АКТ(ф) СПбГУТ

**ОТЧЕТ**

по лабораторным и практическим работам

Технология разработки баз данных

Студент ИСПП-21 19.09.2024 С.А.Миклякова

Преподаватель 19.09.2024 Ю.С.Маломан

Архангельск 2024

# Лабораторная работа №1

**Сбор и анализ требований методом use-case**

1. **Цель работы**
   1. Изучить процесс описания требований к системе методом use-case;
   2. Изучить процесс создания диаграммы вариантов использования.
2. **Ответы на контрольные вопросы**
   1. Диаграмма вариантов использования — это инструмент для визуализации функциональных требований системы. Она описывает взаимодействие между пользователями и системой с точки зрения того, что система должна делать для этих пользователей, а не как она это делает.
   2. Актор — это сущность, которая взаимодействует с системой. Актор находится вне системы и инициирует какое-либо действие в системе или получает от нее информацию. На диаграмме вариантов использования роль актера обозначается в виде стилизованного человечка.
   3. Прецедент — это конкретная функция или цель, которую субъект может достичь, взаимодействуя с системой. Он описывает последовательность действий, выполняемых системой в ответ на запрос субъекта. Прецедент на диаграмме вариантов использования обозначается в виде эллипса. Внутри эллипса обычно пишется имя прецедента, которое должно описывать действие, выполняемое системой.
   4. Отношение ассоциации — это связь между субъектом и прецедентом. Она показывает, что субъект участвует в данном прецеденте.
   5. Отношение обобщения — это иерархическая связь между акторами или прецедентами. Оно показывает, что один актор или прецедент является более общим случаем другого.
   6. Отношение включения — это отношение между двумя прецедентами, при котором один прецедент обязательно включается в другой. Оно показывает, что один прецедент необходим для выполнения другого.
   7. Отношение расширения — это отношение между двумя прецедентами, где один прецедент может быть (но не обязательно) включён в другой. Оно показывает, что один прецедент может дополнять другой при определённых условиях.
3. **Вывод**
   1. В ходе проделанной лабораторной работы был изучен процесс описания требований к системе методом use-case.
   2. В ходе проделанной лабораторной работы был изучен процесс создания диаграммы вариантов использования.

# Лабораторная работа №2

**Проектирование реляционной схемы базы данных в среде СУБД**

1. **Цель работы**
   1. Научиться применять MySQL Workbench в процессе создания схем моделей БД;
   2. Научится представлять логическую модель данных согласно нотациям ERD и IDEF1X.
2. **Ответы на контрольные вопросы**
   1. Сущность (entity) — это реальный или абстрактный объект, который имеет важное значение в контексте предметной области и который необходимо отслеживать в базе данных.
   2. Атрибут (attribute) — это свойство или характеристика сущности. Он описывает определённую особенность сущности и содержит значение данных.
   3. Ключевое поле (key field) — это атрибут или набор атрибутов, значения которых однозначно идентифицируют каждую отдельную сущность.
   4. Первичный ключ (primary key) уникально идентифицирует каждую строку (экземпляр сущности) в таблице. Внешний ключ (foreign key) устанавливает связь между двумя таблицами. Он ссылается на первичный ключ другой таблицы.
   5. Связь (relationship) — это логическая ассоциация между двумя или более сущностями. Она указывает, как сущности связаны друг с другом.
   6. Основные виды связей между сущностями: один-к-одному (one-to-one), один-ко-многим (one-to-many), многие-ко-многим (many-to-many).
   7. Основные элементы ER-диаграммы: сущность, атрибут, первичный ключ, связь, вид связи.
   8. ER-диаграммы применяются для моделирования данных, проектирования баз данных, документирования, планирования, анализа.
3. **Вывод**
   1. В ходе проделанной лабораторной работы научились применять MySQL Workbench в процессе создания схем моделей БД;
   2. В ходе проделанной лабораторной работы научились представлять логическую модель данных согласно нотациям ERD и IDEF1X.

# Лабораторная работа №3

**Приведение БД к нормальной форме**

1. **Цель работы**
   1. Изучить процесс приведения отношений от ненормализованного вида к четвертой нормальной форме;
   2. Изучить процесс декомпозии отношений.
2. **Ответы на контрольные вопросы**
   1. Первичный ключ отношения (primary key) — это атрибут (или набор атрибутов) в таблице (отношении), значения которых однозначно идентифицируют каждую строку (кортеж) в этой таблице.
   2. Внешний ключ отношения (foreign key) — это атрибут (или набор атрибутов) в таблице (отношении), значения которого ссылаются на первичный ключ другой таблицы (отношения). Внешний ключ устанавливает связь между двумя таблицами.
   3. Нормализация отношений (normalization) — это процесс организации данных в базе данных, направленный на уменьшение избыточности и улучшение целостности данных.
   4. Атрибут A функционально зависит от атрибута B (записывается как B -> A), если для каждого значения атрибута B существует только одно значение атрибута A.
   5. Атрибут A транзитивно зависит от атрибута B, если существует атрибут C, такой что:
   6. B→C (B функционально определяет C),
   7. C→A (C функционально определяет A),
   8. в не функционально определяет A (напрямую B не определяет A).
   9. Отношение находится в первой нормальной форме (1НФ), если:
   10. все атрибуты содержат только атомарные значения: то есть каждый атрибут содержит одно неделимое значение (без повторяющихся групп, множественных значений в одном поле или вложенных таблиц),
   11. нет повторяющихся групп: каждая строка должна быть уникальной, и не должно быть повторяющихся блоков данных.
   12. Отношение находится во второй нормальной форме (2НФ), если:
   13. оно находится в 1НФ,
   14. все неключевые атрибуты полностью зависят от всего первичного ключа: то есть каждый неключевой атрибут должен зависеть от всех атрибутов первичного ключа, если первичный ключ составной, а не только от части.
   15. Отношение находится в третьей нормальной форме (3НФ), если:
   16. оно находится во 2НФ,
   17. нет транзитивных зависимостей: все неключевые атрибуты должны зависеть только от первичного ключа, а не от других неключевых атрибутов. То есть не должно быть неключевых атрибутов, которые зависят от других неключевых атрибутов.
3. **Вывод**
   1. В ходе проделанной лабораторной работы был изучен процесс приведения отношений от ненормализованного вида к четвертой нормальной форме;
   2. В ходе проделанной лабораторной работы был изучен процесс декомпозии отношений.

# Лабораторная работа №4

**Установка SQL сервера**

1. **Цель работы**
   1. Научиться устанавливать разные СУБД, используя docker.
2. **Ответы на контрольные вопросы**
   1. Стандартные порты, которые используются для подключения к СУБД:
3. MSSQL (Microsoft SQL Server):

* TCP 1433: основной порт для подключения к SQL Server,
* UDP 1434: используется для службы SQL Server Browser, которая помогает находить экземпляры SQL Server на сервере.

