**Industria de Desenvolvimento de jogos**

**ABRAGAMES** - Associação Brasileira das Empresas de Desenvolvimento de Jogos ([https://www.abragames.org](https://www.abragames.org/))

Eventos relacionados a jogos:

**BIG Festival** - é o maior evento de negócios da américa latina

**Brasil Game Show (*BGS*)** - é a maior feira de games da américa latina voltada para o consumidor final.

**SBGames** - é o maior evento acadêmico sobre jogos e entretenimento digital da américa latina.

**Carreiras no Desenvolvimento de JOGOS**

* Programação

1. Gameplay
2. AI
3. Network
4. Ferramentas
5. GFX/Shaders - efeitos visuais

* Arte gráfica

1. Arte 2D
2. Arte 3D
3. Animação
4. Arte Conceitual
5. UI/UX
6. Tech Art -

* Game design

1. Prototipagem
2. Arquitetura dos Sistemas
3. Idealização do Jogo
4. Balanceamento
5. UX
6. Level Design

* Audio design

1. Efeitos Sonoros
2. Trilhas Sonoras
3. Programação de Áudio

* Narrativa
* Tester(QA)
* Produção
* Marketing

**PESQUISA DE CARGOS**

* Lista de estúdios associados à Abragames

1. SOBRE
2. ASSOCIADOS/AFILIADOS/PARCEIROS

* Canal "The Gamer Inside Brasil"

**POR ONDE COMEÇAR**

* Estudar
* Portifólio

1. **Github** para programadores
2. **ArtStation** paraartistas
3. **Itch.io** para jogos finalizados
4. **Sites próprios** para portifólios profissionais
5. **Notion** para portifólios flexiveis

**DICAS PARA CARREIRA**

Onde você quer estar em 5 anos?

* **Plano de aprendizado**

1. Objetivo final
2. Passo-a-passo
3. Inspirações e motivações

* **Publicar jogos**

1. Desenvolver protótipos
2. Finalizar projetos do começo ao fim
3. Sempre manter o escopo pequeno

* **Portifólio**

1. **Github** para programadores
2. **ArtStation** paraartistas
3. **Itch.io** para jogos finalizados
4. **Sites próprios** para portifólios profissionais
5. **Notion** para portifólios flexiveis

* **Networking**

1. Conversar com pessoas que já estão na indústria

**Introdução ao Desenvolvimento e Criação de Jogos**

Etapas do Desenvolvimento

* Idealização
* Protótipo
* Fatia Vertical
* Início da Produção
* Alpha - feature complete
* Beta - content complete
* Versão Final
* Lançamento

Principais Plataformas e Tecnologias

* Unity
* Unreal Engine
* Godot
* GameMaker Studio
* Construct

Onde Publicar um Jogo?

* Play Store(Google)
* App Store (Apple)
* Steam (Valve)
* Web (Kongregate, Gamejolt, etc)
* Consoles (Xbox, Switch, etc)

Direitos Autorais

Você pode usar comercialmente os assets do seu próprio jogo?

**Versionamento de Código**

configurar o git

ctrl + L = limpar o prompt

nome de usuário:

**git config --global user.name "throliveira"**

e-mail:

**git config --global user.email throliveira00@gmail.com**

Mostra a branch principal configurada

**git config init.defaultBranch**

Muda o nome da branch principal

**git config --global init.defaultBranch main**

Mostra as configurações globais

**git config --global --list**

Armazenamento de credenciais

**git config --global credential.helper *store*** *ou* ***cache***

Criando e clonando repositórios

**mkdir nome-da-pasta** cria o repositório

**cd nome-da-pasta** entra na pasta

**git init** inicia o repositório como um repositório git

**LS** - com letras minúsculas serve para listar o conteúdo do diretório.

**cat** - exibe o conteúdo

**cat** config -

**git clone** - clona um repositório do github para um repositório local

**git clone url nome-do-repositorio-clonado** - clona um repositório externo para um repositório local e altera o nome do mesmo.

**git remot -v** - mostra os repositórios remotos vinculados

**git remote add origin url-do-repositório-externo** - adiciona o repositorio local ao repositório externo.

**git clone URL --branch nome-da-branch --single-branch** - clona apenas a branch selecionada

**git add README.md** - adiciona um arquivo read.me no repositório

CRIAÇÃO DE REPOSITÓRIOS REMOTOS

gitignore - permite selcionar aquivos para que não sejam enviados para o repositório

SALVANDO ALTERAÇÕES.

**git status** - mostra o status da árvore de trabalho e da área de preparação

**touch README.md** - comando do terminal para criar um arquivo readme vazio

**.md** - extensão markdown = linguagem de marcação simples

**git add nome-do-arquivo** - adiciona um arquivo à áread e preparação para ser commitado.

git . - adiciona todos os arquivos alterados e recentes à area de preparação para ser comitado.

**git commit -m"mensagem obrigatória sobre o que está sendo comitado"** - faz o commit o arquivo preparado anteriormente.

