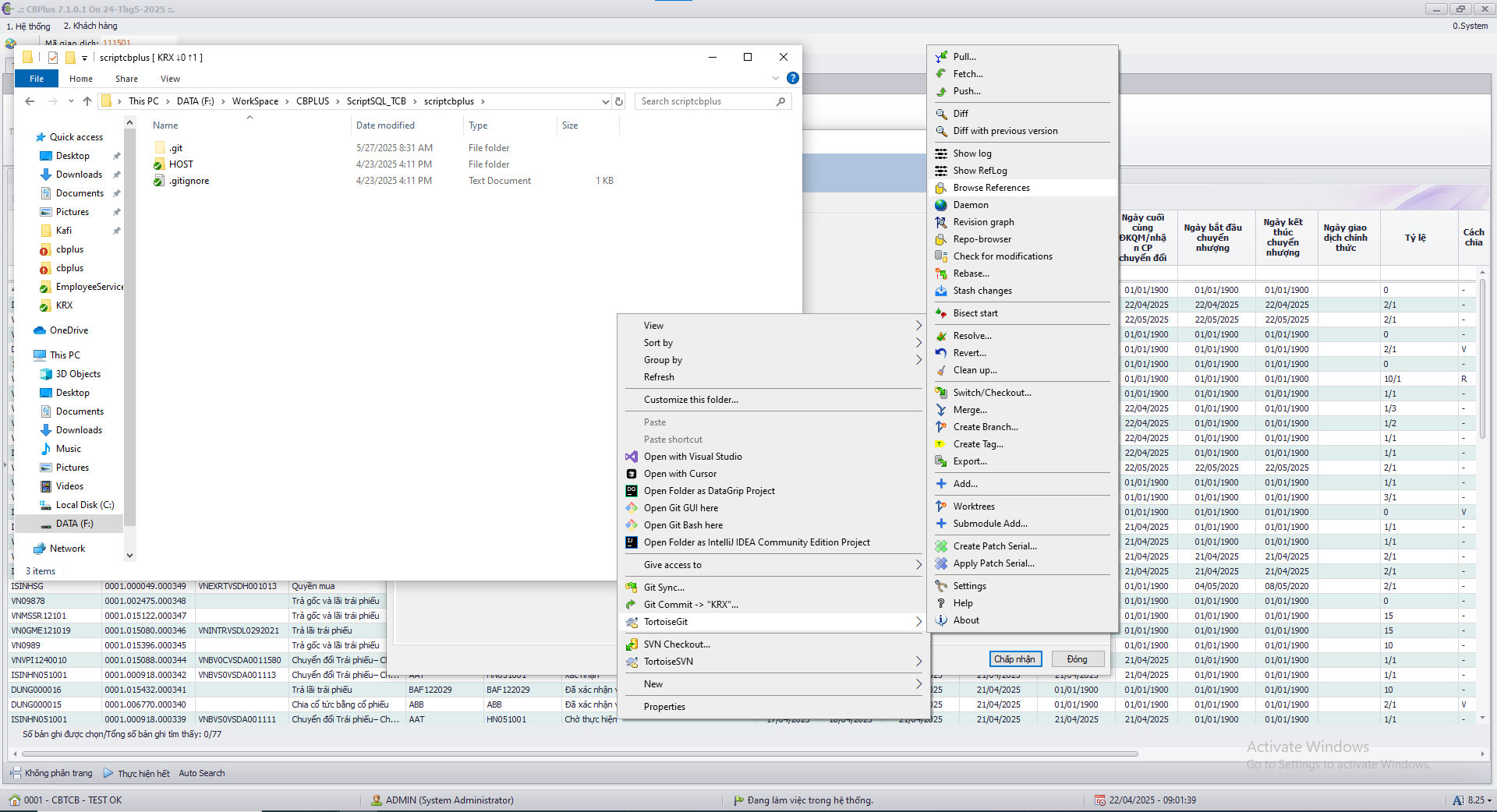
Trong quá trình thực tập tại công ty, em được sử dụng công cụ TortoiseGit để làm việc với hệ thống quản lý mã nguồn Git. Đây là một phần mềm mã nguồn mở, tích hợp trực tiếp vào Windows Explorer, giúp người dùng thao tác với Git một cách trực quan thông qua giao diện đồ họa.

