



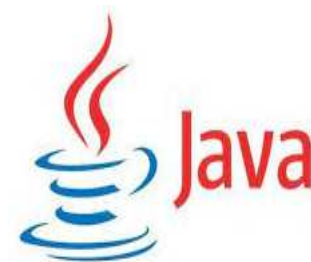
Residência em Software

Residência em Tecnologia da Informação e Comunicação

SpringBoot: Introdução às Requisições

Professor:

Alvaro Degas Coelho



INSTITUIÇÃO EXECUTORA



COORDENADORA



APOIO



Para começar

- HTTP permite comunicação bidirecional
 - De servidor para o cliente e vice versa
- Um teste
- Levante nosso sistema de rede social
- Faça a busca pelo endereço
 - `localhost:8080/listausuarios/?name=mestre`

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Feb 03 16:05:27 GMT-03:00 2024

There was an unexpected error (type=Not Found, status=404).

No static resource listausuarios.

```
org.springframework.web.servlet.resource.NoResourceFoundException: No static resource listausuarios.  
    at org.springframework.web.servlet.resource.ResourceHttpRequestHandler.handleRequest(ResourceHttpRequestHandler.java:585)  
    at org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter.handle(HttpRequestHandlerAdapter.java:52)  
    at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1089)  
    at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:979)  
    at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1014)  
    at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:903)  
    at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:564)  
    at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:885)  
    at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:658)  
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:205)  
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)  
    at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)  
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:174)  
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)  
    at org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:100)  
    at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)  
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:174)  
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)  
    at org.springframework.web.filter.FormContentFilter.doFilterInternal(FormContentFilter.java:93)  
    at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)  
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:174)  
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)  
    at org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter.java:201)  
    at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)  
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:174)  
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)  
    at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:167)  
    at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:90)  
    at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:482)  
    at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:115)  
    at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:93)  
    at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:74)  
    at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:340)  
    at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:391)  
    at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:63)  
    at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:896)  
    at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1744)  
    at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:52)  
    at org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191)  
    at org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:650)
```

