TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO GIỮA KÌ MÔN KIẾN TRÚC HƯỚNG DỊCH VỤ**

*Người thực hiện:* **TRẦN QUỐC VINH - 52000823**

Lớp: **200503401**

Khóa: **24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO GIỮA KÌ MÔN KIẾN TRÚC HƯỚNG DỊCH VỤ**

*Người thực hiện:* **TRẦN QUỐC VINH - 52000823**

Lớp: **200503401**

Khóa: **24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

*TP. Hồ Chí Minh, ngày 31 tháng 3 năm 2024*

*Tác giả*

*Trần Quốc Vinh*

BÁO CÁO ĐƯỢC HOÀN THÀNH

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Khóa luận/Đồ án tốt nghiệp còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Báo cáo cuối kỳ của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 31 tháng 3 năm 2024*

*Tác giả*

*Trần Quốc Vinh*

**TÓM TẮT**

# **MỤC LỤC**

[**TÓM TẮT** 5](#_Toc163289209)

[**MỤC LỤC** 6](#_Toc163289210)

[**DANH MỤC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ** 7](#_Toc163289211)

[1 CHƯƠNG 1: TỔNG QUAN HỆ THỐNG 8](#_Toc163289212)

[**1.1** Giới thiệu đề tài 8](#_Toc163289213)

[**1.1.1** Tổng quan về kiến trúc microservice 8](#_Toc163289214)

[**1.1.2** Hệ thống Ibanking sử dụng microservice 8](#_Toc163289215)

[**1.2** Đặc tả hệ thống 9](#_Toc163289216)

[**1.3** Phạm vi đề tài 9](#_Toc163289217)

[**1.3.1** Giới hạn về đối tượng và chức năng 9](#_Toc163289218)

[**1.3.2** Giới hạn về công nghệ 9](#_Toc163289219)

[**1.4** Ý nghĩa thực tiễn 9](#_Toc163289220)

[**2** CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 10](#_Toc163289221)

[**2.1** Yêu cầu chức năng của hệ thống 10](#_Toc163289222)

[**2.2** Yêu cầu phi chức năng của hệ thống 10](#_Toc163289223)

[**2.3** Lược đồ Use Case của hệ thống 10](#_Toc163289224)

[**2.3.1** Lược đồ Use Case 10](#_Toc163289225)

[2.3.2 Đặc tả Use Case 11](#_Toc163289226)

[2.4 Lược đồ ER của hệ thống 14](#_Toc163289227)

[3 CHƯƠNG 3: HIỆN THỰC HỆ THỐNG 14](#_Toc163289228)

[3.1 Tổng quan về hệ thống 14](#_Toc163289229)

[3.2 Giao diện hệ thống 16](#_Toc163289230)

[3.3 API của hệ thống 17](#_Toc163289231)

[**TÀI LIỆU THAM KHẢO** 19](#_Toc163289232)

**DANH MỤC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

[Hình 1 1: Building Microservices" của Sam Newman 8](#_Toc163062393)

[Hình 2 1: Lược đồ ER hiện thực ở mức vật lý 14](#_Toc163062404)

[Hình 3. 1: Danh sách microservice 15](#_Toc163062510)

[Hình 3. 2 Frontend microservice 15](#_Toc163062511)

[Hình 3. 3: Backend microservice 15](#_Toc163062512)

[Hình 3. 4 UI đăng nhập 15](#_Toc163062513)

[Hình 3. 5 UI trang thanh toán học phí 16](#_Toc163062514)

[Hình 3. 6 UI lịch sử đóng học phí 16](#_Toc163062515)

[Hình 3. 7 Danh sách API 17](#_Toc163062516)

[Bảng 2. 1: Yêu cầu chức năng của hệ thống 10](#_Toc163062470)

[Bảng 2. 2: Yêu cầu phi chức năng của hệ thống 10](#_Toc163062471)

[Bảng 2. 3: Use case Check Account Information 11](#_Toc163062472)

[Bảng 2. 4: Use case Check Newest Tuition Fee 12](#_Toc163062473)

[Bảng 2. 5: Use case Pay Tuition Fee 13](#_Toc163062474)

[Bảng 2. 6: Use case Check Tuition Fee Transaction History 14](#_Toc163062475)

[Bảng 3. 1 Danh sách API của Student Management Microservice 18](#_Toc163289199)

[Bảng 3. 2 Danh sách API của Tuition Fee Payment Microservice 18](#_Toc163289200)

