Universidade Federal do Espírito Santo
Centro Universitário Norte do Espírito Santo
Departamento de Computação e Eletrônica

Fabiano Amaral
Elias Junior

Documentação do software de gestão acadêmica da Universidade Federal de Lugar Nenhum.

A seguir uma breve documentação dos principais pacotes da nossa aplicação e por fim um avaliação do projeto.

model.dao

Esse pacote é responsável por armazenar todas as interfaces do nosso DAO, é onde são listados todos os métodos que trabalharão com nossas entidades, todas as interfaces possuem os seguintes métodos implementados: *create, destroy, find, edit, findall.*

model.dao.jpa

Pacote que tem todas as classes que implementam as interfaces, cada classe desse pacote recebe no seu construtor uma Entity Manager Factory(a partir daqui toda vez que nos referirmos a emf estaremos falando da Entity Manager Factory).

Durante toda a elaboração do projeto foi usada a mesma emf, que foi criada com a mesma *persistence unit*(PU). Internamente as classes do model.dao.jpa funcionam de forma parecida em todas as classes desse pacote então vou falar resumidamente como cada um dos metódos do CRUD funcionam(Será utilizada a entidade aluno como exemplo).

create(Aluno a): A lista de turmas do aluno é verificada, pela remover algum possível null pointer, após isso o aluno é gravado no banco de dados, após o aluno ser gravado no banco de dados é feito um loop que tem como objetivo verificar se aquele aluno está na lista de todas as turmas que ele contém(Já que é um relacionamento bi direcional ManyToMany), os dois lados precisam “se enxergar” então é feita essa checagem.

d**estroy(Aluno a):** O aluno a ser removido é buscado no banco de dados, essa execução está dentro de um bloco try-catch por que o ID informado pode ser inválido, após isso a lista de turmas desse aluno é pega numa list e é iterada removendo o referido aluno de todas as turmas(Caso não fosse feita essa remoção do aluno de todas as turmas existentes teríamos problemas com ID inválidos na lista de alunos de uma determinada turma) que ele está matriculado, após isso é dado um merge em cada turma para que ela seja atualizada no banco e dados e por fim o aluno é excluído do banco de dados utilizado o método remove da classe EntityManager.

findAlunoEntities(boolean all, int maxResult, int firstResult): Neste projeto existe 2 opções de se listar os alunos, por intervalo ou todos. Em ambas opções são feitas chamadas a um método privado da classe, FindAlunoEntities(boolean all, int maxResul, int firstResult) é um método privado que faz a busca, ele é chamado por dois métodos públicos da classe, apenas mudando seus parametros, se você chamar o metodo sem nenhum argumento ele listará todas as entidades, se você chamar com dois argumentos ele listará uma lista que pode ter tamanho maxResult a partir do firstResult. O código desse método está abaixo para melhor entendimento

```
private List<Aluno> findAlunoEntities(boolean all, int maxResults, int firstResult) {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(Aluno.class));
        Query q = em.createQuery(cq);
        if (!all) {
            q.setMaxResults(maxResults);
            q.setFirstResult(firstResult);
        }
        return q.getResultList();
    } finally {
        em.close();
    }
}
```

É criado uma query baseada em critério, após isso verifica se o primeiro argumento do método é true, se for não há por que impor limites a lista de resultado e já é passado para o retorno da query, caso haja esses limitadores eles são impostos a query através de métodos internos da classe. As chamadas dos métodos para listar todos e listar filtrados fica respectivamente da seguinte forma:

```
@Override
public List<Aluno> findAlunoEntities() {
    return findAlunoEntities(true, -1, -1);
}
```

```
@Override
public List<Aluno> findAlunoEntities(int maxResults, int firstResult) {
    return findAlunoEntities(false, maxResults, firstResult);
}
```

findAluno(Long id): É passado um ID para o aluno requerido e após isso dentro de um bloco try-cactch(Afinal esse aluno pode não existir) é feita uma consulta ao banco de dados através da entity manager.

GetAlunoCount(): Um método que não faz nada mais que retornar o número de entidades armazenas na tabela, foi útil em diversas funcionalidades do projeto.

Como mencionado no início, em todas as outras classes esse métodos trabalham de forma parecida, não carecendo de que em cada entidade sejam novamente explicados, claro que em entidades mais complexas, como turma, esses métodos podem crescer um pouco de tamanho, mas o principio básico de funcionamento é o mesmo.

model.dao.jpa.exceptions

É o pacote responsável por armazenas as exceções, como os nomes são bem autoexplicativos é desnecessários explicitar cada comportamento.

Principais desafios do projeto:

Na parte de programação em si não houve muita dificuldade, o grande desafio enfrentado pela dupla na hora de realizar o projeto foi construir uma relação sólida e amigável com o netbeans, já que muitos problemas que surgiram durante o desenvolvimento do projeto não era problema no projeto em si, mas no netbeans, por diversas vezes tivemos de apagar configurações conflituosas da IDE para que o projeto funcionasse, outra coisa que nos deu certa dificuldade, que também foi na parte de configuração, é que não foi tão trivial configurar o banco de dados e conectar, tivemos de usar bibliotecas externas que não conhecíamos(Que até agora não sei na verdade o que faz a jindex). Passado esse primeiro momento de preparação de território fazer as anotações do JPA e construir o CRUD foi brincadeira de criança. Esse trabalho apesar das diversas reclamações da dupla, foi bem proveitoso, aprendemos muitas coisas do

mundo OO que não ainda tínhamos tido oportunidade, como contato com frameworks, banco de dados e design pattern. Agora que o susto de fazer o trabalho já passou podemos dizer que foi um projeto desafiador, devido ao fato de termos de buscar conhecimentos extra classe para a execução do projeto, também tivemos de melhorar nossas skills com sistemas de versionamento, github deu um pau na equipe de desenvolvimento de início mas depois viramos o jogo.