Федеральное агентство железнодорожного транспорта

Омский государственный университет путей сообщения

Кафедра «Автоматика и системы управления»

К ЗАЩИТЕ ДОПУСТИТЬ

Доцент кафедры АиСУ

\_\_\_\_\_\_\_\_ Е.А. Альтман

ПРОЕКТИРОВАНИЕ REST-СЕРВЕРА

Пояснительная записка к курсовой работе

по дисциплине «Объектно – ориентированное программировании»

ИНМВ.400000.000 ПЗ

|  |  |
| --- | --- |
| Оценка  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.  \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись Расшифровка подписи | Студент гр. 28М  \_\_\_\_\_\_\_\_\_\_\_М.М.Маликов  «\_\_» \_\_\_\_\_\_\_\_ 2020 г.  Руководитель – Доцент кафедры «АиСУ»  \_\_\_\_\_\_\_\_\_\_\_Е.А. Альтман  «\_\_» \_\_\_\_\_\_\_\_2020 г. |

Омск 2020

Задание

Разработать REST-сервер оценок.

Сервер позволяет получать и редактировать оценки студентов по различным предметам (а также самих студентов и предметы, студенты объединяются по группам, у каждой группы – свой список предметов), получать списки учащихся по различным критериям (имеет двойки, только 4 и 5 и т. п.).

Реферат

УДК 004.42

Пояснительная записка к курсовой работе содержит 36 страниц, 47 рисунков, 3 использованных источника, 2 приложения.

Объектом курсовой работы является проектирование и реализация REST-сервера.

Цель курсовой работы – получение основных навыков проектирования REST-сервера, знакомство с ORM Kotlin Exposed и фреймворком Ktor и реализация REST-сервера сайта поиска работы.

Результатом курсовой работы является REST-сервер оценок, написанный на языке Kotlin в программе IntelliJ IDEA.

Пояснительная записка выполнена в текстовом редакторе Microsoft Word 2019.

Содержание

[Введение 5](#_Toc59924079)

[1 ER-диаграмма 6](#_Toc59924080)

[2 Диаграмма классов 7](#_Toc59924081)

[3 Диаграмма Прецедентов 8](#_Toc59924082)

[4 Диаграммы последовательности прецедентов 9](#_Toc59924083)

[5 ORM 16](#_Toc59924084)

[6 Rest 21](#_Toc59924085)

[Заключение 32](#_Toc59924086)

[Библиографический список 33](#_Toc59924087)

[Приложение А 34](#_Toc59924088)

[Приложение B 36](#_Toc59924089)

# Введение

REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами.

В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

В курсовой работе необходимо разработать REST-сервер, исходя из REST API (приложение А), клиентский интерфейс для него, а также описать архитектуру приложения. В данной части курсовой работы будет описана архитектура.

# ER-диаграмма

ER-диаграммы отображают отношения набора сущностей, хранящиеся в базе данных. ER-диаграммы могут объяснить логическую структуру баз данных.

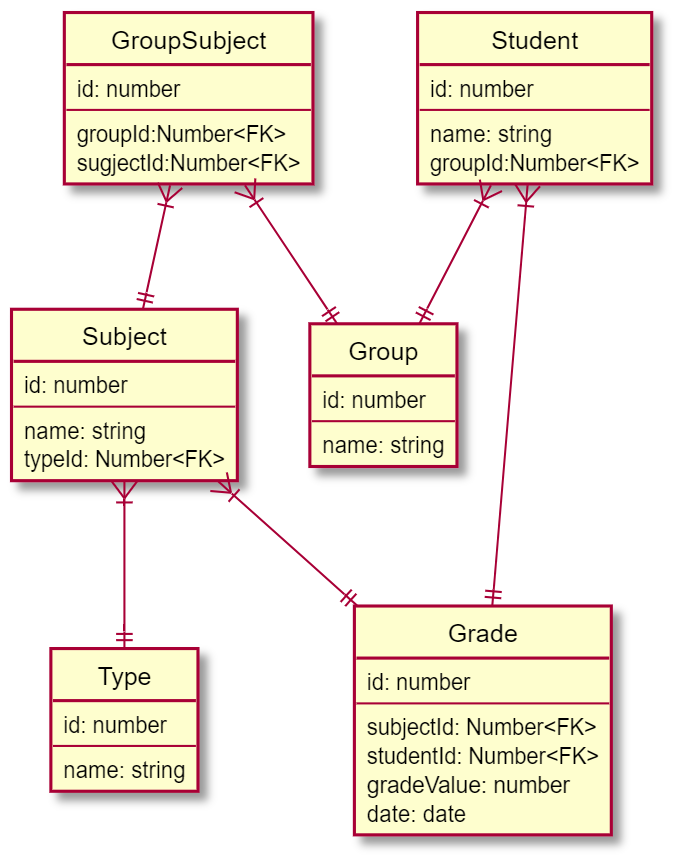


Рисунок 1 – ER-диаграмма

По данной ER-диаграмме была спроектирована база данных.

# Диаграмма классов

Всего в курсовой работе используются 6 классов: Type, Grade, Subject, Student, Group, GroupSubject(рисунок 2).

Класс Type представляет собой тип для класса Subject.

Класс Grade представляет собой оценку и содержит в себе id предмета, id студента, оценку и дату выставления.

Subject представляет собой предмет и содержит id типа и имя.

Student представляет собой студента и содержит имя и id группы, в которой он состоит.

Group представляет собой группу и содержит в себе имя группы.

GroupSubject представляет собой промежуточную таблицу для объединения групп и студентов. Содержит в себе id группы и id студента.

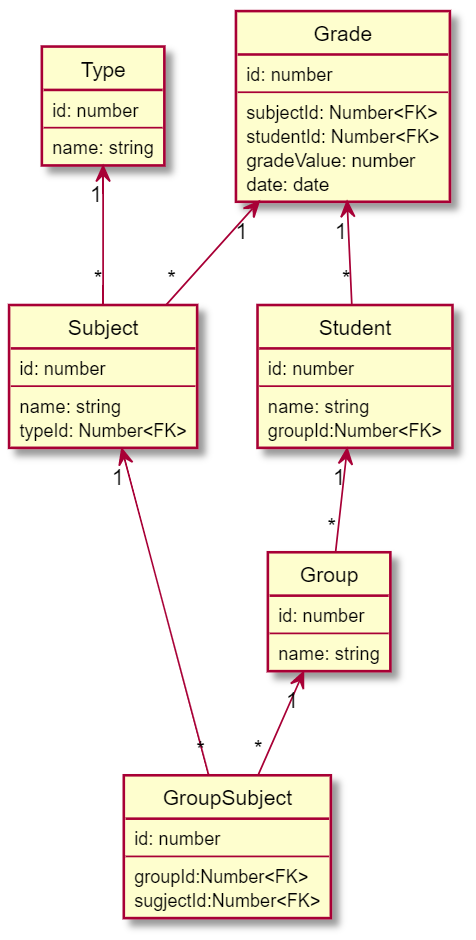


Рисунок 2 – Диаграмма классов

## Диаграмма прецедентов

Для реализации функций получения/ изменения/ создания объектов. Например, получать списки учащихся по различным критериям (имеет двойки, только 4 и 5 и т. п.).

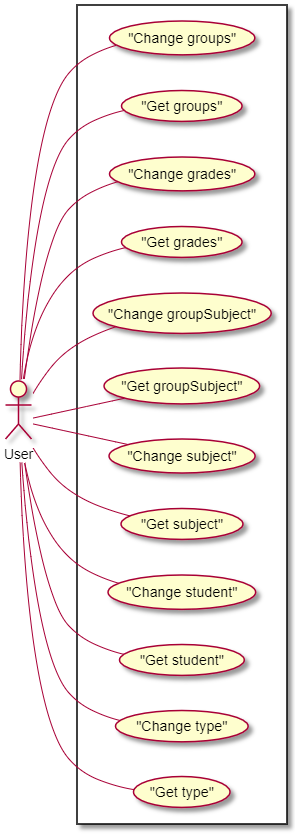


Рисунок 3 – Диаграмма прецедентов

## 4 Диаграммы последовательности прецедентов

При выполнении прецедента change grade, пользователь отправляет соответствующий запрос, в тело которого передает json объект, который будет преобразован в экземпляр класса Grade и будет помещен в новую строку(при методе POST) таблицы GradeTable, или строка данной таблицы будет изменена(при методе PUT). Если пользователь передал верный объект, он будет добавлен, а пользователю будет возвращено подтверждающее сообщение, в ином случае объект не будет добавлен, и пользователь получит уведомление об ошибке.

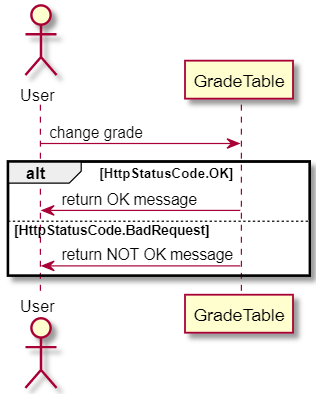


Рисунок 4 – Диаграмма последовательности change grade

При выполнении прецедента change group (создании/изменении/удалении группы), пользователь отправляет соответствующий запрос, в тело которого передает json объект, который будет преобразован в экземпляр класса Group и будет помещен в новую строку(при методе POST) таблицы GroupTable, или строка данной таблицы будет изменена(при методе PUT). Если пользователь передал верный объект, он будет добавлен, а пользователю будет возвращено подтверждающее сообщение, в ином случае объект не будет добавлен, и пользователь получит уведомление об ошибке.

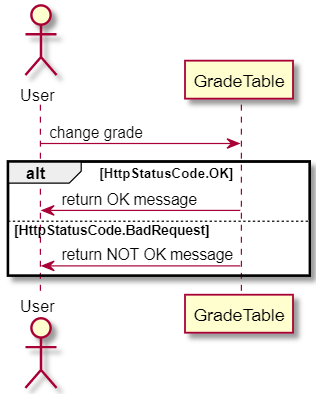


Рисунок 5 – Диаграмма последовательности change grade

При выполнении прецедента change *g*roupsubject (создании/изменении/удалении группы), пользователь отправляет соответствующий запрос, в тело которого передает json объект, который будет преобразован в экземпляр класса GroupSubject и будет помещен в новую строку(при методе POST) таблицы GroupSubjectTable, или строка данной таблицы будет изменена(при методе PUT). Если пользователь передал верный объект, он будет добавлен, а пользователю будет возвращено подтверждающее сообщение, в ином случае объект не будет добавлен, и пользователь получит уведомление об ошибке.

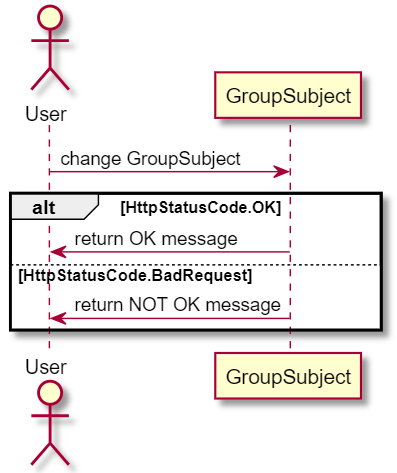


Рисунок 6 – Диаграмма последовательности change group subject

При выполнении прецедента change student (создании/изменении/удалении группы), пользователь отправляет соответствующий запрос, в тело которого передает json объект, который будет преобразован в экземпляр класса Student и будет помещен в новую строку(при методе POST) таблицы StudentTable, или строка данной таблицы будет изменена(при методе PUT). Если пользователь передал верный объект, он будет добавлен, а пользователю будет возвращено подтверждающее сообщение, в ином случае объект не будет добавлен, и пользователь получит уведомление об ошибке.

