Aluno: Guilherme da Silva Delmiro

Agile: Faz parte do desenvolvimento ágil que consiste numa abordagem para gerenciamento de projetos e desenvolvimento de software sendo uma ajuda muito grande para equipes, pois a estrategia agrega muito valor a seus clientes entregando mais rapidamente do que ele precisa e com menos dores de cabeça. Em uma equipe ágil a entrega do trabalho consiste em entregar incrementos pequenos, mas consumíveis ao invés de lançar tudo de uma vez e ter a dor de cabeça de o cliente não gostar, ou precisar mudar algo com constância, com entregas pequenas fica mais fácil para qualquer mudança que ocorra no processo.

- Scrum: É um framework que assim como a metodologia Agile também funciona para gerenciamento de projetos que ajuda as equipes a estruturar e gerenciar seu trabalho por meio de um conjunto de práticas podendo ser usadas tanto por equipes de desenvolvimento de software a todos os tipos de trabalho em equipe fundada no Empirismo e no pensamento Lean. Na prática, o Scrum se divide em ciclos no projeto conhecido de “sprints”. Cada sprint é uma iteração curta com variação de tempo dependendo do projeto, normalmente dura de 1 a 4 semanas e tem um objetivo a ser alcançado. Durantes as sprints, a equipe se reúne diariamente para revisar o progresso e planejar o próximo dia. No final é feito uma revisão do trabalho concluído e identifica oportunidades de melhorias para o próximo sprint. O Scrum se baseia normalmente em três papéis principais: o Product Owner, responsável por desenvolver e comunicar explicitamente o Objetivo do Produto, criação e comunicação clara de itens do Product Backlog, encomendar itens do Product Backlog e garantir que o Product Backlog seja transparente, visível e compreendido;
o Scrum Master, responsável por garantir que a equipe esteja trabalhando de maneira eficaz e esteja seguindo as práticas do Scrum;
e a equipe de desenvolvimento, responsável por criar um plano para o Sprint, o Sprint Backlog, adaptar seu plano a cada dia para a Meta de Sprint, realizar o trabalho necessário e garantir que o objetivo definido pelo PO seja alcançado.

- Kanban: Conhecido por ser uma metodologia de gestão visual, o Quadro Kanban, ferramenta usada em todas as equipes kanban para visualizar o trabalho e otimizar o fluxo do trabalho entre a equipe. Embora aplicáveis a quase qualquer setor, as equipes de desenvolvimento de software encontraram um sucesso particular com a prática ágil. O quadro kanban pode ser usado tanto físico ou digital, tendo com função garantir que o trabalho da equipe seja visualizado, seu fluxo de trabalho seja padronizado e todos os bloqueadores e dependências sejam imediatamente identificados e resolvidos. Um quadro kanban básico tem um fluxo de trabalho organizado em colunas de três etapas: “A Fazer”, “Em Andamento”, “Concluído”. Seu objetivo é que as tarefas se movam pelas colunas do quadro, facilitando a visualização do que precisa ser feito, o que está em andamento e o que foi concluído. O Kanban também enfatiza para limitar as tarefas em andamento, para que a equipe se concentre em determinadas tarefas de cada vez para evitar sobrecarga e atrasos. A tarefas são representadas por cartões (ou post-its) que permitem que os membros da equipe acompanhem o progresso do trabalho por meio de seu fluxo de trabalho de maneira visual. Os cartões apresentam informações sobre o trabalho específico, responsável, descrição, duração e assim por diante.

- Gráfico de BurnDown: É uma ferramenta usada para gerenciar projetos que ajuda a acompanhar o progresso de uma equipe na conclusão de um projeto ao longo do tempo. Ele mostra a quantidade de trabalho que precisa ser feita em relação ao tempo restante até a data de entrega do projeto. São usados para prever a probabilidade de sua equipe concluir seu trabalho no tempo disponível representados com uma linha reta sendo a quantidade total de trabalho que precisa ser feita para completar o projeto. À medida que os trabalhos são concluídos, a linha vai descendo se aproximando de outra linha diagonal que representa a quantidade ideal de trabalho que deveria ser concluída em determinado momento, levando em conta a data de entrega do projeto. Dessa forma,

mantém a equipe ciente do prazo do projeto, se a linha reta estivar abaixo da linha diagonal, significa que a equipe está adiantada em relação ao que foi planejado, e se estiver acima, significa que está atrasada.