**git log** - mostra o autor do commit, a data do commit e a mensagem do commit.

**echo nome-da-pasta/ > .gitignore** = adiciona o diretório ao arquivo gitignore para que não seja reconhecido como algo que deve ser commitado.

**echo > .gitignore** = remove o arquivo atribuído ao gitignore.

**.gitkeep** = é uma convenção utilizada para que o git reconheça diretórios vazios (**touch nome-da-pasta/ .gitkeep**)

Markdown - site: [https://readme.so](https://readme.so/)

GUIA SOBRE MARKDOW: [https://www.markdownguide.org](https://www.markdownguide.org/)

# = título

## = subtítulo

links = [palavra ou fraze a ser transformada em link] em seguida (link para onde o usuário será direcionado ao clicar)

windows + . = abre uma opção para seleção de emojis

- = cria lista com tópicos

Tabelas = |blablabla|blablabla|

|-----------|-----------| uma tabela será criada. O espaço entre os | devem ser prenchidos com o conteúdo da tabela

Caixas de código = ```git init```

DESFAZENDO ALTERAÇÕES NO REPOSITÓRIO LOCAL

**rm -rf .git** - para remover o arquivo git caso tenha iniciado um repositório git na pasta errada.

**git restore nome-do-arquivo** - restaura a ultima atualização de um arquivo que tenha sido erroneamente modificado. CCUIDADO, é importante ter certeza de que totas as alterações feitas no arquivo são indesejáveis, pois este comando descarta todas as alterações feitas e restaura o arquivo para a sua ultima versão commitada.

**git commit --amend -m "Nova mensagem"** - este comando é usado para corrigir ou alterar a mensagem do ultimo commit feito.

**git commit --amend** - abre um editor. **presionar 'I',** depois alterar a mensagem do commit, e para sair presisonar **esc:wq**

**git reset --soft (hash do commit)** - pega os arquivos posteriores ao commit selecionado e os coloca novamente na área de preparação (staging area)

**git reset --mixed (hash do commit)** - pega os arquivos posteriores ao commit selecionado e adiciona na árvore de trabalho, identificando-os como untracked files(arquivos não rastreados), o mixed é o padrão então é possível usar o comando da seguinte forma "**git reset**"

**git reset --hard (hash do commit)** - ignora completamente todos as alterações e commits feitos após o commit selecionado, deletando todos os arquivos e alterações.

**git reflog** - exibe um histórico detalhado das alterações feitas

**git reset nome-do-arquivo** - retira o arquivo da área de preparação e o coloca na árvore de trabalho como arquivo não rastreado.

**git restore --staged pasta/nome-do-arquivo.extensão** - tbm re move o arquivo da área de preparação

ENVIANDO E BAIXANDO ALTERAÇÕES COM O REPOSITÓRIO REMOTO

Criar o repositório no github

**git remote add origin** [**https://github.com/throliveira/dio-resumos-git-github.git**](https://github.com/throliveira/dio-resumos-git-github.git) - faz a conexão do repositório local com o repositório remoto.

**git branch -M main** - serve para mudar o nome da branch principal de master para main.

**git push -u origin main** - faz com que os arquivos commitados sejam adicionados no repositório externo.

**git pull** - pucha as alterações feitas diretamente no repositório externo para o repositório local.

**clicar na tecla .** - abre o web aditor do github.

TRABALHBANDO COM BRANCHES - CRIANDO, MESCLANDO, DELETANDO E TRATANDO CONFLITOS

**echo "#commit-1-branch-main" > commit-1-branch-main.txt** - o que significa esse comando?

**git checkout -b nome-danova-branch** - cria uma nova branch e muda para ela.

**git checkout nome-da-branch** - muda para a branch desejada

**git branch -v** - mostra os ultimos commits feitos em cada branch

**git merge nome-da-branch-que-vai-ser-mesclada** - esse comando mescla a branch secundária à principal. Deve-se estar na branch principal para mesclar a secundária.

**git branch** - lista todas as branchs no repositório

**git branch** -d nome-da-branch - esse comando exclui a branch nomeada

**conflito de merge** - acontece quando mais de um commit é feito na mesma linha de código ou arquivo e ao fazer o push o programador tem que decidir qual alteração será ssalva. Para isso deve-se fazer um pull do repositório remoto e então fazer a alteração manual no arquivo para então dar um add . e depois um commit e push.

TRABALHANDO COM BRANCHES - COMANDOS ÚTEIS DO DIA-A-DIA

**git pull -> git fetch + git merge**

**git fetch origin main** - busca aas alterações feitas no repositório remoto

**git diff main origin/main** - esse comando exibe as diferenças entre os arquivos locais e remotos, o primeiro main representa a branch local e o origin/mmain é a branch no repositório remoto.

**git clone URL --branch nome-da-branch --single-branch** - esse comando permite clonar apenas a branch desejada do repositório local.

**git stash** - arquiva uma alteração feita no repostiório como um commit.

**git stash list** - lista as alterações arquivadas

**git stash pop** - traz as alterações para a branch local e exclui da lista de arquivados

**git stash apply** - faz o mesmo mas mantem as alterações na lista do stash.

github lidianaandrade/dio-curso-git-github