



Domain Driven Design



PROF. ELIANE RODRIGUES MARION SANTA ROSA
Profeliane.rosa@fiap.com.br

1

CONSUMINDO API: CEP



API - Application Programming Interface

FIAP





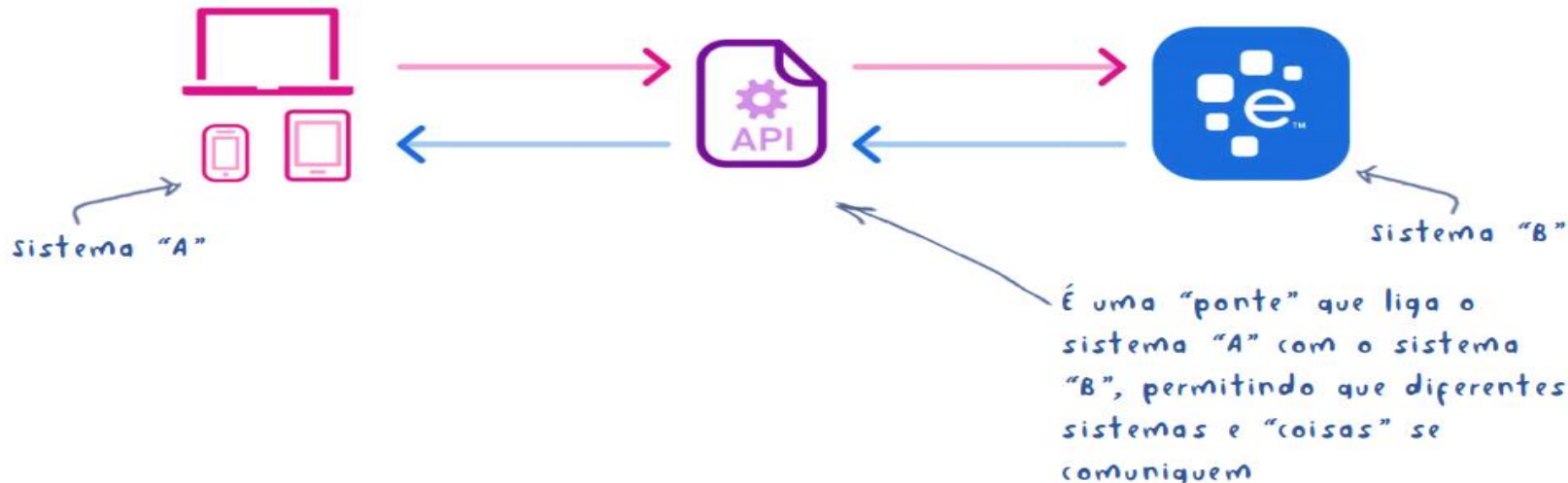
API - Application Programming Interface

- Você pode até não saber o que é API, mas elas certamente já fazem parte de sua rotina: efetuar um pagamento online, reservar um hotel para as férias, usar o Google Maps para descobrir como chegar a um endereço, são apenas alguns exemplos de atividades comuns de nosso dia-a-dia e que funcionam por meio de APIs.
- API é a sigla utilizada para **Application Programming Interface** ou, em português, Interface de Programação de Aplicações. Através das APIs, os **aplicativos** podem se **comunicar** uns com os outros sem conhecimento ou intervenção dos usuários.



API - Application Programming Interface

- Servem para integrar sistemas independente da linguagem de programação, proporcionando uma troca de informações de forma muito segura.





Como API funciona

- Em um app que utiliza GPS, por exemplo, não programamos linhas de código para conectar-se a um satélite para obter as coordenadas do local onde o usuário se encontra. O Google já possui tudo isso desenvolvido e, assim, a 99 Táxi, Uber, Cabify, Correios e etc podem, simplesmente, chamar a API do Google Maps.

Qualquer dispositivo (celular, PC, Carro, Totem, Site, Aplicativo) com conectividade com a Internet pode usar uma API, independentemente do sistema operacional ou da linguagem de programação.



As APIs já são realidade para diversos usos e seu crescimento ainda vai longe



Representações

FIAP

- Vimos que uma API permite a interoperabilidade entre usuários e aplicações. Com isso, é muito importante pensarmos em algo padronizado e, de preferência, de fácil representação e compreensão por humanos e máquinas. Qual dos exemplos abaixo você escolheria para informar o endereço em uma carta?



