Example 01

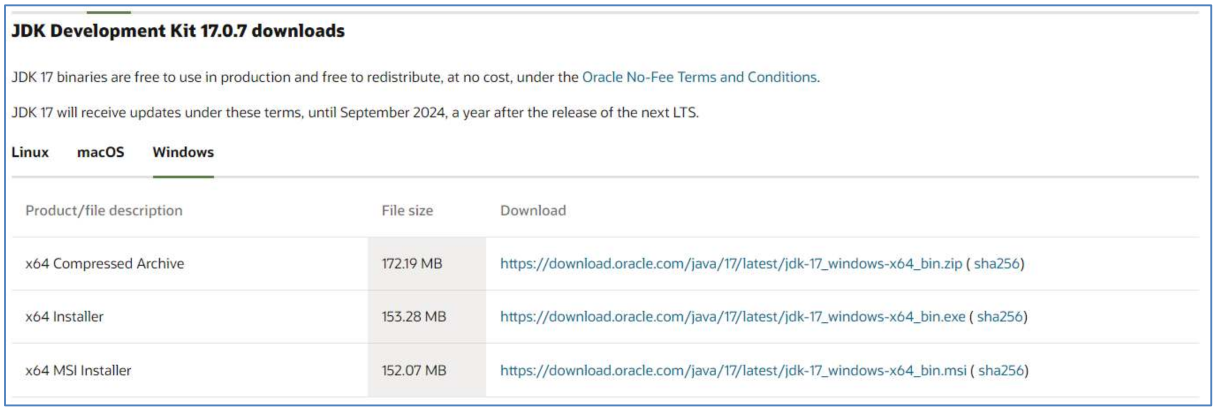
Mục tiêu: Thiết lập môi trường lập trình Java Spring Boot và tạo project

Yêu cầu:

- ✔ Cài đặt JDK và Visual Studio Code
- ✔ Tạo project có tên project là **example01**

Hướng dẫn:

Bước 1: Cài đặt JDK và Visual Studio Code



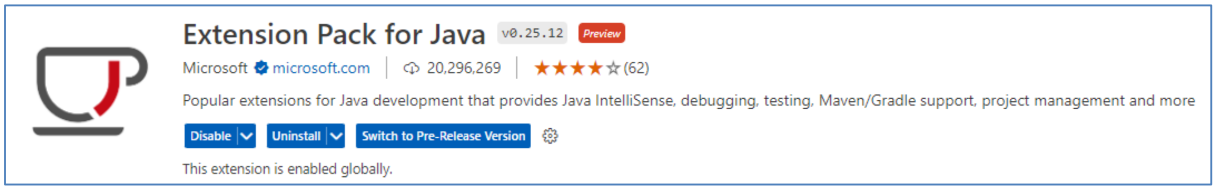
JDK Development Kit 17.0.7 downloads

JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions.

JDK 17 will receive updates under these terms, until September 2024, a year after the release of the next LTS.

Product/file description	File size	Download
x64 Compressed Archive	172.19 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256)
x64 Installer	153.28 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256)
x64 MSI Installer	152.07 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256)

Bước 2: Để cài đặt Extension Java cho Visual Studio Code bạn hãy thực hiện Mở Extensions (Ctrl+Shift+X), tìm kiếm **java**



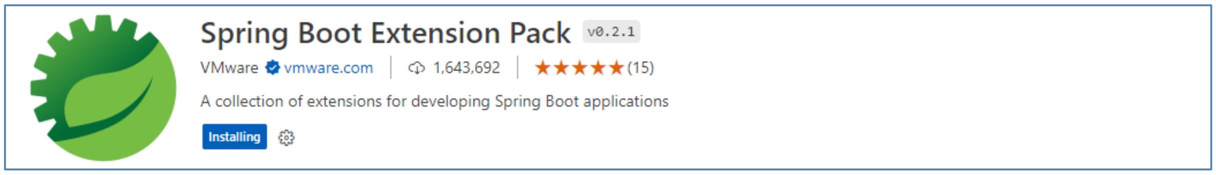
Extension Pack for Java v0.25.12 Preview

Microsoft microsoft.com | 20,296,269 | ★★★★☆ (62)

Popular extensions for Java development that provides Java IntelliSense, debugging, testing, Maven/Gradle support, project management and more

This extension is enabled globally.

Bước 3: Để cài đặt Extension Spring Boot cho Visual Studio Code bạn hãy thực hiện Mở Extensions (Ctrl+Shift+X), tìm kiếm **vscode-spring-initializr**



Spring Boot Extension Pack v0.2.1

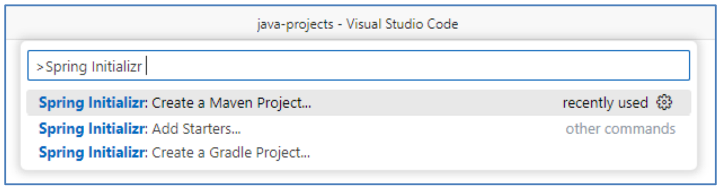
VMware vmware.com | 1,643,692 | ★★★★★ (15)

A collection of extensions for developing Spring Boot applications

Bước 4: Sử dụng Visual Studio Code tạo project **Maven** với thông tin như sau:

- ✓ Spring Boot version: **3.1.1**
- ✓ Language: **Java**
- ✓ Group Id: **com.nguyenanhtu**
- ✓ Artifact Id: **example01**
- ✓ Packaging type: **Jar**
- ✓ Java Version: **17**
- ✓ Dependencies: **Selected 0 dependencies**
- ✓ Vị trí project: **D:\java-projects**

Hãy mở Bảng lệnh (Ctrl+Shift+P) và nhập **Spring Initializr** để bắt đầu tạo dự án **Maven**

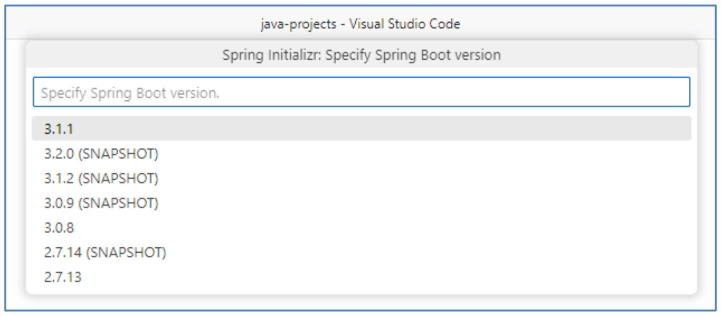

java-projects - Visual Studio Code

>Spring Initializr

Spring Initializr: Create a Maven Project... recently used

Spring Initializr: Add Starters... other commands

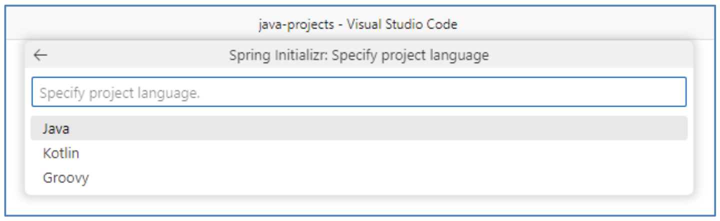
Spring Initializr: Create a Gradle Project...


java-projects - Visual Studio Code

Spring Initializr: Specify Spring Boot version

Specify Spring Boot version.

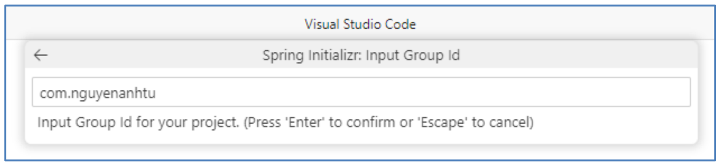
3.1.1
3.2.0 (SNAPSHOT)
3.1.2 (SNAPSHOT)
3.0.9 (SNAPSHOT)
3.0.8
2.7.14 (SNAPSHOT)
2.7.13


java-projects - Visual Studio Code

Spring Initializr: Specify project language

Specify project language.

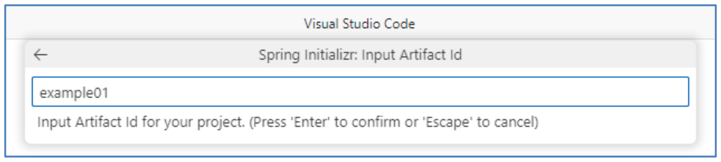
Java
Kotlin
Groovy


Visual Studio Code

Spring Initializr: Input Group Id

com.nguyenanhtu

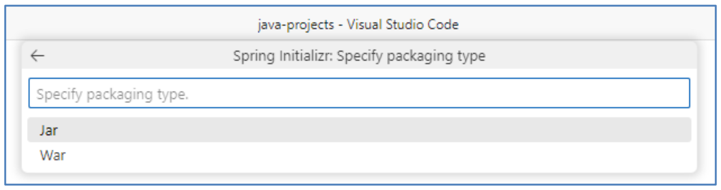
Input Group Id for your project. (Press 'Enter' to confirm or 'Escape' to cancel)


Visual Studio Code

Spring Initializr: Input Artifact Id

example01

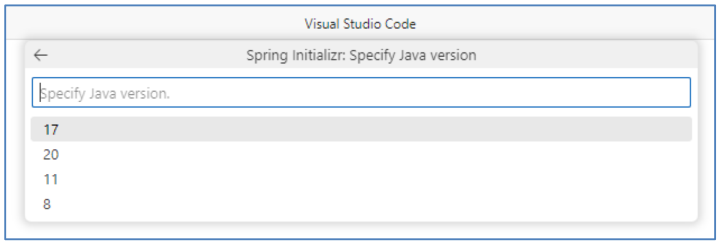
Input Artifact Id for your project. (Press 'Enter' to confirm or 'Escape' to cancel)


java-projects - Visual Studio Code

Spring Initializr: Specify packaging type

Specify packaging type.

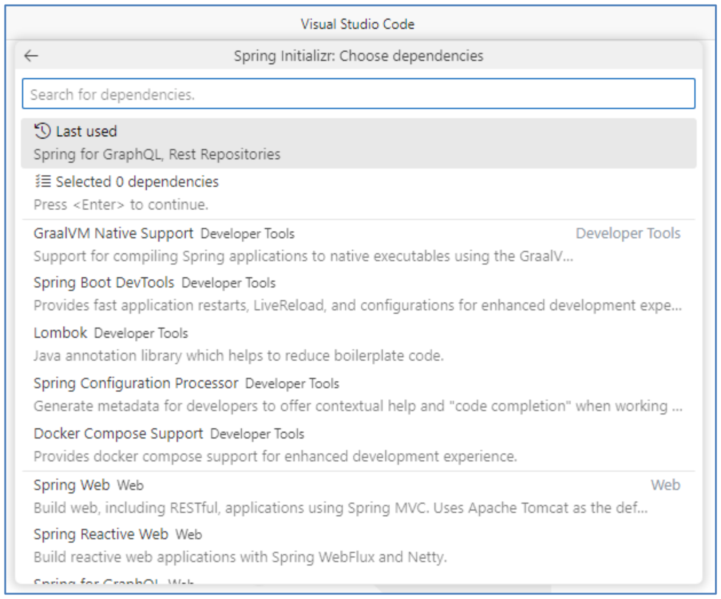
Jar
War


Visual Studio Code

Spring Initializr: Specify Java version

Specify Java version.

17
20
11
8


Visual Studio Code

Spring Initializr: Choose dependencies

Search for dependencies.

Last used
Spring for GraphQL, Rest Repositories

Selected 0 dependencies
Press <Enter> to continue.

GraalVM Native Support Developer Tools — Developer Tools
Support for compiling Spring applications to native executables using the GraalV...

Spring Boot DevTools Developer Tools
Provides fast application restarts, LiveReload, and configurations for enhanced development expe...

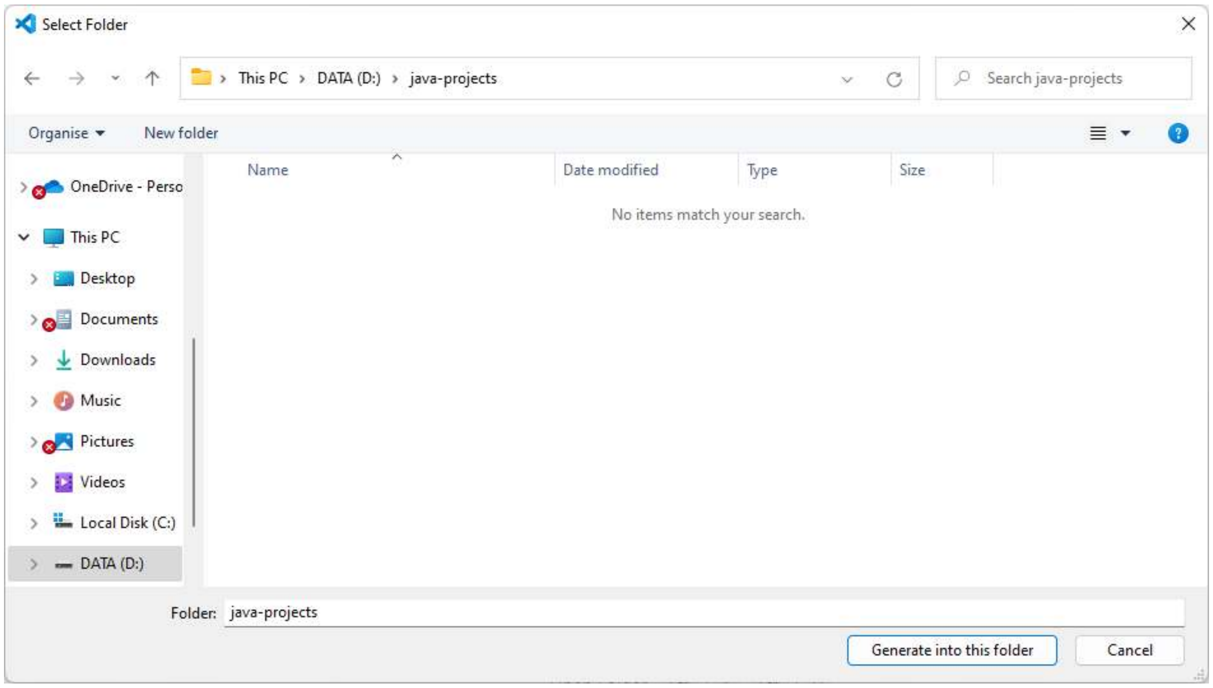
Lombok Developer Tools
Java annotation library which helps to reduce boilerplate code.

