Лабораторная работа 18

ПСКП

ПОИТ-3

**Задание 01**

1. Ознакомьтесь с диаграммой базы данных (БД)



1. Параметры SQL-сервера: ***172.16.193.223, student/fitfit***
2. Скрипт для создания и заполнения таблиц базы данных – ***10.sql***.
3. Создайте собственный экземпляр БД с именем: **XYZ**, где XYZ – инициалы студента. После сдачи лабораторной работы ***БД должна обязательно быть удалена***.

**Задание 02**

1. Разработайте приложение **18-01**, представляющее собой HTTP-сервер, прослушивающий порт ***3000***. Сервер должен обрабатывать запросы, описанные в следующих таблицах. **Для выполнения SQL-запросов используйте пакет sequelize**.

|  |  |  |
| --- | --- | --- |
| Метод  запроса | URI | Назначение |
| GET | / | получить статический HTML-файл, назначение файла описывается в задании 03 |
| GET | api/faculties | получить список всех факультетов в json-формате |
| GET | api/pulpits | получить список всех кафедр в json-формате |
| GET | api/subjects | получить список всех учебных дисциплин в json-формате |
| GET | api/teachers | получить список всех преподавателей в json-формате |
| GET | api/auditoriumstypes | получить список всех типов учебных аудиторий в json-формате |
| GET | api/auditoriums | получить список всех учебных аудиторий в json-формате |
| GET | api/faculties/xyz/subjects | xyz – код факультета, получить за один запрос факультет xyz с его дисциплинами (использовать include) |
| GET | api/auditoriumtypes/xyz/  auditoriums | xyz – код типа аудитории, получить аудитории с типом аудитории xyz (использовать include) |
| POST | api/faculties | добавить новый факультет, данные в json-формате |
| POST | api/pulpits | добавить новую кафедру, данные в json-формате |
| POST | api/subjects | добавить новую учебную дисциплину, данные в json-формате |
| POST | api/teachers | добавить нового препо-давателя, данные в json-формате |
| POST | api/auditoriumstypes | добавить новый тип учебной аудитории, данные в json-формате |
| POST | api/auditoriums | добавить новую учебную аудиторию, данные в json-формате |
| PUT | api/faculties | корректировать информацию о факультете, данные в json-формате |
| PUT | api/pulpits | корректировать информацию о кафедре, данные в json-формате |
| PUT | api/subjects | корректировать информацию об учебной дисциплине, данные в json-формате |
| PUT | api/teachers | корректировать информацию о преподавателе, данные в json-формате |
| PUT | api/auditoriumstypes | корректировать информацию о типе учебной аудитории, данные в json-формате |
| PUT | api/auditoriums | корректировать информацию об аудитории, данные в json-формате |
| DELETE | api/faculties/xyz | xyz - код факультета, удалить факультет с кодом xyz |
| DELETE | api/pulpits/xyz | xyz - код кафедры, удалить кафедру с кодом xyz |
| DELETE | api/subjects/xyz | xyz - код дисциплины, удалить дисциплину с кодом xyz |
| DELETE | api/teachers/xyz | xyz - код преподавателя, удалить преподавателя с кодом xyz |
| DELETE | api/auditoriumtypes/xyz | xyz - код типа аудитории, удалить тип аудитории с кодом xyz |
| DELETE | api/auditoriums/xyz | xyz - код аудитории, удалить аудиторию с кодом xyz |

|  |  |
| --- | --- |
| Метод  запроса | Назначение |
| GET | Возвращает HTML-файл, найденные в БД данные или ***сообщение об ошибке в json-формате*** |
| POST | принимает данные в json-формате,  возвращает добавленные в БД данные или ***сообщение об ошибке в json-формате*** |
| PUT | принимает данные в json-формате,  возвращает измененные в БД данные или ***сообщение об ошибке в json-формате*** |
| DELETE | возвращает удаленные в БД данные или ***сообщение об ошибке в json-формате*** |

1. Обязательно проверяйте входящие данные.
2. Для хранения данных приложение **18-01** должно использовать БД **XYZ.** Для соединения с БД сервердолжен применять пул соединений.
3. Для проверки работоспособности приложения, используйте **POSTMAN**.

**Задание 03**

1. HTML-файл, пересылаемый при запросе **GET /**, должен содержать разметку и JS-код, позволяющий отправлять следующие запросы, получать, обрабатывать и отображать ответы:

**GET api/teachers**

**POST api/teachers**

**PUT api/teachers**

**DELETE api/teachers/xyz**

В случае получения от сервера сообщения об ошибке, это сообщение должно быть отображено.

