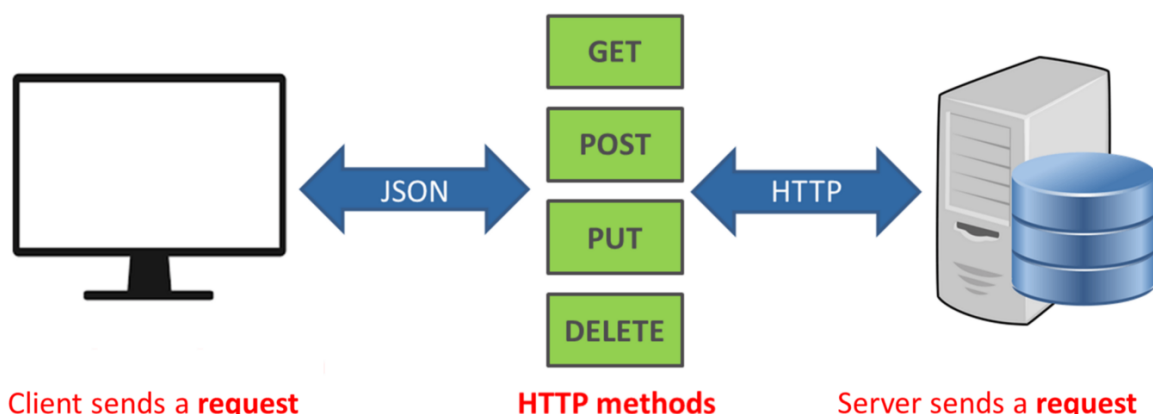


Tarefa: Construir uma API para cadastro, consulta, edição e exclusão de produtos

Projetos do tipo API são desenvolvidos de acordo com um conjunto de especificações definidos pelo padrão **REST**.

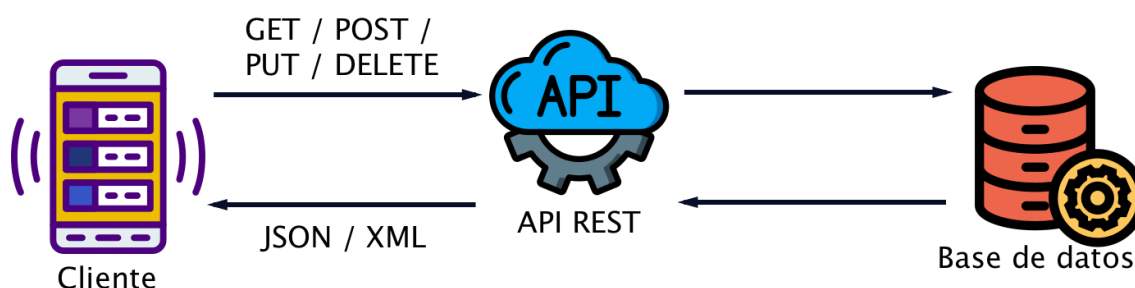
REST

Conjunto de especificações que definem como uma API Web deverá ser construída e distribuída.



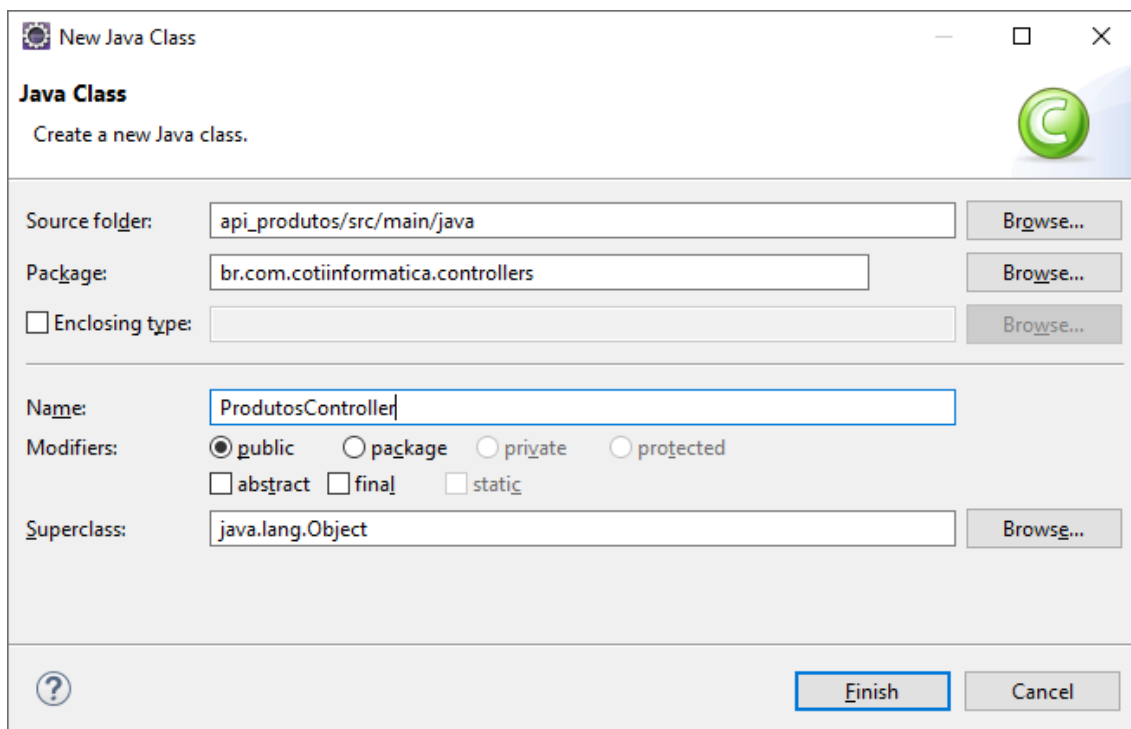
São características de uma **API REST**:

- Os Serviços de uma API deverão ser disponibilizados através de **ENDPOINTS**, por exemplo: **/api/produtos**
- Os ENDPOINTS disponibilizam serviços tais como: cadastro, consulta, edição, exclusão etc. Para cada tipo de operação feita por um ENDPOINT, precisamos definir um método ou verbo. Por exemplo:
 - POST** /api/produtos (Cadastrar produto)
 - PUT** /api/produtos (Atualizar produto)
 - DELETE** /api/produtos (Excluir produto)
 - GET** /api/produtos (Consultar produtos)
- Todos os dados que a API recebe (requisição) ou que a API retorna (resposta) são em formato **JSON**.



Primeiro, vamos criar um **controlador** para definir o **ENDPOINT** de serviços para api de produtos:

Implementar a API REST



@RestController

Annotation do Spring Boot capaz de qualificar uma classe como um controlador de API, que será desenvolvida no padrão REST.

```
package br.com.cotiinformatica.controllers;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController
```

```
public class ProdutosController {  
  
}
```

É muito importante documentarmos cada ENDPOINT e método da API para que o framework Swagger possa exibir uma documentação detalhada sobre os serviços que estamos construindo. Exemplo:

```
package br.com.cotiinformatica.controllers;
```

```
import org.springframework.web.bind.annotation.RestController;
```

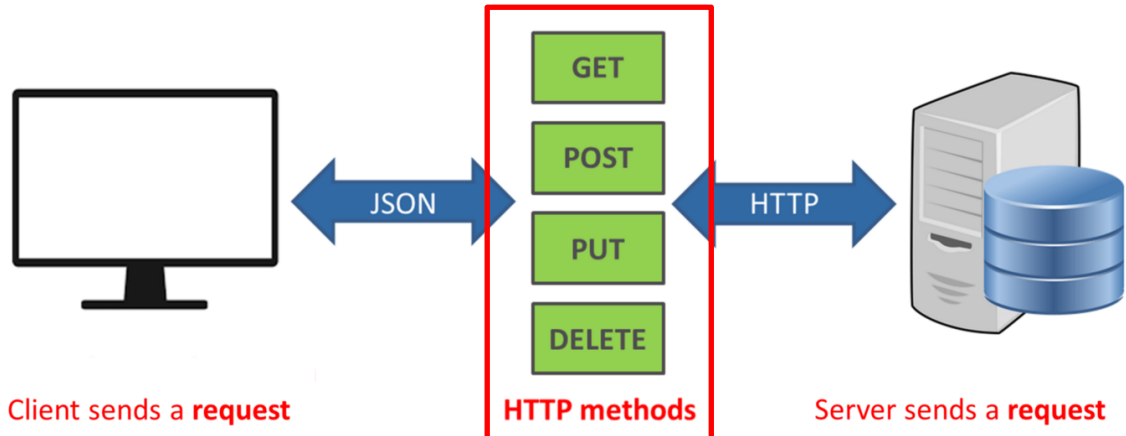
```
import io.swagger.annotations.Api;
```

```
@Api(tags = "Controle de produtos")  
@RestController
```

```
public class ProdutosController {  
  
}
```

Criando os métodos da API:

Operações POST, PUT, DELETE e GET



```
package br.com.cotiinformatica.controllers;
```

```
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RestController;
```

```
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiOperation;
```

```
@Api(tags = "Controle de produtos")
```

```
@RestController
```

```
public class ProdutosController {
```

```
    @ApiOperation("Serviço para cadastro de produto.")
```

```
    @PostMapping("/api/produtos")
```

```
    public ResponseEntity<String> post() {  
        return null;  
    }
```

```
    @ApiOperation("Serviço para atualização de produto.")
```

```
    @PutMapping("/api/produtos")
```

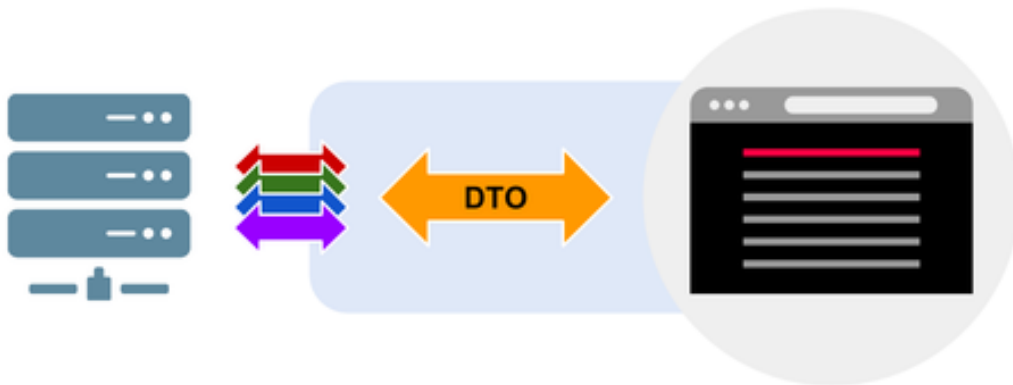
```
    public ResponseEntity<String> put() {  
        return null;  
    }
```

```
@ApiOperation("Serviço para exclusão de produto.")
@DeleteMapping("/api/produtos")
public ResponseEntity<String> delete() {
    return null;
}

@ApiOperation("Serviço para consulta de produtos.")
@GetMapping("/api/produtos")
public ResponseEntity<String> getAll() {
    return null;
}
}
```

DTO – Data Transfer Objects

Padrão utilizado para definição de objeto de transferência de dados.



