iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Módulo 6: Princípios de Engenharia de Software

Aula 1
Introdução à Engenharia de Software

UPskill

Sobre o módulo

O módulo de Princípios de Desenvolvimento de Software pretende promover a compreensão e aquisição de competências iniciais e fundamentais da engenharia de software. Vamos estudar metodologias e boas práticas de desenvolvimento de software, o processo de engenharia de requisitos, desenho de sistemas, controlo de versões e metodologias de testes.

Programa do módulo

Aula	Tema
1	Introdução à Engenharia de Software; Requisitos; Metodologias de Desenvolvimento; Controlo de versões
2	Testes de Software; Documentação; Análise de código;
3	Introdução ao Desenho de Software; Diagramas de Use Case; Exercícios
4	Diagramas de Atividades; Exercícios
5	**Mini-teste; Apresentação dos Projetos**

Avaliação do módulo

Mini-teste Classificado de 0-20	Aula 5 22 / 09 / 2022	**50%**
Projeto Classificado de 0-20	Aula 5 22 / 09 / 2022	**50%**

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

O que é a Engenharia de Software?

Engenharia de software é um ramo da ciência da computação que trata o desenvolvimento e a construção de sistemas e software. Na sua essência, a Engenharia de Software é a aplicação dos campos de engenharia no ramo do desenvolvimento de software.

Campos da Engenharia de Software

- **Requisitos** de Software
- **Desenho** de Software
- **Desenvolvimento** de Software
- **Testes** de Software
- **Manutenção** de Software

Requisitos de Software

- Processo de **definir**, **documentar** e **manter** requisitos no processo de desenho de um software.
- Foco nas tarefas que determinam as necessidades dos clientes ou as condições necessárias para o desenvolvimento, alteração ou continuidade de um projeto ou produto.
- Permite ter uma lista de tarefas em mão e contacto entre os desenvolvedores e gestores.

Tipos de Requisitos

- Requisitos do **Cliente**
- Requisitos de **Arquitetura**
- Requisitos **Estruturais**
- Requisitos de **Comportamento**
- Requisitos **Funcionais**
- Requisitos de ***Performance***
- Requisitos de **Design**

Requisitos do Cliente

Requisitos que podem ser definidos ao responder às seguintes perguntas:

- Onde é que o sistema vai ser usado?
- Como é que o sistema vai realizar o objetivo especificado?
- Quais são os parâmetros críticos a ter em conta?
- Que componentes do objetivo final se podem subdividir?
- Quão eficiente ou eficaz é que precisa ser o sistema?
- Durante quanto tempo vai ser usado o sistema?
- Em que ambientes deverá funcionar?

Desenho de Software

- O Desenho de Software pode ser definido por todas as atividades envolvidas na conceptualização e enquadramento do sistema.
- Passa pelo planeamento e resolução de problemas na especificação **antes** do desenvolvimento.

Desenho de Software

- Principais princípios de desenho de software?
  - Deve-se ter em conta o que foi definido nos requisitos
  - Não se deve reinventar a roda.
  - Deve haver uniformidade e coerência.
  - Deve ser pensado de forma a ser escalável e alterável.
  - Bom software deve ser desenhado de raiz a pensar na robustez e na eficiência.
  - Deve-se ter em conta que o **desenho de software e o seu desenvolvimento são paradigmas diferentes.**

Desenvolvimento de Software

- A atividade principal (e mais demorada) na construção do software. A combinação da programação com os vários processos de definição, implementação, verificação, gestão e melhoria do ciclo de desenvolvimento.

- **Basicamente, o desenvolvimento foi o que fizeram nos projetos do quarto módulo.**

Testes e Manutenção

- Os testes e a manutenção interagem mutuamente no processo da engenharia e desenvolvimento dos projetos.
- Envolve a investigação que permite averiguar a qualidade do produto e que define as atividades necessárias para a continuidade e coerência do software.
- Os testes permitem garantir que o sistema continua a funcionar como esperado (podem ser automatizados) e a manutenção vai corrigindo falhas que vão acontecendo ao longo do tempo: o software precisa de ser mantido.

Hoje vamos focar-nos na

Fase de Desenvolvimento.

