

BEM-VINDOS

Residência de Software 2022/1

Disciplina: Desenvolvimento de API RestFull

O que é uma API?

O que é uma API?

A sigla API deriva da expressão inglesa *Application Programming Interface* que, traduzida para o português, pode ser compreendida como uma interface de programação de aplicação

Ou seja, API é um conjunto de normas que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos.

O que é uma API?

Por meio de APIs, desenvolvedores podem criar novos softwares e aplicativos capazes de se comunicar com outras plataformas.

Por exemplo: caso um desenvolvedor queira criar um aplicativo de fotos para Android, ele poderá ter acesso à câmera do celular através da API do sistema operacional, sem ter a necessidade de criar uma nova interface de câmera do zero.

Exemplos de uso de API

No WhatsApp, por exemplo, podemos perceber a integração da lista de contatos salva no dispositivo com os contatos do aplicativo

No Facebook, temos a integração com o Instagram, que permite que fotos postadas no aplicativo também sejam postadas automaticamente no Facebook.

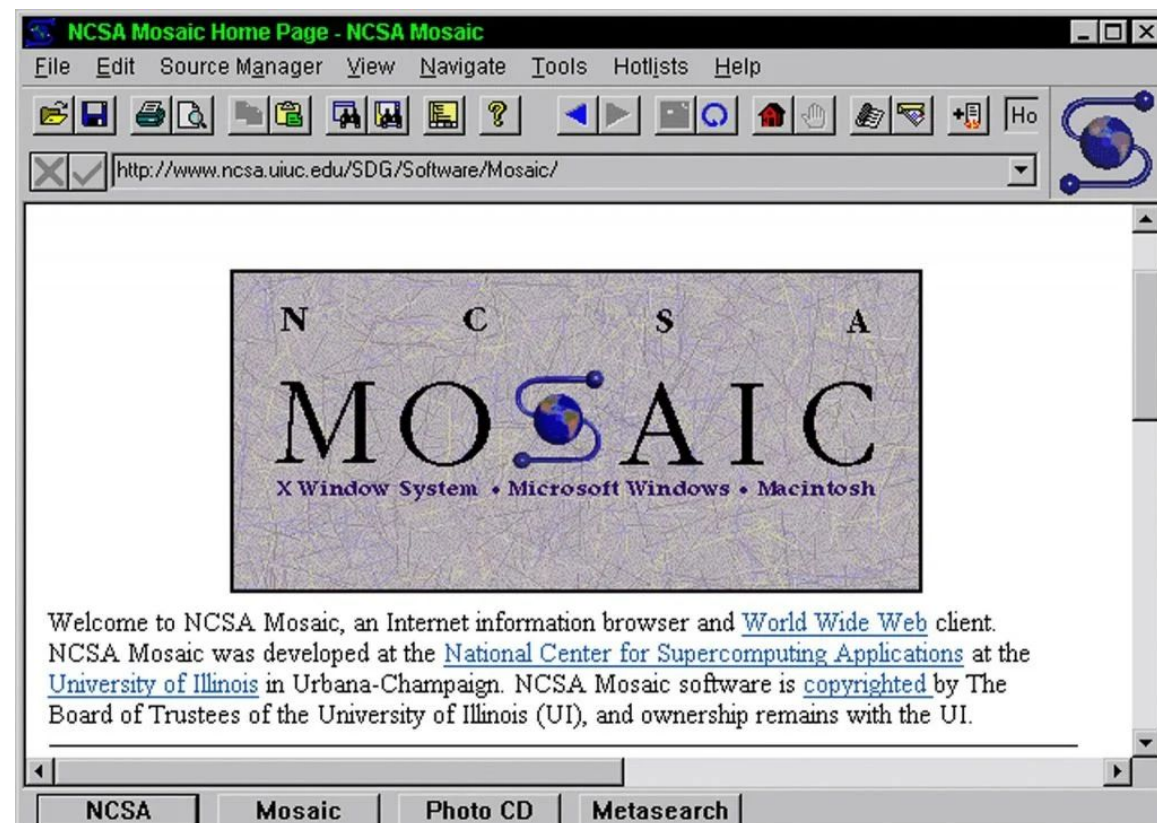
Aplicativos que utilizam os serviços de mapas por meio da API do Google Maps

Um pouco da História

1991 - Star7 (Green Project)



1993 - Navegador Mosaic

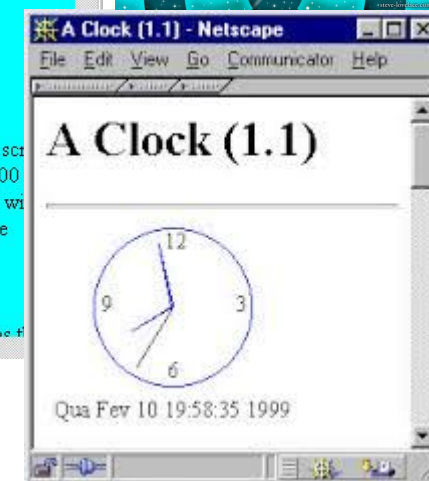
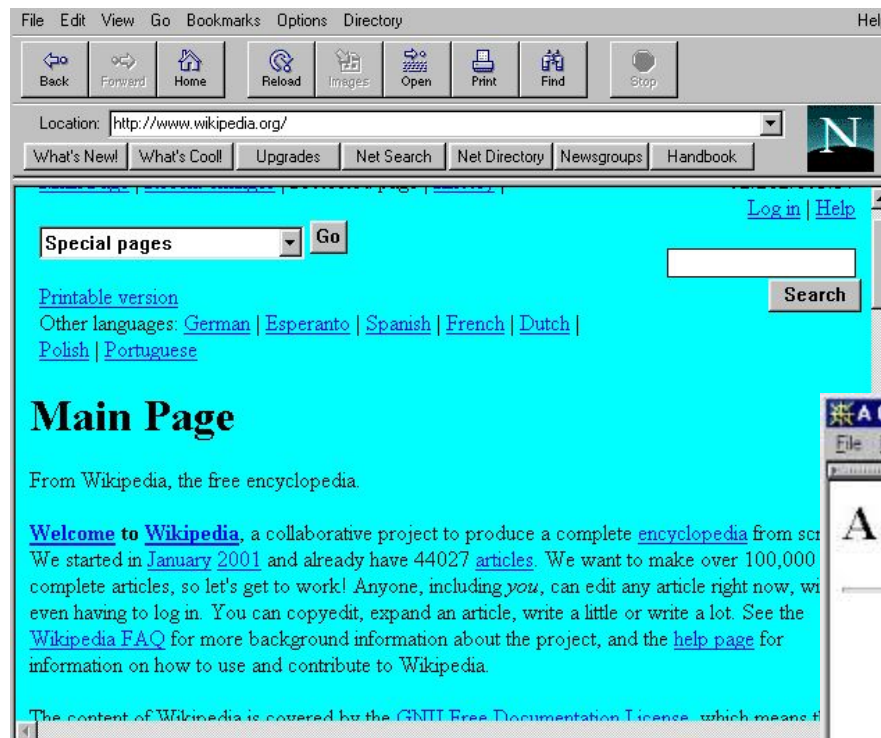