Na API, para cada requisição ou resposta definida pela API iremos criar uma classe DTO que modele esta requisição ou resposta.

Por exemplo, vamos criar DTOS para definir os dados que deverão ser enviados para a API no cadastro de um produto.

/dtos/**PostProdutosDTO.java**

Dados que deverão ser enviados para a API na requisição de cadastro de produtos: **POST /api/produtos**

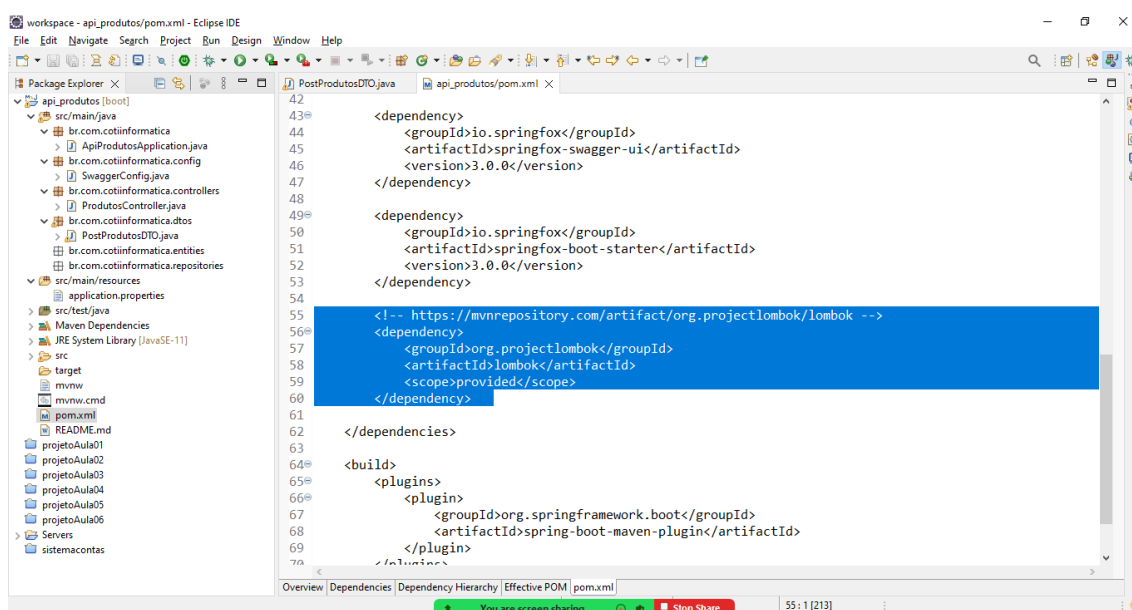
```
package br.com.cotiinformatica.dtos;

public class PostProdutosDTO {

    private String nome;
    private Double preco;
    private Integer quantidade;
}
```

Adicionando o LOMBOK: /pom.xml

```
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <scope>provided</scope>
</dependency>
```



package br.com.cotiinformatica.dtos;

```
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
```

```
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class PostProdutosDTO {
```

```
    private String nome;
    private Double preco;
    private Integer quantidade;
```

```
}
```

Voltando no controlador:

```
package br.com.cotiinformatica.controllers;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.dtos.PostProdutosDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(tags = "Controle de produtos")
@RestController
public class ProdutosController {

    @ApiOperation("Serviço para cadastro de produto.")
    @PostMapping("/api/produtos")
    public ResponseEntity<String> post(@RequestBody PostProdutosDTO dto) {
        return null;
    }

    @ApiOperation("Serviço para atualização de produto.")
    @PutMapping("/api/produtos")
    public ResponseEntity<String> put() {
        return null;
    }

    @ApiOperation("Serviço para exclusão de produto.")
    @DeleteMapping("/api/produtos")
    public ResponseEntity<String> delete() {
        return null;
    }

    @ApiOperation("Serviço para consulta de produtos.")
    @GetMapping("/api/produtos")
    public ResponseEntity<String> getAll() {
        return null;
    }
}
```

Criando um DTO para definir os dados que a API vai receber para atualizar um produto.

/dtos/**PutProdutosDTO.java**

```
package br.com.cotiinformatica.dtos;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
```

```
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class PutProdutosDTO {

    private Integer idProduto;
    private String nome;
    private Double preco;
    private Integer quantidade;
}
```

Voltando no controlador:

```
package br.com.cotiinformatica.controllers;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.dtos.PostProdutosDTO;
import br.com.cotiinformatica.dtos.PutProdutosDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(tags = "Controle de produtos")
@RestController
public class ProdutosController {

    @ApiOperation("Serviço para cadastro de produto.")
    @PostMapping("/api/produtos")
    public ResponseEntity<String> post(@RequestBody PostProdutosDTO dto) {
        return null;
    }

    @ApiOperation("Serviço para atualização de produto.")
    @PutMapping("/api/produtos")
    public ResponseEntity<String> put(@RequestBody PutProdutosDTO dto) {
        return null;
    }

    @ApiOperation("Serviço para exclusão de produto.")
    @DeleteMapping("/api/produtos")
    public ResponseEntity<String> delete() {
        return null;
    }

    @ApiOperation("Serviço para consulta de produtos.")
    @GetMapping("/api/produtos")
    public ResponseEntity<String> getAll() {
        return null;
    }
}
```

Definindo a assinatura para o método de exclusão de produto:

```
package br.com.cotiinformatica.controllers;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.dtos.PostProdutosDTO;
import br.com.cotiinformatica.dtos.PutProdutosDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(tags = "Controle de produtos")
@RestController
public class ProdutosController {

    @ApiOperation("Serviço para cadastro de produto.")
    @PostMapping("/api/produtos")
    public ResponseEntity<String> post(@RequestBody PostProdutosDTO dto) {
        return null;
    }

    @ApiOperation("Serviço para atualização de produto.")
    @PutMapping("/api/produtos")
    public ResponseEntity<String> put(@RequestBody PutProdutosDTO dto) {
        return null;
    }

    @ApiOperation("Serviço para exclusão de produto.")
    @DeleteMapping("/api/produtos/{id}")

    public ResponseEntity<String> delete

        (@PathVariable("id") Integer idProduto) {

        return null;
    }

    @ApiOperation("Serviço para consulta de produtos.")
    @GetMapping("/api/produtos")
    public ResponseEntity<String> getAll() {
        return null;
    }
}
```


Criando uma classe DTO para modelar os dados que serão retornados na consulta de produtos da API:

/dtos/**GetProdutosDTO.java**

```
package br.com.cotiinformatica.dtos;
```

```
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
import lombok.ToString;
```

```
@Setter
```

```
@Getter
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
@ToString
```

```
public class GetProdutosDTO {
```

```
    private Integer idProduto;
```

```
    private String nome;
```

```
    private Double preco;
```

```
    private Integer quantidade;
```

```
}
```

Voltando no controlador:

```
package br.com.cotiinformatica.controllers;
```

```
import java.util.List;
```

```
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.DeleteMapping;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.PathVariable;
```

```
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.PutMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
import br.com.cotiinformatica.dtos.GetProdutosDTO;
```

```
import br.com.cotiinformatica.dtos.PostProdutosDTO;
```

```
import br.com.cotiinformatica.dtos.PutProdutosDTO;
```

```
import io.swagger.annotations.Api;
```

```
import io.swagger.annotations.ApiOperation;
```

```
@Api(tags = "Controle de produtos")
```

```
@RestController
```

```
public class ProdutosController {
```

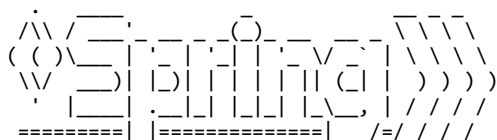
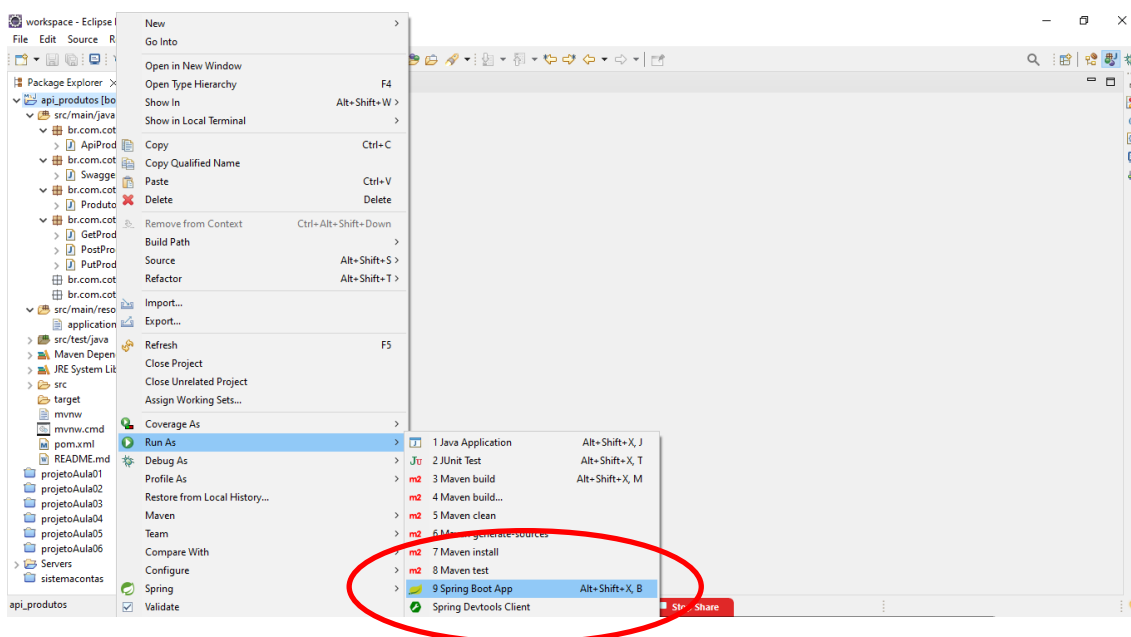
```
@ApiOperation("Serviço para cadastro de produto.")
@PostMapping("/api/produtos")
public ResponseEntity<GetProdutosDTO> post
(@RequestBody PostProdutosDTO dto) {
    return null;
}

@ApiOperation("Serviço para atualização de produto.")
@PutMapping("/api/produtos")
public ResponseEntity<GetProdutosDTO> put
(@RequestBody PutProdutosDTO dto) {
    return null;
}

@ApiOperation("Serviço para exclusão de produto.")
@DeleteMapping("/api/produtos/{id}")
public ResponseEntity<GetProdutosDTO> delete
(@PathVariable("id") Integer idProduto) {
    return null;
}

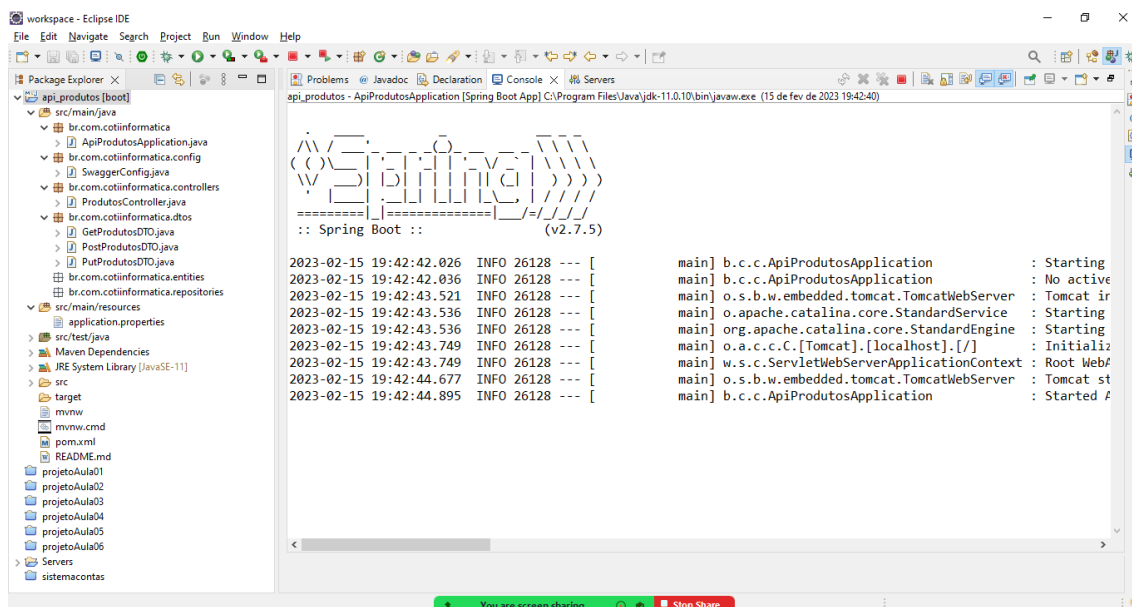
@ApiOperation("Serviço para consulta de produtos.")
@GetMapping("/api/produtos")
public ResponseEntity<List<GetProdutosDTO>> getAll() {
    return null;
}
}
```

