



# COMO FAZER UM APP EM 1 DIA!

POR ALLINE OLIVEIRA

# *Como Fazer um APP em 1 dia!*

AUTORA: ALLINE OLIVEIRA

DESIGNER: AMANDA COSTA

ISBN: 97810.956.4503.1

JULHO/2019

# SUMÁRIO

<i>Sobre o Livro</i>	4
<i>Depoimentos</i>	5
<i>Agradecimentos</i>	5
<b>INTRO</b>	6
<i>GitHub</i>	9
<i>Gitpod</i>	12
<i>Nome e Logo</i>	14
<i>Backend Python</i>	15
<i>Flask</i>	17
<i>Projeto Exemplo</i>	18
<i>Frontend</i>	19
<i>Commit e Push</i>	22
<i>Localhost</i>	24
<i>Heroku Cloud</i>	25
<i>Banco de Dados</i>	30
<i>MongoDB</i>	31
<i>MongoDB Atlas</i>	32
<i>PyMongo</i>	36
<i>API Endpoints</i>	39
<i>Data Explorer</i>	42
<b>EXTRAS</b>	43
<i>Plugins de autenticação</i>	44
<i>Login com o Facebook</i>	45
<i>Google Sign-in</i>	53
<i>Localização do usuário</i>	57
<i>Mapa</i>	59
<b>BÔNUS</b>	65
<i>Amazon AWS</i>	66
<i>Beanstalk</i>	70
<i>Segurança HTTPS</i>	73
<i>Route 53</i>	74
<i>Billing</i>	79
<i>Certificado SSL</i>	80
<i>Load Balancer</i>	84
<i>EC2</i>	89
<i>Mais Billing</i>	90
<i>App Basicão</i>	91
<i>Deploy</i>	93

## ***SOBRE O LIVRO***

"Como fazer um APP em 1 dia! mostra como criar um aplicativo desde o desenvolvimento até o deploy na nuvem em produção! Livro mara de 100 páginas que a Alline Oliveira escreveu." - Loiane Groner

"Quem nunca quis fazer um app em apenas 1 dia? O ebook "Como fazer um App em 1 dia!" promete essa maravilha." - BrazilJS

"Alline escreveu um livro incrível e eu recomendo demais que leiam e pratiquem todo o tutorial que ela ensina. No livro você encontra o link no Github com o código base (ai é só dar um fork e começar a estudar :D ), aprende sobre Gitpod, backend em python, um pouquinho de frontend, host em cloud usando o Heroku, o básico de Big Data usando MongoDB e mais um capítulo bônus para deixar seu app incrível. <3" - Jessica Felix

"Saiu a primeira versão de um livro muito massa para você ver o poder que é saber integrar as coisas em desenvolvimento. No livro Alline mostra como é fácil, usando as ferramentas certas, criar uma aplicação web e que hospedá-la não é um bicho de 7 cabeças." - Pokemao

"Alline é uma craque na relação da sensibilidade do humano com a tecnologia no ecossistema DIGITAL. Super indico a leitura e principalmente a prática do que ela ensina no livro. Alline é uma ativista para o sucesso do espírito empreendedor no Brasil. Gratidão querida." - Fernando Esselin

## **DEPOIMENTOS**

"Se não fosse a @allineo e esse tutorial incrível não saberia nem sobre a existência do flask ! Sou javeiro e resolvi me aventurar com python depois desse tuto !" - Luiz Henrique

"@allineo obrigada pelo tutorial! Ele é maravilhoso. Parabéns. Eu consegui fazer o app de acordo com as funcionalidades q estão no tutorial e quero adequar para minhas funcionalidades. " - Ellen Guimaraes

## **AGRADECIMENTOS**

Não tenho nenhuma condição de listar TANTAS PESSOAS que me ajudaram a chegar até aqui em todos esses meus 25 anos de carreira. Cada uma que foi e é especial com certeza vai se identificar aqui nesse momento porque sabe o quanto foi importante pra mim.

Gosto de frizar também que no fundo a pessoa que mais me ajudou em toda a minha trajetória sempre fui eu mesma, Sabemos o quanto precisamos ser fortes e resistentes para superarmos tantos estereótipos que existem na área de tecnologia, não é mesmo?

Porém existe sim um nome que foi totalmente imprescindível em toda minha vida e que eu não posso deixar de explicitar aqui com todas as letras nesses agradecimentos eternos: **Cleusa, minha mãe.**

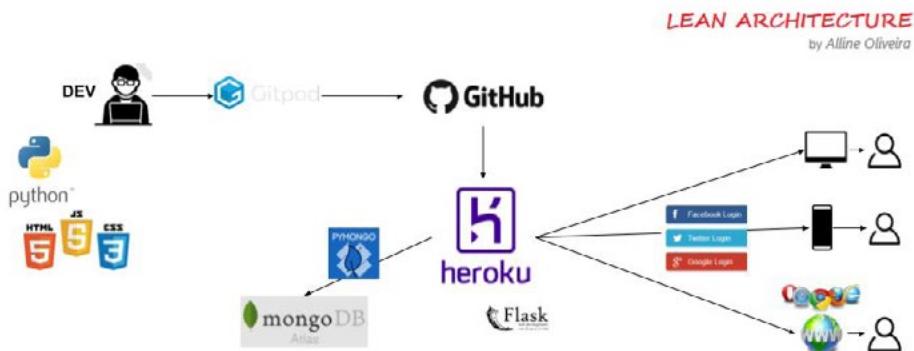
E por fim, o ser que é a base hoje da minha existência, e que agora me acompanha lado a lado nesse meu caminho: **meu filho, Luiz.**

# INTRO

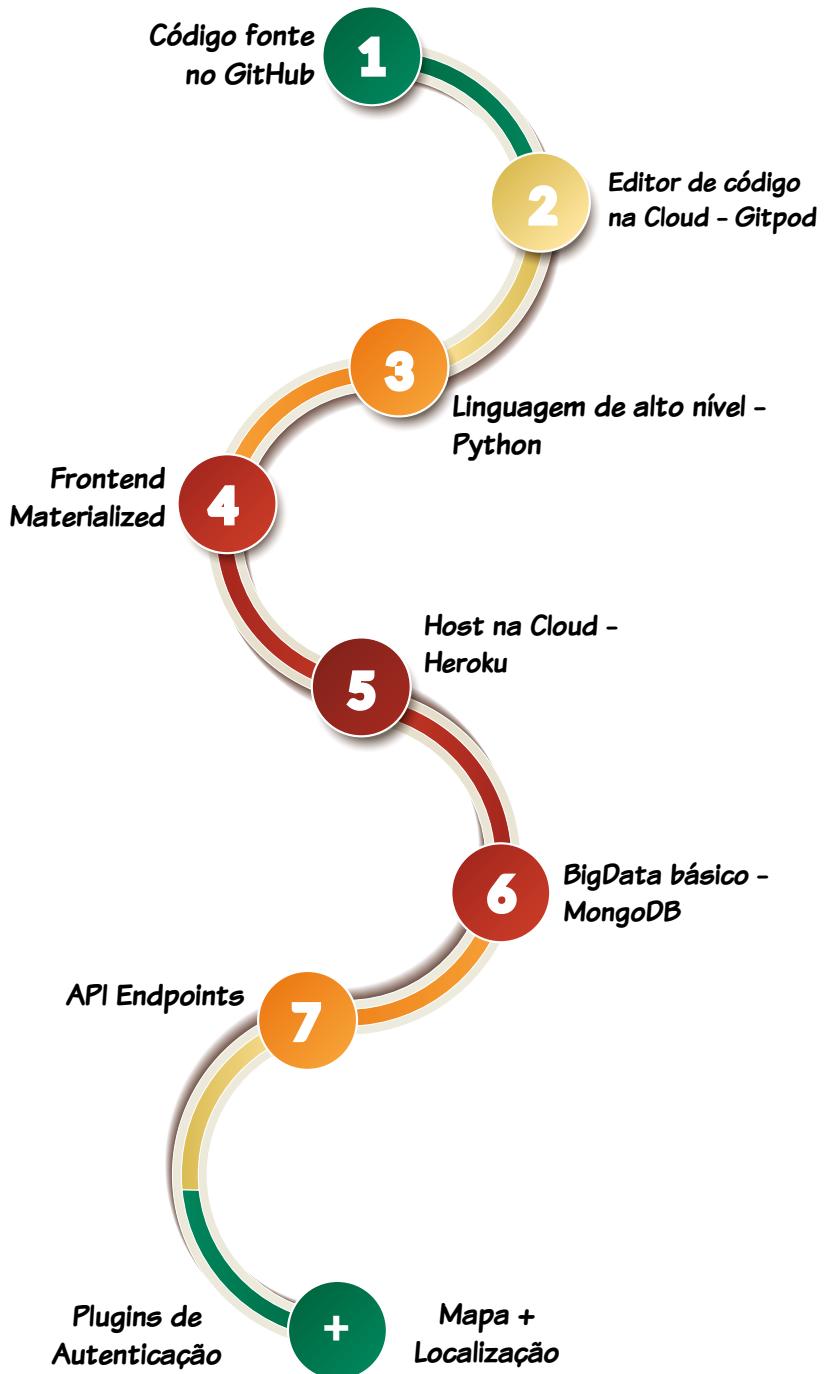
Quer tirar uma ideia da cabeça mas não quer gastar tempo nem dinheiro para testá-la?

Segue aqui uma receitinha básica com ferramentas práticas que vão lhe ajudar a publicar um applicativo mínimo que pode ser feito por uma única pessoa em algumas horas!

Primeiro, um spoiler da arquitetura que vamos usar. Não se preocupe se não estiver entendendo tudo logo de cara. Vamos guiar passo a passo cada um dos pontos apresentados.



A sequência a seguir contém os passos do processo, que depois serão detalhados separadamente já com o código pronto para copiar e colar.



## **\*\* Observações \*\***

Para esse tutorial você vai precisar de cadastro nas seguintes plataformas:

[GitHub.com](#), [Gitpod.io](#), [Heroku.com](#), [MongoDB.com](#)

E atenção! A arquitetura descrita aqui é **SHELL FREE**, ou seja, não utiliza linhas de comando!

Você não precisará nem sair do seu navegador :-D

## **\*\* DICA \*\***

Entre para o nosso grupo no TELEGRAM e tire suas dúvidas conosco!!

Busque pelo nome do grupo: APPemdia

<https://t.me/joinchat/JxXfahbkVFXrY3SOsnINvw>



Vamos começar acessando o site [GitHub.com](https://github.com).

GitHub é um famoso repositório online de código que serve para ajudar a nós, desenvolvedores, a trabalhar em equipe de forma segura e eficiente.

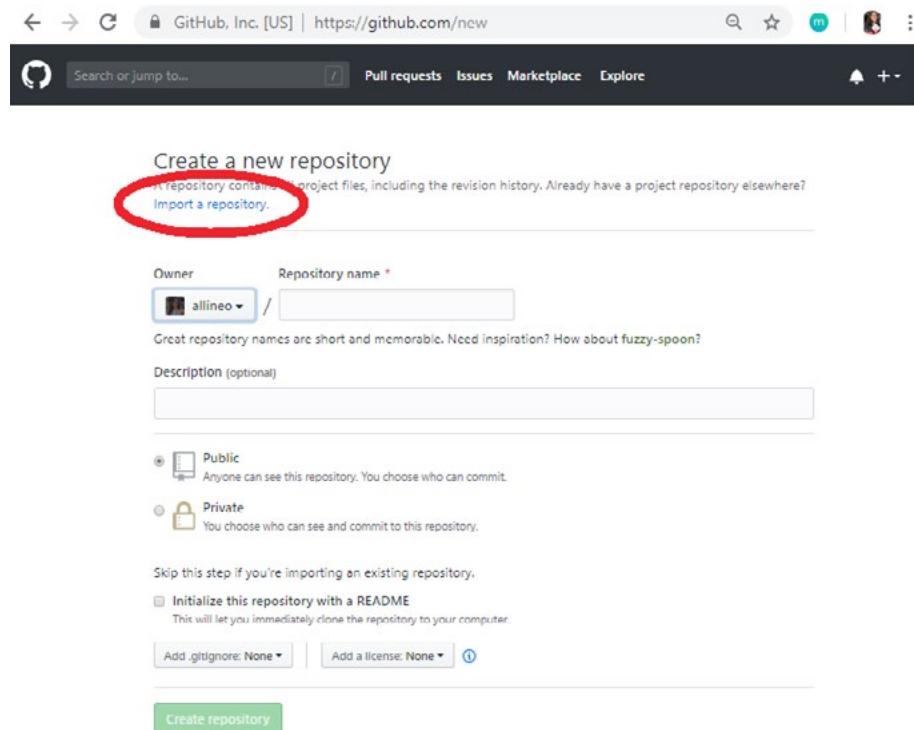
Existem várias ferramentas de armazenamento e versionamento de código também muito boas no mercado, como por exemplo [BitBucket.com](https://bitbucket.org). Escolhemos GitHub por ser a mais padronizada hoje em dia e também por ser a única que se integra facilmente de forma nativa com as outras ferramentas que iremos utilizar aqui nesse tutorial.

A screenshot of the GitHub homepage. At the top, there's a dark header bar with the GitHub logo, a search bar, and links for "Sign up" and a menu icon. Below the header, there's a navigation bar with icons for "Code review", "Project management", "Integrations", "Team management", "Social coding", "Documentation", and "Code hosting". The "Code hosting" icon is highlighted with a blue underline. The main content area features the GitHub logo and the tagline "All your code in one place". Below that, there's a paragraph about GitHub's capabilities and a small note: "\* As of April 2019". To the right, there's a large image of the GitHub cat logo.

Acessou o site? Então agora clique no botão "Sign up" e faça seu cadastro se ainda não o tiver.

Depois clique no botão  New para criar um novo repositório.

Para facilitar sua vida neste tutorial, clique no link "Import a repository" que aparecerá na próxima tela. Você poderá importar um outro repositório que já temos PRONTO com todos os arquivos que iremos utilizar.



The screenshot shows the GitHub interface for creating a new repository. At the top, there's a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below that is a search bar and a notifications section. The main area is titled "Create a new repository". It asks if the repository contains project files and provides a link to "Import a repository.". A red circle highlights this link. Below it, there are fields for "Owner" (set to "allineo") and "Repository name" (empty). There's also a "Description (optional)" field and a choice between "Public" and "Private" visibility. Underneath, there's a note about skipping the step if importing an existing repository, followed by checkboxes for initializing with a README and cloning to the computer. At the bottom is a green "Create repository" button.

Na página de importação, cole o link do repositório abaixo que irá ser utilizado no campo de "Your old repository", como na figura da tela a seguir:

<https://github.com/allineo/AppBasicaoBASE>

Import your project to GitHub  
Import all the files, including the revision history, from another version control system.

Your old repository's clone URL

Learn more about the types of [supported VCS](#).

Your new repository details

Owner  /

Privacy

**Public**  
Anyone can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

[Cancel](#) [Begin import](#)

No campo "Name", dê um nome para o repositório de código do projeto de sua ideia.

Fique a vontade para decidir se gostaria de criar o seu repositório como Público (seu código será aberto, acessível por qualquer pessoa) ou como Privado (acesso somente para quem você autorizar).

Então clique no botão "Begin import".

O seu novo repositório será criado já com a estrutura abaixo que é a que iremos utilizar aqui.

Não se preocupe que iremos lhe guiar em parte por parte.

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

allineo base

			Latest commit
py	base	base	30 minute
static	base	base	24 minute
templates	base	base	24 minute
Profile	base	base	30 minute
requirements.txt	base	base	30 minute
runtime.txt	base	base	30 minute

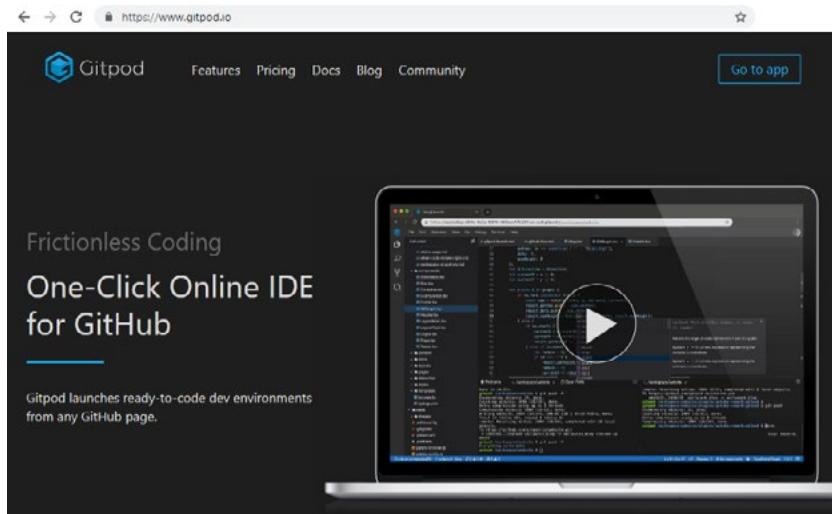
Help people interested in this repository understand your project by adding a README. [Add a README](#)



*Tem coisa mais fantástica do que editor de código na Cloud?!*

O Gitpod é um editor online que se comunica diretamente com seu repositório no GitHub, levando e trazendo as alterações desenvolvidas no código de seu aplicativo de forma bastante intuitiva e sem precisar utilizar as linhas de comando do Git!

Entre no site [Gitpod.io](https://www.gitpod.io), clique no botão "Go to app" e faça o cadastro com a sua conta do GitHub.



Dê todas as permissões de acesso ao Github, e aguarde aparecer o seu Dashboard.

Para clonar automaticamente um repositório do GitHub para o Gitpod, observe que é só usar o padrão de url abaixo:

`https://gitpod.io/#https://github.com/ (url_do_SEU_REPO_no_GitHub)`

Docs

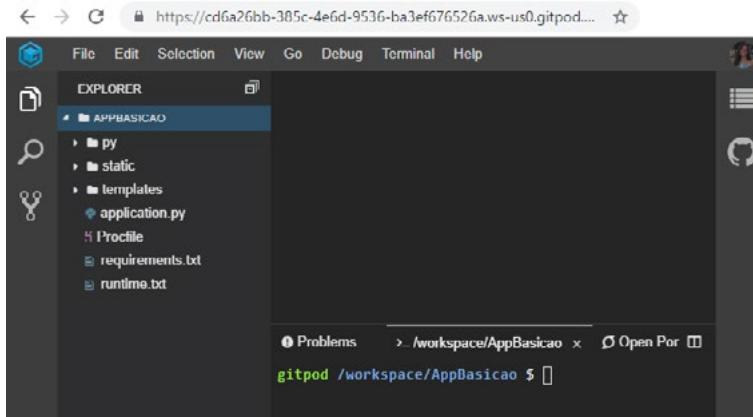
## Getting Started

Gitpod delivers the IDE part for the web-based development flows of common code hosting platforms. The following example shows how to use Gitpod with a GitHub repository.

1. Point your browser to some GitHub repository, pull request, or issue, e.g. <https://github.com/arunoda/learnnextjs-demo>
2. Either prefix the URL in the address bar of your browser with <https://gitpod.io/#https://github.com/arunoda/learnnextjs-demo> or push the **Gitpod** button if you have installed the Gitpod extension.

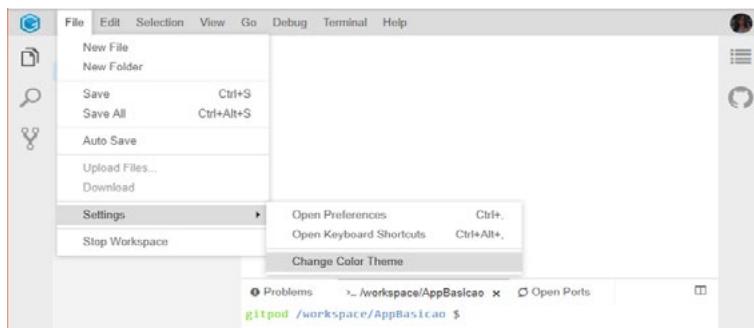
*Sua workspace será criada automaticamente no Gitpod, já com os arquivos de código trazidos do seu repositório no Github.*

*Essa ação é o equivalente ao comando "git clone" do Git.*



### \*\*Observação \*\*

*Se desejar alterar a cor de fundo do Gitpod, é só selecionar: "File" > "Settings" > "Change Color Theme".*



## Nome e Logo

A essa altura seu app já deve ter um nome, é sempre bom pensar em algo bem comercial.

Desenhe uma logo qualquer para começar, algo simples, porque esse app por enquanto é só para testar sua ideia.

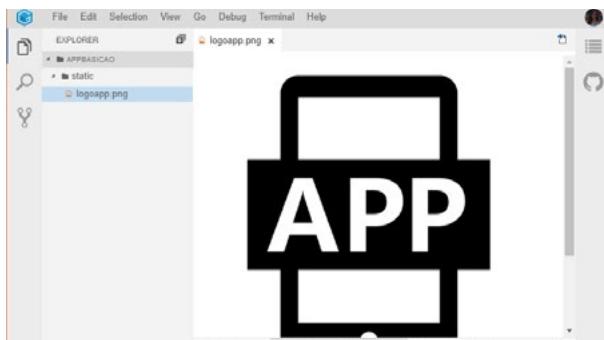
### \*\* DICA \*\*

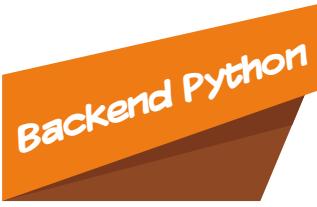
Uma boa ferramenta para desenhar uma logo rapidamente é a [Canva.com](https://www.canva.com)

### \*\* Observação \*\*

Faça o arquivo da logo em formato ".png", com fundo transparente e dimensões mínimas de 512x512 pixels, para já poder usá-lo posteriormente como ícone do app na autenticação com o plugin do Facebook.

