XÂY DỰNG RESTFUL API TÍCH HỢP VÀO ỨNG DỤNG VÍ ĐIỆN TỬ TRÊN ANDROID VÀ WEBSITE

Trương Đặng Minh Tân[1], Nguyễn Duy Luân[2], Nguyễn Tấn Hữu Danh[3], Nguyễn Thị Bích Ngân[4,*]

[1]Sinh viên lớp 12DHTH_TD, mssv: 2001210185, Khoa Công nghệ thông tin, Trường Đại học Công Thương Thành phố Hồ Chí Minh

[2]Sinh viên lớp 12DHTH_TD, mssv: 2001215937, Khoa Công nghệ thông tin, Trường Đại học Công Thương Thành phố Hồ Chí Minh

[3]Sinh viên lớp 12DHTH_TD, mssv: 2001215653, Khoa Công nghệ thông tin, Trường Đại học Công Thương Thành phố Hồ Chí Minh

[4]Khoa Công nghệ thông tin, Trường Đại học Công Thương Thành phố Hồ Chí Minh

[*]Email: truongdangminhtan333@gmail.com, *nganntb@huit.edu.vn*

Ngày nhận bài..... ; Ngày chấp nhận đăng: , 2023

1. TÓM TẮT

Hiện nay, nhu cầu xây dựng hệ thống các phần mềm đa nền tảng đã và đang phát triển mạnh mẽ. REpresentational State Transfer application programming interface (RESTful API) là một thành phần không thể thiếu, giúp giao tiếp giữa các nền tảng trong các hệ thống phần mềm ấy. Trong bài báo này, chúng tôi đề xuất cách xây dựng RESTful API cho giao tiếp nghiệp vụ trong hệ thống ứng dụng "Ví điện tử" bằng cách sử dụng các công cụ như Node.js[1] và Express.js[2]. Đồng thời triển khai máy chủ API lên môi trường Internet[3]. Sau đó tiến hành thực hiện gọi API từ ứng dụng ví điện tử trên thiết bị di động android cho người dùng và trang quản trị với ReactJs[4]. Bên cạnh đó chúng tôi tiến hành thực nghiệm bằng cách gọi các API từ máy chủ và xử lý nghiệp vụ của ứng dụng ví điện tử như: xác thực người dùng, quản lý thông tin cá nhân, thanh toán, chuyển – nhận tiền, hóa đơn, dịch vụ tín dụng, tiết kiệm và nhiều dịch vụ tiện ích khác.

Từ khóa: RESTful APIs, Framework, ví điện tử, mô hình Client – Server, lập trình di động android.

2. GIỚI THIỆU

Ngày nay, với sự phát triển không ngừng của công nghệ thông tin, việc kết nối và truy cập dữ liệu thông qua mạng Internet đã trở nên phổ biến hơn bao giờ hết. Đặc biệt, nhu cầu sử dụng dịch vụ số trên đa nền tảng như: di động, website, phần mềm

[1] https://nodejs.org
[2] https://expressjs.com
[3] https://nt.hadeszhang.tech
[4] https://react.dev/

máy tính… ngày càng cao đòi hỏi các nhà phát triển phải đưa ra những giải pháp mới và cấp thiết cung cấp dịch vụ của họ trên đa nền tảng nhằm phục vụ người dùng mà vẫn đảm bảo tính toàn vẹn dữ liệu và nhất quán, từ đó mô hình Client – Server với RESTful API ra đời, việc truy cập thông tin mọi lúc mọi nơi trên mọi thiết bị đã trở nên dễ dàng hơn bao giờ hết. Hơn nữa, các thiết bị di động, đặc biệt là các thiết bị chạy trên hệ điều hành Android, đã trở thành một phần quan trọng trong cuộc sống của chúng ta. Việc kết hợp RESTful API vào thiết bị di động Android, đồng thời với website triển khai giao diện quản trị hệ thống bằng React.js cho thấy khả năng thích nghi đa nền tảng của mô hình, khả năng mở rộng linh hoạt sẽ giúp các nhà phát triển và doanh nghiệp có nhiều sự lựa chọn, tận dụng các khả năng mở rộng linh hoạt để hoàn thiện hệ thống. Bằng cách sử dụng các thư viện của Android để gọi RESTful API, các nhà phát triển có thể cung cấp các tính năng đáp ứng nhanh chóng cho ứng dụng của mình [1].

Trên cơ sở sử dụng npm (Node package manager) là công cụ quản lý thư viện hiệu quả lớn nhất hiện nay, việc xây dựng máy chủ RESTful API đầy đủ các tính năng nghiệp vụ, hiệu quả, tối ưu, toàn vẹn với tính xác thực cao là hoàn toàn khả thi, bên cạnh đó việc gọi API từ thiết bị di động Android sẽ trở thành xu hướng tương lai trong lĩnh vực phát triển ứng dụng. Bài báo này tập trung vào việc xây dựng RESTful API và gọi API từ ứng dụng di động Android và cung cấp các giải pháp và khuyến nghị cho các nhà phát triển để cải thiện hiệu suất của API [1].

Bài báo này sẽ cung cấp cho độc giả một cái nhìn tổng quan về quá trình xây dựng một hệ thống hoàn chỉnh về nghiệp vụ ví điện tử, cách xây dựng RESTfull API, ứng dụng di động android và cả trang web quản trị, bao gồm cách thiết kế giao diện lập trình ứng dụng API, cách triển khai máy chủ API lên môi trường Internet, gọi và sử dụng các API từ máy chủ trên đa nền tảng.

Phần còn lại của bài báo được tổ chức như sau: trong mục 2, chúng tôi trình bày các sơ sở lý thuyết của việc xây dựng RESTful API, quy trình nghiệp vụ hệ thống ví điện tử; mục 3 trình bày quy trình xây dựng RESTful API và các bước thực hiện gọi API từ một ứng dụng di động; phần thực nghiệm được trình bày trong mục 4 và trong mục 5, chúng tôi tổng kết các vấn đề chính của bài báo.

