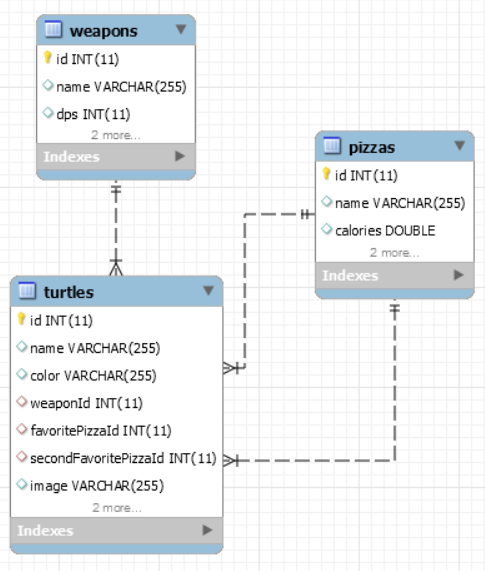
Лабораторная работа 09

ПвИ

**Задание 01**

1. Ознакомьтесь с диаграммой базы данных (БД)



1. Создайте собственный экземпляр БД с именем: **XYZ**, где XYZ – инициалы студента. Можно использовать любую реляционную СУБД, поддерживаемую sequelize.

**Задание 02**

1. Для взаимодействия в БД используйте пакет **sequelize**.
2. Сперва используйте подход database first.
3. Затем удалите БД и для повторного ее создания используйте подход code first.
4. Заполните БД данными.

**Задание 03**

1. Разработайте приложение **09-01**, представляющее собой **EXPRESS**-HTTP-сервер, прослушивающий порт ***3000***.
2. Структура сервера должна соответствовать архитектуре N-layer.
3. Уровень моделей должен содержать описание моделей, соответствующих таблицам в ранее созданной БД, и связей между ними.
4. Уровень сервисов должен содержать бизнес-логику приложения и валидацию данных.
5. Сервис для *черепашек-ниндзя* должен предоставлять следующие операции:
6. Чтение всех черепашек-ниндзя.
7. Чтение черепашки-ниндзя по id, проверка является ли id числом.
8. Чтение всех черепашек-ниндзя, у которых любимая пицца *str* (задается клиентом).
9. Создание новой черепашки-ниндзя, проверять наличие всех полей (поля weaponId, favoritePizzaId, secondFavoritePizzaId, image могут быть пустыми).
10. Обновление черепашки-ниндзя по id с валидацией (описано выше).
11. Удаление черепашки-ниндзя по id.
12. Привязка любимой пиццы (первой и второй).
13. Отвязка любимой пиццы (первой и второй).
14. Привязка оружия.
15. Отвязка оружия.
16. Сервис для *оружия* должен предоставлять следующие операции:
17. Чтение всего оружия.
18. Чтение оружия по id, проверка является ли id числом.
19. Чтение всего оружия, у которого dps больше или меньше *n* (задается клиентом).
20. Создание нового оружия, проверять, чтобы dps был не более 500, а также наличие всех полей.
21. Обновление оружия по id с валидацией (описано выше).
22. Удаление оружия по id.
23. Сервис для *пицц* должен предоставлять следующие операции:
24. Чтение всех пицц.
25. Чтение пиццы по id, проверка является ли id числом.
26. Чтение всех пицц, у которых калорийность больше или меньше *m* (задается клиентом).
27. Создание новой пиццы, проверять, чтобы калорийность была не более 2000, а также наличие всех полей.
28. Обновление пиццы по id с валидацией (описано выше).
29. Удаление пиццы по id.
30. Уровень контроллеров должен принимать запросы клиентов и возвращать ответы.
31. Уровень роутов должен содержать следующие endpoint’ы:

|  |  |
| --- | --- |
| Метод | Маршрут |
| GET | api/turtles |
| GET | api/turtles/id |
| GET | api/turtles?favoritePizza=*str* |
| POST | api/turtles |
| PUT | api/turtles/ |
| PUT | api/turtles/favoritePizzaBind |
| PUT | api/turtles/secondFavoritePizzaBind |
| PUT | api/turtles/weaponBind |
| DELETE | api/turtles/favoritePizzaUnbind |
| DELETE | api/turtles/secondFavoritePizzaUnbind |
| DELETE | api/turtles/weaponUnbind |
| DELETE | api/turtles |
| GET | api/weapons |
| GET | api/weapons/id |
| GET | api/weapons?dps=gt *n*  api/weapons?dps=lt *n* |
| POST | api/weapons |
| PUT | api/weapons |
| DELETE | api/weapons |
| GET | api/pizzas |
| GET | api/pizzas/id |
| GET | api/pizzas?calories=gt *m*  api/pizzas?calories=lt *m* |
| POST | api/pizzas |
| PUT | api/pizzas |
| DELETE | api/pizzas |

|  |  |
| --- | --- |
| Метод  запроса | Назначение |
| GET | Возвращает 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. Для соединения с БД сервердолжен применять *пул соединений*.
2. Для проверки работоспособности приложения, используйте **POSTMAN**.