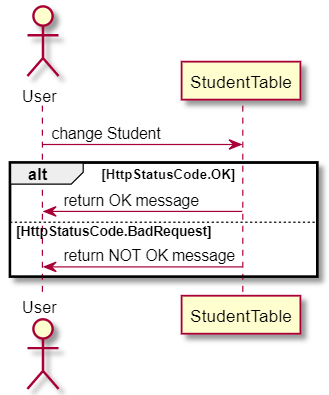


Рисунок 7 – Диаграмма последовательности change student

При выполнении прецедента change subject (создании/изменении/удалении группы), пользователь отправляет соответствующий запрос, в тело которого передает json объект, который будет преобразован в экземпляр класса Subject и будет помещен в новую строку(при методе POST) таблицы SubjectTable, или строка данной таблицы будет изменена(при методе PUT). Если пользователь передал верный объект, он будет добавлен, а пользователю будет возвращено подтверждающее сообщение, в ином случае объект не будет добавлен, и пользователь получит уведомление об ошибке.

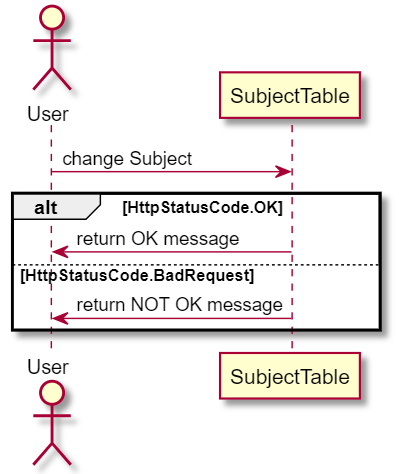


Рисунок 8 – Диаграмма последовательности change subject

При выполнении прецедента change type (создании/изменении/удалении группы), пользователь отправляет соответствующий запрос, в тело которого передает json объект, который будет преобразован в экземпляр класса Type и будет помещен в новую строку(при методе POST) таблицы TypeTable, или строка данной таблицы будет изменена(при методе PUT). Если пользователь передал верный объект, он будет добавлен, а пользователю будет возвращено подтверждающее сообщение, в ином случае объект не будет добавлен, и пользователь получит уведомление об ошибке.

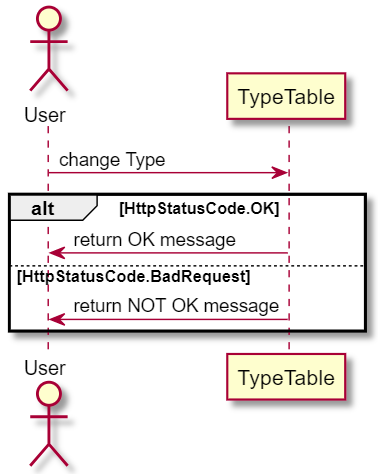


Рисунок 9 – Диаграмма последовательности change type

При выполнении прецедента get grade, пользователь отправляет запрос на соответствующий адрес, принимая который сервер, возвращает пользователю объект в формате JSON, соответствующий экземпляру класса Grade. В случае, если запрос не был успешно обработан, пользователю будет возвращено уведомление об ошибке.

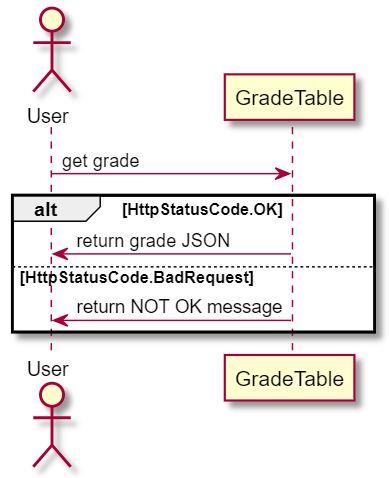


Рисунок 10 – Диаграмма последовательности get grade

При выполнении прецедента get group, пользователь отправляет запрос на соответствующий адрес, принимая который сервер, возвращает пользователю объект в формате JSON, соответствующий экземпляру класса Group. В случае, если запрос не был успешно обработан, пользователю будет возвращено уведомление об ошибке.

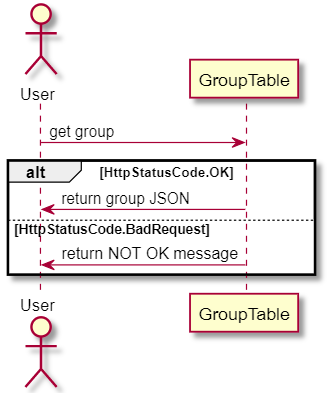


Рисунок 11 – Диаграмма последовательности get group

При выполнении прецедента get group subject, пользователь отправляет запрос на соответствующий адрес, принимая который сервер, возвращает пользователю объект в формате JSON, соответствующий экземпляру класса GroupSubject. В случае, если запрос не был успешно обработан, пользователю будет возвращено уведомление об ошибке.

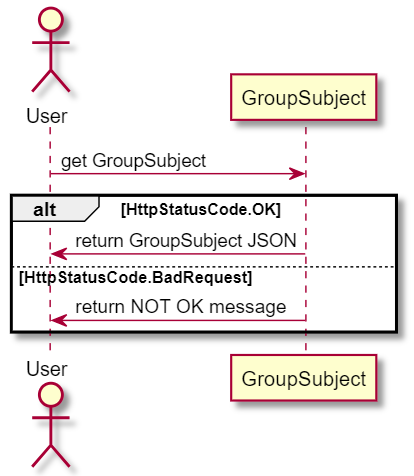


Рисунок 12 – Диаграмма последовательности get group subject

При выполнении прецедента get student, пользователь отправляет запрос на соответствующий адрес, принимая который сервер, возвращает пользователю объект в формате JSON, соответствующий экземпляру класса Student. В случае, если запрос не был успешно обработан, пользователю будет возвращено уведомление об ошибке.

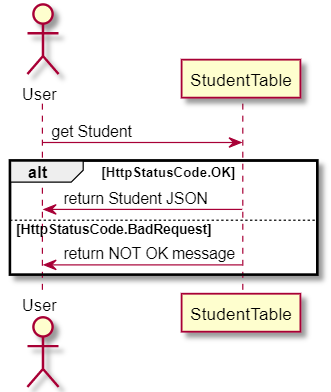


Рисунок 13 – Диаграмма последовательности get student

При выполнении прецедента get subject, пользователь отправляет запрос на соответствующий адрес, принимая который сервер, возвращает пользователю объект в формате JSON, соответствующий экземпляру класса Subject. В случае, если запрос не был успешно обработан, пользователю будет возвращено уведомление об ошибке.

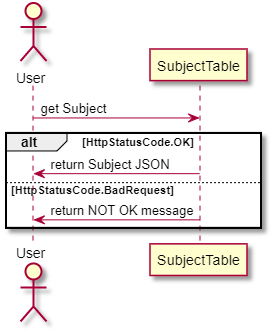


Рисунок 14 – Диаграмма последовательности get subject

При выполнении прецедента get type, пользователь отправляет запрос на соответствующий адрес, принимая который сервер, возвращает пользователю объект в формате JSON, соответствующий экземпляру класса Type. В случае, если запрос не был успешно обработан, пользователю будет возвращено уведомление об ошибке.

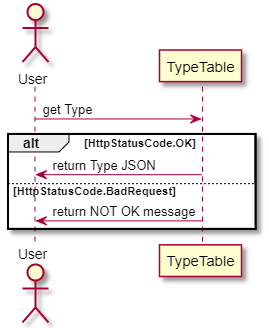


Рисунок 15 – Диаграмма последовательности get type

## ORM

ORM – технология программирования, позволяющая связать базы данных с концепциями объектно-ориентированного программирования. ORM является промежуточным звеном между базой данных и кодом который пишет программист, позволяющим читать, создавать, удалять и редактировать объекты в базе данных.

В программе были реализованы следующие классы для функционирования REST сервера.

Класс Grade представляет собой оценку, выставленную студенту за определенный предмет. Он содержит id студента, id предмета, саму оценку и дату выставления.

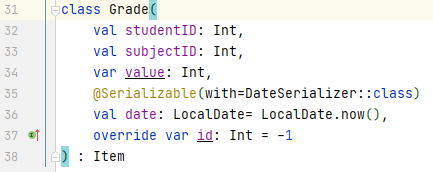


Рисунок 16 – Класс Grade

Класс Group представляет собой группу. Содержит в себе имя группы.

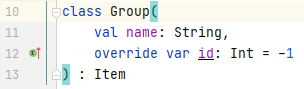


Рисунок 17 – Класс Group

Класс GroupSubject служит для создания списка предметов для групп. Содержит в себе id группы и id предмета.

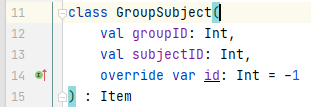


Рисунок 18 – Класс GroupSubject

Класс Student представляет собой студента. Содержит в себе имя и id группы, к которой принадлежит.

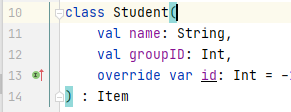


Рисунок 19 – Класс Student

Класс Subject представляет собой предмет. Содержит в себе имя и id типа.

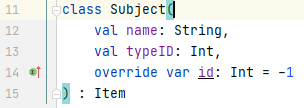


Рисунок 20 – Класс Subject

Класс Type представляет собой тип предмета. Содержит в себе имя.

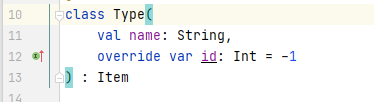


Рисунок 21 – Класс Type

Для взаимодействия с базой данных был использован репозиторий на DSL (рисунок 22).

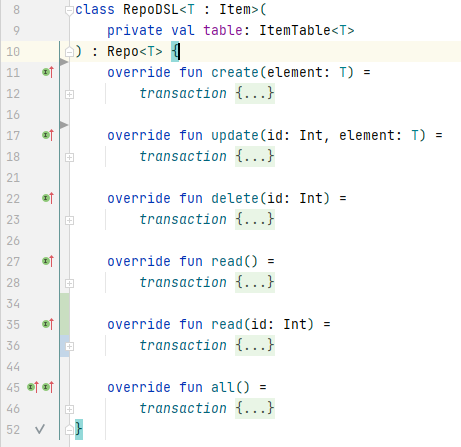


Рисунок 22 – RepoDSL

Так же были реализованы классы для таблиц (рисунок 23,24,25,26,27,28,29).

Таблица GradeTable содержит в себе оценки. Строками данной таблицы являются объекты класса Grade.

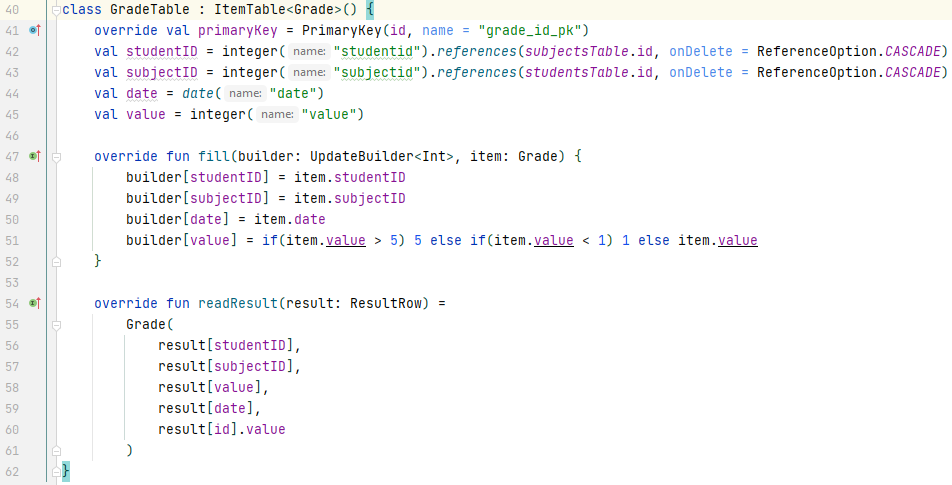


Рисунок 23 – Таблица GradeTable

Таблица GroupsTable содержит в себе группы. Строками данной таблицы являются объекты класса Group.

