ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Департамент программной инженерии

СОГЛАСОВАНО

Приглашенный преподаватель
департамента программной инженерии
факультета компьютерных наук

_____ А. Н. Степанов

«3» *мая* 2022 г.

УТВЕРЖДАЮ

Академический руководитель
образовательной программы
«Программная инженерия», канд. техн. наук

_____ В. В. Шилов

«12» мая 2022 г.

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

СЕРВЕРНАЯ ЧАСТЬ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПОМОЩИ В ЗАПОМИНАНИИ ТЕОРИТИЧЕСКОЙ ЧАСТИ ДИСЦИПЛИН

Программа и методика испытаний

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.05.10-01 ПМИ 01-1-ЛУ

Исполнитель:

_____ / И. Н. Дедов /

«3» *мая* 2022 г.

Москва 2022

УТВЕРЖДЕН

RU.17701729.05.10-01 ПМИ 01-1-ЛУ

СЕРВЕРНАЯ ЧАСТЬ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПОМОЩИ В ЗАПОМИНАНИИ ТЕОРИТИЧЕСКОЙ ЧАСТИ ДИСЦИПЛИН

Программа и методика испытаний

RU.17701729.05.10-01 ПМИ 01-1

Листов 78

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

Москва 2022

АННОТАЦИЯ

Программа и методика испытаний — это документ, в котором содержится информация о программном продукте, а также полное описание приемочных испытаний для данного программного продукта.

Настоящая Программа и методика испытаний для «Серверной части мобильного приложения для помощи в запоминании теоретической части дисциплин» содержит следующие разделы: «Объект испытаний», «Цель испытаний», «Требования к программе», «Требования к программной документации», «Средства и порядок испытаний», «Методы испытаний», «Приложения».

В разделе «Объект испытаний» указано наименование, краткая характеристика и назначение программы.

В разделе «Цель испытаний» указана цель проведения испытаний.

Раздел «Требования к программе» содержит основные требования к программе, которые подлежат проверке во время испытаний (требования к функционалу и интерфейсу).

Раздел «Требования к программным документам» содержит состав программной документации, которая представляется на испытания.

Раздел «Средства и порядок испытаний» содержит информацию о технических и программных средствах, которые следует использовать во время испытаний, а также порядок этих испытаний.

Раздел «Методы испытаний» содержит информацию об используемых методах испытаний.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1];
2) ГОСТ 19.102-77 Стадии разработки [2];
3) ГОСТ 19.103-77 Обозначения программ и программных документов [3];
4) ГОСТ 19.104-78 Основные надписи [4];
5) ГОСТ 19.105-78 Общие требования к программным документам [5];

6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];
7) ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению [10].

Изменения к данному документу оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

СОДЕРЖАНИЕ

ГЛОССАРИЙ 6

1. ОБЪЕКТ ИСПЫТАНИЙ 7

1.1. Наименование программы 7

1.2. Краткая характеристика области применения 7

2. ЦЕЛЬ ИСПЫТАНИЙ 8

2.1. Документы, на основании которых ведется разработка 8

3. ТРЕБОВАНИЯ К ПРОГРАММЕ 9

3.1. Требования к функциональным характеристикам 9

3.1.1. Состав выполняемых функций 9

3.2. Требования к входным и выходным данным 21

3.2.1. Организация входных данных 21

3.2.2. Организация выходных данных 22

3.3. Требования к интерфейсу 22

3.4. Требования к надежности 22

4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ 23

4.1. Состав программной документации 23

4.2. Специальные требования к программной документации 23

5. СРЕДСТВА И ПОРЯДОК ИСПЫТАНИЙ 24

5.1. Технические средства, используемые во время испытаний 24

5.2. Программные средства, используемые во время испытаний 24

5.3. Порядок проведения испытаний 24

5.4. Требования к персоналу 24

6. МЕТОДЫ ИСПЫТАНИЙ 25

6.1. Проверка требований к технической документации 25

6.2. Проверка требований к функциональным характеристикам 25
6.2.1. Регистрация 25
6.2.2. Аутентификация 27
6.2.3. Команды 28
6.2.4. Дисциплины 32
6.2.5. Темы 33
6.2.6. Вопросы 34
6.2.7. Тестирование 39
6.2.8. Статистика 42
6.2.9. Настройки аккаунта пользователя 43
6.2.10. Управление командой 45
6.2.11. Управление предметами 59
6.2.12. Управление темами 64
6.2.13. Управление вопросами 69
6.2.14. Вывод 74
6.3. Проверка входных и выходных данных 74
6.4. Проверка требований к надежности 74
СПИСОК ИСТОЧНИКОВ 76
ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 78

ГЛОССАРИЙ

1) *Пользователь* – человек, который использует приложение «Stoady». В контексте приложения «Stoady» существует два типа пользователей: «студенты» и «администраторы».
2) *Студент* (также, *участник, участник команды*) – пользователь, цель которого – лучше разобраться в теоретическом материале дисциплин.
3) *Администратор* – пользователь, имеющий доступ к редактированию материалов, содержащихся в приложении «Stoady», а именно:
   - добавление, удаление и редактирование прав участников команд;
   - добавление, удаление и редактирование предметов, тем и вопросов, содержащихся в команде.
4) *Команда* (также, *группа*) – группа людей, объединенных общим набором предметов.
5) *Предмет* (также, *дисциплина*) – верхний уровень иерархии теоретических материалов в приложении, который может содержать внутри себя одну или несколько тем.
6) *Тема* – раздел предмета, содержащий необходимую для изучения теорию и соответствующие ей вопросы.
7) *Вопрос* – элемент нижнего уровня иерархии теории в приложении. Вопрос содержит в себе сам текст вопроса, а также ответ на него.
8) *СУБД* – система управления базами данных.
9) *API* (Application Programming Interface) – описание способов (набор классов, методов и т. п.), которыми одна компьютерная программа (в данном случае, клиентская часть приложения) может взаимодействовать с другой (в данном случае, с сервером).
10) *JSON* (JavaScript Object Notation) – текстовый формат обмена данными, основанный на языке программирования JavaScript. Но при этом формат независим от JavaScript и может использоваться в любом языке программирования.

1. ОБЪЕКТ ИСПЫТАНИЙ

1.1. Наименование программы

Наименование программы – «Серверная часть мобильного приложения для помощи в запоминании теоретической части дисциплин».

Наименование программы на английском языке – «Server Side of the Mobile Application for Assisting in the Memorization of Theoretical Parts of Disciplines».

1.2. Краткая характеристика области применения

Программа будет использоваться клиентской частью приложения «Stoady». Ее цель – производить необходимые расчеты и обеспечивать взаимодействие между конечным пользователем приложения и базой данных.

2. ЦЕЛЬ ИСПЫТАНИЙ

2.1. Документы, на основании которых ведется разработка

Целью испытаний является проверка корректности выполнения программой функций, изложенных в п. 4 «Требования к программе» настоящего Технического задания из комплекта документации в соответствии с ЕСПД (Единой системой программной документации).

3. ТРЕБОВАНИЯ К ПРОГРАММЕ

3.1. Требования к функциональным характеристикам

3.1.1. Состав выполняемых функций

Серверная часть приложения «Stoady» должна предоставлять API, обеспечивающий выполнений следующих функций:

Группа требований	Формулировка требования	Требования к конечной точке API
Аутентификация	Аутентификация пользователя по почте и паролю	**Метод запроса:** POST **Параметры:** Нет **Тело запроса:** • *email* – почта пользователя • *password* – пароль пользователя **Возвращаемые данные:** • **Status code:** 200 • **Response body:** информация о пользователе в формате JSON: его пользователя, имя пользователя и идентификатор его аватара
Регистрация	Регистрация нового пользователя	**Метод запроса:** POST **Параметры:** Нет **Тело запроса:** • *name* – имя пользователя • *email* – почта пользователя • *password* – пароль пользователя

		• *avatarId* – идентификатор аватара пользователя **Возвращаемые данные:** • **Status code:** 200 • **Response body:** информация о созданном пользователе в формате JSON: идентификатор созданного пользователя
Команды	Получить список команд, в которых состоит пользователь	**Метод запроса:** GET **Параметры:** • *userId* – идентификатор пользователя **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** информация о найденных командах в формате JSON: идентификатор команды, роль пользователя в этой команде, название команды и ее аватар
	Предоставить информацию о команде	**Метод запроса:** GET **Параметры:** • *teamId* – идентификатор выбранной команды **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200

		• **Response body:** информация о выбранной команде в формате JSON: название команды, аватар команды, а также информация о дисциплинах внутри этой команды (идентификаторы, названия и описание)
	Создание новой команды пользователем	**Метод запроса:** POST **Параметры:** Нет **Тело запроса:** • *userId* – идентификатор пользователя, создавшего команду • *teamName* – имя новой команды **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
Дисциплины	Предоставить информацию о выбранном пользователем предмете	**Метод запроса:** GET **Параметры:** • *subjectId* – идентификатор предмета, выбранного пользователем **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** информация о выбранном предмете в формате JSON: идентификатор предмета, его описание, а также

		идентификаторы и названия тем, входящие в этот предмет
Темы	Предоставить информацию о выбранной теме	**Метод запроса:** GET **Параметры:** • *topicId* – идентификатор темы, выбранной пользователем **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** информация о выбранной пользователем теме в формате JSON: название темы, ее описание и список вопросов, входящих в тему
Вопросы	Получить вопросы, содержащиеся в выбранной теме	**Метод запроса:** GET **Параметры:** • *topicId* – идентификатор темы, вопросы которой необходимо получить **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** информация о вопросах, содержащихся в теме в формате JSON: их идентификаторы, а также текст вопроса и ответа

	Добавить вопрос в сохраненные	**Метод запроса:** PUT **Параметры:** • *userId* – идентификатор пользователя, которому нужно добавить вопрос в сохраненные • *questionId* – идентификатор вопроса, который необходимо добавить в сохраненные **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
	Удалить вопрос из сохраненных	**Метод запроса:** DELETE **Параметры:** • *userId* – идентификатор пользователя, вопрос у которого нужно удалить • *questionId* – идентификатор вопроса, который необходимо удалить из сохраненных **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 **Response body:** нет
	Получение сохраненных	**Метод запроса:** GET **Параметры:**

	вопросов пользователя	• *userId* – идентификатор пользователя, сохраненные вопросы которого необходимо получить **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** информация о сохраненных вопросах пользователя в формате JSON: их идентификаторы, а также текст вопроса и ответа
Тестирование	Сохранение результатов тестирования по теме	**Метод запроса:** POST **Параметры:** Нет **Тело запроса:** • *userId* – идентификатор пользователя, который проходил тестирование • *topicId* – идентификатор темы, тест по которой был пройден • *result* – процент выполнения теста **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
Статистика	Получение статистики прохождения	**Метод запроса:** GET **Параметры:**

