**СОДЕРЖАНИЕ**

Введение…………………………………………………………………………...4

1 Технический проект…………………………………………………………….7

1.1 Анализ предметной области……………………………………………….7

1.2 Постановка задачи………………………………………………………….8

1.3 Требования к программному обеспечению и техническим средствам…8

1.4 Проектирование функциональности программы……………………….10

1.5 Проектирование структуры программы…………………………………11

1.6 Разработка концептуальной модели базы данных……….……………..15

1.7 Выбор средств разработки………………………………………………..18

2 Рабочий проект………………………………………………………………...22

2.1 Проектирование физической модели базы данных…………………….22

2.2 Функциональное взаимодействие модулей программы………………..25

2.3 Входные и выходные данные…………………………………………….31

2.4 Установка и настройка программы……………………………………...31

2.5 Работа с программой……………………………………………………...33

Заключение……………………………………………………………………….37

Список использованных источников…………………………………….……..39

Приложения

А – Use Case диаграмма………………………………………………………40

Б – Код основных классов……………………………………………………41

# **ВВЕДЕНИЕ**

Темой дипломного проекта является «Разработка и создание приложения для перевода иностранного текста с графического изображения».

В реалиях современного мира, особенно в интернете, большая часть современных статей, инструкций, документаций, да даже субтитров к играм и фильмам написана на иностранном языке. Далеко не все современные пользователи интернета знают те самые иностранные языки, а пользоваться онлайн переводчиком не всегда удобно, или, например, вообще возможно.

Целью дипломного проекта будет написание приложения, способного переводить иностранный текст с изображений на русский язык.

Объектом исследования будут графические файлы, содержащие текстовую информацию.

Предметом исследования данного проекта будут являться инструментальные средства, с помощью которых будет происходить распознавание текста на фото и его дальнейший перевод на русский язык.

Написание данного дипломного проекта является целесообразным, потому что:

* приложение будет способно переводить иностранный текст с изображения, что значительно упросит получение информации из интернета;
* приложение будет иметь крайне низкие системные требования к компьютеру пользователя;
* на рынке современных решений крайне мало аналогов данной программы.

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 базы данных (БД);
* разработаны структура и функциональность приложения;
* 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 практически в любых областях работы за компьютером, где необходимо знание иностранного языка.

Основными пользователями данной программы будут люди, которые часто сталкиваются с иностранным текстом при работе за компьютером.

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 приложения, включая разработку логической модели БД. Также в нем будет произведен выбор инструментов для физической реализации построенных моделей данных приложения.

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

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

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

# **1 ТЕХНИЧЕСКИЙ ПРОЕКТ**

## **1.1 Анализ предметной области**

В современном мире, при работе за компьютером, каждый его пользователь постоянно контактирует с возможно незнакомыми для себя иностранными языками. Причём речь идёт не только о работниках IT или подобных сфер. Даже в играх пользователь постоянно сталкивается с иностранным языком.

К большому сожалению людей, проживающих в России, в интернете русский язык не является столь популярным у иностранцев, как тот же английский язык. Поэтому большинство документаций, различных инструкций, да даже субтитров к большинству игр выпускаются исключительно на английском языке. И это вовсе не плохо, за счёт использования самого популярного межнационального языка удаётся получить куда больший охват аудитории, чем при использовании другого, менее популярного языка. Проблемой это может стать в тех случаях, когда пользователь либо вообще не знает иностранного языка, либо знает его на очень низком уровне, что может негативно отразиться не только на качестве его работы за компьютером, но и на его общих впечатлениях от процесса работы.

В связи со всем вышеперечисленным, появилась потребность в приложении, которое поможет облегчить и улучшить работу за компьютером современным пользователям. Такое приложение должно обеспечивать:

* возможность быстрого доступа к системе перевода текста;
* быстрый перевод текста, в независимости от оборудования пользователя;
* отсутствие значительной нагрузки на компьютер пользователя;
* улучшение качества работы пользователя;
* просмотр истории переводов;
* выбор конкретного переводчика для работы из предложенных.

## **1.2 Постановка задачи**

Необходимо разработать клиент-серверное приложение, которое предоставит возможность перевода иностранного текста с графического изображения. Также создать документационное сопровождение приложения, включающее в себе технический и рабочий проекты.

Разрабатываемое приложение должно быть простым и понятным в использовании, иметь простой и понятный интерфейс, а также предоставлять следующий функционал:

* авторизация пользователя;
* доступ к настройкам программы;
* возможность перевода текста на определённой области экрана;
* доступ к истории перевода пользователя;
* возможность выбора переводчика для перевода текста;
* возможность загрузки скриншота для перевода текста.

Необходимые данные должны храниться в удалённой БД.

Вся необходимая информация должна быть распределена по таблицам БД, которая находится в третьей нормальной форме.

Входными данными в приложении будут скриншоты с иностранным текстом.

Выходными данными в приложении будут переведённые на требуемый язык тексты.

Способ решения поставленных задач зависит от выбора средств разработки.