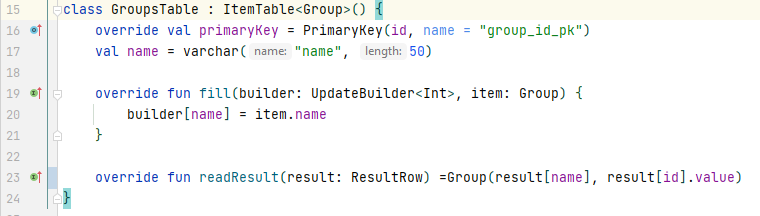


Рисунок 24 – Таблица GroupsTable

Промежуточная таблица GroupSubjectTable объединяет в себе группы и предметы, таким образом формируется список предметов для каждой группы. Строками данной таблицы являются объекты класса Group.



Рисунок 25 – Таблица GroupSubjectTable

Таблица StudentsTable содержит в себе студентов. Строками данной таблицы являются объекты класса Student.

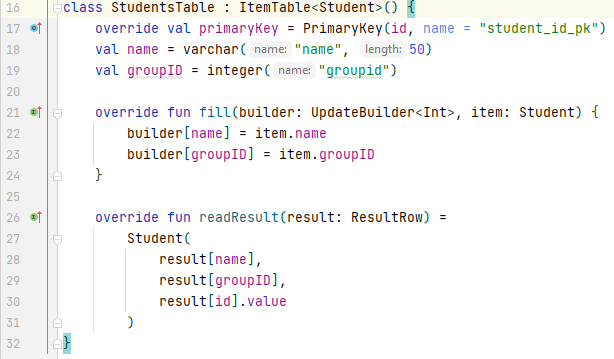


Рисунок 26 – Таблица StudentTable

Таблица SubjectsTable содержит в себе предметы. Строками данной таблицы являются объекты класса Subject.



Рисунок 27 – Таблица SubjectsTable

Таблица TypesTable содержит в себе типы предметов. Строками данной таблицы являются объекты класса Type.

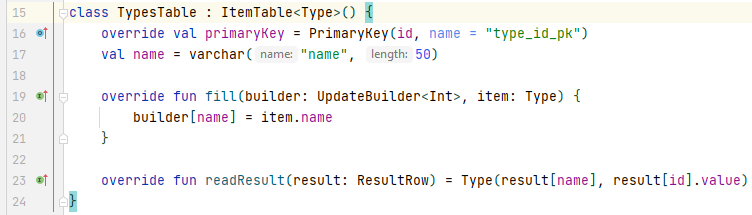


Рисунок 28 – Таблица TypesTable

Подключение к базе данных h2 и создание таблиц (рисунок 29)

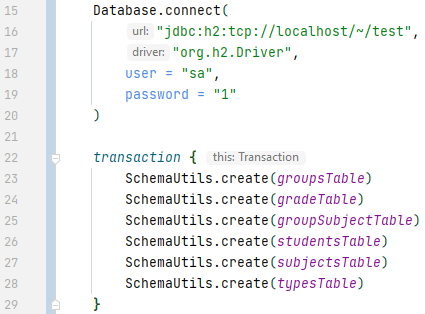


Рисунок 29 – Подключение к базе данных и создание таблиц

## Rest

Для работы rest-сервера был использован Netty. Были установлены заголовки с помощью функции install(), так же для локального подключения к серверу был установлен COR. Маршруты были прописаны в restRepo.



Рисунок 30 – Конфигурация сервера

Функция restRepo принимает в качестве параметров groupRepo, groupSerializer, gradeRepo, gradeSerializer, groupSubjectRepo, groupSubjectSerializer, studentRepo, studentSerializer, subjectRepo, subjectSerializer, typeRepo, typeSerializer.

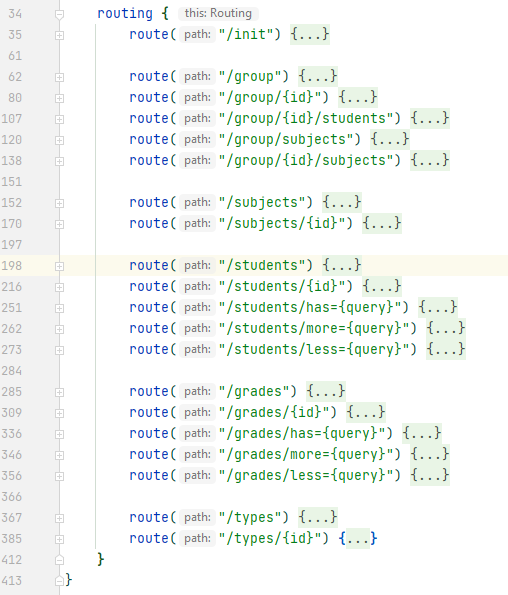


Рисунок 31 – Маршруты

При запросе пользователя GET /group, сервер вернет список всех групп в формате JSON.

При запросе пользователя POST /group, с указанием в теле запроса данных новой группы в формате JSON, будет создана новая запись в groupTable, то есть будет создана новая группа, исходя из переданных пользователем данных.



Рисунок 32 – /group

При запросе пользователя GET /group/id, сервер вернет группу с соответствующим id формате JSON.

При запросе пользователя DELETE/group/id, будет удалена запись с соответствующим id.

При запросе пользователя PUT /group/id, с указанием в теле запроса новых данных для группы в формате JSON, будет изменена запись в groupTable.



Рисунок 33 – /group/id

При запросе пользователя GET /group/id/students, сервер вернет студентов, принадлежащей к группе с соответствующим id, в формате JSON.

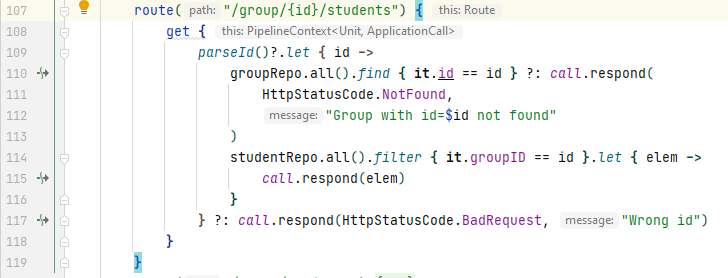


Рисунок 34 – /group/id/students

При запросе пользователя GET /group*/*subjects, сервер вернет все элементы из таблицы groupSubjectTable в формате JSON.

При запросе пользователя POST /group/subjects, с указанием в теле запроса новых данных для GroupSubject в формате JSON, будет изменена запись в groupSubjectsTable.



Рисунок 35 – /group/subjects

При запросе пользователя GET /group/id*/*subjects, сервер вернет все предметы, принадлежащие группе с соответствующим id, в формате JSON.

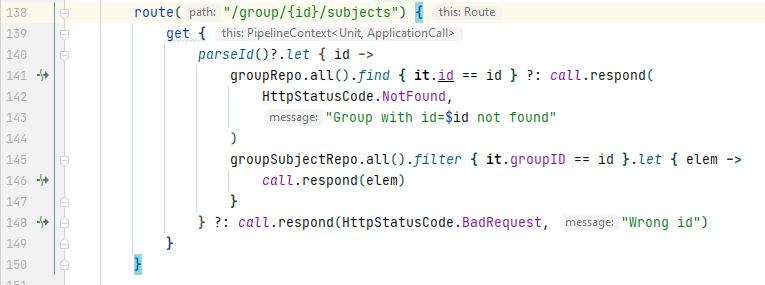


Рисунок 36 – /group/id/subjects

При запросе пользователя GET /subjects, сервер вернет все предметы, из таблицы subjectTable, в формате JSON.

При запросе пользователя POST /subjects, с указанием в теле запроса новых данных для Subject в формате JSON, будет добавлена запись в groupSubjectsTable.



Рисунок 37 – /subjects

При запросе пользователя GET / subjects /id, сервер вернет предмет с соответствующим id формате JSON.

При запросе пользователя DELETE/ subjects /id, будет удалена запись с соответствующим id.

При запросе пользователя PUT / subjects /id, с указанием в теле запроса новых данных для группы в формате JSON, будет изменена запись в subjectsTable.



Рисунок 38 – /subjects/id

При запросе пользователя GET /students, сервер вернет всех студентов, из таблицы studentsTable, в формате JSON.

При запросе пользователя POST /students, с указанием в теле запроса новых данных для Subject в формате JSON, будет добавлена запись в studentsTable.



Рисунок 39 – /students

При запросе пользователя GET / student /id, сервер вернет студента с соответствующим id формате JSON.

При запросе пользователя DELETE/ student /id, будет удалена запись с соответствующим id.

При запросе пользователя PUT / student /id, с указанием в теле запроса новых данных для группы в формате JSON, будет изменена запись в studentsTable.

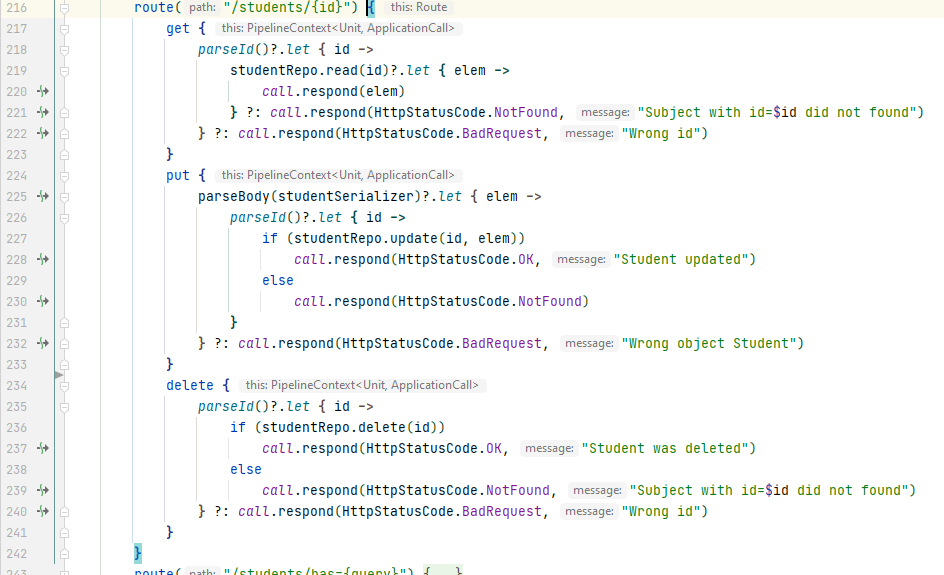


Рисунок 40 – / student /id

При запросе пользователя GET / student /{query} с указателем, сервер вернет студента, соответствующему условию выборки(при query has=n вернет список студентов, имеющие оценку n, при less = m вернет студентов, имеющих оценки меньше m, при more = l вернет студентов, имеющих оценки выше l)в формате JSON.

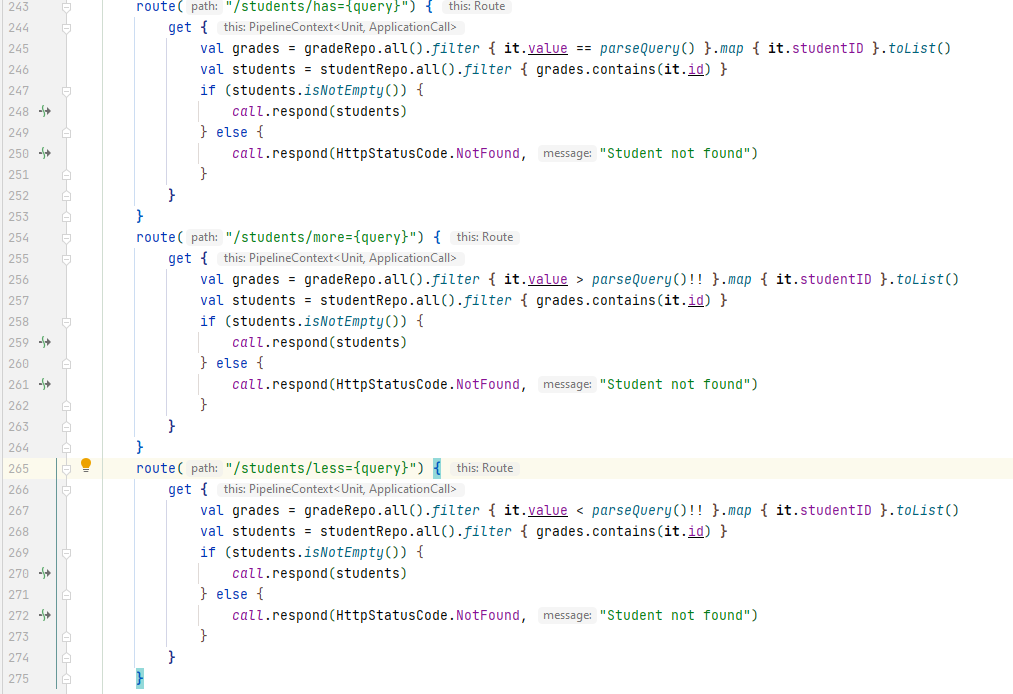


Рисунок 41 – / students/{query}

При запросе пользователя GET /grades, сервер вернет все оценки, из таблицы gradesTable, в формате JSON.

При запросе пользователя POST / grades, с указанием в теле запроса новых данных для Grade в формате JSON, будет добавлена запись в gradesTable.



Рисунок 42 – / grades

При запросе пользователя GET / grades /id, сервер вернет оценку с соответствующим id формате JSON.

При запросе пользователя DELETE/ grades /id, будет удалена запись с соответствующим id.

При запросе пользователя PUT / grades /id, с указанием в теле запроса новых данных для группы в формате JSON, будет изменена запись в gradesTable.

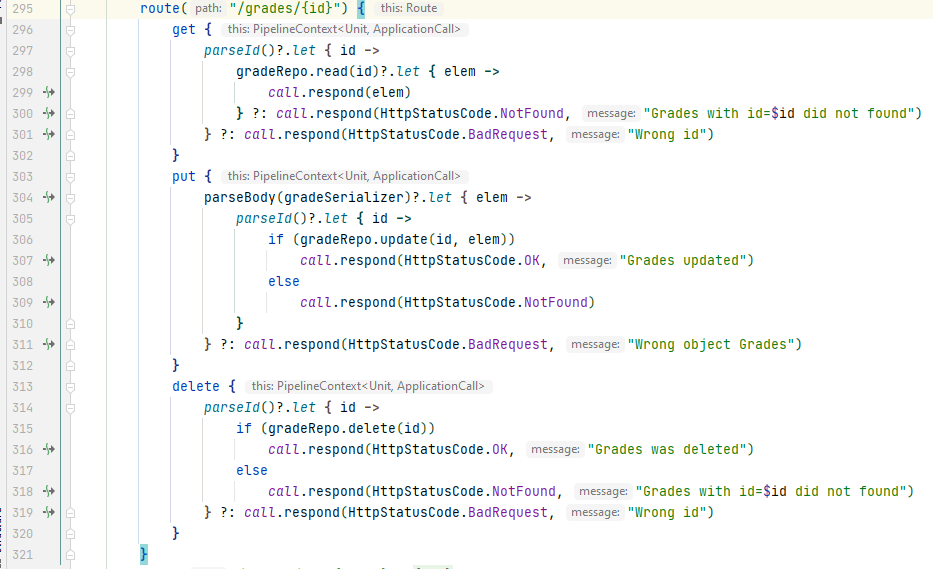


Рисунок 43 – / grades /id

При запросе пользователя GET / grades /{query} с указателем, сервер вернет оценку, соответствующему условию выборки(при query has=n вернет список оценок равных n, при less = m вернет оценки меньше m, при more = l вернет оценки выше l)в формате JSON.



Рисунок 44 – / grades /{query}

При запросе пользователя GET /grades, сервер вернет все типы, из таблицы typesTable, в формате JSON.

При запросе пользователя POST / grades, с указанием в теле запроса новых данных для Type в формате JSON, будет добавлена запись в typesTable.



Рисунок 45 – / types

При запросе пользователя GET / types /id, сервер вернет тип с соответствующим id формате JSON.

При запросе пользователя DELETE/ types /id, будет удалена запись с соответствующим id.

При запросе пользователя PUT / types /id, с указанием в теле запроса новых данных для типа в формате JSON, будет изменена запись в typesTable.

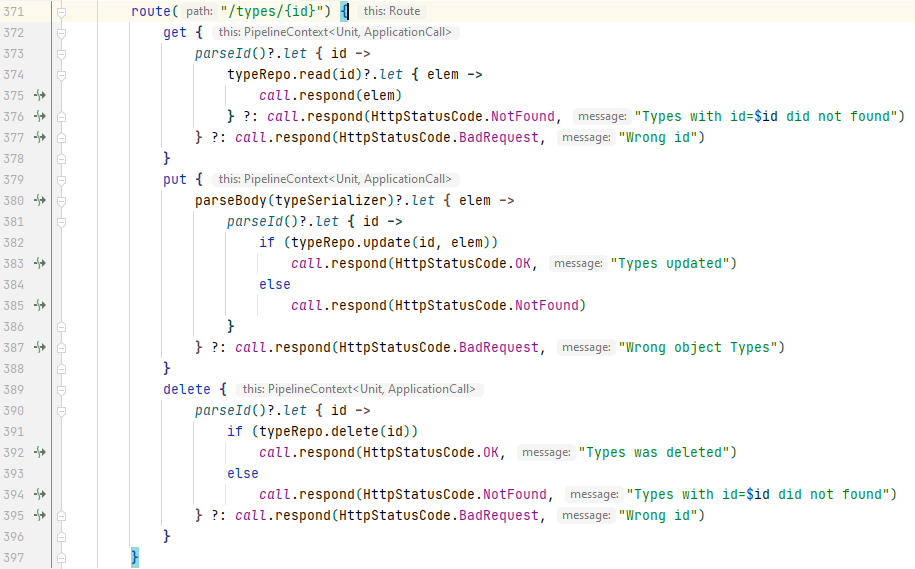


Рисунок 46 – / types /id

# Заключение

В ходе выполнения работы были получены навыки разработки rest сервера на языке программирования kotlin с использованием библиотеки ktor, kotlin exposed, а так же навыки построения uml диаграмм и взаимодействия с технологией ORM.

В курсовой работе был реализован сервер оценок.

# Библиографический список

1. Википедия. Свободная энциклопедия [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма\_(UML)
2. Kotlin exposed [Электронный ресурс] / Режим доступа: [https://github.com/JetBrains/Exposed](https://github.com/JetBrains/Exposed%20)  (Exposed)
3. plantuml [Электронный ресурс] / Режим доступа: [https://plantuml.com/ru/use-case-diagram](https://plantuml.com/ru/use-case-diagram%20)  (UML)

# Приложение А

(обязательное)

