



Curso Presencial Programação Fullstack

Aula 10

Prof. MSc. Kelson | Senior Software Engineer



PROFKELSON.DEV
BORA CODAR?

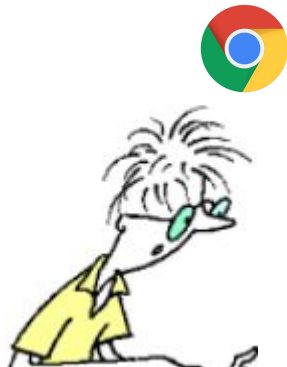
Introdução ao Projeto 🚀

- Este projeto é um Gerenciador de Fornecedores, seguindo a arquitetura em camadas.
- Utiliza Spring Boot, JPA, Lombok, Jakarta Validation, MapStruct, entre outras tecnologias.
- Vamos explorar cada camada, entender seu papel e como elas se conectam! 🔍



Exemplo de Comunicação

Bob acessa
<https://www.profkelson.dev>



GET Request



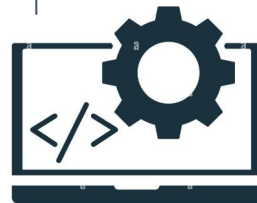
GET Request



Front end

O frontend ficará responsável por fornecer a página web (*interface*)

GET Request



BACK END

Mas quem busca no banco de dados é o nosso amigo *back-end*



PostgreSQL



Atenção: Esta é apenas uma exemplificação, não necessariamente condiz com a realidade das tecnologias utilizadas pelo referido site.

Bob acessou
www.profkelson.dev e
recebeu a página inicial



Response 200 OK



Response 200 OK



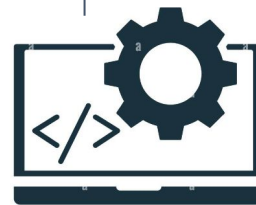
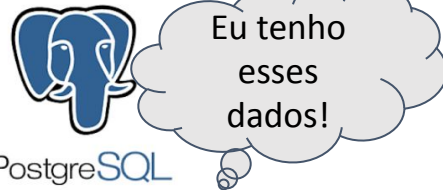
Front end

Body Response
com o JSON da
página inicial do
site



Response 200 OK

PostgreSQL



BACK END

```
1 {  
2   "imagem": "logo.png",  
3   "mensagemTopo": "Curso Presencial Programação Fullstack - Turma 04 - Início: 15/03/2025 ÚLTIMAS VAGAS!",  
4   "mensagemBotaoTopo1": "FALAR NO WHATSAPP",  
5   "linkBotaoTopo1": "https://api.whatsapp.com/...",  
6 }
```

Curso Presencial
Programação Fullstack
- Turma 04 - Início:
15/03/2025 ÚLTIMAS
VAGAS!

FALAR NO WHATSAPP

INSCREVA-SE AGORA, VEM
SER DEU!

Introdução ao REST



- ♦ REST (Representational State Transfer) é um **estilo arquitetural** para comunicação entre sistemas na web.
- ♦ Criado por Roy Fielding em 2000.
- ♦ Baseado no protocolo HTTP, utilizando recursos (resources) representados por URLs.

Principais características:

- ✓ Stateless (sem estado: cada requisição é independente).
- ✓ Client-Server (separação entre cliente e servidor).
- ✓ Cacheable (pode armazenar respostas para melhorar performance).
- ✓ Uniform Interface (padrão de comunicação).
- ✓ Layered System (arquitetura pode ter camadas).

📌 Exemplo de recurso RESTful:

<https://api.meusite.com/fornecedores/123> → Recurso Fornecedor com ID 123.

{ REST }

O que é uma RESTful API?



API RESTful:

💡 Uma API que segue os princípios REST para expor dados e operações via web.

Como funciona?

- 1 O cliente (front-end, app, sistema externo) faz uma requisição HTTP para a API.
- 2 A API RESTful processa a requisição e acessa os dados no banco.
- 3 A API retorna uma resposta HTTP com o resultado.

