ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN

BÁO CÁO ĐỒ ÁN

ĐIỆN TOÁN ĐÁM MÂY

Đề tài:

Quản lý dự án web thương mại điện tử trên môi trường cloud

Giảng viên lý thuyết: ThS. Hà Lê Hoài Trung

Lớp: IS402.P11

Nhóm sinh viên thực hiện:

1. Cao Hoài Sang MSSV: 21522541
2. Thi Thành Công MSSV: 21521897
3. Nguyễn Trần Gia Kiệt MSSV: 21522258

TP. Hồ Chí Minh, Năm 2024

LỜI CẢM ƠN

Trước hết, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến tập thể quý thầy cô trường Đại học Công nghệ Thông tin - Đại học Quốc gia TP.HCM và quý thầy cô khoa Hệ thống thông tin đã tạo điều kiện, giúp chúng em học tập và có được những kiến thức cơ bản làm tiền đề giúp chúng em hoàn thành được dự án này.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy ThS. Hà Lê Hoài Trung (Giảng viên giảng dạy môn Điện toán đám mây – IS402.P11). Nhờ sự hướng dẫn tận tình và chu đáo của thầy, nhóm chúng em đã học hỏi được nhiều kinh nghiệm và hoàn thành thuận lợi, đúng tiến độ cho dự án của mình.

Ngoài ra, chúng em cũng gửi lời cảm ơn đến tập thể lớp IS402.P11 khoảng thời gian qua đã đồng hành cùng nhau. Cảm ơn sự đóng góp của tất cả các bạn cho những buổi học luôn sôi nổi, thú vị và dễ tiếp thu.

Trong quá trình thực hiện đồ án, nhóm chúng em luôn giữ một tinh thần cầu tiến, học hỏi và cải thiện từ những sai lầm, tham khảo từ nhiều nguồn tài liệu khác nhau và luôn mong tạo ra được sản phẩm chất lượng nhất có thể. Tuy nhiên, do vốn kiến thức còn hạn chế trong quá trình trau dồi từng ngày, nhóm chúng em không thể tránh được những sai sót, vì vậy chúng em mong rằng quý thầy cô sẽ đưa ra nhận xét một cách chân thành để chúng em học hỏi thêm kinh nghiệm nhằm mục đích phục vụ tốt các dự án khác trong tương lai. Xin chân thành cảm ơn quý thầy cô!

Nhóm thực hiện

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

....., ngày…….tháng……năm 2024

Người nhận xét

(Ký tên và ghi rõ họ tên**)**

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN

1.1. Giới thiệu loại bài toán thuộc nhóm

- Lưu trữ: Bài toán này liên quan đến việc CI/CD, xử lý, lưu trữ dữ liệu và deploy phần mềm/ứng dụng trên môi trường cloud

1.2. Loại dữ liệu

- Dữ liệu trong bài toán này có thể bao gồm dữ liệu giao dịch thương mại điện tử.

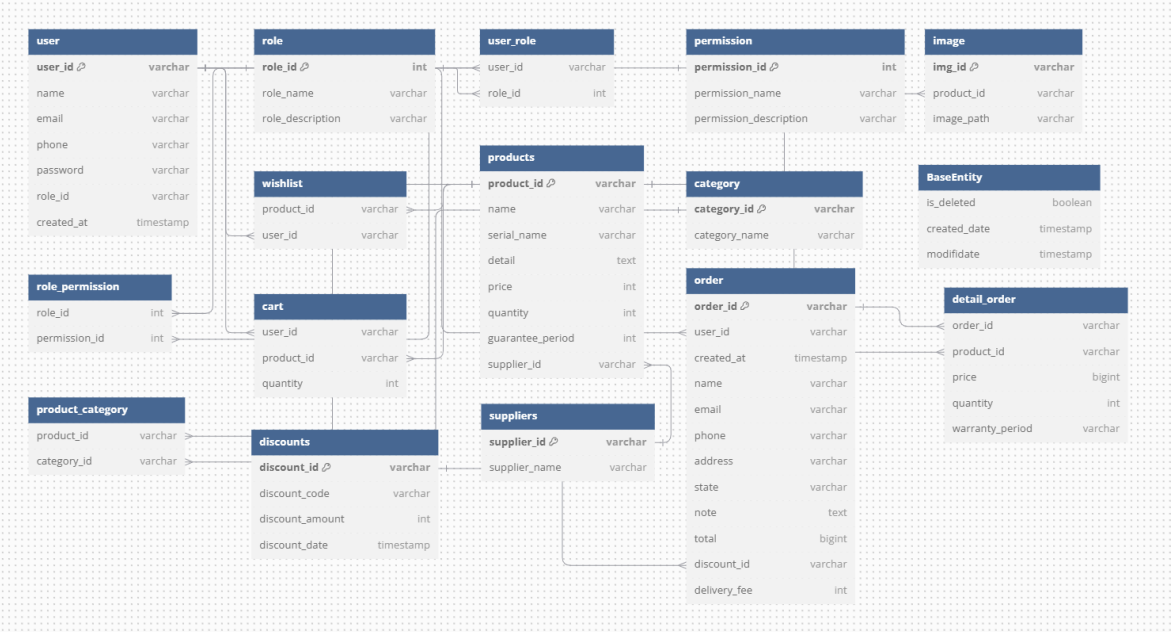

Figure 1 Lượt đồ cơ sở dữ liệu

1.3. Kích thước dữ liệu – dung lượng bộ nhớ cho xử lý

- Kích thước dữ liệu: Dữ liệu cần xử lý có dung lượng lớn hơn 4GB.
- So sánh tốc độ xử lý:
    - Máy chủ truyền thống: Xử lý dữ liệu trên máy chủ truyền thống có thể gặp hạn chế về khả năng mở rộng và hiệu suất, đặc biệt khi dung lượng dữ liệu lớn.

Id	endpoint	method	request_time
1	http://127.0.0.1:8000/api/product	GET	0.0493
2	http://127.0.0.1:8000/api/product	GET	0.0418
3	http://127.0.0.1:8000/api/category	GET	0.0059
4	http://127.0.0.1:8000/api/test	GET	0.0031
5	http://127.0.0.1:8000/api/product/...	POST	0.0185
6	http://127.0.0.1:8000/api/category	GET	0.0058
7	http://127.0.0.1:8000/api/product/...	POST	0.0164
8	http://127.0.0.1:8000/api/product/...	POST	0.0169
9	http://127.0.0.1:8000/api/product/...	POST	0.0155
10	http://127.0.0.1:8000/api/product/...	POST	0.0183
11	http://127.0.0.1:8000/api/cart/GetC...	POST	0.0081
12	http://127.0.0.1:8000/api/cart/Get...	POST	0.0044
13	http://127.0.0.1:8000/api/product/...	POST	0.0138
14	http://127.0.0.1:8000/api/product/...	POST	0.0144
15	http://127.0.0.1:8000/api/product/...	POST	0.0139
16	http://127.0.0.1:8000/api/product/...	POST	0.0169
17	http://127.0.0.1:8000/api/product/...	POST	0.0141
18	http://127.0.0.1:8000/api/cart/GetC...	POST	0.0041
19	http://127.0.0.1:8000/api/cart/Get...	POST	0.0042
20	http://127.0.0.1:8000/api/category	GET	0.0051
21	http://127.0.0.1:8000/api/cart/GetC...	POST	0.0044
22	http://127.0.0.1:8000/api/cart/GetC...	POST	0.0040
23	http://127.0.0.1:8000/api/product/...	POST	0.0177
24	http://127.0.0.1:8000/api/category	GET	0.0052
25	http://127.0.0.1:8000/api/order/Ge...	POST	0.0063
26	http://127.0.0.1:8000/api/product/...	POST	0.0144
27	http://127.0.0.1:8000/api/order/Ge...	POST	0.0054
28	http://127.0.0.1:8000/api/category	GET	0.0053
29	http://127.0.0.1:8000/api/order/Ge...	POST	0.0094

Figure 2 Tốc độ xử lý local

- Hệ thống cloud: Sử dụng hệ thống cloud như Azure cho phép mở rộng linh hoạt và cải thiện hiệu suất xử lý nhờ vào các dịch vụ như AKS và Azure Web Apps. Việc triển khai tự động hóa với Azure DevOps giúp tối ưu hóa quy trình build, test và deploy.

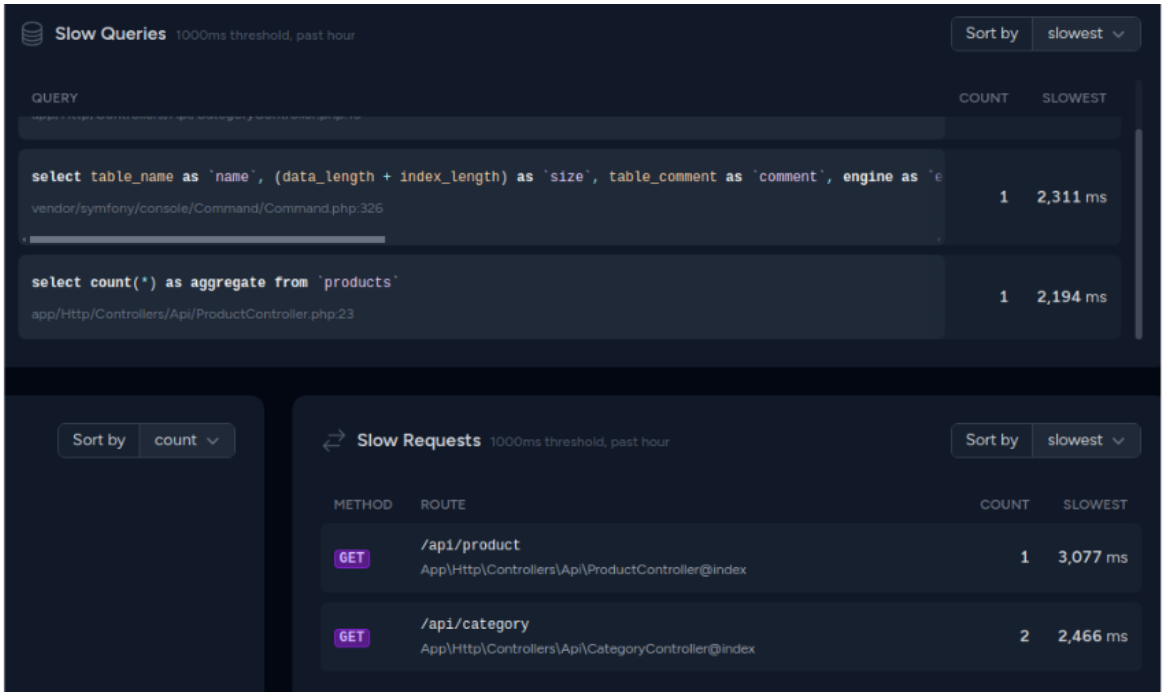

Figure 3 Tốc độ đọc ghi trên cloud (1)