Executando o projeto:

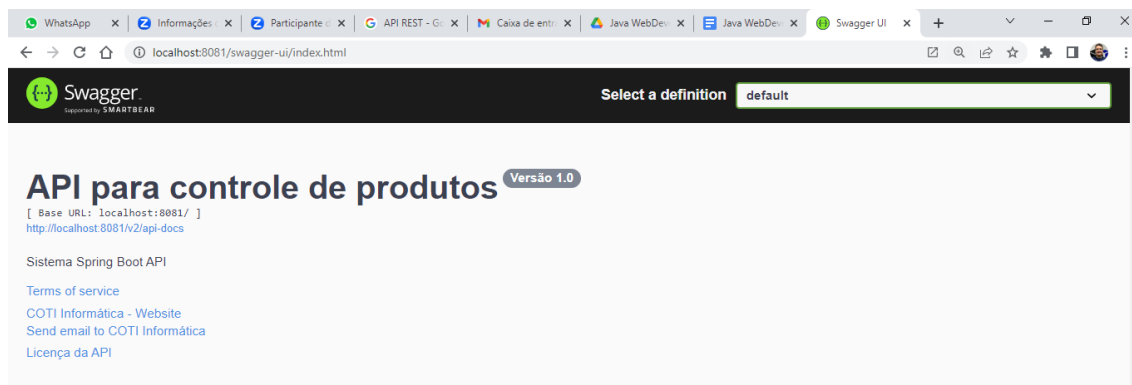


:: Spring Boot :: (v2.7.5)

```
2023-02-15 19:42:42.026 INFO 26128 --- [main] b.c.c.ApiProdutosApplication
: Starting ApiProdutosApplication using Java 11.0.10 on DESKTOP-P9F6D9F with PID 26128
(C:\Users\Samsung\Desktop\COTI - Aulas\2023 - Java WebDeveloper SQS 18h as 22h (Início em
09.01)\workspace\api_produtos\target\classes started by Samsung in C:\Users\Samsung\Desktop\COTI
- Aulas\2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)\workspace\api_produtos)
2023-02-15 19:42:42.036 INFO 26128 --- [main] b.c.c.ApiProdutosApplication
: No active profile set, falling back to 1 default profile: "default"
2023-02-15 19:42:43.521 INFO 26128 --- [main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8081 (http)
2023-02-15 19:42:43.536 INFO 26128 --- [main] o.apache.catalina.core.StandardService
: Starting service [Tomcat]
2023-02-15 19:42:43.536 INFO 26128 --- [main]
org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.68]
2023-02-15 19:42:43.749 INFO 26128 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/]
: Initializing Spring embedded WebApplicationContext
2023-02-15 19:42:43.749 INFO 26128 --- [main]
w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
in 1655 ms
2023-02-15 19:42:44.677 INFO 26128 --- [main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8081 (http) with context
path ''
2023-02-15 19:42:44.895 INFO 26128 --- [main] b.c.c.ApiProdutosApplication
: Started ApiProdutosApplication in 3.306 seconds (JVM running for 4.407)
```



<http://localhost:8081/swagger-ui/index.html>





Swagger
SMARTBEAR

Select a definition default

API para controle de produtos Versão 1.0

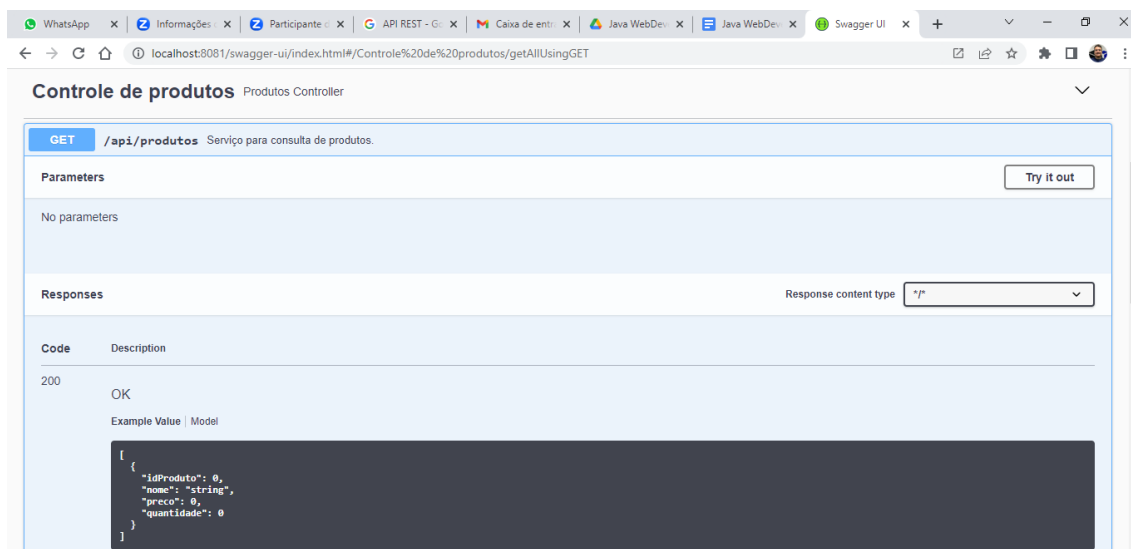
[Base URL: localhost:8081/]
http://localhost:8081/v2/api-docs

Sistema Spring Boot API

[Terms of service](#)
[COTI Informática - Website](#)
[Send email to COTI Informática](#)
[Licença da API](#)

Controle de produtos Produtos Controller

- GET** /api/produtos Serviço para consulta de produtos.
- POST** /api/produtos Serviço para cadastro de produto.
- PUT** /api/produtos Serviço para atualização de produto.
- DELETE** /api/produtos/{id} Serviço para exclusão de produto.



Controle de produtos Produtos Controller

- GET** /api/produtos Serviço para consulta de produtos.

Parameters Try it out

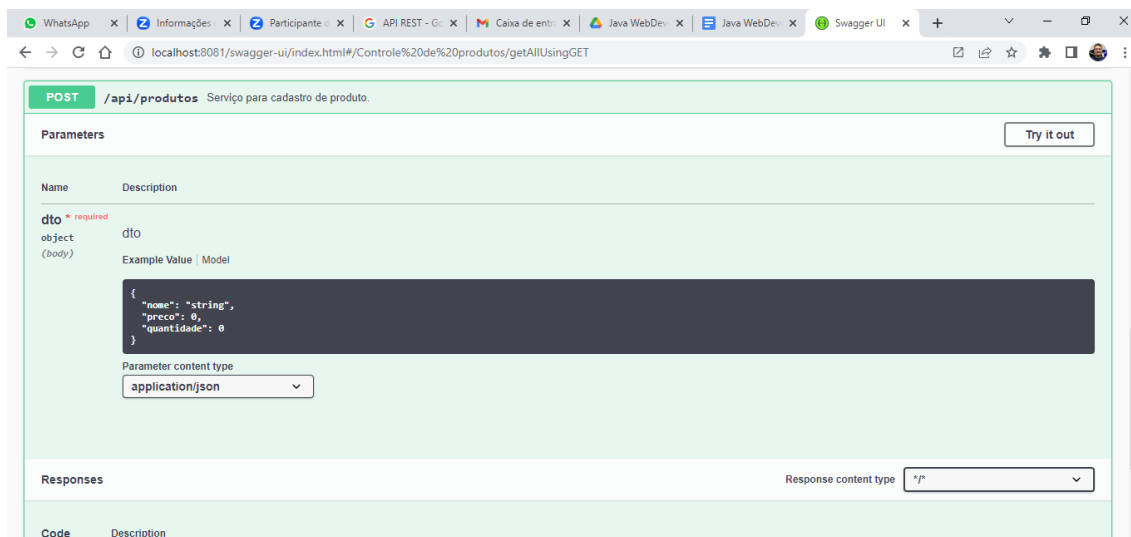
No parameters

Responses Response content type: */*

Code	Description
200	OK

Example Value | Model

```
{
  "idProduto": 0,
  "nome": "string",
  "preco": 0,
  "quantidade": 0
}
```



- POST** /api/produtos Serviço para cadastro de produto.

