



## Exercício Programa – Design de Software 2019.1

### “Escape Insper!”

Data de entrega: **22/04/2019, via GitHub**

Grupos: duplas ou trios. O critério de avaliação muda conforme o tamanho da equipe.

#### Introdução

*Ticking away the moments that make up a dull day  
Fritter and waste the hours in an offhand way.  
Kicking around on a piece of ground in your home town  
Waiting for someone or something to show you the way.*

*Tired of lying in the sunshine staying home to watch the rain.  
You are young and life is long and there is time to kill today.  
And then one day you find ten years have got behind you.  
No one told you when to run, you missed the starting gun.*

*Pink Floyd -- Time  
Dark Side of the Moon, 1973*

O tempo passa muito rápido. Quando mal a gente percebe... já é hora de entregar o EP!  
E agora?

#### Text adventures

Neste exercício-programa vocês desenvolverão um jogo estilo *text adventure*, às vezes chamado também de *interactive fiction*. Estes jogos eram muito populares no início da computação pessoal, nos anos 80. Consistem geralmente no seguinte:

- Um cenário (descrito em texto) é apresentado ao jogador.
- Algumas opções de ação são também apresentadas, se existirem. Caso não existam opções de ação, o jogo termina.
- O jogador escolhe qual ação realizar.
- O jogo então evolui para um novo cenário.

Versões gráficas deste tipo de jogo se tornaram bem populares no final dos anos 80 e começo dos anos 90. Clássicos desta era incluem “*Maniac Mansion*”, “*Day of the Tentacle*”, “*Full Throttle*”, “*The Secret of Monkey Island*”, “*Sam & Max*”, entre tantos outros. Recentemente o *Netflix*, para capitalizar na nostalgia anos 80, lançou um filme interativo chamado *Bandersnatch* onde o personagem principal é um programador de *interactive fiction* nos anos 80.

# Insper

## Escape Insper!

Você deverá desenvolver um *text adventure* onde o objetivo (pelo menos no começo) é tentar um adiamento na entrega do EP. O jogo começa no saguão de entrada do Insper, e você irá navegar pelas opções de jogo para tentar atingir seu objetivo.

O EP deverá ser desenvolvido em etapas, conforme será descrito neste documento mais adiante.

Aqui está um exemplo de uma partida deste jogo:

Na hora do sufoco!

-----

Parecia uma boa ideia: vou só jogar um pouquinho/assistir Netflix/embaçar em geral. Amanhã eu começo o EP. Mas isso não deu certo...

É o dia de entregar o EP e você está muuuuito atrasado! Você está na entrada do Insper, e quer procurar o professor para pedir um adiamento do EP (boa sorte...)

Qual é o seu nome? **Avatar**

Saguão do perigo

-----

Você está no saguão de entrada do Insper  
Escolha sua opção:

andar professor: Tomar o elevador para o andar do professor  
biblioteca: Ir para a biblioteca

O que você quer fazer? **biblioteca**

Caverna da tranquilidade

-----

Você está na biblioteca  
Escolha sua opção:

início: Voltar para o saguão de entrada

O que você quer fazer? **inicio**

Um veterano selvagem apareceu!

Avatar: 100 hit points, 12 pontos de ataque e 7 pontos de defesa  
veterano: 10 hit points, 2 pontos de ataque e 4 pontos de defesa  
O que você deseja fazer: 'lutar' ou 'fugir'? **lutar**  
Avatar atacou e causou 10 de dano!  
Você venceu o veterano! Yay!

Saguão do perigo

-----

Você está no saguão de entrada do Insper  
Escolha sua opção:



```
andar professor: Tomar o elevador para o andar do professor
biblioteca: Ir para a biblioteca

O que você quer fazer? andar professor

Andar do desespero
-----
Você chegou ao andar da sala do seu professor
Escolha sua opção:

início: Tomar o elevador para o saguão de entrada
professor: Falar com o professor

O que você quer fazer? professor

O monstro do Python
-----
Você foi pedir para o professor adiar o EP. O professor revelou que é um monstro
disfarçado e devorou sua alma.
Acabaram-se suas opções! Mwo mwo mwooooo...
Você morreu!
```

## Definição do exercício-programa

Este exercício-programa tem duas fases de desenvolvimento:

- Fase 1: vocês irão iniciar o seu desenvolvimento de código. Nesta fase vocês aprenderão sobre o uso de um sistema de controle de versão muito popular chamado Git. Deverão também desenvolver separadamente duas pequenas *features* do jogo, para mostrar que sabem trabalhar independentemente com Git (e com Python também, claro!).
- Fase 2: Vocês irão aperfeiçoar o jogo.