Spring Configuration Processor Developer Tools
Generate metadata for developers to offer contextual help and "code completion" when working ...

Docker Compose Support Developer Tools
Provides docker compose support for enhanced development experience.

Spring Web Web — Web
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the def...

Spring Reactive Web Web
Build reactive web applications with Spring WebFlux and Netty.


Select Folder

This PC > DATA (D:) > java-projects

Search java-projects

Organise ▾ New folder

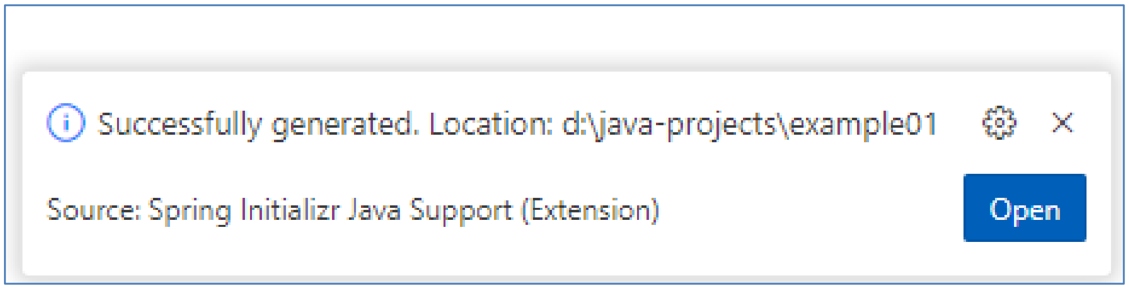
Name	Date modified	Type	Size

No items match your search.

OneDrive - Perso
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
DATA (D:)

Folder: java-projects

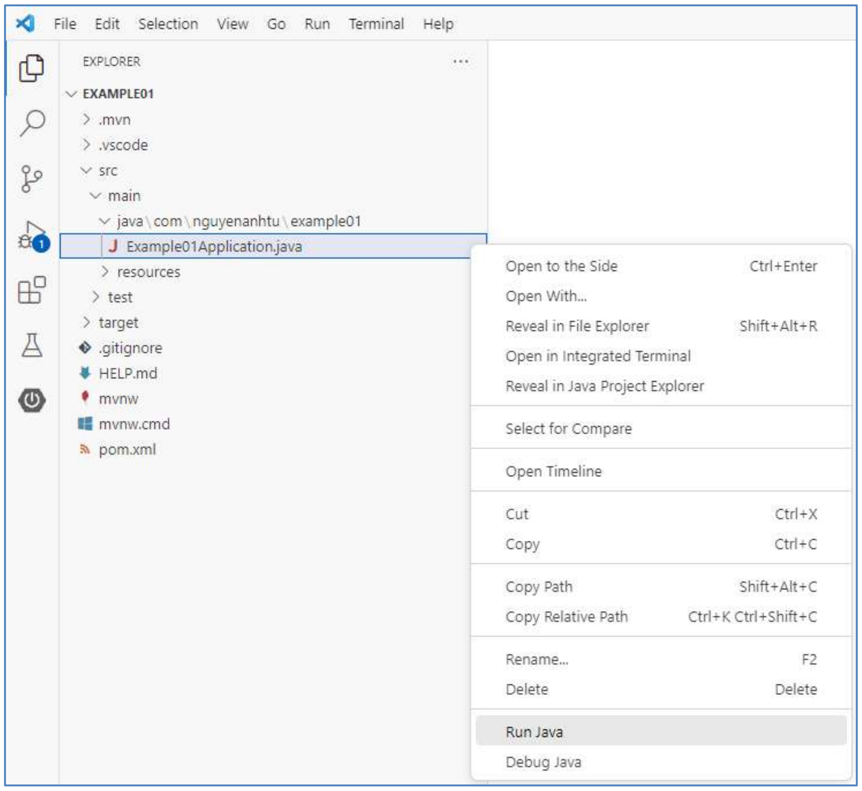
Generate into this folder | Cancel

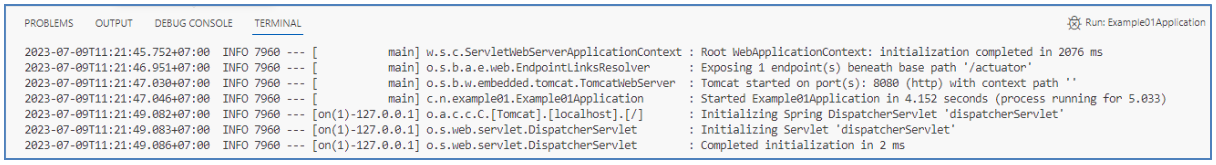

Successfully generated. Location: d:\java-projects\example01

Source: Spring Initializr Java Support (Extension)

Open

Bước 5: Click chuột phải vào lớp **Example01Application** sau đó Click vào **Run Java**.


File Edit Selection View Go Run Terminal Help
EXPLORER
EXAMPLE01
.mvn
.vscode
src
main
java \ com \ nguyenanhtu \ example01
Example01Application.java
resources
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
Open to the Side Ctrl+Enter
Open With...
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Reveal in Java Project Explorer
Select for Compare
Open Timeline
Cut Ctrl+X
Copy Ctrl+C
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename... F2
Delete Delete
Run Java
Debug Java


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Run: Example01Application

```
2023-07-09T11:21:45.752+07:00  INFO 7960 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2076 ms
2023-07-09T11:21:46.951+07:00  INFO 7960 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1 endpoint(s) beneath base path '/actuator'
2023-07-09T11:21:47.030+07:00  INFO 7960 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2023-07-09T11:21:47.046+07:00  INFO 7960 --- [           main] c.n.example01.Example01Application       : Started Example01Application in 4.152 seconds (process running for 5.033)
2023-07-09T11:21:49.082+07:00  INFO 7960 --- [on(1)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-07-09T11:21:49.083+07:00  INFO 7960 --- [on(1)-127.0.0.1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2023-07-09T11:21:49.086+07:00  INFO 7960 --- [on(1)-127.0.0.1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
```

Example 02

Mục tiêu: Tạo và quản lý ứng dụng **Spring Boot Rest API** sử dụng **Hibernate** kết nối cở sở dữ liệu **MySQL**.

Yêu cầu: Thực hiện các chức năng sau:

- ✓ Tạo bảng
- ✓ Thêm dữ liệu
- ✓ Xóa dữ liệu
- ✓ Sửa dữ liệu
- ✓ Truy vấn dữ liệu

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code tạo project với thông tin như sau:

- ✓ Spring Boot version: **3.1.1**
- ✓ Language: **Java**
- ✓ Group Id: **com.nguyenanhtu**
- ✓ Artifact Id: **example02**
- ✓ Packaging type: **Jar**
- ✓ Java Version: **17**
- ✓ Dependencies: **Spring Web, Spring Data JPA, MySQL Driver, Lombok**
- ✓ Vị trí project: **D:\java-projects**

Bước 2: Sửa file **application.properties** như sau:

src/main/resources/application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/example02
spring.datasource.username=root
spring.datasource.password=
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
```

Bước 3: Tao Entity **Product** như sau:

Product.java

```
package com.nguyenanhtu.example02.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "Products")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String title;
    @Column(nullable = false)
```

```java
    private String description;
    @Column(nullable = false, unique = true)
    private String photo;
    @Column(nullable = false)
    private double price;
}
```

Bước 4: Tạo interface cho **Product** Repository như sau:

ProductRepository.java

```java
package com.nguyenanhtu.example02.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.nguyenanhtu.example02.entity.Product;

public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

Bước 5: Tạo Interface cho **Product** Service như sau:

ProductService.java

```java
package com.nguyenanhtu.example02.service;

import java.util.List;
import com.nguyenanhtu.example02.entity.Product;

public interface ProductService {
    Product createProduct(Product product);
    Product getProductById(Long productId);
    List<Product> getAllProducts();
    Product updateProduct(Product product);
    void deleteProduct(Long productId);
}
```

Bước 6: Tao Service Layer cho **Product** như sau:

ProductServiceImpl.java

```java
package com.nguyenanhtu.example02.service.impl;

import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import com.nguyenanhtu.example02.entity.Product;
import com.nguyenanhtu.example02.service.ProductService;
import com.nguyenanhtu.example02.repository.ProductRepository;
import java.util.List;
import java.util.Optional;

@Service
@AllArgsConstructor
public class ProductServiceImpl implements ProductService {

    private ProductRepository productRepository;

    @Override
    public Product createProduct(Product product) {
        return productRepository.save(product);
    }

    @Override
    public Product getProductById(Long productId) {
        Optional<Product> optionalProduct = productRepository.findById(productId);
        return optionalProduct.get();
    }

    @Override
    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }

    @Override
    public Product updateProduct(Product product) {
        Product existingProduct = productRepository.findById(product.getId()).get();
        existingProduct.setTitle(product.getTitle());
        existingProduct.setDescription(product.getDescription());
        existingProduct.setPhoto(product.getPhoto());
        existingProduct.setPrice(product.getPrice());
        Product updatedProduct = productRepository.save(existingProduct);
        return updatedProduct;
    }

    @Override
    public void deleteProduct(Long productId) {
        productRepository.deleteById(productId);
    }
}
```

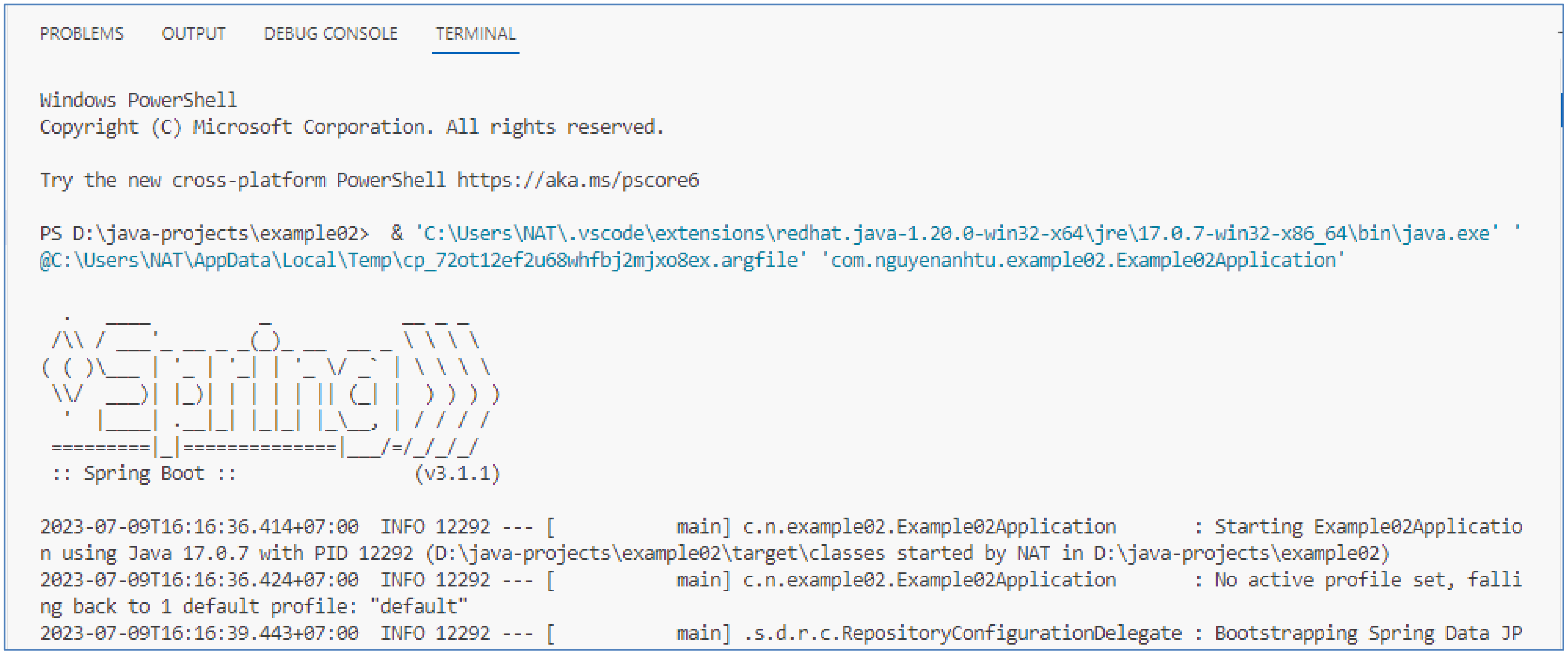
Bước 7: Tạo Controller cho **Product** như sau:

ProductController.java

```java
package com.nguyenanhtu.example02.controller;

import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import com.nguyenanhtu.example02.entity.Product;
import com.nguyenanhtu.example02.service.ProductService;
import java.util.List;

@RestController
```

```java
@AllArgsConstructor
@RequestMapping("api/products")
public class ProductController {

    private ProductService productService;

    // Create Product REST API
    @PostMapping
    public ResponseEntity<Product> createProduct(@RequestBody Product Product) {
        Product savedProduct = productService.createProduct(Product);
        return new ResponseEntity<>(savedProduct, HttpStatus.CREATED);
    }

    // Get Product by id REST API
    // http://localhost:8080/api/Products/1
    @GetMapping("{id}")
    public ResponseEntity<Product> getProductById(@PathVariable("id") Long ProductId) {
        Product Product = productService.getProductById(ProductId);
        return new ResponseEntity<>(Product, HttpStatus.OK);
    }

    // Get All Products REST API
    // http://localhost:8080/api/Products
    @GetMapping
    public ResponseEntity<List<Product>> getAllProducts() {
        List<Product> Products = productService.getAllProducts();
        return new ResponseEntity<>(Products, HttpStatus.OK);
    }

    // Update Product REST API
    @PutMapping("{id}")
    // http://localhost:8080/api/Products/1
    public ResponseEntity<Product> updateProduct(@PathVariable("id") Long ProductId,
            @RequestBody Product Product) {
        Product.setId(ProductId);
        Product updatedProduct = productService.updateProduct(Product);
        return new ResponseEntity<>(updatedProduct, HttpStatus.OK);
    }

    // Delete Product REST API
    @DeleteMapping("{id}")
    public ResponseEntity<String> deleteProduct(@PathVariable("id") Long ProductId) {
        productService.deleteProduct(ProductId);
        return new ResponseEntity<>("Product successfully deleted!", HttpStatus.OK);
    }
}
```

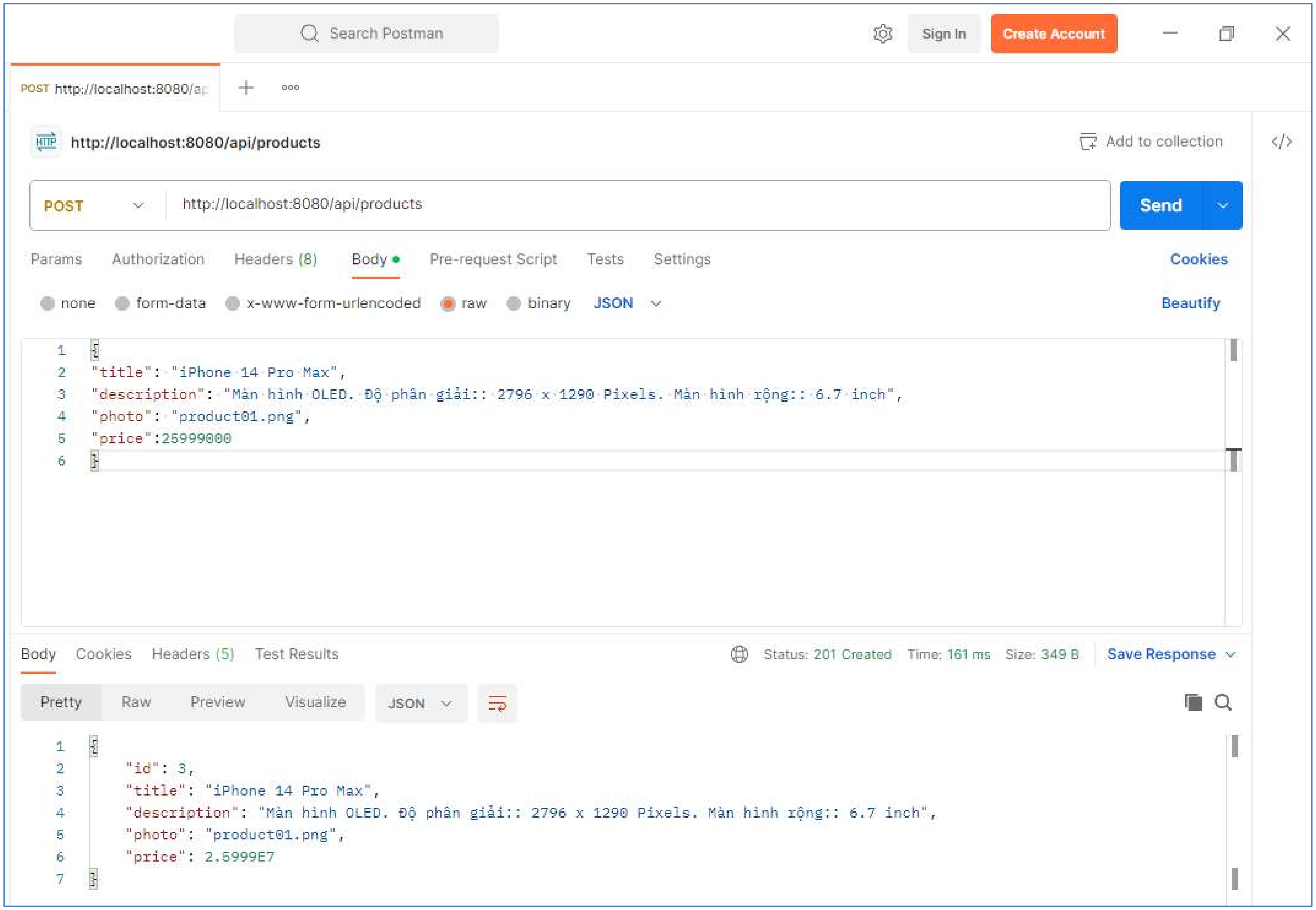
Bước 8: Click chuột phải vào lớp **Example02Application**, sau đó Click vào **Run Java**.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\java-projects\example02>  & 'C:\Users\NAT\.vscode\extensions\redhat.java-1.20.0-win32-x64\jre\17.0.7-win32-x86_64\bin\java.exe' '
@C:\Users\NAT\AppData\Local\Temp\cp_72ot12ef2u68whfbj2mjxo8ex.argfile' 'com.nguyenanhtu.example02.Example02Application'

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.1)

2023-07-09T16:16:36.414+07:00  INFO 12292 --- [           main] c.n.example02.Example02Application       : Starting Example02Applicatio
n using Java 17.0.7 with PID 12292 (D:\java-projects\example02\target\classes started by NAT in D:\java-projects\example02)
2023-07-09T16:16:36.424+07:00  INFO 12292 --- [           main] c.n.example02.Example02Application       : No active profile set, falli
ng back to 1 default profile: "default"
2023-07-09T16:16:39.443+07:00  INFO 12292 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JP
```

Bước 9: Sử dụng **Postman** kiểm tra các API



POST http://localhost:8080/api/products

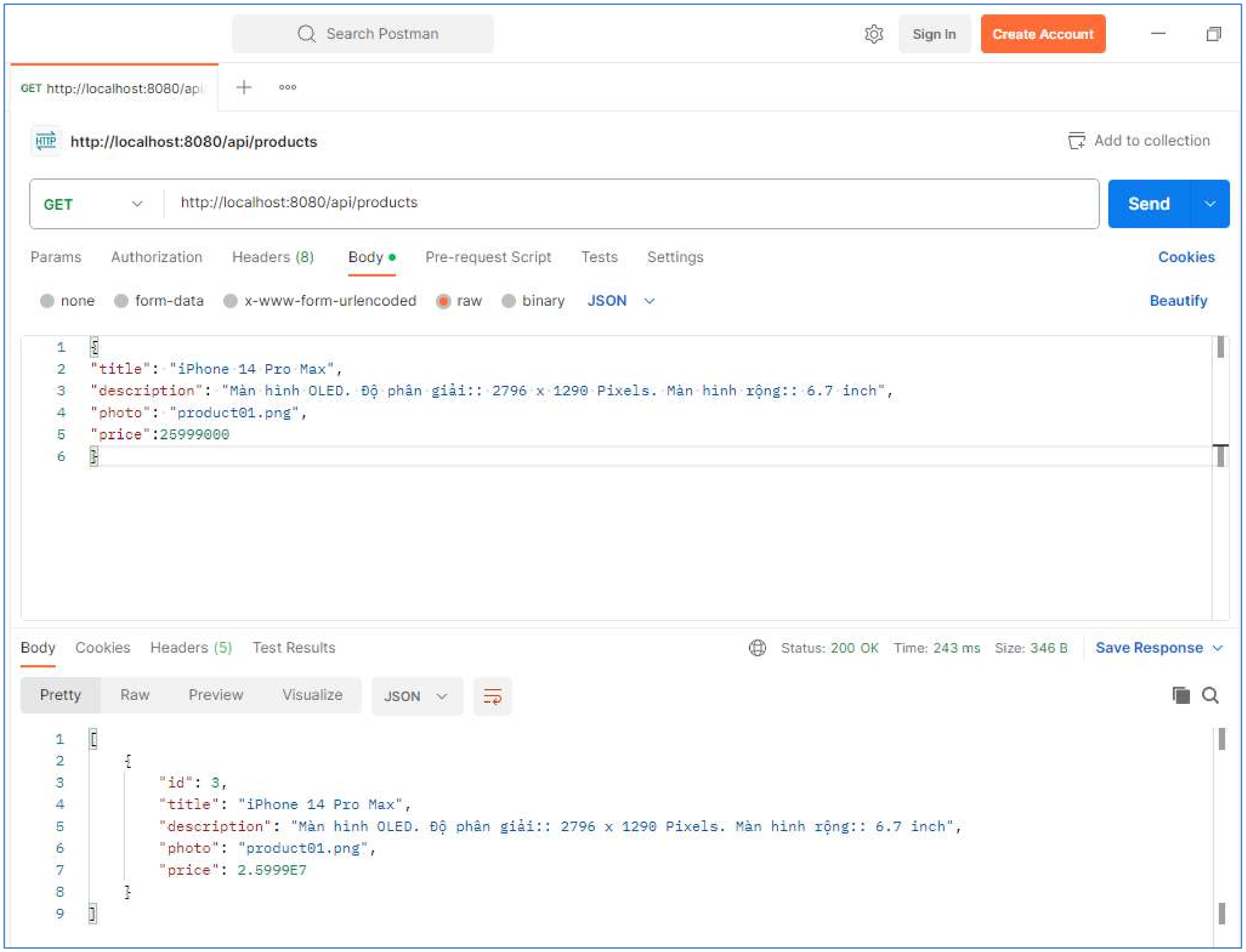
none form-data x-www-form-urlencoded [✓ raw] binary JSON

```
{
"title": "iPhone 14 Pro Max",
"description": "Màn hình OLED. Độ phân giải:: 2796 x 1290 Pixels. Màn hình rộng:: 6.7 inch",
"photo": "product01.png",
"price":25999000
}
```

Body Cookies Headers (5) Test Results — Status: 201 Created Time: 161 ms Size: 349 B

Pretty — JSON

```
{
    "id": 3,
    "title": "iPhone 14 Pro Max",
    "description": "Màn hình OLED. Độ phân giải:: 2796 x 1290 Pixels. Màn hình rộng:: 6.7 inch",
    "photo": "product01.png",
    "price": 2.5999E7
}
```



GET http://localhost:8080/api/products

```
{
"title": "iPhone 14 Pro Max",
"description": "Màn hình OLED. Độ phân giải:: 2796 x 1290 Pixels. Màn hình rộng:: 6.7 inch",
"photo": "product01.png",
"price":25999000
}
```

Body — Status: 200 OK Time: 243 ms Size: 346 B — JSON

```
[
    {
        "id": 3,
        "title": "iPhone 14 Pro Max",
        "description": "Màn hình OLED. Độ phân giải:: 2796 x 1290 Pixels. Màn hình rộng:: 6.7 inch",
        "photo": "product01.png",
        "price": 2.5999E7
    }
]
```

Exercise 01

Mục tiêu: Tạo và quản lý ứng dụng **Spring Boot Rest API** sử dụng **Hibernate** kết nối cở sở dữ liệu **MySQL**.

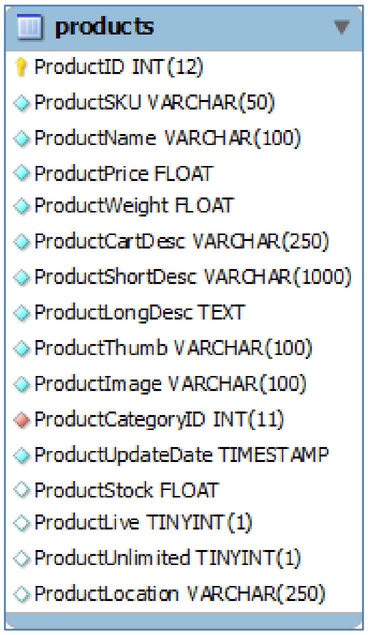
Yêu cầu: Thực hiện các chức năng sau:

- ✓ Tạo bảng
- ✓ Thêm dữ liệu
- ✓ Xóa dữ liệu
- ✓ Sửa dữ liệu
- ✓ Truy vấn dữ liệu

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code tạo project với thông tin như sau:

- ✓ Spring Boot version: **3.1.1**
- ✓ Language: **Java**
- ✓ Group Id: **com.nguyenanhtu**
- ✓ Artifact Id: **exercise01**
- ✓ Packaging type: **Jar**
- ✓ Java Version: **17**
- ✓ Dependencies: **Spring Web, Spring Data JPA, MySQL Driver, Lombok**
- ✓ Vị trí project: **D:\java-projects**



products
ProductID INT(12)
ProductSKU VARCHAR(50)
ProductName VARCHAR(100)
ProductPrice FLOAT
ProductWeight FLOAT
ProductCartDesc VARCHAR(250)
ProductShortDesc VARCHAR(1000)
ProductLongDesc TEXT
ProductThumb VARCHAR(100)
ProductImage VARCHAR(100)
ProductCategoryID INT(11)
ProductUpdateDate TIMESTAMP
ProductStock FLOAT
ProductLive TINYINT(1)
ProductUnlimited TINYINT(1)
ProductLocation VARCHAR(250)

Example 03

Mục tiêu: Tạo và quản lý ứng dụng **Spring Boot Rest API** kết hợp với ReactJS

Yêu cầu: Thực hiện các chức năng sau:

- ✓ Truy vấn dữ liệu

Hướng dẫn:

Bước 0: Sử dụng Visual Studio Code mở thư mục **D:\java-projects\example02** Click chuột phải vào lớp **Example02Application**, sau đó Click vào **Run Java**.

Bước 1: Sử dụng Visual Studio Code mở thự mục **java-projects** sau đó thực hiện lệnh như sau để tạo project có tên là **example03**:

Terminal

