



# Deploy da Aplicação



# Deploy (implantação)

- Depois do programador desenvolver os requisitos da aplicação, é necessário um deploy para fazer os testes em um ambiente de produção
- Tal deploy para testes é chamado de staging, e é a etapa intermediária entre o desenvolvimento e o produto final
- Nessa seção, veremos como fazer uma implantação staging para o app de eventos que criamos ao longo do curso



# Deploy da API



# Deploy da API

- O primeiro passo será o deploy da API
- Para esse exemplo, vamos utilizar a plataforma Heroku, pois ela tem um *tier* gratuito adequado para o exemplo



# 1º passo: configurar o Django

- As configurações do Django atuais são para desenvolvimento (debug=true, por exemplo)
- Objetivo desse passo é criar um arquivo de configuração com configurações adequadas para o deploy



# Comando de gerenciamento *check*

- O django fornece um comando de gerenciamento para verificar se as configurações estão corretas para deploy de produção, basta rodar no terminal, dentro da pasta do projeto:

```
(venv) C:\Users\Lucas Amorim\PycharmProjects\django-course\eventsAPI>python manage.py check --deploy
System check identified some issues:

WARNINGS:
?: (security.W004) You have not set a value for the SECURE_HSTS_SECONDS setting. If your entire site is served
ty. Be sure to read the documentation first; enabling HSTS carelessly can cause serious, irreversible problems.
?: (security.W008) Your SECURE_SSL_REDIRECT setting is not set to True. Unless your site should be available ov
load balancer or reverse-proxy server to redirect all connections to HTTPS.
?: (security.W012) SESSION_COOKIE_SECURE is not set to True. Using a secure-only session cookie makes it more d
?: (security.W016) You have 'django.middleware.csrf.CsrfViewMiddleware' in your MIDDLEWARE, but you have not se
traffic sniffers to steal the CSRF token.
?: (security.W020) ALLOWED_HOSTS must not be empty in deployment.
```



# Comando *check*

- É retornada uma lista com todas as recomendações para o deploy. É importante ler cada uma e consultar a documentação para configurar os valores corretos.



# Carregando um arquivo de configuração

- Uma estrutura de projeto normalmente vai conter dois arquivos de configuração, um para desenvolvimento, o *settings.py* já presente no projeto, e um para deploy, por exemplo, *settings\_prod.py*
- Por padrão, o Django carrega o arquivo *settings.py* como arquivo de configuração, para carregar outro, é necessário configurar a variável de ambiente `DJANGO_SETTINGS_MODULE` com o arquivo desejado





# Arquivo de configuração

- Por exemplo, na API de eventos faríamos  
`DJANGO_SETTINGS_MODULE=eventsAPI.settings_prod`
- Lembrando que só devemos fazer isso no servidor em que será feito o deploy, para desenvolvimento devemos carregar o *settings.py* normalmente, o que é o padrão



# Verificação final

- Seguindo os passos do *check*, as seguintes condições já tem de estar atendidas no arquivo de configuração de produção:
  - Debug desativado
  - ALLOWED\_HOSTS com a lista de hosts a ser utilizada em deploy
  - Redirecionamento SSL ativo
- Essas são as principais, mas existem várias outras que serão mostradas pelo *check* e deverão ser tratadas



## 2º passo: preparar o ambiente


- Esse passo depende do serviço de hospedagem
- Pode ser necessário configurar python, WSGI, entre outros. Como isso depende da plataforma, é interessante olhar o manual da plataforma para configurar adequadamente o ambiente
- No nosso caso, utilizaremos o Heroku





# Heroku

- Primeiro é necessário criar um app no site:


App name

   
**events-manager-api** is available

Choose a region

 United States 

---

 [Add to pipeline...](#)

---

[Create app](#)



# Heroku

- Instalar o [Heroku CLI](#)
- Proporciona várias funções relativas ao heroku, utilizaremos para gerenciar o aplicativo e fazer o deploy



# Heroku: associar repositório com o app

- É necessário associar o repositório atual da API de eventos com o app que acabamos de criar no heroku, para isso, basta utilizar o comando:

```
heroku git:remote -a events-manager-api
```

- Sendo o último argumento o nome do app criado
- Note que para esse comando funcionar, seu projeto precisa já ser um repositório do git, caso ainda não seja, rode git init, git add . e faça commit das alterações antes do comando acima