Os objetivos de aprendizado e as correspondentes rubricas de avaliação são os seguintes:

Objetivo de aprendizado	Conceito atingido				
	I	D	C	B	A
Desenvolvimento em Python	Não conseguiu fazer nem ao menos as tarefas básicas da fase 1	Fez as tarefas básicas e mais nenhuma.	Implementou corretamente as tarefas básicas e pelo menos uma <i>feature</i>	Implementou corretamente as tarefas básicas e pelo menos três <i>features</i>	Implementou corretamente as tarefas básicas e pelo menos cinco <i>features</i>

Objetivo de aprendizado	Conceito atingido				
	I	D	C	B	A
Trabalho em equipe e uso do git	Não conseguiu usar o git	Conseguiu cumprir as tarefas básicas da forma requerida, e em seguida entregou o código em um <i>commit</i> só ou em poucos commits. Não demonstrou que ambos os alunos contribuíram de modo substancial.	O balanço de código desenvolvido é muito inclinado para um dos alunos OU várias features no mesmo <i>commit</i> .	N/A	Balanço adequado de esforços, <i>commits</i> contínuos
Qualidade do trabalho	Entregou um repositório com arquivos sem sentido E código mal feito, cheio de código inútil e ilegível ("lixo")	Entregou um repositório com arquivos sem sentido OU (exclusivo) código mal feito, cheio de código inútil e ilegível ("lixo")	Código poderia ser melhor estruturado e documentado, mas pelo menos não contém "lixo"	Código com estruturação e documentação razoável, mas poderia ser melhor modularizado. Código segue padrão consistente de escrita ("coding standards" e.g. PEP8)	Código bem estruturado e modularizado, bem documentado, e consistente em termos de padrões de escrita.

A nota do projeto será dada pelo seguinte:

- Se a equipe obteve conceito I ou D em algum objetivo de aprendizado, o conceito final será dado pelo menor conceito obtido. Por exemplo: se todas as *features* foram implementadas, e o código está impecável, mas um aluno fez tudo (conforme evidenciado no git), a nota será D (ou até mesmo I se nem a Fase 1 foi seguida corretamente).
- Se todos os conceitos forem iguais ou superiores a C, o conceito final será obtido pela média aritmética dos conceitos obtidos para os objetivos de aprendizado.
- Se a equipe contiver 3 integrantes, todos os conceitos do objetivo de aprendizado "Desenvolvimento em Python" recuam 1 nível (ou seja, o que antes representava C agora é D). Para atingir o conceito A o grupo deve desenvolver pelo menos duas *features* livres, além do antigo conceito A.

# Insper

## Fase 1: Começando o desenvolvimento

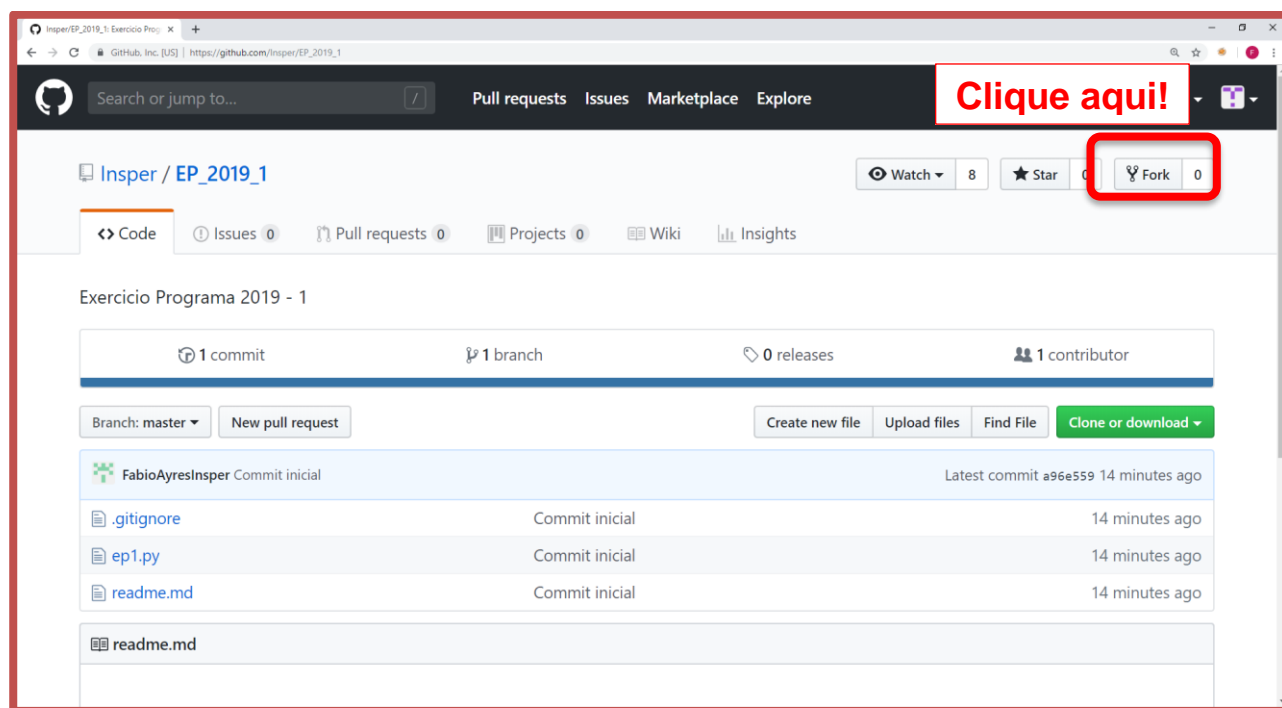
### Iniciando o repositório

Nesta fase vocês irão se familiarizar com uma ferramenta essencial para o desenvolvimento de software em equipe: o sistema de controle de versão **git**.