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

Paradigmas, Modelos e Metodologias

Dentro da engenharia de software, temos práticas comuns no desenvolvimento de soluções, as mais conhecidas:

- **Agile**
- Cleanroom
- Spiral
- V model
- **Waterfall**
- **Scrum**
- Kanban
- **DevOps**

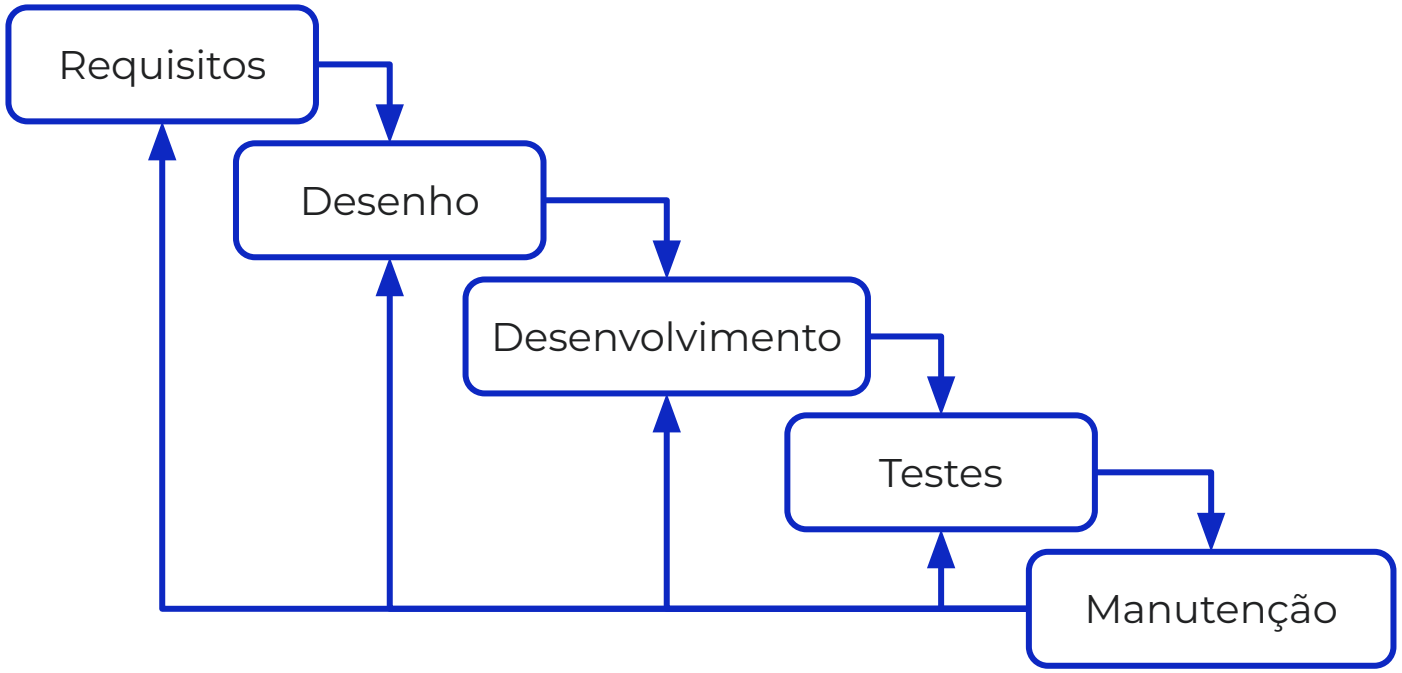
Modelos de Desenvolvimento

- Surgiram da necessidade de aumentar a performance do desenvolvimento de software
- Prevêem aumentar a colaboração dentro das equipas
- São filosofias e formas de pensar


iscte
INSTITUTO
UNIVERSITÁRIO
DE LISBOA
emprego
digital

Modelo *Waterfall*

O modelo de desenvolvimento de software mais antiquado. Baseado nos modelos de desenvolvimento de outras engenharias.