```
PS D:\java-projects\example03>npx create-react-app example03
```

Bước 2: Sử dụng Visual Studio Code mở thự mục **java-projects\example03** sau đó thực hiện lệnh như sau cài đặt ReactJS Bootstrap:

Terminal

```
PS D:\java-projects\example03>npm install react-bootstrap bootstrap
```

Bước 3: Sửa file index.js như sau:

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import "bootstrap/dist/css/bootstrap.min.css";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <App />
);
```

Bước 4: Tạo thự mục **src/components/shared**, sau đó tạo React Bootstrap Menu có tên file **Layout.js** như sau:

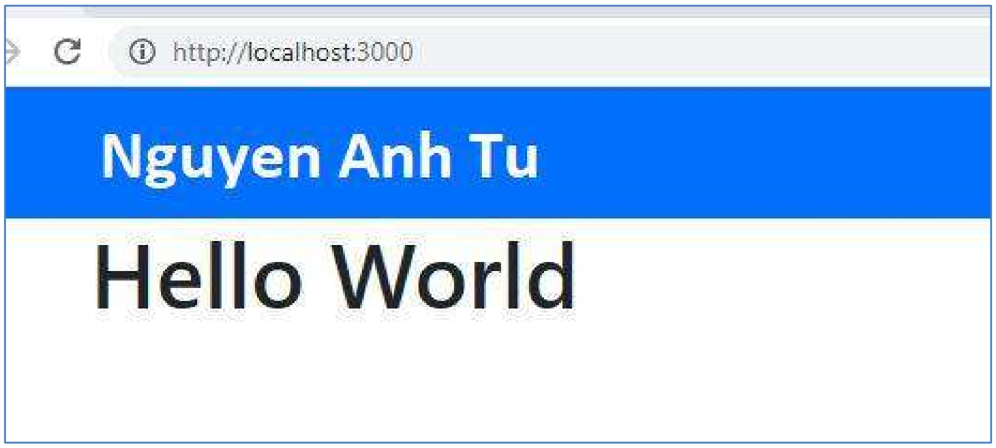
src/components/shared/Layout.js

```
import Container from "react-bootstrap/Container";
import Navbar from "react-bootstrap/Navbar";

function Layout(props) {
  return (
    <div>
      <Navbar bg="primary" variant="dark" expand="lg">
        <Container>
          <Navbar.Brand>Nguyen Anh Tu</Navbar.Brand>
          <Navbar.Toggle aria-controls="basic-navbar-nav" />
        </Container>
      </Navbar>
      <Container>{props.children}</Container>
    </div>
  );
}
export default Layout;
```

Bước 5: Sửa file App.js như sau:

src/App.js

```
import './App.css';
import Layout from './components/shared/Layout';
function App() {
  return (
    <Layout>
      <h1>Hello World</h1>
    </Layout>
  );
}
export default App;
```



Bước 6: Cài đặt thư viện **Axios** với lệnh như sau:

Terminal

```
PS D:\java-projects\example03>npm i axios
```

Bước 7: Tạo mới thư mục **pages** sau đó tạo react component có tên **AllProduct.js**

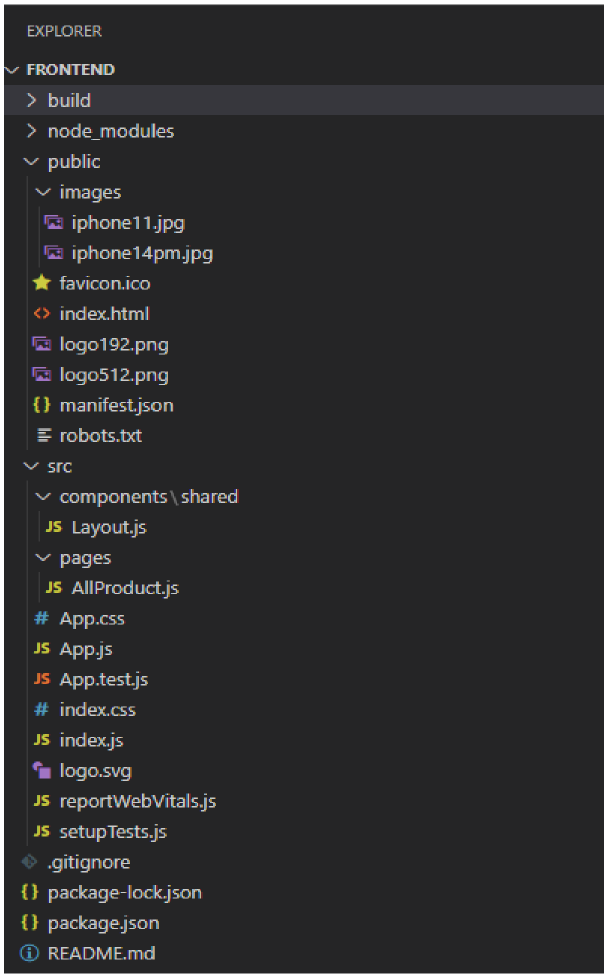
src/pages/AllProduct.js

```
import { Row } from "react-bootstrap";
import Card from "react-bootstrap/Card";
import Col from "react-bootstrap/Col";
import { useEffect, useState } from "react";
import axios from "axios";

function AllProduct() {
  const [products, setProducts] = useState([]);

  useEffect(() => {
    axios.get("https://localhost:8080/api/products").then((response) => {
      setProducts((data) => {
        return response.data;
      });
    });
  }, []);

  return (
    <>
      <Row md={3} className="g-4 mt-1">
        {products.map((sv) => {
          return (
            <Col key={sv.id}>
              <Card>
                <Card.Img variant="top" src={sv.photo} />
                <Card.Body>
                  <Card.Title>{sv.title}</Card.Title>
                  <Card.Text>
                    <b>Thông số kỹ thuật:</b> {sv.description}
                  </Card.Text>
                  <Card.Text>
                    <b>Giá: </b>
                    {sv.price}
                  </Card.Text>
                </Card.Body>
              </Card>
            </Col>
          );
        })}
      </Row>
    </>
  );
}
export default AllProduct;
```

Bước 8: Sửa file App.js như sau:

src/App.js

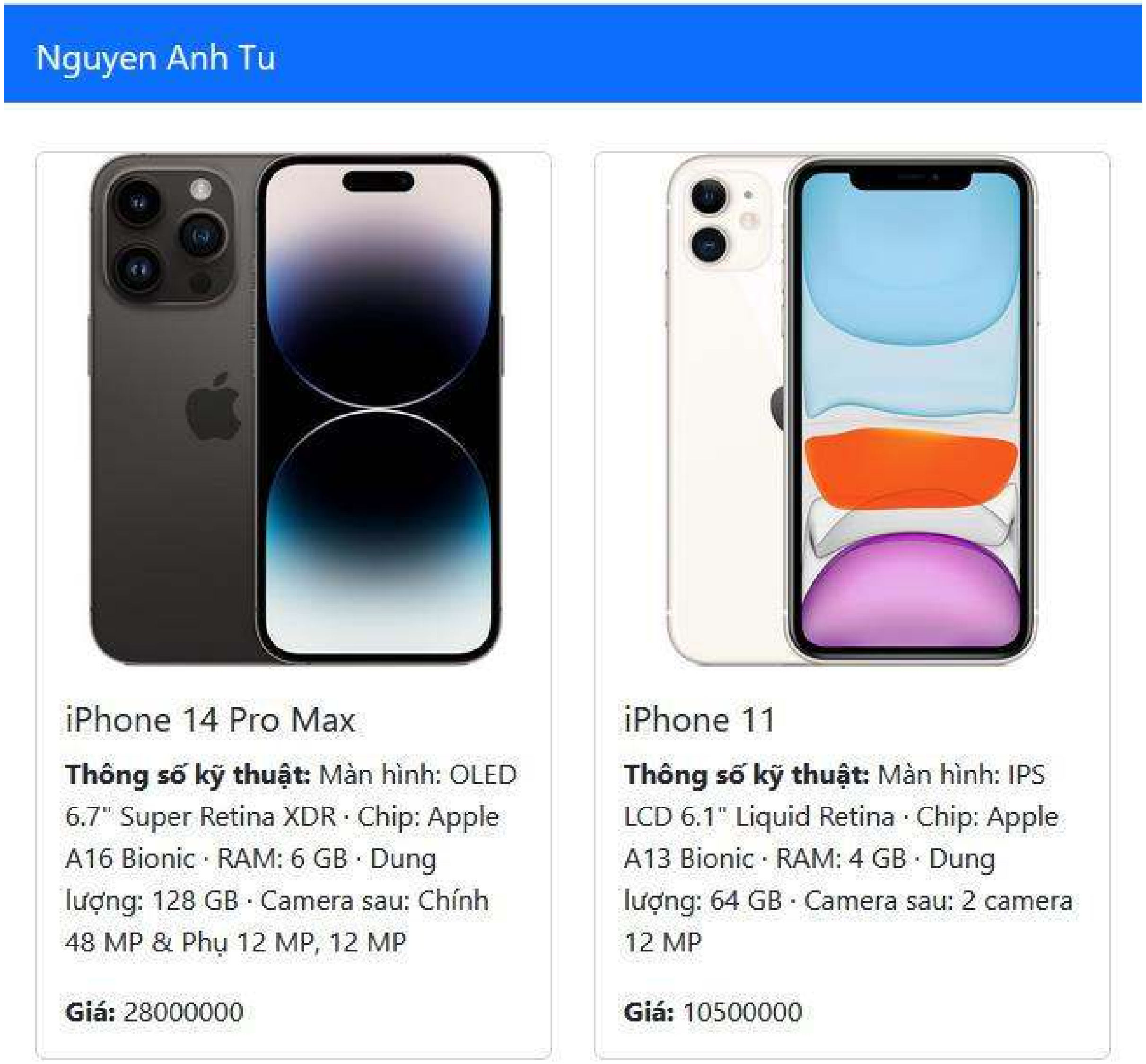
```
import './App.css';
import Layout from './components/shared/Layout';
import AllProduct from './pages/AllProduct';

function App() {
  return (
    <Layout>
      <AllProduct></AllProduct >
    </Layout>
  );
}
export default App;
```



```
EXPLORER
FRONTEND
  > build
  > node_modules
  v public
    v images
      iphone11.jpg
      iphone14pm.jpg
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
  v src
    v components\shared
      Layout.js
    v pages
      AllProduct.js
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Bước 9: Start Frontend

Terminal
PS D:\java-projects\example03>**npm start**



Nguyen Anh Tu

iPhone 14 Pro Max

Thông số kỹ thuật: Màn hình: OLED 6.7" Super Retina XDR · Chip: Apple A16 Bionic · RAM: 6 GB · Dung lượng: 128 GB · Camera sau: Chính 48 MP & Phụ 12 MP, 12 MP

Giá: 28000000

iPhone 11

Thông số kỹ thuật: Màn hình: IPS LCD 6.1" Liquid Retina · Chip: Apple A13 Bionic · RAM: 4 GB · Dung lượng: 64 GB · Camera sau: 2 camera 12 MP

Giá: 10500000

Example 04

Mục tiêu: Tạo và quản lý ứng dụng **Spring Boot Rest API** sử dụng **Hibernate** kết nối cở sở dữ liệu **MySQL** với quan hệ **One to Many**.

Yêu cầu: Thực hiện các chức năng sau:

- ✓ Tạo bảng
- ✓ Thêm dữ liệu
- ✓ Xóa dữ liệu
- ✓ Sửa dữ liệu
- ✓ Truy vấn dữ liệu

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code tạo project với thông tin như sau:

- ✓ Spring Boot version: **3.1.1**
- ✓ Language: **Java**
- ✓ Group Id: **com.nguyenanhtu**
- ✓ Artifact Id: **example03**
- ✓ Packaging type: **Jar**
- ✓ Java Version: **17**
- ✓ Dependencies: **Spring Web, Spring Data JPA, MySQL Driver, Lombok**
- ✓ Vị trí project: **D:\java-projects**

Bước 3: Sửa file **application.properties** như sau:

src/main/resources/application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/example03
spring.datasource.username=root
spring.datasource.password=
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
```