Um pouco da História

1995 - Java



1996 - Netscape 2.0



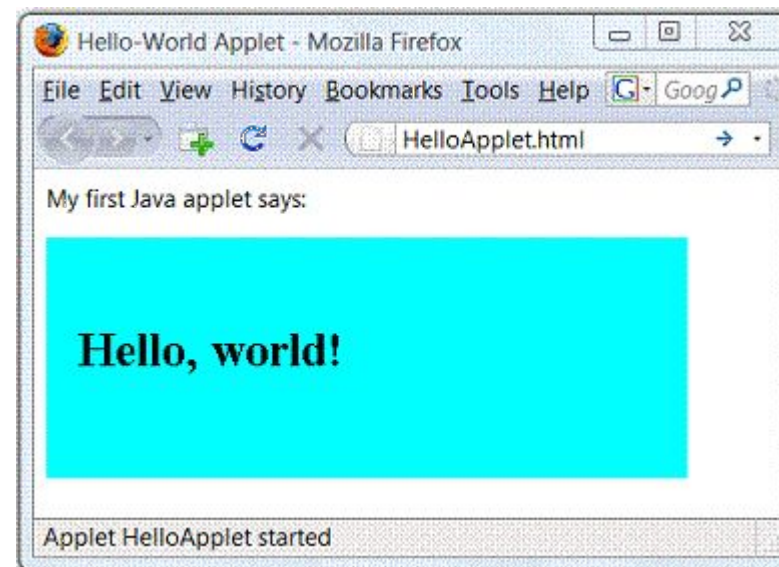
Um pouco da História

HelloApplet.java

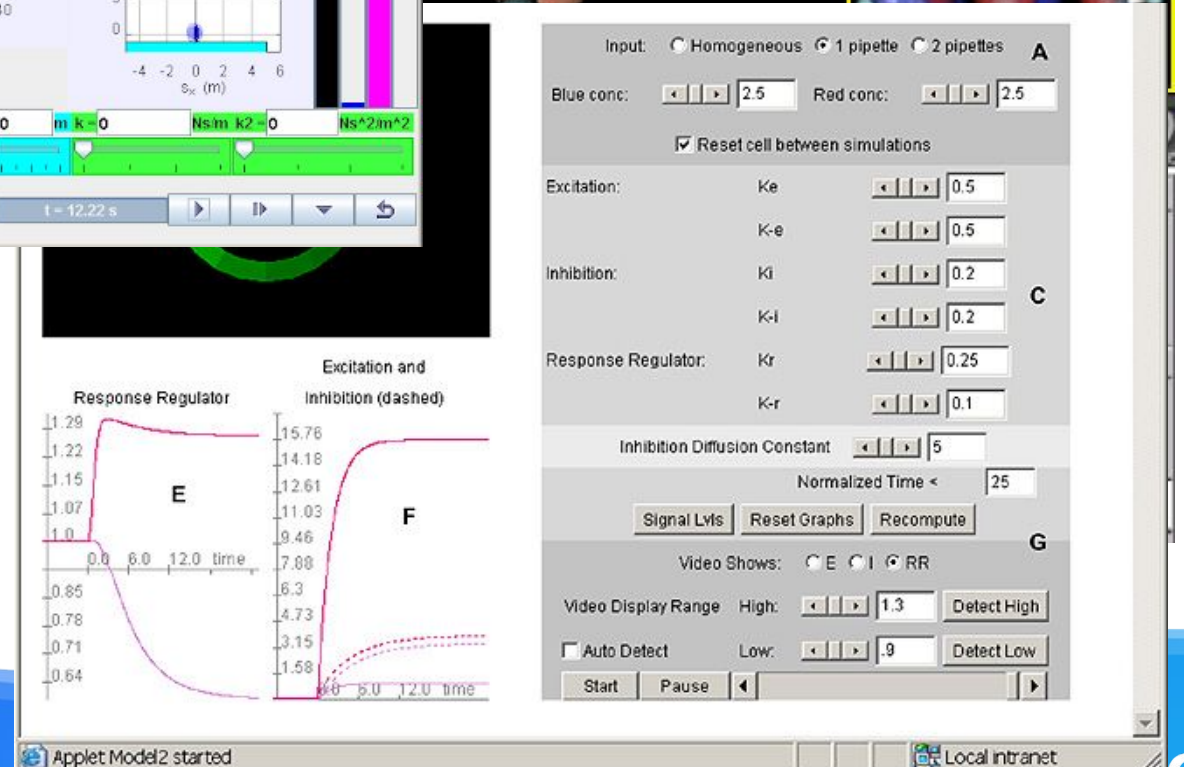
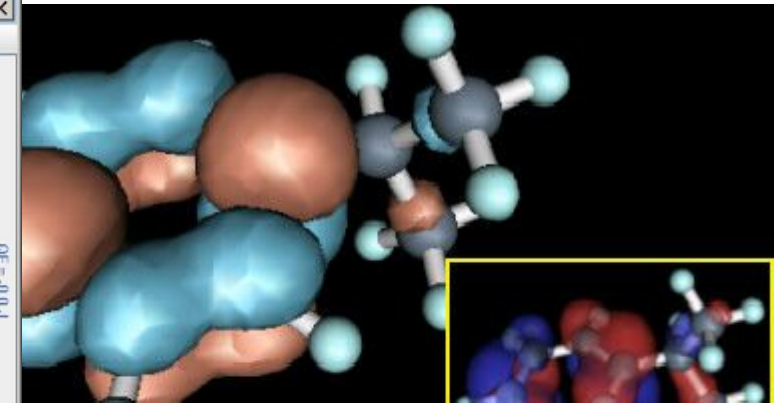
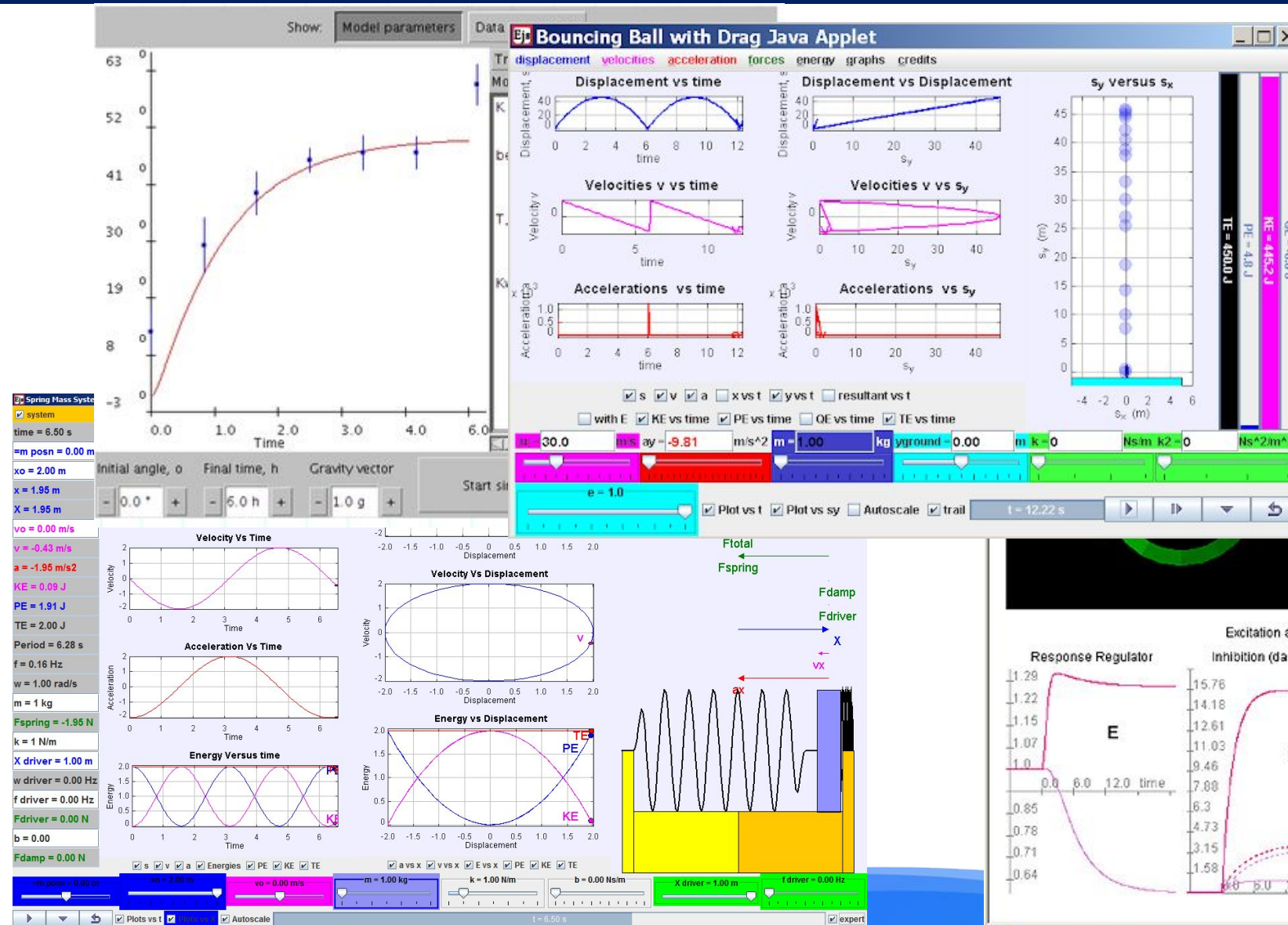
```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
/**
 * First Java Applet to say "Hello, world" on your web browser
 */
public class HelloApplet extends Applet { // save as "HelloApplet.java"
    public void paint(Graphics g) {
        setBackground(Color.CYAN); // set background color
        g.setColor(Color.BLACK); // set foreground text color
        g.setFont(new Font("Times New Roman", Font.BOLD, 30)); // set font face, bold and size
        g.drawString("Hello, world", 20, 80); // draw string with baseline at (20, 80)
    }
}
```

HelloApplet.html

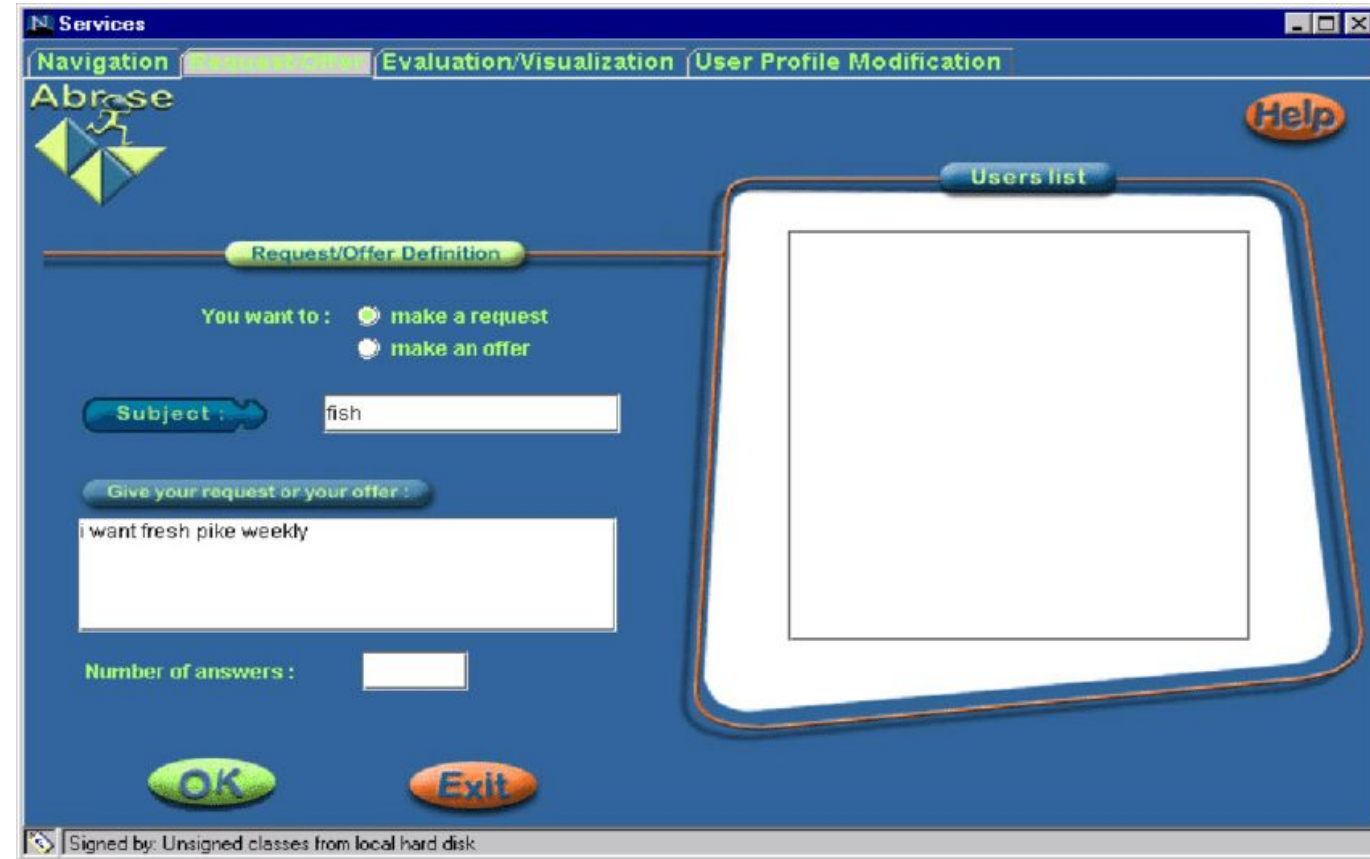
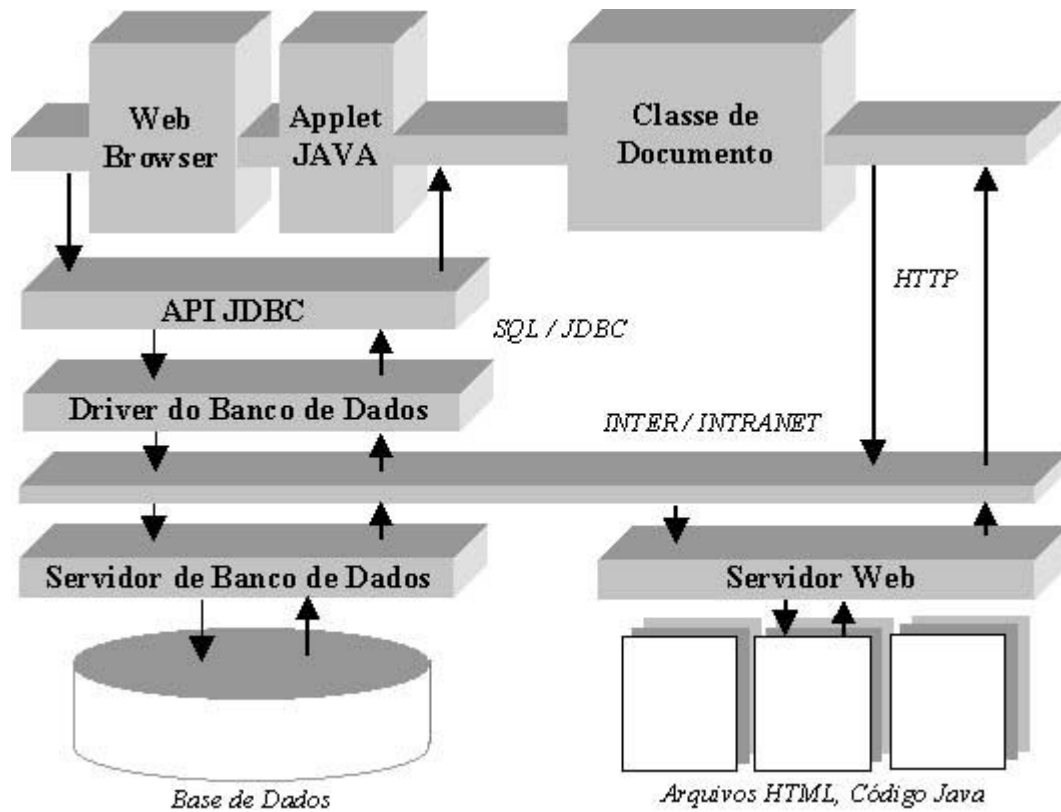
```
1 <html>
2 <head>
3   <title>Hello-World Applet</title>
4 </head>
5 <body>
6   <h3>My first Java applet says:</h3>
7   <applet code="HelloApplet.class" width="400" height="150"
8       alt="Error loading applet!">
9   </applet>
10 </body>
11 </html>
```



