DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

LEIC-A, LEIC-T, LETI

Engenharia de Software

2º Semestre – 2017/2018

Enunciado Geral do Projecto

O objetivo do projeto é desenvolver o sistema ADVENTURE BUILDER . Este sistema deverá permitir a integração de diferentes aplicações com vista à oferta de atividades de lazer.

O que se segue é uma descrição geral do projeto a desenvolver. Aspetos concretos do projeto a realizar serão descritos em enunciados ao longo do semestre. As partes descritas neste enunciado e que não sejam pedidas nas várias fases a realizar não necessitam de ser concretizadas e apenas servem para contextualizar o problema. Informação sobre a forma específica de desenvolver aspetos do projeto é fornecida no **guia de laboratório** deste ano.

1. Introdução

O sistema ADVENTURE BUILDER tem como funcionalidade principal a integração de diversos fornecedores de serviços com vista à oferta de atividades de lazer. Adicionalmente, o sistema permite a reserva de aventuras, nome que iremos usar para referenciar o conjunto de serviços associados a uma atividade de lazer. A concretização deste sistema é alcançada através da integração de 6 aplicações: BROKER, HOTEL, ACTIVITY, CAR, BANK e TAX.

Cada uma destas aplicações é um componente independente e possui uma interface web através da qual os utilizadores podem interagir. Adicionalmente, a aplicação BROKER interage com as restantes aplicações por forma a fornecer os seus serviços por integração dos serviços que elas disponibilizam. Por outro lado, todas as aplicações interagem com a aplicação TAX por forma a enviarem as faturas de todo os item que vendidos.

Uma interação típica com início na aplicação BROKER é iniciada por um utilizador que através da interface web requer a reserva de uma aventura. Este pedido desencadeia uma interação entre a aplicação BROKER e a aplicação BANK, por forma a se proceder ao débito do custo associado à viagem. De seguida, a aplicação BROKER interage com as outras três aplicações, HOTEL, ACTIVITY e CAR, para se proceder à reserva do hotel, da atividade de lazer

e do carro. No caso de todos os pedidos serem bem sucedidos, a base de dados do BROKER é atualizada com a nova informação, na base de dados do BANK uma conta é debitada, na base de dados do HOTEL um quarto do hotel é reservado, na base de dados da ACTIVITY uma actividade de lazer é também reservada, na base de dados de CAR é registado o alugar do carro e na base de dados de TAX é registada a fatura associada à operações comercial associada à fatura.

Uma outra interação que envolve mais do que uma aplicação ocorre sempre que um dos fornecedores transaciona um item, dado que tem que enviar a fatura para a aplicação TAX.

Por forma a se desenvolver um sistema com este nível de complexidade vamos usar uma abordagem incremental, em que o sistema irá ser desenvolvido através de uma sequência de versões, em que cada uma delas aumenta a quantidade total de funcionalidade existente na versão anterior, e iterativa, em que a complexidade tecnológica do sistema irá também ser introduzida por fases. Focando-nos nos aspetos iterativos do desenvolvimento do sistema devemos introduzir dois conceitos que nos ajudam na sua descrição e compreensão:

- *Módulo* - um módulo é uma unidade de implementação, como por exemplo uma classe ou um pacote
- *Componente* - um componente é uma unidade de execução, como por exemplo um processo

Estabelecidos estes dois conceitos, podemos definir a estratégia de desenvolvimento iterativo do sistema ADVENTURE BUILDER como a progressiva separação dos módulos que formam o sistema por diferentes componentes. Considerando que existe um módulo associado a cada uma das 6 aplicações, o código que as implementa, nas versões iniciais do sistemas estes módulos vão executar dentro de um único componente e nas versões finais cada um dos módulos executará no seu componente.

