### МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ

**Государственное профессиональное образовательное учреждение**

**«Воркутинский политехнический техникум»**

# КУРСОВОЙ ПРОЕКТ

По дисциплине МДК.05.02 Разработка кода

информационных систем

Разработка информационной системы

«Учёт расчетов за проживание в общежитии»

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2023

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc134270026)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc134270027)

[1.1 Платформа .NET 4](#_Toc134270028)

[1.2 Язык программирования С# 5](#_Toc134270029)

[1.3 Windows Presentation Foundation (WPF) 6](#_Toc134270030)

[1.4 СУБД SQL server 7](#_Toc134270031)

[1.5 Microsoft SQL Server Management Studio 8](#_Toc134270032)

[1.6 Entity Framework 9](#_Toc134270033)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 10](#_Toc134270034)

[2.1 Разработка диаграммы ERD 10](#_Toc134270035)

[2.2 Разработка базы данных 12](#_Toc134270036)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 16](#_Toc134270037)

[3.1 Разработка интерфейса информационной системы 16](#_Toc134270038)

[3.2 Программирование информационной системы 23](#_Toc134270039)

[ЗАКЛЮЧЕНИЕ 37](#_Toc134270040)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 38](#_Toc134270041)

# ВВЕДЕНИЕ

В среде программных продуктов очень много программ для учёта информации и финансов в офисе, в учреждении или на предприятии.

Основными особенностями специализированных продуктов, предназначенных для отслеживания информации и расчётов финансов – удобство, простота использования и многофункциональность. Разработчики предлагают удобный инструмент для решения большой монотонной работы за короткое время.

Часто имеющиеся функции АИС на рынке программных продуктов избыточны. Их наличие ведёт к повышению системных требований для обеспечения функционирования данных систем и к значительному увеличению расходов у потребителя на приобретение и эксплуатации данных программных продуктов.

Поэтому нужно создать программный продукт, который имеет необходимых минимальный функционал для учёта информации и финансов, который можно приобрести по доступной цене.

Объект курсовой работы – создание программы через различные инструменты программирования.

Предмет курсовой работы – создание с помощью программирования автоматизированной информационной системы «Учёт расчетов за проживание в общежитии».

Целью курсовой работы является создание информационной системы для автоматизированного учета проживания оплат жильцов общежития.

Для достижения цели проекта нужно выполнить следующие задачи:

1. на основании технического задания выбираются определённый инструментарий;
2. в соответствии с техническим заданием спроектировать базу данных;
3. разработать информационную систему.

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# Платформа .NET

**.NET Framework** — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для различных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Хотя **.NET Framework** является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства Windows, существуют независимые проекты (прежде всего это Mono и Portable.NET), позволяющие запускать программы .NET Framework на некоторых других операционных системах. В настоящее время **.NET Framework** развивается в виде **.NET**.

**.NET** (ранее известна как **.NET Core**) — это модульная платформа для разработки программного обеспечения с открытым исходным кодом. Совместима с такими операционными системами как Windows, Linux и macOS. Поддерживает следующие языки программирования: C#, Visual Basic .NET (частично) и F#.

NET основана на .NET Framework. Платформа .NET отличается от неё модульностью, кроссплатформенностью, возможностью применения облачных технологий, и тем, что в ней произошло разделение между библиотекой CoreFX и средой выполнения CoreCLR. CoreFX — это библиотека, интегрированная в .NET. Среди её компонентов: System.Collections, System.IO, System.Xml. CoreCLR — это среда выполнения, включающая в себя RyuJIT, встроенный сборщик мусора и другие компоненты.

**.NET** — модульная платформа. Каждый её компонент обновляется через менеджер пакетов NuGet, а значит можно обновлять её модули по отдельности, в то время как .NET Framework обновляется целиком. Каждое приложение может работать с разными модулями и не зависит от единого обновления платформы.

При помощи платформы .NET можно создать:

1. облачные приложения;
2. кроссплатформенные клиентские приложения;
3. приложения для Windows;
4. Приложения для других операционных систем.

# Язык программирования С#

**C#** (произносится си шарп) — объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core.

Visual C# — это реализация языка C# корпорацией Майкрософт. Поддержка Visual C# в Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и другим полезным, правильным классам, что существенно ускоряет цикл разработки. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Есть две основных версии, зачем Microsoft стала делать язык программирования C#:

1. Официальная: чтобы упростить разработку приложений под Windows. А то языки в то время были сложные и плохо адаптировались под разные Windows-компьютеры.
2. Правдоподобная: чтобы заменить Java, на который у Microsoft не было лицензии. В итоге получился почти такой же язык, но с интеграцией под Windows. Как и Java, он основан на языке C и легко запускается на любом устройстве. А название «Си» с решёткой как бы говорит: «У нас тут не копия Java, а новый язык в линейке „Си“».

Если коротко, этот язык:

1. Кросс-платформенный — запускается почти на любом железе.
2. Объектно-ориентированный — состоит из классов и объектов, которые умеют передавать свойства друг другу.
3. Постоянно развивается.
4. Дружит с экосистемой Windows — для этого и был написан.

# Windows Presentation Foundation (WPF)

**Windows Presentation Foundation** (**WPF**) — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework, использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

XAML представляет собой язык декларативного описания интерфейса, основанный на XML. Также реализована модель разделения кода и дизайна, позволяющая кооперироваться программисту и дизайнеру. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на элементы управления второго уровня, которые разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента, например, кнопки или текстового поля.

Для работы с WPF требуется любой .NET-совместимый язык. В этот список входит множество языков: C#, F#, VB.NET, C++, Ruby, Python, Delphi (Prism), Lua и многие другие. Для полноценной работы может быть использована как Visual Studio, так и Expression Blend.

Ниже приведен список некоторых наиболее существенных изменений, которые привнесла с собой технология WPF в мир программирования Windows-приложений:

1. Веб-подобная модель компоновки;
2. Богатая модель рисования;
3. Развитая текстовая модель;
4. Анимация;
5. Поддержка аудио и видео;
6. Стили и шаблоны;
7. Декларативный пользовательский интерфейс.

# СУБД SQL server

**SQL** (от англ. Structured Query Language — «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Является информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

1. Создание в базе данных новой таблицы; добавление в таблицу новых записей;
2. Изменение записей;
3. Удаление записей;
4. Выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
5. Изменение структур таблиц.

Появление баз данных позволило объединить разное понимание БД со стороны пользователей и системных администраторов. Неискушенные в технических деталях люди «видят» таблицы как некий перечень данных с колонками и строками. Системный подход включает файлы с табличными данными, связанными друг с другом согласно определенному алгоритму.

Клиентами БД являются прикладные программы, их интерфейс, различные интерактивные модули сайтов вроде калькуляторов и онлайн-редакторов. Но есть еще один компонент системы – СУБД. Он предназначен для ручного доступа к информации и позволяет извлекать данные, работать с ними в памяти сервера, в том числе с применением языка SQL.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования. При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

# Microsoft SQL Server Management Studio

**SQL Server Management Studio** (**SSMS**) — утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера.

Среда SSMS предназначена для доступа к службам Службы Analysis Services, Integration Services и Reporting Services, а также для их настройки, администрирования и управления ими.

Главным инструментом SSMS является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

Среда SSMS предоставляет единую комплексную служебную программу, которая сочетает в себе обширную группу графических инструментов с рядом многофункциональных редакторов скриптов для доступа к SQL Server для разработчиков и администраторов баз данных всех профессиональных уровней.

Основной состав функционала SSMS:

1. Подключение к службам SQL Server
2. Обозреватель объектов
3. Обозреватель шаблонов
4. Редактор SQL кода
5. Просмотр плана выполнения запроса
6. Обозреватель решений
7. Конструктор таблиц
8. Конструктор баз данных (Диаграммы баз данных)
9. Конструктор запросов и представлений
10. Просмотр свойств объектов
11. Мастер создания скриптов
12. Управление безопасностью SQL Server
13. Присоединение и отсоединение баз данных
14. Создание резервных копий баз данных и восстановление баз данных из архива
15. Создание связанных серверов (Linked Server)
16. Монитор активности SQL Server
17. Профилировщик XEvent

# Entity Framework

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Первоначально он поставлялся как неотъемлемая часть .NET Framework, однако, начиная с Entity Framework версии 6.0, он поставлялся отдельно от .NET Framework.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Entity Framework представляет собой набор технологий в ADO.NET, который поддерживает разработку программных приложений, ориентированных на данные. Архитекторы и разработчики приложений, ориентированных на данные, обычно борются с необходимостью достижения двух совершенно разных целей. Они должны моделировать сущности, связи и логику бизнес-задач, которые они решают, а также должны работать с обработчиками данных, используемыми для хранения и извлечения данных. Данные могут охватывать несколько систем хранения, каждая из которых имеет свои собственные протоколы; даже приложения, работающие с одной системой хранения, должны уравновешивать требования системы хранения данных с требованиями написания эффективного и поддерживаемого кода приложения. Эту проблему обычно называют «несоответствием объекта и реляционного импеданса».

