Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Управление сущностями
Тема

Преподаватель _____ А. К. Погребников
подпись, дата | инициалы, фамилия

Студент КИ19-16/1б 031939175 _____ А. Д. Непомнящий
номер группы, зачетной книжки | подпись, дата | инициалы, фамилия

Красноярск 2021

1 Цель работы

Цель работы состоит в получении навыков реализации моделей данных на стороне Java и работы с Criteria API.

2 Задачи

Выполнение работы сводится к следующим задачам.

1. Создать классы сущностей для всех таблиц.

2. Написать методы, реализующие запросы, с использованием Criteria API.

3. Написать метод с использованием Criteria API, возвращающий объект класса, не являющегося сущностью БД.

3 Ход работы

3.1 Классы сущностей

Были созданы сущности для всех таблиц БД. На листингах 1-3 приведены классы сущностей для таблиц фракций, командиров и справочника государственных принадлежностей соответственно.

Листинг 1 – Код класса сущности фракции

```
@Table(name = "factions")
@Entity
public class Faction {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Integer id;

    @Lob
    @Type(type = "org.hibernate.type.TextType")
    @Column(name = "name")
    private String name;

    @ManyToOne (fetch = FetchType.LAZY)
    @JoinColumn(name = "home_system_id")
    private System homeSystem;
```

Окончание листинга 1

```
    @ManyToOne
    @JoinColumn(name = "allegiance_id")
    private Allegiance allegiance;

    @ManyToOne
    @JoinColumn(name = "government_id")
    private Government government;

    @Column(name = "is_player_faction")
    private Boolean isPlayerFaction;

    //Get и Set методы

    @Override
    public String toString() {
        final StringBuilder sb = new StringBuilder("Faction ");
        sb.append(name);
        sb.append(" with allegiance ").append(allegiance);
        sb.append(" and ").append(government);
        sb.append(" government");
        return sb.toString();
    }
}
```

Листинг 2 – Код класса сущности командира

```
package com.github.durakin.serverprogramming.lab4.entity;

import org.hibernate.annotations.Type;

import javax.persistence.*;

@Table(name = "commanders")
@Entity
public class Commander {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Integer id;
```

Окончание листинга 2

```
    @Lob
    @Type(type = "org.hibernate.type.TextType")
    @Column(name = "name", nullable = false)
    private String name;

    @ManyToOne
    @JoinColumn(name = "faction_id")
    private Faction faction;

    //Get и Set методы

    @Override
    public String toString() {
        final StringBuilder sb = new StringBuilder("Commander ");
        sb.append(name);
        sb.append(" from ").append(faction);
        return sb.toString();
    }
}
```

Листинг 3 – Код класса сущности государственной принадлежности

```
@Table(name = "allegiances")
@Entity
public class Allegiance {
    @Id
    @Column(name = "allegiance_id", nullable = false)
    private Integer id;

    @Lob
    @Type(type = "org.hibernate.type.TextType")
    @Column(name = "allegiance")
    private String allegiance;

    //Get и Set методы

    @Override
    public String toString() {
        return allegiance;
```

Окончание листинга 3

```
    }
}
   private FactionRepository factionRepository;
```

3.2 Criteria API

Были написаны методы с использованием Criteria API, в частности – для получения списка командиров по названию фракции и для получения государственной принадлежности командира (через принадлежность его фракции).

Листинг 4 – Код некоторых методов с использованием Criteria API

```
    public List<Commander> FindByFactionName(String name) {


        CriteriaQuery<Commander> commanderCriteriaQuery =
criteriaBuilder.createQuery(Commander.class);
        Root<Commander> root =
commanderCriteriaQuery.from(Commander.class);
        Join<Commander, Faction> factionJoin = root.join("faction");

commanderCriteriaQuery.where(criteriaBuilder.equal(factionJoin.get("name"
), name));


        return
entityManager.createQuery(commanderCriteriaQuery).getResultList();
    }


    public List<Commander> FindByAllegianceName(String allegiance) {


        CriteriaQuery<Commander> commanderCriteriaQuery =
criteriaBuilder.createQuery(Commander.class);
        Root<Commander> root =
commanderCriteriaQuery.from(Commander.class);
        Join<Commander, Faction> factionJoin = root.join("faction");
        Join<Faction, Allegiance> allegianceJoin =
factionJoin.join("allegiance");

commanderCriteriaQuery.where(criteriaBuilder.equal(allegianceJoin.get("al
legiance"), allegiance));
        return
entityManager.createQuery(commanderCriteriaQuery).getResultList();
    }
```

3.3 Получение объекта произвольного класса из БД

Был написан метод, возвращающий государственную принадлежность всех командиров. Данный метод возвращает список объектов класса CommanderAllegiance, не являющегося сущностью БД. Код класса и сущности приведен далее на листингах 5 и 6 соответственно.

Листинг 5 – Код класса CommanderAllegiance

```
public class CommanderAllegiance {
    private String name;
    private String allegiance;

    public CommanderAllegiance(String name, String allegiance) {
        this.name = name;
        this.allegiance = allegiance;
    }

    //Get и Set методы

    @Override
    public String toString() {
        final StringBuilder sb = new StringBuilder("Commander ");
        sb.append(name);
        sb.append(", allegiance ").append(allegiance);
        return sb.toString();
    }
}
```

Листинг 6 – Код метода получения списка

```
    public List<CommanderAllegiance> FindCommandersAllegiance() {
        CriteriaQuery<CommanderAllegiance> commanderCriteriaQuery =
criteriaBuilder.createQuery(CommanderAllegiance.class);
        Root<Commander> root =
commanderCriteriaQuery.from(Commander.class);
        Join<Commander, Faction> factionJoin = root.join("faction");
        Join<Faction, Allegiance> allegianceJoin =
factionJoin.join("allegiance");

commanderCriteriaQuery.select(criteriaBuilder.construct(CommanderAllegian
ce.class, root.get("name"), allegianceJoin.get("allegiance")));
```

Окончание листинга 6

```
        return
entityManager.createQuery(commanderCriteriaQuery).getResultList();
    }
```

4 Вывод

В ходе работы были получены навыки реализации моделей данных на стороне Java и работы с Criteria API.