Decisões e especificações do projeto AD_EXTREME.

Guimarães, F.P, Medeiros H.H, Tenorio, M. A. R., Falcão, R.V.

Resumo:

Este documento tem como objetivo prover especificações,decisões de projeto e erros encontrados ao longo do projeto.

Abstract:

This document is intended to provide specifications, design decisions and errors found throughout the project.

Introdução:

O projeto como descrito: Sistema de anúncios ad-extreme: pessoas ou empresas possam publicar vários tipos de anúncios, tanto para venda de imóveis ou móveis bem como serviços e empregos. Os usuários pesquisam por anúncios por meio de tag/categoria ou pelo título do mesmo e ordena os mesmos por: data, preço, ou classificação do anunciante. Eles podem classificar o anunciante e contactar os mesmos por meio da plataforma ou por meio dos contatos disponibilizados pelos anunciantes. Os anunciantes podem determinar o tempo dos anúncios bem como determinar a data do início da publicação do anúncio (similar: olx, mercadolivre).[0]

A partir desta especificação são definidos objetivos, ferramentas que serão utilizadas para um melhor controle do projeto: Armazenamento, comunicação e organização. Timeline e tarefas realizadas, problemas encontradas no código (*bad smells)*, padrões de projeto (design)

Principais objetivos:

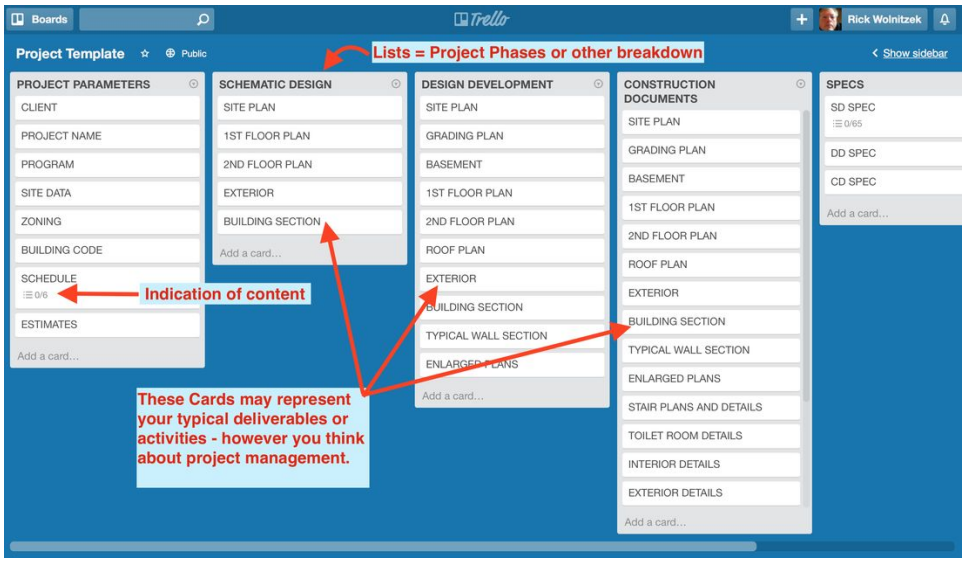
- ➔ Mostrar as principais decisões de projeto.
- ➔ Quais user stories foram escolhidas?
- ➔ Problemas encontrados e corrigidos do código original.

- Erros de codificação.
- Erros de design.

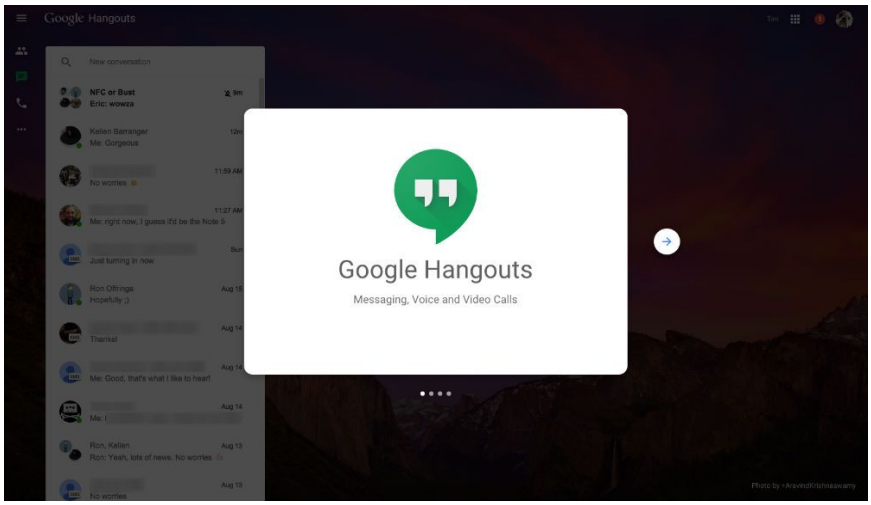
➔ Mostrar quais funções adicionais foram implementadas