	тестов по темам пользователя	• *userId* – идентификатор пользователя, статистику которого необходимо получить **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** информация о темах и результате выполнения тестирования по этим темам в формате JSON
Настройки аккаунта пользователя	Обновить аватар пользователя	**Метод запроса:** PUT **Параметры:** • *userId* – идентификатор пользователя • *avatarId* – идентификатор аватара, который необходимо установить пользователю **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
Управление командой	Получение списка участников команды	**Метод запроса:** GET **Параметры:** • *teamId* – идентификатор команды, участников которой необходимо получить **Тело запроса:** Нет

		Возвращаемые данные: • **Status code:** 200 • **Response body:** информация об участниках команды в формате JSON: их идентификаторы, имена, адреса электронных почт и роли в команде
	Изменение роли участника команды	**Метод запроса:** PUT **Параметры:** • *teamId* – идентификатор команды, в которой нужно изменить роль участника • *userId* – идентификатор участника, роль которого нужно изменить • *userRole* – новая роль участника • *executorId* – идентификатор пользователя, выполняющего действие **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
	Добавление участника в команду	**Метод запроса:** POST **Параметры:** • *teamId* – идентификатор команды, в которую необходимо добавить участника • *email* – адрес электронной почты участника, которого нужно добавить в команду

		• *executorId* – идентификатор пользователя, выполняющего действие **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
	Удаление участника из команды	**Метод запроса:** DELETE **Параметры:** • *teamId* – идентификатор команды, из которой необходимо удалить участника • *userId* – идентификатор участника, которого нужно удалить из команды • *executorId* – идентификатор пользователя, выполняющего действие **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
	Редактирование команды	**Метод запроса:** POST **Параметры:** • *teamId* – идентификатор команды, из которой необходимо удалить участника • *executorId* – идентификатор пользователя, выполняющего действие

		Тело запроса: • *teamName* – строка, новое название команды • *teamAvatar* – строка, ссылка на новый аватар команды **Возвращаемые данные:** • **Status code:** 200 **Response body:** нет
Управление предметами	Создание нового предмета	**Метод запроса:** POST **Параметры:** Нет **Тело запроса:** • *teamId* – идентификатор команды, в которую необходимо добавить предмет • *subjectName* – название нового предмета • *subjectDescription* – описание предмета **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
	Редактирование предмета	**Метод запроса:** POST **Параметры:** • *subjectId* – идентификатор редактируемого предмета **Тело запроса:** • *subjectName* – новое название предмета • *subjectDescription* – новое описание предмета

		Возвращаемые данные: • **Status code:** 200 • **Response body:** нет
	Удаление предмета	**Метод запроса:** DELETE **Параметры:** • *subjectId* – идентификатор предмета, который необходимо удалить **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
Управление темами	Создание новой темы	**Метод запроса:** POST **Параметры:** Нет **Тело запроса:** • *subjectId* – идентификатор предмета, в который необходимо добавить тему • *topicName* – название новой темы • *topicDescription* – описание темы **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
	Редактирование темы	**Метод запроса:** POST **Параметры:**

		• *topicId* – идентификатор редактируемой темы **Тело запроса:** • *topicName* – новое название темы • *topicDescription* – новое описание темы **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
	Удаление темы	**Метод запроса:** DELETE **Параметры:** • *topicId* – идентификатор темы, которую необходимо удалить **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
Управление вопросами	Создание нового вопроса	**Метод запроса:** POST **Параметры:** Нет **Тело запроса:** • *topicId* – идентификатор темы, в которую необходимо добавить вопрос • *questionText* – текст нового вопроса • *answerText* – текст ответа на новый вопрос **Возвращаемые данные:**

		• **Status code:** 200
	Редактирование вопроса	**Метод запроса:** POST **Параметры:** • *questionId* – идентификатор редактируемого вопроса **Тело запроса:** • *questionText* – новый текст вопроса • *answerText* – новый текст ответа на вопрос **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет
	Удаление вопроса	**Метод запроса:** DELETE **Параметры:** • *questionId* – идентификатор вопроса, который нужно удалить **Тело запроса:** Нет **Возвращаемые данные:** • **Status code:** 200 • **Response body:** нет

3.2. Требования к входным и выходным данным

3.2.1. Организация входных данных

Входными данными приложения являются HTTP-запросы от клиентской части приложения, которые могут содержать в себе тело в формате JSON.

3.2.2. Организация выходных данных

Выходными данными приложения являются HTTP-ответы на запросы от клиентской части мобильного приложения, содержащие код ответа, заголовки, а также тело ответа в формате JSON.

В число выходных данных также входят логи работы приложения.

3.3. Требования к интерфейсу

Серверная часть мобильного приложения «Stoady» не предусматривает наличия интерфейса.

3.4. Требования к надежности

Система должна корректно обрабатывать неверные запросы любого вида и выдавать информативные сообщения об ошибках:

1) В случае возникновения ошибки на стороне сервера, API должен отправить ответ с кодом 500 Internal Server Error.
2) В случае возникновения ошибки по вине пользователя, API должен отправить ответ с одним из кодов: 400 Bad Request, 401 Unauthorized, 404 Not Found или 405 Method Not Allowed.

У каждой возвращаемой ошибки должно быть как минимум: сообщение с текстом и трейс стека вызовов.

4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

4.1. Состав программной документации

1) «Серверная часть мобильного приложения для помощи в запоминании теоретической части дисциплин». Техническое задание (ГОСТ 19.201-78) [7];
2) «Серверная часть мобильного приложения для помощи в запоминании теоретической части дисциплин». Программа и методика испытаний (ГОСТ 19.301-78) [10];
3) «Серверная часть мобильного приложения для помощи в запоминании теоретической части дисциплин». Пояснительная записка (ГОСТ 19.404-79) [11];
4) «Серверная часть мобильного приложения для помощи в запоминании теоретической части дисциплин». Руководство оператора (ГОСТ 19.505-79) [12];
5) «Серверная часть мобильного приложения для помощи в запоминании теоретической части дисциплин». Текст программы (ГОСТ 19.401-78) [13].

4.2. Специальные требования к программной документации

1) Документы к программе должны быть выполнены в соответствии с ГОСТ 19.106-78 и ГОСТами к каждому виду документа (см. пункт 5.1.).
2) Пояснительная записка должна быть загружена в систему Антиплагиат через LMS «НИУ ВШЭ». Лист, подтверждающий загрузку пояснительной записки, сдается в учебный офис вместе со всеми материалами не позже, чем за день до защиты курсовой работы.
3) Вся документация также воспроизводится в печатном виде, она должна быть подписана академическим руководителем образовательной программы 09.03.04 «Программная инженерия», руководителем разработки и исполнителями перед сдачей курсовой работы в учебный офис, не позже одного дня до защиты.
4) Документация также сдается в электронном виде в формате .pdf или .docx, а программа – в архиве формата .zip или .rar.
5) Все документы перед защитой курсовой работы должны быть загружены в информационно-образовательную среду НИУ ВШЭ LMS (Learning Management System) в личном кабинете, дисциплина – «Курсовой проект, 2 курс ПИ», одним архивом.

5. СРЕДСТВА И ПОРЯДОК ИСПЫТАНИЙ

5.1. Технические средства, используемые во время испытаний

Во время испытаний использовался персональный компьютер со следующими характеристиками:

1) операционная система Windows 10;
2) процессор Intel® Core™ i5-8265U с 4 ядрами;
3) 16 Гб оперативной памяти;
4) 1 Тб памяти, 501 Гб свободной памяти;
5) видеокарта NVIDIA GeForce GTX 1050 with Max-Q Design.

5.2. Программные средства, используемые во время испытаний

Во время испытаний использовались следующие программные средства:

1) IDE JetBrains Rider версии 2021.1.2;
2) JetBrains DataGrip версии 2021.1.1;
3) Microsoft .NET Framework 6.0.101;
4) Postman API версии 9.17.1.

5.3. Порядок проведения испытаний

Испытания должны проводиться в следующем порядке:

1) скачать и установить IDE JetBrains Rider версии 2021.1.1 или выше;
2) установить .NET версии 6.0;
3) скачать код приложения, открыть его в IDE JetBrains Rider и локально запустить API;
4) провести испытания, описанные в разделе «Методика испытаний»;
5) выйти из программы, закрыть IDE.

5.4. Требования к персоналу

Для проверки корректности программы достаточно одного человека.

Проверяющий должен знать язык программирования C# (версии 9.0 или выше), а также иметь опыт работы с фреймворком ASP .NET Core [14], иметь знания в области реляционных баз данных, а также опыт работы с БД PostgreSQL (версии 13 или выше). Проверяющий должен иметь представление о клиент–серверном взаимодействии.

6. МЕТОДЫ ИСПЫТАНИЙ

6.1. Проверка требований к технической документации

Состав программной документации проверяется наличием полного комплекта документов программной документации в системе SmartLMS и наличием всех требуемых подписей. Также проверяется соответствие документации требованиям ГОСТ.

Комплект документов полный. Все документы удовлетворяют представленным требованиям.

6.2. Проверка требований к функциональным характеристикам

6.2.1. Регистрация

- **Регистрация нового пользователя**

Кейс 1. Пользователя не существует.

С помощью Postman отправляем запрос в конечную точку `/users/register`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/users/register' \
-H 'Content-Type: application/json' \
-d '{
    "username": "test_user",
    "email": "test_user@gmail.com",
    "password": "12345",
    "avatarId": 0
}'
```

Результат получаем ответ 200 OK и JSON-объект с информацией о пользователе:

```
{
    "userId": 28
}
```

С помощью запроса в базу данных проверяем, что новый пользователь создался:

```
SELECT * FROM Users
WHERE id = 28;
```

Результат:

id	28
username	test_user
email	test_user@gmail.com

password	bO+xORA9BdWwDW4GctKu5bFbs0DVkll0+5iRjARHANs=
avatarid	0
salt	aCEpk+aBeDbdnOc7QLGH4vRTpEPyLeS3C/SH3Ci7HsI=

Кейс 2. Пользователь уже существует.

Предусловие: в первом кейсе был зарегистрирован пользователь `test_user`.

С помощью Postman отправляем запрос в конечную точку `/users/register`, подав на вход адрес электронной почты `test_user@gmail.com`, использованный в предыдущем кейсе:

```
curl -X 'POST' 'https://stoady.herokuapp.com/users/register' \
-H 'Content-Type: application/json' \
-d '{
    "username": "another_test_user",
    "email": "test_user@gmail.com",
    "password": "123457890",
    "avatarId": 12
}'
```

Результат: получаем ответ 400 Bad Request и объект с текстом ошибки «User already exists».


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "User already exists",
    "stackTrace": "   at Stoady.Handlers.User.RegisterUser.RegisterUserCommandHandler.Handle
        (RegisterUserCommand request, CancellationToken ct) in /src/Stoady/Handlers/User/
        RegisterUser/RegisterUserCommandHandler.cs:line 48\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 62\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR
```


Кейс 3. Некорректный запрос.