+ Các thao tác em thực hiện với TortoiseGit:

* Clone dự án từ GitLab về máy tính cá nhân để phát triển chức năng.
* Pull code mới nhất từ remote repository về trước khi bắt đầu làm việc.
* Commit và push mã nguồn sau khi hoàn thành chức năng hoặc fix lỗi.
* Tạo branch riêng để làm việc độc lập trên từng task mà không ảnh hưởng đến nhánh chính.
* So sánh sự khác biệt (Diff) giữa các phiên bản file để kiểm tra thay đổi.
* Thực hiện merge hoặc rebase nhánh khi hoàn thành và chuẩn bị đưa lên môi trường tích hợp.

+ Lợi ích khi sử dụng TortoiseGit:

* Giao diện thân thiện, dễ sử dụng, đặc biệt với người mới làm quen Git.
* Có thể thao tác trực tiếp từ File Explorer, không cần dùng dòng lệnh.
* Hỗ trợ hiển thị lịch sử commit, log thay đổi, giúp theo dõi tiến độ dễ dàng.
* Tích hợp công cụ conflict resolver giúp xử lý xung đột khi merge nhanh chóng.



Hình 22 : TortoiseGit - Git UI

- Docker Desktop

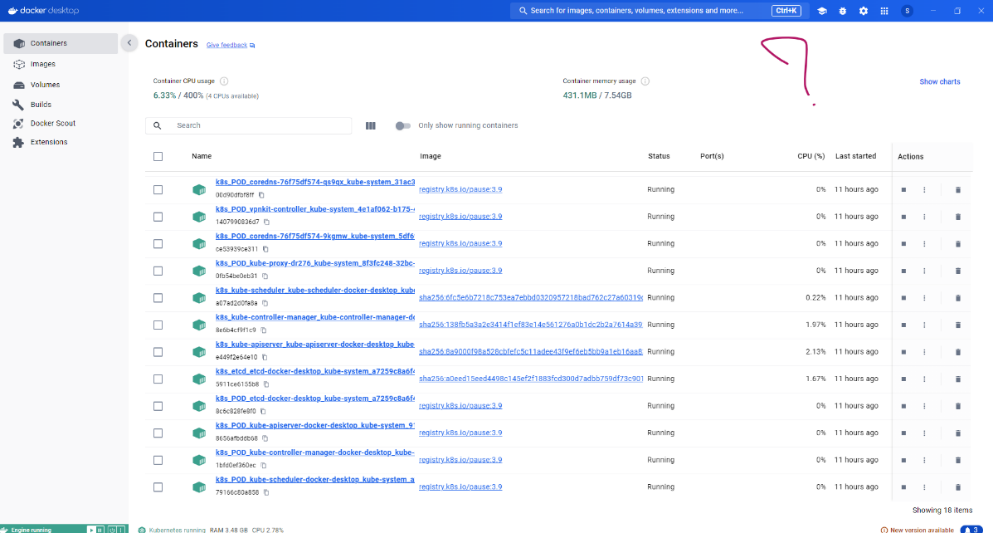
Docker Desktop là một công cụ phổ biến hỗ trợ việc đóng gói, triển khai và chạy ứng dụng trong môi trường container hóa. Trong thời gian thực tập, em sử dụng Docker Desktop để:

+ Các công việc đã thực hiện:

* Khởi động và quản lý các container phục vụ cho hệ thống như: Oracle Database, SQL Server, Kafka, Redis,...
* Tạo và chạy các container ứng dụng .NET hoặc Java từ file Dockerfile.
* Sử dụng docker-compose để triển khai nhiều service đồng thời như API, DB, Kafka, UI,...
* Kiểm tra log container, restart, stop container khi phát sinh lỗi trong quá trình chạy thử hệ thống.
* Làm quen với việc tách biệt môi trường phát triển và môi trường thật, giúp đảm bảo hệ thống hoạt động đồng nhất khi triển khai.

+ Lợi ích:

* Giảm thiểu lỗi do khác biệt môi trường.
* Dễ dàng chia sẻ môi trường phát triển với đồng nghiệp bằng docker-compose.yml.
* Quản lý nhiều dịch vụ phức tạp mà không cần cài đặt thủ công.