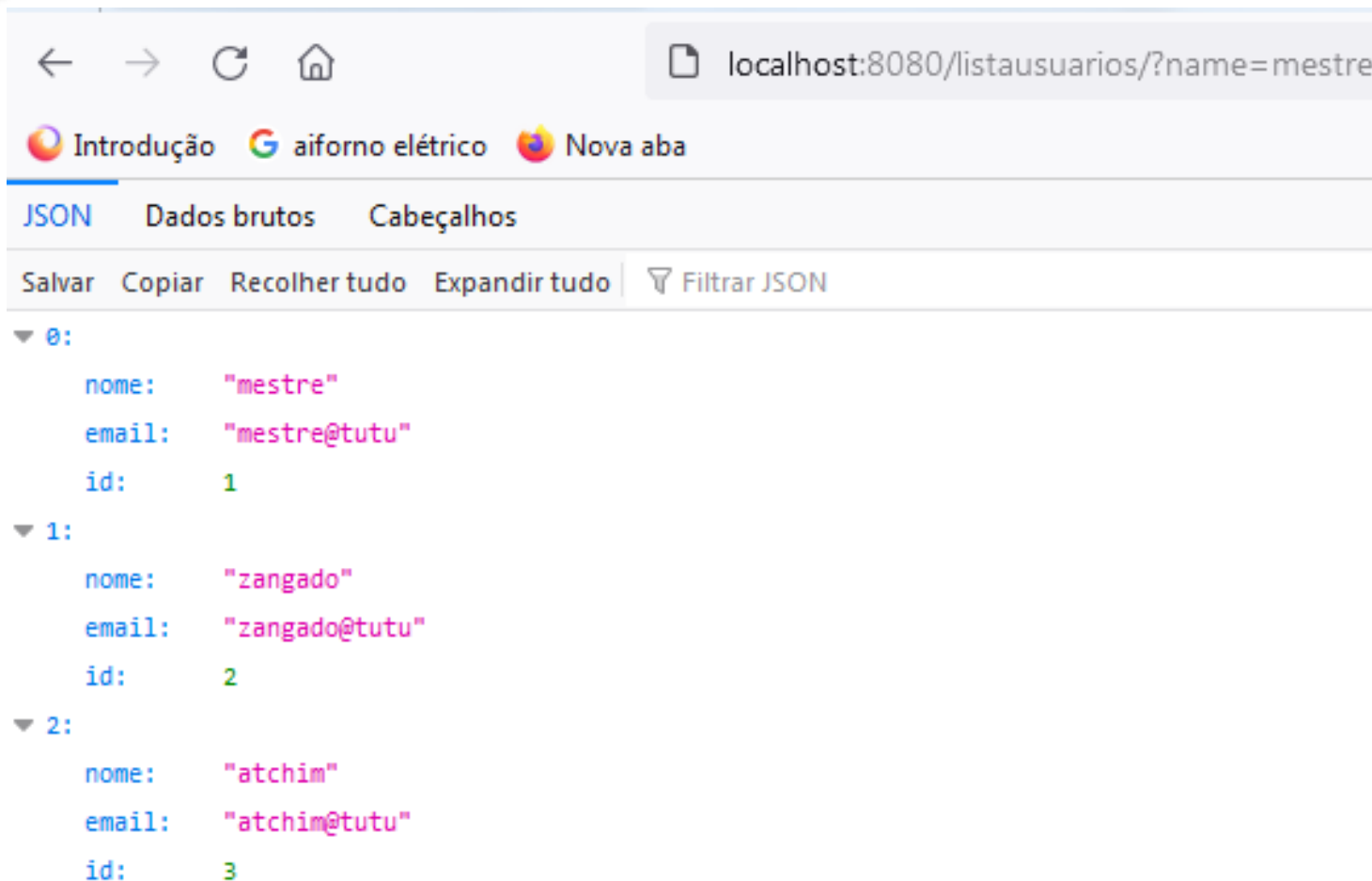
A aplicação não lidou

- Padrão http
 - ? Serve para passar um parâmetro
- Podemos receber este parâmetro!
 - Restrição importante: o nome do parâmetro tem que ser o mesmo
- localhost:8080/listausuarios/?name=mestre
- Parâmetro: name
- Vamos usar este parâmetro para limitar a busca
 - mestre

Seleções de dados

- Por exemplo: usuários chamados “mestre”
 - Lembre que Nome não é obrigatoriamente único
- Alterar o método em ListaUsersController
 - Criar um parâmetro (String name) na entrada da função
- Por enquanto não precisamos fazer nada.
Apenas receber
 - Sysout()
- Again:
`localhost:8080/listausuarios/?name=mestre`

Sem erros na requisição



```
0:
  nome: "mestre"
  email: "mestre@tutu"
  id: 1
1:
  nome: "zangado"
  email: "zangado@tutu"
  id: 2
2:
  nome: "atchim"
  email: "atchim@tutu"
  id: 3
```


Parâmetro recebido

```
Console × JUnit
RedesocialApplication [Java Application] C:\Program Files\Java\jdk-19.0.2\bin\javaw.exe
2024-02-03T16:14:40.280-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.297-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.297-03:00 INFO 11636 --- [ restartedMain] w
2024-02-03T16:14:40.308-03:00 INFO 11636 --- [ restartedMain] c
2024-02-03T16:14:40.312-03:00 INFO 11636 --- [ restartedMain] c
2024-02-03T16:14:40.313-03:00 INFO 11636 --- [ restartedMain] c
2024-02-03T16:14:40.313-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.335-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.338-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.342-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.343-03:00 WARN 11636 --- [ restartedMain] o
2024-02-03T16:14:40.372-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.376-03:00 INFO 11636 --- [ restartedMain] j
2024-02-03T16:14:40.441-03:00 WARN 11636 --- [ restartedMain] J
2024-02-03T16:14:40.529-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.537-03:00 INFO 11636 --- [ restartedMain] o
2024-02-03T16:14:40.542-03:00 INFO 11636 --- [ restartedMain] c
2024-02-03T16:14:40.544-03:00 INFO 11636 --- [ restartedMain] .
2024-02-03T16:14:46.091-03:00 INFO 11636 --- [nio-8080-exec-1] o
2024-02-03T16:14:46.091-03:00 INFO 11636 --- [nio-8080-exec-1] o
2024-02-03T16:14:46.092-03:00 INFO 11636 --- [nio-8080-exec-1] o
Parametro name=mestre
```