Leia a apostila de github “Controle de versão.pdf”. Neste momento os dois alunos devem criar suas contas no GitHub.

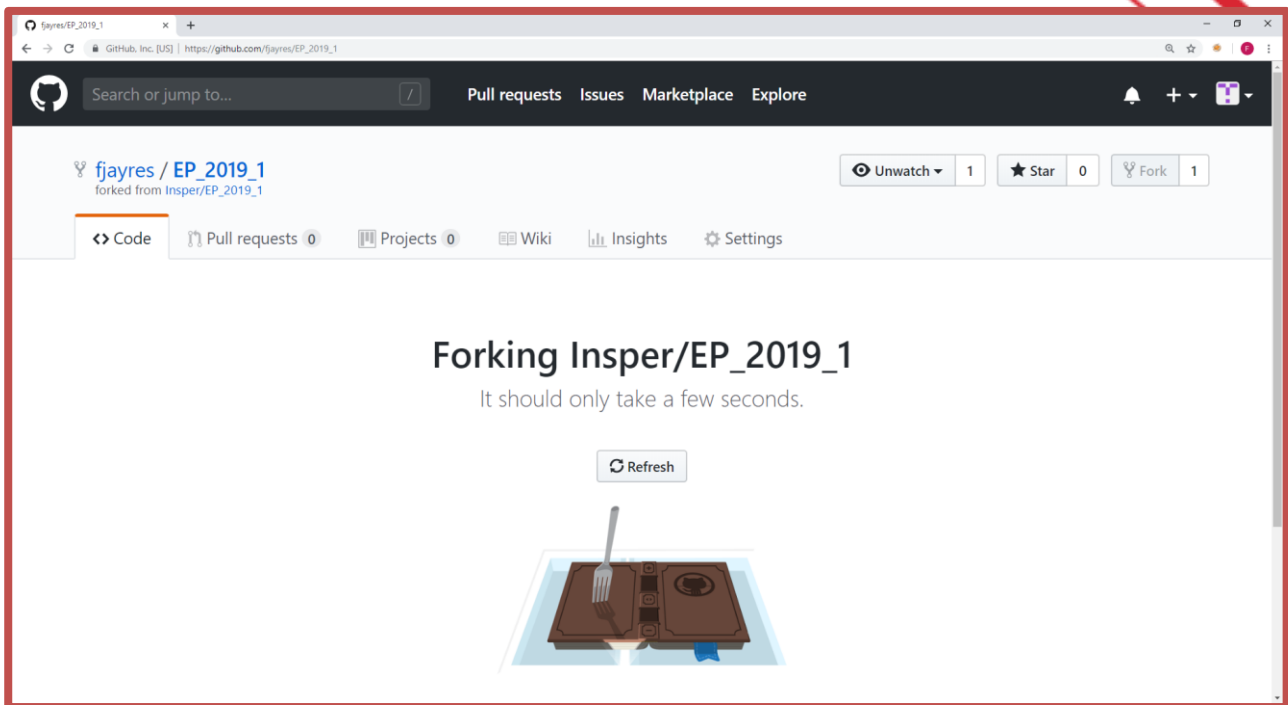
Agora um dos membros da equipe deverá fazer um *fork* do repositório do professor localizado em [https://github.com/Insper/EP\\_2019\\_1](https://github.com/Insper/EP_2019_1). Um *fork* é uma cópia independente do projeto original. Para fazer um *fork* siga esses passos:

- Faça login na sua conta do GitHub.
- Navegue para [https://github.com/Insper/EP\\_2019\\_1](https://github.com/Insper/EP_2019_1). Você deverá a página a seguir. Clique no botão “Fork”