Waterfall: **Vantagens**

- Estrutura clara e concisa
- Fácil para projetos pequenos em que se consegue delimitar concretamente um ínicio e um fim
- Requisitos previsíveis e documentação bem definida
- Alta visibilidade: Passos bem definidos que permitem ao cliente ver o avanço no desenvolvimento
- Passagem de informação sistemática e sem perdas

Waterfall: **Desvantagens**

- Em projetos de longa duração torna-se impraticável
- Não permite realizar alterações a meio do desenvolvimento
- Exclui em grande parte o cliente da equação
- Testes só são feitos no fim, o que leva a um ciclo de vida instável

Modelo *Agile*

- Modelo alternativo ao *Waterfall* que providencia uma abordagem mais realista ao desenvolvimento de software.

- Depende dos requisitos iniciais e da ideia do produto final.

Agile: **Vantagens**

- Ótimo para projetos longos devido à flexibilidade nas alterações e no constante feedback da parte do cliente
- Separação do ciclo de desenvolvimento em "*sprints*" que tornam o processo mais rápido e segmentado
- Foco maior na qualidade do produto final
- Possível testar em qualquer fase
- Altamente cooperativo

Agile: **Desvantagens**

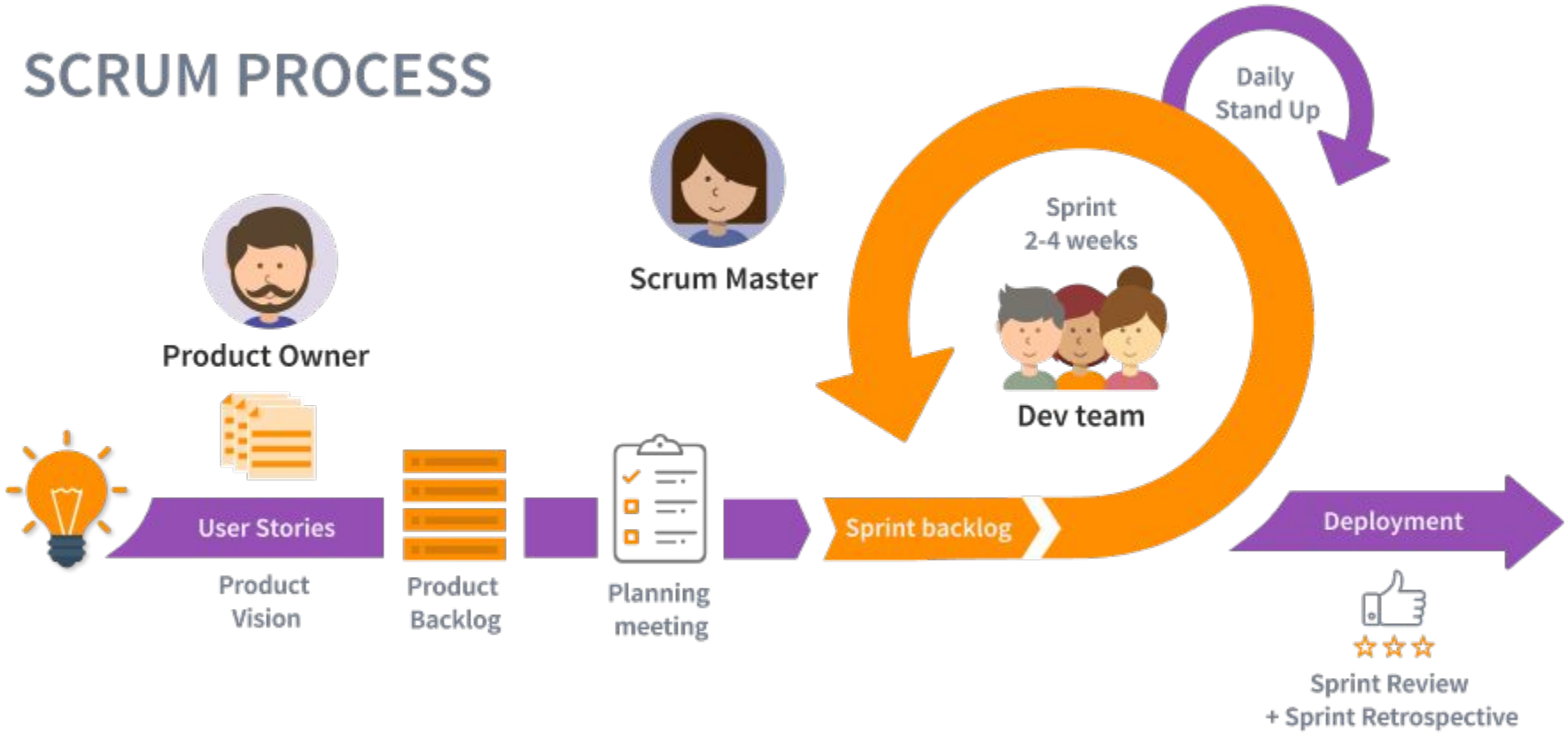
- Alto nível de stress nas equipas de desenvolvimento
- Clientes podem criar ciclos de inoperabilidade
- Difícil estabelecer um fim concreto do projeto
- Degradação da estrutura do projeto a cada iteração

