

BEM VINDOS

DAY 01

FERRAMENTAS

Philipi P. Torres
philipi@litelims.com

 **LiteLiMS**
lims4all.com

A wooden desk with a cup of coffee, a notebook, a pen, and a smartphone.

O QUE É O...

SaaS

Software como serviço, do inglês Software as a Service (SaaS), é uma forma de distribuição e comercialização de software.

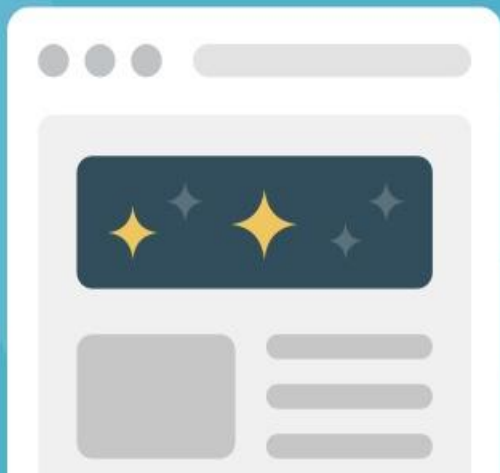
- >_ Benefícios
- >_ Desafios iniciais de um Software como Serviço
- >_ Fases de Crescimento de um Software como Serviço

O QUE É O... Código Fonte

Código-fonte (source code) é o conjunto de palavras ou símbolos escritos de forma ordenada, contendo instruções em uma das linguagens de programação existentes, de maneira lógica.

{ FRONT-END }

CSS
HTML
Javascript



VS

< BACK-END >

PHP
.NET
ExpressionEngine



{FRONT-END}

O desenvolvedor front-end é responsável por "dar vida" à interface. Trabalha com a parte da aplicação que interage diretamente com o usuário. Por isso, é importante que esse desenvolvedor também se preocupe com a experiência do usuário.



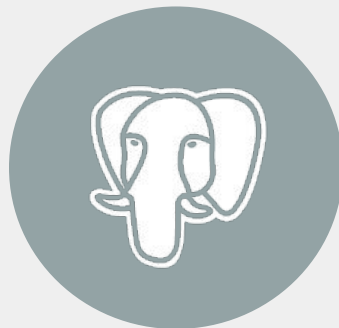
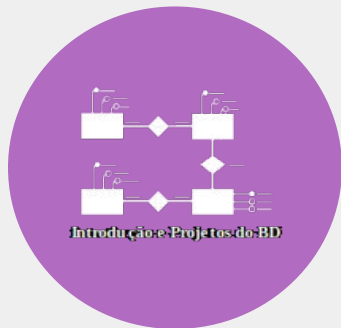
<BACK-END>

Como o nome sugere, o desenvolvedor back-end trabalha na parte de "trás" da aplicação. Ele é o responsável, em termos gerais, pela implementação da regra de negócio.



<BACK-END>

Para que o aprendizado em back-end seja ainda mais completo, é preciso ter conhecimento em banco de dados (ex: MySQL, SQL Server, PostgreSQL, entre outros). Normalmente o banco de dados é escolhido com base no que possuir mais recursos disponíveis na linguagem que será utilizada na aplicação.



Mobile



Qual linguagem escolher?

Para desenvolver nativamente na plataforma escolhida, você precisa aprender a programar com a linguagem utilizada nessa plataforma:

Android, iOS ou Windows Phone.

Android -> Aprenda Java

iOS -> Aprenda Swift ou Objectiv-c

Windows Phone -> Aprender C# ou Visual Basic

Nativo



Híbrido



WTF!? Qual é a diferença?

Aplicativos nativos são desenvolvidos com as linguagens de programação nativas do sistema operacional, oferecidas pelo seu Kit de desenvolvimento de software (*Em inglês, SDK – Software Development Kit*).

Já os aplicativos híbridos são desenvolvidos com linguagens de programação não nativas ao sistema operacional. Para isso, utiliza-se ferramentas (cross-platform ou multiplataforma) criadas especificamente para esse propósito.

WTF!? Qual é a diferença?