BDD: Behaviour-Driven Development ou Desenvolvimento Orientado por Comportamentos, criada por Dan North, que observou que enquanto utilizava a prática ágil do TDD no seu time de desenvolvimento, o time ficava perdido e havia muitas falhas na comunicação. Em sua investigação, descobriu que isso se dava pelo fato de que o time de desenvolvedores não sabia o que testar e por onde iniciar os testes.
O desenvolvedor apresentou ao mundo o BDD, um processo de desenvolvimento de software concebido em resposta ao TDD, ou seja, o BDD é a evolução da metodologia ágil TDD, que tem como principal objetivo agregar valor ao produto e conhecimento ao negócio, aproximando assim, todos os envolvidos do time, seja pessoas técnicas ou não e criar um entendimento comum entre as diferentes partes interessadas do projeto sobre o comportamento do sistema.

BDD pode ser definido de duas formas: uma das definições têm o BDD como uma metodologia de desenvolvimento ágil, que visa integrar regras de negócios com linguagem de programação e possui todo o seu foco voltado para o comportamento do software. Mas também é possível definir o BDD como uma prática em que os membros da equipe discutem o comportamento esperado de um sistema, para criar um entendimento compartilhado da funcionalidade esperada.

Umas das suas principais vantagens é que ele ajuda a garantir que o software atenda aos requisitos do usuário final e não apenas às necessidades técnicas do desenvolvedores, melhorando assim a qualidade do software e a satisfação do cliente, além de tornar o processo mais eficiente eficaz.

- Gherkin: O processo de desenvolvimento do BDD se baseia na escrita de cenários de testes usando uma linguagem ubíqua, essa linguagem se apoia no uso de um vocabulário pequeno, minimizando os problemas de comunicação de forma que todos os envolvidos digam a mesma linguagem, chamada “Gherkin” que permite você escrever história de usuário usando palavras-chaves de sua língua nativa.
O “Gherkin” possui as seguintes palavras-chaves: Funcionalidade (feature), Cenário (scenario), Dado que (given), Quando (when), E (and), Então (Then), sendo cada uma dessas palavras-chaves, responsáveis por uma funcionalidade.

A Funcionalidade (feature), representa as regras do sistema, o comportamento esperado de uma determinada ação do usuário no sistema.

- Given-When-Then: É um estilo de representação de testes. Os Cenários de teste são regidos por três palavras básicas, que definem o corpo da linguagem:
• Dado que (Given) : descreve o estado do mundo antes você começa o comportamento que está especificando nesse cenário. É possível pense nisso como as pré-condições para o teste
• Quando (When) : é o comportamento que você está especificando.
• Então (Then) : descreve as alterações que você esperar devido ao comportamento especificado.
Pense no cenário inicial, onde o usuário irá realizar seu login com sucesso. Então, temos o seguinte escopo:

> **Cenário de Teste**: Realizar login com sucesso no Gmail (aqui estou descrevendo o cenário) **Dado que** estou na página de login do Gmail (aqui temos uma pré-condição para iniciar o teste)
>
> **Quando** preencho o campo e-mail com um e-mail válido

E clico em próximo

E preencho o campo senha com uma senha válida

E clico em próximo (aqui temos todas as ações realizadas para executar o teste)

Então, devo ver a minha caixa de entrada do Gmail (por fim temos o resultado esperado pelo usuário)

BugTracking: Rastreamento de bugs em português, é o processo de registrar e monitorar bugs ou erros durante o teste de software. Também é conhecido como rastreamento de defeitos ou rastreamento de problemas. Sua importância se dá porque pode aumentar a produtividade e a qualidade das equipes de software. Também pode ajudar a reduzir o custo e o tempo de desenvolvimento e alcançar um maior retorno sobre o investimento. O rastreamento de bugs fornece uma estrutura para gerenciamento de defeitos e um fluxo de trabalho para relatar e resolver problemas. Também ajuda a melhorar a comunicação e o feedback dentro da equipe e com outras partes interessadas. Sua função se dá registrando e monitorando bugs ou erros detectados durante o teste de software que pode ser detectador por usuários, testadores ou ferramentas automatizadas. Algumas etapas envolvem a criação de um relatório de bug descrevendo o problema, sua gravidade, localização e etapas para reproduzi-lo. Logo então o relatório é atribuído a um desenvolvedor que fica responsável por corrigi-lo. P desenvolvedor pode precisar se comunicar com a pessoa que reportou ou o seu time, e em seguida, atualiza o status do bug à medida que trabalha nele, como aberto, em andamento, resolvido ou fechado.

