TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

WEB PROGRAMMING AND APPLICATIONS
MIDTERM ESSAY

Research Web Services and build a web service

Người hướng dẫn: **ThS MAI VĂN MẠNH**

Người thực hiện: **BÙI HỮU LỘC – 521H0504**

TRẦN HỮU NHÂN – 521H0507

VŨ QUỐC NHẬT TÂN – 521H0420

Lớp : **21H50301**

Khoá : **25**

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

WEB PROGRAMMING AND APPLICATIONS
MIDTERM ESSAY

Research Web Services and build a web service

Người hướng dẫn: **ThS MAI VĂN MẠNH**
Người thực hiện: **BÙI HỮU LỘC**
TRẦN HỮU NHÂN
VŨ QUỐC NHẬT TÂN
Lớp : **21H50301**
Khoá : **25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

We would like to express our sincere gratitude for lecturer Main Van Manh throughout the mid-term report of the Web Programming and Applications course. Your expertise, knowledge, and commitment have been invaluable in helping me and my classmates navigate this complex and challenging subject.

Your clear explanations and thorough feedback on our projects have been instrumental in our understanding of the course material. Your dedication to teaching and ensuring that each of us reaches our full potential has not gone unnoticed and is greatly appreciated.

Once again, thank you for your time, effort, and support. We have learned a great deal from you and look forward to continuing to learn from you for the remainder of the course.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của ThS Mai Văn Mạnh; 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 đồ án 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 đồ án 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 29 tháng 4 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Bùi Hữu Lộc

Trần Hữu Nhân

Vũ Quốc Nhật Tân

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

This essay research about web services and RESTful API help reader will be able to answer the following questions:

- What is Web Service and why is Web API needed.

- What types of Web services are there, similarities and differences between them.

- What is RESTful web service, components of RESTful API and its working principle.

- Examples of RESTful web services and its application in current practice.

- How to create a RESTful service using PHP-MySQL, how to deploy and use (shop management API), test this API using POSTMAN.

- Some issues in designing, building, and deploying RESTful services.

MỤC LỤC

LỜI CẢM ƠN 1
ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC 2
TÔN ĐỨC THẮNG 2
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 3
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 3
CHAPTER 1: INTRODUCTION 4
CHAPTER 2: BACKGROUND 4
2.1 What are Web Services? 4
2.2 What is it application on modern web development? 5
CHAPTER 3: TYPES OF WEB SERVICES 6
3.1 How many types of web services 6
3.2 Compare between different type of web services. 6
CHAPTER 4: RESTFUL WEB SERVICES 7
4.1 What is RESTful web service? 7
4.2 Components of RESTful Web Services 8
4.2.1 Resources 8
4.2.2 URIs 8
4.2.3 HTTP Methods 8
4.2.4 Representations 8
4.2.5 Hypermedia Links 9
4.3 Examples of RESTful Web Services 9
CHAPTER 5: COMMON ISSUE 10
5.1 Security 10
5.2 Performance 10
5.3 Reliability 10
5.4 Interoperability 11

5.5 Scalability........11
CHAPTER 6: IMPLEMENTATION........11
6.1 Explain demo web services:........11
6.1.1 Register........12
6.1.2 Login........12
6.1.3 Get products.........14
6.1.4 Show product........15
6.1.5 Store product........16
6.1.6 Update product........17
6.1.7 Delete product.........18
6.1.8 Get cart........19
6.1.9 Update cart........20
6.1.10 Payment........21
6.1.11 Verify Payment........23
6.1.12 Get orders:........24
6.1.13 Show orders........25
6.2 Demo webservices on hosting........26
CHAPTER 7: CONCLUSION........26
REFERENCES........27

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Figure 1. Web services model..5
Figure 2. API register..12
Figure 3. API login..13
Figure 4. API Get Products..14
Figure 5. API Get Product with ID..15
Figure 6. API Create Product..16
Figure 7. API Edit Product..17
Figure 8. API Delete Product..18
Figure 9. API Get information of Cart..19
Figure 10. API Update Cart..20
Figure 11. API Get Payment URL..21
Figure 12. VNPay Payment Interface..22
Figure 13. VnPay Confirm payment OPT..22
Figure 14. Verify Payment Status..23
Figure 15. API Get all orders with pagination..24
Figure 16. API Show information about an order..25

DANH MỤC BẢNG

Table 1 Compare API based on features...6

CHAPTER 1: INTRODUCTION

Web Services are an essential part of modern web development. They allow different applications and systems to communicate with each other and exchange information. The primary objective of this essay is to provide a comprehensive overview of Web Services and demonstrate how to build a Web Service using PHP-MySQL. In this essay, we will discuss the significance of Web Services and REST API, their relevance in modern web development, and the objectives of our project.

CHAPTER 2: BACKGROUND

2.1 What are Web Services?

A web service is a collection of open protocols and standards that enable data to be exchanged between various applications or systems. Web services are used by software programs written in various programming languages and running on various platforms to exchange data via computer networks such as the Internet in a manner like inter-process communication on a single machine.

A web service is any software, application, or cloud technology that connects, interoperates, and exchanges data messages - often XML (Extensible Markup Language) - across the internet using standardized web protocols (HTTP or HTTPS).

Web services offer the advantage of allowing applications written in various languages to communicate with one another by sending data between clients and servers via a web service. A web service is invoked by a client submitting an XML request, to which the service responds with an XML response.

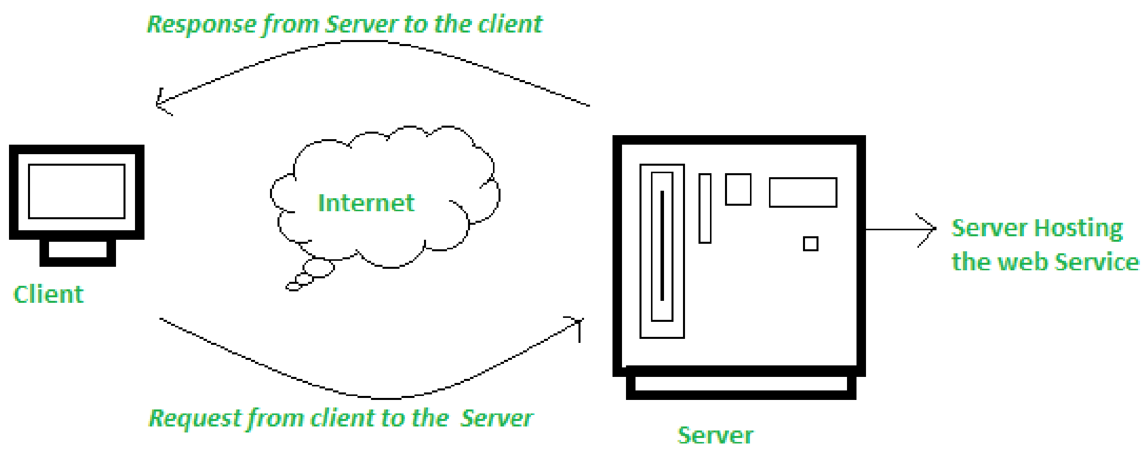

Response from Server to the client
Internet
Client
Server Hosting the web Service
Request from client to the Server
Server

Figure 1. Web services model

2.2 What is it application on modern web development?

The primary benefit of Web Services is that they enable different applications to communicate with each other seamlessly, even if they are built using different languages or frameworks. This means that developers can build complex applications that can interact with each other easily, without worrying about the underlying technology.

Web APIs is a crucial aspect of Web Services, as they define a set of rules and protocols for communication between applications. This makes it easier for developers to build applications that can interact with each other, as they do not need to worry about the underlying technology or protocol.