# CHƯƠNG 1: TỔNG QUAN HỆ THỐNG

## Giới thiệu đề tài

### Tổng quan về kiến trúc microservice

Kiến trúc Microservices là xu hướng mới trong phát triển phần mềm, cho phép chia nhỏ ứng dụng thành các dịch vụ độc lập, tự quản lý. Cách tiếp cận này mang lại nhiều lợi ích, bao gồm:

* Tính linh hoạt: Dễ dàng phát triển và triển khai các tính năng mới.
* Khả năng mở rộng: Khả năng mở rộng từng dịch vụ độc lập để đáp ứng nhu cầu.
* Khả năng bảo trì: Dễ dàng xác định và sửa lỗi trong các dịch vụ riêng lẻ.
* Tính tin cậy: Khả năng chịu lỗi cao hơn do sự cô lập của các dịch vụ.

Để tìm hiểu thêm về kiến trúc Microservices, bạn có thể tham khảo cuốn sách "Building Microservices" của Sam Newman, một chuyên gia hàng đầu trong lĩnh vực này. Cuốn sách cung cấp hướng dẫn toàn diện về việc thiết kế, phát triển và triển khai các hệ thống microservices.

Với kiến trúc microservices và sự hướng dẫn của Sam Newman, bạn có thể xây dựng các ứng dụng mạnh mẽ, linh hoạt và có khả năng mở rộng cao.  
A book cover with bees on a honeycomb

Description automatically generated

Hình 1 1: Building Microservices" của Sam Newman

### Hệ thống Ibanking sử dụng microservice

Dự án của chúng tôi áp dụng kiến trúc microservices để nâng cao trải nghiệm người dùng cho cổng thông tin sinh viên. Bằng cách chia nhỏ ứng dụng thành các dịch vụ độc lập, chúng tôi hướng đến mục tiêu:

* Tăng tốc độ phát triển và triển khai các tính năng mới.
* Đảm bảo khả năng mở rộng để đáp ứng nhu cầu ngày càng tăng của sinh viên.
* Cải thiện khả năng bảo trì, Nâng cao khả năng chịu lỗi của hệ thống

## Đặc tả hệ thống

Dự án bao gồm ba microservices chính:

* **Giao diện người dùng (frontend)**: Cung cấp giao diện trực quan và dễ sử dụng cho sinh viên
* **Quản lý sinh viên (student\_management)**: Lưu trữ và quản lý thông tin sinh viên, bao gồm tài khoản và các môn học sinh viên đăng ký trong 1 kì học
* **Thanh toán học phí (tuition\_fee\_payment)**: Cung cấp phương thức thanh toán học phí an toàn và tiện lợi

## Phạm vi đề tài

### Giới hạn về đối tượng và chức năng

Do scope của dự án nhỏ nên hệ thống UI được thiết kế chỉ dành riêng cho sinh viên đang theo học tại trường. Hệ thống không có UI dành cho giảng viên hoặc admin

Ngoài các chức năng trong phần đặc tả yêu cầu chức năng phía trên, hệ thống **không cung cấp** các chức năng sau: Quản lý lớp học, Quản lý giảng viên, Quản lý tài chính,...

### Giới hạn về công nghệ

Dự án này sử dụng các công nghệ sau:

* Backend: Flask
* Database: SQLite
* Frontend: Bootstrap/HTML/CSS

Các công nghệ trên có thể dẫn tới một số hạn chế như sau:

* Khả năng mở rộng: Flask và SQLite có thể không phù hợp với các ứng dụng có lượng truy cập hoặc dữ liệu lớn
* Khả năng bảo mật: Cần cài đặt thêm các thư viện bảo mật cho Flask và SQLite.
* Hỗ trợ cộng đồng: Cộng đồng Flask và SQLite nhỏ hơn so với Django và RDBMS khác, việc tìm kiếm hỗ trợ có thể gặp khó khăn hơn.
* Khả năng tùy chỉnh: Bootstrap có thể hạn chế khả năng tùy chỉnh giao diện ứng dụng so với các framework khác như Angular hoặc React

## Ý nghĩa thực tiễn

* Qua dự án trên, tôi đã học được các best practice trong việc xây dựng API cho các hệ thống sử dụng microservice
* Ngoài ra tôi còn tìm hiểu được thêm về kiến trúc microservice, cách nó hoạt động cũng như lợi ích và hạn chế của kiến trúc này so với kiến trúc monolith truyền thống