Parameters Try it out

Name	Description
dto * required	dto

object (body)

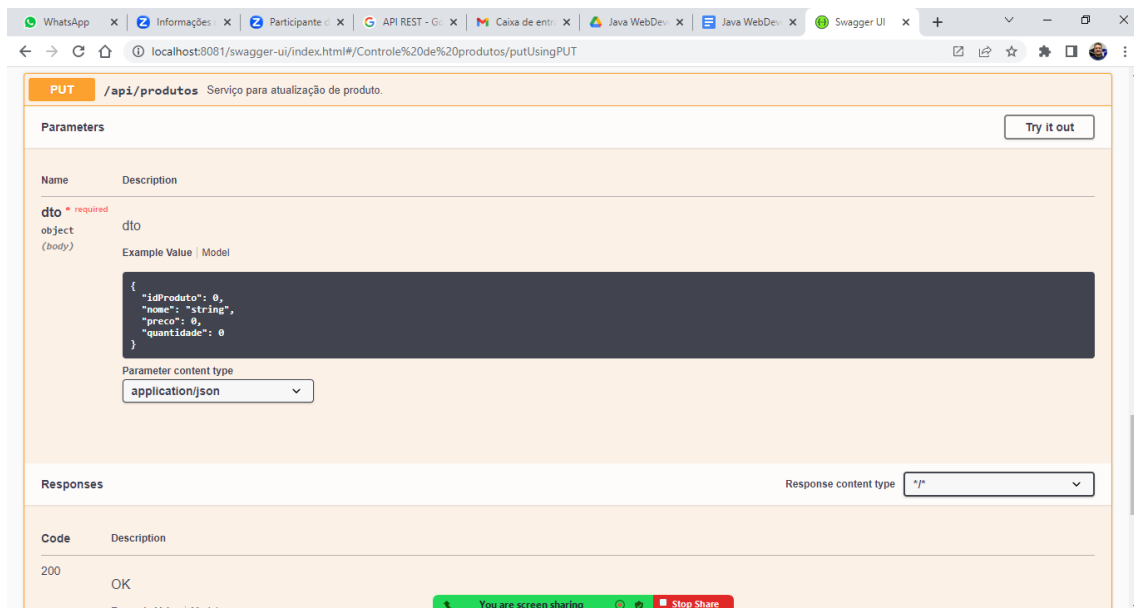
Example Value | Model

```
{
  "nome": "string",
  "preco": 0,
  "quantidade": 0
}
```

Parameter content type: application/json

Responses Response content type: */*

Code	Description
200	OK



PUT /api/produtos Serviço para atualização de produto.

Parameters

Try it out

Name	Description
dto * required	dto

object (body)

Example Value | Model

```
{
  "idProduto": 0,
  "nome": "string",
  "preco": 0,
  "quantidade": 0
}
```

Parameter content type

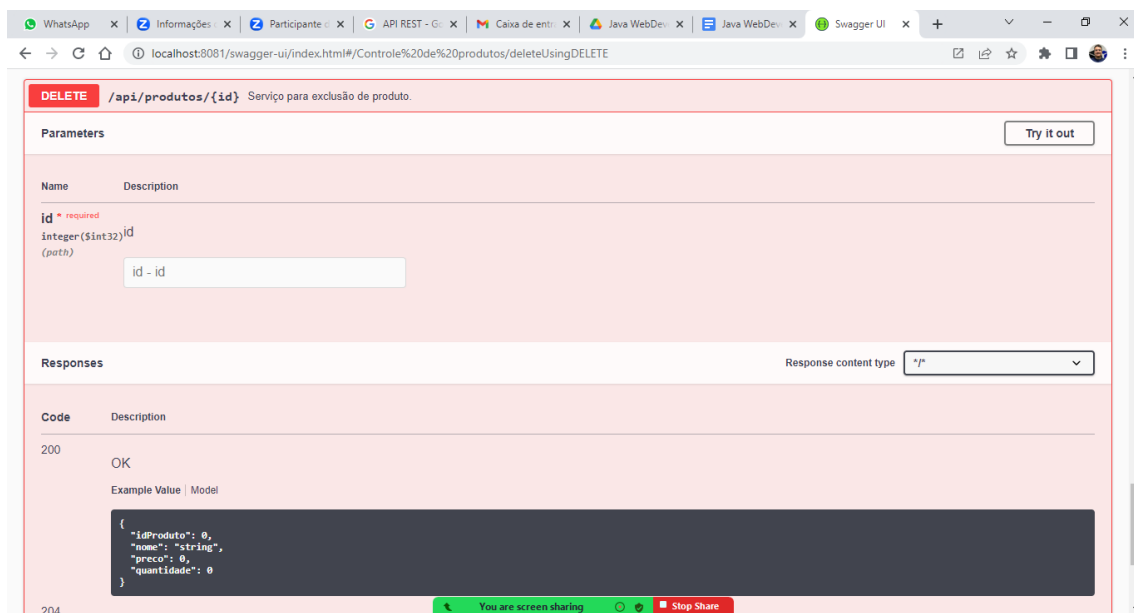
application/json

Responses

Response content type */*

Code	Description
200	OK

Example Value | Model



DELETE /api/produtos/{id} Serviço para exclusão de produto.

Parameters

Try it out

Name	Description
id * required	id

integer(\$int32) (path)

id - id

Responses

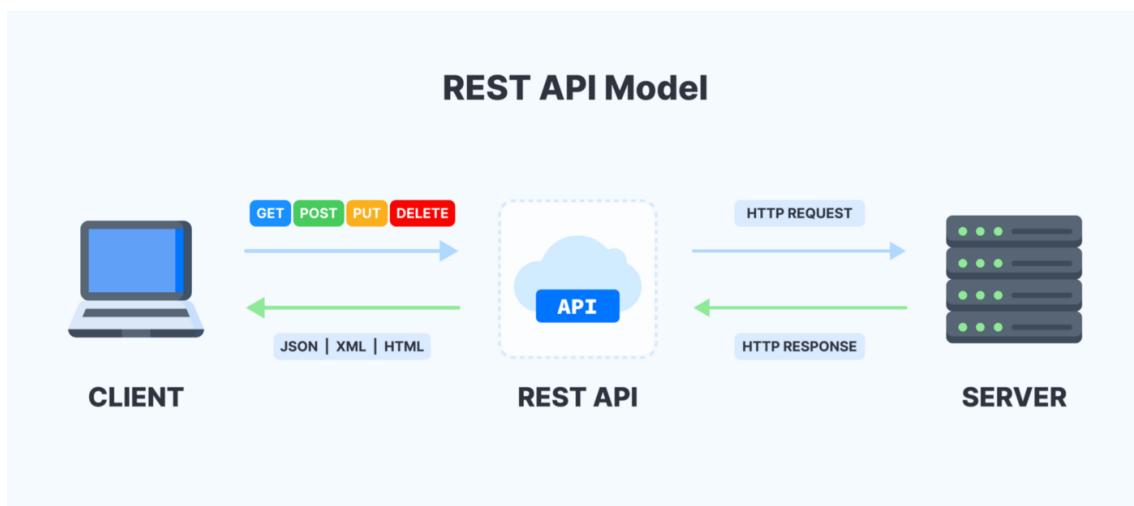
Response content type */*

Code	Description
200	OK

Example Value | Model

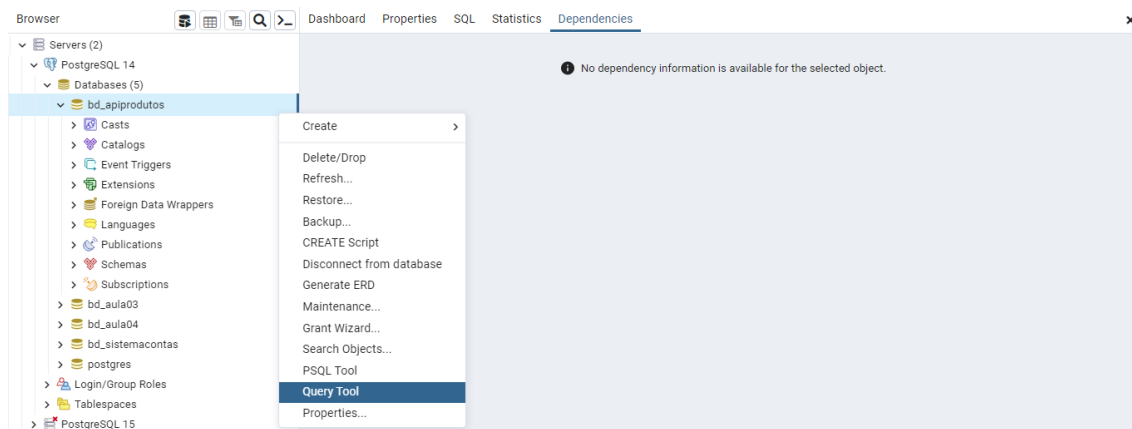
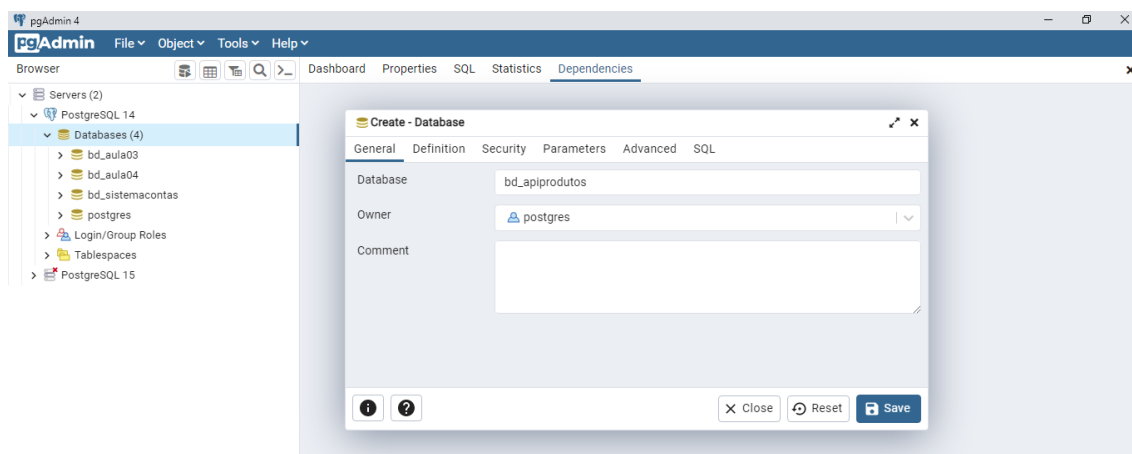
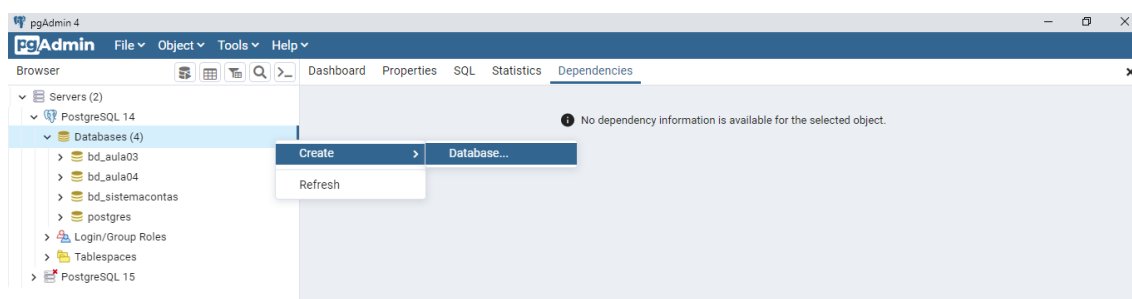
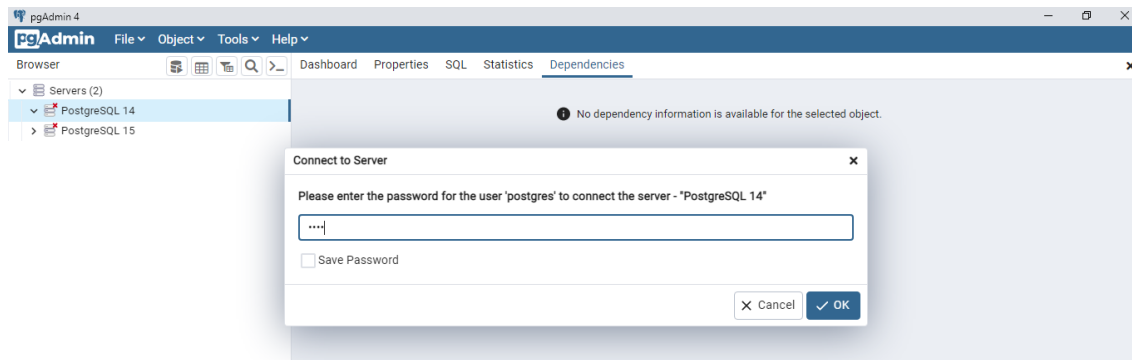
```
{
  "idProduto": 0,
  "nome": "string",
  "preco": 0,
  "quantidade": 0
}
```