Codigo do controller

```
@RestController
public class ListaUsersController {

    @Autowired
    private UsuarioRepository usuarioRepository;

    @RequestMapping("/listausuarios/")
    public List<UserDTO> listaUsuarios(String name) {
        System.out.println("Parametro name="+name);
        List<Usuario> listaUsuarios = (ArrayList<Usuario>) usuarioRepository.findAll();
        List<UserDTO> lista = new ArrayList<UserDTO>();
        for (Usuario u: listaUsuarios) {
            UserDTO ud = new UserDTO(u);
            lista.add(ud);
        }
        return lista;
    }
}
```


Dados do cliente para a aplicação ok

- Usar o parâmetro
 - Listar só “mestre”
- É necessário aplicar um filtro
 - SQL (where)
- Springdata
 - Há um método “padrão”
 - `findBy<Coluna>`
- No caso
 - `findByName(name)`
- Será necessário criar esta assinatura no UserRepository

Como

- Observe o nome do atributo na classe Usuario
 - Nome
- Na interface
 - findByNome()
 - Retorna uma lista de usuários
- O Spring JPA já mapeia o nome do método (findBy**Nome**) com o atributo (**nome**)
- Injeção de dependência tem efeito colateral: acoplamento!
 - E se o nome da coluna mudar?

TO DO...

1. Criar o `findBy?()` na interface `Repository` (retorna uma lista de `Usuarios!`)
2. Trocar o `findAll()` pelo método adequado

Programa aí!

