ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Relatório

Engenharia de Software II

2021/22

Trabalho prático II

Realizado por:

Bruno Dylan Pinto Ferreira, nº 8200586

Gonçalo André Fontes Oliveira, nº 8200595

Jorge Miguel Fernandes Correia, nº 8200592

Nuno de Figueiredo Brito e Castro, nº 8200591

Tabela de conteúdos

1. Introdução **2**

2. Âmbito **2**

3. Glossário **3**

4. Abordagem **5**

Metodologias de Desenvolvimento de SW 5

Metodologia SCRUM 5

Processo SCRUM 6

Pre-Game 6

Development or Game Phase 6

Post-Game Phase 6

Processos do desenvolvimento de software 8

SDLC - Modelo do ciclo de vida do desenvolvimento do software 8

Abordagem do problema 10

5. Desenvolvimento e configurações **11**

6. Ferramentas **14**

7. Conclusão **15**

8. Repositório Git **15**

9. Bibliografia **15**

1. Introdução

Neste projeto foi nos pedido que fosse desenvolvida uma plataforma que permitisse facilitar as transações de produtos a nível nacional. Além disto, deveria ainda permitir a recolha e entrega de encomendas. O software desenvolvido deveria suportar a gestão de toda a informação referente a empresas, produtos, compras e envios de produtos.

Como suporte ao desenvolvimento da nossa solução foi nos dada uma API de suporte à implementação de um livro razão, capaz de gerir e armazenar de modo seguro as transações realizadas entre entidades.

2. Âmbito

Este projeto foi desenvolvido no âmbito da disciplina de Engenharia de Software II, no qual era nos dito que a Market for Business to Business (G4B2) é uma plataforma que permite facilitar a transação de produtos entre empresas a nível nacional e, para além de ser permitir efetuar as transações, faz a recolha e entrega das encomendas. Para melhorar a produtividade da G4B2 , foi proposto o desenvolvimento de um software capaz de processar e otimizar a sua atividade.

Resumidamente, a solução deveria permitir gerir toda a informação sobre as empresas, produtos, compras e envios dos produtos. Era ainda referido que a aplicação faria parte de três módulos com responsabilidades distintas que permitirão aumentar a segurança e produtividade da G4B2.

Para suporte ao desenvolvimento do software, foi nos fornecida uma API de suporte à implementação de um livro razão que permite armazenar de uma forma segura as transações realizadas entre entidades.

3. Glossário

Issue - Issue normalmente significa o desenvolvimento de uma funcionalidade de um produto final desenvolvido por uma equipa de um projeto, tendo em conta requisitos especificados. A uma issue podem ser atribuídos pesos, prioridades, estados etc.

Epics - Uma epic é um conjunto de trabalhos que podem ser divididos em tarefas específicas (user stories / issues) com base nas necessidades/solicitações de clientes ou utilizadores finais.

User Story - As user stories definem quem, o quê e o porquê de um recurso do produto. Normalmente são destinados a developers, mas devem ser facilmente compreendidos por todos os membros de uma equipa, incluindo clientes. Eles descrevem o que um utilizador precisa de fazer, do ponto de vista desse utilizador.

Points and Estimation - Quando existem muitas issues pode tornar-se difícil obter uma visão geral da mesma. Para resolver este problema pode associar-se pesos a issues para obter uma ideia melhor de quanto tempo, valor ou complexidade uma issue pode ter.

Labels - As labels são atribuídas a issues individuais, de modo a posteriormente efetuar pesquisas numa lista de issues utilizando uma ou mais labels como filtro.

Product Backlog - O product backlog é uma lista de novas funcionalidades ou mudanças das mesmas, correção de bugs ou outras atividades que uma equipa pode entregar para alcançar um resultado específico.

Sprint - Define um período de tempo finito em que um trabalho deve ser feito. Pode variar entre uma semana e um mês ou mais. Para definir um período de tempo corretamente é normal estimar o esforço necessário para concluir a sprint em questão.

Burndown Chart - São gráficos que permitem obter uma visão mais geral do rumo de desenvolvimento de determinado projeto. Através deste gráfico é possível verificar, por exemplo, se existem possíveis melhorias para calcular o tempo estimado de determinada issue/sprint.

Agile Board - São quadros subdivididos em labels que permitem fazer uma gestão mais eficiente das issues.

Testes de software - Um teste de software é um software que executa outro software, validando se os resultados são os esperados (teste de estado) ou se executa a sequência de eventos esperado (teste de comportamento).

Resumidamente, os testes de software permitem aos programadores verificar se a lógica do programa desenvolvido está de acordo com os requisitos. A execução automática de testes permite identificar “bugs” resultantes de mudanças no código fonte.