Многие инструменты объектно-реляционного сопоставления (ORM) (также известные как «объектно-реляционные менеджеры») были разработаны, чтобы позволить разработчикам работать с данными в виде объектов и свойств, специфичных для предметной области, таких как клиенты и адреса клиентов, без необходимости беспокоиться о базовых таблицах и столбцах базы данных, где хранятся эти данные. С ORM разработчики могут работать на более высоком уровне абстракции, когда они имеют дело с данными, и могут создавать и поддерживать ориентированные на данные приложения с меньшим количеством кода, чем в традиционных приложениях. Entity Framework — это решение ORM, которое в настоящее время продвигается для использования в стеке разработки Майкрософт.

# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

# Разработка диаграммы ERD

Диаграмма ERD или ER-модель (от англ. Entity-Relationship model, модель «сущность — связь») — модель данных, позволяющая описывать концептуальные схемы предметной области. Это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования, разработки программного обеспечения и информационных систем для бизнеса.

Будет использована графическая нотация (стиль оформления диаграммы) Crow’s Foot. Данная нотация была предложена Гордоном Эверестом под названием “inverted arrow” («перевёрнутая стрелка»), однако сейчас чаще называемая “Crow’s Foot”, или “crow’s foot” («воронья лапка») или “fork” («вилка»).

Согласно данной нотации, сущность изображается в виде прямоугольника, содержащего её имя, выражаемое существительным. Имя сущности должно быть уникальным в рамках одной модели. При этом, имя сущности — это имя типа, а не конкретного экземпляра данного типа. Экземпляром сущности называется конкретный представитель данной сущности.

Связь изображается линией, которая связывает две сущности. Степень конца связи указывается графически, множественность связи изображается в виде «вилки» на конце связи. Модальность связи так же изображается графически — необязательность связи помечается кружком на конце связи. Именование обычно выражается одним глаголом в изъявительном наклонении настоящего времени: «имеет», «принадлежит» и т. д.; или глаголом с поясняющими словами: «включает в себя», и т. п. Наименование может быть одно для всей связи или два для каждого из концов связи. Во втором случае, название левого конца связи указывается над линией связи, а правого — под линией. Каждое из названий располагаются рядом с сущностью, к которой оно относится.

Атрибуты сущности записываются внутри прямоугольника, изображающего сущность, и выражаются существительным в единственном числе (возможно, с уточняющими словами). Среди атрибутов выделяется ключ сущности — не избыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности.

Создадим ERD диаграмму для создания представления об базе данных для автоматизированной информационной системе «Учёт расчетов за проживание в общежитии».

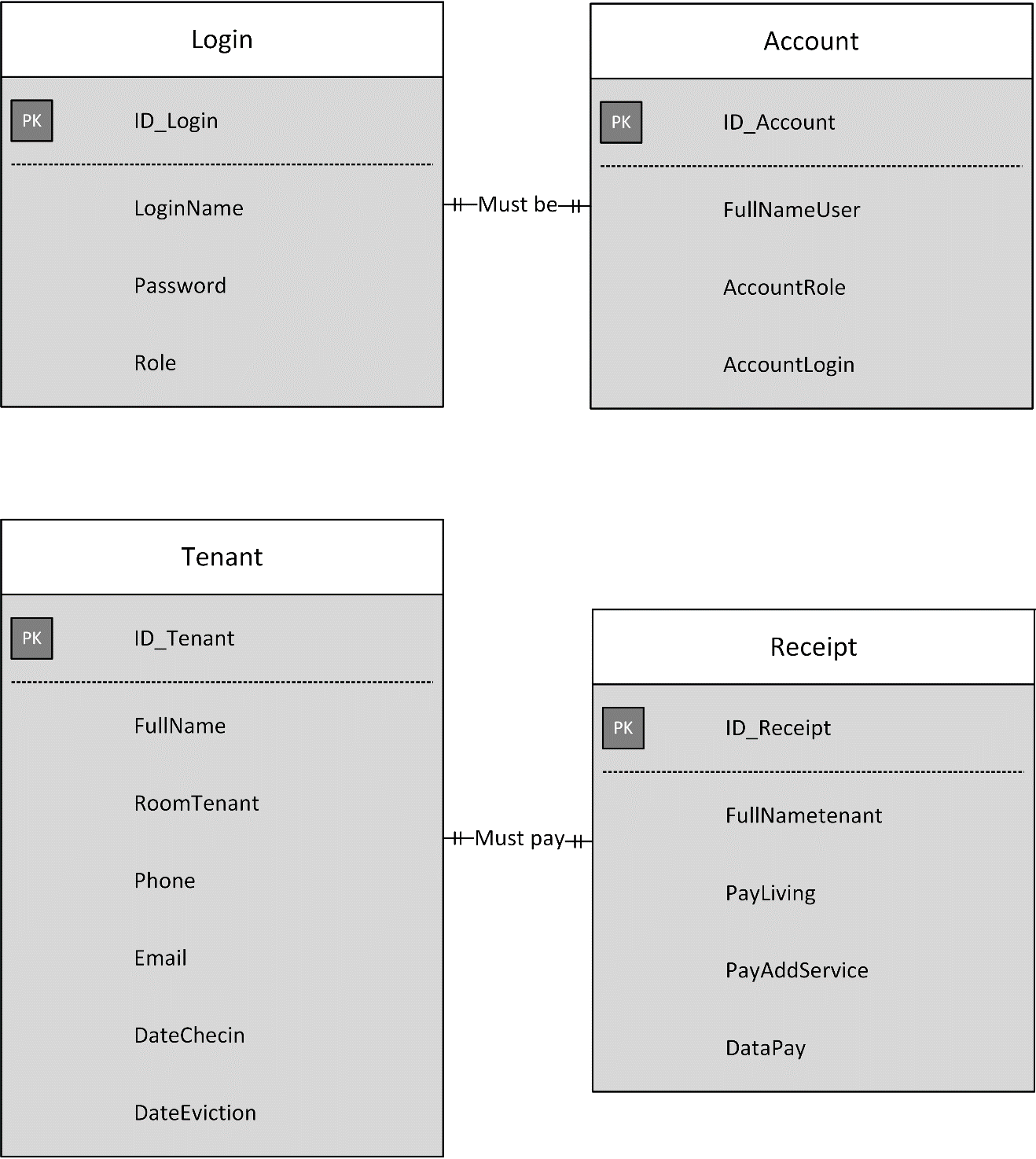


Рис. 2.1. ERD-диаграмма «Учёт расчетов за проживание в общежитии»

В данной ERD-диаграмме есть следующие сущности: «Пользователь» («Account»), «Логин» («Login»), «Житель» («Tenant») и «Квитанция» («Receipt»).

Сущность «Account» имеет следующие атрибуты: ID\_Account – индификайионный номер сущности, FullNameUser – ФИО пользователя системы, AccountLogin – внешний ключ для связки с сущностью Login для связывания пользователя с его логином, паролем и ролью пользователя. «Account» должен иметь один «Login».

Сущность «Login» имеет следующие атрибуты: ID\_Login – индификационный номер сущности, LoginName – логин пользователя для входа в систему, Password – пароль пользователя для входа в систему, Role – роль пользователя в системе. «Login» должен быть связан с одним «Account».

Сущность «Tenant» имеет следующие атрибуты: ID\_Tenant – индификационный номер сущности, FullName – ФИО жильца, RoomTenant – комната жильца, Phone- телефон жильца, Email – почта жильца, DateChecin – дата заселения жильца, DateEviction – дата выселения жильца. «Tenant» должен платить один «Receipt».

Сущность «Receipt» имеет следующие атрибуты: ID\_Receipt – индификационный номер сущности, FullNametenant – ФИО жильца для квитанции, PayLiving – оплата жильца за проживание, PayAddService – оплата жильца за дополнительные услуги, DataPay – дата оплаты квитанции. «Receipt» должен быть оплачен одним «Tenant».

# Разработка базы данных

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или просто базой данных.

Для базы данных обычно требуется комплексное программное обеспечение, которое называется системой управления базами данных (СУБД). СУБД служит интерфейсом между базой данных и пользователями или программами. СУБД обеспечивает контроль и управление данными, позволяя выполнять различные административные операции, такие как мониторинг производительности, настройка, а также резервное копирование и восстановление.