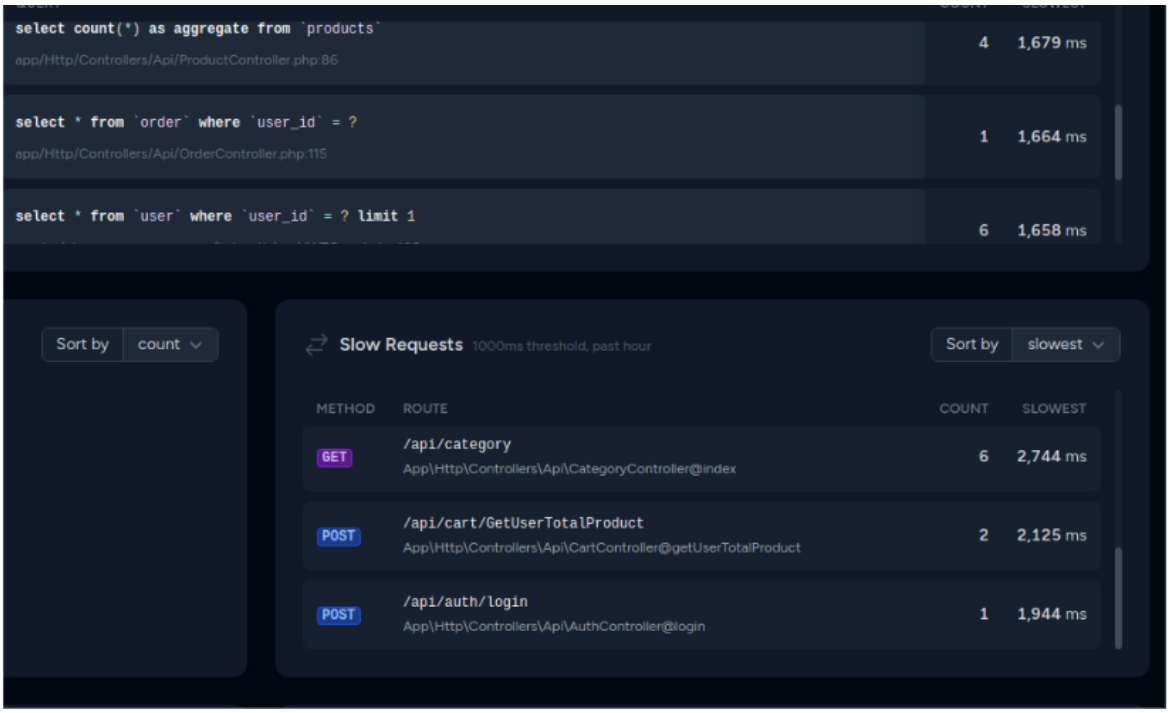

Figure 4 Tốc độ đọc ghi trên cloud (2)

1.4. Liệt kê, Giới thiệu các thành phần sử dụng trong cloud

- PaaS: Azure DevOps (CI/CD), Azure Web Apps, Azure Key Vault, Data Factory (V2), Container Registry Webhook, Storage Account.
- SaaS: Azure Repos, Azure Application Insights, Visual Studio Code, Azure DevOps Backlog

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Cơ sở lý thuyết về định dạng lưu trữ

2.1.1 Định dạng lưu trữ mã nguồn và cấu hình

Trong pipeline CI/CD, mã nguồn ứng dụng, cũng như các cấu hình để xây dựng và triển khai, được lưu trữ trong các kho mã nguồn (repository) và được tổ chức dưới dạng các file có cấu trúc nhất định.

Git repositories (ví dụ: Azure Repos, GitHub, GitLab): Mã nguồn thường được lưu trữ trong các repository, đây là hệ thống phân tán giúp theo dõi các thay đổi trong mã nguồn qua các commit:

- .git: Dấu hiệu của một Git repository.
- .gitignore: Cấu hình để chỉ định các file hoặc thư mục không cần được theo dõi trong Git (như các file tạm hoặc file nhạy cảm).
- README.md: Mô tả tổng quan về dự án hoặc các hướng dẫn sử dụng, thường được viết bằng Markdown.
- Dockerfile: File cấu hình Docker, giúp xây dựng các Docker image từ mã nguồn.

2.1.2 Định dạng file cấu hình trong pipeline:

- Trong các pipeline CI/CD, định dạng file cấu hình rất quan trọng để xác định các bước xử lý và các tham số cần thiết cho quá trình build, test và deploy.

Figure 5 File JSON/YAML

- YAML: Được sử dụng để cấu hình quy trình CI/CD. File này chứa các bước build, test và deploy được tổ chức dưới dạng các job, stage và task. YAML được sử dụng vì tính dễ đọc và khả năng định nghĩa các bước rất linh hoạt. Trong Azure DevOps, bạn có thể sử dụng YAML để tự động hóa các công việc liên quan đến CI/CD, như xây dựng, kiểm thử và triển khai.
- JSON(.json): Được sử dụng trong một số công cụ khác như Jenkins hoặc Terraform để mô tả các pipeline hoặc cấu hình công việc. Cũng giống như YAML, JSON có thể lưu trữ cấu trúc dữ liệu, tuy nhiên YAML thường dễ sử dụng và trực quan hơn trong các tình huống CI/CD.
- Dockerfile: Dockerfile là một tệp cấu hình đặc biệt được sử dụng để xây dựng Docker image. Trong môi trường pipeline, Dockerfile được sử dụng để xác định cách thức xây dựng một container với ứng dụng cụ thể.

2.1.3 Định dạng tệp logs và báo cáo

- Trong quá trình pipeline CI/CD, logs và báo cáo giúp theo dõi và phân tích các lỗi hoặc thông tin quá trình:
  - .log hoặc .txt: Các tệp log thường được tạo ra trong các bước như build, test hoặc deploy, giúp người phát triển phân tích chi tiết lỗi hoặc thông báo trong suốt quá trình.
  - Test reports: Tệp báo cáo kết quả kiểm thử thường có định dạng XML hoặc HTML:
    - JUnit: Định dạng báo cáo kiểm thử phổ biến trong CI/CD. Kết quả kiểm thử có thể được lưu trữ dưới dạng tệp XML để phân tích và tạo báo cáo chi tiết.
    - SonarQube reports: Được sử dụng để kiểm tra chất lượng mã nguồn và báo cáo về các vấn đề bảo mật và hiệu suất.

2.2 Cơ sở lý thuyết về thuật toán xử lý

2.2.1 Thuật toán quản lý mã nguồn (Azure Repos & Git)

- SHA-1 Hashing Algorithm:

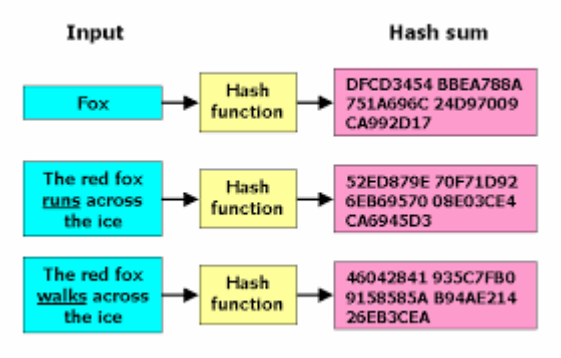

Figure 6 SHA-1 Hashing Algorithm

  - Lý thuyết: Thuật toán băm SHA-1 được sử dụng để tạo một chuỗi 160-bit (40 ký tự) đại diện duy nhất cho nội dung của file hoặc commit. Tất cả các file, thư mục và commit trong Git đều được ánh xạ qua một mã băm này.
  - Ứng dụng:
    - Xác định thay đổi mã nguồn: Khi lập trình viên cập nhật mã, Git sẽ tạo một commit ID bằng thuật toán SHA-1 để theo dõi thay đổi.
    - Trigger pipeline: Khi có thay đổi mã nguồn, Azure Pipelines sẽ kích hoạt một chuỗi các bước tiếp theo dựa trên commit hoặc pull request.