Agora voltando ao Gitpod, vamos começar o aquecimento. Vá para dentro do diretório de nome "static" na raiz de seu projeto e importe o arquivo da sua logo e **renomeie** para "logoapp.png". Se precisar delete o arquivo existente.





## Backend Python

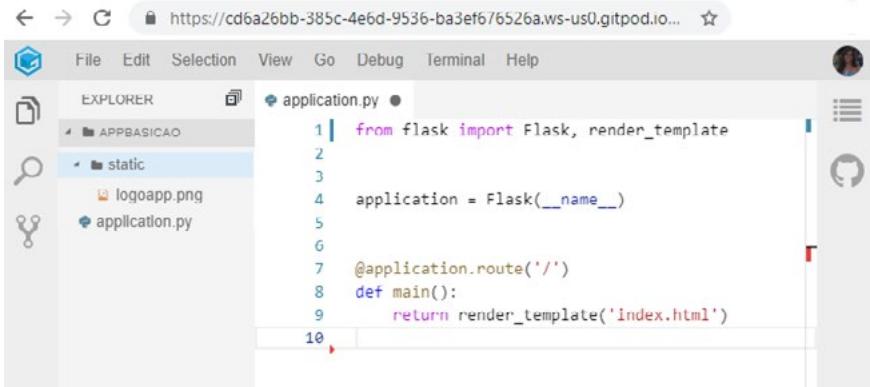
Python atualmente é uma das linguagens de programação mais utilizadas no mercado. Pode ser bastante simples para começar, por isso a escolhemos como o backend do nosso app.

Nesse nosso tutorial você só precisará de noções muito básicas de Python e de programação. Não precisará nem instalar nada, porque utilizaremos o nosso editor de código na Cloud.

Pronto! Chegou o momento de começarmos a codar!

Abra, na raíz de seu projeto do Gitpod, o primeiro arquivo de código chamado "application.py". Copie e cole para esse arquivo o conteúdo de código abaixo:

```
from flask import Flask, render_template  
  
application = Flask(__name__)  
  
@application.route('/')  
  
def main():  
    return render_template('index.html')
```



The screenshot shows a Gitpod interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- EXPLORER:** Shows a file tree with **APPBASICA0** expanded, containing **static** (with **logoapp.png**) and **application.py**.
- Code Editor:** Displays the **application.py** file content:

```
1 | from flask import Flask, render_template
2 |
3 |
4 | application = Flask(__name__)
5 |
6 |
7 | @application.route('/')
8 | def main():
9 |     return render_template('index.html')
10 |
```

## Atenção!

O Python necessita das identações digitadas corretamente para conseguir compilar seu código. Observe o espaçamento de **4 caracteres em branco (espaços)** para delimitar os blocos de comando, como na linha 9 acima.

# FLASK

Como nosso app será para dispositivos móveis e para a web (um **WebApp**), então precisamos de um ambiente com um servidor Web. Observe que para esse nosso tutorial escolhemos utilizar **Flask** por ser bastante leve e simples.

Flask é uma framework web para Python que inclui o servidor web que precisamos aqui.



## \*\* Observação \*\*

Caso deseje instalar o plugin do Flask no próprio Gitpod, digite o comando abaixo na janela bash de sua workspace:

```
$ pip3 install flask
```

A screenshot of a terminal window from the Gitpod workspace. The window shows the command '\$ pip3 install flask' being typed into the input field. The status bar at the bottom indicates the user is in workspace/AppBasicao, has 1 unread message, and a file count of 36.

# Projeto Exemplo

Como falamos anteriormente, já temos um repositório no GitHub com todos os arquivos de código do projeto que vamos utilizar aqui.

Acesse o link abaixo e observe os arquivos disponíveis no projeto de exemplo:

<https://github.com/allineo/AppBasicao>

Vamos lhe ajudar a compreender arquivo por arquivo.

The screenshot shows the GitHub repository page for 'allineo / AppBasicao'. At the top, there's a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below the navigation, there's a search bar and a 'Unwatch' button. The repository name 'allineo / AppBasicao' is displayed, along with a star count of 0 and a fork count of 4. A 'Code' tab is selected, showing 12 commits, 1 branch, 0 releases, and 1 contributor. The 'master' branch is selected. Below this, a list of files is shown, each with a preview icon, name, description, and timestamp. The files listed are: .py, static, templates, Procfile, application.py, requirements.txt, and runtime.txt. At the bottom of the list, there's a note about adding a README and a 'Add a README' button.

File	Description	Last Commit
.py	First commit	4 days ago
static	templates	24 days ago
templates	gitpod	14 days ago
Procfile	profile	last month
application.py	First commit	4 days ago
requirements.txt	first commit	2 months ago
runtime.txt	first commit	2 months ago

Esses arquivos já foram importados anteriormente para o repositório de seu projeto no capítulo do GitHub, lembra?

# Frontend

*Em desenvolvimento de software, definimos frontend como tudo aquilo que toca o usuário final, toda a parte da interface que possui interação com o exterior do código.*

*Nesse exemplo faremos um frontend bem mínimo só para demonstrar a ideia de um desenvolvimento ágil e simples, sem design e sem nenhum objetivo maior.*

*É muito importante que neste momento você consiga simplificar sua ideia ao máximo. Qual a funcionalidade mais crucial de toda a sua ideia? Qual a única página de todo o seu app que não pode ser retirada? Sempre tem como reduzir sua ideia a um Mínimo Produto Viável.*

*Em um próximo momento você poderá continuar a desenvolver funcionalidades próprias seguindo os princípios da simplicidade e do minimalismo, necessários para confrontar suas ideias com o comportamento de usuários reais da forma mais barata e eficiente possível.*

*Para uma tela de início do app, com um menu lateral básico, do tipo hambúrguer, criamos o código HTML a seguir. Comece observando esse código para se ambientar com a arquitetura.*

*Na pasta "templates", do seu projeto no Gitpod, abra o arquivo "index.html".*

```
File Edit Selection View Go Debug Terminal Help
EXPLORER APPBASICO index.html
  static
    logoapp.png
  templates
    index.html
  application.py
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
4     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1.0" />
5     <meta name="google-signin-client_id" content="XXXXXXXXXXXXXXXXXXXX.apps.googleusercontent.com">
6     <title> APP BÁSICO </title>
7
8     <link href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css" rel="stylesheet">
9     <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
10    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
11
12    <script src="static/js/facebook.js"></script>
13    <script async defer crossorigin="anonymous" src="https://connect.facebook.net/pt_BR/sdk.js"></script>
14    <script src="static/js/googleLogin.js"></script>
15    <script src="https://apis.google.com/js/platform.js?onLoad=renderButton" async defer></script>
16    <script src="https://maps.googleapis.com/maps/api/js?key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" async defer></script>
17
18    <script src="static/js/application.js"></script>
19    <script src="static/js/form.js"></script>
20    <script src="static/js/map.js"></script>
21    <style>
22      @import url("static/css/main.css");
23    </style>
24
25  </head>
26
27  <body>
28    <div class="background">
29      <section id="header">
30        <nav>
31          <div class="nav-wrapper">
32            <a href="#" data-target="#mobile-demu" class="sidenav-trigger">
33              <div class="NavbarToggle">
34                <div class="ToggleOne"></div>
35                <div class="ToggleTwo"></div>
36                <div class="ToggleThree"></div>
37              </div>
38            </a>
39          </div>
40        </nav>
41      </section>
42    </div>
43
44    <div class="content">
45      <div>
46        <h1>APP BÁSICO</h1>
47        <p>Este é um projeto de exemplo para demonstrar como construir uma aplicação web com Python e Flask, integrando banco de dados MySQL, formulários, autenticação com Google e Facebook, e gerenciamento de usuários. Utilizamos o framework Materialize para o design responsivo e o SQLAlchemy para a interação com o banco de dados. O projeto também inclui uma interface de usuário com navegação móvel e uma interface administrativa para gerenciar os usuários criados.</p>
48        <hr/>
49        <h2>Autenticação:</h2>
50        <ul>
51          <li>Autenticação com Google e Facebook.</li>
52          <li>Autenticação com usuário e senha (usuários criados no banco de dados).</li>
53        </ul>
54        <hr/>
55        <h2>Gerenciamento de Usuários:</h2>
56        <ul>
57          <li>Criação de novos usuários.</li>
58          <li>Edição e exclusão de usuários existentes.</li>
59        </ul>
60        <hr/>
61        <h2>Integração com Banco de Dados:</h2>
62        <ul>
63          <li>Consulta de usuários no banco de dados MySQL.</li>
64          <li>Inserção de novos usuários no banco de dados MySQL.</li>
65        </ul>
66      </div>
67    </div>
68
69    <div class="Footer">
70      <div>
71        <small>Desenvolvido por Allineo - 2023. Todos os direitos reservados.</small>
72      </div>
73    </div>
74
75  </body>
76
77</html>
```

*Observação. Este arquivo também está disponível no projeto exemplo que disponibilizamos no GitHub pelo link abaixo:*

<https://github.com/allineo/AppBasicao/blob/master/templates/index.html>

*Para esse layout de frontend, com menu retrátil, que funciona responsivamente tanto para computadores padrão como para dispositivos móveis e pequenos, utilizamos uma framework de frontend muito disseminada chamada [Materialize \(materializecss.com\)](#).*

```
8   <link href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css" rel="stylesheet">
9   <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
```

*Você vai precisar olhar também todos os outros arquivos CSS e JS que estão no diretório "static" que foram copiados para o seu projeto.*

*Observação: Esses arquivos também estão no projeto AppBasicao de exemplo do GitHub, que podem ser acessados pelo link:*

<https://github.com/allineo/AppBasicao/tree/master/static>



## \*\* Observação \*\*

Neste exemplo, os códigos de frontend e backend da aplicação encontram-se acoplados num mesmo projeto. Caso seja da preferência da sua equipe de trabalho, construa-os em projetos separados.

E se, ao colocar o frontend e o backend em servidores diferentes, ocorrer um erro de *Cross-Origin*, use a biblioteca *CORS* do Flask, como no exemplo abaixo.

```
application.py
1 from flask import Flask
2 from flask_cors import CORS
3
4
5 app = Flask(__name__)
6 CORS(app)
```

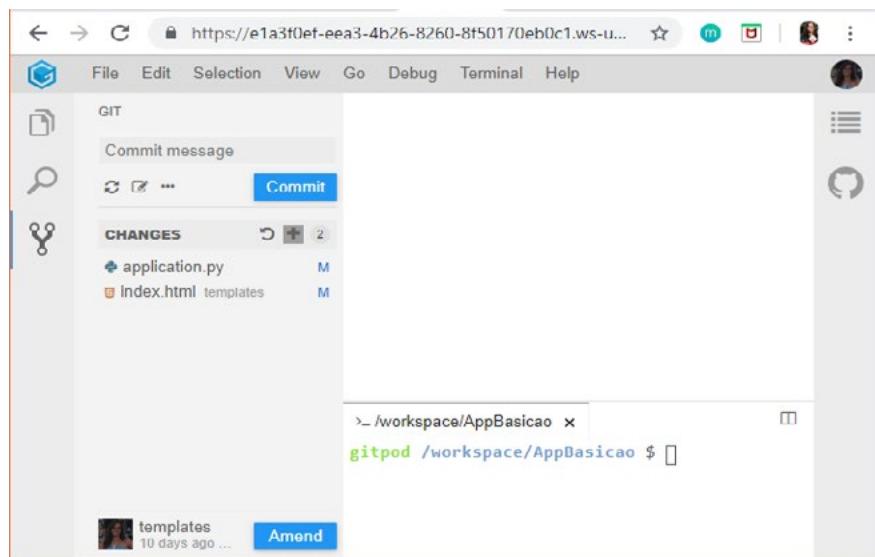
E se for usar *CORS*, não se esqueça de importar a biblioteca no Python adicionando a linha "flask\_cors" no arquivo "requirements.txt".

```
requirements.txt
AppBasicao > requirements.txt
1 flask
2 flask_cors
3 gunicorn
4 pymongo[tls,srv]
5 requests
```

## Commit e Push

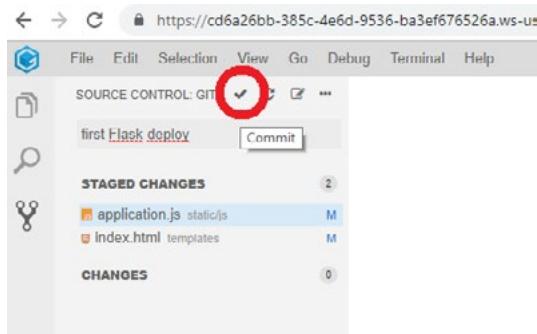
A qualquer momento que desejar, você pode subir para o repositório do GitHub, as alterações realizadas em seu projeto no Gitpod, ou seja, realizar o vulgo "Commit". É importante fazer commit com frequência para deixar seu código fonte resguardado e acessível às outras pessoas do seu time de desenvolvimento.

Clique no último ícone ("Git") do canto esquerdo superior do seu projeto no Gitpod, e você verá a lista dos arquivos ainda não committedos de seu projeto.

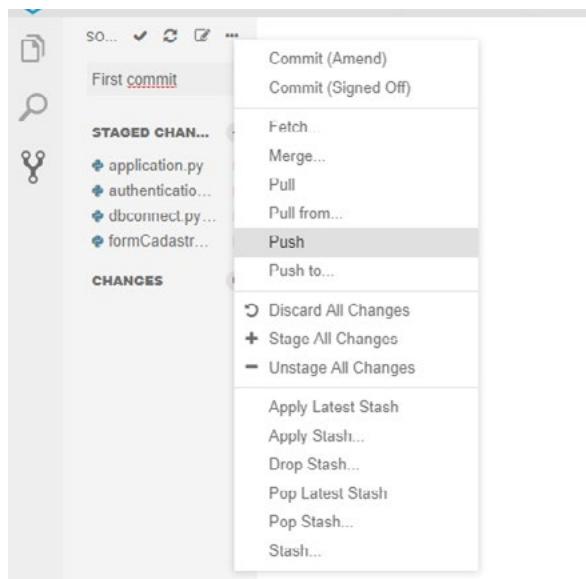


Clique no ícone "+" (Stage All Changes) da seção "Changes", onde estão agrupados os arquivos em que você fez suas últimas alterações. Esse clique é o equivalente ao comando "git add" do Git.

Agora digite no campo "Commit message" uma mensagem que identifica essas suas últimas alterações de código e clique no botão de check para comitar o seu código dentro do Gitpod. Este é o clique equivalente ao comando "git commit" do Git.



Depois de comitado, por último clique nos três pontinhos "..." e selecione a opção "Push", para realmente enviar essas alterações para o seu repositório de origem no GitHub, sendo este finalmente o clique equivalente ao comando "git push" do Git.



Pronto! Pode ir até o site do GitHub e atualize a página que você verá suas alterações por lá!

# localhost

**Não é necessário, mas se desejar você poderá executar e testar o código localmente em seu computador.**

Siga os passos abaixo:

1. Faça o download do seu código do Gitpod
2. Download e instale python 3 em Python.org
3. Instale as bibliotecas necessárias:

```
pip3 install Flask  
pip3 install PyMongo  
pip3 install DNSPython
```

4. Execute o arquivo application.py
- py application.py

```
PS C:\alline\appbasicao\AppBasicao> py .\application.py  
* Serving Flask app "application" (lazy loading)  
* Environment: production  
WARNING: This is a development server. Do not use it in a production deployment.  
Use a production WSGI server instead.  
* Debug mode: on  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 991-088-446  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)  
127.0.0.1 - - [24/Jun/2019 17:28:37] "GET / HTTP/1.1" 200 -
```



## Heroku Cloud

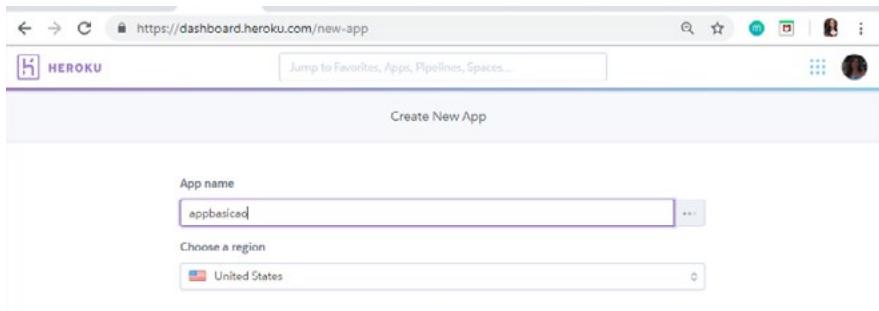
Um bom lugar para disponibilizar seu app atualmente é a tão famosa Cloud.

A plataforma [Heroku.com](https://Heroku.com) é uma excelente opção de Cloud para começar a aprender e a programar para a web.

Heroku é um **PaaS**, uma Plataform as a Service, ou seja, um gerenciador automático que disponibiliza um ambiente completo com servidores de aplicação já previamente instalados e configurados, e ao mesmo tempo totalmente flexível para diferentes linguagens de programação e infraestrutura que você queira usar (incluindo Python).

O Heroku também facilita muito nosso tutorial porque integra diretamente com o GitHub através de um botão **click-to-deploy!**

Crie seu usuário na plataforma Heroku.com e depois clique no botão "Create new app" para hospedar seu novo aplicativo.



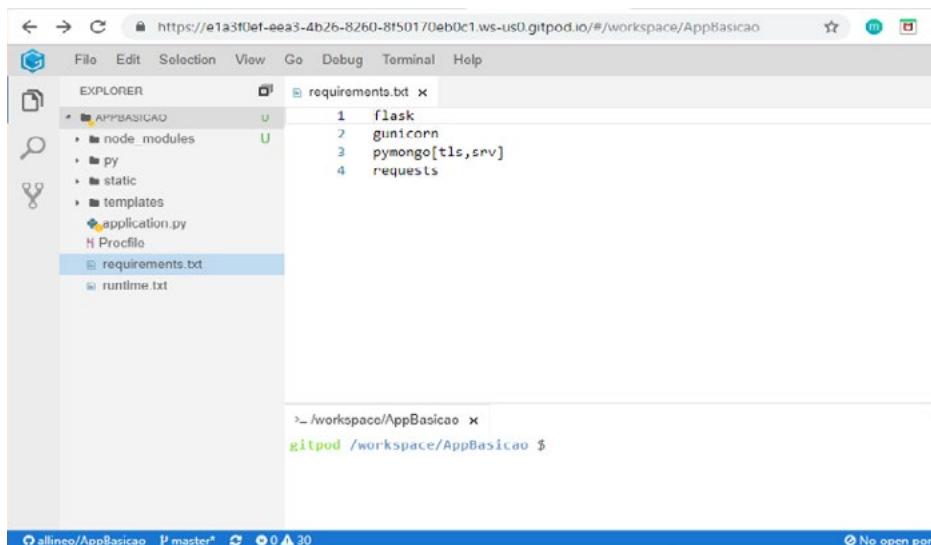
Para o funcionamento do deploy com código Python no Heroku, é necessário adicionar três arquivos de configuração ao projeto.

Volte ao Gitpod e observe na raiz de seu projeto os três arquivos abaixo:

"requirements.txt", "runtime.txt" e "Procfile"

Se desejar, acesse o código desses três arquivos de configuração também através desse link abaixo:

<https://github.com/allineo/AppBasicao/blob/master/>



The screenshot shows a Gitpod interface with the following details:

- File Explorer:** Shows a folder named "APPBASICO" containing "node\_modules", "py", "static", "templates", "application.py", "Procfile", "requirements.txt" (which is selected), and "runtime.txt".
- Terminal:** Displays the command "gitpod /workspace/AppBasicao \$".
- Code View:** The "requirements.txt" file content is shown:

```
flask
gunicorn
pymongo[tls,srv]
requests
```
- Address Bar:** https://e1a3f0ef-eea3-4b26-8260-8f50170eb0c1.ws-us0.gitpod.io/#/workspace/AppBasicao
- Bottom Status Bar:** Shows the repository path "allineo/AppBasicao" and the branch "master".

Aproveite para observar também que esses três arquivos já foram importados para o seu repositório do GitHub quando você importou os arquivos do projeto de exemplo no começo desse tutorial. Vá até o GitHub e certifique-se que eles estão lá.

Agora voltando para o Heroku.com, depois de criado o App e subido os arquivos de configuração para o GitHub, selecione a opção "Deploy" no menu, e então o método GitHub aparecerá entre as três opções de "Deployment Method". Clique no botão "Connect to GitHub" e autorize a conexão.

Digite então o nome de seu repositório do GitHub no campo em aberto e clique em "Search".

Após a autorização, clique no botão "Connect" do repositório.

The screenshot shows the Heroku Dashboard at <https://dashboard.heroku.com/apps/appbasicao/deploy/github>. The page is titled 'appbasicao' under the 'Personal' section. It displays options for connecting the app to a pipeline or GitHub. Under 'Deployment method', the 'GitHub' option is selected, showing a connection to 'allineo' with the repository 'appbasicao'. A 'Connect' button is visible at the bottom right of the GitHub section.