Metodologia **SCRUM**

SCRUM

- Framework de gestão de projetos
- Baseado em desenvolvimento *agile*
- Atualmente utilizado em mais de 60% dos projetos *agile*
- O Scrum consiste num conjunto de técnicas e processos

SCRUM



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Características do ***SCRUM***

- Clientes tornam-se parte da equipe de desenvolvimento (os clientes devem estar genuinamente interessados no produto final)
- Entregas frequentes e intermediárias de funcionalidades 100% desenvolvidas
- Planos frequentes de mitigação de riscos desenvolvidos pela equipa
- Discussões diárias do status com a equipa responsável pelo desenvolvimento: ***“daily”***

Características do ***SCRUM***

- Na ***"daily"***, cada membro da equipa de desenvolvimento responde às seguintes perguntas:
  - O que fiz desde ontem em direção ao objetivo?
  - O que estou a pensar fazer até amanhã em direção ao objetivo?
  - Existe algo que me impede de atingir o objetivo?
- Reuniões frequentes com os *stakeholders* (partes interessadas no projeto) para monitorizar o progresso;
- Problemas não são ignorados e ninguém é penalizado por reconhecer ou descrever qualquer problema não visto;

Sprint

- Um sprint é a unidade básica de desenvolvimento em Scrum. Sprints tendem a durar entre uma semana e um mês, e são um esforço dentro de uma faixa de tempo (ou seja, restrito a uma duração específica).
- A adoção de ciclos relativamente curtos permite entregas de partes dos sistemas, gerando valor para os clientes e permitindo uma avaliação da dinâmica do trabalho.

Sprint

- Durante cada sprint, a equipa incrementa o produto potencialmente entregável (por exemplo, software funcional e testado).
- O conjunto de funcionalidades que entram em uma sprint vêm do Product Backlog, que é um conjunto de prioridades de requisitos de alto nível definidos pelo *Product Owner*.
- Os itens do backlog que entram para a sprint são determinados durante a reunião de planeamento da sprint (*Sprint Planning*).
- Durante esta reunião, o *Product Owner* informa a equipa dos itens no backlog do produto que quer concluídos.

Durante um sprint, ninguém está autorizado a alterar o backlog da sprint, o que significa que os requisitos são congelados para essa sprint.

Como implementar o ***Scrum?***

- O Scrum pode ser implementado através de uma ampla gama de ferramentas.
- Muitas empresas utilizam ferramentas diversas para construir e manter artefatos como o backlog da sprint.
- Há também pacotes de software open-source e proprietários dedicados à gestão de projetos com Scrum.
- Há organizações que implementam o Scrum sem o uso de quaisquer ferramentas, e mantêm os seus artefatos na forma de cópias impressas, como papel, quadros e notas.