Entre as ferramentas mais conhecidos dessa categoria, destaca-se:

- > **_ Apache Cordova**
- > **_ Phonegap**
- > **_ Appcelerator**
- > **_ Ionic**
- > **_ Mobile Angular UI**
- > **_ E outros**

Se nada impossibilitar o desenvolvimento do app utilizando ferramentas multiplataformas (apps híbridos), não hesite!
Aprenda HTML, CSS e JavaScript!



Banco de Dados



Armazenando Dados

Obviamente, as aplicações móveis possuem recursos que permitem o armazenamento destes dados. Este que pode ser realizado de diferentes maneiras:

Armazenamento interno: dados privados;

Armazenamento externo: dados públicos e compartilhado;

Banco de dados: dados estruturados (ou não);

Sim, o armazenamento em **Banco de Dados** é a forma mais utilizada e, portanto, mais profissional. Sendo assim, é nesse momento que eu preciso dizer:



**POR
ONDE
COMEÇAR?**

Ferramentas





IDE

do inglês
**Integrated
Development
Environment.**

Onde armazena o código fonte?

GitHub é uma plataforma de hospedagem de código-fonte com controle de versão usando o **Git**. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo.



GitHub



É um sistema de controle de versão de arquivos. Através deles podemos desenvolver projetos na qual diversas pessoas podem contribuir simultaneamente no mesmo, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.



A close-up photograph showing two hands kneading a large mass of light-brown, textured dough. The hands are positioned on either side of the dough, with fingers pressing and pulling it. The dough has a crumbly, fibrous appearance. The background is a plain, light-colored surface.

MÃO NA MASSA!

Instalando o JDK

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html?ssSourceSiteId=otnpt>

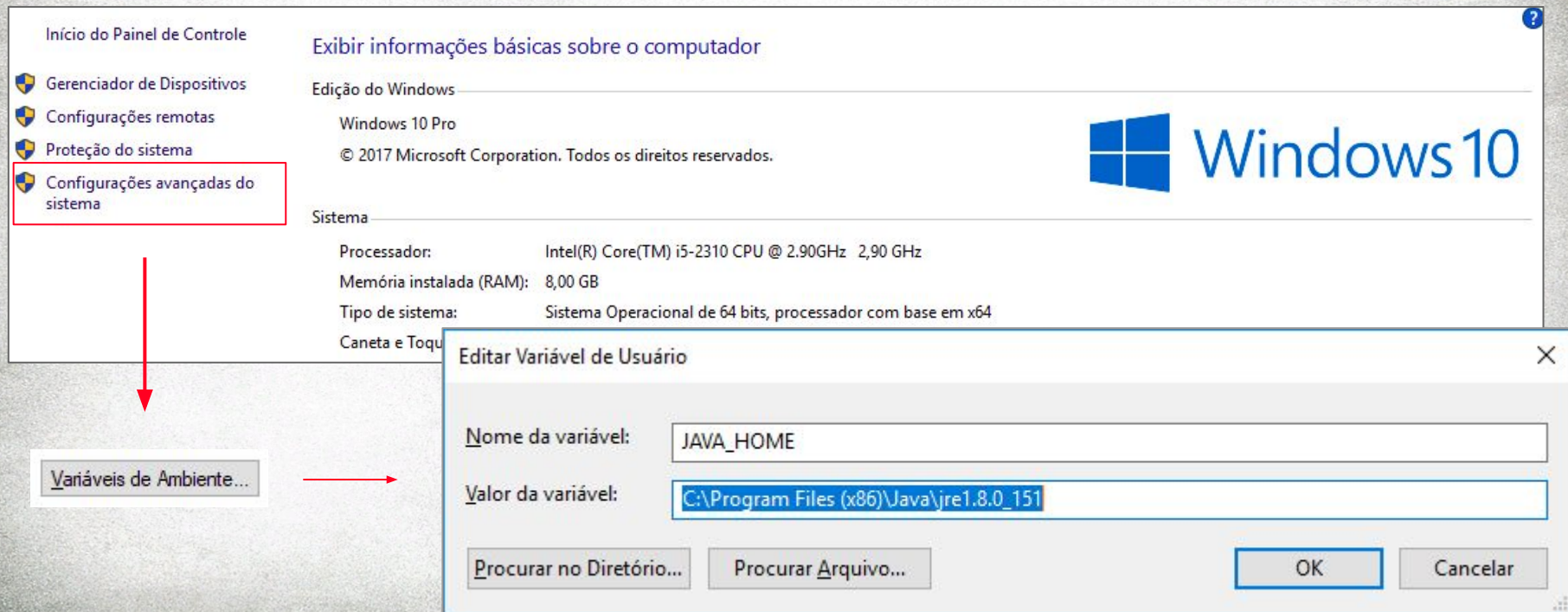
o JDK (Java Development Kit), pacote que inclui tudo o que é necessário para escrever aplicações e também o JRE para poder rodá-los após finalizá-los.