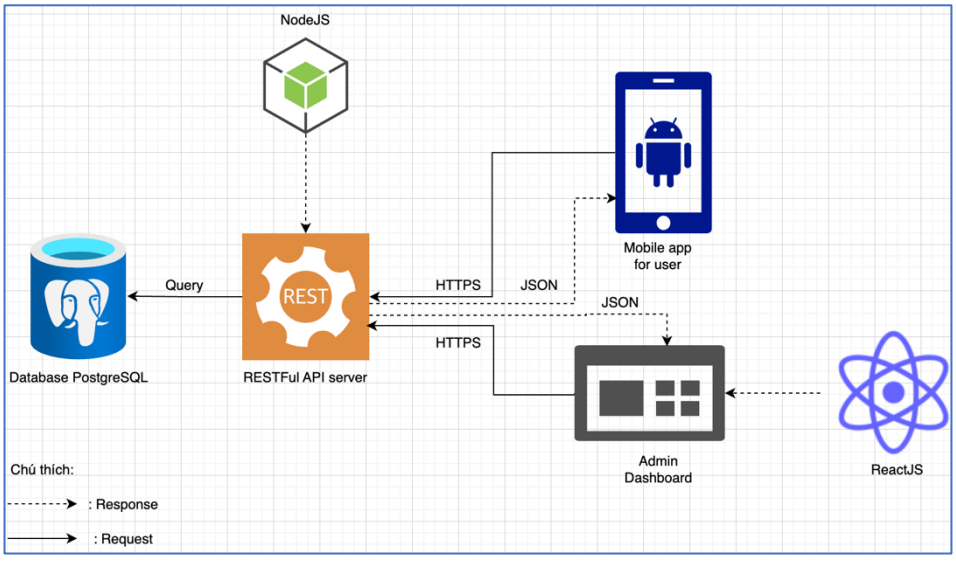
3. CƠ SỞ LÝ THUYẾT

3.1. REST Ful API

API là trung gian máy chủ giao tiếp cho phép hai ứng dụng đầu – cuối tương tác với nhau và tương tác với cơ sở dữ liệu hoặc tài nguyên hệ thống, trung gian máy chủ này nhận các yêu cầu (request) và trả lời (response), các yêu cầu hoặc trả lời có thể hiểu là dữ liệu đầu vào và đầu ra.

Về cơ bản thì RESTful API là tập hợp các quy tắc hay tiêu chuẩn xây dựng hệ thống máy chủ API. Nó chú trọng vào việc quản lý tài nguyên hệ thống (dữ liệu dạng văn bản, hình ảnh, âm thanh,…) cho phép các máy khách giao tiếp với máy chủ, trao đổi dữ liệu thông qua giao HTTP hoặc HTTPS, dữ liệu được truyền đi phổ biến nhất

là dưới dạng JSON[5]. API là một phương thức trung gian cho phép các hệ thống hoặc ứng dụng riêng biệt "giao tiếp" với nhau thông qua các yêu cầu và phản hồi. API cung cấp một cách tiêu chuẩn hóa để các ứng dụng khác tương tác với hệ thống hoặc ứng dụng đó [3]. Hình 1 thể hiện kiến trúc hệ thống sẽ được triển khai, phương thức giao tiếp Client – Server giữa máy chủ RESTFul API với các ứng dụng máy khách.



Hình 1. Kiến trúc hệ thống một hệ thống phần mềm theo mô hình Client – Server

Thông qua việc hiểu và áp dụng các nguyên lý và quy trình hoạt động của RESTful API, các nhà phát triển có thể xây dựng và tích hợp các dịch vụ web linh hoạt và dễ dàng mở rộng.

Tuy nhiên, việc phát triển API là điều không hề đơn giản, đòi hỏi lập trình viên phải có hiểu biết rộng rãi, có kinh nghiệm trong việc tích hợp API cho các hệ thống với nghiệp vụ phân tích phức tạp và cũng phải luôn cập nhật xu hướng mới về lĩnh vực này. Vấn đề bảo mật là mối quan tâm hàng đầu khi xây dựng và tích hợp API. Việc rò rỉ thông tin, dữ liệu nhạy cảm sẽ ảnh hưởng tiêu cực đến hệ thống của bạn và doanh nghiệp, việc rò rỉ thông tin, tấn công thông tin là rất phổ biến đặc biệt là với các hệ thống RESTFul API. Vì vậy khi xây dựng hệ thống không chỉ chú ý đến vấn đề nghiệp vụ hay chỉ đơn giản là chỉ cần tính năng chạy được mà còn phải đảm bảo tính toàn vẹn dữ liệu, tính xác thực và bảo mật của hệ thống [2].

Về phần triển khai RESTful API, chúng tôi đề xuất quá trình xây dựng máy chủ API với Express.js Framework[6]. Express là một framework dành riêng cho NodeJs. Nó cung cấp nhiều tính năng mạnh mẽ trên nền tảng web cũng như ứng dụng di động, hỗ trợ các phương thức HTTP và midleware[7], từ đó tạo ra một API vô cùng mạnh mẽ và dễ sử dụng. Sau đó, chúng tôi sẽ hướng dẫn cách triển khai máy chủ lên môi trường Internet (truy cập từ mọi nơi) để từ đó có thể truy cập đến máy chủ API từ mọi thiết bị khác. Hình 1 mô tả kiến trúc hệ thống chính với máy chủ API được

[5] https://vi.wikipedia.org/wiki/JSON
[6] https://expressjs.com/
[7] https://vi.wikipedia.org/wiki/Middleware

xây dựng với NodeJs(Express.js), PostgreSQL[8] làm DBMS(Database Managerment System), phía máy khách ta có ứng dụng di động xây dựng trên Android studio[9] và trang quản trị xây dựng trên nền tảng trình duyệt với React.js.