- Merkle Tree:

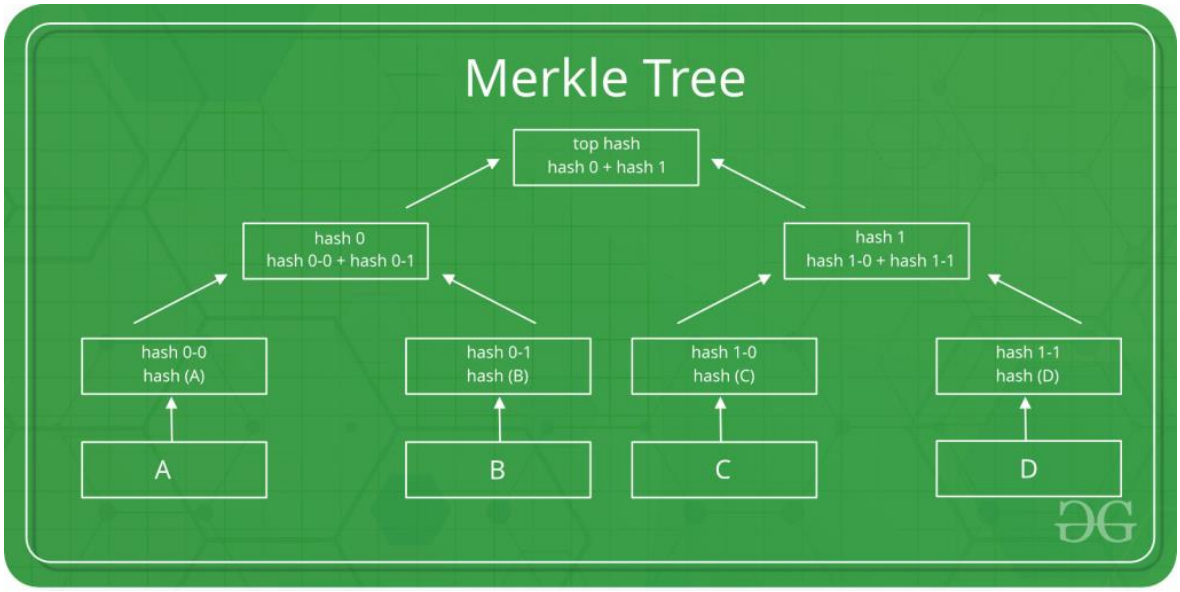

Figure 7 Merkle Tree

- Lý thuyết: Merkle Tree là cấu trúc cây nơi mỗi nút lá chứa mã băm của dữ liệu, và các nút cha chứa mã băm của các nút con. Điều này giúp nhanh chóng phát hiện thay đổi dữ liệu.
- Ứng dụng: Xác định file thay đổi: Chỉ các file thay đổi sẽ được xử lý trong pipeline, giúp tiết kiệm thời gian build.

2.2.2 Thuật toán trong CI/CD Pipelines (Azure Pipelines)

- Dependency Resolution:

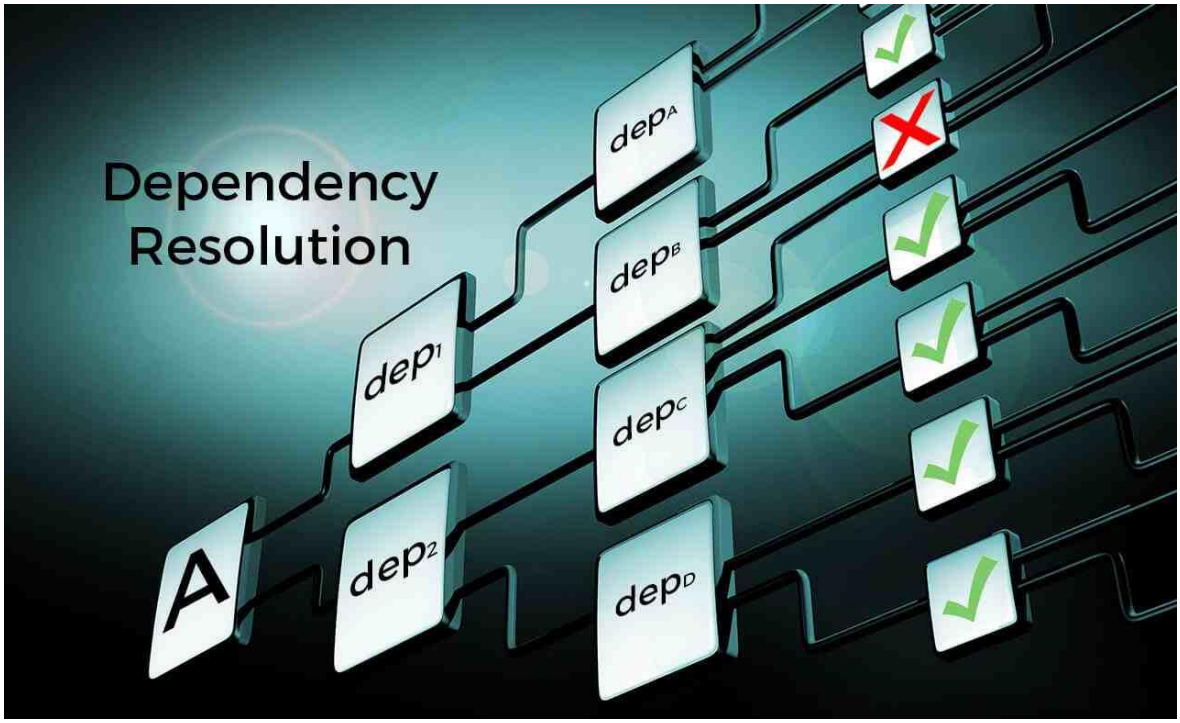

Figure 8 Dependency resolution

- Lý thuyết: Dependency resolution là quá trình xác định, sắp xếp, và xử lý các thư viện, module, hoặc thành phần phụ thuộc cần thiết để một ứng dụng hoạt động chính xác.. Thuật toán này thường sử dụng Directed Acyclic Graph (DAG) để phân tích các phụ thuộc giữa các module và thư viện. Topological Sorting là thuật toán phổ biến được sử dụng cho bài toán này, áp dụng cho đồ thị có hướng không chu trình (DAG - Directed Acyclic Graph).
- Ứng dụng:
  - Trong giai đoạn build, các công cụ như NuGet hoặc npm sẽ phân tích file cấu hình (package.json, .csproj) để tải đúng phiên bản thư viện cần thiết và trong quá trình tải các thư viên hoặc packages, thuật toán sẽ xác định thứ tự tải các thư viên.
  - Nếu có xung đột giữa các phiên bản, hệ thống sẽ tìm cách giải quyết hoặc thông báo lỗi.

2.2.3 Thuật toán trong Azure Monitor & Application Insights

- Anomaly Detection:
    - Lý thuyết: Sử dụng các thuật toán học máy (machine learning) để phát hiện các bất thường trong dữ liệu telemet.
    - Ứng dụng: Azure Application Insights giám sát và phân tích hiệu suất ứng dụng, phát hiện các sự kiện bất thường như tăng đột biến độ trễ hoặc lỗi, và gửi cảnh báo tự động.

2.3 Các dịch vụ sử dụng cloud

- Azure Repos:

Figure 9 Azure Repos

- **Mô tả**: Azure Repos là một dịch vụ quản lý mã nguồn từ Microsoft Azure, cung cấp các kho Git (hoặc TFVC) để lưu trữ và quản lý mã

nguồn trong các dự án phần mềm. Azure Repos cho phép lập trình viên lưu trữ mã nguồn, thực hiện các thay đổi, quản lý lịch sử và cấu trúc phiên bản mã nguồn.

- **Lý thuyết**: Git là một hệ thống quản lý mã nguồn phân tán, nghĩa là mỗi lập trình viên có một bản sao của kho lưu trữ mã nguồn trên máy tính của mình. Khi lập trình viên thực hiện thay đổi, họ có thể commit các thay đổi này vào kho local trước khi đẩy (push) chúng lên kho lưu trữ từ xa (remote repository). Quá trình này giúp theo dõi được lịch sử thay đổi mã nguồn và hỗ trợ nhiều lập trình viên làm việc đồng thời trên cùng một dự án mà không gặp phải xung đột.
- **Ứng dụng bên ngoài**: Azure Repos có thể được sử dụng trong các dự án phần mềm để lưu trữ và quản lý mã nguồn trong các công ty hoặc nhóm phát triển phần mềm. Các công ty có thể tổ chức và theo dõi lịch sử thay đổi mã nguồn, dễ dàng hợp nhất các nhánh (branches), và kiểm tra mã nguồn bằng các pull requests.
- **Ứng dụng trong bài tập lớn**: Trong bài tập DevOps này, **Azure Repos** là nơi lưu trữ mã nguồn của ứng dụng. Mỗi khi có sự thay đổi trong mã nguồn (ví dụ, khi lập trình viên commit thay đổi mới vào repository), nó sẽ kích hoạt một trigger để bắt đầu quy trình build và deploy tự động trong pipeline. Bằng cách này, mọi thay đổi đều được ghi lại và quản lý, đảm bảo rằng quy trình phát triển phần mềm luôn được kiểm soát và đồng bộ.

- Azure Pipelines:

Figure 10 Azure Pipelines

- **Mô tả:** Azure Pipelines là dịch vụ CI/CD (Continuous Integration / Continuous Delivery) của Azure, cho phép tự động hoá quy trình xây dựng (build), kiểm tra (test) và triển khai (deploy) ứng dụng. Nó hỗ trợ nhiều ngôn ngữ và nền tảng khác nhau, bao gồm cả .NET, Java, Node.js, …
- **Lý thuyết:** CI/CD là một phương pháp phát triển phần mềm nhằm đảm bảo rằng mã nguồn được tích hợp và kiểm tra liên tục, từ đó giúp giảm thiểu lỗi và cải thiện hiệu suất. Với CI, mọi thay đổi mã nguồn sẽ được tự động kiểm tra, build và kiểm tra tính đúng đắn. Còn với CD, ứng dụng được tự động triển khai lên môi trường sản xuất sau khi tất cả các bước kiểm tra thành công.
- **Ứng dụng bên ngoài:** Azure Pipelines có thể được sử dụng trong các công ty phát triển phần mềm để tự động hoá các công việc xây dựng và triển khai ứng dụng. Nó có thể hỗ trợ từ việc build ứng dụng, triển khai lên server, đến kiểm tra chất lượng của mã nguồn.

- **Ứng dụng trong bài tập lớn:** Trong pipeline DevOps của bài tập này, Azure Pipelines đóng vai trò quan trọng trong việc tự động hoá quy trình. Khi mã nguồn được cập nhật trong Azure Repos, Azure Pipelines sẽ tự động bắt đầu các bước build và test ứng dụng. Sau khi kiểm tra thành công, ứng dụng sẽ được triển khai lên môi trường đám mây (thông qua Azure Kubernetes Service hoặc các dịch vụ khác).