В качестве примеров популярного программного обеспечения для управления базами данных, или СУБД, можно назвать MySQL, Microsoft Access, 1.5, Microsoft SQL Server Management Studio, СУБД Oracle Database и dBASE.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов, формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

Создадим базу данных для автоматизированной информационной системы «Учёт расчетов за проживание в общежитии». Воспользуемся программой Microsoft SQL Server Management Studio для этой задачи.

Подключимся к серверу базы данных. Именно здесь будет создаваться база данных для информационной системы «Учёт расчетов за проживание в общежитии». Вводим необходимые данные и нажимаем на кнопку «Соединить».

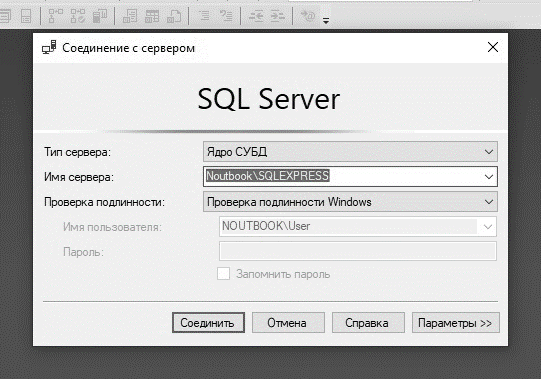


Рис. 2.2. Окно подключения к серверу в Microsoft SQL Server Management Studio

Создадим базу данных под именем «dormitoryManagerBD». Добавим таблицы и в них же параметры (выбираем тип данных у полей таблицы).

После добавляем связи между таблицами. User подключаем к Login, так как пользователь не сможет войти без логина и пароля, а Tenant связываем с Receipt, так в квитанции нужно будет указывать данные жильца.

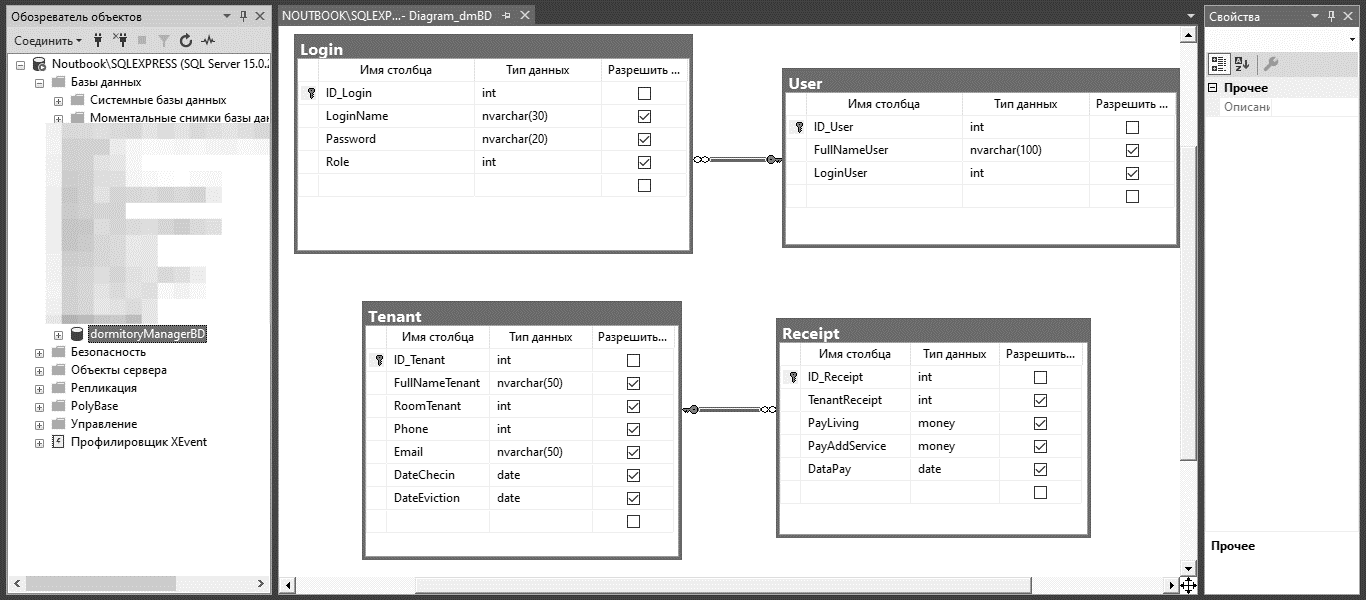


Рис. 2.3. Построение диаграммы таблиц в базе данных «dormitoryManagerBD»

В итоге получилась диаграмма двух связный пар таблиц. Одна пар связана с пользователями системы, а другая с жильцами общежития.

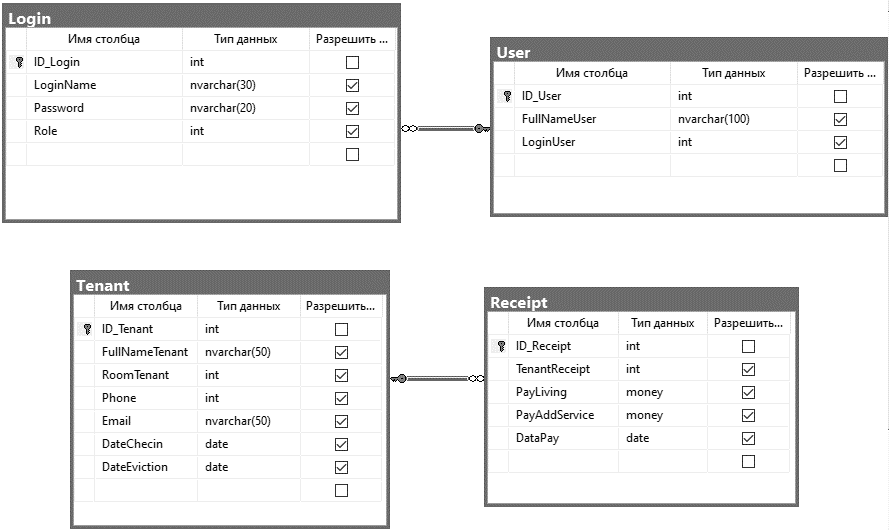


Рис. 2.4. Диаграмма таблиц в базе данных «dormitoryManagerBD»

Основа базы данных была создана. Теперь осталось дополнить её нужной нам информацией для использования в информационной системе «Учёт расчетов за проживание в общежитии».

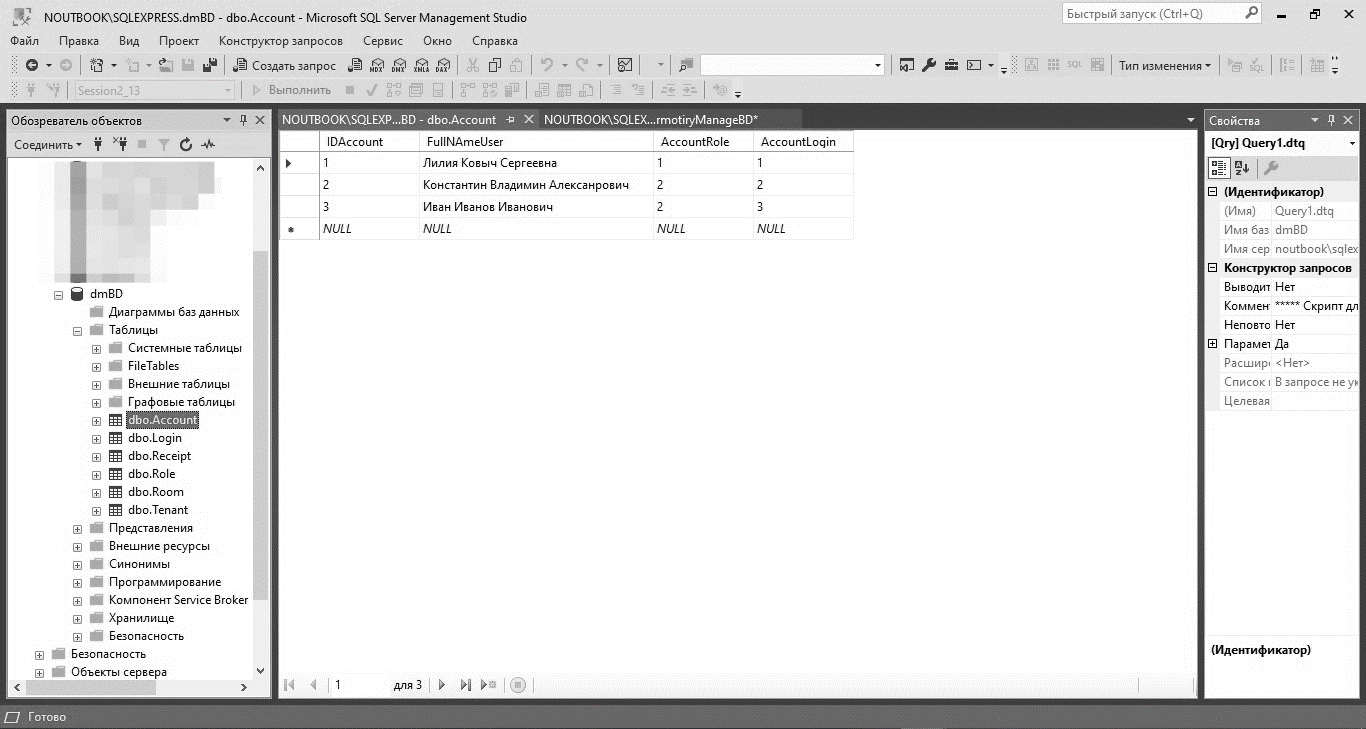


Рис. 2.5. Заполнение данными таблицы «Login»

Добавим в базу данных данные (рис 2.4). В каждой таблице добавим примерную информацию для использования в информационной системе.

