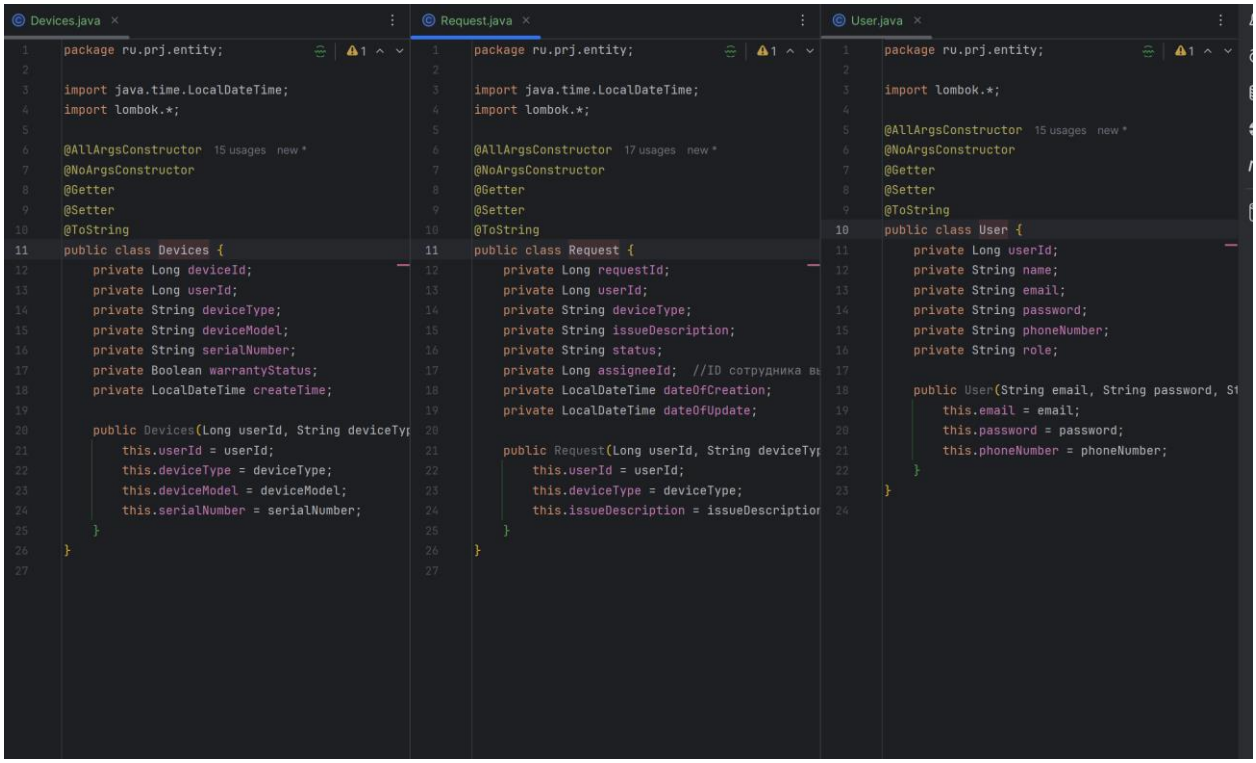
Задание1. Реализовать классы для каждой из таблиц базы данных в папке model.