In summary, Web Services and Web APIs are essential in modern web development, enabling different applications and systems to communicate with each other and share data and functionality. They make it easier for developers to build complex applications that can interact with each other seamlessly, without worrying about the underlying technology or protocol.

CHAPTER 3: TYPES OF WEB SERVICES

3.1 How many types of web services

There are several types of Web Services, each with its own set of advantages and disadvantages, and developers must select the appropriate type based on the specific needs of their project. There exist multiple classifications of Web Services, including **SOAP**, **REST**, and **XML-RPC**. The essay also compares these various types of Web Services, emphasizing their similarities and differences as well as the benefits and drawbacks of each.

3.2 Compare between different type of web services.

The similarities of Web Services are as follows:

- All of them allow different applications to access and use data over the network.
- All of them use the HTTP protocol and request/response methods to transmit data between the server and the user.
- All of them use data format languages such as XML or JSON to transmit information between applications.
- All of them can be used to create complex web applications, such as social networking or e-commerce applications.

Table compare based on feature:

Table 1 Compare API based on features.

Feature	**SOAP**	**REST**	**XML-RPC**
Data Format	Only support XML	Supports multiple data format (XML, JSON, plain text)	Only support XML

Transport	Can use various transport protocols (e.g., HTTP, SMTP)	Primarily uses HTTP	Primarily uses HTTP
Catching	Does not have built-in caching mechanism	Supports caching through HTTP	Does not have built-in caching mechanism
Security	Has built-in WS-Security standard	Relies on transport-level security (e.g., HTTPS)	Relies on transport-level security (e.g., HTTPS)
Easy to use	Requires more setup and configuration	Easier to implement and use	Easier to implement and use than SOAP but more complex than REST

Each type of Web Service has its own strengths and weaknesses, and developers must choose the appropriate type based on the specific requirements of their project. SOAP is a more traditional type of Web Service that uses the XML protocol for communication. RESTful Web Services, on the other hand, use the HTTP protocol to communicate and are designed to be lightweight, scalable, and flexible. XML-RPC is a simple type of Web Service that uses XML for communication and is suitable for simple applications that do not require advanced functionality.

CHAPTER 4: RESTFUL WEB SERVICES

4.1 What is RESTful web service?

RESTful web service is a subset of web services and it is stateless client-server architecture in which web services are resources identified by URIs. REST, which stands for Representational State Transfer, is an architectural style that defines a set of constraints for creating web services. RESTful web services enable requesting systems to access and manipulate textual representations of web resources using a consistent and predefined set of stateless operations. RESTful APIs adhere to software communication standards that are secure, dependable, and efficient.

4.2 Components of RESTful Web Services

RESTful Web Services are composed of several components, including resources, URIs, HTTP methods, representations, and hypermedia links. These components work together to enable clients to interact with the server and perform various operations.

4.2.1 Resources

Resources are the key components of RESTful Web Services. They represent the data or functionality that can be accessed by clients. Resources can be anything from a user profile to a blog post or a product catalog.

4.2.2 URIs

URIs (Uniform Resource Identifiers) are used to identify resources in RESTful Web Services. They are the addresses of the resources that clients can access using HTTP methods. For example, the URI for a user profile might be **`https://yoursite.com/users/1`**

4.2.3 HTTP Methods

HTTP methods are used to perform operations on resources in RESTful Web Services. The most used HTTP methods are GET, POST, PUT, and DELETE. GET is used to retrieve a resource, POST is used to create a new resource, PUT is used to update an existing resource, and DELETE is used to delete a resource.

4.2.4 Representations

Representations are the different ways in which a resource can be presented to clients. These can include JSON, XML, or HTML representations, depending on the needs of the client.

4.2.5 Hypermedia Links

Hypermedia links are links that connect resources to each other. They enable clients to navigate between different resources and perform various operations. Hypermedia links are a crucial aspect of RESTful Web Services, as they enable clients to interact with the server in a flexible and scalable manner.

4.3 Examples of RESTful Web Services

There are many examples of RESTful Web Services in the real world, including social networking sites, e-commerce sites, and online marketplaces. One example of a RESTful Web Service is the GitHub API. The GitHub API enables developers to access and manipulate data on the GitHub platform using RESTful Web Services. Developers can use the API to create, read, update, and delete repositories, pull requests, issues, and other resources on the GitHub platform.

Another example of a RESTful Web Service is the Twitter API. The Twitter API enables developers to access and manipulate data on the Twitter platform using RESTful Web Services. Developers can use the API to create, read, update, and delete tweets, direct messages, user profiles, and other resources on the Twitter platform.

There are some popular RESTful API in Vietnam:

VnExpress API: provides services such as getting news, getting weather information, getting stock information, getting gold price information, getting petrol price information.

ZaloPay API: provides online payment services to business partners.

ViettelPay's API: provides online payment services to business partners.

Momo's API: provides online payment services to business partners.

In summary, RESTful Web Services are a popular choice for modern web development due to their simplicity, scalability, and flexibility. They are composed of several components, including resources, URIs, HTTP methods, representations, and hypermedia links. There are many examples of RESTful Web Services in the real world, including social networking sites, e-commerce sites, and online marketplaces.

CHAPTER 5: COMMON ISSUE

Here are some common issues and concerns that developers need to be aware of when working with Web Services:

5.1 Security

Security is a critical aspect of Web Service development. RESTful services are vulnerable to a variety of attacks, including SQL injection, cross-site scripting, and denial-of-service attacks. In this chapter, we will discuss the security issues that developers must consider when designing, developing, and deploying RESTful services. We will discuss the best practices for securing RESTful services, including using SSL encryption, validating user input, and using authentication and authorization mechanisms.

Ensuring the security of data transmitted between different applications is a major concern when using Web Services. Developers need to implement proper authentication and authorization mechanisms to prevent unauthorized access to sensitive data.

5.2 Performance

Web Services can add overhead to the communication between different applications, which can affect the performance of the system. Developers need to optimize their Web Services to minimize this overhead and ensure that the system performs efficiently.

5.3 Reliability

Ensuring the reliability of Web Services is crucial, as any downtime or errors can affect the functionality of the entire system. Developers need to implement proper error handling and recovery mechanisms to ensure that the system can recover from any failures.

5.4 Interoperability

Web Services need to be able to communicate with different applications built using different languages and frameworks. Ensuring interoperability between different systems can be challenging and requires careful planning and design.

5.5 Scalability

As the usage of a system grows, its Web Services need to be able to handle an increasing number of requests. Developers need to design their Web Services with scalability in mind to ensure that they can handle large volumes of traffic without affecting performance.

CHAPTER 6: IMPLEMENTATION

Creating a RESTful service using PHP-MySQL involves several steps, including designing the database schema, writing the API code, and testing the API. This chapter will describe the process of creating a RESTful service using PHP-MySQL in detail. It also discusses the best practices for designing the database schema, writing the API code, and testing the API, as well as the tools and frameworks available for this purpose. We will also provide examples of RESTful services created using PHP-MySQL. In this project use RESTful web services to management shop have function as: register, login, add product, edit product, delete product, add product to cart, remove product in cart, payment using E-Wallet.

6.1 Explain demo web services:

Example built system shop management RESTful API. These are main APIs:

- Auth API: register, login.
- Product API: add, edit, delete product.
- Cart API: add product to card, remove product from cart.
- Payment API: payment using e-wallet.