ECP - Técnica destinada a reduzir o número de testes necessários dividindo o domínio de entrada (ou saída) em classes de dados em que os casos de teste podem ser derivados para cada operação, o “tester” deve identificar as classes de equivalência dos argumentos e os estados dos objetos

BVA - Técnica baseada na observação de bugs que ocorrem frequentemente em valores fronteira. É focada em testar valores especiais (null, 0, etc) e limites do domínio de entrada (ou saída) imediatamente acima e abaixo (além de ou em vez de valores intermédios)

4. Abordagem

Ao longo do desenvolvimento foram feitas várias abordagens que nos permitiram tomar algumas decisões cruciais para a realização/gestão do projeto.

Metodologias de Desenvolvimento de SW

A realização deste projeto foi baseada em metodologias ágeis, que consistem numa abordagem de desenvolvimento de software baseada no desenvolvimento iterativo. Métodos ágeis dividem tarefas em iterações menores. O alcance e os requisitos do projeto são estabelecidos no início do processo de desenvolvimento. Os planos relativos ao número de iterações, à duração e ao alcance de cada iteração são claramente definidos com antecedência.

Metodologia SCRUM

SCRUM é uma estrutura desenvolvida para gestão do processo de desenvolvimento de sistemas e concentra-se maioritariamente na forma como os membros da equipa devem funcionar para produzir um sistema flexível num ambiente constantemente variável e complexo.

O objetivo é que o desenvolvimento do sistema envolva várias variáveis ambientais e técnicas que estão provavelmente a alterar-se durante o processo, o que torna o processo de desenvolvimento imprevisível e complexo, requerendo flexibilidade para ser capaz de responder às alterações.

Processo SCRUM

O processo scrum está dividido em 3 fases, sendo elas:

Pre-Game

Esta fase encontra-se dividida em duas sub-fases: **Planeamento** e **Arquitetura/Projeto de alto nível**.

É na sub-fase de **Planeamento** que é criada uma Product Backlog list contendo os requisitos atualmente conhecidos. É ainda nesta fase que os requisitos são priorizados e o esforço necessário para a sua implementação é estimado.

Além disto é ainda realizada uma avaliação do risco e aspectos de controlo, necessidades de formação e aprovação da gestão de verificação.

Já a sub-fase de **Arquitetura/Projeto de alto nível** inclui o planeamento da arquitectura com os itens actuais na Product Backlog.

Development or Game Phase

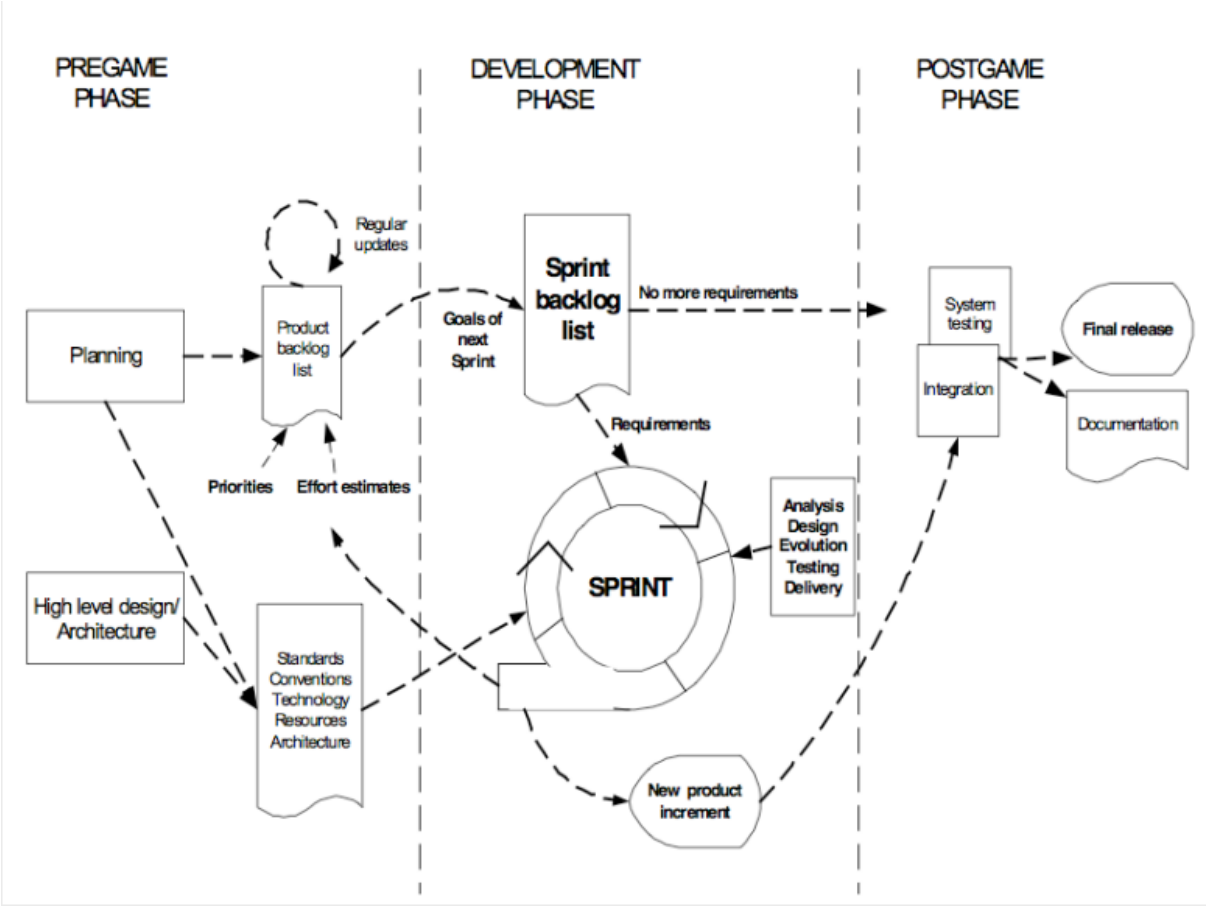
Esta parte é considerada a parte ágil da abordagem SCRUM e é vista como uma “black box” onde o imprevisível é esperado, devido ao facto de variáveis técnicas e ambientais como, prazo, qualidade, requisitos, entre outros podem variar durante o processo.

