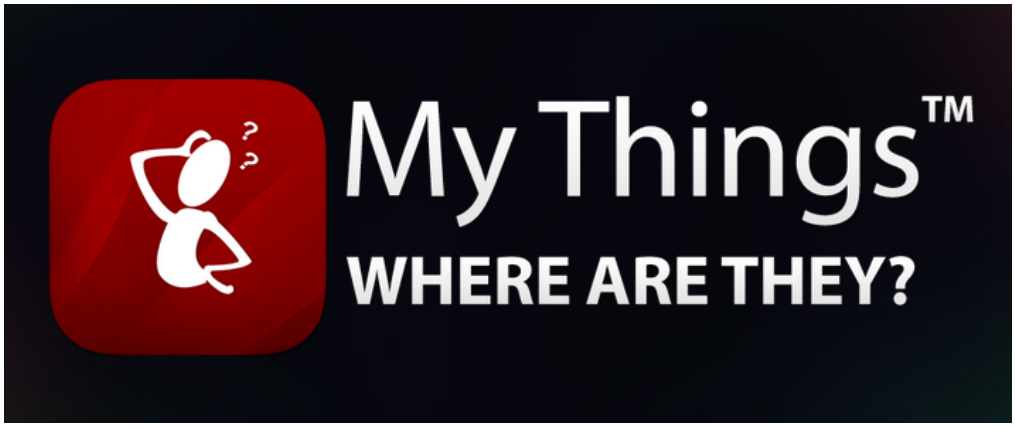
draft - projeto LP2 2017.1



TT - Tracking things

Você e alguns amigos tem um monte de jogos, eletrônicos e de tabuleiro, além de vários blu rays com séries e filmes. Vocês sempre emprestam uns aos outros, mas de vez em quando é difícil lembrar com quem está o quê. Para resolver esse problema vocês resolveram escrever um app chamado TT (de tracking things). Vocês dividiram as tarefas, um amigo seu muito bom em desenvolvimento para android vai fazer toda a parte de interface gráfica e você e seu grupo são responsáveis pela lógica dessa aplicação. Assim, todos os amigos podem acessar o aplicativo via celular[1], facilitando a gerências de seus empréstimos.

A ideia é muito simples. O sistema permite o cadastro de amigos (usuários) com seus pertences (itens que eles aceitam emprestar). O TT deve conseguir manter informações de todos os empréstimos de itens entre os usuários cadastrados. A qualquer momento você deve poder identificar com quem está o seu maravilhoso jogo 1 do Lego Harry Potter pra PS4... Ou onde está seu jogo da vida, uma relíquia, e se alguma peça foi perdida e quem perdeu. Também é possível saber se alguém tem The Sims para PC porque você está interessada(o) em pegar emprestado por uns dias.

Para manter o sistema justo, o TT vai aumentando a reputação de quem cuida bem "das coisas" dos amigos e devolve no prazo, enquanto vai diminuindo a reputação daqueles que não são bons cuidadores ou devolvedores. Assim, um amigo com reputação baixa pode de repente não conseguir mais nada emprestado porque já farrapou demais. Isso é bom para proteger os bens dos usuários.

Abaixo segue o documento que especifica o desenvolvimento desse software.

Requisitos importantes a serem considerados:

É importante ter em mente que ao desenvolver um programa, uma série de decisões importantes de design são tomadas. Uma ótima prática de programação é sempre separar a interface gráfica das classes do domínio do problema. Isso é importante para garantir que mudanças na interface não irão necessitar mudanças nas classes do

domínio do problema e mudanças nas classes do domínio do problema não irão requerer mudanças na interface gráfica. Dessa forma, essas partes conceituais do programa ficam isoladas. Nesse projeto iremos alcançar esse isolamento usando uma classe de fachada (ou facade) e classes controladoras (ou controllers). Veja mais detalhes a seguir:

- Todas as suas classes que fazem parte do domínio do problema não devem ser acessadas diretamente pelo código da interface gráfica. Portanto, você deve usar o padrão de projeto facade. Para saber mais sobre esse padrão de projeto você pode estudar um pouco aqui ou aqui (ou em outro material que ache legal). Essa classe de fachada vai fazer a comunicação entre a interface gráfica que seu colega irá desenvolver e as classes do domínio do problema que seu grupo irá desenvolver.
- A sua classe de fachada também não deve entender muito das classes do domínio do problema. O ideal é que cada método dessa classe tenha apenas uma linha de código, que será uma chamada por delegação a um método de algum controller. Assim, você terá que usar classes controladoras (controllers) para esconder da fachada tratamentos de entradas que a fachada não deveria ter responsabilidade de fazer, mas que você também não quer delegar para uma classe do domínio do problema. Por exemplo, quando um usuário vai adicionar um item de sua coleção de itens a emprestar, ele irá informar o nome do item, o valor, e quaisquer outras informações para que esse item seja criado. Em vez de chamar diretamente o objeto usuário onde o item deve ser inserido, você deve chamar um controlador que vai checar se essa entrada é válida e só então chama o objeto necessário para a operação. O uso de controladores é mais um nível de abstração para separar a interface gráfica das classes do domínio do problema. Esse controlador faz parte de um padrão de projeto muito usado chamado model-view-controller. Você pode entender um pouco mais desse padrão aqui ou aqui.

Você verá que a classe de fachada e os controladores serão escritos aos poucos ao longo do projeto à medida que o grupo avança no desenvolvimento dos casos de uso. Assim, não teremos casos de uso específicos para o desenvolvimento destas entidades, mas lembre-se que a cada caso de uso vocês devem considerar a classe de fachada e os controladores necessários para lidar com o caso de uso sendo desenvolvido.

Caso de uso 1: CRUD dos usuários

Deve ser possível fazer CRUD (Create, read, update and delete) de usuários no TT.
Por enquanto, cada usuário deve manter as seguintes informações obrigatoriamente: nome, email e número do celular do usuário.

No futuro (caso de uso 2) cada usuário deve ter sua própria lista de itens, que podem ser de vários tipos. Então os usuários deverão poder adicionar e remover itens que eles estão disponibilizando para empréstimo no sistema. Por enquanto, basta permitir que usuários sejam adicionados, removidos, pesquisados e atualizados (tenham seus dados atualizados) no TT. Dois usuários são iguais se eles tem o mesmo nome e número de celular. Deve ser possível ter uma representação de String de um usuário da seguinte forma:

Raquel Vigolvino Lopes, raquel@computacao.ufcg.edu.br, (83) 9999-0000

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Para avaliar melhor o desenvolvimento deste caso de uso rode esses testes de aceitação com EasyAccept.

Caso de uso 2: CRUD dos itens emprestáveis

Deve ser possível fazer CRUD de itens de empréstimo do TT. Todo item precisa manter o seu nome, o seu valor (que pode ser o valor de compra) e se está ou não emprestado no momento. No futuro vocês também pretendem incluir livros e material esportivo, então é bom manter um design que permita que novos tipos de itens possam ser facilmente emprestados via esse sistema.

Inicialmente, vocês pretendem considerar os seguintes itens de empréstimos:

- Blu ray. Blu ray pode ser de filmes, séries ou shows. Pode ser um apenas ou uma temporada inteira de uma série. Vocês decidiram que não será possível emprestar um único blu ray de uma temporada, ou empresta a

temporada inteira ou não empresta nada. No blu-ray de uma temporada você deve guardar a coleção de blu-rays que compõem a temporada. Além das informações gerais que todo item tem (descritas anteriormente), todo blu ray deve ter uma duração e uma classificação[2]. Os blu rays de filmes devem ainda ter gênero[3] e ano de lançamento. Os blu rays de shows devem indicar o nome do artista e o número de faixas. A coleção de blu rays que representa uma temporada de uma série deve manter, além da coleção de blu rays, o gênero da série e o número da temporada. Além disso, blu-rays de temporada devem saber informar sua duração total. Dois blu rays são iguais se tiverem o mesmo nome. Dois blu-rays de temporada são iguais se tiverem o mesmo nome e mesmo número de temporada.