6.1.1 Register

Path: /register

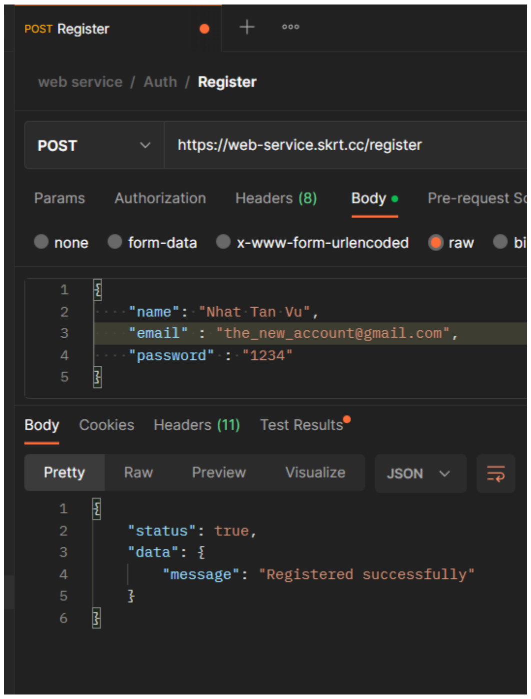
Method: POST

Body json: name, email, password

Description: User register an account

How it works:

- Validate the body json
- Check if there is any user register that email before, if exist, return error
- Create user then response success.



POST https://web-service.skrt.cc/register

```
{
    "name": "Nhat Tan Vu",
    "email" : "the_new_account@gmail.com",
    "password" : "1234"
}
```

```
{
    "status": true,
    "data": {
        "message": "Registered successfully"
    }
}
```

Figure 2. API register

6.1.2 Login

Path: /login

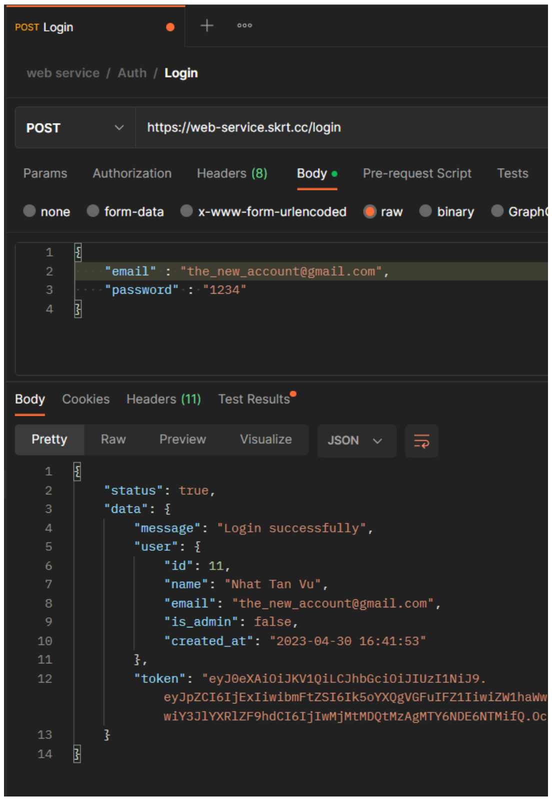
Method: POST

Body json: email, password

Description: User login to the system

How it works:

- Validate the body json
- Check if any user match with that email, if not return error
- Verify the password, if not match, return error
- Generate a json web token and save to database
- Response success with user data and token



POST Login

web service / Auth / Login

POST https://web-service.skrt.cc/login

```
{
    "email" : "the_new_account@gmail.com",
    "password" : "1234"
}
```

```
{
    "status": true,
    "data": {
        "message": "Login successfully",
        "user": {
            "id": 11,
            "name": "Nhat Tan Vu",
            "email": "the_new_account@gmail.com",
            "is_admin": false,
            "created_at": "2023-04-30 16:41:53"
        },
        "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
            eyJpZCI6IjExIiwibmFtZSI6Ik5oYXQgVGFuIFZ1IiwiZW1haWw
            wiY3JlYXRlZF9hdCI6IjIwMjMtMDQtMzAgMTY6NDE6NTMifQ.Oc
    }
}
```

Figure 3. API login

6.1.3 Get products.

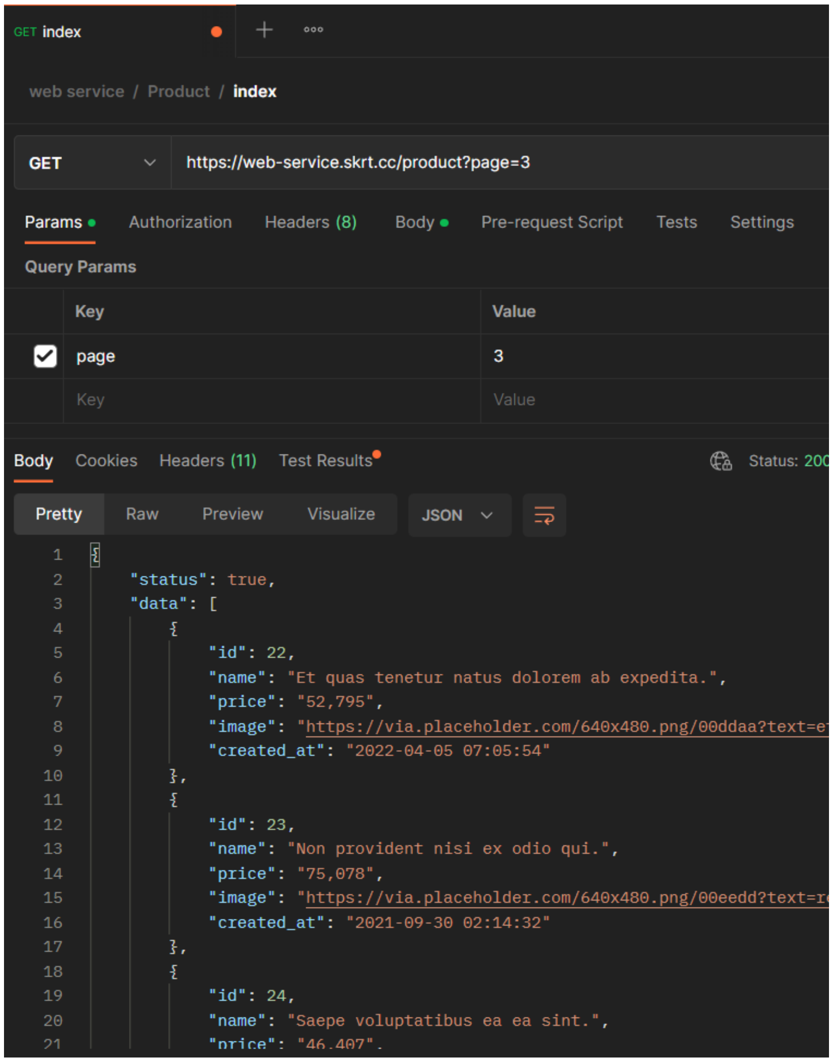
Path: /product

Method: GET

Description: Get all products with pagination

How it works:

- Select all the product with skip and take depends on query params page argument then response data



GET https://web-service.skrt.cc/product?page=3

Key	Value
page	3

Status: 20

```
{
    "status": true,
    "data": [
        {
            "id": 22,
            "name": "Et quas tenetur natus dolorem ab expedita.",
            "price": "52,795",
            "image": "https://via.placeholder.com/640x480.png/00ddaa?text=e
            "created_at": "2022-04-05 07:05:54"
        },
        {
            "id": 23,
            "name": "Non provident nisi ex odio qui.",
            "price": "75,078",
            "image": "https://via.placeholder.com/640x480.png/00eedd?text=r
            "created_at": "2021-09-30 02:14:32"
        },
        {
            "id": 24,
            "name": "Saepe voluptatibus ea ea sint.",
            "price": "46,407",
```