С помощью Postman отправляем запрос в конечную точку `/users/register`, подав на вход слишком короткий пароль и неверную строку в качестве адреса электронной почты:

```
curl -X 'POST' 'https://stoady.herokuapp.com/users/register' \
-H 'Content-Type: application/json' \
-d '{
    "username": "another_test_user",
    "email": "this_is_not_an_email",
    "password": "hi",
```

```
    "avatarId": 11
}'
```

Результат: получаем ответ 400 Bad Request и объект с ошибкой валидации.


```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nEmail: 'Email' is not a valid email address.
        \r\nPassword.Length: 'Password Length' must be greater than or equal to '5'.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request,
        CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 56\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.
        Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
```


6.2.2. Аутентификация

- **Аутентификация пользователя по почте и паролю**

Кейс 1. Пользователь существует.

Предусловие: зарегистрированный пользователь – в предыдущем пункте 6.2.1 был создан пользователь `test_user`.

С помощью Postman отправляем запрос в конечную точку `/auth/authorize`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/auth/authorize' \
-H 'Content-Type: application/json' \
-d '{
    "email": "test_user@gmail.com",
    "password": "12345"
}'
```

Результат: получаем ответ 200 OK и объект со следующим содержанием:

```
{
    "id": 28,
    "name": "test_user",
    "avatarId": 0
}
```

Кейс 2. Пользователя не существует.

С помощью Postman отправляем запрос в конечную точку `/auth/authorize`, подав на вход адрес электронной почты несуществующего пользователя:

```
curl -X 'POST' 'https://stoady.herokuapp.com/auth/authorize' \
-H 'Content-Type: application/json' \
-d '{
    "email": "non_existent_user@gmail.com",
    "password": "12345"
}'
```

Результат: получаем ответ 401 Unauthorized.

Кейс 3. Некорректный пароль.

С помощью Postman отправляем запрос в конечную точку `/auth/authorize`, подав на вход адрес электронной почты существующего пользователя, но с неправильным паролем:

```
curl -X 'POST' 'https://stoady.herokuapp.com/auth/authorize' \
-H 'Content-Type: application/json' \
-d '{
    "email": "test_user@gmail.com",
    "password": "wrong_password"
}'
```

Результат: получаем ответ 401 Unauthorized.

6.2.3. Команды

- **Получение списка команд, в которых состоит пользователь**

Кейс 1. Пользователь состоит в командах.

Предусловие: зарегистрированный пользователь, состоящий в командах – возьмем существующего пользователя «`Ivan`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/users/teams`, передав в параметр идентификатор пользователя:

```
curl -X 'GET' 'https://stoady.herokuapp.com/users/teams?userId=1'
```

Результат: получили ответ 200 OK и информацию о командах пользователя:

```
{
    "teams": [
        {
            "teamId": 3,
            "role": "Member",
            "teamName": "myOtherTeam",
            "teamAvatar": "https://ie.wampi.ru/2022/03/20/lake.png"
        },
        {
            "teamId": 4,
            "role": "Member",
            "teamName": "HSE Android",
            "teamAvatar": "https://s.zefirka.net/images/2017-05-
17/slonyata-milye-malenkie-giganty/slonyata-milye-malenkie-giganty-
9.jpg"
        }
    ]
}
```

Кейс 2. Пользователь не состоит в командах.

Предусловие: зарегистрированный пользователь, не состоящий в командах – возьмем пользователя `test_user` с идентификатором «28».

С помощью Postman отправляем запрос в конечную точку `/users/teams`, передав в параметр идентификатор пользователя:

```
curl -X 'GET' 'https://stoady.herokuapp.com/users/teams?userId=28'
```

Результат: получили ответ 200 OK и пустой список команд:

```
{
    "teams": []
}
```

Кейс 3. Пользователь не существует.

С помощью Postman отправляем запрос в конечную точку `/users/teams`, передав в параметр идентификатор несуществующего пользователя:

```
curl -X 'GET' 'https://stoady.herokuapp.com/users/teams?userId=1000'
```

Результат: получили ответ 200 OK и пустой список команд:

```
{
    "teams": []
}
```

- **Предоставление информации о команде**

Кейс 1. Команда существует.

Предусловие: созданная команда – возьмем команду «`HSE Android`» с идентификатором «4».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}`, взяв `teamId = 4`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/teams/4'
```

Результат: получили ответ 200 OK и корректный ответ:

```
{
    "name": "HSE Android",
    "picture": "https://s.zefirka.net/images/2017-05-17/slonyata-
milye-malenkie-giganty/slonyata-milye-malenkie-giganty-9.jpg",
    "subjects": [
        {
            "id": 1,
            "name": "Algebra",
            "description": "\nThis is about algebra."
        },
        {
            "id": 2,
            "name": "History",
            "description": "hello?"
        },
        {
            "id": 4,
            "name": "Music",
            "description": "This is about musiiiic!"
        }
```

```
    ]
}
```

Кейс 2. Команда не существует.

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}`, взяв `teamId = 1000`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/teams/1000'
```

Результат: получили ответ 400 Bad Request и объект, содержащий информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find team with ID = 1000",
    "stackTrace": "   at Stoady.Handlers.Team.GetTeamInfo.GetTeamInfoCommandHandler.Handle
        (GetTeamInfoCommand request, CancellationToken ct) in /src/Stoady/Handlers/Team/
        GetTeamInfo/GetTeamInfoCommandHandler.cs:line 54\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 31\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


- **Создание новой команды**

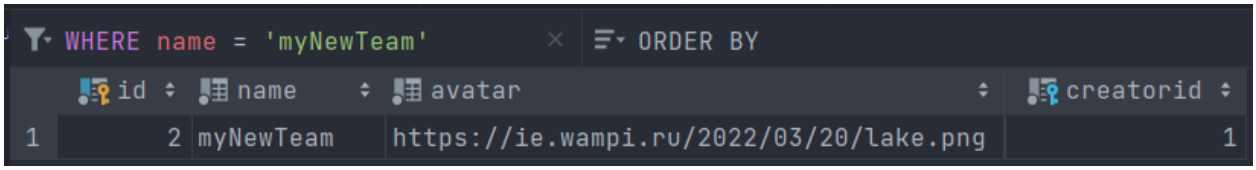
Кейс 1. Пользователь, создающий команду, существует.

Предусловие: существующий пользователь – возьмем пользователя «`Ivan`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/teams/create`:

```
curl -X
'POST' 'https://stoady.herokuapp.com/teams/create?userId=1&teamName=my
NewTeam'
```

Результат: получили ответ 200 OK. В базе появилась новая команда «`myNewTeam`».



Кейс 2. Не существует пользователя, создающего команду.

С помощью Postman отправляем запрос в конечную точку `/teams/create`, взяв `userId = 1000`:

```
curl -X
'POST' 'https://stoady.herokuapp.com/teams/create?userId=1000&teamName
=myNewTeam'
```

Результат: получили ответ 400 Bad Request и объект, содержащий информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find user with ID = 1000",
    "stackTrace": "   at Stoady.Handlers.Team.CreateTeam.CreateTeamCommandHandler.Handle
        (CreateTeamCommand request, CancellationToken ct) in /src/Stoady/Handlers/Team/CreateTeam/
        CreateTeamCommandHandler.cs:line 49\n   at Stoady.Helpers.ValidationBehavior`2.Handle
        (TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/
        Helpers/ValidationBehavior.cs:line 62\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


6.2.4. Дисциплины

- **Предоставление информации о выбранном предмете**

Кейс 1. Предмет существует.

Предусловие: существующая дисциплина – возьмем предмет «`Algebra`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/subjects/{subjectId}`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/subjects/1'
```

Результат: получили ответ 200 OK и объект с корректной информацией о предмете:


```
{
    "name": "Algebra",
    "description": "\nThis is about algebra.",
    "topics": [
        {
            "id": 1,
```

```
            "name": "Matrices"
        },
        {
            "id": 2,
            "name": "Vectors"
        }
    ]
}
```

Кейс 2. Предмета не существует.

С помощью Postman отправляем запрос в конечную точку `/subjects/{subjectId}`, взяв `subjectId = 1000`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/subjects/1000'
```

Результат: получили ответ 400 Bad Request и объект, содержащий верную информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find subject with ID = 1000",
    "stackTrace": "   at Stoady.Handlers.Subject.GetSubjectInfo.GetSubjectInfoCommandHandler.Handle
        (GetSubjectInfoCommand request, CancellationToken ct) in /src/Stoady/Handlers/Subject/
        GetSubjectInfo/GetSubjectInfoCommandHandler.cs:line 54\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


6.2.5. Темы

- **Предоставление информации о выбранной теме**

Кейс 1. Тема существует.

Предусловие: существующая тема – возьмем тему «`Matrices`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/topics/{topicId}`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/topics/1'
```

Результат: получили ответ 200 OK и объект с корректной информацией о содержимом темы.

```
{
    "name": "Matrices",
    "description": "\nIn mathematics, a matrix is a rectangular array
or table of numbers, symbols, or expressions, arranged in rows and col
umns, which is used to represent a mathematical object or a property o
f such an object.",
    "questions": [
        {
            "id": 2
        }
    ]
}
```

Кейс 2. Темы не существует.

С помощью Postman отправляем запрос в конечную точку `/topics/{topicId}`, взяв `topicId = 1000`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/subjects/1000'
```

Результат: получили ответ 400 Bad Request и объект, содержащий верную информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find topic with ID = 1000",
    "stackTrace": "   at Stoady.Handlers.Topic.GetTopicInfo.GetTopicInfoCommandHandler.Handle
       (GetTopicInfoCommand request, CancellationToken ct) in /src/Stoady/Handlers/Topic/
       GetTopicInfo/GetTopicInfoCommandHandler.cs:line 54\n   at Stoady.Helpers.
       ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
       RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
       MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
       CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
       Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
       cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


6.2.6. Вопросы

- **Получение вопросов темы**

Кейс 1. Тема существует.

Предусловие: существующая тема – возьмем тему «`Matrices`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/questions/{topicId}`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/questions/1'
```

Результат: получили ответ 200 OK и объект с корректной информацией о вопросах темы.

```
{
    "questions": [
        {
            "id": 2,
            "questionText": "How do you transpose a matrix?",
            "answerText": "Flip it around its diagonal"
        }
    ]
}
```

Кейс 2. Темы не существует.