Representações

```
<?xml version="1.0" encoding="UTF-8"?>
<endereco>
  <rua>Rua Alcântara,113</rua>
  <cidade>São Paulo</cidade>
</endereco>
```

↖ A representação em XML é mais verbosa e exige um esforço extra de quem está escrevendo

```
{
  "endereco":{
    "rua": "Rua Alcântara,113",
    "cidade":"São Paulo"
  }
}
```

↖ O JSON, além de ser um formato leve para troca de dados, é também muito simples de ler



- Derivado do JavaScript, JSON é um acrônimo de **JavaScript Object Notation**.



CHECK LIST

Vantagens do JSON

- Leitura mais simples
- É tipado
- Arquivo com tamanho reduzido
- Velocidade maior na execução e transporte de dados
- Quem utiliza? Google, Facebook, Yahoo!, Twitter...



Consumindo API – CEP dos Correios

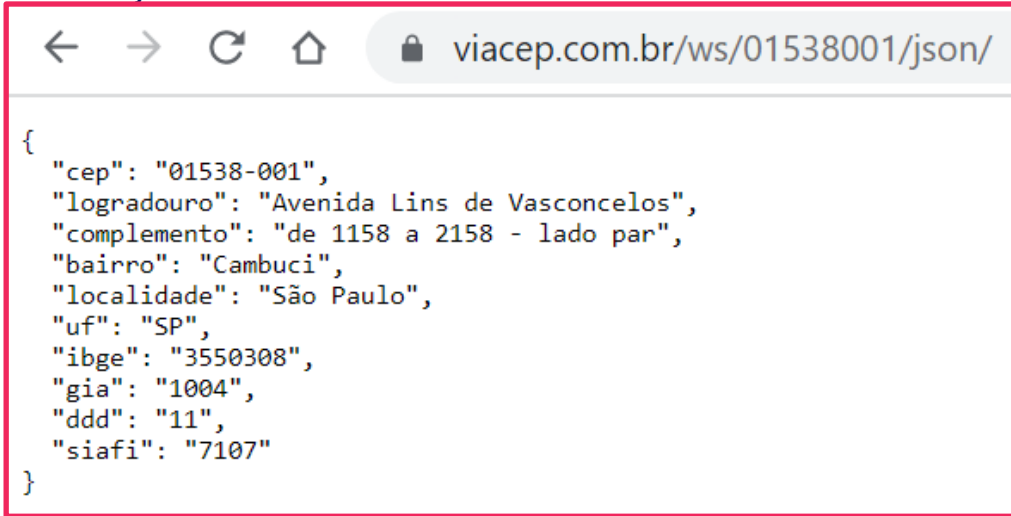
FIAP

No site www.viacep.com.br tem as instruções de como enviar um cep e receber as informações, iremos focar no que está sendo apontado em rosa na próxima imagem.

Para consultar um cep basta entrar no site com o link abaixo e substituir o que está em vermelho pelo CEP que deseja consultar o endereço.

<https://viacep.com.br/ws/01538001/json/>

Após entrar no endereço abaixo o que o site exibiu de endereço, nesse caso é o endereço da FIAP da Aclimação e o CEP é 01538-001





VIACEP

Consulte CEPs de todo o Brasil

Procurando um [webservice](#) gratuito e de alto desempenho para consultar Códigos de Endereçamento Postal (CEP) do Brasil? Utilize nosso serviço, melhore a qualidade de suas aplicações web e colabore para manter esta base de dados atualizada.

Acessando o webservice de CEP

Para acessar o webservice, um CEP no formato de **{8}** dígitos deve ser fornecido, por exemplo: "01001000". Após o CEP, deve ser fornecido o tipo de retorno desejado, que deve ser "json" ou "xml".

