Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Программируемые мобильные системы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему

**ПРОГРАММНОЕ СРЕДСТВО ЗООМАГАЗИНА**

**«PetShop»**

Выполнил: студент группы

Ф.И.О.

Проверил: Ф.И.О. преподавателя

Минск 2023

**СОДЕРЖАНИЕ**

1 Введение…………………………………………………………………………3

1.1 Описание предметной деятельности…………………………………3

1.2 Группы пользователей, их основные задачи и запросы к базе данных…………………………………………………………………………………3

1.3 Цели и задачи курсового проектирования…………………………...4

2 Проектирование…………………………………………………………………5

2.1 Инфологический этап проектирования………………………………5

2.2 Логический этап проектирования…………………………………….5

2.3 Обоснование выбора технологий и описание среды реализации…..8

3 Программная реализация……………………………………………………...10

3.1 Физическая структура базы данных………………………………...10

3.2 Реализация проекта базы данных…………………………………...11

3.2.1 Создание и заполнение таблиц базы данных………………11

3.2.2 Создание представлений……………………………………12

3.2.3 Назначение прав доступа……………………………………12

3.2.4 Создание индексов…………………………………………..14

3.3 Структура приложения………………………………………………14

4 Тестирование…………………………………………………………………...15

5 Описание применения…………………………………………………………18

Заключение……………………………………………………………………….19

Список использованных источников…………………………………………...20

# 1 ВВЕДЕНИЕ

На сегодняшний день базы данных используются во многих сферах нашей жизни: в медицине, в банковской сфере, в торговле, в интернет-магазинах и т.д. Чтобы обеспечить надежность хранения и обработки данных, в 1970-ых годах началось активное развитие технологий баз данных.

Язык SQL (Structured Query Language) был создан в 1974 году. Данный язык активно применяется в разработке и работе с базами данных. На данный момент существует огромное количество систем управления реляционными базами данных, которые используют язык SQL или его расширения.

В процессе данной курсовой работы будут рассмотрены основные виды систем управления базами данных, их преимущества и методы работы с ними. Помимо этого, в данной курсовой работе будут описаны методы проектирования баз данных и методы взаимодействия с базами данных.

Зоомагазины ежедневно обрабатывают огромное количество заказов, к ним поступают различные поставки. В связи с этим разрабатываются различные системы для зоомагазинов.

Программное средство “PetShop” упрощает работу всего магазина по взаимодействию работников, курьеров и покупателями.

Так как это база данных, то всю информацию о товаре можно заполнить один раз, а в будущем только изменять: удалить товар, изменить их количество в наличии, стоимость и добавлять новые товары.

1.1 Описание предметной деятельности

В первую очередь при проектировании базы данных необходимо выбрать цель создания базы данных, основные её возможности и информацию, которую база данных будет обрабатывать.

В создаваемой базе данных для работы с программным средством “PetShop” должна храниться информация, которая необходима для работы зоомагазинов. К необходимой информации можно отнести информацию о товарах, работников, реализованных товарах и чеках. Данная база данных создается для внутреннего использования работниками зоомагазинов.

1.2 Группы пользователей, их основные задачи и запросы к базе данных

Обязанности работника:

- добавлять новые товары в систему;

- обслуживать клиентов.

- директора дополнительно имеют возможность добавления, изменения и удаления пользователей из системы.

С учетом обязанностей работников зоомагазинов, можно выделить следующие свойства проектируемой базы данных:

- каталог реализуемой продукции, которое содержит количество в наличии, их стоимость, категорию и название;

- чеки с информацией о реализованных товарах, суммой чека.

Запросы к базе данных будут дублировать обязанности работников зоомагазинов.

1.3 Цели и задачи курсового проектирования

Цель данной курсовой работы заключается в:

– реализации программного средства, которое упростит и оптимизирует работу зоомагазинов;

- обеспечении работников принимать поставки;

- обеспечении работников проводить продажи покупателям;

- обеспечении директорам редактирования списка пользователей, которые имеют доступ к системе;

Задачи данной курсовой работы:

– анализ предметной области;

– проектирование базы данных;

– разработка программного средства для взаимодействия с базой данных.

Объектом исследования является база данных, разработанная с помощью специальных программных средств.

Предметом программного средства является повышение автоматизации работы зоомагазинов.

Программное средство может использовать в любых зоомагазинах.

# 2 ПРОЕКТИРОВАНИЕ

2.1 Инфологический этап проектирования

Основными задачами концептуального проектирования являются определение предметной области системы и формирование взгляда на предметную область с позиций сообщества будущих пользователей БД, т. е. инфологической модели предметной области.

Концептуальная модель предметной области представляет собой описание структуры и динамики предметной области, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектах предметной области и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.

Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов.

Основные требования к содержанию модели:

* концептуальная модель должна отображать все сущности и связи, значимые для той цели, ради которой мы ее рисуем;
* все объекты модели (и сущности, и связи) должны быть именованы. Именование сущностей и связей должно выполняться в терминах предметной области;
* для связей должна быть указана кратность (один – многие);
* для каждой связи должно быть указано направление чтения.

Для данного программного средства существует несколько главных сущностей.

Взаимоотношения между сущностями иллюстрируются с помощью связей. Правила и ограничения взаимоотношений описываются с помощью свойств связей. Обычно связи определяют либо зависимости между сущностями, либо влияние одной сущности на другую.

Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Слабые сущности называют подчиненными (дочерними), а сильные – базовыми (родительскими).

Таким образом была спроектирована концептуальная модель БД.

2.2 Логический этап проектирования

В качестве логического уровня моделирования при разработке программного обеспечения были выбраны следующие диаграммы:

– диаграмма «сущность-связь»;

– диаграмма вариантов использования;

– диаграмма деятельности;

– диаграмма последовательности.

Диаграммы «сущность-связь» (ERD) (рисунок 2.1) предназначены для разработки моделей данных и позволяют описывать схемы предметной деятельности. С помощью данной модели можно выделить ключевые сущности и выделить связи, которые устанавливаются между сущностями.

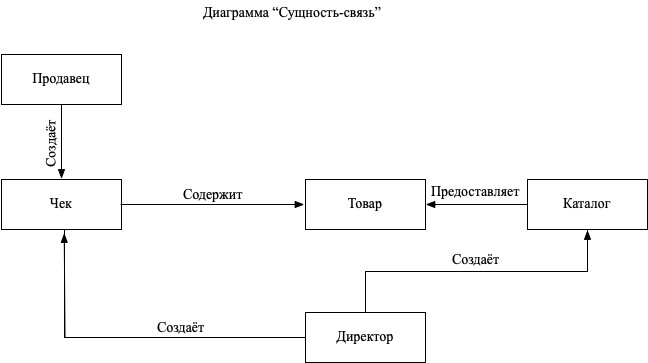


Рисунок 3.1

Диаграмма вариантов использования (рисунок 3.2) – диаграмма, описывающая функционал разрабатываемого программного обеспечения доступен каждой группе пользователей.



Рисунок 3.2

Основными элементами диаграммы являются: группы пользователей и действия.

В нашей информационной системе можно выделить группы пользователей:

– продавец;

– директор.

Каждая из групп пользователей пользуются системой исходя из своих обязанностей.

Продавцы могут:

– создать чек;

– закрыть чек (провести финальный расчет);

– экспортировать чек для дальнейшей печати;

- редактировать каталог.

Менеджеры имеют доступ ко всему функционалу программного обеспечения.

Диаграмма деятельности – диаграмма, которая фокусируется на выполнении и потоке поведения системы вместо реализации. Данный вид диаграмм имеет другое название – объектно-ориентированная блок-схема.

2.3 Обоснование выбора технологий и описание среды реализации

В нынешнее время информационные технологии интенсивно развиваются, и они активно используются во всех сферах жизни человека. Также увеличиваются объемы обрабатываемой информации. В связи с этим появляется необходимость в эффективности обработки, хранения информации. Автоматизированное программное средство является мощным инструментом во многих компаниях.

Перед началом работы были изучены средства разработки и технологии с целью выбора наиболее подходящего варианта для реализации программного обеспечения. Для создания многопользовательских баз данных широко используют следующие системы:

– MySQL;

– Microsoft SQL Server (MS SQL Server);

– Oracle SQL.

Инфраструктура Framework Core решает одну основную задачу: сохранение объектов .Net в базе данных и извлечение их в более позднее время.

Проанализировав различные базы данных, была выбрана система Microsoft SQL Server. Преимущества данного варианта:

– Многопоточность – поддержка нескольких запросов;

– Простота в использовании;

– Наличие множества плагинов и приложений, которые упрощают работу с базами данных;

– Гибкая связь привилегий и паролей;

– Безопасность;

– Масштабируемость – данная система является универсальной и её можно использовать для работы с проектами разного размера;

– Скорость – высокая производительность обеспечивается за счет упрощения некоторых стандартов.

Для разработки программного обеспечения с возможностью работы с базами данных широко используют следующие языки программирования:

– C++;

– C#;

– PHP;

– Java.

Выбор был остановлен на C#, так как он позаимствовал многие вещи из С++ и Java. Так как язык C# является объектно-ориентированным, данный язык позволяет разработать крупные, но одновременно расширяемые приложения. В основном данный язык позаимствовал возможности Java, позаимствовал некоторые возможности из C++ и в нем реализованы собственные идеи разработчиков языка. С каждой новой версией в язык добавляются новые возможности, которые облегчают разработчикам разработку.

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

**3.1 Физическая структура базы данных**

Этап физического проектирования заключается в увязке логической структуры БД и физической среды хранения с целью наиболее эффективного размещения данных, т. е. отображении логической структуры БД в структуру хранения.

Решается вопрос размещения хранимых данных в пространстве памяти, выбора эффективных методов доступа к различным компонентам «физической» БД. Результаты этого этапа документируются в форме схемы хранения на языке определения данных (DDL). Принятые на этом этапе решения оказывают определяющее влияние на производительность системы.

Между физическим и логическим проектированием существует обратная связь, так как иногда с целью повышения эффективности приходится менять структуру БД. Это возникает, если полная нормализация данных не позволяет достичь требуемой эффективности обработки информации. Принимается решение о денормализации отношений. При этом следует учитывать следующее: денормализация усложняет реализацию БД; денормализация снижает гибкость системы; денормализация ускоряет выборку данных, но снижается скорость обновления данных. Иногда денормализацию называют оптимизацией исполнения. Одной из важнейших составляющих проекта базы данных является разработка средств защиты БД. Защита данных имеет два аспекта: защита от сбоев и защита от несанкционированного доступа. Для защиты от сбоев разрабатывается стратегия резервного копирования. Для защиты от несанкционированного доступа каждому пользователю доступ к данным предоставляется только в соответствии с его правами доступа.

Создание базы данных начинается с самых нижних уровней, с таблиц у которых нет внешних ключей.

Таблица «Bills» состоит из таких атрибутов как:

– «ID», который является первичным ключом и имеет тип данных «int»;

– «Date», который имеет тип данных «nvarchar(50)»;

– «Time», который имеет тип данных «nvarchar(50)»;

– «Sum», который имеет тип данных «float»;

– «GivenSum», который имеет тип данных «float»;

– «TypeOfMoney», который имеет тип данных «nvarchar(50)»;

– «Change», который имеет тип данных «float».

Таблица «Items» состоит из таких атрибутов как:

– «ID», который является первичным ключом и имеет тип данных «int»;

– «Name», который имеет тип данных «nvarchar(50)»;

– «Price», который имеет тип данных «float»;

– «Category», который имеет тип данных «nvarchar(50)»;

– «Count», который имеет тип данных «int».

Таблица «Purchases» состоит из таких атрибутов как:

– «ID», который является первичным ключом и имеет тип данных «int»;

– «BillID», который имеет тип данных «int»;

– «DishID», который имеет тип данных «int»;

– «Count», который имеет тип данных «int».

Таблица «Users» состоит из таких атрибутов как:

– «ID», который является первичным ключом и имеет тип данных «int»;

– «Login», который имеет тип данных «nvarchar(50)»;

– «Password», который имеет тип данных «nvarchar(50)»;

– «Role», который имеет тип данных «nvarchar(50)».

Итогом физического проектирования должно быть представление метаданных всех таблиц. Далее будет описано пошаговое создание базы данных.

**3.2 Реализация проекта базы данных**

**3.2.1 Создание и заполнение таблиц базы данных**

На этом этапе необходимо создать базу данных в MsSQL Server. База данных \*Имя БД\* создается при помощи команды: CREATE DATABASE [Имя БД]. Создание таблиц базы данных представлено ниже. Метаданные таблицы «Bills»:

CREATE TABLE [dbo].[Bills]

(

[ID] INT NOT NULL PRIMARY KEY,

[Date] NVARCHAR(50) NOT NULL,

[Time] NVARCHAR(50) NOT NULL,

[Sum] FLOAT NOT NULL,

[GivenSum] FLOAT NOT NULL,

[Change] FLOAT NOT NULL,

[TypeOfGivenMoney] NVARCHAR(50) NOT NULL

);

Метаданные таблицы «Items»:

CREATE TABLE [dbo].[Items] (

[ID] INT NOT NULL,

[Name] NVARCHAR (50) NOT NULL,

[Price] FLOAT (53) NOT NULL,

[Category] NVARCHAR (50) NOT NULL,

[Count] INT NOT NULL,

PRIMARY KEY CLUSTERED ([ID] ASC)

);

Метаданные таблицы «Users»:

CREATE TABLE [dbo].[Users] (

[Id] INT NOT NULL,

[Login] NVARCHAR (50) NOT NULL,

[Password] NVARCHAR (50) NOT NULL,

[Role] NVARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

Метаданные таблицы «Purchases»:

CREATE TABLE [dbo].[Purchases] (

[ID] INT NOT NULL,

[BillID] INT NOT NULL,

[ItemID] INT NOT NULL,

[Count] INT NOT NULL,

PRIMARY KEY CLUSTERED ([ID] ASC)

);

Заполнять данные таблицы необязательно, так как заполнение может быть во время работы данного программного обеспечения.

Заполнение таблицы «Users»:

INSET INTO Users (ID, Login, Password, Role) VALUES (1, ‘manager’, ‘password’, N‘Директор’), (2, ‘worker’, ‘workerPassword’, N’Продавец’)

Заполнение таблицы «Items»:

INSERT INTO Items (ID, Name, Price, Category, Count) VALUES (1, N‘Игрушка для домашних животных’, ’10.55’, N‘Игрушки’, ‘50’), (2, N’Лакомство для собак’, ‘3.45’, ‘Еда для домашних животных’, ‘34’)

**3.2.2 Создание представлений**

Представления представляют из себя виртуальные таблицы, которые содержат запросы, которые динамически извлекают данные. Данные представления дают нам ряд преимуществ, например, они защищают данные, так как представления дают доступ только к части таблицы, а не ко всем данным таблицы.

Создание представления «PurchasesItems»:

CREATE VIEW PurchasesItems AS SELECT Purchases.BillID, Purchases.ItemID, Purchases.Count, Items.Name, Items.Price, Items.Category FROM Purchases, Items WHERE Purchases.ItemID = Items.ID;

**3.2.3 Назначение прав доступа**

Права доступа – права, которые дают возможность доступа к объекту базы данных. Право доступа предоставляются пользователю или группе пользователей для выполнения следующих функций:

– показать данные;

– добавление новых данных;

– обновление данных.

Право доступа предоставляются владельцу или создателю объекта. В дальнейшем они могут принимать решение о предоставлении прав доступа другим пользователям. Существует несколько типов прав доступа к объектам базы данных:

– право на выполнение инструкций – определяет набор прав пользователя на выполнение выражений типа создание базы данных;

– право на работу с объектами – определяет набор прав пользователя при работе с данными;

– предопределенные права – определяют набор действий, которые разрешены для выполнения пользователями, входящих в определенные стандартные роли.

Ограничивая круг пользователей, можно избежать больших проблем.

Список и описание прав доступа к таблицам:

– SELECT – дает возможность пользователю прочитать данные таблицы или представления. Есть возможность дать право доступа SELCT как ко всей таблице или представлению, так и к индивидуальным столбцам;

– INSERT – дает возможность пользователю добавить новые данные в таблицу или в представление;

– UPDATE – дает возможность изменить существующие данные в таблице или в представлении;

– DELETE – дает возможность удалить данные из таблицы или из представления.

**3.2.4 Создание индексов**

Системы баз данных используют индексы для обеспечения быстрого доступа к реляционным данным.

Индекс – отдельная структура данных, позволяющая получать быстрый доступ к одной или нескольким строкам таблицы базы данных.

Индекс базы данных во многом схож с алфавитным указателем книги. Но есть две разницы:

– читатель книги имеет возможность использовать алфавитный указатель или нет. Пользователь базы данных такой возможности не имеет и за него это решение принимает компонент системы, который имеет название «оптимизатор запросов»;

– индекс для определенной книги создается вместе с книгой, после чего он больше не изменяется. В базе данных индекс будет меняться при каждом изменении данных.

Стоит отметить, что обновление таблицы, содержащей индексы, занимает больше времени, чем обновление таблицы без индексов, так как индексы также нуждаются в обновлении.

**3.3 Структура приложения**

При проектировании уделяется отдельное внимание вопросам предоставления информации пользователю, а также взаимодействию пользователя с системой, что в наибольшей степени определяется типом приложения и выдается заказчиком в виде требования.

В настоящее время существует большое количество различных программных решений, которые позволяют разработчику выбрать наиболее оптимальное для конкретной проблемы.

Модули – это законченные программные блоки, состоящие из моделей, представлений, контроллеров и других вспомогательных компонентов.

При установке модулей в приложение, конечный пользователь получает доступ к их контроллерам. По этой причине модули часто рассматриваются как миниатюрные приложения. В отличии от приложений, модули нельзя развертывать отдельно. Модули должны находиться внутри приложений.

Проект содержит 9 страниц, доступных пользователям.

При запуске приложения, пользователь попадает в зону входа, в которой пользователь вводит свой логин и пароль. После успешной авторизации пользователь попадает в меню, которое доступно этому пользователю по роли.

**4 ТЕСТИРОВАНИЕ**

Тестирование – процесс проверки программного средства с целью выявления и предотвращения дефектов, оценки соответствия требованиям заказчика.

Баг (дефект) – ошибка в коде, которая может привести к тому, что программное средство будет работать не так, как ожидается.

Тестирование программного средства – это креативная и интеллектуальная работа. Разработка правильных и эффективных тестов – достаточно непростое занятие. Принципы тестирования, представленные ниже, были разработаны в последние 40 лет и являются общим руководством для тестирования в целом:

- тестирование показывает наличие дефектов. Оно может показать наличие дефектов в программном средстве, но не доказать их отсутствие. Тем не менее, важно составлять тест-кейсы, которые будут находить как можно больше багов. Таким образом, при должном тестовом покрытии, тестирование позволяет снизить вероятность наличия дефектов в программном обеспечении. В то же время, даже если дефекты не были найдены в процессе тестирования, нельзя утверждать, что их нет;

- исчерпывающее тестирование невозможно. Невозможно провести исчерпывающее тестирование, которое бы покрывало все комбинации пользовательского ввода и состояний системы, за исключением совсем уж примитивных случаев. Вместо этого необходимо использовать анализ рисков и расстановку приоритетов, что позволит более эффективно распределять усилия по обеспечению качества ПО;

- раннее тестирование. Тестирование должно начинаться как можно раньше в жизненном цикле разработки программного обеспечения, и его усилия должны быть сконцентрированы на определенных целях;

- скопление дефектов. Разные модули системы могут содержать разное количество дефектов – то есть, плотность скопления дефектов в разных элементах программного средства может отличаться. Усилия по тестированию должны распределяться пропорционально фактической плотности дефектов. В основном, большую часть критических дефектов находят в ограниченном количестве модулей. Это проявление принципа Парето: 80% проблем содержатся в 20% модулей;

- тестирование зависит от контекста. Выбор методологии, техники и типа тестирования будет напрямую зависеть от природы самого программного средства. Например, программное обеспечение для медицинских нужд требует гораздо более строгой и тщательной проверки, чем, скажем, компьютерная игра. Из тех же соображений, сайт с большой посещаемостью должен пройти через серьезное тестирование производительности, чтобы показать возможность работы в условиях высокой нагрузки.

Исполнение тестов необходимо, но не менее важны и сопровождающие действия – планирование и документирование процесса. В обязанности тестировщика входит разработка тестовых сценариев, а также подготовка тестирования и оценка его результатов. Таким образом можно выделить следующие процессы тестирования:

- планирование и управление – определение целей тестирования и описание задач тестирования для достижения этих целей; сопоставление текущей ситуации в процессе тестирования с планом и составление отчетности;

- анализ и проектирование – это процесс написания тестовых сценариев и условий на основе общих целей тестирования;

- внедрение и реализация – написание тест-кейсов, на основе написанных ранее тестовых сценариев, собирается необходимая для проведения тестов информация, подготавливается тестовое окружение и запускаются тесты;

- оценка критериев выхода и написание отчетов – проверка, было ли проведено достаточное количество тестов, достигнута ли нужная степень обеспечения качества системы; убедится в том, что нет необходимости проводить дополнительные тесты. Если все же такая необходимость есть, возможно, потребуется изменить установленный критерий выхода;

- действия по завершению тестирования – сборка, систематизация и анализ информации о результатах тестирования.

Тестирование программного обеспечения имеет следующие уровни:

- компонентное – поиск дефектов и проверка функциональности (модулей, программных средств, объектов, классов), которые можно протестировать изолированно;

- интеграционное – проверяет взаимодействие между компонентами, взаимодействие различных частей системы (ОС, файловая система, аппаратное обеспечение);

- системное – тестирование поведения тестового объекта как целостной системы или программного средства;

- приемочное, в свою очередь, подразделяется на пользовательское, эксплуатационное, контрактное и правовое, альфа и бета. Заключается в проверке работоспособности системы, частей системы или отдельных нефункциональных характеристик системы.

Тестирование программного средства проходило в ходе и после ее разработки. При тестировании использовалось компонентное, системное, интеграционное и приемочное тестирование. Меньше всего времени было затрачено на приемочное тестирование, потому что ежедневно просматривались требования к данному программному обеспечению и продумывались варианты оптимальной реализации. Также использовались методы тестирования черным и белым ящиком, что весьма помогло в поиске дефектов.

**5 ОПИСАНИЕ ПРИМЕНЕНИЯ**

Данное программное средство помогает эффективно управлять зоомагазинами.

За счет автоматизации зоомагазинов, работа данных заведений становится продуктивнее и легче.   
 Плюсы данного программного средства:

- продавцам легко считать нужную сумму чека, размер сдачи;

- легко вносить управлять информацией о каталоге;

– менеджеру легче добавить новых сотрудников или удалить старых.

**Заключение**

Перед созданием компонентов системы было выполнено проектирование БД и приложения. В результате концептуального проектирования и анализа предметной области были выделены основные сущности и составлена диаграмма «сущность-связь». В процессе создания алгоритма решения задачи были спроектированы диаграммы деятельности и вариантов использования, отражающие структуру и функции проекта. Также был спроектирован пользовательский интерфейс.

В результате работы было разработано программное средство зоомагазина. Приложение имеет удобный и интуитивно понятный интерфейс, позволяет пользователям авторизоваться с правами пользователя или администратора. Администратор может легко заполнить базу данных без необходимости изучения языков запросов.

Разработанная автоматизированная система обладает следующими преимуществами:

* удобный интерфейс;
* возможность сохранять данные;
* возможность редактирования БД от имени администратора.

Главной задачей системы является создание статей и тестов. Все цели разработки были достигнуты, система протестирована и готова к использованию.

Были усвоены теоретические основы организации баз данных, включая принципы построения на концептуальном, логическом и физическом уровнях, а также получены навыки ставить и решать практические задачи проектирования и эксплуатации баз данных.

# Список использованных источников

[1] Бошемин, Б. Основы .NET / Б.Бошемин. – Вильямс,2016.

[2] Шилдт, Г. C# 4.0. Полное руководство /  Г. Шилдт .  – М.: Вильямс, 2019.

[3] Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке С# / Б.Шнайер. – М.: Триумф, 2003.

[4] Страуструп, Б. Полное руководство C# / Б.Страуструп.  – М.: Вильямс, 2017.

[5] Троелсен, Э. Язык программирования C# 5.0 и платформа   
.NET 4.5 / Э. Троелсен. – М.: Вильямс, 2015.

[6] Павловская, Т. А. С#. Объектно-ориентированное программирование: практикум / Павловская, Т. А., Щупак. –  СПб. : Питер, 2004.

[7] Керниган, Б. Язык программирования С# / Б. Керниган, Д. Ритчи. – М.: СПб.: Питер, 2002.

[8] Шварц, Б. MySQL по максимуму. / Б. Шварц, Т. Вадим, З. Петр – Питер, 2015.

[9] Ульман, Л. MySQL: руководство по изучению языка / Л. Ульман – М.: Эксмо, 2010.