Com a conexão estabelecida, o deploy ficará disponível na seção "Manual Deploy", através do botão "Deploy Branch" no final da página. A branch escolhida em geral será a "master".

The screenshot shows the Heroku dashboard for the app 'appbasico'. At the top, it says 'App connected to GitHub' and 'Code diffs, manual and auto deploys are available for this app.' A message box indicates it is connected to 'allneo/appBasico' by 'allneo'. Below this, there's a note about releases in the activity feed. Under 'Automatic deploys', it says 'Enable automatic deploys from GitHub' and provides instructions: 'Every push to the branch you specify here will deploy a new version of this app. Deploys happen automatically; be sure that this branch is always in a deployable state and any tests have passed before you push.' It includes a dropdown for 'Choose a branch to deploy' set to 'master', a checkbox for 'Wait for CI to pass before deploy', and a button to 'Enable Automatic Deploys'. Under 'Manual deploy', it says 'Deploy a GitHub branch' and 'This will deploy the current state of the branch you specify below.' It also has a dropdown for 'Choose a branch to deploy' set to 'master' and a 'Deploy Branch' button.

Ao clicar no botão "Deploy Branch", o deploy começará a ser realizado.

Com o término bem sucedido, o Heroku gerará e disponibilizará gratuita e automaticamente uma url de domínio para seu app.

E mais... O Heroku já gera a url segura criptografada automaticamente no formato **HTTPS!**

Para acessar a sua url pública é só clicar no botão "View"!

The screenshot shows the deployment summary for the 'appbasico' app. It lists the steps: 'Choose a branch to deploy' (set to 'master'), 'Deploy Branch' (button), 'Receive code from GitHub' (status green), 'Build master: 5e48eebc' (status green), 'Release phase' (status green), and 'Deploy to Heroku' (status green). A message at the bottom says 'Your app was successfully deployed.' with a 'View' button.



**Maaaas, caso o seu deploy não tenha sido bem sucedido... :-(**  
**Você pode acessar o log do seu deploy no console do Heroku. Vá no botão "More" do console e selecione a primeira opção "View logs".**

```

request_id=310972a2-f497-4a40-a341-149f2f329025 fwd="52.73.113.104" dyno:web.1 connect=1ms bytes=467 protocol=https
2019-06-29T17:26:18.250168+00:00 heroku[router]: at+info method=GET path="/static/policy.html" host=appbasicao.herokuapp.com request_id=de7a7ac494-2f55-48ff-a052-52792d229e56 fwd="52.73.113.104" dyno:web.1 connect=0ms service=796ms status=200 bytes=512B protocol=https
2019-06-29T17:26:18.237716+00:00 app[web.1]: 10.49.234.100 - - [29/Jun/2019:17:26:18 +0000] "GET / HTTP/1.1" 302 241 "-" "Mediumbot-MetaTagFetcher/0.3 (<https://medium.com/>)"
2019-06-29T17:26:18.246181+00:00 app[web.1]: 10.169.214.104 - - [29/Jun/2019:17:26:18 +0000] "GET /static/policy.html HTTP/1.1" 200 0 "-" "Mediumbot-MetaTagFetcher/0.3 (<https://medium.com/>)"

```

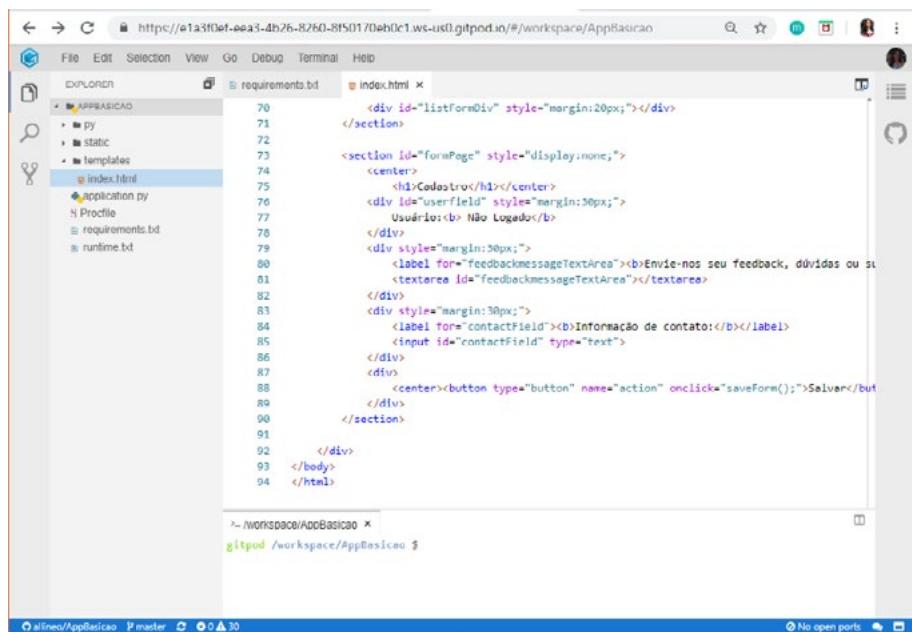
## \*\* Observação \*\*

Se desejar configurar uma url própria para seu app, você pode adquiri-la em algum provedor de domínios, como por exemplo, [GoDaddy.com](#) ou [Registro.br](#), e depois adicione-a ao Heroku na seção "Domains" no menu "Settings".

# Banco de Dados

Continuando na montagem de uma base para seu app, vamos agora criar um formulário de cadastro que usa um banco de dados para armazenar as informações. O frontend para esse formulário de exemplo será bem simples.

Observe o código da seção "formPage" em seu template "index.html":



```
File Edit Selection View Go Debug Terminal Help
EXPLORADOR requirements.txt index.html
APPLICAÇÃO
  - py
  - static
  - templates
    - index.html
    - application.py
    - Procfile
    - requirements.txt
    - runtime.txt

index.html
70      <div id="listFormDiv" style="margin:20px;"></div>
71  </section>
72
73  <section id="formPage" style="display:none;">
74    <center>
75      <h1>Cadastro</h1></center>
76      <div id="userfield" style="margin:30px;">
77        Usuário:<b> Não Logado</b>
78      </div>
79      <div style="margin:30px;">
80        <label for="feedbackmessageTextArea"><b>Envie-nos seu feedback, dúvidas ou sugestões!</b></label>
81        <textarea id="feedbackmessageTextArea"></textarea>
82      </div>
83      <div style="margin:30px;">
84        <label for="contactField"><b>Informação de contato:</b></label>
85        <input id="contactField" type="text">
86      </div>
87      </div>
88      <center><button type="button" name="action" onclick="saveForm();">Salvar</button>
89    </div>
90  </section>
91
92  </div>
93 </body>
94 </html>
```

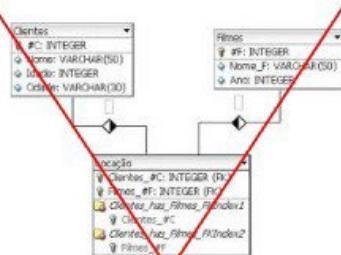
Crie seu próprio formulário com os campos que dizem respeito ao modelo de negócios da ideia de seu app.

# MongoDB

Agora vamos criar o banco de dados que armazenará os campos desse formulário que criamos. Escolhemos um banco de dados NÃO RELACIONAL (NoSQL) porque é o mais simples e ágil hoje em dia.

O MongoDB é uma opção bem disseminada no mercado com bastante recursos para o desenvolvimento. Armazena registros (que chama de documentos) diretamente no formato JSON.

## NoSQL Not Only SQL



- Keys
- Relationships
- Dependencies
- Fixed Schema
- High Integrity

# MongoDB Atlas

Para não termos trabalho de instalação, configuração, manutenção, load balancing, escalonamento de software, etc, vamos utilizar um banco de dados que já seja oferecido na Cloud com todas essas facilidades preestabelecidas.

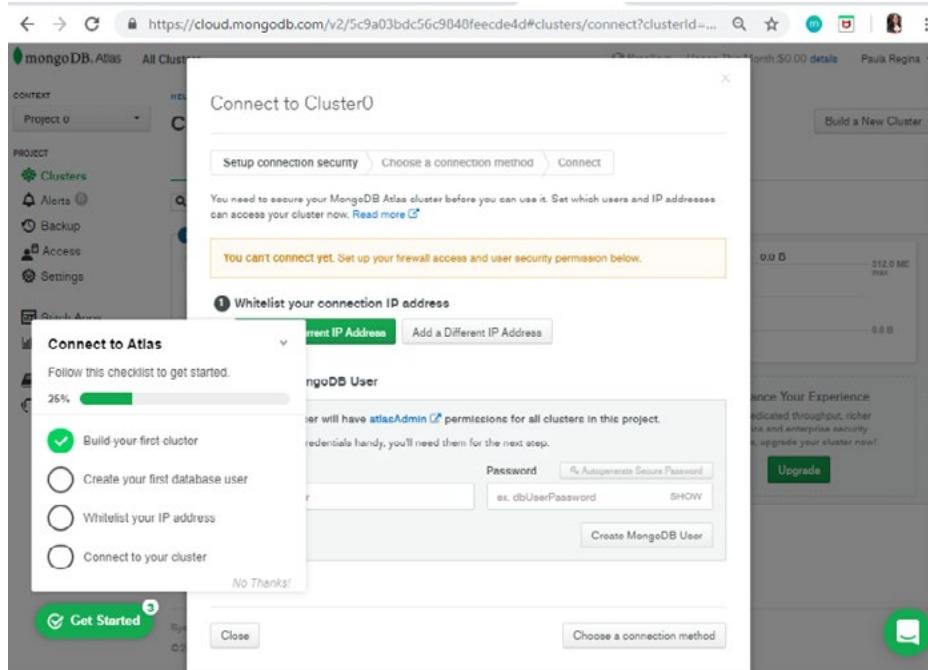
O MongoDB Atlas é uma opção auto gerenciada que tem uma versão de custo zero para desenvolvedores nas fases iniciais de projeto. Tudo o que a gente precisa para começar.

The screenshot shows the MongoDB Atlas web interface for creating a new cluster. At the top, there's a navigation bar with a back arrow, forward arrow, a search icon, a star icon, a user profile icon, and a 'Clusters' button. Below the navigation is a header with the MongoDB logo and the text 'Create New Cluster'. The main area is titled 'Global Cluster Configuration' and contains several sections:

- Cloud Provider & Region:** Set to 'AWS, Oregon (us-west-2)'. It includes buttons for AWS, Google Cloud Platform, and Azure.
- Region Selection:** A section for selecting a region. It shows '★ recommended region' (US East) and lists regions grouped by continent:
  - NORTH AMERICA:** N. Virginia (us-east-1) ★ (FREE TIER AVAILABLE), Ohio (us-east-2) ★, N. California (us-west-1), Oregon (us-west-2) ★ (FREE TIER AVAILABLE).
  - EUROPE:** Stockholm (eu-north-1) ★ (FREE TIER AVAILABLE), Ireland (eu-west-1) ★ (FREE TIER AVAILABLE), London (eu-west-2) ★.
  - AUSTRALIA:** Sydney (ap-southeast-2) ★.
  - ASIA:** Tokyo (ap-northeast-1) ★, Seoul (ap-northeast-2).
- Cost Information:** Shows '\$0.54/hour' and a note: 'Pay-as-you-go! You will be billed hourly and can terminate your cluster anytime. Excludes replicate data transfer, backup, and more.'
- Action Buttons:** 'Cancel' and 'Create Cluster' buttons.

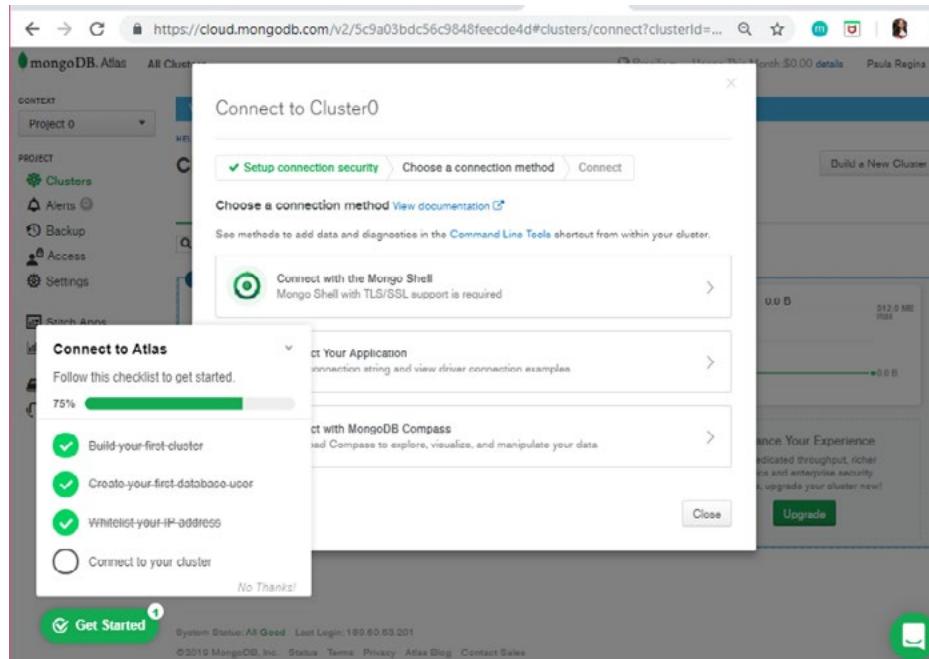
Comece criando o cluster onde seu banco de dados será provisionado. Escolha uma das "Free Tiers" e clique em "Create Cluster".

Quando o cluster estiver criado, clique no botão "Connect" e vá seguindo todos os passos citados no botão "Get Started".

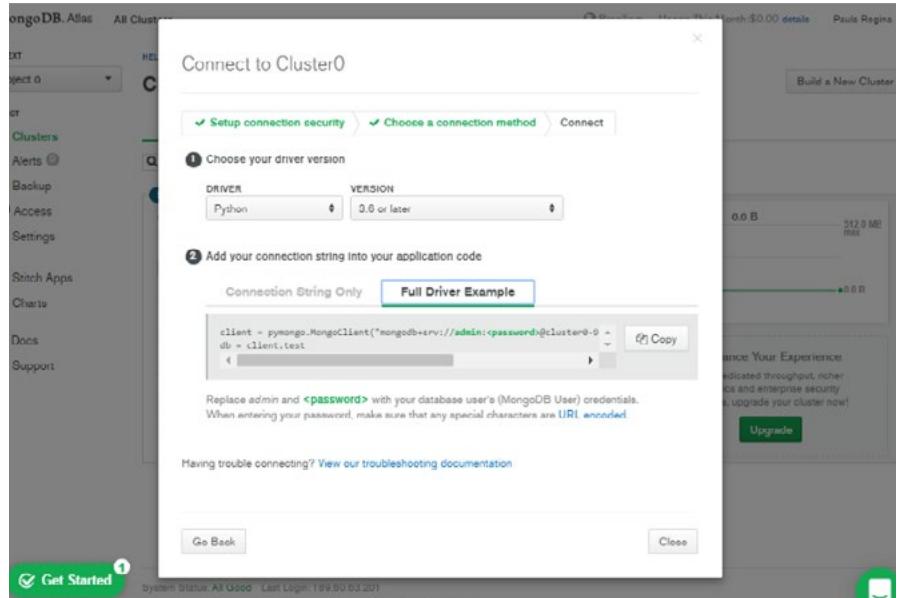


Clique em "Add your current IP Address", e aceite o endereço IP que for detectado clicando no botão "Add IP Address".

Depois crie seu primeiro usuário admin, dê uma senha a ele e clique no botão "Create MongoDB User". Finalmente clique em "Choose a connection method".



Na seção de "Connection Method", nós vamos conectar via aplicação que é a opção "Connect your Application".



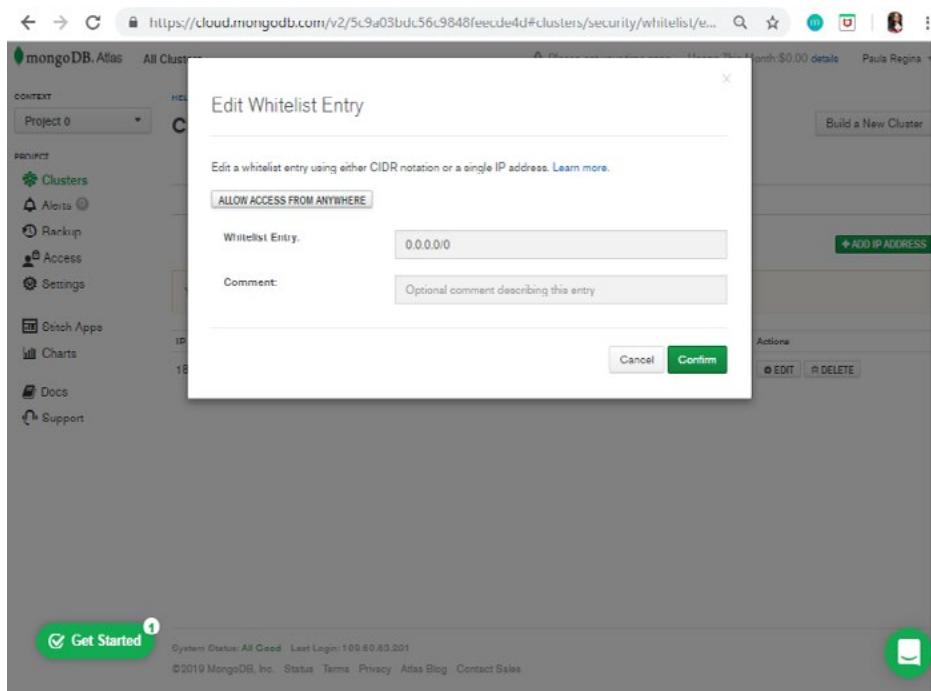
Nesta tela acima, selecione o driver para Python, na versão "3.6 or later" e depois observe o código "Full Driver Example".

Observe que o exemplo utiliza a biblioteca PyMongo. É ela que iremos utilizar para conectar no Python com o MongoDB.

Você precisará copiar a string de conexão com o banco para colar em nosso código de exemplo.

Outra configuração importante, é liberar os endereços IPs do projeto no MongoDB Atlas, para o banco de dados ser acessado de qualquer lugar.

Na aba "Security", clique na opção "IP Whitelist" e depois na action "Edit" de seu IP detectado. No popup, clique no botão "Allow access from anywhere" e depois clique em "Confirm".



# PyMongo

Voltemos ao código.

Para persistir as informações digitadas no form que criamos no frontend precisamos criar o backend em Python. E para a conexão do backend com o banco de dados utilizaremos a biblioteca PyMongo.

Comece criando um diretório chamado "py" no projeto do Gitpod e adicione o arquivo Python "dbconnect.py", que executa a conexão com o MongoDB. É só copiar o código abaixo:

<https://github.com/allineo/AppBasicao/blob/master/py/dbconnect.py>

The screenshot shows a Gitpod workspace interface. On the left, the Explorer sidebar displays a project structure for 'APPBASICO'. Inside 'py', there are files: \_\_init\_\_.py, authentication.py, config.json, dbconnect.py (which is currently selected), and formCadastro.py. The main area shows the content of dbconnect.py:

```
1 import pymongo.json
2
3
4 def getConfig(configitem):
5     with open('py/config.json', 'r') as f:
6         config = json.load(f)
7     return config[configitem]
8
9
10 def dbConnection(dbcollectionname):
11     dburi = getConfig('dburi')
12     dbname = "test"
13     client = pymongo.MongoClient(dburi)
14     db = client[dbname]
15     dbcollection = db[dbcollectionname]
16     return dbcollection
17
18
19 def saveData(dbcollectionname, data):
20     dbcollection = dbConnection(dbcollectionname)
21     id = dbcollection.insert_one(data).inserted_id
22     return str(id)
23
```

Below the code editor, the Terminal tab shows the command: gitpod /workspace/AppBasicao \$

*Repare que o banco de dados criado automaticamente pelo MongoDB Atlas no Cluster0 gratuito já vem com o nome padrão de "test".*

*E ATENÇÃO! Para o diretório "py" funcionar como arquivos de código no seu projeto Python, é só adicionar o arquivo vazio "`__init__.py`" dentro desse diretório.*

*Adicione também o arquivo de configuração "`config.json`", e substitua a string de conexão "dburi" pela url do seu banco de dados disponibilizada no botão "Connect" de seu cluster, lá de volta na interface do MondoDB Atlas.*

*Não esqueça de trocar também sua SENHA dentro da string de conexão.*



The screenshot shows a code editor with an 'EXPLORER' view on the left and a code editor window on the right. The 'APPLICAÇÃO' folder contains several files: 'py', '\_\_init\_\_.py', 'authentication.py', 'config.json', 'dbconnect.py', and 'formCadastro.py'. The 'config.json' file is selected and its content is displayed in the code editor:

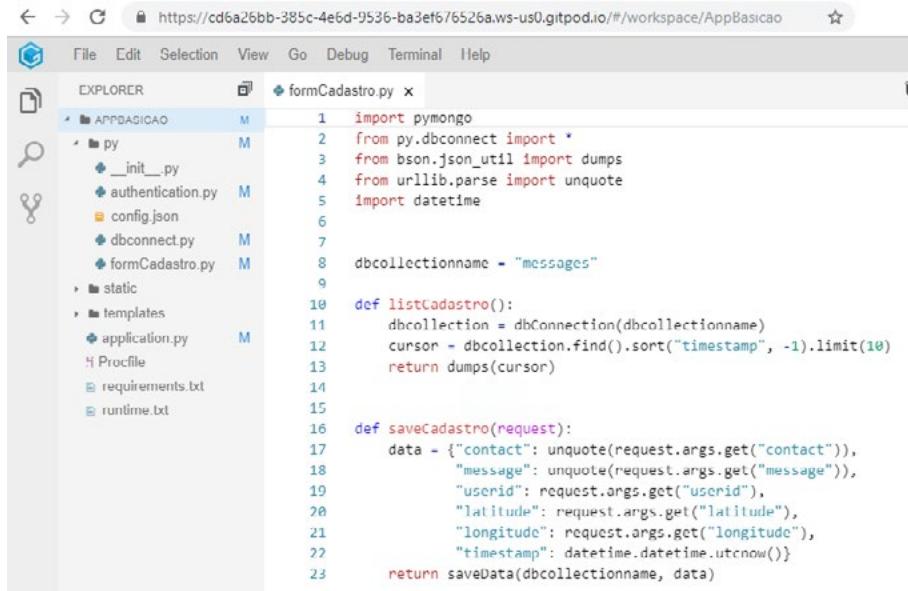
```
1
2   "dburi": "mongodb+srv://XXXXXXXX:XXXXXX@cluster0-jvnpq.mongodb.net/test?retryWrites=true",
3   "fbclientId": "XXXXXXXXXXXX",
4   "fbclientSecret": "XXXXXXXXXXXX"
```

*As outras variáveis de configuração também deverão ser trocadas nas próximas seções desse artigo.*

Agora você já pode adicionar ao mesmo subdiretório "py" do projeto, o arquivo "formCadastro.py", que contém as funções que listam e salvam no banco os dados digitados no frontend através do formulário de cadastro.