Assim podemos considerar as diferentes fases do desenvolvimento iterativo do sistema:

1. Os 6 módulos executam dentro do mesmo componente, concretizado através de uma máquina virtual java. Nesta fase a funcionalidade dos módulos é testada usando testes de unidade e integração com a tecnologia JUNIT
2. Os 6 módulos executam dentro do mesmo componente, mas vão ser adicionados testes que simulam as interações que ocorrerão quando estiverem separados em diversos componentes, utilizando a tecnologia JMOCKIT
3. Os 6 módulos executam dentro do mesmo componente, mas vai ser adicionado um novo componente, um servidor de base de dados que suportará dados persistente, utilizando a tecnologia FENIXFRAMEWORK. Nesta configuração o componente onde executam os 6 módulos é um cliente do componente servidor da base de dados
4. Cada um dos 6 módulos executa dentro de 1 componente distinto. Este componente irá ser um servidor aplicacional que interage com 6 servidores de base de dados diferentes, um componente de base de dados para cada um dos componente servidor aplicacional. Os servidores aplicacionais possuem interfaces web html, através das quais os utilizadores finais interagem, e são implementados usando a tecnologia SPRINGBOOT
5. A configuração anterior é aumentada com mais um componente que irá fazer testes carga, utilizando a tecnologia JMETER, por forma a simular perfis de utilização do sistema próximos do que virá a ser a sua utilização em produção

De acordo com o plano traçado o desenvolvimento decorrerá em 5 fases distintas de 2 semanas e em que cada fase corresponde uma etapa do desenvolvimento iterativo. O desenvolvimento incremental acontecerá ortogonalmente e sempre que seja necessário enriquecer a aplicação com mais funcionalidade. De forma a garantir que todos os grupos acompanham o desenvolvimento, após a entrega associada a uma fase será disponibilizado o enunciado e o código de partida para a fase seguinte. Este código conterá a solução da fase anterior eventualmente enriquecido com mais funcionalidade.

Dado que em cada fase é entregue uma nova versão do código, como ponto de partida para a fase seguinte, todas as alterações que se efetuem para além do solicitado no enunciado não estarão disponíveis na fase seguinte. Assim sendo, não é aconselhável a realização desse tipo de alterações, pois, para além de não terem impacto na avaliação, podem comprometer a conclusão das tarefas nos prazos definidos.

Resumidamente, o foco de cada fase iterativa é:

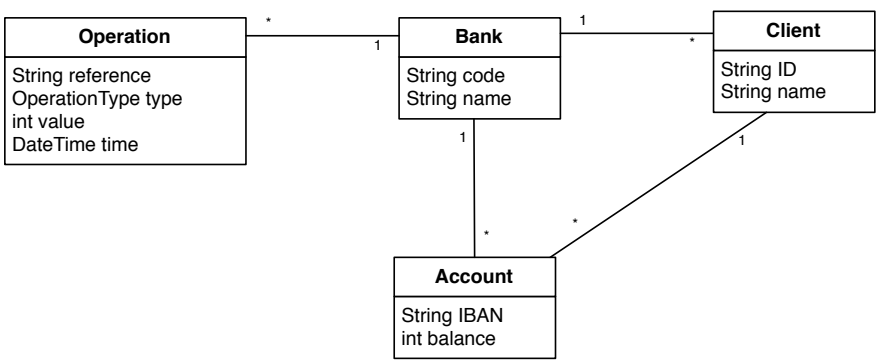
1. Desenvolvimento baseado em testes, testes unitários e de integração com a tecnologia JUNIT
2. Testes de integração com a tecnologia JMOCKIT
3. Introdução da persistência por refatorização do código e utilização da tecnologia MYSQL e FENIXFRAMEWORK
4. Servidores aplicacionais com a tecnologia SPRINGBOOT
5. Concorrência e testes de carga com a tecnologia JMETER

2. Módulos do Sistema

O sistema é constituído por 6 módulos que implementam a funcionalidade das aplicações: BROKER, HOTEL, ACTIVITY, CAR, BANK e TAX.

Sempre que for referida a existência de identificadores associados a entidades do domínio, quer dizer que eles são únicos, i.e., permitem identificar univocamente cada instância dessa entidade. Existem dois tipos de identificador, aqueles que são explicitamente dados como argumentos do construtor da instância, e os que são sequencialmente gerados durante a criação da instância. Sempre que for necessário visualizar uma lista instâncias de entidade do domínio, por omissão, a ordem é alfabética crescente dos identificadores das entidades presentes na lista.