1. MySQL:

* TCP 3306: стандартный порт для подключения к MySQL,
* TCP 5432: стандартный порт для подключения к PostgreSQL.
  1. Стандартные папки для хранения данных СУБД:
     1. MSSQL (Microsoft SQL Server):

1. По умолчанию (Windows):

* C:\Program Files\Microsoft SQL Server\MSSQLxx.MSSQLSERVER\MSSQL\Data (где xx — номер версии SQL Server, например, 15 для SQL Server 2019),
* C:\ProgramFiles\MicrosoftSQLServer\MSSQLxx.MSSQLSERVER\MSSQL\Log (папка для логов).
  1. Настройки пути можно увидеть в SQL Server Management Studio (SSMS) или через запросы к SQL Server
     1. MySQL:

1. По умолчанию (Linux):

* /var/lib/mysql

1. По умолчанию (Windows):

* C:\ProgramData\MySQL\MySQL Server x.x\Data (где x.x - версия MySQL)

1. Файлы конфигурации:

* Linux: /etc/mysql/my.cnf
* Windows: C:\ProgramData\MySQL\MySQL Server x.x\my.ini
* PostgreSQL (Постгрес):

1. По умолчанию (Linux):

* /var/lib/postgresql/<version>/main/ (где <version> — версия PostgreSQL, например, 14)

1. По умолчанию (Windows):

* C:\Program Files\PostgreSQL\<version>\data

1. Файлы конфигурации:

* Linux: /etc/postgresql/<version>/main/postgresql.conf
* Windows: C:\ ProgramFiles\ PostgreSQL\ <version>\ data\ postgresql.conf
  1. Docker — это платформа для создания, развёртывания и управления контейнерами. Контейнер — это стандартизированная единица программного обеспечения, которая объединяет код, библиотеки, зависимости и настройки в одном пакете. Контейнеры позволяют запускать приложения в изолированной среде независимо от операционной системы.

1. **Вывод**
   1. В ходе проделанной лабораторной работы научились устанавливать разные СУБД, используя docker.

# Лабораторная работа №5

**Создание базы данных в среде разработки**

1. **Цель работы**
   1. Изучить способы обеспечения целостности данных в MS SQL Server;
   2. Научиться работать в среде SQL Server Management Studio (SSMS).
2. **Ответы на контрольные вопросы**
   1. SQL Server Management Studio (SSMS) — это интегрированная среда разработки (IDE), разработанная Microsoft для администрирования, разработки и управления базами данных SQL Server.
   2. MS SQL Server поддерживает несколько видов авторизации:
3. Windows Authentication (проверка подлинности Windows),
4. SQL Server Authentication (проверка подлинности SQL Server),
5. Azure Active Directory.
   1. Первичный ключ (primary key) — это атрибут (или набор атрибутов) в таблице, значения которых однозначно идентифицируют каждую строку в этой таблице.
   2. Для указания заполнения столбца автоинкрементными значениями в MS SQL Server используется ключевое слово IDENTITY.
   3. Для указания значения по умолчанию для столбца используется ключевое слово DEFAULT.
   4. Для задания проверочного ограничения используется ключевое слово CHECK.
   5. Для обеспечения уникальности значений в столбце или наборе столбцов используется ключевое слово UNIQUE.
   6. Внешний ключ (foreign key) — это атрибут (или набор атрибутов) в одной таблице, который ссылается на первичный ключ в другой таблице. Он устанавливает связь между двумя таблицами.
   7. Внешний ключ может принимать следующие значения:
6. значение, соответствующее первичному ключу в связанной таблице: это нормальное состояние, когда внешний ключ ссылается на существующую строку в другой таблице,
7. NULL: внешний ключ может быть NULL, если связь является необязательной. Это означает, что в текущей таблице может не быть связанной записи в другой таблице,
8. недействительное значение: (не является нормальным). Внешний ключ может ссылаться на несуществующее значение в другой таблице. Это нарушение ссылочной целостности, и обычно это ошибка в данных или некорректная настройка.
9. **Вывод**
   1. В ходе проделанной лабораторной работы были изучены способы обеспечения целостности данных в MS SQL Server;
   2. В ходе проделанной лабораторной работы было приобретено умение работать в среде SQL Server Management Studio (SSMS).

# Лабораторная работа №6

**Создание представлений в СУБД**

1. **Цель работы**
   1. Научиться создавать и использовать представления в MS SQL Server;
   2. Закрепить навык создания запросов на выборку в MS SQL Server.
2. **Ответы на контрольные вопросы**
   1. Представление (View) в SQL — это виртуальная таблица, которая не хранит данные сама по себе. Вместо этого представление основано на результате запроса SQL к одной или нескольким существующим таблицам. Таблицы хранят данные физически, а представления хранят только логическое определение данных. Таблицы позволяют напрямую управлять данными, а представления предоставляют пользовательский интерфейс для работы с данными.
   2. Представления используются для различных целей, включая:
3. упрощение сложных запросов: представления могут инкапсулировать сложные запросы, позволяя пользователям обращаться к данным с помощью простого имени представления,
4. безопасность данных: представления могут ограничивать доступ к данным, предоставляя пользователям доступ только к нужным столбцам или строкам,
5. согласованность: представления могут предоставлять согласованный интерфейс для доступа к данным, даже если базовая структура данных меняется,
6. реорганизация данных: представления могут перегруппировывать, переименовывать и преобразовывать данные для более удобного использования,
7. использование совместно с другими представлениями: позволяют строить цепочку представлений,
8. сокрытие сложности: скрывает сложную структуру таблиц, позволяя пользователям работать с упрощенным представлением.
   1. Представления могут включать информацию из нескольких таблиц одновременно. Для этого используется оператор JOIN в SQL-запросе, на основе которого создается представление.
   2. Обновляемые представления (то есть те, которые позволяют вставлять, обновлять и удалять данные через представление) имеют ряд ограничений:
9. основано на одной таблице: большинство обновляемых представлений должны быть основаны на одной таблице, (но это не строгое требование, и бывают исключения)
10. простые выборки: представление должно иметь простой запрос без группировки (GROUP BY), агрегатных функций (SUM, COUNT), DISTINCT и UNION,
11. ключевые поля: представление должно включать все первичные ключи базовой таблицы (для обновлений и удалений)
12. НЕ для соединений: если представление основано на нескольких таблицах (через соединение), оно не всегда будет обновляемым, Обновления в таком представлении могут быть неоднозначными,
13. безопасность: обновления могут быть ограничены, если для представления заданы определенные правила безопасности,
14. триггеры: триггеры в базовых таблицах могут влиять на возможность обновления через представление,
15. не все базы данных: не все базы данных полностью поддерживают обновляемые представления.
    1. Создание обновляемого представления зависит от конкретной СУБД. Однако общие принципы схожи. Для создания обновляемого представления, основанного на одной таблице, запрос должен быть простым и включать первичный ключ.