📌 Exemplo de fluxo de requisição:

💻 Cliente: "Quero listar todos os fornecedores!"

✉️ Faz GET em → <https://api.meusite.com/fornecedores>

📬 Recebe resposta JSON:

```
1 ✓ [
2   { "id": 1, "nome": "Fornecedor A" },
3   { "id": 2, "nome": "Fornecedor B" }
4 ]
```

✅ APIs RESTful são padronizadas, simples e fáceis de escalar! 🚀

HTTP: O protocolo das APIs 🌐

📌 HTTP (HyperText Transfer Protocol) é o protocolo usado para comunicação na web e APIs RESTful.

Características do HTTP:

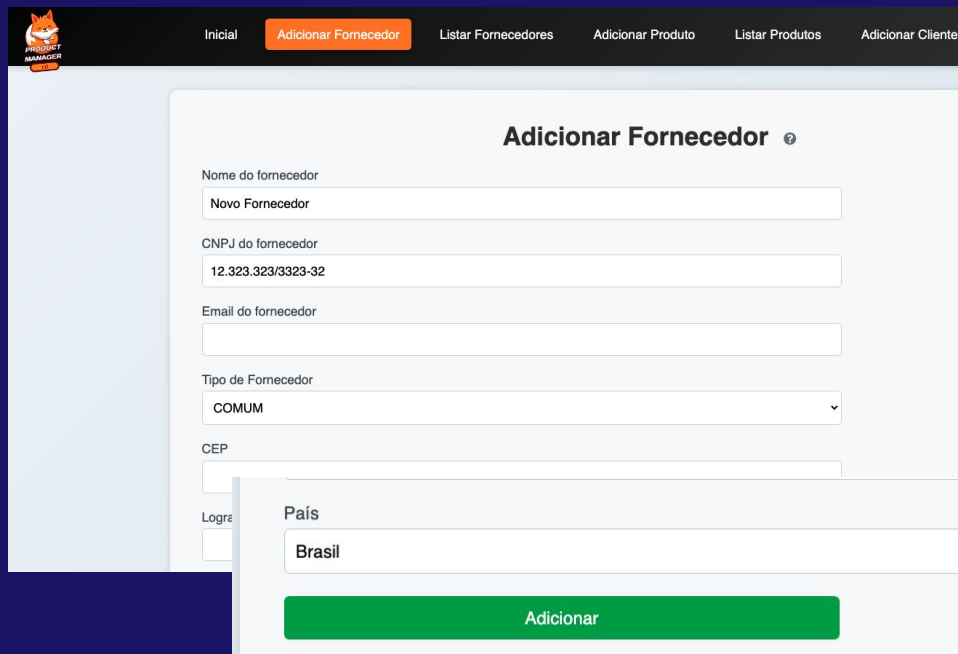
- **Baseado em requisições e respostas.**
- Stateless (cada requisição é independente).
- **Usa verbos HTTP para definir ações.**



Métodos HTTP: O que cada um faz?

Método	Descrição	Exemplo
GET	Buscar dados	<code>GET /fornecedores</code>
POST	Criar um novo recurso	<code>POST /fornecedores</code>
PUT	Atualizar um recurso por completo	<code>PUT /fornecedores/1</code>
PATCH	Atualizar parte de um recurso	<code>PATCH /fornecedores/1</code>
DELETE	Remover um recurso	<code>DELETE /fornecedores/1</code>

Métodos HTTP: O que cada um faz?