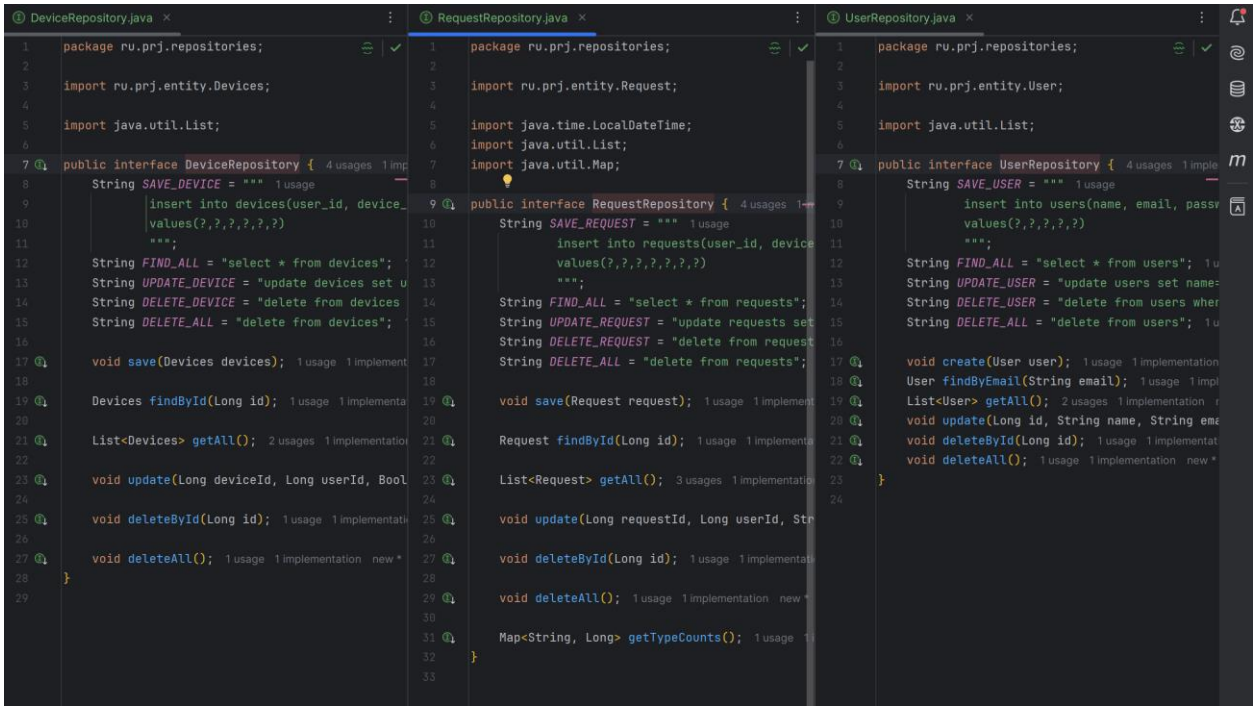


```java
// Devices.java
package ru.prj.entity;

import java.time.LocalDateTime;
import lombok.*;

@AllArgsConstructor  15 usages  new *
@NoArgsConstructor
@Getter
@Setter
@ToString
public class Devices {
    private Long deviceId;
    private Long userId;
    private String deviceType;
    private String deviceModel;
    private String serialNumber;
    private Boolean warrantyStatus;
    private LocalDateTime createTime;

    public Devices(Long userId, String deviceTyp
        this.userId = userId;
        this.deviceType = deviceType;
        this.deviceModel = deviceModel;
        this.serialNumber = serialNumber;
    }
}
```

```java
// Request.java
package ru.prj.entity;

import java.time.LocalDateTime;
import lombok.*;

@AllArgsConstructor  17 usages  new *
@NoArgsConstructor
@Getter
@Setter
@ToString
public class Request {
    private Long requestId;
    private Long userId;
    private String deviceType;
    private String issueDescription;
    private String status;
    private Long assigneeId;  //ID сотрудника вы
    private LocalDateTime dateOfCreation;
    private LocalDateTime dateOfUpdate;

    public Request(Long userId, String deviceTyp
        this.userId = userId;
        this.deviceType = deviceType;
        this.issueDescription = issueDescriptior
    }
}
```

```java
// User.java
package ru.prj.entity;

import lombok.*;

@AllArgsConstructor  15 usages  new *
@NoArgsConstructor
@Getter
@Setter
@ToString
public class User {
    private Long userId;
    private String name;
    private String email;
    private String password;
    private String phoneNumber;
    private String role;

    public User(String email, String password, St
        this.email = email;
        this.password = password;
        this.phoneNumber = phoneNumber;
    }
}
```

Задание2. Реализовать интерфейсы по работе с 3 и более таблицами в папке repository.