Bước 4: Tao Entity **Product** như sau:

Product.java

```java
package com.nguyenanhtu.example03.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "Products")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String title;
    @Column(nullable = false)
    private String description;
    @Column(nullable = false, unique = true)
    private String photo;
    @Column(nullable = false)
    private double price;
    @ManyToOne
    @JoinColumn(name="category_id", nullable=false)
    private Category category;
}
```

Bước 5: Tao Entity **Category** như sau:

Category.java

```
package com.nguyenanhtu.example03.entity;

import java.util.List;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "Categories")
public class Category {

    @Id
```

```java
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String title;
    @Column(nullable = false)
    private String description;
    @Column(nullable = false, unique = true)
    private String photo;
    @OneToMany(mappedBy = "category")
    private List < Product > products;
}
```

Bước 6: Tạo Interface cho **Product Repository** như sau:

ProductRepository.java

```java
package com.nguyenanhtu.example03.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.nguyenanhtu.example03.entity.Product;

public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

Bước 7: Tạo Interface cho **Product Service** như sau:

ProductService.java

```java
package com.nguyenanhtu.example03.service;

import java.util.List;
import com.nguyenanhtu.example03.entity.Product;

public interface ProductService {
    Product createProduct(Product product);
    Product getProductById(Long productId);
    List<Product> getAllProducts();
    Product updateProduct(Product product);
    void deleteProduct(Long productId);
}
```

Bước 8: Tao Service Layer cho **Product** như sau:

ProductServiceImpl.java

```
package com.nguyenanhtu.example03.service.impl;

import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import com.nguyenanhtu.example03.entity.Product;
import com.nguyenanhtu.example03.service.ProductService;
import com.nguyenanhtu.example03.repository.ProductRepository;
import java.util.List;
import java.util.Optional;

@Service
@AllArgsConstructor
public class ProductServiceImpl implements ProductService {

    private ProductRepository productRepository;

    @Override
    public Product createProduct(Product product) {
        return productRepository.save(product);
    }

    @Override
    public Product getProductById(Long productId) {
        Optional<Product> optionalProduct = productRepository.findById(productId);
        return optionalProduct.get();
    }

    @Override
    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }

    @Override
    public Product updateProduct(Product product) {
        Product existingProduct = productRepository.findById(product.getId()).get();
        existingProduct.setTitle(product.getTitle());
        existingProduct.setDescription(product.getDescription());
        existingProduct.setPhoto(product.getPhoto());
        existingProduct.setPrice(product.getPrice());
        existingProduct.setCategory(product.getCategory());
        Product updatedProduct = productRepository.save(existingProduct);
        return updatedProduct;
    }

    @Override
    public void deleteProduct(Long productId) {
        productRepository.deleteById(productId);
    }
}
```

Bước 9: Tạo Interface cho **Category** Repository như sau:

CategoryRepository.java

```
package com.nguyenanhtu.example03.repository;
```

```java
import org.springframework.data.jpa.repository.JpaRepository;
import com.nguyenanhtu.example03.entity.Category;

public interface CategoryRepository extends JpaRepository<Category, Long> {
}
```

Bước 10: Tạo Interface cho **Category** Service như sau:

CategoryRepository.java

```java
package com.nguyenanhtu.example03.service;

import java.util.List;
import com.nguyenanhtu.example03.entity.Category;

public interface CategoryService {
    Category createCategory(Category category);
    Category getCategoryById(Long categoryId);
    List<Category> getAllCategories();
    Category updateCategory(Category category);
    void deleteCategory(Long categoryId);
}
```

Bước 11: Tao Service Layer cho **Category** như sau:

ProductServiceImpl.java

```java
package com.nguyenanhtu.example03.service.impl;

import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import com.nguyenanhtu.example03.entity.Category;
import com.nguyenanhtu.example03.service.CategoryService;
import com.nguyenanhtu.example03.repository.CategoryRepository;
import java.util.List;
import java.util.Optional;

@Service
@AllArgsConstructor
public class CategoryServiceImpl implements CategoryService {

    private CategoryRepository categoryRepository;

    @Override
    public Category createCategory(Category category) {
        return categoryRepository.save(category);
    }

    @Override
    public Category getCategoryById(Long categoryId) {
        Optional<Category> optionalCategory = categoryRepository.findById(categoryId);
        return optionalCategory.get();
    }

    @Override
    public List<Category> getAllCategories() {
        return categoryRepository.findAll();
    }

    @Override
    public Category updateCategory(Category category) {
        Category existingCategory = categoryRepository.findById(category.getId()).get();
        existingCategory.setTitle(category.getTitle());
        existingCategory.setDescription(category.getDescription());
        existingCategory.setPhoto(category.getPhoto());
        existingCategory.setProducts(category.getProducts());
        Category updatedCategory = categoryRepository.save(existingCategory);
        return updatedCategory;
    }

    @Override
    public void deleteCategory(Long categoryId) {
        categoryRepository.deleteById(categoryId);
    }
}
```

Bước 12: Tạo Controller cho **Product** như sau:

ProductController.java

```
package com.nguyenanhtu.example02.controller;

import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import com.nguyenanhtu.example03.entity.Product;
import com.nguyenanhtu.example03.service.ProductService;
import java.util.List;

@RestController
@AllArgsConstructor
@RequestMapping("api/products")
public class ProductController {

    private ProductService productService;

    // Create Product REST API
    @PostMapping
    public ResponseEntity<Product> createProduct(@RequestBody Product Product) {
        Product savedProduct = productService.createProduct(Product);
        return new ResponseEntity<>(savedProduct, HttpStatus.CREATED);
    }

    // Get Product by id REST API
    // http://localhost:8080/api/Products/1
    @GetMapping("{id}")
    public ResponseEntity<Product> getProductById(@PathVariable("id") Long ProductId) {
        Product Product = productService.getProductById(ProductId);
        return new ResponseEntity<>(Product, HttpStatus.OK);
    }

    // Get All Products REST API
    // http://localhost:8080/api/Products
    @GetMapping
    public ResponseEntity<List<Product>> getAllProducts() {
        List<Product> Products = productService.getAllProducts();
        return new ResponseEntity<>(Products, HttpStatus.OK);
    }

    // Update Product REST API
    @PutMapping("{id}")
    // http://localhost:8080/api/Products/1
    public ResponseEntity<Product> updateProduct(@PathVariable("id") Long ProductId,
            @RequestBody Product Product) {
        Product.setId(ProductId);
        Product updatedProduct = productService.updateProduct(Product);
        return new ResponseEntity<>(updatedProduct, HttpStatus.OK);
    }
```

```java
    // Delete Product REST API
    @DeleteMapping("{id}")
    public ResponseEntity<String> deleteProduct(@PathVariable("id") Long ProductId) {
        productService.deleteProduct(ProductId);
        return new ResponseEntity<>("Product successfully deleted!", HttpStatus.OK);
    }
}
```

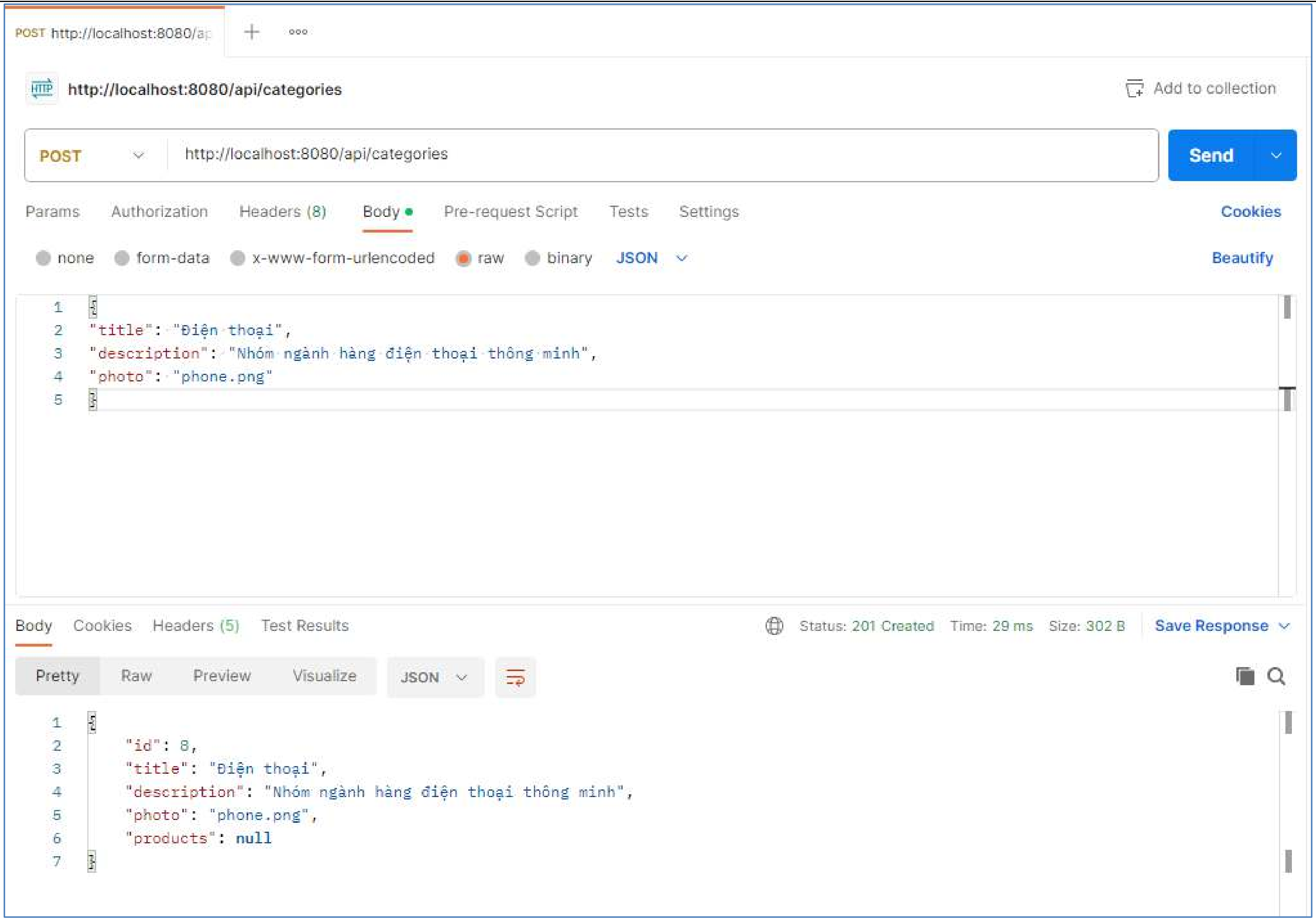
Bước 13: Tạo Controller cho **Category** như sau:

CategoryController.java

```java
package com.nguyenanhtu.example03.controller;

import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import com.nguyenanhtu.example03.entity.Category;
import com.nguyenanhtu.example03.service.CategoryService;
import java.util.List;

@RestController
@AllArgsConstructor
@RequestMapping("api/categories")
public class CategoryController {

    private CategoryService categoryService;

    // Create Category REST API
    @PostMapping
    public ResponseEntity<Category> createCategory(@RequestBody Category category) {
        Category savedCategory = categoryService.createCategory(category);
        return new ResponseEntity<>(savedCategory, HttpStatus.CREATED);
    }

    // Get Category by id REST API
    // http://localhost:8080/api/Categories/1
    @GetMapping("{id}")
    public ResponseEntity<Category> getCategoryById(@PathVariable("id") Long categoryId) {
        Category Category = categoryService.getCategoryById(categoryId);
        return new ResponseEntity<>(Category, HttpStatus.OK);
    }

    // Get All Categorys REST API
    // http://localhost:8080/api/Categories
    @GetMapping
    public ResponseEntity<List<Category>> getAllCategorys() {
        List<Category> Categories = categoryService.getAllCategories();
        return new ResponseEntity<>(Categories, HttpStatus.OK);
    }

    // Update Category REST API
    @PutMapping("{id}")
    // http://localhost:8080/api/Categories/1
    public ResponseEntity<Category> updateCategory(@PathVariable("id") Long categoryId,
            @RequestBody Category Category) {
        Category.setId(categoryId);
        Category updatedCategory = categoryService.updateCategory(Category);
        return new ResponseEntity<>(updatedCategory, HttpStatus.OK);
    }