Figure 4. API Get Products

6.1.4 Show product

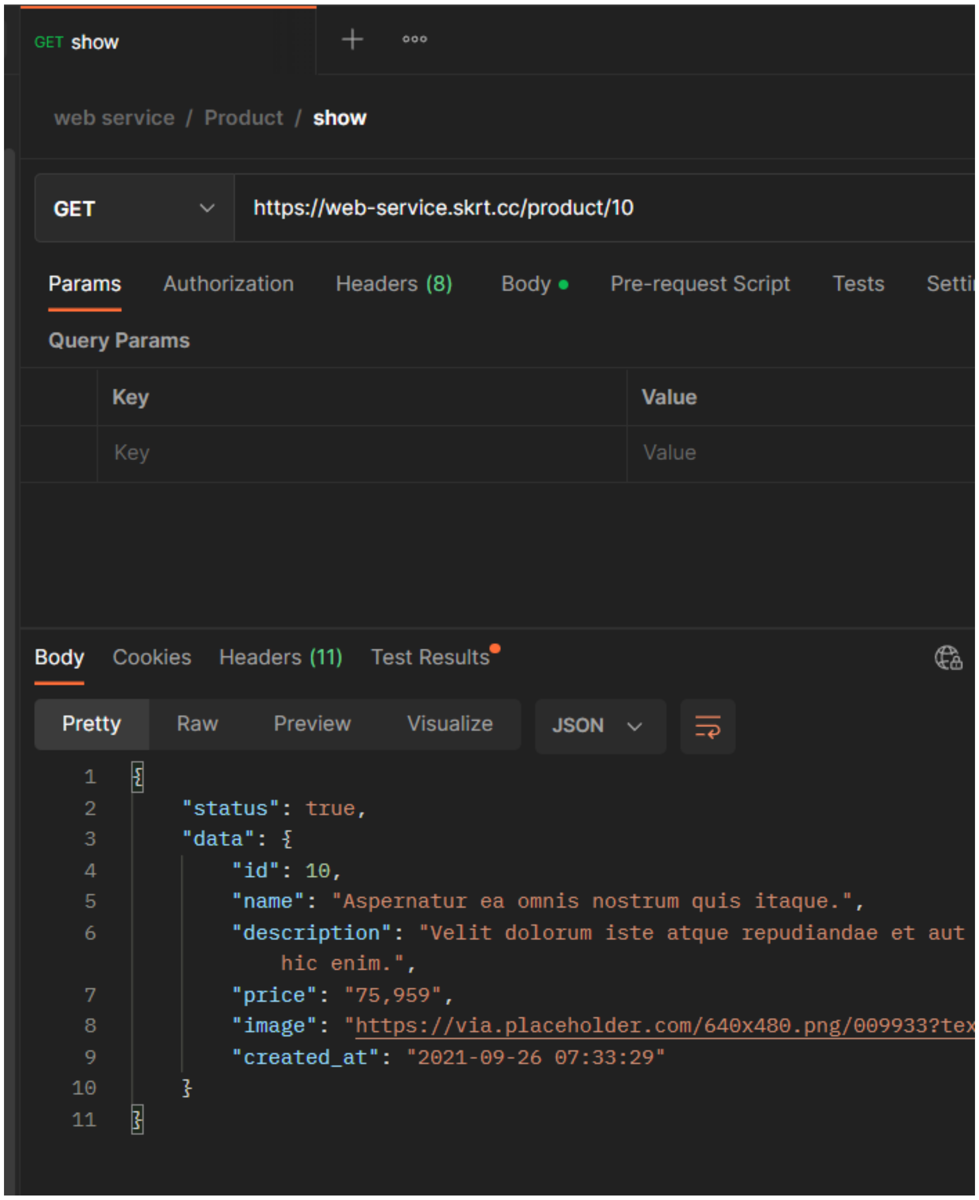
Path: /product/{id}

Method: GET

Description: Show information about a product

How it works:

- Select the product where id provided, if it doesn't exist, return 404



GET https://web-service.skrt.cc/product/10

```
{
    "status": true,
    "data": {
        "id": 10,
        "name": "Aspernatur ea omnis nostrum quis itaque.",
        "description": "Velit dolorum iste atque repudiandae et aut
            hic enim.",
        "price": "75,959",
        "image": "https://via.placeholder.com/640x480.png/009933?tex
        "created_at": "2021-09-26 07:33:29"
    }
}
```

Figure 5. API Get Product with ID

6.1.5 Store product

Path: /product

Method: POST

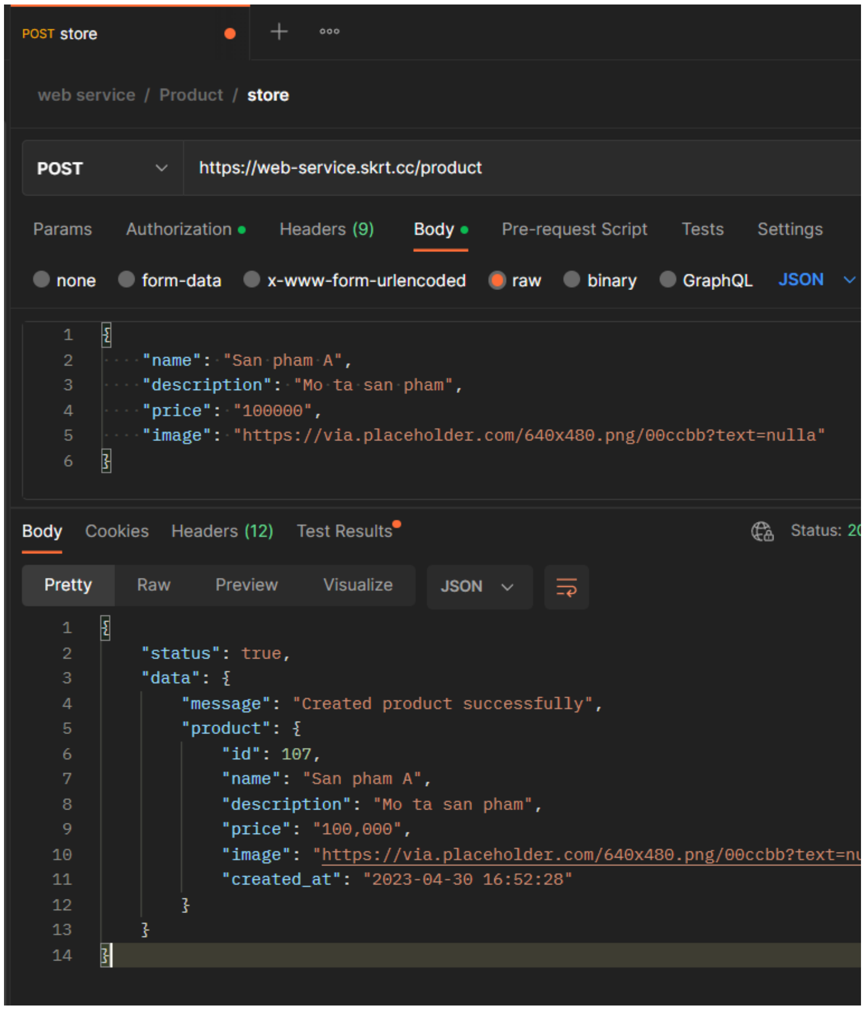
Body json: name, description, price, image

Headers: Authorization:Bearer {token}

Description: Create a product

How it works:

- Validate the body json
- Save the product to database and response success with product's data.



