**NМИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«ТЮМЕНСКИЙ индустриальный университет»**

Высшая школа цифровых технологий

Кафедра Математики и прикладных информационных систем

**КУРСОВОЙ ПРОЕКТ**  
**Дисциплина:** «Управление базами данных»   
**Тема:** «Разработка СУБД по предметной области «Домашняя бухгалтерия» на основе MySQL»

**Выполнил:**

Студент 3 курса группы АСОиУб-21-2

направления 09.03.01 «Информатика и вычислительная техника»

Тувышкин Д.А.

**Руководитель:**

Доцент, канд. пед. наук

Спирин И.С.

Тюмень 2024

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Для реализации курсового проекта необходимо решить ряд задач:

* анализ и описание предметной области;
* определение предметной области и темы курсового проекта;
* оформление технического задания и его утверждение;
* представление концепции интерфейса сервиса;
* реализация концепции интерфейса сервиса;
* создание базы данных;
* реализация функциональной части сервиса;
* предоставление функциональной части сервиса;
* оформление курсового проекта и документации проекта;
* предоставление курсовой проекта и документации проекта.

АННОТАЦИЯ

В данной работе рассматривается проблема эффективного управления домашними финансами. Проводится анализ существующих способов ведения учета доходов и расходов, а также других финансовых данных.

Целью данной работы является создание удобного и функционального приложения, которое позволит пользователям быстро и эффективно управлять своими финансами, улучшая финансовую грамотность и удовлетворяя потребности в контроле бюджета.

Объектом исследования является приложение для эффективного управления домашними финансами.

Предметом исследования является процесс создания приложения для взаимодействия между пользователями и их финансовыми данными.

Основные задачи исследования:

1. Провести анализ предметной области, изучить роль домашней бухгалтерии в жизни пользователей, существующие способы ведения учета финансов, их преимущества и недостатки, а также обосновать необходимость разработки нового приложения для управления финансами.
2. Определить методологию разработки приложения, сформулировать требования к нему, спроектировать архитектуру и выбрать соответствующие технологии и инструменты для реализации;
3. Осуществить процесс разработки приложения, включая реализацию основных функций, создание пользовательского интерфейса, а также провести тестирование и отладку приложения.

Практическая значимость исследования заключается в создании удобного и эффективного инструмента для управления домашними финансами, что может повысить финансовую грамотность пользователей и улучшить их способность контролировать бюджет.

В результате исследования была спроектирована и реализована СУБД для учета домашних финансов, включающая базы данных для хранения информации о доходах, расходах, бюджетах и других важных данных. Система позволяет эффективно управлять этой информацией, осуществлять поиск, добавление, редактирование и удаление записей.

Ключевые слова: домашняя бухгалтерия, финансы, учет доходов и расходов, пользовательская система учёта аптеки, MySQL, информационная система, управление финансами, база данных.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc183558365)

[1 Анализ предметной области 5](#_Toc183558366)

[1.1 Роль домашней бухгалтерии в жизни обычных потребителей 5](#_Toc183558367)

[1.2 Существующие способы взаимодействия 6](#_Toc183558368)

[1.3 Обоснование необходимости разработки нового приложения 8](#_Toc183558369)

[2 Методология разработка приложения 10](#_Toc183558370)

[2.1 Выбор методологии разработки 10](#_Toc183558371)

[2.2 Сбор и анализ требований 11](#_Toc183558372)

[2.3 Проектирование архитектуры СУБД 13](#_Toc183558373)

[2.4 Планирование разработки 17](#_Toc183558374)

[3 Процесс разработки приложения 20](#_Toc183558375)

[3.1 Создание базы данных 20](#_Toc183558376)

[3.2 Выбор фреймворка/библиотек 21](#_Toc183558377)

[ЗАКЛЮЧЕНИЕ 26](#_Toc183558378)

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

Домашняя бухгалтерия - это система учета финансовых операций, которая помогает пользователям отслеживать свои доходы и расходы, планировать бюджет и управлять личными финансами.

WPF (Windows Presentation Foundation) - это подсистема для построения графических интерфейсов, которая является частью экосистемы платформы .NET.

Приложение - программное обеспечение, разработанное для использования клиентами и сотрудниками автосервиса.

Бэкенд - серверная часть приложения, отвечающая за обработку данных, логику и взаимодействие с базами данных.

Фронтенд - клиентская часть приложения, отвечающая за визуализацию данных и взаимодействие с пользователем.

Тестирование - процесс проверки работоспособности и функциональности приложения с целью выявления и устранения ошибок.

API (Application Programming Interface) - Интерфейс программирования приложений, представляющий собой набор готовых классов, процедур, функций, структур для обеспечения взаимодействия различных программных компонентов.

СУБД (Система Управления Базами Данных) - программное обеспечение, предназначенное для создания, хранения и управления базами данных.

ER-диаграмма (Entity-Relationship Diagram) - диаграмма "сущность-связь", используемая для визуального представления концептуальной модели данных.

SQL (Structured Query Language) - структурированный язык запросов, применяемый для управления реляционными базами данных.

Обозначения и сокращения:

1. UI - Пользовательский интерфейс;
2. БД - База данных;
3. API - Программный интерфейс приложения;
4. MVP - Минимально жизнеспособный продукт;
5. IDE - Интегрированная среда разработки.

# **ВВЕДЕНИЕ**

В современном мире цифровые технологии активно внедряются во все сферы жизни, включая управление личными финансами. Эффективное взаимодействие между пользователями и их финансовыми данными является важным аспектом для обеспечения контроля бюджета и повышения финансовой грамотности. Разработка специализированного приложения для домашней бухгалтерии может значительно упростить процессы учета доходов и расходов, планирования бюджета и анализа финансового состояния.

Целью данной работы является создание удобного и функционального приложения, которое позволит пользователям быстро и эффективно управлять своими финансами, улучшая качество учета и удовлетворяя потребности в контроле бюджета.

Объектом исследования является приложение для эффективного управления домашними финансами.

Предметом исследования является процесс создания приложения, обеспечивающего взаимодействие между пользователями и их финансовыми данными.

Задачи исследования:

1. Провести анализ предметной области, изучить роль домашней бухгалтерии в жизни обычных людей, существующие способы учета финансов, их преимущества и недостатки, а также обосновать необходимость разработки нового приложения для управления личными финансами.
2. Определить методологию разработки приложения, сформулировать требования к нему, спроектировать архитектуру и выбрать соответствующие технологии и инструменты для реализации.
3. Осуществить процесс разработки приложения, включая реализацию основных функций, создание пользовательского интерфейса, а также провести тестирование и отладку приложения.

Практическая значимость исследования заключается в создании удобного и эффективного инструмента для управления домашними финансами, что может повысить финансовую грамотность пользователей и улучшить их способность контролировать бюджет.

# **1 Анализ предметной области**

## **1.1 Роль домашней бухгалтерии в жизни обычных потребителей**

Домашняя бухгалтерия играет существенную роль в управлении личными финансами, обеспечивая доступ к важной информации о доходах и расходах, а также способствуя финансовой грамотности. Для многих людей ведение учета финансов становится первым шагом к улучшению их финансового состояния и планированию будущего.

Помимо основной функции учета доходов и расходов, домашняя бухгалтерия выполняет ряд других важных задач.

Предоставление информации:

- приложения для домашней бухгалтерии могут предоставлять пользователям рекомендации по оптимизации расходов, планированию бюджета и накоплению средств;

- пользователи могут получать советы по эффективному управлению финансами, включая информацию о сбережениях и инвестициях.

Обеспечение финансовой безопасности:

- приложения помогают пользователям отслеживать свои расходы и доходы, что позволяет выявлять финансовые риски и предотвращать перерасход;

- они могут напоминать о предстоящих платежах и сроках уплаты долгов, что способствует своевременному выполнению финансовых обязательств.

Упрощение процесса учета:

- приложения предлагают удобные инструменты для ввода и анализа данных, позволяя пользователям легко отслеживать свои финансовые операции;

- пользователи могут создавать отчеты о своих расходах и доходах, что помогает лучше понимать финансовое состояние.

Образование в области финансов:

- приложения могут включать обучающие материалы, направленные на повышение осведомленности о личных финансах, инвестициях и экономии;

- пользователи могут участвовать в вебинарах или получать доступ к ресурсам по финансовому планированию.

Автоматизация процессов:

- многие приложения предлагают функции автоматического импорта данных из банковских счетов, что упрощает процесс учета и минимизирует ошибки;

- пользователи могут настраивать автоматические уведомления о превышении бюджета или изменениях в финансовом состоянии.

В целом, домашняя бухгалтерия является неотъемлемой частью управления личными финансами, обеспечивая доступ к информации, инструментам и рекомендациям, которые способствуют улучшению финансового благополучия пользователей.

## **1.2 Существующие способы взаимодействия**

Эффективное ведение учета финансов является важным фактором для обеспечения контроля бюджета и оптимизации личных финансовых процессов. На данный момент существует несколько основных способов учета, которые могут использоваться в домашней бухгалтерии.

Одним из наиболее распространенных и традиционных способов ведения учета является использование бумажных журналов и форм. При этом способе вся информация о доходах, расходах, сбережениях и других финансовых данных фиксируется вручную в специальных журналах или бланках. Этот метод позволяет быстро вносить записи и имеет низкие первоначальные затраты.

Преимуществами бумажных журналов являются их простота и отсутствие необходимости в специализированном программном обеспечении. Однако этот способ имеет существенные недостатки, такие как ограниченная ёмкость журналов, риск потери или повреждения данных, сложность поиска и анализа информации, а также высокие затраты на хранение большого количества бумажных документов.

Многие пользователи переходят к использованию электронных таблиц, таких как Microsoft Excel, для ведения учета своих финансов. В таблицах могут содержаться списки доходов, расходов, бюджетов и другая важная информация. Этот способ позволяет автоматизировать некоторые процессы, упростить поиск и фильтрацию данных, а также обеспечить их резервное копирование.

Преимуществами использования электронных таблиц являются относительная простота, гибкость и возможность создания различных отчётов и аналитики. Однако недостатками являются ограниченные возможности по управлению большими объёмами данных, риск случайного удаления или повреждения информации, отсутствие надёжной системы контроля доступа и безопасности данных.

В современных условиях пользователи начинают внедрять специализированные приложения для домашней бухгалтерии. Такие приложения позволяют создавать структурированные базы данных для хранения информации о доходах, расходах, бюджетах и других финансовых данных, необходимых для эффективного управления личными финансами.

Преимуществами использования СУБД являются надёжность, масштабируемость, возможность управления большими объёмами данных, контроль доступа и обеспечение целостности и безопасности информации. Кроме того, СУБД предоставляют расширенные возможности для поиска, фильтрации, создания отчётов и аналитики.

Однако внедрение СУБД может быть затратным и требовать определённых технических знаний и ресурсов. Также необходимо обеспечить надлежащую техническую поддержку и обучение персонала для эффективного использования этих систем.

Некоторые пользователи используют комбинацию различных способов учета данных, например, ведение основных записей в электронных таблицах и дополнительных данных в бумажных журналах. Выбор подходящего способа или комбинации способов зависит от объёма данных и специфических потребностей в учёте и управлении информацией.

Важно отметить, что для повышения эффективности управления личными финансами необходимо внедрение современных информационных систем, таких как СУБД, которые обеспечивают надёжное хранение, управление и анализ данных, а также позволяют оптимизировать финансовые процессы.

## **1.3 Обоснование необходимости разработки нового приложения**

Несмотря на наличие различных способов учета и управления личными финансами, существующие решения имеют ряд недостатков и ограничений, которые могут привести к неудовлетворенности пользователей и снижению эффективности финансового контроля. Разработка нового специализированного приложения для домашней бухгалтерии может стать ключевым инструментом, направленным на улучшение качества учета и повышение удовлетворенности пользователей.

Использование современных технологий и удобного пользовательского интерфейса может сделать процесс управления финансами более привлекательным и эффективным. Кроме того, новое приложение может способствовать повышению финансовой грамотности пользователей и укреплению их уверенности в собственных финансовых решениях.

Важно отметить, что при разработке приложения необходимо учитывать потребности и предпочтения как пользователей, так и специалистов по финансам, чтобы обеспечить максимальную эффективность и удобство использования для всех заинтересованных сторон. Регулярный мониторинг отзывов и предложений пользователей поможет совершенствовать функциональность приложения и адаптировать его под изменяющиеся потребности рынка и технологические тренды.

В заключение можно сказать, что разработка нового приложения для управления личными финансами является актуальной и перспективной задачей, решение которой может принести значительные выгоды как для пользователей, так и для бизнеса. Повышение удобства учета, оптимизация процессов управления финансами и улучшение качества сервиса – вот лишь некоторые из потенциальных преимуществ, которые могут быть достигнуты благодаря внедрению современных цифровых технологий в сферу личных финансов.

# **2 Методология разработка приложения**

## **2.1 Выбор методологии разработки**

При разработке любого программного обеспечения, включая систему управления базами данных (СУБД) для аптеки, выбор подходящей методологии разработки имеет решающее значение для успешной реализации проекта. Существует несколько популярных методологий разработки программного обеспечения, каждая из которых имеет свои преимущества и недостатки.

Каскадная (Waterfall) методология – одна из старейших и наиболее традиционных методологий. Она предполагает последовательное выполнение этапов: сбор требований, проектирование, реализация, тестирование и внедрение. Каждый этап должен быть завершен полностью, прежде чем начнется следующий. Такой подход хорошо подходит для проектов с хорошо определенными и стабильными требованиями, однако он может быть неэффективным при необходимости внесения изменений или адаптации к меняющимся условиям.

Итеративная методология, такая как Rational Unified Process (RUP), предполагает разделение проекта на итерации, каждая из которых включает планирование, анализ, проектирование, реализацию и тестирование. Это позволяет получать работающие версии программного обеспечения на каждой итерации и вносить необходимые изменения на основе обратной связи. Однако данная методология может быть сложной в управлении и требует значительных усилий по документированию.

В последние годы все большую популярность приобретают гибкие методологии разработки, такие как Agile, Scrum и Kanban. Они основаны на принципах адаптивного планирования, итеративной разработки, постоянного взаимодействия с заказчиком и быстрого реагирования на изменения. Вместо детального планирования всего проекта, работа разбивается на небольшие итерации (спринты), в ходе которых реализуются наиболее приоритетные функции.

Для разработки СУБД для домашней бухгалтерии наиболее подходящей методологией является методология Scrum. Ее преимущества заключаются в следующем:

Гибкость и адаптивность: Scrum позволяет легко адаптироваться к изменяющимся требованиям или условиям, что важно при разработке сложных систем, таких как СУБД.

Прозрачность процесса: Использование информационных досок (Scrum boards) и ежедневных встреч команды (daily standups) обеспечивает прозрачность процесса разработки для всех участников.

Итеративная разработка: Разбиение работы на короткие итерации (спринты) позволяет постепенно наращивать функциональность СУБД и своевременно вносить корректировки.

Концентрация на наиболее важных задачах: В каждом спринте команда фокусируется на реализации наиболее приоритетных функций, что обеспечивает постоянный прогресс и ценность для заказчика.

Таким образом, выбор гибкой методологии Scrum для разработки СУБД для аптеки обеспечит необходимую гибкость, вовлеченность заказчика, прозрачность процесса и концентрацию на наиболее важных задачах, что повысит эффективность разработки и удовлетворенность конечным результатом.

## **2.2 Сбор и анализ требований**

Сбор и анализ требований являются одними из ключевых этапов при разработке любого приложения для управления личными финансами. От полноты и точности собранных требований зависит успех всего проекта и удовлетворенность конечных пользователей.

Для сбора требований можно использовать следующие методы:

1. Интервью - проведение структурированных или неструктурированных бесед с пользователями (например, людьми, управляющими своими финансами, финансовыми консультантами) для выяснения их потребностей, пожеланий и существующих проблем в учете финансов.
2. Анкетирование - подготовка и распространение анкет среди пользователей приложения для сбора количественных и качественных данных о требованиях к системе.
3. Наблюдение - изучение реальных процессов управления финансами путем наблюдения за поведением пользователей и их взаимодействием с существующими инструментами учета.
4. Анализ документации - изучение существующих материалов, таких как инструкции по ведению учета, советы по финансовому планированию и другие документы, связанные с управлением личными финансами.

После сбора исходных данных необходимо провести их анализ и сформулировать требования к разрабатываемой системе. Требования можно разделить на две основные категории:

1. Функциональные требования определяют функции и возможности, которые должны быть реализованы в приложении. Для системы домашней бухгалтерии функциональные требования могут включать:

* ведение базы данных о доходах, расходах и бюджетах;
* возможность поиска, добавления, редактирования и удаления записей в базах данных;
* формирование различных отчетов;
* контроль доступа и разграничение прав пользователей.

1. Нефункциональные требования определяют критерии качества, производительности, безопасности, удобства использования и другие характеристики системы. Для СУБД и приложения аптеки нефункциональные требования могут включать:

* отказоустойчивость и надежность;
* масштабируемость и производительность;
* простота и интуитивность пользовательского интерфейса;
* обеспечение безопасности и конфиденциальности данных;
* кроссплатформенность и совместимость с различными операционными системами.

Для эффективного управления требованиями при использовании гибкой методологии Scrum рекомендуется применять подход User Story Mapping и формулировать требования в виде коротких пользовательских историй (user stories). Каждая история должна описывать конкретный сценарий использования системы с точки зрения пользователя.

Пример пользовательской истории: "Как пользователь, я хочу иметь возможность быстро найти информацию о своих расходах за месяц, чтобы лучше планировать свой бюджет".

Сбор, анализ и четкое документирование требований являются критически важными факторами для успешной разработки СУБД и приложения, отвечающих потребностям аптеки и его клиентов.

## **2.3 Проектирование архитектуры СУБД**

Проектирование архитектуры является важным этапом в разработке любой информационной системы, включая СУБД и настольное приложение. Архитектура определяет общую структуру системы, взаимосвязи между ее компонентами и принципы их взаимодействия.

Архитектура программного обеспечения (ПО) — это совокупность основополагающих принципов, использованных при проектировании и реализации системы. Она обеспечивает концептуальную модель, которая определяет структуру, поведение и взаимодействие компонентов системы.

При проектировании архитектуры СУБД и приложения для аптеки необходимо учитывать следующие принципы:

1. Модульность - система должна быть разделена на отдельные модули или компоненты, каждый из которых отвечает за определенную функциональность. Это упрощает разработку, тестирование и дальнейшее сопровождение системы.
2. Расширяемость - архитектура должна предусматривать возможность добавления новых функций и компонентов без значительных изменений в существующем коде.
3. Производительность - система должна обеспечивать приемлемое быстродействие и отзывчивость, особенно при работе с большими объемами данных.
4. Безопасность - архитектура должна включать механизмы контроля доступа, аутентификации и авторизации пользователей, а также защиты данных от несанкционированного доступа и изменения.
5. Масштабируемость - архитектура должна позволять наращивать мощность и ресурсы системы по мере роста требований к ее производительности и объему обрабатываемых данных.

Для разрабатываемой СУБД и приложения наиболее подходящим является использование классической клиент-серверной архитектуры с многоуровневой организацией. В этой архитектуре система разделена на следующие уровни:

1. Уровень представления (клиентское приложение) - настольное приложение на C# Windows Presentation Foundation (WPF), предоставляющее пользовательский интерфейс для взаимодействия с системой.
2. Уровень бизнес-логики - компонент, реализующий основную бизнес-логику обработки данных и взаимодействия с базой данных.
3. Уровень данных (СУБД) - система управления базами данных MySQL, отвечающая за хранение, обработку и управление данными.

Такая архитектура обеспечивает четкое разделение ответственности между компонентами системы, повышает модульность и расширяемость, а также облегчает тестирование и сопровождение.

Проектирование базы данных является одним из наиболее важных аспектов разработки СУБД. Оно включает в себя следующие этапы:

1. Концептуальное проектирование - определение основных сущностей (объектов) предметной области и связей между ними. На этом этапе создается концептуальная модель данных, например, с использованием нотации Entity-Relationship Diagram (ER-диаграмма).
2. Логическое проектирование - преобразование концептуальной модели в логическую модель данных, соответствующую выбранной СУБД (в данном случае MySQL). На этом этапе определяются таблицы, атрибуты и ограничения целостности.
3. Физическое проектирование - создание физической схемы базы данных, оптимизированной для конкретной СУБД. На этом этапе определяются типы данных, индексы, стратегии хранения и другие параметры физической реализации базы данных.

Тщательное проектирование архитектуры СУБД и приложения для аптеки обеспечит надёжность, производительность, безопасность и масштабируемость системы, а также упростит ее дальнейшее сопровождение и развитие.

Была спроектирована концептуальная модель ER – диаграмма в нотации Чена. Она представлена на рисунке 2.1.

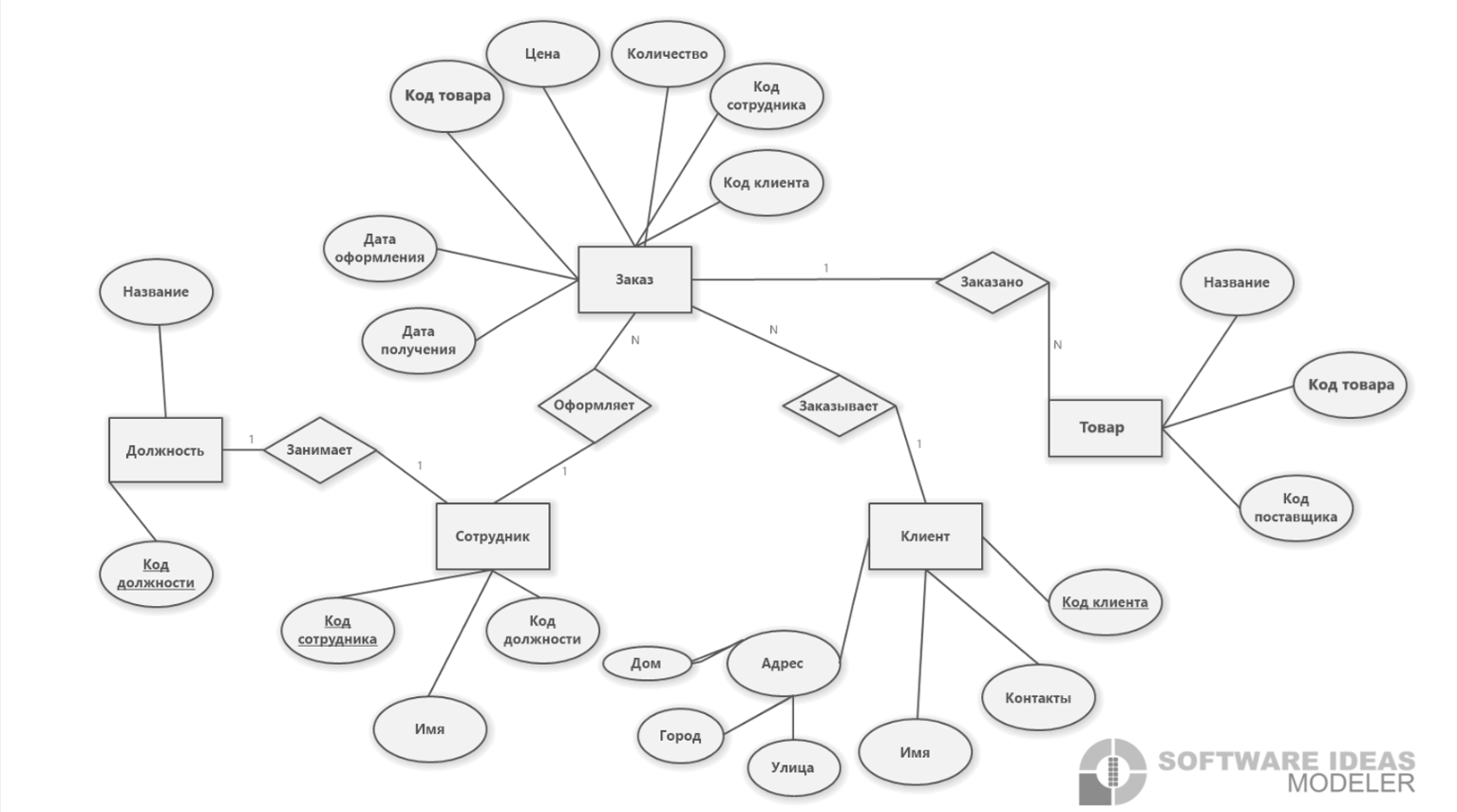


Рисунок 2.1 - ER – диаграмма в нотации Чена

Затем была спроектирована ER-диаграмма в нотации Мартина. Она представлена на рисунке 2.2.

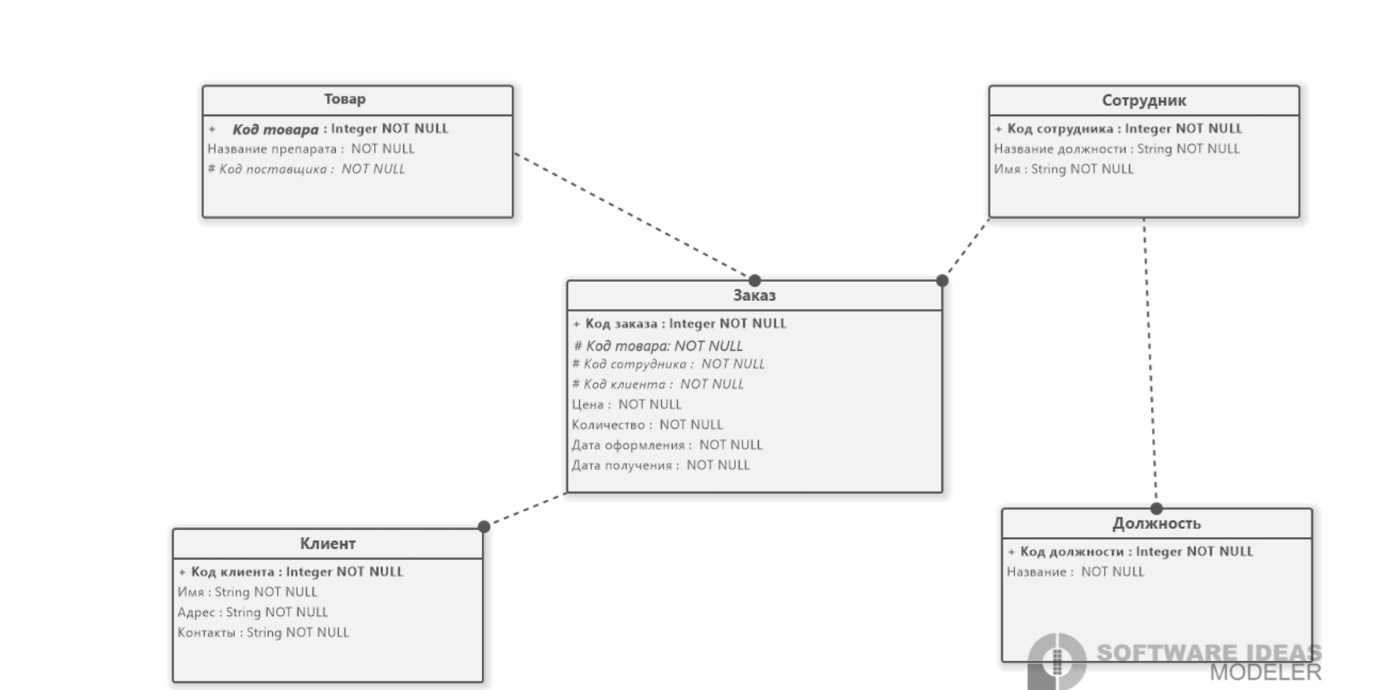


Рисунок 2.2 - ER-диаграмма в нотации Мартина

И также была спроектирована ER-диаграмма в нотации IDEF1X. Она представлена на рисунке 2.3.

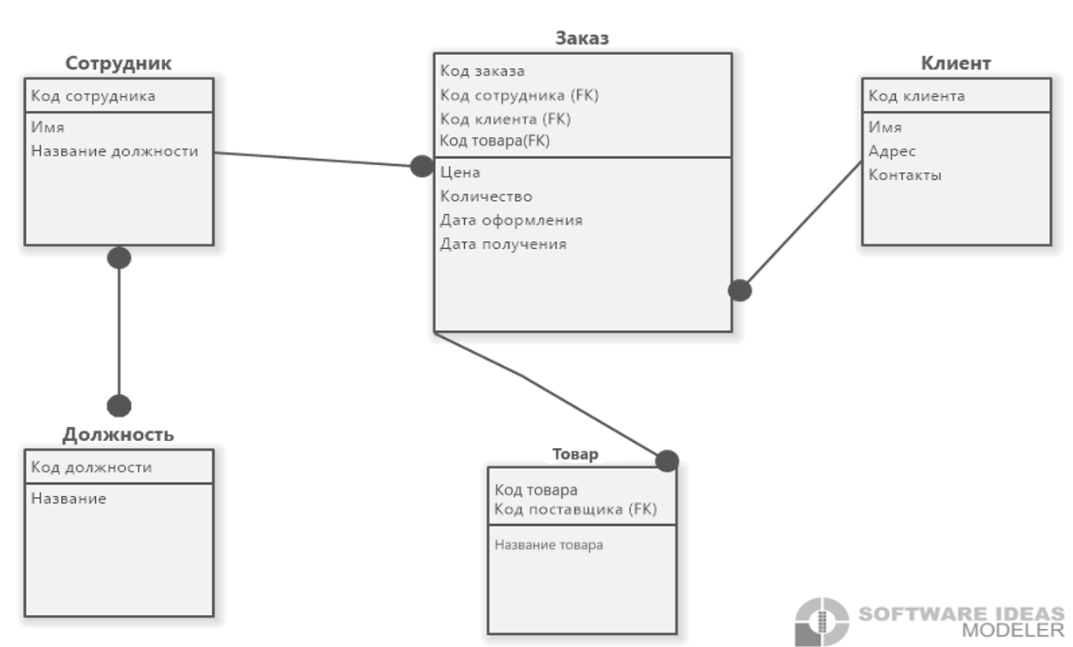


Рисунок 2.3 - ER-диаграмма в нотации IDEF1X

После разработки архитектуры СУБД можно перейти к ее планированию, что мы разберем в следующей главе.

## **2.4 Планирование разработки**

Для успешной разработки СУБД и настольного приложения необходимо выбрать подходящие технологии и инструменты, которые обеспечат эффективную реализацию, соответствие требованиям и удобство работы команды разработчиков. Рассмотрим выбранные технологии и инструменты, а также обоснование их выбора.

Система управления базами данных (СУБД) Одним из ключевых компонентов разрабатываемой системы является СУБД. Для данного проекта была выбрана MySQL - одна из наиболее популярных и широко используемых реляционных СУБД с открытым исходным кодом.

Обоснование выбора MySQL:

* высокая производительность и масштабируемость, позволяющая работать с большими объемами данных;
* кроссплатформенность и совместимость с различными операционными системами;
* наличие широкого сообщества поддержки и обилие документации, что упрощает обучение и устранение проблем;
* возможность интеграции с различными языками программирования и фреймворками.

Для разработки настольного приложения, которое будет взаимодействовать с СУБД и предоставлять пользовательский интерфейс, был выбран язык C# и фреймворк WPF.

C# — это современный объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он широко используется для создания различных типов приложений, включая настольные, веб-приложения и мобильные приложения.

WPF — это фреймворк для создания оконных приложений с графическим интерфейсом пользователя (GUI) для операционных систем Microsoft Windows. Он предоставляет обширный набор визуальных элементов управления и инструментов для разработки интуитивно понятных и удобных в использовании приложений.

Обоснование выбора C# и WPF:

* C# является одним из наиболее популярных и широко используемых языков программирования, что облегчает поиск квалифицированных разработчиков и доступность ресурсов для обучения;
* WPF предоставляет богатый набор визуальных элементов управления и инструментов для создания удобных пользовательских интерфейсов;
* наличие официальной поддержки и обширной документации от Microsoft;
* высокая производительность и надежность приложений, созданных с использованием C# и WPF.

Для управления исходным кодом и совместной работы над проектом была выбрана система контроля версий Git и хостинг репозиториев GitHub.

Git — распределенная система управления версиями, которая позволяет отслеживать изменения в коде, создавать ветки разработки, объединять изменения и эффективно работать в команде.

GitHub — веб-сервис для хостинга репозиториев Git, который предоставляет дополнительные возможности для совместной работы, отслеживания ошибок, управления проектами и обзора кода.

Обоснование выбора Git и GitHub:

* Git является de-facto стандартом для управления версиями кода в современной разработке программного обеспечения;
* GitHub обеспечивает централизованное хранилище кода, упрощает совместную работу и предоставляет дополнительные инструменты для управления проектом;
* наличие обширного сообщества пользователей и огромного количества ресурсов для обучения.

В качестве интегрированной среды разработки (IDE) была выбрана Visual Studio 2022 от Microsoft. Это мощная и функциональная среда, предназначенная для разработки приложений на различных языках программирования, включая C#, и предоставляющая широкий набор инструментов для повышения производительности разработчиков.

Выбор подходящих технологий и инструментов имеет решающее значение для успешной реализации проекта, обеспечения качества, производительности и удобства разработки. Сочетание MySQL, C#, WPF, Git, GitHub и Visual Studio 2022 предоставляет надежную и мощную платформу для создания СУБД и настольного приложения для домашней бухгалтерии.

# **3 Процесс разработки приложения**

# **3.1 Создание базы данных**

Разработка базы данных для СУБД аптеки была одной из ключевых задач проекта. Для ее реализации были выполнены следующие шаги:

Проектирование схемы базы данных.

На основе собранных требований и анализа предметной области была спроектирована концептуальная модель данных в виде ER-диаграммы. Она включала основные сущности, такие как "Клиент", "Препарат", "Заказ", "Должность", "Сотрудник" и связи между ними.

Затем концептуальная модель была преобразована в логическую реляционную модель базы данных, соответствующую СУБД MySQL. На этом этапе были определены таблицы, поля (атрибуты), типы данных и ограничения целостности (первичные и внешние ключи).

Создание физической базы данных.

После проектирования логической модели была создана физическая база данных в СУБД MySQL. Для этого было выполнено следующее:

1. Создана пустая база данных " Accounting" с использованием инструмента администрирования MySQL Microsoft SQL Server Management Studio.
2. Сгенерированы CREATE TABLE операторы на основе спроектированной логической модели для создания таблиц базы данных.
3. Выполнены операторы CREATE TABLE в MySQL Microsoft SQL Server Management Studio для создания всех необходимых таблиц: Client, Preparation, Order, Employee, JobTitle.
4. Определены первичные и вторичные ключи, а также другие ограничения целостности для обеспечения связности и непротиворечивости данных.

Наполнение базы данных тестовыми данными.

Для тестирования и отладки приложения база данных была наполнена тестовыми данными, имитирующими реальную информацию об аптеке. Это включало:

* создание тестовых записей клиентов с различными комбинациями имён, фамилий и контактной информации;
* добавление записей препаратов с разными названиями, ценами, характеристиками и id-номерами;
* создание записей заказов с различными статусами, датами заказа и связями с клиентами, препаратами и ценами;
* внесение информации о препаратах с описаниями и ценами;
* добавление записей сотрудников с разными ролями, именами, фамилиями и учётными данными.

Наличие тестовых данных позволило более эффективно протестировать функциональность приложения, проверить работу с различными сценариями и отладить возможные ошибки на ранней стадии разработки.

Таким образом, была создана полноценная база данных, готовая для использования в разрабатываемом приложении для аптеки. Тщательное проектирование и реализация базы данных обеспечили надежность, согласованность и целостность данных, а также заложили основу для дальнейшей разработки функциональности приложения.

## **3.2 Выбор фреймворка/библиотек**

После создания базы данных следующим важным этапом стала разработка клиентского приложения, предоставляющего пользовательский интерфейс для взаимодействия с СУБД автосервиса. Приложение было реализовано на языке C# с использованием фреймворка Windows Presentation Foundation (WPF).

Реализация основных функций.

На начальном этапе были реализованы основные функции приложения, необходимые для работы автосервиса:

1. Авторизация и аутентификация пользователей. Была создана форма авторизации, где пользователи (сотрудники аптеки) могут вводить свои учетные данные - имя пользователя и пароль. Реализована проверка введенных данных с информацией в базе данных таблицы Employee. В зависимости от роли пользователя (администратор, фармацевт) определялся доступ к различным функциям приложения.
2. Управление клиентами. Разработана форма для просмотра, добавления, редактирования и удаления информации о клиентах. Данные отображались в виде таблицы, связанной с таблицей Client в базе данных. Реализованы функции для выполнения операций CRUD (создание, чтение, обновление, удаление) над записями клиентов.
3. Управление препаратами. Создана форма для работы с информацией о заказываемых препаратов клиентами. Пользователи могли просматривать, добавлять, редактировать и удалять записи о препаратах, связанные с таблицей Preparate в базе данных.
4. Управление заказами. Разработана форма для создания и управления заказами. Пользователи могли создавать новые заказы, выбирать клиента и нужный препарат. Информация о заказах сохранялась в таблицах Order.

Разработка пользовательского интерфейса.

Для обеспечения удобного взаимодействия с приложением был разработан интуитивно понятный и привлекательный пользовательский интерфейс с использованием стандартных элементов управления Windows Presentation Foundation.

Главная форма приложения содержала меню с пунктами для перехода к различным функциям, таким как управление клиентами, препаратами, заказами, сотрудниками и должностями.

Для отображения данных использовались табличные представления (DataGridView), позволяющие просматривать, фильтровать и сортировать записи.

Формы для ввода и редактирования данных содержали текстовые поля, выпадающие списки и другие элементы для удобного заполнения информации.

Для повышения удобства использования были добавлены диалоговые окна для подтверждения действий, а также всплывающие сообщения об успешном выполнении операций или возникших ошибках.

Интеграция с базой данных.

Для взаимодействия клиентского приложения с базой данных MySQL использовалась библиотека MySql.Data.

Был реализован класс DatabaseConnection, инкапсулирующий логику подключения к базе данных и выполнения SQL-запросов.

В каждой форме приложения, требующей доступ к базе данных, создавался экземпляр класса DatabaseConnection, через который выполнялись необходимые SQL-запросы.

Для обеспечения безопасности использовались параметризованные запросы, предотвращающие SQL-инъекции и другие атаки на уровне базы данных.

Результаты выполнения запросов отображались в соответствующих элементах пользовательского интерфейса, таких как таблицы данных или текстовые поля.

Таким образом, был реализован полнофункциональный клиентский интерфейс для взаимодействия с базой данных аптеки, обеспечивающий удобство работы пользователей и эффективное управление информацией.

**3.3 Тестирование и отладка**

После завершения разработки основных функций клиентского приложения и его интеграции с базой данных, следующим важным этапом стало тестирование для выявления и устранения ошибок и неисправностей. Тестирование являлось неотъемлемой частью процесса разработки, обеспечивающей высокое качество и надежность приложения.

Модульное тестирование.

На начальном этапе проводилось модульное тестирование отдельных компонентов и функций приложения. Для этого были созданы специальные модульные тесты, охватывающие различные сценарии использования и граничные условия.

Модульные тесты включали в себя:

* тестирование методов класса DatabaseConnection для корректности выполнения SQL-запросов и обработки результатов;
* тестирование методов классов, реализующих бизнес-логику приложения (валидация данных, вычисления, преобразования и т.д.);
* тестирование компонентов пользовательского интерфейса (форм, элементов управления) на корректность отображения и обработки событий.

Модульное тестирование позволило выявить и устранить ошибки на уровне отдельных компонентов до их интеграции в общую систему.

Интеграционное тестирование.

После успешного прохождения модульного тестирования было проведено интеграционное тестирование для проверки корректного взаимодействия различных компонентов приложения между собой и с базой данных.

Интеграционное тестирование включало в себя:

* проверку процесса авторизации пользователей с различными ролями и привилегиями доступа;
* тестирование функций CRUD (создание, чтение, обновление, удаление) для таблиц Client, Preparation, Order, Employee, JobTitle и других сущностей базы данных;
* проверку создания и сохранения заказов на обслуживание с привязкой к клиентам, препаратам и выбранным функциям.

В ходе интеграционного тестирования были обнаружены и устранены ошибки, связанные с несогласованностью данных, и другими проблемами взаимодействия компонентов.

Тестирование пользовательского интерфейса.

Важной частью тестирования стала проверка удобства использования и корректности работы пользовательского интерфейса приложения. Тестирование пользовательского интерфейса включало:

* проверку отображения и расположения элементов управления на формах приложения;
* тестирование навигации между формами и функциональными областями приложения;
* проверку реакции приложения на различные действия пользователя (ввод данных, нажатие кнопок, изменение настроек и т.д.);
* тестирование отображения сообщений об ошибках и подсказок для пользователя.

В процессе тестирования пользовательского интерфейса были выявлены и устранены ошибки, связанные с некорректным отображением данных, неинтуитивным расположением элементов управления и другими проблемами, влияющими на удобство работы пользователей.

# **ЗАКЛЮЧЕНИЕ**

В рамках данного проекта была разработана информационная система «Бухгалтерия», включающая централизованное хранилище данных (базу данных) и настольное приложение с удобным пользовательским интерфейсом. База данных «Бухгалтерия» была спроектирована с использованием реляционной модели данных и включала связанные таблицы, хранящие информацию о препаратах, заказах, сотрудниках и т.д. Настольное приложение, разработанное с использованием принципов ориентированного на данные проектирования, обеспечивает удобный пользовательский интерфейс для доступа ко всем функциям системы.

Все поставленные задачи были выполнены. Основные результаты работы включают:

1. Анализ предметной области:

* были исследованы основные процессы и требования, связанные с управлением аптеки;
* определены ключевые сущности, такие как клиенты, препараты, заказы, должности и сотрудники.

1. Проектирование и создание базы данных:

* на основе концептуальной и логической моделей была разработана физическая схема базы данных;
* в СУБД MySQL были созданы необходимые таблицы с определением связей, первичных и внешних ключей для обеспечения целостности данных.

1. Разработка клиентского приложения:

* реализовано настольное приложение на C# с использованием фреймворка WРF;
* основные функции приложения включают авторизацию и аутентификацию пользователей, управление препаратами и сотрудниками, создание и управление заказами, а также просмотр истории заказов.

1. Интеграция с базой данных:

* для взаимодействия с базой данных MySQL была использована библиотека MySql.Data;
* реализованы методы для выполнения SQL-запросов и получения данных из базы.

1. Тестирование и отладка:

* проведено тщательное тестирование всех компонентов системы, включая модульное и интеграционное тестирование, тестирование пользовательского интерфейса;
* были выявлены и исправлены ошибки, что позволило обеспечить надежность и стабильность работы приложения.

Проект продемонстрировал преимущества использования гибкой методологии разработки (Scrum), что позволило эффективно управлять процессом разработки, своевременно реагировать на изменения требований и обеспечить высокое качество конечного продукта. Планирование разработки, разбиение задач на спринты и регулярные ревизии результатов способствовали достижению поставленных целей в установленные сроки.

Практическая значимость данного проекта заключается в создании удобного и функционального инструмента для управления аптек. Система позволяет автоматизировать и упростить многие процессы, связанные с обслуживанием клиентов, управлением заказами и ресурсами, что способствует повышению эффективности работы аптеки и улучшению качества обслуживания клиентов.

Использование современных информационных технологий и системного подхода к управлению информационной системой «Бухгалтерия» позволяет упорядочить все процессы, оцифровать и усовершенствовать ручные и автоматизированные операции обслуживания клиентов.

Таким образом цель курсового проекта была выполнена. Было создано удобное и функциональное приложение, которое позволит клиентам и администрации аптеки быстро и эффективно взаимодействовать.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Агальцов В.П. Базы данных. – М.: Мир, 2002. – 376 с
2. Бажин И.И. Информационные системы менеджмента. – М.: ГУ-ВШЭ,
3. Барановская Т.П., В.И. Лойко, М.И. Семенов, А.И. Трубилин. Информационные системы и технологии в экономике. Учебник. 2-е изд., доп. и перераб. М.: Финансы и статистика, 2005. – 416 с.
4. Ватсон К.А. C#. / К. Ватсон, М. Беллиназо, О. Корнс, Д. Эспиноза. - М.: Лори, 2005. - 863 с.
5. Гагарина Л.Г. Разработка и эксплуатация автоматизированных информационных систем: Учебное пособие / Л.Г. Гагарина. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 384 с.
6. Гвоздева В.А. Введение в специальность программиста: Учебник / В.А. Гвоздева. - 2-e изд., испр. и доп. - М.: ИД ФОРУМ: ИНФРА-М, 2011. - 208 с.
7. Голицына О.Л., Попов И.И. Программирование на языках высокого уровня: Учебное пособие / О.Л. Голицына, И.И. Попов. - М.: Форум, 2008. - 496 с.
8. Документация по языку C# // Microsoft URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 13.05.2024).
9. Емельянова Н.З., Партыка Т.Л., Попов И.И. Проектирование информационных систем: учеб.пособие / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. - М.: Форум, 2009. - 432 с.
10. Заботина Н.Н. Проектирование информационных систем: Учебное пособие / Н.Н. Заботина. - М.: ИНФРА-М, 2011. - 331 с.
11. Колдаев В.Д. Структуры и алгоритмы обработки данных: Учебное пособие / В.Д. Колдаев. - М.: ИЦ РИОР: НИЦ ИНФРА-М, 2014. - 296 с.
12. Култыгин О. П. Администрирование баз данных. СУБД MS SQL Server: учеб. пособие / О. П. Култыгин. - М.: МФПА, 2012. - 232 с
13. Немцова Т.И. Программирование на языке высокого уровня. Программир. на языке Object Pascal: Учеб.пос. / Т.И. Немцова и др; Под ред. Л.Г. Гагариной - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013 – 496 с
14. Нортроп Т.Е. Основы разработки приложений на платформе Microsoft.NET Framework. Учебный курс Microsoft. Перевод с англ./ Т. Нортроп, Ш. Уилдермьюс, Б. Райан. - М.: «Русская редакция», 2007. - 864 с.
15. Базы данных : учебное пособие / составители Т. Ж. Базаржапова [и др.]. — Улан-Удэ : Бурятская ГСХА им. В.Р. Филиппова, 2022. — 84 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/284240 (дата обращения: 13.05.2024).