A instalação não tem segredos, é **Next, Next e Finish**.

Java SE Development Kit 8u161		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.92 MB	jdk-8u161-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u161-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.96 MB	jdk-8u161-linux-i586.rpm
Linux x86	183.76 MB	jdk-8u161-linux-i586.tar.gz
Linux x64	166.09 MB	jdk-8u161-linux-x64.rpm
Linux x64	180.97 MB	jdk-8u161-linux-x64.tar.gz
macOS	247.12 MB	jdk-8u161-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.99 MB	jdk-8u161-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u161-solaris-sparcv9.tar.gz
Solaris x64	140.57 MB	jdk-8u161-solaris-x64.tar.Z
Solaris x64	97.02 MB	jdk-8u161-solaris-x64.tar.gz
Windows x86	198.54 MB	jdk-8u161-windows-i586.exe
Windows x64	206.51 MB	jdk-8u161-windows-x64.exe

Config: JAJA_HOME

Para tal vá em **Meu Computador** e clique com o botão direito em **Propriedades**



Testando a instalação

Abra o prompt do DOS e digite: `java -version`.

Se tudo estiver certo irá aparecer a version do JRE no prompt do DOS

```
C:\Windows\system32\cmd.exe
Microsoft Windows [versão 10.0.15063]
(c) 2017 Microsoft Corporation. Todos os direitos reservados.

C:\Users\DEV - LiteLiMS>java -version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)

C:\Users\DEV - LiteLiMS>
```

Instalando o Git

msysgit.github.com

Basta entrar aí e baixar o software e dar um double click em cima do executável para começar a instalação.

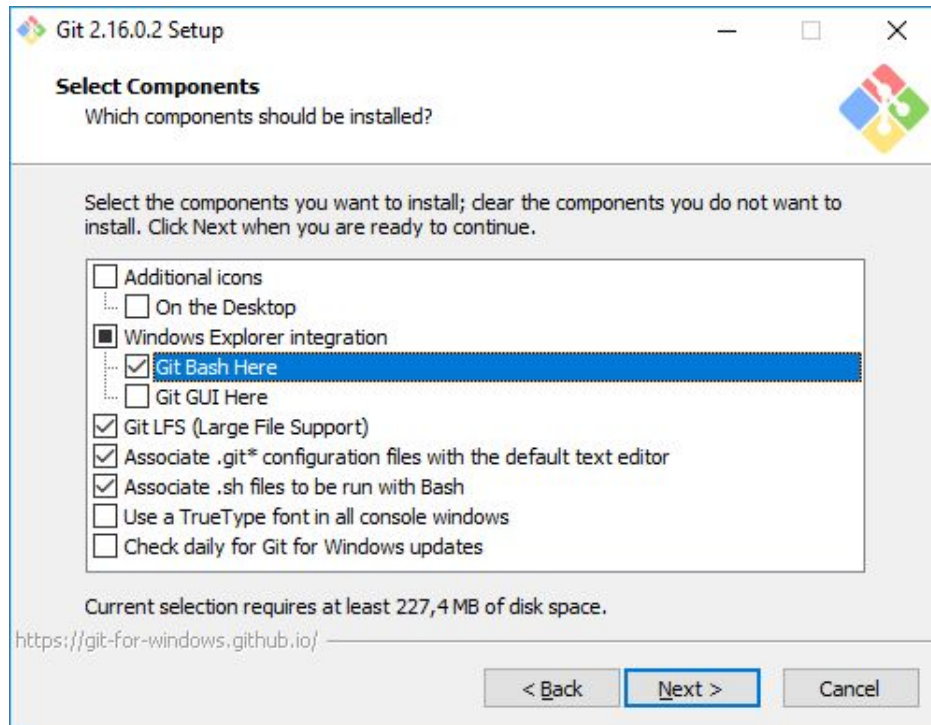


Instalando o Git

Aqui é simples: **Next, Next** até chegar a essa tela:

Deixe marcado a opção **Git Bash Here** **Here**.