Hình 23 : Docker Desktop - Quản lý container

- IntelliJ IDEA

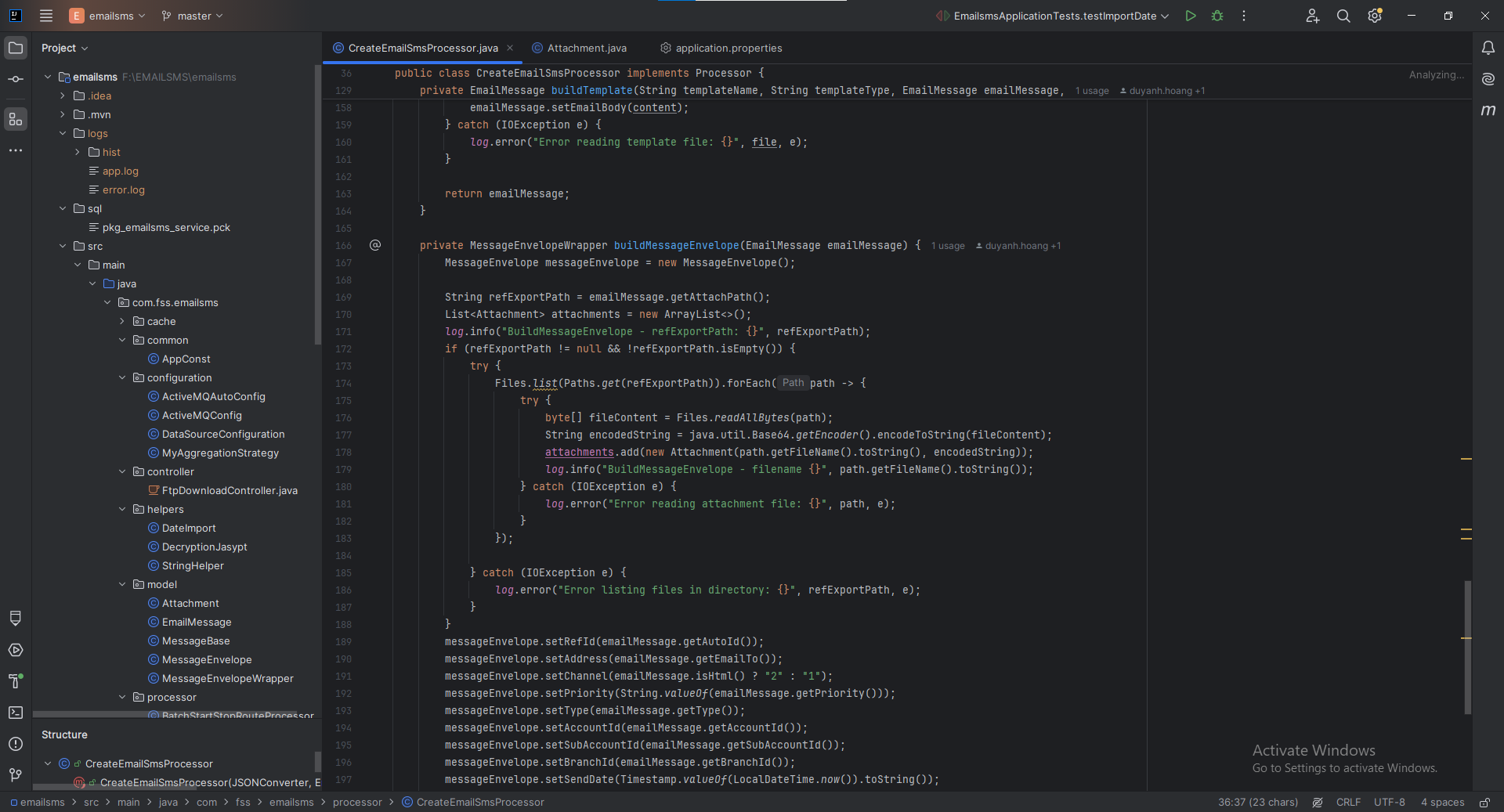
IntelliJ IDEA là một IDE chuyên dụng cho ngôn ngữ Java và các công nghệ liên quan như Spring Boot, Maven, Hibernate,... Em sử dụng IntelliJ IDEA để:

Các công việc đã thực hiện:

* Phát triển các module Java sử dụng Spring Boot hoặc Camel trong hệ thống vi mô (microservices).
* Tích hợp Maven/Gradle để quản lý thư viện và cấu hình project.
* Viết và kiểm thử các API REST, xử lý dữ liệu qua Kafka, hoặc truy vấn database.
* Viết service gửi mail tự động thông báo khách hang bằng spring boot java.
* Sử dụng các plugin hỗ trợ như Lombok, Spring Assistant, Docker, Kafka Tool,...
* Debug chương trình trực tiếp trong IDE để theo dõi giá trị biến và xử lý lỗi.