Um pouco da História



Um pouco da História



Java Enterprise Edition

J2EE 1.2 - 1999

- Servlets
- JSP - JavaServer Pages
- Filters
- Tag Libraries
- EJB
- JNDI
- JavaMail

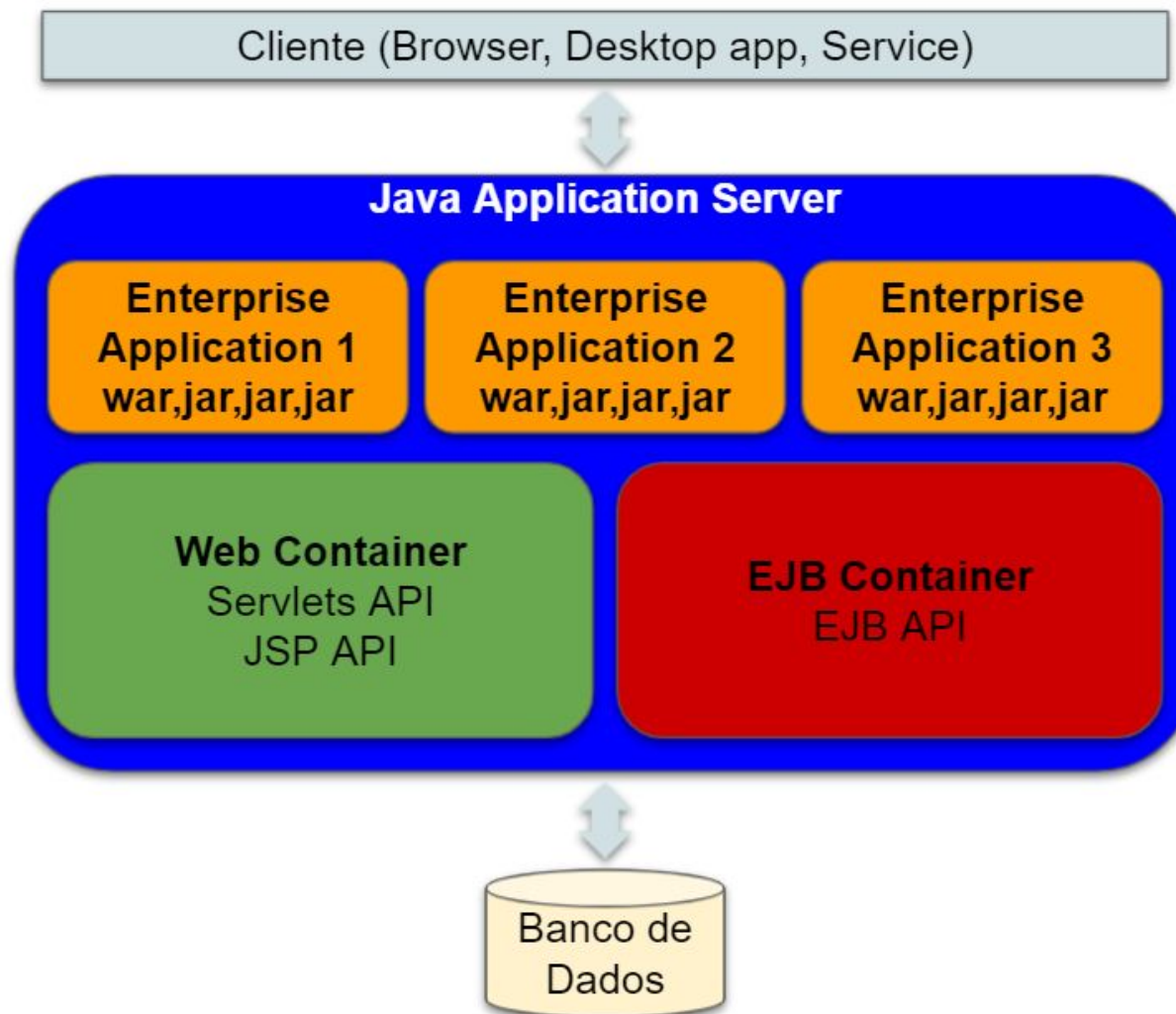
Aplicação J2EE

Servidor de Aplicação

Java Virtual Machine

Sistema Operacional

Sistema Operacional



Servlets

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println( "<!DOCTYPE html >\n" +
            "<html>\n" +
            "<head><title>Olá mundo</title></head>\n" +
            "<body>\n" +
            "<h1>Bem vindos ao curso de Java Backend</h1>\n" +
            "</body></html>"
        );
    }
}
```

Servlets

JEE 8 - 2017

Enterprise Application Technologies

- Batch Applications for the Java Platform 1.0
- Concurrency Utilities for Java EE 1.0
- Contexts and Dependency Injection for Java 2.0
- Dependency Injection for Java 1.0
- Bean Validation 2.0
- Enterprise JavaBeans 3.2
- Interceptors 1.2
- Java EE Connector Architecture 1.7
- Java Persistence 2.2
- Common Annotations for the Java Platform 1.3
- Java Message Service API 2.0
- Java Transaction API (JTA) 1.2
- JavaMail 1.6

Web Services Technologies

- Java API for RESTful Web Services (JAX-RS) 2.1
- Implementing Enterprise Web Services 1.3
- Web Services Metadata for the Java Platform 2.1
- Java API for XML-Based RPC (JAX-RPC) 1.1 (Optional)
- Java API for XML Registries (JAXR) 1.0 (Optional)
- Management and Security Technologies
- Java EE Security API 1.0
- Java Authentication Service Provider Interface for Containers 1.1
- Java Authorization Contract for Containers 1.5
- Java EE Application Deployment 1.2 (Optional)
- J2EE Management 1.1
- Debugging Support for Other Languages 1.0
- Java EE-related Specs in Java SE
- Java Management Extensions (JMX) 2.0
- SOAP with Attachments API for Java (SAAJ) Specification 1.3
- Streaming API for XML (StAX) 1.0
- Java API for XML Processing (JAXP) 1.6
- Java Database Connectivity 4.0
- Java Architecture for XML Binding (JAXB) 2.2
- Java API for XML-Based Web Services (JAX-WS) 2.2
- JavaBeans Activation Framework (JAF) 1.1

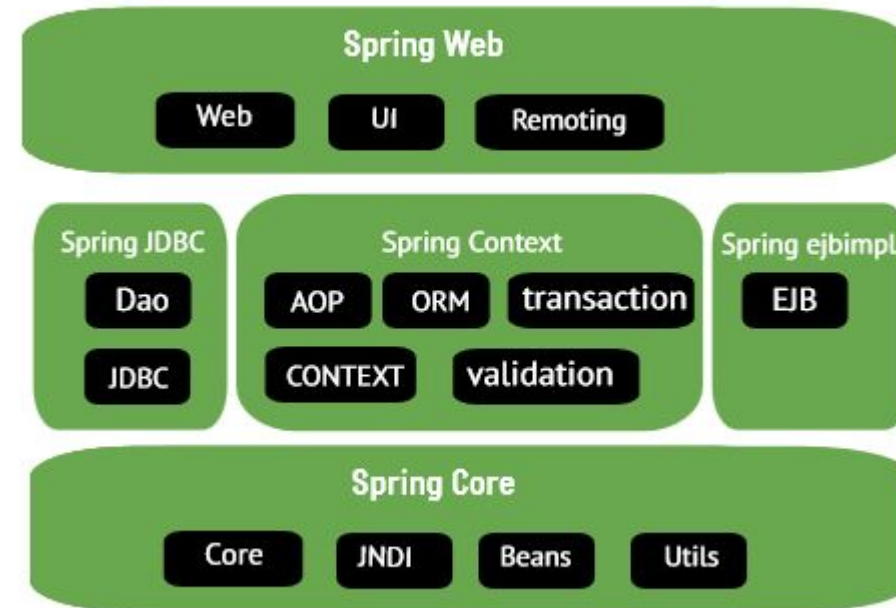
Spring Framework

Outubro de 2002



Junho de 2003

Spring Framework 0.9 Modules (June 2003)



springtutorials.com

Spring Framework

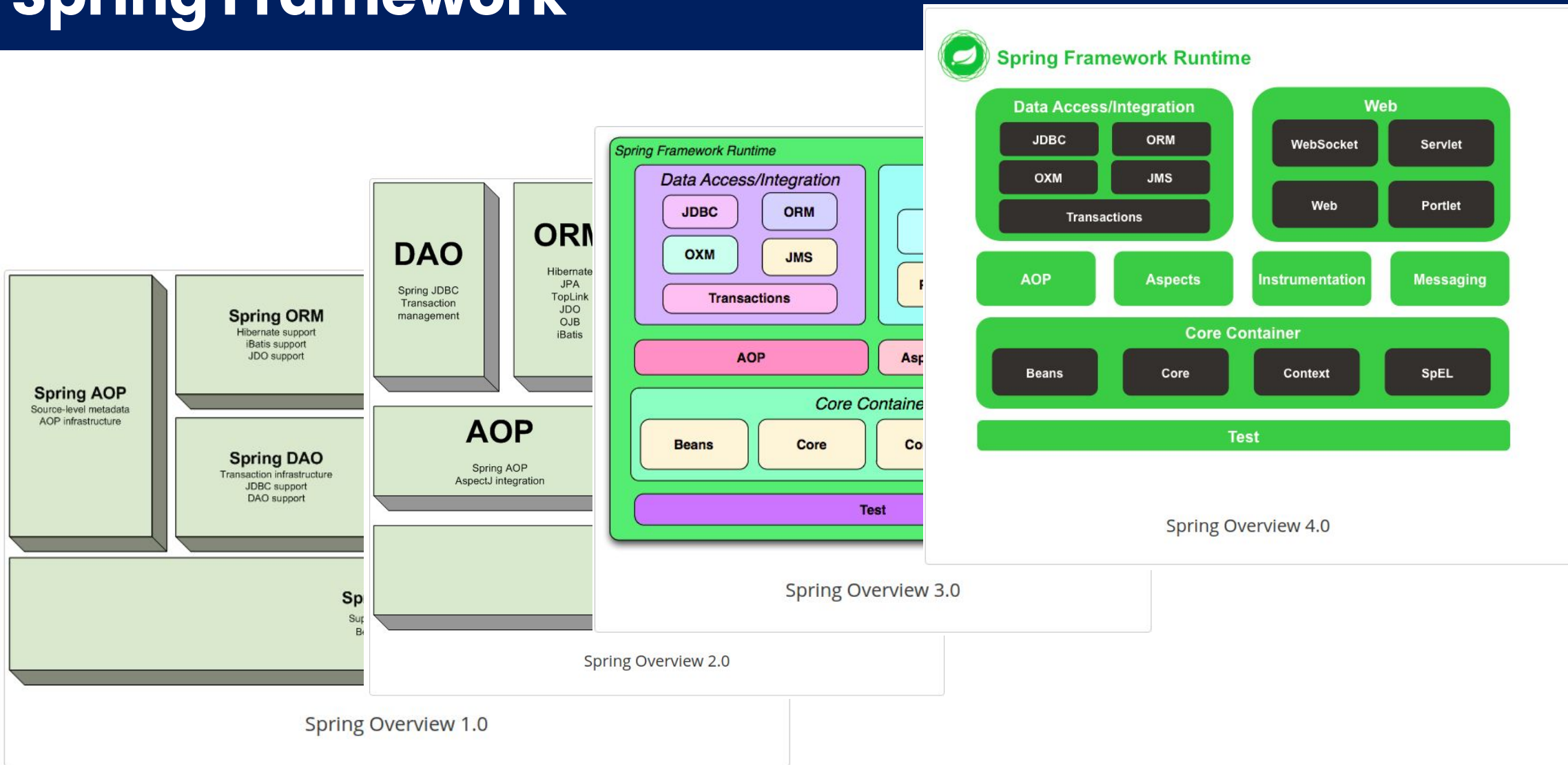
Consulta no banco com JDBC:

```
public class JdbcTest {  
  
    public static Connection getConnection() throws ClassNotFoundException, SQLException {  
        String url = "jdbc:postgresql://localhost:5432/testdb";  
        String user = "usuario";  
        String password = "senha";  
        Class.forName("org.postgresql.Driver");  
        Connection con = DriverManager.getConnection(url, user, password);  
        return con;  
    }  
  
    public static void main(String args[]) {  
        Connection con = null;  
        try {  
            con = getConnection();  
            Statement st = con.createStatement();  
            ResultSet rs = st.executeQuery("SELECT COUNT(*) FROM NOTAS");  
            int totalNotas = rs.getInt(1);  
            System.out.println("Total de notas: " + totalNotas);  