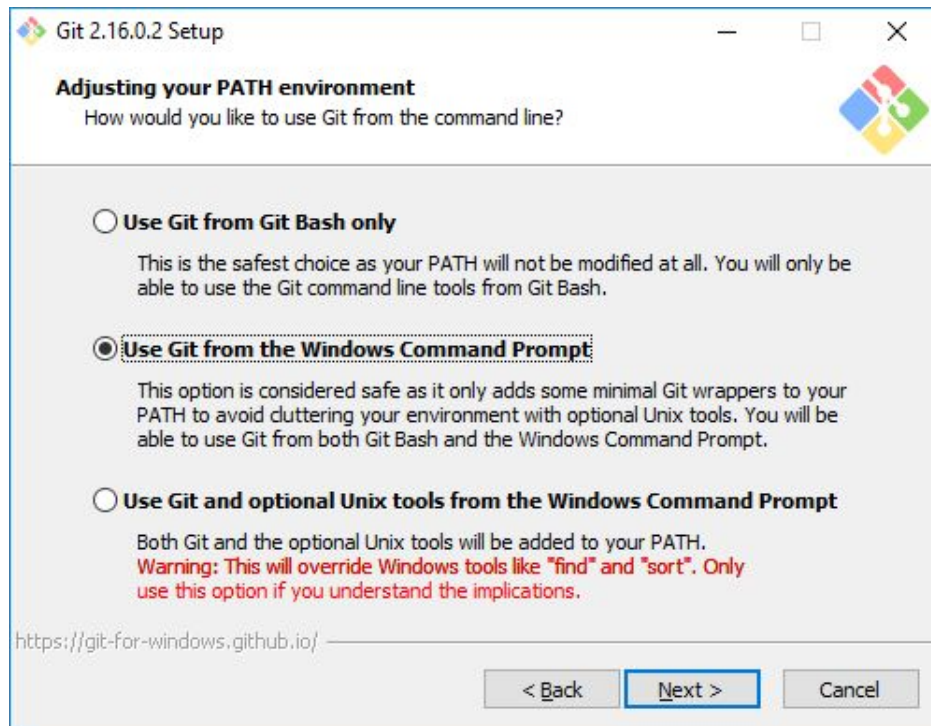
Com essa opção você poderá abrir o terminal do Git a partir de qualquer pasta em que estiver com o botão direito do mouse.



Instalando o Git

Ai vem o **Next, Next** de novo até essa tela:

Deixe marcado a opção **Use git from Command Prompt** para que consiga usar o Git a partir do CMD do Windows também, sem precisar abrir o Git Bash.