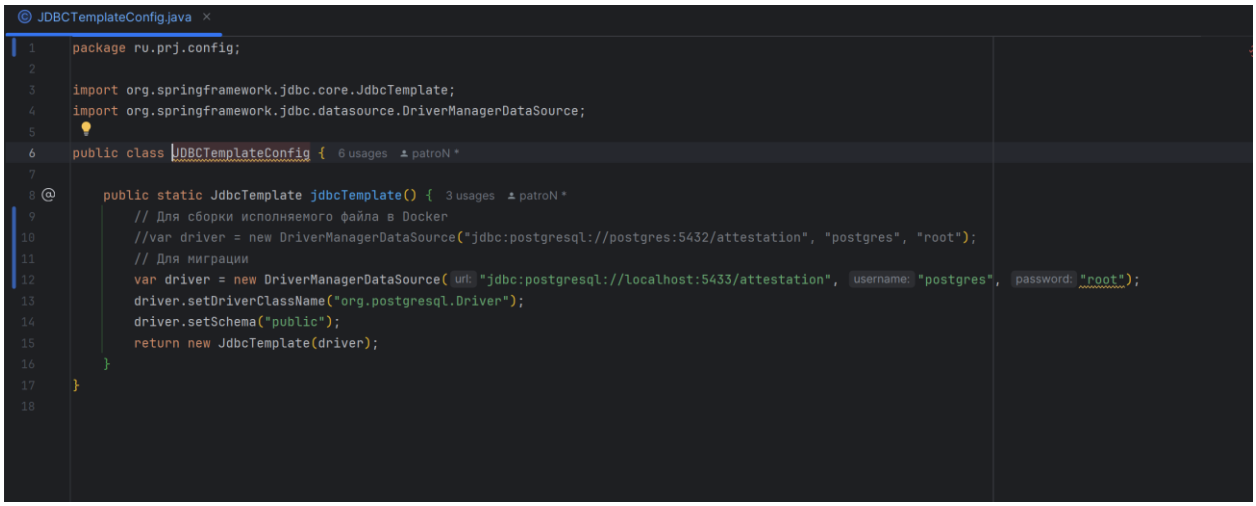


```java
// DeviceRepository.java
package ru.prj.repositories;

import ru.prj.entity.Devices;

import java.util.List;

public interface DeviceRepository {  4 usages  1 imp
    String SAVE_DEVICE = """  1 usage
            insert into devices(user_id, device_
            values(?,?,?,?,?,?)
            """;
    String FIND_ALL = "select * from devices";
    String UPDATE_DEVICE = "update devices set u
    String DELETE_DEVICE = "delete from devices
    String DELETE_ALL = "delete from devices";

    void save(Devices devices);  1 usage  1 implement

    Devices findById(Long id);  1 usage  1 implementa

    List<Devices> getAll();  2 usages  1 implementatio

    void update(Long deviceId, Long userId, Bool

    void deleteById(Long id);  1 usage  1 implementati

    void deleteAll();  1 usage  1 implementation  new *
}
```

```java
// RequestRepository.java
package ru.prj.repositories;

import ru.prj.entity.Request;

import java.time.LocalDateTime;
import java.util.List;
import java.util.Map;

public interface RequestRepository {  4 usages  1
    String SAVE_REQUEST = """  1 usage
            insert into requests(user_id, device
            values(?,?,?,?,?,?,?)
            """;
    String FIND_ALL = "select * from requests";
    String UPDATE_REQUEST = "update requests set
    String DELETE_REQUEST = "delete from request
    String DELETE_ALL = "delete from requests";

    void save(Request request);  1 usage  1 implement

    Request findById(Long id);  1 usage  1 implementa

    List<Request> getAll();  3 usages  1 implementatio

    void update(Long requestId, Long userId, Str

    void deleteById(Long id);  1 usage  1 implementat

    void deleteAll();  1 usage  1 implementation  new *

    Map<String, Long> getTypeCounts();  1 usage  1
}
```

