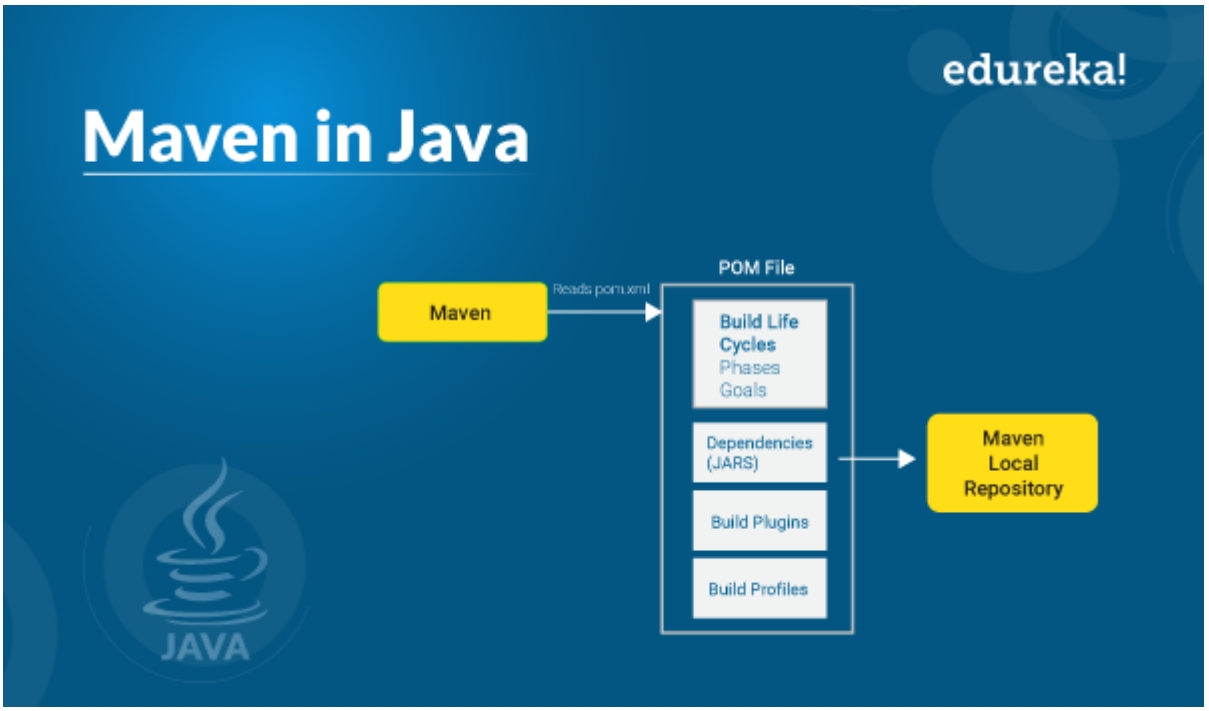
Aula 10 - API

1. Maven

Para trabalharmos com API em Java, precisamos garantir que o Maven está instalado.

O Apache Maven é uma excelente ferramenta de apoio a qualquer equipe que trabalhe com projetos Java (outras tecnologias também são suportadas), fornecendo aos desenvolvedores uma forma de automatizar e padronizar a construção e publicação de suas aplicações.



Para verificar se o Maven está instalado, vamos no prompt de comando e executamos esse comando:

mvn -version

Se estiver instalado, teremos o seguinte resultado:

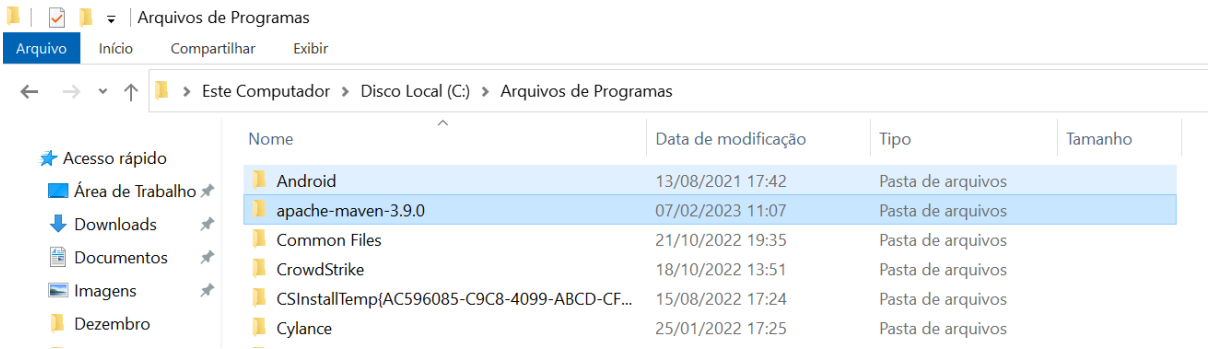
Se não estiver instalado, não teremos o resultado acima e teremos que fazer o seguinte procedimento:

Acessar: https://maven.apache.org/download.cgi

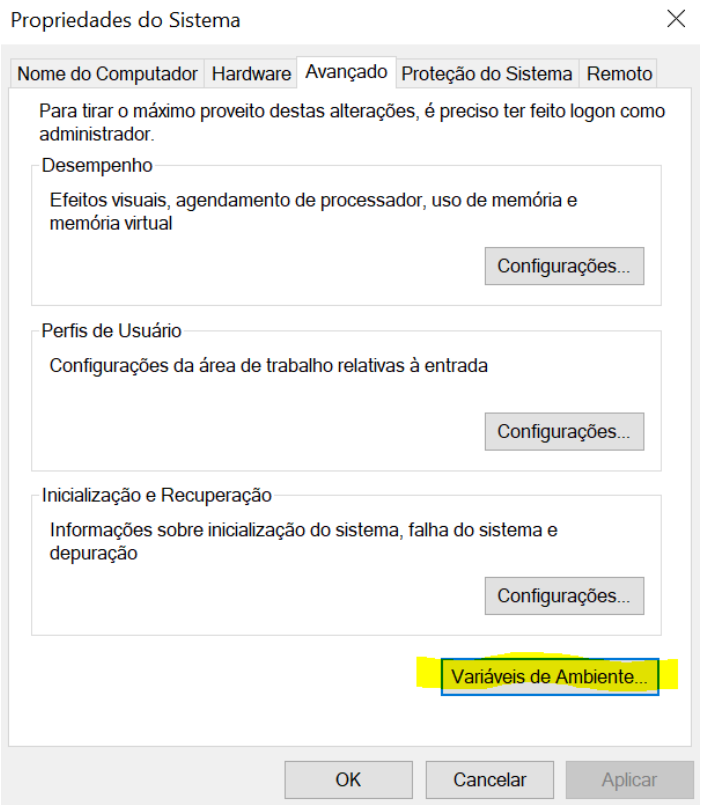
Baixar o seguinte zip:

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.9.0-bin.tar.gz	apache-maven-3.9.0-bin.tar.gz.sha512	apache-maven-3.9.0-bin.tar.gz.asc
Binary zip archive	apache-maven-3.9.0-bin.zip	apache-maven-3.9.0-bin.zip.sha512	apache-maven-3.9.0-bin.zip.asc
Source tar.gz archive	apache-maven-3.9.0-src.tar.gz	apache-maven-3.9.0-src.tar.gz.sha512	apache-maven-3.9.0-src.tar.gz.asc
Source zip archive	apache-maven-3.9.0-src.zip	apache-maven-3.9.0-src.zip.sha512	apache-maven-3.9.0-src.zip.asc

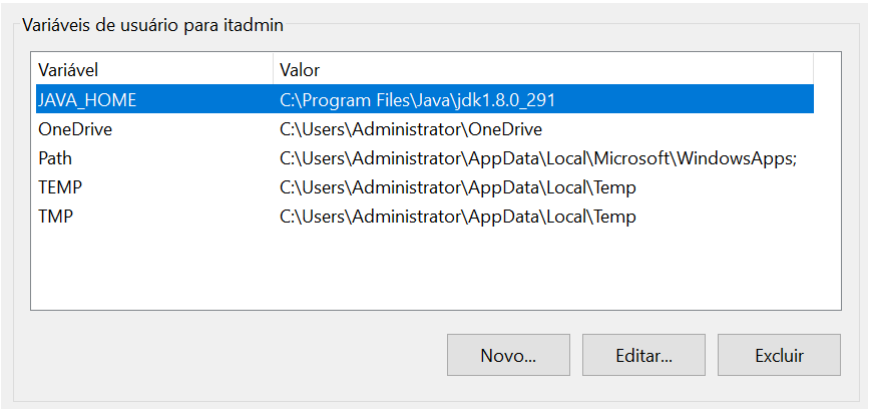
Descompactar o zip e copiar a pasta “apache-maven-3.9.0” para a pasta **Arquivos de Programas** no Meu Computador