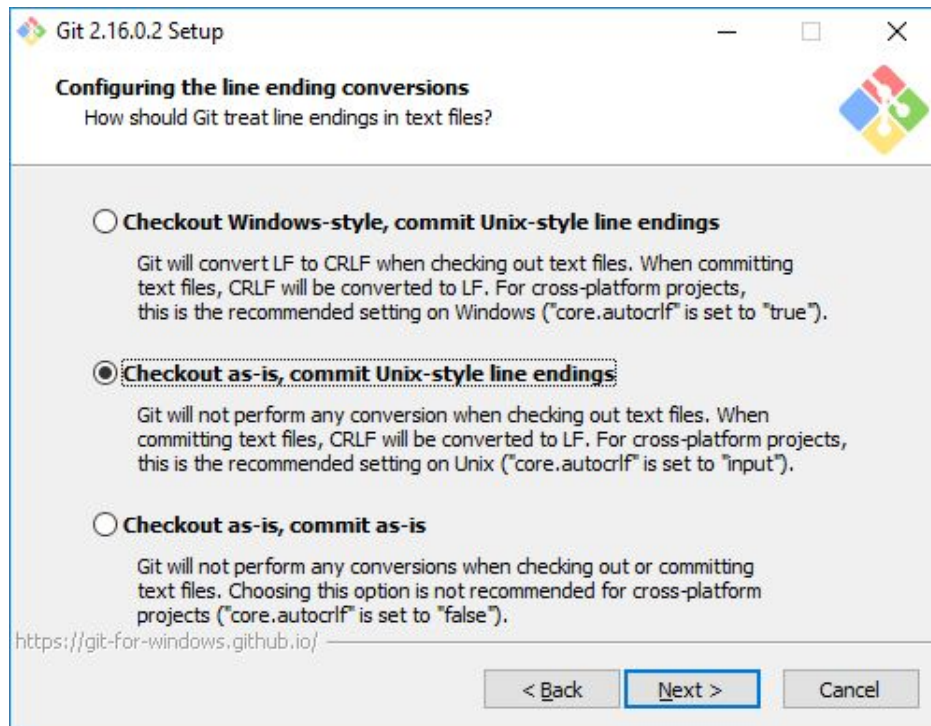


Instalando o Git

Next, Next até essa tela:

Selecione a segunda opção para que não existam conflitos entre as quebras de linha que o Windows coloca com as que os Sistemas Unix usam.

Next, Next e podemos então configurar.



Configuração do Git

```
MINGW64:/c/Users/DEV - LiteLiMS
```

```
DEV - LiteLiMS@DESKTOP-1QH HARU MINGW64 ~  
$ git  
usage: git [--version] [--help] [-C <path>] [-v  
[--exec-path[=<path>]] [--html-path  
[-p | --paginate | --no-pager] [--n  
[--git-dir=<path>] [--work-tree=<pa  
<command> [<args>]
```

These are common Git commands used in various

start a working area (see also: git help tutor
clone Clone a repository into a new dir
init Create an empty Git repository or

work on the current change (see also: git help
add Add file contents to the index
mv Move or rename a file, a director
reset Reset current HEAD to the specif
rm Remove files from the working tre

examine the history and state (see also: git h
bisect Use binary search to find the co
grep Print lines matching a pattern

```
MINGW64:/c/Users/DEV - LiteLiMS
```

```
DEV - LiteLiMS@DESKTOP-1QH HARU MINGW64 ~  
$ git config --global user.name "Philipi P. Torres"
```

```
DEV - LiteLiMS@DESKTOP-1QH HARU MINGW64 ~  
$ git config --global user.email "philipi@litelims.com"
```


```
DEV - LiteLiMS@DESKTOP-1QH HARU MINGW64 ~  
$ |
```


Criando uma conta no GitHub


Vamos até o site do GitHub (<https://github.com/join>) e crie uma conta.

Join GitHub

The best way to design, build, and ship software.

 **Step 1:**
Create personal account

 **Step 2:**
Choose your plan

 **Step 3:**
Tailor your experience

Create your personal account

Username

This will be your username. You can add the name of your organization later.

Email address

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

You'll love GitHub

- Unlimited collaborators
- Unlimited public repositories
- ✓ Great communication
- ✓ Frictionless development
- ✓ Open source community

Na hora de se cadastrar o site vai te oferecer uns planos pagos.

A diferença básica é que nos planos pagos você pode ter repositórios privados, coisa que não temos no gratuito.

Configurando o GitHub

1- Precisamos gerar uma chave SSH que seu computador vai usar pra se autorizar com o Github. Digite o seguinte comando no Git Bash:

```
ssh-keygen -t rsa -b 4096 -C "seu_email@dominio.com"
```

O resultado será:

```
Generating public/private rsa key pair.  
Enter file in which to save the key (/Users/you/.ssh/id_rsa): [Press enter]
```

Você quer salvar a chave nesse arquivo mesmo, só dê enter.

Configurando o GitHub

Depois, ele vai pedir uma senha:

```
Enter passphrase (empty for no passphrase): [Type a passphrase]  
Enter same passphrase again: [Type passphrase again]
```

Essa senha você vai ter que digitar toda vez que for baixar algo de um repositório ou enviar algo pra lá. Eu deixo sem mesmo.

