Projeto Prático 1
Programação Orientada a Objetos - MC302

Alunos: André Papoti, Bruno Falkenburg, Lucas Ramos, Nicolas França, Sophia Estrêla

Professora: Esther Colombini

Unicamp - Instituto de Computação

Abril de 2018

Resumo

A intenção do projeto é ser um sistema interno de uma imobiliária, ou seja, é feito considerando os corretores como usuários e a imobiliária em sí como um administrador (root). Esses usuários gerenciam imóveis, clientes, proprietários e propostas.

È importante destacar que o usuário final são os corretores e outros possíveis funcionários da imobiliária (que é o caso do nosso administrador).

Queremos fazer buscas de imoveis do melhor agrado do cliente, assim como servir como um banco de dados afim de armazenar os dados do cliente e do proprietário em sí.

Queremos enfatizar que o projeto não foi pensado para acesso direto de terceiros, como compradores e proprietários, pois a nossa regra de negócio diz que esses dados são sigilosos.

A nossa ambição é que em sua etapa final o próprio sistema dê sugestões deimóveis do agrado do cliente, cruzando suas preferências com os imóveis disponíveis.

O sistema foi pensado consultando uma corretora real, de forma que este seadapte as necessidades de um usuário do mundo real. Este também trará o benefício de tornar processos do dia-a-dia de uma imobiliaria, como a oficialização de uma proposta se tornar mais fácil e ágil.

1 Introdução

2 Funcionalidades

2.1 Funcionalidades Implementadas

Foram implementadas as seguintes funcionalidades nesta etapa do projeto:

1. Adição de novos Imoveis;
2. Adição de novos Corretores
3. Adição de novos Clientes;
4. Adição de Propostas;
5. Adição de Proprietários dos imóveis;
6. Remoção de Imoveis;
7. Remoção de Corretores
8. Remoção de Clientes;
9. Remoção de Propostas;
10. Finalização de Propostas: propostas pode estar ativas ou não;
11. Listagem dos Imoveis disponíveis no sistema como um todo;
12. Listagem dos Imoveis cujo corretor responsável é o usuário;
13. Listagem das Propostas da Imobiliaria;
14. Listagem das Propostas cujo o usuário (Corretor) foi intermediário;
15. Listagem dos Clientes cujo responsável é o Corretor;
16. Listagem dos Proprietarios de todos os Imoveis disponíveis no sistema;
17. Alteração dos dados do Imovel;
18. Alteração dos dados do Corretor;
19. Alteração dos dados do Proprietario;
20. Busca por um Imovel dado sua respectiva id ou objeto;
21. Busca por um Proposta dado sua respectiva id ou objeto;
22. Busca por um Corretor dado seu respectivo id ou objeto ou creci;

2.2 Implementações Futuras

Pensamos em implementar as seguintes funcionalidades na segunda parte do projeto:

1. Correlacione as preferências do cliente e as características dos imóveis para dar dicas para o corretor automaticamente com a adição de um novo cliente;
2. Possua a funcionalidade Visita, para agendamento de visitas a imóveis;
3. Possua uma classe Administrador (administrador da imobiliária com maior acesso que o Corretor);
4. Tenha a lista de imóveis ordenada conforme o número de visitas agendadas de forma que os mais visitados tenham destaque;
5. Tenha um banco de dados a fim de que os dados não sejam perdidos;
6. Possua algoritmos mais avançados em termos de busca;
7. Utilize uma API de mapas para busca de imóveis no raio de preferência do cliente;
8. Esteja disponível na nuvem (tenha um site);
9. Tenha um mecanismo de login seguro;

3 Ferramentas Utilizadas

3.1 Lagom

O Lagom Framework será utilizado para desenvolver serviços que serão consumidos pelo sistema. Entende-se Framework como uma coletânea de códigos genéricos - ou seja, que podem ser utilizados em diversos projetos - que têm por finalidade auxiliar o programador no desenvolvimento, servindo de esqueleto para o código que será escrito. Ao contrário de bibliotecas, frameworks ditam o fluxo de controle da aplicação.

O framework contém um conjunto de APIs que facilitam ao desenvolvedor o trabalho de escrever microserviços em Java, que podem fazer uso de ferramentas já incluídas no Lagom, como servidores para permanência dos dados e compartilhamento de informações com outros serviços. Todas as ferramentas utilizadas e os serviços desenvolvidos podem ser inicializados com um único comando, ou separadamente.