Papeis em ***Scrum***

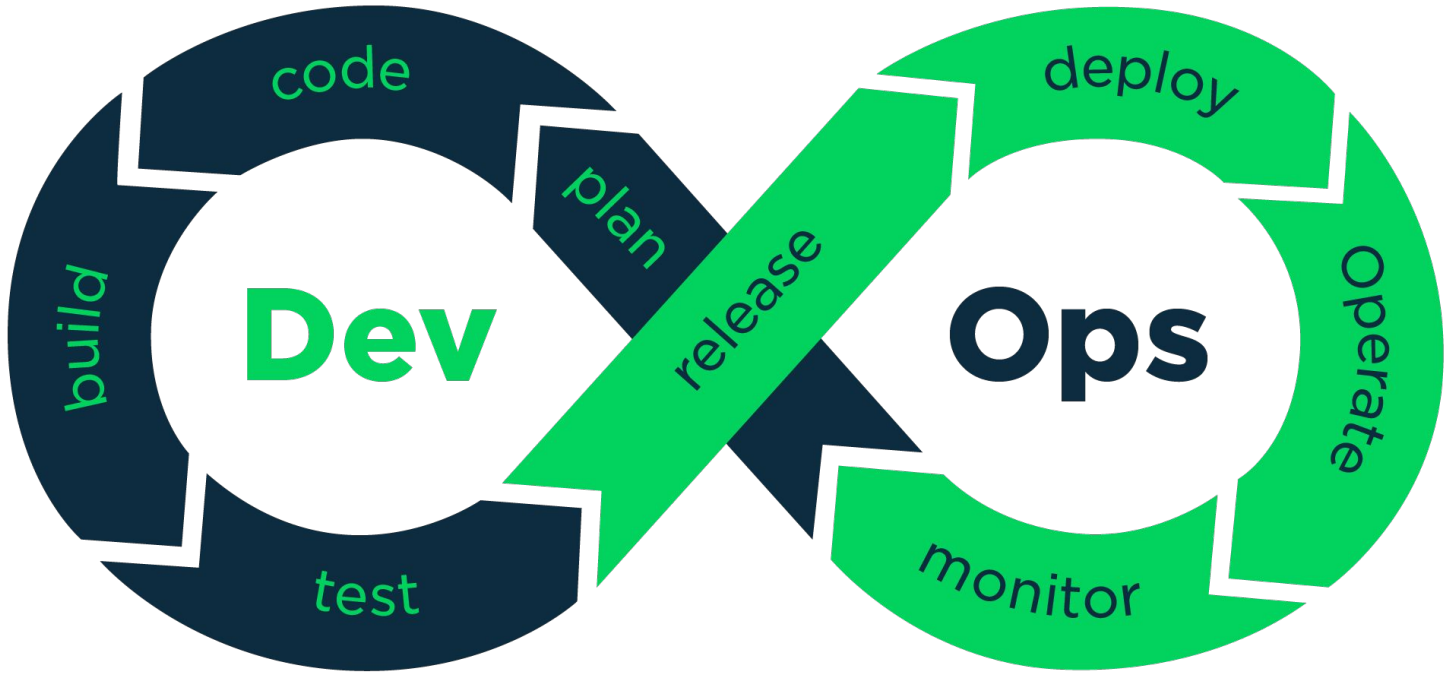
- O **Scrum Master**, que mantém os processos (normalmente no lugar de um gestor de projeto);
- O **Product Owner** (Dono do Produto), que representa os *stakeholders* e o negócio (ou seja, o cliente);
- A **equipa de desenvolvimento**, (ou DevTeam), um grupo multifuncional entre 3 a 9 pessoas e que fazem a análise, projeto, implementação, teste etc.

Metodologia **DevOps**

DevOps

- Baseado na metodologia *Agile*, tem como objetivo misturar o trabalho de duas equipas - desenvolvimento (Dev) e de operações (Ops).
- É uma filosofia e um paradigma cultural nas equipas de desenvolvimento de software.
- Toma proveito da automação e de ferramentas de engenharia de software para criar ambientes de desenvolvimento em que todos os membros da equipa partilham funções.

DevOps



Ferramentas de interesse

- Git
- StackOverflow
- Docker
- Heroku
- AWS

Controlo de **Versões**


iscte
INSTITUTO
UNIVERSITÁRIO
DE LISBOA
emprego
digital

Controlo de Versões

Sistemas que permitem manter um histórico de alterações a ficheiros e pastas de um projeto. Desenhadas para coordenar trabalho entre equipas de desenvolvimento de software para trabalharem no mesmo código ao mesmo tempo.