- Rastreabilidade: No contexto de desenvolvimento de software, a rastreabilidade refere-se à capacidade de acompanhar como os requisitos são convertidos em funcionalidades específicas do software, bem como como essas funcionalidades são testadas e implementadas. Essa prática pode ser útil para garantir que o software atenda aos requisitos definidos pelos usuários ou clientes, bem como para identificar problemas que possam surgir durante o processo de desenvolvimento. Para isso, é necessário manter registros de todas as mudanças feitas no software ao longo do tempo e rastrear como essas mudanças estão relacionadas aos requisitos originais. A rastreabilidade permite que as equipes mantenham registros de todos os requisitos, especificações, design, código, testes e outros aspectos do processo de desenvolvimento. Além disso, a rastreabilidade também pode ser útil para fins de auditoria, permitindo que os desenvolvedores demonstrem que o software foi desenvolvido e testado adequadamente.

- Mantis: Ferramenta de software de código aberto usada para gerenciar bugs e rastrear problemas de softwares. Com sua interface simples e fácil de usar, ela permite que os desenvolvedores registrem e rastreiem bugs, problemas, melhorias e atribuam tarefas a membro da equipe. Além disso, oferece gerenciamento de fluxo de trabalho, possibilitando a capacidade de definir prioridades para bugs específicos e acompanhar o status de cada bug.
O Mantis permite a criação de relatórios personalizados sobre os bugs e problemas registrados, permitindo que as equipes tomem decisões informadas sobre as próximas etapas do processo. O seu uso é normalmente usado por muitas empresas e projetos de software de código aberto em todo o mundo sendo conhecido por ser uma ferramenta confiável e robusta para gerenciamento de bugs e rastreamento de problemas.

- Jira: Ferramenta de software desenvolvida pela Atlassian para gerenciamento de projetos, rastreamento de problemas e gerenciamento de tarefas. Muito comum em equipes de desenvolvimento de software, mas não se limita somente a essa área. Ele permite que equipes criem projetos e tarefas, atribuam tarefas a membro e por aí vai. Sua interface é bastante amigável para

qualquer usuário, que permite que os usuários vejam o status das tarefas em tempo real e atualizem o progresso das tarefas conforme andamento da tarefa ou projeto. É especialmente útil para gerenciar bugs e problemas em projeto, pois permite que as equipes criem tickets para problemas específicos e atribuam a membros da equipe e também permitem a definição de prioridade para bugs, problemas e acompanhem seus status ao logo do tempo.

- Azure DevOps:Plataforma de gerenciamento de ciclo de vida de aplicativos baseada na nuvem. É também um conjunto de ferramentas projetados para gerenciamento de projetos, rastreamento de problemas, colaboração em equipe, integração e entrega contínua de software.
Essas ferramentas são integradas e trabalham juntas para fornecer um ambiente colaborativo para equipes de desenvolvimento de software. Elas permitem que as equipes gerenciem e rastreiem problemas e tarefas usando Azure Boards, gerenciem e controlem o código-fonte usando Azures Repos, criem e gerenciem testes automatizados usando Azure Test Plans, gerenciem pacotes de software e dependências usando Azure Artifacts, e implantem e monitorem seus aplicativos na nuvem.

Testes Automatizado: É a aplicação de ferramentas de software para automatizar um processo manual orientado por humanos de revisão e validação de um produto de software. O objetivo desses testes é avaliar a qualidade do software, garantir que funcione corretamente e identificar possíveis bugs e problemas.
Para cria-los é possível usando uma variedade de ferramentas de software que automatizam o processo de teste, como frameworks de teste e ferramentas de automação. Ferramentas essas que permitem que scripts de teste sejam executados automaticamente a partir do que um desenvolvedor ou QA escrever, permitindo que os desenvolvedores identifiquem problemas de qualidade do software de forma rápida e eficiente.
Os testes automatizados permitem testar a funcionalidade, desempenho, segurança e usabilidade de um software, podendo ser executados em diferentes níveis do ciclo de vida de desenvolvimento de software, incluindo testes unitários, testes de integração e testes de aceitação.