Em seguida, você verá uma mensagem dizendo que deu tudo certo:

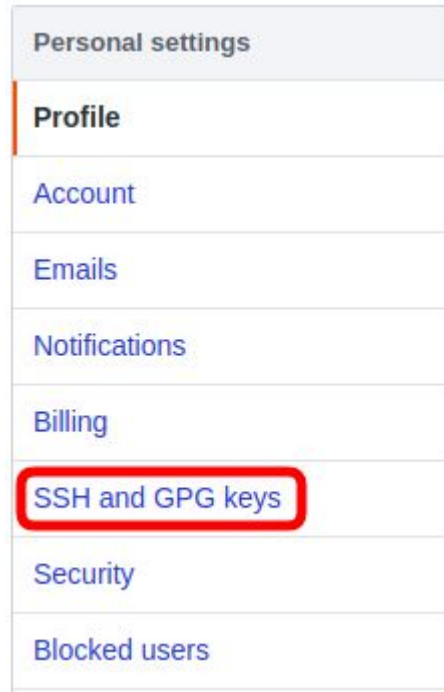
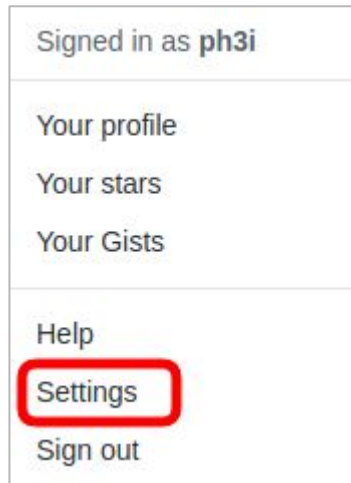
```
Your identification has been saved in /Users/you/.ssh/id_rsa.  
# Your public key has been saved in /Users/you/.ssh/id_rsa.pub.  
# The key fingerprint is:  
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db seuemail@dominio.com
```

Configurando o GitHub

3- Agora, vamos associar a chave que geramos à nossa conta do Github. Para copiar a chave do *bash*, digite o seguinte comando:


```
clip < ~/.ssh/id_rsa.pub
```

A chave agora está no nosso **ctrl+v** :P Abra o Github no navegador, faça o login e selecione a opção *settings* em seu menu:



Configurando o GitHub

SSH keys

A green rectangular button with rounded corners and a red border, containing the text "New SSH key".

New SSH key

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

Em *title*, coloque o nome que você quiser dar pra essa chave. Você pode colocar o nome que quiser, não faz diferença. Em key, **cole** a chave que você copiou do terminal com **ctrl-v**. Se não funcionar, volte uns passos acima e copie novamente. Clique em *Add Key*:

Fizemos tudo isso para configurar seu Git com a sua conta do Github. Vamos ver se funcionou?

Testando a configuração

Digite o seguinte no *Git Bash*:

```
ssh -T git@github.com
```

O resultado deve ser:

```
The authenticity of host 'github.com (207.97.227.239)' can't be established.  
# RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.  
# Are you sure you want to continue connecting (yes/no)?
```

Digite **yes** e aperte enter.

```
Hi username! You've successfully authenticated, but GitHub does not provide shell  
access.
```

#gogogo **Git** + **GitHub**

Abra o Github, faça login com a sua conta e clique em ***Start a project***

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#)

[Start a project](#)

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



ph3i

Repository name

zero2one

Great repository names are short and memorable. Need inspiration? How about [sturdy-carnival](#).

Description (optional)