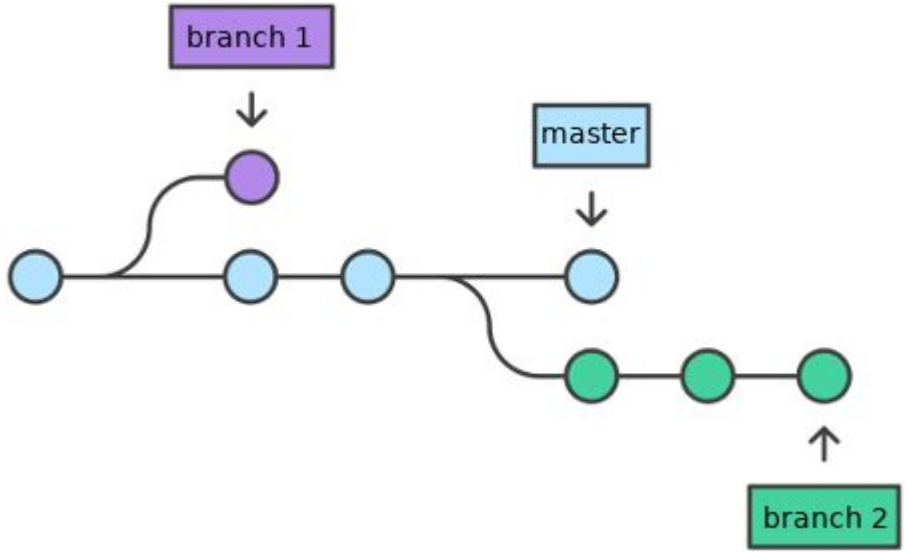
- Subversion (SVN), Mercurial, **Git**, CVS, etc.

Git

O Git distingue-se dos outros sistemas de controlo de versionamento ao permitir acesso completo a todos os ficheiros, *branches* e iterações de um projeto. Permite aceder ao **histórico completo de todas as mudanças** que foram efetuadas desde o ínicio do repositório e ter acesso a um log transparente de **quem e quando** se fez as alterações.

Git - **Funcionamento Geral**

- O Git permite criar vários ramos (***branches***) de desenvolvimento que por sua vez permitem distinguir funcionalidades a serem feitas, versões ou ambientes *(dev, QA, staging, prod*).
- Num ambiente estável de desenvolvimento, cada alteração concreta no projeto é “gravada” num branch através de ***commits.***



Git - **Comandos Básicos**

- `git init` - Inicializa um novo repositório git no diretório atual
- `git clone` - Cria uma cópia local de um projeto já existente remotamente
- `git add` - Marca um ficheiro ou diretório como “modificado”
- `git commit` - Grava as modificações no histórico do projeto
- `git push` - Atualiza o repositório remoto com commits feitos localmente
- `git status` - Mostra as modificações pendentes ou à espera de commit

Git - **Gestão de *Branches***

- `git branch` - Mostra os *branches* locais
- `git checkout` - Altera o *branch* atual para o especificado
- `git merge` - Junta dois *branches*
- `git revert` - Retrocede commits existentes e as alterações introduzidas
- `git cherry-pick` - Aplica alterações de um commit de outro branch ao atual

GitHub

- O ***GitHub*** é uma plataforma de alojamento de código-fonte e ficheiros com controlo de versão, que utiliza o Git.
- Permite que programadores, utilitários ou qualquer utilizador registado contribua para projetos privados e/ou Open Source de qualquer lugar do mundo.
- É amplamente utilizado por programadores e promove uma fácil comunicação através de recursos para relatar problemas ou juntar repositórios remotos (*issues, pull request*).

GitHub

- O GitHub é mundialmente usado e chega a ter mais de 36 milhões de utilizadores ativos contribuindo em projetos comerciais ou pessoais.
- Hoje, o GitHub aloja mais de 100 milhões de projetos, alguns deles que são conhecidos mundialmente: WordPress, GNU/Linux, Atom, Electron.
- **O GitHub também vai alojar o vosso projeto final!**

Git - **Instalação e configuração**

- Existem várias ferramentas que permitem fazer a gestão de repositórios Git: SourceTree, SmartGit, GitForce...
- **Contudo, o IntelliJ também inclui as todas funcionalidades do Git, pelo que será a ferramenta que vamos utilizar para gerir os nossos repositórios.**

- Instalar o Git no vosso computador: **https://git-scm.com/download**
- Criar uma conta no GitHub, onde os projetos ficarão alojados: **https://github.com**

Exercício 1

Criar e inicializar um novo repositório chamado “hello-world” no GitHub, adicionar um ficheiro *README* ao projeto, marcar a alteração no ficheiro criado, fazer um commit com uma mensagem adequada, adicionar o url do github como remote ao projeto e fazer *push* das alterações.