С помощью Postman отправляем запрос в конечную точку `/questions/{topicId}`, взяв `topicId = 1000`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/questions/1000'
```

Результат: получили ответ 200 OK и пустой список вопросов темы:

```
{
    "questions": []
}
```

- **Добавление вопроса в сохраненные**

Кейс 1. Вопрос и пользователь существуют.

Предусловие: существующий вопрос – возьмем вопрос про матрицы с идентификатором «2», – и пользователь – возьмем пользователя «`Ivan`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/questions/save/{questionId}`:

```
curl -X 'PUT' 'https://stoady.herokuapp.com/questions/save/2' \
-H 'userId: 1'
```

Результат: получили ответ 200 OK.

Кейс 2. Пользователя не существует.

Предусловие: существующий вопрос – возьмем вопрос про матрицы с идентификатором «2».

С помощью Postman отправляем запрос в конечную точку `/questions/save/{questionId}`, подав в качестве параметра `userId` число 1000:

```
curl -X 'PUT' 'https://stoady.herokuapp.com/questions/save/2' \
-H 'userId: 1000'
```

Результат: получаем ответ 400 Bad Request и информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when saving this question: 23503: insert or update on table
        \"userquestions\" violates foreign key constraint \"userquestions_userid_fkey\"\n\nDETAIL:
        Detail redacted as it may contain sensitive data. Specify 'Include Error Detail' in the
        connection string to include this information.",
    "stackTrace": "   at Stoady.Handlers.Question.SaveQuestion.SaveQuestionCommandHandler.Handle
        (SaveQuestionCommand request, CancellationToken ct) in /src/Stoady/Handlers/Question/
        SaveQuestion/SaveQuestionCommandHandler.cs:line 60\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 31\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
```


Кейс 3. Вопроса не существует.

Предусловие: существующий пользователь – возьмем пользователя «`Ivan`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/questions/save/{questionId}`, подав в качестве параметра `questionId` число 1000:

```
curl -X 'PUT' 'https://stoady.herokuapp.com/questions/save/1000' \
-H 'userId: 1'
```

Результат: получаем ответ 400 Bad Request и информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when saving this question: 23503: insert or update on table
        \"userquestions\" violates foreign key constraint
        \"userquestions_questionid_fkey\"\n\nDETAIL: Detail redacted as it may contain sensitive
        data. Specify 'Include Error Detail' in the connection string to include this information.
        ",
    "stackTrace": "   at Stoady.Handlers.Question.SaveQuestion.SaveQuestionCommandHandler.Handle
        (SaveQuestionCommand request, CancellationToken ct) in /src/Stoady/Handlers/Question/
        SaveQuestion/SaveQuestionCommandHandler.cs:line 60\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 31\n   at
```

- **Удаление вопроса из сохраненных**

Кейс 1. Вопрос и пользователь существуют.

Предусловие: существующий вопрос – возьмем вопрос про матрицы с идентификатором «2», – и пользователь – возьмем пользователя «`Ivan`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/questions/unsave/{questionId}`:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/questions/unsave/2' \
-H 'userId: 1'
```

Результат: получили ответ 200 OK.

Кейс 2. Пользователя не существует.

Предусловие: существующий вопрос – возьмем вопрос про матрицы с идентификатором «2».

С помощью Postman отправляем запрос в конечную точку `/questions/unsave/{questionId}`, подав в качестве параметра `userId` число 1000:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/questions/unsave/2' \
-H 'userId: 1000'
```

Результат: получаем ответ 400 Bad Request и информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when unsaving this question. Please, try again.",
    "stackTrace": "   at Stoady.Handlers.Question.RemoveSavedQuestion.
        RemoveSavedQuestionCommandHandler.Handle(RemoveSavedQuestionCommand request,
        CancellationToken ct) in /src/Stoady/Handlers/Question/RemoveSavedQuestion/
        RemoveSavedQuestionCommandHandler.cs:line 51\n   at Stoady.Helpers.ValidationBehavior`2.
        Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/
        Stoady/Helpers/ValidationBehavior.cs:line 31\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
```


Кейс 3. Вопроса не существует.

Предусловие: существующий пользователь – возьмем пользователя «Ivan» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/questions/unsave/{questionId}`, подав в качестве параметра `questionId` число 1000:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/questions/unsave/1000' \
-H 'userId: 1'
```

Результат: получаем ответ 400 Bad Request и информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when unsaving this question. Please, try again.",
    "stackTrace": "   at Stoady.Handlers.Question.RemoveSavedQuestion.
        RemoveSavedQuestionCommandHandler.Handle(RemoveSavedQuestionCommand request,
        CancellationToken ct) in /src/Stoady/Handlers/Question/RemoveSavedQuestion/
        RemoveSavedQuestionCommandHandler.cs:line 51\n   at Stoady.Helpers.ValidationBehavior`2.
        Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/
        Stoady/Helpers/ValidationBehavior.cs:line 31\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
```


- **Получение списка сохраненных вопросов**

Кейс 1. Пользователь существует.

Предусловие: существующий пользователь – возьмем пользователя «Ivan» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/questions/saved/{userId}`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/questions/saved/1'
```

Результат: получили ответ 200 OK и объект с корректной информацией о сохраненных вопросах пользователя.


```
{
    "savedQuestions": [
        {
            "id": 3,
            "questionText": "When did WWI start?",
```

```
            "answerText": "20th century"
        },
        {
            "id": 5,
            "questionText": "Q2",
            "answerText": "A2"
        },
        {
            "id": 6,
            "questionText": "Q3",
            "answerText": "A3"
        }
    ]
}
```

Кейс 2. Пользователя не существует.

С помощью Postman отправляем запрос в конечную точку `/questions/saved/{userId}`, взяв `userId = 1000`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/questions/saved/1000'
```

Результат: получили ответ 200 OK и пустой список вопросов пользователя.

```
{
    "savedQuestions": []
}
```

6.2.7. Тестирование

- **Сохранение результатов тестирования**

Кейс 1. Пользователь и тема существуют.

Предусловие: существующий пользователь – возьмем пользователя «`Ivan`» с идентификатором «1», – и тема – возьмем тему «`Matrices`» с идентификатором «1».

Отправим с помощью Postman запрос в конечную точку `/tests`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/tests' \
-H 'Content-Type: application/json' \
-d '{
    "userId": 1,
    "topicId": 1,
    "result": 75
```

```
}'
```

Результат: получили ответ 200 OK.

Кейс 2. Пользователь не существует.

Предусловие: существующая тема – возьмем тему «`Matrices`» с идентификатором «1».

Отправим с помощью Postman запрос в конечную точку `/tests`, взяв пользователя с идентификатором «1000», которого не существует:

```
curl -X 'POST' 'https://stoady.herokuapp.com/tests' \
-H 'Content-Type: application/json' \
-d '{
    "userId": 1000,
    "topicId": 1,
    "result": 50
}'
```

Результат: получили ответ 400 Bad Request и ошибку с соответствующей информацией.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Exception occurred when saving statistics for user with ID = 1000 and topic with ID = 1: 23503: insert or update on table \"statistics\" violates foreign key constraint \"statistics_userid_fkey\"\n\nDETAIL: Detail redacted as it may contain sensitive data. Specify 'Include Error Detail' in the connection string to include this information.",
    "stackTrace": "   at Stoady.Handlers.Testing.SaveTestResults.SaveTestResultsCommandHandler.Handle(SaveTestResultsCommand request, CancellationToken ct) in /src/Stoady/Handlers/Testing/SaveTestResults/SaveTestResultsCommandHandler.cs:line 75\n   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 62\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
```


Кейс 3. Тема не существует.

Предусловие: существующий пользователь – возьмем пользователя «`Ivan`» с идентификатором «1».

Отправим с помощью Postman запрос в конечную точку `/tests`, положив в качестве параметра `topicId` число 1000:

```
curl -X 'POST' 'https://stoady.herokuapp.com/tests' \
-H 'Content-Type: application/json' \
```

```
-d '{
    "userId": 1,
    "topicId": 1000,
    "result": 20
}'
```

Результат: аналогично предыдущему кейсу, получаем ответ 400 Bad Request и информацию об ошибке.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Exception occurred when saving statistics for user with ID = 1 and topic with ID = 1000: 23503: insert or update on table \"statistics\" violates foreign key constraint \"statistics_topicid_fkey\"\n\nDETAIL: Detail redacted as it may contain sensitive data. Specify 'Include Error Detail' in the connection string to include this information.",
    "stackTrace": "   at Stoady.Handlers.Testing.SaveTestResults.SaveTestResultsCommandHandler.Handle(SaveTestResultsCommand request, CancellationToken ct) in /src/Stoady/Handlers/Testing/SaveTestResults/SaveTestResultsCommandHandler.cs:line 75\n   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 62\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
```


Кейс 4. Некорректный запрос.

Предусловие: существующий пользователь – возьмем пользователя «`Ivan`» с идентификатором «1», – и тема – возьмем тему «`Matrices`» с идентификатором «1».

Отправим с помощью Postman запрос в конечную точку `/tests`, но в поле «`result`» укажем число 150 (ограничение стоит на целое число от 0 до 100 включительно):

```
curl -X 'POST' 'https://stoady.herokuapp.com/tests' \
-H 'Content-Type: application/json' \
-d '{
    "userId": 1,
    "topicId": 1,
    "result": 150
}'
```

Результат: получаем ошибку валидации и ответ 400 Bad Request.

```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nResult: 'Result' must be less than or equal to
        '100'.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request,
        CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 56\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.
        Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
```


6.2.8. Статистика

- **Получение статистики прохождения тестов по темам**

Кейс 1. Пользователь существует.

Предусловие: существующий пользователь – возьмем пользователя «Ivan» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/stats/{userId}`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/stats/1'
```

Результат: получили ответ 200 OK и объект с корректной информацией о статистике прохождения тестов пользователем.


```
{
    "results": [
        {
            "topicId": 3,
            "topicName": "World War I",
            "result": 50
        },
        {
            "topicId": 7,
            "topicName": "Красивые кошьки",
            "result": 99
        },
        {
            "topicId": 4,
            "topicName": "Classical Russian",
            "result": 66
```

```
        },
        {
            "topicId": 1,
            "topicName": "Matrices",
            "result": 75
        }
    ]
}
```

Кейс 2. Пользователя не существует.

С помощью Postman отправляем запрос в конечную точку `/stats/{userId}`, взяв `userId = 1000`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/stats/1000'
```

Результат: получили ответ 200 OK и пустой список результатов по пройденным темам.