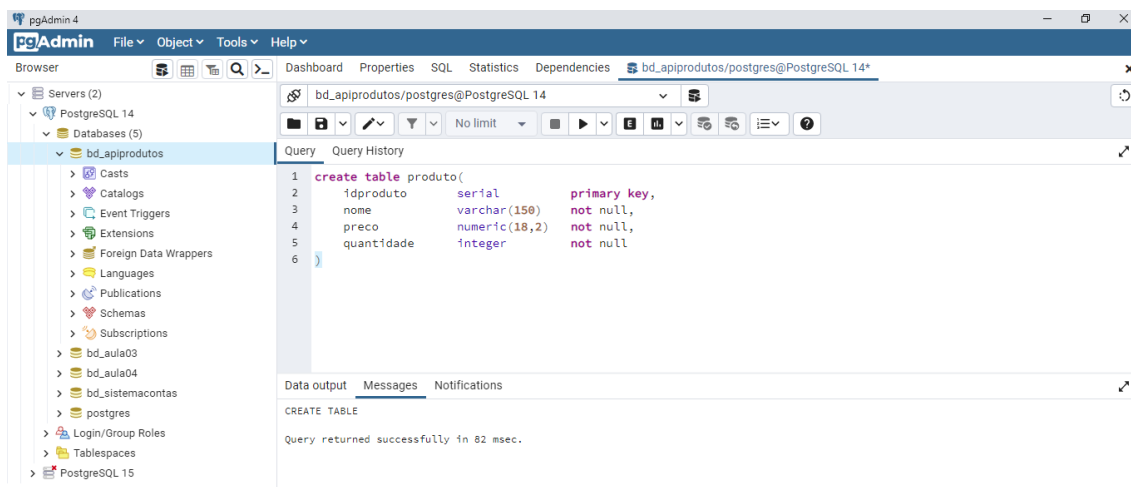
204



Construindo o banco de dados:

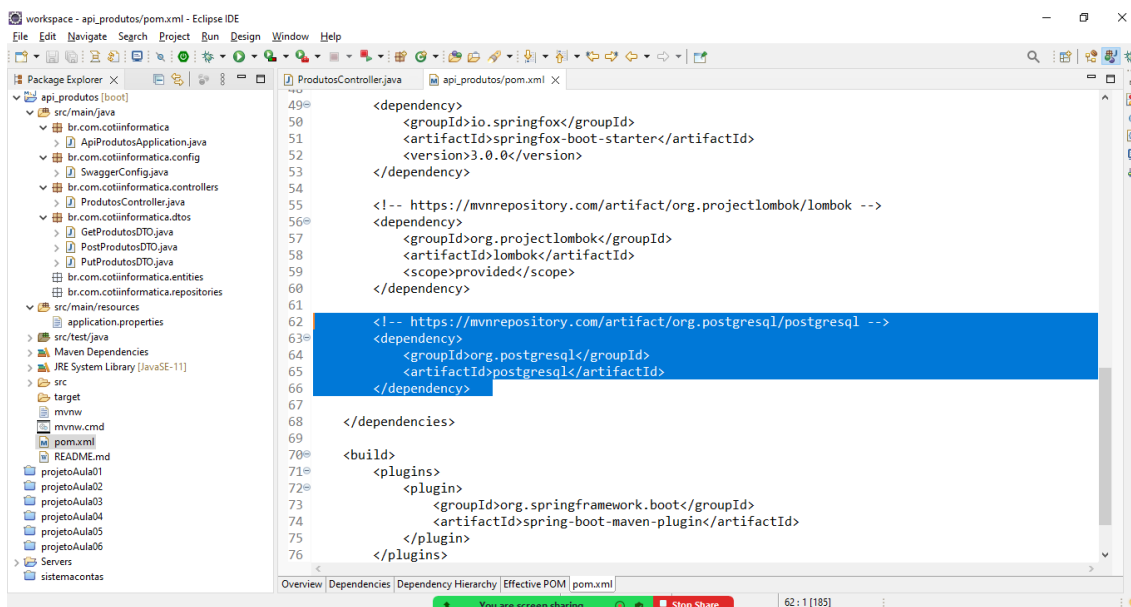
Utilizando o PostgreSQL:



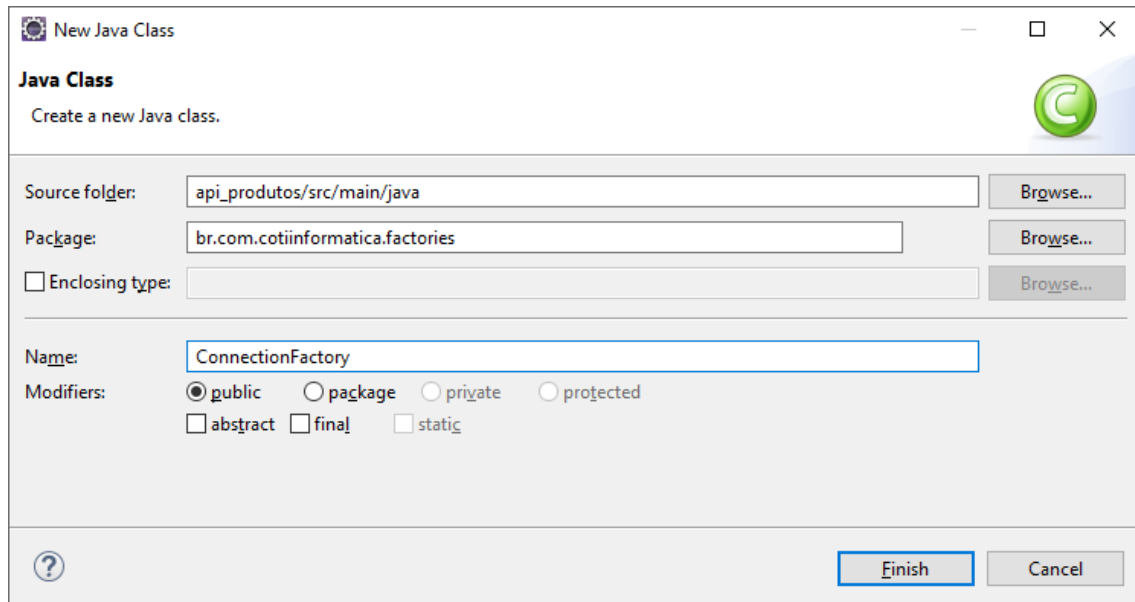


```
create table produto(
  idproduto          serial          primary key,
  nome               varchar(150)    not null,
  preco              numeric(18,2)   not null,
  quantidade         integer         not null
)
```

Adicionando no projeto API a biblioteca para acesso ao PostgreSQL:
/pom.xml



```
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
</dependency>
```



```
package br.com.cotiinformatica.factories;

import java.sql.Connection;
import java.sql.DriverManager;

public class ConnectionFactory {

    private static final String DRIVER = "org.postgresql.Driver";
    private static final String URL = "jdbc:postgresql:
        //localhost:5432/bd_apiprodutos";
    private static final String USER = "postgres";
    private static final String PASS = "coti";

    public static Connection getConnection() throws Exception {
        Class.forName(DRIVER);
        return DriverManager.getConnection(URL, USER, PASS);
    }
}
```

Criando a entidade:

/entities/**Produto.java**

```
package br.com.cotiinformatica.entities;

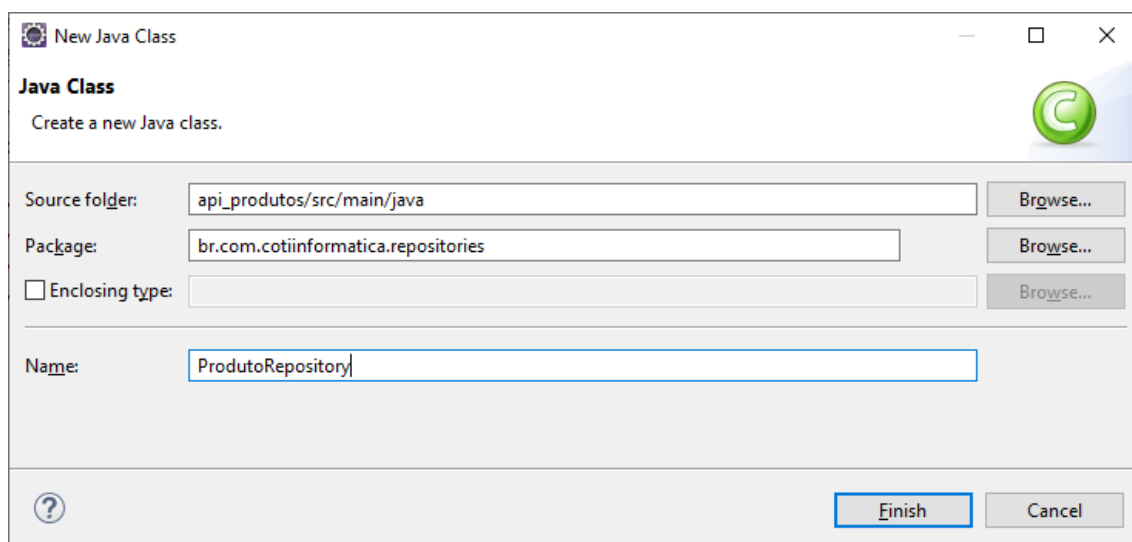
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
```



```
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Produto {

    private Integer idProduto;
    private String nome;
    private Double preco;
    private Integer quantidade;
}
```

Desenvolvendo a classe de repositório para produto:
/repositories/**ProdutoRepository.java**



```
package br.com.cotiinformatica.repositories;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import br.com.cotiinformatica.entities.Produto;
import br.com.cotiinformatica.factories.ConnectionFactory;

public class ProdutoRepository {

    public void create(Produto produto) throws Exception {

        Connection connection = ConnectionFactory.getConnection();
```

```
PreparedStatement statement = connection
    .prepareStatement("insert into produto(nome,
        preco, quantidade) values(?,?,?)");
statement.setString(1, produto.getNome());
statement.setDouble(2, produto.getPreco());
statement.setInt(3, produto.getQuantidade());
statement.execute();

statement = connection.prepareStatement
    ("select lastval() as idproduto");
ResultSet resultSet = statement.executeQuery();

if(resultSet.next())
    produto.setIdProduto(resultSet.getInt("idproduto"));

connection.close();
}

public void update(Produto produto) throws Exception {

    Connection connection = ConnectionFactory.getConnection();

    PreparedStatement statement = connection
        .prepareStatement("update produto set nome=?,
            preco=?, quantidade=? where idproduto=?");
    statement.setString(1, produto.getNome());
    statement.setDouble(2, produto.getPreco());
    statement.setInt(3, produto.getQuantidade());
    statement.setInt(4, produto.getIdProduto());
    statement.execute();

    connection.close();
}

public void delete(Integer idProduto) throws Exception {

    Connection connection = ConnectionFactory.getConnection();

    PreparedStatement statement = connection
        .prepareStatement
            ("delete from produto where idproduto=?");
    statement.setInt(1, idProduto);
    statement.execute();
}
```

```
        connection.close();
    }

    public List<Produto> findAll() throws Exception {

        Connection connection = ConnectionFactory.getConnection();

        PreparedStatement statement = connection
            .prepareStatement("select * from produto");
        ResultSet resultSet = statement.executeQuery();

        List<Produto> lista = new ArrayList<Produto>();

        while(resultSet.next()) {

            Produto produto = new Produto();

            produto.setIdProduto(resultSet.getInt("idproduto"));
            produto.setNome(resultSet.getString("nome"));
            produto.setPreco(resultSet.getDouble("preco"));
            produto.setQuantidade
                (resultSet.getInt("quantidade"));

            lista.add(produto);
        }

        connection.close();
        return lista;
    }

    public Produto findById(Integer idProduto) throws Exception {

        Connection connection = ConnectionFactory.getConnection();

        PreparedStatement statement = connection.prepareStatement
            ("select * from produto where idproduto=?");
        statement.setInt(1, idProduto);
        ResultSet resultSet = statement.executeQuery();

        Produto produto = null;

        if(resultSet.next()) {

            produto = new Produto();
```

```

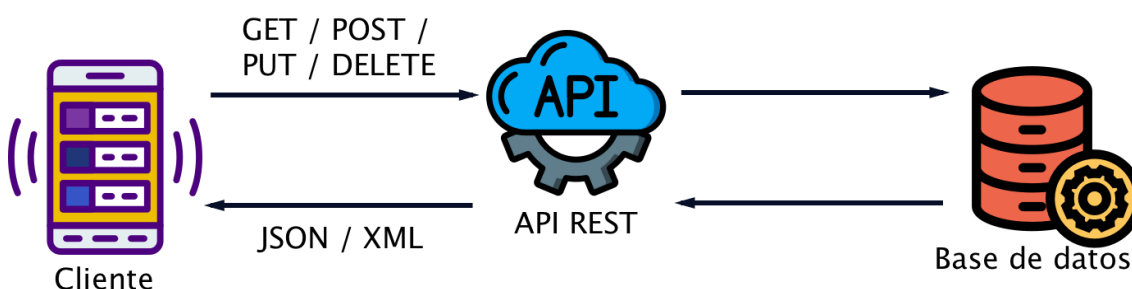
        produto.setIdProduto(resultSet.getInt("idproduto"));
        produto.setNome(resultSet.getString("nome"));
        produto.setPreco(resultSet.getDouble("preco"));
        produto.setQuantidade
            (resultSet.getInt("quantidade"));
    }

    connection.close();
    return produto;
}
}

```

Voltando no controlador para implementarmos os serviços da API:

/controllers/**ProdutosController.java**



Desenvolvendo o cadastro do produto:

```

package br.com.cotiinformatica.controllers;

import java.util.List;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.dtos.GetProdutosDTO;
import br.com.cotiinformatica.dtos.PostProdutosDTO;
import br.com.cotiinformatica.dtos.PutProdutosDTO;
import br.com.cotiinformatica.entities.Produto;
import br.com.cotiinformatica.repositories.ProdutoRepository;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

