Лабораторная работа 10

ПвИ

**Задание 01**

1. Используйте БД из предыдущей лабораторной работы.

**Задание 02**

1. Разработайте приложение **10-01**, представляющее собой HTTP-сервер, прослушивающий порт ***3000***. Сервер должен обрабатывать запросы, описанные в следующих таблицах. **Для выполнения SQL-запросов используйте пакет prisma**.

|  |  |  |
| --- | --- | --- |
| Метод  запроса | URI | Назначение |
| GET | / | получить статический HTML-файл, назначение файла описывается в задании 04 |
| 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 через связанные записи. В результате должно возвращаться следующее: |
| GET | api/auditoriumtypes/xyz/  auditoriums | xyz – код типа аудитории, получить аудитории с типом аудитории xyz через связанные записи. В результате должно возвращаться следующее: |
| GET | api/auditoriumsWithComp1 | Получить список всех компьютерных классов первого корпуса |
| GET | api/puplitsWithoutTeachers | Получить список всех кафедр без преподавателей |
| GET | api/pulpitsWithVladimir | Получить список всех кафедр, на которых есть хотя бы один преподаватель с именем Владимир |
| GET | api/auditoriumsSameCount | Получить количество всех аудиторий с одинаковым типом и вместимостью |
| POST | api/faculties | добавить новый факультет, данные в json-формате. Должна быть возможность вместе с новым факультетом сразу создавать кафедры, которые к нему относятся. Связанные кафедры должны создаваться лишь в том случае, когда они передаются в запросе. |
| POST | api/pulpits | добавить новую кафедру, данные в json-формате. Факультет, к которому относится кафедра передавать в формате «faculty, faculty\_name». В случае, если факультета с указанным кодом faculty не существует, то создавать новый (faculty\_name брать из строки) |
| POST | api/subjects | добавить новую учебную дисциплину, данные в json-формате. |
| POST | api/teachers | добавить нового преподавателя, 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 **10-01** должно использовать БД **XYZ.** Для соединения с БД сервердолжен применять пул соединений.
3. Для проверки работоспособности приложения, используйте **POSTMAN**.

**Задание 03**

1. Напишите какой-то пример использования Fluent API. Объясните, что это такое и для чего применяется.

**Задание 04**

1. Разработайте HTML-страницу, на которой будет выводиться информация обо всех кафедрах.
2. Список кафедр должен выводиться по 10 записей на странице. Внизу должны отображаться кнопки с номером страницы.
3. Рядом с информацией о каждой кафедре должно дополнительно выводиться количество преподавателей на ней (использовать \_count).

**Задание 05**

1. Добавьте транзакцию, в которой происходит изменение во всех аудиториях вместительности на 100 (использовать инкремент), а потом сразу же изменения откатываются.
2. Продемонстрируйте работу транзакции.

**Задание 06** Ответьте на следующие вопросы

1. Что такое Prisma?

Prisma — это современный ORM (Object-Relational Mapping) инструмент, предназначенный для упрощения доступа к базе данных и управления данными в приложениях на Node.js и TypeScript, обеспечивая при этом эффективную работу с базами данных через генерацию типизированных клиентских библиотек.

1. Назовите отличия Prisma от других фреймворков ORM?

* Автоматическая генерация типизированного клиента на основе схемы базы данных.
* Удобный и интуитивно понятный синтаксис запросов.
* Поддержка миграций баз данных с помощью Prisma Migrate.
* Упор на типы данных и их безопасность

1. Какова роль Prisma Client в доступе к данным и управлении ими?

Роль Prisma Client заключается в предоставлении разработчику удобного и мощного API для выполнения запросов к базе данных, автоматически генерируя типизированные запросы на основе схемы базы данных, что обеспечивает безопасность типов и упрощает доступ к данным и их управление.

1. Можно ли использовать Prisma с базами данных SQL и NoSQL?

Использование Prisma с базами данных SQL и NoSQL: Prisma первоначально была сосредоточена на поддержке SQL баз данных (PostgreSQL, MySQL, SQLite, SQL Server), но также экспериментально поддерживает некоторые NoSQL базы данных, такие как MongoDB.

1. Какие языки программирования совместимы с Prisma?

Совместимость Prisma с языками программирования: Прямая поддержка предоставляется для TypeScript и JavaScript, обеспечивая интеграцию с Node.js приложениями.

1. Расскажите, где и как можно использовать ***every***, ***none*** и ***some***?

Эти операторы используются в фильтрации запросов для определения соответствия элементов в массиве определенным условиям. Примеры:

* every: проверяет, удовлетворяют ли все элементы массива определенному условию.
* none: проверяет, что ни один из элементов массива не соответствует условию.
* some: проверяет, что хотя бы один элемент массива соответствует условию.

1. Для чего нужен метод ***upsert***?

Метод upsert используется для выполнения операции вставки новой записи в базу данных, если она не существует, или обновления существующей записи, если она уже существует, в одной операции.

1. Как включить регистронезависимый поиск?

Регистронезависимый поиск можно включить, используя опции запроса, такие как mode: 'insensitive' в параметрах поиска, что позволяет выполнять поиск без учета регистра символов.

1. Объясните назначение методов ***$queryRaw***, ***$executeRaw***.

* $queryRaw: используется для выполнения сырого SQL-запроса, который возвращает данные из базы данных.
* $executeRaw: используется для выполнения сырого SQL-запроса, который не возвращает данные (например, для операций INSERT, UPDATE, DELETE).

1. Какие виды транзакций существуют в Prisma?

* Неявные транзакции: выполняются автоматически для каждого запроса.
* Явные транзакции: позволяют группировать несколько запросов в одну транзакцию, используя метод transaction для контроля над выполнением группы операций как единой транзакции.