СОДЕРЖАНИЕ

Введение 4

1 Технический проект 8

1.1 Анализ предметной области 8

1.2 Постановка задачи 9

1.3 Проектирование системы хранения данных 10

1.4 Проектирование функциональности программы 13

1.5 Проектирование структуры программы 14

1.6 Выбор средств реализации 17

1.7 Требования к программному обеспечению и техническим средствам 21

2 Рабочий проект 25

2.1 Физическая модель данных 25

2.2 Функциональное взаимодействие модулей программы 27

2.3 Входные и выходные данные 35

2.4 Установка и настройка программы 36

2.5 Работа с программой 38

Заключение 44

Список использованных источников 46

Приложение А – Use Case диаграмма 47

Приложение Б – Код основных классов 48

ВВЕДЕНИЕ

Для выполнения своих задач многим учебным заведениям необходимо осуществлять контроль за обучающимися и аппаратным обеспечением, находящимся в имуществе предприятия. Поэтому возникла необходимость автоматизировать некоторые процессы с использованием современных подходов и средств реализации информационных систем.

Любая информационная система включает в себя одну или несколько баз данных (БД), что значительно упрощает процессы хранения и обработки информации.

Одной из основных задач отделения является накопление и обработка сведений об аппаратном обеспечении и приложениях запущенных в данный момент на персональном компьютере (ПК). Так же немаловажной частью программы будет возможность управления ПК через локальную сеть. Именно данные процессы будет автоматизироваться в рамках данного дипломного проекта.

Темой дипломного проекта является разработка и создание электронных карточек обучающихся отделения информационных технологий МГГТК ФГБОУ ВО «АГУ».

Объектом исследования является деятельность отделения информационных технологий по контролю действия за обучающимися за персональными компьютерами.

Предметом исследования являются ПК, установленные на отделении, а также локальная сеть, которая позволит осуществлять контроль за всеми ПК удаленно.

Целью данного дипломного проекта является разработка расширяемой информационной системы, которая позволит производить контроль и собирать информацию об ПК, установленных в кабинете организации.

Написание данного дипломного проекта является целесообразным, потому что информационная система:

* позволит производить контроль за ПК, подключенными в локальной сети предприятия;
* облегчит работу учителей за счет возможности просмотра рабочего стола учащегося;
* позволит собирать актуальную информацию об аппаратном обеспечении, находящимся внутри ПК;
* снизит объем бумажной документации.

Разработка нового приложения позволит учесть индивидуальные предпочтения пользователя в интерфейсе и предоставляемых функциях, не перегружая информационную систему лишним, реализованным «впрок», инструментарием. Это снизит системные требования к развертыванию данной информационной системы.

Для достижения поставленной цели будут рассмотрены и решены следующие задачи:

* произведен анализ предметной области и выделены необходимые данные;
* разработаны концептуальная и логическая модели БД;
* разработаны структура и функциональность приложения для управления созданной БД;
* произведён выбор средств реализации;
* описаны минимальные требования к программно-техническим средствам;
* произведена физическая реализация БД;
* описаны входные и выходные данные;
* разработан программный код приложения;
* разработана пользовательская система помощи;
* разработана инсталляция.

Практическая значимость проекта заключается в возможности использования приложения в образовательных учреждения с целью автоматизации процесса контроля за учащимися в процессе работы за ПК.

Основными пользователями данной программы будут преподаватели, работающие в кабинете, где находится большое количество ПК.

Данные для приложения будут взяты из документации иностранных аналогов и деятельности отделения.

Теоретическими основами и методами решения поставленных задач будут методологии проектирования функциональности приложения, структуры БД, особенности реализации программного кода с помощью конкретного языка программирования, принципы и алгоритмы применения инструментальных средств на различных этапах разработки программного продукта.

В техническом проекте будет произведен анализ предметной области и представлены этапы проектирования архитектуры, интерфейса и функциональности реализуемой информационной системы, включая разработку логической модели БД. Также в нем будет произведен выбор инструментов для физической реализации построенных моделей данных и информационной системы.

Наиболее удобной для реализации поставленной задачи является свободная реляционная система управления базами данных. В соответствии с этим будут подобраны система управления БД (СУБД) и инструментальные средства реализации интерфейса приложения, которое будет работать с данной БД.

Для проектирования интерфейса и функциональности приложения, а также архитектуры информационной системы будут использоваться UML-диаграммы.

В рабочем проекте представлены достигнутые практические результаты, в частности, описаны физическая модель данных и разработанные классы, реализующие функциональность приложения. Также в нем описаны процессы установки информационной системы и особенности ее эксплуатации.

1 ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1 Анализ предметной области

Отделение является основным учебно-научным структурным подразделением средних учебных заведений, которое проводит учебную, методическую и научно-исследовательскую деятельность по одной или нескольким родственным специальностям.

На отделение возлагаются следующие задачи и обязанности:

* реализация федеральных государственных образовательных стандартов (ФГОС СПО), целевая ориентация учебного процесса на формирование общих и профессиональных компетенций, установленных ФГОС СПО;
* обеспечение качественного образования, повышение ответственности за результаты образовательной деятельности;
* внедрение в образовательный процесс современных подходов к обучению и воспитанию обучающихся;
* усиление работы по сохранению контингента и адаптации обучающихся.

Реализация данных задач предполагает планирование и управление процессом проведения лекций, лабораторных, практических, семинарских и других видов учебных занятий, руководства учебной и производственной практикой, исследовательскими работами, курсовыми и дипломными проектами (работами), проведения экзаменов и зачётов на промежуточной аттестации, а также итоговой аттестации.

Работа учащихся за учебными местами требует жесткого контроля. Не менее важным является фиксация, хранение и обработка данных об аппаратном и программном обеспечении.

Сопровождение данной деятельности требует оперирование большими объемами информации. Поэтому существует практическая потребность в автоматизации данных функций для облегчения процесса накопления, хранения, контроля и обработки информации.

Разрабатываемое приложение будет автоматизировать и упрощать следующие процессы:

* хранение данных об аппаратном обеспечении каждого ПК в локальной сети;
* возможность редактирования информации об сотруднике отделения;
* добавление новых сотрудников в систему;
* обеспечение контроля во время учебной деятельности;

Разрабатываемое приложение будет оперировать данными об аппаратном и программном обеспечении всех ПК в локальной сети, а также данными о сотрудниках отделения.

1.2 Постановка задачи

Необходимо разработать клиент-серверную информационную систему для контроля, поддержки и мониторинга за учащимися отделения информационных технологий МГГТК ФГБОУ ВО «АГУ», а также создать документационное сопровождение приложения, включающее в себе технический и рабочие проекты.

Необходимые данные должны храниться в удалённой БД.

Данная программа должна будет содержать все нужные сведения о преподавателях и ПК в локальной сети, иметь удобный и понятный пользователю графический интерфейс, обеспечивать необходимый уровень защиты данных от несанкционированного использования.

Программный продукт должен включать в себя две части:

* информационно-справочная часть, которая будет отображать необходимую информацию;
* операционная часть, в которой преподаватели и сотрудники отделения смогут изменять необходимую информацию и контролировать учебный процесс.

Приложение должно обеспечивать выполнение следующих функций:

* авторизация;
* отображение меню;
* просмотр, добавление и редактирование информации о сотрудниках,
* хранить и передавать информацию об аппаратном и программном обеспечении;
* удаленная поддержка и работа с ПК, работающим в локальной сети;
* взаимодействие с данными, хранящимися в удалённой БД.

Данные будут предоставлены в виде графической и текстовой информации, загруженной из целевой БД и файлов.

Входными данными в приложении будут данные сотрудников и сведения об аппаратном и программном обеспечении персональных компьютеров (ПК).

Выходными данными в приложении будут информация об действиях на ПК в локальной сети.

Приложение будет состоять из окон и страниц, на которых будет располагаться вся необходимая пользователю информация.

Способ решения поставленных задач зависит от выбора средств разработки.

Тестирование программы будет проводиться путем запуска ряда автоматизированных Unit-тестов.

1.3 Проектирование системы хранения данных

Первым шагом на пути реализации проекта будет создание БД.

В широком смысле БД – это хранилище элементов данных, называемых «записями», имеющее определенную физическую и логическую структуру, а также программный интерфейс, позволяющий пользователю взаимодействовать с сохраняемой в ней информацией. Чтобы универсальным способом извлекать из нее группы записей, обрабатывать их, изменять и удалять, требуются специальные программы, которые называются СУБД [2].

Наиболее удобной для реализации поставленной задачи является свободная реляционная система управления базами данных, которая использует декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

В теории баз данных эти таблицы называют отношениями (relations) –

поэтому и базы данных называются реляционными. Отношение – это

математический термин. При определении свойств таких отношений

используется теория множеств. В терминах данной теории строки таблицы

будут называться кортежами (tuples), а колонки – атрибутами. Отношение

имеет заголовок, который состоит из атрибутов, и тело, состоящее из

кортежей. Количество атрибутов называется степенью отношения, а

количество кортежей – кардинальным числом.

Реляционная модель данных – данные представлены посредством строк в таблицах.

В данном дипломном проекте предметной областью является хранение, контроль и поддержка ПК в локальной сети. БД будет предназначена для хранения данных о сотрудниках отделения, а также информацию о действиях, происходящих на каждом ПК.

Первый этап процесса проектирования БД заключается в создании концептуальной модели данных.

Компонентами концептуальной модели являются сущности и взаимосвязи. Она служит средством общения между различными пользователями, и поэтому разрабатывается без учета особенностей физического представления данных. При проектировании концептуальной модели все усилия разработчика должны быть направлены в основном на структуризацию данных и выявление взаимосвязей между ними без рассмотрения особенностей реализации и вопросов эффективности обработки. Проектирование концептуальной модели базируется на основе анализа решаемых в предметной области задач по обработке данных [2].

На основе анализа предметной области и пункта 1.2 были выделены основные сущности:

* пользователи (идентификатор, логин, пароль, имя, фамилия, отчество, идентификатор роли, идентификатор кабинета);
* кабинеты (идентификатор, название кабинета);
* роли (идентификатор, наименование);
* специальности сотрудников (идентификатор, название специальности);
* названия действий (идентификатор, название)
* статусы действий (идентификатор, название статуса)
* действия в кабинетах (идентификатор, время действия, идентификатор кабинета, идентификатор статуса действия, идентификатор названия действия)

Затем была разработана концептуальная модель БД (рисунок 1) и логическая модель, которая отражает связи между сущностями (рисунок 2).

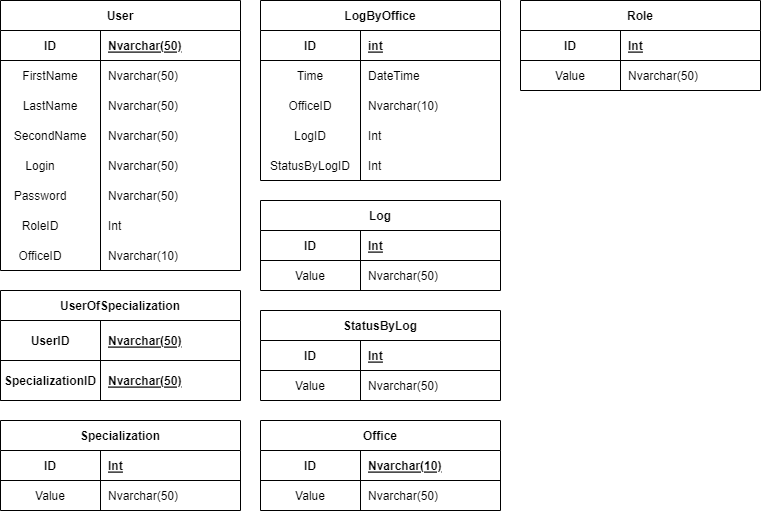


Рисунок 1 – Концептуальная модель БД

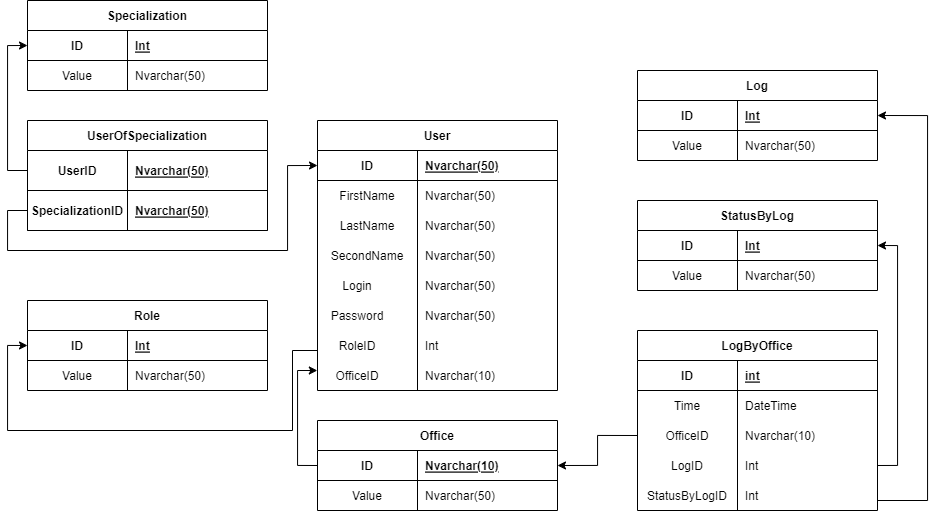


Рисунок 2 – Логическая модель БД

Так как в данном проекте потребуется большая работа с данными об аппаратуре и характеристиках ПК, необходимо разработать файловую модель для хранения этих данных.

Для хранения логов, информации о комплектующих и характеристиках ПК, разрешенных для использования программ и статистики работы программы была разработана файловая структура хранилища данных (рис. 3).

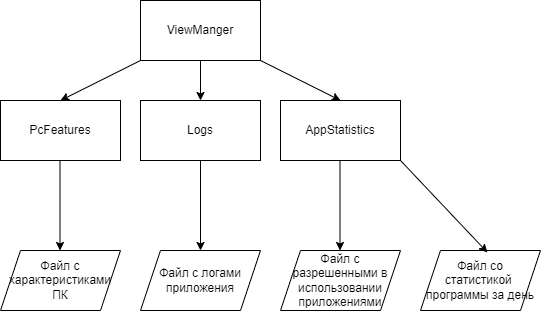


Рисунок 3 – Логическая файловая модель для хранения данных

1.4 Проектирование функциональности приложения

Для описания основных функций проектируемой информационной системы воспользуемся методологией UML и построим диаграмму Use Case (приложение А). Она определяет:

* основных пользователей системы, демонстрируя их иерархию;
* основные функции системы и их взаимосвязи;
* группирование функций системы по доступности определенным пользователям;
* зависимость функционирования проектируемой системы от внешних приложений.

На ней видно разграничение уровней доступа к реализованным функциям и основные сценарии использования представленной информационной системы, предусмотренные при её разработке.

Обучающемуся, который является одним из пользователей системы доступны функции просмотра характеристик ПК, закрытие программы, а также настройка приложения.

Секретарю, кроме функции настройки приложения, доступны функции редактирования информации о сотрудниках, а также возможность создания нового сотрудника.

Преподавателю, опять же, доступна функция настройка приложения, а также функции с выбранным ПК: запросить информацию, выключить. Также учитель может сформировать отчеты по всем ПК за день.

При выполнении всех функций система взаимодействует с удаленной БД, расположенной на сервере. Взаимодействие происходит через стандартизированный интерфейс, предоставляемый СУБД.

1.5 Проектирование структуры приложения

При создании информационной системы одним из самых важных аспектов является её архитектура. Она представляет собой концептуальное видение структуры будущих функциональных процессов и технологий на системном уровне и во взаимосвязи.

Так как разработать нужно клиент-серверную информационную систему, то необходимо определиться с её архитектурой. Основными клиент-серверными архитектурами, реализуемыми в настоящее время являются:

* выделенный сервер;
* активный сервер;
* сервер приложений [5].

В архитектуре «выделенный сервер» средства управления БД и БД размещены на машине-сервере, а функциональная обработка данных выполняется на стороне клиента. Сервер только предоставляет данные по запросу клиента. В данной архитектуре предъявляются повышенные требования к аппаратному и программному обеспечению клиента, так как вся бизнес-логика реализуется на стороне клиента, а сервер выполняет только функции доступа к данным и их защиты.

В архитектуре «активный сервер» функции бизнес-логики разделяются между клиентской и серверной частями. Общие или критически значимые функции оформляются в виде хранимых процедур, включаемых в состав БД. Кроме этого, вводится механизм отслеживания событий БД – триггеров, также включаемых в состав БД. При возникновении соответствующего события (обычно изменения данных) СУБД вызывает для выполнения хранимую процедуру, связанную с триггером, что позволяет эффективно контролировать изменение БД. Хранимые процедуры и триггеры могут быть использованы любыми клиентскими приложениями, работающими с БД. Это снижает дублирование программных кодов и исключает необходимость компиляции каждого запроса.

В данной архитектуре снижаются требования к аппаратному и программному обеспечению клиента, но возрастают, применительно к серверу.

В архитектуре «сервер приложений» снижение уровня требований к ресурсам клиента достигается за счет введения промежуточного звена – сервера приложений, на который переноситься значительная часть программных компонентов управления данными и большая часть бизнес-логики. При этом серверы БД обеспечивают исключительно функции СУБД по ведению и обслуживанию БД [4].

Так как клиентская часть информационной системы будет устанавливаться на компьютерах, не обладающих повышенными характеристиками производительности процессора, оперативной и постоянной памяти, то выбор был остановлен на архитектуре «активный сервер».

Сервер БД будет осуществлять целый комплекс действий по управлению данными. Основными его обязанностями будут:

* поддержка ссылочной целостности данных согласно определенным в БД правилам;
* хранение и резервное копирование данных;
* выполнение пользовательских запросов на выбор и модификацию данных и метаданных, получаемых от клиентских приложений, функционирующих на персональных компьютерах локальной сети;
* обеспечение авторизованного доступа к данным на основе проверки прав и привилегий пользователей;

В качестве рабочих мест пользователя будут использоваться персональные компьютеры секретаря и преподавателей, расположенные в локальной сети отделения, что позволяет не отказываться от привычной рабочей среды.

С целью удовлетворения предъявляемых к клиентскому приложению требований, приведенных в разделе 1.2, и представленных на Use Case диаграмме функций, разрабатываемое приложение будет состоять из следующих страниц:

* авторизации – будет запускаться перед отображением главного окна для идентификации пользователя;
* главная – будет содержать навигационные кнопки для перехода на страницы, настроек, редактирования, добавления, страницы с графиком работ за день, страница, с характеристиками выбранного ПК, страница со списком ПК;
* список компьютеров – страница где выведены все зарегистрированные ПК в системе;
* добавления и редактирования – страница для добавления и редактирования сотрудников в системе;
* график – окно для графика действий проведенных за день (учебный план, группы, обучающиеся, приказы, новости).

Необходимые данные будут хранится в БД.

Проектируемая структура информационной системы представлена на рисунке 4.



Рисунок 4 – Схема проектируемой структуры информационной системы

Проектируемая структура клиентского приложения представлена на рисунке 5.



Рисунок 5 – Схема проектируемой структуры клиентского приложения

Тестирование программы планируется проводить путём выбора каждого пункта меню и проверки корректности получаемого отклика программы.

1.6 Выбор средств реализации

Интерфейс и функциональность клиентского приложения могут быть написаны на следующих языках программирования: С++, С#, Python, Java. Все они позволяют работать с БД, файлами, текстом и графикой, с использованием объектно-ориентированного подхода.

Выбор был остановлен на языке программирования C#, с использованием .NET 6, и интегрированной среде разработки Microsoft Visual Studio 2019.

Visual Studio обеспечивает поддержку новейших языковых функций на протяжении всего процесса разработки. Данная среда разработки поддерживает C#, Visual Basic, C++, TypeScript, F# и даже сторонние языки, например, JavaScript [3].

Язык С# актуален потому, что:

* позволяет более рационально создавать популярные на сегодня типы приложений;
* интегрировал в себе преимущества языков Java и С++, при этом в объединённом языке исключены некоторые спорные директивы, макросы, отменены глобальные переменные;
* является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
* является мощным объектным языком с возможностями наследования и универсализации;
* мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из БД и других хранилищ данных [4].

.NET 6 – следующий шаг в .NET Core. Проект призван улучшить .NET в нескольких ключевых аспектах:

* улучшить производительность работы программы за счет: прокачаной предварительной компиляции (через утилиту Crossgen2), оптимизации на основе профилирования (PGO) и горячей перезагрузке приложений во время отладки;
* profile-Guided Optimization (PGO) - техника оптимизации программы компилятором, нацеленная на увеличение производительности выполнения программы;
* Горячая перезагрузка приложений. Такая возможность была и раньше, но в очень сильно упрощённом варианте и только в мощной IDE, вроде Visual Studio. Теперь же её прокачали настолько, что она реально позволит сэкономить уйму времени, избавившись от постоянных действий остановка-правка-ребилд-деплой-запуск-достижение точки отладки, причём, в любой IDE, даже в VS Code.

Если вы используете.NET Core или.NET 5, вам нужно будет взглянуть на обновление, и путь миграции будет довольно минимальным. Visual Studio 2022 также вышел сегодня и, наконец, дает нам 64-разрядную версию Visual Studio, а также собственный набор преимуществ производительности для повышения вашей продуктивности. Если вы используете Raygun, вы обнаружите, что наш поставщик отчетов о сбоях уже настроен для.NET 6, и мы находимся в процессе выпуска обновленной поддержки.NET 6 для APM, наряду с поддержкой профилирования приложений.NET, развернутых в Linux.

Сохранилось всё, что было улучшено в .NET Core:

* open source и ориентированность на сообщество GitHub;
* кроссплатформенная реализация;
* поддержка использования специфических платформозависимых возможностей, таких как Windows Forms и WPF под Windows, а также нативных привязок (bindings) к каждой нативной платформе из Xamarin;
* высокая производительность;
* side-by-side инсталляция;
* маленький размер файлов проектов (SDK-стиль);
* интерфейс командной строки (CLI) с широкими возможностями;
* интеграция с Visual Studio, Visual Studio for Mac и Visual Studio Code.

Нововведения:

* возможность вызова кода Java из .NET 6 будет доступна на всех платформах;
* вызов кода Objective-C и Swift из .NET 6 будет поддерживаться в нескольких операционных системах;
* CoreFX будет расширен, чтобы поддерживать статическую компиляцию .NET (ahead-of-time – AOT), для уменьшения потребления ресурсов (footprints) и поддержки большего количества операционных систем [7].

Для проектирования физической модели БД будет использоваться СУБД Microsoft SQL Server и утилита SSMS. Данная СУБД является высокоэффективной и надежной, в ней есть возможности для использования ее в любых критичных бизнес-приложениях.

Основные преимущества Microsoft SQL Server:

* масштабируемость и производительность;
* база данных менее уязвима;
* инструменты бизнес-аналитики с поддержкой самообслуживания;
* Базовые выпуски SQL Server обладают расширенными функциями обеспечения безопасности;
* Стандартные и корпоративные выпуски Oracle обеспечивают лишь базовую безопасность;
* Центр обновления Windows позволяет снизить риски в области безопасности, а также свести к минимуму простой системы во время установки пакетов исправлений.

Для создания инсталляции была выбрана Inno Setup – система создания инсталляторов для Windows-программ с открытым исходным кодом. Она представляет собой среду для настройки проекта, создания сценария и конечного релиза дистрибутива.

Ключевыми особенностями Inno Setup являются:

* поддержка всех современных версий Windows: 10, 8, 7, 2008 R2, Vista. Поддерживает установку 64-битных программ на 64-битных выпусках. Также поддерживает 64-разрядные процессоры;
* поддержка создания одиночных EXE для облегчения установки и распространения программ через сеть Интернет;
* стандартный интерфейс мастера установки в стиле Windows 2000/XP;
* возможность выбора типа установки, например: полный, минимальный, выборочный;
* имеет встроенную поддержку DEFLATE, bzip2 и 7-Zip LZMA/LZMA2 сжатий;
* установщик умеет сравнивать версии файлов, заменять встроенные файлы, устанавливать разделяемые файлы, регистрировать DLL/OCX библиотеки и устанавливать шрифты;
* позволяет создавать ярлыки в меню «Пуск» и на «Рабочем столе»;
* позволяет создавать записи в реестре и .ini-файлы;
* поддерживает создание многоязычных инсталляторов программ;
* поддерживает Unicode и языки с направлением письма справа налево;
* поддерживает установку пароля и шифрование инсталляторов программ [10].

1.7 Требования к программному обеспечению и техническим средствам

Программное обеспечение должно удовлетворять всем реализуемым функциям системы, а также иметь определенный набор средств для организации всех требуемых процессов обработки данных, которые позволяют своевременно выполнять все функции во всех режимах функционирования системы.

Системное программное обеспечение должно реализовывать комплекс задач управления. Для корректной работы разрабатываемой программы необходимо, чтобы клиентский компьютер, с целью удовлетворения минимальным требованиям, был оснащен:

* операционной системой Windows 10 / 11;
* лицензионным программным обеспечением Microsoft Office 2019 +;
* .NET 6 +;

Сервер должен быть оснащён:

* операционной системой Windows Server 2019 / Ubuntu OS / CentOS Server / Oracle Linux Server;
* Microsoft SQL Server 2019;
* SQL Server Management Studio.

При выборе технических средств, применяемых для функционирования разрабатываемой программы, должны учитываться следующие требования:

* выбор технических средств должен обеспечивать рациональное соотношение между затратами на создание системы и достигаемым эффектом;
* технические параметры системы управления не должны налагать ограничения на регламент технологического процесса функционирования системы.

Для реализации вышеуказанных требований со стороны клиентского оборудования необходим следующий состав технических средств:

* центральный процессор фирмы INTEL или AMD с тактовой частотой не менее 1 ГГц;
* стабильное Ethernet-соединение.
* оперативная память не менее 2 ГБ;
* дисплей с разрешением не менее 1024x768 точек;
* 130 МБ свободного места на диске;
* клавиатура;
* манипулятор типа «мышь».

Серверное оборудование должно соответствовать следующим аппаратным требованиям:

* центральный процессор фирмы INTEL или AMD с тактовой частотой не менее 4 ГГц;
* стабильное Ethernet-соединение.
* оперативная память не менее 8 ГБ;
* 150 ГБ свободного места на диске.

Для разработки, отладки и расширения реализуемой информационной системы необходим следующий набор технических средств:

* операционная система Windows 10 / 11;
* .NET 6+, включая компоненты для разработки универсальных приложения (Universal Windows Platform);
* Visual Studio 2019 +;
* Microsoft SQL Server 2019;
* SQL Server Management Studio.

2 РАБОЧИЙ ПРОЕКТ

* 1. Физическая модель данных

Физическая модель данных описывает данные средствами конкретной СУБД. Ограничения, имеющиеся в логической модели данных, реализуются различными средствами СУБД, например, при помощи индексов, декларативных ограничений целостности, триггеров, хранимых процедур. При этом решения, принятые на уровне логического моделирования, определяют некоторые границы, в пределах которых можно развивать физическую модель данных. Точно также, в пределах этих границ можно принимать различные решения. Например, отношения, содержащиеся в логической модели данных, должны быть преобразованы в таблицы или коллекции, но для каждой таблицы можно дополнительно объявить различные индексы, повышающие скорость обращения к данным.

SQL имеет много преимуществ, что делает его популярным и востребованным. Это надежный и эффективный язык, используемый для взаимодействия с базой данных.

На основе логической модели, представленной в пункте 1.3, была создана физическая модель БД. Она описывает то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа. Также описываются типы, идентификаторы и разрядность полей (таблицы 1-8).

Таблица 1 – Физическая модель таблицы «Пользователь»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Обязательность |
| ID | Nvarchar(50) | Да |
| FirstName | Nvarchar(50) | Да |
| LastName | Nvarchar(50) | Да |
| SecondName | Nvarchar(50) | Нет |
| Login | Nvarchar(50) | Да |
| Password | Nvarchar(150) | Да |
| RoleID | Int | Да |
| OfficeID | Nvarchar(50) | Да |

Таблица 2 – Физическая модель таблицы «Роль»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Обязательность |
| ID | Int | Да |
| Value | Nvarchar(50) | Да |

Таблица 3 – Физическая модель таблицы «Кабинет»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Обязательность |
| ID | Nvarchar(10) | Да |
| Value | Nvarchar(100) | Да |

Таблица 4 – Физическая модель таблицы «Специальность»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Обязательность |
| ID | Int | Да |
| Value | Nvarchar(50) | Да |

Таблица 5 – Физическая модель таблицы «Специальность пользователя»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Обязательность |
| UserID | Nvarchar(50) | Да |
| SpecializationID | Int | Да |

Таблица 6 – Физическая модель таблицы «Логи кабинета»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Обязательность |
| ID | Int | Да |
| OfficeID | Nvarchar(10) | Да |
| LogID | Int | Да |
| StatusByLogID | Int | Да |
| Time | Datetime | Да |

Таблица 7 – Физическая модель таблицы «Логи»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Обязательность |
| ID | Int | Да |
| Value | Nvarchar(50) | Да |

Таблица 8 – Физическая модель таблицы «Статус логов»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Обязательность |
| ID | Int | Да |
| Value | Nvarchar(50) | Да |

2.2 Функциональное взаимодействие модулей программы

На основе разработанного технического проекта было написано приложение, обеспечивающее ведение электронных карточек обучающихся отделения информационных технологий МГГТК ФГБОУ ВО «АГУ». Для создания программы был использован язык программирования C# и интегрированная среда разработки Visual Studio 2022. В программе используются следующие языковые модули:

* System – содержит фундаментальные и базовые классы;
* System.Media – содержит классы для воспроизведения звуковых файлов и работы с системными звуками;
* System.Drawing – предоставляет доступ к основным графическим функциям GDI+ (Graphics Device Interface);
* Microsoft.Win32 – предоставляет два типа классов: те, которые обрабатывают события, вызванные операционной системой, и те, которые управляют системным реестром;
* System.Collections.Generic – содержит интерфейсы и классы, определяющие универсальные коллекции, которые позволяют пользователям создавать строго типизированные коллекции, обеспечивающие повышенную производительность и безопасность типов по сравнению с неуниверсальными строго типизированными коллекциями;
* System.IO – содержит типы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также типы для базовой поддержки файлов и папок;
* System.Net.Http – предоставляет интерфейс программирования для современных приложений HTTP.
* Newtonsoft.Json – библиотека чтобы работать с JSON объектами;
* System.Linq – предоставляет классы и интерфейсы, поддерживающие запросы с использованием LINQ;
* System.Diagnostics – предоставляет классы, позволяющие осуществлять взаимодействие с системными процессами, журналами событий и счетчиками производительности;
* System.Text – cодержит классы, которые представляют кодировки ASCII и Юникода; абстрактные базовые классы для преобразования блоков знаков в блоки байтов и обратно; вспомогательный класс, который обрабатывает и форматирует объекты String, не создавая промежуточные экземпляры String;
* System.Threading.Tasks – предоставляет типы, которые упрощают работу по написанию параллельного и асинхронного кода;
* System.Windows – предоставляет несколько важных классов базовых элементов Windows Presentation Foundation (WPF), различные классы, которые поддерживают систему свойств и логику событий WPF, а также другие типы, более широко применяемые в ядре и инфраструктуре WPF;
* System.Windows.Input – предоставляет типы для поддержки системы ввода Windows Presentation Foundation (WPF). Сюда входят классы абстрагирования устройств для устройств мыши, клавиатуры и пера, часто используемые классы диспетчера ввода, поддержка для команд и пользовательских команд, а также различные служебные классы [6].

Также были разработаны следующие элементы для взаимодействия пользователя с программой:

* MainWindow.xaml.cs (главное окно программы) – осуществляет переход между страницами приложения, а также обеспечивает вывод руководства по использованию программы;
* AuthPage.xaml.сs (страница авторизации) – осуществляет функции авторизации в приложении и переход на главную страницу;
* ComputerManagementPage.xaml.cs (страница для управления компьютерами) – обеспечивает интерфейс для получения и дистанционного управления компьютерами;
* StudentPage.xaml.cs (страница со списком студентов) – обеспечивает интерфейс для вывода списка студентов;
* CreateUserPage.xaml.cs (страница для создания нового пользователя) – обеспечивает интерфейс для создания и добавления пользователя в БД;
* MainPage.xaml.cs (главная страница приложения) – обеспечивает интерфейс, который разграничивает роли пользователя, а также предоставляет основной функционал;
* SettingsPage.xaml.cs (страница с настройками приложения) – обеспечивает настройку параметров приложения;
* StatisticsPage.xaml.cs (страница со статистикой работы приложения) – обеспечивает интерфейс, который выводит информацию и графики работы за день;
* UpdateUserListPage.xaml.cs (страница для обновления информации о пользователях) – обеспечивает интерфейс для обновления информации о пользователе;

Для реализации функциональности описанного пользовательского интерфейса были разработаны следующие модули, взаимодействие которых представлено на рисунке 6:

* Core (ядро системы) – модуль содержит классы, которые реализуют ключевую логику взаимодействия приложения и операционной системы (ОС) компьютера;
* Command (команды) – модуль содержит классы, реализующие логику различных «команд», которые пользователь отдаёт приложению на выполнение;
* Controllers (контроллеры) – модуль содержит классы, реализующие логику работы с API;
* Assets (ресурсы) – модуль содержит классы и файлы, которые являются общими ресурсами приложения. Классы модуля описывают логику доступа ко внутренним файлам;
* View (интерфейс) – модуль содержит классы и объекты пользовательского интерфейса, необходимые для его отрисовки и взаимодействия с пользователем;
* Model (модели) – модуль, который хранит различный модели представления объектов;
* ViewModel – модуль, являющейся связующим звеном между интерфейсом и логикой программы.

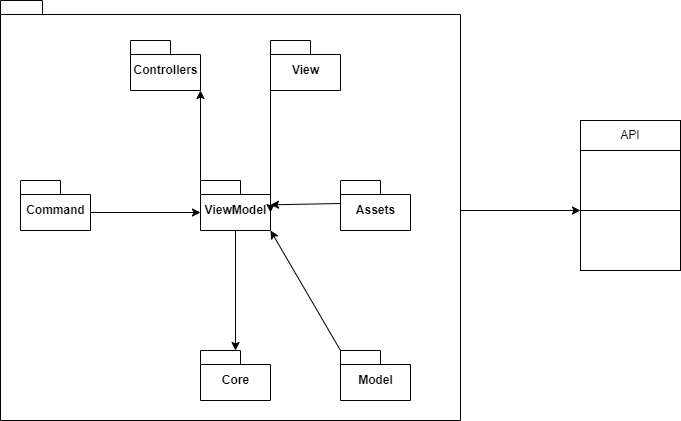


Рисунок 6 – Схема функционального взаимодействия модулей

Для реализации функциональности модуля Command были написаны следующие классы:

* DelegateCommand.cs – абстрактный класс, реализующий паттерн команды;

Методы класса DelegateCommand.cs:

* public bool CanExecute(object parameter) – вызывается при выполнении команды с предусловием;
* public void Execute(object parameter) – вызывается для немедленного выполнения команды.

Взаимодействие данных классов представлено на рисунке 7.

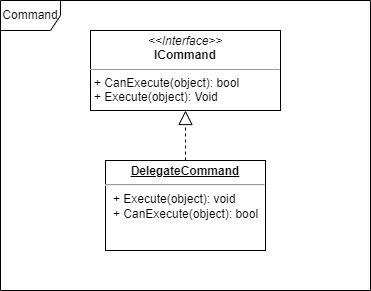


Рисунок 7 – Диаграмма классов модуля Command

Для реализации функциональности модуля Controllers были написаны следующие классы:

* AuthPageViewModelController.cs – класс, реализующий логику авторизации в приложении;
* TcpController.cs – класс, реализующий логику для передачи данных между сервером и клиентом;
* UniversalController<T>.cs – класс, реализующий универсальный контроллер для работы с API;

Код классов AuthPageViewModelController.cs, TcpController.cs и UniversalController<T>.cs представлен в приложении Б.

Метод класса AuthPageViewModelController.cs: public async Task<bool> AuthHelper(User user) – метод, который отправляет введенные данные для авторизации на API и ждет ответа для дальнейшей работы программы.

Методы класса TcpController.cs:

* public async Task StartTcp() – запускает TcpListener для подключения и работы с клиентами;
* public void GetDataTcp(object? obj) – получает данные отправленные от клиента.

Методы класса UniversalController<T>.cs:

* public async Task<IEnumerable<T>> GetList() – универсальный метод для получения списка нужной модели;
* public async Task<T> Get(string id) – универсальный метод для получения списка нужной модели по Id;
* public async Task<bool> Create(T obj) – универсальный метод для добавления новой модели в БД;
* public async Task<bool> Put(T obj) – универсальный метод для обновления выбранной модели в БД;
* public async Task<bool> Delete(string id) – универсальный метод для удаления выбранной модели в БД;

Взаимодействие данных классов представлено на рисунке 8.

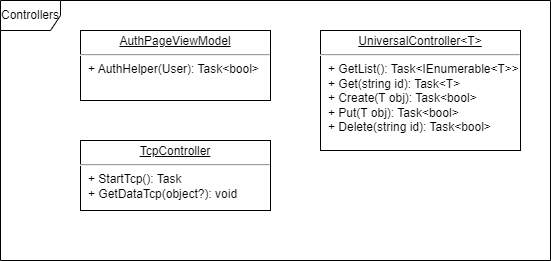


Рисунок 8 – Диаграмма классов модуля Controllers

Для реализации функциональности модуля Core были написаны следующие классы:

* FrameManager.cs – класс, реализующий логику переключений страниц в приложении;
* ISerringsManager.cs – интерфейс, описывающий работу с настройками приложения;
* SettingsManager.cs – класс, реализующий логику настройки приложения;
* StatisticsSort.cs – класс, реализующий логику сортировки статистики за день.

Метод класса FrameManager.cs: public static void SetSource<T>(T target, string nameFrame) where T : Page – вызывается при установке нового источника для отображения данных.

Метод класса StatisticsSort.cs: public IEnumerable<double> Sort(string allStat) – сортирует собранную статистику за день.

Методы класса SettingsManager.cs:

* public void SetTheme(string theme) – метод, который устанавливает тему в приложении;
* public void SetLanguage(string language) – метод, который устанавливает язык в приложении;

Схема класса представлена на рисунке 9.

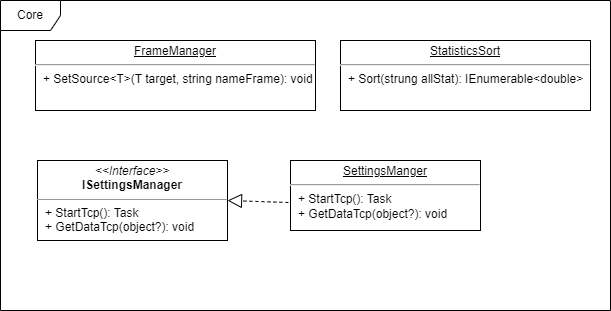


Рисунок 9 – Диаграмма классов модуля Core

Для реализации функциональности модуля ViewModel был написан класс BaseViewModel.cs, реализующий логику взаимодействия приложения с интерфейсом.

Методы класса BaseViewModel.cs:

* protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null) – вызывается при установке нового значения у элемента интерфейса для его обновления;

Схема класса представлена на рисунке 10.

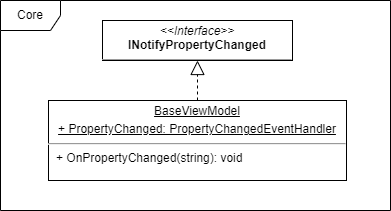


Рисунок 10 – Диаграмма классов модуля ViewModel

2.3 Входные и выходные данные

Входными данными в модуле View являются текстовые и графические данные из удалённой БД.

Выходными данными для модуля View является информация, вводимая в различные компоненты на форме. Изменения автоматически сохраняются в соответствующих документах в БД.

Входными данными в модуле AuthPage.xaml.cs являются данные из таблицы «Пользователь». При авторизации пользователя входные и выходные данные хранятся в сущности User в следующих полях:

* string Login – содержит введенный логин пользователя при регистрации;
* string Password – содержит введенный пароль пользователя.

Входными данными в модулях ComputerManagementPage.xaml.cs являются данные из удаленного компьютера.

Выходными данными для модулей ComputerManagementPage.xaml.cs является информация, сформированная в правильном виде информация о ПК.

Выходными данными для модуля CreateUserPage.xaml.cs является информация, вводимая в различные компоненты форм добавления данных.

Входными данными в модуле CreateUserPage.xaml.cs являются данные из таблицы «Пользователь».

Входными данными в модуле MainPage.xaml.cs являются данные из документа «Пользователь».

Выходными данными для модуля SettingsPage.xaml.cs является информация, вводимая в различные компоненты форм добавления данных.

Входными данными в модуле SettingsPage.xaml.cs являются выбранные настройки программы из ресурсов приложения.

Выходными данными для модуля StaticsPage.xaml.cs является статистика выводимая на экран пользователя в виде графика.

Входными данными в модуле StaticsPage.xaml.cs являются данные из файла «AllowedApplications».

Выходными данными для модуля UpdateUserListPage.xaml.cs является информация, вводимая в различные компоненты форм добавления данных.

Входными данными в модуле UpdateUserListPage.xaml.cs являются данные из таблицы «Пользователь».

2.4 Установка и настройка программы

Программа устанавливается на компьютер при запуске файла «Setup.exe». При установке программы происходит создание папки «ViewManager» в папке программ системы и осуществляется копирование файлов, необходимых для запуска и корректной работы программы:

* папки:
* Assets;
* runtimes;
* ru;
* libraries;
* CIS MGGTK.exe;
* uninstall.exe.

Для корректной работы приложения компьютер должен иметь стабильное интернет-соединение.

Программу можно удалить, выбрав пункт меню «Деинсталлировать» в меню Пуск/Программы или запустив файл uninstall.exe в папке программы.

Для нормальной работы программы необходимо:

* центральный процессор фирмы INTEL или AMD с тактовой частотой не менее 1 ГГц;
* стабильное Ethernet-соединение.
* оперативная память не менее 2 ГБ;
* дисплей с разрешением не менее 1024x768 точек;
* 130 МБ свободного места на диске;
* клавиатура;
* манипулятор типа «мышь».

Для создания инсталляции была использована программа Inno Setup. В ходе инсталляции пользователю показываются следующие окна:

* приветствие – в нем предлагается закрыть все запущенные приложения;
* краткая информация об устанавливаемом приложении – назначение, минимальные требования к программному обеспечению и техническим средствам;
* выбор папки установки;
* создание дополнительных ярлыков – позволяет создать ярлык на рабочем столе или отказаться от создания ярлыков;
* ход выполнения процесса установки;
* завершение установки – позволяет выбрать файлы, которые нужно запустить после установки.

2.5 Работа с программой

Для запуска программы нужно нажать на иконку с названием «ViewManager.exe», двойным кликом. В результате открывается главное окно со страницей для авторизации, в котором необходимо указать логин и пароль для входа в систему (рисунок 11).

После авторизации открывается главная страница c активной страницей настроек (рисунок 12). В зависимости от роли пользователя у него будет доступен разный функционал.

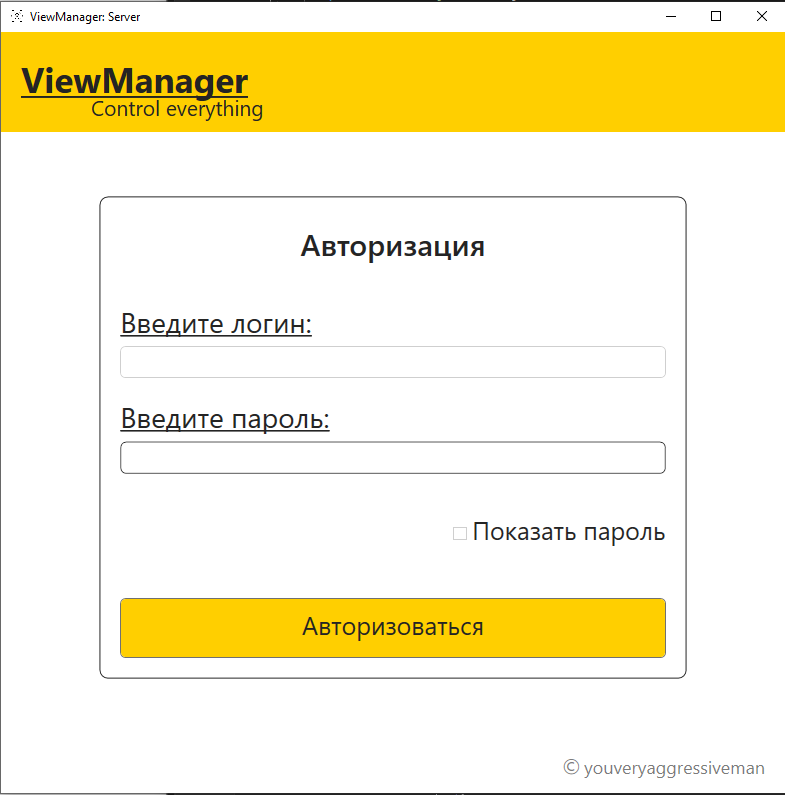


Рисунок 11 – Страница авторизации

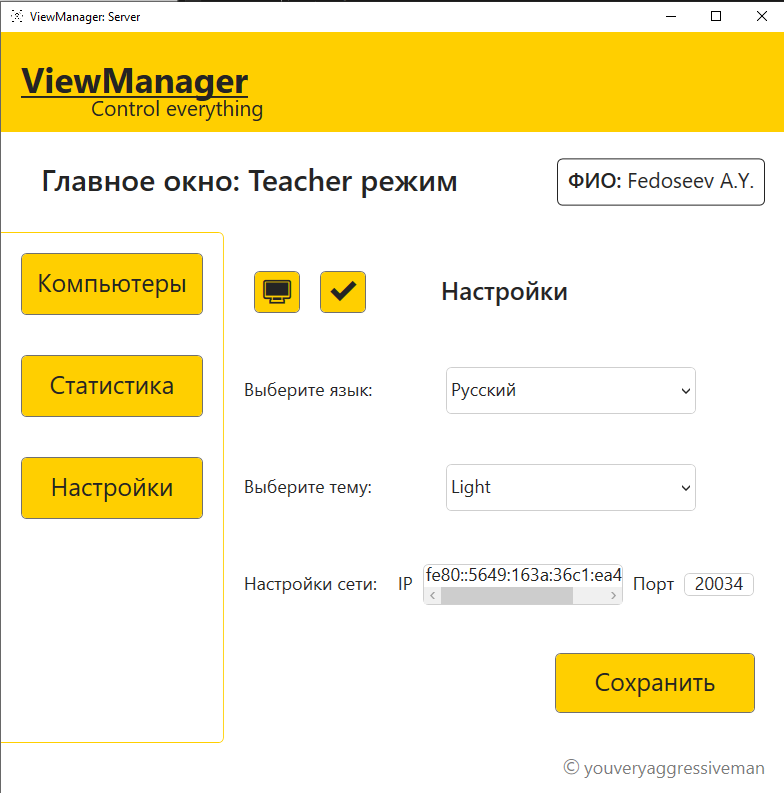


Рисунок 12 – Главная страница

На данной странице можно изменить тему и язык в приложении. А также поменять порт для подключения к серверу. Так же можно просмотреть информацию о ПК (рисунок 13) и задать разрешенные приложения на компьютерах (рисунок 14).

В данном примере открывается окно с ролью преподавателя. При нажатии на кнопку «Компьютеры» открывается страница для работы с подключенными пользователями (рисунок 15).

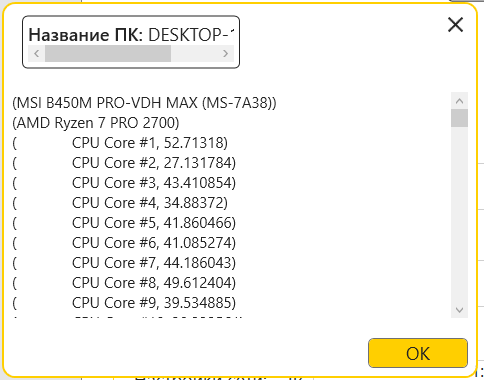


Рисунок 13 – Страница с информацией о ПК

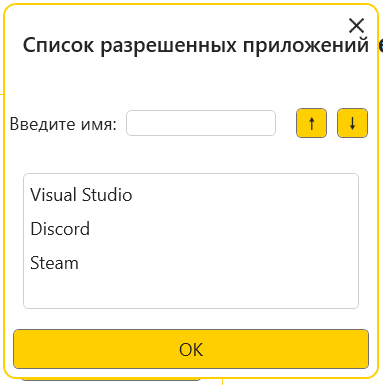


Рисунок 14 – Страница с разрешенными приложениями

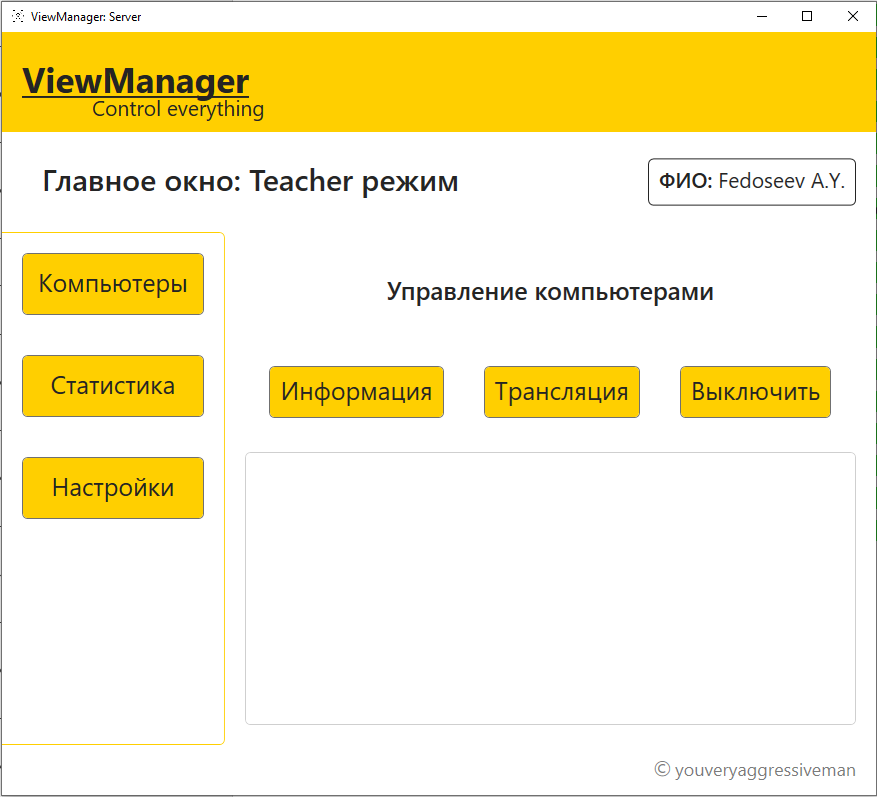


Рисунок 15 – Страница для удаленного управления ПК

На данном окне выводятся все подключенные ПК к серверу. Так же можно есть кнопки для управления данными компьютерами. Вы можете выключить ПК, получить информацию о нем, и запросить трансляцию.

При нажатии на кнопку «Статистика» открывается окно со статистикой за день (рисунок 16). Здесь в виде графиков вы можете увидеть и контролировать работу студентов.

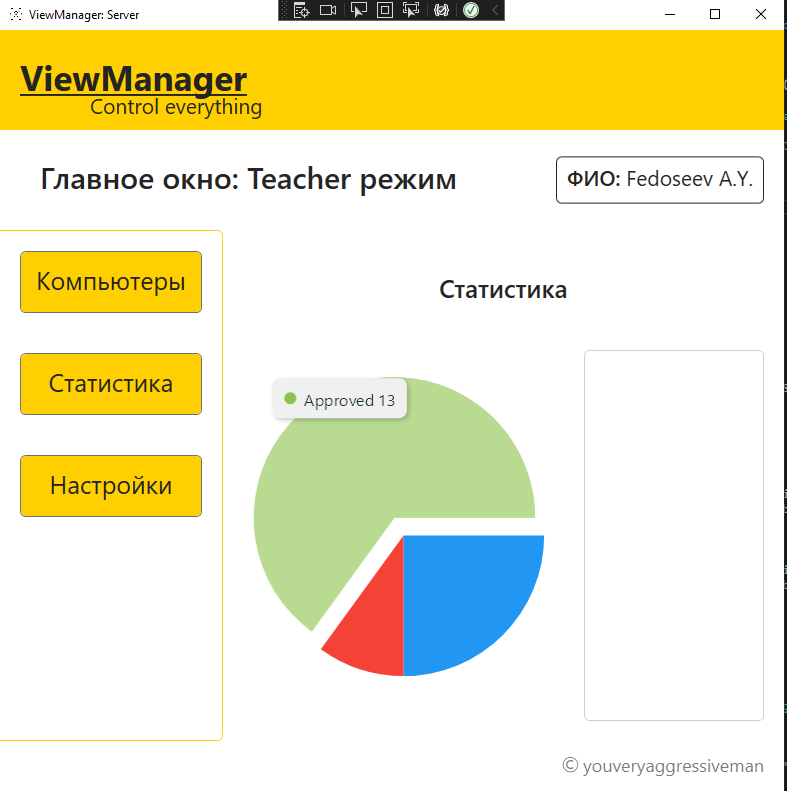


Рисунок 16 – Страница со статистикой

Если пользователь зайдет под ролью бухгалтера, ему будет доступен функционал: добавление нового преподавателя (рисунок 17) и обновление информации (рисунок 18).

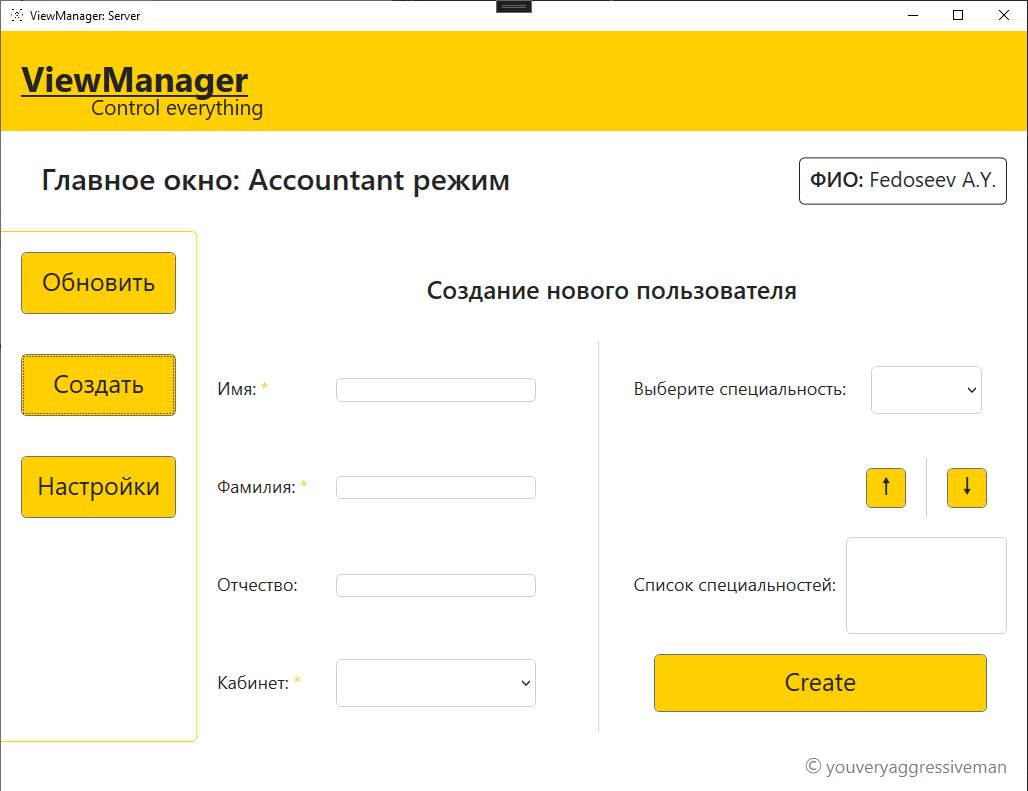


Рисунок 17 –Страница добавление нового преподавателя

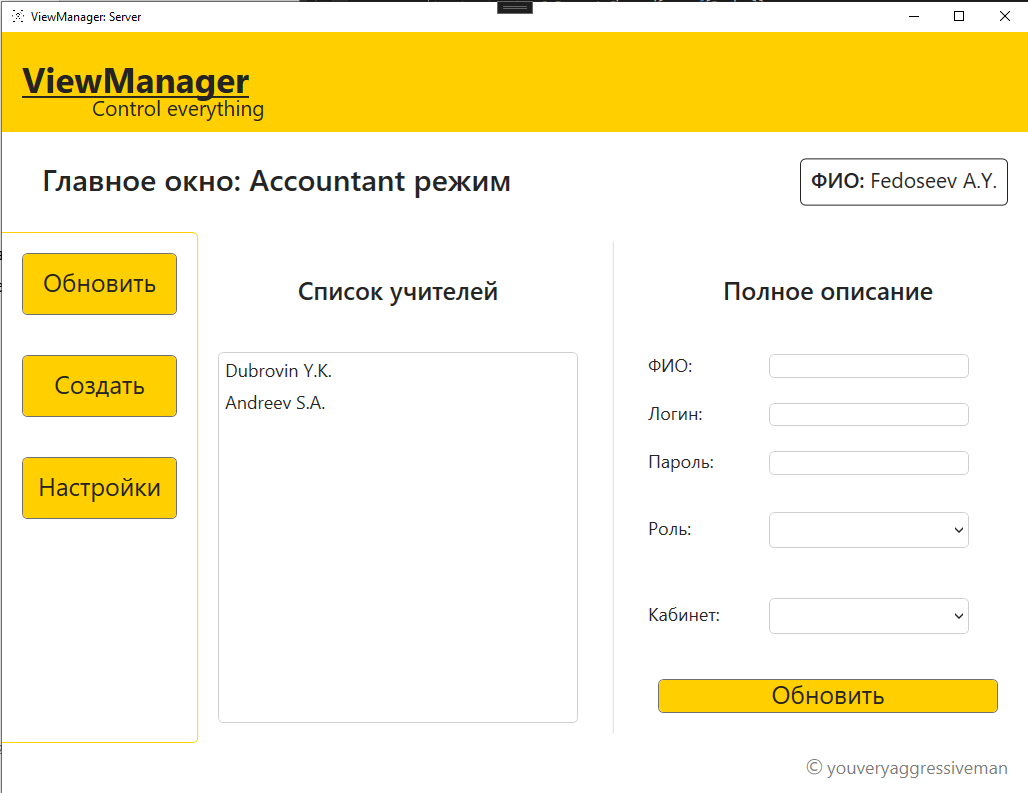


Рисунок 18 – Страница обновления информации

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была разработана клиент-серверная информационная система для учета и хранения электронных карточек обучающихся отделения информационных технологий МГГТК ФГБОУ ВО «АГУ». Также было разработано документальное сопровождение программы, включающее в себя технический и рабочий проекты.

В ходе выполнения дипломного проекта была достигнута поставленная цель и решены следующие задачи:

* выполнен анализ предметной области и спроектированы концептуальная и логическая модели БД для хранения необходимой информации;
* спроектирована функциональность и структура приложения, работающего с созданной БД. При проектировании использовался инструментарий UML-диаграмм;
* произведен выбор средств реализации БД, приложения и инсталляции;
* определены минимальные требования к техническим и программным средствам;
* создана физическая модель БД с помощью СУБД SSMS;
* создана физическая модель файловой системы для хранения данных;
* с помощью средств интегрированной среды разработки Visual Studio 2022 и языка программирования С# был разработан интерфейс приложения, позволяющий получить доступ ко всем необходимым функциям;
* с помощью средств интегрированной среды разработки Visual Studio 2022 и языка программирования С# были разработаны модули, обеспечивающие выполнение всех необходимых функций;
* создана инсталляция с помощью средств Inno Setup.

Полученные в результате анализа предметной области данные были распределены по сущностям, которые логически связаны между собой. Основываясь на логической связи сущностей, строились запросы к БД.

Для доступа к БД были использованы компоненты среды программирования C# (EnitityFramework).

Удобный интерфейс программы, с одной стороны, позволяет легко ориентироваться в приложении, не требуя от пользователя каких-либо специальных навыков работы, с другой стороны – предоставляет пользователю оперативную информацию.

Разработанное приложение обеспечивает следующие возможности:

* авторизацию пользователей;
* доступ к окнам и функциям системы через основное меню;
* внесение информации;
* редактирование информации;
* поиск информации;
* работу с удаленным сервером;
* изменения настроек приложения;
* управление компьютером через локальную сеть;
* работу с информационной БД.

Основные достоинства разработанного приложения:

* низкая требовательность к ресурсам системы;
* простота установки и настройки;
* удобство эксплуатации;
* изменение темы и языка в системе, в зависимости от предпочтений пользователя.

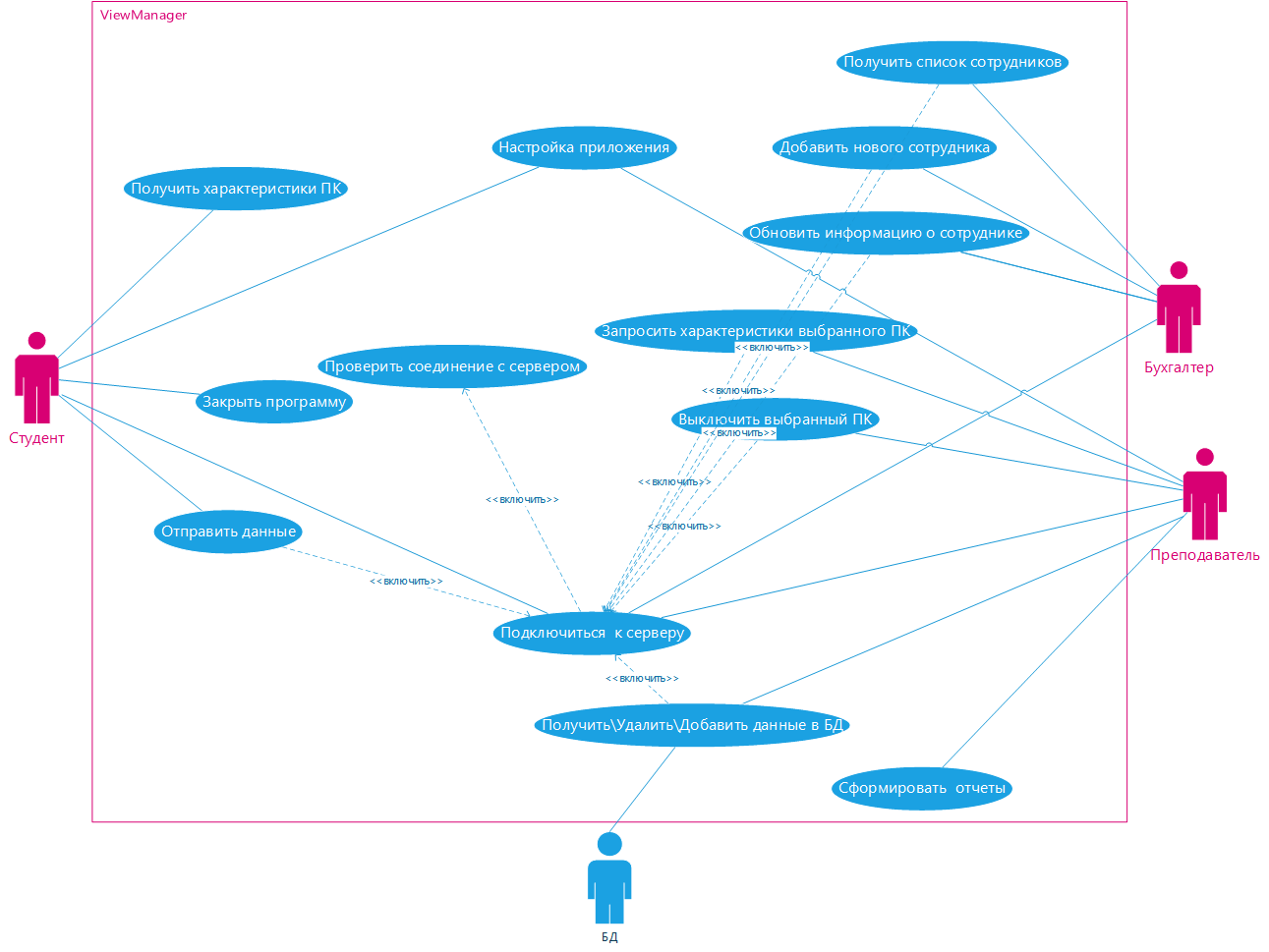
К недостаткам данной программы можно отнести то, что данные о настройках приложения не хранятся в БД.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Албахари Б., Албахари Дж. С# 7.0. Справочник. Полное описание языка. М.: Вильямс, 2018. 1024 с.
2. Виссер Д. Разработка обслуживаемых программ на языке С#. М.: ДМК Пресс, 2019. 194 с.
3. Кугаевских А. Проектирование информационных систем. Системная и бизнес-аналитика. Новосибирск: НГТУ, 2018. 256 с.
4. Лотка Р. C# и CSLA .NET Framework. Разработка бизнес-объектов. М.: Диалектика / Вильямс, 2018. 842 c.
5. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. C# 5.0 и платформа .NET 4.5 для профессионалов. М.: Вильямс, 2018. 693 c.
6. Скит Д. С# Для профессионалов. Тонкости программирования. М.: Вильямс, 2019. 608 с.
7. Borland B. Pentaho Analytics for Mongodb. М.: Книга по Требованию, 2018. 146 c.
8. Документация Microsoft. URL: https://docs.microsoft.com/ru-ru/documentation (дата обращения: 28.04.2022).
9. Официальный сайт Inno Setup. URL: https://jrsoftware.org/isinfo.php (дата обращения: 28.04.2022).
10. Простое руководство по UML-диаграммам и моделированию баз данных. URL: https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling (дата обращения: 28.04.2022).
11. Построение графиков на WPF форме под .NET Framework 4. URL: https://habr.com/ru/post/145343/

ПРИЛОЖЕНИЕ А

Use Case диаграмма проектируемой информационной системы



ПРИЛОЖЕНИЕ Б

Код основных классов

Класс AuthPageViewModelController.cs

using Newtonsoft.Json;

using ServerApp.Core.Singleton;

using ServerApp.Core.Tokens;

using ServerApp.Model;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Net.Sockets;

using System.Text;

using System.Text.Json;

using System.Text.Json.Serialization;

using System.Threading.Tasks;

using JsonSerializer = System.Text.Json.JsonSerializer;

namespace ServerApp.Controllers

{

public class AuthPageViewModelController

{

private readonly string \_connectionString;

public AuthPageViewModelController(string connectionString) =>

\_connectionString = connectionString;

public async Task<bool> AuthHelper(User user)