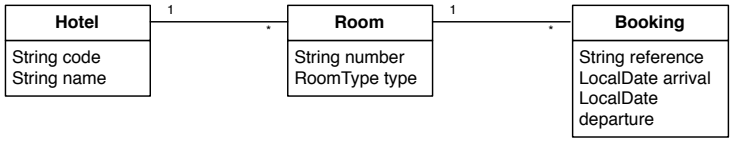
2.1. BANK



O módulo BANK é responsável por suportar a funcionalidade associada aos pagamentos das aventuras. Permite a criação de instâncias da entidade Bank, e cada banco possui associadas instâncias de Client e Account. O identificador de Bank é explícito enquanto que os identificadores de Client e Account são sequencialmente gerados.

Adicionalmente, sempre que ocorre uma operação de depósito, ou de levantamento, é criada uma instância da entidade Operation como forma de registar a sua ocorrência. Os identificadores das instâncias de Operation, que são gerados sequencialmente, servem como referência da ocorrência da operação. Por exemplo, quando é efetuado o pagamento de uma aventura é enviada a referência da operação de pagamento para as entidades do módulo Broker. As instâncias de Bank possuem a lista de todas as operações que ocorreram no seu contexto.

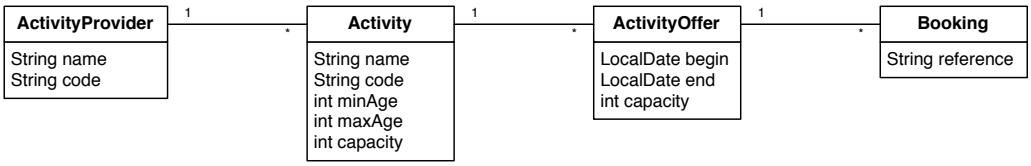
2.2. HOTEL



O módulo HOTEL é responsável por suportar a funcionalidade associada à reserva de quartos. Possui as seguintes entidades Hotel, Room e Booking. O hotel é identificado por um identificador explícito, assim como o quarto, sendo no caso do quarto o número, ou seja, os quartos de um hotel são identificados pelo seu número. É possível ter várias instâncias de Hotel e cada hotel possui várias instâncias de Room. Os quartos podem ser singulares ou de casal.

A entidade Booking representa uma reserva de um quarto num determinado período temporal e tem associada uma referência, que serve de identificação da reserva, e que é gerada sequencialmente. Naturalmente, não é possível ter a sobreposição de reservas de um quarto num mesmo período.

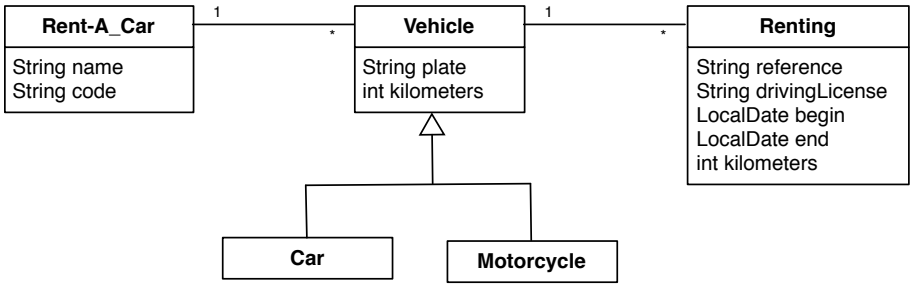
2.3. ACTIVITY



O módulo ACTIVITY é responsável por efetuar a gestão de atividades de lazer. Para isso contém as entidades ActivityProvider, Activity, ActivityOffer e Booking. Cada instância de ActivityProvider possui o conjunto das suas atividades, representadas pela entidade Activity, a qual descreve as idades aconselhadas dos participantes e o número limite de participantes. Uma actividade pode ser oferecida num certo intervalo de tempo, entidade ActivityOffer, podendo ser oferecida várias vezes. As instâncias de ActivityOffer possuem associado o conjunto das suas reservas, entidade Booking.