```

```
@Api(tags = "Controle de produtos")
@RestController
public class ProdutosController {

    @ApiOperation("Serviço para cadastro de produto.")
    @PostMapping("/api/produtos")
    public ResponseEntity<GetProdutosDTO> post
        (@RequestBody PostProdutosDTO dto) {

        GetProdutosDTO result = new GetProdutosDTO();

        try {

            Produto produto = new Produto();

            produto.setNome(dto.getNome());
            produto.setPreco(dto.getPreco());
            produto.setQuantidade(dto.getQuantidade());

            ProdutoRepository produtoRepository
                = new ProdutoRepository();
            produtoRepository.create(produto);

            //retornando os dados do produto cadastrado
            result.setIdProduto(produto.getIdProduto());
            result.setNome(produto.getNome());
            result.setPreco(produto.getPreco());
            result.setQuantidade(produto.getQuantidade());

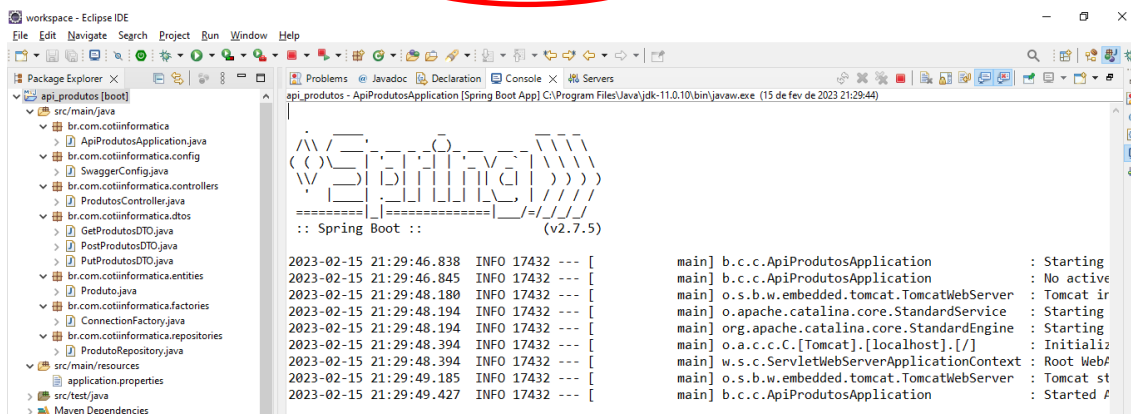
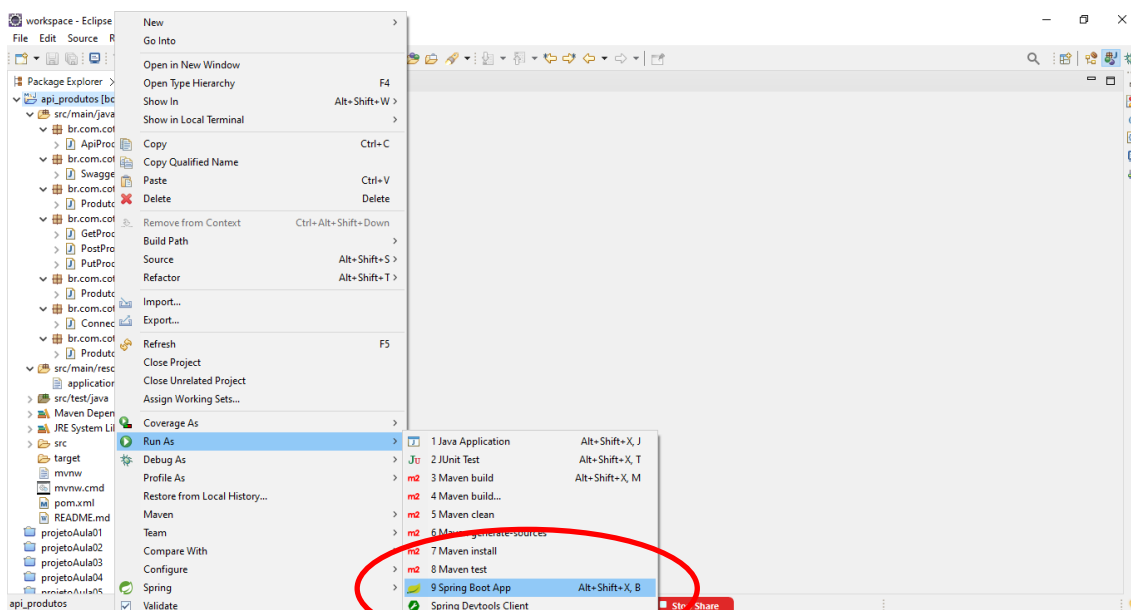
            return ResponseEntity.status
                (HttpStatus.CREATED).body(result);
        }
        catch (Exception e) {
            e.printStackTrace();
            return ResponseEntity.status
                (HttpStatus.INTERNAL_SERVER_ERROR).body(result);
        }
    }

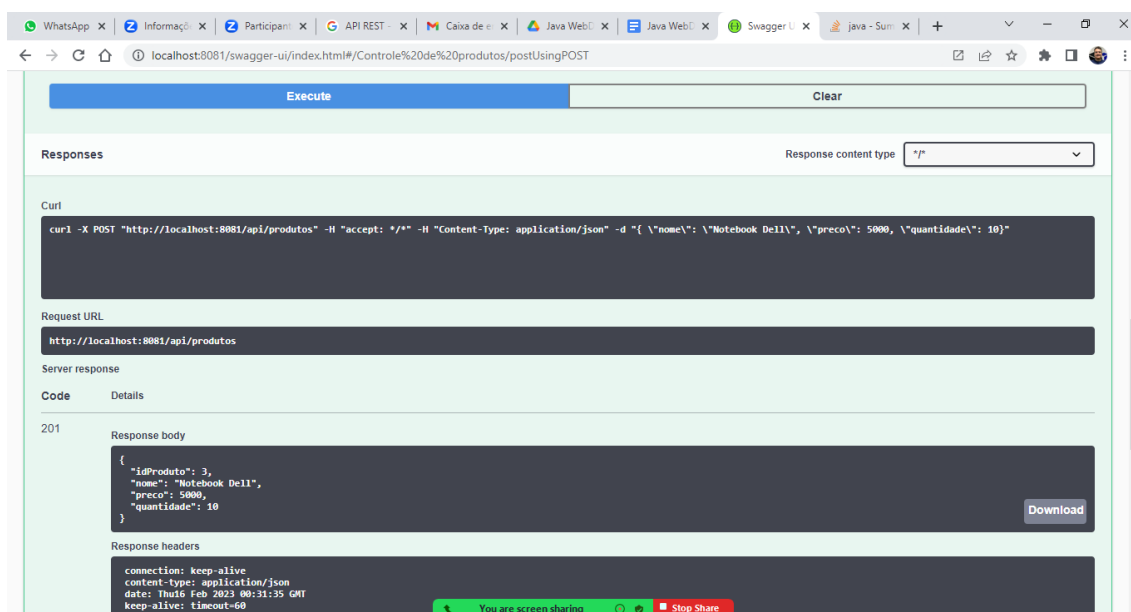
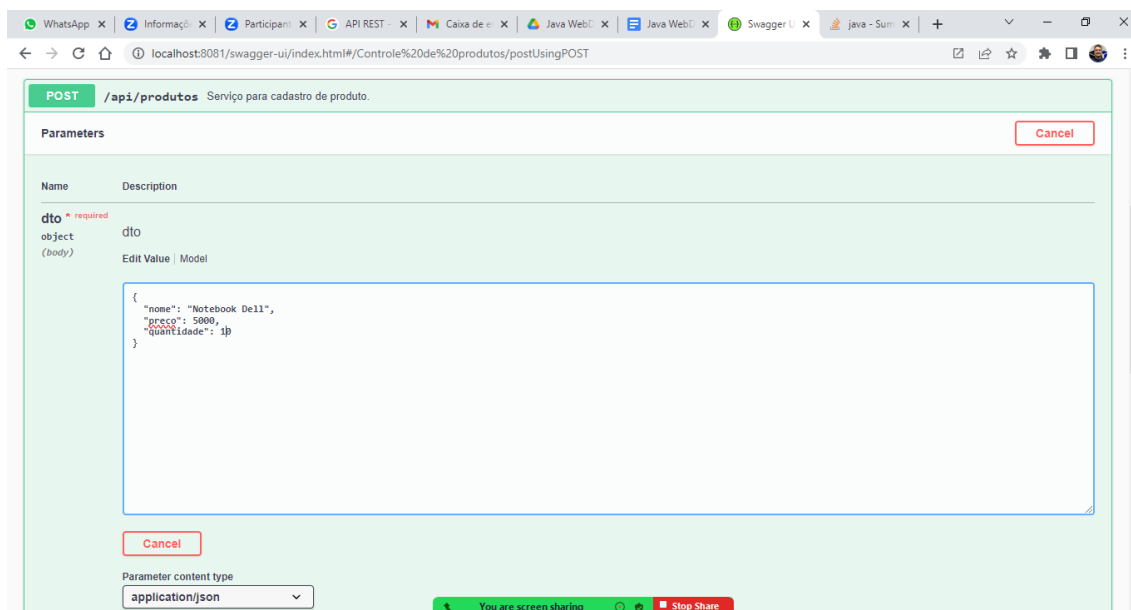
    @ApiOperation("Serviço para atualização de produto.")
    @PutMapping("/api/produtos")
    public ResponseEntity<GetProdutosDTO> put
        (@RequestBody PutProdutosDTO dto) {
        return null;
    }

    @ApiOperation("Serviço para exclusão de produto.")
    @DeleteMapping("/api/produtos/{id}")
    public ResponseEntity<GetProdutosDTO> delete
        (@PathVariable("id") Integer idProduto) {
        return null;
    }
}
```

```
@ApiOperation("Serviço para consulta de produtos.")
@GetMapping("/api/produtos")
public ResponseEntity<List<GetProdutosDTO>> getAll() {
    return null;
}
```

Executando e testando através do Swagger:





Implementando os demais métodos da API:

```
package br.com.cotiinformatica.controllers;
```

```
import java.util.ArrayList;  
import java.util.List;
```

```
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;
```

```
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.dtos.GetProdutosDTO;
import br.com.cotiinformatica.dtos.PostProdutosDTO;
import br.com.cotiinformatica.dtos.PutProdutosDTO;
import br.com.cotiinformatica.entities.Produto;
import br.com.cotiinformatica.repositories.ProdutoRepository;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(tags = "Controle de produtos")
@RestController
public class ProdutosController {

    @ApiOperation("Serviço para cadastro de produto.")
    @PostMapping("/api/produtos")
    public ResponseEntity<GetProdutosDTO> post
        (@RequestBody PostProdutosDTO dto) {

        GetProdutosDTO result = new GetProdutosDTO();

        try {

            Produto produto = new Produto();

            produto.setNome(dto.getNome());
            produto.setPreco(dto.getPreco());
            produto.setQuantidade(dto.getQuantidade());

            ProdutoRepository produtoRepository = new ProdutoRepository();
            produtoRepository.create(produto);

            //retornando os dados do produto cadastrado
            result.setIdProduto(produto.getIdProduto());
            result.setNome(produto.getNome());
            result.setPreco(produto.getPreco());
            result.setQuantidade(produto.getQuantidade());

            return ResponseEntity.status(HttpStatus.CREATED).body(result);
        }
        catch(Exception e) {
            e.printStackTrace();
            return ResponseEntity.status
                (HttpStatus.INTERNAL_SERVER_ERROR).body(result);
        }
    }

    @ApiOperation("Serviço para atualização de produto.")
    @PutMapping("/api/produtos")
    public ResponseEntity<GetProdutosDTO> put(@RequestBody PutProdutosDTO dto) {

        GetProdutosDTO result = new GetProdutosDTO();
```



```
try {

    ProdutoRepository produtoRepository = new ProdutoRepository();
    Produto produto = produtoRepository.findById(dto.getIdProduto());

    if(produto != null) {

        produto.setNome(dto.getNome());
        produto.setPreco(dto.getPreco());
        produto.setQuantidade(dto.getQuantidade());

        produtoRepository.update(produto);

        result.setIdProduto(produto.getIdProduto());
        result.setNome(produto.getNome());
        result.setPreco(produto.getPreco());
        result.setQuantidade(produto.getQuantidade());

        return ResponseEntity.status(HttpStatus.OK).body(result);
    }
    else {
        return ResponseEntity.status
            (HttpStatus.BAD_REQUEST).body(result);
    }
}
catch(Exception e) {
    e.printStackTrace();
    return ResponseEntity.status
        (HttpStatus.INTERNAL_SERVER_ERROR).body(result);
}
}

@ApiOperation("Serviço para exclusão de produto.")
@DeleteMapping("/api/produtos/{id}")
public ResponseEntity<GetProdutosDTO> delete(@PathVariable("id")
Integer idProduto) {

    GetProdutosDTO result = new GetProdutosDTO();

    try {

        ProdutoRepository produtoRepository = new ProdutoRepository();
        Produto produto = produtoRepository.findById(idProduto);

        if(produto != null) {

            produtoRepository.delete(idProduto);

            result.setIdProduto(produto.getIdProduto());
            result.setNome(produto.getNome());
            result.setPreco(produto.getPreco());
            result.setQuantidade(produto.getQuantidade());
```

```
        return ResponseEntity.status(HttpStatus.OK).body(result);
    }
    else {
        return ResponseEntity.status
            (HttpStatus.BAD_REQUEST).body(result);
    }
}
catch(Exception e) {
    e.printStackTrace();
    return ResponseEntity.status
        (HttpStatus.INTERNAL_SERVER_ERROR).body(result);
}

}

@ApiOperation("Serviço para consulta de produtos.")
@GetMapping("/api/produtos")
public ResponseEntity<List<GetProdutosDTO>> getAll() {

    List<GetProdutosDTO> lista = new ArrayList<GetProdutosDTO>();

    try {

        ProdutoRepository produtoRepository = new ProdutoRepository();
        for(Produto produto : produtoRepository.findAll()) {

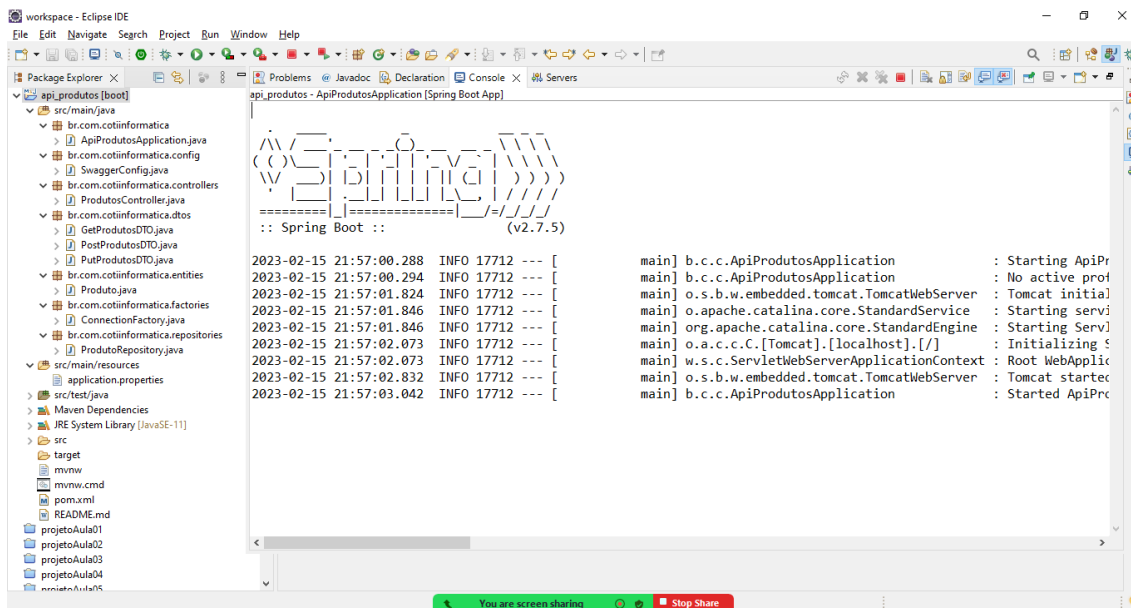
            GetProdutosDTO dto = new GetProdutosDTO();
            dto.setIdProduto(produto.getIdProduto());
            dto.setNome(produto.getNome());
            dto.setPreco(produto.getPreco());
            dto.setQuantidade(produto.getQuantidade());

            lista.add(dto);
        }

        return ResponseEntity.status(HttpStatus.OK).body(lista);
    }
    catch(Exception e) {
        e.printStackTrace();
        return ResponseEntity.status
            (HttpStatus.INTERNAL_SERVER_ERROR).body(lista);
    }
}

}
```