CREATE VIEW ProductsView AS

SELECT

ProductID,

ProductName,

Price

FROM

Products;

1. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение создавать и использовать представления в MS SQL Server;
   2. В ходе проделанной лабораторной работы был закреплен навык создания запросов на выборку в MS SQL Server.

# Лабораторная работа №7

**Создание функций пользователя в СУБД**

1. **Цель работы**
   1. Научиться создавать и использовать скалярные и табличные функции;
   2. Закрепить навык создания запросов на выборку в MS SQL Server.
2. **Ответы на контрольные вопросы**
   1. Пользовательские функции (User-Defined Functions — UDF) — это подпрограммы или блоки кода, которые определяются пользователем для выполнения конкретной задачи в базе данных. Они позволяют расширить функциональность SQL, предоставляя возможность выполнять операции, которые не входят в стандартные возможности SQL.
   2. Скалярные функции – это функции, которые возвращают одно значение заданного типа данных; Табличные функции – это функции, которые возвращают табличные данные, т.е. данные типа TABLE.
   3. Примеры (MS SQL Server):
   4. объявление переменной и присваивание значения

DECLARE @myVariable INT;

SET @myVariable = 10;

* 1. объявление и присваивание при инициализации

DECLARE @myString VARCHAR(255) = 'Hello World';

* 1. Использование переменной

SELECT @myVariable AS MyNumber, @myString AS MyText;

* 1. Для указания параметра по умолчанию в функции используется ключевое слово DEFAULT при объявлении параметра.
  2. Скалярная функция вызывается в запросе так же, как и встроенные функции.

SELECT ProductName,

dbo.CalculatePriceWithDiscount(Price) AS DiscountedPrice

FROM Products;

SELECT dbo.CalculatePriceWithDiscount(100, 0.20);

* 1. Табличная функция вызывается в операторе FROM или JOIN.

1. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение создавать и использовать скалярные и табличные функции;
   2. В ходе проделанной лабораторной работы был закреплен навык создания запросов на выборку в MS SQL Server.

# Лабораторная работа №8

**Создание хранимых процедур в СУБД**

1. **Цель работы**
   1. Научиться создавать и использовать хранимые процедуры в MS SQL Server;
   2. Закрепить навык создания запросов на выборку и модификацию данных в MS SQL Server.
2. **Ответы на контрольные вопросы**
   1. Хранимая процедура — это объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры применяются для повышения производительности, расширения возможностей программирования, поддержки функций безопасности данных.
   2. Хранимая процедура является более гибкой для написания нужного кода, в то время как функции имеют жесткую структуру и функциональность.
   3. Входные параметры (input parameters) хранимной процедуры определяются в ее объявлении:

CREATE PROCEDURE GetProductsByPriceRange (

@minPrice DECIMAL(10,2), -- Входной параметр minPrice

@maxPrice DECIMAL(10,2) -- Входной параметр maxPrice

)

AS

* 1. Выходные параметры (output parameters) хранимной процедуры определяются в ее объявлении с ключевым словом OUTPUT.
  2. Вызов без выходных параметров

EXEC GetProductsByPriceRange @minPrice = 10, @maxPrice = 50;

-- Вызов с выходными параметрами

DECLARE @count INT;

EXEC GetProductCount @category = 'Electronics', @productCount = @count OUTPUT;

SELECT @count AS TotalProducts;

1. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение создавать и использовать хранимые процедуры в MS SQL Server;
   2. В ходе проделанной лабораторной работы был закреплен навык создания запросов на выборку и модификацию данных в MS SQL Server.

# Лабораторная работа №9

**Создание триггеров в СУБД**

1. **Цель работы**
   1. Научиться создавать и использовать триггеры в MS SQL Server;
   2. Закрепить навык создания запросов на выборку и модификацию данных в MS SQL Server.
2. **Ответы на контрольные вопросы**
   1. Триггер (Trigger) — это особый вид хранимой процедуры, которая автоматически выполняется в ответ на определенное событие в базе данных. Триггеры обычно связаны с определенной таблицей и выполняются при вставке (INSERT), обновлении (UPDATE) или удалении (DELETE) данных в этой таблице.
   2. Триггеры — это автоматические «сторожа», которые срабатывают при возникновении событий в базе данных. Хранимые процедуры — это вызываемые программные блоки для выполнения определённых действий.
   3. Триггер нельзя запустить на выполнение напрямую. Он выполняется автоматически при возникновении определенного события, которое было указано при его создании.
   4. Основные назначения триггеров:
3. обеспечение целостности данных: триггеры могут гарантировать, что данные соответствуют определенным правилам и ограничениям,
4. аудит: триггеры могут записывать все изменения в таблицах (журнал изменений),
5. автоматизация действий: триггеры могут автоматически выполнять определенные действия в ответ на события, такие как:
6. обновление связанных таблиц,
7. отправка уведомлений,
8. проверка данных перед их добавлением/изменением,
9. реализация сложных бизнес-правил: триггеры могут реализовывать сложные бизнес-правила, которые трудно реализовать другими способами,
10. поддержка ссылочной целостности: триггеры могут каскадно обновлять или удалять связанные записи, если родительские записи были обновлены или удалены,
11. управление доступом: триггеры могут контролировать доступ к таблицам, ограничивая действия пользователя на основе определённых правил.
    1. Триггеры типа AFTER вызываются после выполнения действия, запускающего триггер, а триггеры типа INSTEAD OF выполняются вместо действия, запускающего триггер.
12. **Вывод**
    1. В ходе проделанной лабораторной работы было приобретено умение создавать и использовать триггеры в MS SQL Server;
    2. В ходе проделанной лабораторной работы был закреплен навык создания запросов на выборку и модификацию данных в MS SQL Server.