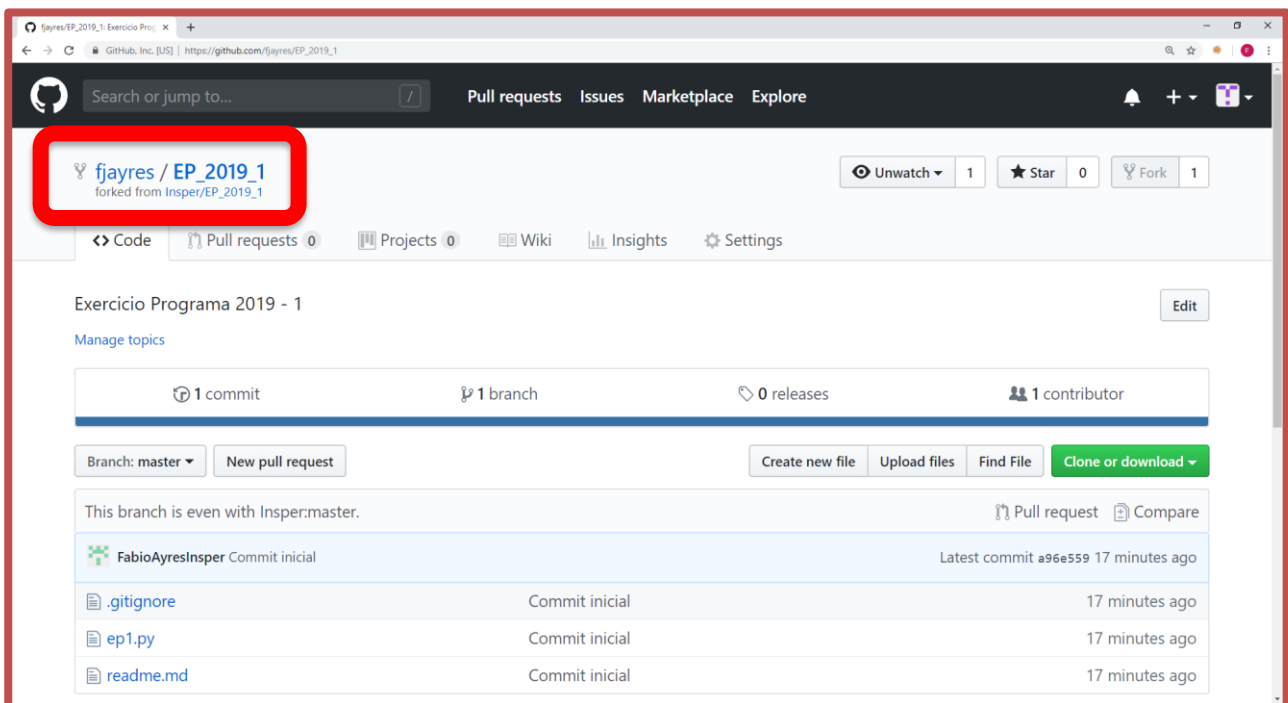


A seguinte página aparecerá:

# Insper



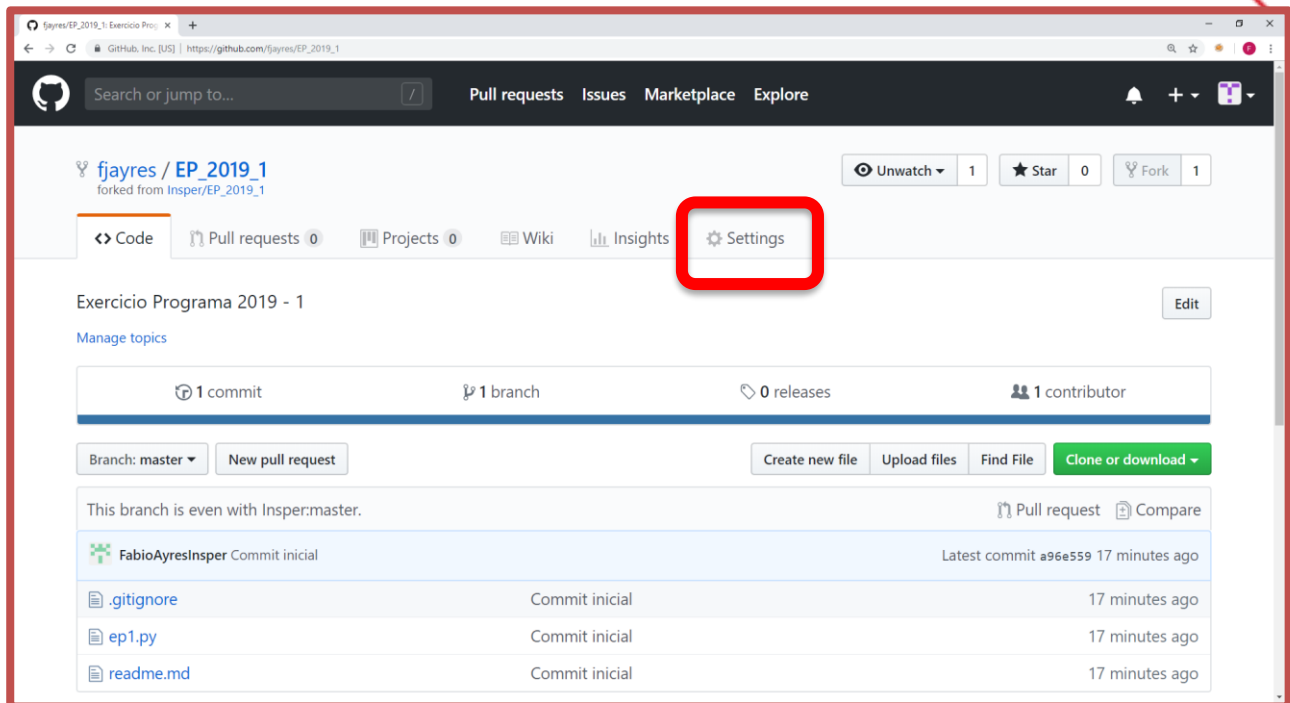
E finalmente aparecerá a página da sua cópia independente (seu *fork*) do projeto (veja no destaque que se trata de um *fork*):