    // Delete Category REST API
    @DeleteMapping("{id}")
    public ResponseEntity<String> deleteCategory(@PathVariable("id") Long categoryId) {
        categoryService.deleteCategory(categoryId);
        return new ResponseEntity<>("Category successfully deleted!", HttpStatus.OK);
    }
}
```

Bước 14: Click chuột phải vào lớp **Example03Application**, sau đó Click vào **Run Java**.

Bước 15: Sử dụng Postman kiểm tra các API



POST http://localhost:8080/api/categories

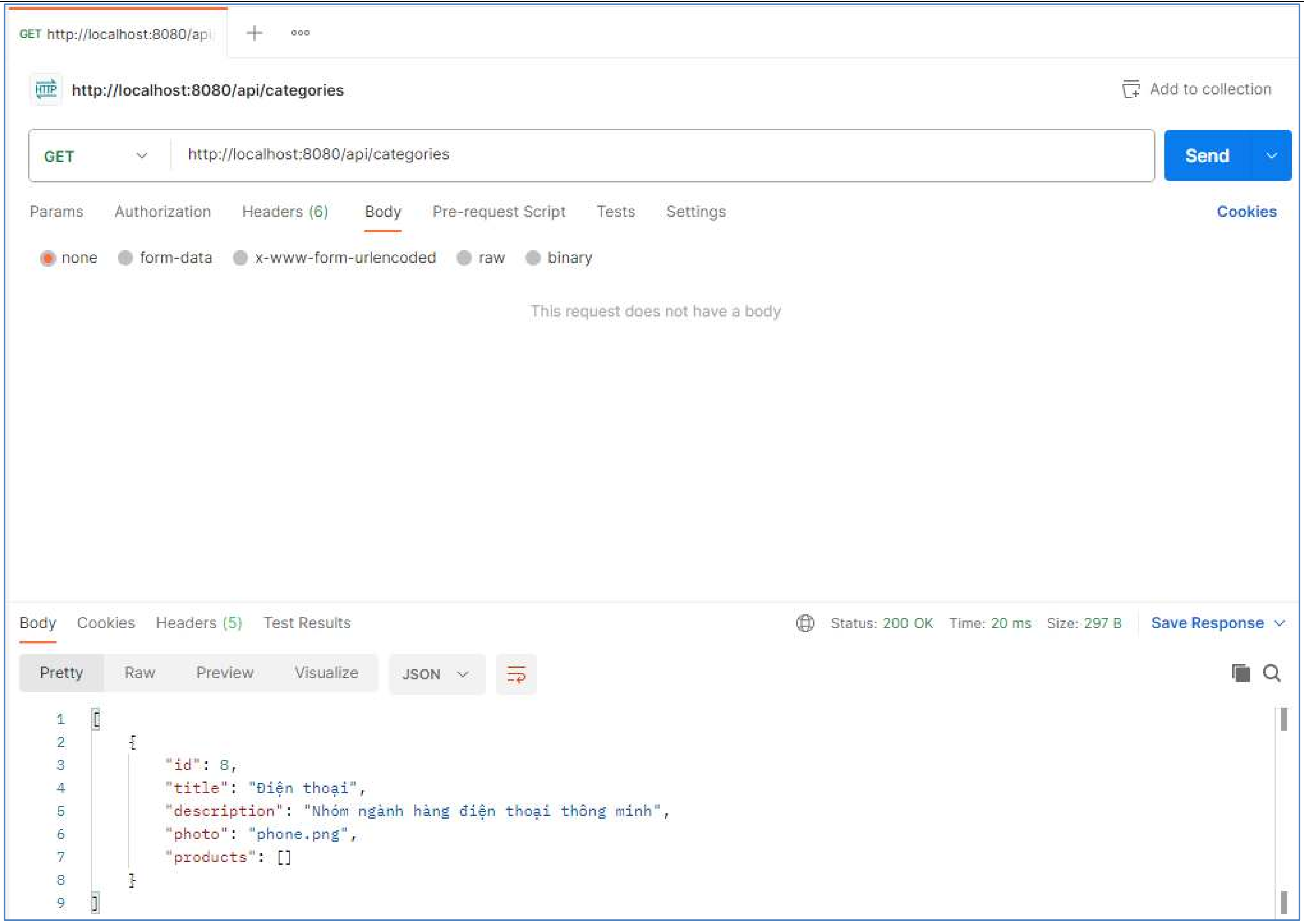
Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | JSON

```
{
"title": "Điện thoại",
"description": "Nhóm ngành hàng điện thoại thông minh",
"photo": "phone.png"
}
```

Status: 201 Created Time: 29 ms Size: 302 B

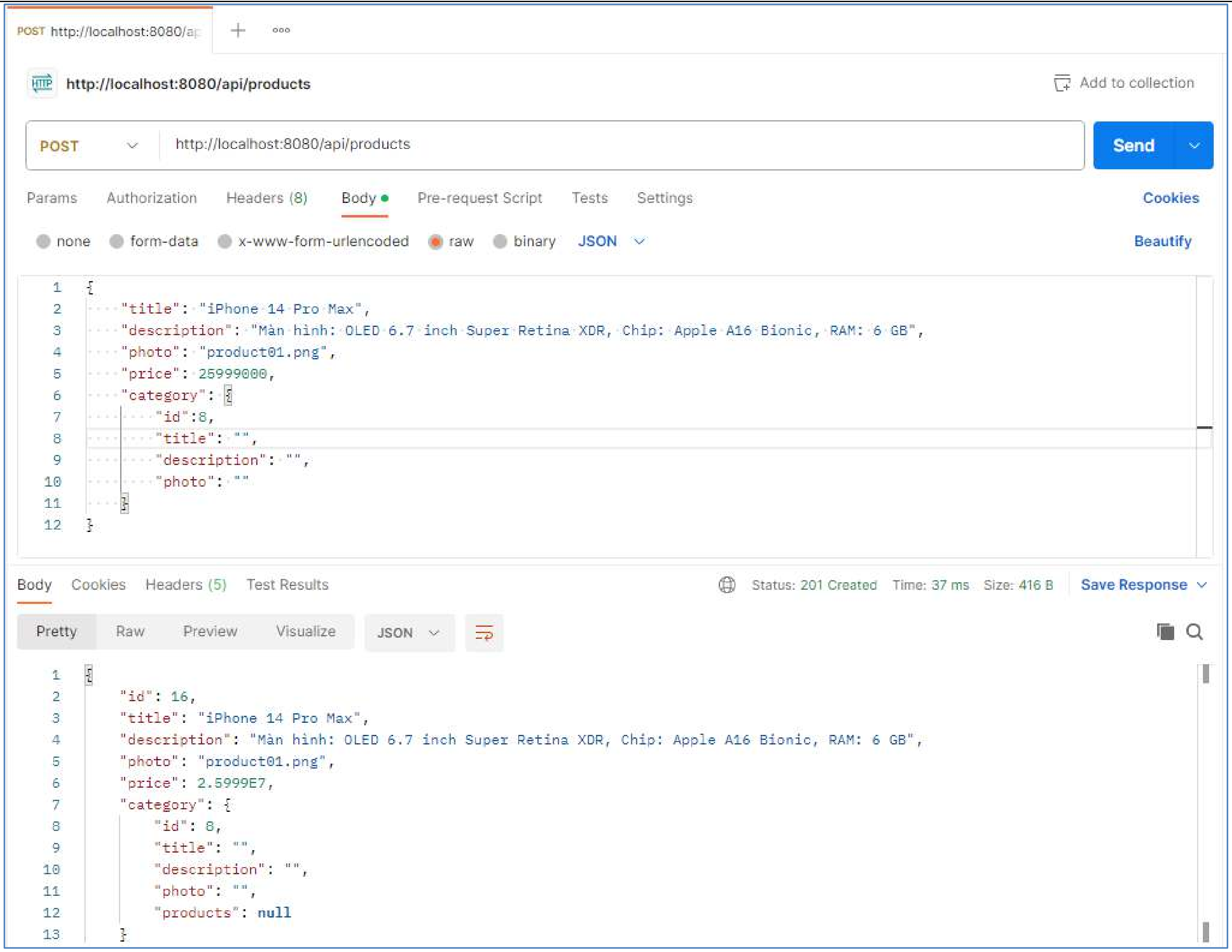
```
{
    "id": 8,
    "title": "Điện thoại",
    "description": "Nhóm ngành hàng điện thoại thông minh",
    "photo": "phone.png",
    "products": null
}
```


GET http://localhost:8080/api/categories

This request does not have a body

Status: 200 OK Time: 20 ms Size: 297 B

```
[
    {
        "id": 8,
        "title": "Điện thoại",
        "description": "Nhóm ngành hàng điện thoại thông minh",
        "photo": "phone.png",
        "products": []
    }
]
```



POST http://localhost:8080/api/products

```
{
    "title": "iPhone 14 Pro Max",
    "description": "Màn hình: OLED 6.7 inch Super Retina XDR, Chip: Apple A16 Bionic, RAM: 6 GB",
    "photo": "product01.png",
    "price": 25999000,
    "category": {
        "id":8,
        "title": "",
        "description": "",
        "photo": ""
    }
}
```

Status: 201 Created Time: 37 ms Size: 416 B

```
{
    "id": 16,
    "title": "iPhone 14 Pro Max",
    "description": "Màn hình: OLED 6.7 inch Super Retina XDR, Chip: Apple A16 Bionic, RAM: 6 GB",
    "photo": "product01.png",
    "price": 2.5999E7,
    "category": {
        "id": 8,
        "title": "",
        "description": "",
        "photo": "",
        "products": null
    }
```

Example 06

Mục tiêu: Tạo và quản lý ứng dụng **Spring Boot Rest API** kết hợp với ReactJS

Yêu cầu: Thực hiện các chức năng sau:

- ✓ Tạo bảng
- ✓ Thêm dữ liệu
- ✓ Xóa dữ liệu
- ✓ Sửa dữ liệu
- ✓ Truy vấn dữ liệu

Hướng dẫn:

Bước 0: Sử dụng Visual Studio Code mở thư mục **D:\java-projects\example04** Click chuột phải vào lớp **Example04Application**, sau đó Click vào **Run Java**.

Bước 1: Sử dụng Visual Studio Code mở thụ mục **java-projects** sau đó thực hiện lệnh như sau để tạo project có tên là **example05**:

Terminal

```
PS D:\java-projects>npx create-react-app example05
```

Bước 2: Sử dụng Visual Studio Code mở thụ mục **java-projects\example05** sau đó thực hiện lệnh như sau cài đặt package:

Terminal

```
PS D:\java-projects\example05>npm install axios
PS D:\java-projects\example05>npm install bootstrap
PS D:\java-projects\example05>npm install react
PS D:\java-projects\example05>npm install react-dom
PS D:\java-projects\example05>npm install react-bootstrap
PS D:\java-projects\example05>npm install react-router-bootstrap
PS D:\java-projects\example05>npm install react-router-dom
PS D:\java-projects\example05>npm install @mui/material
PS D:\java-projects\example05>npm install @mui/icons-material
PS D:\java-projects\example05>npm install @mui/styles
PS D:\java-projects\example05>npm install @emotion/react
PS D:\java-projects\example05>npm install @emotion/styled
PS D:\java-projects\example05>npm start
```

Bước 3: Tạo file apiService.js như sau:

/src/api/apiService.js

```
import axios from 'axios';
let API_URL="https://localhost:8080/api";
export function callApi(endpoint, method='GET',body){
        return axios({
            method,
            url:`${API_URL}/${endpoint}`,
            data:body,
        }).catch(e=>{
            console.log(e)
        })
}
export function GET_ALL_PRODUCTS(endpoint){
    return callApi(endpoint,"GET");
}
export function GET_PRODUCT_ID(endpoint,id){
    return callApi(endpoint+"/"+id,"GET");
}
export function POST_ADD_PRODUCT(endpoint,data){
    return callApi(endpoint,"POST",data);
}
export function PUT_EDIT_PRODUCT(endpoint,data){
    return callApi(endpoint,"PUT",data);
}
export function DELETE_PRODUCT_ID(endpoint){
    return callApi(endpoint,"DELETE");
}
export function GET_ALL_CATEGORIES(endpoint){
    return callApi(endpoint,"GET");
}
```

Bước 4: Tạo thư mục **components** và file **MenuTop.js** như sau:

/src/components/MenuTop.js

```
import React from 'react';
import {Link} from 'react-router-dom'
import { makeStyles } from '@mui/styles';
import {AppBar,Toolbar,Typography,IconButton} from '@mui/material';
import MenuIcon from '@mui/icons-material/Menu';
import MenuItem from '@mui/material/MenuItem';
import Menu from '@mui/material/Menu';
import MoreVertIcon from '@mui/icons-material/Menu';

const useStyles = makeStyles((theme) => ({
    root: {
       width:'100%',
       flexGrow: 1,
    },
    title:{
        flexGrow:1
    },
    linkTo:{
      textDecoration:'none',
      color:'#000'
    },
    linkHome:{
      textDecoration:'none',
      color:'#fff'
    }
  }));
export default function MenuTop() {
  const classes = useStyles();
  const [anchorEl, setAnchorEl] = React.useState(null);
  const isMenuOpen = Boolean(anchorEl);
  const handleProfileMenuOpen = (event) => {
    setAnchorEl(event.currentTarget);
  };
  const handleMenuClose = () => {
    setAnchorEl(null);
  };
  const menuId = 'primary-search-account-menu';
  const renderMenu = (
    <Menu
      anchorEl={anchorEl}
      anchorOrigin={{ vertical: 'top', horizontal: 'right' }}
      id={menuId}
      keepMounted
      transformOrigin={{ vertical: 'top', horizontal: 'right' }}
      open={isMenuOpen}y
      onClose={handleMenuClose}
    >
      <MenuItem onClick={handleMenuClose}><Link to="/products" className={classes.linkTo}>Product</Link></MenuItem>
      <MenuItem onClick={handleMenuClose}><Link to="/categories"
className={classes.linkTo}>Categories</Link></MenuItem>
      </Menu>
  );
    return (
        <div className={classes.root}>
            <AppBar position="static" color="primary">
              <Toolbar>
                <IconButton edge="start" color="inherit" aria-label="menu">
                    <MenuIcon />
                </IconButton>
                <Typography variant="h6" className={classes.title}>
                     <Link to="/" className={classes.linkHome}>Demo React ASP Core</Link>
                </Typography>
                <IconButton edge="end" color="inherit" aria-label="MoreVert" aria-controls={menuId}
                 aria-haspopup="true"
              onClick={handleProfileMenuOpen}>
```