3.2. Các công nghệ, framework và thư viện được dùng trong hệ thống Ví điện tử

3.2.1. Một số thư viện chính phía máy chủ (NodeJS):

- **bcrypt:** Thư viện để mã hóa và giải mã mật khẩu, giúp bảo vệ mật khẩu của người dùng trong cơ sở dữ liệu.
- **dotenv[10]:** Cho phép tải các biến môi trường từ tệp .env vào quá trình của ứng dụng, giúp quản lý cấu hình dễ dàng và gia tăng tính bảo mật với các thông tin về máy chủ, thông tin xác thức database, mã hóa,…
- **express:** Framework web cho Node.js, cung cấp các tính năng cơ bản cho việc xây dựng ứng dụng web, như routing, middleware và quản lý yêu cầu.
- **jsonwebtoken[11]:** Thư viện để tạo và xác thực JSON Web Tokens (JWT) cho việc xác thực và quản lý phiên làm việc của người dùng, đây là một framework cực kỳ quan trọng cho khả năng bảo mật trong hệ thống.
- **nodemon[12]:** Công cụ hữu ích cho việc phát triển Node.js, tự động khởi động lại ứng dụng khi các tập tin thay đổi, giúp tiết kiệm thời gian và công sức trong quá trình phát triển.
- **pg:** Thư viện Node.js cho phép tương tác với cơ sở dữ liệu PostgreSQL.
- **request-ip:** Middleware Express để trích xuất địa chỉ IP của người dùng từ yêu cầu HTTP, thư viện này còn dùng cho việc chống tấn công, ghi nhận nguồn gốc ip đã sử dụng tính năng nhằm ghi nhận trong logs.
- **sequelize[13]:** Một ORM (Object-Relational Mapping) cho Node.js, giúp tương tác với cơ sở dữ liệu một cách dễ dàng và linh hoạt.
- **socket.io[14]:** Thư viện cho phép giao tiếp thời gian thực giữa máy khách và máy chủ thông qua các kết nối WebSocket và HTTP.
- **winston[15]:** Cung cấp tính năng ghi logs theo mức độ nghiêm trọng và theo ngày.
- **Hệ quản trị cơ sở dữ liệu PostgreSQL**.
- **Triển khai máy chủ API với Nginx:** Nginx là một máy chủ web và cũng có thể hoạt động như một máy chủ proxy ngược, cung cấp một loạt các chức năng quan trọng cho việc xử lý yêu cầu web và cân bằng tải.

3.2.2. Phía máy khách ứng dụng di động trên Android:

[8] https://www.postgresql.org/
[9] https://developer.android.com/studio
[10] https://www.dotenv.org/
[11] https://jwt.io/
[12] https://viblo.asia/p/nodejs-bai-9-gioi-thieu-nodemon-package-3Q75wmyeZWb
[13] https://sequelize.org/
[14] https://socket.io/
[15] https://viblo.asia/p/write-logger-mot-cach-don-gian-voi-winston-V3m5WPygKO7

Retrofit[16] là một thư viện phổ biến được sử dụng trong phát triển ứng dụng Android để tương tác với các API.

3.2.3. Phía máy khách website quản trị với ReactJS:

Axios: Axios là một thư viện HTTP client phổ biến và linh hoạt, được sử dụng rộng rãi trong cả ReactJS và Node.js. Nó hỗ trợ các yêu cầu HTTP như GET, POST, PUT, DELETE và hỗ trợ xử lý yêu cầu bất đồng bộ thông qua các Promise.

Triển khai máy chủ với pm2: PM2[17] là một trình quản lý tiến trình sản xuất cho Node.js, được sử dụng rộng rãi để quản lý ứng dụng Node.js trong môi trường sản xuất.

4. XÂY DỰNG HỆ THỐNG (SYSTEM DEVELOPMENT):

4. 1. Cơ sở dữ liệu (Database):

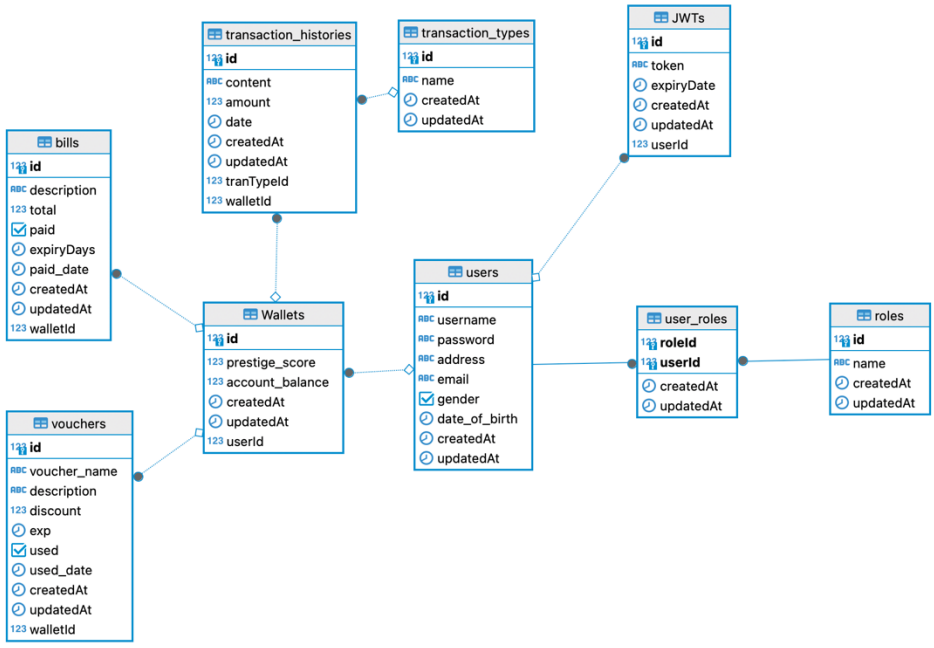
Dựa vào phân tích quy trình nghiệp vụ chúng tôi đưa ra các tính năng chính của hệ thống như sau:

- Phía người dùng:
  + Xác thực người dùng (đăng ký, đăng nhập).
  + Giao dịch nạp, thanh toán, chuyển tiền.
  + Giao dịch quản lý và thanh toán hóa đơn, sử dụng phiếu giảm giá.
  + Dịch vụ tín dụng và tiết kiệm.
  + Chat trực tuyến và báo cáo.
- Phía quản trị:
  + Quản lý thông tin người dùng (CRUD).
  + Quản lý danh sách hóa đơn, mã giảm giá.
  + Xác nhập nạp tiền.
  + Tạo và quản lý các chương trình tín dụng.
  + Chat trực tuyến và nhận báo cáo.