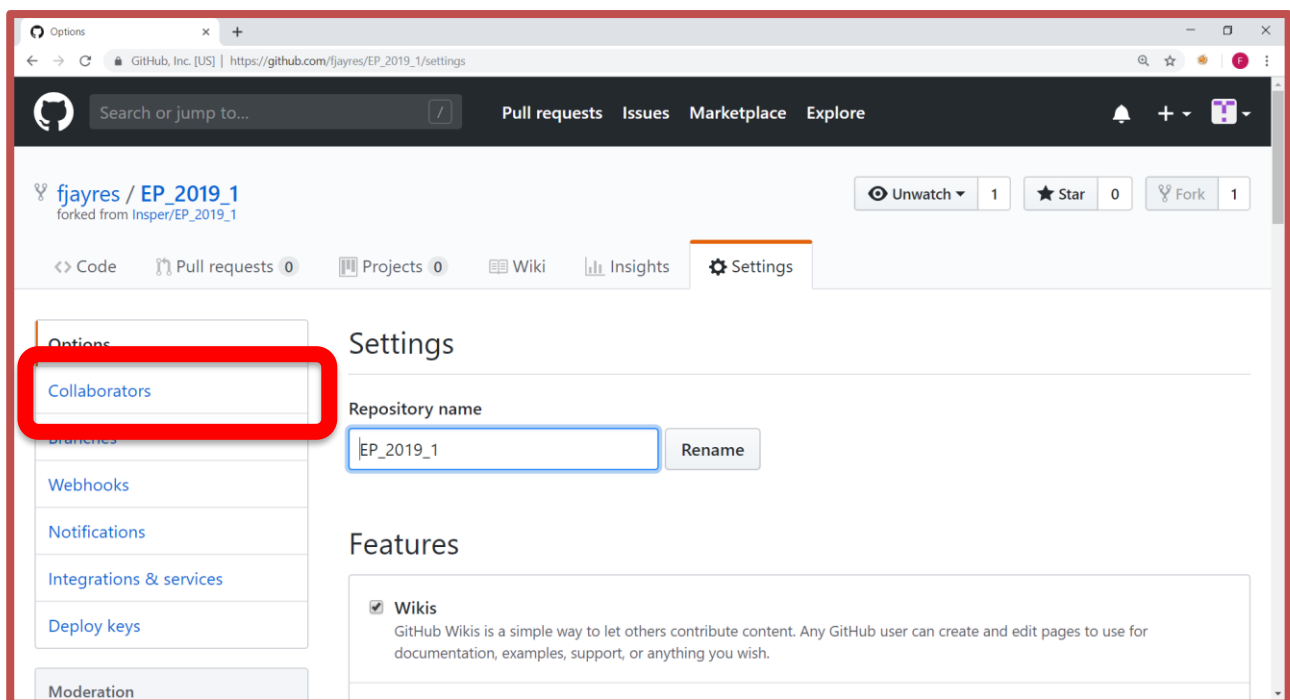
- **Mande o endereço do seu repositório para o professor, via entrega no Blackboard.** A avaliação será feita exclusivamente a partir do material presente no GitHub. Entregas via email, Dropbox, etc., não serão aceitas!
- Adicione o outro colega do time como contribuidor:



Vá para Settings...