Public

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



Private

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **None**

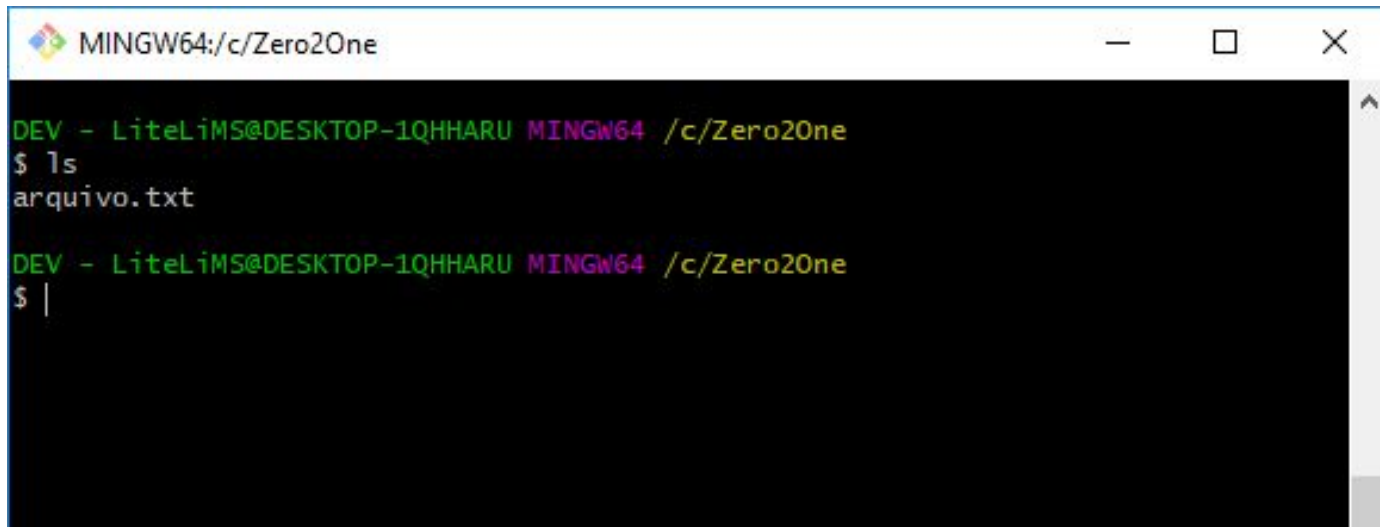


[Create repository](#)

Iniciando um repositório

Abra o Git Bash e vá até a pasta onde está o seu projeto.

Eu no caso vou até a minha pasta Zero2One, onde tenho um arquivo salvo no formato txt



```
MINGW64:/c/Zero2One

DEV - LiteLiMS@DESKTOP-1QH HARU MINGW64 /c/Zero2One
$ ls
arquivo.txt

DEV - LiteLiMS@DESKTOP-1QH HARU MINGW64 /c/Zero2One
$ |
```

Init: Repositório local

Agora, quero transformar essa pasta em um repositório Git.

Pra isso, basta digitar **git init** e dar enter:

```
$ git init
```

```
Initialized empty Git repository in c:/Users/Philipi/Zero20ne/.git/
```

Ele está falando que iniciou um repositório vazio nesta pasta. Como eu sei que agora tenho um repositório nesta pasta?

Se você abrir a pasta no File Explorer e configurar ele para mostrar os arquivos ocultos, verá que tem uma pasta a mais chamada .git:

Add: Origin

Vamos agora executar esse comando:

```
git remote add origin git@github.com:ph3i/zero2one.git
```

Basicamente o que estamos dizendo nessa linha de comando é:

"Git, esse meu repositório local se conectará com um remoto, o caminho dele(origin) é `git@github.com:ph3i/zero2one.git`, por favor, estabeleça essa comunicação pra mim".

Sending to GitHub

Quando estou dentro de uma pasta que é um repositório Git e quero saber o que eu fiz de alterações, eu digito o seguinte comando:

```
git status
```

Ele irá mostrar:

```
On branch master
```

```
Initial commit
```

```
Untracked files:
```

```
    (use "git add <file>..." to include in what will be committed)
```

```
    arquivo.txt
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

Sending to GitHub

Então:

```
git add arquivo.txt
```

Com esse comando, eu adicionei o meu arquivo ao Git.

Se eu der git status de novo:

```
$ git status
On branch master
Initial commit
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
    new file:   arquivo.txt
```


Sending to GitHub

Depois de ter adicionado o arquivo no controle de versões do Git.

```
git commit -m "aqui coloco uma mensagem, tipo: Meu primeiro commit"
```

Beleza, comitado.

Agora quero enviar isso para o meu repositório remoto:

```
git push -u origin master
```

Para baixar do repositório do GitHub para o repositório local

```
git pull -u origin master
```

```
def eat_brain  
    increment(:brains_eaten, 1)  
end
```

ARE YOU
READY?



Philipi P. Torres
philipi@lifelims.com



CONSIDERAÇÕES
FINAIS



**Simplicidade é o último
grau de sofisticação.**

- Leonardo Da Vinci