- Azure Container Registry (ACR)

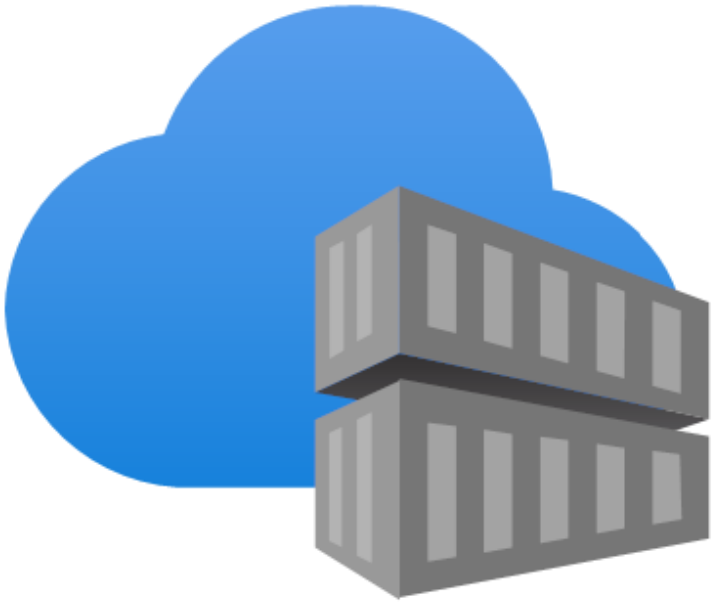

Figure 11 Azure Container Registry (ACR)

- **Mô tả:** Azure Container Registry (ACR) là dịch vụ của Azure để lưu trữ và quản lý các Docker images trong đám mây. Nó giúp bạn dễ dàng lưu trữ, phân phối và quản lý các container images cho các ứng dụng sử dụng Docker.
- **Lý thuyết**: Docker là một nền tảng cho việc phát triển, vận chuyển và chạy các ứng dụng trong các container. Container là các môi trường ảo hoá nhẹ, giúp ứng dụng chạy trên bất kỳ môi trường nào mà không gặp phải sự khác biệt về hệ điều hành hoặc cấu hình. ACR cho phép bạn lưu trữ các Docker images mà bạn có thể pull từ đó để chạy trên các môi trường như Azure Kubernetes Service (AKS)

- **Ứng dụng bên ngoài:** ACR được sử dụng để lưu trữ các Docker images của ứng dụng, giúp các đội phát triển và triển khai ứng dụng dễ dàng quản lý các images mà không phải tự quản lý máy chủ Docker hoặc cấu hình phức tạp.
- **Ứng dụng trong bài tập lớn:** Trong bài tập này, ACR sẽ được sử dụng để lưu trữ các Docker images mà Azure Pipelines tạo ra từ mã nguồn của ứng dụng. Sau khi các image được build trong pipeline, chúng sẽ được đẩy lên ACR. Khi cần triển khai lên môi trường đám mây, các image này sẽ được kéo (pull) từ ACR và triển khai trên Azure Kubernetes Service.

- Azure Kubernetes Service (AKS)

Figure 12 Azure Kubernetes Service (AKS)

- **Mô tả:** Azure Kubernetes Service (AKS) là dịch vụ container orchestration của Azure, sử dụng Kubernetes để quản lý, triển khai và tự động scale các container ứng dụng. AKS giúp bạn quản lý và vận hành các container trên môi trường đám mây một cách dễ dàng.
- **Lý thuyết:** Kubernetes là một nền tảng mã nguồn mở giúp tự động hoá việc triển khai, mở rộng và quản lý các container ứng dụng. AKS cung cấp một dịch vụ Kubernetes được quản lý sẵn, giúp giảm thiểu sự phức tạp khi tự vận hành Kubernetes trên các máy chủ của bạn.
- **Ứng dụng bên ngoài:** AKS có thể được sử dụng trong các tổ chức lớn để quản lý các ứng dụng container hoá trên đám mây. Các ứng dụng có

thể được tự động scale lên hoặc xuống tuỳ vào lưu lượng truy cập và tài nguyên hệ thống.

- **Ứng dụng trong bài tập lớn:** Trong bài tập này, AKS sẽ là môi trường để triển khai các container ứng dụng sau khi chúng được build và lưu trữ trong ACR. Khi các Docker images sẵn sàng, AKS sẽ giúp triển khai chúng lên các pod trong cluster Kubernetes, tự động scale ứng dụng tuỳ theo nhu cầu.

- Azure Monitor & Application Insights:

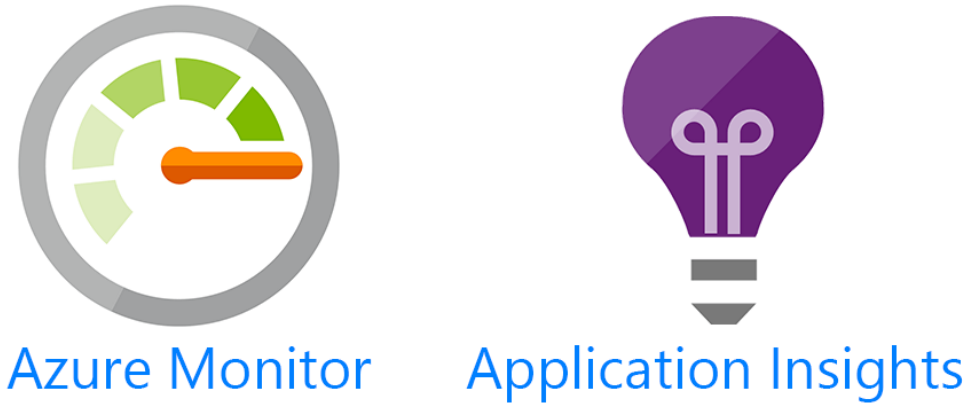

Figure 13 Azure Monitor and Application Insights

- **Mô tả:** Azure Monitor là dịch vụ giám sát toàn diện từ Azure, giúp bạn theo dõi và phân tích các tài nguyên và ứng dụng. Application Insights là một phần của Azure Monitor, giúp giám sát và phân tích hiệu suất của ứng dụng.
- **Lý thuyết:** Azure Monitor thu thập dữ liệu về hiệu suất, trạng thái và hoạt động của các dịch vụ trong Azure. Application Insights bổ sung thêm khả năng giám sát ứng dụng, từ việc ghi lại lỗi, theo dõi thời gian phản hồi của ứng dụng đến phân tích cách thức người dùng tương tác với ứng dụng.

- **Ứng dụng bên ngoài:** Các công ty sử dụng Azure Monitor và Application Insights để đảm bảo rằng các ứng dụng của họ hoạt động hiệu quả, tìm và khắc phục lỗi nhanh chóng, và tối ưu hóa trải nghiệm người dùng.
- **Ứng dụng trong bài tập lớn:** Sau khi ứng dụng được triển khai lên AKS, Azure Monitor và Application Insights sẽ giúp giám sát hiệu suất của ứng dụng và theo dõi các vấn đề tiềm ẩn trong suốt quá trình hoạt động của ứng dụng. Chúng cũng cung cấp các thông báo khi có lỗi xảy ra hoặc hiệu suất ứng dụng giảm.

- Azure Key Vault:

Figure 14 Azure Key Vault

- **Mô tả:** Azure Key Vault là dịch vụ của Microsoft Azure dùng để lưu trữ và quản lý các thông tin bảo mật như bí mật (secrets), khóa (keys) và chứng chỉ (certificates). Key Vault giúp bảo vệ các thông tin nhạy cảm và cung cấp cơ chế quản lý truy cập chặt chẽ đối với các tài nguyên này.
- **Lý thuyết:** Key Vault được thiết kế để giúp bảo vệ dữ liệu nhạy cảm khỏi việc truy cập trái phép. Nó hỗ trợ lưu trữ và truy cập các thông tin

như mật khẩu, chuỗi kết nối cơ sở dữ liệu, khóa API và chứng chỉ. Key Vault sử dụng các chính sách bảo mật nghiêm ngặt để đảm bảo chỉ những người hoặc ứng dụng có quyền mới có thể truy cập thông tin bảo mật này. Các khóa và bí mật được mã hóa và có thể được quản lý và truy xuất qua API.

- **Secrets**: Lưu trữ thông tin nhạy cảm như mật khẩu, chuỗi kết nối cơ sở dữ liệu, hoặc các giá trị cấu hình khác.
- **Keys**: Lưu trữ các khóa mã hóa để bảo vệ dữ liệu hoặc các hoạt động bảo mật khác.
- **Certificates**: Lưu trữ và quản lý các chứng chỉ SSL/TLS, giúp bảo vệ các kết nối an toàn.

- **Ứng dụng bên ngoài:**
  - Các công ty sử dụng Azure Key Vault để bảo vệ thông tin nhạy cảm trong môi trường đám mây. Thông qua Key Vault, bạn có thể dễ dàng quản lý các khóa bảo mật, mật khẩu và chứng chỉ mà không cần phải lưu trữ chúng trong mã nguồn hoặc cơ sở dữ liệu, giúp tăng cường bảo mật cho ứng dụng và hạ tầng.
  - Các ứng dụng trong thực tế như ứng dụng web hoặc dịch vụ API có thể sử dụng Key Vault để lấy thông tin bảo mật cần thiết trong suốt quá trình vận hành, chẳng hạn như chuỗi kết nối cơ sở dữ liệu hoặc API keys.
- **Ứng dụng trong bài tập lớn:**
  - Trong pipeline DevOps này, Azure Key Vault sẽ được sử dụng để bảo vệ các thông tin bảo mật trong quá trình triển khai và vận hành ứng dụng trên cloud. Ví dụ, các thông tin nhạy cảm như chứng chỉ SSL dùng cho giao tiếp an toàn, mật khẩu cơ sở dữ liệu, hoặc API keys có thể được lưu trữ trong Key Vault. Khi ứng dụng triển khai trên Azure Kubernetes Service (AKS) hoặc Azure Web Apps, nó có thể lấy các thông tin bảo mật này một

cách an toàn từ Key Vault mà không cần phải lưu trữ chúng trong mã nguồn.

