ADVENTURE_3

Setup

Para realizar a terceira parte do projeto deve começar por resolver na totalidade o exercício prático da aula-dml.

Uma vez que domine os conceitos de Fénix Framework e DML pode iniciar a resolução do projeto, obtendo a nova versão do código disponibilizada em https://github.com/tecnico-softeng/reference.git. Para isso faça

```
$git remote add reference https://github.com/tecnico-softeng/re
$git fetch --all
$git checkout master
$git reset --hard reference/master
$git push origin master --force
```

de seguida deve assegurar-se que o código da segunda parte foi entregue através do ramo second-deliver, mudando para esse ramo

```
$git checkout second-deliver
```

pode então apagar o ramo develop, mudar para o ramo master, criar um novo ramo develop a partir do master e colocar a nova versão do develop em origin

```
$git branch -d develop
$git checkout master
$git checkout -b develop
$git push origin develop --force
```

Verifique que consegue correr os testes existentes:

```
$mvn clean test
```

Enunciado

A terceira parte do projeto corresponde a introduzir suporte persistente no domínio dos 6 módulos utilizando a FénixFramework. Para tal, deve refatorizar o código de forma a que seja usada apenas uma base de dados e a domain root possui os conjuntos de instâncias de Hotel, de ActivityProvider, de Bank, RentACar, IRS, e de Broker. Nos módulos broker, activity e hotel já existem implementações iniciais que podem ser usadas como referência. O módulo bank já persiste toda a informação necessária, e pode ser usado como exemplo.

As tarefas básicas de refatorização de código são:

- Tornar persistente uma classe
- Tornar persistente a associação entre duas classes
- Tornar persistente os atributos de uma classe

Cada um dos passos acima deve ser verificado por sucessivos enriquecimentos da classe de teste de persistência do módulo onde há duas transações, a primeira de escrita, onde se altera o domínio, e a segunda de leitura, onde se fazem as verificações que o estado ficou persistente.

Ao iniciar as tarefas (para cada uma delas compile e corra os testes) de um módulo, por exemplo para o módulo car, deve começar por:

- Definir a classe abstrata RollbackTestAbstractClass
- Alterar todos os testes para subclasses de RollbackTestAbstractClass de forma a garantir que a execução dos testes não altera a base de dados
- Tornar persistente a classe de topo do módulo, RentACar
- Associar a classe de topo do módulo à DomainRoot
- Definir classe CarPersistenceTest para testar a persistência da criação de objetos RentACar
- Remover a variável estática RentACar.rentACars
- Adicionar as variáveis code, name, nif e iban na DML

Os grupos devem começar por reunir para identificar as tarefas, associá-las ao backlog, e definir dependências de precedência entre elas, de forma a calendarizá-las no período de 2 semanas disponíveis.

Cada grupo deve dividir-se em sub-grupos de 3/4 alunos. Um dos grupos deve ficar com o módulo car e o outro com o módulo tax. Devem ainda dividir as tarefas associadas à nova informação que é necessário tornar persistente nos restantes módulos, por exemplo, as classes Processor e os novos atributos de algumas classes que já são persistentes. **É obrigatório indicar no README.md quais os elementos que trabalham em cada um dos módulos, indicando o número de aluno, o nome e o username no GitHub.**

Para a entrega deverão fazer:

```
$git checkout develop
$git checkout -b third-deliver
$git tag ADVENTURE_3
$git push origin --tags third-deliver:third-deliver
```

Esta tag colocada deve ter uma data anterior à data limite de entrega, dia **22 de Abril pelas 20:00**.

Durante os laboratórios, os alunos dos grupos serão avaliados com base no trabalho desenvolvido durante cada uma das semanas. Para isso é necessária a criação de um sprint no GitHub, usando a interface de Project, em que cada tarefa deve ser representada por um Note. Neste caso o projeto a criar deve-se chamar Sprint Three. Uma vez criado o projeto, escolhendo como template pré-definido "Kanban (automatic)", deve ser alterado o nome da coluna To Do para Backlog,. Nesta coluna serão colocadas todas as tarefas a realizar, as quais serão definidas durante a reunião do grupo. Cada tarefa é criada como uma Note, contendo a descrição da tarefa a realizar, e deve ser convertida num Issue, por forma a que os commits possam ser-lhe associados. Cada aluno que vá realizar uma tarefa move o respetivo note da coluna de Backlog para a coluna In progress, e atribui-se essa tarefa no Issue respetivo. Quando a tarefa é terminada, o commit deve ser associado ao respetivo Issue, passando este para o estado fechado. Para se associar um commit a um Issue deve colocar-se na mensagem de commit close #n, em que n é o número do Issue. Quando o Issue é fechado o respetivo Note é automaticamente passado da coluna In progress para a coluna Done. **Cada commit deve estar associado a uma única tarefa.**