REST API

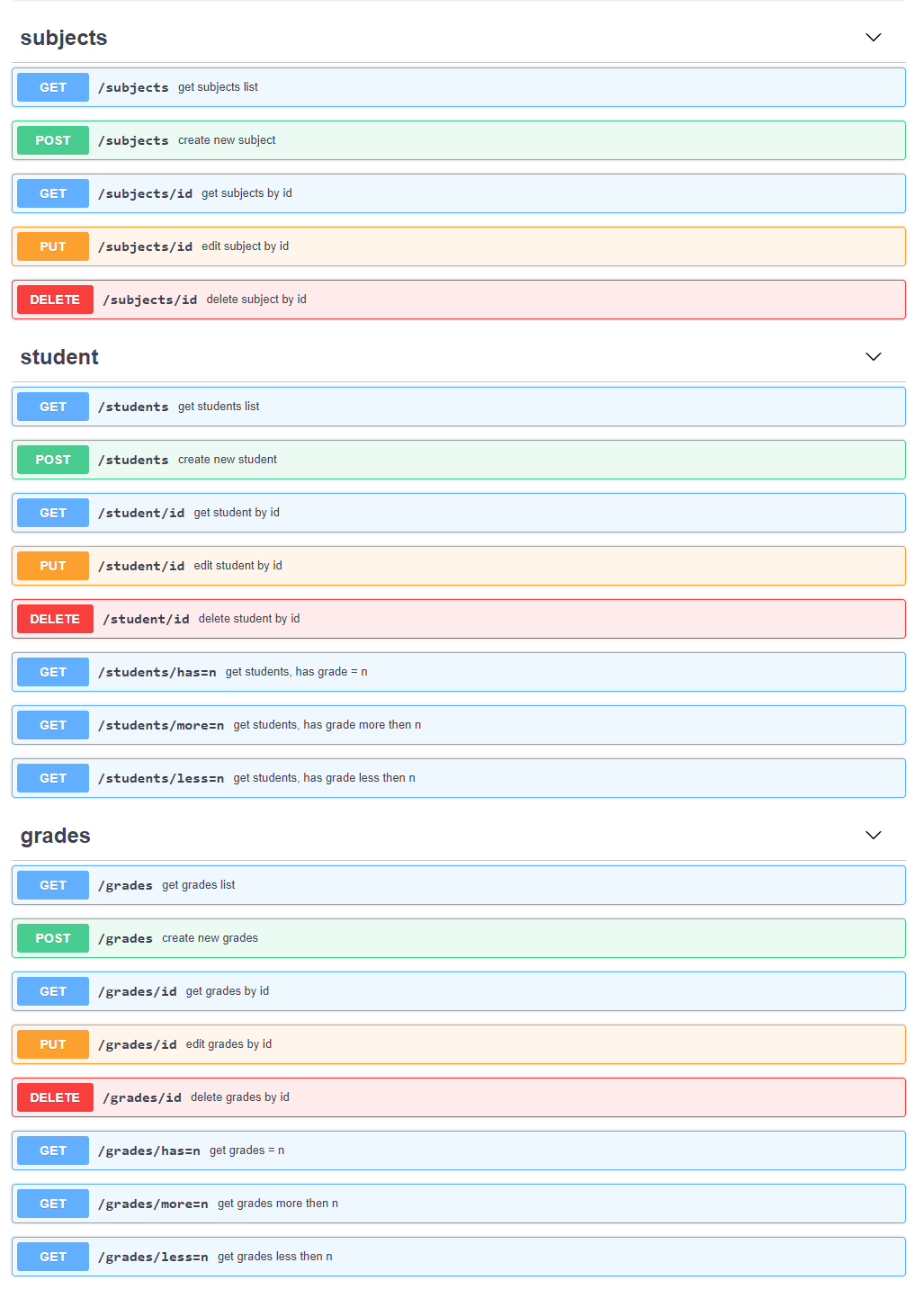


Рисунок А.1 – REST API

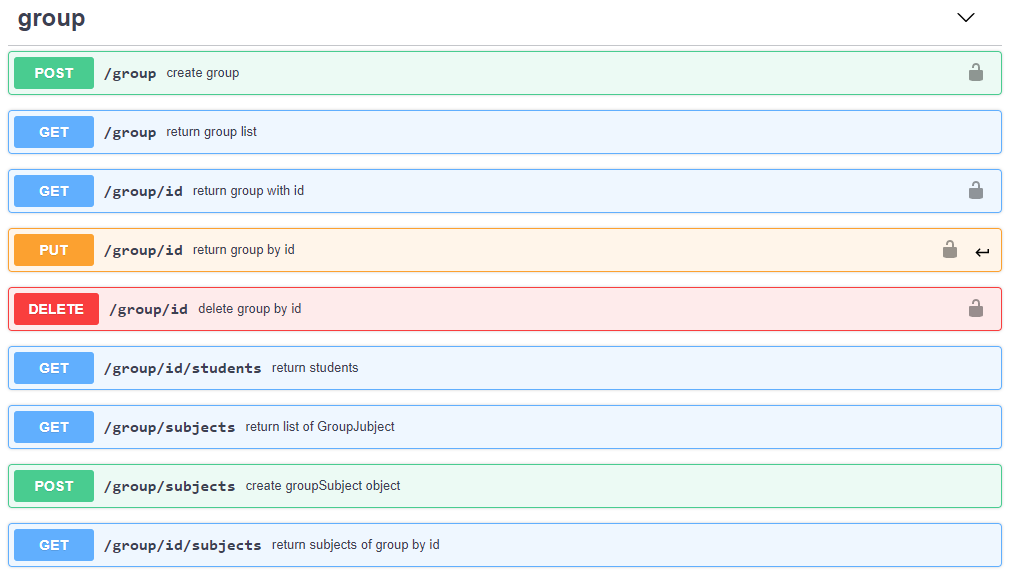


Рисунок А.2 – REST API

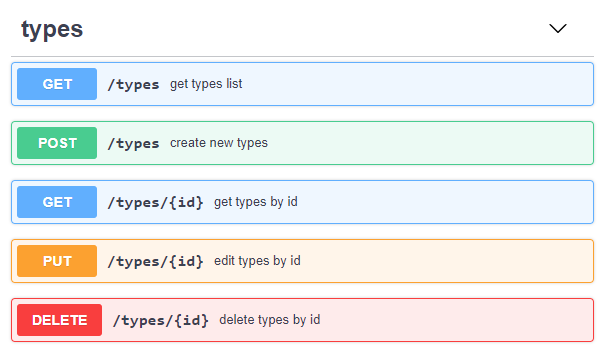


Рисунок А.3 – REST API

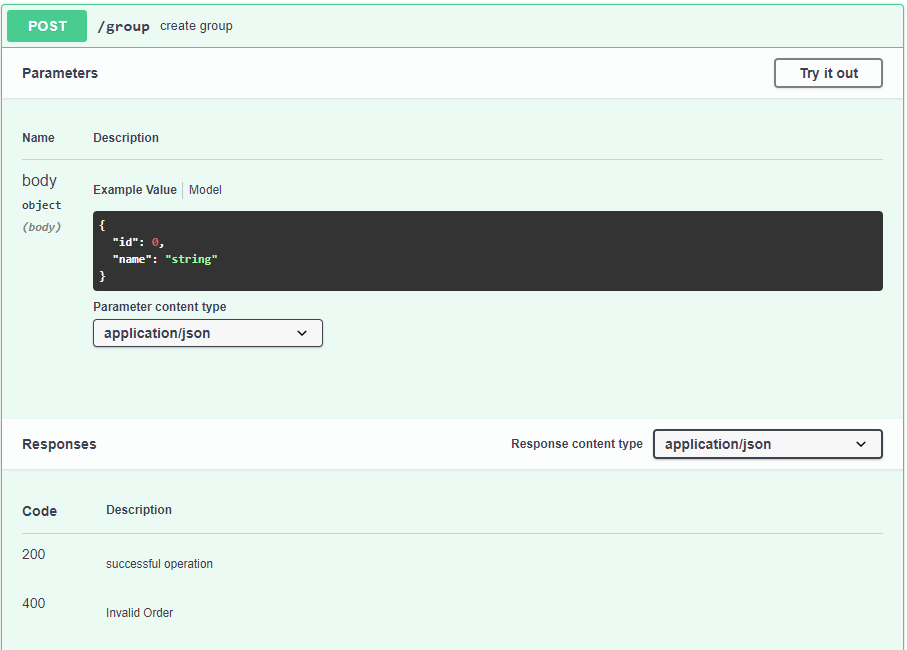


Рисунок А.4 – REST API

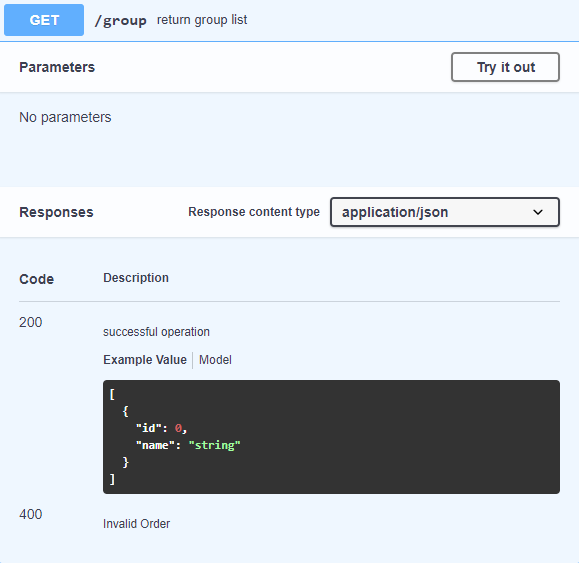


Рисунок А.5 – REST API

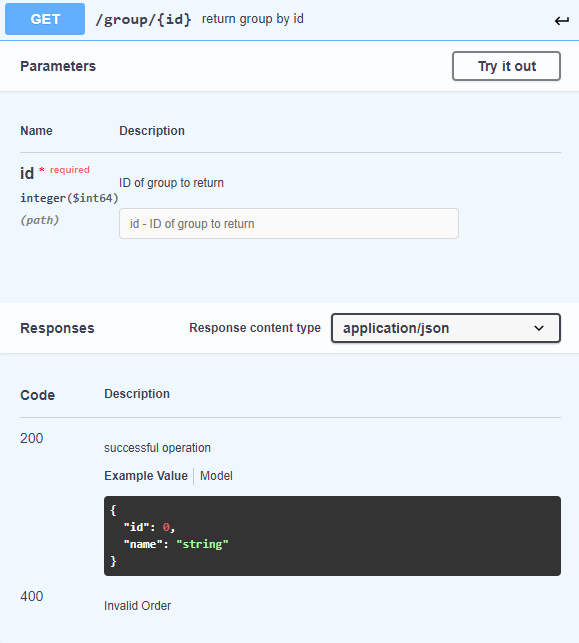


Рисунок А.6 – REST API

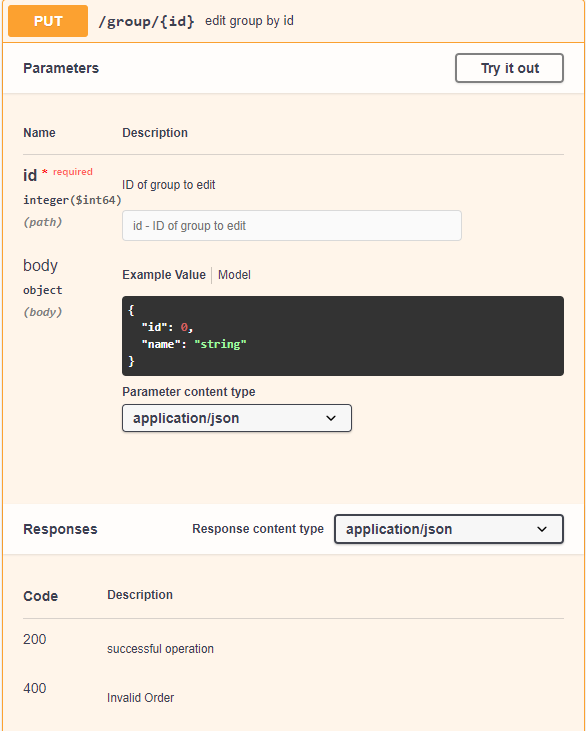


Рисунок А.7 – REST API

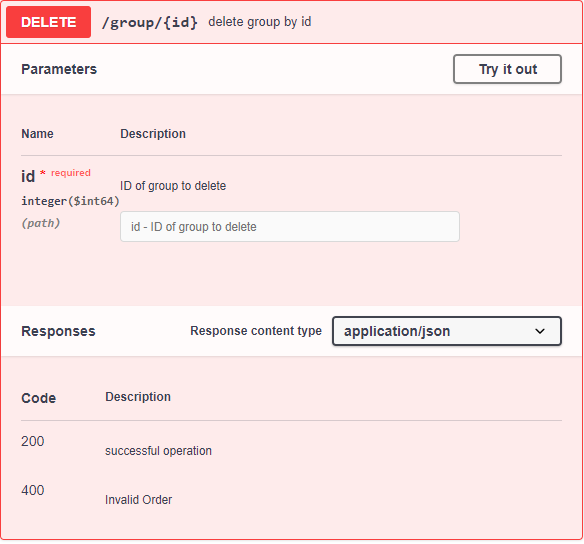


Рисунок А.8 – REST API

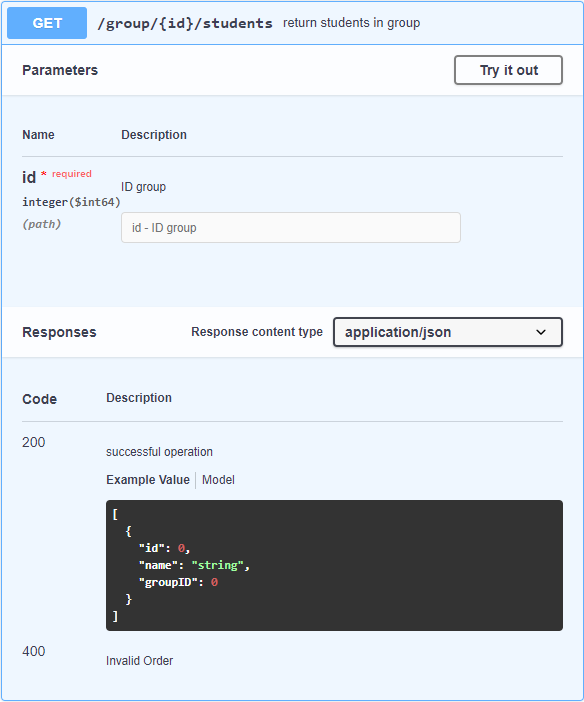


Рисунок А.9 – REST API

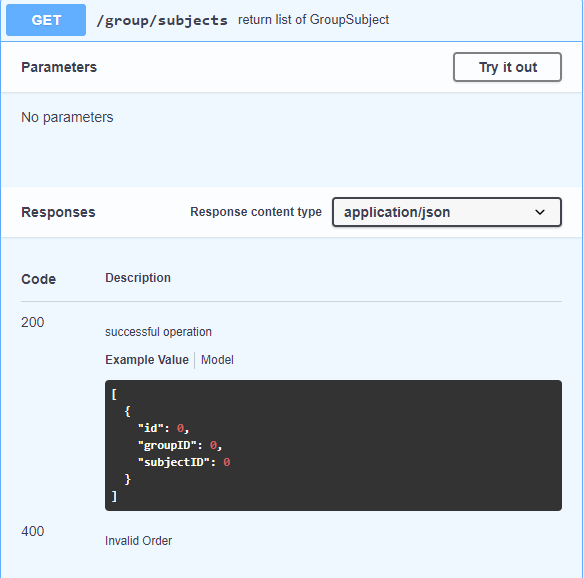


Рисунок А.10 – REST API

