UniverSitter

Boas Prática de Engenharia de Software

Semestre: 2024.1

Professora: Antonia Diana Braga Nogueira

Alunos:

Luis Felippe Morais de Lima — 538605

Marcelo Mikael Pinheiro Lessa Peres — 536011

Renan Victor de Almeida Silva — 538428

Introdução

O **UniverSitter** é uma plataforma que visa conectar donos de pets e plantas a colegas cuidadores apaixonados, oferecendo uma alternativa segura e eficaz para o cuidado temporário desses companheiros. Com o crescente número de brasileiros que possuem animais de estimação e a popularidade crescente do cultivo de plantas, essa plataforma surge como uma solução prática para os universitários que precisam se ausentar de suas casas por longos ou, até mesmo, curtos períodos de tempo.

A aplicação de boas práticas de Engenharia de Software (ES) no desenvolvimento do **UniverSitter** foi fundamental para garantir a qualidade do sistema, a eficiência no desenvolvimento e a fácil manutenção futura. Este relatório descreve as etapas de planejamento e execução dessas práticas no projeto.

Posicionamento do Problema

O Problema de:	Não ter amparo para pets/plantas em períodos não letivos
Afeta:	Universitários locais
Seu impacto é:	Obstáculo para adoção de animais, universitários cada vez mais sozinhos.
Uma boa solução seria:	Uma plataforma que conecta possíveis cuidadores a estes universitários.

Posicionamento do Produto

Para:	Universitários locais
Que:	Que constantemente passam pela incessante busca de um bom cuidador para seus dependentes em períodos não letivos.
O: é um:	UniverSitter Rede social web para conexão de pessoas dentro do nicho alvo
Que:	Conecta os donos aos cuidadores de modo efetivo, claro e rápido, oportunizando um âmbito de livre acordo entre eles.
Ao contrário de:	DogHero, TrustedHouseSitter e/ou PetWalk
Nossa solução:	Estamos focados em conectar pessoas e na troca de favores, enquanto as outras plataformas estão oferecendo serviços terceirizados.

Planejamento das boas práticas

O planejamento da aplicação de boas práticas envolveu a definição de processos, padrões e ferramentas para cada fase do ciclo de vida do software. O foco principal foi garantir que o desenvolvimento fosse orientado por qualidade, escalabilidade e manutenibilidade. As principais diretrizes planejadas incluem:

- **Arquitetura Modular**: Utilização de uma arquitetura modularizada que permitisse a separação clara entre diferentes componentes, como autenticação, gerenciamento de cuidadores e interação com os donos de pets/plantas.
- **Boas Práticas de Código**: Definição de padrões de código utilizando as melhores práticas da linguagem escolhida (React, Node.js, entre outras), como o uso correto de identação, nomenclatura descritiva de variáveis, e divisão adequada de funções e componentes.
- **Controle de Versão**: Uso de Git como ferramenta de controle de versão, definindo uma política de branches para facilitar o trabalho colaborativo, com revisão de código através de pull requests.
- **Testes**: Implementação de testes unitários e de integração para garantir a robustez do sistema em seus componentes críticos, como o gerenciamento de cadastro de cuidadores e sistema de avaliação.
- **Gestão de Requisitos**: Definição de uma documentação clara de requisitos que pudesse ser iterada conforme feedback de usuários e stakeholders, além da manutenção de uma backlog com priorização de funcionalidades.

Execução das boas práticas

Durante a execução do projeto **UniverSitter**, a equipe seguiu um ciclo iterativo e incremental, com foco na entrega contínua de valor e na adaptação conforme o feedback obtido ao longo do processo. Abaixo, destacamos as boas práticas aplicadas em cada fase do desenvolvimento:

Scrum no ciclo de desenvolvimento:

Durante cada Sprint, o foco sempre foi a entrega de funcionalidades prioritárias e o cumprimento dos requisitos. A metodologia Scrum nos permitiu identificar pontos críticos a serem corrigidos rapidamente, enquanto mantínhamos o projeto avançando em direção às suas metas principais.