ListaUsersController

```
@RestController
public class ListaUsersController {

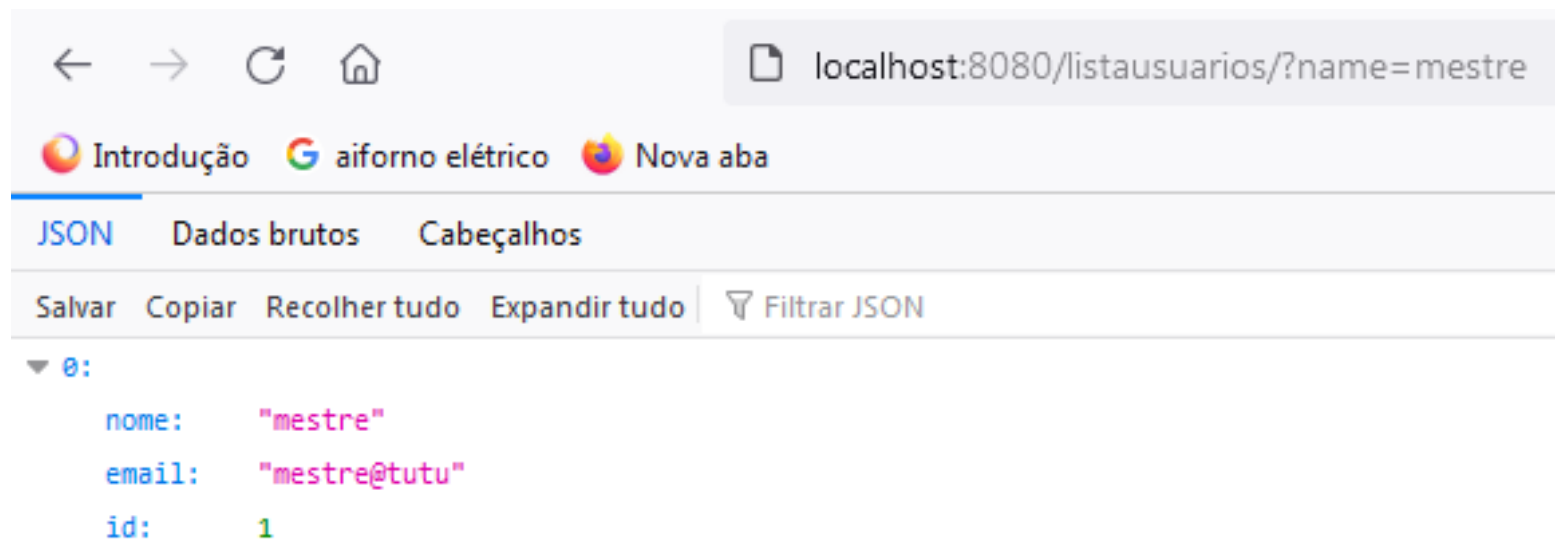
    @Autowired
    private UsuarioRepository usuarioRepository;

    @RequestMapping("/listausuarios/")
    public List<UserDTO> listaUsuarios(String name) {
        List<Usuario> listaUsuarios =
            (ArrayList<Usuario>) usuarioRepository.findByNome(name);
        List<UserDTO> lista = new ArrayList<UserDTO>();
        for (Usuario u: listaUsuarios) {
            UserDTO ud = new UserDTO(u);
            lista.add(ud);
        }
        return lista;
    }
}
```

UsuarioRepository

```
package com.redesocial.redesocial.repository;  
  
import java.util.List;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import com.redesocial.redesocial.modelo.Usuario;  
  
public interface UsuarioRepository extends JpaRepository<Usuario, Integer> {  
    List<Usuario> findByName(String nome);  
}
```

localhost:8080/listausuarios/?name=mestre



Mas agora dá erro quando o
parâmetro não é usado



Corrijaí!



Verbos HTTP

- Padrão HTTP (e HTTPS) define métodos de requisição
- Responsáveis or indicar a ação requerida
- Comumente chamados de HTTP Verbs (Verbos HTTP)
- Os mais comuns (que usaremos aqui)
 - GET, POST, PUT e DELETE

Propriedades dos verbos

- **Safe** - é seguro de ser executado porque não altera o estado do servidor (apenas busca dados)
 - GET
- **Cacheable** - a resposta pode ser guardada (cache) para ser usada depois (se for possível)
 - Evita uma requisição desnecessária ao servidor
 - GET
- **Idempotent** - Uma requisição feita repetidas vezes tem o mesmo efeito
 - Deixa o servidor no mesmo estado que a primeira delas deixou
 - GET, PUT, DELETE

Usar os verbos corretamente

- Melhora o desempenho de sua aplicação
 - Exemplo: pode-se usar PUT para inserir um novo dado
 - Deixamos de usar a nosso favor sua idempotência
 - Perda de desempenho
- Há uma regra geral que é proposta como padrão
- Seguiremos aqui

Utilizando os verbos HTTP

- GET
 - Para recuperar dados
- POST
 - Para inserir dados
- PUT
 - Para inserir dados (update)
- DELETE
 - Para remover dados
- ...