- Các tác vụ trong pipeline như xây dựng (build), kiểm tra (test), và triển khai (deploy) sẽ cần các bí mật này để kết nối với các dịch vụ bên ngoài hoặc truy cập vào cơ sở dữ liệu. Việc sử dụng **Azure Key Vault** giúp bảo vệ các thông tin này khỏi các lỗ hổng bảo mật hoặc rủi ro từ việc lưu trữ thông tin bảo mật không an toàn.
- Ví dụ, khi **Azure Pipelines** triển khai ứng dụng, nó sẽ lấy chuỗi kết nối cơ sở dữ liệu hoặc các khóa API từ **Azure Key Vault** thay vì cứng mã chúng trong mã nguồn hoặc trong các file cấu hình. Điều này giúp giảm thiểu rủi ro bảo mật trong quá trình triển khai và hoạt động của ứng dụng trên môi trường cloud.

CHƯƠNG 3: MÔ HÌNH DỮ LIỆU

3.1 Tốc độ cho phép đọc ghi

- Frontend:

OPERATION NAME	DURATION (AVG)	COUNT	PIN
Overall	**151 ms**	**128**	
GET /_next/static/media/SF-Pro-Display-Bold.f0cb6ea8.otf	598 ms	1	
GET /robots933456.txt	517 ms	1	
GET /_next/static/media/SF-Pro-Display-Light.4b4166b9.otf	381 ms	1	
GET /_next/static/media/SF-Pro-Display-Semibold.f868526c.otf	380 ms	1	
GET /_next/static/media/SF-Pro-Display-Regular.b5c65667.otf	326 ms	1	
GET /images/product_images/pccase-review.png	308 ms	1	
GET /_next/static/media/SF-Pro-Display-Medium.f74d27cd.otf	250 ms	1	
GET /images/product_images/LocationPin_01.png	230 ms	1	
GET /_next/image	191 ms	75	
GET /images/product_images/Laptop_01.png	175 ms	1	

Figure 15 Tốc độ đọc ghi trên FE

- Backend: Trung bình ~3000ms cho cả đọc và ghi

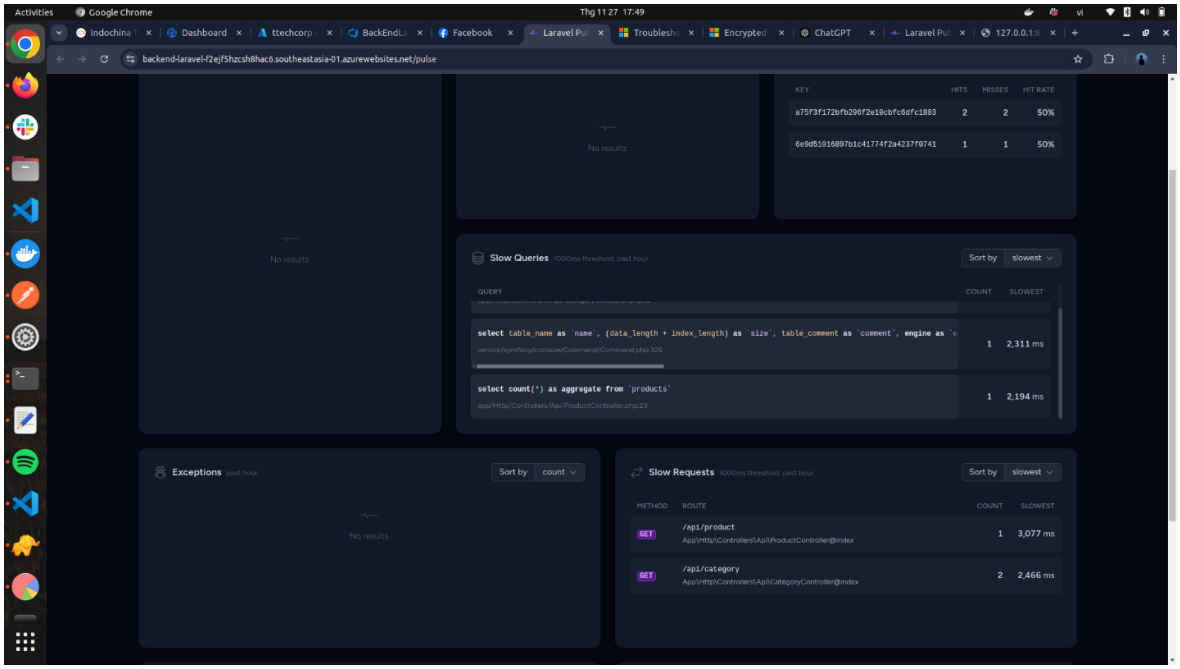

Figure 16 Tốc độ đọc ghi dưới BE(1)

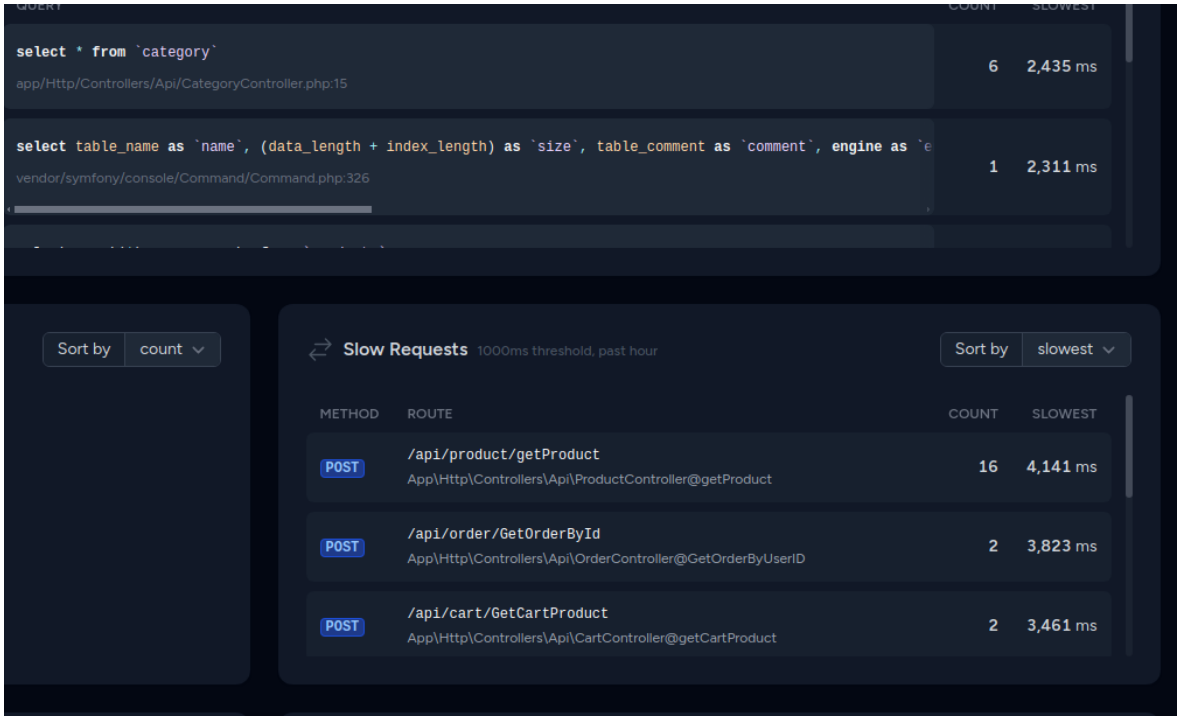

Figure 17 Tốc độ đọc ghi dưới BE(2)

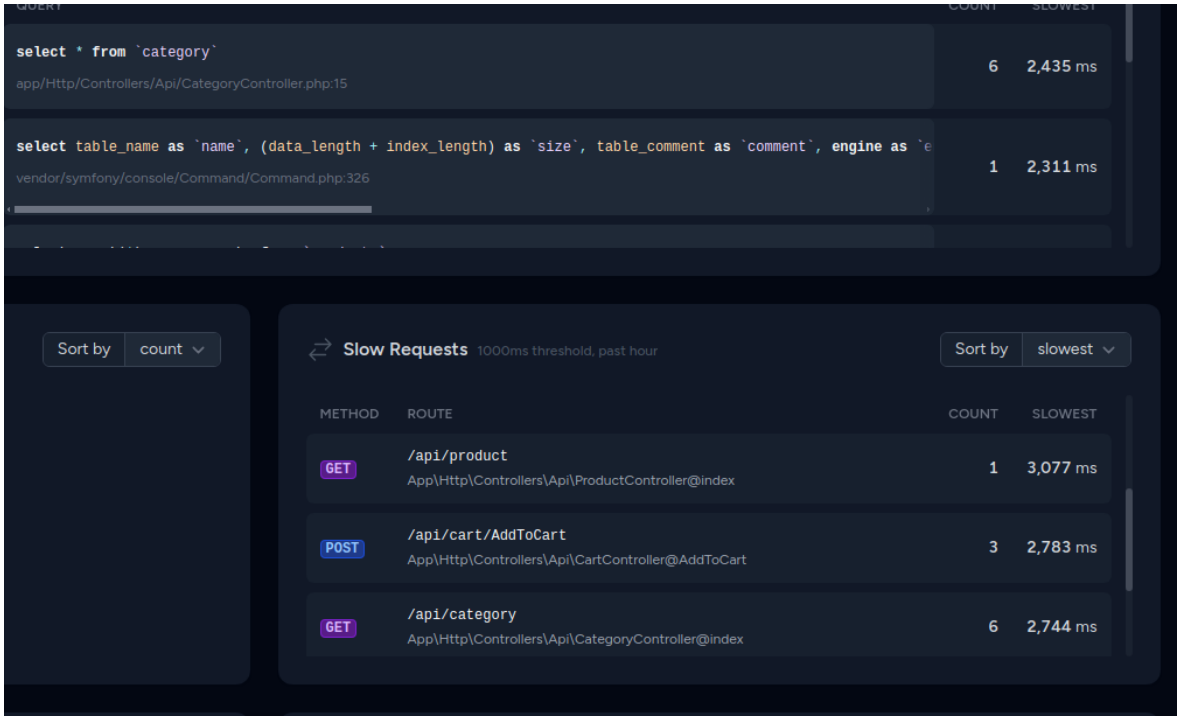

Figure 18 Tốc độ đọc ghi dưới BE (3)