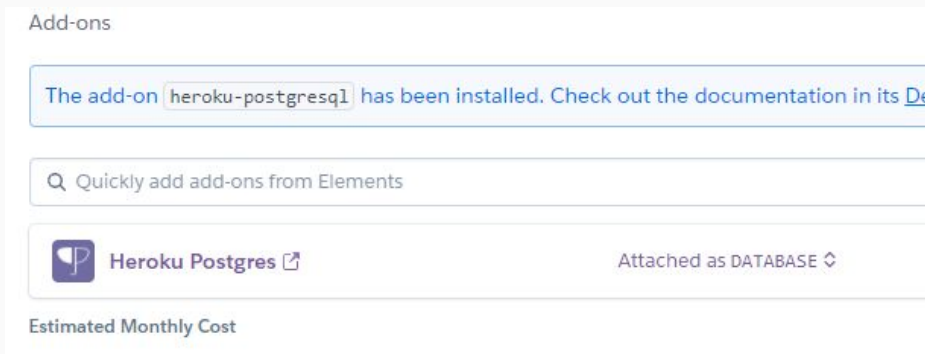
# Heroku: adicionar o banco de dados

- Um app do heroku não vem com banco de dados incluído, e toda vez que um app é reiniciado, seu conteúdo é perdido, apagando todo o banco SQLite (banco que estamos usando atualmente no Django)
- Vamos então utilizar o add-on gratuito Heroku Postgres, assim associando um banco de dados com nossa aplicação



# Heroku: adicionar o banco de dados

- Na página do app, acessar a aba *Resources*
- Na barra de busca abaixo de *Add-ons*, buscar por “Heroku Postgres”
- Clique sobre o add-on e autorize
- O add-on estará então associado:

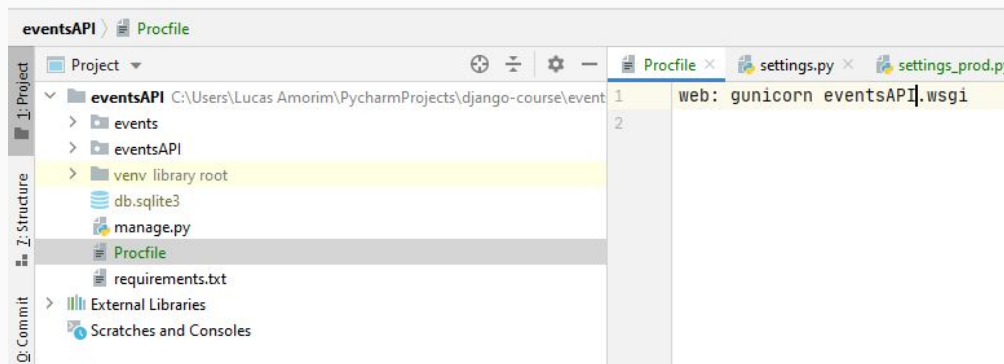






# Configurando o Django para o heroku

- O próximo passo é configurar o arquivo Procfile na raiz do projeto
  - O arquivo Procfile diz para o Heroku qual o comando que deve ser executado ao iniciar a aplicação
  - No caso do nosso Procfile, teremos apenas uma linha: `web: gunicorn eventsAPI.wsgi`
  - Ou seja, rodaremos o servidor gunicorn, passando o módulo wsgi da nossa aplicação, que foi criado ao iniciar o projeto
  - A estrutura final deve ficar assim:
  - Não esquecer de adicionar o gunicorn no requirements.txt





# Outras configurações

- Algumas outras configurações precisam ser feitas, como associar o banco de dados
- Felizmente, o heroku proporciona um pacote python que já faz o restante das configurações automaticamente, bastando instalá-lo via PIP:

```
pip install django-on-heroku
```

- Esse pacote deve também ser adicionado no requirements.txt!



# Outras configurações

- Para o django-on-heroku aplicar as configurações automáticas, deve ser feita a seguinte chamada no arquivo de produção (nosso caso *settings\_prod.py*, colocar na última linha):
  - `django_on_heroku.settings(locals())`
  - Essa chamada vai automaticamente associar banco de dados e arquivos estáticos



# Configurar variável de ambiente no heroku

- Agora precisamos configurar a variável de ambiente para alterar nosso módulo de configurações para o de produção (*settings\_prod.py*)
- Conforme dito antes, devemos alterar o valor da variável `DJANGO_SETTINGS_MODULE`
- Podemos fazer isso no heroku abrindo a aba “Settings” do app e adicionando uma nova “Config var”, conforme abaixo:

The screenshot shows the Heroku 'Config Vars' management interface. It has a title 'Config Vars' and a 'Hide Config Vars' button in the top right. Below the title, there are two rows of configuration variables, each with a 'KEY' field, a 'VALUE' field, and edit/delete icons. The first row shows 'DATABASE\_URL' with a PostgreSQL connection string. The second row shows 'DJANGO\_SETTINGS\_MODULE' with the value 'eventsAPI.settings\_prod'. At the bottom, there is a form to add a new variable with 'KEY' and 'VALUE' input fields and an 'Add' button.