Uma mudança na aplicação

- Vamos excluir o endpoint /listausuarios/
- Vamos usar o endpoint /usuarios/
 - E os verbos para as operações correspondentes
- GET: recuperar usuários
- POST: inserir usuários
- PUT: para alterar dados de usuários
- DELETE: para excluir usuários

Associando verbos a métodos

- No Controller
- A anotação `@RequestMapping` vai ser modificada
 - Acrescenta o método (verbo)
 - `@RequestMethod(value = "/usuarios", method=RequestMethod.GET)`
- Outra opção (melhor: padrão Expert)
 - Uma anotação associando o Controller a uma URL
 - Todos os métodos responderão no mesmo endpoint!
 - Outra anotação em cada método dando o verbo
 - `@RestController + @RequestMapping("/usuarios")` -> na classe
 - `@<Get/Post/Put/Delete>Mapping` -> Nos métodos

Novas annotations

- **@GetMapping**
 - Indica que o método será executado com o verbo GET
- **@PostMapping**
 - Indica que o método será executado com o verbo POST
- **@PutMapping**
 - Indica que o método será executado com o verbo PUT
- **@DeleteMapping**
 - Indica que o método será executado com o verbo DELETE



Residência
em Software

Corrigir o Controller

Corrigir o Controller

```
+ import java.util.ArrayList;

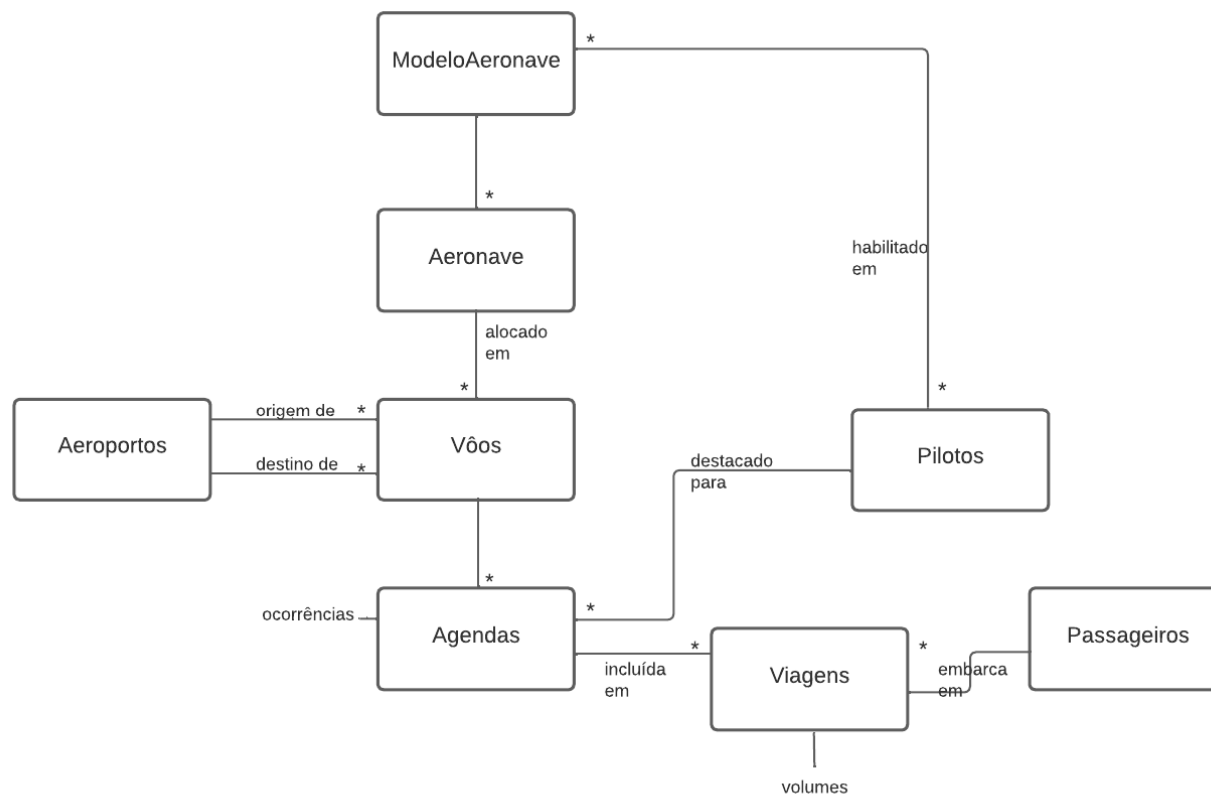
@RestController
@RequestMapping("/listausuarios/")
public class ListaUsersController {

    @Autowired
    private UsuarioRepository usuarioRepository;

    @GetMapping
    public List<UserDTO> listaUsuarios(String name) {
        List<Usuario> listaUsuarios;
        if (name!=null)
            listaUsuarios =
                (ArrayList<Usuario>) usuarioRepository.findByNome(name);
        else
            listaUsuarios =
                (ArrayList<Usuario>) usuarioRepository.findAll();
        List<UserDTO> lista = new ArrayList<UserDTO>();
        for (Usuario u: listaUsuarios) {
            UserDTO ud = new UserDTO(u);
            lista.add(ud);
        }
        return lista;
    }
}
```