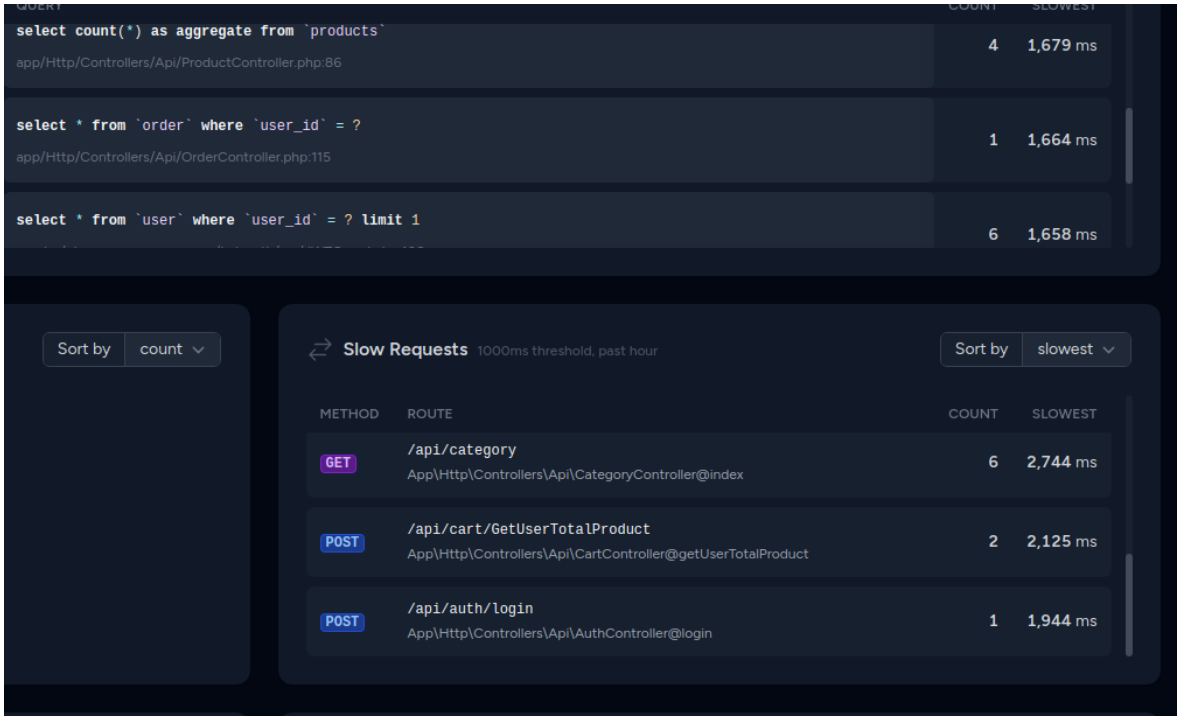

Figure 19 Tốc độ đọc ghi dưới BE (4)

3.2 Luồng xử lý dữ liệu tự động ETL

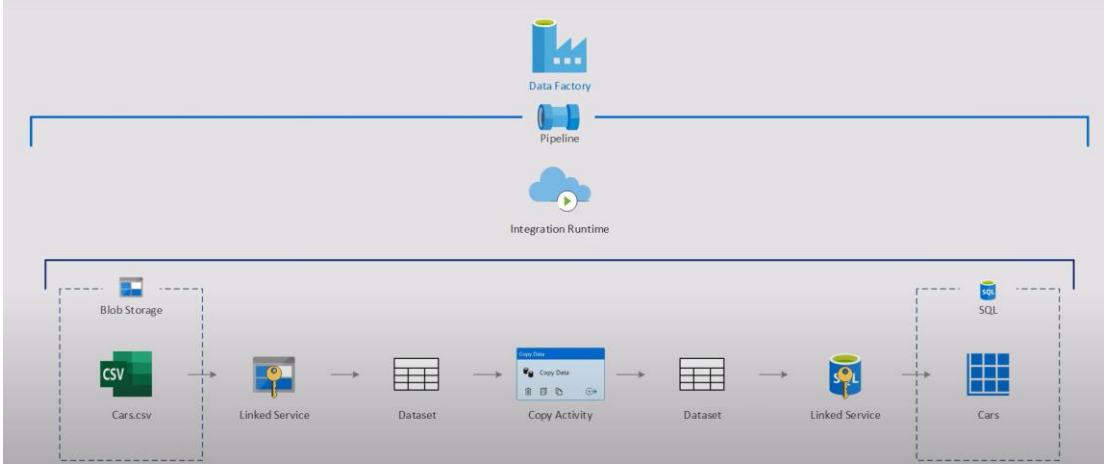

Figure 20 Luồng dữ liệu ETL

- **Data Factory**:
  - Đây là dịch vụ chính trên Azure dùng để xây dựng, quản lý và tự động hóa các quy trình ETL/ELT. Nó điều phối luồng dữ liệu giữa các nguồn dữ liệu và đích.
- **Pipeline**:
  - Là một tập hợp các bước công việc (activities) để xử lý và di chuyển dữ liệu. Trong sơ đồ, pipeline chứa các tác vụ ETL.
- **Integration Runtime**:
  - Đây là thành phần tính toán trong Azure Data Factory, chịu trách nhiệm thực thi các hoạt động dữ liệu (ví dụ: sao chép dữ liệu) giữa nguồn và đích. Nó đảm bảo kết nối giữa các thành phần.
- **Blob Storage**:
  - Là nguồn dữ liệu gốc trong sơ đồ, lưu trữ file **Cars.csv**. Đây là nơi chứa dữ liệu thô được trích xuất (Extract).
- **Cars.csv**:

- Tệp dữ liệu đầu vào, chứa thông tin liên quan đến các bản ghi "Cars". Dữ liệu trong file này sẽ được trích xuất.

- **Linked Service (Blob Storage)**:
  - Một kết nối được cấu hình trước để truy cập dữ liệu từ Blob Storage. Nó xác định thông tin xác thực và các thiết lập cần thiết để truy cập dữ liệu trong Blob Storage.
- **Dataset (Blob)**:
  - Định nghĩa cấu trúc dữ liệu của tệp Cars.csv. Dataset chỉ định thông tin về cách trích xuất dữ liệu từ nguồn.
- **Copy Activity**:
  - Tác vụ chính trong pipeline để sao chép dữ liệu từ nguồn (Blob Storage) sang đích (SQL). Đây là bước xử lý chính trong quy trình ETL.
- **Dataset (SQL)**:
  - Định nghĩa cấu trúc dữ liệu đích trong cơ sở dữ liệu SQL. Nó xác định thông tin về cách dữ liệu sẽ được lưu trữ.
- **Linked Service (SQL)**:
  - Kết nối cấu hình trước để truy cập cơ sở dữ liệu SQL. Tương tự như Linked Service cho Blob Storage, nó định nghĩa thông tin xác thực và các thiết lập cần thiết.
- **SQL**:
  - Đích của dữ liệu sau quá trình ETL. Dữ liệu từ tệp Cars.csv sẽ được tải (Load) vào bảng SQL.
- **Cars (SQL Table)**:

- Bảng đích trong cơ sở dữ liệu SQL, nơi lưu trữ dữ liệu đã được sao chép từ tệp Cars.csv.g

3.3 Các giải pháp thực hiện tối ưu hoá:

- **Lựa chọn cơ sở dữ liệu lưu trữ chính:**
  - Chọn Azure SQL Database (quan hệ):
    - Phù hợp với dữ liệu có cấu trúc như sản phẩm, danh mục, thuộc tính (màu sắc, kích thước).
    - Hỗ trợ truy vấn phức tạp, như tìm kiếm sản phẩm theo nhiều điều kiện (giá, đánh giá, v.v.).
    - Tính năng gợi ý:
      - **Indexing:** Tạo chỉ mục cho các cột tìm kiếm phổ biến (tên sản phẩm, giá).
      - **Partitioning:** Phân vùng dữ liệu theo danh mục hoặc khu vực.
- **Cải thiện tốc độ đọc/ghi dữ liệu:**
  - **Sử dụng giải pháp bộ nhớ đệm (Caching):**
    - **Azure Cache for Redis**: Cải thiện tốc độ truy cập dữ liệu bằng cách lưu trữ tạm thời các bản sao dữ liệu trong bộ nhớ đệm.
    - **Azure Content Delivery Network (CDN):** Tăng tốc phân phối nội dung tĩnh như hình ảnh, video đến người dùng cuối.
  - **Tăng hiệu năng truy cập dữ liệu:**
    - **Azure Premium Storage:** Sử dụng ổ SSD cho các ứng dụng yêu cầu độ trễ thấp và hiệu suất cao (IOPS).
    - **Azure Ultra Disk Storage:** Cung cấp IOPS cực cao và độ trễ thấp cho các ứng dụng quan trọng như cơ sở dữ liệu OLTP.

- **Read/Write Optimization:** Sử dụng giao thức truy cập dữ liệu hiệu quả như **ADLS Gen2 REST API** hoặc **Azure Data Factory IR Integration**.