- Cucumber: Ferramenta usada para testes automatizados de comportamento (BDD). O Cucumber é uma linguagem de especificação que permite que desenvolvedores e stakeholders descrevam o comportamento esperado do software, usando palavras-chaves como “dado que”, “quando” e “então” e em inglês Given-When-Then. Ele também é capaz de executar automaticamente os cenários de teste em linguagem natural que foram escritos, convertendo-os em código executável, permitindo que os testes sejam executados automaticamente e garantindo que o software funcione corretamente. Além disso, ele pode ser integrado em outras ferramentas de automação de teste, como Selenium e Appium, para testar a interface do usuário e a funcionalidade do software em diferentes plataformas.

- SpecFlow: É uma extensão do Cucumber, bastante popular para testes BDD de código aberto e para .NET, que permite seja escrito especificações em linguagem natural para descrever o comportamento desejado do software usando Given-When-Then, podendo ser escritas em colaboração com stakeholders (partes interesadas) do projeto, permitindo que todos envolvidos tenham compreensão dos requisitos e do comportamento esperado do software. Suas especificações em linguagem natural escritas no SpecFlow são automaticamente convertidas em código executável, o que permite que os testes sejam executados automaticamente, garantindo o funcionamento correto em todo ciclo de vida do desenvolvimento

- Selenium WebDriver: Ferramenta de automação de testes que permite a criação de scripts automatizados escritos usando linguagens de programação como Java, Python e Ruby, que são usados para interagir com elementos da interface do usuário. Sua vantagem é que seus scripts podem ser executados em diferentes navegadores da internet, permitindo testar funcionalidades em

diferentes ambientes. Ele também pode ser usado para testar aplicativos web em dispositivos móveis, permitindo que criem scripts que interajam com elementos de interface do usuário em smartphones e tablets.

- Appium: É uma ferramenta de automação de testes focada em aplicativos móveis que permite criar scripts automatizados para testar funcionalidades de aplicativos móveis nativos, web mobile e híbridos em plataformas IOS e Android e Windows usando linguagens de programação para interagir com elementos da interface do usuário, como botões e menus.

- Java: Linguagem de programação de alto nível e orientada a objetos criada nos anos 90 e projetada para ter o menor número possível de dependências de implementação. Seu uso geral se destina a permitir que programadores escrevam uma vez, executem em qualquer lugar (WORA), permitindo que o código compilado possa ser executado em todas as plataformas que suportam Java sem a necessidade de recompilar.

Controle de Versão: Prática de rastrear e gerenciar alterações no código do software. Os sistemas de controle de versão são ferramentes que ajudam equipes a gerenciar alterações no código-fonte ao longo do tempo, mantendo o controle de todas as modificações no código em um tipo especial de banco de dados. Se um erro for cometido, os desenvolvedores podem voltar no histórico daquele código e comparar versões anteriores do código para ajudar a corrigir algum erro. Para muitos o código fonte é como se fosse um repositório do conhecimento que os desenvolvedores coletaram e refinaram por seu esforço, e o controle de versão existe para protegê-lo tanto da catástrofe quanto da degradação casual do erro humano e das consequências não intencionais.

- Git: Sistemas de controle de versão criado em 2005 pelo criador do Linux, Linus Torvalds. Utilizado principalmente para gerenciar código fonte de projetos de software, permitindo que vários desenvolvedores trabalhem em um mesmo projeto ao mesmo tempo, mantendo um histórico das mudanças realizadas nos arquivos. O seu armazenamento não depende de um servidor central, ou seja, o Git mantém uma cópia completa de todos o projeto em cada um dos computadores do desenvolvedores, sendo muito útil para fazer mudanças em versões que não afetem a estabilidade da versão principal. Além disso, as mudanças nos arquivos são acompanhadas por commits, que são descrições detalhadas das alterações em tal momento, facilitando a visualização do histórico de alterações no projeto e identificando falhas.