Exercício

- Voltando ao diagrama da companhia aérea



Relembrando as entidades já implementadas

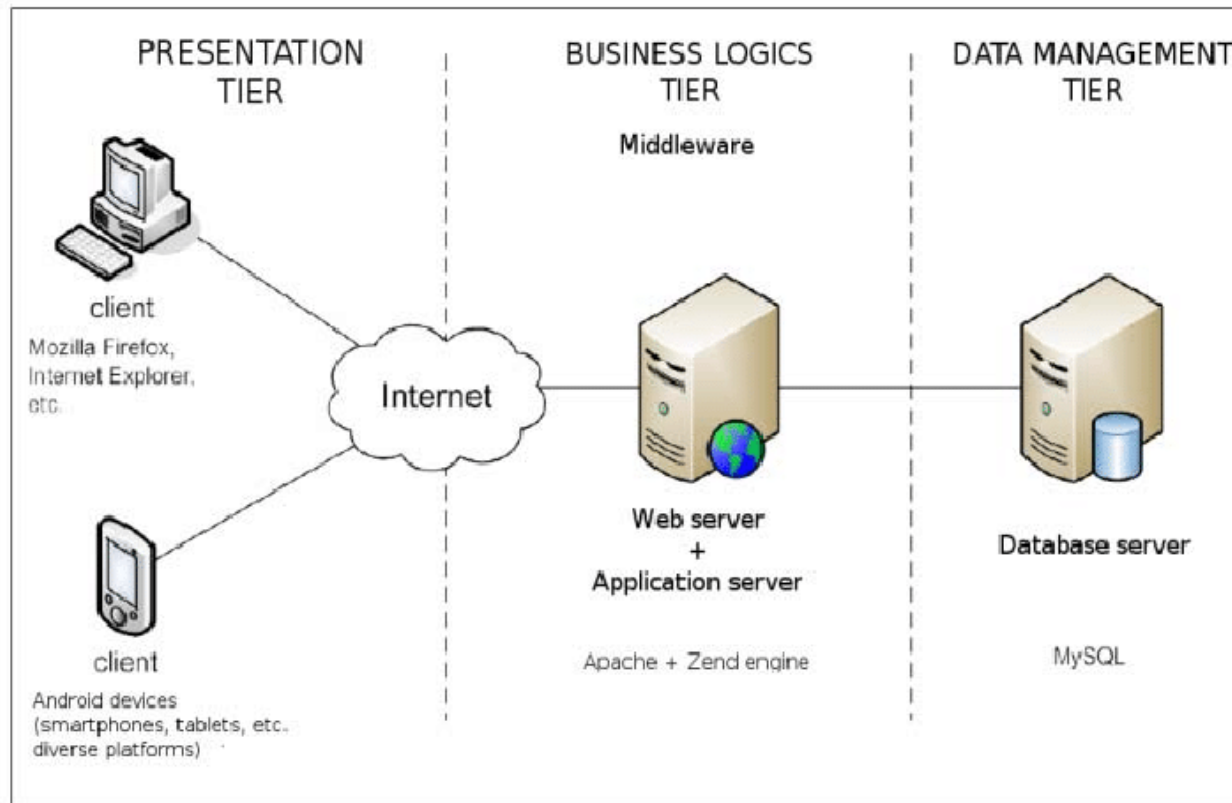
- Piloto(\$Id, Nome, NumBreve)
 - Ex. (01, “Juca Pires”, “12354”)
- ModeloAeronave(\$Id, Fabricante, Nome)
 - Ex. (01, “Embraer”, “EMB195”)
- Aeroporto(\$Id, ICAO, Nome)
 - Ex. (01, “SBIL”, “Jorge Amado”)