```
                    <MoreVertIcon />
                </IconButton>
              </Toolbar>
            </AppBar>
            {renderMenu}
        </div>
    )
}
```

Bước 5: Tạo file Home.js như sau:

/src/components/Home.js

```
import React,{useEffect,useState} from 'react'
import { makeStyles } from '@mui/material/styles';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';
import Alert from '@mui/material/Alert';
import { Redirect } from 'react-router-dom';
import IconButton from '@mui/icons-material/IosShare';
import CloseIcon from '@mui/icons-material/IosShare';
import Button from '@mui/material/Button';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import {Link} from 'react-router-dom'
import { GET_ALL_PRODUCTS,DELETE_PRODUCT_ID } from '../api/apiService';
const useStyles = makeStyles((theme) => ({
  root: {
    flexGrow: 1,
    marginTop:20
  },
  paper:{
    width:'100%',
    margin:'auto'
  },
  removeLink:{
    textDecoration:'none'
  }
}));
export default function Home() {
  const classes = useStyles();
  const [products,setProducts] = useState({});
  const [checkDeleteProduct,setCheckDeleteProduct] = useState(false);
  const [close, setClose] = React.useState(false);
  useEffect(() => {
    GET_ALL_PRODUCTS(`products`).then(item=>setProducts(item.data))
  }, [])
  const RawHTML = (body,className) =>
    <div className={className} dangerouslySetInnerHTML={{ __html: body.replace(/\n/g, '<br />')}} />
  const deleteProductID = (id)=>{
    DELETE_PRODUCT_ID(`products/${id}`).then(item=>{
      console.log(item)
      if(item.data===1){
        setCheckDeleteProduct(true);
        setProducts(products.filter(key=>key.idProduct!==id))
      }
    })
  }
  return (
    <div className={classes.root}>
         <Grid container spacing={3}>
            <Grid item xs={12}>
              <Paper className={classes.paper}>
                {checkDeleteProduct && <Alert
```

```
                  action={
                  <IconButton
                    aria-label="close"
                    color="inherit"
                    size="small"
                    onClick={() => {
                      setClose(true);
                      setCheckDeleteProduct(false)
                    }}
                  >
                    <CloseIcon fontSize="inherit" />
                  </IconButton>
                }
                >Detele successfuly</Alert>}
              <TableContainer component={Paper}>
                  <Table className={classes.table} aria-label="simple table">
                    <TableHead>
                      <TableRow>
                        <TableCell>Title</TableCell>
                        <TableCell align="center">Body</TableCell>
                        <TableCell align="center">Price</TableCell>
                        <TableCell align="center">Category</TableCell>
                        <TableCell align="center">Modify</TableCell>
                        <TableCell align="center">Delete</TableCell>
                      </TableRow>
                    </TableHead>
                    <TableBody>
                      {products.length>0 && products.map((row) => (
                        <TableRow key={row.idProduct}>
                          <TableCell component="th" scope="row">{row.title}</TableCell>
                          <TableCell align="left">{RawHTML(row.description,"body")}</TableCell>
                          <TableCell align="center">{row.price}</TableCell>
                          <TableCell align="center">{row.category.name}</TableCell>
                          <TableCell align="center">
                            <Link to={`/edit/product/${row.id}`} className={classes.removeLink}>
                               <Button size="small" variant="contained" color="primary">Edit</Button></Link>
                            </TableCell>
                          <TableCell align="center">
                              <Button  size="small" variant="contained" color="secondary"
onClick={()=>deleteProductID(row.idProduct)}>Remove</Button>
                          </TableCell>
                        </TableRow>
                      ))}
                    </TableBody>
                  </Table>
                </TableContainer>
              </Paper>
            </Grid>
        </Grid>
    </div>
  )
}
```

Bước 6: Sửa file App.js như sau:

/src/App.js

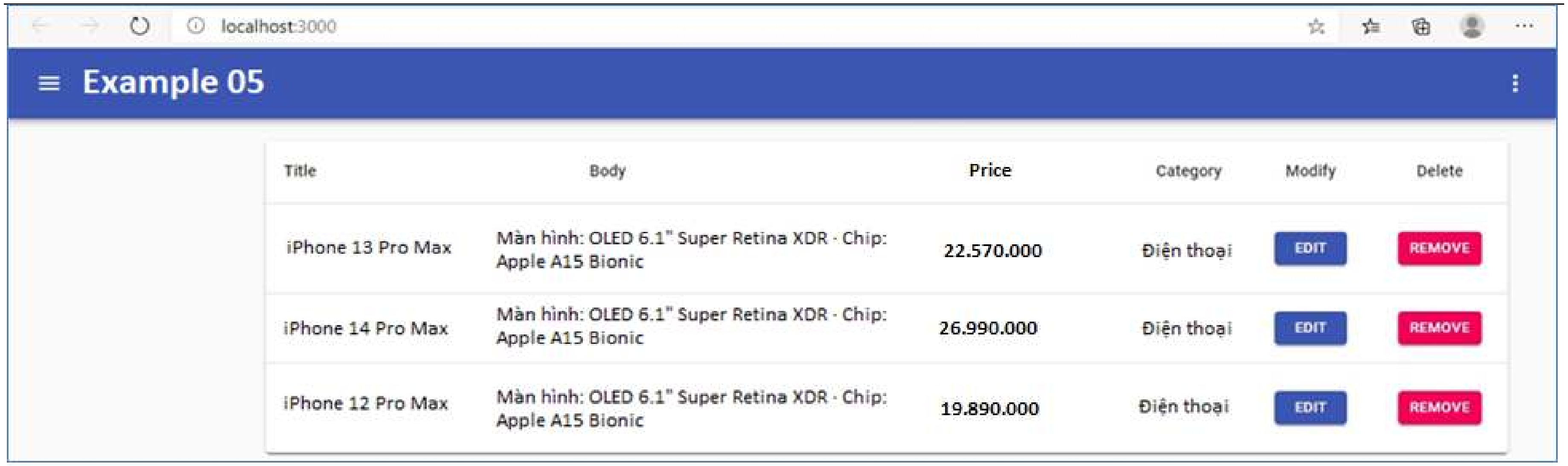
```
import React, { Component } from 'react';
import { Route } from 'react-router';
import { Container } from '@mui/material';
import CssBaseline from '@mui/material/CssBaseline';
import MenuTop from './components/MenuTop';
import Home from './components/Home';
export default class App extends Component {
  displayName = App.name;
  render() {
    return (
       <React.Fragment>
        <CssBaseline />
        <MenuTop />
        <Container maxWidth="md">
           <Route exact path='/' component={Home} />
           <Route path='/home' component={Home} />
        </Container>
    </React.Fragment>
    );
  }
}
```

Bước 7: Sửa file **index.js** như sau:

/src/index.js

```
import './index.css';
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
const rootElement = document.getElementById('root');

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
rootElement);
```

Bước 8: Thực hiện lệnh **npm start** để Start FrontEnd trong Visual Code



Title	Body	Price	Category	Modify	Delete
iPhone 13 Pro Max	Màn hình: OLED 6.1" Super Retina XDR - Chip: Apple A15 Bionic	22.570.000	Điện thoại	EDIT	REMOVE
iPhone 14 Pro Max	Màn hình: OLED 6.1" Super Retina XDR - Chip: Apple A15 Bionic	26.990.000	Điện thoại	EDIT	REMOVE
iPhone 12 Pro Max	Màn hình: OLED 6.1" Super Retina XDR - Chip: Apple A15 Bionic	19.890.000	Điện thoại	EDIT	REMOVE

Bước 9: Tạo file Product.js như sau:

/components/Product.js

```
import React,{useEffect,useState} from 'react'
import { makeStyles } from '@mui/material/styles';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';
import Typography from '@mui/material/Typography';
import TextField from '@mui/material/TextField';
import Button from '@mui/material/Button';
import Alert from '@mui/material/Alert';
import { Redirect } from 'react-router-dom';

import {GET_ALL_CATEGORIES, POST_ADD_PRODUCT} from '../api/apiService';
const useStyles = makeStyles((theme) => ({
  root: {
    flexGrow: 1,
    marginTop:20
  },
  paper: {
    padding: theme.spacing(2),
    margin: 'auto',
    maxWidth: 600,
  },
  title:{
    fontSize:30,
    textAlign:'center'
  },
  link:{
    padding:10,
    display:'inline-block'
  },
  txtInput:{
    width:'98%',
    margin:'1%'
  },
  submit: {
    margin: theme.spacing(3, 0, 2),
  },
}));
const currencies = [
    {
      value: 'USD',
      label: '$',
    },
    {
      value: 'EUR',
      label: '€',
    },
    {
      value: 'BTC',
   label: '฿',
    },
    {
      value: 'JPY',
      label: '¥',
    },
  ];
export default function Product() {
  const classes = useStyles();
  const [checkAdd,setCheckAdd] = useState(false);
  const [title,setTitle] = useState(null)
  const [body,setBody] = useState(null)
  const [Price,setPrice] = useState(null)
  const [category, setCategory] = useState(0);
  const [categories,setCategories] = useState({});

  useEffect(() => {
    GET_ALL_CATEGORIES('categories').then(item=>{
      setCategories(item.data);
    });

  }, [])
  const handleChangeTitle = (event) => {
    setTitle(event.target.value)
  }
  const handleChangeBody = (event) => {
    setBody(event.target.value)
  }
  const handleChangePrice = (event) => {
    setPrice(event.target.value)
  }
  const handleChangeCategory = (event) => {
    setCategory(event.target.value);
```

```
};

const addProduct = (event)=>{
  event.preventDefault();
  if(title!=="" && body!=="" && Price!=="" && category>0){
    let product = {
       Title:title,
       Body:body,
       Price:Price,
       idCategory:category
    }
    POST_ADD_PRODUCT(`products`,product).then(item=>{
       if(item.data===1){
        setCheckAdd(true);
       }
    })
  }
  else{
     alert("Bạn chưa nhập đủ thông tin!");
  }
}

if(checkAdd){
  return <Redirect to="/" />
}

return (
  <div className={classes.root}>
       <Grid container spacing={3}>
          <Grid item xs={12}>
            <Paper className={classes.paper}>
                  <Typography className={classes.title} variant="h4">
                       Add Product
```

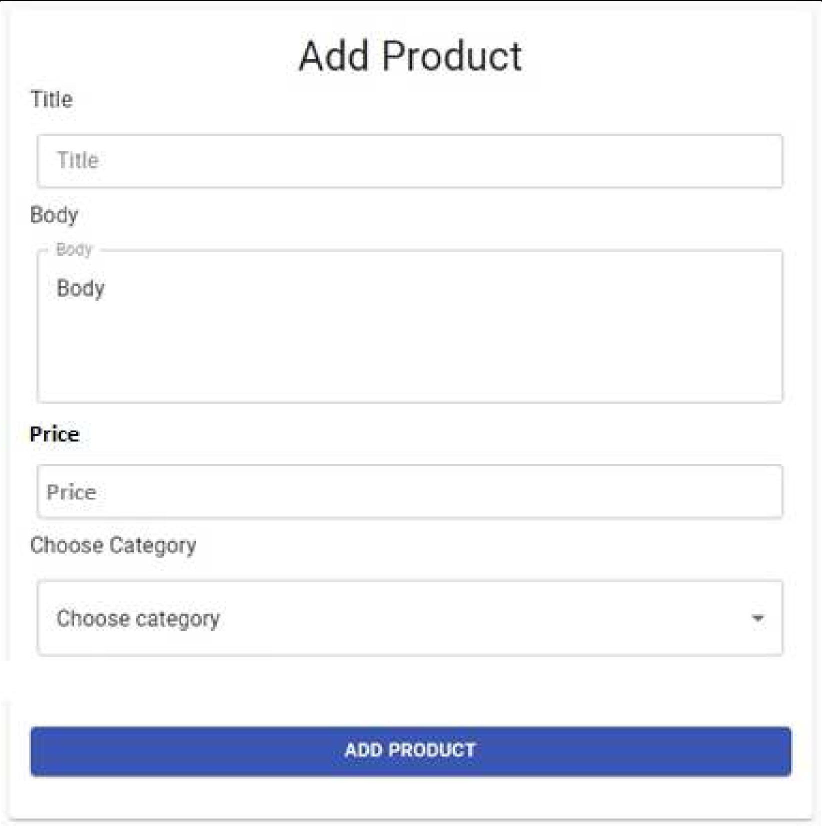
```
                    </Typography>
                    <Grid item xs={12} sm container>
                        <Grid item xs={12}>
                            <Typography gutterBottom variant="subtitle1">
                                Title
                            </Typography>
                            <TextField id="Title" onChange = {handleChangeTitle} name="Title" label="Title" 
variant="outlined" className={classes.txtInput} size="small"/>
                        </Grid>
                        <Grid item xs={12}>
                            <Typography gutterBottom variant="subtitle1">
                                Body
                            </Typography>
                            <TextField id="outlined-multiline-static" onChange = {handleChangeBody} label="Body" 
name="Body"  className={classes.txtInput} multiline rows={4} defaultValue="Body" variant="outlined"/>
                        </Grid>
                        <Grid item xs={12}>
                            <Typography gutterBottom variant="subtitle1">
                                Price
                            </Typography>
                            <TextField id="Price" onChange= {handleChangePrice} name="Price" label="Price" 
variant="outlined" className={classes.txtInput} size="small"/>
                        </Grid>
                        <Grid item xs={12}>
                           <Typography gutterBottom variant="subtitle1">
                                Choose Category
                            </Typography>
                            <TextField
                                id="outlined-select-currency-native"
                                name="idCategory"
                                select
                                value={category}
                                onChange={handleChangeCategory}
                                SelectProps={{
                                    native: true,
                                }}
                                helperText="Please select your currency"
                                variant="outlined"
                                className={classes.txtInput}
                                >
                                <option value="0">Choose category</option>
                                {categories.length>0 && categories.map((option) => (
                                    <option key={option.idCategory} value={option.idCategory}>
                                    {option.name}
                                    </option>
                                ))}
                              </TextField>
                        </Grid>
                        <Grid item xs={12}>
                            <Button type="button" onClick={addProduct}  fullWidth variant="contained" color="primary"
className={classes.submit} >
                                Add product
                            </Button>
                        </Grid>
                    </Grid>
              </Paper>
            </Grid>
        </Grid>
    </div>
  )
}
```

Bước 10: Tạo file App.js như sau:

/App.js