Acesse o código aqui:

<https://github.com/allineo/AppBasicao/blob/master/py/formCadastro.py>



```
File Edit Selection View Go Debug Terminal Help
EXPLORER formCadastro.py
APPOBASICO
  py
    __init__.py
    authentication.py
    config.json
    dbconnect.py
    formCadastro.py
  static
  templates
  application.py
  Profile
  requirements.txt
  runtime.txt

1 import pymongo
2 from py.dbconnect import *
3 from bson.json_util import dumps
4 from urllib.parse import unquote
5 import datetime
6
7 dbcollectionname = "messages"
8
9
10 def listCadastro():
11     dbcollection = dbConnection(dbcollectionname)
12     cursor = dbcollection.find().sort("timestamp", -1).limit(10)
13     return dumps(cursor)
14
15
16 def saveCadastro(request):
17     data = {"contact": unquote(request.args.get("contact")),
18             "message": unquote(request.args.get("message")),
19             "userid": request.args.get("userid"),
20             "latitude": request.args.get("latitude"),
21             "longitude": request.args.get("longitude"),
22             "timestamp": datetime.datetime.utcnow()}
23
24 return saveData(dbcollectionname, data)
```

Atualize o nome de sua collection (que é o equivalente às tabelas de dados dos bancos relacionais) na variável "dbcollectionname".

### \*\* Observação \*\*

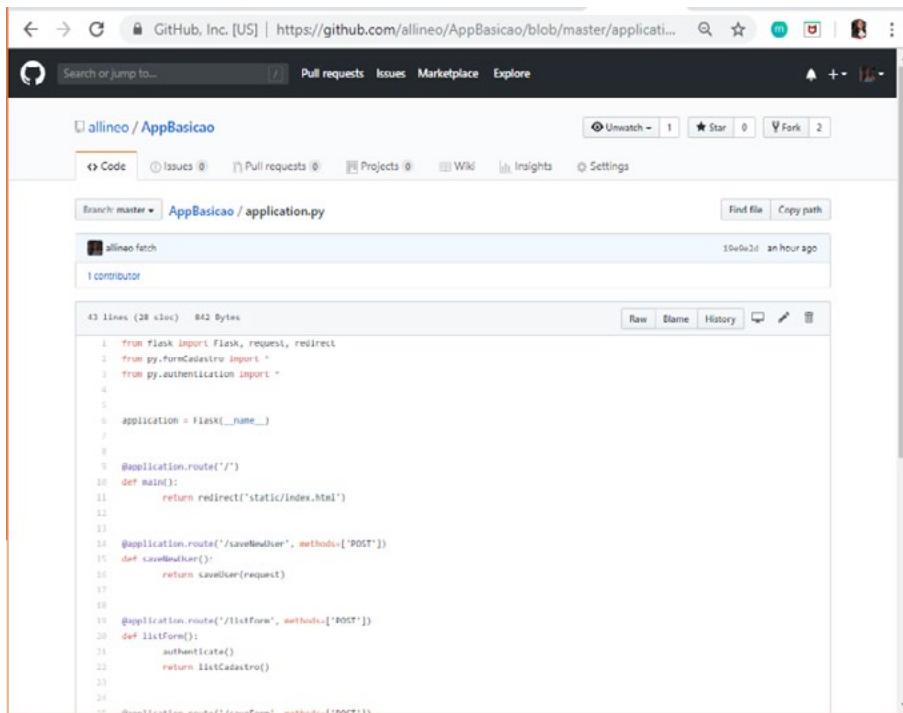
Não se preocupe porque sua collection será criada automaticamente no MongoDB caso ela ainda não exista. Essas são as vantagens de um banco de dados *Schemaless*.

# API Endpoints

Para manter o backend e o frontend completamente desacoplados, independentes e flexíveis, criamos API endpoints para todas as chamadas feitas do frontend para o backend.

Observe no código do arquivo "application.py", os endpoints (cuja sintaxe se inicia pela anotação: `@application.route`) para salvar e recuperar os dados do formulário no banco de dados.

O código pronto para esses métodos encontra-se no link abaixo:  
<https://github.com/allineo/AppBasicao/blob/master/application.py>



The screenshot shows a GitHub repository page for 'allineo / AppBasicao'. The 'application.py' file is displayed. The code defines a Flask application with routes for saving and listing users. The GitHub interface includes navigation tabs like 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. The file has 43 lines (28 sloc) and 842 bytes. It contains imports for Flask, redirect, render\_template, and authentication. The main function sets up a route for '/' and a POST route for '/saveUser' to handle user saves. Another route for '/listForm' handles user lists.

```
from flask import Flask, request, redirect
from pyturnredirec import *
from py.authentication import *

application = Flask(__name__)

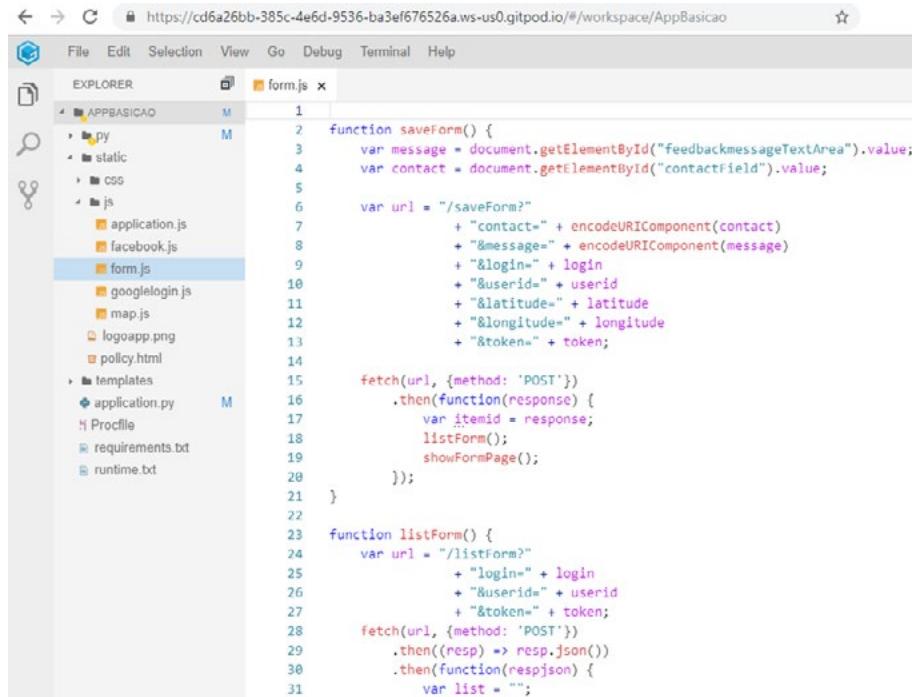
@application.route('/')
def main():
    return redirect('static/index.html')

@application.route('/saveUser', methods=['POST'])
def saveUser():
    return saveUser(request)

@application.route('/listForm', methods=['POST'])
def listForm():
    authenticate()
    return listCadastro()
```

Agora perceba que as chamadas no frontend para o acesso aos dados do backend estão sendo feitas através das funções de **FETCH** do JavaScript, que se encontram no arquivo "form.js".

<https://github.com/allineo/AppBasicao/blob/master/static/js/form.js>



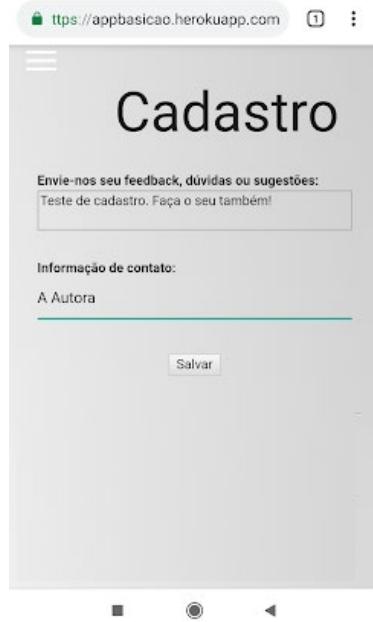
The screenshot shows a code editor interface with a top navigation bar and a sidebar labeled 'EXPLORER'. The 'EXPLORER' sidebar lists several files and folders under 'APPBASICAO': py, static, CSS, js (with form.js selected), application.js, facebook.js, googlelogin.js, map.js, logoapp.png, policy.html, templates, application.py, Profile, requirements.txt, and runtime.txt. The main editor area displays the 'form.js' file content:

```
1 function saveForm() {
2     var message = document.getElementById("feedbackmessageTextArea").value;
3     var contact = document.getElementById("contactField").value;
4
5     var url = "/saveForm?"
6         + "contact=" + encodeURIComponent(contact)
7         + "&message=" + encodeURIComponent(message)
8         + "&login=" + login
9         + "&userid=" + userid
10        + "&latitude=" + latitude
11        + "&longitude=" + longitude
12        + "&token=" + token;
13
14     fetch(url, {method: 'POST'})
15         .then(function(response) {
16             var itemid = response;
17             listForm();
18             showFormPage();
19         });
20 }
21
22
23 function listForm() {
24     var url = "/listForm?"
25         + "login=" + login
26         + "&userid=" + userid
27         + "&token=" + token;
28     fetch(url, {method: 'POST'})
29         .then((resp) => resp.json())
30         .then(function(resjson) {
31             var list = "";
32         });
33 }
```

Substitua o nome das variáveis de acordo com os campos do formulário que você desejar construir para o seu próprio app.

Atualize todos esses arquivos no seu código do Gitpod e depois faça o commit e o push para o repositório remoto no GitHub.

Depois volte a interface do Heroku e clique no botão "Deploy Branch" como explicado anteriormente.



*Gostou do tutorial até aqui?*

*Deixe um feedback no próprio formulário de cadastro do nosso app de exemplo!*

*Acesse: <https://appbasicao.herokuapp.com>*

# Data Explorer

Para visualizar os dados após eles serem criados, existe uma interface dentro do MongoDB Atlas.

É só clicar no botão "Collections" dentro da tela do Cluster. Este botão vai abrir a tela abaixo, com todos seus databases e os dados dentro de cada uma de suas collections.

Explore à vontade.

The screenshot shows the MongoDB Atlas Cluster interface. On the left, there's a sidebar with various project management options like Clusters, Alerts, Backup, Access, Settings, Stitch, Charts, Docs, and Support. The main area shows a cluster named 'Cluster0' with details: VERSION 4.0.9 and REGION N. Virginia (us-east-1). Below this, there are tabs for Overview, Real Time, Metrics, and Collections. The Collections tab is currently active, displaying information about the 'test.messages' collection. It shows a collection size of 1.79KB, 15 total documents, and 30KB total size. There are buttons for Find and Indexes. A search bar at the top right says 'Find' and has a 'Refresh' button. At the bottom, there's a section for QUERY RESULTS showing 1-15 OF 16 documents. One document is highlighted with its \_id, contact, message, user id, latitude, longitude, and timestamp. The timestamp is listed as 2019-04-12T22:18:15, 758+00:00.

```
_id: ObjectId("5c14cc5fef1d90000fd9775")
contact: ""
message: "TESTANDO LOCALIZAÇÃO"
userId: "101242936706498883349"
latitude: "-22.932648"
longitude: "-43.757837000000005"
timestamp: 2019-04-12T22:18:15, 758+00:00
```

```
_id: ObjectId("5c14cc5fef1d90000fd9775")
contact: "info"
message: "Olá, teste"
userId: "100894984851435732335"
latitude: "-29.698160"
longitude: "-43.796663"
timestamp: 2019-04-13T11:49:41.171+00:00
```

# **EXTRAS**

A partir de agora nós vamos começar a nos aprofundar bem mais nas funcionalidades e na complexidade dessas funcionalidades dentro de nosso app.

Mas não se preocupe que será possível acompanhar o desenvolvimento mesmo para quem ainda não tem muita prática, pois vamos descrever tela por tela, com cada passo que precisa ser dado para adicionar autenticação, mapa ou até mesmo migrar de Cloud para a plataforma da Amazon AWS.



Existem várias formas de autenticar um usuário em seu app. A pior delas é reinventar a roda e fazer todo o código de forma customizada com seu próprio punho.

São muitos os plugins de autenticação disponíveis no mercado. Vamos utilizar aqui, a título de exemplo, dois dos mais utilizados no mercado, que são o botão de Sign in do Facebook e o do Google.

### **\*\* Observação \*\***

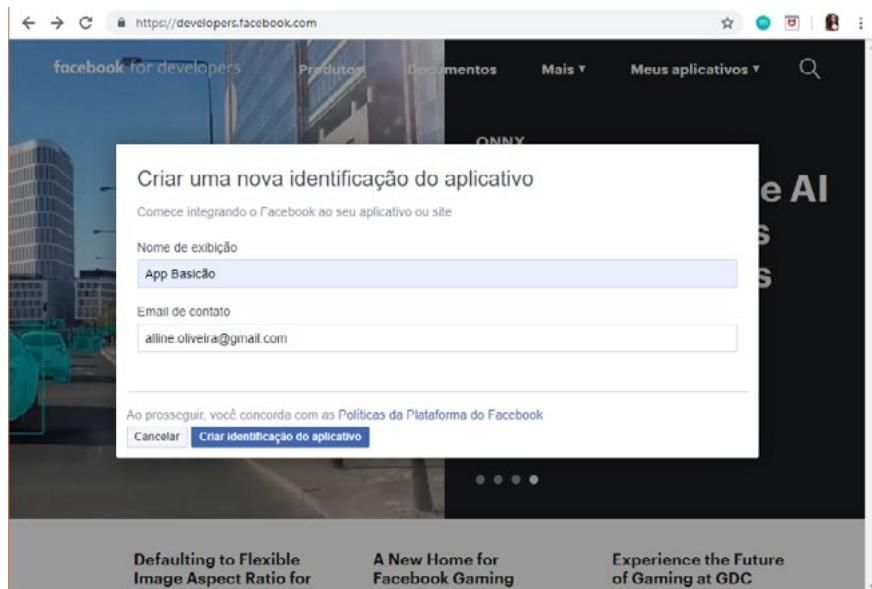
Existem também plataformas que já automatizam vários logins sociais ao mesmo tempo em um único plugin, como por exemplo o Firebase Authentication ou o [Auth0.com](https://auth0.com).

Login com  
o Facebook

Para adicionar aquele botão de "Login com o Facebook" em seu app, você precisa começar associando a sua conta normal do Facebook (sim, eu sei que você tem uma) à uma conta de desenvolvedor do Facebook.

Acesse a plataforma [Developers.Facebook.com](https://developers.facebook.com), e clique na opção de "Adicionar Novo Aplicativo" à sua conta.

Digite o nome e o e-mail de contato de seu app e clique no botão "Criar identificação de aplicativo".



**Na próxima tela de "Cenários", selecione a opção de "Integrar Login do Facebook" e clique em "Confirmar".**

**\*\* Observação \*\***

**Posteriormente você vai precisar do número "ID do Aplicativo", que se encontra no topo da tela, para configurar o código do botão de login.**



**Na próxima tela de "Configurações Básicas", preencha os campos de acordo com as informações de seu app.**

**O "Domínio do aplicativo" é a url configurada pelo Heroku. Adicione também a logo no tamanho mínimo de 512x512 pixels e selecione uma categoria para seu app.**

A política de privacidade pode ser gerada por algum site disponível na web, e você pode adicionar o arquivo gerado diretamente na pasta "static" do código do seu aplicativo.

Assim como esse exemplo que utilizamos aqui:

<https://appbasicao.herokuapp.com/static/policy.html>



Clique em "Salvar alterações".

Você vai precisar também configurar a url de retorno de seu app quando o usuário se autenticar pelo Facebook.

No menu esquerdo da tela, selecione a opção "Login do Facebook" e depois "Configurações".

No campo "URIs de redirecionamento do OAuth válidos", você pode cadastrar a própria url da página de início do seu app configurada pelo Heroku.

The screenshot shows the 'Configurações de OAuth do cliente' (OAuth Client Settings) section of the Facebook developer console. It includes several configuration options:

- Login no OAuth do cliente**: Allows the flow of tokens from the standard OAuth. Protects your application from abuse using options like rate limits and IP restrictions.
- Login do OAuth na Web**: Allows users to log in via the Facebook login button on your website.
- Exigir HTTPS**: Requires the use of HTTPS for URLs of the application and the token endpoint. Recommended.
- Forçar reautenticação do OAuth na Web**: Forces users to log in again when they visit your website after logging in via the Facebook login button.
- Login do OAuth no navegador incorporado**: Allows users to log in via the embedded Facebook login button.
- User mode strict for OAuth redirection URLs**: Allows users to log in via the Facebook login button even if the URL corresponds to a different URL for redirection.
- URLs de redirecionamento do OAuth válidas**: A text input field containing `https://openclassroom.herokuapp.com/`.
- Login de dispositivos**: A checkbox option.

At the bottom right are 'Desmarcar' and 'Salvar alterações' buttons.

Clique novamente em "Salvar alterações".

Agora para finalizar. Ative suas configurações clicando na opção "Desativado" que se encontra em cinza no topo da tela. Ao clicar aparecerá uma janela pop-up.

Confirme que deseja "Tornar o aplicativo público" e o botão ficará verde indicando que seu app está ativado no Facebook.

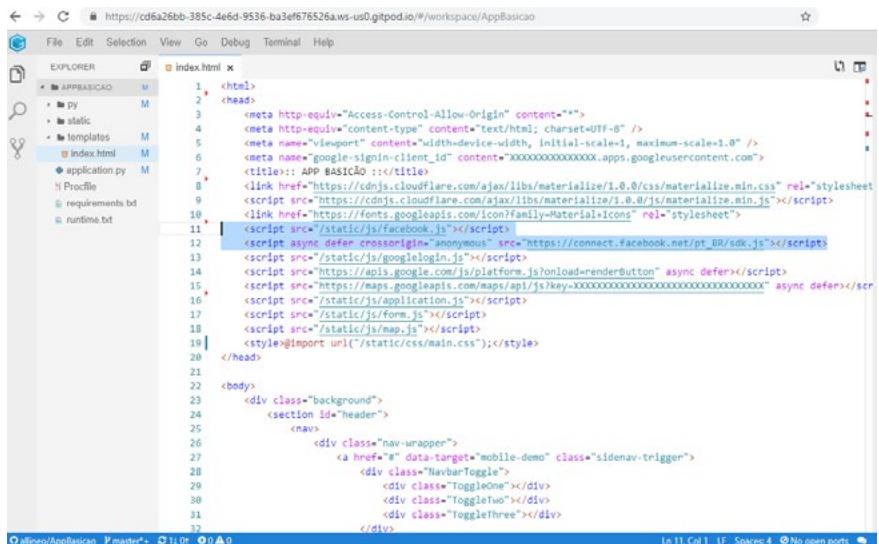
The screenshot shows the 'Meus aplicativos' (My Apps) section of the Facebook developer dashboard. The app status is shown as 'ATIVADO' (Enabled) with a green button. Other visible buttons include 'Status: ao vivo' (Live Status), 'Visualizar o Analytics' (View Analytics), and 'Ajuda' (Help). A search bar at the top says 'Pesquisar em developers.facebook.com'.

Agora vamos ao código do botão de login do Facebook que se encontra em nosso app.

No arquivo "index.html", lá no seu projeto do Gitpod, abra a seção "head" e adicione os scripts de importação do plugin do Facebook, assim como o ressaltado na figura abaixo.