        } catch (SQLException ex) {  
            ex.printStackTrace();  
        } catch (ClassNotFoundException e) {  
            e.printStackTrace();  
        } finally {  
            if (con != null) {  
                try {  
                    con.close();  
                } catch (SQLException e) {  
                    e.printStackTrace();  
                }  
            }  
        }  
    }  
}
```

Consulta no banco com Spring Framework usando JdbcTemplate:

```
public class TesteJdbcTemplate {  
  
    public static SimpleDriverDataSource getDataSource() {  
        SimpleDriverDataSource ds = new SimpleDriverDataSource();  
        ds.setDriver(new org.postgresql.Driver());  
        ds.setUrl("jdbc:postgresql://localhost:5432/testdb");  
        ds.setUsername("usuario");  
        ds.setPassword("senha");  
        return ds;  
    }  
  
    public static void main(String args[]) {  
        SimpleDriverDataSource ds = getDataSource();  
        int totalNotas = new JdbcTemplate(ds)  
            .queryForObject("SELECT COUNT(*) FROM NOTAS", Integer.class);  
        System.out.println("Total de notas: " + totalNotas);  
    }  
}
```

Spring Framework



Spring Boot

É uma boa plataforma para desenvolvedores Java desenvolverem uma aplicação Spring independente e pronta para produção que você pode simplesmente "roda".

- Microframework
- Utiliza o “eco-sistema” do Spring Framework
- *Convenção sobre Configuração*
- Baseado em POJOs - Plain Old Java Object

Vantagens

- Começar com configurações mínimas
- Fácil de entender e desenvolver
- Aumento de produtividade
- Reduzir o tempo de desenvolvimento



Spring Boot

Programador com foco em negócio ao invés da tecnologia.

Controller

```
@RestController("/todo")
public class TodoController {
    @Autowired
    TodoService todoService;

    @GetMapping("/count")
    public Integer getCount() {
        return todoService.getCount();
    }
}
```

Service

```
@Service
public class TodoService {

    @Autowired
    TodoRepository todoRepository;

    @Transactional
    public Integer getCount() {
        return todoRepository.contar();
    }
}
```

Repository

```
public interface TodoRepository
    extends JpaRepository<Todo, Integer>{

    @Query("Select count(*) from Todo")
    public Integer contar();
}
```

Entity

```
@Entity()
@Table(name="todo")
public class Todo {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;
    private String titulo;
    private String descricao;
    private Boolean completada;
}
```

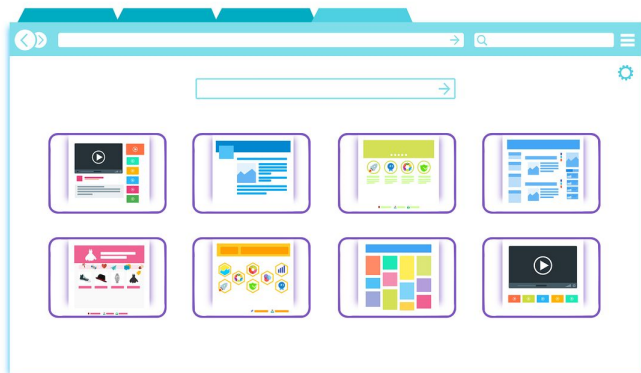
Arquivo de Configuração

```
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
spring.datasource.url=jdbc:postgresql://localhost:5432/testdb
spring.datasource.username=usuario
spring.datasource.password=senha
```

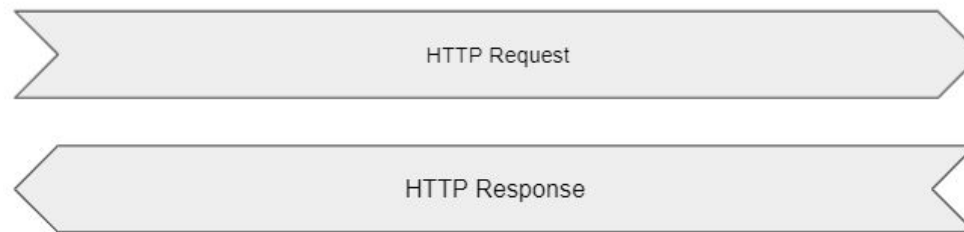

O que é HTTP?

O Hypertext Transfer Protocol, sigla HTTP (em português Protocolo de Transferência de Hipertexto) é um protocolo de comunicação... utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos. Ele é a base para a comunicação de dados da World Wide Web.

O que é HTTP?