Seu trabalho

- Modifique a arquitetura
- Crie um controller para cada assunto
 - /pilotos/
 - /aeroportos/
 - /modelo aeronave/
- Crie os procedimentos para atender aos requests
 - /pilotos/
 - ?nome= Juca Pires
 - /modelo aeronave/
 - ?fabricante=Boeing - ?nome=Emb195
 - /aeroportos/
 - ?icao=SBIL - ?nome=Jorge Amado

Epílogo da aula

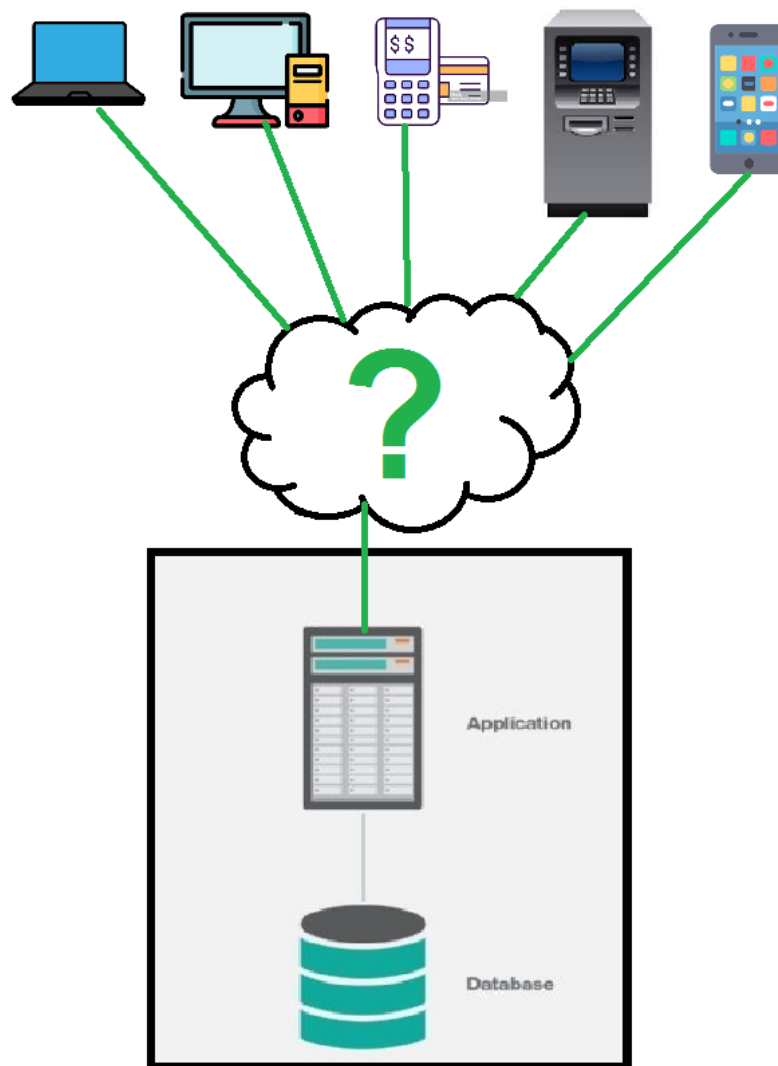
- E prólogo da próxima
- Voltando às 3 (ou mais) camadas



Comunicação entre Camadas

- Da camada de Aplicação (lógica de negócio) para a camada de Dados
 - Drivers proprietários (SQL), ODBC, JDBC, ORM (JPA-Hibernate), etc.
- E da camada de apresentação para a camada de Aplicação?
- Problema mais complexo
 - Hardware, Sistemas Operacionais, Software...