Pode copiar e colar o código do arquivo de exemplo a seguir:

<https://github.com/allineo/AppBasicao/blob/master/templates/index.html>



```
File Edit Selection View Go Debug Terminal Help
EXPLORER index.html
1 <html>
2   <head>
3     <meta http-equiv="Access-Control-Allow-Origin" content="">
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1.0" />
6     <meta name="google-signin-client_id" content="XXXXXXXXXXXXXX.apps.googleusercontent.com">
7     <title>:: APP BASICO ::</title>
8     <link href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css" rel="stylesheet"
9       <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
10    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
11    <script src="/static/js/facebook.js"></script>
12    <script async defer crossorigin="anonymous" src="https://connect.facebook.net/pt_BR/sdk.js"></script>
13    <script src="/static/js/googlelogin.js"></script>
14    <script src="https://apis.google.com/js/platform.js?onload=renderButton" async defer></script>
15    <script src="https://maps.googleapis.com/maps/api/js?key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"></script>
16    <script src="/static/js/application.js"></script>
17    <script src="/static/js/form.js"></script>
18    <script src="/static/js/map.js"></script>
19    <style>@import url('/static/css/main.css');</style>
20   </head>
21
22   <body>
23     <div class="background">
24       <section id="header">
25         <nav>
26           <div class="nav-wrapper">
27             <a href="#" data-target="#mobile-demo" class="sidenav-trigger">
28               <div class="NavbarToggle">
29                 <div class="ToggleOne"></div>
30                 <div class="ToggleTwo"></div>
31                 <div class="ToggleThree"></div>
32             </div>
33         </nav>
34       </section>
35     </div>
36   </body>
37 
```

Depois vá até a section "loginPage" no "body" desse mesmo arquivo "index.html" e adicione o código do botão de login:

```
50   <section id="loginPage">
51     <center>
52       <br/><br/>
53       
54       <br/><br/><br/>
55
56       <div id="fb-root"></div>
57       <div class="fb-login-button" data-size="large" data-button-type="login_with"
58           data-auto-logout-link="true" data-use-continue-as="false"
59           scope="public_profile,email" onlogin="checkLoginState();"></div>
60
61       <br/><br/>
62       <div id="google-signin2"></div>
63     </center>
64   </section>
```

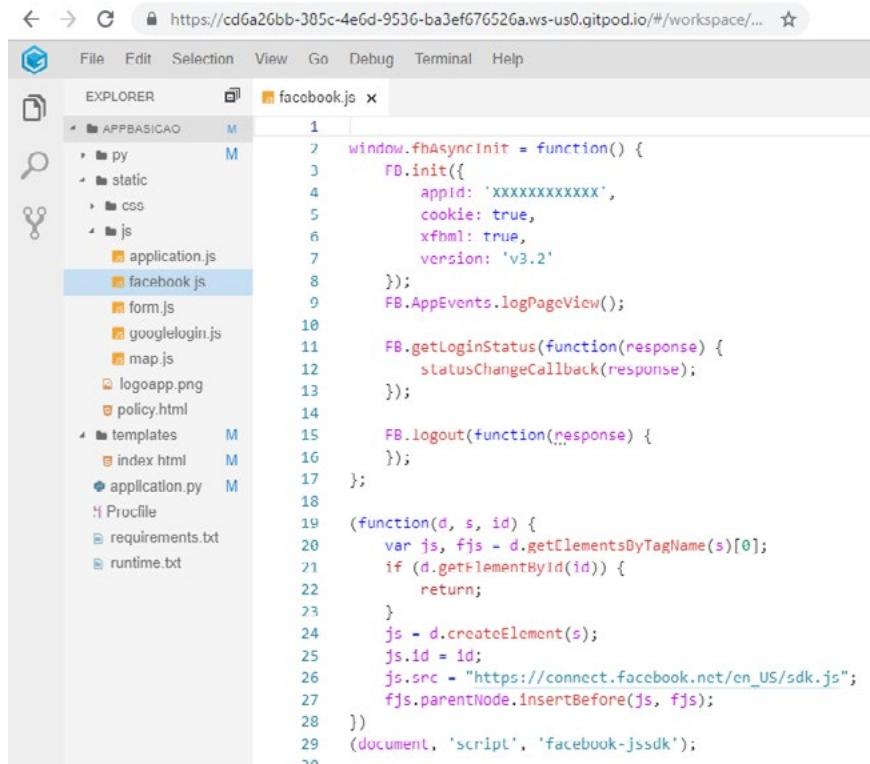
Continuando.

Agora copie todo o arquivo "facebook.js" para dentro da pasta "/static/js/" do seu projeto do Gitpod.

Use o código de exemplo abaixo:

<https://github.com/allineo/AppBasicao/blob/master/static/js/facebook.js>

Não esqueça de substituir a variável "appId" da função "FB.init" pelo seu "ID do Aplicativo".



The screenshot shows a Gitpod workspace interface. At the top, there's a navigation bar with back, forward, and search icons, followed by the URL [https://cd6a26bb-305c-4e6d-9536-ba3ef676526a.ws-us0.gitpod.io/#/workspace/...](https://cd6a26bb-305c-4e6d-9536-ba3ef676526a.ws-us0.gitpod.io/#/workspace/). Below the URL is a star icon. The main area has tabs for File, Edit, Selection, View, Go, Debug, Terminal, and Help. On the left is an Explorer sidebar showing a project structure: APPBASICAO (M) containing py, static (with css and js), js (with application.js, facebook.js, form.js, googlelogin.js, map.js, logoapp.png, policy.html), templates (with index.html), application.py, and Profile, requirements.txt, and runtime.txt. The js folder is expanded, and facebook.js is selected. To the right of the sidebar is a code editor window titled "facebook.js" with the following content:

```
1 window.fbAsyncInit = function() {
2     FB.init({
3         appId: 'XXXXXXXXXXXX',
4         cookie: true,
5         xfbml: true,
6         version: 'v3.2'
7     });
8     FB.AppEvents.logPageView();
9
10    FB.getLoginStatus(function(response) {
11        statusChangeCallback(response);
12    });
13
14    FB.logout(function(response) {
15    });
16
17 };
18
19 (function(d, s, id) {
20     var js, fjs = d.getElementsByTagName(s)[0];
21     if (d.getElementById(id)) {
22         return;
23     }
24     js = d.createElement(s);
25     js.id = id;
26     js.src = "https://connect.facebook.net/en_US/sdk.js";
27     fjs.parentNode.insertBefore(js, fjs);
28 })
29 (document, 'script', 'facebook-jssdk');
```

Observe a segunda linha da função "statusChangeCallback" onde é passado o token de acesso do usuário que acabou de se logar.

Esse token, assim como o id do usuário logado, serão guardados em variáveis no JavaScript, para depois serem verificados no backend, quando o aplicativo fizer chamadas para os endpoints das APIs.

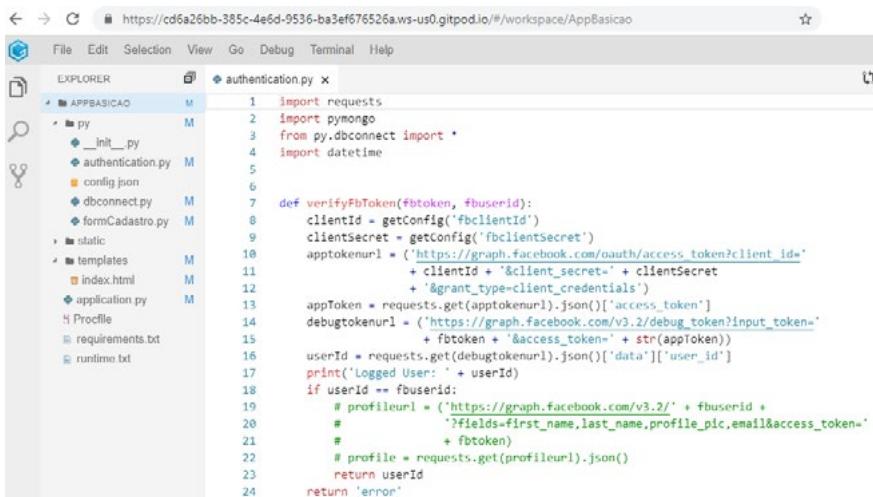
Observe também nesse mesmo arquivo a função "FB.api('/me")" na linha abaixo, que busca os dados de cadastro no Facebook do usuário logado e envia para o backend para serem salvos no banco.

```
47     function getFbUser() {
48         FB.api('/me?fields=id,email,name', function(response) {
49             saveNewUser("fb", response.id, response.name, response.email);
50         });
51     }
```

No backend, o token de login do Facebook deve ser verificado antes dos dados serem acessados, para ter certeza que o usuário foi devidamente autenticado no frontend.

Segue o código para essa verificação:

<https://github.com/allineo/AppBasicao/blob/master/py/authentication.py>



The screenshot shows a code editor interface with the following details:

- File Path:** https://cd5a26bb-385c-4e6d-9536-ba3ef676526aws-us0.gitpod.io/#/workspace/AppBasicao
- File:** authentication.py
- Content:** The code defines a function verifyFbToken that takes fbtoken and fbuserid as parameters. It uses requests to make API calls to Facebook's OAuth endpoint to get an access token. Then it uses another request to the debug\_token endpoint to get user information. Finally, it prints the user ID and returns it or an error message if the user ID doesn't match the provided fbuserid.

```
1  import requests
2  import pymongo
3  from pydbconnect import *
4  import datetime
5
6
7  def verifyFbToken(fbtoken, fbuserid):
8      clientId = getConfig('fbclientId')
9      clientSecret = getConfig('fbclientsecret')
10     appTokenUrl = ('https://graph.facebook.com/oauth/access_token?client_id='
11                   + clientId + '&client_secret=' + clientSecret
12                   + '&grant_type=client_credentials')
13     appToken = requests.get(appTokenUrl).json()['access_token']
14     debugTokenUrl = ('https://graph.facebook.com/v3.2/debug_token?input_token='
15                      + fbtoken + '&access_token=' + str(appToken))
16     userId = requests.get(debugTokenUrl).json()['data']['user_id']
17     print('Logged User: ' + userId)
18     if userId == fbuserid:
19         # profileUrl = ('https://graph.facebook.com/v3.2/' + fbuserid +
20         #                 '?fields=first_name,last_name,profile_pic,email&access_token='
21         #                 + fbtoken)
22         # profile = requests.get(profileUrl).json()
23         # return userId
24     return 'error'
```

Após a validação do token, os dados do usuário logado já podem ser salvos no banco de dados.

A função de salvamento desses dados é chamada pelo endpoint "`/saveNewUser`" do arquivo "application.py", que por sua vez chama a função "saveUser" do arquivo "authentication.py" acima.

Você pode acompanhar os dados armazenados de todos os usuários que estão se logando em seu aplicativo, visualizando-os diretamente lá no banco de dados, através do Data Explorer do MongoDB Atlas, como foi mostrado anteriormente.

The screenshot shows the MongoDB Atlas interface. In the top navigation bar, the URL is https://cloud.mongodb.com/v2/5c979d51c56c90eb23a41592#metrics/repl... and the project is set to 'Project 0'. The main area displays the 'Cluster0' configuration. On the left, there's a sidebar with 'PROJECT' sections for Clusters, Alerts, Backup, Access, Settings, Stitch Apps, Charts, Docs, and Support. The 'Clusters' section shows '1 DATABASES' and '2 COLLECTIONS'. The 'Collections' tab is selected, showing the 'test' database with two collections: 'messages' and 'users'. The 'users' collection has 3 documents and a total size of 36KB. Below the collection table, there are 'Find' and 'Indexes' buttons, and a search bar with the filter '{filter: "example"}'. At the bottom, it says 'QUERY RESULTS 1-3 OF 3' and shows the details of one document:

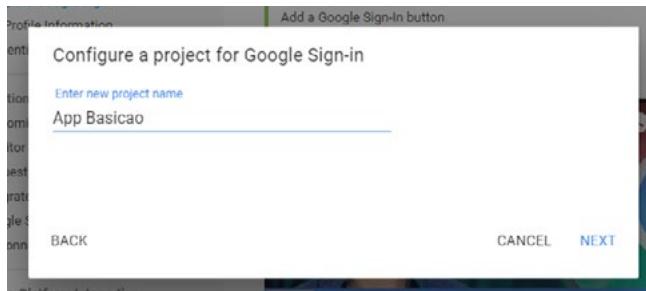
```
_id: ObjectId("5cbdd5d4d4070000a8d499")
login: "g1"
userId: "11722903025814867638"
latitude: "-22.9877442090000008"
longitude: "-41.21511315"
name: "Alline Oliveira"
email: "alline.oliveira@gmail.com"
timestamp: 2019-04-08T23:46:37.482+00:00
```

## Google Sign-in

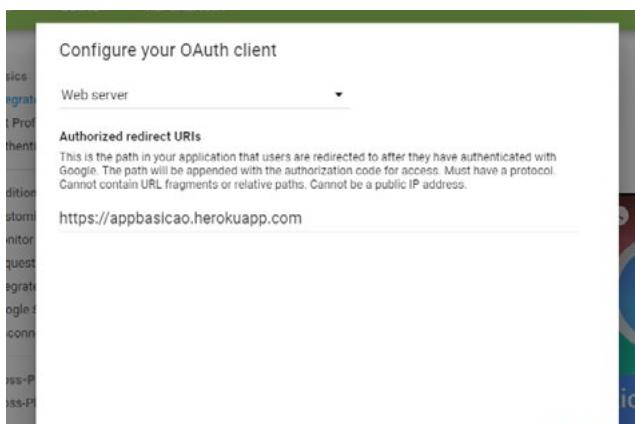
Para adicionar o botão de "Login com o Google" em seu app, você precisa configurar um projeto na plataforma do Google.

Acesse a url abaixo, comece clicando no botão "Configure a project" e adicione um novo projeto:

<https://developers.google.com/identity/sign-in/web/sign-in>

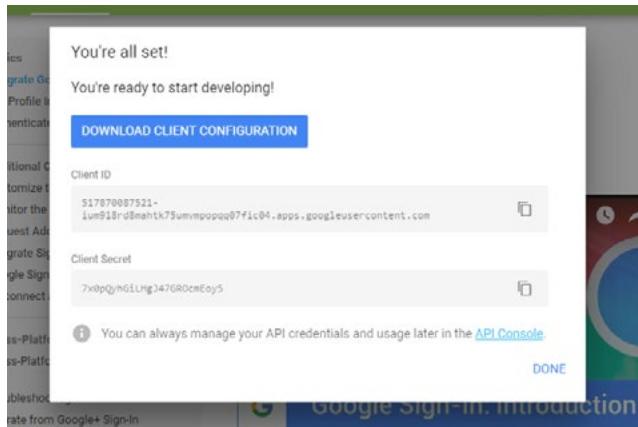


Selecione a opção "Web server" e digite a url de domínio do Heroku para o seu app no campo de "Authorized redirect URIs". Depois clique em "Create".



Na próxima tela aparecerão as strings "Client ID" e "Client Secret" para a conexão do código do seu app.

Copie-as antes de clicar em "Done", ou então selecione a opção: "Download client configuration".



### \*\* Observação \*\*

Se precisar buscar o id de autenticação do cliente desse projeto depois de ter clicado no botão "DONE", acesse a url abaixo e copie o valor da coluna "Client ID" da variável "OAuth client".

<https://console.cloud.google.com/apis/credentials>

*Voltando ao código do seu projeto no Gitpod.*

No arquivo "index.html", adicione as linhas necessárias à configuração do plugin do botão de login do Google, segundo a linha de código destacada no exemplo abaixo:

<https://github.com/allineo/AppBasicao/blob/master/templates/index.html>

```
1 <html>
2   <head>
3     <meta http-equiv="Access-Control-Allow-Origin" content="">
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1.0" />
6     <meta name="google-signin-client_id" content="00000000000000000000.apps.googleusercontent.com">
7     <title>:: APP BÁSICO ::</title>
8     <link href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css" rel="stylesheet">
9     <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
10    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
11    <script src="/static/js/facebook.js"></script>
12    <script async defer crossorigin="anonymous" src="https://connect.facebook.net/pt_BR/sdk.js"></script>
13    <script src="/static/js/googlelogin.js"></script>
14    <script src="https://apis.google.com/js/platform.js?onload=renderButton" async defer></script>
15    <script src="https://maps.googleapis.com/maps/api/js?key=XXXXXXXXXXXXXXXXXXXXXX" async defer></script>
16    <script src="/static/js/application.js"></script>
17    <script src="/static/js/form.js"></script>
18    <script src="/static/js/map.js"></script>
19    <style>@import url("/static/css/main.css");</style>
20  </head>
```

Para o valor "content" da linha do "google-signin-client\_id", substitua pelo valor do "Client ID" gerado na criação de seu projeto Google descrito acima.

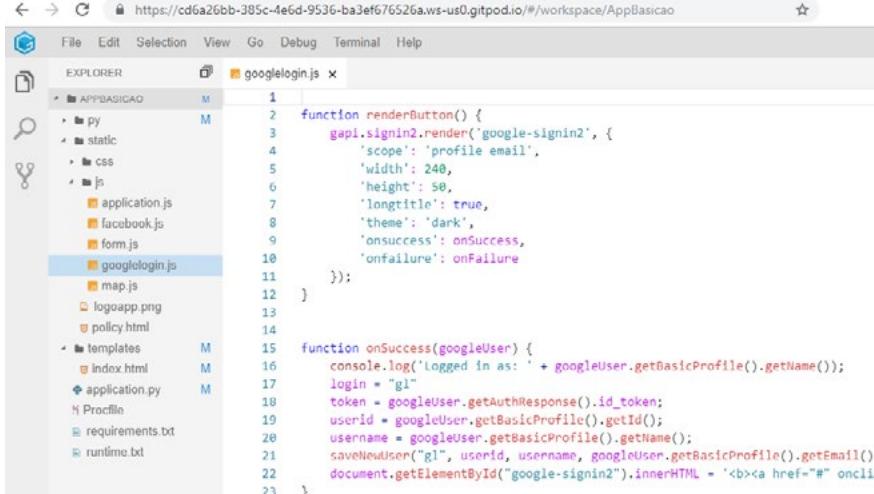
Observe também a importação do arquivo "googlelogin.js", e o script de inicialização das apis do Google tudo na mesma seção de cabeçalho.

E então na seção “body”, as tags html do botão em si, com o id “**“google-signin2”**”, como na linha selecionada abaixo:

```
50     <section id="loginPage">
51         <center>
52             <br/><br/><br/>
53             
54             <br/><br/><br/>
55
56             <div id="fb-root"></div>
57             <div class="fb-login-button" data-size="large" data-button-type="login_with"
58                 data-auto-logout-link="true" data-use-continue-as="false"
59                 scope="public_profile,email" onlogin="checkLoginState();"></div>
60
61             <br/>
62             <div id="google-signin2"></div>
63         </center>
64     </section>
```

Agora no diretório "static/js", adicione o arquivo "googlelogin.js", para fazer a inicialização do botão de login com o Google e para pegar o token de login após o usuário se autenticar.

<https://github.com/allineo/AppBasicao/blob/master/static/js/googlelogin.js>



```
function renderButton() {
  gapi.signin2.render('google-signin2', {
    'scope': 'profile email',
    'width': 240,
    'height': 50,
    'longtitle': true,
    'theme': 'dark',
    'onsuccess': onSuccess,
    'onfailure': onFailure
  });
}

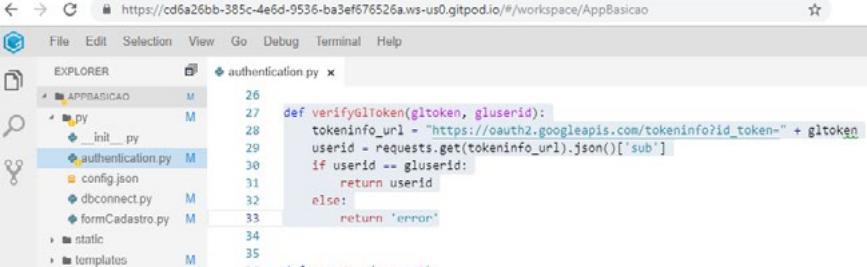
function onSuccess(googleUser) {
  console.log('Logged in as: ' + googleUser.getBasicProfile().getName());
  login = "gl"
  token = googleUser.getAuthResponse().id_token;
  userid = googleUser.getBasicProfile().getId();
  username = googleUser.getBasicProfile().getName();
  saveNewUser(gl, userid, username, googleUser.getBasicProfile().getEmail())
  document.getElementById("google-signin2").innerHTML = '<bx><a href="#" onclick="'
}

function onFailure(error) {
  console.log(error)
}
```

Nesse arquivo "googlelogin.js" também são enviados os dados do usuário logado para o backend para ser salvo no banco de dados.

No backend, a verificação é feita pela função "verifyGLToken" do arquivo "authentication.py".

<https://github.com/allineo/AppBasicao/blob/master/py/authentication.py>



```
def verifyGLToken(gltoken, gluserid):
  tokeninfo_url = "https://oauth2.googleapis.com/tokeninfo?id_token=" + gltoken
  userid = requests.get(tokeninfo_url).json()['sub']
  if userid == gluserid:
    return userid
  else:
    return 'error'
```

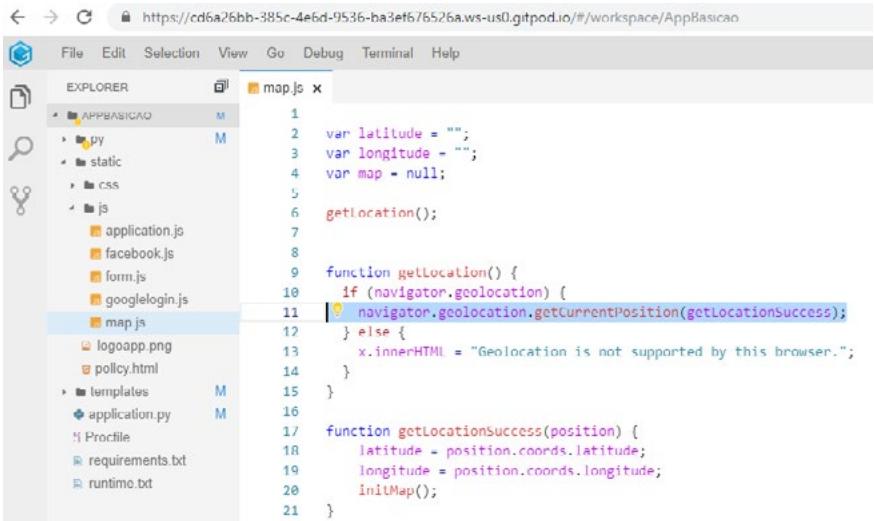
# Localização do usuário

Sua ideia pode necessitar de várias tecnologias diferentes ao ser implementada. Estamos demonstrando aqui algumas das funcionalidades mais comuns de um app, como por exemplo formulários de cadastro, login de usuário, campos de busca, mapas, etc.