# Лабораторная работа №10

**Реализация доступа пользователей к базе данных**

1. **Цель работы**
   1. Научиться использовать системные хранимые процедуры и DDL-команды для управления именами входа и пользователями БД в СУБД;
   2. Научиться назначать привилегии пользователю БД;
   3. Закрепить навык создания объектов БД.
2. **Ответы на контрольные вопросы**
   1. Имя входа — это ключ от входной двери здания. Позволяет войти в здание. Пользователь БД — это ключ от конкретной комнаты. Позволяет получить доступ к конкретной комнате и что-то в ней делать.
   2. Пользователи идентифицируются в MS SQL Server на двух уровнях:
3. идентификация на уровне сервера (Authentication),
4. идентификация на уровне базы данных (Authorization).
   1. Система безопасности MS SQL Server делится на уровень экземпляра (сервера) и уровень базы данных.
   2. Роли сервера (Server Roles) используются для управления разрешениями на уровне сервера. Роли сервера предоставляют общие права на выполнение определенных задач, связанных с администрированием сервера.
   3. Роли базы данных (Database Roles) используются для управления разрешениями на уровне базы данных. Роли базы данных предоставляют права на выполнение определенных операций с данными и объектами в конкретной базе данных.
5. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение использовать системные хранимые процедуры и DDL-команды для управления именами входа и пользователями БД в СУБД;
   2. В ходе проделанной лабораторной работы было приобретено умение назначать привилегии пользователю БД;
   3. В ходе проделанной лабораторной работы был закреплен навык создания объектов БД.

# Лабораторная работа №11

**Выполнение резервного копирования и восстановления БД**

1. **Цель работы**
   1. Научиться выполнять резервное копирование БД;
   2. Научиться сохранять во внешних файлах описание структуры и данные БД;
   3. Научиться выполнять восстановление БД.
2. **Ответы на контрольные вопросы**
   1. Резервные копии баз данных создаются для защиты данных и обеспечения возможности восстановления базы данных в случае возникновения проблем.
   2. Полное резервное копирование (Full Backup):
3. сохраняет все данные и объекты базы данных целиком,
4. создает автономную резервную копию, из которой можно восстановить базу данных,
5. требует больше времени и места для создания,
6. является основой для других видов резервного копирования,
7. обеспечивает наиболее простое восстановление.

Разностное резервное копирование (Differential Backup):

1. сохраняет только те данные, которые были изменены с момента последнего полного резервного копирования,
2. создает зависимую резервную копию, которая требует наличия последней полной резервной копии для восстановления,
3. требует меньше времени и места для создания по сравнению с полным,
4. позволяет быстрее восстанавливать данные, чем при использовании только полных резервных копий,
5. имеет смысл, если данные меняются часто.
   1. Расписание резервного копирования зависит от многих факторов, включая частоту изменения данных, критичность данных, требования к восстановлению и доступные ресурсы.
   2. Для формирования скриптов для объектов базы данных можно использовать следующие методы:
6. мастер создания сценариев. позволяет генерировать сценарии для всей базы данных или для конкретных объектов базы данных (базы данных, таблицы, хранимых процедур и т. д.),
7. опция «SCRIPT AS» (без данных). позволяет создавать скрипты для объектов базы данных, используя опцию «SCRIPT OBJECT AS». в случае базы данных опция «SCRIPT DATABASE AS» воссоздаёт её и параметры её конфигурации.
   1. Восстановление базы данных из резервной копии обычно выполняется с помощью SQL Server Management Studio (SSMS) или с помощью команд SQL.
   2. Для выполнения восстановления базы данных используется команда RESTORE DATABASE в T-SQL.
   3. Резервные копии восстанавливаются в следующем порядке:
8. полное резервное копирование: восстанавливается сначала последняя полная резервная копия,
9. разностное резервное копирование: затем восстанавливается последнее разностное резервное копирование, относящееся к восстановленной полной копии (если таковая имеется),
10. файлы журналов транзакций: после восстановления полной и разностной копий последовательно восстанавливаются файлы журналов транзакций для восстановления базы данных на конкретный момент времени.
    1. RECOVERY: Этот параметр запускает восстановление базы данных и переводит ее в рабочее состояние после выполнения всех этапов восстановления. NORECOVERY: Этот параметр отключает восстановление базы данных после применения резервных копий. База данных остается в режиме восстановления, что позволяет применять дополнительные резервные копии журналов транзакций. База данных не может использоваться в этом режиме.
    2. Мастер импорта и экспорта SQL Server (SQL Server Import and Export Wizard) — это инструмент для перемещения данных между различными источниками и приёмниками данных.
11. **Вывод**
    1. В ходе проделанной лабораторной работы было приобретено умение выполнять резервное копирование БД;
    2. В ходе проделанной лабораторной работы было приобретено умение сохранять во внешних файлах описание структуры и данные БД;
    3. В ходе проделанной лабораторной работы было приобретено умение выполнять восстановление БД.

# Лабораторная работа №12

**Экспорт данных базы в документы пользователя**

1. **Цель работы**
   1. Научиться выполнять экспорт данных из БД.
2. **Ответы на контрольные вопросы**
   1. Для экспорта данных SQL Server в CSV-файл в SQL Server Management Studio нужно выполнить следующие шаги:
3. откройте обозреватель объектов,
4. щёлкните правой кнопкой мыши по базе данных и выберите «задачи» — «экспортировать данные»,
5. в открывшемся мастере импорта и экспорта введите необходимую информацию (имя сервера, база данных и т, д,) и нажмите «далее»,
6. выберите «назначение плоского файла» в качестве пункта назначения и укажите путь к файлу,
7. выберите название таблицы и запятую в качестве разделителя, нажмите «далее»,
8. в окне «сохранения и запуска пакета» изменений не требуется, нажмите «далее»,
9. в окне мастера «завершить» проверьте все настройки, установленные в процессе экспорта,
10. нажмите «готово», чтобы начать экспорт базы данных sql в csv.
    1. Для экспорта данных SQL Server в xlsx-файл в SQL Server Management Studio нужно выполнить следующие шаги:
11. шелкнуть базу данных правой кнопкой мыши,
12. выбрать «Задачи»,
13. выбратт «Импортировать данные» или «Экспортировать данные».
    1. Чтобы открыть в Excel файлы формата txt/csv нужно:
14. открыть Excel,
15. выбрать «Файл»→«Открыть»,
16. найти и выбрать файл,
17. следовать инструкциям мастера импорта, если потребуется.

Чтобы открыть в Excel файлы формата xml нужно:

1. В Excel выбрать «Файл»→«Открыть»,
2. найти и выбрать XML-файл.
   1. В файле CSV можно хранить данные из одной или нескольких таблиц. Обычно, если экспортируются данные из нескольких таблиц, каждая таблица представляет собой отдельный раздел в файле CSV, или же это может быть один или несколько файлов CSV.
   2. Как и в случае с JSON, для экспорта данных в XML в SQL Server обычно используется SQL-запрос. Вы можете использовать функции FOR XML для формирования XML-структуры из результатов запроса. Это мощный метод для сложных XML-экспортов, который обычно не выполняется напрямую из SSMS.
3. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение выполнять экспорт данных из БД.

# Лабораторная работа №13

**Импорт данных пользователя в базу данных**

1. **Цель работы**
   1. Научиться выполнять импорт данных в БД.
2. **Ответы на контрольные вопросы**
   1. Для импорта Excel-файла в SQL Server с помощью SQL Server Management Studio нужно выполнить следующие шаги:
3. откройте мастер импорта: щелкнуть правой кнопкой мыши на базе данных, перейти в раздел «Задачи» и выбрать «Импорт данных».
4. выбрать источник данных как Microsoft Excel и выбрать файл Excel.
5. Присвоить столбцы из Excel -файла столбцам в таблице SQL.
6. Запустить процесс импорта, следуя подсказкам мастера импорта.
   1. Для импорта CSV-файла в SQL Server Management Studio нужно выполнить следующие шаги:
7. в обозревателе объектов Object Explorer выбрать целевую базу данных, щелкнуть по ней правой кнопкой мыши и выбрать из контекстного меню команду Tasks→Import Data,
8. в появившемся диалоговом окне выбрать источник данных DataSource из выпадающего списка. выбрать вариант Flat File Source и нажать кнопку Next,
9. на следующем экране указать путь до файла CSV на жёстком диске.
   1. Способы импорта XML-данных в SQL Server:

* с помощью массовой вставки,
* с помощью SSIS (SQL Server Integration Services),
* с помощью SQLCLR (SQL Common Language Runtime),
* с помощью функции OPENXML.
  1. Существует несколько способов импорта данных из файла JSON в SQL Server. Вот некоторые из них:
* с помощью SQL Server Integration Services (SSIS),
* с помощью OPENROWSET,
* с помощью BULK INSERT.

1. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение выполнять импорт данных в БД.

# Лабораторная работа №14

**Создание слоя доступа к данным БД**

1. **Цель работы**
   1. Научиться создавать приложение C# для организации доступа к БД;
   2. Изучить свойства и методы компонентов SqlConnection, SqlCommand,

SqlDataReader, научиться их применять и настраивать.

1. **Ответы на контрольные вопросы**
   1. SqlConnection — это класс в .NET, который представляет соединение с базой данных Microsoft SQL Server.
   2. SqlCommand — это класс в .NET, который представляет SQL-запрос или хранимую процедуру, которую нужно выполнить в базе данных SQL Server.
   3. SqlDataReader — это класс в .NET, который используется для чтения результатов SQL-запроса, выполненного с помощью объекта SqlCommand.
   4. Для подключения к разным СУБД в .NET требуются разные пространства имён (namespaces):

MS SQL Server: using System.Data.SqlClient;

MySQL: using MySql.Data.MySqlClient;

* 1. Класс SqlCommand предоставляет несколько методов для выполнения SQL-запросов:

1. ExecuteNonQuery():

* выполняет запрос, который не возвращает данные (например, INSERT, UPDATE, DELETE, CREATE, DROP),
* возвращает количество затронутых строк,
* используется для выполнения действий с данными.

1. ExecuteReader():

* выполняет запрос, который возвращает данные (например, SELECT),
* возвращает объект SqlDataReader, который используется для чтения результата запроса,
* используется для получения данных из базы данных.

1. ExecuteScalar():

* выполняет запрос, который возвращает одно скалярное значение (SELECT COUNT(\*), SELECT SUM(price)),
* возвращает первое значение из первого столбца первого ряда,
* Используется для получения единственного значения из результата.

1. **Вывод**
   1. В ходе проделанной лабораторной работы было сформировано умение создавать приложение C# для организации доступа к БД;
   2. В ходе проделанной лабораторной работы были изучены свойства и методы компонентов SqlConnection, SqlCommand, SqlDataReader, научиться их применять и настраивать.

# Лабораторная работа №15

**Применение ORM для доступа к данным БД**

1. **Цель работы**
   1. Научиться создавать приложение C# для организации доступа к БД;
   2. Научиться проектировать приложение, использующее паттерн репозиторий и Dapper.
2. **Ответы на контрольные вопросы**
   1. Dapper — это легковесный ORM (объектно-реляционный преобразователь) для .NET. Он представляет собой микро-ORM, который обеспечивает высокую производительность при работе с базами данных, предоставляя простой и эффективный способ выполнения SQL-запросов и отображения результатов в объектах.
   2. Dapper предоставляет несколько методов для извлечения данных из БД:

* Query<T>(),
* QuerySingle<T>(),
* QuerySingleOrDefault<T>(),
* QueryFirst<T>(),
* QueryFirstOrDefault<T>(),
* Query<dynamic>(),
* QueryMultiple().
  1. Dapper предоставляет следующие методы для изменения данных в БД:
* Execute(),
* ExecuteScalar<T>().
  1. Паттерн «Репозиторий» — это шаблон проектирования, который используется для инкапсуляции логики доступа к данным. Он предоставляет абстрактный интерфейс для работы с данными, скрывая детали реализации (например, работу с SQL-запросами и ORM) от остальной части приложения.

1. **Вывод**
   1. В ходе проделанной лабораторной работы было сформировано умение создавать приложение C# для организации доступа к БД;
   2. В ходе проделанной лабораторной работы было сформировано умение проектировать приложение, использующее паттерн репозиторий и Dapper.

# Лабораторная работа №16

**Применение ORM EF Core**

1. **Цель работы**
   1. Научиться создавать приложение C# для организации доступа к БД;
   2. Научиться разрабатывать приложение, использующее EF Core.
2. **Ответы на контрольные вопросы**
   1. ORM (объектно-реляционный преобразователь) — это программный инструмент или технология, которая позволяет сопоставлять объекты в объектно-ориентированном языке программирования (например, C#, Java) с данными в реляционной базе данных (например, SQL Server, MySQL, PostgreSQL).
   2. EF Core (Entity Framework Core) — это современный кроссплатформенный ORM от Microsoft для .NET. Он является переработанной и упрощённой версией оригинального Entity Framework (EF) и предоставляет удобный способ работы с реляционными базами данных.
   3. Для получения данных из БД в Entity Framework Core можно использовать метод FromSqlRaw() у объектов DbSet.
   4. Для редактирования, вставки, обновления и удаления данных с помощью EF Core можно использовать следующие методы:
3. вставка данных: используйте метод DbSet.Add для добавления новой сущности в контекст (instance of DbContext). При вызове метода SaveChanges() будет вставлена новая запись в базу данных,
4. обновление данных: В контексте EF API отслеживает все сущности, полученные с помощью контекста. Поэтому при редактировании данных EF автоматически устанавливает EntityState на «Modified». При вызове метода SaveChanges() будет выполнен обновленный оператор в базе данных,
5. удаление данных: используйте метод DbSet.Remove() для удаления записи в таблице базы данных.Стандартные классы делегатов для создания событий в C# - это EventHandler и EventHandler.
6. **Вывод**
   1. В ходе проделанной лабораторной работы было сформировано умение создавать приложение C# для организации доступа к БД.
   2. В ходе проделанной лабораторной работы было сформировано умение разрабатывать приложение, использующее EF Core.

