



INSTITUTO TECNOLÓGICO DE AERONÁUTICA
DIVISÃO DE CIÊNCIA DA COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE SOFTWARE

**PROJETO GISLENE: RELATÓRIO TÉCNICO DA
METODOLOGIA ÁGIL APLICADA**

Gabriela Lima
Lindemberg Teixeira

São José dos Campos - SP
Junho de 2018

**Gabriela Lima
Lindemberg Teixeira**

**PROJETO GISLENE: RELATÓRIO TÉCNICO DA METODOLOGIA ÁGIL
APLICADA**

Trabalho para o resumo e explicação das atividades realizadas ao longo do semestre no projeto Gislene, seguindo os moldes da metodologia Ágil e dos conceitos ensinados em sala de aula

Professor: Inaldo Capistrando Costa

SUMÁRIO

1	PLANEJAMENTO DO PROJETO	4
1.1	TAREFAS A SEREM REALIZADAS	4
1.2	DIVISÃO DE TAREFAS	5
2	USER STORIES	6
2.1	ÉPICOS	6
2.2	HISTÓRIAS DO USUÁRIO	6
2.3	MÉTRICAS ADOTADAS	7
3	PLANEJAMENTO DAS SPRINTS	10
3.1	MÉTRICAS DE PLANEJAMENTO	10
3.2	GRÁFICOS DE PROJETO	10
3.2.1	Gráfico de Burndown do projeto	10
4	FERRAMENTA DE TESTES	13
4.1	FERRAMENTA ESCOLHIDA	13
4.2	TESTES REALIZADOS	13
5	LIÇÕES APRENDIDAS	15

1. PLANEJAMENTO DO PROJETO

1.1. TAREFAS A SEREM REALIZADAS

Gislene é um projeto para o desenvolvimento de um aplicação Web para georreferenciamento seguro para a Força Aérea em uma rede restrita, sendo voltado para a criação, manipulação e associação de objetos com características a serem definidas pelo administrador do software, criando rotas de voo, autorizações e áreas de tráfego. O aplicativo deveria estar dotado de uma interface gráfica que facilitasse o uso dos mapas e objetos armazenados no banco de dados.

Tendo em vista o código legado fornecido pela turma anterior, decidiu-se que seria melhor refazer o projeto, uma vez que entender o código e corrigir os problemas remanescentes exigiria muito tempo e esforço por parte da equipe.

Para as linguagens de programação do projeto foram escolhidas HTML, CSS e Javascript para o front-end, Ruby on Rails para o back-end e PostgreSQL para o banco de dados.

A partir da primeira reunião com o product owner e do relatório final fornecido pela turma anterior, obteve-se o seguinte backlog para esse projeto:

- Marcar pontos no mapa
- Estabelecer conexão com o banco de dados
- Editar pontos
- Criar relações entre pontos
- Cadastrar usuário com login e senha
- Buscar objetos a partir de parâmetros
- Importar e Exportar dados

Essa lista, além da interface gráfica amigável e back-end seguro para ser usado em uma intranet da FAB, compõe os requisitos do projeto.

Ao longo do semestre, durante as várias reuniões com o Product Owner, houveram algumas mudanças no escopo do projeto. Entre essas mudanças está a adoção do modelo AOM – Assessability Object Model –, que é responsável por possibilitar a modificação da estrutura de classes e relações dos objetos implementados em tempo de execução pelo administrador do projeto. Essa tarefa foi considerada prioritária para a conclusão do projeto e tomou precedência sobre itens do backlog, como a importação/exportação de dados e busca de objetos.

Para agilizar a implementação dessa aplicação, decidiu-se utilizar como base o OSM – Open Street Map – que é uma aplicação de código livre, open-source, para objetos georreferenciados.

1.2. *DIVISÃO DE TAREFAS*

Devido aos diferentes graus de prioridade atribuídos às diversas funções do projeto, a divisão das tarefas não foi constante, mudando a cada sprint conforme as necessidades das tarefas e histórias escolhidas.

Para os primeiros Sprints, onde, uma vez que o projeto estava sendo começado do início, precisava-se implementar toda a estrutura básica do aplicativo, dispensando, assim, uma interface gráfica robusta e complexa, um maior número de pessoas foi designado para o back-end e para o banco de dados: uma pessoa ficou responsável pelo front; quatro pelo back-end; e duas pelo banco de dados.

Após essa fase inicial, três pessoas ficaram responsáveis pelo front-end – Gabriela Lima, Gustavo Nahum e Lindemberg Teixeira – e as demais – Davi Hasuda, Dylan Sugimoto, Eduardo Henrique e Gabriel Adriano – pelo back-end, já que o banco de dados não-relacional (postgresql) já havia sido implementado e estava completamente funcional para o escopo do projeto.

Essa configuração se manteve até o fim do projeto, uma vez que a mudança de prioridade do modelo AOM pelo Product Owner exigiu maior esforço no back-end e na interface entre front e back.

2. USER STORIES

2.1. ÉPICOS

A partir do backlog do produto, apresentado anteriormente, o grupo desenvolveu as seguintes Epics para o projeto:

- Permitir que um usuário autorizado manipule objetos
- Criar sistema de cadastro de usuários
- Estabelecer a conexão com o banco de dados
- Permitir que administradores modifiquem a estrutura de objetos do programa
- Ser independente da conexão da internet

Esses cinco épicos descrevem muito bem o projeto e fazem a cobertura de todos os itens do backlog apresentado. Além disso, nenhum deles pode ser dividido em dois, ou seja, cada um cobre apenas um aspecto do projeto.

2.2. HISTÓRIAS DO USUÁRIO

Utilizando os épicos escolhidos para o projeto, tomando um a um e os fracionando em porções menores – user stories –, que cobrem o suficiente para serem implementadas em uma única sprint.

A lista de User Stories definidas para o projeto está apresentada abaixo.

- Como usuário autorizado, quero poder marcar pontos no mapa
- Como usuário autorizado, quero poder editar objetos
- Como usuário autorizado, quero poder criar relações entre objetos
- Como usuário não autorizado, quero poder fazer meu cadastro
- Como usuário autorizado, quero poder visualizar e ocultar pontos no mapa
- Como usuário não autorizado, quero poder fazer login
- Como usuário autorizado, quero poder fazer logoff
- Como administrador, quero ter uma interface que me permita criar novos objetos, hierarquias e atributos

- Como usuário autorizado, quero poder buscar objetos a partir de parâmetros
- Como usuário autorizado, quero poder importar e exportar dados em KML
- Como usuário e futuro desenvolvedor, quero uma documentação precisa e concisa
- Como dono do sistema, quero um rápido deployment do produto

Essas histórias se mostraram bastante satisfatórias para o escopo do projeto, uma vez que não foi necessário realizar grandes mudanças ao longo do semestre. Assim, pode-se afirmar que as histórias escolhidas satisfazem muito bem a metodologia proposta.

A Figura 1 apresenta as histórias de usuário criadas pelo grupo no padrão Kanban da plataforma de gerenciamento de projeto Taiga.



Figura 1. Print screen das histórias de usuário conforme apresentadas no taiga

2.3. MÉTRICAS ADOTADAS

Para o andamento do projeto de acordo com a metodologia Ágil, decidiu-se quantificar as histórias de usuário e as tasks provenientes de cada uma a partir de um simples sistema de horas. Cada story point atribuído no projeto estava relacionado a 1 homem-hora.

Por estar utilizando o site tree.taiga.io para gerenciar o projeto e as sprints, realizou-se a atribuição de pontos de cada story e cada task em quatro categorias disponíveis na plataforma: UX; Design; Front; e Back.

Como o projeto do semestre seria desenvolvido em 8 sprints, cada uma quinzenal e com cada aluno disponível para trabalhar durante 10 horas – conforme previsto na ementa do curso –, havia um total de 510 horas a serem distribuídas no projeto – incluindo horas de estudo.

A divisão da carga horária foi feita da seguinte forma:

- Como usuário autorizado, quero poder marcar pontos no mapa.
UX: 3
Design: 2
Front: 10
Back: 5
- Como usuário autorizado, quero poder editar objetos
UX: 1
Design: 1
Front: 5
Back: 2
- Como usuário autorizado, quero poder criar relações entre objetos
UX: 3
Design: 3
Front: 10
Back: 8
- Como usuário não autorizado, quero poder fazer meu cadastro
UX: 1
Design: 1
Front: 2
Back: 8
- Como usuário autorizado, quero poder visualizar e ocultar pontos no mapa
UX: 1
Design: 1
Front: 3
Back: 2
- Como usuário não autorizado, quero poder fazer login
UX: 1
Design: 1
Front: 3

Back: 5

- Como usuário autorizado, quero poder fazer logoff
UX: 1
Design: 1
Front: 2
Back: 3
- Como administrador, quero ter uma interface que me permita criar novos objetos, hierarquias e atributos
UX: 8
Design: 8
Front: 20
Back: 20
- Como usuário autorizado, quero poder buscar objetos a partir de parâmetros
UX: 2
Design: 1
Front: 2
Back: 5
- Como usuário autorizado, quero poder importar e exportar dados em KML
UX: 1
Design: 1
Front: 2
Back: 3
- Como usuário e futuro desenvolvedor, quero uma documentação precisa e concisa
UX: 0
Design: 0
Front: 5
Back: 5
- Como dono do sistema, quero um rápido deployment do produto
UX: 0
Design: 0
Front: 0
Back: 8

A carga horária em cada uma dessas stories não leva em consideração tarefas de estudo. Ao definir tasks de cada user story, o tempo de estudo foi levado em consideração, mas, para efeito de gerenciamento de projeto, cada user story só apresenta o tempo estimado prático de implementação.

3. PLANEJAMENTO DAS SPRINTS

3.1. MÉTRICAS DE PLANEJAMENTO

Durante o planejamento das sprints, buscou-se realizar reuniões com o Product Owner a cada 2 sprints para apresentá-lo o andamento do projeto e verificar a necessidade de alguma alteração no escopo do projeto. Durante algumas dessas reuniões, em especial a que ele solicitou a alteração da prioridade do modelo AOM, o planejamento das sprints seguintes teve de ser alterado para satisfazer as novas exigências.

O grupo realizou reuniões quinzenais para apresentação do progresso individual e planejamento da próxima sprint. Entretanto, através de comunicação virtual – Whatsapp –, foi feito um acompanhamento mais fracionado do progresso do projeto, possibilitando a montagem dos gráficos de burndown de sprints individuais, em vez de apenas o gráfico do projeto completo.

3.2. GRÁFICOS DE PROJETO

3.2.1 Gráfico de Burndown do projeto

O gráfico de burndown do projeto inteiro está apresentado na Figura 2

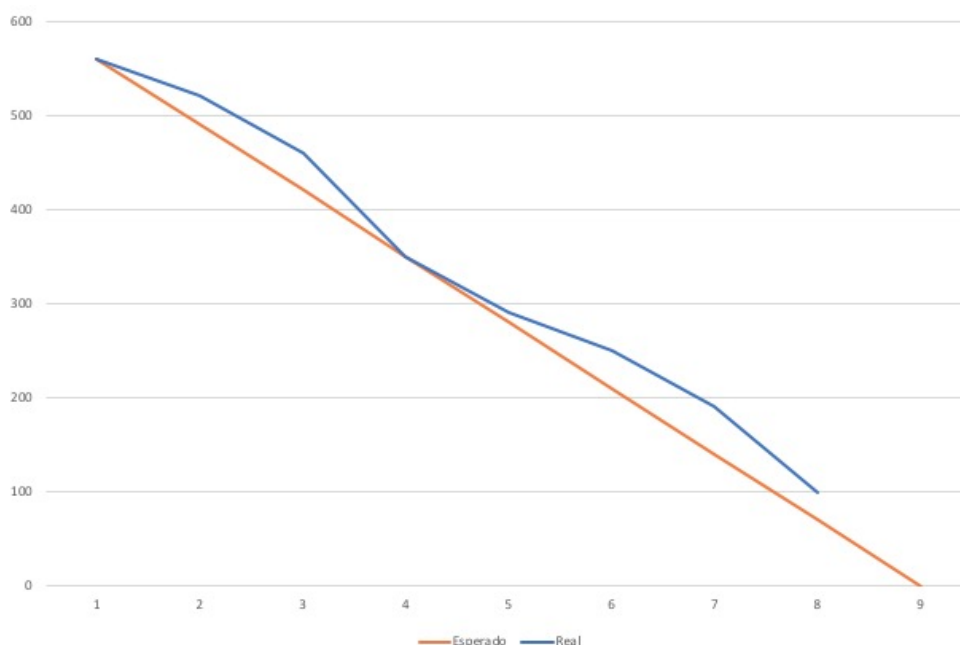


Figura 2. Gráfico de burndown para todo o projeto

Os gráficos apresentados abaixo, das Figuras 3 a 8 são referentes ao burndown das sprints.

A figura 3 apresenta o gráfico de burndown para a Sprint 2.

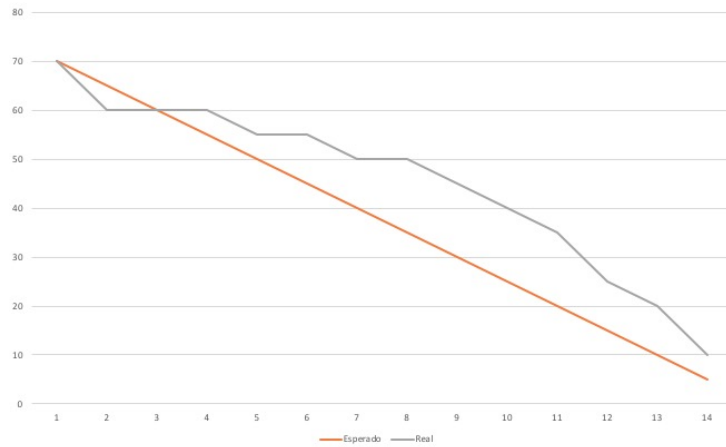


Figura 3. Gráfico de burndown da sprint 2

A figura 4 apresenta o gráfico de burndown para a Sprint 3.

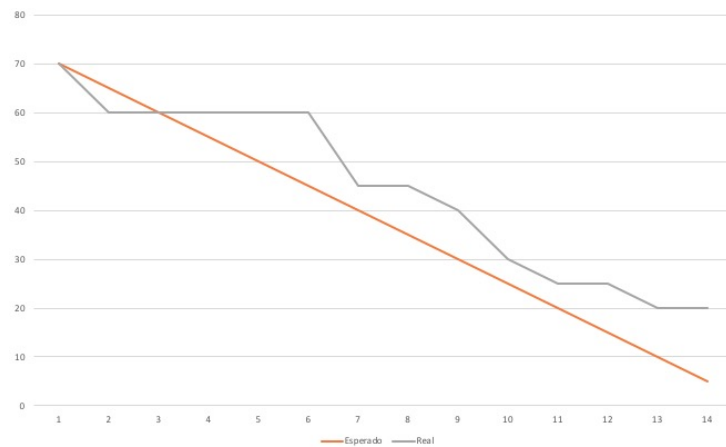


Figura 4. Gráfico de burndown da sprint 3

A figura 5 apresenta o gráfico de burndown para a Sprint 4.

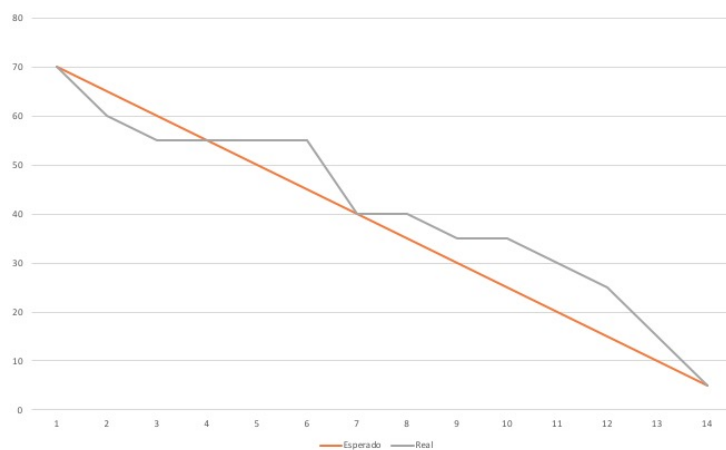


Figura 5. Gráfico de burndown da sprint 4

A figura 6 apresenta o gráfico de burndown para a Sprint 5.

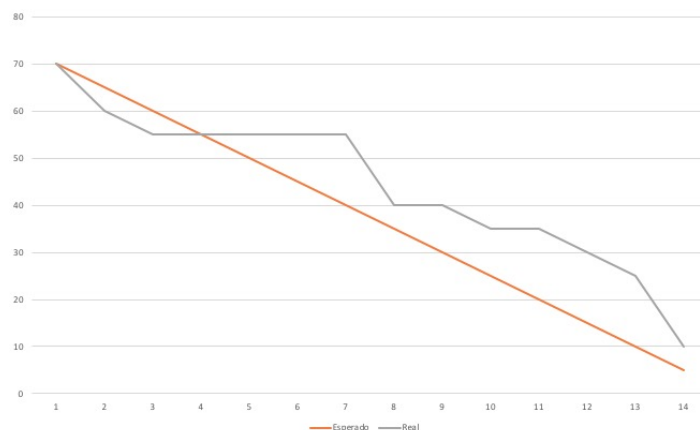


Figura 6. Gráfico de burndown da sprint 5

A figura 7 apresenta o gráfico de burndown para a Sprint 6.

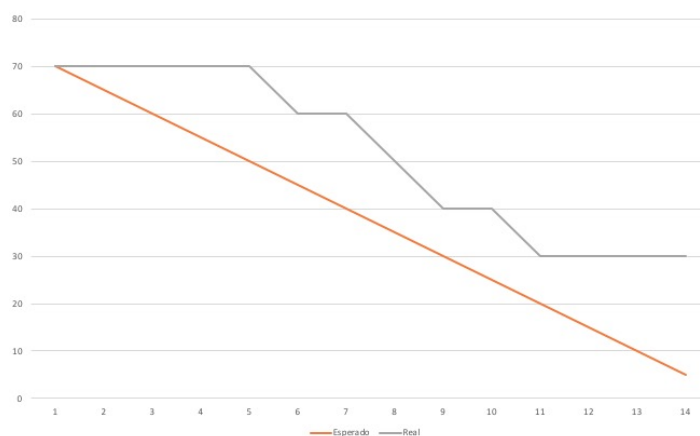


Figura 7. Gráfico de burndown da sprint 6

A figura 8 apresenta o gráfico de burndown para a Sprint 7.

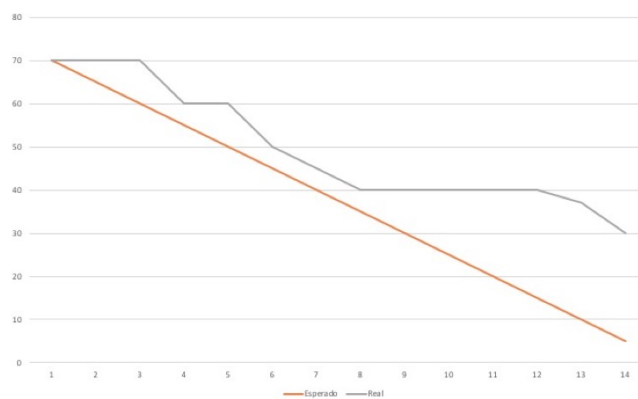


Figura 8. Gráfico de burndown da sprint 7

4. FERRAMENTA DE TESTES

4.1. FERRAMENTA ESCOLHIDA

Para a implementação de testes no projeto, utilizou-se o próprio framework de teste do Rails. O ambiente de testes desse framework é bastante eficiente e, com alguns comandos no terminal, ele realiza diversos testes automatizados no código existente.

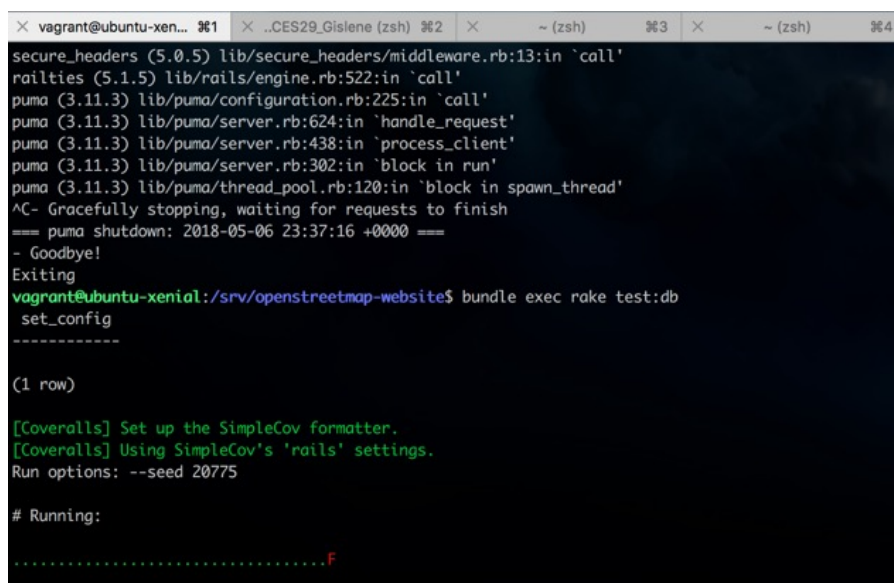
Os motivos principais de termos escolhido essa plataforma de testes é o fato dela já estar implementada no Ruby on Rails e funcionar tão bem com o código do OpenStreetMap. Esses fatores fizeram com que outra ferramenta não fosse tão vantajosa, uma vez que exigiria um esforço para sincronizá-la com o código implementado.

Para identificar quais partes do código não estão cobertas pela interface de testes, utilizou-se o Coveralls (coveralls.io).

Para realizar testes automáticos de uma interface web, utilizou-se o Capybara, que é um sistema de testes que simula o uso de um aplicativo web como se fosse um usuário real, evitando assim o desgaste de testar o sistema enquanto garante o funcionamento do software de forma adequada. Essa aplicação é voltada especificamente para web apps.

4.2. TESTES REALIZADOS

A Figura 9 apresenta o print de uma tela de terminal onde são testados alguns módulos do código, obtendo resultado satisfatório. Nesse caso, os testes foram realizados no início do projeto e ainda não havia grande variedade de módulos a serem testados.



```
✕ vagrant@ubuntu-xen... 361 ✕ ..CES29_Gislene (zsh) 362 ✕ ~ (zsh) 363 ✕ ~ (zsh) 364
secure_headers (5.0.5) lib/secure_headers/middleware.rb:13:in `call'
railties (5.1.5) lib/rails/engine.rb:522:in `call'
puma (3.11.3) lib/puma/configuration.rb:225:in `call'
puma (3.11.3) lib/puma/server.rb:624:in `handle_request'
puma (3.11.3) lib/puma/server.rb:438:in `process_client'
puma (3.11.3) lib/puma/server.rb:302:in `block in run'
puma (3.11.3) lib/puma/thread_pool.rb:120:in `block in spawn_thread'
^C- Gracefully stopping, waiting for requests to finish
== puma shutdown: 2018-05-06 23:37:16 +0000 ==
- Goodbye!
Exiting
vagrant@ubuntu-xenial:/srv/openstreetmap-website$ bundle exec rake test:db
set_config
-----
(1 row)

[Coveralls] Set up the SimpleCov formatter.
[Coveralls] Using SimpleCov's 'rails' settings.
Run options: --seed 20775

# Running:

.....F
```

Figura 9. Print de um teste em algumas unidades do código

A Figura 10 apresenta o print do terminal Linux para um teste um pouco mais extensivo do código implementado. Nela, é possível observar, pelos pontos verdes – que indicam teste aprovado –, que o código é completamente funcional nos módulos de teste implementados. Diferentemente da figura anterior, aqui o espaço amostral é bem mais vasto, assegurando maior credibilidade do teste.

```
gabriel@notega:~/Projetos/teste/CES29_Gislene/projeto_padrao_OSM$ rake test

[Coveralls] Set up the SimpleCov formatter.
[Coveralls] Using SimpleCov's 'rails' settings.
Run options: --seed 33407

# Running:

.....

Finished in 262.527708s, 3.9882 runs/s, 1297.8135 assertions/s.
1047 runs, 340712 assertions, 0 failures, 0 errors, 0 skips
[Coveralls] Outside the CI environment, not sending data.
```

Figura 10. Print de um teste mais extensivo em várias unidades do código

5. LIÇÕES APRENDIDAS

Ao final do projeto, aprendemos muito sobre o gerenciamento de um projeto de programação nos moldes da metodologia Ágil. Foi possível observar diversas das dificuldades enfrentadas em um projeto real: imprevistos na implementação e desenvolvimento do código; problemas de gerenciamento de tempo; imprevisibilidade nas estimativas de tempo e esforço por falta de experiência; e, principalmente, mudanças repentinas no escopo do projeto pelo Product Owner, forçando uma completa reavaliação das prioridades e tarefas a serem desenvolvidas.

Durante o semestre, pôde-se aplicar nesse projeto grande parte do conhecimento apresentado em sala de aula acerca do desenvolvimento de um projeto real, seja em questões envolvendo programação, quanto em questões mais sociais, como conversar com um cliente.

O único ponto nas conversas com o POs é que, no caso dessa disciplina, todos os Product Owners são pessoas técnicas, que entendem as dificuldades do desenvolvimento de uma atividade como essa, coisa que é extremamente raro de ocorrer em um ambiente corporativo real.