- Jogos eletrônicos. Os jogos eletrônicos são os jogos de video games (como PS4, xbox one, etc.) e também os jogos de PC que tem mídias (por exemplo CD) que podem ser emprestadas. Nosso sistema não vai compartilhar senhas de STEAM, Origin ou outros congregadores de jogos online. Além das informações gerais que todo item tem (descritas anteriormente), todo jogo eletrônico deve ter um nome e indicar a plataforma[4] do jogo. Dois jogos eletrônicos são iguais se tiverem o mesmo nome e funcionarem na mesma plataforma.
- Jogos de tabuleiro. Os jogos de tabuleiro são os mais delicados, já que pequenas peças podem ser perdidas. Além das informações gerais que todo item tem (descritas anteriormente), todo jogo de tabuleiro deve manter informação sobre peças perdidas de um jogo de tabuleiro. Essas informações podem vir em uma lista de strings que representam as peças perdidas. Com base nessa lista de peças perdidas o jogo deve saber informar se está ou não completo. Dois jogos de tabuleiro são iguais se tiverem o mesmo nome e mesmas peças perdidas.

Todos os itens devem poder ser apresentados através de uma string. Você também deve acrescentar os métodos get, set e comportamentais que achar necessários.

Tem um amigo mais experiente de vocês que resolveu ajudar e está escrevendo testes de aceitação com EasyAccept. Esses testes podem ajudar um pouco mais a definir essas informações.

Depois de criados os tipos de itens emprestáveis, certifique-se de que os usuários cadastrados no sistema tenham sua própria lista de itens (seus pertences que desejam emprestar). De posse do nome do item e do nome do dono do item, deve ser possível identificar um item único no sistema. Assim, não deve ser permitido que um usuário tenha itens com o mesmo nome. A lista de itens disponíveis para empréstimo de cada usuário pode ser de quaisquer tipos criados neste caso de uso. Então, **adicionar/remover/pesquisar/atualizar** um item no TT significa adicionar/remover/pesquisar/atualizar um item na coleção de itens de um usuário específico. A pesquisa de itens oferecida pelo sistema deve usar o nome do item e portanto pode retornar vários itens, se estes estiverem repetidos dentre as listas de empréstimo dos usuários. Nesse caso de uso você deve garantir que itens (de quaisquer tipos) podem ser adicionados, removidos, atualizados e pesquisados na rede TT.

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Testes de aceitação disponíveis aqui.

Caso de uso 3: funcionalidades de pesquisa

Uma vez que o sistema já conhece os usuários e seus pertences, chegou a hora de escrever algumas funcionalidades interessantes.

1. O sistema deve permitir uma listagem de todos os itens previamente cadastrados de todos os usuários no sistema no momento da pesquisa (estejam eles já emprestados ou não). Essa listagem deve ser ordenada em ordem alfabética pelo nome do item;
2. O sistema deve permitir uma listagem de todos os itens inseridos no sistema no momento da pesquisa (estejam eles já emprestados ou não). Essa listagem deve ser ordenada em ordem de valor dos itens;
3. O sistema deve permitir que um usuário pesquise mais detalhes sobre um item usando o nome do item, o nome do dono do item e o celular do dono do item. Por mais detalhes entenda: informação sobre o estado atual do item (se está emprestado ou não) e as caraterísticas específicas do item, que vão depender do item específico sendo pesquisado.

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Testes de aceitação disponíveis aqui.

Caso de uso 4: criação de empréstimos

O sistema deve permitir empréstimos entre usuários do sistema. Para tal uma nova entidade Emprestimo deve existir, que vai guardar informações sobre o empréstimo específico:

- O dono do item emprestado;
- O usuário que pegou o item emprestado;
- O item emprestado;
- Data inicial do empréstimo;
- Número de dias combinados para empréstimo;
- Data real de devolução do item.