```
import React, { Component } from 'react';
import { Route } from 'react-router';
import Container from '@mui/material/Container';
import CssBaseline from '@mui/material/CssBaseline';
import MenuTop from './components/MenuTop';
import Home from './components/Home';
import Product from './components/Product';
import Category from './components/Category';
import EditProduct from './components/EditProduct';
export default class App extends Component {
  displayName = App.name;
  render() {
    return (
       <React.Fragment>
        <CssBaseline />
        <MenuTop />
        <Container maxWidth="md">
           <Route exact path='/' component={Home} />
           <Route path='/home' component={Home} />
           <Route path='/products' component={Product} />
        </Container>
    </React.Fragment>
    );
  }
}
```

Bước 11: Thực hiện lệnh **npm start** để Start FrontEnd trong Visual Code



Bước 12: Tạo file EditProduct.js như sau:

/components/EditProduct.js

```
import React,{useEffect,useState} from 'react'
import { makeStyles } from '@mui/material/styles';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';
import Typography from '@mui/material/Typography';
import TextField from '@mui/material/TextField';
import Button from '@mui/material/Button';
import Alert from '@mui/material/Alert';
import { Redirect } from 'react-router-dom';

import {GET_ALL_CATEGORIES, GET_PRODUCT_ID, PUT_EDIT_PRODUCT} from '../api/apiService';
const useStyles = makeStyles((theme) => ({
  root: {
    flexGrow: 1,
    marginTop:20
  },
  paper: {
    padding: theme.spacing(2),
    margin: 'auto',
    maxWidth: 600,
  },
  title:{
    fontSize:30,
    textAlign:'center'
  },
  link:{
    padding:10,
    display:'inline-block'
  },
  txtInput:{
    width:'98%',
    margin:'1%'
  },
  submit: {
    margin: theme.spacing(3, 0, 2),
  },
}));
export default function EditProduct ({ match, location }){
  const classes = useStyles();
  const [checkUpdate,setCheckUpdate] = useState(false);
  const [idProduct,setIdProduct] = useState(0);
  const [title,setTitle] = useState(null)
  const [body,setBody] = useState(null)
  const [Price,setPrice] = useState(null)
  const [category, setCategory] = useState(0);
  const [categories,setCategories] = useState({});

  useEffect(() => {
    GET_PRODUCT_ID(`products`,match.params.id).then(product=>{
        setIdProduct(product.data.idProduct)
        setTitle(product.data.title);
        setBody(product.data.body);
        setPrice(product.data.Price);
        setCategory(product.data.category.idCategory)
    })
    GET_ALL_CATEGORIES('categories').then(item=>{
      setCategories(item.data);
    });

  }, [])

  const handleChangeTitle = (event) => {
    setTitle(event.target.value)
  }
  const handleChangeBody = (event) => {
    setBody(event.target.value)
  }
  const handleChangePrice = (event) => {
    setPrice(event.target.value)
  }
  const handleChangeCategory = (event) => {
    setCategory(event.target.value);
```

```
};

const EditProduct = (event)=>{
  event.preventDefault();
  if(title!=="" && body!=="" && Price!=="" && category>0 && idProduct>0){
    let product = {
       Title:title,
       Body:body,
       Price:Price,
       idCategory:category
    }
    PUT_EDIT_PRODUCT(`products/${idProduct}`,product).then(item=>{
       if(item.data===1){
        setCheckUpdate(true);
       }
    })
  }
  else{
     alert("Bạn chưa nhập đủ thông tin!");
  }
}
if(checkUpdate){
  return <Redirect to="/" />
}
return (
  <div className={classes.root}>
       <Grid container spacing={3}>
          <Grid item xs={12}>
            <Paper className={classes.paper}>
                  <Typography className={classes.title} variant="h4">
                       Edit Product
                  </Typography>
                  <Grid item xs={12} sm container>
```

```
                        <Grid item xs={12}>
                            <Typography gutterBottom variant="subtitle1">
                                Title
                            </Typography>
                            <TextField id="Title" onChange = {handleChangeTitle} value={title} name="Title"
variant="outlined" className={classes.txtInput} size="small"/>
                        </Grid>
                        <Grid item xs={12}>
                            <Typography gutterBottom variant="subtitle1">
                                Body
                            </Typography>
                            <TextField id="outlined-multiline-static" onChange = {handleChangeBody} defaultValue={body}
name="Body"  className={classes.txtInput} multiline rows={4}  variant="outlined"/>
                        </Grid>
                        <Grid item xs={12}>
                           <Typography gutterBottom variant="subtitle1">
                                Choose Category
                            </Typography>
                            <TextField
                                id="outlined-select-currency-native"
                                name="idCategory"
                                select
                                value={category}
                                onChange={handleChangeCategory}
                                SelectProps={{
                                    native: true,
                                }}

                                variant="outlined"
                                className={classes.txtInput}
                                >
                                <option value="0">Choose category</option>
                                {categories.length>0 && categories.map((option) => (
                                    <option key={option.idCategory} value={option.idCategory}>
```

```
                              {option.name}
                              </option>
                            ))}
                          </TextField>
                      </Grid>
                      <Grid item xs={12}>
                          <Button type="button" onClick={EditProduct}  fullWidth variant="contained" color="primary"
className={classes.submit} >
                            Update product
                          </Button>
                      </Grid>
                  </Grid>
              </Paper>
            </Grid>
        </Grid>
    </div>
  )
}
```

Bước 13: Tạo file App.js như sau:

/App.js

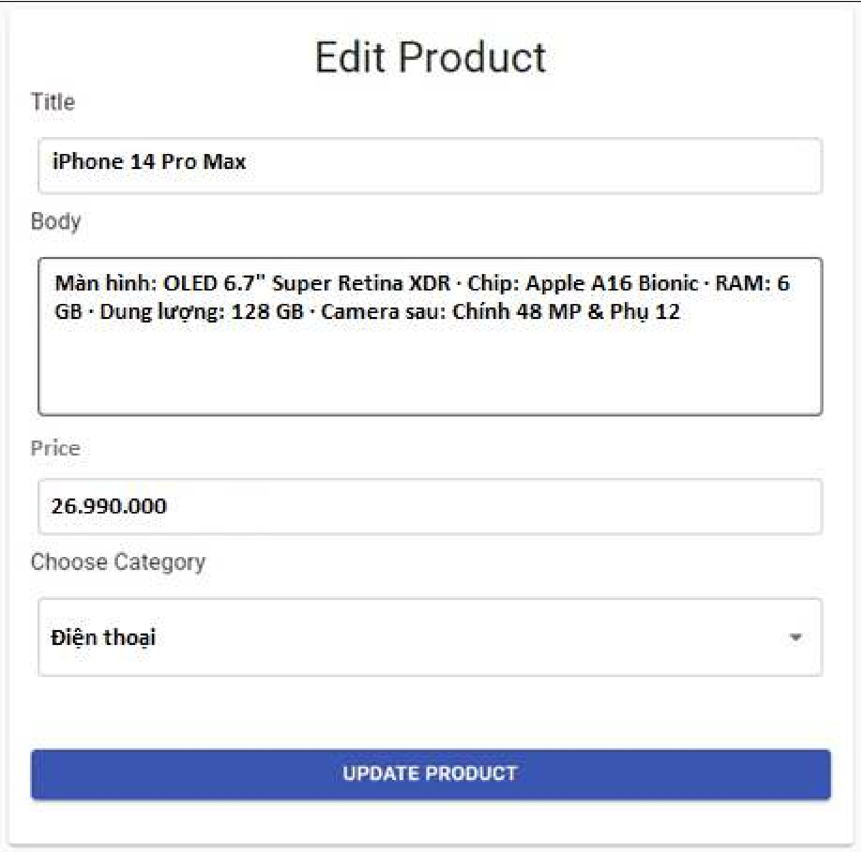
```
import React, { Component } from 'react';
import { Route } from 'react-router';
import Container from '@mui/material/Container';
import CssBaseline from '@mui/material/CssBaseline';
import MenuTop from './components/MenuTop';
import Home from './components/Home';
import Product from './components/Product';
import Category from './components/Category';
import EditProduct from './components/EditProduct';
export default class App extends Component {
  displayName = App.name;
  render() {
    return (
        <React.Fragment>
         <CssBaseline />
         <MenuTop />
         <Container maxWidth="md">
            <Route exact path='/' component={Home} />
            <Route path='/home' component={Home} />
            <Route path='/products' component={Product} />
            <Route path='/edit/product/:id' component={EditProduct} />
         </Container>
    </React.Fragment>
    );
  }
}
```

Bước 14: Sửa file index.js như sau:

/index.js

```
import './index.css';
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
import reportWebVitals from './reportWebVitals';
const rootElement = document.getElementById('root');

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  rootElement);

reportWebVitals();
```

Bước 15: Thực hiện lệnh **npm start** để Start FrontEnd trong Visual Code



Edit Product

Title

iPhone 14 Pro Max

Body

Màn hình: OLED 6.7" Super Retina XDR · Chip: Apple A16 Bionic · RAM: 6 GB · Dung lượng: 128 GB · Camera sau: Chính 48 MP & Phụ 12

Price

26.990.000

Choose Category

Điện thoại

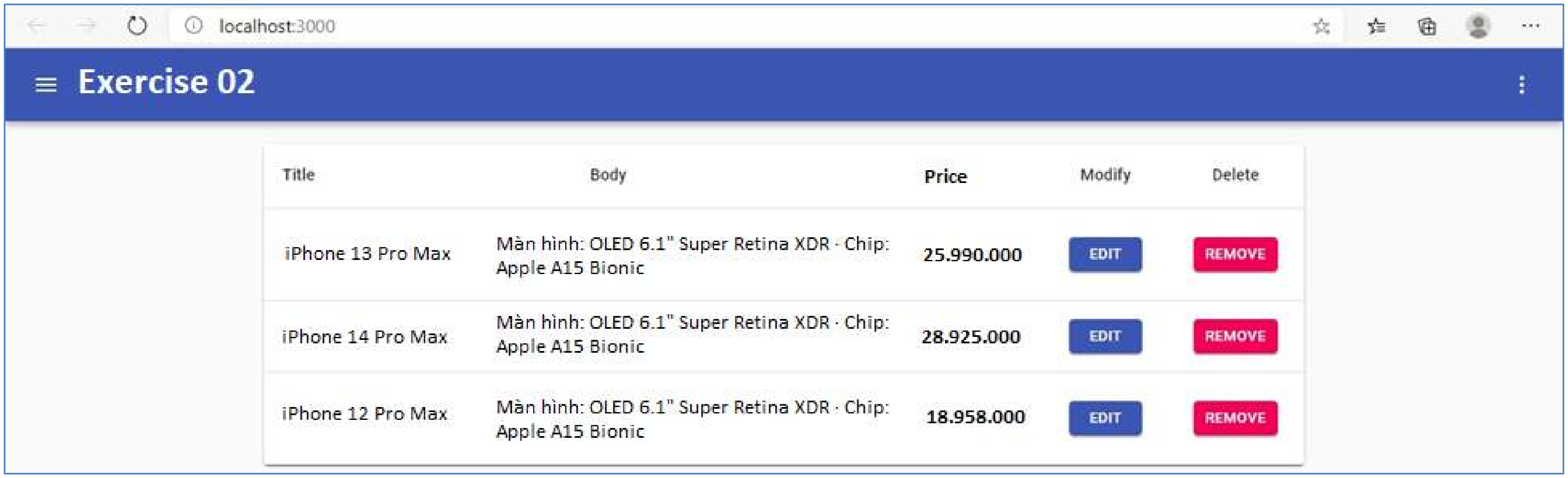
Exercise 02

Mục tiêu: Tạo và quản lý ứng dụng **Spring Boot Rest API** kết hợp với ReactJS

Yêu cầu: Thực hiện các chức năng sau:

- ✓ Tạo bảng
- ✓ Thêm dữ liệu
- ✓ Xóa dữ liệu
- ✓ Sửa dữ liệu
- ✓ Truy vấn dữ liệu

Hướng dẫn:



Title	Body	Price	Modify	Delete
iPhone 13 Pro Max	Màn hình: OLED 6.1" Super Retina XDR · Chip: Apple A15 Bionic	25.990.000	EDIT	REMOVE
iPhone 14 Pro Max	Màn hình: OLED 6.1" Super Retina XDR · Chip: Apple A15 Bionic	28.925.000	EDIT	REMOVE
iPhone 12 Pro Max	Màn hình: OLED 6.1" Super Retina XDR · Chip: Apple A15 Bionic	18.958.000	EDIT	REMOVE