```
GET /index.html HTTP/1.1  
Host: www.exemplo.com
```



```
HTTP/1.1 200 OK  
Date: Mon, 23 May 2020 22:38:34 GMT  
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)  
Last-Modified: Wed, 08 Jan 2020 23:11:55 GMT  
Etag: "3f80f-1b6-3e1cb03b"  
Accept-Ranges: bytes  
Content-Length: 438  
Connection: close  
Content-Type: text/html; charset=UTF-8  
<!DOCTYPE html >  
<html>  
<head>  
  <title>Titulo da página</title>  
  <meta charset="utf-8"/>  
</head>  
<body>  
  <h1>Curso de java backend</h1>  
  ...
```


Representational State Transfer (REST), em português Transferência Representacional de Estado, é um estilo de arquitetura de software que define um conjunto de restrições a serem usadas para a criação de web services (serviços Web). Os Web services que estão em conformidade com o estilo arquitetural REST, denominados Web services RESTful, fornecem interoperabilidade entre sistemas de computadores na Internet.

- Maneira simples de realizar interações entre sistemas
- “Resource-based”: baseado no conceito de recursos (coisas e não ações, como produto, pessoa, email, pedido)
- Utiliza o protocolo HTTP e seus *verbos* para as operações
- Recursos são identificados por URIs (Uniform Resource Identifier)
- Podem ser representados em *JSON* ou *XML*
- Podem ser armazenados no cache

Rest: exemplo simples



HTTP Request

```
GET /index.html HTTP/1.1  
Host: www.exemplo.com
```



Rest: exemplo simples



HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2020 22:38:34 GMT
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2020 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
<!DOCTYPE html>
<html>
<head> <title>Rest exemplo</title> </head>
<body>
  <h1>Lista de e-mail</h1>
  <table border=1>
    <tr><td>Nome</td><td>E-mail</td></tr>
    <tr><td></td><td></td></tr>
  </table>
</body>
<script>
  function carregaDados() {
    .....
  }
</script>
</html>
```



Rest: exemplo simples



HTTP Request

```
GET /emails HTTP/1.1  
Host: www.exemplo.com
```



Rest: exemplo simples



HTTP Response

HTTP/1.1 200 OK
Date: Mon, 23 May 2020 22:38:34 GMT
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2020 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/json; charset=UTF-8

```
[  
  {  
    "nome": "antonino",  
    "email": "antonio@mail.com"  
  },  
  {  
    "nome": "joaquim",  
    "email": "joaquim@mail.com"  
  },  
  {  
    "nome": "maria",  
    "email": "maria@mail.com"  
  }  
]
```



Rest – JSON / XML?

JSON: JavaScript Object Notation

```
[
  {
    "nome": "Edson Sales Arantes",
    "notas": [ "8", "9", "5" ]
  },
  {
    "nome": "Luiz Livelli ",
    "notas": [ "8", "10", "7" ]
  },
  {
    "nome": "Caique Caicedo De Plata",
    "notas": [ "10", "10", "9" ]
  }
]
```

XML (Extensible Markup Language)

```
<?xml version="1.0" encoding="UTF-8"?>
<lista-alunos>
  <aluno>
    <nome>Edson Sales Arantes</nome>
    <notas>
      <element>8</element><element>9</element><element>5</element>
    </notas>
  </aluno>
  <aluno>
    <nome>Luiz Livelli </nome>
    <notas>
      <element>8</element><element>10</element><element>7</element>
    </notas>
  </aluno>
  <aluno>
    <nome>Caique Caicedo De Plata</nome>
    <notas>
      <element>10</element><element>10</element><element>9</element>
    </notas>
  </aluno>
</lista-alunos>
```

Rest – JSON / XML?

JSON

- Fácil e simples de usar.
- Consome menos memória, menor banda para transferência de dados.
- Não é necessário criar “Mapeamento”, Jackson resolve tudo ;-)
- No browser/javascript, não há a necessidade de biblioteca extra para converter os dados
 - `string = JSON.stringify(objeto)`
 - `Objeto = JSON.parse(string)`
- Muito comum em serviços web, aplicativos e sites de dados

XML

- Tags não são pré-definidas (como em html), é possível criar tags customizadas.
- Criado principalmente para transportar dados
- Código de marcação é fácil de ser entendido e legível por seres humanos.
- Formato bem estruturado, fácil de por seres humanos.
- Possibilidade de se ter validadores de formatos
- Muito comum em aplicações/ferramentas corporativas.
- Utilizado em arquivos de configuração e documentos (Office).

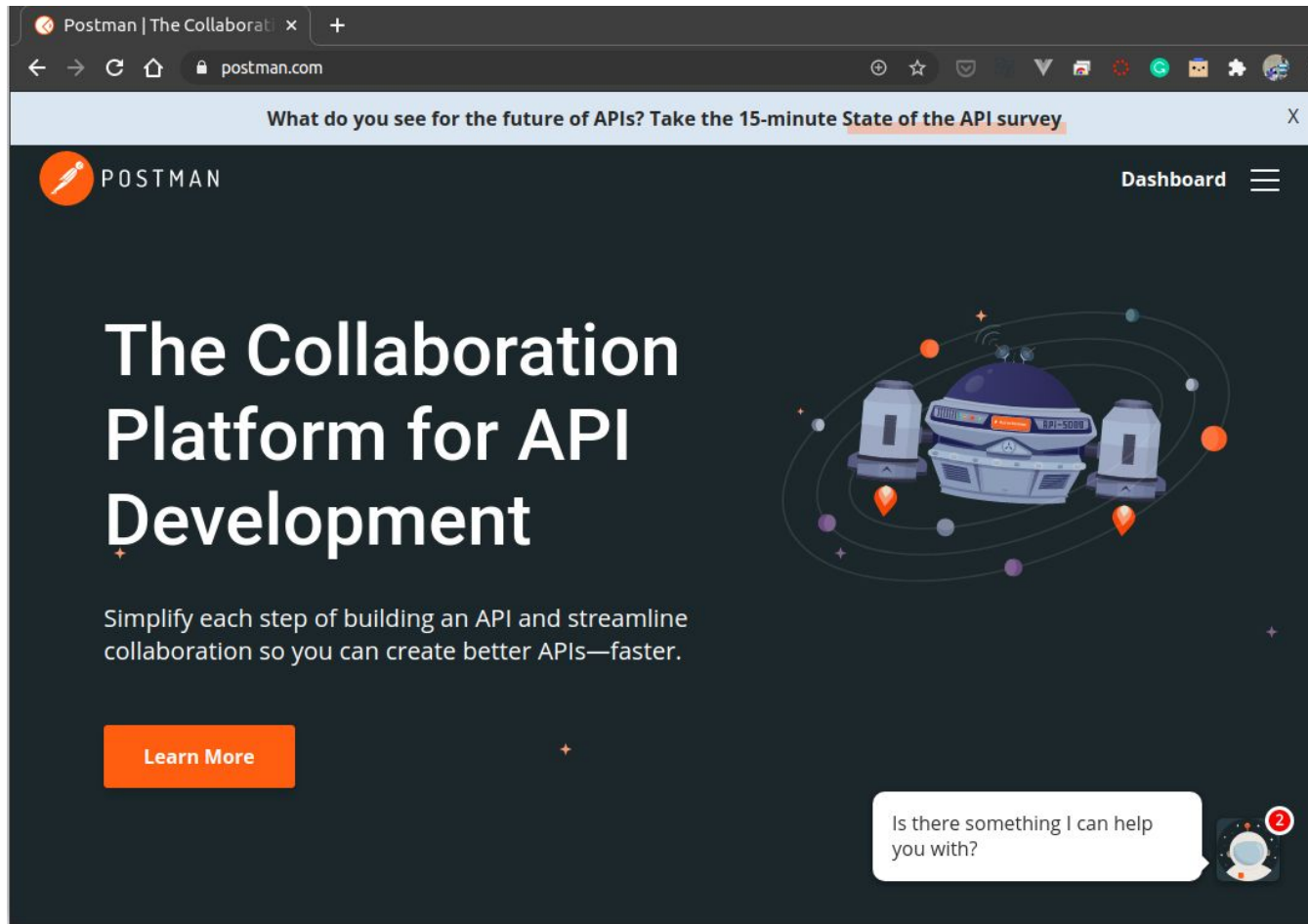
Rest – JSON / XML?

Verbo HTTP	Operação CRUD	Comando SQL
GET	R - read (ler)	SELECT
POST	C - create (criar)	INSERT
PUT ou PATCH	U - update (atualizar)	UPDATE
DELETE	D - delete (apagar)	DELETE