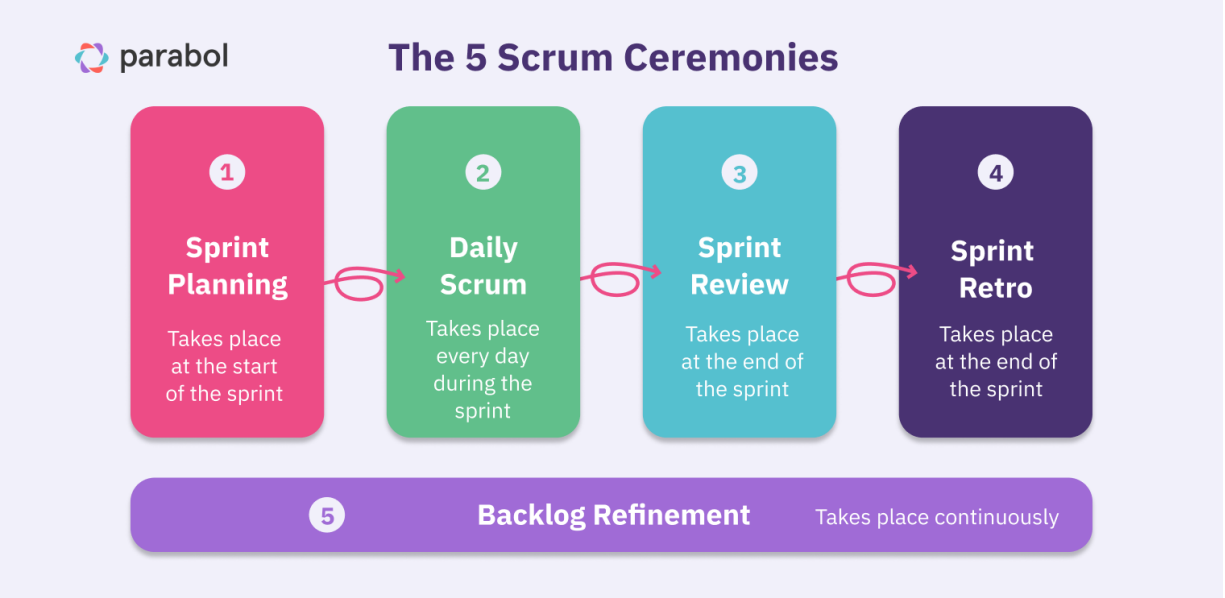
São observadas e controladas através de várias práticas durante os Sprints da fase de desenvolvimento em que, cada Sprint inclui as fases tradicionais do desenvolvimento de software: análise, requisitos, projeto, evolução e entrega.

Post-Game Phase

Nesta fase o sistema está pronto para ser entregue ao cliente e inclui tarefas tais como integração, testes do sistema e documentação.

Mais abaixo pode ser observado um esquema que representa exatamente o funcionamento do processo SCRUM:



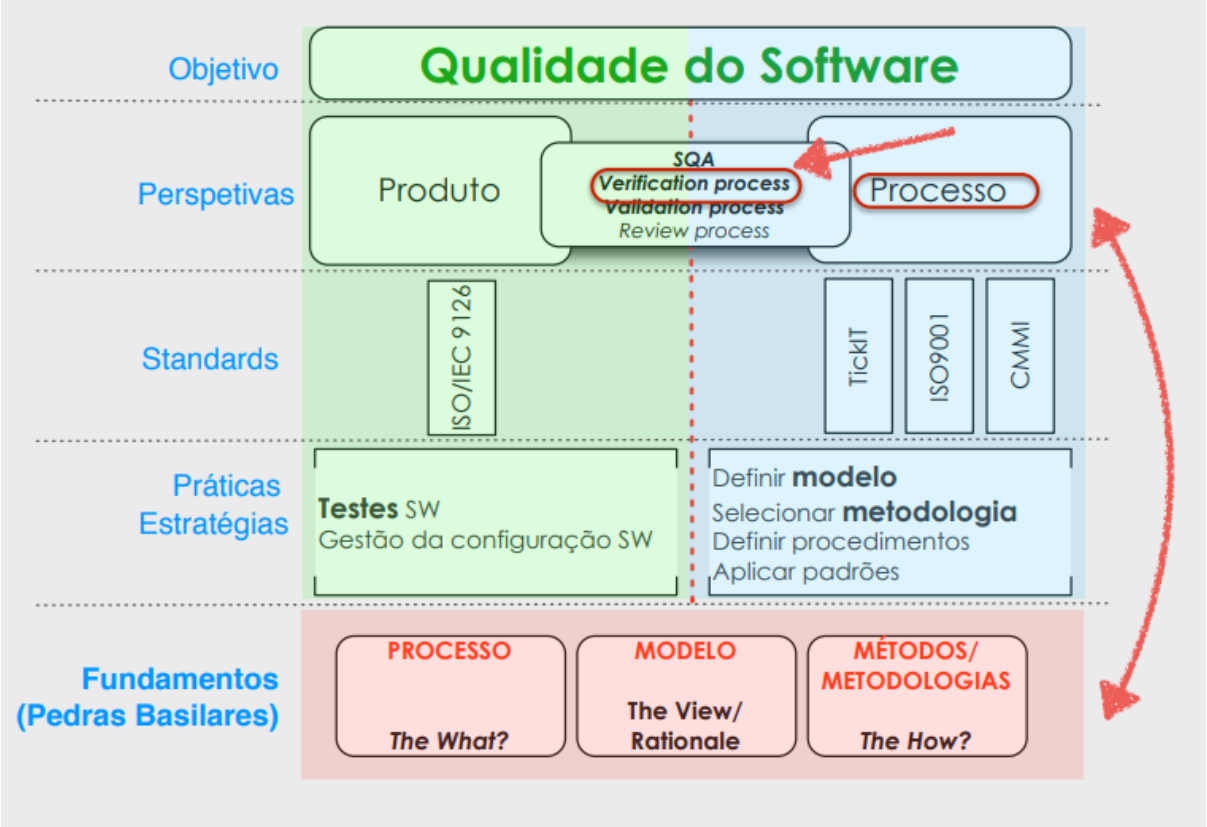


Processos do desenvolvimento de software

Os processos principais de desenvolvimento de software são:

1. Aquisição
2. Fornecimento
3. Desenvolvimento
4. Operação
5. Manutenção.

Para garantir a qualidade do software foram seguidas as práticas do roadmap representado abaixo



SDLC - Modelo do ciclo de vida do desenvolvimento do software

A implementação consiste na definição ou seleção de um modelo de ciclo de vida de software apropriado ao âmbito, magnitude e complexidade do projeto e na execução de documentação dos resultados, de acordo com o processo de documentação.

O ciclo de vida é a estrutura que contém processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema, desde a definição de seus requisitos até o término de seu uso.
Existem vários modelos de SDLC, no entanto o modelo escolhido para o desenvolvimento deste projeto foi o **Modelo em V**.

O modelo em V, também conhecido como modelo de Verificação e Validação, é um modelo SDLC onde a execução dos processos acontece de forma sequencial, em forma de V.
O V-Model é uma extensão do modelo cascata e baseia-se na associação de uma fase de teste para cada fase de desenvolvimento correspondente. Isso significa que para cada fase do ciclo de desenvolvimento, há uma fase de teste diretamente associada. Este é um modelo altamente disciplinado e a próxima fase começa somente após a conclusão da fase anterior.