CHƯƠNG 4: HIỆN THỰC ĐỀ TÀI

4.1 Hình ảnh trang web

- Trang chủ:

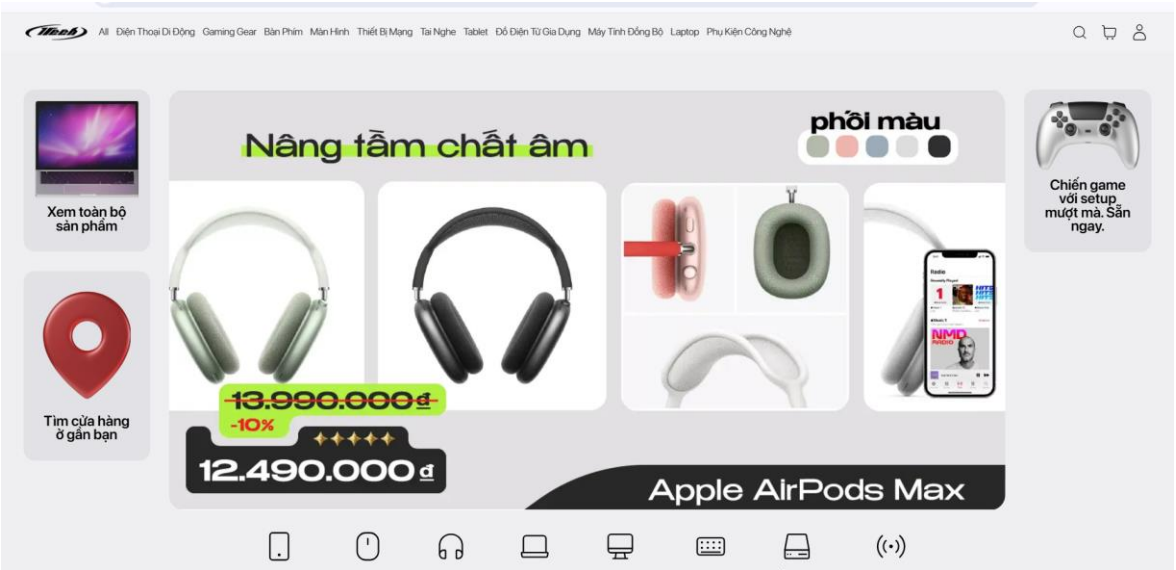

Figure 21 Giao diện trang web

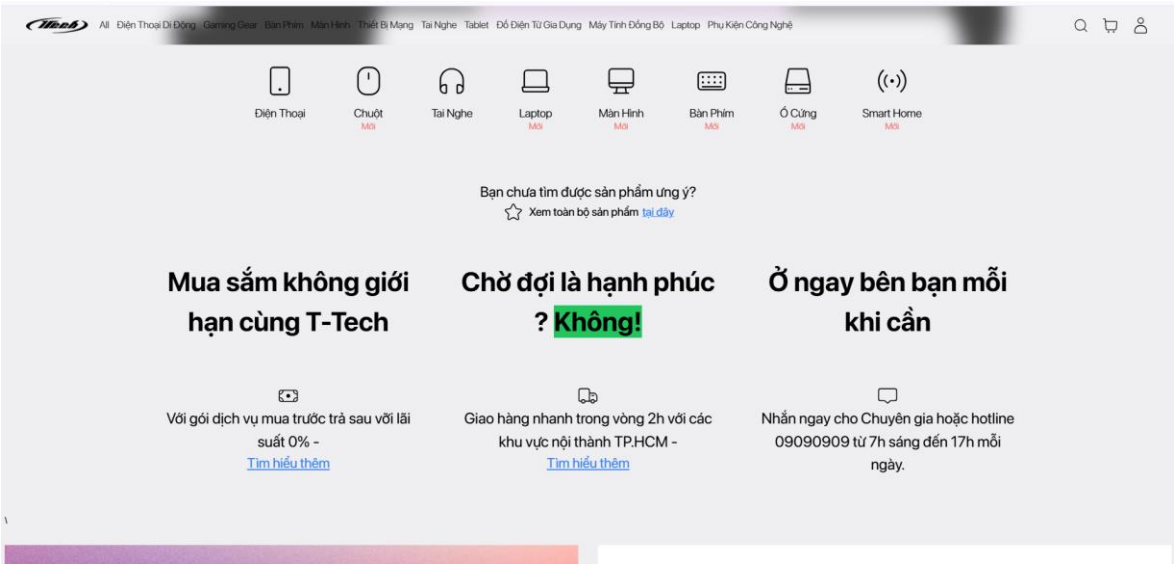

Figure 22 Giao diện trang web

- Trang chi tiết user:

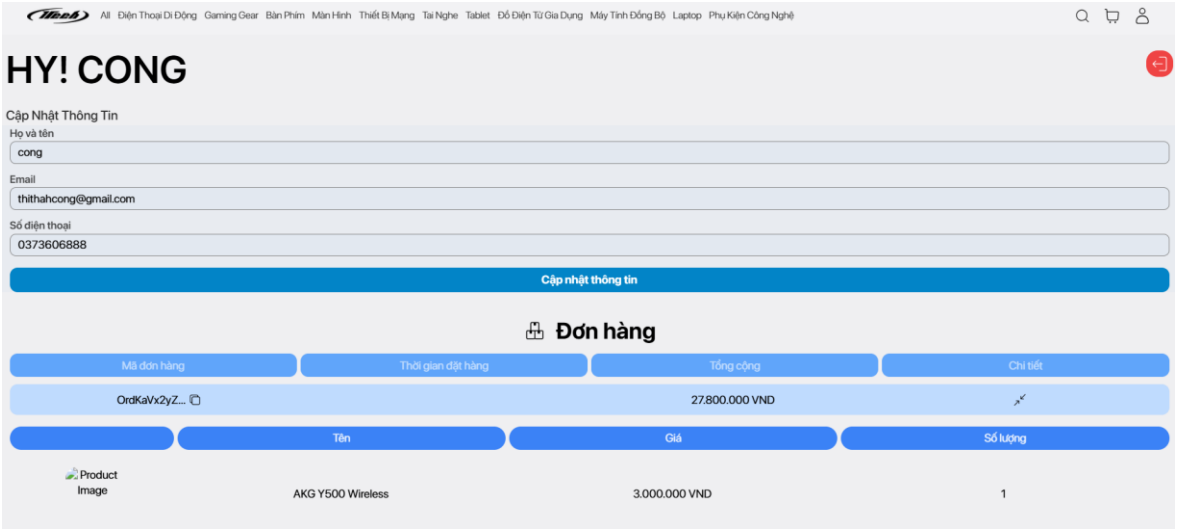

Figure 23 Giao diện trang user

- Trang danh sách sản phẩm để thêm vào giỏ hàng:

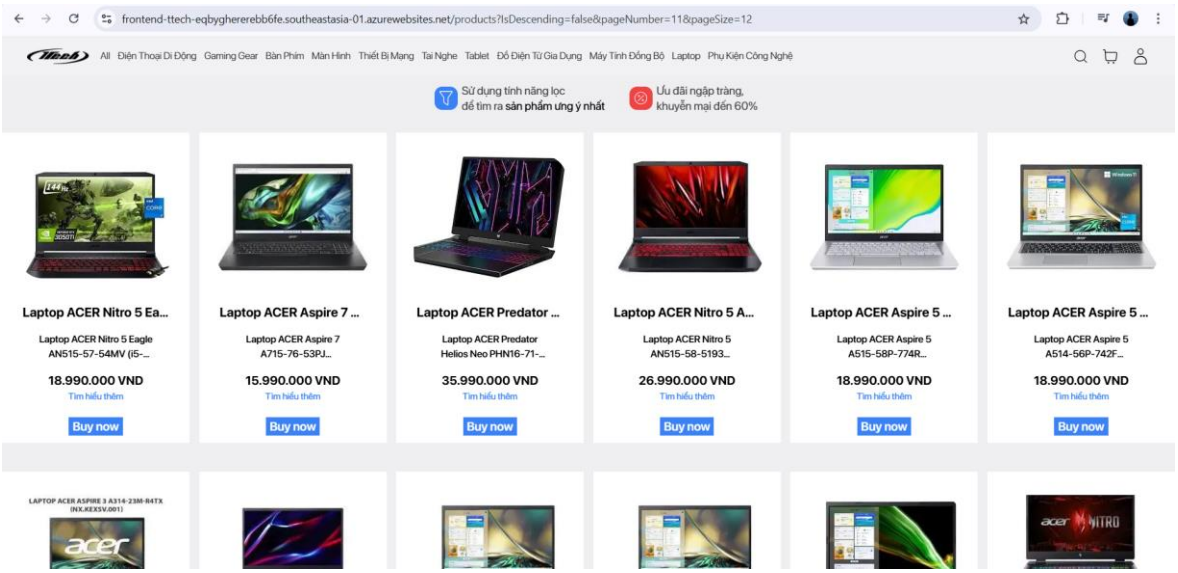

Figure 24 Giao diện trang sản phẩm

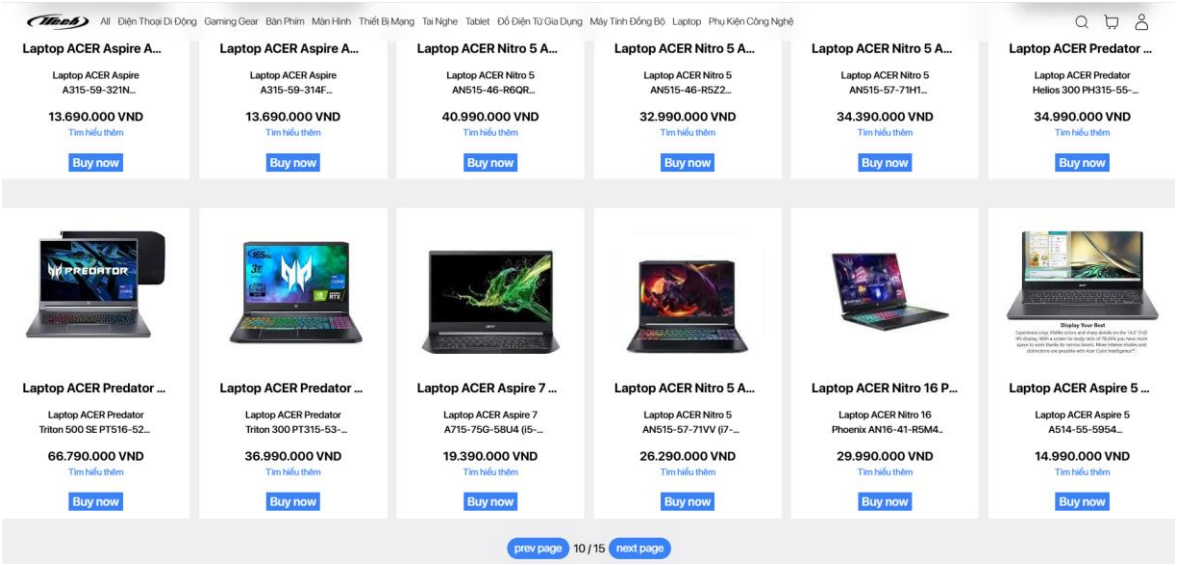

Figure 25 Giao diện trang sản phẩm

- Trang giỏ hàng:

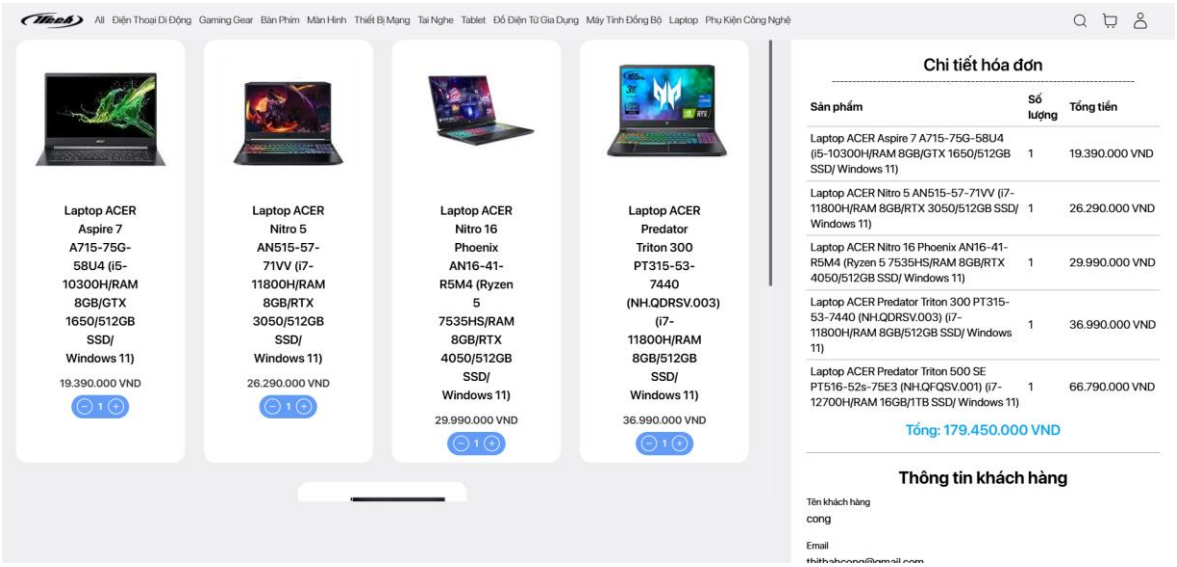

Figure 26 Giao diện trang giỏ hàng

4.2 Azure function

- Trigger report daily cho admin qua email

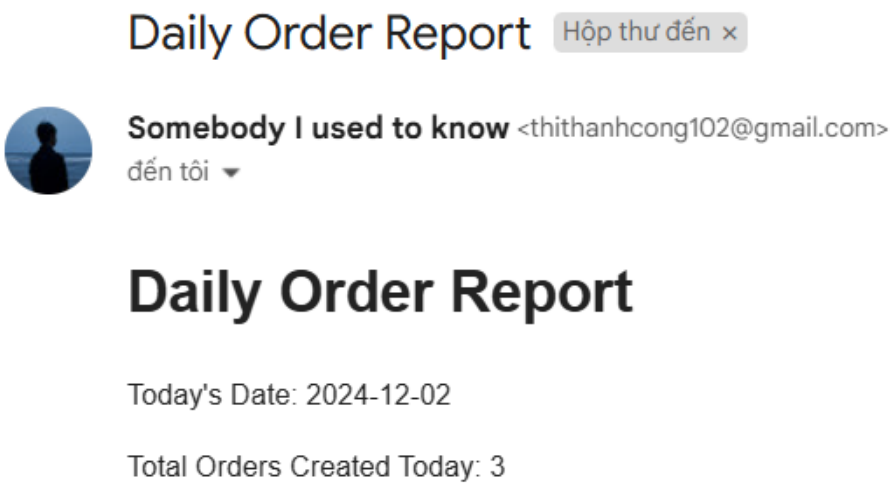

Figure 27 Email trigger report

4.3 Hệ thống blue/green:

- Hệ thống **Blue/Green Deployment** là một chiến lược triển khai giúp giảm thiểu thời gian downtime và rủi ro trong việc phát hành ứng dụng hoặc tính năng mới. Với website e-commerce, nơi tính khả dụng (availability) và trải nghiệm người dùng (UX) là rất quan trọng, cách triển khai này đảm bảo rằng việc nâng cấp hệ thống không ảnh hưởng đến trải nghiệm khách hàng.
- Cách hoạt động Blue/Green Deployment
    - **Blue environment (Hiện tại):** Đây là môi trường đang hoạt động và phục vụ người dùng. Đây có thể là phiên bản ổn định hiện tại của trang e-commerce.
    - **Green environment (Mới):** Đây là môi trường chứa phiên bản mới (ví dụ: code, tính năng mới hoặc cập nhật), được chuẩn bị để thay thế môi trường Blue.
    - **Quy trình:**
        - **Triển khai phiên bản mới** (Green) trên một môi trường độc lập.

- **Kiểm tra và xác thực** môi trường Green (qua testing và staging).
- **Chuyển hướng traffic** từ Blue sang Green khi Green được xác nhận là hoạt động đúng.
- **Duy trì khả năng rollback**, cho phép quay lại Blue nếu có vấn đề xảy ra.

- Môi trường **blue/green** trên azure:

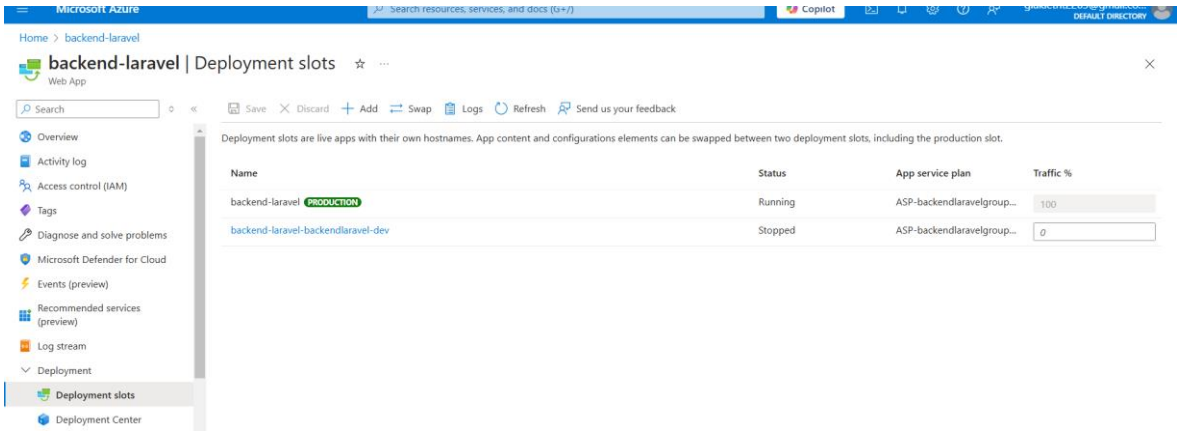

Figure 28 Môi trường blue/green

- **Triển khai Blue/Green Deployment bằng Azure Web App Deployment Slots:**

  Azure Web App Deployment Slots là một tính năng mạnh mẽ cho phép bạn tạo nhiều "slot" (môi trường) trên một ứng dụng Azure App Service. Tính năng này rất phù hợp để triển khai hệ thống Blue/Green Deployment, giúp giảm downtime và rủi ro khi phát hành ứng dụng mới.

- **Khái niệm về Deployment Slots**
  - **Production Slot (Blue):** Đây là slot mặc định, đang chạy phiên bản hiện tại của ứng dụng và phục vụ người dùng.
  - **Staging Slot (Green):** Đây là slot khác, nơi triển khai phiên bản mới để thử nghiệm và kiểm tra trước khi chuyển sang môi trường production.
  - **Restores:** đây là các slot đặc biệt nếu restore trang web bằng backup, nó hoạt động như trang web lúc tạo backup.

NGUỒN THAM KHẢO

[1] Docker, "Docker," [Online]. Available: https://www.docker.com/.

[2] Microsoft, "Azure documentation," Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/azure/.

[3] GeeksforGeeks, "SHA-1 Hash," 18 07 2024. [Online]. Available: https://www.geeksforgeeks.org/sha-1-hash-in-java/.

[4] GeeksforGeeks, "Introduction to Merkle Tree," [Online]. Available: https://www.geeksforgeeks.org/introduction-to-merkle-tree/. [Accessed 03 11 2024].

[5] Gradle, "Dependency Resolution," [Online]. Available: https://docs.gradle.org/current/userguide/dependency_resolution.html. [Accessed 03 11 2024].