POST https://web-service.skrt.cc/product

```
{
    "name": "San pham A",
    "description": "Mo ta san pham",
    "price": "100000",
    "image": "https://via.placeholder.com/640x480.png/00ccbb?text=nulla"
}
```

Status: 2

```
{
    "status": true,
    "data": {
        "message": "Created product successfully",
        "product": {
            "id": 107,
            "name": "San pham A",
            "description": "Mo ta san pham",
            "price": "100,000",
            "image": "https://via.placeholder.com/640x480.png/00ccbb?text=nu
            "created_at": "2023-04-30 16:52:28"
        }
    }
}
```

Figure 6. API Create Product

6.1.6 Update product

Path: /product/{id}

Method: PUT

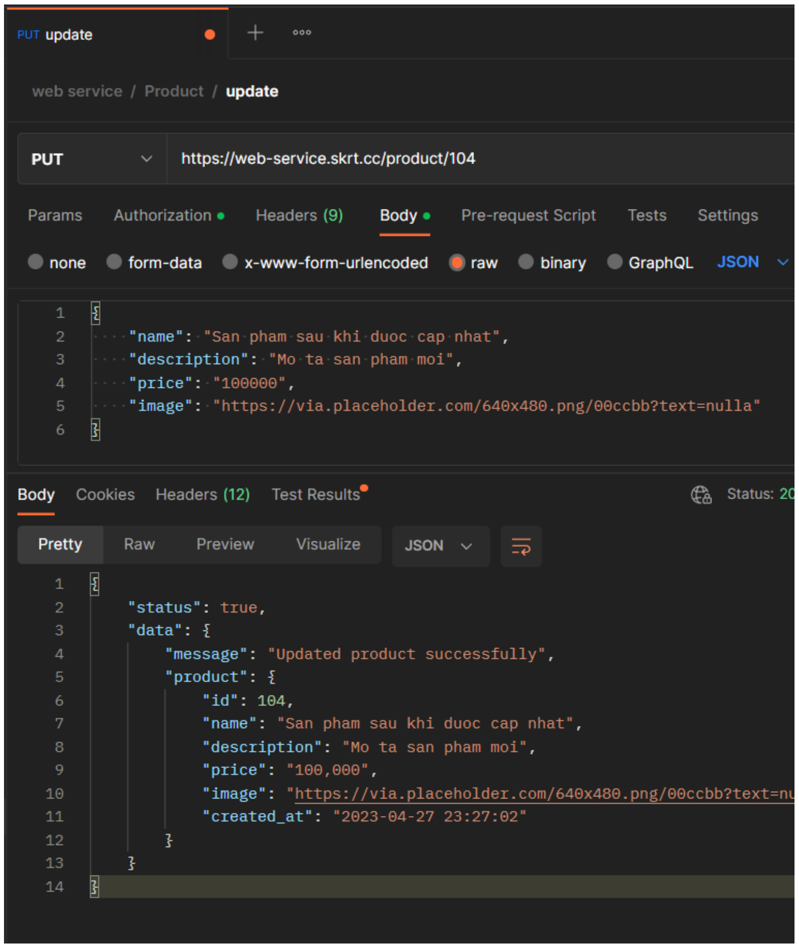
Body json: name, description, price, image

Headers: Authorization:Bearer {token}

Description: Update a product by id

How it works:

- Validate the body json
- Find that product, if it does not exist, return 404
- Update the product and response new product data



PUT https://web-service.skrt.cc/product/104

```
{
    "name": "San pham sau khi duoc cap nhat",
    "description": "Mo ta san pham moi",
    "price": "100000",
    "image": "https://via.placeholder.com/640x480.png/00ccbb?text=nulla"
}
```

```
{
    "status": true,
    "data": {
        "message": "Updated product successfully",
        "product": {
            "id": 104,
            "name": "San pham sau khi duoc cap nhat",
            "description": "Mo ta san pham moi",
            "price": "100,000",
            "image": "https://via.placeholder.com/640x480.png/00ccbb?text=nu
            "created_at": "2023-04-27 23:27:02"
        }
    }
}
```

Figure 7. API Edit Product

6.1.7 Delete product.

Path: /product/{id}

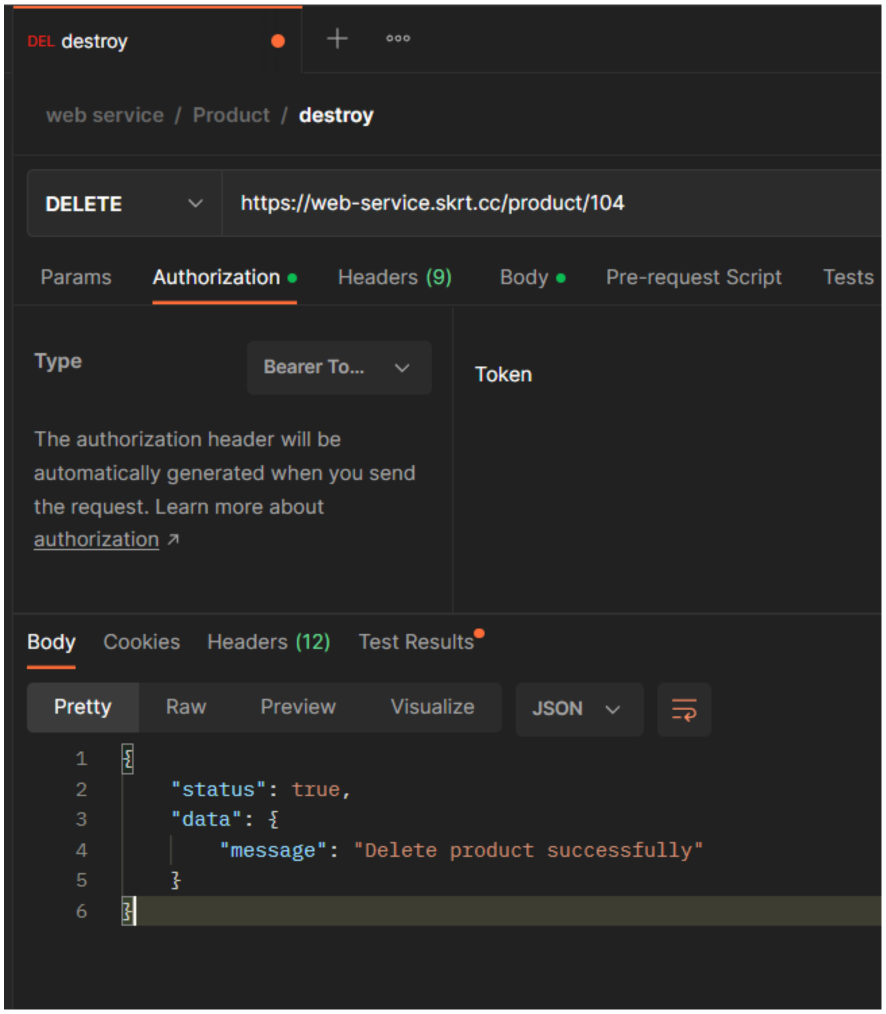
Method: DELETE

Headers: Authorization:Bearer {token}

Description: Delete a product by id

How it works:

- Find that product, if it does not exist, return 404.
- Try to delete it, if it has relation to orders, return error.
- Delete the product and response success.



https://web-service.skrt.cc/product/104

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

```
{
    "status": true,
    "data": {
        "message": "Delete product successfully"
    }
}
```