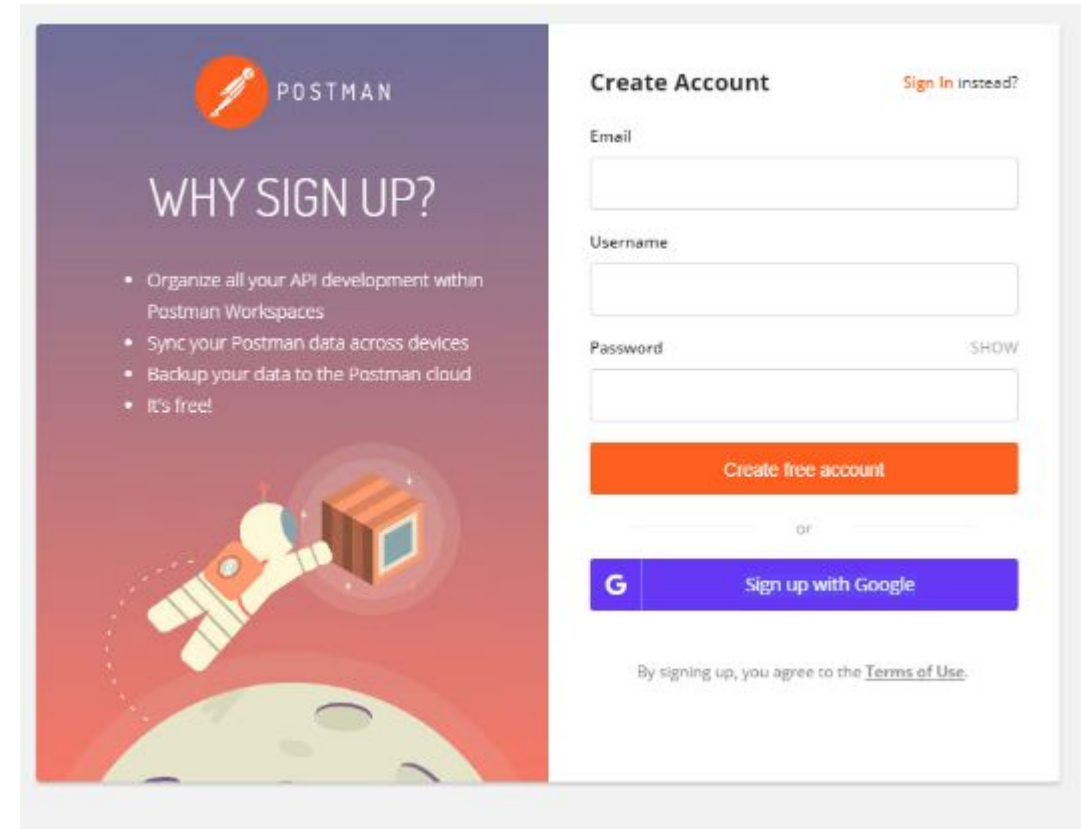
Postman

Ferramenta de apoio ao desenvolvedor Rest

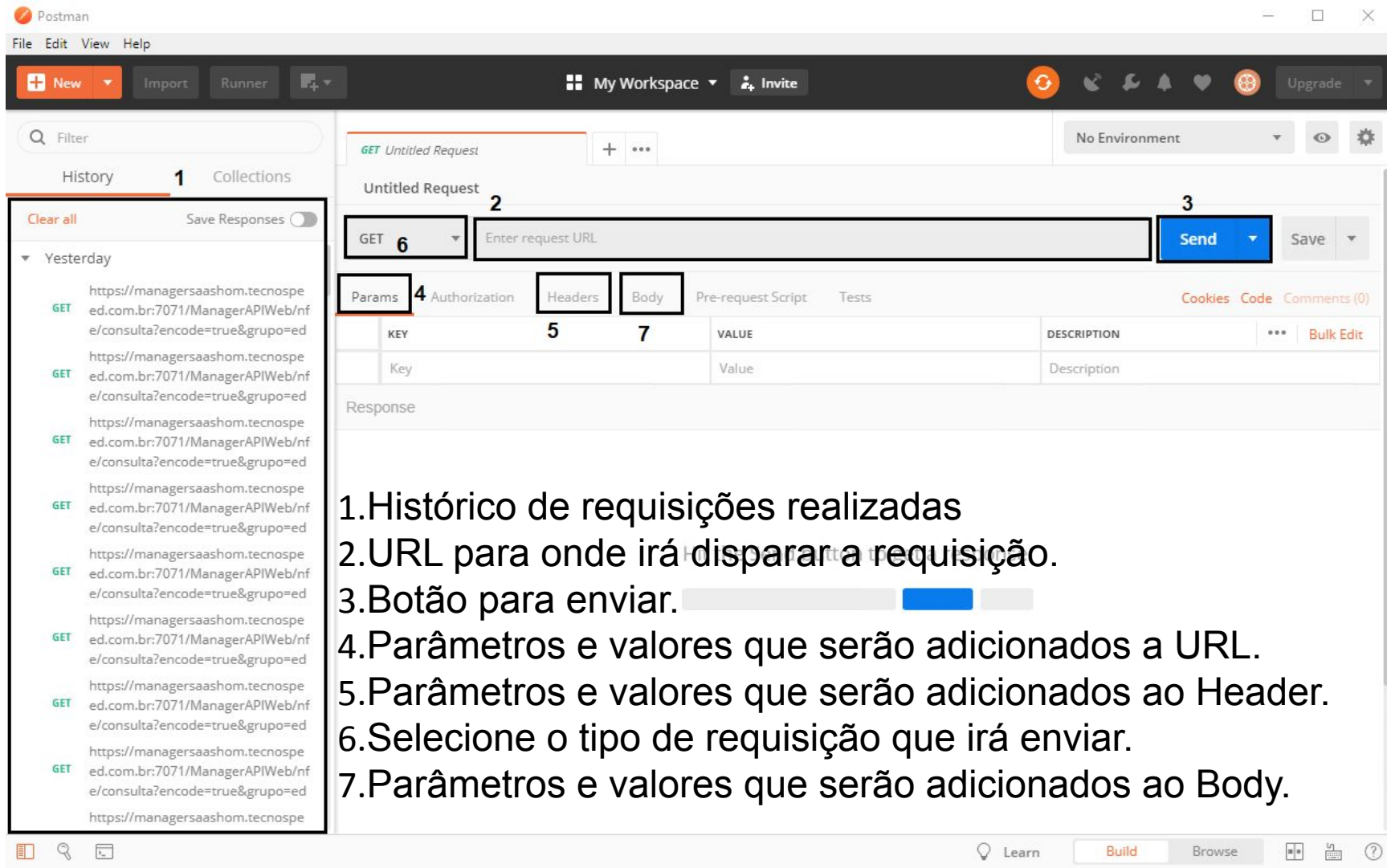
<https://www.postman.com/>



- Fazer o Download
- Instalar
- Criar usuário
- Autenticar



Postman



Rest: playground

Servidor com dados Fake para testes

<https://github.com/bulinha/residencia-serratec/blob/master/crud.jar?raw=true>

Executar:

```
java -jar crud.jar
```

```
(base) bulinha@bula-idk-notebook:~/residencia/exemplos/respositorio$ java -jar crud.jar
```

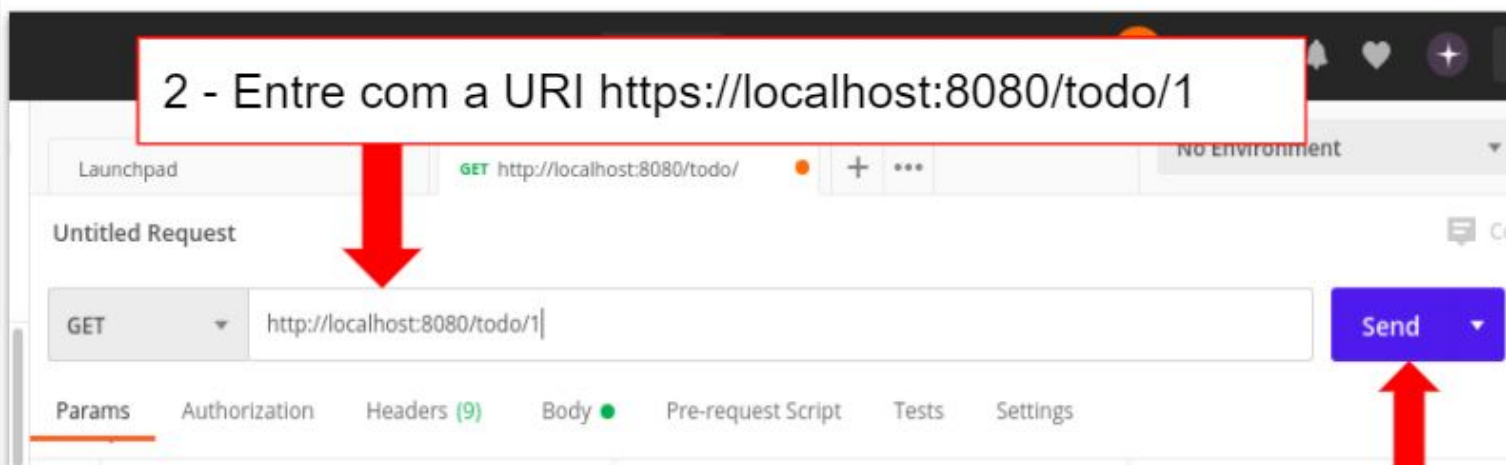
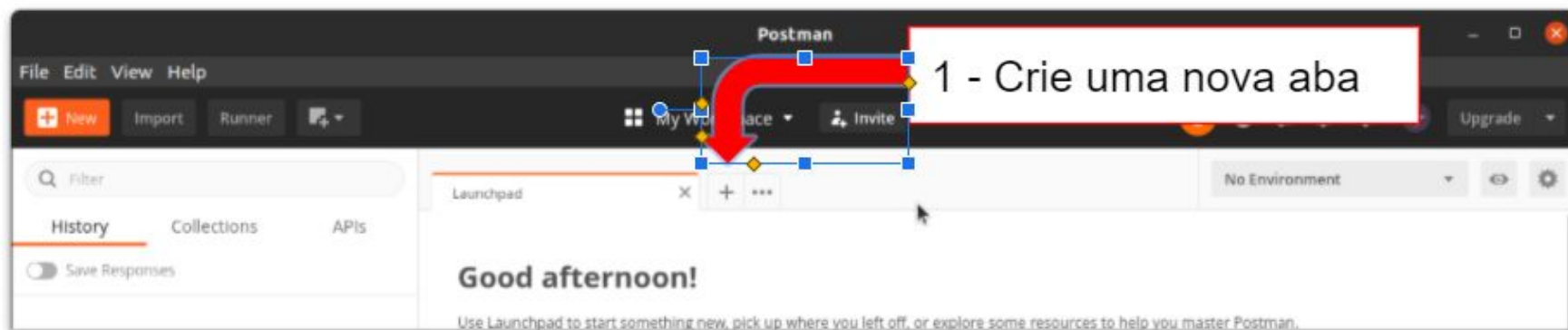
[illegible]

```

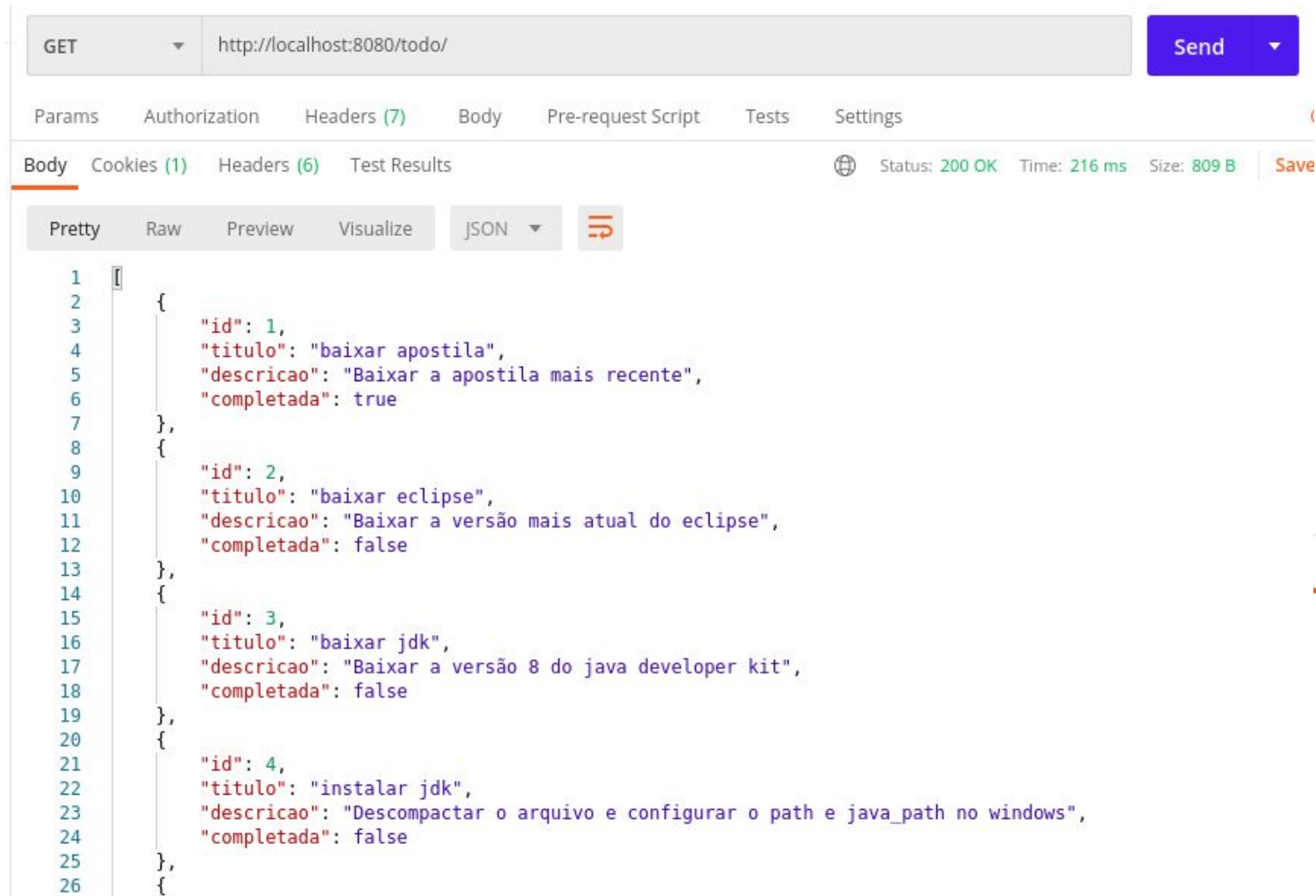
2020-08-02 17:00:23.668 INFO 813524 --- [           main] o.s.cursojava2.crud.CrudApplication : Starting CrudApplication v0.0.1-SNAPSHOT on bula-idk-notebook with
PID 813524 (/home/bulinha/residencia/exemplos/respositorio/crud.jar started by bulinha in /home/bulinha/residencia/exemplos/respositorio)
2020-08-02 17:00:23.671 INFO 813524 --- [           main] o.s.cursojava2.crud.CrudApplication : No active profile set, falling back to default profiles: default
2020-08-02 17:00:24.757 INFO 813524 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-08-02 17:00:24.771 INFO 813524 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-08-02 17:00:24.771 INFO 813524 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.37]
2020-08-02 17:00:24.854 INFO 813524 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-08-02 17:00:24.854 INFO 813524 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1131 ms
2020-08-02 17:00:25.096 INFO 813524 --- [           main] o.s.c.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-08-02 17:00:25.341 INFO 813524 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-08-02 17:00:25.356 INFO 813524 --- [           main] o.s.cursojava2.crud.CrudApplication : Started CrudApplication in 2.08 seconds (JVM running for 2.5)