**Задание 04**

1. Создайте scope для auditoriums для получения всех аудиторий, где вместительность от 10 до 60 человек. Используйте созданный scope в выборке.

**Задание 05**

1. Добавьте хуки beforeCreate и afterCreate к модели faculties.
2. Продемонстрируйте их работу.
3. Добавьте хук по умолчанию для beforeDelete.

**Задание 06**

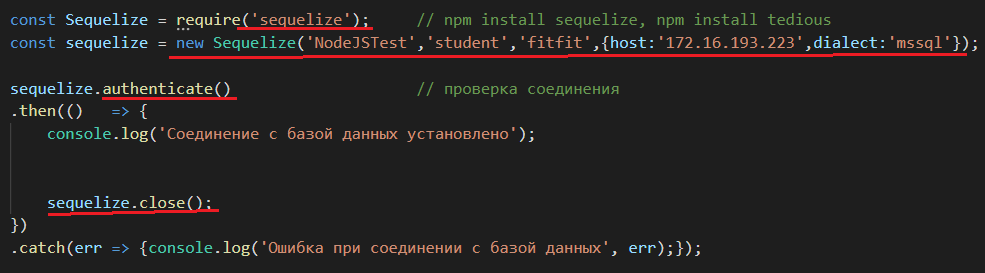
1. Добавьте транзакцию, в которой происходит изменение во всех аудиториях вместительности на 0, а спустя 10 секунд изменения откатываются.
2. Продемонстрируйте работу транзакции.

**Задание 07** Ответьте на следующие вопросы

1. Поясните понятие «ORM»

**ORM – Object Relational Mapping** – технология программирования, которая позволяет работать с SQL базой данных как с набором программных объектов

1. Какой дополнительный пакет необходимо установить для работы sequelize c Microsoft SQL Server?

****

tedious

1. Поясните понятие «модель».

Модели описывают стуктуру хранящихся в бд данных. Кроме того, через модели в основном идет взаимодействие с бд.

1. Какая функция sequelize позволяет выполнить SELECT-запрос?

FindAll()

1. Какая функция sequelize позволяет выполнить INSERT-запрос?

create

1. Какая функция sequelize позволяет выполнить UPDATE-запрос?

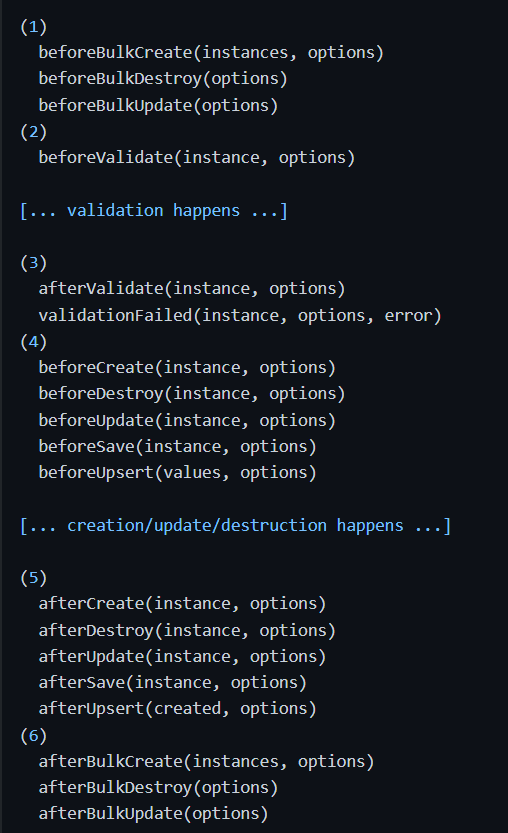
Update()

1. Какая функция sequelize позволяет выполнить DELETE-запрос?

destroy

1. Поясните, что такое «хуки»? Какие они бывают?

Хуки (также известные как события жизненного цикла) - это функции, которые вызываются до и после выполнения вызовов в sequelize.



1. Поясните назначение функции sequelize *Модель*.**hasMany**.

1 ко многим

1. Поясните понятие «raw query».

Для того чтобы выполнить заранее подготовленные запросы

sequelize.query(‘select …’)

1. Поясните понятие «транзакция».

Набор операций, которые либо выполняются все, либо не выполняются вообще