The screenshot shows a web application interface for a 'PRODUCT MANAGER' system. The top navigation bar includes links for 'Inicial', 'Adicionar Fornecedor' (highlighted in orange), 'Listar Fornecedores', 'Adicionar Produto', 'Listar Produtos', and 'Adicionar Cliente'. The main content area is titled 'Adicionar Fornecedor' with a help icon. It contains a form with the following fields: 'Nome do fornecedor' (text input with placeholder 'Novo Fornecedor'), 'CNPJ do fornecedor' (text input with placeholder '12.323.323/3323-32'), 'Email do fornecedor' (text input), 'Tipo de Fornecedor' (dropdown menu with 'COMUM' selected), 'CEP' (text input), and 'País' (text input with 'Brasil' selected). A green 'Adicionar' button is at the bottom of the form. A 'Logar' checkbox is partially visible on the left side of the form.

Adicionar Fornecedor ⓘ

Nome do fornecedor
Novo Fornecedor

CNPJ do fornecedor
12.323.323/3323-32

Email do fornecedor

Tipo de Fornecedor
COMUM

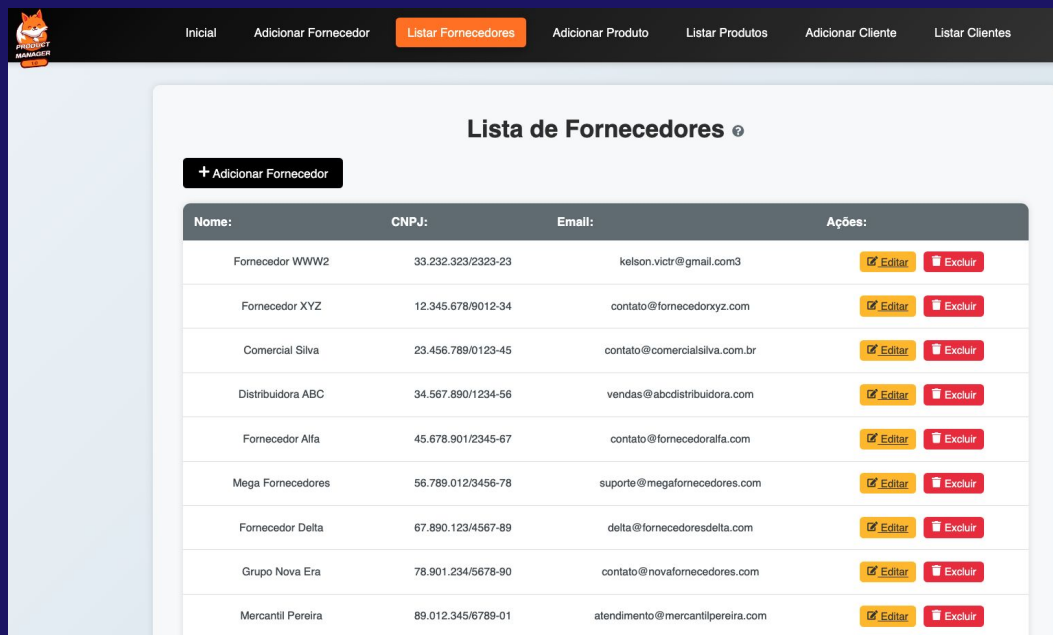
CEP

Logar

País
Brasil

Adicionar

Métodos HTTP: O que cada um faz?




The screenshot shows a web application with a dark blue header. The header contains a logo on the left and a navigation menu with the following items: 'Inicial', 'Adicionar Fornecedor', 'Listar Fornecedores' (highlighted in orange), 'Adicionar Produto', 'Listar Produtos', 'Adicionar Cliente', and 'Listar Clientes'. Below the header, the main content area is titled 'Lista de Fornecedores' with a small help icon. There is a button '+ Adicionar Fornecedor' and a table with the following data:

Nome:	CNPJ:	Email:	Ações:
Fornecedor WWW2	33.232.323/2323-23	kelson.victr@gmail.com3	✎ Editar 🗑 Excluir
Fornecedor XYZ	12.345.678/9012-34	contato@fornecedortex.com	✎ Editar 🗑 Excluir
Comercial Silva	23.456.789/0123-45	contato@comercialsilva.com.br	✎ Editar 🗑 Excluir
Distribuidora ABC	34.567.890/1234-56	vendas@abcdistribuidora.com	✎ Editar 🗑 Excluir
Fornecedor Alfa	45.678.901/2345-67	contato@fornecedoralfa.com	✎ Editar 🗑 Excluir
Mega Fornecedores	56.789.012/3456-78	suporte@megafornevedores.com	✎ Editar 🗑 Excluir
Fornecedor Delta	67.890.123/4567-89	delta@fornecedoresdelta.com	✎ Editar 🗑 Excluir
Grupo Nova Era	78.901.234/5678-90	contato@novafornevedores.com	✎ Editar 🗑 Excluir
Mercantil Pereira	89.012.345/6789-01	atendimento@mercantilpereira.com	✎ Editar 🗑 Excluir

Métodos HTTP: O que cada um faz?



 Inicial Adicionar Fornecedor Listar Fornecedores Adicionar Produto Listar Produtos Adicionar Cliente Listar Clientes

Editar Fornecedor ⓘ

Nome do fornecedor

CNPJ do fornecedor

Email do fornecedor

Tipo de Fornecedor

CEP

Logradouro

Número

Métodos HTTP: O que cada um faz?

Bairro

Lourdes

Cidade

Belo Horizonte

Estado

MG


País

Brasil

Editar

Métodos HTTP: O que cada um faz?



 Inicial Adicionar Fornecedor **Listar Fornecedores** Adicionar Produto Listar Produtos Adicionar Cliente Listar Clientes

Lista de Fornecedores

+ Adicionar Fornecedor

Nome:	CNPJ:	Email:	Ações:
Fornecedor WWW2	33.232.323/2323-23	kelson.victr@gmail.com3	✎ Editar 🗑 Excluir
Fornecedor XYZ	12.345.678/9012-34	contato@fornecedorxyz.com	✎ Editar 🗑 Excluir
Comercial Silva	23.456.789/0123-45	contato@comercialsilva.com.br	✎ Editar 🗑 Excluir
Distribuidora ABC	34.567.890/1234-56	vendas@abcdistribuidora.com	✎ Editar 🗑 Excluir
Fornecedor Alfa	45.678.901/2345-67	contato@fornecedoralfa.com	✎ Editar 🗑 Excluir
Mega Fornecedores	56.789.012/3456-78	suporte@megaforneceadores.com	✎ Editar 🗑 Excluir

HTTP Responses: O que significam?



Quando uma API responde a uma requisição, ela envia um código de status HTTP.

```
1 {  
2   "status": 404,  
3   "error": "Not Found",  
4   "message": "Fornecedor não encontrado"  
5 }
```

Código	Significado	Exemplo
200 OK	Sucesso na requisição	GET /fornecedores
201 Created	Recurso criado	POST /fornecedores
204 No Content	Requisição sem resposta	DELETE /fornecedores/1
400 Bad Request	Erro no envio de dados	Enviar JSON errado
401 Unauthorized	Requer autenticação	Sem token válido
403 Forbidden	Acesso negado	Sem permissão
404 Not Found	Recurso não encontrado	ID inexistente
500 Internal Server Error	Erro no servidor	Falha interna