KEY	VALUE
DATABASE_URL	postgres://bwlryrgpcfdlrc:1cdabb4bc0ede2e
DJANGO_SETTINGS_MODULE	eventsAPI.settings_prod

Buttons: Hide Config Vars, Add



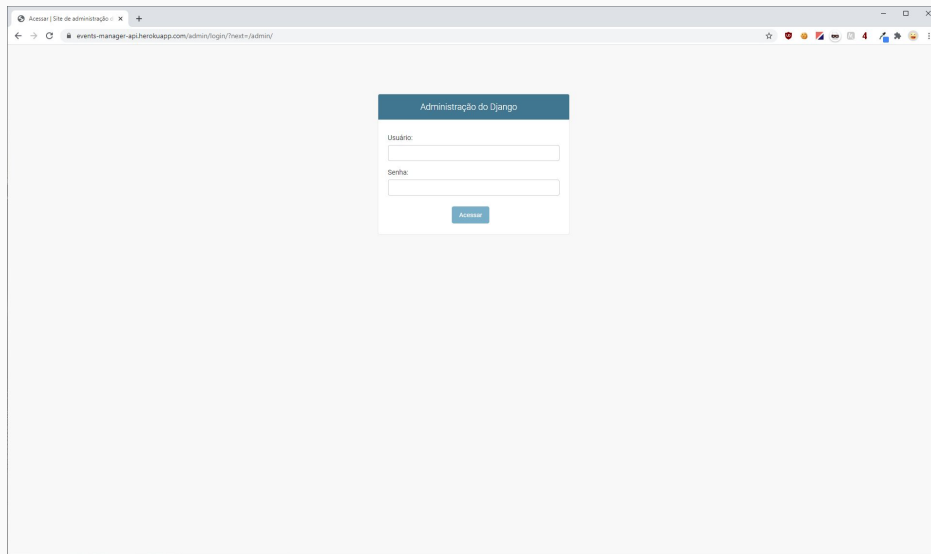
# Enviando para o heroku

- Com tudo configurado, basta fazer commit das alterações e enviar para o heroku utilizando o `git push heroku master`
- Será feito o deploy da aplicação



# Verificar funcionamento

- Se tudo deu certo, a aplicação estará rodando em modo produção na URL fornecida pelo Heroku:





# Próximos passos

- São necessários realizar alguns passos extras, assim como no desenvolvimento:
  - Migrar o banco de dados
  - Criar um superusuário para gerenciamento
- São os mesmos comandos do desenvolvimento, só que dessa vez precisam ser executados no heroku. Utilize a opção More -> Run Console na página do app para executar comandos.



# Deploy da aplicação Angular





# Passo 1: configurar e utilizar ambientes

- Utilizar os *environments* criados ao utilizar o comando `ng new`, preencher as informações de desenvolvimento no `environment.ts` e as de produção no `environment.prod.ts`
- Em nosso caso, mudará apenas a URL da API

```
environment.prod.ts
1 export const environment = {
2   production: true,
3   baseUrl: 'https://events-manager-api.herokuapp.com/'
4 };
```

```
environment.ts
1 // This file can be replaced during build by using the `fileReplacements` array.
2 // `ng build --prod` replaces `environment.ts` with `environment.prod.ts`.
3 // The list of file replacements can be found in `angular.json`.
4
5 export const environment = {
6   production: false,
7   baseUrl: 'http://localhost:8000/'
8 };
9
```