```
{
    "results": []
}
```

6.2.9. Настройки аккаунта пользователя

- **Обновление аватара**

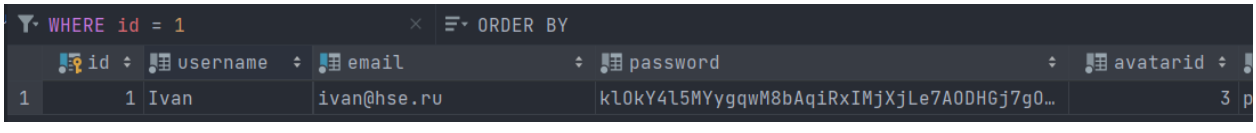
Кейс 1. Пользователь существует.

Предусловие: существующий пользователь – возьмем пользователя «Ivan» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/users/{userId}/avatar/set`:

```
curl -X
'PUT' 'https://stoady.herokuapp.com/users/1/avatar/set?avatarId=3'
```

Результат: получили ответ 200 OK. Аватар пользователя обновлен в базе данных.



Кейс 2. Пользователя не существует.

С помощью Postman отправляем запрос в конечную точку `/users/{userId}/avatar/set`, взяв `userId = 1000`:

```
curl -X
'PUT' 'https://stoady.herokuapp.com/users/1000/avatar/set?avatarId=0'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when changing the avatar. Please, try again.\n (UserId = 1000)
        ",
    "stackTrace": "    at Stoady.Handlers.User.UpdateUserAvatar.UpdateUserAvatarCommandHandler.
        Handle(UpdateUserAvatarCommand request, CancellationToken ct) in /src/Stoady/Handlers/User/
        UpdateUserAvatar/UpdateUserAvatarCommandHandler.cs:line 51\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 62\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
```


Кейс 3. Некорректный идентификатор аватара.

Предусловие: существующий пользователь – возьмем пользователя «`Ivan`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/users/{userId}/avatar/set`, подав в запрос `avatarId = 100` (ограничивается от 0 до 14 включительно):

```
curl -X
'PUT' 'https://stoady.herokuapp.com/users/1/avatar/set?avatarId=100'
```

Результат: получили ошибку валидации с соответствующим текстом.


```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nAvatarId: 'Avatar Id' must be less than or equal
        to '14'.",
    "stackTrace": "    at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request,
        CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 56\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.
        Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
```

6.2.10. Управление командой

- **Получение списка участников команды**

Кейс 1. Команда существует.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members`:

```
curl -X 'GET' 'https://stoady.herokuapp.com/teams/4/members'
```

Результат: получили ответ 200 OK и JSON-объект с членами этой команды.

```
{
    "members": [
        {
            "id": 4,
            "username": "Anna",
            "email": "anna@hse.ru",
            "role": "Creator"
        },
        {
            "id": 1,
            "username": "Ivan",
            "email": "ivan@hse.ru",
            "role": "Member"
        }
    ]
}
```

Кейс 2. Команда не существует.

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members`, подав в качестве параметра `teamId` число 1000:

```
curl -X 'GET' 'https://stoady.herokuapp.com/teams/1000/members'
```

Результат: получили ответ 200 OK и пустой список участников.

```
{
    "members": []
}
```

- **Изменение роли участника команды**

Кейс 1. Команда существует, пользователь состоит в ней, у редактирующего есть права на изменение роли.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь – возьмем пользователя «Ivan» с идентификатором «1», состоящего в команде «HSE Android», редактировать роль будет создатель команды «HSE Android», пользователь «Anna» с идентификатором «4».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members:

```
curl -X
'PUT' 'https://stoady.herokuapp.com/teams/4/members?userId=1&userRole=
Admin' \
-H 'executorId: 4'
```

Результат: получили ответ 200 OK, роль пользователя в базе данных изменилась.

	teamid	userid	roleid	name
1	4	4	1	Creator
2	4	1	2	Admin

Кейс 2. Команда существует, пользователь состоит в ней, у редактирующего нет прав на изменение роли.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь, состоящий в команде – возьмем пользователя «Ivan» с идентификатором «1», редактировать роль будет участник команды «HSE Android», пользователь «Andrey» с идентификатором «5» и ролью «Member».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members:

```
curl -X
'PUT' 'https://stoady.herokuapp.com/teams/4/members?userId=1&userRole=
Admin' \
-H 'executorId: 5'
```

Результат: получаем ответ 400 Bad Request с сообщением, что у пользователя нет прав на редактирование роли.

```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "You do not have permission to change this user's role.",
    "stackTrace": "   at Stoady.Handlers.Team.ChangeMemberStatus.ChangeMemberStatusCommandHandler.Handle(ChangeMemberStatusCommand request,
        CancellationToken ct) in /src/Stoady/Handlers/Team/ChangeMemberStatus/ChangeMemberStatusCommandHandler.cs:line 64\n   at Stoady.
        Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 62\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest
        request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request,
```


Кейс 3. Команда существует, пользователь не состоит в ней, у редактирующего есть права на изменение роли.

Предусловие: существующая команда – возьмем команду «`HSE Android`» с идентификатором «4», существующий пользователь, не состоящий в команде – возьмем пользователя «`Daniel`» с идентификатором «8», редактировать роль будет участник команды «`HSE Android`», пользователь «`Ivan`» с идентификатором «1» и ролью «`Admin`».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members`:

```
curl -X
'PUT' 'https://stoady.herokuapp.com/teams/4/members?userId=8&userRole=
Admin' \
-H 'executorId: 1'
```

Результат: получили ответ 400 Bad Request с сообщением о том, что пользователь с идентификатором «8» не состоит в команде.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find user with ID = 8 in team with ID = 4.",
    "stackTrace": "   at Stoady.Handlers.Team.ChangeMemberStatus.ChangeMemberStatusCommandHandler.Handle(ChangeMemberStatusCommand request,
        CancellationToken ct) in /src/Stoady/Handlers/Team/ChangeMemberStatus/ChangeMemberStatusCommandHandler.cs:line 54\n   at Stoady.
        Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 62\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest
        request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request,
```


Кейс 4. Команда не существует.

Предусловие: существующие пользователи – возьмем пользователей с `userId = 4` и `executorId = 1`).

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members`, взяв в качестве параметра `teamId` число 1000:

```
curl -X
'PUT' 'https://stoady.herokuapp.com/teams/1000/members?userId=4&userRo
le=Admin' \
-H 'executorId: 1'
```

Результат: ответ 400 Bad Request с сообщением о том, что в команде с идентификатором «1000» нет пользователя с идентификатором «4».


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find user with ID = 4 in team with ID = 1000.",
    "stackTrace": "   at Stoady.Handlers.Team.ChangeMemberStatus.ChangeMemberStatusCommandHandler.Handle(ChangeMemberStatusCommand request,
        CancellationToken ct) in /src/Stoady/Handlers/Team/ChangeMemberStatus/ChangeMemberStatusCommandHandler.cs:line 54\n   at Stoady.
        Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 62\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest
        request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request,
```


Кейс 5. Команда существует, пользователь состоит в ней, у редактирующего есть права на изменение роли, роль меняется на «Creator».

Предусловие: существующая команда – возьмем команду «`HSE Android`» с идентификатором «4», существующий пользователь, состоящий в команде – возьмем пользователя «`Ivan`» с идентификатором «1», редактировать роль будет создатель команды «`HSE Android`», пользователь «`Anna`» с идентификатором «4».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members`, передав в качестве параметра `userRole` значение «`Creator`»:

```
curl -X
'PUT' 'https://stoady.herokuapp.com/teams/4/members?userId=1&userRole=
Creator' \
-H 'executorId: 4'
```

Результат: получили ответ 400 Bad Request с ошибкой валидации и сообщением о невозможности изменения роли на «Creator».

```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nRole: Cannot assign Creator role manually.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next)
        in /src/Stoady/Helpers/ValidationBehavior.cs:line 56\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest
        request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n
        at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle
        (TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestPreProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at
```


Кейс 6. Команда существует, пользователь редактирует свою же роль, имея права на редактирование.

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members`, передав в качестве параметров `userId` и `executorId` одно и то же значение «1»:

```
curl -X
'PUT' 'https://stoady.herokuapp.com/teams/4/members?userId=1&userRole=
Creator' \
-H 'executorId: 1'
```

Результат: получили ответ 400 Bad Request с ошибкой валидации на неравенство этих параметров.


```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nUserId: 'UserId' must not be equal to 'ExecutorId'.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next)
        in /src/Stoady/Helpers/ValidationBehavior.cs:line 56\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest
        request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n
        at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle
        (TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestPreProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at
```


- **Добавление участника в команду**

Кейс 1. Команда существует, пользователь существует, у добавляющего есть права на добавление.

Предусловие: существующая команда – возьмем команду «`HSE Android`» с идентификатором «4», существующий пользователь, не состоящий в команде – возьмем пользователя «`Daniel`» с идентификатором «8» и адресом электронной почты

daniel@hse.ru, редактировать роль будет участник команды «HSE Android», пользователь «Ivan» с идентификатором «1» и ролью «Admin».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members/add:

```
curl -X
'POST' 'https://stoady.herokuapp.com/teams/4/members/add?email=daniel%
40hse.ru' \
-H 'executorId: 1'
```

Результат: получаем ответ 200 OK, в команде появился новый пользователь с ролью «Member» (идентификатор «3»).

	id	teamid	userid	roleid
1	6	4	4	1
2	42	4	1	2
3	104	4	5	3
4	105	4	8	3

Кейс 2. Команда существует, пользователь существует, у добавляющего нет прав на добавление.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь, не состоящей в команде – возьмем пользователя «Kermit» с идентификатором «17» и адресом электронной почты kermit_the_frog@mail.ru, редактировать роль будет участник команды «HSE Android», пользователь «Daniel» с идентификатором «8» и ролью «Member».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members/add:

```
curl -X
'POST' 'https://stoady.herokuapp.com/teams/4/members/add?email=Kermit_
the_frog%40mail.ru' \
-H 'executorId: 8'
```

Результат: получаем ответ 400 Bad Request и ошибкой с текстом об отсутствии прав на добавление.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "You do not have permission to add users to this team.",
    "stackTrace": "   at Stoady.Handlers.Team.AddMember.AddMemberCommandHandler.Handle(AddMemberCommand request, CancellationToken ct) in /src/Stoady/Handlers/Team/AddMember/AddMemberCommandHandler.cs:line 53\n   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 62\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
```


Кейс 3. Команда существует, пользователя не существует, у добавляющего есть права на добавление.