Executando e testando:

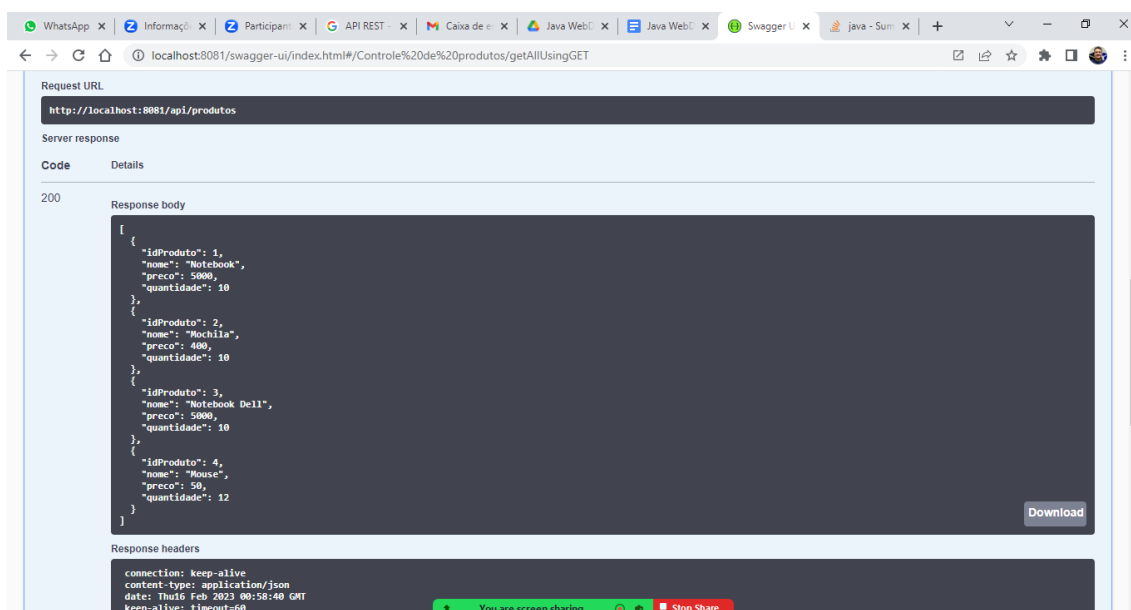
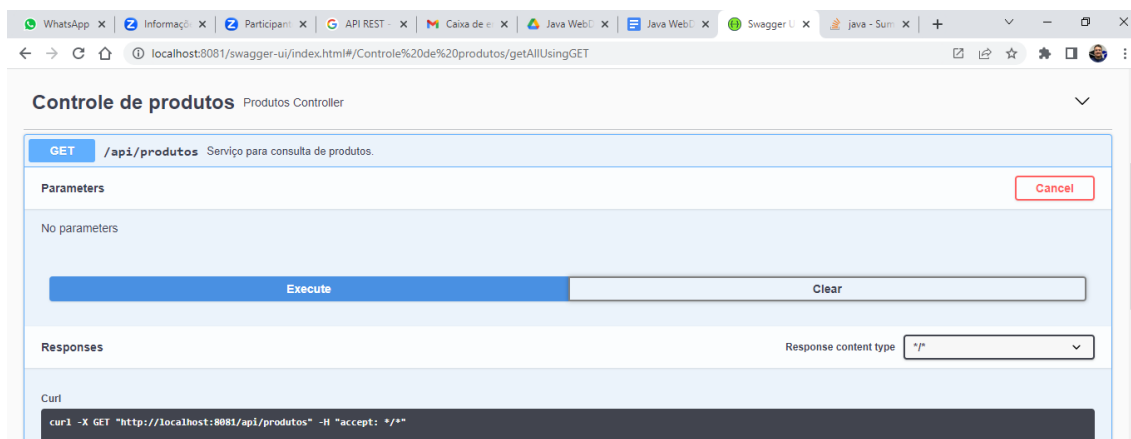


```

workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer X Javadoc Declaration Console X Servers
api_produtos [boot]
src/main/java
  br.com.cotiinformatica
    ApiProdutosApplication.java
  br.com.cotiinformatica.config
    SwaggerConfig.java
  br.com.cotiinformatica.controllers
    ProdutosController.java
  br.com.cotiinformatica.dtos
    GetProdutosDTO.java
    PostProdutosDTO.java
    PutProdutosDTO.java
  br.com.cotiinformatica.entities
    Produto.java
  br.com.cotiinformatica.factories
    ConnectionFactory.java
  br.com.cotiinformatica.repositories
    ProdutoRepository.java
src/main/resources
  application.properties
src/test/java
Maven Dependencies
JRE System Library [JavaSE-11]
src
  target
  mvnw.cmd
  pom.xml
  README.md
projetoAula01
projetoAula02
projetoAula03
projetoAula04
projetoAula05

:: Spring Boot ::
(v2.7.5)

2023-02-15 21:57:00.288 INFO 17712 --- [main] b.c.c.ApiProdutosApplication : Starting ApiPr
2023-02-15 21:57:00.294 INFO 17712 --- [main] b.c.c.ApiProdutosApplication : No active prof
2023-02-15 21:57:01.824 INFO 17712 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initial
2023-02-15 21:57:01.846 INFO 17712 --- [main] o.apache.catalina.core.StandardService : Starting servi
2023-02-15 21:57:01.846 INFO 17712 --- [main] org.apache.catalina.core.StandardEngine : Starting Servi
2023-02-15 21:57:02.073 INFO 17712 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing S
2023-02-15 21:57:02.073 INFO 17712 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplic
2023-02-15 21:57:02.832 INFO 17712 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat startec
2023-02-15 21:57:03.042 INFO 17712 --- [main] b.c.c.ApiProdutosApplication : Started ApiPr
  
```



PUT /api/produtos Serviço para atualização de produto.

Parameters

Cancel

Name	Description
dto * required	dto

object (body)

Edit Value | Model

```
{
  "idProduto": 1,
  "nome": "Caderno Escolar",
  "preco": 50,
  "quantidade": 20
}
```

Cancel

Parameter content type

application/json

You are screen sharing Stop Share

Curl

```
curl -X PUT "http://localhost:8081/api/produtos" -H "accept: */*" -H "Content-Type: application/json" -d '{"idProduto": 1, "nome": "Caderno Escolar", "preco": 50, "quantidade": 20}'
```

Request URL

http://localhost:8081/api/produtos

Server response

Code	Details
200	<p>Response body</p> <pre>{ "idProduto": 1, "nome": "Caderno Escolar", "preco": 50, "quantidade": 20 }</pre> <p>Download</p> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Thu 16 Feb 2023 01:00:06 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Responses

Code	Description
200	OK

You are screen sharing Stop Share

DELETE /api/produtos/{id} Serviço para exclusão de produto.

Parameters

Cancel

Name	Description
id * required	id

integer(\$int32) (path)

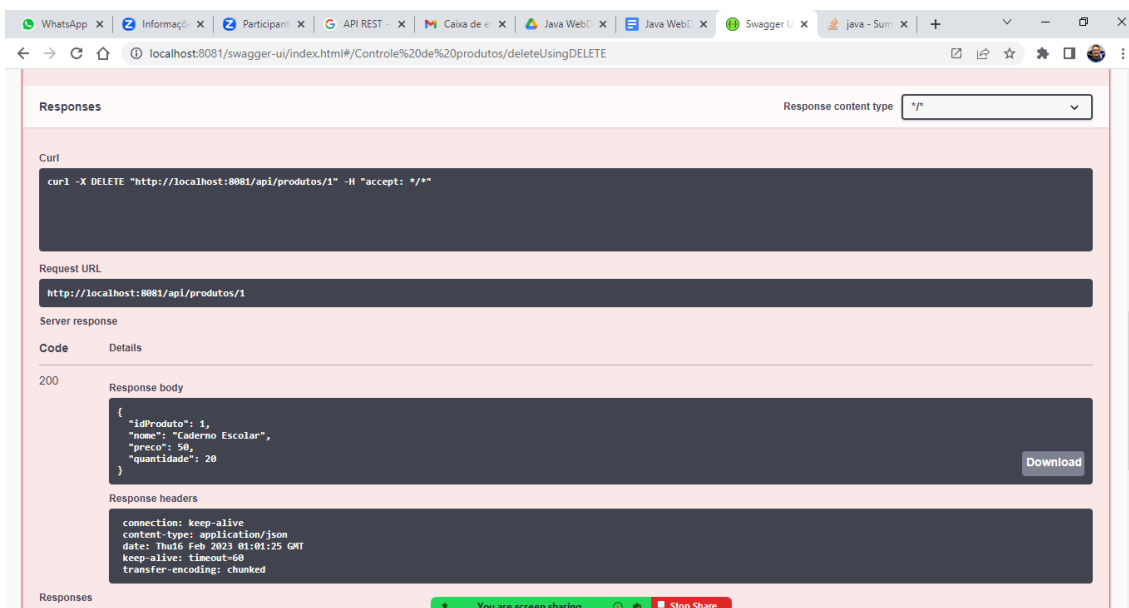
1

Execute

Responses

Response content type */*

Code	Description
200	OK



Enviando para o GITHUB:

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/api_produtos (main)
$ git add .
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/api_produtos (main)
$ git commit -m 'Desenvolvimento da API de produtos'
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/api_produtos (main)
$ git push -u origin main
```