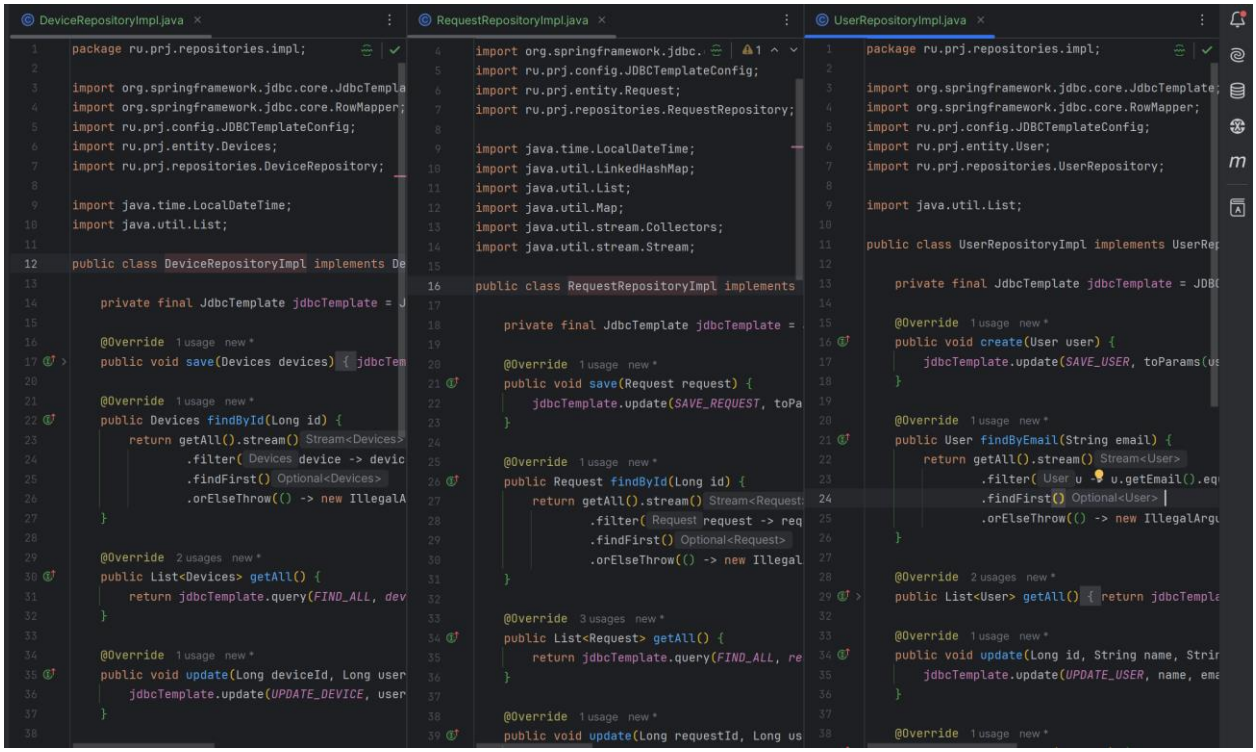
```java
// UserRepository.java
package ru.prj.repositories;

import ru.prj.entity.User;

import java.util.List;

public interface UserRepository {  4 usages  1 imple
    String SAVE_USER = """  1 usage
            insert into users(name, email, passv
            values(?,?,?,?,?)
            """;
    String FIND_ALL = "select * from users";  1 u
    String UPDATE_USER = "update users set name=
    String DELETE_USER = "delete from users wher
    String DELETE_ALL = "delete from users";  1 u

    void create(User user);  1 usage  1 implementation
    User findByEmail(String email);  1 usage  1 impl
    List<User> getAll();  2 usages  1 implementation
    void update(Long id, String name, String ema
    void deleteById(Long id);  1 usage  1 implementat
    void deleteAll();  1 usage  1 implementation  new *
}
```

Задание 3. Для работы с базой данных предварительно реализуются методы выгрузки данных из базы. Использовать JDBCTemplate.



Прошу обратить внимание, на вариации реализаций driver, перед сборкой исполняемого файла, необходимо раскомментировать соответствующую строку, после сборки и развертывания на Docker, закомментировать и раскомментировать другую, для выполнения миграции баз данных, после чего есть возможность взаимодействовать с приложением и базой данных средствами Docker.

Задание 4. Реализовать классы-наследники (имплементация) интерфейсов по работе с таблицами базы данных в папке repository.



Задание 5. Реализовать класс App для проверки работоспособности приложения. Если был опыт работы с JUnit5 тестами на Java допустимо заменить данный пункт написанием тестов.