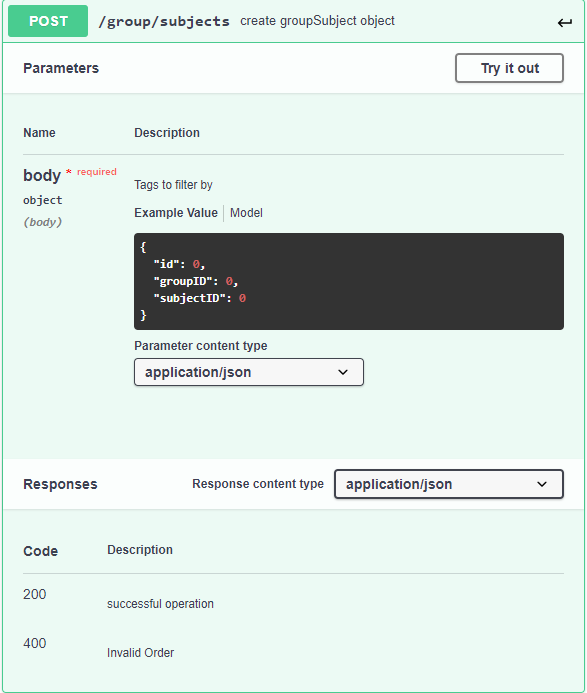


Рисунок А.11 – REST API

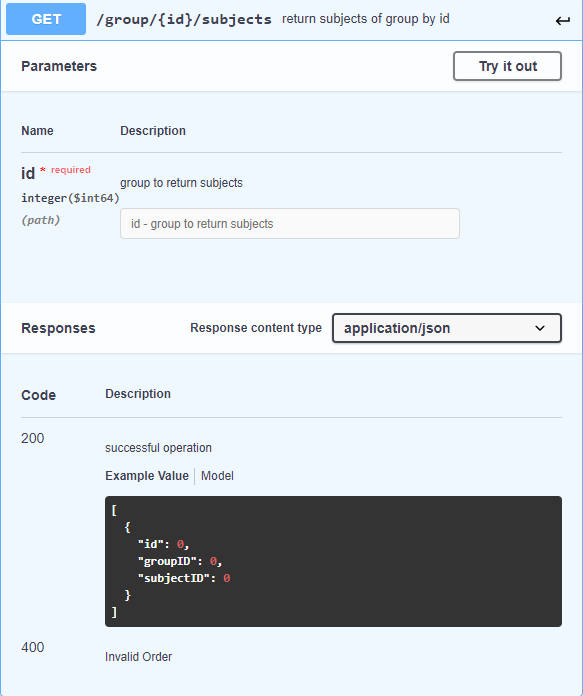


Рисунок А.12 – REST API

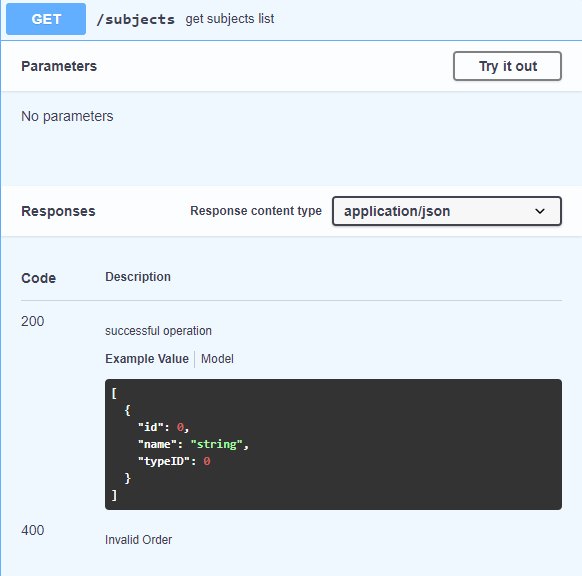


Рисунок А.13 – REST API

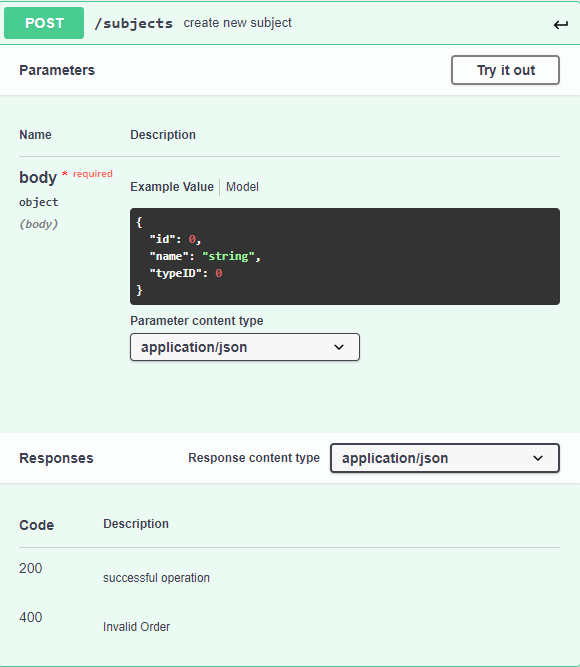


Рисунок А.14 – REST API

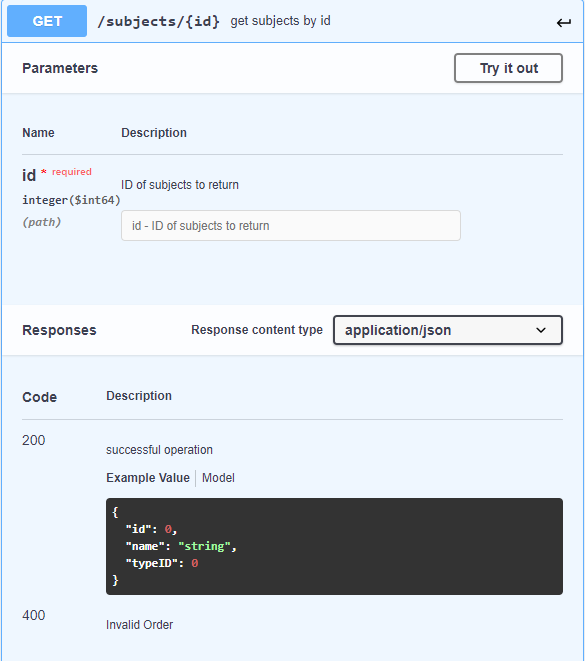


Рисунок А.15 – REST API

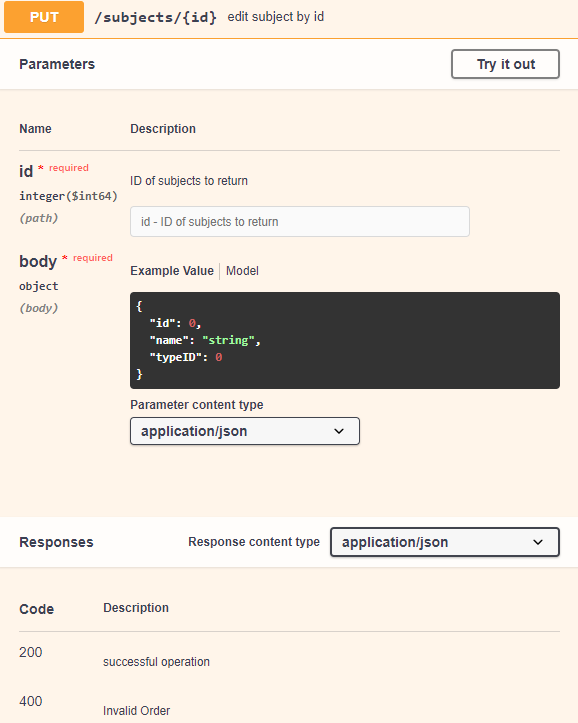


Рисунок А.16 – REST API

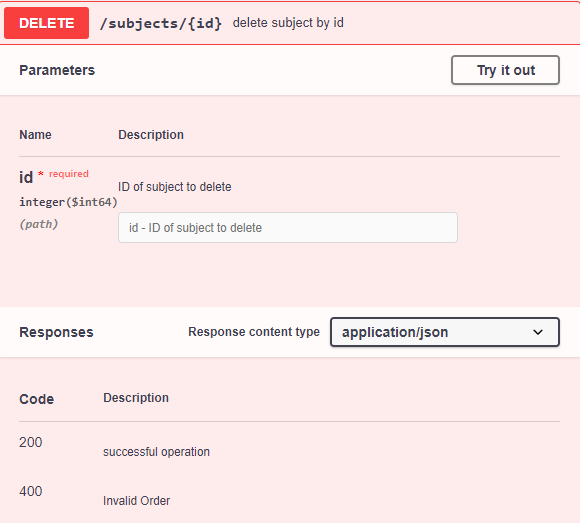


Рисунок А.17 – REST API

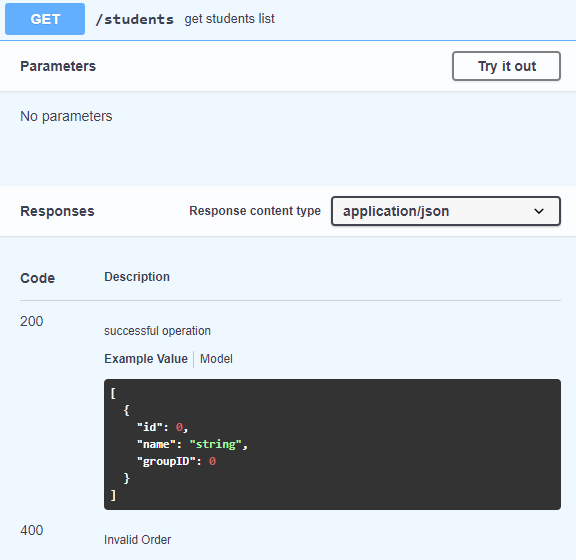


Рисунок А.18 – REST API

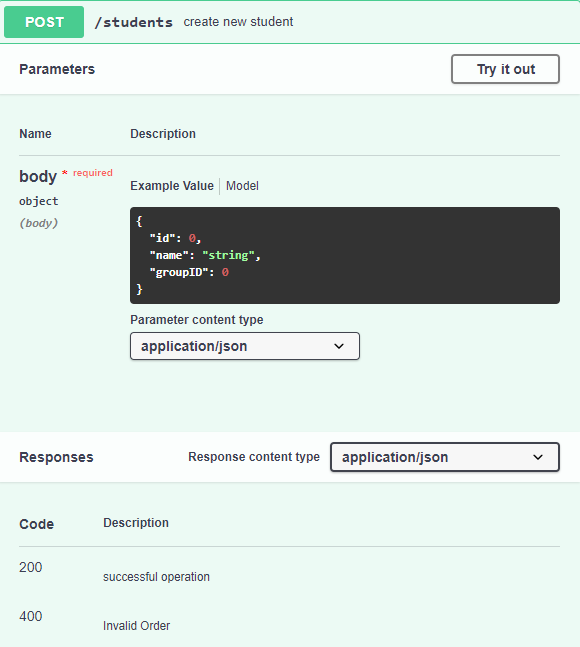


Рисунок А.19 – REST API

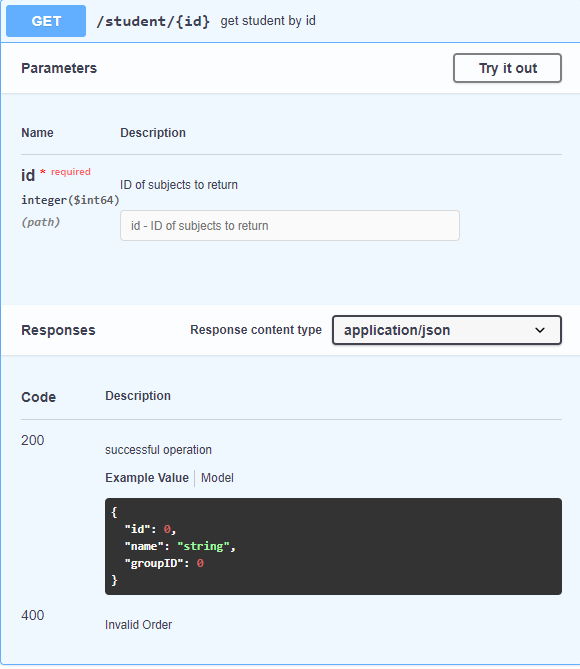


Рисунок А.20 – REST API

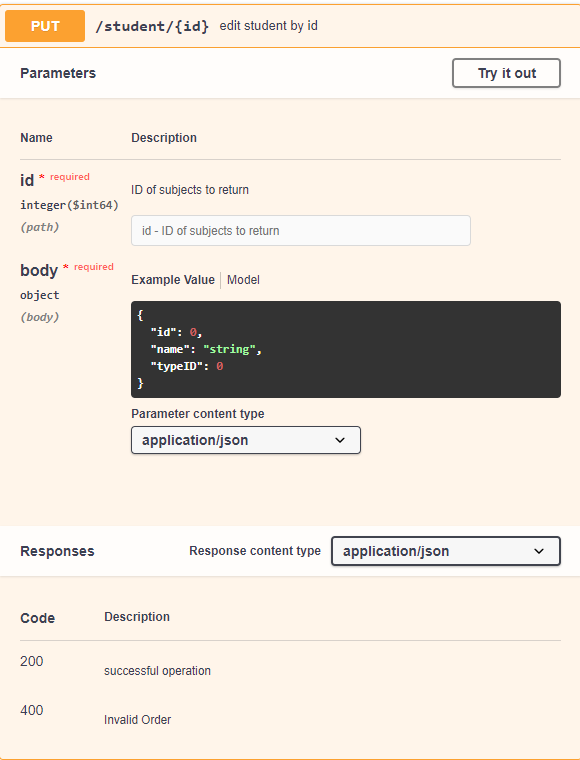