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Yêu cầu chức năng của hệ thống

|  |  |
| --- | --- |
| **Yêu cầu chức năng** | * Hệ thống hỗ trợ sinh viên đăng nhập và đăng xuất * Hệ thống hỗ trợ sinh viên xem thông tin cá nhân như số dư trong tài khoản,... * Hệ thống hỗ trợ xem thông tin học phí mới nhất khi nhập mã số sinh viên (MSSV) * Hệ thống hỗ trợ sinh viên đóng học phí qua MSSV và có bảo mật 2 lớp, sau khi thành công sẽ có mail gửi về * Hệ thống hỗ trợ xem lịch sử giao dịch của sinh viên |

Bảng 2. 1: Yêu cầu chức năng của hệ thống

## Yêu cầu phi chức năng của hệ thống

|  |  |
| --- | --- |
| **Yêu cầu phi chức năng** | * Hiệu suất hoạt động hệ thống * Sao lưu dữ liệu * Cho phép người dùng tương tác dễ dàng * Khả năng tương thích với đa nền tảng * Khả năng bảo trì hệ thống |

Bảng 2. 2: Yêu cầu phi chức năng của hệ thống

## Lược đồ Use Case của hệ thống

### Lược đồ Use Case

A diagram of a system

Description automatically generated

### Đặc tả Use Case

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC-1 | |
| **Use Case Name** | Check Account Information | |
| **Description** | Student muốn kiểm tra thông tin tài khoản cá nhân | |
| **Trigger Event** | Student chọn vào nút Account Information | |
| **Actor(s)** | Student | |
| **Pre-condition(s)** | * Tài khoản khả dụng và sinh viên đã đăng nhập vào hệ thống * Có kết nối internet khi thực hiện đăng nhập | |
| **Post-condition(s)** | Bảng thông tin tài khoản hiện lên trên UI | |
| **Flow of event** | Actor | System |
| 1. Chọn mục Account Information tại navigation bar | 1.1 Hệ thống hiển thị modal gồm thông tin tài khoản |
| **Exception** | Hệ thống xảy ra lỗi trong quá trình xử lý dữ liệu và hiển thị thông báo lỗi | |

Bảng 2. 3: Use case Check Account Information

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC-2 | |
| **Use Case Name** | Check Newest Tuition Fee | |
| **Description** | Student muốn kiểm tra học phí của học kì mới nhất | |
| **Trigger Event** | Student điền MSSV vào ô “Student ID” | |
| **Actor(s)** | Student | |
| **Pre-condition(s)** | * Tài khoản khả dụng và sinh viên đã đăng nhập vào hệ thống * Có kết nối internet khi thực hiện đăng nhập | |
| **Post-condition(s)** | Sinh viên có thể xem số tiền cụ thể cần thanh toán cho học phí | |
| **Flow of event** | Actor | System |
| 1. Nhập MSSV vào ô Student ID trong trang Pay Tuition Fee | 1.1 UI hiển thị số tiền sinh viên cần đóng trong học kì mới nhất trong ô Tuition Fee |
| **Exception** | * Nếu MSSV không tồn tài hệ thống sẽ hiện ra lỗi * Nếu MSSV có tồn tại nhưng học phí đã được đóng hoặc không tồn tại, hệ thống sẽ hiện ra lỗi | |

Bảng 2. 4: Use case Check Newest Tuition Fee

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC-3 | |
| **Use Case Name** | Pay Tuition Fee | |
| **Description** | Student muốn thực hiện đóng học phí của học kì mới nhất | |
| **Trigger Event** | Student chọn vào ô Pay Tuition Fee | |
| **Actor(s)** | Student | |
| **Pre-condition(s)** | * Tài khoản khả dụng và sinh viên đã đăng nhập vào hệ thống * Dữ liệu về sinh viên và học phí của sinh viên đó phải tồn tại * Sinh viên phải có số dư trong tài khoản lớn hơn hoặc bằng mức học phí | |
| **Post-condition(s)** | Bảng confirmation sẽ hiện lên, nếu SV chọn Yes sẽ dẫn tới gửi xác thực mã OTP | |
| **Relationship with other Use case** | Include: Send OTP, Send success email | |
| **Flow of event** | Actor | System |
| 1. Nhập MSSV vào ô Student ID trong trang Pay Tuition Fee 2. Nhấp chọn nút Pay Tuition Fee 3. Chọn Send OTP để nhận mã OTP về email 4. Nhập mã OTP và nhấn verify, nếu đúng thì giao dịch học phí sẽ được thực hiện | 1. UI hiển thị số tiền sinh viên cần đóng trong học kì mới nhất trong ô Tuition Fee 2. Hệ thống hiện bảng Confirmation và yêu cầu nhấn Yes 3. Hệ thống gửi email OTP tới email của sinh viên 4. Hệ thống thực hiện giao dịch, thông báo thành công, gửi mail và đưa sinh viên về lại trang chính |
| **Exception** | * Nếu MSSV không tồn tài hệ thống sẽ hiện ra lỗi * Nếu MSSV có tồn tại nhưng học phí đã được đóng hoặc không tồn tại, hệ thống sẽ hiện ra lỗi * Số dư trong tài khoản không đủ, hệ thống sẽ hiện lỗi * Xác thực mã OTP sai, hệ thống sẽ hiện lỗi | |