Neste exercício, deve utilizar os comandos estudados, e não as ferramentas do IntelliJ.

Exercício 1 - **Resolução**

```
# create a new directory, and initialize it with git-specific functions
git init hello-world

# change into the `my-repo` directory
cd my-repo

# create the first file in the project
touch README.md

# git isn't aware of the file, stage it
git add README.md

# take a snapshot of the staging area
git commit -m "add README to initial commit"
```

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

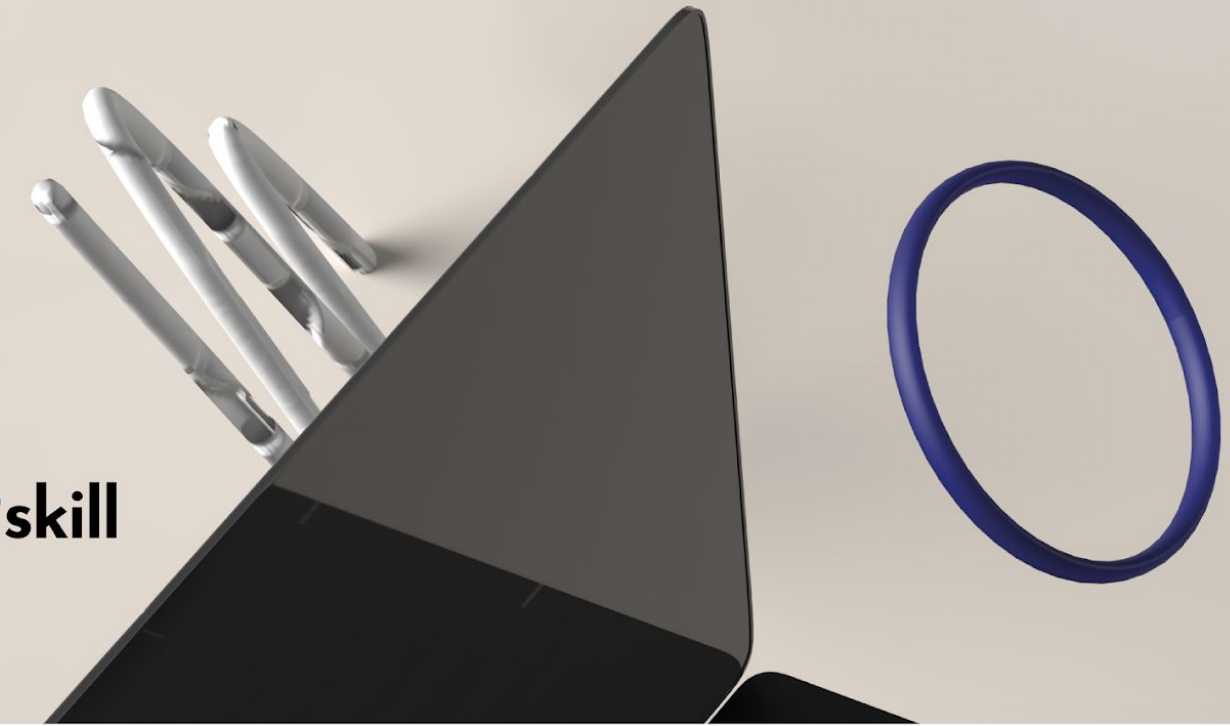
Exercício 1 - **Resolução**

```
# provide the path for the repository you created on github
git remote add origin https://github.com/YOUR-USERNAME/YOUR-REPOSITORY.git

# push changes to github
git push --set-upstream origin main

# para executar o último comando é necessário fazer login no github, e para tal é necessário criar um access token:
https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token
```

Git - Recursos de Aprendizagem

- https://learngitbranching.js.org/
- https://git-scm.com/docs

O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill