Universidade Fereral de Uberlândia
Bacharelado em Sistemas de Informação
Engenharia de Software

Engenharia de Software: Trabalho Final

Aluno(a): Lucas Fernando Borges

Aluno(a): Kelly Cristina Alves

Professora: Fabiola S. F. Pereira

Monte Carmelo
2021

Universidade Fereral de Uberlândia

Lucas Fernando Borges

Kelly Cristina Alves

Engenharia de Software: Trabalho Final

Trabalho apresentado à disciplina de Engenharia de Software do Curso de Bacharelado em Sistemas de Informação - FACOM - da Universidade Federal de Uberlândia - Campus Monte Carmelo.

Professora Fabiola S. F. Pereira.

Monte Carmelo
2021

1. Introdução

Este trabalho é feito a cerca de um sistema de anúncio de lixo eletrônico, DESCARTAQUI. O sistema nasceu pela necessidade de se fazer o descarte correto de materiais eletrônicos, bem como reaproveitar os que ainda estão em bom estado, ou reciclar peças. O descarte correto traz diversos benefícios para o meio ambiente, e consequentemente para a sociedade. O descarte desse tipo de lixo aumenta a cada ano, então é de suma importância que comecemos a nos preocupar com o destino destes materiais, promovendo ações de sustentabilidade para com o meio ambiente, de forma a impedir, ou ao menos diminuir, os impactos poluentes gerados pelo lixo eletrônico.

A regra de negócio do sistema funciona como um e-commerce, todavia com algumas diferenças. Os autores do sistema são as Empresas - que se cadastram para anunciar seu interesse em coletar os lixos eletrônicos ofertados - os Doadores - que se cadastram para ofertar um eletrônico - e o Administrador - que faz o controle dos cadastros realizados. A Empresa realiza seu cadastro, informando seus dados, para serem encontradas por pessoas que desejam descartar, ou doar, materiais eletrônicos. Os doadores realizam um cadastro, informando seus dados, para que possam anunciar um descarte,ou doação, e este ser encontrado por algum interessado. O cadastro de um novo eletrônico somente pode ser realizado caso o autor Doador esteja logado.

Para aqueles que desejam doar um eletrônico, sem realizar cadastro no sistema, basta pesquisar uma Empresa interessada em receber. Aqueles que desejam ofertar um eletrônico, basta fazer seu cadastro e anunciar o lixo. Aqueles que desejam receber/buscar por uma doação, seja Empresa ou não, basta acessar a listagem de ofertas de lixo do site.

Figura 1. Página inicial do sistema DESCARTAQUI.

2. Requisitos

2.1. Histórias

Para este projeto foram criadas 9 histórias, sendo elas:

Figura 2. Página que aborda informações do ”Projeto viver bem”no sistema DESCARTAQUI.

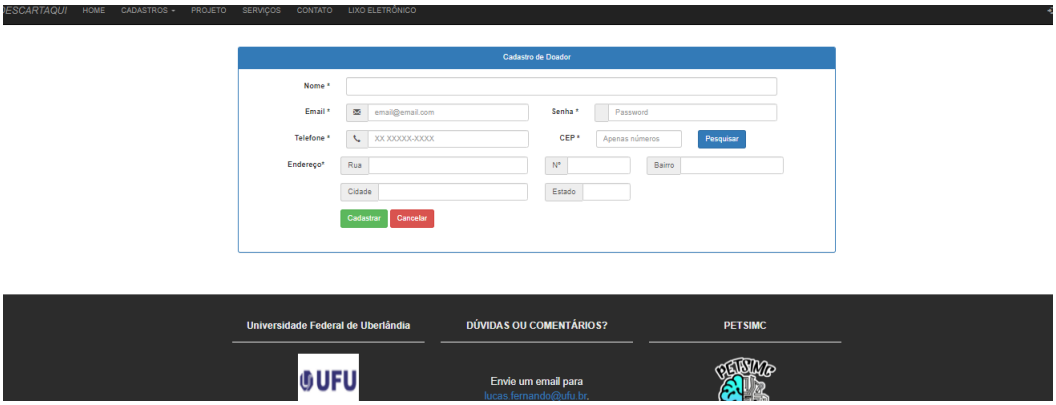

Figura 3. Página de cadastro do doador do sistema DESCARTAQUI.

1. Ajustes iniciais do projeto, tendo com subtarefas: criação do repositório no gitHub, definição da linguagem e tecnologias, definição final do que vai ser o sistema, escrita e organização das histórias no backlog;
2. Desenvolver BackEnd, tendo como subtarefas: desenvolver classes controller, criar classes referentes às entidades, criar tabelas no banco de dados, desenvolver telas de cadastro;
3. CRUD doador;
4. CRUD empresa;
5. CRUD lixo;
6. Criar FrontEnd, tendo como subtarefas: definir layout, definir conteúdo, criar imagens relacionadas.
7. Listagem de empresa;
8. Listagem aberta de lixo disponível;
9. Login(autenticação) de usuários com as seguintes subtarefas: definir usuários que farão login, ligar back,front e banco.