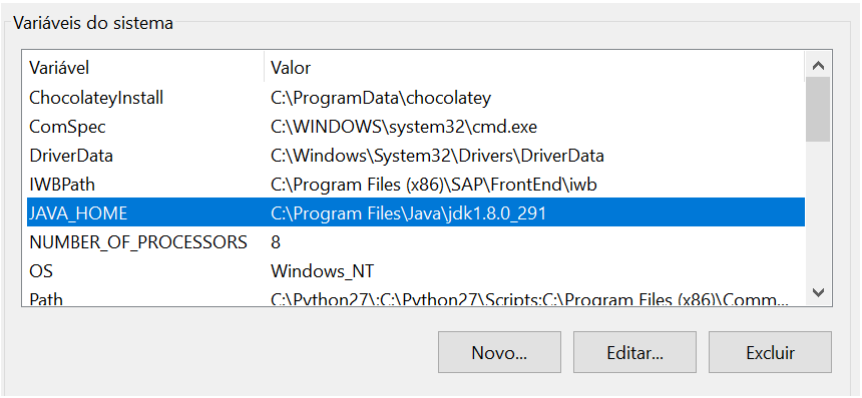
Após isso, vamos alterar as variáveis de ambiente. Para isso acesse o Meu Computador, botão direito Propriedades, Variáveis de ambiente…



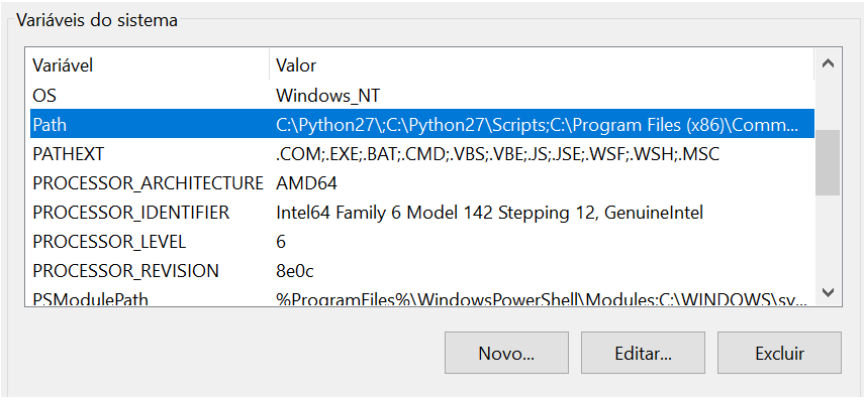
Em variáveis de usuário, clique em Novo e adicione a variável **JAVA_HOME** com o valor C:\Program Files\Java\jdk1.8.0_291 **(Não copie esse caminho aqui e sim, copie, o da sua máquina)**

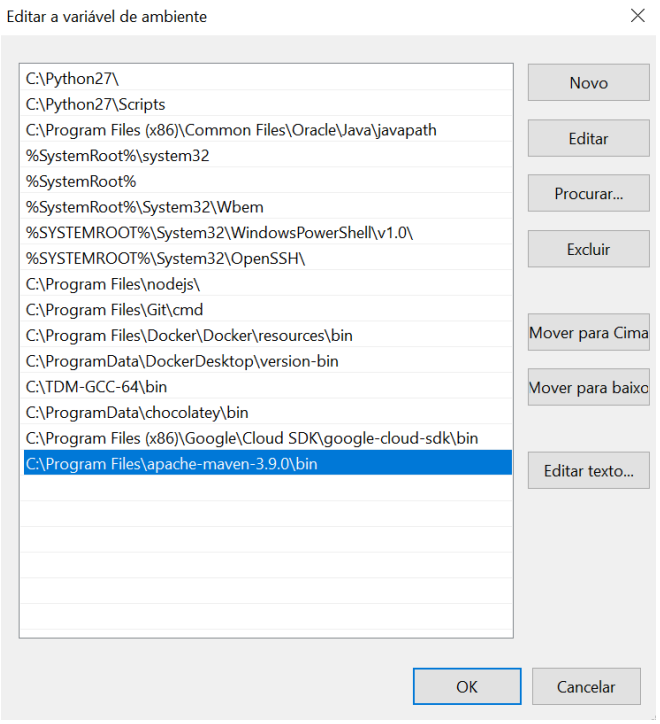


Em variáveis do sistema, clique em Novo e adicione também a variável **JAVA_HOME** com o valor C:\Program Files\Java\jdk1.8.0_291 **(Não copie esse caminho aqui e sim, copie, o da sua máquina)**



