


Spring + Angular: a stack de desenvolvimento usada no GPES



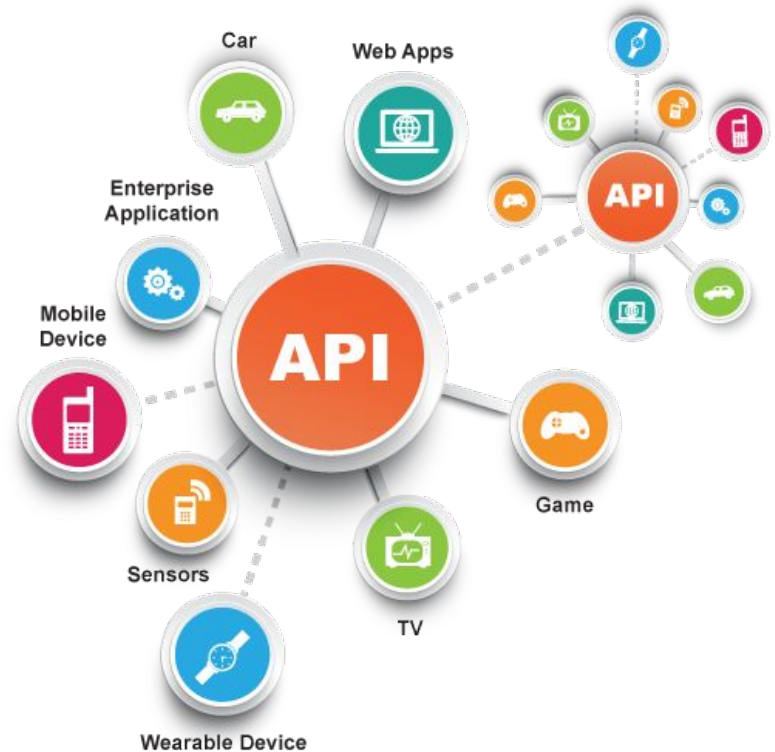
Nosso principal objetivo é estimular a pesquisa, inovação e extensão na Área de Engenharia de Software para disponibilização e reuso das suas boas práticas na academia e na indústria.

Criamos uma API para divulgar as boas práticas pesquisadas. Agora queremos aprender e compartilhar com os colegas sobre Spring Framework e Angular.

API

- Modelo de desenvolvimento utilizado
- Software distribuído
- Independente de tecnologia
- Endpoint's
- Comunicação via JSON

Exemplo: <https://viacep.com.br/ws/58037345/json/>





Spring Stack

- Evolução do spring framework
- Spring Boot
- Spring Data
- Spring MVC



Spring Boot

- Convenção sobre configuração
- Starters
- Facilidade de implantação



Inicializador Spring

- Dependências
- Inclusão de spring starters



Site: <https://start.spring.io/>



Iniciar pela criação da camada MODEL

1. Criar classe de modelo
2. Incluir o mapeamento objeto relacional (JPA - Hibernate)
3. Conexão de banco de dados



Criar repositório

- Operações prontas para manipulação do banco de dados
- Facilitador para criação de consultas



Criar classe de serviço

- Adicionar lógica de funcionamento dos métodos
- Adicionar conexão com o repository
- Injeção de dependência spring (Autowired)
- Componentes Spring



Criar controller

- Função do controlador
- Endpoint da aplicação
- Comunicação com o service
- Padrões para nomenclatura de endpoints



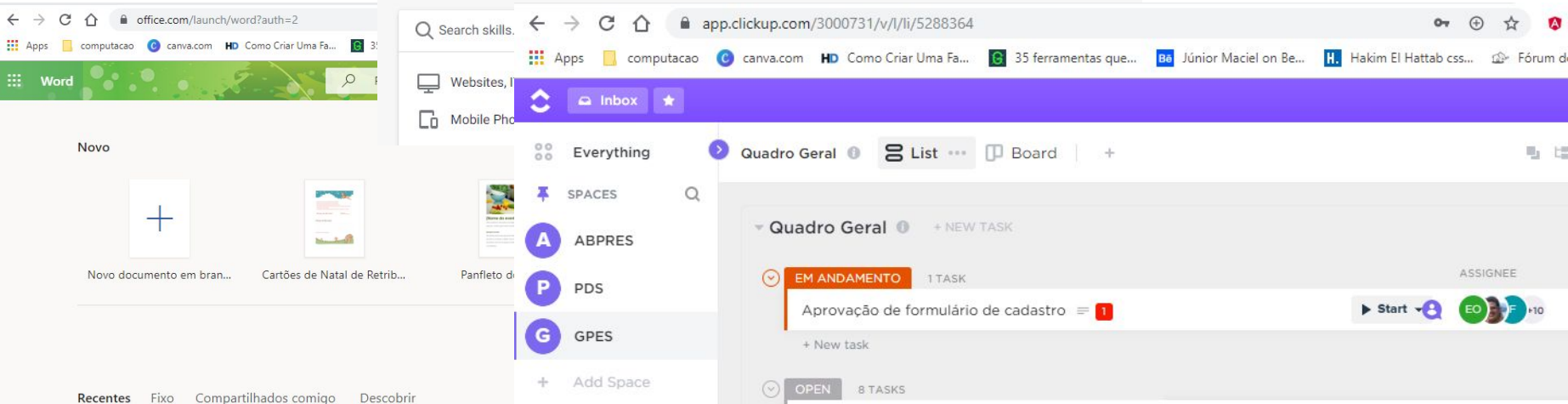
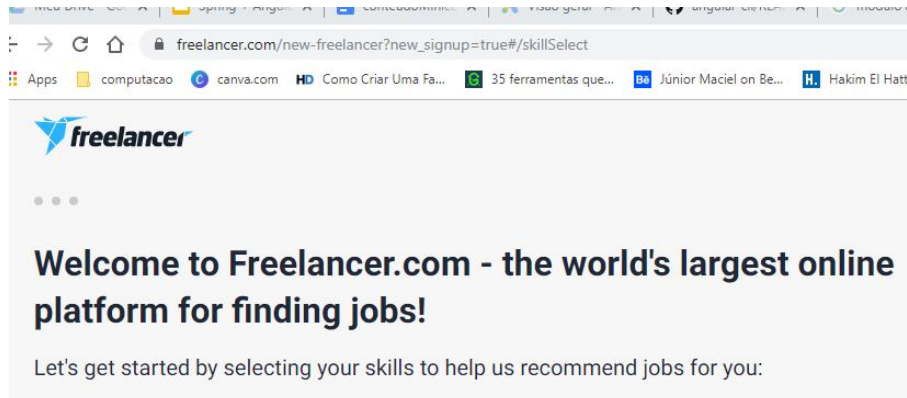
Testar endpoint

- Utilizar Swagger
- Testes de integração

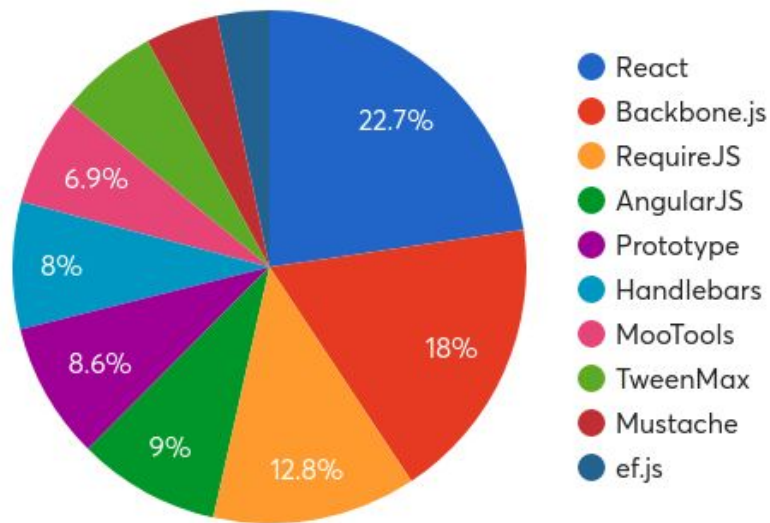


Angular

- Baseado em Typescript
- Arquitetura de componentes
- Modularização
- Conceitos OO
- Angular 8.3.19 Novembro de 2019
- stackblitz.com
- Angular x React x Vue



Market Leaders



Blocos principais do framework



COMPONENTES

DATA BINDING

DIRETIVAS

INJEÇÃO DE
DEPENDÊNCIA

METADATA

MÓDULO

ROTEAMENTO

SERVIÇOS

TEMPLATE

Blocos principais do framework

MÓDULO

Ajuda a organizar e modularizar a aplicação. Assim, podem haver diversos diretórios. Com o módulo se escolhe o que será exposto para o resto da aplicação, por exemplo.

```
src > app > TS app.module.ts > ...
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { CategoriaComponent } from './abpres/categoria/categoria.component';
7  import { TemplateModule } from './template/template.module';
8  import { FormsModule } from '@angular/forms';
9  import { HttpClientModule } from '@angular/common/http';
10
11
12  @NgModule({
13    declarations: [
14      AppComponent,
15      CategoriaComponent
16    ],
17    imports: [
18      BrowserModule,
19      AppRoutingModule,
20      FormsModule,
21      HttpClientModule,
22      TemplateModule
23    ],
24    providers: [],
25    bootstrap: [AppComponent]
26  })
27  export class AppModule { }
```


Blocos principais do framework

O *componente* é o que o usuário vai ver. Ele encapsula o Template, Metadada e associação dos dados do componente com o html para mostra-los através do data binding.

São controlados pelo componentes:

- A raiz do aplicativo com os links de navegação.
- Listagem de dados etc

COMPONENTES

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'abpres';
}
```

Blocos principais do framework

TEMPLATE

Diz ao Angular como processar uma classe. Com ele o Angular sabe onde encontrar os principais blocos de construção. Metadados, templates e componentes integram a visualização do framework.

```
<!--- HEADER --->
<app-header></app-header>
<div class="container">
  <!--- AREA DE CONTEUDO --->
  <ng-content></ng-content>
</div>
<!--- FOOTER --->
<app-footer></app-footer>
```

```
<nav class="navbar navbar-expand-lg navbar-light bg-light" style="margin-bottom: 50px;">
  <div class="container">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
        </li>
      </ul>
    </div>
  </div>
</nav>
<footer class="text-muted fixed-bottom">
  <div class="container">
    <p>Rodapé</p>
    <p>New to Bootstrap? <a href="https://getbootstrap.com/">Visit the homepage</a> or read our <a
      href="/docs/4.3/getting-started/introduction/">getting
      started guide</a>.</p>
  </div>
</footer>
```

Blocos principais do framework



METADATA

Diz ao Angular como processar uma classe. Com ele o Angular sabe onde encontrar os principais blocos de construção.

Metadados, templates e componentes integram a visualização do framework.

SEM o Metadata o complemento é só uma classe

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

Blocos principais do framework

SERVIÇOS

Um serviço é uma classe com um propósito. Faz algo bem específico, que pode ser usado por toda a aplicação. Aqui fica a chamada lógica de negócio. Ele pode ser injetado em outras classes usando a Injeção de Dependência.

```
@Injectable({
  providedIn: 'root'
})
export class CategoriaService {

  url = 'http://localhost:8080/categorias'

  constructor(private http: HttpClient) {}

  create(data) {
    return this.http.post(this.url, data)
      .toPromise()
  }
}
```

Blocos principais do framework

INJEÇÃO DE DEPENDÊNCIA

Em geral se injeta serviços em componentes, que é uma maneira de fornecer uma nova instância de uma classe com as dependências totalmente formadas.

```
@Injectable({
  providedIn: 'root'
})
export class CategoriaService {

  url = 'http://localhost:8080/categorias'

  constructor(private http: HttpClient) {}

  create(data) {
    return this.http.post(this.url, data)
      .toPromise()
  }
}
```

Blocos principais do framework



DATA BINDING

Dados da aplicação são apresentados no componente utilizando o recurso de data binding.

Blocos principais do framework

DATA BINDING

`[(ngModel)]` = "someValue"



= "someValue"

```
<app-layout>
  <div class="col-md-6">
    <form #categoriaForm="ngForm" (ngSubmit)="onSubmit()">
      <div class="form-group">
        <label for="titulo">Título</label>
        <input type="text" class="form-control" id="titulo" [(ngModel)]="categoria.titulo" name="titulo" placeholder="Digite o título">
      </div>
      <input type="submit" class="btn btn-primary" value="Cadastrar">
    </form>
  </div>
</app-layout>
```

Blocos principais do framework

DIRETIVAS

Responsável por modificar elementos DOM e/ou seu comportamento. Os Componentes também são diretivas, pois faz display de formulário, <div> etc.

```
<app-layout>
  ....<div class="col-md-6">
  ....  ....<form #categoriaForm="ngForm" (ngSubmit)="onSubmit()">
  ....  ....  ....<div class="form-group">
  ....  ....  ....  ....<label for="titulo">Título</label>
  ....  ....  ....  ....<input type="text" class="form-control" id="titulo" [(ngModel)]="categoria.titulo" name="titulo" placeholder="Digite o título">
  ....  ....  ....  ....</div>
  ....  ....  ....<input type="submit" class="btn btn-primary" value="Cadastrar">
  ....  ....</form>
  ....</div>
</app-layout>
```


Blocos principais do framework

ROTEAMENTO

Esqueça páginas, use o roteamento para mostrar tudo em uma só página. Com o Router podemos ir para as diversos componentes que são as telas da aplicação. Tudo em uma página.

```
const routes: Routes = [  
  {  
    path: '',  
    component: CategoriaComponent  
  }  
];
```

<router-outlet></router-outlet>



Criando projeto com Angular CLI

- Precisa de NodeJS e NPM instalados
- Criação do aplicativo
 - > npm install -g @angular/cli
 - > “ng new repositorio”
 - > “ng serve”
- Adicionando Bootstrap
 - > npm install bootstrap jquery
 - Adicionar no angular.json
 - "node_modules/jquery/dist/jquery.js"
 - "node_modules/bootstrap/dist/css/bootstrap.css"



Criando um componente

- Utilizando o angular CLI
- `> ng g c nome_componente`
- Estrutura de um componente



Criando estrutura de template

- Criar novo módulo
- Criar componente de cabeçalho
- Criar área de conteúdo no template
- Adicionando roteamento



Criando componente para categoria

- Criando componente para cadastro de categoria
- Criando componente para listagem de categoria
- Criando componente para agrupar os componentes acima



Criando camada de serviço

- Adicionando chamada HTTP para o backend
- Adicionando função no TS para receber resultado
- Atualizando componentes