# Лабораторная работа №17

**Разработка приложения для фильтрации, поиска и сортировки данных**

1. **Цель работы**
   1. Научиться создавать приложение C# для организации доступа к БД средствами EF Core;
   2. Научиться выполнять сортировку, фильтрацию и постраничный вывод данных, используя LINQ-запросы.
2. **Ответы на контрольные вопросы**
   1. Методы для сортировки данных в LINQ-запросах.
3. OrderBy(keySelector): сортирует последовательность по возрастанию на основе указанного ключа (keySelector),
4. OrderByDescending(keySelector): сортирует последовательность по убыванию на основе указанного ключа (keySelector),
5. ThenBy(keySelector): используется для вторичной сортировки, когда нужно отсортировать данные по нескольким ключам,
6. ThenByDescending(keySelector): используется для вторичной сортировки, когда нужно отсортировать данные по нескольким ключам,
   1. Методы Take и Skip используются для пагинации данных.
7. Skip(count):

* пропускает указанное количество элементов в последовательности,
* используется для пропуска уже отображенных элементов при переходе на следующую страницу,
* count — это количество элементов, которые нужно пропустить.

1. Take(count):

* возвращает указанное количество элементов из начала последовательности,
* используется для взятия элементов, которые нужно отобразить на текущей странице,
* count — это количество элементов, которые нужно взять.метод select используется в linq для проекции каждого элемента источника данных (например, коллекции) в новую форму.
  1. Метод Where(predicate) используется для фильтрации элементов в последовательности на основе указанного условия (предиката).
  2. При составлении условий (предикатов) в LINQ-запросах можно использовать следующие логические операторы:

1. && (логическое “И”)
2. || (логическое “ИЛИ”)
3. ! (логическое “НЕ”)
   1. Для проверки, что значение есть в списке, можно использовать метод Contains().
   2. Для проверки, что строка начинается с определенного текста, можно использовать метод StartsWith().
   3. Для проверки, что строка содержит определенный текст, можно использовать метод Contains().
   4. Метод Select() позволяет выбрать определенные поля из объектов, возвращаемых запросом.
4. Выбор определённых свойств:

var productNamesAndPrices = context.Products

.Select(p => new {p.ProductName, p.Price})

.ToList();

// результаты вернутся как анонимные объекты

1. Выбор с преобразованием типов

var productPrices = context.Products

.Select(p => (decimal)p.Price)

.ToList();

// результаты вернутся как список `decimal`

1. Выбор в конкретный класс:

var productDtos = context.Products

.Select(p => new ProductDto

{

Name = p.ProductName,

Cost = p.Price

})

.ToList();

1. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение создавать приложение C# для организации доступа к БД средствами EF Core.
   2. В ходе проделанной лабораторной работы было приобретено умение выполнять сортировку, фильтрацию и постраничный вывод данных, используя LINQ-запросы.

# Лабораторная работа №18

**Разработка приложения для редактирования данных**

1. **Цель работы**
   1. Научиться выполнять вставку, обновление и удаление записей средствами EF Core;
   2. Научиться обеспечивать обратную связь при редактировании данных.
2. **Ответы на контрольные вопросы**
   1. Методы Add() и AddRange() используются для добавления новых сущностей (записей) в контекст EF Core, чтобы они были добавлены в базу данных при вызове SaveChanges().
   2. Метод Update() используется для обновления существующей сущности в базе данных через контекст EF Core.
   3. Методы Remove() и RemoveRange() используются для удаления сущностей из контекста EF Core, чтобы они были удалены из базы данных при вызове SaveChanges().
   4. Метод SaveChanges() (или SaveChangesAsync()) используется для сохранения всех изменений, внесённых в контекст EF Core, в базе данных. SaveChanges — это синхронный метод, а SaveChangesAsync — асинхронный метод.
   5. Значения полей объекта изменяются путём прямого присвоения новых значений соответствующим свойствам объекта.
   6. По умолчанию, если идентификатор является числовым типом (int, long и т. д.) и ключевым полем, то он либо устанавливается автоматически в базе данных, либо имеет значение по умолчанию, если это поле не помечено как автоинкрементное (например, 0 или null).
   7. Есть несколько способов передачи объекта из одной формы в другую в Windows Forms (.NET Framework или .NET):

* через конструктор,
* через свойство,
* через публичный статический класс-хранилище,
* через публичное событие.

1. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение выполнять вставку, обновление и удаление записей средствами EF Core;
   2. В ходе проделанной лабораторной работы было приобретено умение обеспечивать обратную связь при редактировании данных.

# Лабораторная работа №19

**Выполнение SQL-команд и SQL-подпрограмм**

1. **Цель работы**
   1. Научиться выполнять SQL-команды и вызывать хранимые процедуры и функции пользователя средствами EF Core.
2. **Ответы на контрольные вопросы**
   1. В EF Core для выборки данных обычно используются LINQ-запросы. Однако в случаях, когда требуется выполнить SQL-запрос напрямую, можно использовать методы FromSqlRaw() или FromSqlInterpolated().
   2. Для выполнения команд по изменению данных (INSERT, UPDATE, DELETE) в EF Core можно использовать следующие методы:

* ExecuteSqlRaw(sql, parameters),
* ExecuteSqlInterpolated(sql).
  1. В EF Core параметры передаются в SQL-команды через FromSqlRaw() / FromSqlInterpolated() или ExecuteSqlRaw()/ExecuteSqlInterpolated().
  2. Использовать FromSqlRaw() или FromSqlInterpolated() или создать представление (view) на основе функции.
  3. Для вызова хранимой процедуры в EF Core можно использовать метод FromSqlRaw()/FromSqlInterpolated() для выборки данных или ExecuteSqlRaw()/ ExecuteSqlInterpolated() для изменения данных.

1. **Вывод**
   1. В ходе проделанной лабораторной работы было сформировано умение выполнять SQL-команды и вызывать хранимые процедуры и функции пользователя средствами EF Core.

# Лабораторная работа №20

**Реализация разграничения прав доступа пользователей**

1. **Цель работы**
   1. Научиться разграничивать права доступа пользователей на уровне интерфейса приложения;
   2. Научиться изменять настройки подключения к БД средствами Entity Framework Core.
2. **Ответы на контрольные вопросы**
   1. Самым простым способом управления подключением к базе данных является использование конфигурационного файла приложения.
   2. Существует несколько распространённых способов хранения пользователей и ролей в базе данных:

* простое хранение пользователей и ролей,
* более гибкое хранение ролей и разрешений.
  1. Авторизация — это процесс определения того, имеет ли аутентифицированный пользователь права на выполнение определенной операции или доступ к определенным ресурсам.
  2. Регистрация — это процесс создания новой учётной записи пользователя в системе.