E clique na variável **PATH**, Editar e adicione o caminho da pasta do Maven C:\Program Files\apache-maven-3.9.0\bin **(Não copie esse caminho aqui e sim, copie, o da sua máquina)**





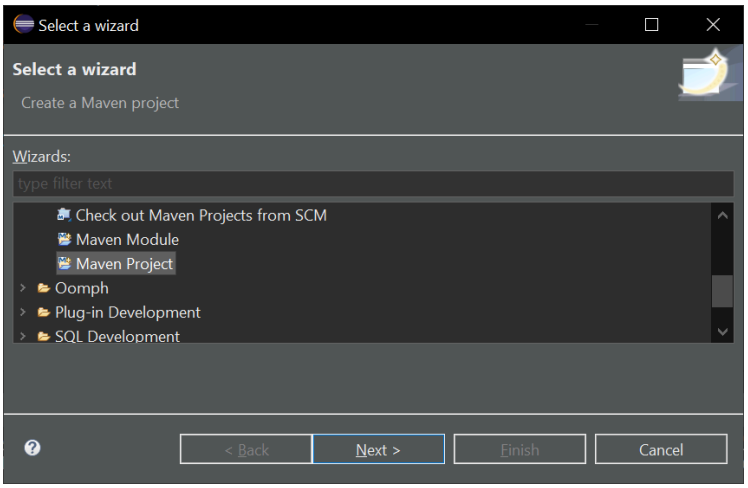
Pronto, após isso, verifique via cmd se agora o Maven está instalado, com o seguinte comando:

mvn -version

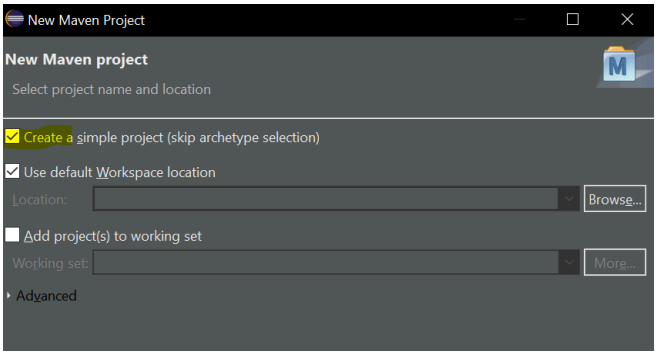
Se estiver instalado, teremos o seguinte resultado:

2. Desenvolvimento JAVA

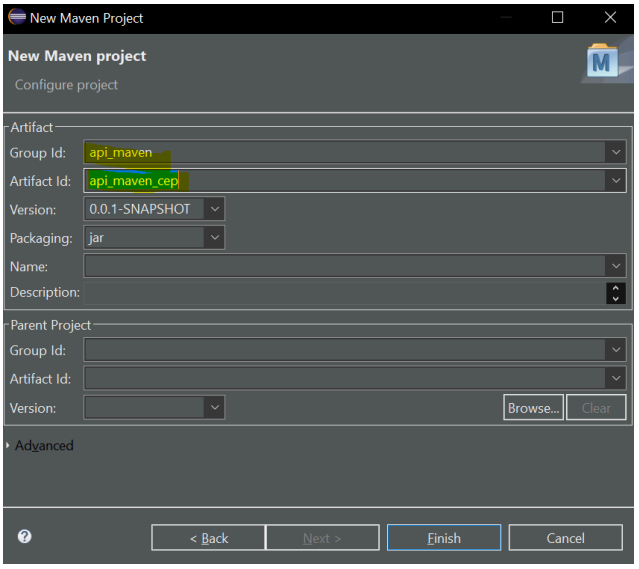
No Eclipse, crie um novo projeto Maven, clicando em New > Project > Other > Maven Project