Предусловие: существующая команда – возьмем команду «`HSE Android`» с идентификатором «4», редактировать роль будет участник команды «`HSE Android`», пользователь «`Anna`» с идентификатором «4» и ролью «`Creator`».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members/add`, в качестве параметра `teamId` взяв число 1000:

```
curl -X
'POST' 'https://stoady.herokuapp.com/teams/4/members/add?email=non_exi
stent_user%40hse.ru' \
-H 'executorId: 4'
```

Результат: получаем ответ 400 Bad Request и ошибкой.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "User with email = non_existent_user@hse.ru does not exist.",
    "stackTrace": "   at Stoady.Handlers.Team.AddMember.AddMemberCommandHandler.Handle(AddMemberCommand request, CancellationToken ct) in C:\\Studies\\Programming\\Stoady\\Stoady\\Handlers\\Team\\AddMember\\AddMemberCommandHandler.cs:line 63\r\n   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in C:\\Studies\\Programming\\Stoady\\Stoady\\Helpers\\ValidationBehavior.cs:line 62\r\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\r\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\r\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\r\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
```


Кейс 4. Команда существует, пользователь уже состоит в команде, у добавляющего есть права на добавление.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь – возьмем пользователя «Daniel» с идентификатором «8» и адресом электронной почты daniel@hse.ru, состоящий в команде «HSE Android», редактировать роль будет участник команды «HSE Android», пользователь «Ivan» с идентификатором «1» и ролью «Admin».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members/add:

```
curl -X
'POST' 'https://stoady.herokuapp.com/teams/4/members/add?email=daniel%
40hse.ru' \
-H 'executorId: 1'
```

Результат: получаем ответ 400 Bad Request.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when adding user:\r\n23505: duplicate key value violates unique constraint
        \"teamusers_teamid_userid_key\"\r\n\r\nDETAIL: Detail redacted as it may contain sensitive data. Specify 'Include Error Detail' in
        the connection string to include this information.",
    "stackTrace": "   at Stoady.Handlers.Team.AddMember.AddMemberCommandHandler.Handle(AddMemberCommand request, CancellationToken ct) in
        C:\\Studies\\Programming\\Stoady\\Stoady\\Handlers\\Team\\AddMember\\AddMemberCommandHandler.cs:line 80\r\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in
        C:\\Studies\\Programming\\Stoady\\Stoady\\Helpers\\ValidationBehavior.cs:line 62\r\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)
        \r\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate`1 next)\r\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
```


Кейс 5. Команда не существует.

Предусловие: существующий пользователь – возьмем пользователя «Kermit» с идентификатором «17» и адресом электронной почты kermit_the_frog@mail.ru, редактировать роль будет пользователь «Ivan» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members/add, взяв teamId = 1000:

```
curl -X
'POST' 'https://stoady.herokuapp.com/teams/4/members/add?email=daniel%
40hse.ru' \
-H 'executorId: 1'
```

Результат: получаем ответ 400 Bad Request с ошибкой о том, что пользователя с идентификатором «1» нет в команде с идентификатором «1000».

```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find user with ID = 1 in team with ID = 1000.",
    "stackTrace": "   at Stoady.Services.RightsValidatorService.ValidateRights(Int64 teamId, Int64 userId, CancellationToken ct) in /src/Stoady/Services/RightsValidatorService.cs:line 38\n   at Stoady.Handlers.Team.AddMember.AddMemberCommandHandler.Handle(AddMemberCommand request, CancellationToken ct) in /src/Stoady/Handlers/Team/AddMember/AddMemberCommandHandler.cs:line 51\n   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 62\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
```


- **Удаление участника из команды**

Кейс 1. Команда существует, участник существует, у пользователя есть права на удаление.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий в ней пользователь – возьмем пользователя «Daniel» с идентификатором «8», редактировать роль будет участник команды «HSE Android», пользователь «Ivan» с идентификатором «1» и ролью «Admin».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members/remove`:

```
curl -X
'DELETE' 'https://stoady.herokuapp.com/teams/4/members/remove?userId=8
' \
-H 'executorId: 1'
```

Результат: получаем ответ 200 OK, в команде больше нет пользователя с идентификатором «8».

	id	teamid	userid	roleid
1	6	4	4	1
2	42	4	1	2
3	104	4	5	3

Кейс 2. Команда существует, участник существует, у пользователя нет прав на удаление.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий в ней пользователь – возьмем пользователя «Ivan» с идентификатором «1», редактировать роль будет участник команды «HSE Android», пользователь «Andrey» с идентификатором «5» и ролью «Member».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members/remove:

```
curl -X
'DELETE' 'https://stoady.herokuapp.com/teams/4/members/remove?userId=1
' \
-H 'executorId: 5'
```

Результат: получаем ответ 400 Bad Request с сообщением о невозможности удаления пользователя из-за отсутствия прав.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "You do not have permissions to remove users from this team.",
    "stackTrace": "   at Stoady.Handlers.Team.RemoveMember.RemoveMemberCommandHandler.Handle(RemoveMemberCommand request, CancellationToken
        ct) in /src/Stoady/Handlers/Team/RemoveMember/RemoveMemberCommandHandler.cs:line 62\n   at Stoady.Helpers.ValidationBehavior`2.Handle
        (TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 62\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request,
```


Кейс 3. Команда существует, участника не существует, у пользователя есть права на удаление.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь, не состоящий в команде – возьмем пользователя «Daniel» с идентификатором «8», редактировать роль будет участник команды «HSE Android», пользователь «Ivan» с идентификатором «1» и ролью «Admin».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members/remove:

```
curl -X
'DELETE' 'https://stoady.herokuapp.com/teams/4/members/remove?userId=8
' \
-H 'executorId: 1'
```

Результат: ответ 400 Bad Request и сообщение о невозможности найти указанного пользователя в команде.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find user with ID = 8 in team with ID = 4.",
    "stackTrace": "   at Stoady.Handlers.Team.RemoveMember.RemoveMemberCommandHandler.Handle(RemoveMemberCommand request, CancellationToken
        ct) in /src/Stoady/Handlers/Team/RemoveMember/RemoveMemberCommandHandler.cs:line 52\n   at Stoady.Helpers.ValidationBehavior`2.Handle
        (TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request,
```


Кейс 4. Команда не существует.

Предусловие: существующие пользователи – возьмем пользователя «Andrey» с идентификатором «5», редактировать роль будет участник команды «HSE Android», пользователь «Ivan» с идентификатором 1 и ролью «Admin».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/members/remove`, подав вместо параметра `teamId` число 1000:

```
curl -X
'DELETE' 'https://stoady.herokuapp.com/teams/1000/members/remove?userI
d=5' \
-H 'executorId: 1'
```

Результат: ответ 400 Bad Request и сообщение о невозможности найти указанного пользователя в команде.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find user with ID = 5 in team with ID = 1000.",
    "stackTrace": "   at Stoady.Handlers.Team.RemoveMember.RemoveMemberCommandHandler.Handle(RemoveMemberCommand request, CancellationToken
        ct) in /src/Stoady/Handlers/Team/RemoveMember/RemoveMemberCommandHandler.cs:line 52\n   at Stoady.Helpers.ValidationBehavior`2.Handle
        (TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request,
```


Кейс 5. Удаление создателя команды.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь – возьмем пользователя «Anna» с

идентификатором «4», являющегося создателем команды, редактировать роль будет участник команды «HSE Android», пользователь «Ivan» с идентификатором «1» и ролью «Admin».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/members/remove:

```
curl -X
'DELETE' 'https://stoady.herokuapp.com/teams/4/members/remove?userId=4
' \
-H 'executorId: 1'
```

Результат: ответ 400 Bad Request и ошибкой с текстом о невозможности удалить создателя из команды.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "You cannot remove the creator from their team.",
    "stackTrace": "   at Stoady.Handlers.Team.RemoveMember.RemoveMemberCommandHandler.Handle(RemoveMemberCommand request, CancellationToken
        ct) in /src/Stoady/Handlers/Team/RemoveMember/RemoveMemberCommandHandler.cs:line 57\n   at Stoady.Helpers.ValidationBehavior`2.Handle
        (TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request,
```


- **Редактирование команды**

Кейс 1. Команда существует, у пользователя есть права на редактирование.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь, состоящий в команде и имеющий права на редактирование – возьмем пользователя «Ivan» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку /teams/{teamId}/avatar:

```
curl -X 'POST' 'https://stoady.herokuapp.com/teams/4/avatar' \
-H 'executorId: 1' \
-H 'Content-Type: application/json' \
-d '{
    "teamName": "new team name",
    "teamAvatar": "new team avatar link"
}'
```

Результат: получаем ответ 200 OK, в базе данных поменялись название и аватар команды.

	id	name	avatar	creatorid
1	4	new team name	new team avatar link	1

Кейс 2. Команда существует, у пользователя нет прав на редактирование.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь, не имеющий прав на редактирование – возьмем пользователя «Andrey» с идентификатором «5».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/avatar`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/teams/4/avatar' \
-H 'executorId: 5' \
-H 'Content-Type: application/json' \
-d '{
    "teamName": "new team name 2",
    "teamAvatar": "new team avatar link 2"
}'
```

Результат: получаем ответ 400 Bad Request с ошибкой с текстом о том, что пользователь не имеет прав на редактирование команды.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "You do not have permission to edit this team.",
    "stackTrace": "   at Stoady.Handlers.Team.EditTeam.EditTeamCommandHandler.Handle(EditTeamCommand request, CancellationToken ct) in /src/
        Stoady/Handlers/Team/EditTeam/EditTeamCommandHandler.cs:line 47\n   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request,
        CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n
        at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestPostProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken,
```


Кейс 3. Команда существует, пользователь не состоит в команде.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь, не состоящий в команде – возьмем пользователя «Kermit» с идентификатором «17».

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/avatar`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/teams/4/avatar' \
-H 'executorId: 17' \
-H 'Content-Type: application/json' \
-d '{
    "teamName": "new team name 3",
    "teamAvatar": "new team avatar link 3"
}'
```

Результат: получаем ответ 400 Bad Request с ошибкой о том, что пользователь в команде не найден.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find user with ID = 17 in team with ID = 4.",
    "stackTrace": "   at Stoady.Services.RightsValidatorService.ValidateRights(Int64 teamId, Int64 userId, CancellationToken ct) in /src/
        Stoady/Services/RightsValidatorService.cs:line 38\n   at Stoady.Handlers.Team.EditTeam.EditTeamCommandHandler.Handle(EditTeamCommand
        request, CancellationToken ct) in /src/Stoady/Handlers/Team/EditTeam/EditTeamCommandHandler.cs:line 45\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest
        request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
```


Кейс 4. Команда не существует.