1. **Вывод**
   1. В ходе проделанной лабораторной работы было сформировано умение разграничивать права доступа пользователей на уровне интерфейса приложения;
   2. В ходе проделанной лабораторной работы было сформировано умение изменять настройки подключения к БД средствами Entity Framework Core.

# Лабораторная работа №21

**Разработка приложения для импорта данных**

1. **Цель работы**
   1. Импорт данных пользователя в базу данных средствами EF Core.
2. **Ответы на контрольные вопросы**
   1. Для импорта данных из CSV-файла в базу данных через .NET-приложение можно использовать следующие подходы:

* чтение CSV и импорт с использованием ADO.NET или EF Core,
* использование BULK INSERT (только для MS SQL Server),
* использование SQL Server Integration Services (SSIS).
  1. Для импорта изображения в базу данных через приложение можно использовать следующие подходы:
* сохранение изображения в виде массива байтов,
* сохранение пути к файлу.
  1. Для импорта данных из JSON-файла в базу данных через приложение можно использовать следующие подходы:
* десериализация JSON и импорт в БД (ADO.NET или EF Core),
* использование OPENJSON (для MS SQL Server).

1. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умения импортировать данные пользователя в базу данных средствами EF Core.

# Лабораторная работа №22

**Разработка приложения для экспорта данных в текстовые файлы**

1. **Цель работы**
   1. Научиться выполнять экспорт данных из БД в формате текстового файла;
   2. Научиться выполнять экспорт данных из БД с применением Microsoft Office Word.
2. **Ответы на контрольные вопросы**
   1. Для работы с Microsoft Word в .NET-приложении обычно используется библиотека COM Interop (Component Object Model Interoperability), поскольку напрямую .NET-библиотеки для работы с Word не существует. Чтобы работать с Word, вам нужно подключить пространство имен: using Microsoft.Office.Interop.Word.
   2. Для сохранения данных в формате docx в .NET можно использовать следующие библиотеки:
3. Open XML SDK: для полного контроля над процессом, высокой производительности и независимости от установленного Office,
4. DocX: для более простого и быстрого решения, когда не требуется сложная обработка,
5. Aspose.Слова: когда требуется широкий набор функций, но вы готовы платить за лицензию,
6. COM Interop: если нужно автоматизировать работу с установленным Word.
   1. Экспорт данных в формат txt (.txt) является самым простым из всех представленных форматов, поскольку txt-файл — это обычный текстовый файл, поэтому для экспорта данных в .txt-файл можно использовать обычные средства .NET.
7. **Вывод**
   1. В ходе проделанной лабораторной работы было сформировано умение выполнять экспорт данных из БД в формате текстового файла.
   2. В ходе проделанной лабораторной работы было сформировано умение выполнять экспорт данных из БД с применением Microsoft Office Word.

# Лабораторная работа №23

**Разработка приложения для экспорта данных в табличные файлы**

1. **Цель работы**
   1. Научиться выполнять экспорт данных из БД в формате табличного файла;
   2. Научиться выполнять экспорт данных из БД с применением Microsoft Office Excel.
2. **Ответы на контрольные вопросы**
   1. Для работы с Microsoft Word в .NET-приложении обычно используется библиотека COM Interop (Component Object Model Interoperability), поскольку напрямую .NET-библиотеки для работы с Word не существует. Чтобы работать с Word, вам нужно подключить пространство имен: using Microsoft.Office.Interop.Word.
   2. Для сохранения данных в формате docx в .NET можно использовать следующие библиотеки:
3. Open XML SDK: для полного контроля над процессом, высокой производительности и независимости от установленного Office,
4. DocX: для более простого и быстрого решения, когда не требуется сложная обработка,
5. Aspose.Слова: когда требуется широкий набор функций, но вы готовы платить за лицензию,
6. COM Interop: Если нужно автоматизировать работу с установленным Word.
   1. Экспорт данных в формат txt (.txt) является самым простым из всех представленных форматов, поскольку txt-файл — это обычный текстовый файл, поэтому для экспорта данных в .txt-файл можно использовать обычные средства .NET.
7. **Вывод**
   1. В ходе проделанной лабораторной работы было приобретено умение выполнять экспорт данных из БД в формате табличного файла.
   2. В ходе проделанной лабораторной работы было приобретено умение выполнять экспорт данных из БД с применением Microsoft Office Excel.

# Практическая работа №1

**Разработка web-API для доступа к данным**

1. **Цель работы**
   1. Научиться выполнять разработку web-API для доступа к БД.
2. **Ответы на контрольные вопросы**
   1. REST-запрос — это запрос, отправляемый по протоколу HTTP на сервер для получения или отправки данных.
   2. RESTful — это архитектурный стиль для проектирования веб-сервисов, основанный на принципах REST.
   3. Метод GET используется для получения данных с сервера.
   4. Метод POST используется для отправки данных на сервер для создания нового ресурса или выполнения операции, которая может изменить состояние ресурсов на сервере.
   5. Метод PUT используется для полной замены существующего ресурса на сервере.
   6. Метод DELETE используется для удаления существующего ресурса с сервера.
3. **Вывод**
   1. В ходе проделанной практической работы было приобретено умение выполнять разработку web-API для доступа к БД.

# Практическая работа №2

**Вызов REST API**

1. **Цель работы**
   1. Научиться проверять работоспособность RESTful API в клиентском приложении;
   2. Научиться выполнять тестирование RESTful API методом черного ящика.
2. **Ответы на контрольные вопросы**
   1. У объекта HttpClient базовый адрес указывается при создании экземпляра HttpClient с использованием свойства BaseAddress в конструкторе HttpClientHandler или можно установить свойство BaseAddress после создания объекта HttpClient.
   2. Основные группы кодов ответа:
3. информационные: запрос принят, обработка продолжается

* 100 Continue – сервер получил часть запроса, клиент может продолжить отправку остальной части,
* 101 Switching Protocols – запрос клиента о переходе на другой протокол,

1. успешные: запрос успешно выполнен

* 200 OK – запрос успешно выполнен (обычно для GET-запросов),
* 201 Created – запрос успешно выполнен и был создан новый ресурс(обычно для POST-запросов),
* 202 Accepted – запрос был принят для обработки, но обработка еще не закончена,
* 204 No Content – запрос выполнен успешно, но не возвращается контент (обычно DELETE),

1. переадресация: ресурс был перемещен, и клиент должен выполнить дополнительный запрос

* 301 Moved Permanently – ресурс перемещен на новый URI,
* 302 Found (или 302 Moved Temporarily) – ресурс временно перемещен на другой URI,
* 304 Not Modified – ресурс не был изменён с момента последнего запроса (используется для кэширования),

1. клиентские ошибки: ошибка в запросе клиента

* 400 Bad Request – некорректный синтаксис запроса,
* 401 Unauthorized – необходима аутентификация,
* 403 Forbidden – доступ к ресурсу запрещен,
* 404 Not Found – ресурс не найден,
* 405 Method Not Allowed – данный метод не разрешен для этого ресурса,
* 409 Conflict – конфликт при изменении состояния ресурса,
* 429 Too Many Requests – слишком много запросов (для защиты от DDoS),

1. серверные ошибки: ошибка на стороне сервера.