Lợi ích:

* Tăng hiệu suất viết code nhờ tính năng gợi ý thông minh (IntelliSense) và tự động sửa lỗi.
* Giao diện dễ nhìn, hỗ trợ tốt Spring Framework và các công nghệ phổ biến.
* Tích hợp Git, Docker, Database viewer, rất tiện lợi khi làm việc đa nhiệm.



Hình 24 : IntelliJ IDE - Lập trình Java

- SonarQube : Trong quá trình thực tập, để kiểm tra code clean, em đã được tiếp xúc thêm với 1 công nghệ là SonarQube.

SonarQube là một công cụ phân tích mã nguồn tĩnh (Static Code Analysis) giúp phát hiện:

* Bug (lỗi logic),
* Code Smell (đoạn code "có mùi" – khó đọc, khó bảo trì),
* Vulnerability (lỗ hổng bảo mật),
* Duplicated Code (mã bị lặp lại),
* Test Coverage (độ bao phủ của unit test),
* …và nhiều chỉ số chất lượng khác.

A diagram of a software company

AI-generated content may be incorrect.

Hình 25 : SonarQube Architecture

## 2.4. Môi trường làm việc tại FSS

* Trong suốt quá trình thực tập tại Công ty Cổ phần Giải pháp Phần mềm Tài chính FSS, em có cơ hội trải nghiệm một môi trường làm việc chuyên nghiệp, thân thiện và đầy tính học hỏi.
* Không gian làm việc tại FSS được bố trí hiện đại, rộng rãi, với đầy đủ tiện nghi như phòng họp, phòng training, khu vực pantry và khu nghỉ ngơi dành cho nhân viên. Mỗi nhân viên được trang bị máy tính cấu hình cao, có màn hình kép phục vụ việc lập trình và kiểm thử hiệu quả hơn. Mạng nội bộ nhanh và ổn định, hỗ trợ truy cập các hệ thống nội bộ và môi trường test/deploy dễ dàng.
* Văn hóa công ty đề cao sự hợp tác, chủ động và chia sẻ. Mặc dù em chỉ là một thực tập sinh, nhưng luôn được anh chị trong team nhiệt tình hướng dẫn, từ việc đọc tài liệu hệ thống đến cách phân tích yêu cầu và thực hiện các tác vụ lập trình cụ thể. Các buổi daily meeting diễn ra vào đầu mỗi sáng giúp tôi nắm được tiến độ chung của team và học cách trình bày công việc một cách rõ ràng.
* Ngoài giờ làm việc, công ty còn tổ chức các buổi sinh hoạt nội bộ như “FSS Friday Talk”, nơi các kỹ sư chia sẻ kiến thức, kỹ thuật hoặc kinh nghiệm thực tế. Đây là điểm cộng rất lớn trong việc tạo nên một môi trường học hỏi và phát triển bền vững cho nhân viên, kể cả thực tập sinh.

## 2.5. Đánh giá và đề xuất

- Đánh giá chung : Sau gần 2 tháng thực tập tại FSS, em nhận thấy công ty có một quy trình làm việc bài bản, đặc biệt là trong việc quản lý dự án, kiểm thử và triển khai hệ thống. Các công nghệ được sử dụng đều là những công nghệ tiên tiến, phù hợp với yêu cầu của doanh nghiệp lớn như ngân hàng, tài chính. Em đã học được rất nhiều từ những dự án thực tế mà mình tham gia, cả về mặt kỹ thuật (.NET, Oracle Database, GitLab CI/CD, SonarQube) lẫn quy trình làm việc chuyên nghiệp (Agile, Code Review, Deployment,...).

- Ưu điểm nổi bật:

* Văn hóa hỗ trợ, chia sẻ giữa các thành viên rất tốt.
* Quy trình làm việc rõ ràng, có tài liệu và hướng dẫn cụ thể.
* Được tiếp cận với công nghệ và hệ thống quy mô lớn.
* Đội ngũ mentor có kinh nghiệm và luôn sẵn sàng hỗ trợ.