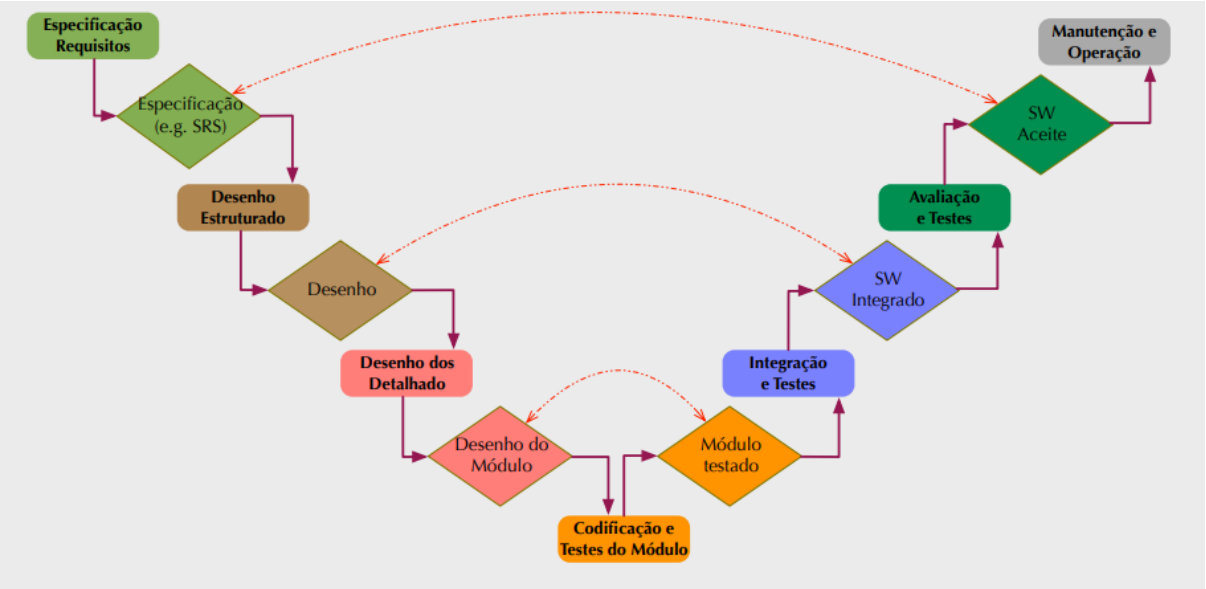
Vantagens:

- Os testes têm resultados de maior efetividade, uma vez que são testados contra requisitos e não contra especificações.
- Este modelo possibilita que se encontre erros durante os processos de se derivar especificações de requisitos.
- Ajuda a desenvolver novos requisitos.
- Melhora a qualidade do produto resultante, uma vez que valida o processo de engenharia de sistemas durante a integração do sistema.

Desvantagens:

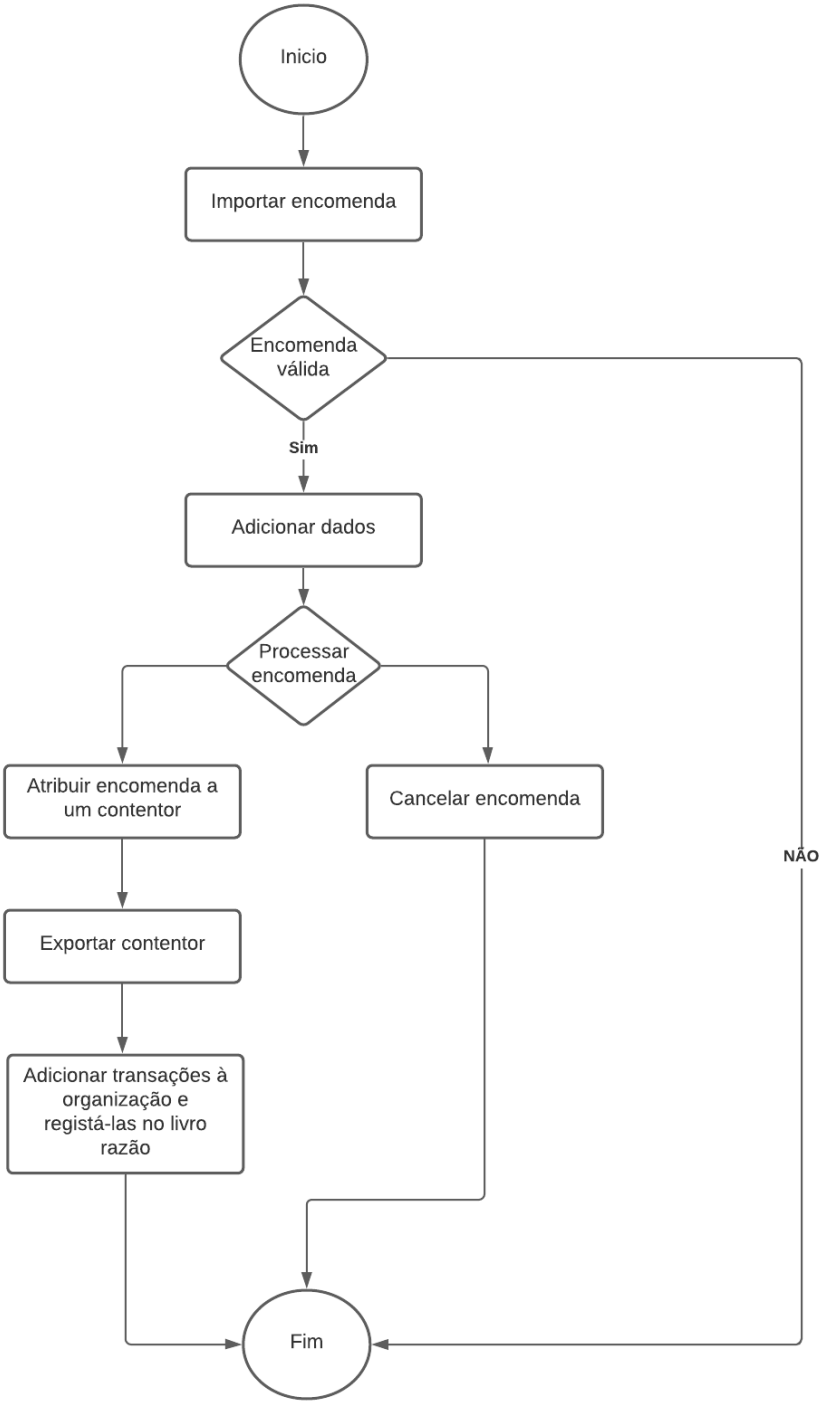
- Não considera o paralelismo que geralmente ocorre em projetos de maior complexidade.
- Não considera as diversas dimensões do projeto.
- Há ciclos de revisão em etapas tardias do processo, quando se encontra erros, a sua correção é pesada.