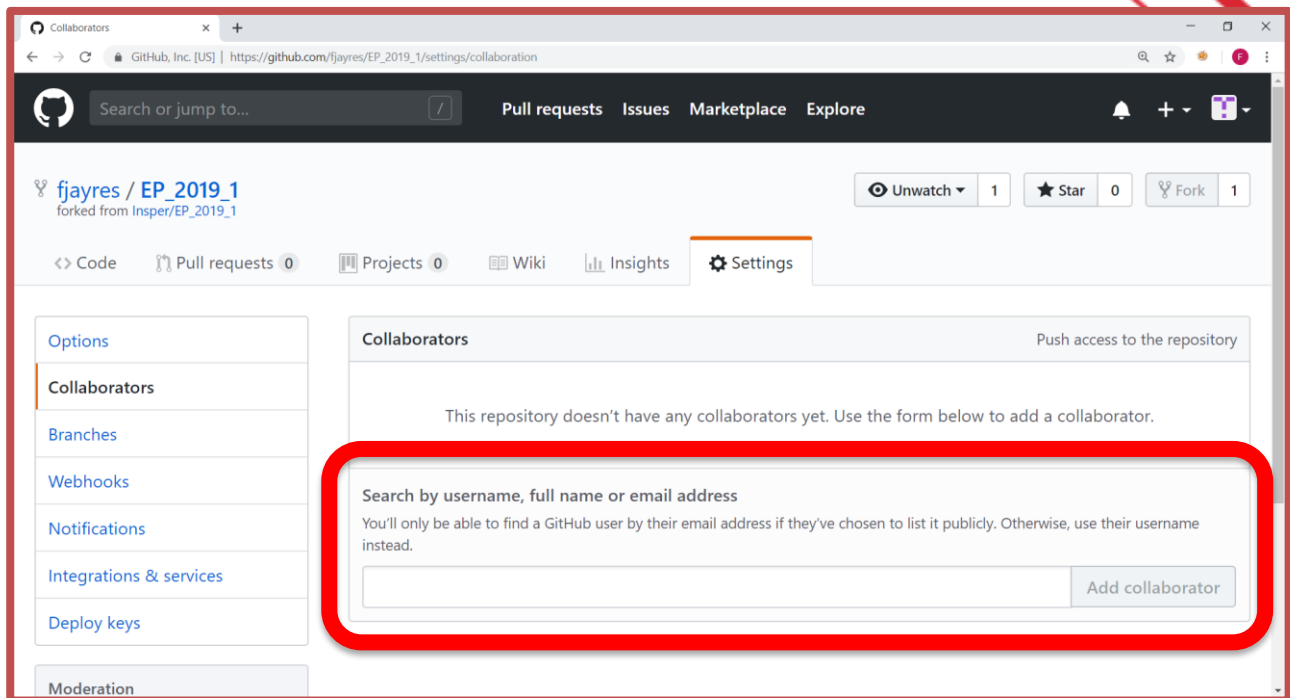


Depois para “Collaborators”



Agora digite o nome do seu colega e o adicione como colaborador:

# Insper



## Primeiros commits

Vamos começar a trabalhar no repositório. Nossa primeira tarefa será clonar o repositório remoto para começar a trabalhar. Neste *handout* vamos apenas apresentar a forma de fazer isso via terminal: os professores não vão dar suporte ao uso da ferramenta gráfica, pois esta muda ano após ano.

Abra o terminal do git. Faça a clonagem do repositório remoto:

```
Anaconda Prompt

(base) C:\Users\Fabio\Projetos\EP>git clone https://github.com/usuario/EP_2019_1.git
Cloning into 'EP_2019_1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.

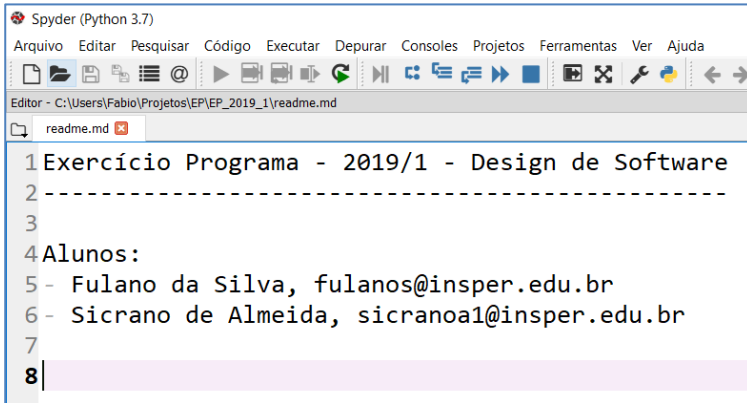
(base) C:\Users\Fabio\Projetos\EP>
```

onde **usuario** indica o usuário github da pessoa que fez o *fork*.



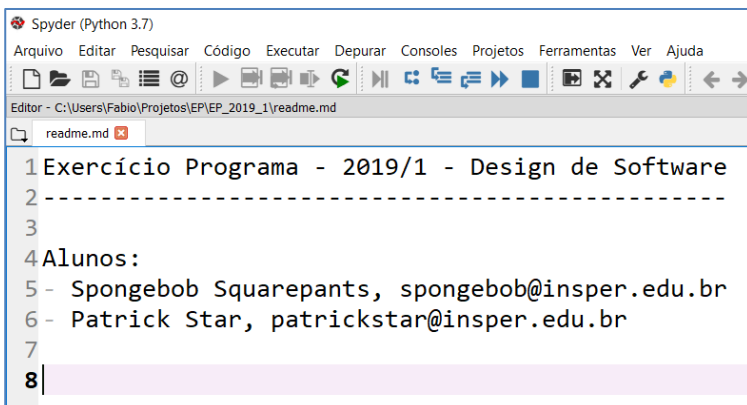
# Insper

Como primeira tarefa, vocês devem editar o arquivo `readme.md` e colocar os nomes dos membros da equipe. Abra este arquivo em um editor de texto qualquer (pode ser até mesmo o Spyder):



