**Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Информационные системы
Отчёт по лабораторной работе №1
Вариант 5366377

Выполнил: Зинченко Иван Николаевич
Группа: P3312

Санкт-Петербург
2025 год

Текст задания

Реализовать информационную систему, которая позволяет взаимодействовать с объектами класса `Vehicle`, описание которого приведено ниже:

```
public class Vehicle {
    private int id; //Значение поля должно быть больше 0, Значение этого поля должно
быть уникальным, Значение этого поля должно генерироваться автоматически
    private String name; //Поле не может быть null, Строка не может быть пустой
    private Coordinates coordinates; //Поле не может быть null
    private java.util.Date creationDate; //Поле не может быть null, Значение этого
поля должно генерироваться автоматически
    private VehicleType type; //Поле может быть null
    private int enginePower; //Значение поля должно быть больше 0
    private Long numberOfWheels; //Поле может быть null, Значение поля должно быть
больше 0
    private Double capacity; //Поле может быть null, Значение поля должно быть больше
0
    private int distanceTravelled; //Значение поля должно быть больше 0
    private double fuelConsumption; //Значение поля должно быть больше 0
    private FuelType fuelType; //Поле может быть null
}
public class Coordinates {
    private double x;
    private Double y; //Максимальное значение поля: 910, Поле не может быть null
}
public enum VehicleType {
    BOAT,
    SHIP,
    MOTORCYCLE,
    CHOPPER;
}
public enum FuelType {
    KEROSENE,
    ALCOHOL,
    MANPOWER,
    PLASMA,
    ANTIMATTER;
}
```

Разработанная система должна удовлетворять следующим требованиям:

- Основное назначение информационной системы - управление объектами, созданными на основе заданного в варианте класса.
- Необходимо, чтобы с помощью системы можно было выполнить следующие операции с объектами: создание нового объекта,- получение информации об объекте по ИД, обновление объекта (модификация его атрибутов), удаление объекта. Операции- должны осуществляться в отдельных окнах (интерфейсах) приложения. При получении информации об объекте класса должна- также выводиться информация о связанных с ним объектах.
- При создании объекта класса необходимо дать пользователю возможность связать новый объект с объектами- вспомогательных классов, которые могут быть связаны с созданным объектом и уже есть в системе.
- Выполнение операций по управлению объектами должно осуществляться на серверной части (не на клиенте), изменения- должны синхронизироваться с базой данных.

- На главном экране системы должен выводиться список текущих объектов в виде таблицы (каждый атрибут объекта – отдельная колонка в таблице). При отображении таблицы должна использоваться пагинация (если все объекты не- помещаются на одном экране).
- Нужно обеспечить возможность фильтровать/сортировать строки таблицы, которые показывают объекты (по значениям любой- из строковых колонок). Фильтрация элементов должна производиться по неполному совпадению.
- Переход к обновлению (модификации) объекта должен быть возможен из таблицы с общим списком объектов и из области с- визуализацией объекта (при ее реализации).
- При добавлении/удалении/изменении объекта, он должен автоматически появиться/ исчезнуть/измениться в интерфейсах у- других пользователей, авторизованных в системе.
- Если при удалении объекта с ним связан другой объект, связанные объекты должны быть связаны с другим объектом (по- выбору пользователя), а изначальный объект удален.
- Пользователи должны иметь возможность просмотра всех объектов. Для модификации объекта должно открываться отдельное диалоговое окно. При вводе некорректных значений в поля объекта должны появляться информативные сообщения о соответствующих ошибках.

В системе должен быть реализован отдельный пользовательский интерфейс для выполнения специальных операций над объектами:

- Вернуть один (любой) объект, значение поля enginePower которого является минимальным.
- Вернуть количество объектов, значение поля fuelType которых больше заданного.
- Вернуть массив объектов, значение поля name которых содержит заданную подстроку.
- Найти все транспортные средства с мощностью двигателя в заданном диапазоне.
- «Скрутить» счётчик пробега транспортного средства с заданным id до нуля.

Представленные операции должны быть реализованы в качестве функций БД, которые необходимо вызывать из уровня бизнес-логики приложения.

Особенности хранения объектов, которые должны быть реализованы в системе:

- Организовать хранение данных об объектах в реляционной СУБД (PostgreSQL). Каждый объект, с которым работает ИС, должен быть сохранен в базе данных.
- Все требования к полям класса (указанные в виде комментариев к описанию классов) должны быть выполнены на уровне ORM и БД.
- Для генерации поля id использовать средства базы данных.
- Для подключения к БД на кафедральном сервере использовать хост pg, имя базы данных - studs, имя пользователя/пароль совпадают с таковыми для подключения к серверу.

