Behavier driven development (BDD)

Introdução:

Behavier driven development, ou seja, desenvolvimento guiado por comportamento é uma metodologia de desenvolvimento de software que visa promover a colaboração entre os desenvolvedores, pessoas da área de negócio e testadores, por meio da criação de especificações de comportamento claras e compreensíveis. O BDD utiliza uma linguagem específica (geralmente em formato de histórias ou cenários) para descrever o comportamento desejado do software em termos como interações com o usuário entradas e saídas. O BDD se concentra em definir o comportamento esperado do software antes que ele seja desenvolvido.

## Processo:

O processo do BDD começa com o alinhamento entre a visão de todos os envolvidos após isso são criados cenários que descrevem como o software deve se comportar em diferentes situações, esses cenários são escritos em linguagem natural (Gherkin) para garantir que todas as partes envolvidas no projeto possam compreender.

## Ferramentas Comuns de BDD:

Algumas ferramentas populares que auxiliam na implementação da metodologia BDD:

* Cucumber: Utiliza a linguagem Gherkin para descrever comportamento de forma compreensível. No ponto de vista da programação a utilização desta ferramenta pode ser utilizada em conjunto com Java, Javascript para a criação de código através desses cenários.
* SpecFlow: É outra ferramenta que integra BDD com .NET o que permite a utilização de diversas linguagens e diversas plataformas é possível através do specflow é possível “traduzir “as histórias de usuário para linguagem de programação.
* JBehave: Primeiro framework criado para BDD ela assim como Cucumber suporta a escrita de histórias de usuário.

## Benefícios do BDD:

O BDD oferece diversos benefícios para o desenvolvimento de software principalmente para as equipes envolvidas no projeto iremos listar algumas destas vantagens a seguir:

* Melhoria na comunicação: importante salientar que BDD não é um processo de teste de software, mas sim um processo de desenvolvimento que foca principalmente na conversa na troca de visões entre envolvidos no projeto e o BDD se utilizado corretamente gera uma melhora na comunicação.
* Documentação dinâmica: BDD facilita a criação de documentos atualizados e compreensíveis já que todas as partes do processo são documentadas o que permite um controle maior dos passos do projeto por exemplo requisitos essa documentação clara e atualizada sobre os mesmos permite um melhor desenvolvimento do projeto.
* Redução de erros: O BDD identifica problemas no início do projeto, economizando tempo e dinheiro. Por conta das conversas realizadas entre todos os envolvidos e por conta da estrutura de desenvolvimento que parte de uma visão de como o projeto deve ser logo no início isso irá mostrar vários erros problemas ocasionando numa redução de custos já que erros quanto mais tarde forem encontrados mais caros eles serão.
* Feedback rápido: permite que os desenvolvedores obtenham feedback rapidamente, garantindo que o software atenda as expectativas, por conta de todos esses benefícios como Melhora na comunicação, Documentação dinâmica e Redução de erros os desenvolvedores acabam por terem um feedback rápido já que sempre que algo que não for agradável a outra parte envolvida como por exemplo o PO (product owner) será rapidamente evidenciada.

## Como é aplicado?

Para a aplicação eficiente do BDD é necessário seguir quatro etapas distintas:

1. Descoberta: O Product Owner (PO) explica a história e a visão geral do negócio. A equipe utiliza o processo de discussão do BDD para reunir exemplos de usuários e entender como a função fornecida é usada.
2. Definição: A equipe faz perguntas para determinar regras, critérios de aceitação ou novas histórias. O esclarecimento dessas ideias ocorre por meio de diálogo entre os participantes.
3. Formalização: Todas as questões discutidas são formalizadas com uma definição, criando um documento que contém todas as notas reunidas em um idioma neutro. A linguagem Gherkin é geralmente recomendada para criar critérios de aceitação.
4. Entrega: A equipe de desenvolvimento apresenta a história validada ao PO para validação. Após a validação, a história é utilizada durante a revisão da produção do projeto para garantir que o produto corresponda aos padrões estabelecidos.

## A Relação entre BDD e TDD:

O Behavior Driven Development (BDD) e o Test Driven Development (TDD) são práticas que têm um grande impacto no desenvolvimento de software. Embora sejam semelhantes, eles diferem em suas abordagens.

O BDD se concentra no comportamento do software e nas interações do usuário, enquanto o TDD se concentra nos testes unitários. O BDD promove uma linguagem clara e concisa para escrever as especificações, tornando-as compreensíveis tanto para os clientes quanto para os desenvolvedores. Em contraste, o TDD se concentra mais na automação dos testes unitários, o que permite que os desenvolvedores se dediquem à funcionalidade do software.

Apesar de suas diferenças, BDD e TDD trabalham juntos para garantir que o software seja desenvolvido de forma eficiente e com qualidade. O BDD fornece a estrutura para as especificações e cenários de teste, enquanto o TDD fornece a estrutura para os testes unitários. A integração do BDD e TDD garante que o desenvolvimento do software esteja de acordo com as especificações do usuário e que o código funcione adequadamente em todos os níveis.

## A importância da automação de Testes no processo de Behavior Driven Development.

A automação de testes é essencial no processo de BDD, pois garante a eficiência e precisão dos testes realizados. Com a automação, é possível testar cenários complexos de forma rápida e consistente, além de reduzir a chance de erros humanos que podem comprometer a qualidade do produto final.

Além disso, a automação de testes contribui para a documentação do software, uma vez que os testes automatizados servem como exemplos vivos de como a aplicação deve funcionar. Isso pode ser muito útil para futuras manutenções ou para a integração de novas funcionalidades, já que os desenvolvedores têm acesso a um conjunto de testes que validam o comportamento esperado do sistema.

Por fim, a automação de testes também permite a criação de uma cultura de qualidade no processo de desenvolvimento de software, uma vez que incentiva a equipe a escrever testes automatizados de forma consistente e a monitorar a qualidade do software de forma contínua. Isso pode resultar em um produto final mais confiável e satisfatório para os usuários.

Conclusão:

O Behavior Driven Development (BDD) é uma metodologia ágil que se concentra na definição do comportamento esperado de um software antes de seu desenvolvimento. Ao promover uma melhor comunicação e colaboração entre as equipes, o BDD garante que o software desenvolvido atenda às expectativas do usuário final e seja entregue com alta qualidade. Ferramentas como Cucumber, SpecFlow e JBehave facilitam a implementação do BDD, tornando-o uma prática valiosa para qualquer equipe de desenvolvimento de software. Ao adotar o BDD, as empresas podem melhorar a eficiência do desenvolvimento de software, reduzir erros e garantir uma experiência positiva para os usuários finais.

## Referencias e leituras realizadas:

* https://www.devmedia.com.br/behavior-driven-development-bdd-com-specflow/29405
* https://coodesh.com/blog/candidates/metodologias/bdd-na-pratica-entenda-o-que-e-e-como-funciona/
* https://www.objective.com.br/insights/bdd/?utm\_campaign=linkedin.com-blog-bdd&utm\_term={keyword}\_{matchtype}&utm\_content=247783217&utm\_medium=social&utm\_source=linkedin&hss\_channel=lcp-1056597
* https://www.devmedia.com.br/desenvolvimento-orientado-por-comportamento-bdd/21127