Fases do modelo em V



Abordagem do problema

A abordagem relativa à parte principal do problema do trabalho está representada no fluxograma abaixo:



5. Desenvolvimento e configurações

Ao longo do desenvolvimento do projeto foram utilizadas várias estratégias para a gestão do projeto.

Inicialmente foi criada uma Epic nomeada de ***Market for Business to Business*** que, possui 4 sub-epics que dizem respeito às diferentes fases de desenvolvimento do projeto, sendo elas:

- ➔ Análise do problema
- ➔ Módulo de Transações
- ➔ Módulo de Cálculo de custos
- ➔ Módulo de expedição

Tal como o nome indica, a primeira epic diz respeito à análise do problema onde foi estabelecido o product backlog, organizando e criando os respectivos user stories.

Após toda a fase de análise do problema, as restantes epics foram geridas da mesma forma, seguindo o seguinte fluxo:

- ➔ Criação de uma Sprint que visava estimar o tempo que seria necessário para terminar o desenvolvimento da epic em questão.
- ➔ Criar issues para cada uma das etapas a realizar, sendo elas:
  - ◆ Desenho estruturado
  - ◆ Desenho detalhado
  - ◆ Codificação das funcionalidades
  - ◆ Análise ECP/BVA
  - ◆ Criação dos casos de teste
  - ◆ Codificação dos casos de teste
  - ◆ WhiteBox
  - ◆ Manutenção

Este foi o fluxo seguido ao longo do desenvolvimento, respeitando a metodologia que foi escolhida, tal como requisitado no enunciado.

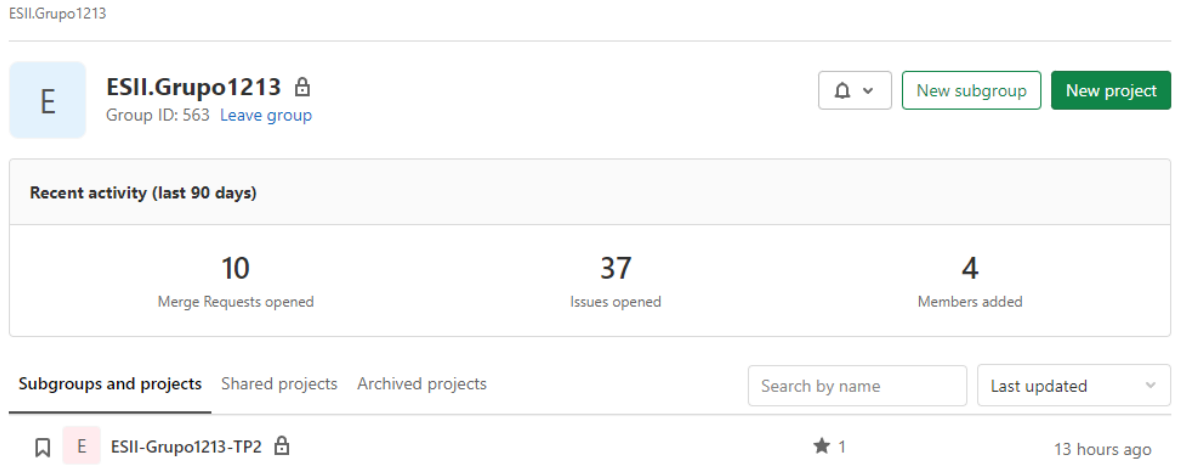
Foram criados dois Scrum Boards utilizados para gerir Change Requests e a evolução de cada etapa, de forma a assegurar a qualidade.

Ao longo do desenvolvimento de cada fase do projeto foram realizados diversos tipos de documentação que podem ser acedidos através da Wiki do projeto onde, se podem observar todos os documentos, bem como as suas respetivas versões.