- Một số đề xuất cải thiện (mang tính xây dựng):

* Nên có thêm các tài liệu hướng dẫn chi tiết hơn cho người mới (onboarding doc, sơ đồ hệ thống).
* Có thể tổ chức thêm các buổi workshop chuyên sâu về kỹ thuật như tối ưu truy vấn Oracle, bảo mật trong .NET,...
* Tăng cường cơ hội để thực tập sinh được tham gia sâu hơn vào quá trình phân tích nghiệp vụ hoặc demo sản phẩm.

# Kinh nghiệm và học hỏi

* Kinh nghiệm về chuyên môn kỹ thuật
* PL/SQL và tư duy xử lý nghiệp vụ tại tầng database  
  Trước khi thực tập, em chỉ quen viết các câu truy vấn SQL đơn giản, chưa có nhiều kinh nghiệm xử lý logic tại tầng cơ sở dữ liệu. Trong quá trình thực tập, em đã được tham gia sửa đổi và viết mới nhiều đoạn mã PL/SQL như Function, Procedure, Package, giúp em:
  + Biết cách chia nhỏ nghiệp vụ thành các bước xử lý logic rõ ràng;
  + Sử dụng biến, vòng lặp, con trỏ (cursor), và xử lý lỗi (EXCEPTION) trong PL/SQL;
  + Tối ưu hóa truy vấn để giảm thời gian xử lý và hạn chế ảnh hưởng tới hiệu năng hệ thống;
  + Viết các hàm xử lý nghiệp vụ kiểm tra điều kiện giao dịch như: xác minh hạn mức tài khoản, kiểm tra quyền đặt lệnh, tính phí và thuế.
* Làm quen với hệ thống client sinh động từ database  
  Đây là lần đầu em làm việc với một client không code tay từng form, mà sử dụng các bảng metadata trong Oracle để render giao diện động. Việc này giúp em hiểu sâu hơn về cách dữ liệu có thể “lái” giao diện:
  + Em đã hiểu rõ cấu trúc bảng lưu field, tab, group trong hệ thống;
  + Học được cách sửa metadata để thêm trường, sửa label, ẩn/hiện các thành phần trên giao diện;
  + Giúp giảm thiểu việc sửa code ở client và linh hoạt cập nhật giao diện theo yêu cầu mới.
* Hiểu được mô hình xử lý nghiệp vụ “bên trong database”  
  Khác với mô hình nhiều lớp MVC mà em từng học, hệ thống Flex xử lý rất nhiều nghiệp vụ ngay trong DB – do đặc thù hiệu năng và tính chính xác cao của ngành tài chính. Em học được cách thiết kế business logic “gần với dữ liệu”, giúp phản hồi nhanh và thống nhất luồng xử lý.
* Kinh nghiệm về công cụ và môi trường phát triển
* Crystal Report:  
  Em đã trực tiếp sửa và bổ sung các báo cáo in ấn theo yêu cầu nghiệp vụ. Từ đó học được:
  + Cách liên kết Crystal Report với stored procedure;
  + Cách định dạng, nhóm dữ liệu, tính tổng động;
  + Kỹ năng debug lỗi do sai kiểu dữ liệu hoặc dữ liệu đầu vào không đúng định dạng.
* TortoiseGit:  
  Qua việc commit, pull và xử lý conflict, em hiểu được quy trình làm việc nhóm và tầm quan trọng của việc quản lý phiên bản mã nguồn. Em cũng học được cách viết commit message rõ ràng và sử dụng nhánh riêng khi cần thử nghiệm.
* Docker Desktop:  
  Em dùng Docker để dựng môi trường test nhanh chóng, học cách chạy các container của client hoặc middleware, điều mà trước đây em chưa từng làm. Điều này giúp tiết kiệm thời gian và hạn chế sự phụ thuộc vào môi trường máy thật.
* IntelliJ IDEA & Visual Studio:  
  Em được tiếp xúc với cả hai IDE – phục vụ cho việc đọc source code của client và chỉnh sửa backend Java/Pascal (tùy theo hệ thống). Nhờ vậy, em hiểu thêm về cách debug đa tầng và sử dụng các công cụ này hiệu quả hơn.
* Kỹ năng mềm và phong cách làm việc
* Kỹ năng phân tích và đọc hiểu nghiệp vụ:  
  Em học được cách đọc tài liệu mô tả yêu cầu nghiệp vụ, hoặc trao đổi với anh/chị nghiệp vụ để hiểu mục tiêu thực sự của việc sửa. Từ đó, biết cách xác định điểm cần can thiệp (ở client hay dưới DB), tránh làm sai yêu cầu.
* Kỹ năng làm việc theo quy trình chuyên nghiệp:  
  Em được hướng dẫn cách viết mô tả công việc khi nhận task, ghi lại kết quả kiểm thử, gửi review trước khi commit code. Điều này giúp em hiểu rằng kỹ thuật tốt chưa đủ – cần có quy trình rõ ràng để đảm bảo chất lượng phần mềm.
* Tư duy chủ động học hỏi và không ngại thử nghiệm:  
  Có nhiều chức năng ban đầu em chưa hiểu, nhưng qua thử nghiệm, đọc log, đặt breakpoint và hỏi đồng nghiệp, em đã từng bước tự khám phá được cách hệ thống hoạt động. Đây là bài học lớn về sự chủ động trong công việc.