Với những nghiệp vụ và tính năng chính trong hệ thống mà chúng tôi đã đưa ra ở trên, chúng tôi xây dựng cơ sở dữ liệu như hình 2. Mỗi người dùng khi tạo tài khoản sẽ sở hữu một ví (wallet) tương ứng, ví này ràng buộc với các đối tượng hóa đơn (bill), mã giảm giá (voucher), lịch sử giao dịch (transaction_histories), điều này giúp xác định user nào sẽ sở hữu những gì, các loại giao dịch chính mặc định là: nạp, chuyển tiền và thanh toán, các loại giao dịch này sẽ xác định tính chất của lịch sử giao dịch. Bên cạnh đó chúng tôi còn xây dựng phân quyền các người dùng (user), mỗi user sẽ sở hữu quyền (role) tương ứng, mặc định khi người dùng đăng ký tài khoản sẽ có quyền "user",

[16] https://square.github.io/retrofit/
[17] https://pm2.keymetrics.io/

ngoài ra quyền "admin" giúp xác định những người dùng quản trị có khả năng tương tác và quản lý các tài nguyên hệ thống hiệu quả hơn. Để tăng cường tính bảo mật và xác thực chặt chẽ cho hệ thống, chúng tôi đề xuất sử dụng JWT(Json web token) là một dịch vụ mã hóa xác thực, nó đóng vai trò là mã xác thực cho hầu hết mọi tác vụ của một người dùng trong hệ thống trong toàn bộ một phiên làm việc (phiên đăng nhập), mỗi lần đăng nhập các token này sẽ được tạo mới và chúng là duy nhất bằng cách sử dụng các trường dữ liệu duy nhất của người dùng như : id, username, email, các token được tạo ra cũng sẽ là duy nhất và xác định người dùng nào đang dùng tính năng gì và có được phép thực hiện nó hay không.



Hình 1: Mô hình thiết kế cơ sở dữ liệu.

4. 2. API:

4. 2. 1. Cấu trúc dữ liệu và dạng response:xw

Trong hình 3, chúng ta có thể dễ dàng nhìn thấy dữ liệu mà máy chủ API giao tiếp và trả về là một chuỗi json, dữ liệu xác định bởi một key và value giá trị tương ứng(bảng 1):

```
{
    "id": 1,
    "username": "Hades",
    "email": "hades@gmail.com",
    "roles": [
        "ROLE_USER"
    ],
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6MSwiaWF0IjoxNzEyMzc0MTQyLCJleHAiOjE3MTIzNzc3NDJ9.
        le-TyicVZniLx53Pgan6FC3ujtG2LjMlgf8QlF7ArIs",
    "refreshToken": "b4656609-3f9f-4f37-ba8d-97d9a7864061",
    "address": "Tỉnh Bà Rịa - Vũng Tàu",
    "gender": true,
    "date_of_birth": "2003-02-24T00:00:00.000Z"
}
```


Hình 2: Chuỗi dữ liệu giao tiếp dưới dạng Json trong tính năng đăng nhập.

Bảng 1: Mô tả key và value tương ứng trong chuỗi phản hồi json**.**

Key	Value
ID	1
Username	Hades
Email	hades@gmail.com
Roles	ROLE_USER
Access Token	Trong_hình 3
Refresh Token	b4656609-3f9f-4f37-ba8d-97d9a7864061
Address	Tỉnh Bà Rịa - Vũng Tàu
Gender	True
Date_of_birth	2003-02-24T00:00:00.000Z

Dựa vào cấu trúc dữ liệu này, phía máy khách sẽ xây dựng tính năng gọi API để đọc và phân tích lấy ra các trường dữ liệu tương ứng.

Header: là nơi chứa các thông tin cần thiết của request tương ứng, nhưng người dùng cuối (end-users) không biết đến sự tồn tại của nó. RESTfull API hỗ trợ một số tiêu chuẩn HTTP header chung. Một số header thường gặp:

- HTTP Accept: cho biết định dạng mà máy phía người dùng chấp nhận cho nội dung phản hồi (response), có giá trị dưới dạng application/json hoặc application/xml. Thường thì mặc định nó là application/json.
- HTTP Content-type: cho biết định dạng của nội dung yêu cầu (request body), có giá trị dưới dạng application/json hoặc application/xml. Thường thì mặc định nó là application/json.
- HTTP Authorization: cung cấp "khóa truy cập" Oauth 2.0 để xác thực người dùng.

Request body (Nội dung yêu cầu): là một chuỗi trong request nhằm để cung cấp thông tin, chẳng hạn như các trường để cập nhật bản ghi. Request body có thể là dữ liệu dưới dạng JSON hoặc XML [2].

Bảng 2: API Endpoint và tương tác tài nguyên hệ thống tương ứng.

API endpoint	Phương thức HTTP	Mô tả
`/users/getAllBills`	GET	Hiện thị danh sách các hóa đơn
`/admin/bill/createBill`	POST	Thêm hóa đơn mới vào CSDL.
`/users/transferMoney`	POST	Chuyển tiền từ một ví của người dùng hiện tại sang một ví khác.
`/admin/updateUser/:id`	PUT	Cập nhật thông tin user trong hệ thống.
`/admin/deleteUser/:id`	DELETE	Xóa toàn bộ thông tin người dùng và ví.