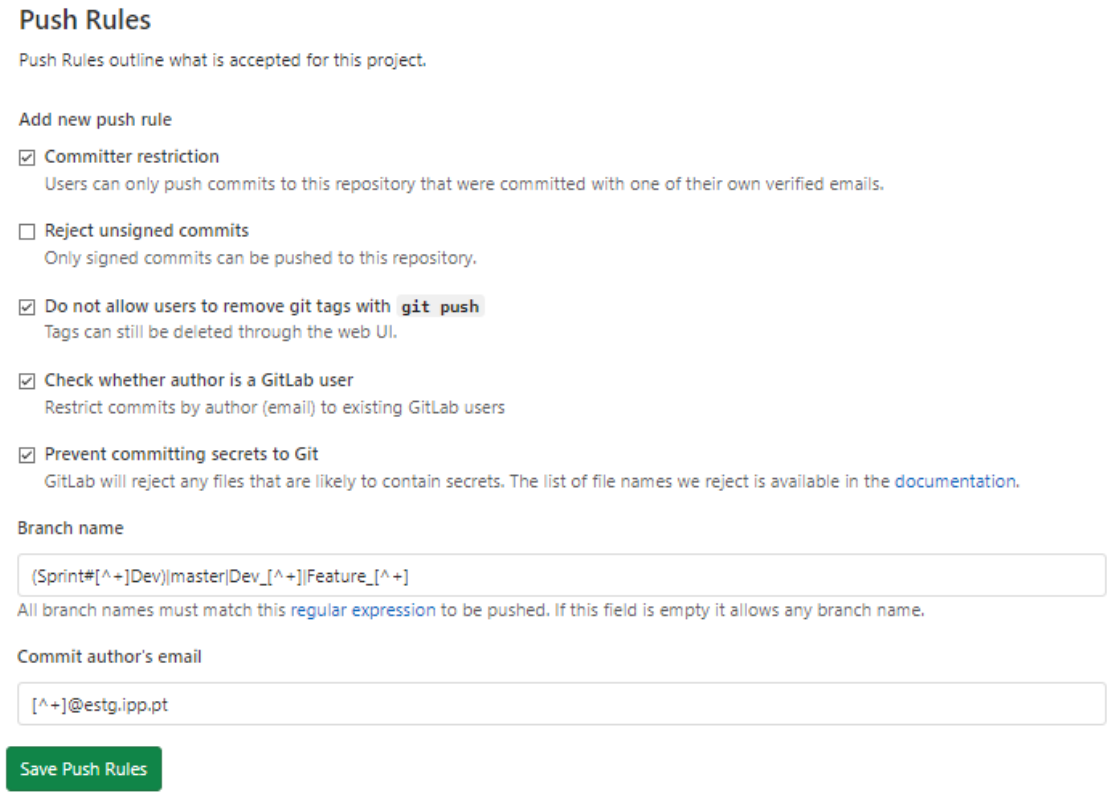
- ➔ https://gitlab.estg.ipp.pt/esii.grupo1213/esii-grupo1213-tp2/-/wikis/home

Configuração do GitLab

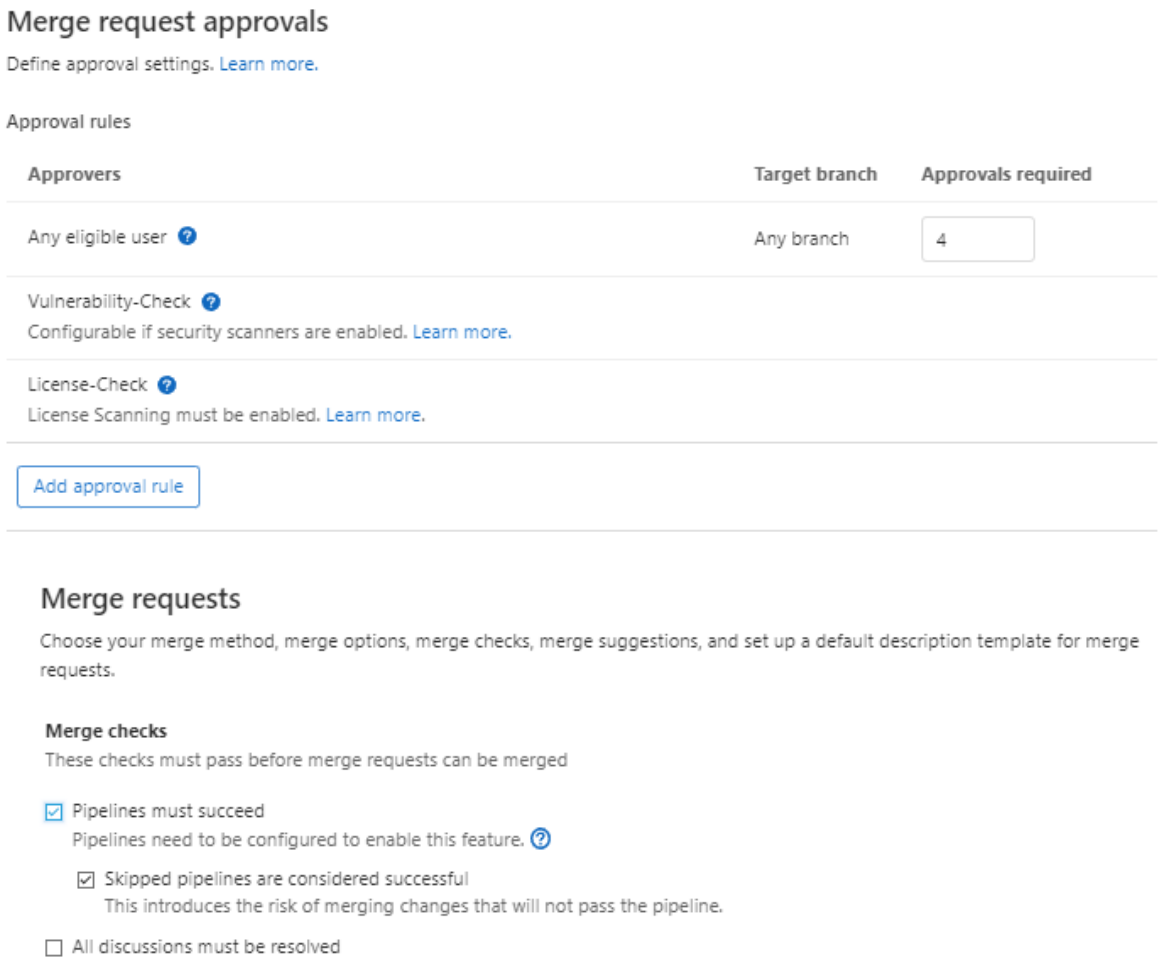
A configuração do gitlab começou pela criação de um grupo com os 4 elementos. Esse grupo foi criado para obtermos uma melhor gestão dos projetos usando Epic's e o Roadmap.