2.2. Requisitos não-funcionais

Requisitos não-funcionais são funcionalidades restritivas e quantificáveis e que possuem métricas como desempenho, espaço, confiabilidade, robustez, entre outros. O sistema em questão, DISCARTAQUI, não exige restrições para seu funcionamento. Trata-se de um site simples, de funcionamento tradicional e, por tal, não demanda requisitos não-funcionais.

3. Gestão do Projeto

3.1. Metodologia

A metodologia ágil usada no processo de desenvolvimento do DESCARTAQUI foi a Scrum. Esta é uma metodologia proposta por Jeffrey Sutherland e Ken Schwaber, que pode ser aplicada em diversos tipos de domínios, não apenas no âmbito de projetos de software. Métodos ágeis são baseados em iterações, isso significa que o desenvolvimento do projeto é dividido em iterações . Em Scrum, cada iteração é denominada Sprint, principal evento, que tem duração média de até um mês .Em cada sprint, histórias de usuários(funcionalidades) são implementadas.

Dessa forma, aplicando a metodologia para domínio do sistema DESCARTAQUI, o desenvolvimento foi dividido em sprints 3 e, em cada sprint, uma ou mais funcionalidades foram propostas e implementadas, como veremos mais adiante.

O time, como são chamadas as equipes de desenvolvimento nesta metodologia, é composto por duas pessoas - Lucas (aluno 1) e Kelly (aluno 2). O aluno 1 foi responsável pelo desenvolvimento do frontend, bem como parte do backend. O aluno 2 foi responsável pelo desenvolvimento de parte do backend, bem como o banco de dados.

3.2. Descrição Qualitativa

Nesta seção, veremos um esboço quantitativo dos resultados no que tange a gestão do projeto.

3.2.1. Sprints

Como já mencionado anteriormente, neste projeto tivemos um total de 3 sprints. Listaremos agora as sprints, acompanhadas de suas respectivas histórias:

1. **Sprint 1**. História(s): ajustes iniciais do projeto.
2. **Sprint 2**. História(s): desenvolver backend.
3. **Sprint 3**. História(s): desenvolver backend, CRUD doador, CRUD lixo, CRUD empresa, listagem aberta do lixo, login(autenticação) de usuários, criar frontend, listagem de empresas.

Ao observarmos as imagens, veremos que houve transbordo de tarefas entre o sprint 2 e o sprint 3. A história em questão é “desenvolver backend”. O desenvolvimento do backend é uma tarefa difícil, pois deve contemplar e conciliar dois elementos: frontend e banco de dados. Por tal, a definição de regra de negócio, bem como a integridade conceitual, devem estar bem alinhadas para dificultar o surgimento de conflitos durante o desenvolvimento individual dos módulos do projeto. Nesta etapa em questão, tivemos problemas quanto ao mencionado e, por isso não, foi possível completar a história do sprint 2 no tempo proposto.

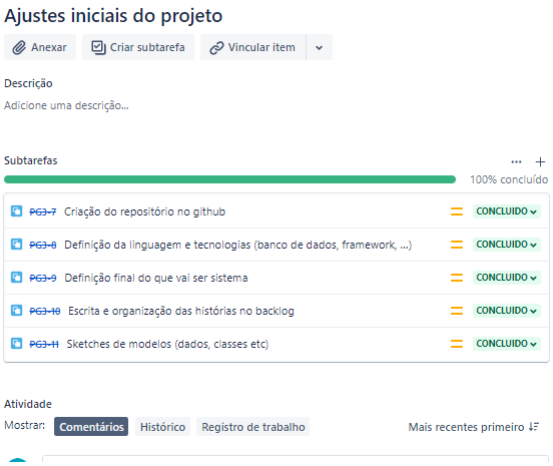

Figura 4. Quadro de tarefas e subtarefas da Sprint 1.

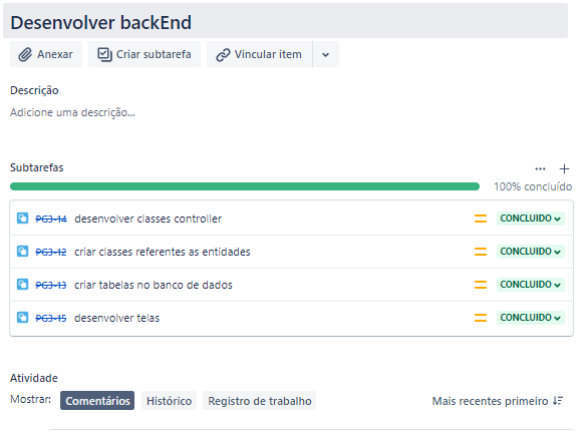

Figura 5. Quadro de tarefas e subtarefas da Sprint 2.

Items concluídas

Chave	Resumo	Tipo de item	Prioridade	Status
PG3-1	Desenvolver backEnd	História	Medium	CONCLUÍDO
PG3-2	CRUD lixo	História	Medium	CONCLUÍDO
PG3-3	CRUD empresa	História	Medium	CONCLUÍDO
PG3-4 *	Listagem aberta do lixo disponivel	História	Medium	CONCLUÍDO
PG3-5 *	Login(autenticação) de usuários	História	Medium	CONCLUÍDO
PG3-16	Criar frontEnd	História	Medium	CONCLUÍDO
PG3-19 *	Listagem de Empresas	História	Medium	CONCLUÍDO
PG3-22 *	CRUD Doador	História	Medium	CONCLUÍDO

Figura 6. Quadro de tarefas da Sprint 3.

3.2.2. Análise gráfica

Nesta seção analisaremos graficamente o andamento dos sprints e, a partir daí, analisaremos a produtividade do grupo. A ferramenta utilizada para o monitoramento das tarefas e acompanhamento do projeto chama-se Jira. Trata-se de um software desenvolvido em linguagem Java, pela empresa Atlassian. Todos os gráficos que aqui serão vistos foram

retirados deste sistema.

Para melhor interpretação dos resultados, faremos uma breve descrição acerca dos gráficos que aqui serão apresentados.

Através do gráfico de relatório do sprint , é possível acompanhar informações sobre o andamento dos trabalhos, como sua conclusão ou recolocamento no backlog.

O gráfico burndown nos fornece informações acerca do trabalho restante total. Isso pode possibilitar a projeção de prazo de entrega do sprint, auxiliando assim o gerenciamento do andamento.

1. **Sprint 1** Na Figura 7, temos o gráfico de relatório do sprint 1. Nele, podemos visualizar o tempo de duração da sprint 1, 9 dias, sua data de kick-off, ou seja, data de início, 29 de abril, e data de fim, dia 7 de maio. Podemos interpretar que, desde sua criação, o sprint não recebeu novas histórias além daqueles que já lhe eram incubidas.
   Na Figura 8 , temos o gráfico de burndown da sprint 1. Podemos notar que, no dia 30 de abril foi realizada uma alteração no escopo. Como o gráfico está subindo, isso significa que um item foi adicionado no sprint 1. O ponto de declínio no dia 7 de maio indica a conclusão de um item. Como a sprint em questão possui apenas uma história, o fim da reta indica o fim da sprint 1.
2. **Sprint 2** Na Figura 9, temos o gráfico de relatório do sprint 2. Nele, podemos visualizar o tempo de duração da sprint 2, 13 dias, sua data de kick-off, ou seja, data de início, 8 de maio, e data de fim, dia 21 de maio. Podemos interpretar que, desde sua criação, o sprint não recebeu novas histórias além daqueles que já lhe eram incubidas.
   Na Figura 10 , temos o gráfico de burndown da sprint 2. A reta em vermelho indica a contagem total de itens da sprint 2. A reta cinza indica o restante total de trabalho. Note que, entre os dias 10 e 19 de maio, a reta cinza decaiu. Isso indica que as tarefas do sprint 2 ou estão em andamento ou foram concluídas.
3. **Sprint 3** Na 11 Figura , temos o gráfico de relatório do sprint 3. Nele, podemos visualizar o tempo de duração da sprint 3, 7 dias, sua data de kick-off, 8 de junho, e data de fim, dia 15 de junho.
   Na Figura 12 , temos o gráfico de burndown da sprint 3. A reta em vermelho indica a contagem total de itens da sprint 3. Note que no dia 15 de junho ela sofreu um desvio tanto para cima, quanto para baixo. O desvio para cima indica a inserção de novas histórias no sprint 3. O desafio para baixo indica o encerramento da sprint 3.

4. Análise e Projeto do Software

4.1. Projeto Arquitetural e Projeto de Componentes

O projeto foi desenvolvido aos moldes do paradigma Orientado a Objetos. Assim sendo, ele possui 9 classes:

-ADM: modelo de administrador do sistema.

- BANCO: modelo responsável pelas comunicações com o banco de dados.

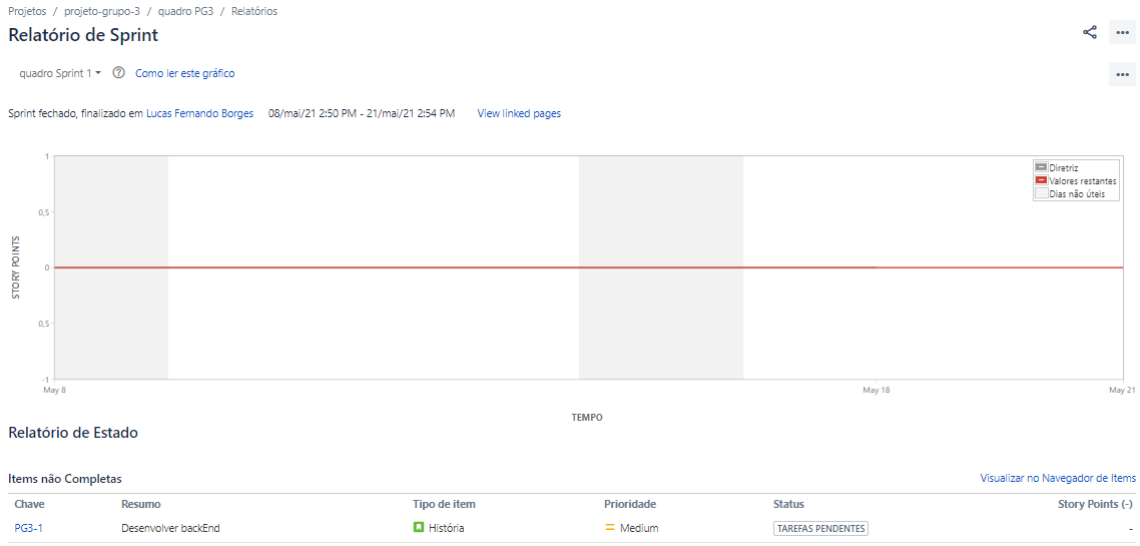

Figura 7. Grafico relatório da Sprint 1.

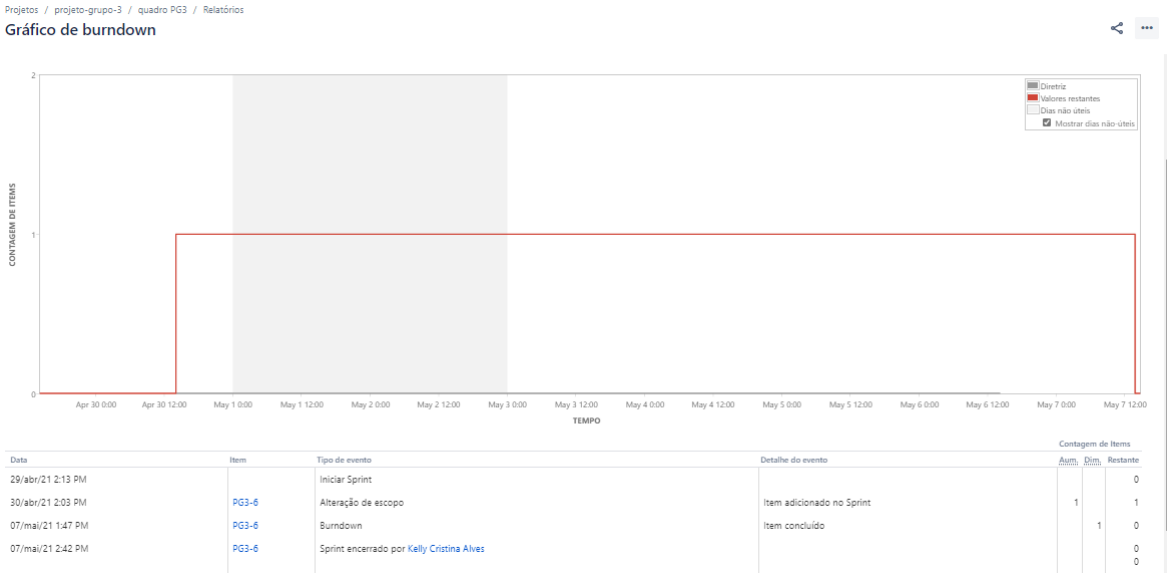

Figura 8. Grafico burndown da Sprint 1.

-ENDEREÇO: modelo de endereço para todas as entidades do banco. - ENTIDADE: classe abstrata, modelo para as entidades concretas do sistema.

-DOADOR: modelo de classe para um dos autores do domínio. representa o usuário que deseja realizar uma doação.

-EMPRESA: classe modelo para um dos autores do domínio. Esta classe representa a empresa parceira do projeto.

-LIXO: classe que representa o principal objeto do sistema.

-LOGIN: classe modelo para realização do login.

-CONTATO: classe modelo para todo contato recebido.

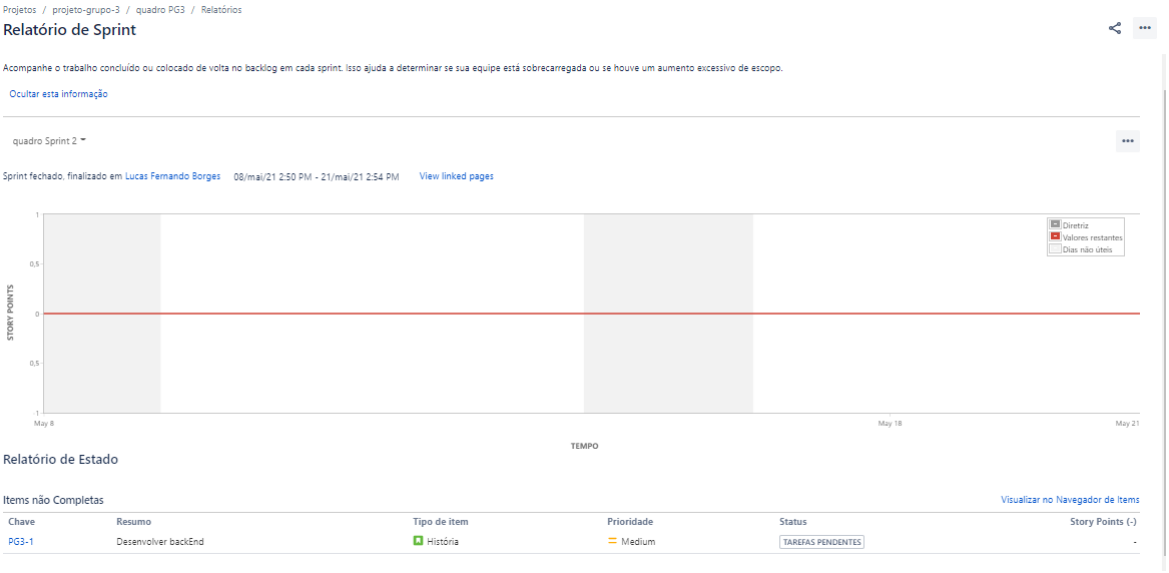

Figura 9. Grafico relatório da Sprint 2.

Figura 10. Grafico burndown da Sprint 2.

A arquitetura usada para este sistema foi a MCV Web. As páginas e classes foram divididas em três módulos: Model (Módulo com as classes-modelo do sistema. Elas são responsáveis pela comunicação com o banco de dados), Controller(Módulo que recebe as requisições da interface gráfica por protocolo HTTP e as repassa para o Model) e View(Módulo de comunicação com o usuário. Através dele, o usuário realiza uma requisição).

4.2. Propriedades e princípios de Projeto

4.2.1. Propriedades

Em relação a integridade conceitual do código, a escrita do backend foi padronizada. Foram utilizados os mesmos frameworks em todas as páginas. As telas com saídas de

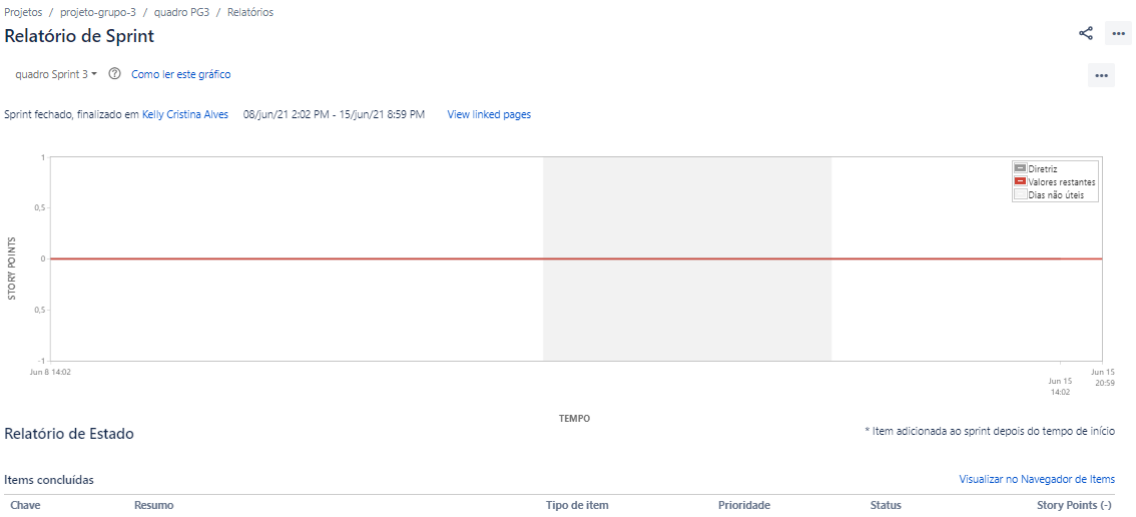

Figura 11. Grafico relatório da Sprint 3.

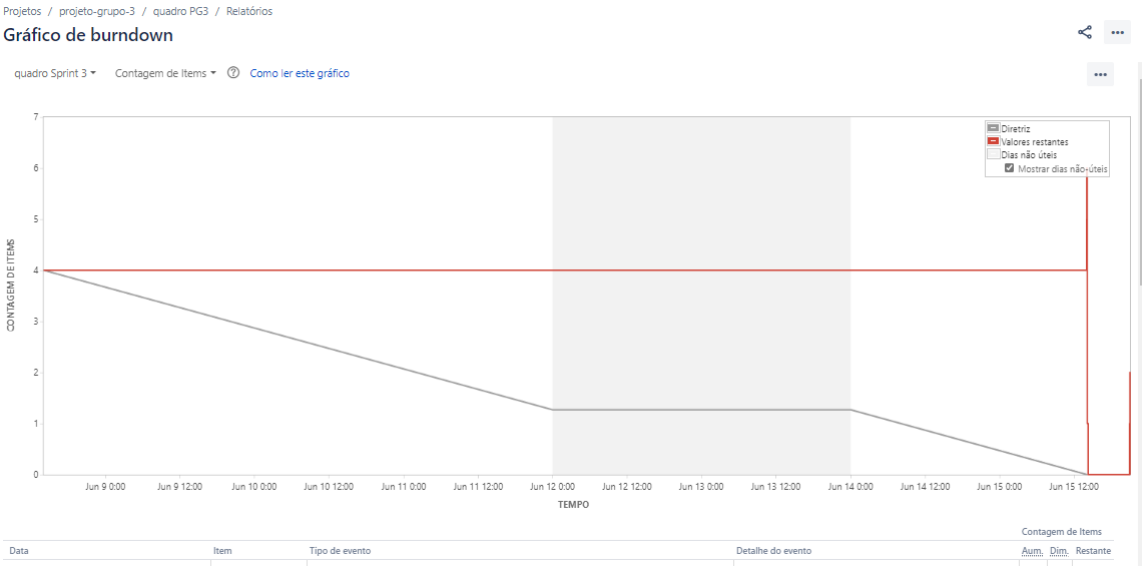

Figura 12. Grafico burndown da Sprint 3.

informação(tabelas), formulários e botões seguem o mesmo padrão gráfico (Figura 15).

O que tange o ocultamento de informação, todas as classes possuem seus atributos privados, construtores e métodos getters e setters(Figura 16).

Buscamos o máximo de coesão possível, dentro dos limites de nosso conhecimento. Trata-se de um sistema simples, dessa forma não há muito que possa gerar problemas. Todavia, buscamos controlar as funções de cada classe para que elas fizessem apenas o que lhes era proposto(Figura 17).

Existem alguns acoplamentos neste projeto, porém buscamos o fazer de uma boa forma. Exemplificando, existe um acoplamento entre as classes Endereço e Entidade não feita via interface, mas a classe Endereço é uma classe estável.

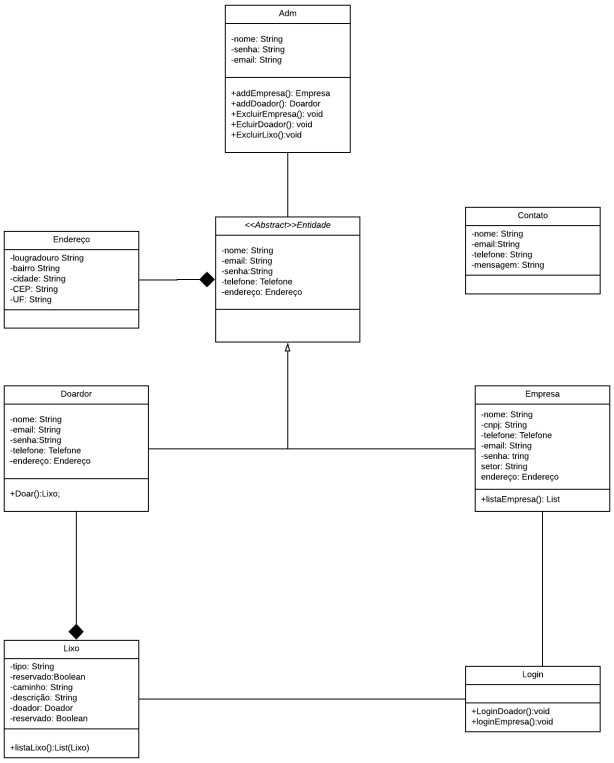

Figura 13. Diagrama de classes do projeto DESCARTAQUI.

4.2.2. Princípios

Para este projeto, buscamos seguir os seguintes princípios de projeto:

1. Responsabilidade única. Aplicamos este princípio para chamadas de métodos entre classes. Isso significa que, por exemplo, ao solicitar uma busca à classe banco, ela apenas retornava a busca. A classe que a chamou passava a informação ao controller e, este, repassava à interface. Isso ajudou a aumentar a coesão do código(Figura 18).
2. Prefira Composição a Herança. Este princípio foi um dos focos do desenvolvimento. As classes que se relacionam, possuem um objeto do tipo da dependência. Exemplificando, existe um acoplamento entre as classes Endereço e Entidade: uma Entidade "possui" um endereço.
3. Demeter Evitamos longas cadeias de chamadas, através do acoplamento entre as classes. Isso contribuiu para o ocultamento da informação. Aberto/Fechado O uso dessa diretriz pode ser vista pelo uso da classe abstrata na criação de outras entidades no projeto. O uso desse tipo de herança é um bom passo a se adotar para a extensibilidade do sistema (Figura 19).

5. Testes e Qualidade

Como já mencionado, trata-se de um clássico sistema de cadastros e listagem de informações. Aqui, as funções presentes apenas auxiliam para com o ocultamento da informação. Em suma, as funções do sistema são getters e setters. Como se sabe, não são

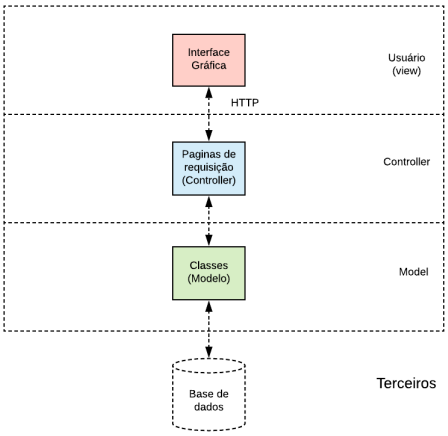

Figura 14. Diagrama de módulos do projeto DESCARTAQUI.

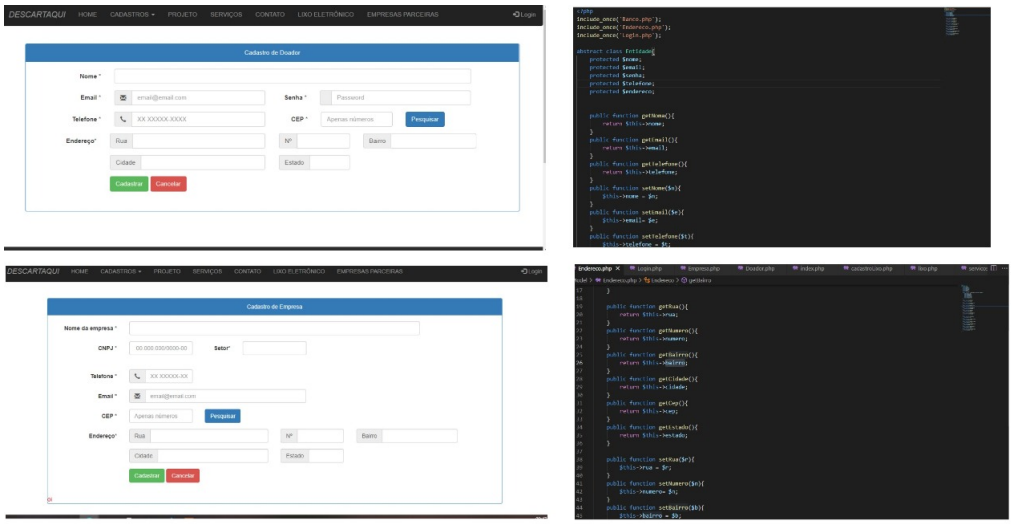

Figura 15. Exemplo de aplicação de Integridade conceitual.

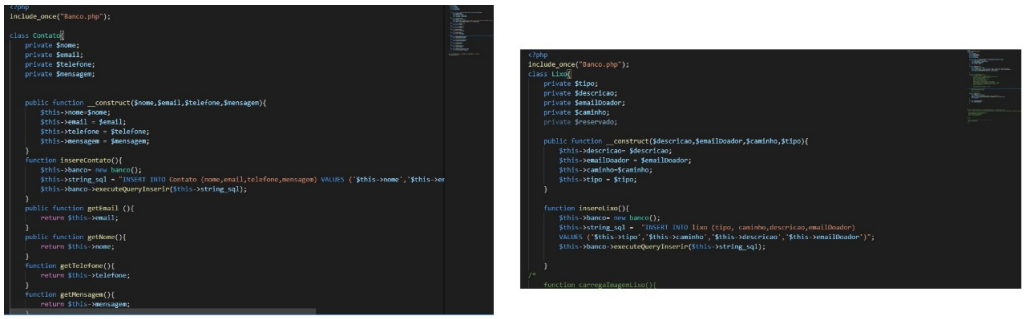

Figura 16. Exemplo de ocutamentento de informações.

feitos testes para estes tipos de função. Por isso, o sistema em questão não foi submetido a testes.

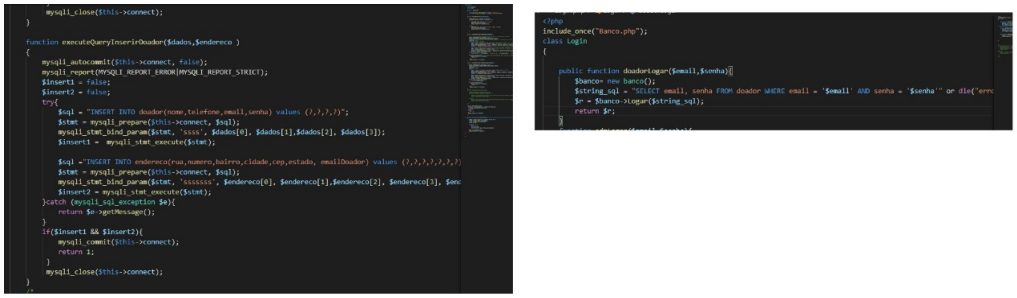

Figura 17. Exemplo de Coesão entre as classes.

Figura 18. Exemplo de aplicação da propriedade de Responsabilidade única.

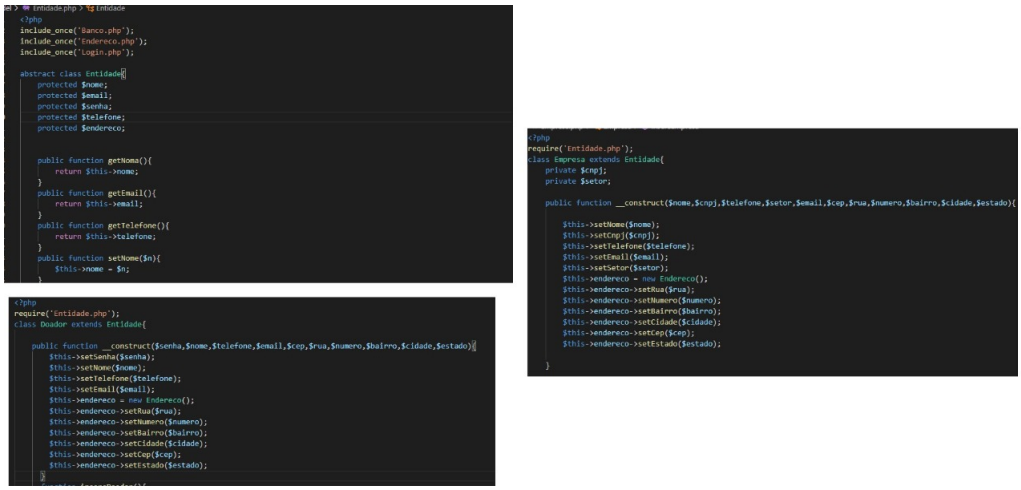

Figura 19. Exemplo de aplicação da propriedade Aberto/Fechado.

6. DevOps

O processo de implantação foi ligeiramente rápido. O DESCARTAQUI já era uma plataforma em desenvolvimento, e os trabalhos acerca deste projeto foram em prol da mudança de paradigma, aplicação das propriedades e diretrizes de bons projetos. Primeiramente buscou-se melhorias mediante a regra de negócio. Após definirmos tal, o processo de modelagem do sistema foi realizado. Então, a partir daí, a implementação do novo pa-

radigma, através da criação das classes, teve início. Paralelo a isto, tentamos aplicar as diretrizes de bons projetos, bem como suas propriedades. Por fim, com o backend pronto, a etapa de ligação entre o frontend, que já estava pronto, foi realizada. Não tivemos datas de entregas de etapas bem definidas, e não usamos as práticas de CI/CD.

7. Conclusão

O trabalho foi importante tanto no aprendizado de novos aspectos, quanto no resgate de conhecimento. Isto porque, para alicerçar bem o projeto, precisamos primeiramente revisitar conceitos de modelagem de software. Isso nos ajudou a compreender melhor o problema e visualizar possíveis soluções. A prática na aplicação de princípios e diretrizes, fixou com excelência estes conceitos, que antes eram abstratos em nosso entendimento. A aplicação de uma arquitetura nos proporcionou o entendimento sobre a organização de um projeto, que outrora, nunca foi questionado.

Nosso maior problema foi seguir fielmente a metodologia Scrum. Ainda não estamos acostumados com o desenvolvimento remoto, então nos adaptar a essa realidade, bem como adaptar nossos horários, foi desafiador.

A dificuldade, em suma, foi realizar a mudança de paradigma, já que o projeto escolhido estava em desenvolvimento. A aplicação das diretrizes e propriedades de projeto também geram algumas dúvidas, mas que contribuíram para o enriquecimento de nosso conhecimento.