A reserva de uma atividade tem que obedecer a um conjunto de condições, como seja, a data e a disponibilidade de vaga. Uma vez feita uma reserva é gerado um identificador, através de geração sequencial, que serve como referência da reserva.

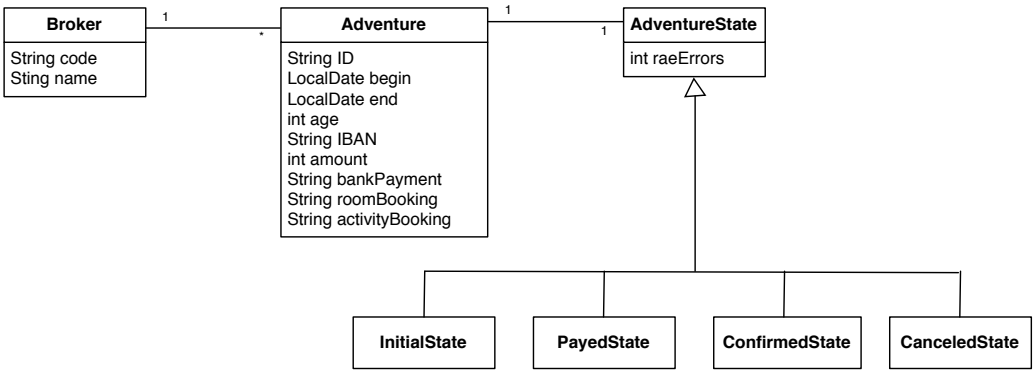
2.4. CAR



O módulo CAR é responsável por efetuar a gestão do aluguer de veículos, pois inclui também o aluguer de veículos motorizados de 2 rodas. Para isso contém as entidades Rent-A-Car, Vehicle com 2 subclasses, Car e Motorcycle, e Booking. Cada instância de Rent-A-Car representa uma empresa de aluguer de veículos motorizados, representados pela entidade Vehicle, a qual indica o número de quilómetros que o veículo possui. As instâncias de Vehicle possuem associado o conjunto dos seus alugueres, entidade Renting, as quais incluem o número de quilómetros que o veículo percorreu durante o aluguer.

O aluguer de um veículo tem que obedecer a um conjunto de condições, como seja, a data e a disponibilidade do veículo. Uma vez efetuada uma reserva é gerado um identificador, através de geração sequencial, que serve como referência. Quando ocorre o fim do aluguer é calculado o número de quilómetros percorridos durante o aluguer e atualizada a quilometragem do veículo.

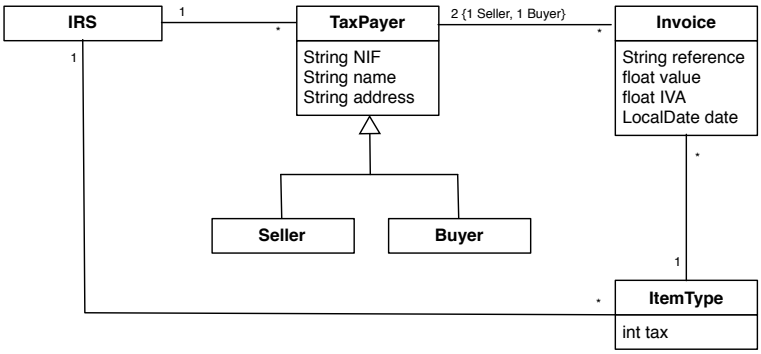
2.5. BROKER



O módulo BROKER é responsável pela gestão das aventuras integrando os serviços dos outros módulos. As suas entidades principais são: Broker, que representa uma empresa que fornece aos seus clientes pacotes integrados de atividades de lazer, e Adventure, que representa o processo de marcação de uma atividade de lazer para um determinado cliente.

O processo de marcação de uma aventura é um processo complexo que ocorre em diversas fases, desde a sua criação inicial até aos pedido de pagamento e reserva. Os pedidos de reserva de hotel e atividade não podem ocorrer antes do pagamento ser efetuado.

2.6. TAX



O módulo TAX é responsável por calcular o IVA gerado em cada compra com dois objetivos, (1) tributar as transações comerciais e (2) calcular os benefícios fiscais que os compradores usufruem por solicitarem a emissão de faturas. Existe uma única instância da entidade IRS a qual tem associada todos os contribuintes, considerando-se, neste caso, dois tipos, disjuntos, de contribuintes, Seller e Buyer, os quais estão associados a uma transação comercial. Cada transação comercial corresponde à venda de um Item, que possui obrigatoriamente associada um comprador e um vendedor. O cálculo do valor de IVA de um Item é efetuado com base no seu valor e na taxa de IVA que está depende do tipo de item, Item Type. Anualmente são calculados os benefícios fiscais dos compradores e a tributação fiscal dos vendedores.

3. Avaliação do projeto

Semanalmente haverá uma avaliação da gestão do projeto. Cada aluno deve contribuir semanalmente com pelo menos uma tarefa que implique o desenvolvimento de código, atualizar o repositório em conformidade e manter o planeamento semanal atualizado (trabalho realizado na semana anterior e trabalho previsto para a semana seguinte). Esta avaliação será feita durante a aula de laboratório de cada equipa. Grupos que não compareçam ao laboratório terão uma avaliação de 0 (zero) nesta componente. Trabalhadores estudantes estão isentos de comparecer nas aulas de avaliação mas devem fornecer, semanalmente, o mesmo material que os restantes colegas. A avaliação terá em conta a qualidade do planeamento do projeto e da gestão do projeto que está a ser aplicada.

O código produzido deve ser guardado no repositório do github.com atribuído a cada equipa.

Cada equipa deve gerir o ficheiro *.gitignore* no diretório base do projeto, indicando os ficheiros que não devem ser colocados no sistema de controlo de versões, tais como o diretório *target* ou ficheiros .class, _Base.java, .jar, ... Projetos que guardem ficheiros desnecessários no repositório terão uma penalização na nota.

O github.com apresenta estatísticas do desenvolvimento do projeto, por exemplo, número de *commits*, linhas adicionadas e removidas por cada membro da equipa. Pessoas em que o github.com mostra uma fraca participação no esforço coletivo de realização do projeto terão uma penalização na nota.

Apenas a informação pedida deve ser impressa no terminal. Qualquer informação de depuração (*debug* ou *trace*) pode ser impressa no terminal através de uma instância da classe Logger do **log4j2** (http://logging.apache.org/log4j/2.x/).

Todo o código e respetiva documentação, comentários no código, *issues* ou páginas *wiki*

devem ser escritos em língua inglesa.

4. Fases de desenvolvimento

O projeto será realizado em equipas de 6 alunos de forma faseada. Cada equipa desenvolve o código no seu repositório github.com.

Para se criar um *workspace* local do projeto do grupo

1. git clone https://github.com/tecnico-softeng/es18CC_GG-project.git, onde, em es18CC_GG-project, CC pode ser "al"ou "tg", e GG é o número do grupo
2. git branch –all, para verificar que existem *branches* remotes/origin

A fase final do projeto é avaliada com visualização e discussão a realizar no final do semestre. Após as discussões, será atribuído a cada aluno uma nota individual que procurará refletir a sua participação no projeto ao longo do semestre.

O projeto contribui com 30% da nota da disciplina e tem uma nota mínima final individual de 8 valores.

O corpo docente da disciplina de Engenharia de Software prevê que a realização do projeto exigirá cerca de 35 horas de trabalho a cada um dos membros da equipa. Nesta previsão, o corpo docente assume que os alunos já compreenderam o funcionamento das tecnologias utilizadas (JUNIT, JMOCKIT, FENIXFRAMEWORK, ...) antes da realização do trabalho e que existe um planeamento do projeto por forma a definir e compreender o trabalho a realizar, bem como a sua distribuição pelos vários elementos da equipa.

Serão publicados, para cada fase, pequenas adendas ao enunciado a detalhar o que será avaliado em cada semana e quaisquer alterações aos requisitos ao trabalho a desenvolver na fase seguinte.