4. 2. 2. Đề xuất một số phương án tối ưu và bảo mật hệ thống:

chúng tôi đề xuất một số phương án tăng cường khả năng quản lý máy chủ, giảm thiểu rủi ro bảo mật có thể xảy ra:

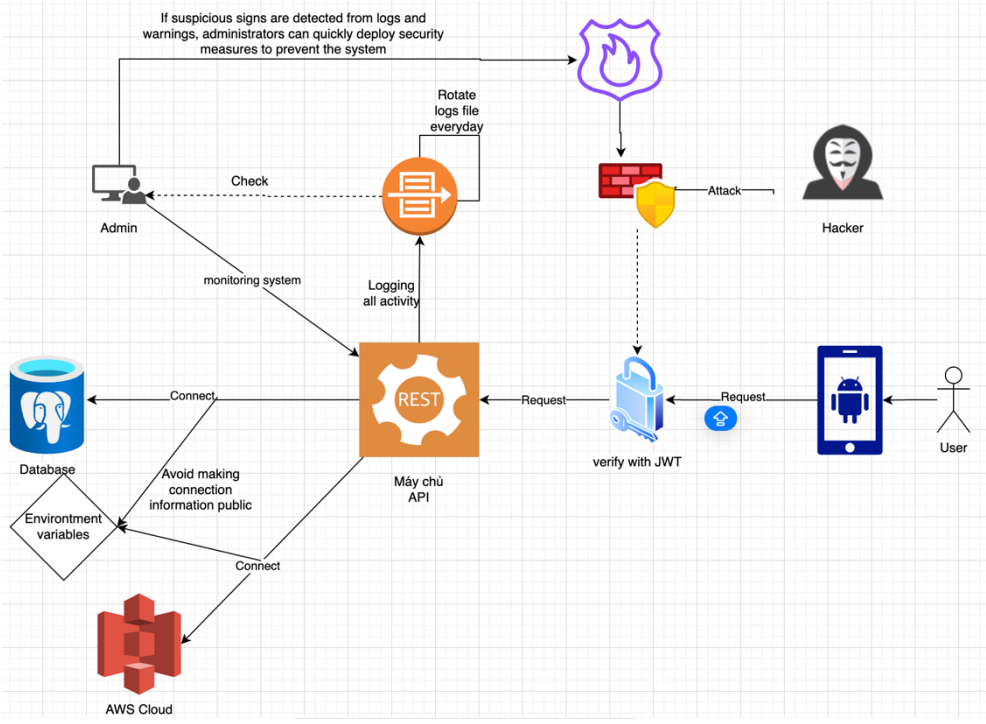
Đầu tiên là sử dụng các biến môi trường trong tệp môi trường trong hầu hết mọi thông tin truy cập: cơ sở dữ liệu, cổng, mã refreshToken, thời gian hết hạn token và nhiều thông tin cần hạn chế công khai khác.

Bênh cạnh đó việc triển khai xác thực bằng Json web tokens [18]là cần thiết để xác thực mọi yêu cầu từ máy khách đến máy chủ, JWT sử dụng chuẩn mở (RFC 7519) định nghĩa một cách nhỏ gọn và khép kín để truyền một cách an toàn thông tin giữa các bên dưới dạng đối tượng JSON, các token này giúp xác định chính xác người dùng nào đang thực hiện hành động gì và họ có được phép không, vì mỗi token xác định một người dùng duy nhất và chúng có hạn sử dụng.

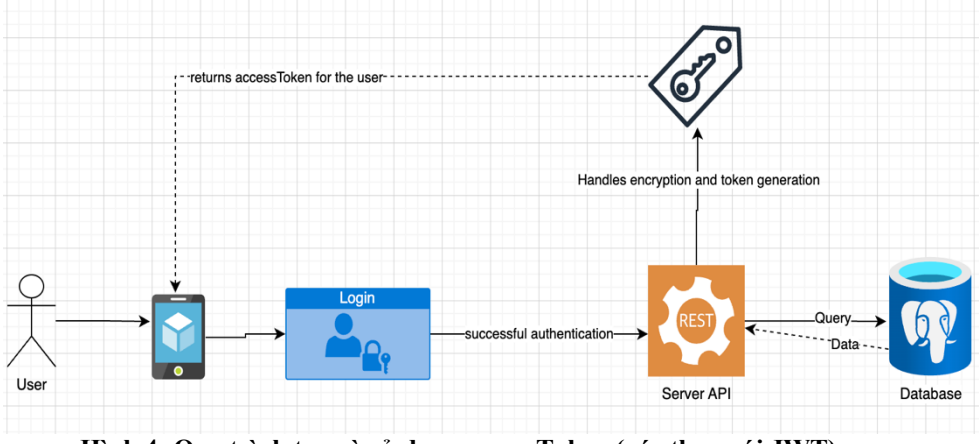
Tiếp theo là phương pháp ghi nhật ký (logs) hoạt động máy chủ phân loại dựa trên mức độ nghiêm trọng (FATAL, ERROR, WARNING, INFO, DEBUG), các tệp nhật ký được ghi theo ngày và xoay vòng nén sau mỗi vòng thời gian quy định hoặc độ lớn tùy theo yêu cầu nghiệp vụ mà hệ thống cần, chúng giúp ghi lại mọi hoạt động tương tác với máy chủ với thông tin cụ thể và thông thường thông tin phản hồi với người dùng có dạng mã hóa với Error code, nhưng phía máy chủ phải ghi nhận thông tin cụ thể hơn bao gồm cả truy xuất địa chỉ IP của máy khách, điều này giúp người quản trị hệ thống dễ dàng phát hiện các vấn đề bảo mật, giảm thiểu rủi ro có thể xảy ra và quản lý máy chủ hiệu quả hơn.

Phương pháp hay và hiệu quả khác là triển khai các bản sao lưu nhật ký, máy chủ lưu trữ tài nguyên và dữ liệu qua dịch vụ điện toán đám mây như AWS S3, giúp quản lý hệ thống hiệu quả và tối ưu hơn. Hình 3 mô tả các phương pháp quản lý và bảo mật hệ thống.