Neste capítulo vamos falar de Geolocalização. Para pegar a localização atual do usuário do seu app, iremos utilizar a API de Geolocalização da linguagem HTML.

Segue abaixo o código Javascript para tal, observe a função "getLocation()" do arquivo "map.js":

<https://github.com/allineo/AppBasicao/blob/master/static/js/map.js>



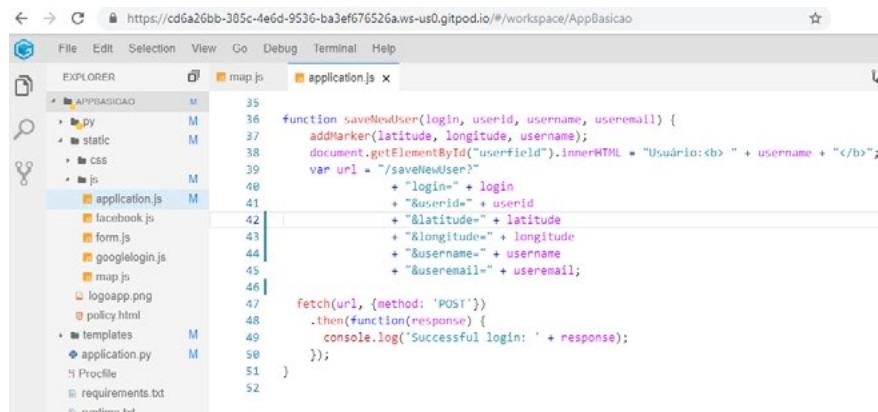
```
File Edit Selection View Go Debug Terminal Help
EXPLORER map.js x
1
2 var latitude = "";
3 var longitude = "";
4 var map = null;
5
6 getLocation();
7
8
9 function getLocation() {
10   if (navigator.geolocation) {
11     navigator.geolocation.getCurrentPosition(getLocationSuccess);
12   } else {
13     x.innerHTML = "Geolocation is not supported by this browser.";
14   }
15 }
16
17 function getLocationSuccess(position) {
18   latitude = position.coords.latitude;
19   longitude = position.coords.longitude;
20   initMap();
21 }
```

Ao iniciarmos o código do aplicativo, já podemos ir logo pegando na localização do usuário suas variáveis de latitude e longitude.

E então, junto com seus dados de login, já podemos salvar o log de acesso do usuário corrente no banco de dados.

A função javascript "saveNewUser" do arquivo "application.js" é que inicia esse processo de chamada fetch para o banco de dados no backend.

<https://github.com/allineo/AppBasicao/blob/master/static/js/application.js>



The screenshot shows a browser window with the URL <https://cd6a26bb-385c-4e6d-9536-ba3ef676526a.ws-us0.gitpod.io/#/workspace/AppBasicao>. The page displays a code editor for the file `application.js`. The code is a function named `saveNewUser` that takes parameters for login, userid, username, and useremail. It adds a marker to a map with the provided coordinates and then performs a POST request to a server endpoint at `/saveNewUser?` followed by the query parameters `&login=`, `&userid=`, `&latitude=`, `&longitude=`, `&username=`, and `&useremail=`. The response from the server is logged to the console.

```
function saveNewUser(login, userid, username, useremail) {
    addMarker(latitude, longitude, username);
    document.getElementById("userfield").innerHTML = "Usuário:<b> " + username + "</b>";
    var url = "/saveNewUser?";
    url += "&login=" + login;
    url += "&userid=" + userid;
    url += "&latitude=" + latitude;
    url += "&longitude=" + longitude;
    url += "&username=" + username;
    url += "&useremail=" + useremail;
    fetch(url, {method: 'POST'})
        .then(function(response) {
            console.log('Successful login: ' + response);
        });
}
```

# Mapa

Se tivermos informações de latitude e longitude de qualquer coisa, como por exemplo, pessoas, estabelecimentos, objetos em movimento ou não, etc, podemos plotar essa localização em um mapa e mostrá-lo em nosso app, caso essa funcionalidade seja relevante para o modelo de negócios de nossa ideia.

Por ser o mapa mais utilizado globalmente, como exemplo nesse tutorial, vamos utilizar o plugin de Mapas da Google para plotar as localizações que capturamos dos usuários que acessaram nosso app.

Vamos começar adicionando a Map Javascript API, da biblioteca da Google, ao seu arquivo "index.html":

<https://github.com/allineo/AppBasicao/blob/master/templates/index.html>

```
19 <script src="https://apis.google.com/js/platform.js" onload="renderButton" async defer></script>
20 <script src="https://maps.googleapis.com/maps/api/js?key=XXXXXXXXXXXXXXXXXXXXXX" async defer></script>
21
```

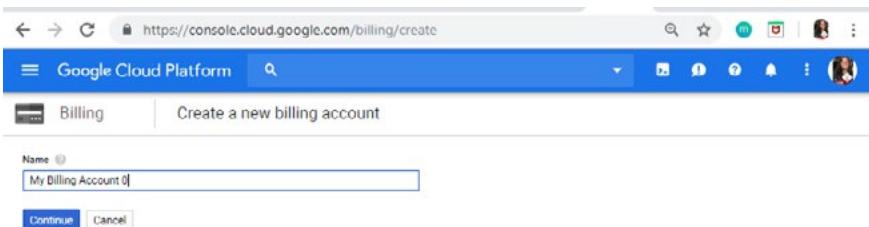
Agora, para realmente mostrar esse mapa na tela do nosso app e demarcar localizações nele, vamos precisar criar uma chave de autenticação na plataforma Google.

E ainda porém, para criarmos essa chave de autenticação, a Google exige que tenhamos um projeto em sua plataforma de Cloud e além disso, exige também que o "Billing" (autorização de cobrança) esteja habilitado nesse projeto, com um cartão de crédito válido e aceitável por eles.

*Então vamos começar esse processo criando uma "Billing account" para depois adicioná-la ao nosso projeto da Google Cloud.*

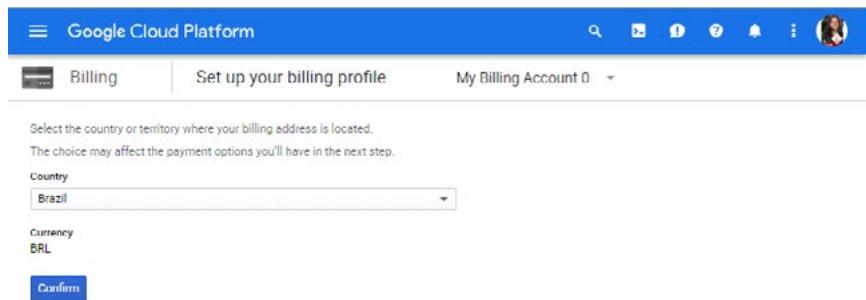
*Acesse o link abaixo:*

*<https://console.cloud.google.com/billing/create>*



*Deixe ou altere o nome da Billing Account, e clique em "Continue".*

*Na seção do país, selecione o correspondente ao seu cartão de crédito. Clique em "Confirm".*



*Na seção de "Payments Profile" informe todos os seus dados pessoais e de cartão de crédito para a Google. Sim, para disponibilizar o Google Maps em seu app você terá que ter um cartão de crédito válido e aceitável por eles.*

*Clique em "Confirm" novamente.*

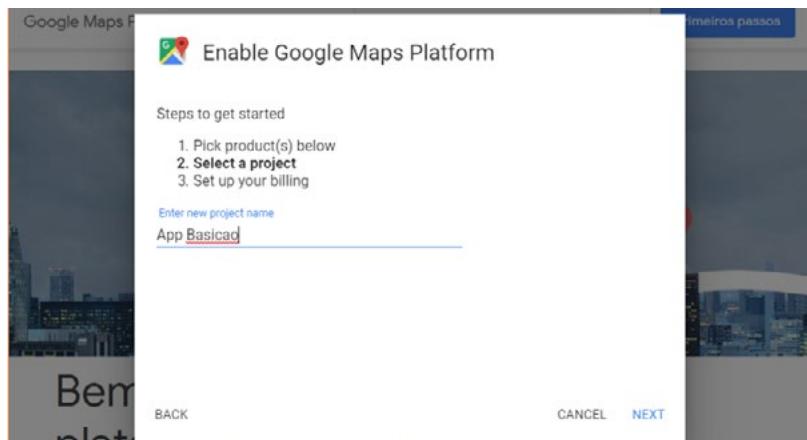
**Pronto, Billing configurado.**

**Agora acesse o link abaixo, selecione a opção "Maps" e clique em "Continuar":**

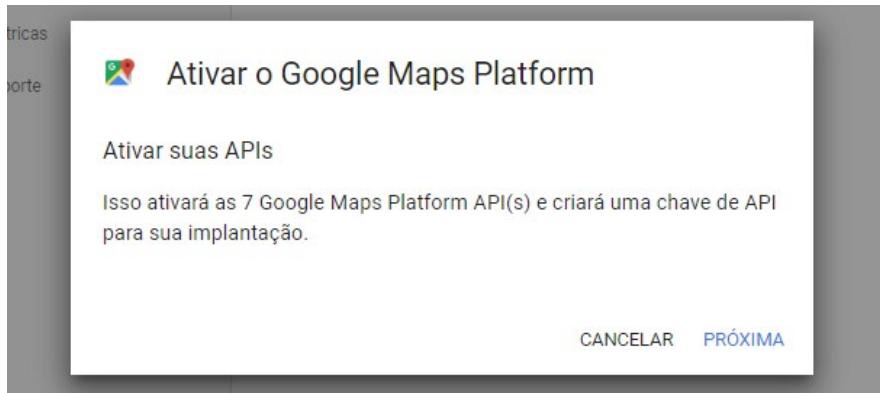
**<https://cloud.google.com/maps-platform/?hl=pt-br#get-started>**



**Na próxima tela, selecione o mesmo projeto criado no capítulo anterior do Google Sign-in e clique em "Next".**



*Depois selecione a "Conta de Faturamento" (ou "Billing Account") que você acabou de criar e clique em "Next". O console da plataforma Google Cloud se abrirá automaticamente na tela abaixo:*



*Clique no botão "Próxima" e sua "Key" aparecerá. Copie essa chave para substitui-la na chave da API do Google Maps do seu arquivo "index.html":*

```
19 <script src="https://apis.google.com/js/platform.js?onload=renderButton" async defer></script>
20 <script src="https://maps.googleapis.com/maps/api/js?key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" async defer></script>
21
```

#### **\*\* Observação \*\***

*Se por acaso precisar pegar a Maps API Key posteriormente, clique no link abaixo e busque a "API Key" com a data em que você a criou:*

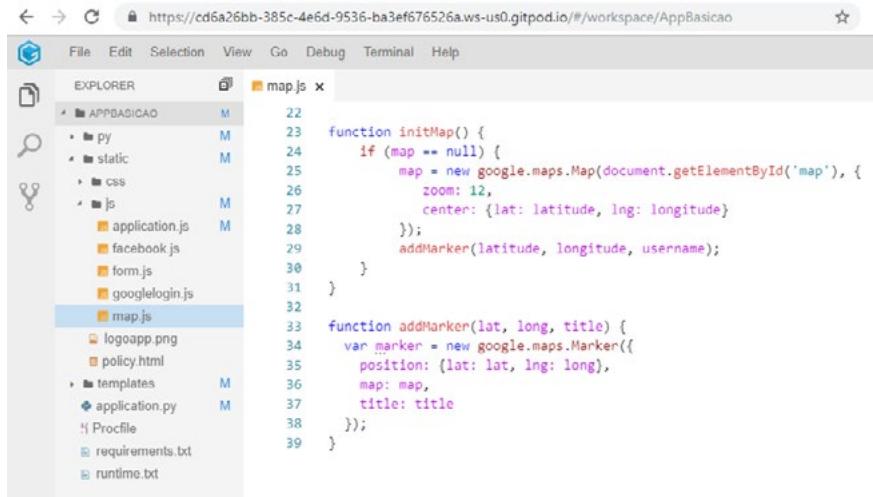
<https://console.cloud.google.com/apis/credentials>

*Ok! Superamos as credenciais.*

*Vamos agora finalmente adicionar o mapa ao nosso app!*

*O código para manipular o mapa encontra-se no arquivo "map.js":*

<https://github.com/allineo/AppBasicao/blob/master/static/js/map.js>



The screenshot shows a code editor interface with the URL <https://cd6a26bb-385c-4e6d-9536-ba3ef676526a.ws-us0.gitpod.io/#/workspace/AppBasicao>. The left sidebar shows a file tree for the 'APPBASICAO' directory, including files like py, static (css, js), templates, application.py, Procfile, requirements.txt, runtime.txt, logoapp.png, policy.html, and several JavaScript files (application.js, facebook.js, form.js, googlelogin.js, map.js). The right pane displays the content of the 'map.js' file:

```
22
23     function initMap() {
24         if (map == null) {
25             map = new google.maps.Map(document.getElementById('map'), {
26                 zoom: 12,
27                 center: {lat: latitude, lng: longitude}
28             });
29             addMarker(latitude, longitude, username);
30         }
31     }
32
33     function addMarker(lat, long, title) {
34         var marker = new google.maps.Marker({
35             position: {lat: lat, lng: long},
36             map: map,
37             title: title
38         });
39     }
```

*Observe a função "initMap()" para instanciar o mapa na sessão do usuário corrente.*

*E depois obeserve a função "addMarker()" para plotar os pinos no mapa.*

*Quer ver como ficou?!*

Então faça todo o processo de subir o código criado para o GitHub (commit e push no Gitpod).

Depois vá ao Heroku e faça o deploy clicando no botão "Deploy Branch".

E então acesse a url pública de seu app clicando no botão "View" quando o Heroku finalizar o deploy com sucesso.

Tchan nan !!! :-D



*BÔNUS*



## Amazon AWS

A AWS da Amazon é hoje a Cloud mais utilizada no mundo. Porém não é a mais simples.

Por isso vamos adicionar essa sequência de passos aqui de como fazer um APP em 1 dia, só que agora na cloud da Amazon.

Porém, já vamos começar com uma advertência muito importante. O serviço de cloud da Amazon, o Amazon Web Services (AWS), possui uma quota de utilização gratuita onde conseguimos fazer todo o nosso app básico.

NO ENTANTO, a cloud AWS exige que você cadastre um **cartão de crédito** para ativar essa conta e além disso, infelizmente, eles também NÃO ACEITAM qualquer tipo de cartão.

**Então, vamos começar!**

Comece criando sua conta sem custo no console da [AWS.com](https://aws.amazon.com/pt/console/):

The screenshot shows the AWS Management Console homepage at https://aws.amazon.com/pt/console/. The top navigation bar includes links for 'Entre em contato com o setor de vendas', 'Suporte', 'Português', 'Minha conta', 'Faça login no console', 'Produtos', 'Soluções', 'Definição de preço', 'Documentação', 'Aprenda', 'Rede de parceiros', 'AWS Marketplace', 'Explore mais', and a search bar. A sidebar on the left titled 'PRODUTOS E SERVIÇOS' lists 'AWS Console' (selected), 'Aplicativo móvel do Console da AWS', 'Perguntas frequentes', 'LINKS RELACIONADOS', 'Documentação', 'Artigos e tutoriais', 'Ferramentas de desenvolvedor', 'Conjuntos de dados públicos', 'Imagens de máquina da Amazon (AMIs)', 'Vídeos e webinars', and 'Novidades'. The main content area features a large orange header 'Console de Gerenciamento da AWS' and a central text block: 'Acesse e gerencie a Amazon Web Services através de uma interface de usuário simples e intuitiva, baseada na web. Você também pode usar o aplicativo móvel do AWS Console para visualizar rapidamente os recursos, em qualquer lugar.' To the right, there's a callout box with 'Comece a usar a AWS gratuitamente', a 'Crie uma conta gratuita' button, and another box with 'Ou faça login no console' and 'Receba doze meses ao nível de uso gratuito da AWS e aproveite os recursos do AWS Basic Support, como atendimento ao cliente 24x7x365 e fóruns de suporte, entre outros recursos.'

Nesta página inicial, clique no botão "Crie uma conta gratuita", digite seus dados pessoais e continue.

The screenshot shows the AWS sign-up page at https://portal.aws.amazon.com/billing/signup#/start. The top navigation bar includes 'Português'. The main content area has a heading 'Criar uma conta da AWS'. On the left, there's a section titled 'As contas da AWS incluem 12 meses de acesso ao nível gratuito' with sub-points about EC2, S3, and DynamoDB usage, and a link to the free terms. On the right, there's a form with fields for 'Endereço de e-mail', 'Senha', 'Confirmar senha', and 'Nome da conta da AWS'. Below the form is a 'Continuar' button and a link 'Fazer login com uma conta existente da AWS'. At the bottom, there's small print about copyright and terms.

**Quando chegar a hora, digite suas informações de cartão de crédito:**

The screenshot shows the 'Informações de pagamento' (Payment Information) section of the AWS billing setup. It includes fields for card number, expiration date, cardholder name, shipping address, and a radio button for 'User my shipping contact'. A yellow 'Enviar protegido' (Protected Send) button is at the bottom.

Digite as informações de pagamento para que possamos verificar sua identidade. Não haverá cobrança, salvo se você ultrapassar os [Limites do nível gratuito da AWS](#). Contra as [pessoas frequentes](#) para obter mais informações.

Número do cartão de crédito/débito  
Data de validade  
Nome do titular do cartão  
Endereço de pagamento  
 User meu endereço de contato  
 User um novo endereço  
Enviar protegido

© 2013 Amazon Web Services, Inc. ou suas afiliadas. Todos os direitos reservados.  
[Política de privacidade](#) | [Termos de uso](#) | [Ajuda](#)

**Ao passar dessa fase, você poderá então escolher a opção de plano da AWS. Selecione o "Plano Básico" clicando no botão "Gratuito". Será suficiente aqui para esse tutorial.**

The screenshot shows the 'Selecionar um plano de suporte' (Select a Support Plan) page. It lists three plans: 'Plano Básico' (Free), 'Plano Desenvolvedor' (Developer Plan), and 'Plano Business' (Business Plan). The 'Plano Básico' is highlighted with a yellow box and labeled 'Gratuito'. The 'Plano Desenvolvedor' starts at 'A partir de 29 USD/mês' and the 'Plano Business' starts at 'A partir de 100 USD/mês'. Each plan has a list of features.

A AWS oferece uma seleção de planos de suporte para atender às suas necessidades. Escolha um plano de suporte que se alinhe melhor com seu uso da AWS. [Saiba mais](#)

**Plano Básico** **Plano Desenvolvedor** **Plano Business**

**Gratuito** **A partir de 29 USD/mês** **A partir de 100 USD/mês**

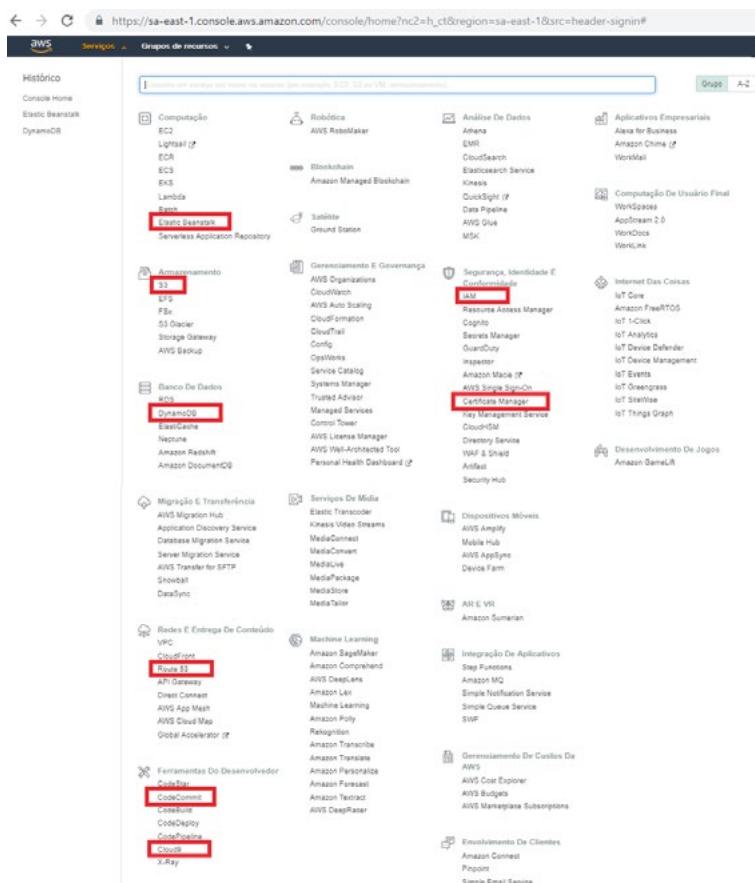
- Incluído com todas as contas
- Acesso por autotendimento 24 horas, todos os dias a fórum e recursos
- Verificações de melhores práticas para auxiliar a aprimorar a segurança e o desempenho
- Acesso a status de integridade e notificações
- Para adição precoce, testes e desenvolvimento
- Acesso por e-mail ao AWS Support durante o horário comercial
- 1 contato primário pode abrir um número ilimitado de casos de suporte
- Tempo de resposta de 12 horas para sistemas que não sejam de produção
- Para cargas de trabalho de produção e operações críticas para os negócios
- Acesso por e-mail, telefone e chat 24 horas, todos os dias ao AWS Support
- Contatos ilimitados podem abrir um número ilimitado de casos de suporte
- Tempo de resposta de 1 hora para sistemas de produção

Precisa de suporte de nível empresarial?  
Entre em contato com seu gerente de conta para obter informações adicionais sobre a execução de cargas de trabalho de negócios e de missão crítica na AWS (a partir de 15.000 USD/mês). [Saiba mais](#)

Nas próximas etapas, selecione a região em que gostaria de usar os recursos. No caso, sugiro escolher a nossa região nacional (América do Sul/São Paulo), para que suas informações armazenadas numa plataforma de cloud fiquem aqui pelo país.