```


Postman



Rest: GET



GET http://localhost:8080/todo/ Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (6) Test Results Status: 200 OK Time: 216 ms Size: 809 B Save

Pretty Raw Preview Visualize JSON ≡

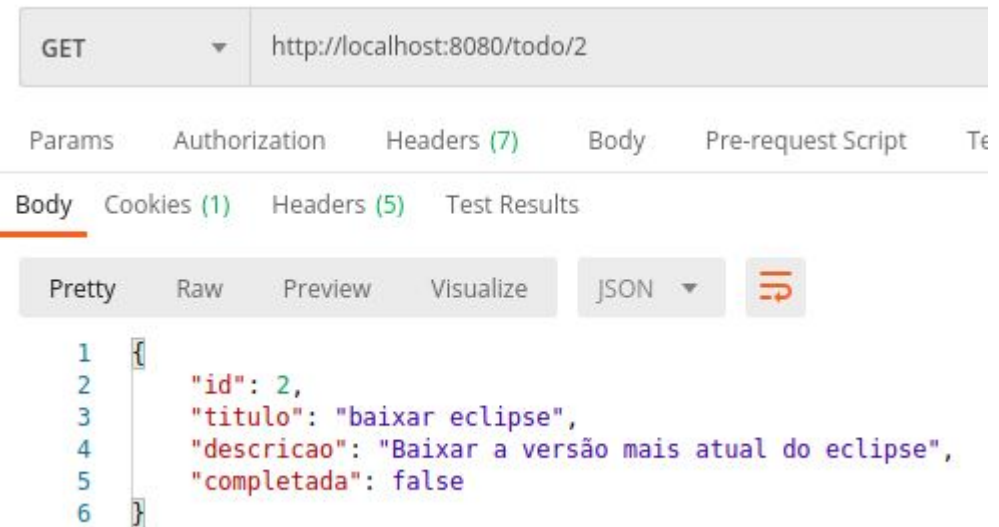
```
1 [
2   {
3     "id": 1,
4     "titulo": "baixar apostila",
5     "descricao": "Baixar a apostila mais recente",
6     "completada": true
7   },
8   {
9     "id": 2,
10    "titulo": "baixar eclipse",
11    "descricao": "Baixar a versão mais atual do eclipse",
12    "completada": false
13  },
14  {
15    "id": 3,
16    "titulo": "baixar jdk",
17    "descricao": "Baixar a versão 8 do java developer kit",
18    "completada": false
19  },
20  {
21    "id": 4,
22    "titulo": "instalar jdk",
23    "descricao": "Descompactar o arquivo e configurar o path e java_path no windows",
24    "completada": false
25  },
26  ]
```

Recuperar todos os “todos”

GET /todo

Recuperar um único “todo”

GET /todo/2



GET http://localhost:8080/todo/2

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize JSON ≡

```
1 {
2   "id": 2,
3   "titulo": "baixar eclipse",
4   "descricao": "Baixar a versão mais atual do eclipse",
5   "completada": false
6 }
```

Rest: GET

Recuperar todos com o atributo “completed” = true

GET /todo/?completada=true

Como retorno, além da lista de “todos”, o serviço retorna o Status 200 OK

GET ▼ http://localhost:8080/todo?completada=false

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCR
<input checked="" type="checkbox"/>	completada	false	
	Key	Value	Descr

Body Cookies (1) Headers (5) Test Results 🌐 Status: 200 OK

Pretty Raw Preview Visualize JSON ↺

```
1 [
2   {
3     "id": 2,
4     "titulo": "baixar eclipse",
5     "descricao": "Baixar a versão mais atual do eclipse",
6     "completada": false
7   },
8   {
9     "id": 3,
10    "titulo": "baixar jdk",
11    "descricao": "Baixar a versão 8 do java developer kit",
12    "completada": false
13  },
14 ]
```

Rest: Status

Categoria	Descrição
1xx: Informacional	Comunica informações no nível do protocolo de transferência. <ul style="list-style-type: none">•101 - indica mudança de protocolo (ex: de http para websocket)
2xx: Sucesso	Indica que a requisição foi completada: <ul style="list-style-type: none">•200 - ok•201 - criado
3xx: Redirecionar	Indica que o cliente deve realizar alguma operação adicional. <ul style="list-style-type: none">•301 - indica que o recurso mudou de localização (indicando a nova no cabeçalho)
4xx: Erro do Cliente	Alguma informação fornecida pelo cliente é inválida <ul style="list-style-type: none">•400 - Bad Request (dados inválidos ou incompletos)•401 - não autorizado•403 - proibido•404 - não encontrado•405 - método não permitido (POST e PUT proibidos)•
5xx: Erro no Servidor	Erro de responsabilidade do servidor <ul style="list-style-type: none">•500 - Erro interno (ex.: exception não tratada no java)

Rest: Get

Requisitar um recurso não existente

GET /todo/999

Retorna o erro 404

GET

▼

http://localhost:8080/todo/999

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE	DESCR
	Key	Value	Descr

Body

Cookies (1)

Headers (4)

Test Results

🌐

Status: 404 Not Found

T

Rest: Get

Pesquisar um recurso por parâmetros com valores não existentes

GET /todo/completed=999

Retorna o 200 OK e uma lista vazia

GET <http://localhost:8080/todo/?titulo=comprar sorvete>

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIP
<input checked="" type="checkbox"/>	titulo	comprar sorvete	
	Key	Value	Descri

Body Cookies (1) Headers (5) Test Results 🌐 Status: 200 OK

Pretty Raw Preview Visualize JSON ↺

1 `[]`

Rest: Post

Incluir um novo “todo”:

1. Selecionar Body
2. Selecionar raw
3. Selecionar Json
4. Preencher o JSON do objeto
5. Enviar (Send)

Retorna o objeto criado (com o novo id)

```
{
  "id": 6,
  "titulo": "fazer exercicio",
  "descricao": "pesquisar em varias apis rest",
  "completada": false
}
```

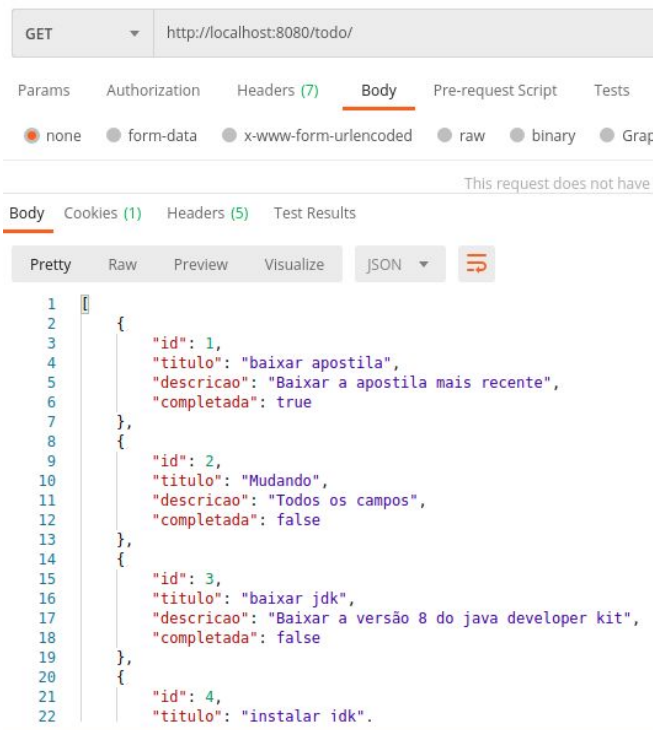
Status 201 (created)

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/todo/`. The 'Body' tab is selected, showing a JSON object: `{ "titulo": "fazer exercicio", "descricao": "pesquisar em varias apis rest", "completada": false }`. The 'raw' radio button is selected. Below the request, the 'Body' tab of the response is shown, displaying a JSON object: `{ "id": 6, "titulo": "fazer exercicio", "descricao": "pesquisar em varias apis rest", "completada": false }`. The status bar indicates a `201 Created` response.

