Relatório da Sprint Scrum

Isabela Yabe
Lavinia Dias
Rodrigo Kalil

17 de outubro

1 Introdução

Este documento serve como um relatório da sprint realizada de **03** a **17** de outubro de 2024. Incrementado após a revisão da mesma.

2 Escolhas realizadas

2.1 DoR

Para ser considerado pronto para ser realizado, um item de backlog precisa ter:

- História do usuário bem definida – A user story deve estar claramente escrita, com os objetivos e necessidades especificados.
- Critérios de aceitação definidos – As condições que o trabalho precisa satisfazer para ser aceito pelo Product Owner.
- Dependências identificadas – Qualquer bloqueio ou dependência com outros times, tarefas ou sistemas deve ser levantado.
- Tamanho da história estimado – A equipe deve ter uma estimativa de tempo/esforço.
- Requisitos técnicos ou funcionais claros – Detalhes técnicos que a equipe precisará saber para implementar a história.
- Prioridade definida – A prioridade da tarefa deve estar alinhada com o Product Backlog.
- Validação de que o time entende a história – Todos os membros da equipe devem compreender os requisitos.

2.2 DoD

- Testes unitários e de integração realizados – A funcionalidade foi devidamente testada para garantir que funciona conforme esperado.
- Critérios de aceitação cumpridos – A história atende a todos os critérios de aceitação especificados no início.
- Código revisado e aprovado – Passou por revisão de código por outro desenvolvedor.
- Documentação completa - Foram adicionadas intruções correspondentes ao código do item.

2.3 XP

No projeto, nós acordamos em aplicar algumas práticas do Extreme Programming. O método ágil XP é muito usado por incluir práticas bem quistas. Esse método se baseia em pequenos incrementos ao produto, o que se interliga bem com a estrutura de sprints presente no scrum.

As práticas que pretendemos seguir durante o projeto são a de **incremental planning**, que consiste basicamente em quebrar histórias de usuários em tarefas e determinar sua prioridade; **test-first development**, na qual os teste para uma funcionalidade são definidos antes da implementação, garantindo o cumprimento direto das exigências com menor risco de bugs; **pair-programming**, em que duas pessoas participam do desenvolvimento de cada parte, evitando bugs e garantindo a corretude do código; **continous integration**, garantindo que cada nova funcionalidade criada seja integrada imediatamente ao sistema, realizando todos os testes.

O XP possui outras práticas, como **collective ownership**, na qual todos executam trabalho em cada parte do projeto, evitando especializações e atribuindo o sucesso como responsabilidade geral. Também há **refactoring**, em que os programadores devem se atentar para oportunidades de modificar a estrutura do código de forma a melhorá-la. Nós decidimos não focar diretamente nessas e outras práticas do XP por enquanto, dando mais ênfase nas mencionadas anteriormente. Isso porque não havia necessidade de realizar o collective ownership, já que cada item de backlog era simples e funcionava de forma razoavelmente similar na estrutura de programação, assim, não haveria grandes dificuldades de todos entenderem o código feito por outros devs. Na próxima sprint, se atentar a oportunidades, gostaríamos de utilizar o refactoring, pois consideramos importante para etapas posteriores, mas não identificamos a necessidade de utilizá-la agora visto a estrutura mais ampla do aplicativo ainda estar sendo definida.

2.3.1 Decorrer da sprint

O incremental planning foi realizado, com a quebra dos itens de backlog em tarefas menores a serem executadas pelo time.

O test-first development foi seguido no início, mas devido a abordagems confusas do problema nas primeiras tarefas, foi preciso se reunir e redefinir a percepção do time, comprometendo a validade de alguns testes definidos inicialmente. Assim, fizemos inicialmente os teste para algumas das classes e depois que compreendemos que a classe em python funcionaria apenas como um meio de interligação do banco de dados com o front-end, tentamos sem sucesso realizar testes para javacript com o módulo jtest. Após essa tentativa frustrada e o alinhamento de ideias entre o grupo, outros unittest tradicionais para as classes foram realizados ao longo do processo.

O pair-programming foi usado em diversos momentos, principalmente em tarefas que demandaram mais tempo e atenção, como a criação da classe de gerenciador do BD. Para facilitar o trabalho, definimos uma rede de apoio dentro no time, na qual cada membro é responsável por prestar suporte e verificar o trabalho de um outro.

A continuous integration não foi bem seguida nessa primeira sprint, pois os itens foram desenvolvidos em paralelo e todos foram unidos ao final, sem intervalos, com exceção da classe database manager que foi utilizada por todos os membros nos diferentes requisitos logo após sua criação.