Comunicação entre Camadas



Algum Padrão

- Anos 80/90 (e até hoje)
- Brasil (sim, somos protagonistas)
 - Febraban
- Padrão para comunicação
 - Descrição de operações bancárias
 - Os “campos” do código de barras
- Saber mais?
 - <https://portal.febraban.org.br/paginas/33/pt-br/>

Mais genérico

- Padrão febraban serve para dados bancários
 - Bancos obrigam empresas a aderir
 - Ou então fica de fora das transações eletrônicas
- Precisa-se de um padrão mais flexível
 - Aberto
- Voltamos aos modelos baseados em Bancos de Dados Hierárquicos
 - XML ou...



Residência
em Software

{JSON}

Como funciona

- Uma aplicação na camada de Interface precisa fazer uma requisição
 - Android, IOS, Windows, Linux, ...
- Formato da requisição
 - Endpoint (endereço completo mais porta)
 - Verbo
 - Corpo
- Formato da resposta
 - Código (HTML)
 - Corpo

O corpo da requisição

- E da resposta
- Padrão JSON
- Exemplo
 - {“nome” : “atchim”,
 - “email” : “atchim@tutu”,
 - “Id” : 3 }

Buscar CEP (outro protagonismo brasileiro)

GET /consulta/cep/{cep} ▼

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  "cep": "string",
  "tipoCep": "string",
  "subTipoCep": "string",
  "uf": "string",
  "cidade": "string",
  "bairro": "string",
  "endereco": "string",
  "complemento": "string",
  "codigoIBGE": "string"
}
```

Exemplo

Chamada

```
https://h-apigateway.conectagov.estaleiro.serpro.gov.br/api-cep/v1/consulta/cep/60130240
```

Retorno

```
{  
  "cep": "60130240",  
  "tipoCep": "logradouro",  
  "subTipoCep": "S",  
  "uf": "CE",  
  "cidade": "Fortaleza",  
  "bairro": "São João do Tauape",  
  "endereco": "Avenida Pontes Vieira",  
  "complemento": "De 2 Até 1550 Lado Par",  
  "codigoIBGE": ""  
}
```

Exemplo

Não é necessário
saber de que
plataforma
(Hardware, SO,
Linguagem, App)
vem a chamada

Chamada

`https://`

`.serpro.gov.br/api-cep/v1/consulta/cep/60130240`

Retorno

```
{  
  "cep": "60130240",  
  "tipoCep": "logradouro",  
  "subTipoCep": "S",  
  "uf": "CE",  
  "cidade": "Fortaleza",  
  "bairro": "São João do Tauape",  
  "endereco": "Avenida Pontes Vieira",  
  "complemento": "De 2 Até 1550 Lado Par",  
  "codigoIBGE": ""  
}
```

Exemplo

Chamada

`https://`

Retorno

```
{  
  "cep": "60130240",  
  "tipoCep": "logradouro",  
  "subTipoCep": "S",  
  "uf": "CE",  
  "cidade": "Fortaleza",  
  "bairro": "São João do Tauape",  
  "endereco": "Avenida Pontes Vieira",  
  "complemento": "De 2 Até 1550 Lado Par",  
  "codigoIBGE": ""  
}
```

Não é necessário
saber de que
plataforma
(Hardware, SO,
Linguagem, App)
vem a chamada

`.serpro.gov.br/api-cep/v1/consulta/cep/60130240`

Não é necessário
saber de que
plataforma
(Hardware, SO,
Linguagem, App)
volta a resposta