Rest: Delete

Excluir um recurso existente

Retorna apenas o código de Status Ok 200 (se for possível remover o recurso)



GET http://localhost:8080/todo/

Params Authorization Headers (7) **Body** Pre-request Script Tests

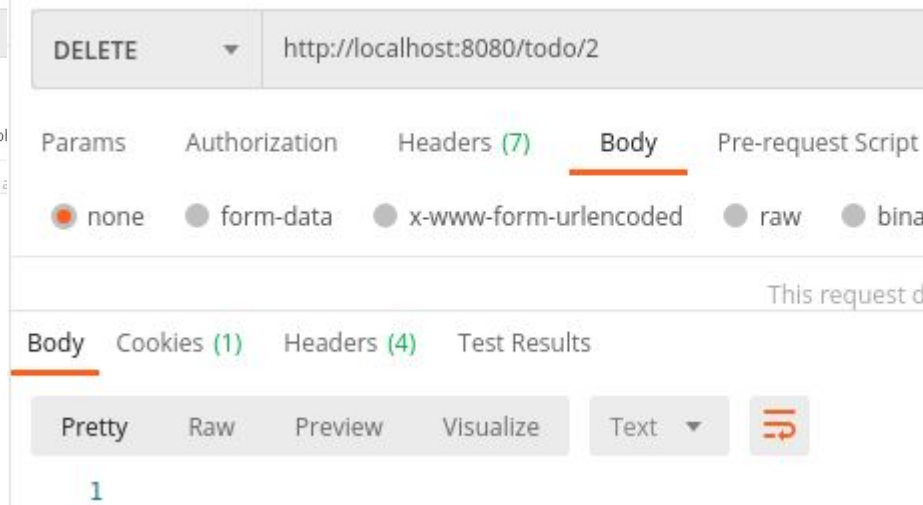
none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 [{
2   {
3     "id": 1,
4     "titulo": "baixar apostila",
5     "descricao": "Baixar a apostila mais recente",
6     "completada": true
7   },
8   {
9     "id": 2,
10    "titulo": "Mudando",
11    "descricao": "Todos os campos",
12    "completada": false
13  },
14  {
15    "id": 3,
16    "titulo": "baixar jdk",
17    "descricao": "Baixar a versão 8 do java developer kit",
18    "completada": false
19  },
20  {
21    "id": 4,
22    "titulo": "instalar idk".
23  }
24 }]
```



DELETE http://localhost:8080/todo/2

Params Authorization Headers (7) **Body** Pre-request Script Tests

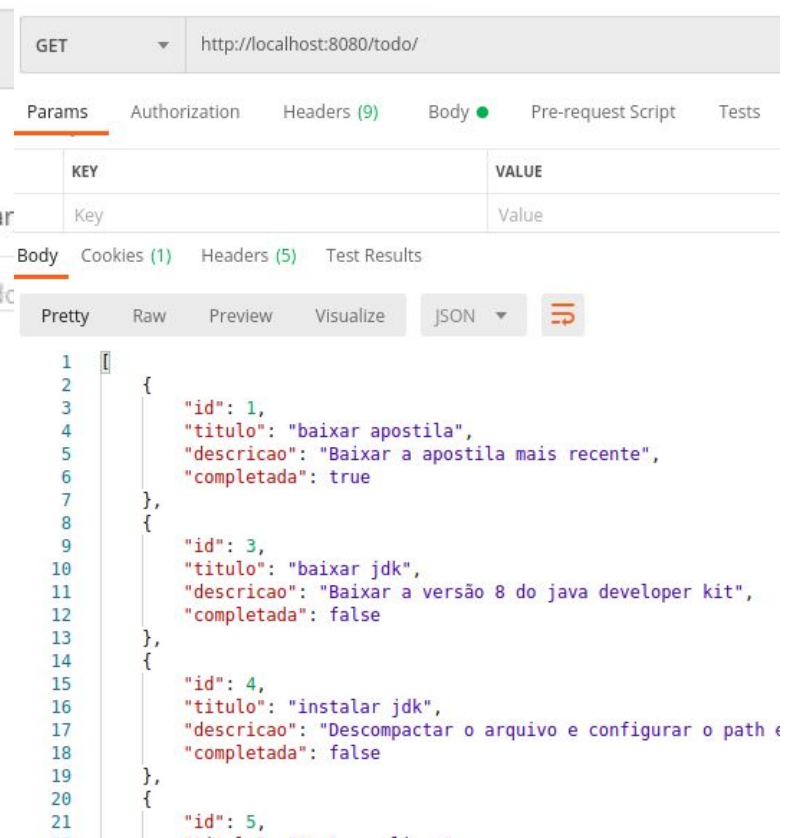
none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies (1) Headers (4) Test Results

Pretty Raw Preview Visualize Text

```
1 1
```



GET http://localhost:8080/todo/

Params Authorization Headers (9) **Body** Pre-request Script Tests

KEY	VALUE
Key	Value

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 [{
2   {
3     "id": 1,
4     "titulo": "baixar apostila",
5     "descricao": "Baixar a apostila mais recente",
6     "completada": true
7   },
8   {
9     "id": 3,
10    "titulo": "baixar jdk",
11    "descricao": "Baixar a versão 8 do java developer kit",
12    "completada": false
13  },
14  {
15    "id": 4,
16    "titulo": "instalar jdk",
17    "descricao": "Descompactar o arquivo e configurar o path e
18    "completada": false
19  },
20  {
21    "id": 5,
22    "titulo": "instalar idk".
23  }
24 }]
```

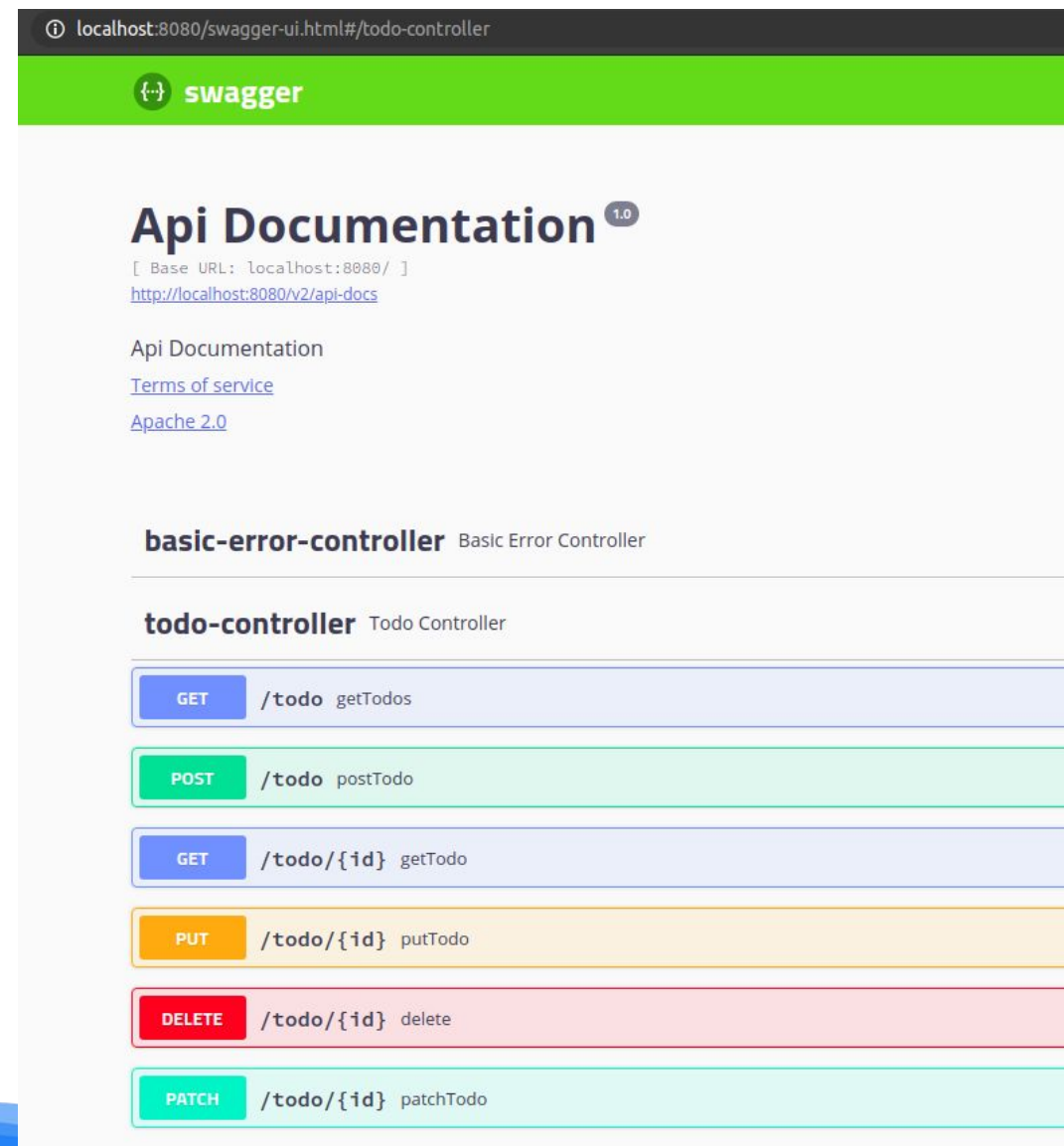
Rest: Swagger

O Swagger é um conjunto de ferramentas de software de código aberto para projetar, criar, documentar e usar serviços da Web RESTful, desenvolvidos pela SmartBear Software. Inclui documentação automatizada, geração de código e geração de casos de teste.

Wikipedia (inglês)

Para acessar o Swagger do servidor Fake de testes:
<http://localhost:8080/swagger-ui.html>

Nem todos os serviços disponíveis na Web implementam o Swagger.



Rest: Exercícios

Acessar diversas APIs Rests abertas:

<https://dadosabertos.camara.leg.br/swagger/api.html>

<https://covid19-brazil-api-docs.now.sh/>

<https://api.le-systeme-solaire.net/swagger>