Relato de experiência de desenvolvimento do FaseCerta com o *framework Scrum*

Caio S. Ferreira, Luiz Miguel Da Silva, Daniel A. Moura, Francisco S. de Oliveira, Gabriel dos A. Lucim, Evanise A. C. Ruiz, Marcelo F. Terenciani, Frank W. C. de Oliveira

[1]Instituto Federal do Paraná (IFPR) – Campus Paranavaí
Paranavaí – PR – Brasil

20241pvai100300{35, 21, 23, 33, 05}@estudantes.ifpr.edu.br,
{evanise.ruiz, frank.willian, marcelo.terenciani}@ifpr.edu.br

Este resumo relata a experiência dos alunos do IFPR - Campus Paranavaí ao desenvolver um sistema *web* como parte das disciplinas de Construção de *Software*, Projeto Integrador e Interação Humano-Computador, do 3° ano do curso de Engenharia de *Software*. O trabalho tem o objetivo de inserir os alunos em um ambiente de trabalho similar ao encontrado no mercado de trabalho real, apresentando-os aos desafios e protocolos encontrados no processo de desenvolvimento de *software* com metodologias ágeis.

O projeto da equipe está sendo desenvolvido para auxiliar na gestão de uma empresa de pequeno porte de soluções elétricas, que até então realizava a maior parte das tarefas administrativas e operacionais através de sistemas terceirizados ou registros em papel. Até o fim do período de desenvolvimento, espera-se que o sistema permita a gestão dos clientes, funcionários, ordens de serviço, movimentações financeiras e serviços ofertados pela empresa.

Durante o desenvolvimento foi adotado o *framework Scrum*, baseado em ciclos iterativos e incrementais, com inspeção e adaptação contínuas [Schwaber and Sutherland 2020]. Neste *framework*, o cronograma é dividido em *Sprints*: ciclos de trabalho com 1 a 4 semanas de duração com reuniões de acompanhamento diárias. A equipe é composta por um PO (*Product Owner*) responsável pela ordenação do *Product Backlog* e pela comunicação com o cliente, um SM (*Scrum Master*) responsável por apoiar a aplicação do Scrum e auxiliar a equipe na identificação e remoção de impedimentos, e quatro desenvolvedores, responsáveis pelo desenvolvimento do código em si. No contexto do projeto, as *sprints* duraram de 4 a 6 semanas, dependendo do cronograma definido pelos avaliadores.

Os requisitos do sistema foram levantados principalmente por meio de reuniões e entrevistas com um especialista do domínio, com a comunicação sendo mantida também por meio do WhatsApp. As necessidades identificadas foram transformadas em histórias de usuário e organizadas no *Product Backlog* de um quadro Kanban mantido no *GitHub Projects*[1], contendo descrição, critérios de aceitação e tarefas de desenvolvimento [Cohn 2004, Patton 2014]. Nas primeiras *Sprints*, a equipe priorizou a construção do *front-end* e dos protótipos das funcionalidades, permitindo a validação das interfaces e dos requisitos antes da implementação e integração com o *back-end*.

[1]https://github.com/CS-PI-2026-FaseCerta

Depois do planejamento inicial do *Product Backlog*, foi realizada a primeira reunião de *Sprint Planning*, na qual o PO apresenta os itens de maior prioridade, conforme a ordenação do *Product Backlog*, e os desenvolvedores selecionam, em conjunto, aqueles que poderão ser desenvolvidos durante a *Sprint*, compondo assim o *Sprint Backlog*. No contexto do projeto, foram realizadas reuniões de acompanhamento inspiradas na *Daily Scrum* nos dias com aulas das matérias de Construção de Software e Projeto Integrador, ocorrendo, portanto, com frequência semanal em razão da organização das disciplinas. Nessas reuniões, os desenvolvedores relatavam o seu progresso e quaisquer impedimentos de trabalho. Essa periodicidade constitui uma adaptação do *Scrum* ao contexto acadêmico do projeto.

No fim de cada *Sprint*, foram realizadas as *Sprint Reviews*, nas quais a equipe apresentou e inspecionou, juntamente com os *stakeholders*, os resultados obtidos durante a *Sprint* (no contexto das disciplinas, estes são os professores e o especialista do domínio). No mesmo dia, a equipe realizava a *Sprint Retrospective*, uma reunião na qual discutia os resultados do último ciclo e possíveis melhorias para o próximo.

A ferramenta central usada para o desenvolvimento do projeto foi o *GitHub*, uma plataforma *online* de gestão de repositórios com versionamento *Git*. Todos os repositórios do projeto estavam vinculados à uma organização, permitindo que os desenvolvedores trabalhassem remotamente, às vezes desenvolvendo várias funcionalidades simultaneamente por meio de *branches*, ao mesmo tempo em que a plataforma oferece ferramentas de revisão para o SM e os avaliadores.