Camadas do Projeto

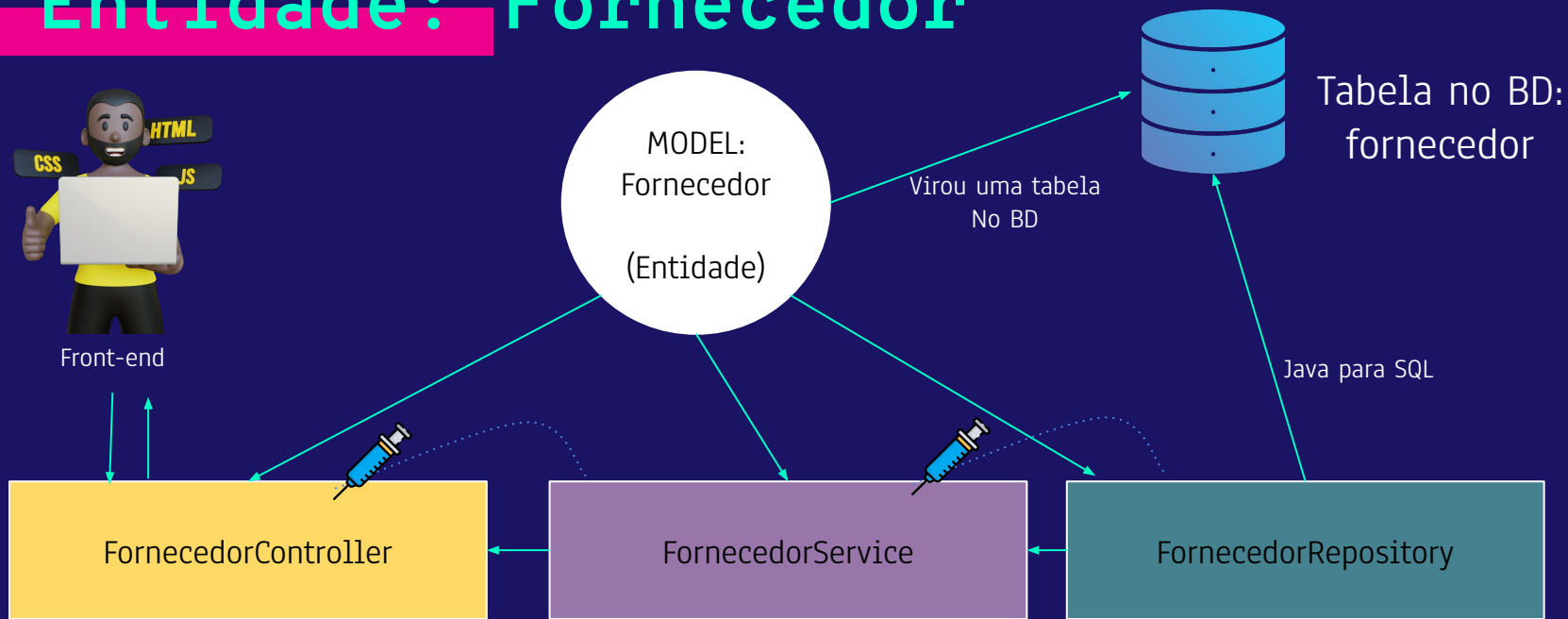


Nosso projeto é organizado da seguinte forma:

- **Model (Entidades)** 🏛️
- DTOs (Data Transfer Objects) 📦
- Enums (Enumerações) 📜
- **Repository (Acesso a Dados)** 🗄️
- **Service (Regras de Negócio)** 🛠️
- Mapper (Conversões DTO <-> Entidade) 🔄
- Exception Handling (Tratamento de Erros) ⚠️
- **Controller (Exposição via API)** 🌐

```
br.com.gerenciador.api
├── controller
│   └── FornecedorController
├── dto
│   ├── EnderecoDTO
│   ├── FornecedorRequestDTO
│   └── FornecedorResponseDTO
├── enums
│   └── TipoFornecedorEnum
├── exception
│   └── GlobalExceptionHandler
├── mapper
│   └── FornecedorMapper
├── model
│   ├── Endereco
│   └── Fornecedor
├── repository
│   └── FornecedorRepository
├── service
│   ├── FornecedorService
│   └── FornecedorServiceImpl
```

Entidade: Fornecedor



1 Model – Representação do Banco de Dados

A camada Model contém as entidades JPA, que representam tabelas no banco de dados.

Destaques:

- @Entity → Marca a classe como uma entidade do JPA.
- @Table(name = "fornecedor") → Define o nome da tabela no banco.
- @Id @GeneratedValue(strategy = GenerationType.IDENTITY) → Indica chave primária auto-incrementada.
- @Column(nullable = false, unique = true, length = 14) → Restrições e tamanho do campo.

```
1  @Entity
2  @Table(name = "fornecedor")
3  public class Fornecedor {
4      @Id
5      @GeneratedValue(strategy = GenerationType.IDENTITY)
6      private Long id;
7
8      @Column(name = "nome", nullable = false, length = 100)
9      private String nome;
10 }
```

 **Dica:** Use @CreationTimestamp e @UpdateTimestamp para salvar datas automaticamente.

2



DTO – Transferência de Dados



Os DTOs evitam expor diretamente as entidades e melhoram a validação dos dados.



Benefícios:

- Segurança  (não expõe toda a estrutura do banco).
- Facilidade de manipulação de dados .
- Melhor controle sobre a entrada e saída da API.




Dica: Use `@Valid` nos DTOs no Controller para validar os dados automaticamente.

```
1 public record FornecedorRequestDTO(  
2     @NotBlank(message = "Nome é obrigatório")  
3     @Size(max = 100) String nome,  
4  
5     @NotBlank(message = "CNPJ é obrigatório")  
6     @Pattern(regexp = "\\d{14}", message = "CNPJ deve conter 14 dígitos numéricos") String cnpj  
7 ) {}
```

3 Enums – Representação de Tipos Fixos

- Utilizado para representar tipos predefinidos no sistema.
- Armazena valores fixos para evitar erros em campos como tipoFornecedor.

```
1 public enum TipoFornecedorEnum {  
2     COMUM, PREMIUM  
3 }
```


 **Dica:** Use `@Enumerated(EnumType.STRING)` para armazenar enums como texto no banco.

4 Repository – Acesso ao Banco de Dados



- Responsável pela persistência dos dados.
- Estende JpaRepository, que já fornece operações CRUD sem precisar escrever SQL.

```
1 @Repository
2 public interface FornecedorRepository extends JpaRepository<Fornecedor, Long> {
3 }
```

 Dica: Você pode criar métodos personalizados, como `findByName(String nome)`.

5

Service – Regras de Negócio



- Contém a lógica do sistema e evita código no Controller.
- Trabalha diretamente com os Repositories.

```
1  @Service
2  @RequiredArgsConstructor
3  public class FornecedorServiceImpl implements FornecedorService {
4      private final FornecedorRepository fornecedorRepository;
5      private final FornecedorMapper fornecedorMapper;
6
7      @Override
8      public FornecedorResponseDTO criarFornecedor(FornecedorRequestDTO dto) {
9          Fornecedor fornecedor = fornecedorMapper.toEntity(dto);
10         Fornecedor fornecedorSalvo = fornecedorRepository.save(fornecedor);
11         return fornecedorMapper.toDTO(fornecedorSalvo);
12     }
13 }
```

🔧 Dica: Use @Transactional em métodos que envolvem múltiplas operações no banco.

6 Mapper – Conversão DTO ↔ Entidade

- Usa MapStruct para transformar objetos de forma automática e eficiente.

```
1 @Mapper(componentModel = "spring")
2 ✓ public interface FornecedorMapper {
3     @Mapping(target = "id", ignore = true)
4     Fornecedor toEntity(FornecedorRequestDTO dto);
5
6     FornecedorResponseDTO toDTO(Fornecedor fornecedor);
7 }
```

📌 Dica: MapStruct reduz o código boilerplate e melhora a legibilidade.

7

Controller – Expondo a API 🌐

- Responsável por expor os endpoints da aplicação.

```
1  @RestController
2  @RequestMapping("/fornecedores")
3  @RequiredArgsConstructor
4  ✓ public class FornecedorController {
5      private final FornecedorService fornecedorService;
6
7      @PostMapping
8  ✓  public ResponseEntity<FornecedorResponseDTO> criarFornecedor(@Valid @RequestBody FornecedorRequestDTO dto) {
9      return ResponseEntity.ok(fornecedorService.criarFornecedor(dto));
10 }
11 }
```