При создании системы нужно учитывать следующие особенности организации взаимодействия с пользователем:

- Система должна реагировать на некорректный пользовательский ввод, ограничивая ввод недопустимых значений и информируя пользователей о причине ошибки.
- Переходы между различными логически обособленными частями системы должны осуществляться с помощью меню.
- При добавлении/удалении/изменении объекта, он должен автоматически появиться/ исчезнуть/измениться на области у всех других клиентов.

При разработке ИС должны учитываться следующие требования:

- В качестве основы для реализации ИС необходимо использовать Jakarta EE, CDI beans.
- Для создания уровня хранения необходимо использовать EclipseLink.

- Разные уровни приложения должны быть отделены друг от друга, разные логические части ИС должны находиться в отдельных компонентах.

Реализация

SQL модель

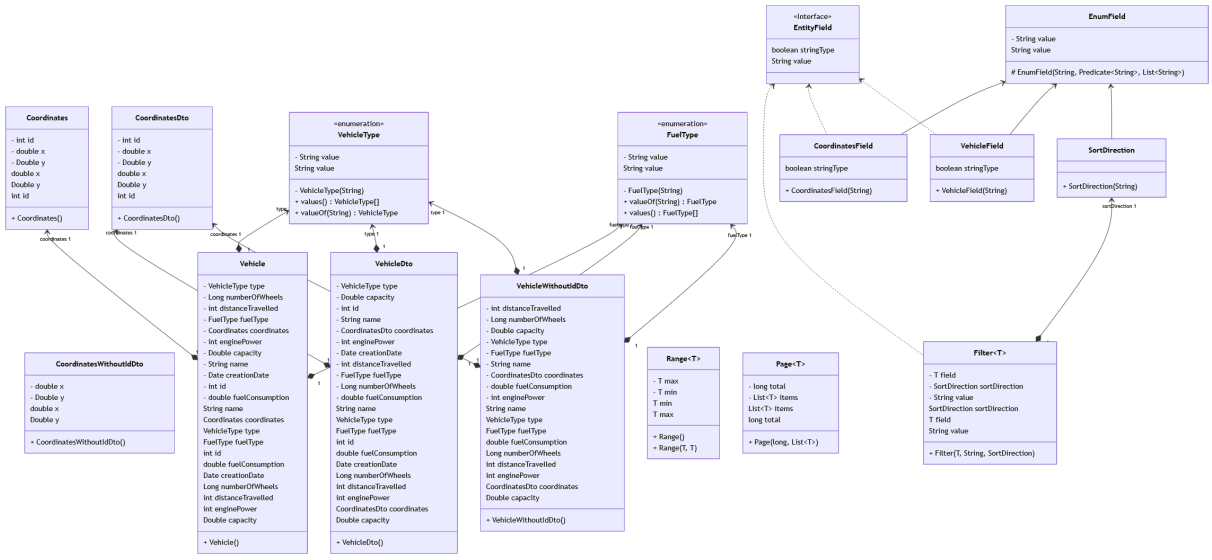
```
CREATE TYPE VEHICLE_TYPE AS ENUM ('BOAT', 'SHIP', 'MOTORCYCLE', 'CHOPPER');

CREATE TYPE FUEL_TYPE AS ENUM ('KEROSENE', 'ALCOHOL', 'MANPOWER', 'PLASMA',
'ANTIMATTER');

CREATE TABLE IF NOT EXISTS coordinates (
  id SERIAL PRIMARY KEY,
  x NUMERIC NOT NULL,
  y NUMERIC NOT NULL CHECK (y <= 910),
  UNIQUE(x, y)
);

CREATE TABLE IF NOT EXISTS vehicles (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL CHECK (LENGTH(name) > 0),
  coordinateId SERIAL REFERENCES coordinates(id),
  creationDate TIMESTAMP DEFAULT NOW() NOT NULL,
    type VEHICLE_TYPE,
  enginePower INTEGER NOT NULL CHECK (enginePower > 0),
  numberOfWheels BIGINT CHECK (numberOfWheels > 0),
  capacity NUMERIC CHECK(capacity > 0),
  distanceTravelled INTEGER CHECK (distanceTravelled > 0),
  fuelConsumption INTEGER CHECK (fuelConsumption > 0),
  fuelType FUEL_TYPE NOT NULL
);

CREATE OR REPLACE FUNCTION get_vehicle_with_min_engine_power()
RETURNS TABLE(
    id INTEGER,
    name TEXT,
    coordinateId INTEGER,
    creationDate TIMESTAMP,
    type VEHICLE_TYPE,
    enginePower INTEGER,
    numberOfWheels BIGINT,
    capacity NUMERIC,
    distanceTravelled INTEGER,
    fuelConsumption INTEGER,
    fuelType FUEL_TYPE
) AS $$
BEGIN
    RETURN QUERY
    SELECT v.*
    FROM vehicles v
    WHERE v.enginePower = (SELECT MIN(vv.enginePower) FROM vehicles vv)
    LIMIT 1;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION count_vehicles_with_fuel_type_greater_than(given_fuel_type
```

```
FUEL_TYPE)
RETURNS INTEGER AS $$
DECLARE
    vehicle_count INTEGER;
BEGIN
    SELECT COUNT(*) INTO vehicle_count
    FROM vehicles
    WHERE fuelType > given_fuel_type;

    RETURN vehicle_count;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get_vehicles_with_name_containing(substring_text TEXT)
RETURNS TABLE(
    id INTEGER,
    name TEXT,
    coordinateId INTEGER,
    creationDate TIMESTAMP,
    type VEHICLE_TYPE,
    enginePower INTEGER,
    numberOfWheels BIGINT,
    capacity NUMERIC,
    distanceTravelled INTEGER,
    fuelConsumption INTEGER,
    fuelType FUEL_TYPE
) AS $$
BEGIN
    RETURN QUERY
    SELECT v.*
    FROM vehicles v
    WHERE v.name ILIKE '%' || substring_text || '%';
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get_vehicles_in_engine_power_range(min_power INTEGER,
max_power INTEGER)
RETURNS TABLE(
    id INTEGER,
    name TEXT,
    coordinateId INTEGER,
    creationDate TIMESTAMP,
    type VEHICLE_TYPE,
    enginePower INTEGER,
    numberOfWheels BIGINT,
    capacity NUMERIC,
    distanceTravelled INTEGER,
    fuelConsumption INTEGER,
    fuelType FUEL_TYPE
) AS $$
BEGIN
    RETURN QUERY
    SELECT v.*
    FROM vehicles v
    WHERE v.enginePower BETWEEN min_power AND max_power
    ORDER BY v.enginePower;
```

