

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

```
Devices.java x
1 package ru.prj.entity;
2
3 import java.time.LocalDateTime;
4 import lombok.*;
5
6 @AllArgsConstructor 15 usages new *
7 @NoArgsConstructor
8 @Getter
9 @Setter
10 @ToString
11 public class Devices {
12     private Long deviceId;
13     private Long userId;
14     private String deviceType;
15     private String deviceModel;
16     private String serialNumber;
17     private Boolean warrantyStatus;
18     private LocalDateTime createTime;
19
20     public Devices(Long userId, String deviceType,
21         this.userId = userId;
22         this.deviceType = deviceType;
23         this.deviceModel = deviceModel;
24         this.serialNumber = serialNumber;
25     }
26 }
27

Request.java x
1 package ru.prj.entity;
2
3 import java.time.LocalDateTime;
4 import lombok.*;
5
6 @AllArgsConstructor 17 usages new *
7 @NoArgsConstructor
8 @Getter
9 @Setter
10 @ToString
11 public class Request {
12     private Long requestId;
13     private Long userId;
14     private String deviceType;
15     private String issueDescription;
16     private String status;
17     private Long assigneeId; //ID сотрудника в
18     private LocalDateTime dateOfCreation;
19     private LocalDateTime dateOfUpdate;
20
21     public Request(Long userId, String deviceType,
22         this.userId = userId;
23         this.deviceType = deviceType;
24         this.issueDescription = issueDescription
25     }
26 }
27

User.java x
1 package ru.prj.entity;
2
3 import lombok.*;
4
5 @AllArgsConstructor 15 usages new *
6 @NoArgsConstructor
7 @Getter
8 @Setter
9 @ToString
10 public class User {
11     private Long userId;
12     private String name;
13     private String email;
14     private String password;
15     private String phoneNumber;
16     private String role;
17
18     public User(String email, String password, St
19         this.email = email;
20         this.password = password;
21         this.phoneNumber = phoneNumber;
22     }
23 }
24 }
```

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

```
DeviceRepository.java x
1 package ru.prj.repositories;
2
3 import ru.prj.entity.Devices;
4
5 import java.util.List;
6
7 public interface DeviceRepository { 4 usages 1 imple
8     String SAVE_DEVICE = "insert into devices(user_id, device_
9         values(?, ?, ?, ?, ?)
10         """;
11     String FIND_ALL = "select * from devices";
12     String UPDATE_DEVICE = "update devices set u
13     String DELETE_DEVICE = "delete from devices";
14     String DELETE_ALL = "delete from devices";
15
16     void save(Devices devices); 1 usage 1 implement
17
18     Devices findById(Long id); 1 usage 1 implementa
19
20     List<Devices> getAll(); 2 usages 1 implementation
21
22     void update(Long deviceId, Long userId, Bool
23
24     void deleteById(Long id); 1 usage 1 implementa
25
26     void deleteAll(); 1 usage 1 implementation new *
27 }
28
29

RequestRepository.java x
1 package ru.prj.repositories;
2
3 import ru.prj.entity.Request;
4
5 import java.time.LocalDateTime;
6 import java.util.List;
7 import java.util.Map;
8
9 public interface RequestRepository { 4 usages 1 imple
10     String SAVE_REQUEST = "insert into requests(user_id, device_
11         values(?, ?, ?, ?, ?, ?)
12         """;
13     String FIND_ALL = "select * from requests";
14     String UPDATE_REQUEST = "update requests set
15     String DELETE_REQUEST = "delete from request
16     String DELETE_ALL = "delete from requests";
17
18     void save(Request request); 1 usage 1 implement
19
20     Request findById(Long id); 1 usage 1 implementa
21
22     List<Request> getAll(); 3 usages 1 implementation
23
24     void update(Long requestId, Long userId, Str
25
26     void deleteById(Long id); 1 usage 1 implementa
27
28     void deleteAll(); 1 usage 1 implementation new *
29
30     Map<String, Long> getTypeCounts(); 1 usage 1
31 }
32
33

UserRepository.java x
1 package ru.prj.repositories;
2
3 import ru.prj.entity.User;
4
5 import java.util.List;
6
7 public interface UserRepository { 4 usages 1 imple
8     String SAVE_USER = "insert into users(name, email, passw
9         values(?, ?, ?, ?)
10         """;
11     String FIND_ALL = "select * from users"; 1 u
12     String UPDATE_USER = "update users set name=
13     String DELETE_USER = "delete from users wher
14     String DELETE_ALL = "delete from users"; 1 u
15
16     void create(User user); 1 usage 1 implementation
17
18     User findByIdEmail(String email); 1 usage 1 impl
19
20     List<User> getAll(); 2 usages 1 implementation r
21
22     void update(Long id, String name, String eme
23     void deleteById(Long id); 1 usage 1 implementa
24     void deleteAll(); 1 usage 1 implementation new *
```

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

```
JDBCTemplateConfig.java
1 package ru.prj.config;
2
3 import org.springframework.jdbc.core.JdbcTemplate;
4 import org.springframework.jdbc.datasource.DriverManagerDataSource;
5
6 public class JDBCTemplateConfig {
7
8     @
9     public static JdbcTemplate jdbcTemplate() {
10         // Для сборки исполняемого файла в Docker
11         //var driver = new DriverManagerDataSource("jdbc:postgresql://postgres:5432/attestation", "postgres", "root");
12         // Для миграции
13         var driver = new DriverManagerDataSource("jdbc:postgresql://localhost:5433/attestation", "username: \"postgres\", password: \"root\"");
14         driver.setDriverClassName("org.postgresql.Driver");
15         driver.setSchema("public");
16         return new JdbcTemplate(driver);
17     }
18 }
```