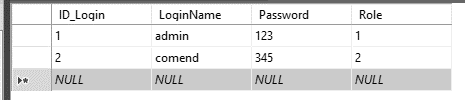


Рис. 2.6. Данные таблицы «Login»

Теперь база данных «dormitoryManagerBD» готова к использованию в автоматизированной информационной системы «Учёт расчетов за проживание в общежитии».

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Разработка интерфейса информационной системы

Интерфейс в информационной системе (ИС) - это точка взаимодействия между пользователями или другими системами и самой ИС. Он определяет, какие данные могут быть получены и отправлены, а также как пользователи будут использовать различные функции и возможности, которые предоставляются в рамках ИС.

В ИС интерфейс может включать в себя графический пользовательский интерфейс (GUI), веб-интерфейс, интерфейсы программирования приложений (API) и другие типы интерфейсов. Кроме того, он может быть настроен и управляем администратором системы, чтобы обеспечить наилучшую эффективность и удобство использования для конечных пользователей.

Главная задача интерфейса в ИС - создать для пользователей доступный и интуитивно понятный портал для выполнения задач, связанных с использованием ИС. Чтобы достичь этой цели, интерфейс должен иметь юзабилити (используемость).

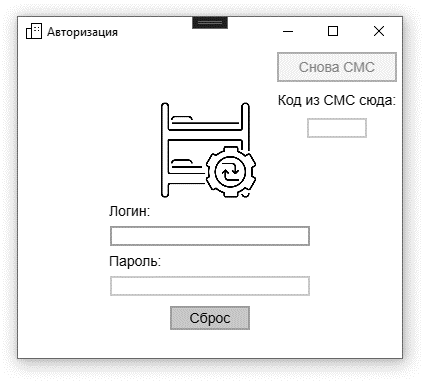


Рис. 3.1. Окно авторизации

Сначала создадим интерфейс авторизации автоматизированной информационной системы «Учёт расчетов за проживание в общежитии» (рис 3.1.).

Создаём его в Microsoft Visual Studio при помощи xaml кода.

XAML (eXtensible Application Markup Language) - это язык разметки, который используется для описания пользовательского интерфейса в приложениях WPF, Silverlight и Universal Windows Platform (UWP). Он позволяет разработчикам создавать графические интерфейсы, используя декларативный подход, что делает код более понятным и удобочитаемым. XAML-файлы могут содержать элементы пользовательского интерфейса, такие как кнопки, текстовые поля, изображения и многое другое, а также свойства элементов, их расположение и стили. XAML также поддерживает привязку данных, анимацию, шаблоны и другие возможности, которые делают процесс создания пользовательского интерфейса более эффективным и гибким.

Приведём в пример код дизайна окна авторизации.

XAML код окна авторизации:

<Window x:Class="AppDormitoryManager.MainWindow"

…

Title="Авторизация" Height="350" Width="400">

<Grid>

<StackPanel HorizontalAlignment="Right" VerticalAlignment="Top">

<Button x:Name="btnSMSgen" Content="Снова СМС" Margin="5" Height="30" Width="120" VerticalAlignment="Center" HorizontalAlignment="Center" Click="btnSMScode\_Click"/>

<TextBlock Text="Код из СМС сюда:" VerticalAlignment="Center" HorizontalAlignment="Center" Margin="5"/>

<TextBox x:Name="tbAccessCode" IsEnabled="False" KeyDown="AccessCodeTextBox\_KeyDown" Margin="5" VerticalAlignment="Center" HorizontalAlignment="Center" Width="60"/>

</StackPanel>

<StackPanel VerticalAlignment="Bottom" HorizontalAlignment="Center" Margin="20">

<StackPanel HorizontalAlignment="Center">

<Image Source="IMG/iconavto.png" Height="95" Width="110" VerticalAlignment="Top" HorizontalAlignment="Left"/>

</StackPanel>

<Label Content="Логин:"/>

<TextBox x:Name="txtLogin" KeyDown="LoginTextBox\_KeyDown" Width="200" Margin="6 2" HorizontalAlignment="Left" BorderThickness="2"/>

<Label Content="Пароль:"/>

<PasswordBox x:Name="txtPassword" IsEnabled="False" KeyDown="PasswordBox\_KeyDown" Margin="6 2" Width="200" HorizontalAlignment="Left"/>

<Button Name="CancelButton" Content="Сброс" Click="btnCancel\_Click" Margin="8" Width="80" Height="24"/>

</StackPanel>

<Button Visibility="Hidden" Name="LoginButton" Content="Вход" Click="btnLogin\_Click" VerticalAlignment="Center" IsEnabled="False" Margin="8" Width="80" Height="24"/>

</Grid>

</Window>

Данный код представляет собой графический интерфейс пользовательской формы "Авторизация" на платформе Windows, созданной на языке разметки XAML. Форма содержит несколько элементов управления, таких как кнопки, текстовые поля и метки, расположенные в двух вертикальных StackPanel'ах и одной кнопке, скрытой изначально.

Первый StackPanel, расположенный справа сверху, содержит кнопку "Снова СМС" для повторной отправки SMS-кода, который необходим для получения доступа к приложению. Также в этом StackPanel'е находится текстовое поле "Код из СМС сюда:", предназначенное для ввода полученного кода, и текстовое поле "tbAccessCode", которое отключено и используется для программного доступа к коду, введенному пользователем.

Второй StackPanel, расположенный внизу по центру, содержит изображение логотипа программы, метки "Логин:" и "Пароль:", а также текстовые поля "txtLogin" и "txtPassword" для ввода логина и пароля соответственно. Элемент "txtPassword" является элементом PasswordBox, то есть поле для ввода пароля, скрывающее введенные символы. Также в этом StackPanel'е расположена кнопка "Сброс" для очистки введенных данных.

Код содержит также обработчики событий "KeyDown" для полей "tbAccessCode", "txtLogin" и "txtPassword", которые позволяют пользователю нажимать клавишу "Enter" для перехода между полями и ввода данных. Кроме того, есть обработчик события "btnCancel\_Click" для кнопки "Сброс", который очищает все поля и отменяет авторизацию.

Приложение использует XAML для создания интерфейса авторизации, состоящего из элементов StackPanel, Button, TextBlock, TextBox, Label и PasswordBox. Элементы расположены на сетке, которая позволяет управлять их положением и размерами. XAML-код позволяет легко изменять элементы пользовательского интерфейса, не затрагивая логику приложения.

Заблокированные статусы некоторых строк и кнопки были сделаны через C# код авторизации этого окна.

Общий дизайн всех элементов окна системы будет запрограммирован в файле «DictionaryADM.xaml» - в словаре стилей элементов на экране. В других случаях редактируем XAML код окна под необходимые условия.

После успешной авторизации пользователь попадает общее для всех видов пользователей окно – «Главное меню» (рис 3.2.), в котором есть три кнопки.

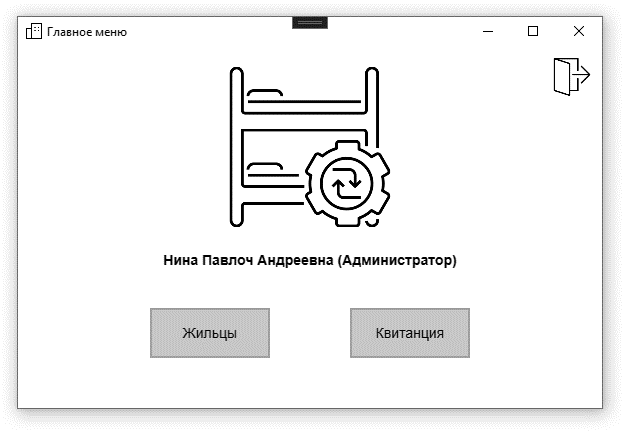


Рис. 3.2. Главное меню системы для пользователя

По центру окна изображён логотип программы. Под логотипом есть строчка об пользователе, который пользуется системой на данный момент.