Além disso, o acompanhamento visual do projeto foi realizado por meio de um quadro Kanban no *GitHub Projects*. O quadro foi dividido em colunas para os diferentes estágios do desenvolvimento (“*Product backlog*”, “*Sprint backlog*”, “Em desenvolvimento”, “Revisão do *Scrum Master*”, “Revisão dos docentes” e “Entregue”). Isso permitiu que a equipe acompanhasse os responsáveis e o andamento das tarefas, mantendo todos cientes do progresso de cada *Sprint*. O *front-end* do projeto foi desenvolvido com React, com *deploy* na Vercel[2]. O *back-end* foi desenvolvido com *Spring Boot*, com *deploy* realizado na *Render*[3] por meio de contêineres construídos a partir de imagens *Docker*.

Os principais desafios foram a falta de tempo para desenvolvimento e o aprendizado simultâneo ao desenvolvimento. Com apenas um dia por semana dedicado exclusivamente ao desenvolvimento do projeto, algumas tarefas demoravam mais do que o esperado, já que outras responsabilidades dos estudantes ocupavam o cronograma. Além disso, conhecimentos necessários para o desenvolvimento do projeto estavam sendo ensinados simultaneamente a implementação, resultando em retrabalhos constantes.

Adicionalmente, a demora na validação uma funcionalidade ligada a múltiplas histórias de usuário, a desistência do curso de um dos membros da equipe, que reduziu o esforço total que a equipe era capaz de aceitar em cada *Sprint*. As adaptações feitas para o ambiente acadêmico também determinaram o SM como o único revisor obrigatório da equipe, possibilitando gargalos, e gerando a necessidade de comunicação frequente entre SM e PO, ou desenvolvedores e PO, para garantir o alinhamento entre os critérios de aceitação explicitados nas HUs e as expectativas reais do cliente.

[2]https://fase-certa-deploy-two.vercel.app
[3]https://fasecerta-backend-latest.onrender.com

A questão do tempo foi mitigada por meio de ajustes no escopo entre as *sprints*. A disparidade de conhecimento foi reduzida ao longo do ano com autoestudo e pela atribuição de tarefas considerando aquelas com as quais os desenvolvedores estavam mais familiarizados. Além disso, a equipe passou a considerar a possibilidade de retrabalho ao estimar o esforço das histórias de usuário, reduzindo riscos de atrasos imprevistos.

A desistência de um desenvolvedor exigiu uma maior revisão do escopo planejado e uma reorganização das atividades previstas para as *Sprints* seguintes. A revisão exclusiva pelo SM foi mitigada pela utilização do *Github Copilot* para análises preliminares do código desenvolvido, uma política determinada pelos docentes avaliadores. Esses desafios foram identificados a partir das discussões realizadas nas retrospectivas, do acompanhamento das atividades no GitHub Projects e das experiências relatadas pelos integrantes da equipe ao longo do desenvolvimento.

Como resultados parciais, em 4 *sprints* foram implementadas interfaces de autenticação e gestão de usuário, alteração e recuperação de senha, *Dashboard*, gestão de clientes, serviços, pedidos, e operações financeiras. As funcionalidades de *back-end* foram desenvolvidas para cadastro e autenticação de usuários, clientes, serviços e operações financeiras de despesas, e foram integradas com suas interfaces respectivas, totalizando 45 histórias de usuário e itens técnicos, sendo que mais de 70% do esforço do projeto até agora foi entregue nas *sprints* 3 e 4. Ao longo do processo, o *Scrum* contribuiu para a organização das atividades e para que os integrantes direcionassem seus esforços a tarefas específicas. Na percepção da equipe, os eventos e práticas adotados favoreceram a identificação de problemas, a comunicação entre os participantes e a organização do trabalho.

Antes do desenvolvimento do projeto, os membros da equipe possuíam pouca experiência com o *Scrum*, a aplicação dos conceitos de Interação Humano-Computador e a operação de *Git*/*GitHub*. O projeto não só proporcionou essas experiências, como também contribuiu para o trabalho em equipe e a comunicação dos estudantes, expandindo tanto o conhecimento técnico quanto o interpessoal. O uso do *Scrum* foi relevante para o crescimento dos estudantes, familiarizando-os com os eventos, papéis e desafios associados ao *framework*, bem como com a dinâmica de times com funções diferentes entre si, competência relevante para a atuação em equipes de desenvolvimento de software.

A experiência evidencia que o desenvolvimento de projetos acadêmicos em um contexto próximo ao profissional pode contribuir para a formação dos estudantes, permitindo vivenciar práticas, responsabilidades e desafios presentes no desenvolvimento de software. Nesse contexto, o *Scrum* auxiliou na organização e no acompanhamento das atividades da equipe.

Referências

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Patton, J. (2014). *User Story Mapping: Discover the whole story, build the right product*. O'Reilly Media Inc.

Schwaber, K. and Sutherland, J. (2020). *O Guia do Scrum: O guia definitivo para o Scrum: As regras do jogo*. ScrumGuides.org.