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.

```
DeviceRepositoryImpl.java
1 package ru.prj.repositories.impl;
2
3 import org.springframework.jdbc.core.JdbcTemplate;
4 import org.springframework.jdbc.core.RowMapper;
5 import ru.prj.config.JDBCTemplateConfig;
6 import ru.prj.entity.Devices;
7 import ru.prj.repositories.DeviceRepository;
8
9 import java.time.LocalDateTime;
10 import java.util.List;
11
12 public class DeviceRepositoryImpl implements DeviceRepository {
13
14     private final JdbcTemplate jdbcTemplate = JDBCTemplateConfig.jdbcTemplate();
15
16     @Override
17     public void save(Devices devices) {
18         jdbcTemplate.update("INSERT INTO devices (name, device_id, user_id) VALUES (?, ?, ?)", devices.getName(), devices.getDeviceId(), devices.getUserId());
19     }
20
21     @Override
22     public Devices findById(Long id) {
23         return getAll().stream().filter(device -> device.getDeviceId().equals(id)).findFirst().orElseThrow(() -> new IllegalArgumentException("Device not found"));
24     }
25
26     @Override
27     public List<Devices> getAll() {
28         return jdbcTemplate.query(FIND_ALL, dev);
29     }
30
31     @Override
32     public void update(Long deviceId, Long userId, String name) {
33         jdbcTemplate.update(UPDATE_DEVICE, deviceId, userId, name);
34     }
35 }
```

```
RequestRepositoryImpl.java
1 import org.springframework.jdbc.core.JdbcTemplate;
2 import ru.prj.config.JDBCTemplateConfig;
3 import ru.prj.entity.Request;
4 import ru.prj.repositories.RequestRepository;
5
6 import java.time.LocalDateTime;
7 import java.util.List;
8 import java.util.stream.Collectors;
9 import java.util.stream.Stream;
10
11 public class RequestRepositoryImpl implements RequestRepository {
12
13     private final JdbcTemplate jdbcTemplate = JDBCTemplateConfig.jdbcTemplate();
14
15     @Override
16     public void save(Request request) {
17         jdbcTemplate.update("INSERT INTO requests (request_id, user_id, request_text) VALUES (?, ?, ?)", request.getRequestId(), request.getUserId(), request.getRequestText());
18     }
19
20     @Override
21     public Request findById(Long id) {
22         return getAll().stream().filter(request -> request.getRequestId().equals(id)).findFirst().orElseThrow(() -> new IllegalArgumentException("Request not found"));
23     }
24
25     @Override
26     public List<Request> getAll() {
27         return jdbcTemplate.query(FIND_ALL, req);
28     }
29
30     @Override
31     public void update(Long requestId, Long userId, String requestText) {
32         jdbcTemplate.update(UPDATE_REQUEST, requestId, userId, requestText);
33     }
34 }
```

```
UserRepositoryImpl.java
1 package ru.prj.repositories.impl;
2
3 import org.springframework.jdbc.core.JdbcTemplate;
4 import org.springframework.jdbc.core.RowMapper;
5 import ru.prj.config.JDBCTemplateConfig;
6 import ru.prj.entity.User;
7 import ru.prj.repositories.UserRepository;
8
9 import java.util.List;
10
11 public class UserRepositoryImpl implements UserRepository {
12
13     private final JdbcTemplate jdbcTemplate = JDBCTemplateConfig.jdbcTemplate();
14
15     @Override
16     public void create(User user) {
17         jdbcTemplate.update("INSERT INTO users (username, email, password) VALUES (?, ?, ?)", user.getUsername(), user.getEmail(), user.getPassword());
18     }
19
20     @Override
21     public User findByIdEmail(String email) {
22         return getAll().stream().filter(user -> user.getEmail().equals(email)).findFirst().orElseThrow(() -> new IllegalArgumentException("User not found"));
23     }
24
25     @Override
26     public List<User> getAll() {
27         return jdbcTemplate.query(FIND_ALL, user);
28     }
29
30     @Override
31     public void update(Long id, String name, String email, String password) {
32         jdbcTemplate.update(UPDATE_USER, id, name, email, password);
33     }
34 }
```

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

```
SecondAttest.java ×
15
16 public class SecondAttest { new *
17
18     private static final Scanner scanner = new Scanner(System.in); 34 usages
19     private static final UserRepository USER_REPOSITORY = new UserRepositoryImpl(); 6 usages
20     private static final DeviceRepository DEVICE_REPOSITORY = new DeviceRepositoryImpl(); 6 usages
21     private static final RequestRepository REQUEST_REPOSITORY = new RequestRepositoryImpl(); 7 usages
22
23 public static void main(String[] args) { new *
24     int continueProgramm = 1;
25
26     while (continueProgramm == 1) {
27         System.out.println("Please choose with which table you would like to search:\n");
28         System.out.println("1. Devices\n2. Requests\n3. Users\n4. Exit");
29
30         int choice = getValidChoice();
31         switch (choice) {
32             case 1:
33                 takeInformFromDevices();
34                 break;
35             case 2:
36                 takeInformFromRequests();
37                 break;
38             case 3:
39                 takeInformFromUsers();
40                 break;
41             case 4:
42                 continueProgramm = 0;
43                 break;
44             default:
45                 System.out.println("Invalid choice");
46         }
47     }
48 }
49
```