No início de cada Sprint, as tarefas eram quebradas em subtarefas e alocadas para membros da equipe de acordo com suas especialidades. A definição clara de metas para cada ciclo ajudou a gerenciar expectativas e possibilitou a entrega de resultados de forma contínua.

Arquitetura modular e Componentização:

A plataforma foi desenvolvida com uma arquitetura baseada em componentes, utilizando o React no front-end e Node.js no back-end. Cada funcionalidade principal (cadastro de cuidadores, listagem de cuidadores, sistema de avaliação, …) foi isolada em módulos separados, permitindo um desenvolvimento mais organizado e facilitando a manutenção futura.

Testes

Testes unitários foram implementados para funções essenciais, como validação de dados de cuidadores e cálculo da média de avaliações dos cuidadores (implementada nos componentes referentes ao **Sitter**). Além disso, testes de integração garantiram a correta interação entre os módulos do sistema, como o registro de cuidadores e a busca eficiente por eles, utilizando os dados armazenados.

Ferramentas de controle de versão e Integração contínua

O projeto foi versionado via GitHub, com o uso de branches para diferentes funcionalidades e correções de bugs. A prática de revisão de código foi adotada para garantir a consistência das contribuições ao código-base.

Avaliação de desempenho e segurança

O desempenho da plataforma foi continuamente avaliado, com foco na otimização de carregamento de dados e uso eficiente da API. Preocupações com segurança, como a encriptação de dados sensíveis, foram consideradas cruciais, especialmente no armazenamento de informações de usuários.

Desafios enfrentados

- **Integração do sistema de avaliações:** Um dos principais desafios foi a implementação de uma lógica eficiente para calcular e atualizar a coluna de *“rating”* na tabela do *sitter*, que dependia das avaliações fornecidas pelos usuários. A solução envolveu o uso de consultas SQL otimizadas, cachimbo para evitar sobrecarga no banco de dados e o uso de *Triggers* SQL para criar regras que automatizam esse cálculo a cada inserção, edição e/ou exclusão de valores no banco de dados.
- **Garantia de responsividade e usabilidade:** manter a interface fluída e responsiva para diferentes dispositivos exigiu ajustes frequentes com base em feedback de usuários e testes múltiplos navegadores.
- **Manutenção de fluxo constante:** A utilização do Scrum combinado com o quadro Trello foi crucial para manter um fluxo constante de entregas e garantir que tanto as

funcionalidades prioritárias quanto as correções menores fossem tratadas de maneira eficiente.

- **Firebase:** Ao decorrer do desenvolvimento do sistema viu-se necessário utilizar-se de recursos do Firebase. Devido a pouca experiência dos membros com a ferramenta, sentimos certa dificuldade na hora de implementar as funcionalidades necessárias. Com ajuda de colegas universitários, foi possível o uso correto da ferramenta e a funcionalidade conseguiu cumprir seus objetivos.

Conclusão

O desenvolvimento do **UniverSitter** seguiu rigorosamente práticas modernas de Engenharia de Software, combinando as metodologias ágeis Scrum e Kanban para otimizar o fluxo de trabalho. Isso permitiu à equipe entregar um produto inicial bom, com um ciclo de desenvolvimento ágil e transparente.

A aplicação dessas práticas, como arquitetura modular, testes e controle de versão, garantiu que o projeto fosse escalável e fácil de manter. O uso do Scrum para planejar Sprints e Kanban para gerenciar melhorias contínuas possibilitou entregas mais rápidas de valor ao cliente, sem comprometer a qualidade ou a estabilidade do sistema.

O projeto continua em desenvolvimento com busca nas melhorias baseadas nos feedbacks dos usuários e clientes. Nosso foco é construir algo fácil de utilizar e que seja simples e cumpra bem seus objetivos.