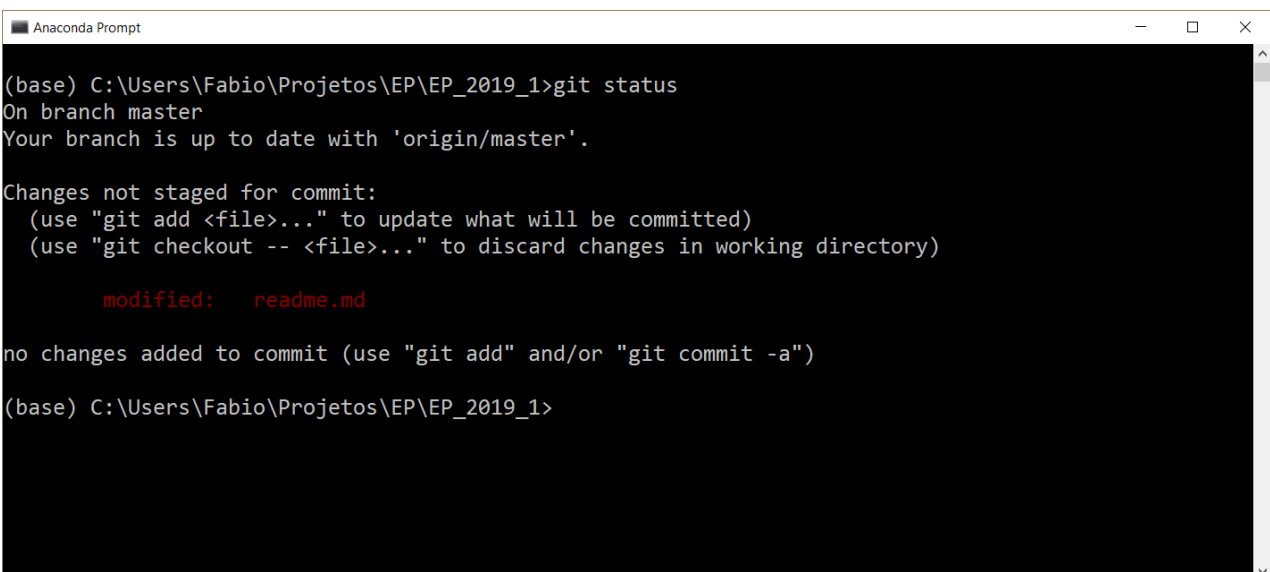
```
Spyder (Python 3.7)
Arquivo  Editar  Pesquisar  Código  Executar  Depurar  Consoles  Projetos  Ferramentas  Ver  Ajuda
[Icons]
Editor - C:\Users\Fabio\Projetos\EP\EP_2019_1\readme.md
readme.md
1 Exercício Programa - 2019/1 - Design de Software
2 -----
3
4 Alunos:
5 - Fulano da Silva, fulanos@insper.edu.br
6 - Sicrano de Almeida, sicranoa1@insper.edu.br
7
8 |
```

Coloque o nome dos membros da equipe e grave o arquivo:



```
Spyder (Python 3.7)
Arquivo  Editar  Pesquisar  Código  Executar  Depurar  Consoles  Projetos  Ferramentas  Ver  Ajuda
[Icons]
Editor - C:\Users\Fabio\Projetos\EP\EP_2019_1\readme.md
readme.md
1 Exercício Programa - 2019/1 - Design de Software
2 -----
3
4 Alunos:
5 - Spongebob Squarepants, spongebob@insper.edu.br
6 - Patrick Star, patrickstar@insper.edu.br
7
8 |
```

Verifique agora no terminal que o git detecta que uma mudança ocorreu:



```
Anaconda Prompt
(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme.md

no changes added to commit (use "git add" and/or "git commit -a")
(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>
```

# Insper

Adicione esta mudança ao *staging area* e faça o *commit*.

```
Anaconda Prompt

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>git add readme.md

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme.md

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>
```

```
Anaconda Prompt

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>git commit -m "Atualizei nomes dos membros da equipe."
[master 8c70b8d] Atualizei nomes dos membros da equipe.
 1 file changed, 2 insertions(+), 2 deletions(-)

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>
```

Faça então o *push* (não esqueça de dar um *pull* antes, sempre, para evitar problemas):

# Inspier

```
Anaconda Prompt

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>git pull
Already up to date.

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>
```