Marque a opção Create a simple project



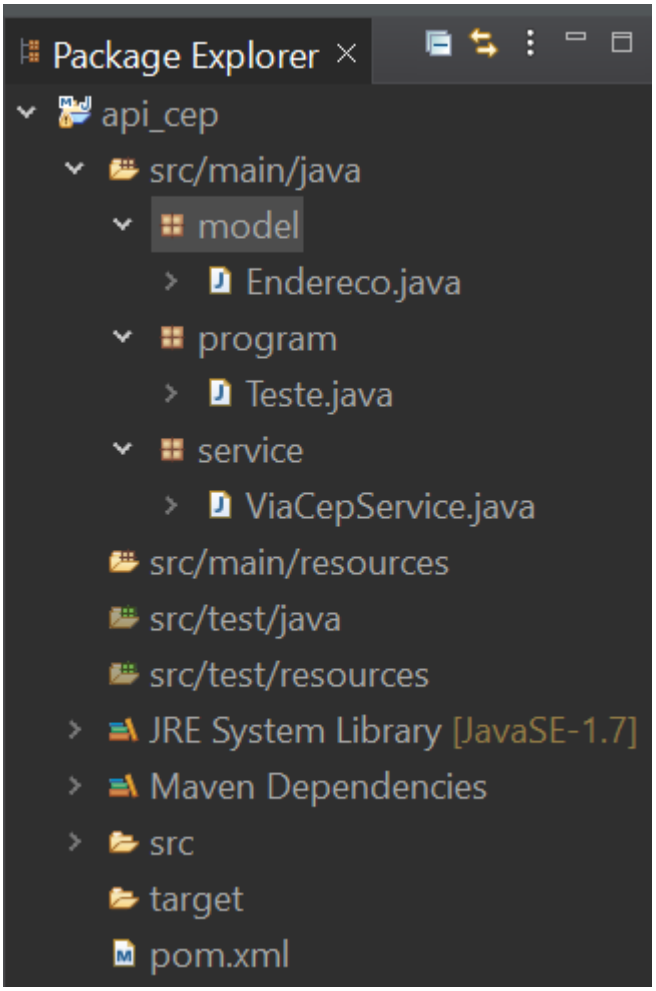
Defina um Group Id e Artifact Id, conforme exemplo:



Vamos criar a seguinte arquitetura de desenvolvimento:

Criar o package model > Endereco.java
Criar o package service > ViaCepService.java
Criar o package program > Teste.java

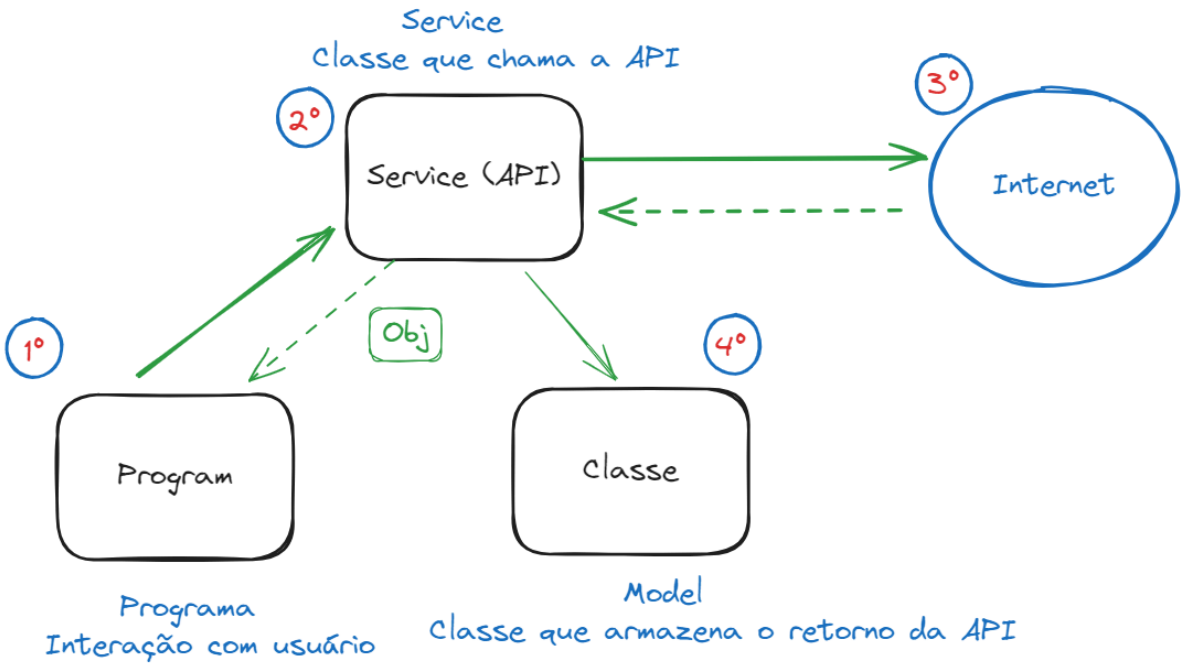
De modo que fique assim:



Vamos incluir no arquivo **pom.xml** logo abaixo da tag <version> o seguinte conteúdo:

```
<dependencies>
  <dependency>
      <groupId>org.apache.httpcomponents</groupId>
      <artifactId>httpclient</artifactId>
      <version>4.5.14</version>
  </dependency>
  <dependency>
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      <version>2.10.1</version>
  </dependency>
</dependencies>
```