Bảng 2. 5: Use case Pay Tuition Fee

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC-4 | |
| **Use Case Name** | Check Tuition Fee Transaction History | |
| **Description** | Student muốn kiểm tra lịch sử đóng học phí của mình | |
| **Trigger Event** | Student chọn vào ô Transaction History trên thanh nav bar | |
| **Actor(s)** | Student | |
| **Pre-condition(s)** | * Tài khoản khả dụng và sinh viên đã đăng nhập vào hệ thống | |
| **Post-condition(s)** | Bảng lịch sử giao dịch học phí hiện lên | |
| **Flow of event** | Actor | System |
| 1. Chọn vào ô Transaction History trên thanh nav bar | 1. UI hiển thị bảng lịch sử giao dịch |
| **Exception** | * Hệ thống xảy ra lỗi trong quá trình xử lý dữ liệu và hiển thị thông báo lỗi | |

Bảng 2. 6: Use case Check Tuition Fee Transaction History

## Lược đồ ER của hệ thống

A screenshot of a computer

Description automatically generated

Hình 2 1: Lược đồ ER hiện thực ở mức vật lý

# CHƯƠNG 3: HIỆN THỰC HỆ THỐNG

## Tổng quan về hệ thống

Dự án được hiện thực bằng các công nghệ sau:

* Backend: Flask
* Database: SQLite
* Frontend: Bootstrap/HTML/CSS

Tổng cộng 3 microservice trong hệ thống: Frontend, Student Management, Tuition Fee Payment

A black background with white numbers

Description automatically generated

Hình 3. 1: Danh sách microservice

Đối với Frontend: Chứ các file template html, css liên quan tới UI

A screenshot of a computer

Description automatically generated

Hình 3. 2 Frontend microservice

Đối với **2 microservice backend** còn lại bao gồm Student Management, Tuition Fee Payment. Mỗi microservice sẽ bao gồm requirements.txt (thư viện yêu cầu cho hệ thống), “myapp” folder để chứa code hiện thực của microservice đó

A screenshot of a computer screen

Description automatically generated

Hình 3. 3: Backend microservice

Trong mỗi microservice sẽ có các folder như sau

* Controllers: chứa code logic và route của API
* Models: chứa ORM model của database
* Database: chứa file .sqlite database của microservice đó
* Data\_sample: chứa dữ liệu mẫu để test

A screenshot of a computer screen

Description automatically generated

## Giao diện hệ thống

A screen shot of a login screen

Description automatically generated

Hình 3. 4 UI đăng nhập

A screenshot of a chat

Description automatically generated

Hình 3. 5 UI trang thanh toán học phí

A white screen with black text

Description automatically generated

Hình 3. 6 UI lịch sử đóng học phí

## API của hệ thống

A screenshot of a computer

Description automatically generated

Hình 3. 7 Danh sách API

|  |  |  |
| --- | --- | --- |
| Type | Name | Route |
| GET | Get User | api/users/<user\_id> |
| GET | Get Logged In User | api/users/myself |
| GET | Get Tuition Fee | api/get\_tuition\_fee?semester\_id=? |
| POST | Login | api/login |
| POST | Update Logged In User | api/users/myself |

Bảng 3. 1 Danh sách API của Student Management Microservice

|  |  |  |
| --- | --- | --- |
| Type | Name | Route |
| GET | Get Latest Tuition Fee | api/latest\_tuition\_fees/<student\_id> |
| GET | Get Paid Tuition Fee of Logged In User | api/paid\_tuition\_fees/myself |
| POST | Upsert Tuition Fees | api/upsert\_tuition\_fee?username=admin&password=admin&semester\_id=? |
| POST | Pay Tuition Fee | api/latest\_tuition\_fees/<payee\_id> |
| POST | Send OTP | api/send\_otp |
| POST | Send Success Email | api/send\_success\_email |

Bảng 3. 2 Danh sách API của Tuition Fee Payment Microservice

**TÀI LIỆU THAM KHẢO**

1. [CS504070 — Service-Oriented Architecture [Phuc H. Duong] (duonghuuphuc.com)](https://cs504070.duonghuuphuc.com/)