- Gitflow: É um modelo alternativo de ramificação (branches) do Git que envolve o uso de ramificações de recursos e várias ramificações primárias. A estratégia Gitflow é baseada na criação de duas ramificações principais: a ramificação de desenvolvimento e a ramificação de produção.
A partir da ramificação de desenvolvimento que são criadas as ramificações de funcionalidades (feature branches), utilizadas para desenvolver novas funcionalidades ou corrigir bugs específicos. Quando a funcionalidade está completa, é realizada uma fusão (merge) da ramificação de funcionalidade na ramificação de desenvolvimento.
Já na ramificação de produção são criadas as ramificações de lançamento (release branches), que são utilizadas para preparar o código para o lançamento. As correções de bugs e outras melhorias são realizadas nesta ramificação, e quando estiver pronta para ser lançada, é realizada uma fusão da ramificação de lançamento na ramificação de produção.

Pipeline: Em engenharia de software, consiste em uma cadeia de elementos de processamento, dispostos de modo que a saída de cada elemento seja a entrada do próximo. É um fluxo de trabalho composto por várias etapas, que podem incluir desde a escrita do código, passando pelos testes de qualidade e funcionalidade, até a implantação e entrega do software ao usuário final.

Seu objetivo é garantir que o processo de desenvolvimento seja estruturado e organizado, permitindo a realização de testes e a correção de erros em cada etapa. Isso ajuda a evitar falhas que possam surgir no final do processo, quando é mais difícil e custoso fazer correções.

- CI: Integração Contínua em português, é uma prática comum que consiste em integrar o código produzido pelos desenvolvedores em um repositório central de forma contínua e automática. Tem o objetivo de detectar problemas de integração o mais cedo possível, antes que eles se tornem mais difíceis e caros de serem corrigidos. Em um ambiente de desenvolvimento de software, cada desenvolvedor tem sua máquina local com uma cópia do código fonte, quando o dev conclui uma tarefa e stá pronto para integras ao repositório do código base, ele pode fazer o push de suas mudanças para o repositório central, lá uma ferramenta de integração contínua irá automaticamente realizar uma série de ações que vai depender de cada projeto, alguns exemplos são: Compilar o código para verificar erros; Executar testes automatizados para verificar se as novas alterações tem bugs ou afetaram a funcionalidade existente…
Essa prática permite uma integração de forma contínua e rápida, reduzindo risco de conflitos de códigos e bugs.

- CICD: Integração Contínua/Desenvolvimento Contínuo em português, é uma abordagem que possibilita a entrega de aplicativos com frequência aos clientes, introduzindo a automação nos estágios de desenvolvimento de aplicativos. A prática envolve o uso de ferramentas de automação que ajudam a integrar as mudanças de código dos devs em um repositório central, e em seguida, automaticamente construir, testar e implantar o software para produção.
O processo começa com a prática de CI, em que o código é integrado continuamente no repositório central e testado automaticamente para garantir que ele funcione corretamente com o resto do código existente.
Após o CI, a prática de CD entra em ação, onde a aplicação é implantada automaticamente em um ambiente de produção após ser testada e aprovada na fase de CI. Essa abordagem garante que as mudanças no código sejam implantadas rapidamente e de forma confiável em um ambiente de produção, reduzindo o tempo e o custo de implantação do software.

- Jenkins: É um servidor de automação autônomo de código aberto podendo ser usado para automatizar toda tarefa relacionada à construção, teste e entrega ou implantação de software.
Sua ideia é, cada alteração feita no código fonte, o Jenkins automaticamente irá verificar a integridade do código, construir o software e executar testes automatizados para garantir que a mudança não tenha impacto negativo. Essa verificação ajuda a reduzir risco de erros e problemas de integração, permitindo entregas de forma mais rápida e confiável.

-Gitlab: É uma plataforma para gerenciamento de repositórios de código fonte, muito utilizado em projeto de desenvolvimento de software. Ele inclui várias funcionalidades e recursos, incluindo controle de versão, rastreamento de problemas, integração contínua e entrega contínua (CI/CD), gerenciamento e revisão de código e muitos outros.
Ele se baseia no sistema de controle de versão Git, isso significa que ele é capaz de lidar com grandes projetos de desenvolvimento de software de forma eficiente e segura. Oferecido com um serviço em nuvem, ou seja, pode ser acessado através da internet, ou instalado em um servidor próprio, o que permite mais controle sobre sua infraestrutura e segurança.