{

using (HttpClient client = new HttpClient())

{

var url = \_connectionString + "token?login=" + user.Login +

"&password=" + user.Password;

var result = await client.PostAsync(url, null);

if (result.IsSuccessStatusCode)

{

var endResult = JsonConvert.DeserializeObject<AuthUser>(await result.Content.ReadAsStringAsync());

TokenForApi.SetTokenApi(endResult.Token.ToString());

AuthUserSingleton.AuthUser.Id = endResult.Id.ToString();

AuthUserSingleton.AuthUser.RoleValue = endResult.RoleValue.ToString();

return true;

}

else if(result.StatusCode == System.Net.HttpStatusCode.NotFound)

{

return false;

}

throw new Exception(result.RequestMessage?.ToString());

}

}

}

}

Класс TcpController.cs

using GeneralLogic.Services.Files;

using ServerApp.Core.Singleton;

using ServerApp.Model;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Net.Sockets;

using System.Net;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ServerApp.Controllers

{

public class TcpController

{

private readonly int \_port;

private readonly string \_server;

private static TcpListener s\_listener = null;

public static ObservableCollection<string> S\_AnswerList { get; set; } = new ObservableCollection<string>();

public TcpController(int port, string server) =>

(\_port, \_server) = (port, server);

public async Task StartTcp()

{

try

{

s\_listener = new TcpListener(IPAddress.Parse(\_server), \_port);

s\_listener.Start();

while (true)

{

TcpClient client = await s\_listener.AcceptTcpClientAsync();

Thread clientThread = new Thread(new ParameterizedThreadStart(GetDataTcp));

clientThread.Start(client);

}

}

catch (Exception ex)

{

LogManager.SaveLog("Server", DateTime.Today, $"TcpClient: {ex.Message}.");

}

finally

{

if (s\_listener != null)

s\_listener.Stop();

}

}

public void GetDataTcp(object? obj)

{

var client = (TcpClient)obj;

NetworkStream stream = null;

try

{

stream = client.GetStream();

byte[] data = new byte[64];

while (true)

{

StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = stream.Read(data, 0, data.Length);

builder.Append(Encoding.UTF8.GetString(data, 0, bytes));

}

while (stream.DataAvailable);

string message = builder.ToString();

switch (message[0])

{

case 'N':

IPEndPoint ipep = (IPEndPoint)client.Client.RemoteEndPoint;

ConnectedClient connectedClient = new()

{

Ip = ipep.Address.ToString(),

Port = ipep.Port,

Name = message.Remove(0, 6)

};

ConnectedClientSingleton.ListConnectedClient.Add(connectedClient);

LogManager.SaveLog("Server", DateTime.Today, "TcpClient: " + connectedClient.Name + ": Successful connection to the server.");

break;

case 'C':

LogManager.SaveLog("Server", DateTime.Today, $"TcpClient: {message}.");

break;

case 'A':

S\_AnswerList.Add(message);

LogManager.SaveLog("Server", DateTime.Today, $"TcpClient: {message}.");

break;

default:

break;

}

}

}

catch (Exception ex)

{

LogManager.SaveLog("Server", DateTime.Today, $"TcpClient: {ex.Message}.");

}

finally

{

if (stream != null)

stream.Close();

if (client != null)

client.Close();

}

}

}

}

Класс UniversalController.cs

using GeneralLogic.Services.Files;

using HidSharp.Utility;

using Newtonsoft.Json;

using ServerApp.Core.Tokens;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Reflection.Metadata.Ecma335;

using System.Text;

using System.Threading.Tasks;

namespace ServerApp.Controllers

{

public class UniversalController<T>

{

protected readonly string \_connectionString;

private readonly string \_nameT;

public UniversalController(string connectionString)

{

\_connectionString = connectionString;

var type = typeof(T);

\_nameT = type.Name;

}

public async Task<IEnumerable<T>> GetList()

{

try

{

using (HttpClient client = new())

{

var header = client.DefaultRequestHeaders;

header.Accept.Clear();

header.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));

header.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", TokenForApi.GetTokenApi());

var result = await client.GetAsync(\_connectionString + $"api/{\_nameT}/Get{\_nameT}List");

if (result.IsSuccessStatusCode)

{

var endResult = JsonConvert.DeserializeObject<IEnumerable<T>>(await result.Content.ReadAsStringAsync());

LogManager.SaveLog("Server", DateTime.Today, "Api: The response was received successfully.");

return endResult;

}

else if (result.StatusCode == System.Net.HttpStatusCode.NotFound)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: There is no user with such data.");

return null;

}

else if (result.StatusCode == System.Net.HttpStatusCode.Unauthorized)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: The user is not logged in to the system.");

return null;

}

return null;

}

}

catch (Exception ex)

{

LogManager.SaveLog("Server", DateTime.Today, $"Api: {ex.Message}.");

throw new OperationCanceledException(ex.Message);

}

}

public async Task<T> Get(string id)

{

try

{

using (HttpClient client = new())

{

var header = client.DefaultRequestHeaders;

header.Accept.Clear();

header.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));

header.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", TokenForApi.GetTokenApi());

var url = \_connectionString + $"api/{\_nameT}/Get{\_nameT}ById?id={id}";

var result = await client.GetAsync(url);

if (result.IsSuccessStatusCode)

{

var endResult = JsonConvert.DeserializeObject<T>(await result.Content.ReadAsStringAsync());

LogManager.SaveLog("Server", DateTime.Today, "Api: The response was received successfully.");

return endResult;

}

else if (result.StatusCode == System.Net.HttpStatusCode.NotFound)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: There is no user with such data.");

return default;

}

else if (result.StatusCode == System.Net.HttpStatusCode.Unauthorized)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: The user is not logged in to the system.");

return default;

}

return default;

}

}

catch (Exception ex)

{

LogManager.SaveLog("Server", DateTime.Today, $"Api: {ex.Message}.");

throw new OperationCanceledException(ex.Message);

}

}

public async Task<bool> Create(T obj)

{

try

{

using (HttpClient client = new())

{

var header = client.DefaultRequestHeaders;

header.Accept.Clear();

header.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));

header.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", TokenForApi.GetTokenApi());

var url = \_connectionString + $"api/{\_nameT}/Create{\_nameT}";

var jsonObject = JsonConvert.SerializeObject(obj);

var content = new StringContent(jsonObject, Encoding.UTF8, "application/json");

content.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/json");

var result = await client.PostAsync(url, content);

if (result.IsSuccessStatusCode)

{

var endResult = JsonConvert.DeserializeObject<bool>(await result.Content.ReadAsStringAsync());

LogManager.SaveLog("Server", DateTime.Today, "Api: The response was received successfully.");

return endResult;

}

else if (result.StatusCode == System.Net.HttpStatusCode.NotFound)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: There is no user with such data.");

return false;

}

else if (result.StatusCode == System.Net.HttpStatusCode.Unauthorized)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: The user is not logged in to the system.");

return false;

}

return false;

}

}

catch (Exception ex)

{

LogManager.SaveLog("Server", DateTime.Today, $"Api: {ex.Message}.");

throw new OperationCanceledException(ex.Message);

}

}

public async Task<bool> Put(T obj)

{

try

{

using (HttpClient client = new())

{

var header = client.DefaultRequestHeaders;

header.Accept.Clear();

header.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));

header.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", TokenForApi.GetTokenApi());

var url = \_connectionString + $"api/{\_nameT}/Update{\_nameT}";

var jsonObject = JsonConvert.SerializeObject(obj);

var content = new StringContent(jsonObject, Encoding.UTF8, "application/json");

content.Headers.ContentType = new System.Net.Http.Headers.MediaTypeHeaderValue("application/json");

var result = await client.PutAsync(url, content);

if (result.IsSuccessStatusCode)

{

var endResult = JsonConvert.DeserializeObject<bool>(await result.Content.ReadAsStringAsync());

LogManager.SaveLog("Server", DateTime.Today, "Api: The response was received successfully.");

return endResult;

}

else if (result.StatusCode == System.Net.HttpStatusCode.NotFound)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: There is no user with such data.");

return false;

}

else if (result.StatusCode == System.Net.HttpStatusCode.Unauthorized)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: The user is not logged in to the system.");

return false;

}

return false;

}

}

catch (Exception ex)

{

LogManager.SaveLog("Server", DateTime.Today, $"Api: {ex.Message}.");

throw new OperationCanceledException(ex.Message);

}

}

public async Task<bool> Delete(T obj, string id)

{

try

{

using (HttpClient client = new())

{

var header = client.DefaultRequestHeaders;

header.Accept.Clear();

header.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));

header.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", TokenForApi.GetTokenApi());

var url = \_connectionString + $"api/{\_nameT}/Remove{\_nameT}";

var result = await client.DeleteAsync(url);

if (result.IsSuccessStatusCode)

{

var endResult = JsonConvert.DeserializeObject<bool>(await result.Content.ReadAsStringAsync());

LogManager.SaveLog("Server", DateTime.Today, "Api: The response was received successfully.");

return endResult;

}

else if (result.StatusCode == System.Net.HttpStatusCode.NotFound)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: There is no user with such data.");

return false;

}

else if (result.StatusCode == System.Net.HttpStatusCode.Unauthorized)

{

LogManager.SaveLog("Server", DateTime.Today, "Api: The user is not logged in to the system.");

return false;

}

return false;

}

}

catch (Exception ex)

{

LogManager.SaveLog("Server", DateTime.Today, $"Api: {ex.Message}.");

throw new OperationCanceledException(ex.Message);

}

}

}

}