Предусловие: существующая команда – возьмем команду «HSE Android» с идентификатором «4», существующий пользователь – возьмем пользователя «Ivan» с идентификатором «1», состоящий в команде и имеющий права на редактирование.

С помощью Postman отправляем запрос в конечную точку `/teams/{teamId}/avatar`, вместо параметра `teamId` подадим число 1000:

```
curl -X 'POST' 'https://stoady.herokuapp.com/teams/1000/avatar' \
-H 'executorId: 1' \
-H 'Content-Type: application/json' \
-d '{
    "teamName": "new team name 4",
    "teamAvatar": "new team avatar link 4"
}'
```

Результат: получили ответ 400 Bad Request и ошибкой о том, что пользователь в команде не найден.

```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find user with ID = 1 in team with ID = 1000.",
    "stackTrace": "   at Stoady.Services.RightsValidatorService.ValidateRights(Int64 teamId, Int64 userId, CancellationToken ct) in /src/
        Stoady/Services/RightsValidatorService.cs:line 38\n   at Stoady.Handlers.Team.EditTeam.EditTeamCommandHandler.Handle(EditTeamCommand
        request, CancellationToken ct) in /src/Stoady/Handlers/Team/EditTeam/EditTeamCommandHandler.cs:line 45\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest
        request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken
```


6.2.11. Управление предметами

- **Создание нового предмета**

Кейс 1. Команда существует.

Предусловие: существующая команда для добавления предмета – возьмем команду «HSE Android» с идентификатором «4».

С помощью Postman отправляем запрос в конечную точку `/subjects/add`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/subjects/add' \
-H 'Content-Type: application/json' \
-d '{
    "teamId": 4,
    "subjectName": "new subject",
    "subjectDescription": "new subject description"
}'
```

Результат: получили ответ 200 OK. В базе данных появился новый предмет.

	id	title	description	teamid
1	26	new subject	new subject description	4

Кейс 2. Команда не существует.

С помощью Postman отправляем запрос в конечную точку `/subjects/add`, взяв `teamId = 1000`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/subjects/add' \
-H 'Content-Type: application/json' \
-d '{
    "teamId": 1000,
    "subjectName": "new subject",
```

```
    "subjectDescription": "new subject description"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой и текстом о том, что не удалось найти команду с нужным идентификатором.

Кейс 3. Некорректный предмет.

Предусловие: существующая команда для добавления предмета – возьмем команду «`HSE Android`» с идентификатором «4».

С помощью Postman отправляем запрос в конечную точку `/subjects/add`. В качестве названия напишем пустую строку, так как длина названия должна быть от 1 до 50 символов включительно. Также вставим пустое описание – это делать можно. Убедимся, что в ответе не будет сказано про пустое описание.

```
curl -X 'POST' 'https://stoady.herokuapp.com/subjects/add' \
-H 'Content-Type: application/json' \
-d '{
    "teamId": 4,
    "subjectName": "",
    "subjectDescription": ""
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой валидации на длину названия, но не описания.

```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nSubjectName: 'Subject Name' must be between 1 and 50 characters. You entered 0 characters.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
```


- **Редактирование предмета**

Кейс 1. Предмет существует.

Предусловие: существующий предмет – возьмем предмет с идентификатором «26», созданный в предыдущем пункте.

С помощью Postman отправляем запрос в конечную точку `/subjects/edit/{subjectId}`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/subjects/edit/26' \
-H 'Content-Type: application/json' \
-d '{
    "subjectName": "edited name",
    "subjectDescription": "edited description"
}'
```

Результат: получили ответ 200 OK. В базе данных появился обновились название и описание темы.

	id	title	description	teamid
1	26	edited name	edited description	4

Кейс 2. Предмет не существует.

С помощью Postman отправляем запрос в конечную точку `/subjects/edit/{subjectId}`, взяв `subjectId = 1000`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/subjects/edit/1000' \
-H 'Content-Type: application/json' \
```

```
-d '{
    "topicName": "edited name 2",
    "topicDescription": "edited description 2"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when editing the subject. Please, try again.",
    "stackTrace": "   at Stoady.Handlers.Subject.EditSubject.EditSubjectCommandHandler.Handle
        (EditSubjectCommand request, CancellationToken ct) in /src/Stoady/Handlers/Subject/
        EditSubject/EditSubjectCommandHandler.cs:line 53\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


Кейс 3. Некорректный предмет.

Предусловие: существующий предмет – возьмем предмет с идентификатором «26».

С помощью Postman отправляем запрос в конечную точку `/subjects/edit/{subjectId}`. В нового названия напишем пустую строку, так как длина названия должна быть от 1 до 50 включительно.

```
curl -X 'POST' 'https://stoady.herokuapp.com/topics/edit/26' \
-H 'Content-Type: application/json' \
-d '{
    "subjectName": "",
    "subjectDescription": "edited description 3"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой валидации на длину названия предмета.

```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nSubjectName: 'Subject Name' must be between 1 and
        50 characters. You entered 0 characters.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request,
        CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.
        Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
```


- **Удаление предмета**

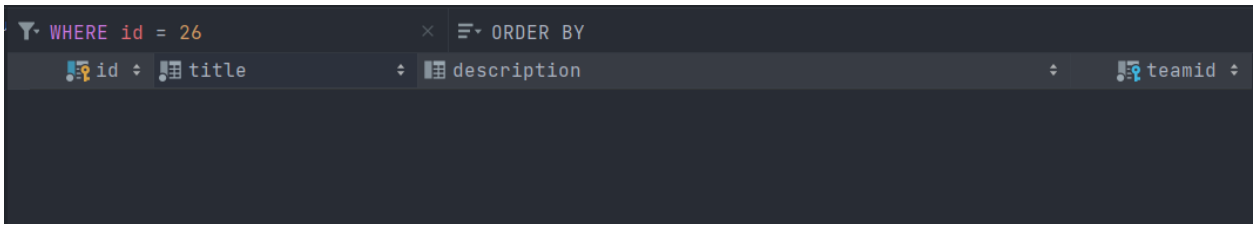
Кейс 1. Предмет существует.

Предусловие: существующий предмет – возьмем предмет с идентификатором «26».

С помощью Postman отправляем запрос в конечную точку `/subjects/remove/{subjectId}`:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/subjects/remove/26'
```

Результат: получили ответ 200 OK. В базе данных темы с идентификатором «26» больше не существует.



Кейс 2. Предмет не существует.

С помощью Postman отправляем запрос в конечную точку `/subjects/remove/{subjectId}`, взяв `subjectId = 26`, который только что удалили:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/subjects/remove/26'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой.

```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when removing the subject. Please, try again.",
    "stackTrace": "    at Stoady.Handlers.Subject.RemoveSubject.RemoveSubjectCommandHandler.Handle
        (RemoveSubjectCommand request, CancellationToken ct) in /src/Stoady/Handlers/Subject/
        RemoveSubject/RemoveSubjectCommandHandler.cs:line 43\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


6.2.12. Управление темами

- **Создание новой темы**

Кейс 1. Предмет существует.

Предусловие: существующий предмет для добавления темы – возьмем тему «`Algebra`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/topics/add`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/topics/add' \
-H 'Content-Type: application/json' \
-d '{
    "subjectId": 1,
    "topicName": "new topic",
    "topicDescription": "new topic description"
}'
```

Результат: получили ответ 200 OK. В базе данных появилась новая тема.

	id	title	description	subjectid
1	28	new topic	new topic description	1

Кейс 2. Предмет не существует.

С помощью Postman отправляем запрос в конечную точку `/topics/add`, взяв `subjectId = 1000`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/topics/add' \
-H 'Content-Type: application/json' \
```

```
-d '{
    "subjectId": 1000,
    "topicName": "new topic",
    "topicDescription": "new topic description"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой и текстом о том, что не удалось найти предмет с нужным идентификатором.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find topic with ID = 1000",
    "stackTrace": "   at Stoady.Handlers.Topic.AddTopic.AddTopicCommandHandler.Handle
        (AddTopicCommand request, CancellationToken ct) in /src/Stoady/Handlers/Topic/AddTopic/
        AddTopicCommandHandler.cs:line 47\n   at Stoady.Helpers.ValidationBehavior`2.Handle
        (TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/
        Helpers/ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
        RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


Кейс 3. Некорректная тема.

Предусловие: существующий предмет для добавления темы – возьмем предмет «`Algebra`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/topics/add`. В качестве названия напишем пустую строку, так как длина названия должна быть от 1 до 50 символов включительно. Также вставим пустое описание – это делать можно. Убедимся, что в ответе не будет сказано про пустое описание.

```
curl -X 'POST' 'https://stoady.herokuapp.com/topics/add' \
-H 'Content-Type: application/json' \
-d '{
    "subjectId": 1000,
    "topicName": "",
    "topicDescription": ""
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой валидации на длину названия, но не описания.

```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nTopicName: 'Topic Name' must be between 1 and 50 characters. You entered 0 characters.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request, CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
```


- **Редактирование темы**

Кейс 1. Тема существует.

Предусловие: существующая тема – возьмем тему с идентификатором «28», созданную в предыдущем пункте.

С помощью Postman отправляем запрос в конечную точку `/topics/edit/{topicId}`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/topics/edit/28' \
-H 'Content-Type: application/json' \
-d '{
    "topicName": "edited name",
    "topicDescription": "edited description"
}'
```

Результат: получили ответ 200 OK. В базе данных появился обновились название и описание темы.

	id	title	description	subjectid
1	28	edited name	edited description	1

Кейс 2. Темы не существует.

С помощью Postman отправляем запрос в конечную точку `/topics/edit/{topicId}`, взяв `topicId = 1000`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/topics/edit/1000' \
-H 'Content-Type: application/json' \
-d '{
```

```
    "topicName": "edited name 2",
    "topicDescription": "edited description 2"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой.

Кейс 3. Некорректная тема.

Предусловие: существующая тема – возьмем тему с идентификатором «28».

С помощью Postman отправляем запрос в конечную точку `/topics/edit/{topicId}`. В нового названия напишем пустую строку, так как длина названия должна быть от 1 до 50 включительно.