Exemplo de pesquisa por CEP:

viacep.com.br/ws/01001000/json/





POSTMAN

FIAP

GET https://viacep.com.br/ws/01538001/json/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (14) Test Results Status: 200 OK Time: 387 ms Size: 755 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "cep": "01538-001",
3   "logradouro": "Avenida Lins de Vasconcelos",
4   "complemento": "de 1158 a 2158 - lado par",
5   "bairro": "Cambuci",
6   "localidade": "São Paulo",
7   "uf": "SP",
8   "ibge": "3550308",
9   "gia": "1004",
10  "ddd": "11",
11  "siafi": "7107"
12 }
```

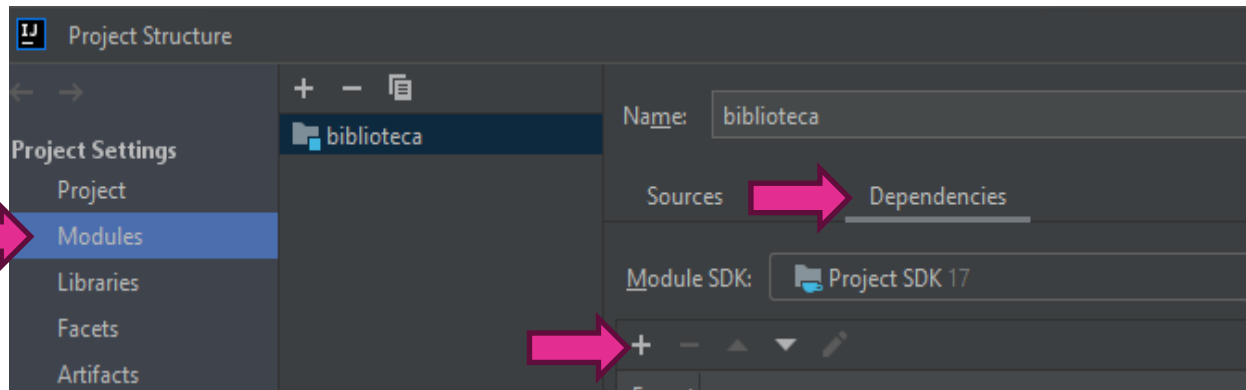
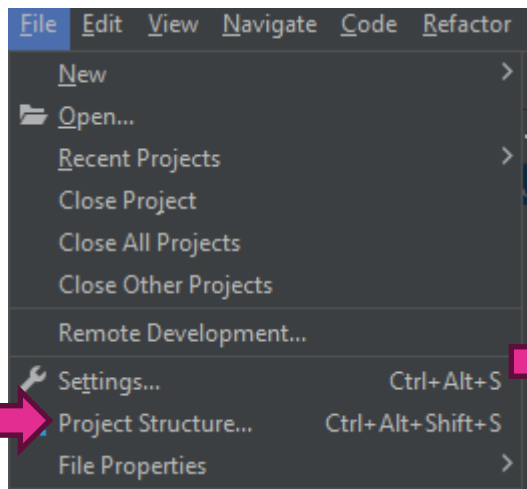
2

Usando o IntelliJ



Inserindo as dependências

FIAP



Adicionar as dependências que foram disponibilizadas no Teams



Classe ViaCepService -Entendendo o código

```
public class ViaCepService {  
    public Endereco buscaEndereco(String cep){  
        try {  
            String endereco = "https://viacep.com.br/ws/" + cep + "/json/";  
            HttpClient client = HttpClient.newHttpClient();  
            HttpRequest request = HttpRequest.newBuilder()  
                .uri(URI.create(endereco))  
                .build();  
  
            HttpResponse<String> response = client  
                .send(request,  
                    HttpResponse.BodyHandlers.ofString());  
            return new Gson().fromJson(response.body(), Endereco.class);  
        } catch (Exception e) {  
            throw new RuntimeException(e);  
        }  
    }  
}
```



Classe Teste -Entendendo o código

```
public class TestaInsercaoComBuscaCep {  
    public static void main(String[] args) {  
        Scanner leitor = new Scanner(System.in);  
        Scanner ent = new Scanner(System.in);  
        ViaCepService consulta = new ViaCepService();  
        Endereco meuEndereco = new Endereco();  
        System.out.println("Digite o número do CEP para consulta: ");  
        var cep = leitor.nextLine();  
        try{  
            meuEndereco = consulta.buscaEndereco(cep);  
            System.out.println("Digite o código do endereco: ");  
            meuEndereco.setId(ent.nextInt());  
            meuEndereco.setCep(cep);  
            System.out.println(meuEndereco);  
        }catch (RuntimeException e){  
            System.out.println(e.getMessage());  
            System.out.println("Finalizando a aplicação");  
        }  
    }  
}
```




Testando sua conexão

Podemos criar outra classe ou utilizarmos a classe TestaInsercao que já criamos.