Рисунок А.21 – REST API

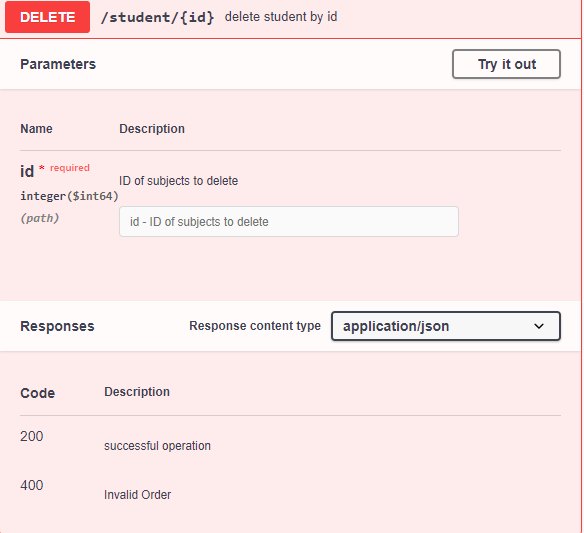


Рисунок А.22 – REST API

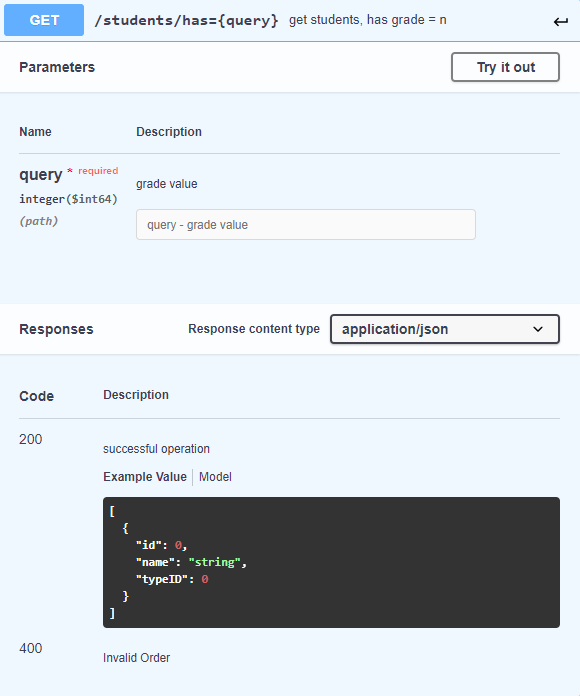


Рисунок А.23 – REST API

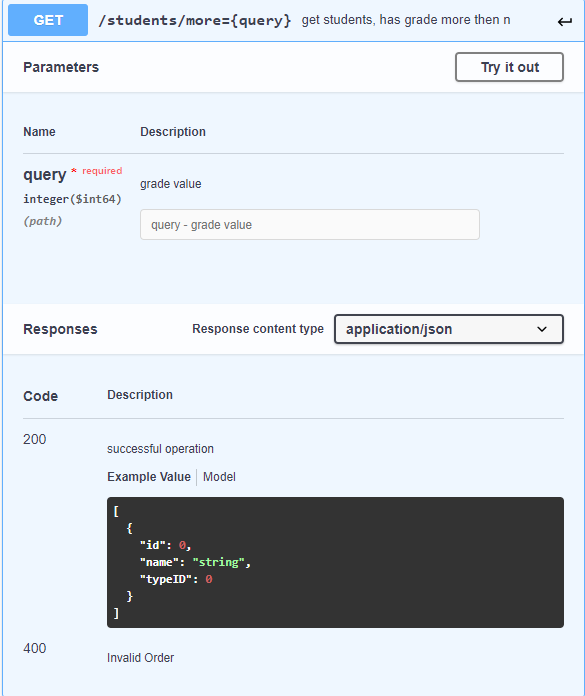


Рисунок А.24 – REST API

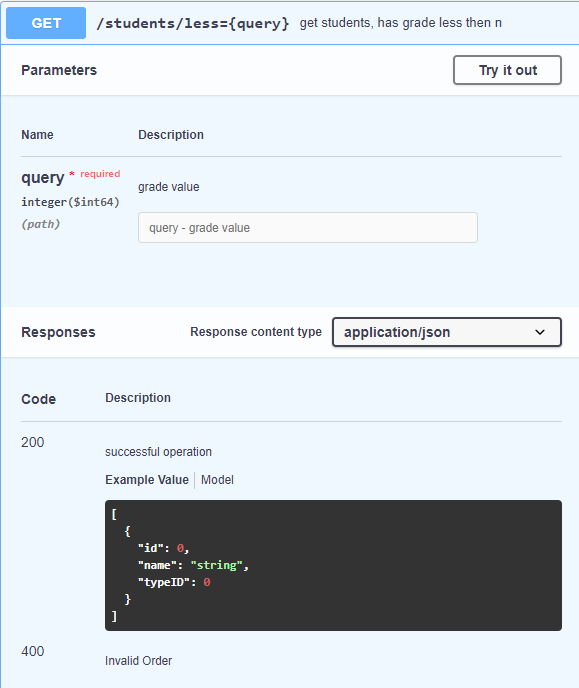


Рисунок А.25 – REST API

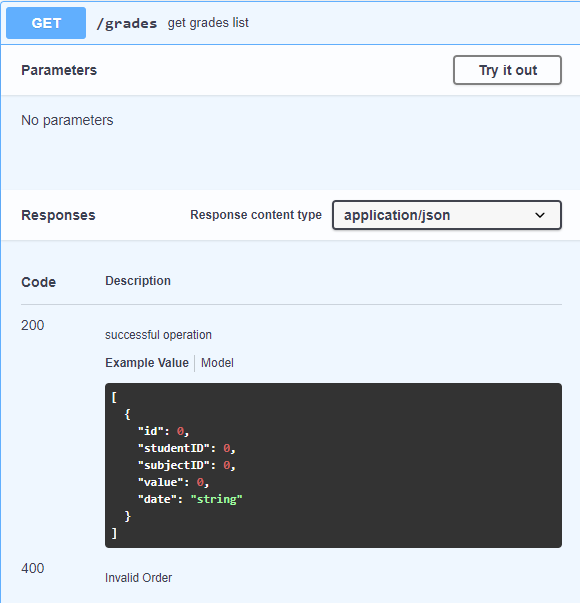


Рисунок А.26 – REST API

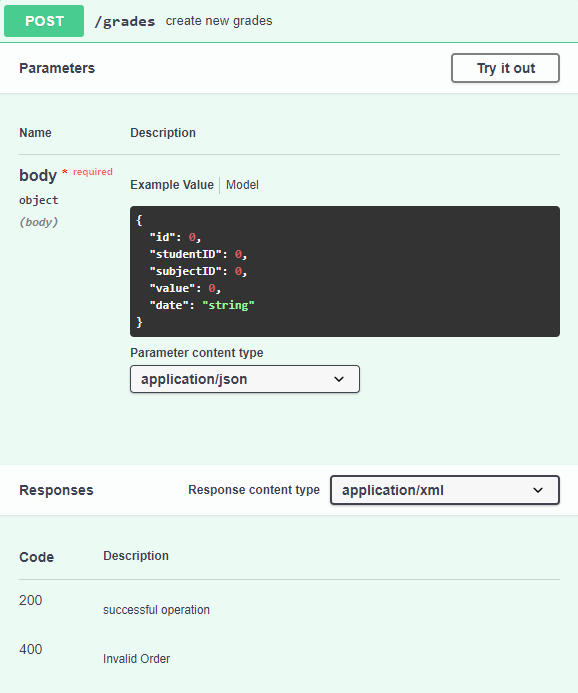


Рисунок А.27 – REST API

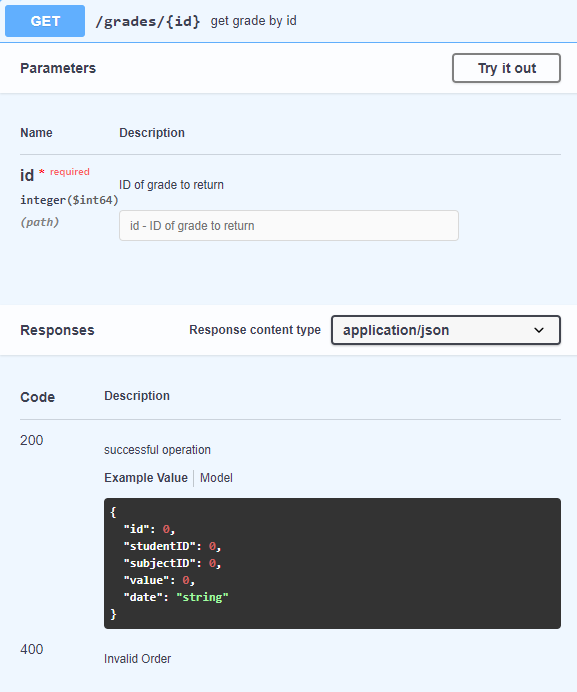


Рисунок А.28 – REST API

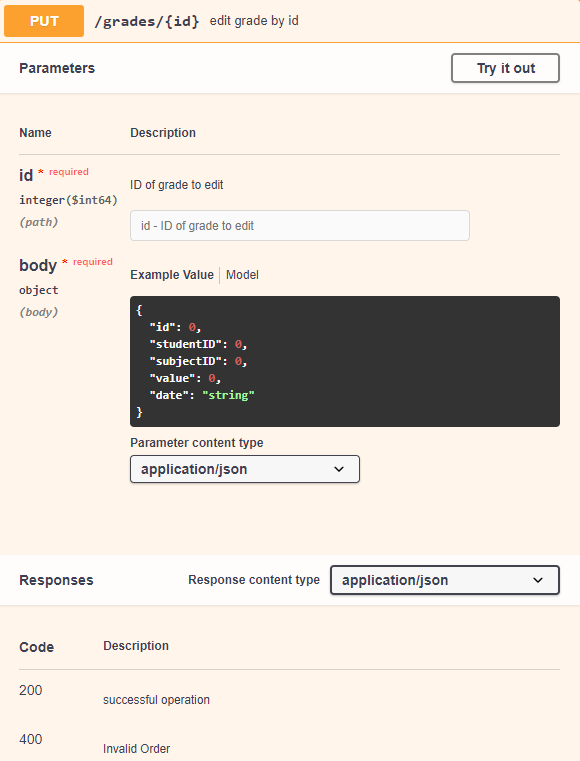


Рисунок А.29 – REST API

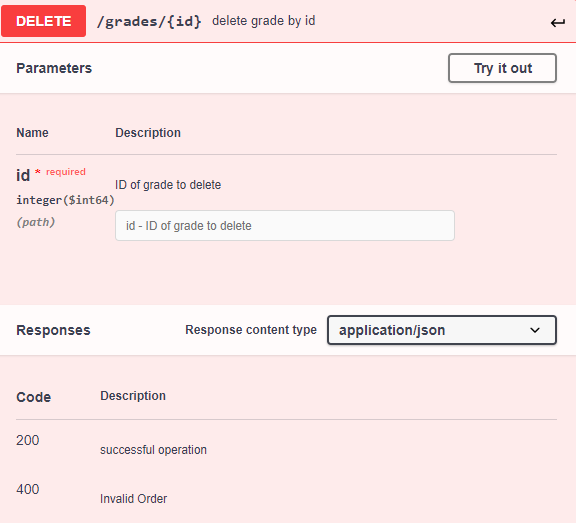


Рисунок А.30 – REST API

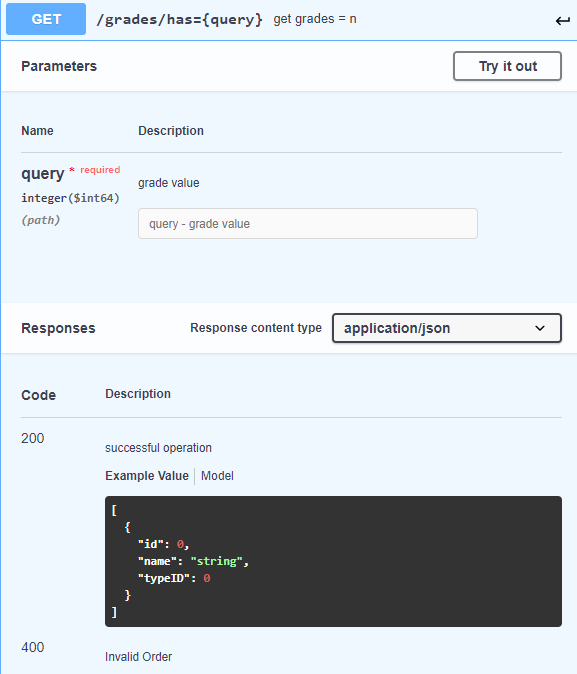