# Kết luận

Thực tập kỹ thuật tại Công ty Cổ phần Chứng khoán FPT (FSS) là một cột mốc quan trọng và đáng nhớ trong hành trình học tập và phát triển nghề nghiệp của em. Trong thời gian thực tập từ ngày 10/02/2025, em đã có cơ hội tiếp cận trực tiếp với hệ thống phần mềm lõi giao dịch chứng khoán Flex – một trong những nền tảng cốt lõi hỗ trợ các hoạt động giao dịch, đặt lệnh và xử lý nghiệp vụ tại FSS.

Thông qua quá trình thực tập, em đã được giao nhiều đầu việc thực tế như: hiệu chỉnh giao diện client để render động từ dữ liệu cấu hình trong cơ sở dữ liệu, tham gia chỉnh sửa các package và function trên Oracle PL/SQL để xử lý logic nghiệp vụ, hỗ trợ kiểm tra và đảm bảo tính ổn định khi deploy các thay đổi vào môi trường UAT/Production. Ngoài ra, em cũng đã làm việc với các công cụ kỹ thuật phổ biến như TortoiseGit để quản lý mã nguồn, Visual Studio cho xử lý form client, cũng như tham gia chỉnh sửa Crystal Report để in ấn báo cáo giao dịch và dữ liệu người dùng.

Môi trường làm việc tại FSS với đặc thù nghiệp vụ tài chính – chứng khoán đòi hỏi tính chính xác, bảo mật và hiệu suất cao đã giúp em hiểu sâu hơn về cách một hệ thống tài chính lớn vận hành. Qua đó, em không chỉ rèn luyện được kỹ năng kỹ thuật mà còn nâng cao tư duy hệ thống, khả năng phân tích nghiệp vụ, kỹ năng làm việc nhóm và phối hợp với các bộ phận khác như QA, BA, DevOps.

Quá trình thực tập đã giúp em củng cố định hướng nghề nghiệp rõ ràng hơn, đồng thời trang bị cho em nền tảng thực tế vững chắc để bước vào môi trường làm việc chính thức sau khi tốt nghiệp. Em xin chân thành cảm ơn Ban Lãnh đạo FSS, các anh chị trong Phòng Công nghệ thông tin – đặc biệt là đội Core Flex – đã tạo điều kiện, hỗ trợ và hướng dẫn tận tình trong suốt quá trình em thực tập tại công ty.

Em hy vọng những kinh nghiệm và bài học thu nhận được trong kỳ thực tập tại FSS sẽ là hành trang quý báu giúp em tiếp tục phát triển bản thân, đóng góp nhiều hơn cho ngành công nghệ và tài chính trong tương lai.

# Tài liệu tham khảo

[1] FSS, Giới thiệu về Công ty Cổ phần Giải pháp phần mềm Tài chính FSS, Trang chủ FSS, 2024.  
[2] FSS, Tài liệu đào tạo nhân viên mới, Hướng dẫn quy trình phát triển phần mềm tại FSS, 2023.  
[3] FSS, Tài liệu hướng dẫn sử dụng hệ thống Core giao dịch chứng khoán Flex, Nội bộ công ty, 2023.  
[4] FSS, Quy trình kiểm thử và triển khai phần mềm tại FSS, 2023.