A prática de collective ownership não foi aplicada, pois cada membro acabou tomando tarefas focadas numa parte do projeto.

O refactoring foi honrado no momento em que percebemos ter modelado o problema insatisfatoriamente, quando então refizemos algumas partes do produto.

3 Backlog da Sprint

3.1 Itens de backlog selecionados para a Sprint

- **RF33 - Reportar Problemas:** Como usuário quero poder reportar problemas com as vending machines ou com a rede social e o gestor pode visualizá-las para que sejam resolvidas.
- **RF14 - Perfil da Vending Machine:** Como usuário, quero acessar o perfil de cada vending machine para ver sua localização, produtos disponíveis e histórico de avaliações, me ajudando a escolher a máquina mais adequada.
- **NR A - Visualizar Perfil de Produto:** Como usuário, quero poder visualizar informações de um produto para decidir sobre sua compra.

3.2 Itens de backlog preparados para acionamento

- **RF7 - Comentar sobre Produto:** Como usuário, quero poder escrever um comentário sobre um produto para compartilhar minha experiência com outros usuários.
- **RF10 - Visualizar Estoque:** Como gestor, quero visualizar o estoque atual de cada produto em cada vending machine para tomar decisões informadas.
- **RF20 - Relatórios Básicos:** Como administrador ou membro da equipe de manutenção, quero gerar relatórios básicos sobre vendas, avaliações e estoque para monitorar o desempenho das vending machines.

4 Resultados da Sprint

- RF33: **Concluída** - Foi feita uma visualização. Foi feita uma lógica de registro das mensagens. O banco de dados é acessado para escrita.
- RF14: **Concluída** - Foi feita uma visualização. As informações das Vending Machines são acessadas por requisições ao banco de dados. Como a funcionalidade de avaliar ainda não existe, não aparece o histórico de avaliações.
- NR A: **Não Concluída** - Aqui foi criada uma classe DatabaseManager a fim de ser um intermediador entre o nosso banco de dados MySql e o python. Dessa classe surgirão as classes filhas que serão nossas tabelas de produtos, VM, reclamações, comentários... Também foi criada a classe DatabaseManagerCenter, para controlar globalmente o banco de dados. É nessa instância onde são instânciadas todas demais tabelas e toda manipulação é feita através dessa classe.

  Para o front-end foi criado de maneira análoga as mesmas classes como Client, como por exemplo o DatabaseManagerClient, para integrarmos através do flask o back e o front. A finalização do front-end ainda não foi finalizada, falta apenas corrigirmos erro de URL.
- RF7: **Concluída** - Foi feita uma visualização. Foi feita uma lógica similar à da RF33 para registro. O banco de dados é acessado para escrita e para a busca dos dados .
- RF10: **Concluída** - A visualização foi feita pelo carregamento dos dados em uma tabela em uma página html, os dados são carregados por meio de uma consulta ao banco de dados.
- RF20: **Não Concluída** - Nenhuma parte deste requisito foi desenvolvido na sprint. No entanto, a criação do banco de dados permite que ele seja desenvolvido com menos dificuldade nas próximas sprints.

5 Retrospectiva

O que funcionou bem

- O time se comunicou frequentemente, e seguiu o acordo em relação ao uso de testes e à rede de apoio.
- O time avançou na modelagem e implementação da base de dados, o que deve facilitar o trabalho futuro, pois muitos requisitos dependem dessa interação com os dados.
- Após estudos conseguimos entender bem o problema e estruturar o projeto de uma forma coesa.

O que pode ser melhorado

- No meio da sprint, houve dúvidas quanto à modelagem inicial, o que levou o time dev a buscar o scrum master. Chegou-se a um consenso, mas a modelagem inadequada de início causou atrasos e necessidade de refatoração de algumas partes do código. Dessa forma uma boa prática para as próximas sprints seria a pesquisa sobre os problemas de programação envolvidos no objetivo da sprint e formas de solução comuns,principalmente para lidar com o encapsulamento do código.

6 Conclusão

Nessa sprint, o time solidificou sua percepção sobre o problema e trabalhou na modelagem de classes e da base de dados.

Enquanto o desenvolvimento de alguns itens seguiu de forma autônoma e obteve uma entrega, o de outros, nos quais o time tentou desenvolver o banco de dados e seu acesso a partir de uma mesma base acoplada de código, não foi concluído por dificuldade do time em relação a essa base.

Alguns itens foram, pois, postergados, mas o melhor direcionamento da estruturação da base de dados e do acesso a ela deve tornar o restante da implementação deles mais rápida.