* 500 Internal Server Error – внутренняя ошибка сервера,
* 501 Not Implemented – сервер не поддерживает метод,
* 502 Bad Gateway – сервер действует как шлюз и получил недопустимый ответ от другого сервера,
* 503 Service Unavailable – сервер временно недоступен (например, перегрузка).
  1. Для получения данных (обычно для GET запросов) используется метод GetAsync().
  2. Для вставки данных (обычно для POST запросов) используется метод PostAsync().
  3. Для изменения данных (обычно для PUT запросов) используется метод PutAsync().
  4. Для удаления данных (обычно для DELETE запросов) используется метод DeleteAsync().

1. **Вывод**
   1. В ходе проделанной практической работы было сформировано умение проверять работоспособность RESTful API в клиентском приложении.
   2. В ходе проделанной практической работы было сформировано умение выполнять тестирование RESTful API методом черного ящика.

# Практическая работа №3

**Разграничение прав доступа на уровне REST API**

1. **Цель работы**
   1. Научиться выполнять разработку web-API для доступа к БД;
   2. Научиться работать с JWT.
2. **Ответы на контрольные вопросы**
   1. Основные атрибуты:

* [Authorize],
* [AllowAnonymous],
* [Authorize(Roles = "role1, role2")],
* [Authorize(Policy = "policyName")].
  1. JWT (JSON Web Token) — это открытый стандарт (RFC 7519) для передачи безопасных утверждений (claims) между двумя сторонами в виде JSON-объекта. JWT обычно используются для авторизации и обмена данными в веб-приложениях.
  2. JWT менее уязвим для CSRF-атак, так как JWT отправляется через заголовок, а не через файл cookie.
  3. Основные настройки:

1. Issuer (издатель): указывает, кто выпустил токен,
2. аудитория (Audience): указывает, для кого предназначен токен,
3. субъект (Subject): идентификатор пользователя, для которого выдан токен,
4. утверждения (Claims): дополнительная информация о пользователе (например, роли, разрешения, права доступа и т. д.),
5. срок действия (Expiration): время, по истечении которого токен станет недействительным,
6. не ранее (время не раньше): время, когда токен вступит в силу,
7. ключ подписи (Signing Key): секретный ключ, используемый для подписи токена,
8. алгоритм (алгоритм подписи): алгоритм, используемый для подписи токена (например, HS256, RS256).
   1. Для авторизации (входа пользователя в систему) обычно используется метод POST, а не GET. Причины:
9. безопасность:

* пароль пользователя обычно передается в теле POST запроса (а не в URI),
* данные, передаваемые в теле POST-запроса, обычно шифруются, что обеспечивает большую безопасность, чем использование GET-запроса,
* GET-запросы обычно предназначены только для получения данных и не должны передавать конфиденциальную информацию,

1. передача больших объемов данных:

* POST позволяет передавать большие объёмы данных в теле запроса, в то время как GET имеет ограничения по длине URI,

1. изменение состояния на сервере:

* авторизация изменяет состояние на сервере (создание сеанса / токена), поэтому, согласно принципам REST, для таких операций предпочтительнее использовать POST.

1. **Вывод**
   1. В ходе проделанной практической работы было сформировано умение выполнять разработку web-API для доступа к БД.
   2. В ходе проделанной практической работы было сформировано умение работать с JWT.

# Практическая работа №4

**Разработка веб-клиента**

1. **Цель работы**
   1. Научиться разрабатывать клиентское веб-приложение для доступа к БД;
   2. Научиться проверять работоспособность RESTful API в клиентском приложении.
2. **Ответы на контрольные вопросы**
   1. Fetch API — это современный интерфейс в JavaScript для выполнения HTTP-запросов к серверу. Он основан на промисах (Promises), что делает его более удобным и понятным по сравнению со старым XMLHttpRequest. Пример GET-запроса:

fetch('https://api.example.com/products')

.then(response => {

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

return response.json(); // если получаем JSON

})

.then(data => {

console.log('Данные получены:', data);

// ... работа с полученными данными

})

.catch(error => {

console.error('Ошибка при выполнении запроса:', error);

});

* 1. Обработка ошибок при вызове API важна для обеспечения стабильности и надёжности приложения. Используйте блоки try...catch и проверки статуса HTTP-ответа.
  2. Асинхронные функции (async/await) — это синтаксическое расширение JavaScript, которое позволяет работать с асинхронным кодом более простым и понятным способом. Асинхронные функции основаны на промисах. Связь с вызовом API:
* вызовы API (через fetch()) являются асинхронными, так как они могут занимать некоторое время,
* async/await делают асинхронный код похожим на синхронный, что упрощает чтение и понимание кода.
  1. Razor Pages — это фреймворк для создания веб-интерфейсов в ASP.NET Core.
  2. Чтобы создать новое приложение Razor Page в ASP.NET Core, нужно выполнить следующие шаги:

1. перейти в меню «Файл»→«Создать»→«Проект».
2. в окне «Новый проект» выбрать .NET Core на левой панели и «ASP.NET Основное веб-приложение» на центральной панели.
3. указать название приложения и местоположение для его сохранения и нажать «ОК».
4. в следующем окне выбрать шаблон, который отвечает за создание страницы Razor. Выбрать «Веб-приложение».
5. Изменить аутентификацию на «Без аутентификации» и нажать «ОК».

Чтобы добавить новую страницу Razor, нужно щелкнуть правой кнопкой мыши папку Pages или любую папку внутри Pages и выберите «Добавить»→«Страница Razor». В новом окне нужно выбрать тип страницы Razor.

1. **Вывод**
   1. В ходе проделанной практической работы было сформировано умение разрабатывать клиентское веб-приложение для доступа к БД.
   2. В ходе проделанной практической работы было сформировано умение проверять работоспособность RESTful API в клиентском приложении.