Кнопка «Жильцы» открывает окно «Лист жильцов» со списком жильцов на данный момент. Для разных ролей системы это окно будет по-разному работать.

Кнопка «Квитанция» открывает окно создания квитанции для жильцов.

Изображение двери в правом углу экрана – кнопка закрытия системы.

У всех ролей в окне «Лист жильцов» показан список жильцов и их данные: ФИО, комната, номер телефона, электронная почта, дата заселения, дата выселения, а также помимо данных о жильцах есть кнопка «Назад» для возврата в «Главное меню».

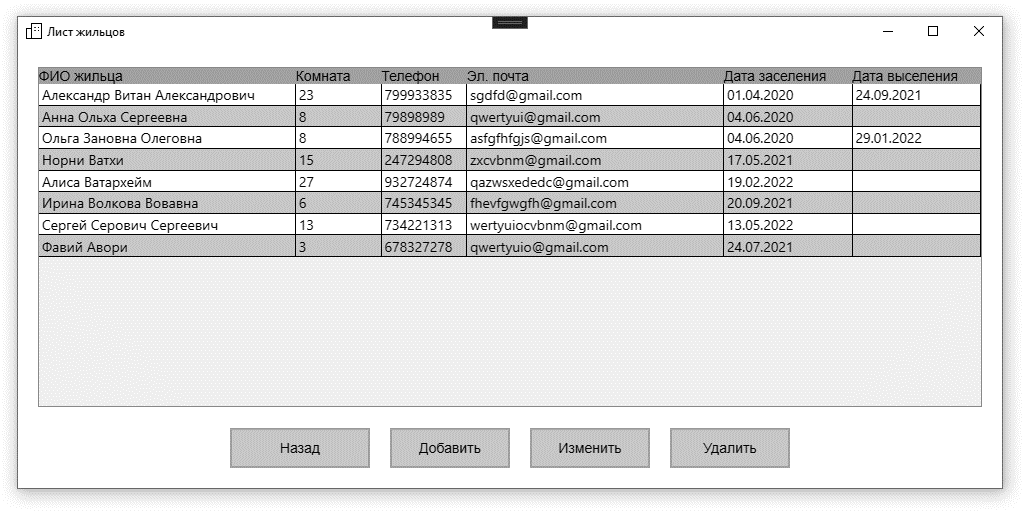


Рис. 3.3. Окно «Лист жильцов» для администратора

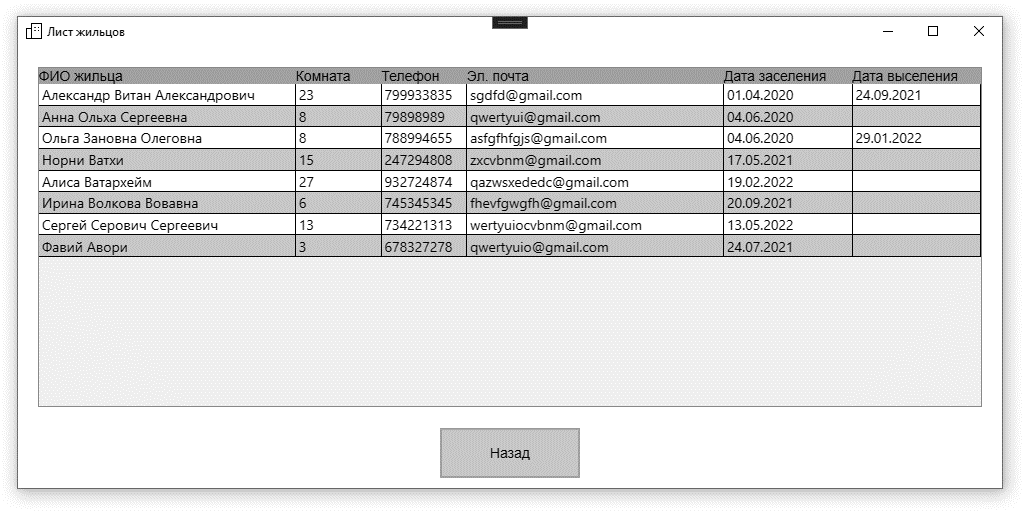


Рис. 3.4. Окно «Лист жильцов» для Коменданта

Добавлять, изменять и удалять данные в этой таблице может только Администратор – для него для этих задач доступны соответствующие кнопки. (рис 3.3.). Комендант сможет только смотреть на данные (рис 3.4.).

Если нажать на кнопку «Добавить» или «Изменить» в окне листа жильцов общежития, то откроется окно для создания и имения данных о жильце (рис. 3.5.). В этом окне или создают новые записи об жильце или изменяют уже имеющиеся данные в базе данных.

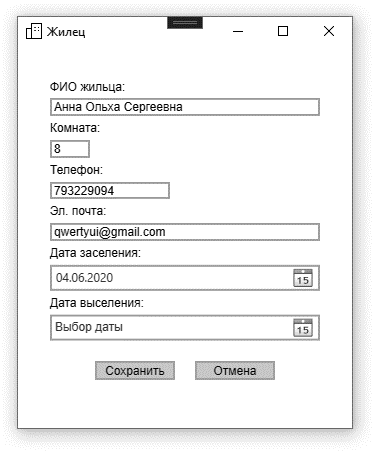


Рис. 3.5. Окно «Жилец» для администратора

Если нажать на кнопку «Добавить» в окне «Лист жильцов», то поля в этом окне будут пусти, а если из списка жильцов выбрать одного и нажать на кнопку «Изменить» в окне «Лист жильцов», то поля в этом окне будут заполнены данными из базы данных «dormitoryManagerBD».

В любом из этих процессов мы нажимаем на кнопку «Сохранить» для сохранения изменений или можем нажать на кнопку «Отмена» для закрытия окна «Жильца».

Если нажать на «Квитанция» в Главном меню, откроется окно создания квитанций для жильцов (рис 3.6). И Комендант, и Администратор могут создавать в нём квитанции.

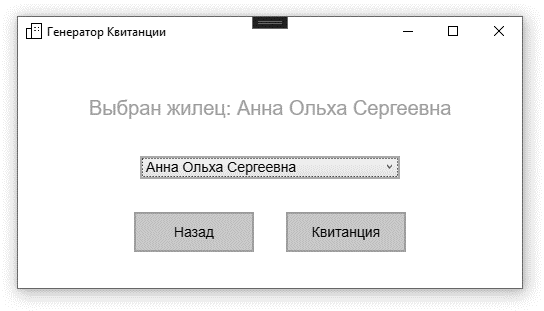


Рис. 3.6. Окно «Создание квитанции»

В этом окне пользователь выбирает нужного человека из выпадающего списка. После нажимаем кнопку «Квитанция», выбираем место сохранения на компьютере и получаем квитанцию для жильца в формате Word файла. Тут также есть кнопка «Назад» для возврата в «Главное меню».

Вот таким получился интерфейс системы по автоматизированной информационной системе «Учёт расчетов за проживание в общежитии».

# Программирование информационной системы

Разработка информационной системы «Учёт расчетов за проживание в общежитии» является одним из актуальных задач в сфере образования и программирования. Система должна обеспечивать учет платежей жильцов за проживание в общежитии, а также предоставлять информацию о задолженности и оплате.

Для решения этой задачи была разработана программа на языке программирования C#, которая позволяет авторизоваться в системе, просматривать информацию и создавать отчёты.

Программа состоит из трех основных модулей: модуля авторизации, модуля просмотра данных и модуля создания отчётов. Модуль авторизации позволяет пользователю войти в систему, используя свой логин, пароль и сгенерированный рандомном код и переходит к другим модулям. После успешной авторизации пользователь получает доступ к модулю просмотра информации, где может узнать о жильцах, и к модулю созданий отчётов, в котором создаются квитанции.

**Модуль авторизации**

Алгоритм авторизации будет состоять из нескольких этапов. Сотрудник сначала вводит свой логин и пароль. При вводе пароля сотрудником и нажатии клавиши Enter, на служебный телефон отправляется СМС с единоразовым кодом доступа. Далее, сотрудник должен ввести полученный код и получить доступ к необходимому функционалу.

Для реализации данного функционала с условием эмуляции работы с СМС, система должна работать следующим образом:

1. При открытии окна активны только поле для ввода логина сотрудника и кнопка “Сброс”.

2. После ввода логина сотрудника и нажатию Enter происходит проверка логина в базе данных.

3. Если логин сотрудника есть в базе данных, поле для ввода пароля становится активным с установленным курсором внутри него.

4. Если логин сотрудника в базе отсутствует, появляется сообщение об ошибке.

5. После ввода правильного пароля сотрудником и нажатия на Enter, открывается модальное окно с сгенерированным кодом доступа из 4 чисел.

6. Пользователь должен ввести полученный код в строку для кода в окне авторизации и авторизоваться в течение 10 секунд после закрытия окна с кодом.

7. Если в течение 10 секунд код не был введен, пользователь должен нажать кнопку «Генератор СМС» для повторной отправки кода.

8. Если введенный пароль неверный, то генерация кода доступа не происходит, и система сообщает пользователю об ошибке.

9. Для очистки всех полей ввода, пользователь может использовать кнопку "Сброс".

10. После успешной авторизации сотруднику будет выведено сообщение с названием его роли и будет открыто общее окно для всех типов пользователей.

С# код окна авторизации по вышесказанному описанию:

using System;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

namespace AppDormitoryManager

{

public partial class MainWindow : Window

{

private dormitoryManagerBDEntities \_db = new dormitoryManagerBDEntities();

private int \_accessCode;

private bool \_isCodeValid = true;

public MainWindow()

{

InitializeComponent();

// Блокируем доступ ко всем элемнтам окна, кроме строки Логина и кнопки "Сброс"

LoginButton.IsEnabled = false;

txtPassword.IsEnabled = false;

tbAccessCode.IsEnabled = false;

btnSMSgen.IsEnabled = false;

}

// Основа аторизации

private void btnLogin\_Click(object sender, RoutedEventArgs e)

{

VerifyUser();

}

//Кнопка броса данных в окне

private void btnCancel\_Click(object sender, RoutedEventArgs e)

{

txtLogin.Clear();

txtPassword.Clear();

tbAccessCode.Clear();

LoginButton.IsEnabled = false;

txtPassword.IsEnabled = false;

tbAccessCode.IsEnabled = false;

btnSMSgen.IsEnabled = false;

}

// Проверка логина

private void LoginTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

var query = from login in \_db.Logins

where login.LoginName == txtLogin.Text

select login;

var userLogin = query.FirstOrDefault();

if (userLogin != null)

{

txtPassword.IsEnabled = true;

txtPassword.Focus();

}

else

{

MessageBox.Show("Логин не найден.");

}

}

}

// Проверка пароля

private void PasswordBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

VerifyUser();

}

}

// Часть авторизации и сохранение необходимых данных

private async void VerifyUser()

{

var query = from user in \_db.Users

join login in \_db.Logins on user.LoginUser equals login.ID\_Login

where login.LoginName == txtLogin.Text && login.Password == txtPassword.Password

select new { user.FullNameUser, login.Role };

var userInfo = query.FirstOrDefault();

if (userInfo != null)

{

GenerateAccessCode();

tbAccessCode.IsEnabled = true;

tbAccessCode.Focus();

btnSMSgen.IsEnabled = true;

LoginButton.IsEnabled = true;

await Task.Delay(10000);

\_isCodeValid = false;

}

else

{

MessageBox.Show("Неверный логин или пароль.");

}

}

// Проверка кода доступа и открытие доступа к функционалу системы

private void AccessCodeTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

if (\_isCodeValid && tbAccessCode.Text == \_accessCode.ToString())

{

var query = from user in \_db.Users

join login in \_db.Logins on user.LoginUser equals login.ID\_Login

where login.LoginName == txtLogin.Text && login.Password == txtPassword.Password

select new { user.FullNameUser, login.Role };

var userInfo = query.FirstOrDefault();

string role = userInfo.Role == 1 ? "Администратор" : "Комендант";

MessageBox.Show($"Добро пожаловать, {userInfo.FullNameUser}! Ваша роль: {role}.");

WindowTenants tenantsWindow = new WindowTenants(userInfo.FullNameUser, userInfo.Role);

WindowGenerReceipt grWindow = new WindowGenerReceipt(userInfo.FullNameUser, userInfo.Role);

WindowGeneral generalWindow = new WindowGeneral(userInfo.FullNameUser, userInfo.Role);

generalWindow.Show();

this.Close();

}

else

{

MessageBox.Show("Неверный код доступа.");

}

}

}

private void btnSMScode\_Click(object sender, RoutedEventArgs e)

{

GenerateAccessCode();

}

// Генерация рандомного кода из 4 цифр

private async void GenerateAccessCode()

{

\_accessCode = new Random().Next(1000, 10000);

MessageBox.Show($"Код доступа: {\_accessCode}");

\_isCodeValid = true;

await Task.Delay(10000);

\_isCodeValid = false;

}

}

}

Этот код является частью приложения для управления студенческим общежитием. Код также содержит некоторые другие функции для очистки формы, отмены ввода данных и обработки нажатия кнопки "Снова СМС".

Он содержит реализацию функционала авторизации и генерации кода доступа.

Некоторые ключевые моменты этого кода:

1. ‘dormitoryManagerBDEntities’ — это объект контекста базы данных, используемой приложением.
2. ‘LoginTextBox\_KeyDown’ и ‘PasswordBox\_KeyDown’ - обработчики событий после ввода логина и ввода пароля соответственно. Они вызывают метод ‘VerifyUser’, который проверяет пароль и логин в базе данных, и если они верны, вызывает метод ‘GenerateAccessCode’ для генерации случайных 4 чисел кода доступа и запускает таймер на 10 секунд.
3. ‘AccessCodeTextBox\_KeyDown’ - обработчик события ввода кода доступа. Если введенный код совпадает с сгенерированным кодом доступа и таймер не истек, пользователь получает доступ к функционалу приложения.
4. ‘GenerateAccessCode’ - метод для генерации случайного 4-значного кода доступа, который сохраняется в переменной ‘\_accessCode’. Также он запускает таймер на 10 секунд, после которого ‘\_isCodeValid’ становится равным false.

После успешной авторизации, пользователю даётся выбор между двумя модулями: модулем просмотра информации и модуля создания отчётов из общего окна «Главное меню».

С# код Главного окна системы:

using System.Windows;

namespace AppDormitoryManager

{

public partial class WindowGeneral : Window

{

private int? role;

private string fullNameUser;

public WindowGeneral(string fullNameUser, int? role)

{

InitializeComponent();

UserInfoLabel.Content = $"{fullNameUser} ({(role == 1 ? "Администратор" : "Комендант")})"; // Выводится на экран информация об пользователе

this.fullNameUser = fullNameUser;

this.role = role;

}

// Открывает окно со списком жильцов

private void btnTenant\_Click(object sender, RoutedEventArgs e)

{

var residentsWindow = new WindowTenants(fullNameUser, role);

residentsWindow.Show();

}

// Открывает окно для создания квитанции

private void btnReceipt\_Click(object sender, RoutedEventArgs e)

{

WindowGenerReceipt residentsWindow = new WindowGenerReceipt(fullNameUser, role);

residentsWindow.Show();

}

// Кнопка выхода из программы

private void btnVCL\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

}

}

Этот код является частью программы для управления общежитием. Класс содержит члены данных `role` и `fullNameUser`, полученных после успешной авторизации, которые используются для отображения информации о пользователе в интерфейсе программы.

Конструктор класса принимает два аргумента: `fullNameUser` и `role`. Аргумент `fullNameUser` — это строка, содержащая полное имя пользователя, который вошел в систему, а `role` — это целочисленное значение, представляющее роль пользователя в системе (1 - администратор, 2 - комендант). В конструкторе вызывается метод `InitializeComponent()`, который инициализирует компоненты интерфейса.

Метод `btnTenant\_Click()` вызывает создание нового экземпляра класса `WindowTenants` (окно «Лист жильцов») и отображает его в качестве нового окна.

Метод `btnReceipt\_Click()` вызывает создание нового экземпляра класса `WindowGenerReceipt` (окно «Генератор квитанции») и отображает его в качестве нового окна.

Метод `btnVCL\_Click()` просто завершает приложение, вызывая метод `Shutdown()` из класса `Application`.

**Модуль просмотра данных**

В модуле просмотра (окно «Лист жильцов») данных пользователь Комендант может только смотреть на них, а пользователь Администратор может менять данные, удалять их и добавлять нового жильца.