```
public class TestaInsercao {  
  
    public static void main(String[] args) {  
        Scanner ent = new Scanner(System.in);  
        Endereco endereco = new Endereco();  
        EnderecoDao dao = new EnderecoDao();  
        System.out.println("Digite o id do endereco: ");  
        int id = ent.nextInt();  
        endereco = dao.buscarPorId(id);  
        System.out.println(endereco);  
    }  
}
```

Para esta última linha funcionar precisamos sobrescrever o método toString() na classe Endereco



Reescrita do método toString

Na classe `Endereco` podemos sobrescrever o método `toString`.

```
@Override
public String toString() {
    return "Endereco\n" +
        logradouro +
        ", " + complemento +
        ", cep = " + cep +
        bairro +
        ", " + localidade +
        ", " + uf;
}
```



Exibição dos dados

Caso optarmos por não sobrescrever o método `toString` precisaremos montar a mensagem no método `main` utilizando os getters.

```
System.out.println(endereco.getLogradouro());  
System.out.println(", " + endereco.getComplemento());
```

3

Buscar todos endereços



Método buscarTodosEnderecos - EnderecoDao

```
public List<Endereco> buscarTodosEnderecos(){
    List<Endereco> enderecos = new ArrayList<>();
    conexao = GerenciadorBD.obterConexao();
    PreparedStatement comandoSql = null;
    try{
        comandoSql = conexao.prepareStatement( sql: "Select * from endereco ");
        ResultSet rs = comandoSql.executeQuery();
        while(rs.next()){
            Endereco endereco = new Endereco();
            endereco.setId(rs.getInt( columnIndex: 1));
            endereco.setCep(rs.getString( columnIndex: 2));
            endereco.setLogradouro(rs.getString( columnIndex: 3));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return enderecos;
}
```



Método buscarTodosEnderecos - EnderecoDao

FIAP

```
        endereco.setComplemento(rs.getString( columnIndex: 4));  
        endereco.setBairro(rs.getString( columnIndex: 5));  
        endereco.setLocalidade(rs.getString( columnIndex: 6));  
        endereco.setUf(rs.getString( columnIndex: 7));  
        enderecos.add(endereco);  
    }  
} catch (SQLException e){  
    e.printStackTrace();  
}  
return enderecos;  
}
```



Entendendo o código

Criamos um método chamado `buscarTodosEnderecos` que retorna uma lista de endereço e dentro declaramos um objeto chamado `enderecos` do tipo `Lista` de endereços.

```
public ArrayList<Endereco> buscarTodosEnderecos(){  
    List<Endereco> enderecos = new ArrayList<Endereco>();  
}
```

Obtemos a conexão através do método `obterConexao` que escrevemos na classe `GerenciadorBD`.

```
conexao = GerenciadorBD.obterConexao();
```



Entendendo o código

Nesta linha estamos criando um objeto chamado `comandoSQL` do tipo `PreparedStatement`. Dentro do `try` tentaremos buscar todos os endereços cadastrados no banco de dados.

```
PreparedStatement comandoSQL = null;
```




Entendendo o código

FIAP

```
comandoSQL = conexao.prepareStatement("select * from endereco ");
```

- Nesta linha fazemos o uso da conexão com o banco de dados para preparar um statement com a instrução SQL select. Devemos escrever a instrução da mesma forma que escrevemos no banco de dados.



- Após montarmos nosso objeto comandoSQL o próximo passo é de fato executar a instrução select no banco, isso é feito com o método `executeQuery()`, e o resultado da busca estamos guardando no objeto `rs` do tipo `ResultSet`.