De seguida definimos os nomes específicos para os branches, e que os commits só poderiam ser efetuados por um utilizador com e-mail da ESTG.



Também definimos que para o merge ser efetuado necessita de aprovação dos 4 elementos do projeto e passar no pipeline ou ter sido efetuado skip ao pipeline.



Para o pipeline ser aprovado tem de passar em 3 job's:

- Build, o projeto tem de ser compilado;
- BackBox, o projeto tem de passar em todos os teste criados;
- WhiteBox, os testes do projeto tem de cobrir no mínimo 65% dos ramos.

Além disso, definimos que os testes só seriam executados quando houvesse um commit na pasta do projeto ou quando ouve-se um merge request.

Criamos também um grupo no teams com ligação ao gitlab usando a ferramenta Incoming Webhook.

Por fim criamos páginas na wiki para melhor gestão do projeto.

6. Ferramentas

Java - A linguagem Java foi a utilizada para este projeto

IntelliJ - IDE utilizado para a codificação e execução do código

GitLab - A plataforma Gitlab deve ser vista como uma ferramenta SCM que permite a gestão de código fonte, gestão de alterações e gestão de building e releases. Foi o repositório git utilizado para o desenvolvimento deste trabalho.

JUnit - Framework open-source para realizar testes unitários para código em Java. Permite a execução de testes de forma:

- Fácil
- Regular
- Fiável

Contém várias funcionalidades para testing entre elas a capacidade de testar cada componente de um programa de forma independente do resto do programa

Gradle - É uma ferramenta que permite integrar e automatizar várias tarefas relacionadas processo de desenvolvimento de software em várias linguagens de programação. O Gradle determina quais os componentes do projeto que estão atualizados, evitando a recompilação de todo o projeto.

PMD - O plugin PMD é um analisador de código fonte estático que encontra falhas de programação comuns, como variáveis não utilizadas, blocos catch vazios, criação desnecessária de objetos e assim por diante.

Jacoco - É uma biblioteca gratuita de cobertura de código para Java, que disponibiliza informações, tal como o nome indica, de taxas de cobertura, quer de ramos quer de instruções, identificando ainda quais as instruções/ramos que não foram cobertos.

7. Conclusão

Com este projeto tivemos a oportunidade de aprofundar conhecimentos no desenvolvimento de software garantindo a qualidade do mesmo através de técnicas lecionadas na Unidade Curricular de Engenharia de Software II e práticas da metodologia ágil SCRUM e do modelo V.

Foram ainda aplicadas técnicas de análise/estudo, testagem, nomeadamente testes de caixa preta e caixa branca que permitem assegurar uma maior robustez e resistência a erros.

8. Repositório Git

http://gitlab.estg.ipp.pt/esii.grupo1213/esii-grupo1213-tp2.git

9. Bibliografia

- https://about.gitlab.com/blog/2018/03/05/gitlab-for-agile-software-development/
- https://docs.gitlab.com/ee/user/project/integrations/microsoft_teams.html
- https://about.gitlab.com/blog/2018/03/05/gitlab-for-agile-software-development/
- https://pmd.github.io/latest/index.html
- Conteúdos da disciplina no moodle