C# код функционала Администратора в окне «Лист жильцов» (фрейм TenantsAdmin):

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace AppDormitoryManager

{

public partial class TenantsAdmin : Page

{

public TenantsAdmin()

{

InitializeComponent();

Loaded += TenantsComendant\_Loaded;

}

private void TenantsComendant\_Loaded(object sender, RoutedEventArgs e)

{

LoadData();

}

// Собирает информацию об жильцах из базы данных

private void LoadData()

{

using (var context = new dormitoryManagerBDEntities())

{

var tenants = context.Tenants.ToList();

TenantsList.ItemsSource = tenants;

}

}

// Кнопка возврата в Главное меню пользователя

private void btnBackMenu\_Click(object sender, RoutedEventArgs e)

{

Window.GetWindow(this).Close();

}

// Кнопка добавления жильца

private void btnSaveTenant\_Click(object sender, RoutedEventArgs e)

{

var addTenantWindow = new AddTenant();

addTenantWindow.ShowDialog();

LoadData();

}

// Кнопка изменения данных об жильце

private void btnChangeTenant\_Click(object sender, RoutedEventArgs e)

{

if (TenantsList.SelectedItem is Tenant selectedTenant)

{

var editTenantWindow = new AddTenant(selectedTenant);

editTenantWindow.ShowDialog();

LoadData();

}

}

// Кнопка удаления жильца из базы данных

private void btnDeleyeTenant\_Click(object sender, RoutedEventArgs e)

{

if (TenantsList.SelectedItem is Tenant selectedTenant)

{

if (MessageBox.Show($"Вы уверены, что хотите удалить жильца {selectedTenant.FullNameTenant}?", "Подтверждение удаления", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

using (var db = new dormitoryManagerBDEntities())

{

var existingTenant = db.Tenants.FirstOrDefault(t => t.ID\_Tenant == selectedTenant.ID\_Tenant);

if (existingTenant != null)

{

db.Tenants.Remove(existingTenant);

db.SaveChanges();

}

}

LoadData();

}

}

}

}

}

Данный код представляет собой класс `TenantsAdmin`, который является страницей (наследуется от класса `Page`) в приложении для управления общежитием. Он содержит методы для загрузки списка жильцов из базы данных, добавления нового жильца, изменения и удаления существующего жильца.

В методе `LoadData` выполняется запрос к базе данных с помощью контекста `dormitoryManagerBDEntities` (который был создан при помощи Entity Framework), чтобы получить список всех жильцов из таблицы `Tenants`. Затем результат запроса используется для заполнения элемента управления `ItemsControl`, связанного с элементом `TenantsList` на странице.

Методы `btnSaveTenant\_Click`, `btnChangeTenant\_Click` и `btnDeleyeTenant\_Click` вызывают соответствующие окна для добавления, изменения или удаления жильца. В случае успешного добавления или изменения жильца, метод `LoadData` вызывается для обновления списка жильцов на странице.

Метод `btnBackMenu\_Click` вызывается при нажатии на кнопку "Назад", чтобы вернуться на главное меню приложения.

В методах `btnChangeTenant\_Click` и `btnDeleyeTenant\_Click` происходит проверка, что выбран элемент из списка жильцов перед выполнением соответствующей операции. Если выбранный элемент не является объектом `Tenant` или не выбран вовсе, то операция не будет выполнена.

Если в окне «Лист жильцов» нажмём на кнопку «Добавить» или «Изменить», то откроется окно «Жилец» для изменения данных в базе данных.

C# код окна «Жилец» (AddTenant) для добавления и изменения данных об жильцах общежития администратором:

```charp

using System.Linq;

using System.Windows;

namespace AppDormitoryManager

{

public partial class AddTenant : Window

{

public Tenant Tenant { get; set; }

public AddTenant()

{

InitializeComponent();

}

public AddTenant(Tenant tenant) : this()

{

// Собираем данные с элемнтов этоого окна

Tenant = tenant;

FullNameTenantTextBox.Text = tenant.FullNameTenant;

RoomTextBox.Text = tenant.RoomTenant.ToString();

PhoneTextBox.Text = tenant.Phone.ToString();

EmailTextBox.Text = tenant.Email;

DateCheckinDatePicker.SelectedDate = tenant.DateChecin;

DateEvictionDatePicker.SelectedDate = tenant.DateEviction;

}

// Создаёт или сохраняет изменния о данных жильца в базе данных

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

// Проверка на правильность записи номера комнаты

int room;

if (!int.TryParse(RoomTextBox.Text, out room))

{

MessageBox.Show("Неверный номер комнаты");

return;

}

// Проверка на правильность записи номера телефона

int phone;

if (!int.TryParse(PhoneTextBox.Text, out phone))

{

MessageBox.Show("Неверный номер телефона");

return;

}

using (var db = new dormitoryManagerBDEntities())

{

// Если добавляем жильца в базу данных

if (Tenant == null)

{

var newTenant = new Tenant

{

FullNameTenant = FullNameTenantTextBox.Text,

RoomTenant = room,

Phone = phone,

Email = EmailTextBox.Text,

DateChecin = DateCheckinDatePicker.SelectedDate,

DateEviction = DateEvictionDatePicker.SelectedDate

};

var maxId = db.Tenants.Max(t => t.ID\_Tenant);

newTenant.ID\_Tenant = maxId + 1;

db.Tenants.Add(newTenant);

}

// Если изменяем данные жильца в базе данных

else

{

var existingTenant = db.Tenants.FirstOrDefault(t => t.ID\_Tenant == Tenant.ID\_Tenant);

if (existingTenant != null)

{

existingTenant.FullNameTenant = FullNameTenantTextBox.Text;

existingTenant.RoomTenant = room;

existingTenant.Phone = phone;

existingTenant.Email = EmailTextBox.Text;

existingTenant.DateChecin = DateCheckinDatePicker.SelectedDate;

existingTenant.DateEviction = DateEvictionDatePicker.SelectedDate;

}

}

db.SaveChanges();

}

this.Close();

}

// Кнопка закрытия этого окна

private void CancelButton\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

}

}

```

Этот код - класс ‘AddTenant’, который является окном для добавления нового жильца или изменения данных существующего жильца в базе данных.

Класс содержит свойство ‘Tenant’, которое представляет объект жильца, данные которого будут изменены, если окно было открыто для редактирования. Если окно было открыто для добавления нового жильца, свойство ‘Tenant’ будет равно null.

Конструктор ‘AddTenant’ без параметров используется для создания нового объекта жильца. Конструктор ‘AddTenant’ с параметром ‘Tenan't’ используется для изменения данных жильца. В этом случае метод инициализации конструктора устанавливает значения элементов управления этого окна, используя данные из объекта ‘Tenant’.

Класс также содержит два обработчика событий. Обработчик события для кнопки "Сохранить" сохраняет новый объект жильца или изменяет существующий объект в базе данных. В случае добавления нового жильца создается новый объект ‘Tenant’ и добавляется в базу данных. В случае изменения данных существующего жильца, существующий объект ‘Tenant’ в базе данных изменяется. После изменения или добавления объекта, окно закрывается.

Обработчик события для кнопки "Отмена" закрывает это окно, не сохраняя данные.

В случае, если в систему войдёт пользователь в роли «Комендант» и откроет окно «Лист жильцов», то этот пользователь сможет только смотреть на имеющиеся данные.

C# код функционала Коменданта в окне «Лист жильцов» (фрейм TenantsComendant):

```charp

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace AppDormitoryManager

{

public partial class TenantsComendant : Page

{

public TenantsComendant()

{

InitializeComponent();

Loaded += TenantsComendant\_Loaded;

}

// Выводит информацию об жильцах из базы данных

private void TenantsComendant\_Loaded(object sender, RoutedEventArgs e)

{

using (var context = new dormitoryManagerBDEntities())

{

var tenants = context.Tenants.ToList();

TenantsList.ItemsSource = tenants;

}

}

// Кнопка возврата в Главное меню пользователя

private void btnBackMenu\_Click(object sender, RoutedEventArgs e)

{

Window.GetWindow(this).Close();

}

}

}

```

**Модуль создания отчётов**

В модуле создания отчётов и пользователь Комендант и пользователь Администратор могут создать специальный отчёт - квитанцию для жильца. В генераторе квитанции надо было выбрать с выпадающего списка человека, после нажимаем на кнопку «Квитанция» и появляется окно сохранения документа. В выбранном место сохранения появится отчёт-квитанция в формате текстового документа в формате «.docx».