[18] https://viblo.asia/p/tim-hieu-ve-json-web-token-jwt-7rVRqp73v4bP



Hình 3: Kiến trúc triển khai các phương pháp quản lý và bảo mật.



Hình 4: Quy trình tạo và sử dụng accessToken (xác thực với JWT).

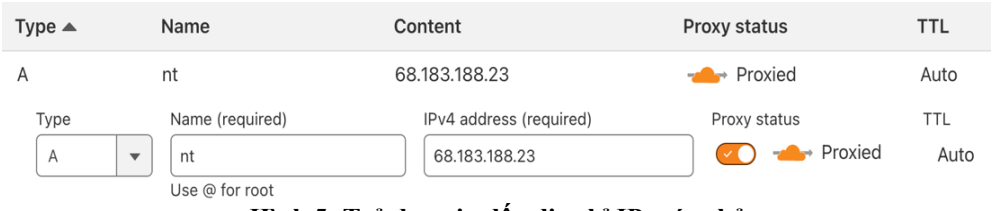
4. 2. 3. Triển khai máy chủ API lên môi trường Internet:

Để triển khai máy chủ API lên môi trường Internet đòi hỏi phải có kiến thức rộng rãi về mạng, công nghệ triển khai, các giao thức, cách kết nối giữa các phần…quan trọng và cơ bản nhất vẫn là tương tác với máy chủ hệ điều hành linux.

Bước đầu tiên cần phải có tên miền (domain)[19], có thể mua hoặc thuê ở bất cứ doanh nghiệp nào, ở đây chúng tôi dùng dịch vụ của Cloudflare vì họ cung cấp cả Proxy và SSL cho tên miền. Dưới đây là từng triển khai:

[19] https://vi.wikipedia.org/wiki/T%C3%AAn_mi%E1%BB%81n

Vào trang Console của Cloudflare đi đến trang DNS, chọn add record, tại mục



Hình 5: Trỏ domain đến địa chỉ IP máy chủ.

content nhập vào địa chỉ IP của máy chủ linux mà bạn sử dụng, bây giờ thì tên miền đã được trỏ đến máy chủ.

Truy cập vào máy chủ Linux bằng giao thức SSH[20], tiếp tục cài đặt Nginx, NodeJS, PostgreSQL, PM2, thay đổi cấu hình PostgreSQL trong tệp config để cho phép kết nối từ mọi nơi.

Truy cập tệp cấu hình Nginx tại vị trí: `/etc/nginx/sites-available/` và định cấu hình như hướng dẫn bên dưới, thay server_name thành tên miền và lắng nghe tại cổng 3000 hoặc bất kỳ cổng nào khác, lúc này khi kích hoạt máy chủ NodeJS tại cổng 3000 thì Nginx sẽ lắng nghe tại cổng 3000.

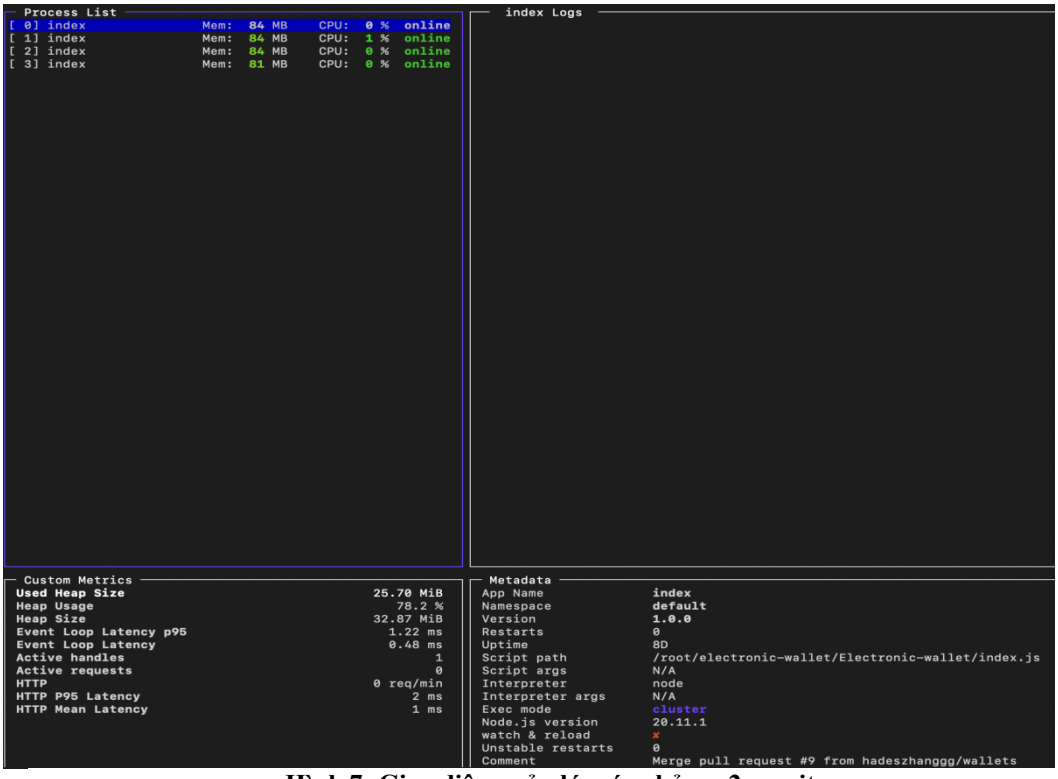
```
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name nt.hadeszhang.tech;
        location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        }
}
```

Hình 6: Cấu hình Nginx.

[20] https://www.hostinger.vn/huong-dan/ssh-la-gi-va-cach-su-dung-ssh-cho-nguoi-moi-bat-dau

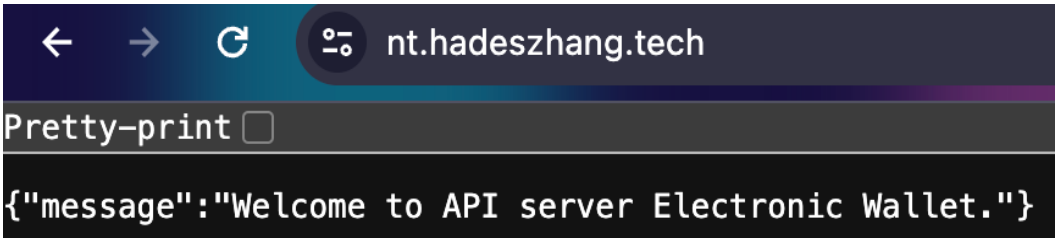
Khởi chạy máy chủ NodeJS bằng PM2: `pm2 start -i 4 /root/electronic-wallet/Electronic-wallet/index.js`

Sau đó dùng lệnh: `pm2 monit -` để kiểm tra máy chủ NodeJS có được khởi chạy chưa, như trong hình 23 máy chủ được chạy với 4 luồng nhằm chia sẽ các kết nối giảm tải cho máy chủ:



Hình 7: Giao diện quản lý máy chủ pm2 monit.

Giờ thì truy cập tên miền https://nt.hadeszhang.tech và kiểm tra kết quả triển khai:



Hình 8: Có thể truy cập máy chủ từ Internet.

5. KẾT QUẢ THỰC NGHIỆM (EXPERIMENT)

Chúng tôi đã xây dựng thành công một máy chủ API cho nghiệp vụ ví điện tử với các tính năng cơ bản như quản lý thông tin người dùng, quản lý hóa đơn, mã giảm giá, tính năng chuyển tiền, các tính năng bảo mật máy chủ như ghi nhật ký, truy xuất thông IP của người dùng, xác thực bằng JWT mã hóa, chống tấn công DDdos qua việc gửi số lượng lớn yêu cầu từ cùng một địa chỉ IP, xây dựng được connection

pool nhằm quản lý truy vấn hiệu quả và tối ưu hơn, triển khai được máy chủ API lên Internet bằng Nginx và PM2, gọi và sử dụng một số API cơ bản trong ứng dụng android. Kết quả thực nghiệm truy xuất API sẽ được thống kê ở bảng 30 và đường dẫn đến máy chủ API tại địa chỉ: https://nt.hadeszhang.tech[21].

Bảng 3: Thống kê API Endpoint thực nghiệm.

API Endpoint	Method	Mô tả	INPUT
`/auth/signup`	POST	Đăng ký tài khoản	Username, email, password, addresss, date_of_birth, gender.
`/auth/signin`	POST	Đăng nhập	Username/Email, password
`/users/getAllBills`	GET	Lấy danh sách hóa đơn	AccessToken
`/users/getAllVouchers`	GET	Lấy danh sách mã giảm giá	AccessToken
`/users/getUnpaidBills`	GET	Lấy danh sách hóa đơn chưa thanh toán	AccessToken
`/users/getUnusedVouchers`	GET	Lấy danh sách mã giảm giá chưa dùng	AccessToken
`/users/transferMoney`	POST	Chuyển tiền cho một người dùng khác	AccessToken, recipientUsernameOrEmail, content, amount
`/admin/bill/createBill`	POST	Tạo một hóa đơn mới và gán cho người dùng	AccessToken, description, total, wallet_id
`/admin/voucher/createVoucher`	POST	Tạo mới một mã giảm giá và gán cho người dùng	AccessToken, voucher_name, description,discount,expiry
`/admin/deleteUser/:id`	DELETE	Xóa người dùng	AccessToken, userId
`/admin/updateUser/:id`	PUT	Sửa thông tin người	AccessToken, userId, username, password, email, address, gender,

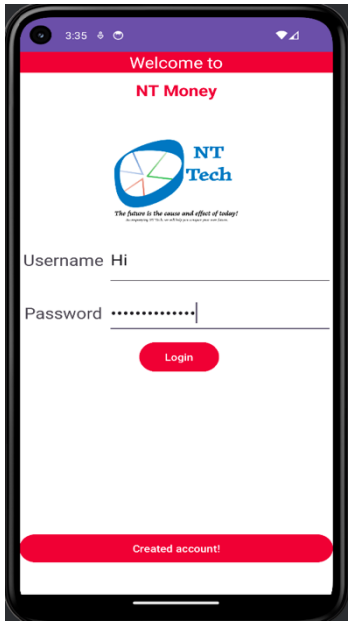
[21] https://nt.hadeszhang.tech

		dùng	date_of_birth

Mặc dù chưa thực hiện gọi đầy đủ các API cho ứng dụng android, nhưng chúng tôi cũng đã thực nghiệm với hai API chính là đăng ký tài khoản và đăng nhập kèm với đó là kết nối socket từ android đến máy chủ (hình 30, 31, 32).

Tôi đã dùng tính năng ghi log để ghi ra logs để ghi ra các thông tin trả về từ máy chủ khi đăng nhập thành công, và khi đăng ký thông tin user cũng được cập nhật vào cơ sở dữ liệu (hình 32, 33). Chúng ta tiến hành tạo một user mới với các thông

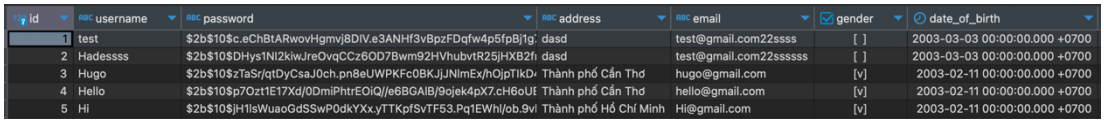
Hình 10: Đăng ký tài khoản.



Hình 9: Đăng nhập với tài khoản vừa đăng ký.

tin này và sử dụng nó trong tất cả các activity khác.