Figure 8. API Delete Product

6.1.8 Get cart

Path: /cart

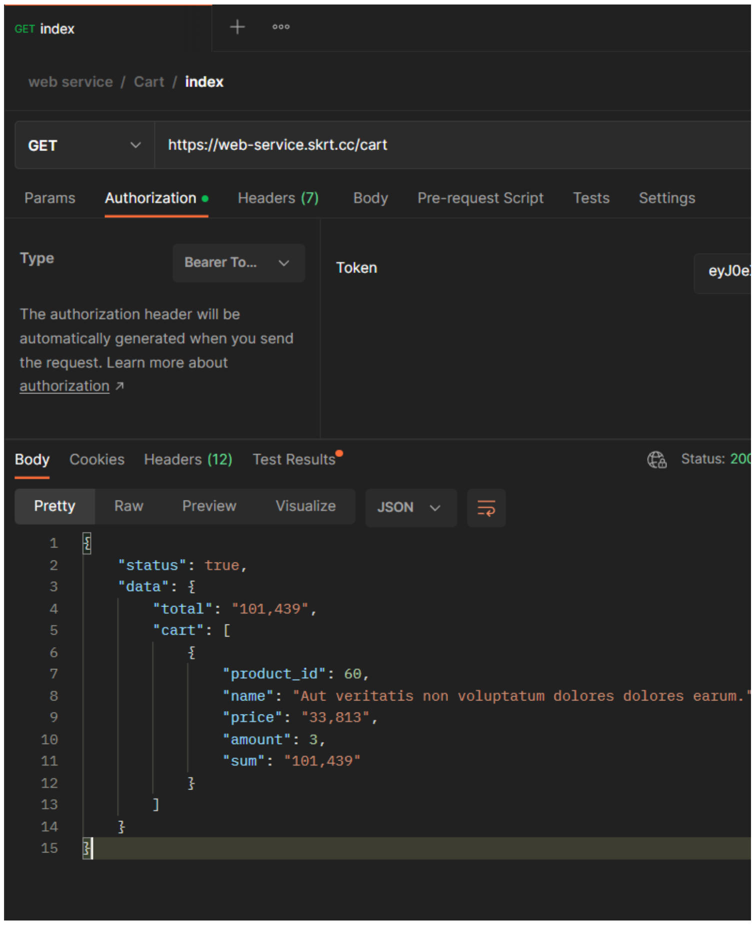
Method: GET

Headers: Authorization:Bearer {token}

Description: Get a cart include products with its amount and price

How it works:

- Select from order where user ID is match with the user requested and order status is In cart, if does not exist, create one to orders table.
- Select the order detail by order id from above then calculate the total.
- Response total and products in the cart.



https://web-service.skrt.cc/cart

Type: Bearer To...

Token: eyJ0e

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

```
{
    "status": true,
    "data": {
        "total": "101,439",
        "cart": [
            {
                "product_id": 60,
                "name": "Aut veritatis non voluptatum dolores dolores earum."
                "price": "33,813",
                "amount": 3,
                "sum": "101,439"
            }
        ]
    }
}
```

Figure 9. API Get information of Cart

6.1.9 Update cart

Path: /cart

Method: PUT

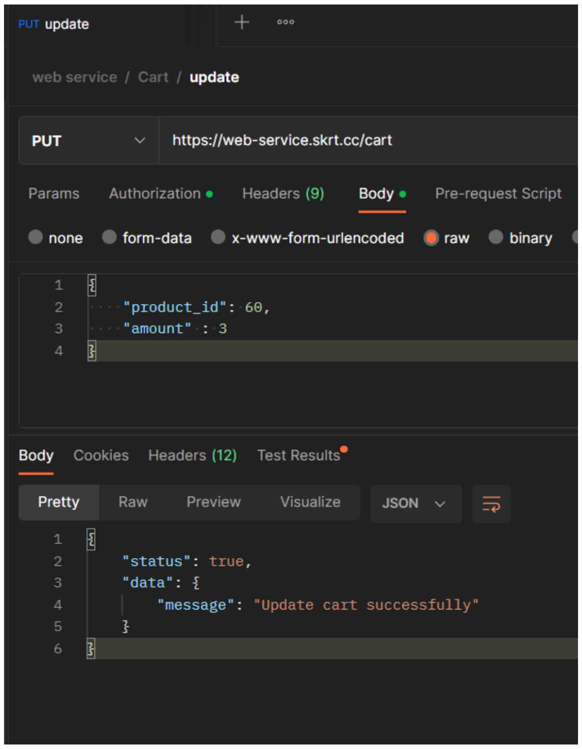
Headers: Authorization:Bearer {token}

Body json: order_id, amount

Description: Add product to cart, remove product from cart, change the product amount

How it works:

- Get the cart
- If the amount is 0, delete that product out of cart, else update that product's amount
- Update order detail to database
- Response success



https://web-service.skrt.cc/cart

```
{
    "product_id": 60,
    "amount" : 3
}
```

```
{
    "status": true,
    "data": {
        "message": "Update cart successfully"
    }
}
```

Figure 10. API Update Cart

6.1.10 Payment

Path: /cart

Method: POST

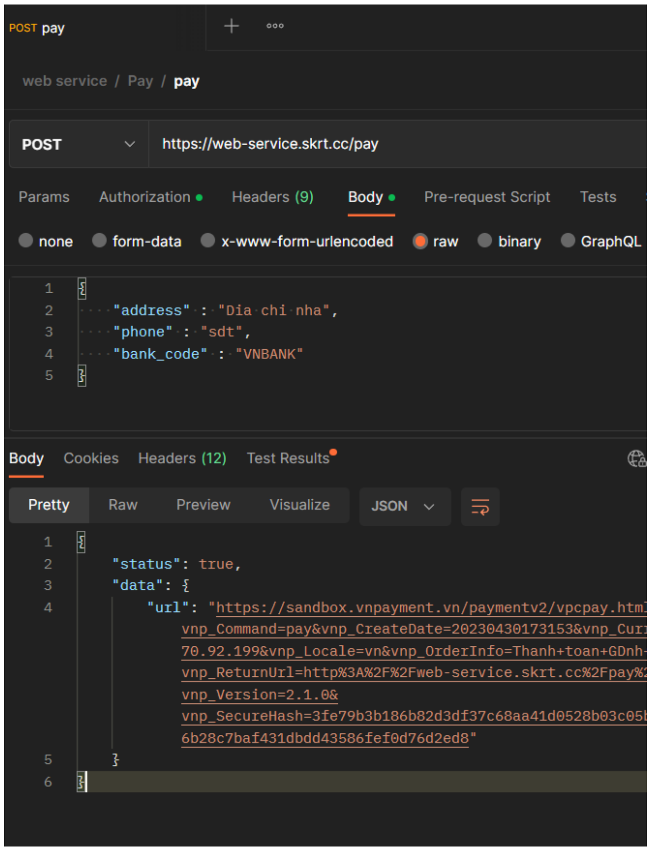
Headers: Authorization:Bearer {token}

Body json: address, phone, bank_code

Description: Get the payment url

How it works:

- Validate the body json
- Get the cart, if the cart value is 0, return error no thing in cart
- Update the order information
- Create vnpay payment url then response



POST https://web-service.skrt.cc/pay

```
{
    "address" : "Dia chi nha",
    "phone" : "sdt",
    "bank_code" : "VNBANK"
}
```

```
{
    "status": true,
    "data": {
        "url": "https://sandbox.vnpayment.vn/paymentv2/vpcpay.htm
            vnp_Command=pay&vnp_CreateDate=20230430173153&vnp_Cur
            70.92.199&vnp_Locale=vn&vnp_OrderInfo=Thanh+toan+GDnh
            vnp_ReturnUrl=http%3A%2F%2Fweb-service.skrt.cc%2Fpay%
            vnp_Version=2.1.0&
            vnp_SecureHash=3fe79b3b186b82d3df37c68aa41d0528b03c05
            6b28c7baf431dbdd43586fef0d76d2ed8"
    }
}
```