```
ResultSet rs = comandoSQL.executeQuery();
```

- Diferente da busca por id, neste caso precisamos verificar se a busca retornou algum resultado, e percorrer o `resultSet` já que o esperado são várias linhas de retorno.



Entendendo o código

FIAP

```
while(rs.next()){  
    Endereco endereco = new Endereco();  
    endereco.setId(rs.getInt( columnIndex: 1));  
    endereco.setCep(rs.getString( columnIndex: 2));  
    endereco.setLogradouro(rs.getString( columnIndex: 3));  
    endereco.setComplemento(rs.getString( columnIndex: 4));  
    endereco.setBairro(rs.getString( columnIndex: 5));  
    endereco.setLocalidade(rs.getString( columnIndex: 6));  
    endereco.setUf(rs.getString( columnIndex: 7));  
    enderecos.add(endereco);  
}
```

Criamos um objeto endereco do tipo Endereço e atribuímos o resultado da linha do resultSet no objeto



Entendendo o código

```
while(rs.next()){  
    Endereco endereco = new Endereco();  
    endereco.setId(rs.getInt( columnIndex: 1));  
    endereco.setCep(rs.getString( columnIndex: 2));  
    endereco.setLogradouro(rs.getString( columnIndex: 3));  
    endereco.setComplemento(rs.getString( columnIndex: 4));  
    endereco.setBairro(rs.getString( columnIndex: 5));  
    endereco.setLocalidade(rs.getString( columnIndex: 6));  
    endereco.setUf(rs.getString( columnIndex: 7));  
    enderecos.add(endereco);  
}
```

Enquanto existir linha
no ResultSet, avançamos



Entendendo o código

```
Endereco endereco = new Endereco();
```

Criamos um objeto
endereco do tipo
Endereço e atribuímos
o resultado da linha
do resultSet no
objeto

```
endereco.setId(rs.getInt(1));
```

endereco

id:

1

cep:

logradouro:

complemento:

bairro:

localidade:

uf:

```
select * from endereco;
```

Saída do Script x

Resultado da Consulta x

Todas as Linhas Extraídas: 1 em 0,058 segundos

IDENDereco	CEP	RUA	COMPLEMENTO	BAIRRO	CIDADE	UF
1	101310914	AV. PAULISTA, 1106 4 ANDAR		BELA VISTA	SÃO PAULO	SP



Entendendo o código

FIAP

```
while(rs.next()){  
    Endereco endereco = new Endereco();  
    endereco.setId(rs.getInt( columnIndex: 1));  
    endereco.setCep(rs.getString( columnIndex: 2));  
    endereco.setLogradouro(rs.getString( columnIndex: 3));  
    endereco.setComplemento(rs.getString( columnIndex: 4));  
    endereco.setBairro(rs.getString( columnIndex: 5));  
    endereco.setLocalidade(rs.getString( columnIndex: 6));  
    endereco.setUf(rs.getString( columnIndex: 7));  
    enderecos.add(endereco)  
}
```

Adicionamos objeto
endereco na lista



Entendendo o código

FIAP

- Acabamos de atribuir os dados devolvidos pela consulta no banco à lista de endereços (enderecos).

- Por fim fechamos nossa conexão

```
conexao.close();  
comandoSQL.close();
```

- E precisamos devolver a lista que geramos, por isso retornamos o enderecos.

```
return enderecos;
```

3

**Testando a busca de
todos enderecos**



Testando sua conexão

Podemos criar outra classe ou utilizarmos a classe TestaInsercao que já criamos.

```
public static void main(String[] args) {  
    Scanner ent = new Scanner(System.in);  
    Endereco endereco = new Endereco();  
    EnderecoDao dao = new EnderecoDao();  
  
    List<Endereco> listaEnderecos = new ArrayList<Endereco>();  
    String dados = "";  
    listaEnderecos = dao.buscarTodosEnderecos();  
}
```



Testando sua conexão

Continuação.....

```
for (Endereco endereco2 : listaEnderecos) {  
    dados += "=====\n";  
    dados += "Id: " + endereco2.getId() + "\n";  
    dados += "Endereço: " + endereco2.getLogradouro() + "\n";  
    dados += "Complemento: " + endereco2.getComplemento() + "\n";  
    dados += "CEP: " + endereco2.getCep() + "\n";  
    dados += "Bairro: " + endereco2.getBairro() + "\n";  
    dados += "Cidade: " + endereco2.getLocalidade() + "\n";  
    dados += "Estado: " + endereco2.getUf() + "\n";  
    dados += "=====\n";  
}  
System.out.println(dados);  
}
```