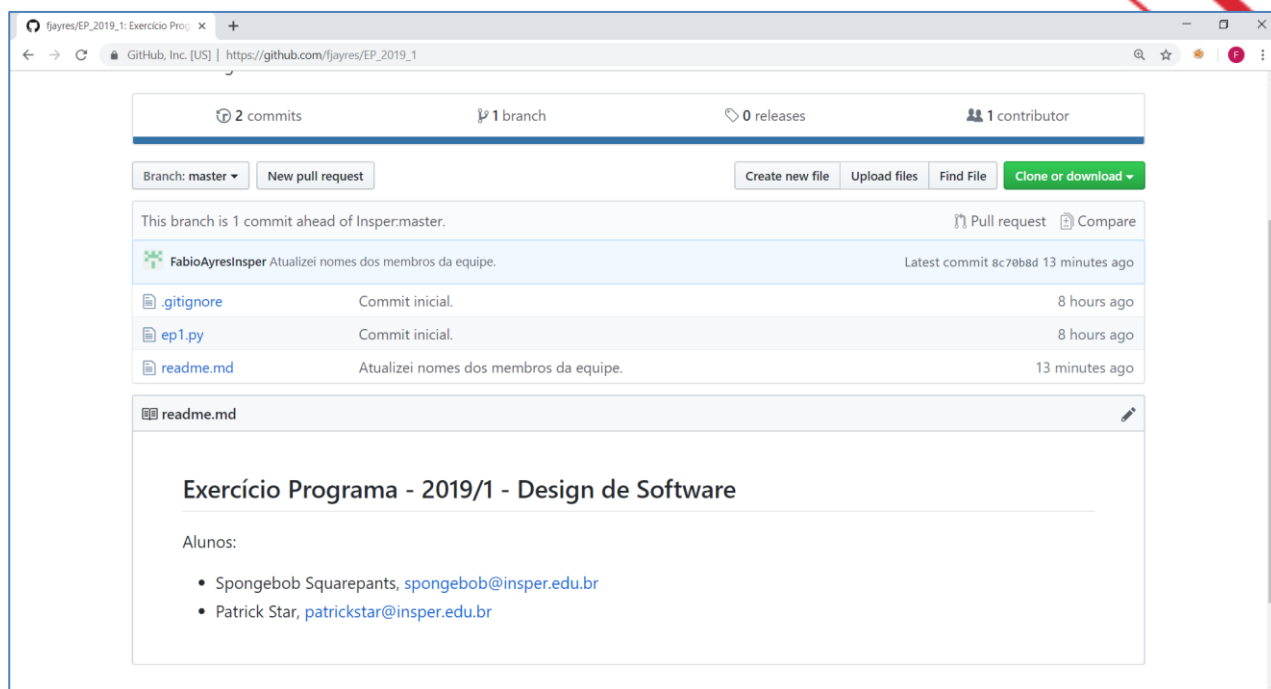
```
Anaconda Prompt

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 401 bytes | 401.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/fjayres/EP_2019_1.git
   d415ebf..8c70b8d  master -> master

(base) C:\Users\Fabio\Projetos\EP\EP_2019_1>
```

Verifique que o código do repositório remoto foi atualizado:

# Insper



Primeiro desenvolvimento separado por aluno

Abram o código em `ep1.py`. Observe que tem alguns espaços para preencher, e que estão marcados como “Aluno A” e “Aluno B”. Isso significa que cada aluno deverá desenvolver o seu próprio pedaço de código. Consequentemente, **cada aluno irá fazer commits de seu próprio pedaço de código.**

**ATENÇÃO: É FUNDAMENTAL QUE CADA ALUNO EXECUTE COMMITS DO SEU PRÓPRIO PEDAÇO DE CÓDIGO CONFORME INDICADO NO CÓDIGO INICIAL, USANDO SUA PRÓPRIA CONTA DO GITHUB!** Os professores vão verificar essa condição no histórico de *commits*: **se isso não for respeitado vocês terão falhado neste EP!**

Os trechos de código a serem desenvolvidos nesta etapa são explicados a seguir:

Para o aluno A:

- Imprimir o cenário atual: o código deverá imprimir detalhes do cenário atual de acordo com a seguinte especificação:
  - Título do cenário.
  - Na linha seguinte, um número de caracteres “-” igual ao comprimento do título do cenário.
  - Descrição do cenário.

Por exemplo:

```
Saguão do perigo
-----
Você está no saguão de entrada do Insper
```



Para o aluno B:

- Imprimir a lista de opções e ler a resposta do usuário.
- A resposta do usuário deverá ser guardada na variável `escolha`

Por exemplo:

Escolha sua opção:

andar professor: Tomar o elevador para o andar do professor

biblioteca: Ir para a biblioteca

O que você quer fazer? `biblioteca`

## Fase 2: desenvolvendo novas *features*

Agora é com vocês: escolha quais *features* vocês querem desenvolver dentre as descritas abaixo:

- **Feature 1: Monstros e prêmios!**
  - Adicione um mecanismo de aparição de monstros e prêmios ao acaso e que podem causar danos ou benefício.
  - Claro que para isso o jogador deverá ter *hit points*.
- **Feature 2: Combate!**
  - Se você implementou a *feature 1*, pode então implementar combates! Os combates podem ser até a morte, ou você pode implementar um mecanismo de fuga também!
- **Feature 3: Inventário!**
  - Implemente um sistema onde o jogador pode adquirir itens quando passa por certas salas, e que servem para permitir acesso a outras salas!
- **Feature 4: Teleporte!**
  - Implemente uma *feature* onde o aluno poderá, a partir de uma sala de teleporte, se transportar para uma sala qualquer desde que saiba de cor o nome da sala!
    - Isso inclui salas secretas e *easter-eggs*!
- **Feature 5: Arquivos!**
  - Ao invés de ter um dicionário descrevendo as salas, descubra como ler essa mesma estrutura a partir de um arquivo de texto, usando o padrão JSON. Veja aqui um exemplo: <https://realpython.com/python-json/>
- **Feature 6: Feature livre!**
  - Você tem carta branca para desenvolver o que quiser, DESDE QUE TENHA IMPLEMENTADO AO MENOS 4 DAS FEATURES ANTERIORES!



## Entrega

Envie no BlackBoard o endereço do seu repositório GitHub.

Exercícios copiados ou muito parecidos podem gerar notificações de infrações do código de ética Insper, resultando em reprovação automática da disciplina ou até mesmo jubramento.

O professor irá clonar o código presente no GitHub ao final do dia de entrega.