Figure 11. API Get Payment URL

Payment process:

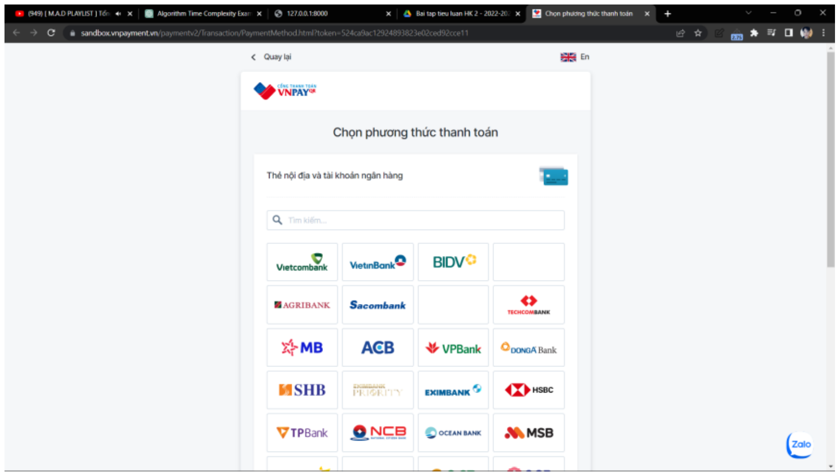
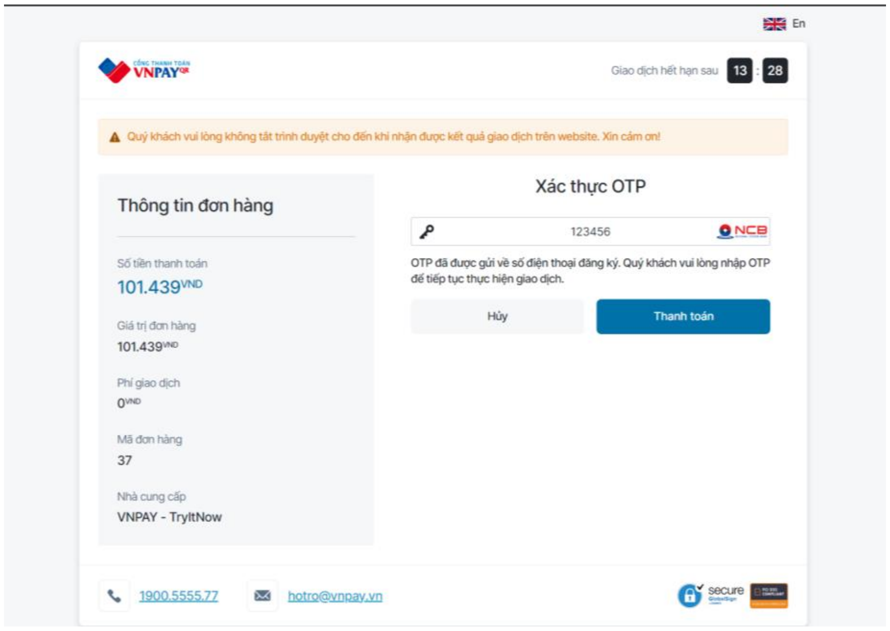


Figure 12. VNPay Payment Interface


Quý khách vui lòng không tắt trình duyệt cho đến khi nhận được kết quả giao dịch trên website. Xin cảm ơn!

Thông tin đơn hàng

Số tiền thanh toán: 101.439 VND
Giá trị đơn hàng: 101.439 VND
Phí giao dịch: 0 VND
Mã đơn hàng: 37
Nhà cung cấp: VNPAY - TryItNow

Xác thực OTP

123456

OTP đã được gửi về số điện thoại đăng ký. Quý khách vui lòng nhập OTP để tiếp tục thực hiện giao dịch.

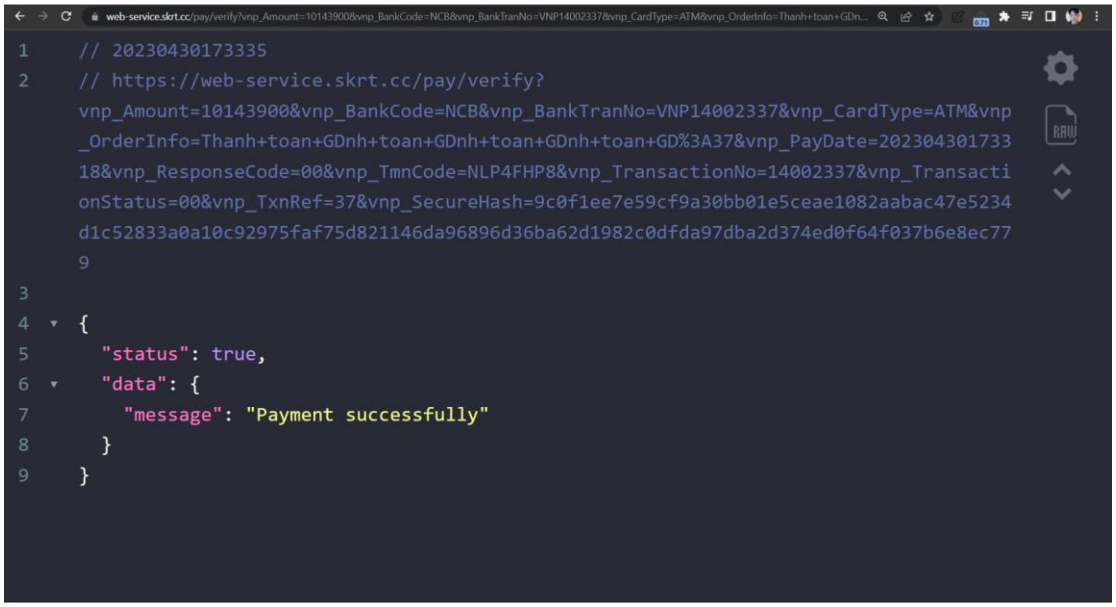
Figure 13. VnPay Confirm payment OPT

6.1.11 Verify Payment

Path: /pay/verify

Method: GET

Description: Verify the payment, this is called by VNPAY



web-service.skrt.cc/pay/verify?vnp_Amount=10143900&vnp_BankCode=NCB&vnp_BankTranNo=VNP14002337&vnp_CardType=ATM&vnp_OrderInfo=Thanh+toan+GDn...

```
// 20230430173335
// https://web-service.skrt.cc/pay/verify?
vnp_Amount=10143900&vnp_BankCode=NCB&vnp_BankTranNo=VNP14002337&vnp_CardType=ATM&vnp
_OrderInfo=Thanh+toan+GDnh+toan+GDnh+toan+GDnh+toan+GD%3A37&vnp_PayDate=202304301733
18&vnp_ResponseCode=00&vnp_TmnCode=NLP4FHP8&vnp_TransactionNo=14002337&vnp_Transacti
onStatus=00&vnp_TxnRef=37&vnp_SecureHash=9c0f1ee7e59cf9a30bb01e5ceae1082aabac47e5234
d1c52833a0a10c92975faf75d821146da96896d36ba62d1982c0dfda97dba2d374ed0f64f037b6e8ec77
9

{
  "status": true,
  "data": {
    "message": "Payment successfully"
  }
}
```

Figure 14. Verify Payment Status

6.1.12 Get orders:

Path: /order

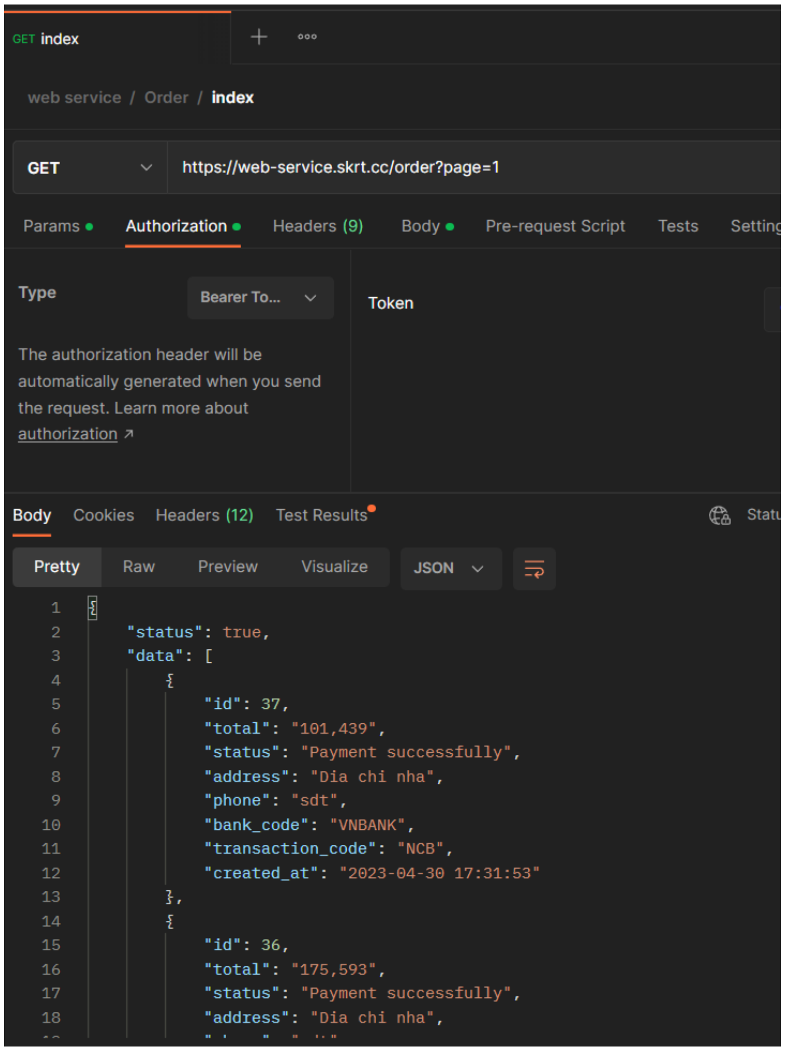
Method: GET

Description: Get all orders with pagination.

Headers: Authorization:Bearer {token}

How it works:

- Select all the orders where user id matched the jwt with skip and take depends on query params page argument then response data.



https://web-service.skrt.cc/order?page=1

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

```
{
    "status": true,
    "data": [
        {
            "id": 37,
            "total": "101,439",
            "status": "Payment successfully",
            "address": "Dia chi nha",
            "phone": "sdt",
            "bank_code": "VNBANK",
            "transaction_code": "NCB",
            "created_at": "2023-04-30 17:31:53"
        },
        {
            "id": 36,
            "total": "175,593",
            "status": "Payment successfully",
            "address": "Dia chi nha",
```

Figure 15. API Get all orders with pagination

6.1.13 Show orders

Path: /order/{id}

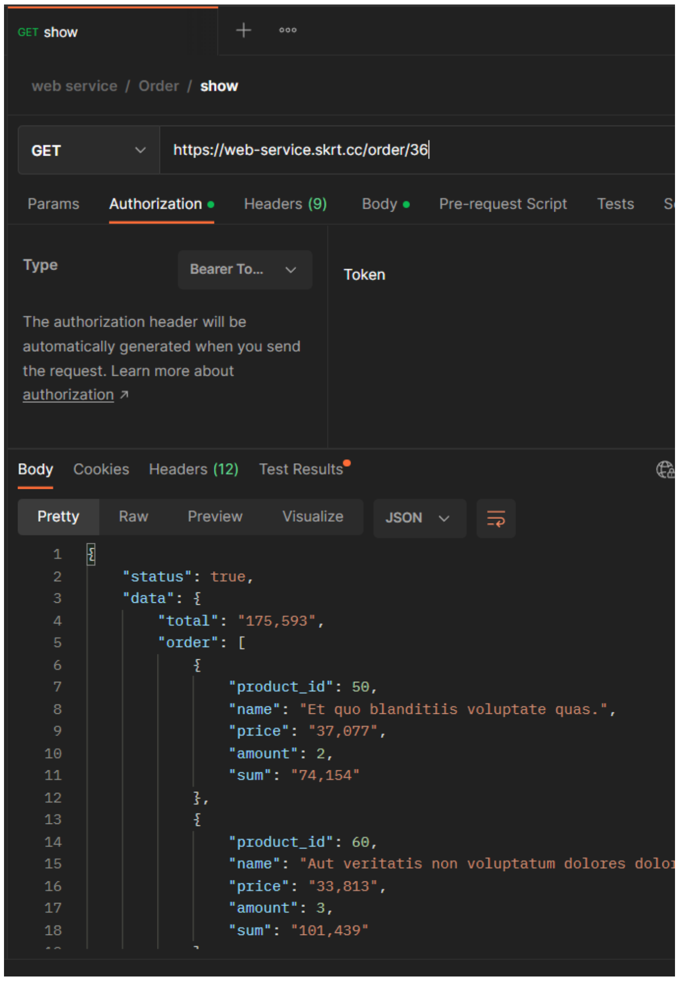
Method: GET

Description: Show information about an order

Headers: Authorization:Bearer {token}

How it works:

- Find the order by id, if it does not exist, return 404.
- Select the order detail, calculate the total then return data.



https://web-service.skrt.cc/order/36

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

```
{
    "status": true,
    "data": {
        "total": "175,593",
        "order": [
            {
                "product_id": 50,
                "name": "Et quo blanditiis voluptate quas.",
                "price": "37,077",
                "amount": 2,
                "sum": "74,154"
            },
            {
                "product_id": 60,
                "name": "Aut veritatis non voluptatum dolores dolo
                "price": "33,813",
                "amount": 3,
                "sum": "101,439"
```

Figure 16. API Show information about an order

6.2 Demo webservices on hosting

We use **Linux** server and **Apache** to deploy the service, you can try immediately at https://web-service.skrt.cc/

CHAPTER 7: CONCLUSION

Throughout this essay, we have explored the concepts of web services and RESTful APIs, and their significance and application in web development. A web service is a software system designed to support interoperable machine-to-machine interaction over a network. We also explain how to deploy RESTful APIs, and test it using tools such as Postman.

In summary, RESTful APIs are an important part of modern web development, enabling developers to create scalable and flexible web services that can be easily consumed by other systems.

REFERENCES

[1] GeeksforGeeks. (2023). What are Web Services? [Online]. Available: https://www.geeksforgeeks.org/what-are-web-services/ [Accessed: April 27, 2023]

[2] Juviler, J. (2023). Web Service vs. API, Explained [Online]. Available: https://blog.hubspot.com/website/web-services-vs-api [Accessed: April 28, 2023].