E pronto, você terá então acesso aos vários serviços da Amazon.  
E eles são inúmeros!

Mas para facilitar, já deixamos indicado abaixo onde encontrar alguns dos serviços que utilizaremos aqui nesse tutorial e outros também interessantes:





O AWS Elastic Beanstalk é o serviço de hospedagem não gerenciável, ou seja o PaaS (Platform as a Service), da cloud da Amazon.

Lá no console da AWS.com, no menu de serviços, vá na primeira seção de "Computação" e clique no item "Elastic Beanstalk".

E no dashboard do Beanstalk vamos criar um novo aplicativo.

Na tela de novo aplicativo, digite o campo de nome e selecione a plataforma "Pré-configurado > Python". Também deixe a opção "Aplicativo de exemplo" selecionada e clique no botão "Criar aplicativo".

The screenshot shows the 'Criar um aplicativo Web' (Create New Application) page. At the top, there's a heading 'Criar um aplicativo Web'. Below it, a sub-section 'Informações do aplicativo' (Application Information) has a 'Nome do aplicativo' (Application Name) field containing 'appbasicao'. Under 'Configuração básica' (Basic Configuration), the 'Plataforma' (Platform) dropdown is set to 'Python'. The 'Código do aplicativo' (Application Code) section contains two options: 'Aplicativo de exemplo' (Example Application) which is selected, and 'Fazer upload do código' (Upload code). A file upload button 'Upload ZIP ou WAR' is visible. At the bottom of the form are three buttons: 'Cancelar' (Cancel), 'Configurar mais opções' (Configure more options), and 'Criar aplicativo' (Create application).

Após todo o ambiente necessário para a hospedagem de seu novo app ser criado, o dashboard abaixo aparecerá.

[Todos os aplicativos](#) > [AppBasico](#) > [Appbasico-env](#) (No ambiente: a-ryunusam, URL: [appbasico.sa-east-1.amazonaws.com](http://appbasico.sa-east-1.amazonaws.com))

[Ações](#)

**Painel**

**Configuração**

**Logs**

**Integridade**

**Monitoramento**

**Alertas**

**Atualizações gerenciadas**

**Eventos**

**Tags**

**Visão geral**

**Integridade**  
OK  
 [Causas](#)

**Verão em execução**  
Sample Application  
[Fazer upload e implantar](#)

**Configuração**  
Python 3.6 running on 64bit Amazon Linux 2.8.3  
[Alterar](#)

**Eventos recentes**

Tempo	Tipo	Detalhes
08-05-2019 22:05:34 UTC-0300	INFO	Successfully launched environment: appbasico-env
08-05-2019 22:05:34 UTC-0300	INFO	Application available at <a href="http://appbasico.sa-east-1.elasticbeanstalk.com">appbasico.sa-east-1.elasticbeanstalk.com</a>
08-05-2019 22:05:32 UTC-0300	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 4 seconds ago and took 2 minutes.
08-05-2019 22:04:49 UTC-0300	INFO	Waiting for EC2 instances to launch. This may take a few minutes.

[Mostrar todos](#)

Então, é só clicar na URL do ambiente criada automaticamente que se encontra no topo do dashboard.

Exemplo: <http://appbasicao.sa-east-1.elasticbeanstalk.com/>

E pronto! Você verá a tela da aplicação de exemplo implantada pela cloud da Amazon.

← → C ⓘ Not secure | appbasicao.sa-east-1.elasticbeanstalk.com

# Congratulations

Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud

## What's Next?

- AWS Elastic Beanstalk overview
- AWS Elastic Beanstalk concepts
- [Deploy a Django Application to AWS Elastic Beanstalk](#)
- [Deploy a Flask Application to AWS Elastic Beanstalk](#)
- Customizing and Configuring a Python Container
- [Working with Logs](#)

Agora vá no menu de "Configurações" do próprio Beanstalk, para que você possa visualizar as configurações que a plataforma gerou automaticamente para seu ambiente.

The screenshot shows the AWS Elastic Beanstalk configuration interface for the 'Teste-env' environment. The left sidebar lists navigation options: Painel, Configuração (selected), Logs, Integridade, Manutenção, Alertas, Atualizações gerenciadas, Eventos, and Tags. The main area is titled 'Visão geral da configuração' (General Configuration Overview) and contains several sections:

- Software**: Shows the application version (v1.0) and its deployment group (v1.0). It includes links to 'Visualizar' (View) and 'Modificar' (Modify).
- Instâncias**: Shows instance types (t2.micro), instance counts (1), and instance health (green). It includes links to 'Visualizar' (View) and 'Modificar' (Modify).
- Capacidade**: Shows capacity type (auto-scaling) and scaling mode (target tracking). It includes links to 'Visualizar' (View) and 'Modificar' (Modify).
- Segurança**: Shows security group (sg-012345678901234567) and IAM role (arn:aws:iam::123456789012:role/AmazonCloudWatchLogsRole). It includes links to 'Visualizar' (View) and 'Modificar' (Modify).
- Atualizações e implementações contínuas**: Shows deployment strategy (blue/green) and status (disponível). It includes links to 'Visualizar' (View) and 'Modificar' (Modify).
- Monitoreamento**: Shows CloudWatch Metrics and CloudWatch Logs integration status (ok). It includes links to 'Visualizar' (View) and 'Modificar' (Modify).
- Atualizações gerenciadas**: Shows managed updates status (desabilitado). It includes links to 'Visualizar' (View) and 'Modificar' (Modify).
- Notificações**: Shows notification status (desabilitado). It includes links to 'Visualizar' (View) and 'Modificar' (Modify).
- Teste**: Shows the environment's test status (não faz parte de teste CDT). It includes a 'Modificar' (Modify) link.
- Banco de dados**: Shows database connection details (MySQL, CLOUD\_DB, Amazon Aurora MySQL, MySQL 5.6). It includes a 'Modificar' (Modify) link.



Para a autenticação de usuário funcionar em nosso app é necessário usarmos a segurança do protocolo HTTPS.

Infelizmente a AWS não libera o protocolo HTTPS automaticamente junto com seus recursos e nem libera seus domínios internos para que nós o configuremos manualmente.

Na AWS é preciso ter um nome de domínio próprio para depois poder habilitar o protocolo HTTPS nessa cloud.

Então vamos a isso. Tenha bastante paciência, porque o processo de configuração de domínio próprio e protocolo HTTPS na AWS, é bem longo.

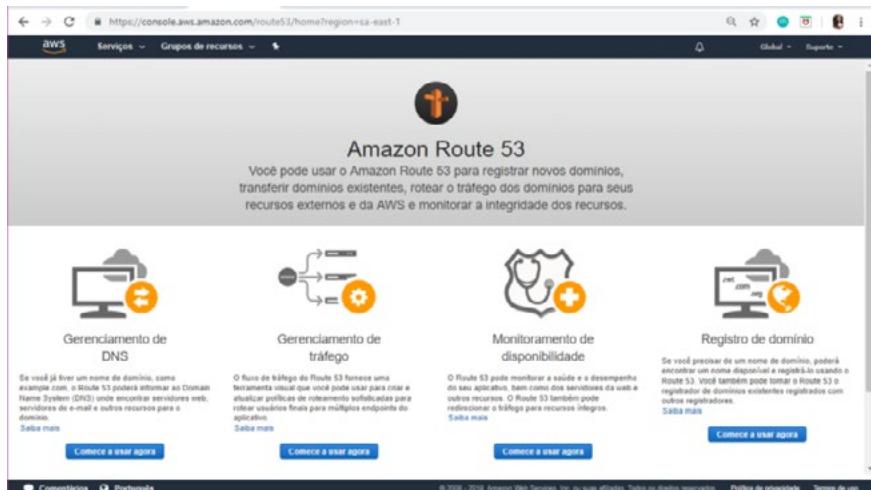
Para começar, se já não possuir, você precisa registrar um domínio próprio.

Existem inúmeras opções de registradores de nomes de domínio no mercado, como por exemplo a [GoDaddy.com](#), ou o [Registro.br](#).

No entanto, se desejar, a própria AWS também possui um registrador de nome de domínio que se chama "Route 53".

# Route 53

*Se você decidir pela facilidade de comprar o nome de domínio dentro da própria plataforma da AWS, vá na seção "Redes" dentro do menu de serviços e clique na opção "Route 53".*



*Clique no botão "Comece a usar agora" da última opção "Registro de domínio", e na próxima tela, clique no botão "Registrar domínio".*

*Digite o nome de domínio que deseja para seu app e clique no botão "Verificar" para saber se ele não já foi comprado por outra pessoa anteriormente.*

The screenshot shows the AWS Route 53 domain registration interface. In the search bar, 'appbasicao' is entered. Below it, a table lists domain suggestions:

Nome do domínio	Status	Preço / 1 Ano	Ação
appbasicao.com	Disponível - No caminho	\$12,00	Adicionar ao carrinho
apbaao.com	Available	\$12,00	Adicionar ao carrinho
apbaao.net	Available	\$11,00	Adicionar ao carrinho
appbasicao.info	Available	\$12,00	Adicionar ao carrinho
appbasicao.net	Available	\$11,00	Adicionar ao carrinho
ondbasicao.info	Available	\$11,00	Adicionar ao carrinho

To the right, a shopping cart summary shows 'appbasicao.com' selected for one year at \$12,00, with a subtotal of \$12,00. A note indicates monthly fees for DNS management.

*Se o nome de domínio escolhido para seu app estiver disponível para compra, clique então no botão "Adicionar ao carrinho" correspondente. Se não estiver disponível, tente outras opções de nome, como por exemplo as sugeridas pelo próprio registrador da AWS.*

*Após adicionar um nome de domínio disponível ao seu carrinho de compras, clique no botão "Continuar" no final da página.*

*Preencha então as informações de contrato e de pagamento e adquira o novo nome de domínio para seu app.*

Para a URL personalizada (ou nome de domínio) de seu app funcionar é necessário ainda roteá-la para a cloud AWS.

Volte à tela principal do Route 53, que se encontra na seção "Redes" dentro do menu de serviços da AWS, e clique no botão "Comece a usar" da primeira opção de serviços "Gerenciamento de DNS".

The screenshot shows the AWS Route 53 service management console. The left sidebar lists various services like Zonas hospedadas, Domínios, and Resolver. The main area has a large orange 'Create hosted zone' button. Below it, there's a brief description of what Route 53 does: it converts domain names into IP addresses. It also mentions that Route 53 is authoritative for domains in datacenters around the world, making it reliable, scalable, and fast. A 'Criar zona hospedada' button is located at the bottom right of the main content area.

Na tela que se abrirá, clique no botão "Criar zona hospedada" e depois no mesmo botão novamente na próxima tela.

Digite então o nome de domínio adquirido para seu app e clique no botão "Criar".

This screenshot shows the 'Create hosted zone' wizard in the AWS Route 53 console. The left sidebar is identical to the previous screenshot. The main area displays a message: 'Você não tem zonas hospedadas'. On the right, there's a form to create a new hosted zone. It asks for the 'Nome do domínio' (Domain name), which is filled with 'basicos.app'. There are fields for 'Commentário' (Comment) and 'Tipo' (Type), both currently set to their defaults. A note below explains that a public hosted zone makes your domain available on the Internet. At the bottom right of the form is a large blue 'Criar' (Create) button.

The screenshot shows the AWS Route 53 console with the URL <https://console.aws.amazon.com/route53/home?region=sa-east-1#resource-record-set/Z1291SGA7GK3GD>. The left sidebar shows 'Zonas hospedadas' with 'beanstack.app' selected. The main area shows 'Criar conjunto de registros' with two entries: 'beanstack.app' (NS) and 'basicstack.app' (SOA). The right panel shows the 'Editar conjunto de registros' dialog for 'beanstack.app' with 'TTL (segundos)' set to 172800 and 'Value' set to 'ns-2041.awsdns-03.uk.'. Buttons at the bottom include 'Salve conjunto de registros', 'Comentários', 'Partilhe', and links to 'Politica de privacidade' and 'Termos de uso'.

Dentro dos conjuntos de registros criados para seu nome de domínio, selecione a opção de tipo igual à "NS" (Name Service).

Agora você terá que ir até o site no qual comprou esse seu domínio e cadastrar esses nomes de serviços dados pela AWS.

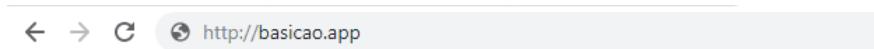
E finalmente, para o domínio funcionar, ainda falta redirecionar o domínio para o ambiente Beanstalk criando um registro A, um alias.

Na tela de "Zonas hospedadas" de seu nome de domínio dentro do "Route 53", clique no botão "Criar conjunto de registros".

No lado direito do dashboard, selecione o "Tipo": "A - Endereço IPv4", e no campo "Alvo do alias", selecione seu ambiente Beanstalk criado. Clique então no botão "Criar".

The screenshot shows the AWS Route 53 service console. On the left, there's a sidebar with various navigation links like Painel, Zonas hospedadas, Fluxo de tráfego, Domínios, Resolver, VPCs, and Regras. The main area is titled "Grupos de recursos" and shows a list of existing resource record sets. A modal window is open for creating a new "conjunto de registros". The modal has fields for "Nome do conjunto de registro" (basicao.app), "Tipo" (A - Endereço IPv4), and "Alias" (checkbox checked). It also contains a detailed description of what an alias is, how it works with CloudFront distributions, and examples of CNAME and website endpoints. At the bottom of the modal is a "Salvar conjunto de registros" button.

Pronto, teste agora no browser o direcionamento de seu nome de domínio para o seu ambiente Beanstalk.



# Billing

Infelizmente temos que conversar sobre um assunto chato aqui agora. Custo.

Na AWS a hospedagem de domínio próprio da NÃO É GRATUITA. Seu cartão de crédito será cobrado no final do mês se você utilizar esse serviço da Amazon.

Para acompanhar esses custos, vá para a seção de nome "Meu painel de faturamento" que se encontra no menu superior direito do nome de sua conta na AWS e depois clique no menu "Faturas" da lateral esquerda da tela. . Como na figura abaixo:

The screenshot shows the AWS Billing Home interface. On the left, a sidebar menu is open under the 'Faturamento' section, with 'Faturas' selected. The main content area displays a summary for June 2019. At the top right, there's a link to 'Minhas credenciais de segurança'. Below it, a table lists charges. A red box highlights the 'Amazon Route 53 HostedZone' entry, which costs \$0.50 per Hosted Zone for the first 25 Hosted Zones. The table also includes other service charges like CloudWatch, Data Transfer, and DynamoDB.

Serviço	Detalhes	Valor
Amazon Web Services, Inc. - Service Charges	Resumo de pagamento	\$80.67
Amazon Route 53 HostedZone	Amazon Route 53 DNS-Queries \$0.40 per 1,000,000 queries for the first 1 Billion queries	6,750 Queries \$2.00
	Amazon Route 53 Intra-AWS-DNS-Queries Queries to Alias records are free of charge	2,041 Queries \$0.00
	Simple Notification Service	\$0.00
	Simple Storage Service	\$0.00

O Route 53 irá cobrar 50 centavos de dólar (US\$0,50) por mês para hospedar seu domínio próprio.

# Certificado SSL

Após ter seu nome de domínio registrado dentro do Route 53 da AWS, você precisará configurar um certificado SSL para funcionar com o protocolo HTTPS junto a esse seu nome de domínio.

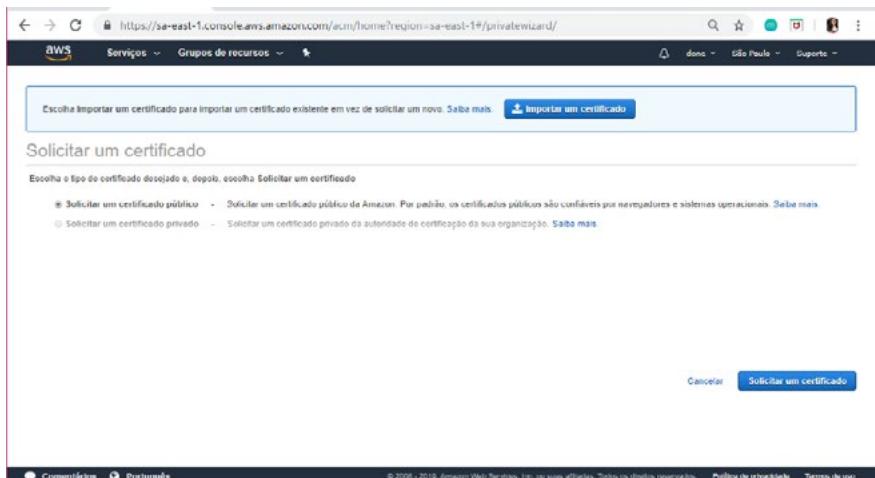
No menu serviços da AWS, na seção de "Segurança", abra a opção "Certificate Manager". Dentro da opção "Provisione certificados" (lado esquerdo) clique no botão "Comece a usar".

The screenshot shows the AWS Certificate Manager console at the URL <https://sa-east-1.console.aws.amazon.com/acm/home?region=sa-east-1#/firstrun/>. The interface is in Portuguese. It displays two main options for certificate provisioning:

- Provisione certificados**: This option is for users who want to provision certificates for their own websites or APIs. It includes a brief description: "Forneca o nome do seu site, estabeleça sua identidade e deixe o ACM fazer o resto. O ACM garante a renovação de certificados SSL/TLS emitidos pela Amazon ou por sua própria autoridade de certificação privada." Below this is a "Comece a usar" button.
- Autoridade de certificação privada**: This option is for users who want to establish a private CA infrastructure. It includes a brief description: "Você ou seu administrador da TI pode estabelecer uma infraestrutura gerenciada segura para emitir e revogar certificados digitais privados. Certificados privados identificam e protegem aplicativos, serviços, dispositivos e usuários dentro de uma organização." Below this is a "Comece a usar" button.

At the bottom right, there is a callout box titled "Região não compatível" with the message: "CAs privadas não estão disponíveis nesta região. Leia a documentação referenciada no link abaixo para conhecer as regras disponíveis." Below this is a link: "Tabela de regras disponíveis para CAs privadas".

**Na próxima tela deixe a opção "Solicitar um certificado público" selecionada e clique no botão "Solicitar um certificado".**



Escolha Importar um certificado para Importar um certificado existente em vez de solicitar um novo. [Saiba mais.](#) [Importar um certificado](#)

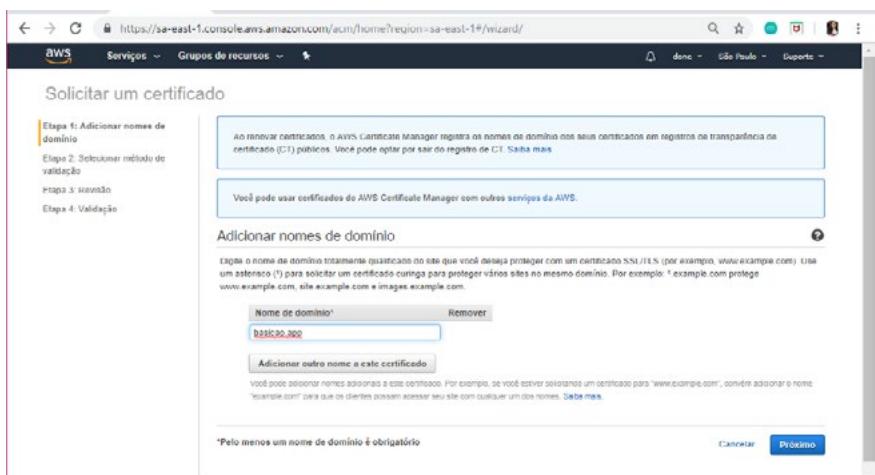
### Solicitar um certificado

Escolha o tipo do certificado desejado e, depois, escolha Solicitar um certificado

- Solicitar um certificado público** • [Solicitar um certificado público da Amazon](#). Por padrão, os certificados públicos são confiáveis por provedores e sistemas operacionais. [Saiba mais.](#)
- [Solicitar um certificado privado](#) • [Solicitar um certificado privado da autoridade de certificação da sua organização](#). [Saiba mais.](#)

[Cancelar](#) [Solicitar um certificado](#)

**Digite o seu nome de domínio próprio e clique no botão "Próximo".**



Etapa 1: Adicionar nomes de domínio

Etapa 2: Selecionar método de validação

Etapa 3: Resumo

Etapa 4: Validação

Ao renovar certificado, o AWS Certificate Manager registra os nomes de domínio dos seus certificados em registros de transparência na certificado (CNAME) públicos. Você pode optar por sair do registro de CNAME. [Saiba mais.](#)

Você pode usar certificados do AWS Certificate Manager com outros serviços da AWS.