C# код окна Генератора квитанции:

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using Word = Microsoft.Office.Interop.Word;

using Microsoft.Win32;

using System.Data;

using System;

using System.Data.Entity.Infrastructure;

using System.Data.SqlClient;

namespace AppDormitoryManager

{

public partial class WindowGenerReceipt : Window

{

private int? role;

private string fullNameUser;

public WindowGenerReceipt(string fullNameUser, int? role)

{

InitializeComponent();

// фильтр для выпадающего списка - не выводим ушедших жильцов

using (var db = new dormitoryManagerBDEntities())

{

DateTime currentDate = DateTime.Now;

TenantsComboBox.ItemsSource = db.Tenants

.Where(t => !string.IsNullOrEmpty(t.FullNameTenant) && t.DateEviction == null)

.ToList();

TenantsComboBox.DisplayMemberPath = "FullNameTenant";

TenantsComboBox.SelectedValuePath = "ID\_Tenant";

}

}

// Кнопка для генерации Word квитанций

private void btnReceipt\_Click(object sender, RoutedEventArgs e)

{

var selectedResidentId = (int)TenantsComboBox.SelectedValue;

var wordApp = new Word.Application();

wordApp.Visible = false;

var wordDoc = wordApp.Documents.Add();

using (var db = new dormitoryManagerBDEntities())

{

// Фиксируем выбранного жильца для дальнйшей работы

var resident = db.Tenants.FirstOrDefault(r => r.ID\_Tenant == selectedResidentId);

if (resident == null)

{

throw new Exception("Такого жильца нету в базе данных.");

}

// счётчик даты для более точного определения дня выплаты квитанции

DateTime DataPay;

int year = DateTime.Now.Year;

int month = DateTime.Now.Month;

int day = resident.DateChecin.Value.Day;

if (day > DateTime.Now.Day)

{

DataPay = new DateTime(year, month, day);

}

else

{

if (month == 12)

{

year += 1;

month = 1;

}

else

{

month += 1;

}

DataPay = new DateTime(year, month, day);

}

var receipt = new Receipt()

{

TenantReceipt = resident.ID\_Tenant, // выбранный жилец из выпадающего списка

PayLiving = new Random().Next(1000, 5000), // имитация необходимой опалты за проживание

PayAddService = new Random().Next(1000, 3000), // имитация необходимой опалты за доп. услуги

DataPay = DataPay // дата выплаты квитанции

};

var paragraph = wordDoc.Paragraphs.Add();

paragraph.Range.Text = "ФИО жильца: " + resident.FullNameTenant + "\n";

paragraph.Range.Text += "Номер комнаты: " + resident.RoomTenant + "\n";

paragraph.Range.Text += "Дата оплаты квитанции: " + receipt.DataPay.Value.ToShortDateString() + "\n";

paragraph.Range.Text += "Сумма оплаты за проживание: " + receipt.PayLiving.ToString() + "\n";

paragraph.Range.Text += "Сумма оплаты доп. услуг: " + receipt.PayAddService.ToString() + "\n";

paragraph.Range.Text += "Итоговая сумма оплаты: " + (receipt.PayLiving + receipt.PayAddService).ToString() + "\n";

}

// Сохранение документа Word

var saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Word documents|\*.docx";

if (saveFileDialog.ShowDialog() == true)

{

wordDoc.SaveAs2(saveFileDialog.FileName);

}

wordDoc.Close();

wordApp.Quit();

}

private void btnBackMenu\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

//Выводит текст выбранного жильца из выпадающего списка

private void TenantsComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var selectedTenant = TenantsComboBox.SelectedItem as Tenant;

if (selectedTenant != null)

{

TenantInfoTextBox.Text = $"Выбран жилец: {selectedTenant.FullNameTenant}";

}

}

}

}

Данный код - Класс WindowGenerReceipt (окна «Генератор квитанции»). Это окно для генерации квитанций на оплату проживания и дополнительных услуг для выбранного жильца.

В конструкторе окна происходит инициализация выпадающего списка жильцов, для которых еще не указана дата выселения.

При нажатии на кнопку "Сгенерировать квитанцию" происходит выборка выбранного жильца из выпадающего списка, формирование объекта ‘Receipt’ с случайными значениями оплаты за проживание и дополнительные услуги, а также датой оплаты, которая определяется на основе текущей даты и даты заселения жильца.

Затем происходит заполнение документа Word с информацией о жильце и сформированной квитанцией. Пользователю предлагается сохранить документ в формате docx.

Также реализована возможность вывода информации о выбранном жильце в текстовом поле при выборе из выпадающего списка.

В данном коде также присутствуют обработка ошибок при отсутствии выбранного жильца в базе данных.

# ЗАКЛЮЧЕНИЕ

Таким образом, курсовой проект по разработке информационной системы «Учёт расчетов за проживание в общежитии» является важным инструментом для упрощения и автоматизации обслуживания общежитий. Автоматизация процесса учёта и контроля задолженностей позволит ускорить работу бухгалтеров и администраторов общежитий.

В процессе выполнения проекта мы разработали базу данных и пользовательский интерфейс, позволяющий быстро удобно вводить просматривать данные о жителях общежития и их расчетах. Мы также реализовали функциональность для авторизации пользователей и управления доступом.

Кроме того, мы заботились о том, чтобы интерфейс был интуитивно понятным и простым в использовании, а база данных была надежной и безопасной для хранения информации.

Работа над проектом позволила улучшить навыки разработки информационных систем и расширить понимание принципов хранения и обработки данных. Кроме того, мы понимаем важность использования информационных технологий в повседневной жизни и в работе с пользовательскими данными.

Посмотреть на информационную систему можно по это ссылке:

https://github.com/KamillaYesa/kursovicexam2023-AppDormitoryManager

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

# Книги

1. Акулов, А. Язык программирования C#. Основы синтаксиса и разработки Windows-приложений. Издательство: Питер, 2021 год. - 368 с.
2. Ли, Джонсон. Windows Presentation Foundation. Полное руководство. Издательство: Вильямс, 2020 год. - 672 с.
3. Петкович, Д. Microsoft SQL Server 2019: Руководство для начинающих, 7-е издание / Д. Петкович. - М.: Издательский дом "Питер", 2020. - 512 с.
4. Петкович, Д. Microsoft SQL Server 2019: Руководство для начинающих / Д. Петкович. - М.: Издательский дом "Питер", 2019. - 464 с.
5. Петкович, Д. Microsoft SQL Server 2019: Руководство для начинающих, 7-е издание. Издательский дом "Питер", 2020 год. - 944 с.
6. Уорд, Б. SQL Server 2019 Revealed: Включая Big Data Clusters и Машинное Обучение / Б. Уорд. - М.: Издательский дом "Питер", 2019. - 320 с.

# Интернет-ресурсы

1. "Введение в .NET: что такое .NET и обзор основ" – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/core/introduction
2. "Начало работы с .NET: что такое среда .NET и как она работает?" – Режим доступа: https://habr.com/ru/sandbox/13849/
3. "Язык программирования С#" – Режим доступа: https://flexberry.github.io/ru/gbt\_csharp.html
4. "Язык С#: где его используют, что пишут, как появился и чем хорош" – Режим доступа: https://skillbox.ru/media/code/yazyk-s-gde-ispolzuyut-chto-pishut-kak-poyavilsya-i-chem-khorosh/
5. Info-Comp (Особенности SQL Server Management Studio). – Режим доступа: <https://info-comp.ru/ssms-feature-overview>
6. Metanit (Entity Framework). – Режим доступа: <https://metanit.com/sharp/entityframework/1.1.php>
7. Metanit (Основы работы с Microsoft SQL Server). – Режим доступа: <https://metanit.com/sql/sqlserver/1.1.php>
8. Metanit (Основы работы с SQL Server Management Studio). – Режим доступа: <https://metanit.com/sql/sqlserver/1.3.php>
9. Microsoft Docs (Entity Framework). – Режим доступа: https://learn.microsoft.com/en-us/ef/
10. Microsoft Docs (SQL Server Management Studio (SSMS)). – Режим доступа: <https://learn.microsoft.com/ru-ru/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15>
11. Microsoft Docs (WPF overview). – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0>
12. Professor Web (WPF). – Режим доступа: https://professorweb.ru/my/WPF/base\_WPF/level1/1\_3.php
13. Timeweb (Основы работы с MS SQL Server). – Режим доступа: <https://timeweb.com/ru/community/articles/osnovy-raboty-s-mssqlserver>
14. Wikipedia (Entity Framework). – Режим доступа: https://en.wikipedia.org/wiki/Entity\_Framework
15. Wikipedia (Microsoft SQL Server). – Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_SQL_Server>
16. Wikipedia (SQL Server Management Studio). – Режим доступа: <https://ru.wikipedia.org/wiki/SQL_Server_Management_Studio>
17. Wikipedia (SQL). – Режим доступа: <https://ru.wikipedia.org/wiki/SQL>
18. Wikipedia (Windows Presentation Foundation). – Режим доступа: https://en.wikipedia.org/wiki/Windows\_Presentation\_Foundation
19. Wikipedia (Особенности технологии Windows Presentation Foundation). – Режим доступа: <https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation#Особенности_технологии>
20. Википедия (.NET Framework) – Режим доступа: https://ru.wikipedia.org/wiki/.NET\_Framework
21. Википедия (.NET) – Режим доступа: https://ru.wikipedia.org/wiki/.NET
22. Википедия (C Sharp) – Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp
23. Документация Microsoft. Краткий обзор языка C# – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/