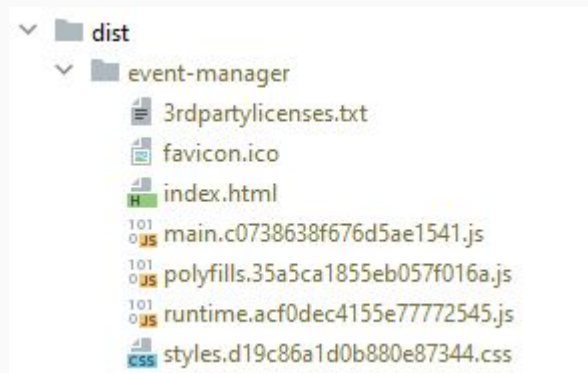


## Passo 2: criar um build de produção

- Utilizar o comando do Angular CLI “ng build --prod” para criar um bundle da aplicação já pronto para hospedagem
- Ao final do processo, será gerada uma pasta *dist* com a aplicação compilada

```
D:\asus-course\Angular-course\event-manager>ng build --prod
```

```
chunk {} runtime.acf0dec4155e77772545.js (runtime) 1.45 kB [entry] [rendered]
chunk {1} main.c0738638f676d5ae1541.js (main) 309 kB [initial] [rendered]
chunk {2} polyfills.35a5ca1855eb057f016a.js (polyfills) 36 kB [initial] [rendered]
chunk {3} styles.d19c86a1d0b880e87344.css (styles) 705 bytes [initial] [rendered]
Date: 2020-10-27T00:21:24.257Z - Hash: a643fd41e3c30107f95a - Time: 17734ms
```





## Passo 3: hospedar a aplicação

- Utilizar a plataforma/servidor de preferência para hospedar a aplicação
- Pode ser feito o uso do Apache, nginx, etc.
- No nosso exemplo vamos utilizar outro app do heroku para hospedar o front-end



# Heroku

- Assim como a API, criar um novo app:

App name

the-events-manager

the-events-manager is available

Choose a region

United States

Add this app to a pipeline

Pipelines form a continuous deployment pipeline

Choose a pipeline

Create app

```
D:\asus-course\Angular-course\event-manager>heroku git:remote -a the-events-manager
set git remote heroku to https://git.heroku.com/the-events-manager.git
```



# Criar um repositório

- Vamos utilizar o node.js para servir a aplicação, mas poderia ser utilizado qualquer outro, escolhemos pela simplicidade
- Criar um repositório git (git init) numa nova pasta
  - Dentro desta pasta, criar uma outra pasta para guardar a aplicação angular compilada (por exemplo, public)
  - Criar uma aplicação node básica para servir o conteúdo da pasta public, usando express
- A configuração do servidor node é fora do escopo do curso, mas disponibilizamos [um código de exemplo no GitHub](#)



# Enviando ao heroku

- Com o servidor node.js criado, basta então fazer *commit* das alterações e enviar para o heroku usando “git push heroku master”

```
C:\Users\Lucas Amorim\Desktop\deploy-events-manager>git push heroku master
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 8 threads
Compressing objects: 100% (27/27), done.
Writing objects: 100% (29/29), 109.97 KiB | 5.79 MiB/s, done.
Total 29 (delta 2), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: ----- Node.js app detected
remote:
remote: ----- Creating runtime environment
remote:
```

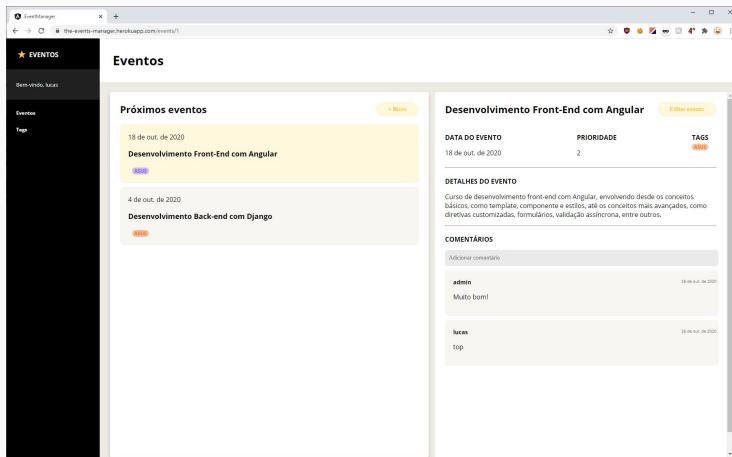
```
remote: ----- Build succeeded!
remote: ----- Discovering process types
remote:   Procfile declares types   -> (none)
remote:   Default types for buildpack -> web
remote:
remote: ----- Compressing...
remote:   Done: 23M
remote: ----- Launching...
remote:   Released v3
remote:   https://the-events-manager.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/the-events-manager.git
 * [new branch]      master -> master
```

```
C:\Users\Lucas Amorim\Desktop\deploy-events-manager>
```



# Testar

- Com ambas aplicações implantadas, verificar se está tudo funcionando
- Caso ocorra um erro, verifique os logs na plataforma/servidor (Heroku: View -> Logs)





# Observações:

- Código fonte: <https://github.com/GabriellBP/Angular-course>
- Exemplo deploy: <https://github.com/amorim/events-manager-deploy/>
- Dúvidas:
  - gabriel.pereira@edge.ufal.br
  - lucas.amorim@edge.ufal.br





Obrigado!