Ferramentas utilizadas para acompanhamento do projeto e comunicação:

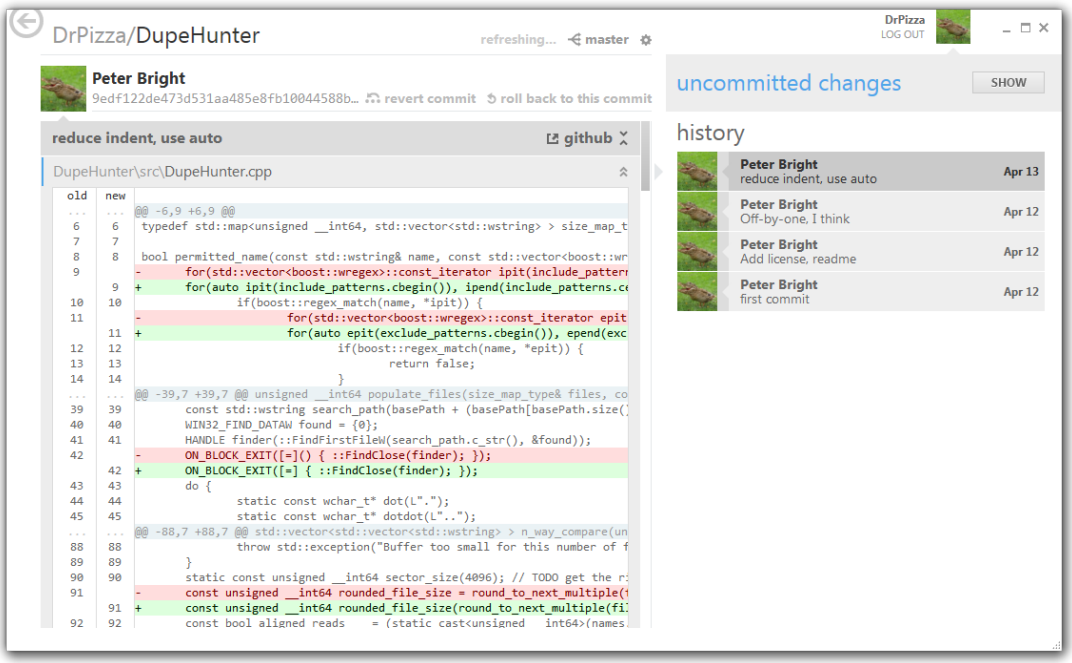
- Trello
  - Trello é uma plataforma online para gerenciamento de projetos, está sendo utilizada para separação de *tasks* do grupo:



- Hangouts
  - Devido aos diferentes horários dos integrantes do grupo, foram programadas reuniões presenciais (uma por semana) e o restante está sendo feita utilizando a ferramenta do google hangouts.



- GitHub: Sistema de controle de versão GIT hospedada no github, sendo utilizado pelo grupo para um desenvolvimento colaborativo.



Design do software:

Durante o passar da disciplina nos foram apresentados conceitos,princípios e padrões do projeto, alguns destes são relembrados abaixo:

- Legibilidade
- Coesão
- Acoplamento
- Duplicação
- Padrões de projeto: expert, controller, iterator etc.
- Código com testes e tratamento de erros.
- Refatoramento

Como o projeto tem um viés educacional, o código foi construído com erros intencionais para assim os alunos terem como alterar o código utilizando conceitos, princípios e padrões de projeto, a partir destes pontos um estudo foi feito no código original[1] e os pontos encontrados são os listados abaixo:

1. Classe An:
    a. Uma classe com nome não intuitivo *an* ferindo assim legibilidade e coesão, tal classe era utilizada para a comunicação entre o *backend* e o *frontend.*
2. Métodos com nomes genéricos
    a. Na classe usuário temos *getters and setters* com nomes não intuitivos e que não indicam sua funcionalidade *getR* e *setR.* Nota-se que aqui estão sendo feridos os princípios de legibilidade e coesão.
3. Uso do *thymeleaf*:
    a. Devido ao uso do *thymeleaf* ocorre um alto acoplamento que não é desejado (sempre desejamos, alta coesão e baixo acoplamento) isso reflete na dificuldade de implementação de algumas funcionalidades.
4. Alguns métodos são demasiadamente extensos:
    a. Devido ao tamanho do corpo de alguns métodos se perde informação e não se entende direito o que se desejava fazer ali
5. Documentação fraca ou inexistente:
    a. Não existe uma documentação para o projeto e não é feito o uso de ferramentas tal como javadoc.
6. Classes não possuem testes ou uma quantidade mínima:
    a. Só uma das classes possui testes
7. Classes ferindo padrões de projeto
    a. Classes realizando tarefas de outras ferindo padrões como o expert dentre outros.
8. Pouca ou inexistência de tratamento de erro

    a. Não existe tratamento de erro para maioria dos casos e quando existe ela é falha ou inexistente, por exemplo com a limitação dos caracteres que é dita de uma forma na especificação e implementada de outra maneira.
9. Não uso de um enum na classe notas
    a. A classe notas possui somente um atributo que tem uma coleção de constantes.
10. Uso de números mágicos para indicação de tipo de usuário
    a. Na parte do frontend o usuário é definido por um inteiro e é escolhido via um switch.
11. Classes que deveriam estar em pacotes distintos estão no mesmo pacote.
    a. Classes que tem atributos distintos e não deveriam estar no mesmo pacote.

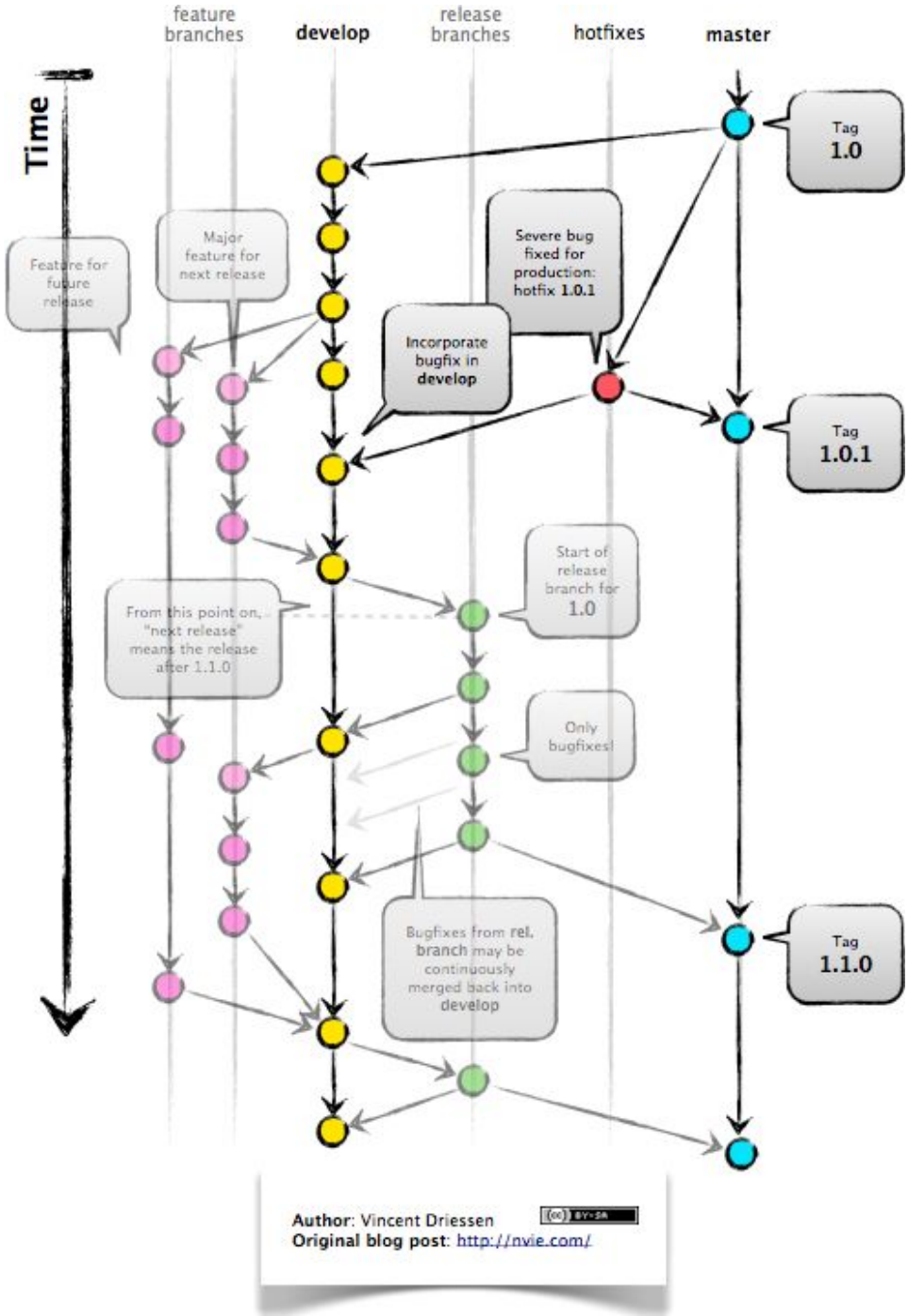
Após tais falhas de projeto serem encontradas soluções foram implementadas :

Lista de soluções:

1. Mudança de nome da classe an para um nome mais intuitivo para UserAnuncioController, aumentando a coesão;
2. Substituição de nomes de métodos para nomes mais intuitivos, assim novamente aumentando a legibilidade e coesão;
3. Troca do *thymeleaf* por javascript devido a familiaridade de integrantes do grupo com a tecnologia e também o uso de angular pelo motivo supracitado;
4. Com o uso de técnicas como extract method foi possível a diminuição de linhas de código para um código mais robusto,legível e enxuto;
5. O projeto está sendo documentado neste documento e estamos começando a realizar o javadoc para em anexo com a especificação, repositório ir na entrega final;
6. Construindo uma hierarquia de classes de testes para uma melhor cobertura da área do código;
7. Classes como a UsuarioServiceImpl tem obrigações e contratos que não condizem com a classe;
8. Criação de uma hierarquia e tratamento de erros para melhor legibilidade e facilidade de mudança;
9. Uso do padrão Type-safe ENUM.
10. Uso de outros tipos e fim do switch case aparenta fazer mais sentido para um desenvolvimento orientado a objeto.
11. Classes que não deveriam estar no mesmo pacote interagem entre si, sem necessidade e/ou sentido assim

Metodologia para controle de versão:

A partir do ‘GitFlow é um modelo de ramificação para Git, criado por Vincent Driessen. Ele tem atraído muita atenção porque é muito bem adequado para colaboração e dimensionamento da equipe de desenvolvimento.’[3]



Uso de padrões de projeto para resolução de problemas:

1. **Padrão Iterator**
    a. Sendo utilizado na pesquisa.
2. **Padrão Controller**
    a. Anúncio, e era o padrão “visível” da versão original.
3. **Padrão Builder**
    a. Separação da construção.

Fluxograma:

0. Primeira Entrega:

- Reuniões presenciais.
- Começo da escrita do documento.
- Busca por erros no código.
- Implementações.
    - Anuncios
    - Pesquisa
- Uso de javascript, familiaridade de um integrante com a tecnologia.
- Uso de angular pelo mesmo motivo.

1.Segunda Entrega:

- Reuniões presenciais
- Continuação da escrita do documento
- aplicação de mais projetos,princípios e conceitos de design de software

Entregáveis:

- 17/03/2017:
    - Versão 0.1 do documento
    - Repositório com o código fonte.[2]
- 28/03/2017:
    - Versão final do documento.
    - Repositório com o código fonte.
    - Projeto hospedado no heroku.
    - Javadoc pronto.

Trabalho em andamento:

.

- Estudo do deploy no heroku.
- Complemento de algumas user stories.
- Usar *https*
- Interessante usar containers para isolar a aplicação?
    - Flow do heroku

Referências.

[0] Projeto S.I 1 2016.2.
<https://docs.google.com/document/d/10Zt_vvNYfyVgI50dHmww4ryOkxJ5b0jYjULZNjY8vLk/pub > Acessado em 11 de Março de 2017
[1] https://files.slack.com/files-pri/T2TSJ35DX-F4CN3UA1E/download/ad-extreme-master.zip
[2] https://github.com/rafaelvfalc/Project-SI1-
[3] https://datasift.github.io/gitflow/IntroducingGitFlow.html
[4]