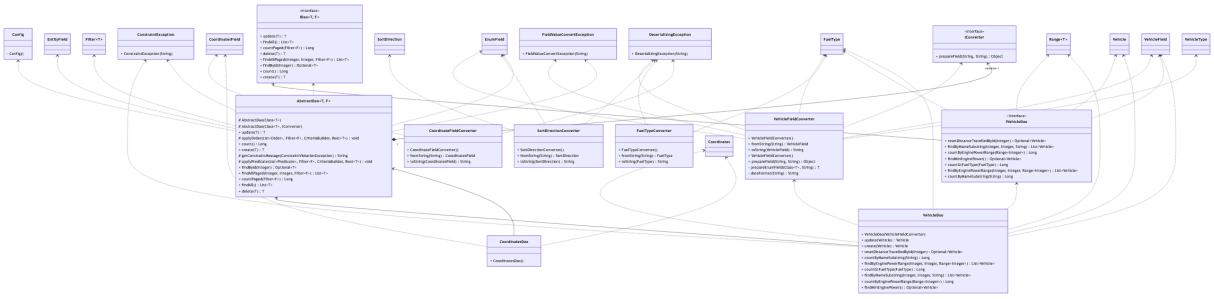
```
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION reset_vehicle_distance(vehicle_id INTEGER)
RETURNS VOID AS $$
BEGIN
    UPDATE vehicles
    SET distanceTravelled = 0
    WHERE id = vehicle_id;

    IF NOT FOUND THEN
        RAISE EXCEPTION 'Vehicle with id % not found', vehicle_id;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

UML Диаграммы

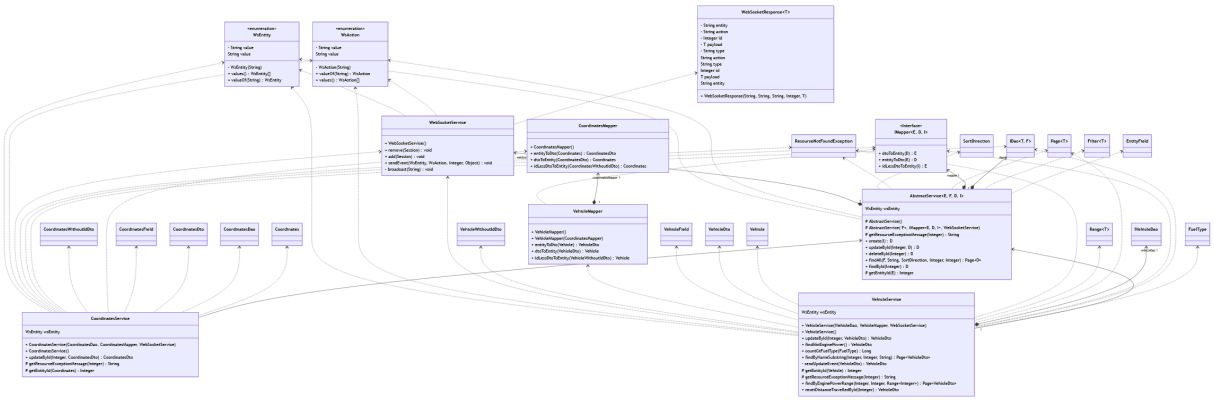
Слой моделей



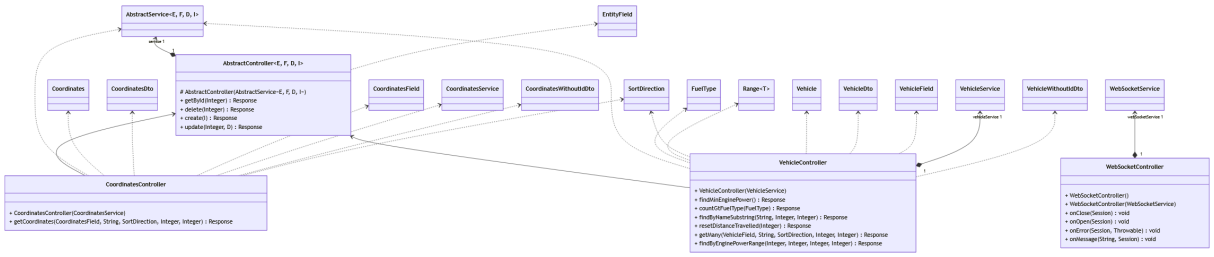
Слой DAO


Config
SortOption
Filter<T>
ConstraintException
CoordinateField
«interface»