Рисунок А.31 – REST API

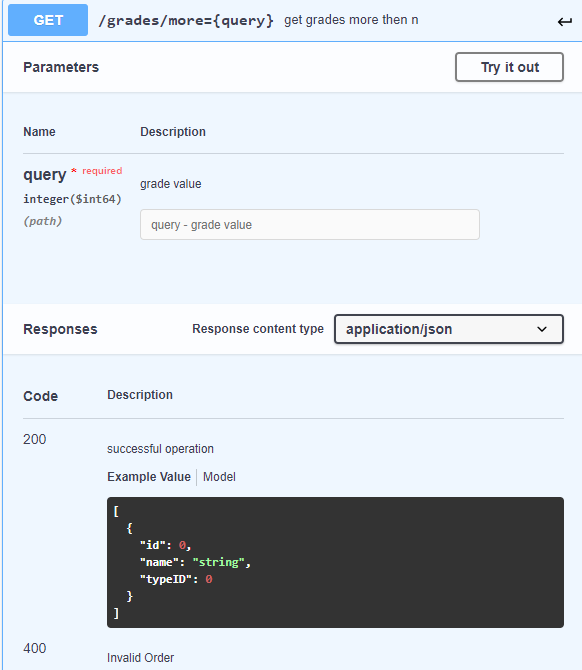


Рисунок А.32 – REST API

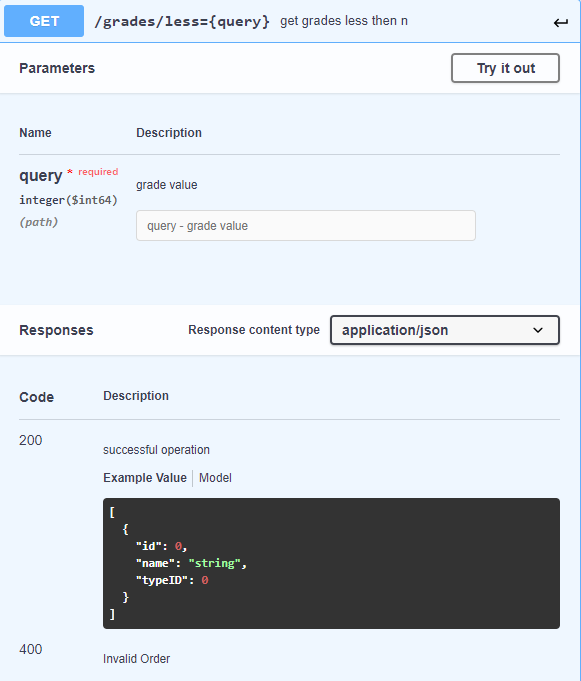


Рисунок А.33 – REST API

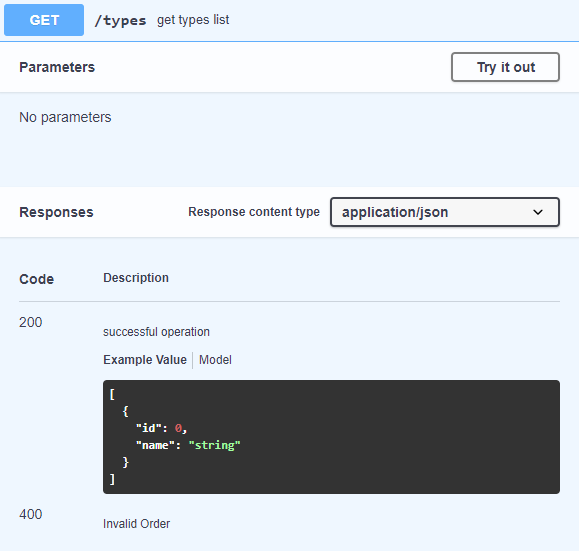


Рисунок А.34 – REST API

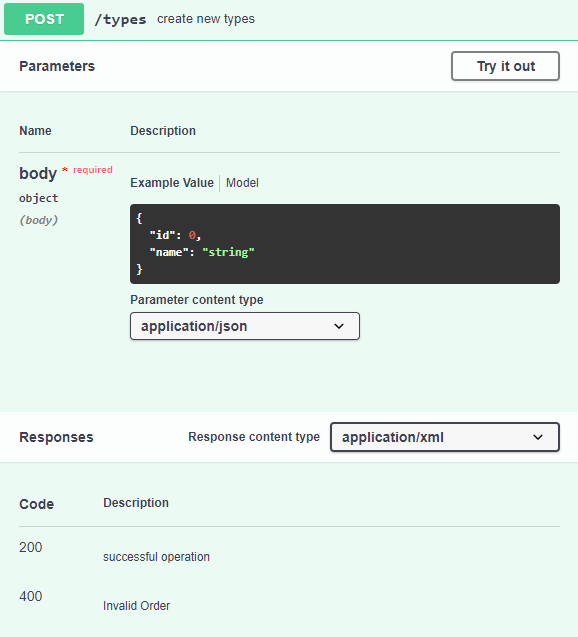


Рисунок А.35 – REST API

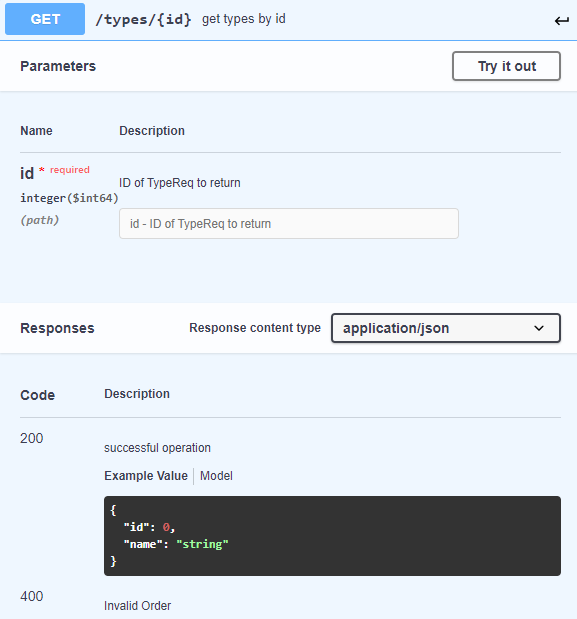


Рисунок А.36 – REST API

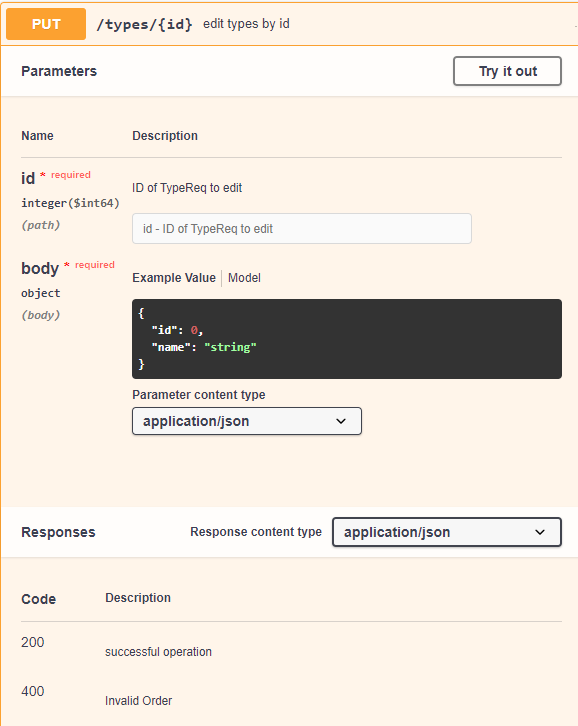


Рисунок А.37 – REST API

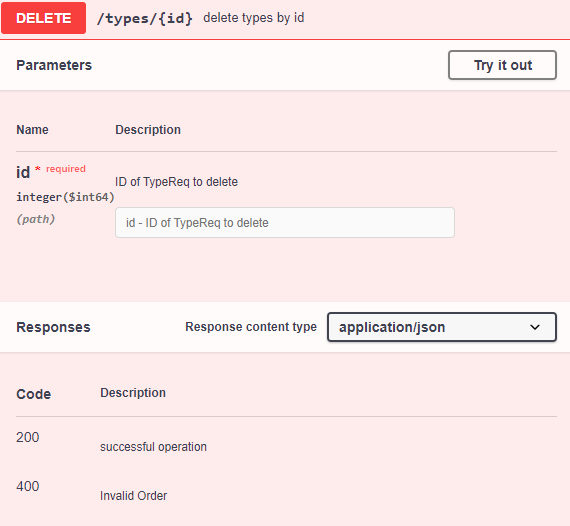


Рисунок А.38 – REST API

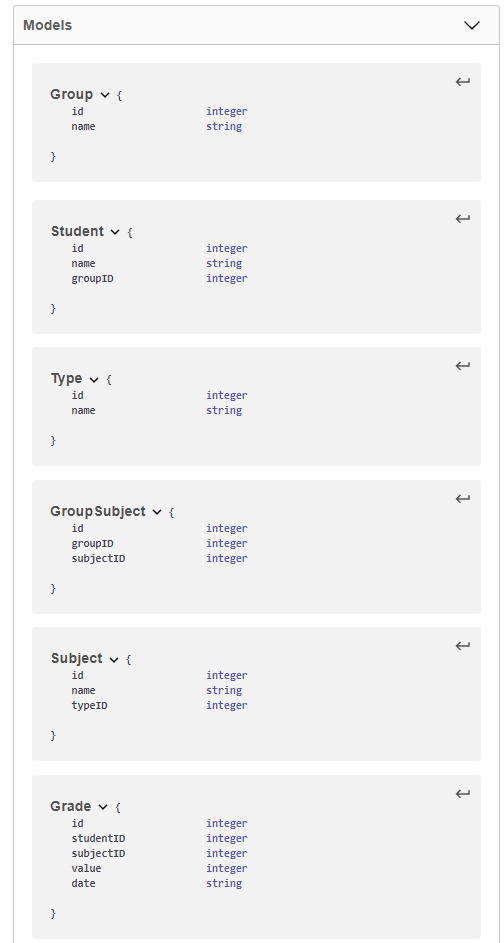


Рисунок А.39 – REST API

Приложение B

(обязательное)

Код проекта

Код проекта доступен по ссылке:

https://github.com/Mmalikov1337/grade\_server