O objetivo desse desenvolvimento é termos a seguinte dinâmica:



Agora vamos desenvolver a classe **Endereco.java** da seguinte forma:

```
package model;

public class Endereco {

    private String cep;
    private String logradouro;
    private String complemento;
    private String bairro;
    private String localidade;
    private String uf;
    private String ibge;
    private String gia;
    private String ddd;
    private String siafi;

    public String getCep() {
    return cep;
    }

    public void setCep(String cep) {
        this.cep = cep;
    }

    public String getLogradouro() {
```

```
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getComplemento() {
        return complemento;
    }

    public void setComplemento(String complemento) {
        this.complemento = complemento;
    }

    public String getBairro() {
        return bairro;
    }

    public void setBairro(String bairro) {
        this.bairro = bairro;
    }

    public String getLocalidade() {
        return localidade;
    }

    public void setLocalidade(String localidade) {
        this.localidade = localidade;
    }

    public String getUf() {
        return uf;
    }

    public void setUf(String uf) {
        this.uf = uf;
    }

    public String getIbge() {
        return ibge;
    }
```

```
    public void setIbge(String ibge) {
        this.ibge = ibge;
    }

    public String getGia() {
        return gia;
    }

    public void setGia(String gia) {
        this.gia = gia;
    }

    public String getDdd() {
        return ddd;
    }

    public void setDdd(String ddd) {
        this.ddd = ddd;
    }

    public String getSiafi() {
        return siafi;
    }

    public void setSiafi(String siafi) {
        this.siafi = siafi;
    }

    @Override
    public String toString() {
        return "Endereco [cep=" + cep + ", logradouro=" + logradouro +
", complemento=" + complemento + ", bairro="
    + bairro + ", localidade=" + localidade + ", uf=" + uf + ", ibge="
+ ibge + ", gia=" + gia + ", ddd="
    + ddd + ", siafi=" + siafi + "]";
    }
}
```

Vamos desenvolver a classe **ViaCepService.java** da seguinte forma:

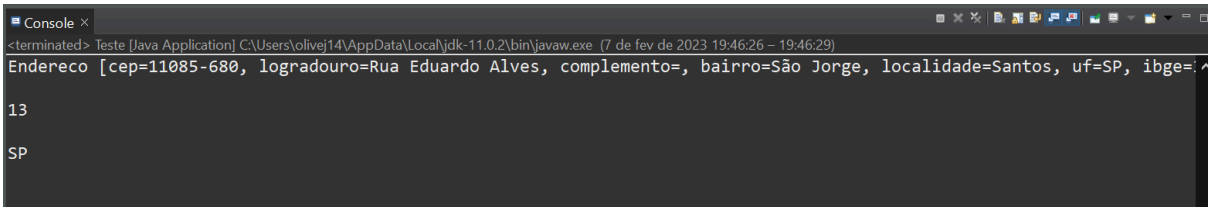
```java
package service;

import java.io.IOException;

import org.apache.http.HttpEntity;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.util.EntityUtils;

import com.google.gson.Gson;

import model.Endereco;

public class ViaCepService {

    public Endereco getEndereco(String cep) throws
ClientProtocolException, IOException {

        Endereco endereco = null;

        HttpGet request = new
HttpGet("https://viacep.com.br/ws/"+cep+"/json/");

        try(CloseableHttpClient httpClient =
HttpClientBuilder.create().disableRedirectHandling().build();

                CloseableHttpResponse response =
httpClient.execute(request)) {

            HttpEntity entity = response.getEntity();

            if(entity != null) {

                String result = EntityUtils.toString(entity);

                Gson gson = new Gson();

                endereco = gson.fromJson(result, Endereco.class);
            }
        }
```

```
        return endereco;
    }
}
```

E por fim, vamos desenvolver a classe **Teste.java** da seguinte forma:

```
package program;

import java.io.IOException;

import model.Endereco;
import service.ViaCepService;

public class Teste {

    public static void main(String[] args) {

        ViaCepService viacepservice = new ViaCepService();

        try {
            Endereco endereco = viacepservice.getEndereco("11085680");

            String ddd = endereco.getDdd();
            String uf = endereco.getUf();

            System.out.println(endereco + "\n");
            System.out.println(ddd + "\n");
            System.out.println(uf  + "\n");

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Bora executar o projeto e teremos esse resultado:



Pronto, o nosso programa fez o consumo da API e está exibindo na tela \o/

Ou podemos ter essa implementação também na classe **Teste.java**:

```java
package program;

import java.io.IOException;
import java.util.Scanner;

import model.Endereco;
import service.ViaCepService;

public class Teste {

    public static void main(String[] args) {

        ViaCepService viacepservice = new ViaCepService();

        Scanner ler = new Scanner(System.in);

        String cep;

        System.out.print("Digite o cep: ");
        cep = ler.next();


        try {
            Endereco endereco = viacepservice.getEndereco(cep);

            System.out.println(endereco.getLogradouro() + "\n");
            System.out.println(endereco.getBairro() + "\n");
            System.out.println(endereco.getLocalidade()  + "\n");

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```
    }
}
```

Entretanto, existe uma outra forma de consumirmos API, sem utilizar a lib **Gson**.

A outra possibilidade é a utilização da lib JSONObject. Com ela é possível navegar em diferentes níveis do JSON de retorno das APIs.

Para isso é necessário importar a dependência do JSONObject na arquivo pom.xml da seguinte forma:

```
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20220320</version>
</dependency>
```

A sua implementação na Classe Service deve ser da seguinte maneira:

```
package service;

import java.io.IOException;

import org.apache.http.HttpEntity;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.util.EntityUtils;
import org.json.JSONObject;

public class ConversaoService {

    public float getConversao(String conversor) throws
ClientProtocolException, IOException {

        HttpGet request = new
HttpGet("https://economia.awesomeapi.com.br/json/last/" + conversor);

        float valorMoeda = 0;
```

```
        try(CloseableHttpClient httpClient =
HttpClientBuilder.create().disableRedirectHandling().build();

                CloseableHttpResponse response =
httpClient.execute(request)) {

            HttpEntity entity = response.getEntity();

            if(entity != null) {

                String result = EntityUtils.toString(entity);

                JSONObject payload = new JSONObject(result);

                conversor = conversor.replace("-", "");


//System.out.print(payload.getJSONObject(conversor).get("ask"));

                valorMoeda = (float)
Double.parseDouble(payload.getJSONObject("USDBRL").get("ask").toString(
));
            }
        }

        return valorMoeda;
    }
}
```

Deixando o programa principal dessa maneira:

```
package program;

import java.io.IOException;
import java.util.Scanner;

import org.apache.http.client.ClientProtocolException;

import service.ConversaoService;

public class Teste {

```

```java
    public static void main(String[] args) throws
ClientProtocolException, IOException {
        // TODO Auto-generated method stub
        Scanner ler = new Scanner(System.in);

        float valorReal, valorConvertido, valorCotacao;

        System.out.print("Digite um valor em R$ para ser convertido em
Dólar: ");
        valorReal = ler.nextFloat();

        ConversaoService convService = new ConversaoService();
        valorCotacao = convService.getConversao("USD-BRL");

        valorConvertido = valorReal / valorCotacao;