```
curl -X 'POST' 'https://stoady.herokuapp.com/topics/edit/28' \
-H 'Content-Type: application/json' \
-d '{
    "topicName": "",
    "topicDescription": "edited description 3"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой валидации на длину названия темы.

```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nTopicName: 'Topic Name' must be between 1 and 50
        characters. You entered 0 characters.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request,
        CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.
        Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
```


- **Удаление темы**

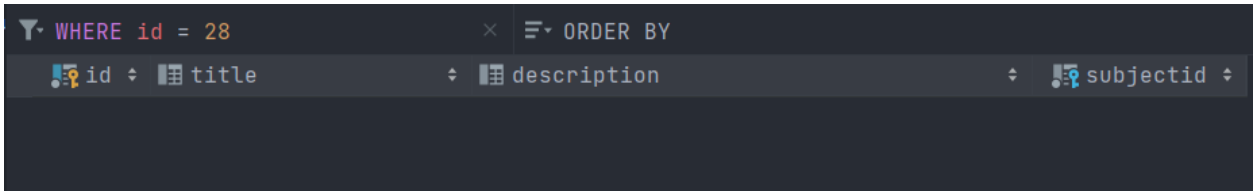
Кейс 1. Тема существует.

Предусловие: существующая тема – возьмем тему с идентификатором «28».

С помощью Postman отправляем запрос в конечную точку `/topics/remove/{topicId}`:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/topics/remove/28'
```

Результат: получили ответ 200 OK. В базе данных темы с идентификатором «28» больше не существует.



Кейс 2. Тема не существует.

С помощью Postman отправляем запрос в конечную точку `/topics/remove/{topicId}`, взяв `topicId = 28`, которую только что удалили:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/topics/remove/28'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой.

```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when removing this topic. Please, try again.",
    "stackTrace": "   at Stoady.Handlers.Topic.RemoveTopic.RemoveTopicCommandHandler.Handle
        (RemoveTopicCommand request, CancellationToken ct) in /src/Stoady/Handlers/Topic/
        RemoveTopic/RemoveTopicCommandHandler.cs:line 43\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


6.2.13. Управление вопросами

- **Создание нового вопроса**

Кейс 1. Тема существует.

Предусловие: существующая тема для добавления вопроса – возьмем тему «`Matrices`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/questions/add`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/questions/add' \
-H 'Content-Type: application/json' \
-d '{
    "topicId": 1,
    "questionText": "new question",
    "answerText": "new answer"
}'
```

Результат: получили ответ 200 OK. В базе данных появился новый вопрос.

	id	questiontext	answertext	topicid
1	43	new question	new answer	1

Кейс 2. Тема не существует.

С помощью Postman отправляем запрос в конечную точку `/questions/add`, взяв `topicId = 1000`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/questions/add' \
-H 'Content-Type: application/json' \
-d '{
```

```
    "topicId": 1000,
    "questionText": "new question",
    "answerText": "new answer"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой и текстом о том, что не удалось найти тему с нужным идентификатором.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Could not find topic with ID = 1000",
    "stackTrace": "   at Stoady.Handlers.Question.AddQuestion.AddQuestionCommandHandler.Handle
        (AddQuestionCommand request, CancellationToken ct) in /src/Stoady/Handlers/Question/
        AddQuestion/AddQuestionCommandHandler.cs:line 47\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


Кейс 3. Некорректный вопрос.

Предусловие: существующая тема для добавления вопроса – возьмем тему «`Matrices`» с идентификатором «1».

С помощью Postman отправляем запрос в конечную точку `/questions/add`. В качестве текста вопроса напишем пустую строку, так как длина вопроса должна быть от 1 до 250 включительно.

```
curl -X 'POST' 'https://stoady.herokuapp.com/questions/add' \
-H 'Content-Type: application/json' \
-d '{
    "topicId": 1,
    "questionText": "",
    "answerText": "new answer"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой валидации на длину вопроса.

```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nQuestionText: 'Question Text' must be between 1
        and 100 characters. You entered 0 characters.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request,
        CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.
        Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
```


- **Редактирование вопроса**

Кейс 1. Вопрос существует.

Предусловие: существующий вопрос – возьмем вопрос с идентификатором «43».

С помощью Postman отправляем запрос в конечную точку `/questions/edit/{questionId}`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/questions/edit/43' \
-H 'Content-Type: application/json' \
-d '{
    "questionText": "edited question",
    "answerText": "edited answer"
}'
```

Результат: получили ответ 200 OK. В базе данных появился обновились текст вопроса и ответа.

	id	questiontext	answertext	topicid
1	43	edited question	edited answer	1

Кейс 2. Вопрос не существует.

С помощью Postman отправляем запрос в конечную точку `/questions/edit/{questionId}`, взяв `questionId = 1000`:

```
curl -X 'POST' 'https://stoady.herokuapp.com/questions/edit/1000' \
```

```
-H 'Content-Type: application/json' \
-d '{
    "questionText": "edited question 2",
    "answerText": "edited answer 2"
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой.


```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when editing the question. Please, try again.",
    "stackTrace": "   at Stoady.Handlers.Question.EditQuestion.EditQuestionCommandHandler.Handle
        (EditQuestionCommand request, CancellationToken ct) in /src/Stoady/Handlers/Question/
        EditQuestion/EditQuestionCommandHandler.cs:line 53\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


Кейс 3. Некорректный вопрос.

Предусловие: существующий вопрос – возьмем вопрос с идентификатором «43».

С помощью Postman отправляем запрос в конечную точку `/questions/edit/{questionId}`. В качестве текста ответа напишем пустую строку, так как длина ответа должна быть от 1 до 250 включительно.

```
curl -X 'POST' 'https://stoady.herokuapp.com/questions/edit/43' \
-H 'Content-Type: application/json' \
-d '{
    "topicId": 1,
    "questionText": " edited question 3",
    "answerText": ""
}'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой валидации на длину текста ответа.

```
{
    "title": "Server Error",
    "code": 400,
    "source": "Stoady",
    "message": "Validation error(s) occurred:\r\nAnswerText: 'Answer Text' must be between 1 and
        100 characters. You entered 0 characters.",
    "stackTrace": "   at Stoady.Helpers.ValidationBehavior`2.Handle(TRequest request,
        CancellationToken ct, RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/
        ValidationBehavior.cs:line 63\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.
        Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate`1
        next)\n   at MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionActionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
```


- **Удаление вопроса**

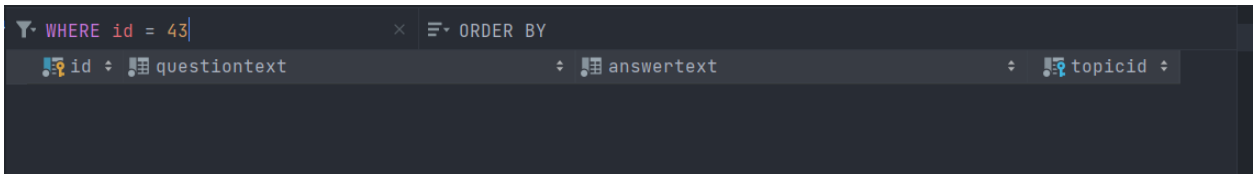
Кейс 1. Вопрос существует.

Предусловие: существующий вопрос – возьмем вопрос с идентификатором «43».

С помощью Postman отправляем запрос в конечную точку `/questions/remove/{questionId}`:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/questions/remove/43'
```

Результат: получили ответ 200 OK. В базе данных вопрос больше не существует.



Кейс 2. Вопрос не существует.

С помощью Postman отправляем запрос в конечную точку `/questions/remove/{questionId}`, взяв `questionId = 43`, который только что удалили:

```
curl -X 'DELETE' 'https://stoady.herokuapp.com/questions/remove/43'
```

Результат: получили ответ 400 Bad Request и объект с ошибкой.

```
{
    "title": "Stoady",
    "code": 400,
    "source": "Stoady",
    "message": "Something went wrong when removing the question. Please, try again.",
    "stackTrace": "   at Stoady.Handlers.Question.RemoveQuestion.RemoveQuestionCommandHandler.
        Handle(RemoveQuestionCommand request, CancellationToken ct) in /src/Stoady/Handlers/
        Question/RemoveQuestion/RemoveQuestionCommandHandler.cs:line 43\n   at Stoady.Helpers.
        ValidationBehavior`2.Handle(TRequest request, CancellationToken ct,
        RequestHandlerDelegate`1 next) in /src/Stoady/Helpers/ValidationBehavior.cs:line 63\n   at
        MediatR.Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request,
        CancellationToken cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.
        Pipeline.RequestExceptionProcessorBehavior`2.Handle(TRequest request, CancellationToken
        cancellationToken, RequestHandlerDelegate`1 next)\n   at MediatR.Pipeline.
```


6.2.14. Вывод

По итогам тестирования, можно утверждать, что программа удовлетворяет функциональным требованиям настоящего Технического задания.

6.3. Проверка входных и выходных данных

API принимает на вход заявленные HTTP-запросы. API возвращает ответы в формате JSON-объектов. Логи приложения пишутся. Найти их можно по ссылке https://dashboard.heroku.com/apps/stoady/logs.

6.4. Проверка требований к надежности

Ошибка	Как воспроизвести
400 Bad Request	Подать в запрос идентификатор несуществующей сущности. Например, запрос `curl -X 'DELETE' \ 'https://stoady.herokuapp.com/questions/remove/1000'` удаляет вопрос с идентификатором 1000, которого не существует в базе данных
401 Unauthorized	Подать в конечную точку для авторизации некорректные данные. Например, запрос `curl -X 'POST' 'https://stoady.herokuapp.com/auth/authorize' \ -H 'Content-Type: application/json' \ -d '{ "email": "ivan@hse.ru",`

	`"password": "wrong_password"`
	`}'`
	вернет ошибку 401 Unauthorized, так как введен неверный пароль.

404 Not Found	Обратиться к несуществующей конечной точке. Например, запрос
	`curl -X 'GET' \`
	`'https://stoady.herokuapp.com/info'`
	вернет код 404 Not Found, так как такой конечной точки API не предоставляет.

405 Method Not Allowed	Послать неверный тип запроса. Например, запрос
	`curl -X 'GET' \`
	`'https://stoady.herokuapp.com/questions/remove/15'`
	вернет ошибку 405 Method Not Allowed, так как тип конечной точки API /questions/remove/{questionId} – DELETE, а не GET.

500 Internal Server Error	Попытаться добавить в базу данных сущность с идентификатором, внешний ключ которого не соответствует никакому объекту. Этот ответ нельзя получить из API, так как по результатам тестирования все конечные точки обрабатывают исключения такого рода.

СПИСОК ИСТОЧНИКОВ

1) ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2) ГОСТ 19.102-77 Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3) ГОСТ 19.103-77 Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4) ГОСТ 19.104-78 Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5) ГОСТ 19.105-78 Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7) ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8) ГОСТ 19.603-78 Общие правила внесения изменений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9) ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10) ГОСТ 19.404-79 Программа и методика испытаний. Требования к содержанию и оформлений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
12) ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
13) ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

14) Документация ASP .NET Core [Электронный ресурс] / Режим доступа: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0, свободный.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					