Base<T, F>
SortDirection
EnumField
FieldValueConvertException
DeserializingException
FuelType
«interface»
IConverter
Range<T>
Vehicle
VehicleField
VehicleType
AbstractDao<T, F>
CoordinateFieldConverter
SortDirectionConverter
FuelTypeConverter
Coordinates
VehicleFieldConverter
«interface»
VehicleDao
CoordinatesDao
VehicleDao

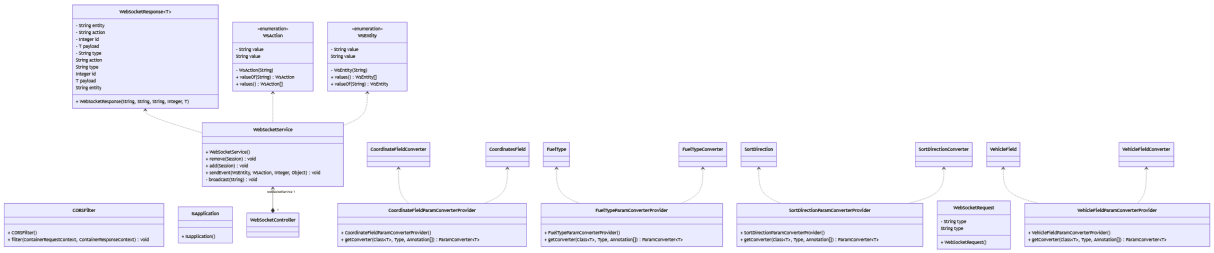
Сервисный Слой



Слой контроллеров

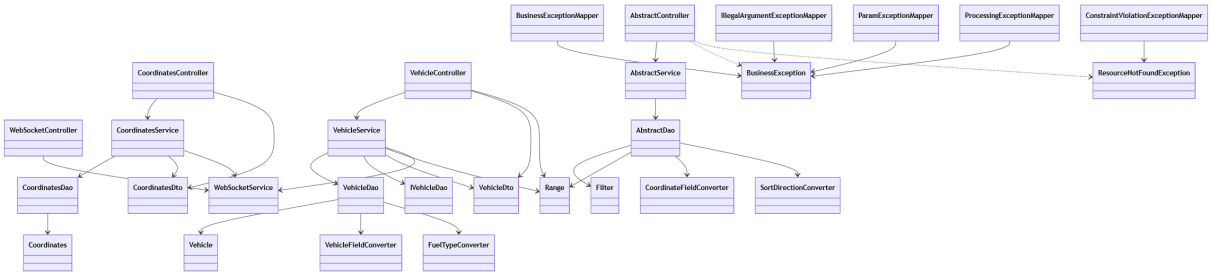


Слой серверных перехватчиков



Слой исключений

Связь между слоями



Исходный код

Ссылка на репозиторий – https://github.com/zinchenko291/is-lab-1.git