De posse do nome do item, do nome do dono do item e seu telefone, deve ser possível identificar um item único no sistema. Dois empréstimos são iguais se envolver os mesmos usuários, o mesmo nome do item e mesma data do empréstimo.

Cada usuário deve manter uma lista dos empréstimos de que participou, seja ao emprestar, seja ao tomar emprestado. Obviamente, itens que já estão emprestados não podem ser novamente emprestados até que sejam devolvidos.

A data marcada para devolução do item é por padrão 7 dias após o empréstimo. Então, por enquanto, todos os itens podem ser emprestados por no máximo 7 dias. Na hora da devolução de um item, o sistema deve receber o número de dias em atraso e a data da devolução (para manter em histórico).

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Testes de aceitação disponíveis aqui.

Caso de uso 5: funcionalidades relacionadas aos empréstimos

O sistema deve ser capaz de analisar os empréstimos da seguinte forma:

- Listar os empréstimos de um usuário quando este usuário está pegando emprestado. Ordenar em ordem alfabética crescente do nome do item;
- Listar os empréstimos de um usuário quando este usuário está emprestando um item. Ordenar em ordem alfabética crescente do nome do item;
- Listar o histórico de empréstimos de um item (identificado no sistema pelo seu nome) na ordem em que o item foi sendo emprestado. Se existirem itens de usuários diferentes com o mesmo nome, então o histórico de todos esses itens deve ser apresentado;
- O sistema deve permitir uma listagem de todos os itens não emprestados no momento da pesquisa. Ordenar essa lista de itens pelo nome do item em ordem crescente;
- Identificar os top-10 itens. Estes são os 10 itens que mais são emprestados . Se existirem itens duplicados no sistema (com mesmo nome) eles devem ser tratados de forma separada. Por exemplo, se dois usuários diferentes tiverem o mesmo jogo de tabuleiro (cada um tem um jogo com mesmo nome), e o de um usuário foi emprestado 23 vezes e o do outro foi emprestado 31 vezes. Se esses jogos fizerem parte dos itens mais emprestados, então ambos devem aparecer na lista top-10 em posições diferentes.;
- Listar os itens que estão emprestados no momento, incluindo o nome do item e o seu dono.

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Testes de aceitação disponíveis aqui.

Caso de uso 6: manter reputações de usuários

Cada usuário do sistema deve manter uma reputação no sistema. A reputação de um usuário deve ser representada por um número de ponto flutuante. A reputação de um usuário inicia com 5% do valor total dos os itens disponíveis pelo usuário no início. Assim, cada vez que um usuário adiciona um item para empréstimo, a sua reputação sobe um

valor de 5% do valor do item adicionado. Os usuários que não dispõem de itens para empréstimo tem reputação inicial zero.

Cada vez que um usuário empresta um item, sua reputação deve subir um valor de 10% do valor do item emprestado. Assim, se um usuário emprestar um jogo eletrônico e esse jogo custa R$ 99,00, então a reputação do usuário deve aumentar em 9,9 pontos. Essa funcionalidade de modificar a reputação do usuário que empresta deve ser implementada.

Quando um usuário que pegou um item emprestado vai entregar o item, o sistema deve ser capaz de atualizar a reputação do usuário que está devolvendo o item como explicado a seguir. Cada vez que um usuário tomar algo emprestado e devolver obedecendo o prazo, a sua reputação deve aumentar um valor de 5% do valor do item emprestado. Assim, se um usuário tomar um jogo eletrônico emprestado e esse jogo custa R$ 99,00, então a reputação do usuário que devolveu o jogo em dia deve aumentar em 4,95 pontos. Quando o usuário entrega um item com atraso, sua reputação deve ser diminuída de um valor proporcional ao valor do item emprestado da seguinte forma: para cada dia de atraso adicionamos 1% nessa proporção. Com 3 dias de atraso a proporção é 3%. Considerando o exemplo do item que custa R$ 99,00, se o usuário entrega o item com 1 dia de atraso sua reputação é diminuída em 0,99, já se entrega com 3 dias de atraso a sua reputação deve ser diminuída de 2,97.

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Testes de aceitação disponíveis aqui.

Caso de uso 7: criar cartões fidelidade de usuários

Os usuários iniciam como usuários noob, com reputação conforme explicada no caso de uso 6. Depois que começam a usar o TT o usuário vai ter sua reputação modificada. Dependendo da reputação do usuário o usuário pode ser considerado como noob, caloteiro, freeRider, ou bomAmigo segundo a tabela abaixo:

noob	O usuário que tem itens para emprestar e reputação no intervalo [0,100]
caloteiro	O usuário que tem reputação menor que zero
bomAmigo	O usuário que tem reputação maior que 100
freeRider	O usuário que tem reputação maior ou igual a zero, mas não tem itens para emprestar

O TT se baseia no tipo de usuário para decidir sobre detalhes de empréstimos. Veja na tabela abaixo.

	noob	**caloteiro**	**bomAmigo**	**freeRyder**
Usuário desse tipo pode pegar algo emprestado?	sim	não	sim	sim
Qual o período máximo do empréstimo para usuário desse tipo?	7 dias	0	14 dias	5 dias

Com base nessas novas informações é possível que você tenha que rever algum código criado anteriormente relacionado aos empréstimos e cálculo das reputações. Lembre-se que esses cartões fidelidade modificam a forma como a reputação e o empréstimo são feitos e o tipo de cartão de um usuário vai mudar ao longo da execução do programa.

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Testes de aceitação disponíveis aqui.

Caso de uso 8: funcionalidades baseadas nas reputações de usuários

- Os usuários que estiverem com reputação negativa não podem mais tomar itens emprestados até que sua reputação seja maior ou igual a zero novamente;
- O TT deve saber listar todos os usuários que estão com reputação negativa e que portanto não podem pegar itens emprestados no momento. Esta listagem deve vir ordenada pelo nome dos usuários;
- O TT deve saber listar os top-10 melhores usuários, aqueles com as maiores reputações (ordenado da maior reputação para a menor);
- O TT deve saber listar os top-10 piores usuários, aqueles com as menores reputações (ordenado da pior reputação para a maior);

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Testes de aceitação disponíveis aqui.

Caso de uso 9: persistência

O TT deve armazenar em disco todos os dados necessários para manter o seu estado mesmo que o programa seja fechado. Assim, se eu usar o programa, adicionar usuários e seus itens, realizar empréstimos, tudo isso deve ser visto se eu fechar o programa e abrir novamente.

Implemente a parte da classe de fachada e o(s) controlador(es) necessários para lidar com as classes/funcionalidades criadas neste caso de uso. Testes de aceitação disponíveis aqui.

---

[1] No mundo real essa lógica do TT deveria se preocupar com sincronização entre processos, para evitar condições de corrida (coisas que vocês só vão aprender mais na frente do curso). Então esta versão a ser desenvolvida será uma versão simplificada do TT.
[2] Existem vários sistemas de classificação, nesse projeto os seguintes valores de classificação são usados: LIVRE, DEZ_ANOS, DOZE_ANOS, QUATORZE_ANOS, DEZESSEIS_ANOS, DEZOITO_ANOS.
[3] Os gêneros considerados nesse projeto são os seguintes: ACAO, ANIMACAO, AVENTURA, COMEDIA, DOCUMENTARIO, DRAMA, EROTICO, FAROESTE, FICCAO, MUSICAL, POLICIAL, ROMANCE, SUSPENSE, TERROR, OUTRO.
[4] As plataformas consideradas nesse software são as seguintes: PC, MAC, PS3, PS4, XBOX360, XBOX_ONE, NINTENDO_3DS, OUTRO.