**Задание 04**

1. Разработайте HTML-страницу, на которой будет выводиться информация обо всех черепашках-ниндзя.
2. Список черепашек должен выводиться по 3 записи на странице. Внизу должны отображаться кнопки с номером страницы.

**Задание 05**

1. Фотографии черепашек-ниндзя хранить на жестком диске в директории images. В БД в поле image хранить полный путь к изображению.
2. В ответ на GET-запрос к /upload высылать HTML-страницу c формой, позволяющей отправить POST-запрос /upload. На форме должны находиться поле для ввода идентификатора черепашки-ниндзя, для которой добавляется изображение, элемент для выбора изображения для загрузки, а также кнопка для отправки формы на сервер.
3. Сервер должен возвращать ошибку, если черепашки-ниндзя с указанным идентификатором не существует, иначе же он должен сохранять файл в директорию images с названием «turtle\_*id*.jpg», где id – указанный идентификатор черепашки-ниндзя, а также добавлять соответствующую запись в БД**.**
4. Фотографии черепашек-ниндзя раздавать с сервера по адресу /images с использованием express.static.
5. В случае если запрошенной фотографии нет, возвращать ошибку 404.

**Задание 06**

1. Добавьте транзакцию, в которой происходит изменение всех пиццы с количеством калорий больше 1500 путем добавления к описанию «SUPER FAT!».
2. Продемонстрируйте работу транзакции.

**Задание 07.** Ответьте на следующие вопросы

1. **Поясните понятие «ORM»**

это техника программирования, позволяющая преобразовывать данные между несовместимыми типами систем, в частности, между объектно-ориентированными языками программирования и реляционными базами данных. ORM позволяет работать с данными в базе данных, используя объекты программирования, не писать прямые запросы SQL.

1. Поясните понятие «модель».

**Модель** в контексте программирования и ORM - это абстракция, представляющая таблицу базы данных в виде класса в программном коде. Объекты этого класса соответствуют записям в таблице. Модели позволяют взаимодействовать с данными в базе данных: создавать, читать, обновлять и удалять записи.

1. Поясните понятие «пул соединений» и его назначение.

**Пул соединений** - это кэш соединений с базой данных, поддерживаемый для обработки запросов к базе данных. Это позволяет избежать затрат на установку и разрыв соединения для каждого запроса. Пул соединений повышает производительность приложений, уменьшая время, необходимое для выполнения операций с базой данных.

1. Что такое Sequelize?

Sequelize - это обещающая ORM для Node.js, поддерживающая реляционные базы данных, такие как PostgreSQL, MySQL, MariaDB, SQLite и MSSQL. Sequelize обеспечивает удобный интерфейс для работы с базами данных, используя JavaScript объекты и асинхронный код.

1. Поясните, что такое «scope» в Sequelize?

**Scope в Sequelize** - это предварительно определенный набор ограничений для модели, который можно легко применять при запросах. Scopes позволяют определять часто используемые запросы на уровне модели для их повторного использования, что упрощает организацию кода и его поддержку.

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

**Хуки в Sequelize** - это функции, которые вызываются в определенные моменты жизненного цикла экземпляра модели (например, перед или после создания, обновления, удаления записей). Хуки позволяют выполнять пользовательский код в эти моменты, например, для валидации или логирования. Они могут быть глобальными (действуют на все модели) или специфичными для модели.

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

Функция Sequelize Модель.hasMany устанавливает связь один ко многим между моделями. Это означает, что одна запись в первичной модели может быть связана с несколькими записями во вторичной модели. Это полезно для представления отношений, как, например, один пользователь имеет множество сообщений.

1. Поясните понятие «raw query» в Sequelize.

**Raw query в Sequelize** - это возможность выполнения прямого SQL-запроса к базе данных, минуя абстракции ORM. Это полезно в случаях, когда нужно выполнить специфический запрос, который трудно или неэффективно выразить средствами ORM. Raw queries предоставляют гибкость, но требуют от разработчика знания SQL и особенностей работы с конкретной базой данных.