Hình 11: Dòng logs in ra từ thông tin trả về của máy chủ khi đăng nhập thành công.



id	username	password	address	email	gender	date_of_birth
1	test	$2b$10$c.eChBtARwovHgmvj8DIV.e3ANHf3vBpzFDqfw4p5fpBj1g	dasd	test@gmail.com22ssss	[ ]	2003-03-03 00:00:00.000 +0700
2	Hadessss	$2b$10$DHys1NI2kiwJreOvqCCz6OD7Bwm92HVhubvtR25jHXB2fi	dasd	test@gmail.com22ssssss	[ ]	2003-03-03 00:00:00.000 +0700
3	Hugo	$2b$10$zTaSr/qtDyCsaJ0ch.pn8eUWPKFc0BKJJNlmEx/hOjpTIkD	Thành phố Cần Thơ	hugo@gmail.com	[v]	2003-02-11 00:00:00.000 +0700
4	Hello	$2b$10$p7Ozt1E17Xd/0DmiPhtrEOiQ/e6BGAlB/9ojek4pX7.cH6oUI	Thành phố Cần Thơ	hello@gmail.com	[v]	2003-02-11 00:00:00.000 +0700
5	Hi	$2b$10$jH1lsWuaoGdSSwP0dkYXx.yTTKpfSvTF53.Pq1EWhl/ob.9v	Thành phố Hồ Chí Minh	Hi@gmail.com	[v]	2003-02-11 00:00:00.000 +0700

Hình 12: Cơ sở dữ liệu được cập nhật.

Ngay khi mở chat activity trên android, phía máy chủ sẽ nhận được sự kiện này và tạo ra một phòng chat (room chat) dựa trên userId của người dùng, điều này giúp tạo ra nhiều room khác nhau và tương ứng với mỗi người dùng (hình 34).

Hình 13: Ngay khi người dùng đăng nhập thành công và chuyển đến trang chat, máy chủ sẽ

```
Server is running at http://localhost:333
A user connected with ID: Zxozz6To_97MwOfoAAAB
A user connected with ID: NwjBfD_ECj7LCisDAAAD
Has user connect to chat with UserID: 5
```

nhận được sự kiện này và tạo ra một room dựa trên userId của người dùng.

6. KẾT LUẬN

Trong bài báo này chúng ta đã nghiên cứu sâu về quy trình nghiệp vụ ví điện tử, cách xây dựng máy chủ RESTful API với NodeJS, các phương pháp tối ưu và bảo mật cho máy chủ, cách triển khai máy chủ lên môi trường Internet, cách gọi và sử dụng API trên ứng dụng di động Android với thư viện retrofit2, kết nối socket.io giữa ứng dụng di động và máy chủ.

Tóm lại, việc tìm hiểu các phương pháp xây dựng máy chủ API, nghiên cứu những giải pháp bảo mật và tối ưu, phân tích quy trình nghiệp vụ để xây dựng hệ thống là một kỹ năng vô cùng quan trọng mà một nhà phát triển thực thụ cần có, bài báo này không chỉ giúp mọi người nắm rõ cơ cấu tổ chức của một máy chủ API mà còn giúp người đọc hiểu rõ hơn về quy trình nghiệp vụ của một hệ thống lớn. Nghiên cứu này có thể là giải pháp chuyển đổi số cho nhiều doanh nghiệp mong muốn dịch vụ của mình được phổ biến rộng rãi trên đa nền tảng mà vẫn đảm bảo được tính bảo mật, tối ưu và toàn vẹn.

7. TÀI LIỆU THAM KHẢO

1. Do Hung, Android Library: Tìm hiểu Retrofit 2.0, truy cập ngày 28/03/2024 từ https://viblo.asia/p/android-library-tim-hieu-retrofit-20-AQrMJVojM40E.
2. Vladimir Maskov, Implementing REST Client for Android, Metropolia, University of Applied Sciences, 2020.
3. Ian Blair (CEO and Co-Founder of BuildFire), How to Create a RESTful API For Your Mobile App, 2022. Truy cập ngày 04/04/2024 từ https://buildfire.com/create-restful-api-mobile-app/.
4. Sarrah Pitaliya, A Tutorial to Deploy the Node.Js App to Nginx Server, truy cập ngày 04/04/2024 từ https://hackernoon.com/a-tutorial-to-deploy-the-nodejs-app-to-nginx-server.
5. Rahul, Deploying Node Application using PM2, truy cập ngày 30/03/2024 từ https://tecadmin.net/deploy-nodejs-application-pm2/.

ABSTRACT

BUILDING RESTFUL API INTEGRATING INTO E-WALLET APPLICATION ON ANDROID AND WEBSITE

Truong Dang Minh Tan[1*], Nguyen Duy Luan[1], Nguyen Tan Huu Danh[1], Nguyen Thi Bich Ngan[1]

Falcuty of Information Technology, Ho Chi Minh city of Industry and Trade

[*]Email: truongdangminhtan333@gmail.com, *nganntb@hufi.edu.vn*

Currently, there is a growing demand for developing cross-platform software systems. The Representational State Transfer application programming interface (RESTful API) plays an indispensable role in facilitating communication among platforms within such software systems. In this paper, we propose a method for building a RESTful API for an "E-wallet" business domain using tools like Node.js and Express.js, while deploying the API server to the Internet environment. Subsequently, we proceed to demonstrate API calls from the E-wallet application on Android for end-users and an administrative dashboard using ReactJs. Additionally, we conduct experiments by invoking APIs from the server and handling business logic of the E-wallet application, including user authentication, personal information management, payments, fund transfers, invoicing, credit services, savings, and various other utility services.

Keywords: RESTful APIs, Framework, Electronic wallet, Client – Server model, mobile programing with Android.

Tôi xin cam kết bài báo này chưa được đăng và gửi đăng ở bất kỳ tạp chí nào khác.
Tác giả:

Họ và tên: Trương Đặng Minh Tân - 12DHTH_TD
Số điện thoại: 0965917852