Microserviço é uma abordagem que visa construir um sistema como um conjunto de pequenos serviços, com funcionalidades específicas e completas, e que se comunicam por meios leves, havendo baixa dependência entre os

módulos. Este tipo de arquitetura colabora com a manutenção da modularidade, permitindo que alterações em serviços específicos não alterem o resultado do sistema como um todo. A estrutura dos serviços Lagom segue firmemente este modelo, fortalecido pela separação entre a declaração da interface do serviço e a implementação em si.

O Lagom oferece também a possibilidade dos microserviços nele construídos consumirem serviços externos, que não precisam seguir a estrutura dos microserviços Lagom. Isto será utilizado, por exemplo, quando for preciso extrair dados de rotas utilizando a API do Google Maps para descobrir uma boa rota entre os imóveis que um corretor deseja mostrar a um cliente.

Das APIs que o Lagom fornece, podem-se destacar a Service API, útil para as declarações das interfaces dos serviços desenvolvidos, bem como sua implementação, e a Persistence API, que auxilia no controle da persistência de dados. Apesar de não ser o banco de dados padrão, o Lagom tem suporte ao PostgreSQL, que será futuramente utilizado.

No momento, há escrito um conjunto de instruções básicas sobre as funcionalidades do Lagom, bem como instruções de instalação e configuração. O texto está disponível no arquivo "DocumentacaoLagom.pdf"localizado no diretório raiz do primeiro projeto ("Projeto1/").

3.2 APIs do Google Maps

A Google oferece alguns serviços para utilização do Maps por outros desenvolvedores. Dentre as APIs oferecidas, constam a Directions API, cuja finalidade principal é a de encontrar direções entre diferentes localidades; a Distance Matrix API, que calcula tempo e distância entre pontos em uma rota, e a Geocoding API, que transforma um endereço em coordenadas e vice-versa.

Para facilitar o uso dessas APIs no código Java, está sendo utilizado o Java Client for Google Maps Services, uma biblioteca desenvolvida pela equipe do Google Maps.

Os serviços que a Google oferece serão utilizados para calcular, sob demanda do corretor, uma boa rota entre diversos imóveis que ele deseja mostrar ao cliente.

Até o momento, foi utilizado o cliente Java do Google Maps para calcular uma rota entre dois endereços, e retornar as etapas do movimento em formato JSON, para que o resultado possa ser facilmente utilizado em outras aplicações, como numa interface gráfica. Utilizou-se a Directions API, que possibilita encontrar mais de uma rota para diversos meios de transporte. Entretanto, no exemplo optou-se por buscar apenas uma rota de carro, configuração que mais se assemelha com o futuro uso dos serviços no sistema. Os códigos fonte podem ser encontrados em "Projeto1/Lagom/maps-sem-lagom/maps-testes/src/".

3.3 PostgreSQL

PostgreSQL é um SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional. Isso significa que os dados no banco são modelados como entidades relacionadas, semelhantes a tabelas, acrescidos de estruturas típicas de orientação a objetos. A linguagem utilizada no PostgreSQL é a SQL.

Será utilizado para criação e gerenciamento de um banco de dados que armazenará os dados referentes às entidades do sistema.

3.4 React

Durante a conclusão do nosso projeto, além do Lagom/Maps/PostgreSQL, queremos montar uma página web baseada no Framework Javascript chamado React. O React foi criado pelo facebook principalmente para facilitar a criação de páginas que se atualizam a todo momento e que utilizam APIs externas a todo momento. Esse tipo de trabalho sem o React precisa de tratamento de DOM e Ajax, que não é uma tarefa muito trivial.

Dessa forma, o nosso Front-end vai utilizar um serviço externo, o Lagom vai gerar a API do Backend e o React vai consumir essa API e mostrar para o usuário final através do navegador. Dessa forma, o cliente do nosso projeto vai usar um programa Backend em Java sem necessariamente precisar ter o Java instalado, trazendo maior flexibilidade para o sistema.

3.5 Git e GitHub

O GitHub é uma plataforma que permite hospedar e compartilhar arquivos, com foco em arquivos de código-fonte.

Por utilizar o Git para controle de versão, os programadores podem trabalhar em ramificações locais do projeto e enviar ao repositório hospedado no GitHub os arquivos que trabalharam, registrando todas alterações e permitindo que outro desenvolvedor que esteja trabalhando no projeto possa permanecer atualizado sobre o progresso do outro.

Por conta dos benefícios que esta plataforma traz, o grupo está utilizando para controle do código - garantindo que todos estejam com versões atualizadas e possam com a mesma facilidade revisar e alterar o próprio código ou o de outro membro da equipe - e dos demais arquivos relacionados ao projeto.

3.6 Trello

O Trello é o nosso organizador de projetos. Com ele temos um board, que tem um conjunto de listas. E cada lista tem é um conjunto de card. Como usamos o método Kanban, os cards transitam entre lists até chegar a lista "Conclusão".

A estrutura do nosso board pode ser vista na Figura 1.

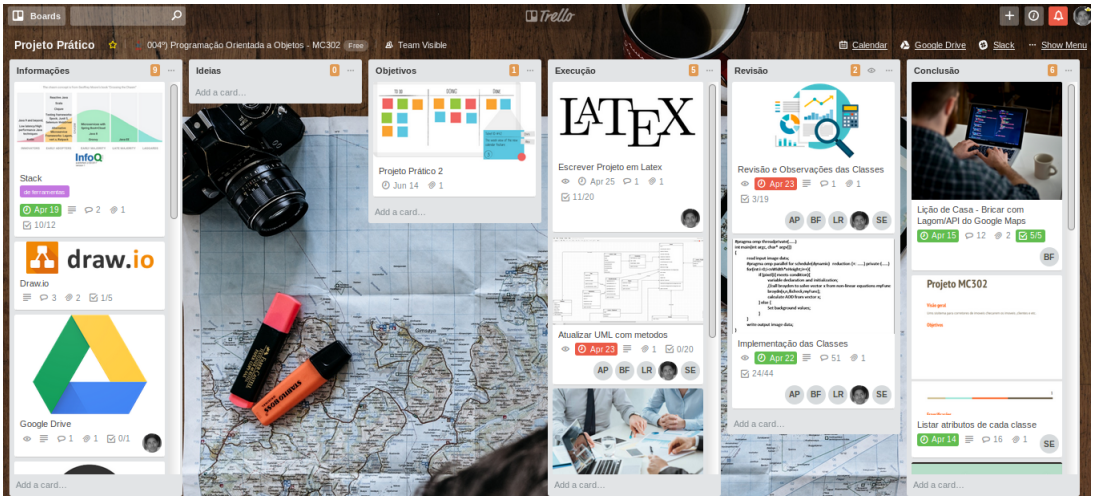

Figura 1: Como o Trello é usado no nosso projeto

O nosso board é dividido em 6 listas: Informações, Ideias, Objetivos, Execução, Revisão e Conclusão.

Informações: Tem cards com informações gerais, links, recursos e descrições e documentações das ferramentas que estamos utilizando.

Ideias: Lista para cards de ideias gerais.

Objetivos: Lista com os objetivos futuros que estamos galgando.

Execução: Lista com as tarefas que estão sendo executadas naquela fase do projeto.

Revisão: Revisão das tarefas terminadas em Execução.

Conclusão: Tarefas que foram em grande parte feitas.

3.7 Google Hangouts

O Google Hangouts é o serviço padrão da Google para comunicação em tempo real. Usamos muito durante nossas reuniões remotas. Normalmente nos encontramos mais online do que fisicamente devido aos diferentes horários em comum entre os membros.

Conseguimos com ele compartilhamento de tela compartilhamento de áudio e essas features são as mais usadas entre nós.

3.8 Google Drive

Usamos para ter os dados mais atualizados possível de forma sincronizada. O Trello e o draw.io tem integração com o Drive que usamos constantemente entre nós.

3.9 draw.io

Para a criação do Diagrama de Classes UML presente neste arquivo foi utilizado o draw.io, uma ferramenta online de criação e edição de diversos tipo de diagramas. O site permite integração dos diagramas com o Google Drive, que é útil para compartilhamento dos diagramas e edição simultânea por mais de um membro.

3.10 Whatsapp

Usado para comunicação rápida entre os membros.

3.11 LaTeX

O LaTeXé utilizado para a formatação do nosso relatório final sobre o projeto. O arquivo lido atualmente foi compilado pelo LaTeX e nos proporciona vários benefícios relacionados a escrita. É uma ferramenta com grande uso pela comunidade acadêmica.

3.12 WindowsBuilder

O WindowsBuilder é um plugin para Eclipse para facilitar a criação de interfaces gráficas (GUI). Nessa fase do projeto essa ferramenta vai nos ajudar a entregar um projeto com uma interface gráfica simples. Vamos ter a estrutura de um CRUD (mas sem banco de dados).

Referências

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2009.

[2] Ben Pfaff. Performance analysis of bsts in system software. Available at https://benpfaff.org/papers/libavl.pdf.

[3] Ben Pfaff. Performance analysis of bsts in system software. *SIGMETRICS Perform. Eval. Rev.*, 32(1):410–411, June 2004.