### Adicionar nomes de domínio

Crie o nome de domínio totalmente qualificado no site que você deseja proteger com um certificado SSL/TLS (por exemplo, www.example.com). Use um asterisco (\*) para solicitar um certificado único para proteger vários sites no mesmo domínio. Por exemplo: \*.example.com protege www.example.com, site.example.com e images.example.com.

[Remover](#)

[Adicionar outro nome a este certificado](#)

Você pode adicionar nomes adicionais a esse certificado. Por exemplo, se você tiver solicitando um certificado para "www.example.com", também adicionar o nome "example.com" para que os clientes possam acessar seu site com qualquer um dos nomes. [Saiba mais.](#)

\*Pelo menos um nome de domínio é obrigatório

[Cancelar](#) [Próximo](#)

**Escolha o método de validação, por exemplo, e clique no botão "Revisão". Na próxima tela clique no botão "Confirmar e solicitar".**

The screenshot shows the 'Solicitar um certificado' (Request a certificate) wizard, Step 2: Select validation method. The 'Validation by DNS' option is selected. The page includes instructions about how ACM validates the domain and links to 'Next Step' and 'Review' buttons.

**Caso tenha selecionado validação por DNS, faça o download das configurações de CNAME do seu domínio clicando no link "Exportar configuração do DNS para um arquivo".**

The screenshot shows the 'Validation' step of the wizard. It displays a message about a pending validation request for 'basicapp'. Below this, it shows a table of validation records for 'basicapp' with one entry in progress. A download link for the DNS configuration file is provided, along with a 'Continue' button.

*Agora vá até o site onde você adquiriu seu nome de domínio e configure nas opções de DNS o seu CNAME exportado da AWS.*

*Depois volte para o Certificate Manager na AWS e clique no botão "Continuar" da tela de validação do seu certificado.*

*Se tudo estiver correto, depois de algum tempo a autenticidade de seu nome de domínio terá sido automaticamente validada pela AWS e seu certificado estará com o status de "Emitido".*

The screenshot shows the AWS Certificate Manager interface. On the left, there's a sidebar with 'Certificados' and 'Gerenciador de certificados'. The main area has a title 'Certificados' with a note about renewing certificates. Below it are buttons for 'Solicitar um certificado', 'Importar um certificado', and 'Ações'. A table lists a single certificate: 'basicos.app' (Status: Emisso, Issued by Amazon, Not in use, Not qualified). Under 'Status', it says the certificate was issued on 2019-05-16T14:02:08 UTC. Under 'Detalhes', it shows the type is 'Nome de domínio' (basicos.app), 'Em uso?' is 'Não', and 'Número de nomes' is '0'. The 'Solicitado em' date is 2019-05-16T13:53:54 UTC, and the 'Emitido em' date is 2019-05-16T14:02:09 UTC. The 'Validade' period ends on 2020-05-16T12:00:00 UTC.

# Load Balancer

Vamos voltar agora lá para o menu de "Configurações" do Beanstalk.

No primeiro item de "Computação" do menu de serviços da AWS, clique na opção "Elastic Beanstalk" e depois dentro da caixa verde de seu aplicativo, clique no link no nome do ambiente (environment).

Exemplo: link verde do nome Appbasicao-env

The screenshot shows the AWS Elastic Beanstalk console at the URL <https://sa-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=sa-east-1>. The interface includes a top navigation bar with links for Services, Groups of resources, and a search bar. Below this is a secondary navigation bar with links for Elastic Beanstalk and AppBasicao. The main content area displays a list of applications under 'Todos os aplicativos'. One application, 'AppBasicao' (highlighted with a green background), is selected, showing its details: Environment: 'Sample Web', Platform: 'Python 2.6 running on 64bit Amazon Linux 2.8.3', Version: 'Sample Application', Last modified: '16-05-2019 17:00:00', URL: 'AppBasicao-env.jkypf5m.sa-east-1.amazonaws.com', and Status of integrity: 'Ok'. On the left side, there's a sidebar with sections like 'Saiba mais', 'Em destaque', and 'Interface da linha de comando (v3)'. At the bottom, there are links for 'Comentários' and 'Português', and a footer with copyright information for Amazon Web Services and links for 'Política de privacidade' and 'Termos de uso'.

Ao abrir o painel de seu app, clique no menu "Configuração" da lateral esquerda da tela.

Então, dentro das várias opções de configuração do Beanstalk, clique no link "Modificar" da opção "Capacidade".

The screenshot shows the AWS Elastic Beanstalk configuration interface for an application named 'AppBasicao'. On the left sidebar, 'Configuração' is selected. In the main area, under 'Capacidade', there is a table with two columns: 'Instâncias' and 'Capacidade'. The 'Instâncias' column contains settings like 'Tipo de instância do EC2: t2.micro', 'ID da imagem do EC2: ami-0e883337740ee30e9', and 'Intervalo de monitoramento: 5 minutos'. The 'Capacidade' column contains settings like 'Tipo de ambiente: balançamento de carga, escalabilidade automática', 'Zonas de disponibilidade: Qualquer', and 'Instâncias: 1-1'. Below the table are 'Revisar alterações' and 'Aplicar configuração' buttons. The URL in the browser bar is https://sa-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=sa-east-1#/env.../AppBasicao-env.

Na seção "Grupo de Auto Scaling" das configurações de Capacidade, selecione o "Tipo de Ambiente" como "Carga balanceada".

Também altere o valor de "Instâncias" para Máx = 1. E não se esqueça de clicar no botão "Aplicar" no final da página.

Todos os aplicativos > AppBasico > Appbasico-env (ID do ambiente: e-systam77vg, URL: Appbasico-env.jkypfym.sa-east-1.elasticbeanstalk.com) Ações

Painel Configuração Logs Integridade Monitoramento Alertas Atualizações gerenciadas Eventos Tags

Modificar capacidade

Grupo de Auto Scaling Configure a capacidade computacional do seu ambiente e as configurações do Auto Scaling para otimizar o número de instâncias usadas.

Tipo de ambiente Carga balanceada

Instâncias Min: 1 Max: 1

Zonas de disponibilidade Qualquer Número de Zonas de disponibilidade (AZs) a serem usadas.

Posicionamento sa-east-1a, sa-east-1c Especifique as Zonas de disponibilidade (AZs) a serem usadas.

Desaquecimento da escalabilidade 300 segundos

Gatilhos de escalabilidade

Métrica NetworkOut Altere a métrica que é monitorada para determinar se a capacidade do ambiente está muito alta ou muito baixa

Estatística Média

De volta às opções de Configuração do Beanstalk, clique agora no botão "Modificar" da opção "Load Balancer".

Clique no botão "Adicionar Listener".

Todos os aplicativos > AppBasico > Appbasico-env (ID do ambiente: e-systam77vg, URL: Appbasico-env.jkypfym.sa-east-1.elasticbeanstalk.com) Ações Adicionar Listener

Painel Configuração Logs Integridade Monitoramento Alertas Atualizações gerenciadas Eventos Tags

Modificar load balancer

Balanceador de carga clássico Você pode especificar listeners para o load balancer. Cada listener direciona o tráfego de entrada do cliente em uma porta especificada usando um protocolo especificado para suas instâncias. Por padrão, configuramos seu load balancer com um servidor web padrão na porta 80.

Porta	Protocolo	Porta da Instância	Protocolo de Instância	Certificado SSL	Habilitar
80	HTTP	80	HTTP	—	<input checked="" type="checkbox"/>

Sessões As configurações a seguir permitem que você controle se o load balancer roteará solicitações da mesma sessão para a instância do Amazon EC2 com a menor carga, ou de forma consistente para a mesma instância.

Permanecer na sessão habilitada Duração do cookie 0 segundos

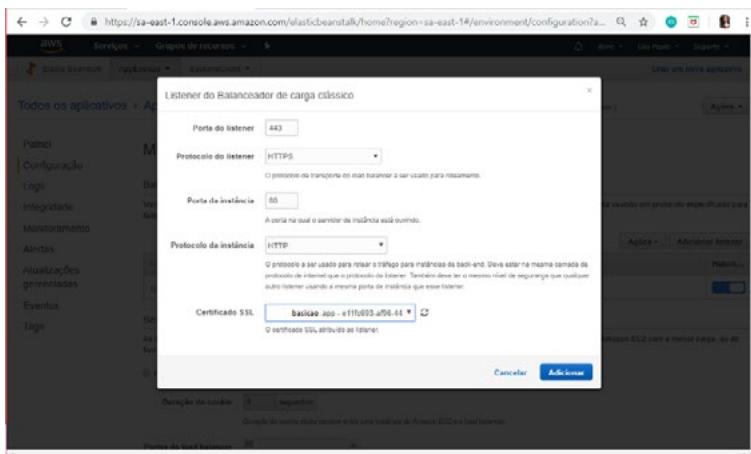
Curtação do cookie sticky session entre uma instância do Amazon EC2 e o load balancer.

Portas do load balancer 80 443

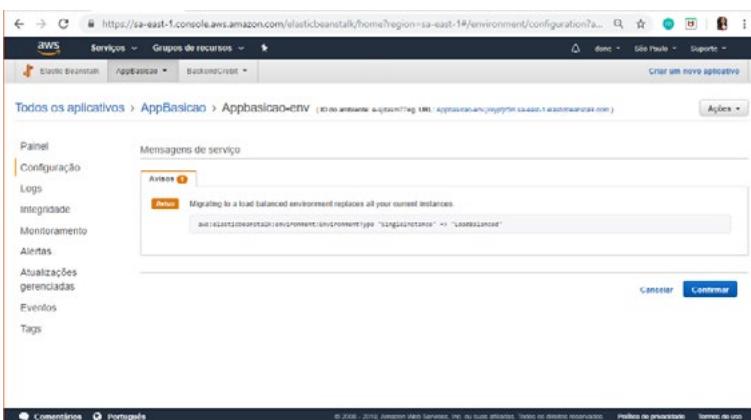
No popup, digite **443** para o número da "Porta do Listener" e selecione o "Protocolo de listener" como "HTTPS".

Depois corrija a "Porta da instância" para "80" e o "Protocolo da instância" para "HTTP".

Selecione o certificado SSL gerado para você e clique no botão "Adicionar".



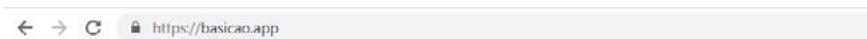
Na próxima tela clique em "Confirmar".



O protocolo HTTPS aparecerá na tela seguinte.

The screenshot shows the AWS Elastic Beanstalk interface for modifying a load balancer. On the left, a sidebar lists navigation options: Painel, Configuração (selected), Logs, Integridade, Monitoramento, Alertas, Atualizações gerenciadas, Eventos, and Tags. The main content area is titled "Modificar load balancer" and "Balanceador de carga clássico". It includes a note about specifying listeners for the load balancer. A table lists two listeners: port 80 (HTTP) and port 443 (HTTPS). The HTTPS row has a green button labeled "Criar novo listener" and a status bar indicating "Criação pendente". Below the table, a section titled "Sessões" discusses session persistence. At the bottom, there are buttons for "Próximo" and "Avançado".

Finalmente, agora teste seu nome de domínio no browser, dessa vez com o protocolo HTTPS.





O ambiente do Beanstalk cria e gerencia automaticamente, lá no ambiente **Elastic Compute Cloud (EC2)**, a máquina necessária ao funcionamento de nosso aplicativo.

Para ver o detalhamento dessa máquina, vá no menu "Serviços" do console, e selecione na primeira seção "Computação", a opção "EC2".

A screenshot of the AWS EC2 Dashboard. On the left sidebar, under the "INSTANCES" section, the "Instances" tab is selected. In the main content area, there's a summary table showing 1 Running Instances, 0 Dedicated Hosts, 1 Volumes, 0 Key Pairs, 0 Elastic IPs, 0 Snapshots, 1 Load Balancers, and 3 Security Groups. Below this, a "Create Instance" section has a "Launch Instance" button. To the right, there's an "Account Attributes" panel with options like "Supported Platforms", "VPC", "Default VPC", and "Resource ID length management". A "Learn more about the latest in AWS Compute from AWS re:Invent by viewing the EC2 Videos." link is also present.

E nesta tela selecione o link "Running Instances".

A screenshot of the AWS EC2 Instances page. The URL is https://sa-east-1.console.aws.amazon.com/ec2/v2/home?region=sa-east-1#instancessort:instancetype. The left sidebar shows the "Instances" section with the "Instances" tab selected. The main table lists one instance: "Instance ID: i-09234d5202bf8f8ee (CreateTime: 2019-05-22T10:48:55Z) Public DNS: ec2-10-220-23-85.sa-east-1.compute.amazonaws.com". The instance status is "running". The table includes columns for Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS (IPv4), IPv6 Public IP, IPv6 IPs, and Key Name. Below the table, detailed information for the instance is shown, including Platform (Amazon Linux 2), IAM role (aws-elasticbeanstalk-2018-03-05\_14-pyton3.6), and various network and security details.

# Mais Billing

Mais uma vez vamos ter que conversar sobre custo aqui nesse tutorial.

Na AWS seu cartão de crédito será cobrado no final do mês se você PASSAR DA COTA GRATUITA de "Load Balancing" no serviço de hospedagem da Amazon.

Para acompanhar esse custo, vá para a seção de nome "**Meu painel de faturamento**" que se encontra no menu superior direito do nome de sua conta na AWS, e depois clique no menu "Faturas" da lateral esquerda da tela. Como na figura abaixo:

Service	Description	Quantity	Unit	Cost
Amazon Elastic Compute Cloud	South America (Sao Paulo)			\$78.67
Amazon Elastic Compute Cloud	Amazon Elastic Compute Cloud running Linux/UNIX			\$30.73
Amazon Elastic Compute Cloud	\$0.00 per Linux t2.micro instance-hour (or partial hour) under monthly free tier			\$0.00
Amazon Elastic Compute Cloud	\$0.0198 per On Demand Linux t2.micro Instance Hour	750 Hrs		\$30.73
EBS		1,652 Hrs		\$0.00
EBS	\$0.00 per GB-month of General Purpose (SSD) provisioned storage under monthly free tier	28,700 GB-Mo		\$0.00
Elastic Load Balancing - Classic				\$47.94
Elastic Load Balancing - Classic	\$0.00 per GB Data Processed by the LoadBalancer under monthly free tier	0.094 GB		\$0.00
Elastic Load Balancing - Classic	\$0.00 per LoadBalancer-hour (or partial hour) under monthly free tier	750 Hrs		\$0.00
Elastic Load Balancing - Classic	\$0.034 per LoadBalancer-hour (or partial hour)	1,410 Hrs		\$47.94
Key Management Service				\$0.00
Route 53				\$2.00
Simple Notification Service				\$0.00
Simple Storage Service				\$0.00



## APP Basicão

Pronto. Seu ambiente AWS está pronto para receber seu app.

E, como já descrito anteriormente, nós já temos um app básico de exemplo com código Python pronto para você subir para o Elastic Beanstalk e testar seu funcionamento, que se encontra no repositório do GitHub do link abaixo:

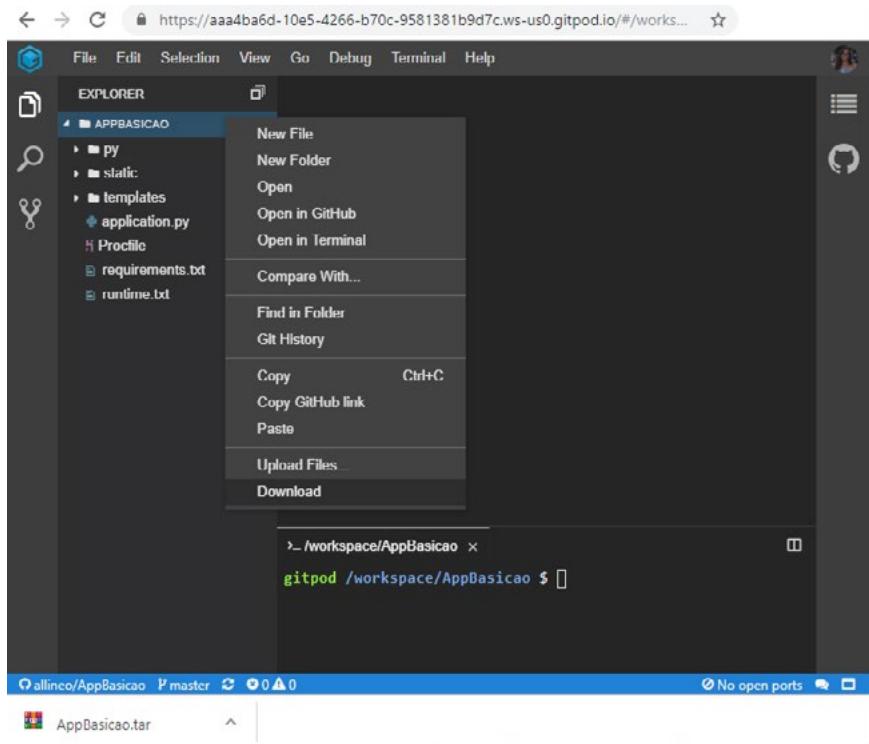
<https://github.com/allineo/AppBasicao.git>

Importe esse código de exemplo para o editor de cloud online [Gitpod.io](#) (como descrito anteriormente no início desse tutorial).

Resumidamente, o que você precisa é fazer seu cadastro na ferramenta e digitar na url do seu navegador o padrão abaixo:  
[https://gitpod.io/#https://github.com/url\\_do\\_repo\\_de\\_seu\\_app](https://gitpod.io/#https://github.com/url_do_repo_de_seu_app)

Faça então todas as alterações que julgar necessário para adaptar o app ao modelo de negócios da sua ideia.

E quando estiver pronto, faça o download de seu código fonte pelo Gitpod clicando com o botão direito do mouse na raiz de sua workspace.



## \*\* ATENÇÃO \*\*

O Gitpod gera um arquivo com a extensão .tar e o AWS Beanstalk necessita de arquivos com extensão .zip para fazer o upload do código. Faça o download do seu código no Gitpod, porém descompacte o arquivo .tar e compacte-o novamente com a extensão .zip.

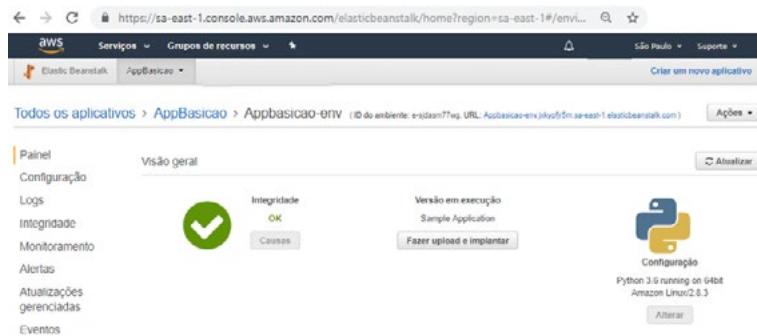
## \*\* Observação \*\*

O AWS Beanstalk necessita que os arquivos de configuração (requirements.txt) e de inicialização de seu app (o arquivo application.py) estejam necessariamente no diretório RAÍZ do arquivo compactado em formato .zip.

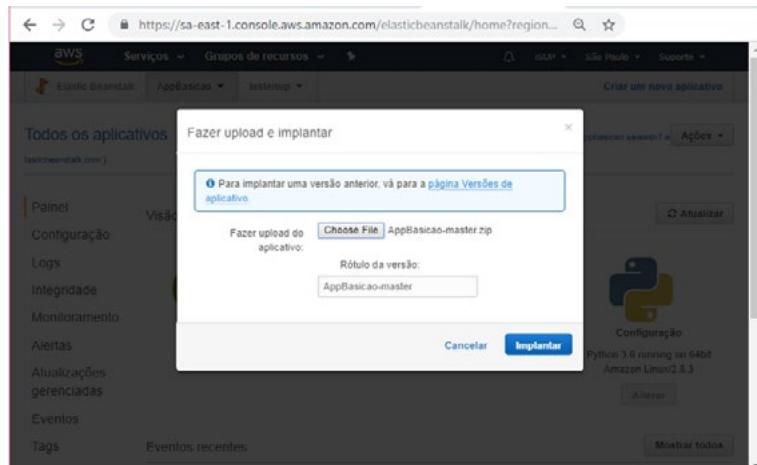
# Deploy

Agora voltemos para a tela do "Painel" de seu aplicativo na plataforma Beanstalk da AWS.

Clique no botão "Fazer upload e implantar".



Na tela pop-up, selecione o arquivo de seu código compactado com a extensão .zip, dê um nome único para a versão desse seu deploy e então clique no botão "Implantar".



Ao terminar a implantação, digite no navegador o seu nome de domínio.

Exemplo: <https://basicao.app>

**Sua URL pública e segura já poderá ser vista inclusive no seu celular!**