        System.out.print("Resultado em US$: " + valorConvertido);
    }

}
```

Sugestão de APIs Públicas:

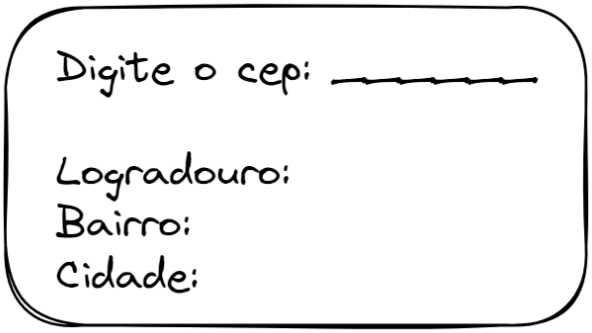
Repositório de APIs Públicas:
https://github.com/public-apis/public-apis

API Star Wars:
https://swapi.dev/

Brasil API:
https://brasilapi.com.br/

Exercícios

Crie um programa em Java que permita consultar endereços através de um Cep. O programa deve consumir a API do ViaCep (**Exemplo: viacep.com.br/ws/01001000/json/**), permitindo o usuário digitar o seu cep e obter as informações abaixo:



Correção: https://drive.google.com/drive/u/1/folders/1LtelMiaxkcDpVfsXZ4qphCT0TFisCJ_8

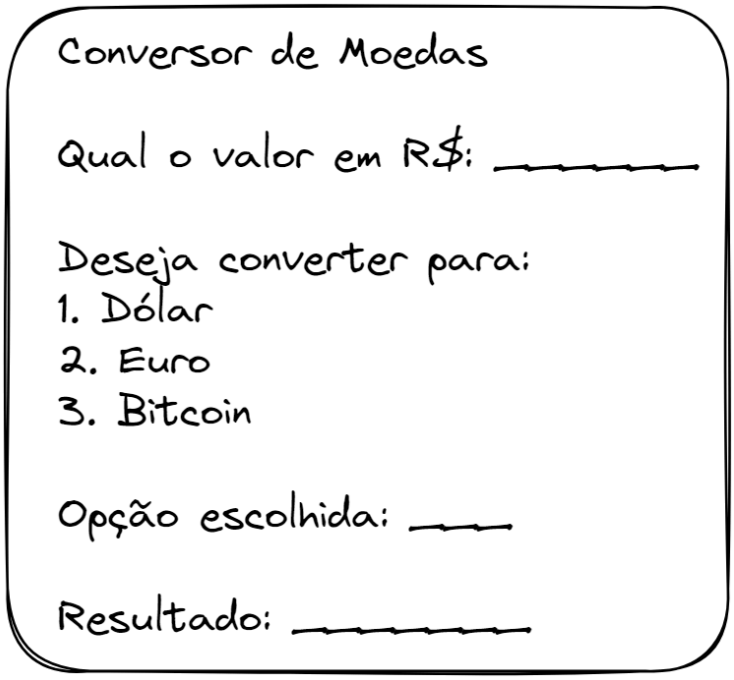
Crie um programa em Java que permita consultar Devs no GitHub. O programa deve consumir a API do GitHub (**Exemplo: https://api.github.com/users/joseffe10**), permitindo o usuário digitar algum login do GitHub e após isso retornar as informações abaixo:

Correção:
https://drive.google.com/drive/u/1/folders/19A7kcZsbaPNLEgvqEYUsbUmfSCXcE6SC

Crie um programa em Java que permita converter o valor de R$ para outras moedas. O programa deve consumir a API de Conversão (**Documentação:** https://docs.awesomeapi.com.br/api-de-moedas), permitindo o usuário digitar o valor em reais e escolher qual o destino da conversão:

Exemplo de uso (o atributo **ask** é o que possui o valor da conversão):

Real para Dólar: https://economia.awesomeapi.com.br/json/last/USD-BRL
Real para Euro: https://economia.awesomeapi.com.br/json/last/EUR-BRL
Real para Bitcoin: https://economia.awesomeapi.com.br/json/last/BTC-BRL



Correção:
https://drive.google.com/drive/u/1/folders/1t9furMNQWx1pGg1Akj63tPsJPciZuoS3

Crie um programa em Java que permita cadastrar pessoas e seu respectivo endereço. Ao cadastrar o endereço deixe a pessoa digitar apenas o cep, e o restante dos campos, deve ser completado de acordo com o retorno da API, como rua, bairro, cidade e estado. A pessoa, só deve cadastrar seu nome, email, número da casa e complemento.
Todo esse conteúdo deve ser cadastrado em um ArrayList. O programa deve permitir, Incluir, Alterar, Excluir e Consultar Pessoas.
As informações da Pessoa, devem ser: Nome, Email, Cep, **Rua, Bairro, Cidade, Estado**, Número da Casa e Complemento.

Correção:
https://drive.google.com/drive/u/1/folders/1mDA-PKMWRwspRkWMB0wVWOqp6KqkVJsW

Crie um programa em Java que permita cadastrar desenvolvedores e seu respectivo GitHub. Ao cadastrar o GitHub deixe a pessoa digitar apenas o seu login do Git, e o restante dos campos, deve ser completado de acordo com o retorno da API, como **id, nome no GitHub, quantidade de repos e quantidade de seguidores**. A pessoa, só deve cadastrar seu nome e email e o restante deve ser obtido pela api do GitHub.
Todo esse conteúdo deve ser cadastrado em um ArrayList. O programa deve permitir, Incluir, Alterar, Excluir e Consultar Devs.
API do GitHub: https://api.github.com/users/<<seu login no GitHub >>

Correção:
https://drive.google.com/drive/u/1/folders/10O4iHmN0D2_1bnnLtBd-J3aIVNmzbM9M

Crie um programa para testar se o nome de um domínio na Web está disponível para criação de um site. Exemplo: joseffe.com.br

https://brasilapi.com.br/api/registrobr/v1/joseffe.com.br
Se o campo "status" retornar AVAILABLE, aparecer na tela Domínio disponível, caso contrário, aparecer Domínio já utilizado e exibir a data de expiração.

Correção: