

# WEBAAPP





# Bem vindo!

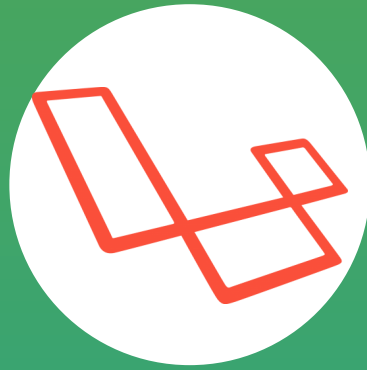




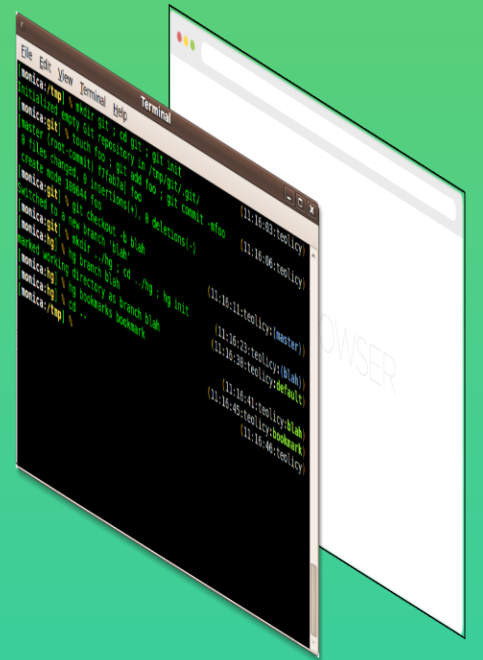
# Formato das Aulas?



BOAS PRÁTICAS



LARAVEL



45min

45min

# COMO FUNCIONA UM WEB APP?



NUVEM



NUVEM

AWS, Digital  
Ocean, Google  
App Engine

NUVEM

AWS, Digital  
Ocean, Google  
App Engine

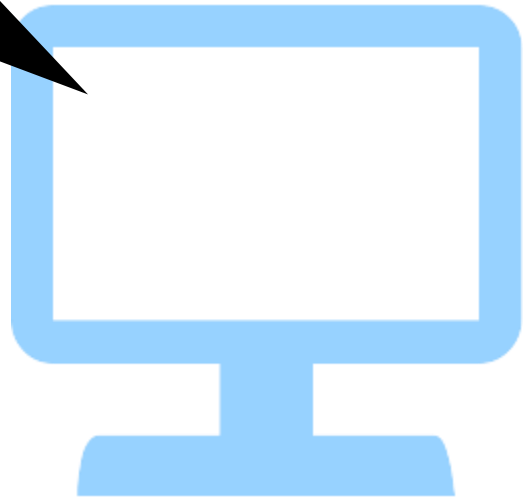


SERVIDORES

NUVEM

PHP, Ruby,  
ASP.NET,  
Python

AWS, Digital  
Ocean, Google  
App Engine



SERVIDORES

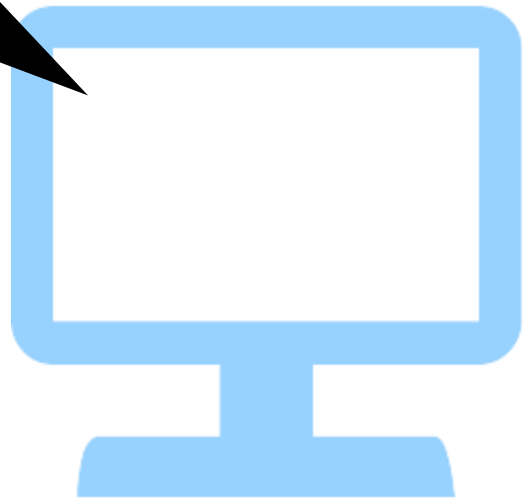


NUVEM

AWS, Digital  
Ocean, Google  
App Engine

PHP, Ruby,  
ASP.NET,  
Python

BANCO DE  
DADOS



SERVIDORES

NUVEM

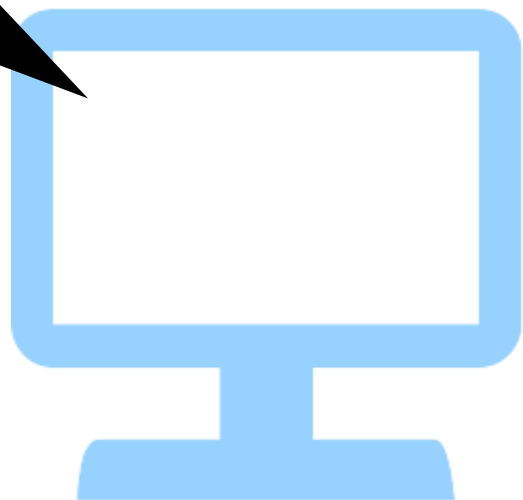
AWS, Digital  
Ocean, Google  
App Engine

BANCO DE  
DADOS

PHP, Ruby,  
ASP.NET,  
Python

MySQL/Maria,  
Mongo,  
Postgre, Neo4J

SERVIDORES





# Como eu escolho qual linguagem usar?

- Documentação
- Produtividade
- Curva de aprendizagem
- Suporte
- Atualizações e correções
- Pacotes de código já desenvolvidos
- Frameworks



```
/helloworld.php  
<html>  
  <head>  
    <title>PHP Test</title>  
  </head>  
  <body>  
    <?php  
      echo '<p>Hello World</p>';  
    ?>  
  </body>  
</html>
```



**4.500.000**  
de programadores



os maiores CMS do mundo usam

**open source**  
e lá vem a versão 7.0



# Facebook

Desenvolveu um compilador de código em PHP chamado HHVM na tentativa de aumentar a performance

## orientação a objeto

suporte a classes, traits, namespaces, e o que há de mais moderno!



HTML

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php
      echo '<p>Hello World</p>';
    ?>
  </body>
</html>
```

PHP



# Pera aí!

NUNCA misture linguagens SERVER SIDE com linguagens de MARCAÇÃO! ò\_\_Ó

- Deixa o código ilegível;
- Impossível de dar manutenção;
- Não permite trabalho em equipe;
- Não tem segurança alguma;
- Além de ficar impossível de adicionar futuras funcionalidades;



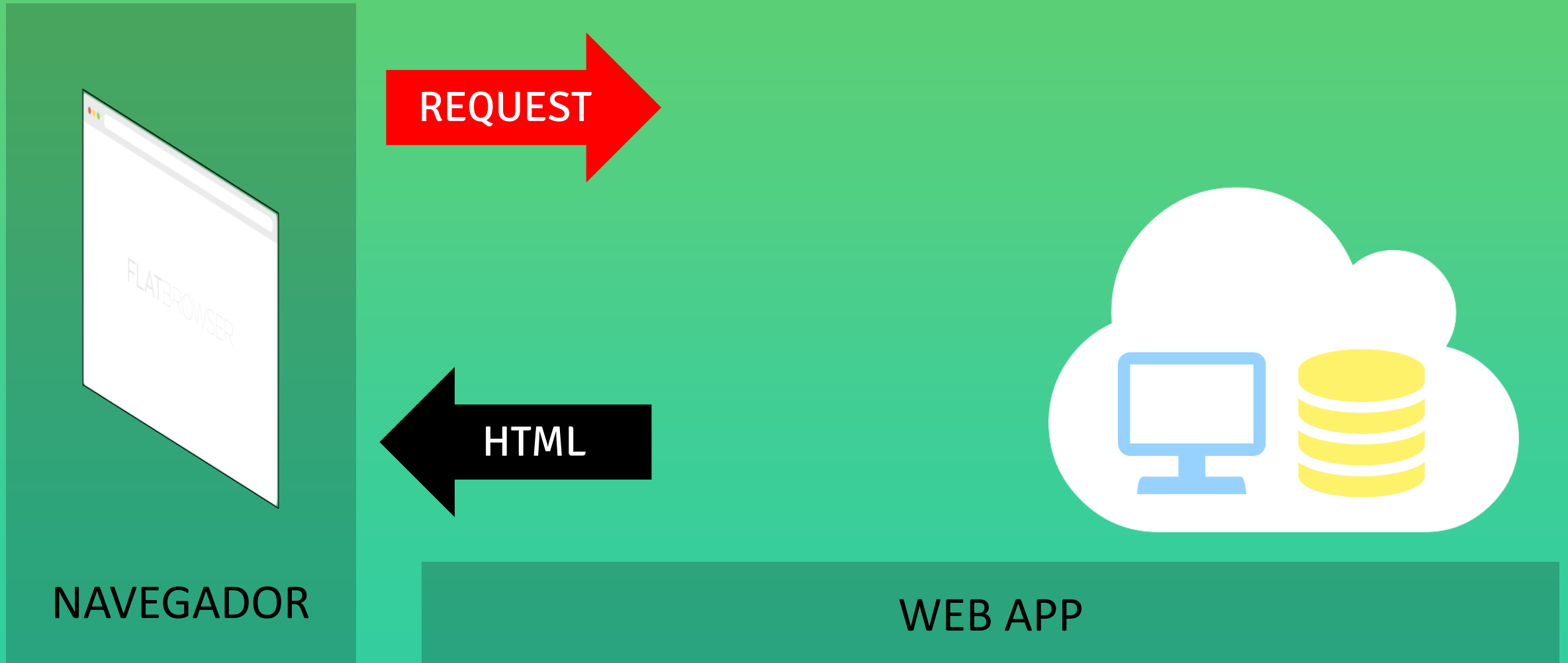
# MVC – Model View Controller



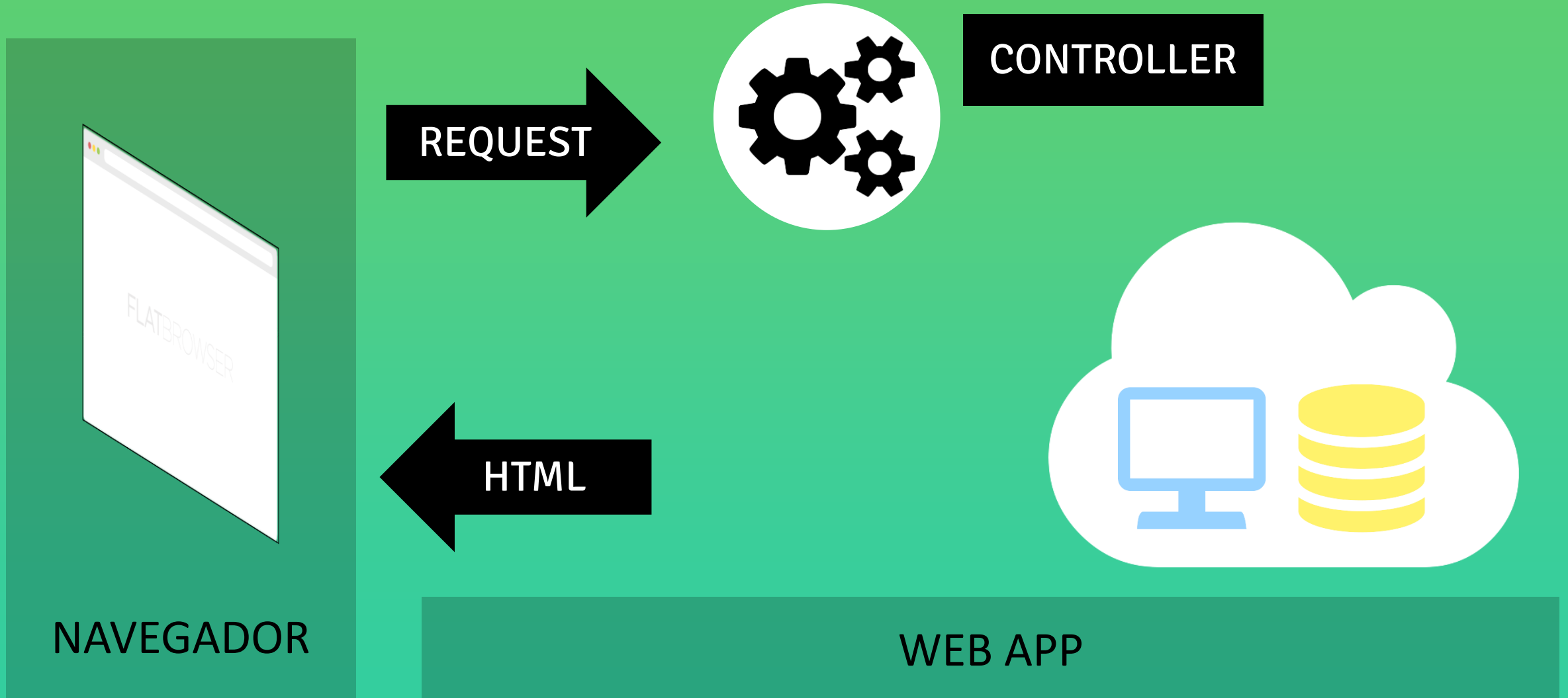
# MVC – Model View Controller



# MVC – Model View Controller



# MVC – Model View Controller





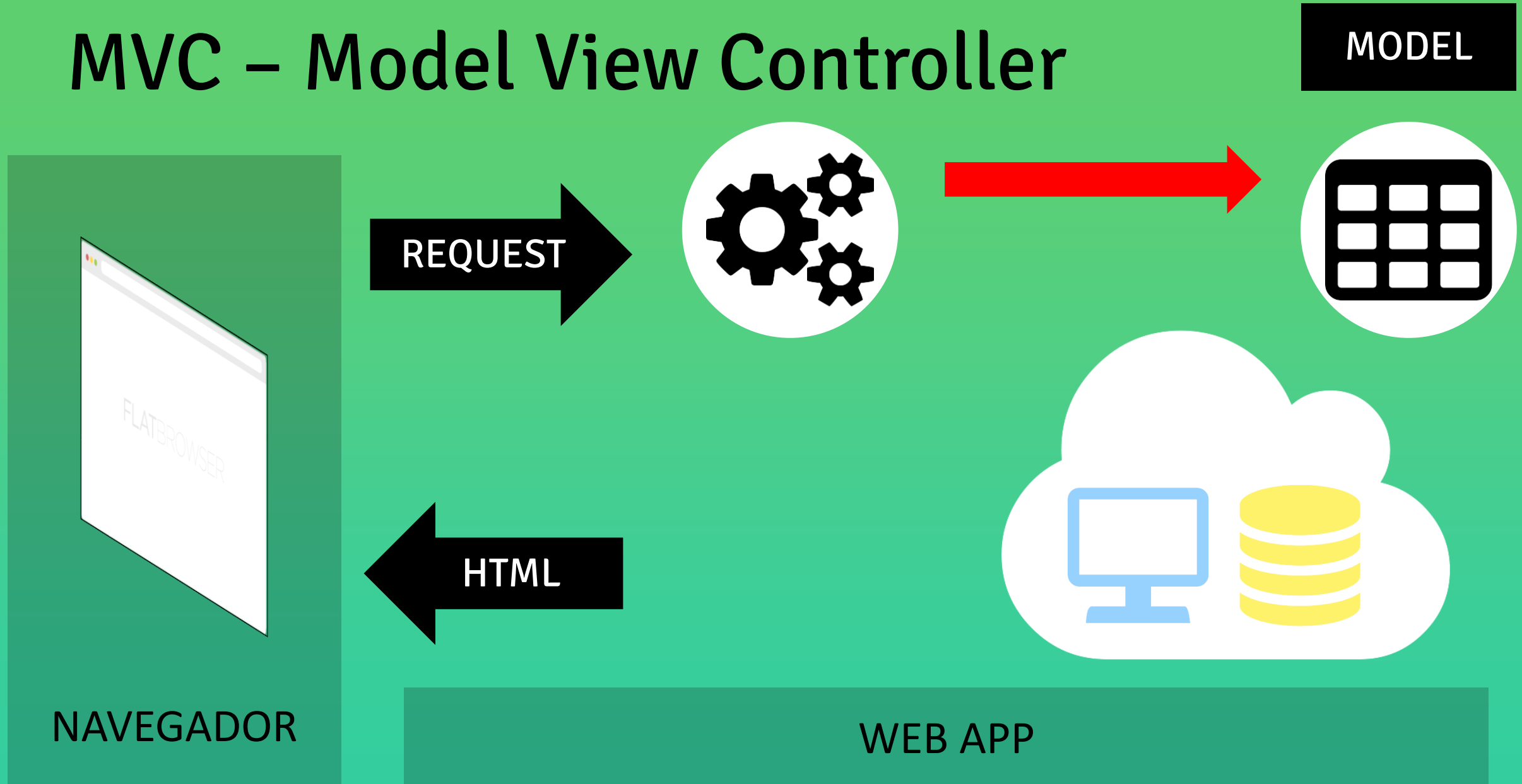
# Controllers

São classes destinadas a fazer o controle de acesso ao conteúdo e povoar as views com informação dinâmica!

Alguns pontos interessantes:

- **RESTful:** preocupação com uma padronização de manipulação dos recursos do seu sistema;
- **APIs:** interface de comunicação entre aplicações funcionam com requisições entre sistemas!

# MVC – Model View Controller





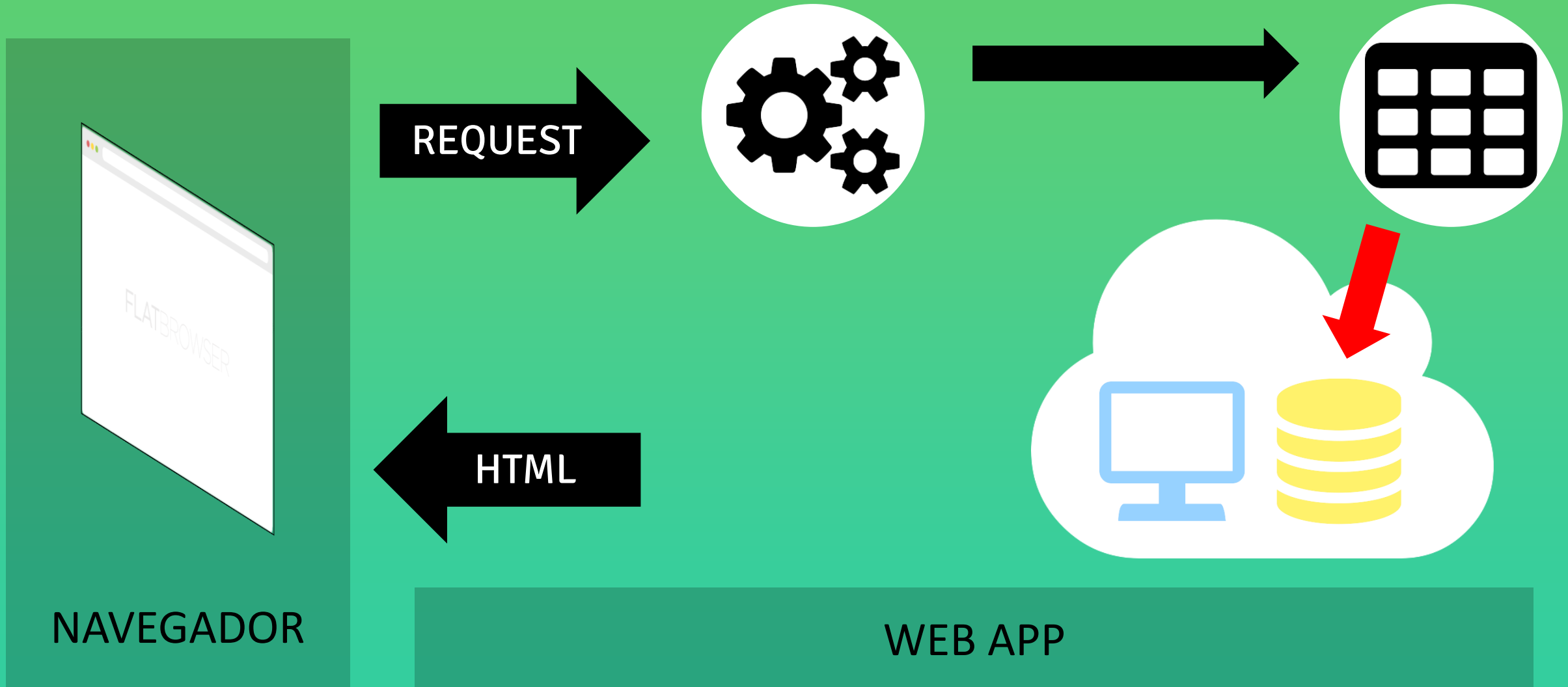
# Models

São classes destinadas a servir de interface entre sua aplicação e seu banco de dados! Geralmente seguem o diagrama de classes!

Alguns pontos interessantes:

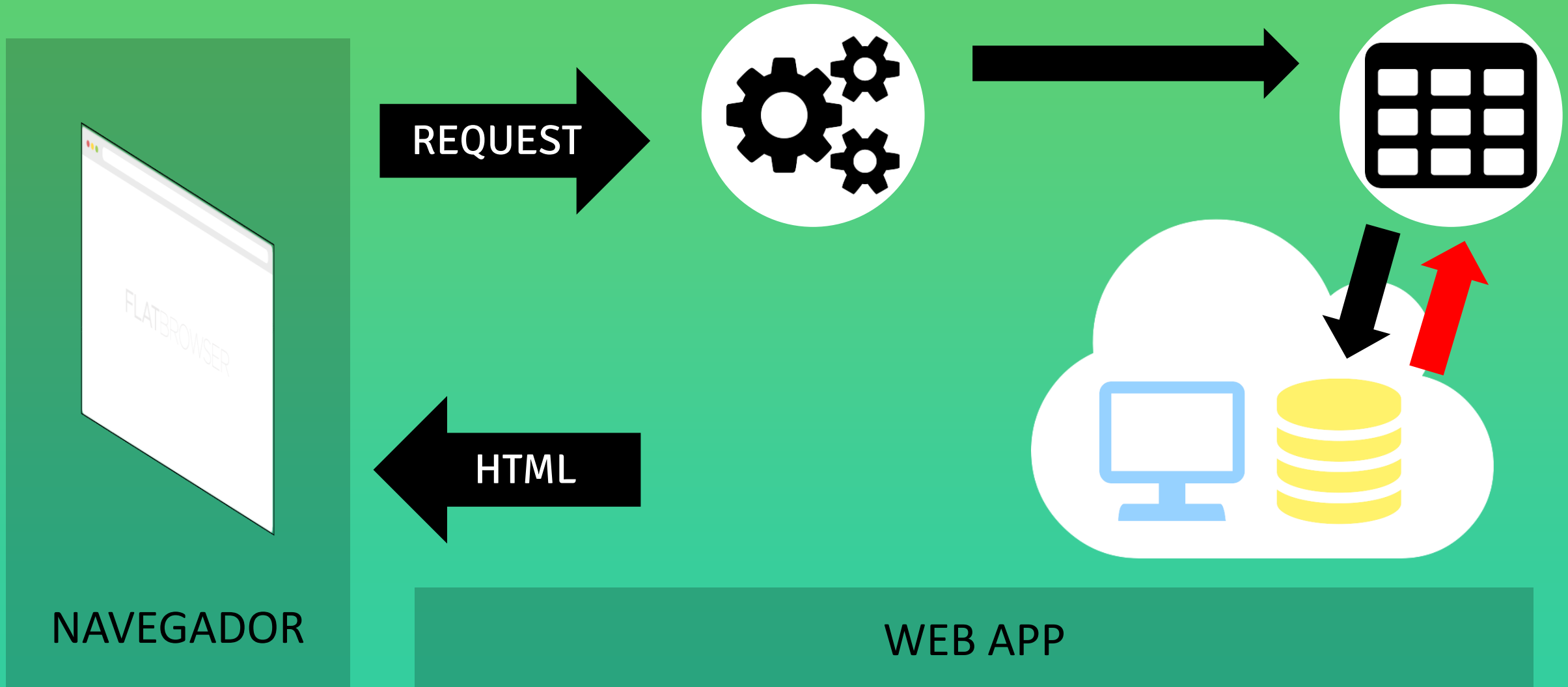
- **Frameworks** modernos mapeiam as interações entre os models assim como definimos chaves estrangeiras! Isto torna as consultas ao banco mto mais fáceis e seguras;
- Disparam as **operações SQL** de SELECT, UPDATE e DELETE.

# MVC – Model View Controller

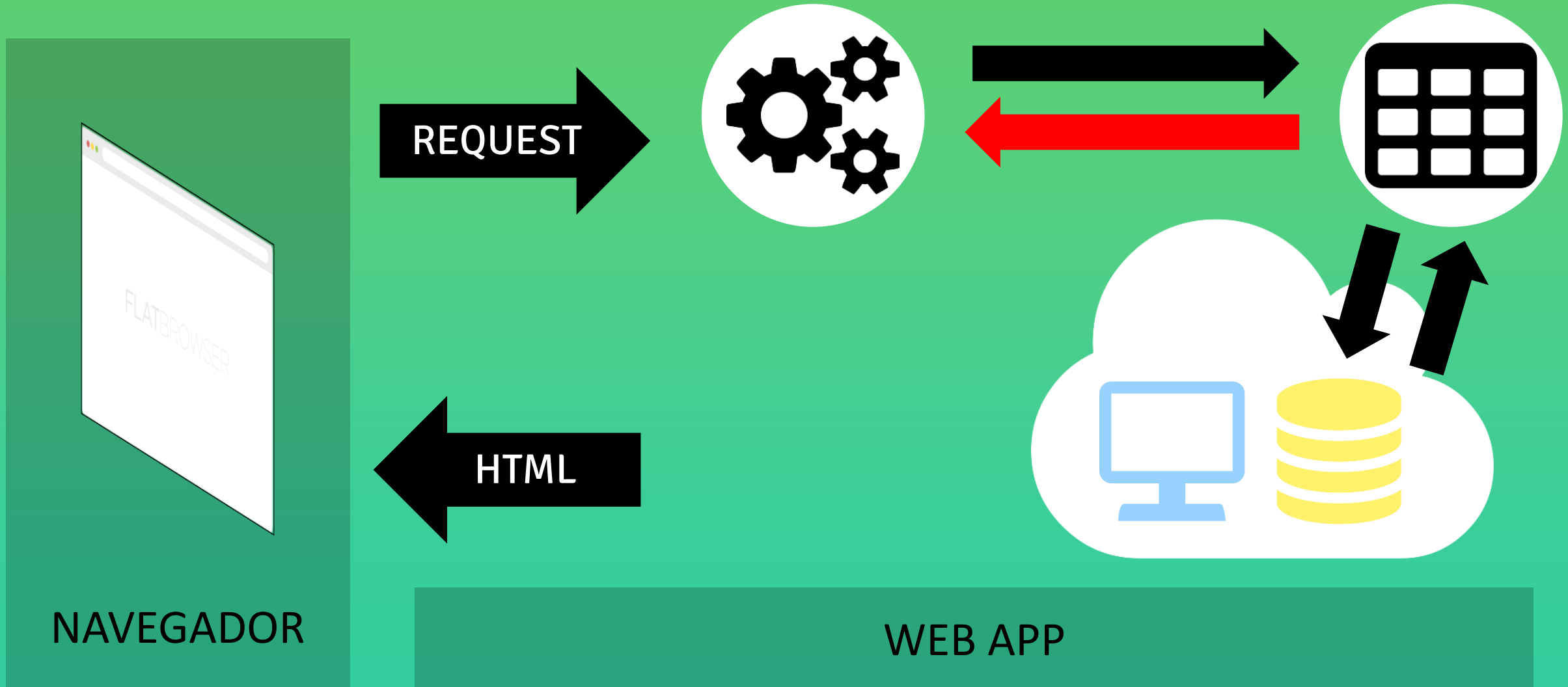




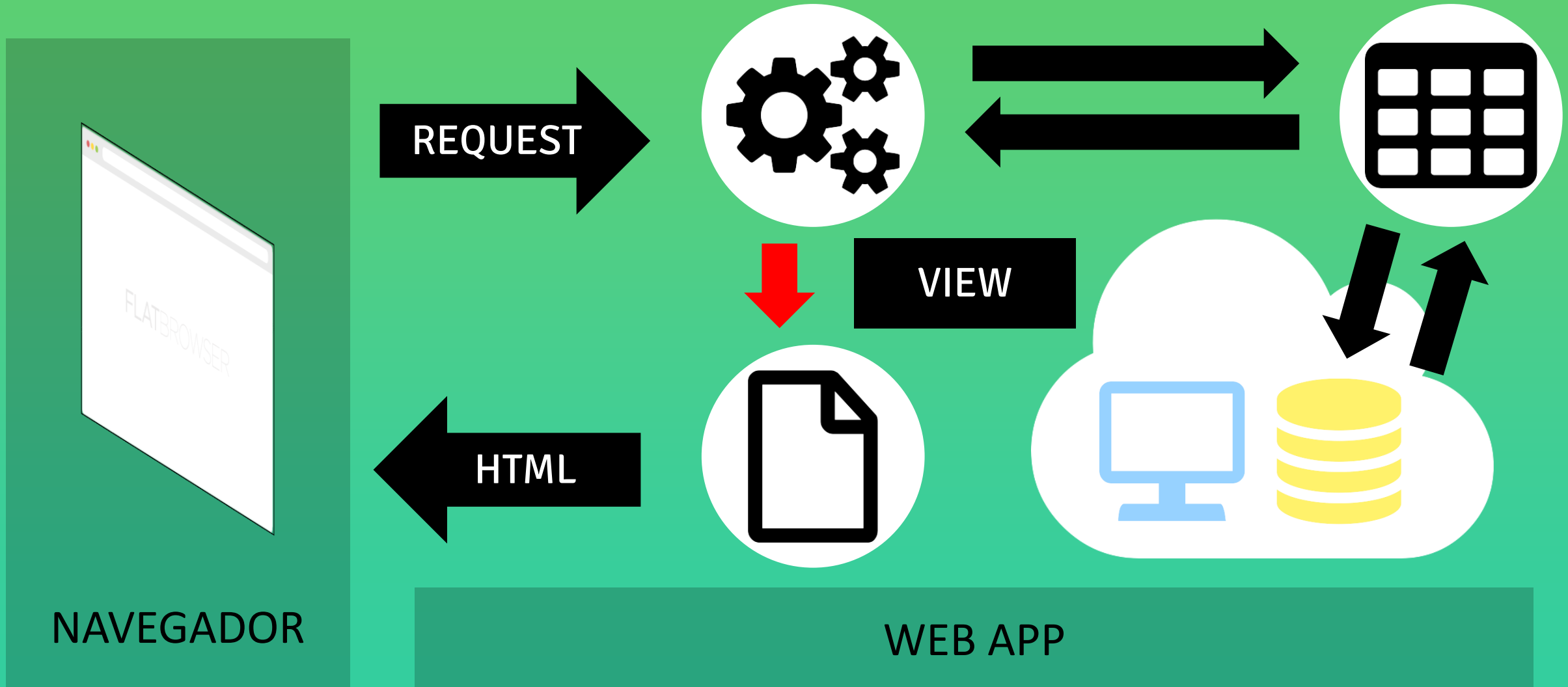
# MVC – Model View Controller



# MVC – Model View Controller



# MVC – Model View Controller





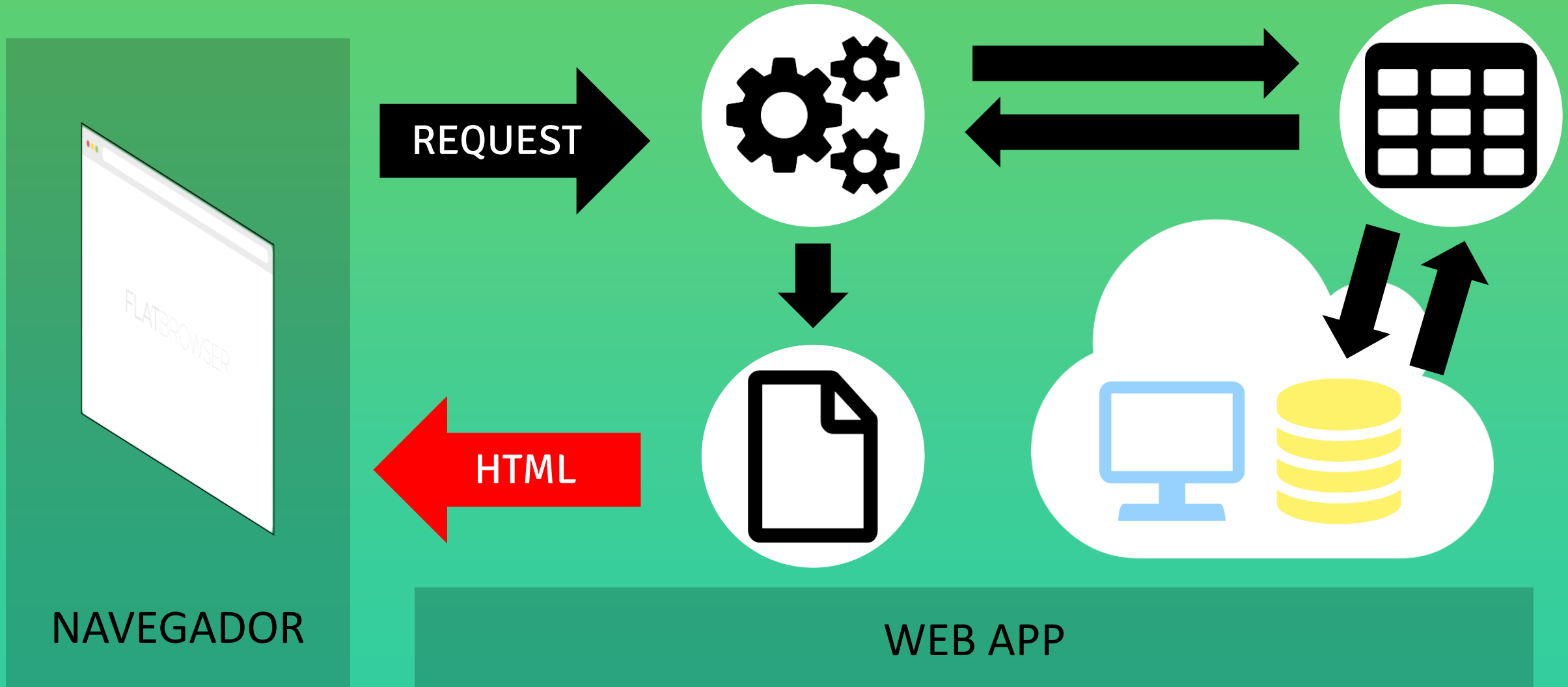
# Views

São objetos HTML já pré-definidos que podem ou não receber injeções de variáveis do Controller

Alguns pontos interessantes:

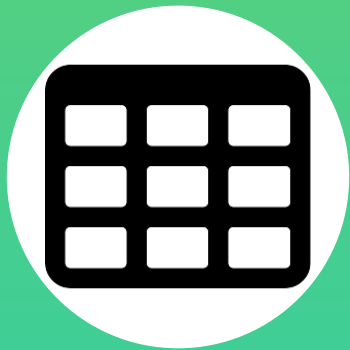
- **Templates:** você pode definir estruturas pré renderizadas do HTML e inserir suas informações que variam de usuário para usuário!

# MVC – Model View Controller



# Frameworks

São projetos genéricos de software que oferece uma arquitetura robusta de base para o desenvolvimento de qualquer aplicação!



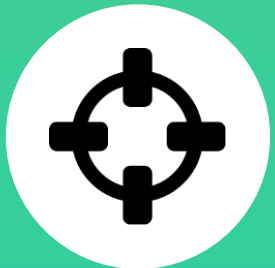
São sempre MVCs (ou MVVMs ou outros) !

# Vantagens do uso de Frameworks



Alta qualidade de documentação de todas as funcionalidades que o sistema pode desempenhar

Otimização de arquitetura e padrões de desenvolvimento



Noções avançadas de segurança web tanto quanto boas práticas de código

# Vantagens do uso de Frameworks



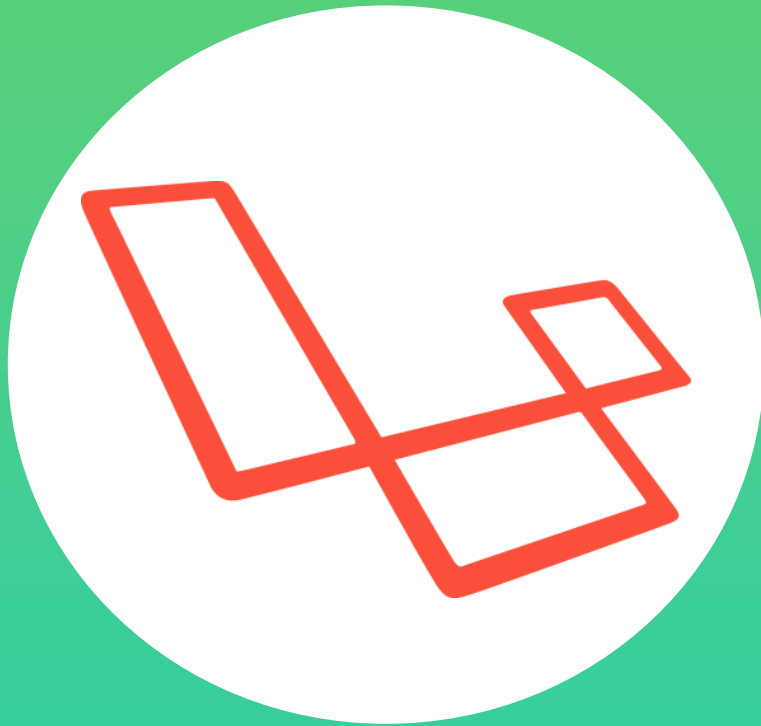
Compartilhamento de código já desenvolvido por outros usuários

Desenvolvimento ágil!





E o nosso framework é o...



LARAVEL

<http://laravel.com/docs/5.1>



**237.000**  
websites usando



segurança contra ataques mais  
frequentes de XSRF e SQL  
injection e várias interfaces para  
diferentes bancos de dados!



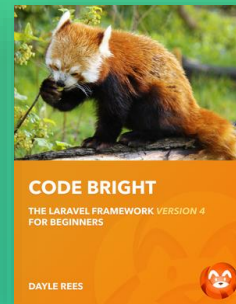
# Taylor Otwell

<https://github.com/taylorotwell>



# Dayle Rees

<https://github.com/daylerees>



# MEU PRÓPRIO WEB APP





# 0 ambiente de desenvolvimento!














<https://wtriple.com/wtnmp/>

Servidor

Banco de  
Dados






PHP


WT-NMP - Development Stack of Nginx Mysql Php


	Nginx: <input checked="" type="checkbox"/>	2 Running	   Kill
	MariaDB: <input checked="" type="checkbox"/>	Running	   Kill
	Php-Cgi: <input checked="" type="checkbox"/>	2 Running	  Kill

php-7.0.0rc7  2

All servers are running in background from c:\wt-nmp. You can close this window.

Projects:      

 [WT-NMP Home](#) ☒ Start with Windows **Kill ALL**

 [Managed Nginx Hosting](#) ☒ Daily MySql Backup **Close**

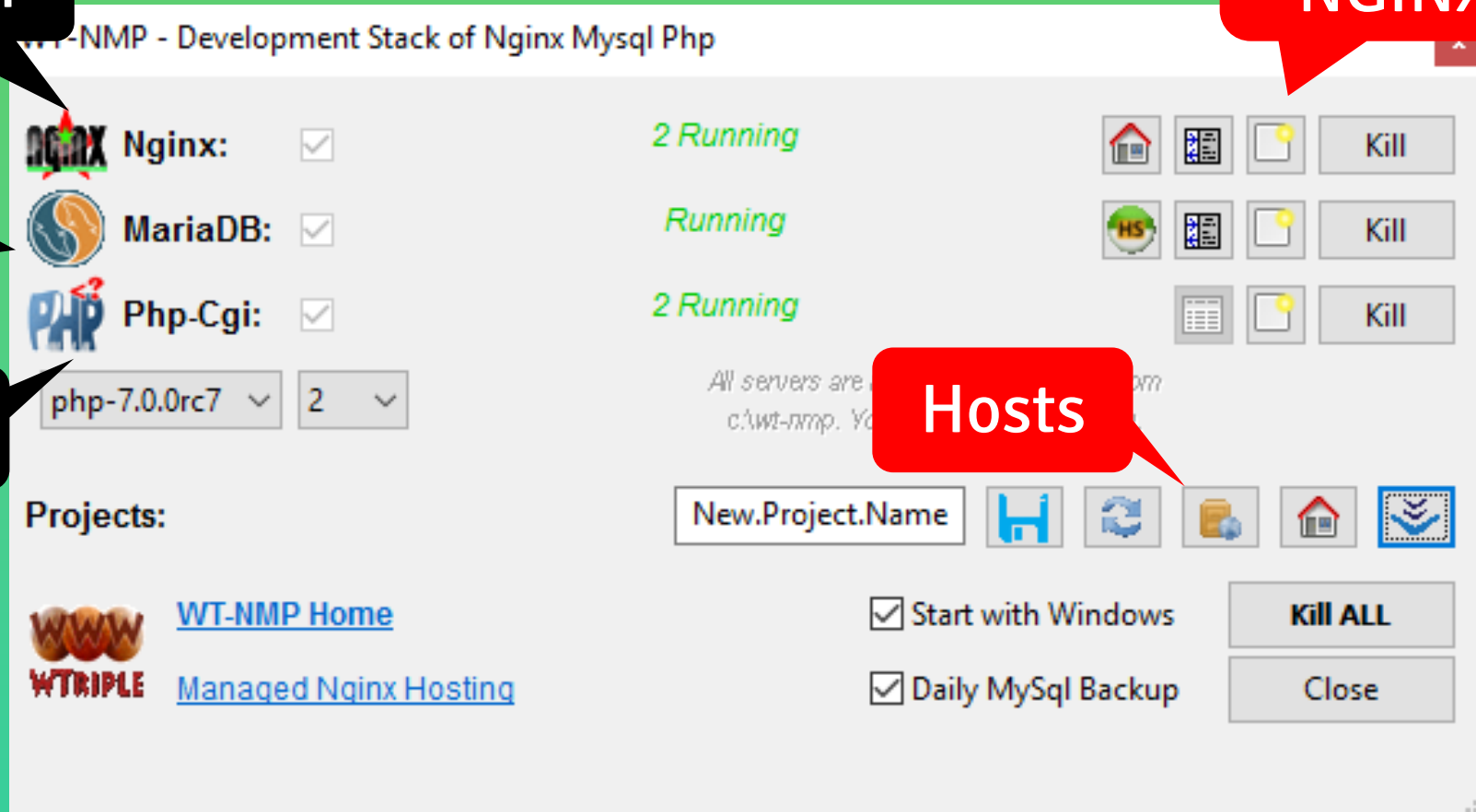
Servidor

Banco de  
Dados

PHP

Config do  
NGINX

Hosts







C:\WINDOWS\SYSTEM32\drivers\etc\hosts



192.168.10.10      webapp.dev

C:\WT-NMP\conf\nginx.conf



```
server {  
    listen 192.168.10.10 ;  
    server_name webdev.dev;  
    root C:\WT-NMP\WWW\webapp\public;  
    index index.php;  
    charset utf-8;  
  
    location / {  
        try_files $uri $uri/ /index.php?$args;  
    }  
    location ~ \.php$ {  
        try_files $uri /index.php =404;  
        fastcgi_split_path_info ^(.+\.php)(/.+)$;  
        fastcgi_pass   php_farm;  
        fastcgi_param  SCRIPT_FILENAME  
$document_root$fastcgi_script_name;  
        include nginx.fastcgi.conf;  
    }  
}
```



/etc/hosts



192.168.10.10	webapp.dev
---------------	------------

/etc/nginx/conf/nginx.conf



```
server {  
    listen 192.168.10.10 ;  
    server_name webdev.dev;  
    root C:\WT-NMP\WWW\webapp\public;  
    index index.php;  
    charset utf-8;  
  
    location / {  
        try_files $uri $uri/ /index.php?$args;  
    }  
    location ~ \.php$ {  
        try_files $uri /index.php =404;  
        fastcgi_split_path_info ^(.+\.php)(/.+)$;  
        fastcgi_pass php_farm;  
        fastcgi_param SCRIPT_FILENAME  
$document_root$fastcgi_script_name;  
        include nginx.fastcgi.conf;  
    }  
}
```



# Mas como eu faço usar o LARAVEL?

A resposta começa com a seguinte palavra: **GIT!**

O **GIT**, é um programa de controle de versão de software que:

- Permite o desenvolvimento em equipes
- Compartilhamento de código tanto para desenvolvimento
- quanto para adaptação (fork)



**Faça sua conta no Github!**

<https://github.com/>



# Ambientes com Proxy

Alguns ambientes como o de universidades possuem barreiras de seguranças a mais, como Firewall e Proxies. Digite esses comandos:

```
$ git config --global http.proxy http://proxy.ufu.br:3128  
$ git config --global https.proxy http://proxy.ufu.br:3128
```



```
$ git clone [repositório]
```

faz uma cópia na sua máquina do repositório alvo  
gerando um subdiretório com o seu conteúdo



```
$ git checkout [branch-alvo]
```

com este comando você seleciona qual branch você irá atualizar seu repositório, branches são caminhos que um código pode percorrer ao longo da vida do aplicativo



# Branches garantem seu sucesso!

O GIT permite você criar outros caminhos para seu código em desenvolvimento não influenciar no código anterior. Esses caminhos são chamados de **branches**, plural de branch.







## Repositório da Aula

<https://github.com/vitta-health/vitta-cursos/>

Cada turma é um **branch diferente** no **vitta-health/vitta-cursos**.

Branch
webapp-ufu-2015-2

Faça um **git clone** do nosso repositório e um **git checkout** para o branch descrito acima nesta pasta abaixo:

C:\WT-NMP\WWW



# Assine a lista de presença!



No GitHub, aperte o botão **Fork**

Faça git clone do SEU repositório **[você]/vitta-cursos**



Branch
webapp-ufu-2015-2



# Assine a lista de presença!



Crie um arquivo **presenca/aula\_01/seu\_nome.json** e neste arquivo siga o seguinte padrão:

presenca/aula\_01/seu\_nome.json



```
{  
  "nome": "Fulano de Tal",  
  "email": "meu-email@email.com",  
  "git": "meu-usuario-git"  
}
```



# Assine a lista de presença!



# Descrição dos comandos nos próximos slides!

```
$ git add --all
```

```
$ git commit -m "Meu-Nome está presente"
```

```
$ git push
```



# Assine a lista de presença!

No GitHub botão **New Pull Request**

**vitta-health/vitta-cursos** branch **webapp-ufu-2015-2**



**seu-usuario/vitta-cursos** branch **webapp-ufu-2015-2**

Botão **Create Pull Request**



# Criando um repositório do zero

No GitHub botão **New Repository**

Siga os próximos slides com os passos para  
inserir conteúdo no **seu** repositório!



```
$ git init
```

começa um novo repositório de controle de versão, você usará este código somente uma vez no começo do seu projeto!





```
$ git remote add origin https://github.com/[u]/[r].git
```

onde:

[u] = seu usuário

[r] = nome do seu repositório

neste passo informamos qual o servidor Git vai ser sincronizado, poderíamos por exemplo escolher outra fonte como Bitbucket ou uma instalação própria de servidor



```
$ git add [arquivo]
```

onde:

[arquivo] é o nome do arquivo que você quer adicionar

com este comando você adicionará todos os arquivos modificados para poder enviá-los ao servidor no local do parâmetro [arquivo] você pode passar o nome do arquivo ou uma determinada pasta



```
$ git commit -m "Meu primeiro Commit"
```

comando para empacotar os dados localmente para envio, é criado um hash para rastreá-lo e o parametro -m é definido para descrever as modificações que foram feitas



```
$ git push origin master
```

envia para mim tudo que eu empacotei para o branch master



```
$ git branch [novo-branch]
```

com este comando você cria um novo branch



# Não reinvente a roda!

Existem pacotes de funcionalidades específicas já desenvolvidas por outros usuários! Pare de perder tempo! Diferente de um framework, o pacote tende a ser específico, como por exemplo:

**“intervention/image”**: *“Image handling and manipulation library with support for **Laravel** integration”*

Com este pacote seu projeto ganha já funcionalidades de poder editar uma imagem usando php!



# Gerenciadores de Dependências

Se seu site tem funcionalidades mínimas necessárias para funcionar, seja ela um:

- gerador de QR Code;
- um gerador de PDF;
- uma conexão com alguma API específica como a AWS;

Você usará gerenciadores de dependências!



# Gerenciadores de Dependências

Existem vários deles no mercado, geralmente são focados nas linguagens de desenvolvimento:

- Javascript: Npm, Bower, Grunt
- Java: Maven
- PHP: Composer (<https://getcomposer.org/>)
- Python: Pip





`$ composer init`

este comando acionará o composer para criar um arquivo de dependências para o seu projeto assim como toda a descrição do mesmo

/composer.json



```
{
  "name": "vitta/webapp_like_a_boss",
  "description": "Este eh meu primeiro app",
  "require": {
    "php": ">=5.5.9",
    "laravel/framework": "5.1.*"
  },
  "require-dev": {
    "phpunit/phpunit": "~4.0",
  }
}
```



```
$ composer require "fzaninotto/faker"
```

este comando adicionará esta nova dependência no projeto editando o composer.json

/composer.json



```
{
  "name": "vitta/webapp_like_a_boss",
  "description": "Este eh meu primeiro app",
  "require": {
    "php": ">=5.5.9",
    "laravel/framework": "5.1.*"
  },
  "require-dev": {
    "fzaninotto/faker": "~1.4",
    "phpunit/phpunit": "~4.0",
  }
}
```

# PACKAGIST

Existe um site onde se indexam vários pacotes para Composer desenvolvidos em PHP. Dê uma olhada aqui:

<https://packagist.org/>



# Criando um projeto com Laravel

Existe alguns comandos especiais no composer como este:

```
$ composer create-project laravel/laravel --prefer-dist
```

Criará um sub-diretório com a estrutura do framework mais a descrição de toda a sua dependência. Então ao final de sua criação, acesse o diretório e rode o seguinte comando:

```
$ composer install
```

MÃOS À  
OBRA!





# Montando o seu projeto!

1. Crie seu próprio projeto usando o framework Laravel **usando o Composer**.
2. Defina **sobre o quê será o app** que você desenvolverá nas próximas semanas e escreva na descrição do composer.
3. Crie uma pasta **seu-projeto** dentro da subpasta **projetos** com o seguinte arquivo no próximo slide:



projetos/seu-projeto/webapp.json



```
{  
  "nome": "Fulano de Tal",  
  "projeto": "Nome do meu Projeto",  
  "git": "link-para-o-repositório"  
}
```



# Idéias para projetos

Locadoras



Lojas

