Университет ИТМО

Мегафакультет компьютерных технологий и управления

Факультет программной инженерии и компьютерной техники

ОТЧЕТ ПО ПРОЕКТИРОВАНИЮ И РАЗРАБОТКЕ ДИЗАЙН ПРОЕКТА

Курса «Рефакторинг баз данных и приложений»

Санкт-Петербург 2022

В данном спринте было проведено следующее:

1. В контроллерах в обоих приложениях стоит возвращать объект класса ResponseEntity и статус response
2. В некоторых контроллерах поменять метод запроса, например запрос на удаление вместо POST использовать DELETE
3. Составить отчет с описанием всех возможных вариантов запросов и http response к ним

Изменения в коде.

Пример (1 приложение, class PatientsController, метод deletePatient):

Было:

```
@PostMapping("delete")
public MessageDto deletePatient(@RequestBody PatientDto patientDto) {
    return patientsService.deletePatient(patientDto);
}
```

Стало:

```
@DeleteMapping("delete")
public ResponseEntity<MessageDto> deletePatient(@RequestBody PatientDto patientDto) {
    MessageDto messageDto = patientsService.deletePatient(patientDto);
    if(messageDto.getMessage().equals("The patient was successfully removed")) {
        return ResponseEntity.status(HttpStatus.OK).body(messageDto);
    }
    return ResponseEntity.status(HttpStatus.NOT_FOUND).body(messageDto);
}
```

(аналогично с другими методами)

Добавлен ExceptionHandler, который ловит ошибки, связанные с неправильным использованием HTTP-метода для каждого запроса:

```
@ExceptionHandler({ MethodArgumentTypeMismatchException.class})
public ResponseEntity handleBaseExceptions() {
    return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Invalid method type");
}
```

Для каждого метода был определен свой набор http-response и статусов, приведем несколько примеров.

1. appointments/edit_time_pattern (1 приложение)

```
    @PatchMapping("edit_time_pattern")
    public ResponseEntity<MessageDto> editTimePattern(@RequestBody AppointmentDto appointmentDto) {
        MessageDto messageDto = appointmentsService.editTimePattern(appointmentDto);
        if(messageDto.getMessage().equals("Time pattern has been successfully changed")) {
            return ResponseEntity.status(HttpStatus.OK).body(messageDto);
        }
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(messageDto);
    }
```

Варианты статусов:

- 200 (OK) – статус успешно изменен

- 400 (BAD_REQUEST) – невалидное значение передано

- 400 (BAD_REQUEST) – невалидный метод запроса (например, если вместо PATCH был отправлен PUT запрос)

2. events/all (2 приложение)

```
@PostMapping("all")
public ResponseEntity getAll(@RequestBody EventDto eventDto) throws IOException {
    List<EventDto> events = eventsService.getAll(eventDto.getPageNumber());
    if(!events.isEmpty()) {
        return ResponseEntity.status(HttpStatus.OK).body(eventsService.getAll(eventDto.getPageNumber()));
    }
    return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Not found events for today");
}
```

Варианты статусов:

- 200 (OK) – успешно, получен список событий на сегодняшний день

- 404 (NOT_FOUND) – события на сегодняшний день не найдены

- 400 (BAD_REQUEST) – невалидное тело запроса

- 400 (BAD_REQUEST) - невалидный метод запроса

3. appointments/add (1 приложение)

```
@PostMapping("add")
public ResponseEntity<MessageDto> addAppointment(@RequestBody AppointmentDto appointmentDto) {
    MessageDto messageDto = appointmentsService.addAppointment(appointmentDto);
    if(messageDto.getMessage().equals("Add new appointment")) {
        return ResponseEntity.status(HttpStatus.CREATED).body(messageDto);
    }
    return ResponseEntity.status(HttpStatus.NOT_FOUND).body(messageDto);
}
```

- 201 (CREATED) – назначение успешно создано

- 404 (NOT_FOUND) – не найдено поле, введенное в запросе (например, пациент не найден)

- 400 (BAD_REQUEST) – невалидное тело запроса

- 400 (BAD_REQUEST) - невалидный метод запроса