## **1.3 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 7 / 10 / 11;
* .NET 6 +.

Сервер должен быть оснащён:

* операционной системой Windows Server 2019;
* SQL Server Express 2019 +;
* .NET 6 +;
* 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 и иметь не менее 1 ядра;
* оперативная память не менее 1 Гб;
* дисплей с разрешением не менее 1024x768 точек;
* 100 Мб свободного места на диске;
* клавиатура;
* манипулятор типа «мышь».

Также должно быть стабильное Ethernet-соединение.

Серверное оборудование должно соответствовать следующим аппаратным требованиям:

* центральный процессор фирмы INTEL или AMD с тактовой частотой не менее 3.2 ГГц и иметь не менее 8 ядер;
* оперативная память не менее 16 Гб;
* 150 Гб свободного места на диске.

Также должно быть стабильное Ethernet-соединение.

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 2022 +;
* SQL Server Express 2019 +;
* SQL Server Management Studio.

## **1.4 Проектирование функциональности программы**

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.

Неавторизованному пользователю будет доступен только базовый функционал программы, а конкретно:

* возможность перевода текста с выбранной области экрана;
* возможность настройки программы.

Также у неавторизованного пользователя будут ограничения на количество переводимых символов в день.

Для того, чтобы получить полный доступ, пользователю необходимо будет авторизоваться.

После авторизации пользователь получит доступ к большему количеству функционала программы:

* возможность загрузить файл изображения и получить из него текст на оригинальном и переведённом языках;
* возможность просматривать историю своих переводов.

Помимо этого, у пользователя значительно возрастёт лимит на количество переведённых символов в день.

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

## **1.5 Проектирование структуры программы**

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 [3].

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 [3].

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.

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 окон:

* окно авторизации – будет вызываться по нажатию на выделенную кнопку/сочетание клавиш; здесь пользователь сможет войти в свой аккаунт;
* окно регистрации – будет вызываться по нажатию на выделенную кнопку/сочетание клавиш; здесь пользователь сможет создать новый аккаунт;
* окно настроек программы и профиля – будет вызываться при запуске программы, а также по нажатию на выделенную кнопку/сочетание клавиш; здесь пользователь может изменять параметры программы и своего профиля;
* окно истории переводов и загрузки скриншота – будет вызываться по нажатию на выделенную кнопку/сочетание клавиш; здесь пользователь сможет сделать скриншот области экрана для работы программы и посмотреть историю своих переводов.

Необходимые данные будут хранится в БД.

Проектируемая структура информационной системы представлена на рисунке 1.

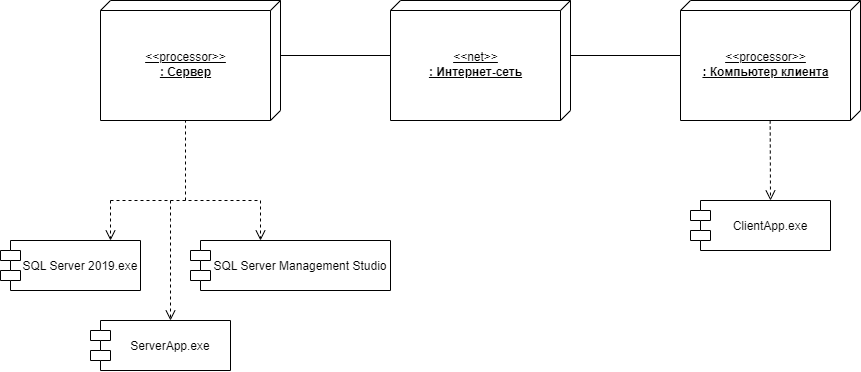


Рисунок 1 – Схема проектируемой структуры информационной системы

Проектируемая структура клиентского приложения представлена на рисунке 2.



Рисунок 2 – Схема проектируемой структуры клиентского приложения

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

## **1.6 Разработка концептуальной модели базы данных**

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 [3].

Наиболее удобной для реализации поставленной задачи является свободная реляционная СУБД, которая использует декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной БД, управляемой соответствующей СУБД.

В теории БД эти таблицы называют отношениями (relations) – поэтому и БД 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.

В данном курсовом проекте предметной областью является процесс перевода иностранного текста на нужный язык. БД будет предназначена для хранения, контроля и поддержка сведений о пользователях, истории их переводов и вспомогательных средствах.

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 [3].

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

* роли (идентификатор, наименование);
* страны (идентификатор, наименование);
* пользователи (идентификатор, идентификатор роли, логин, пароль, псевдоним, почта, имя, фамилия, идентификатор страны, дата рождения);
* истории переводов (идентификатор, идентификатор пользователя, скриншот, идентификатор выбранного OCRa, распознанный текст, точность распознанного текста, идентификатор выбранного переводчика, идентификатор входного языка, идентификатор выходного языка, перевод);
* языки (идентификатор, название, псевдоним для переводчика, псевдоним для OCRa);
* настройки (идентификатор пользователя, название, идентификатор выбранного OCRa, идентификатор выбранного переводчика, API ключ для переводчика, идентификатор входного языка, идентификатор выходного языка, цвет вывода результата);
* OCRs (идентификатор, название);
* переводчики (идентификатор, название).

Затем была разработана концептуальная модель БД (рисунок 3) и логическая модель, которая отражает связи между сущностями (рисунок 4).

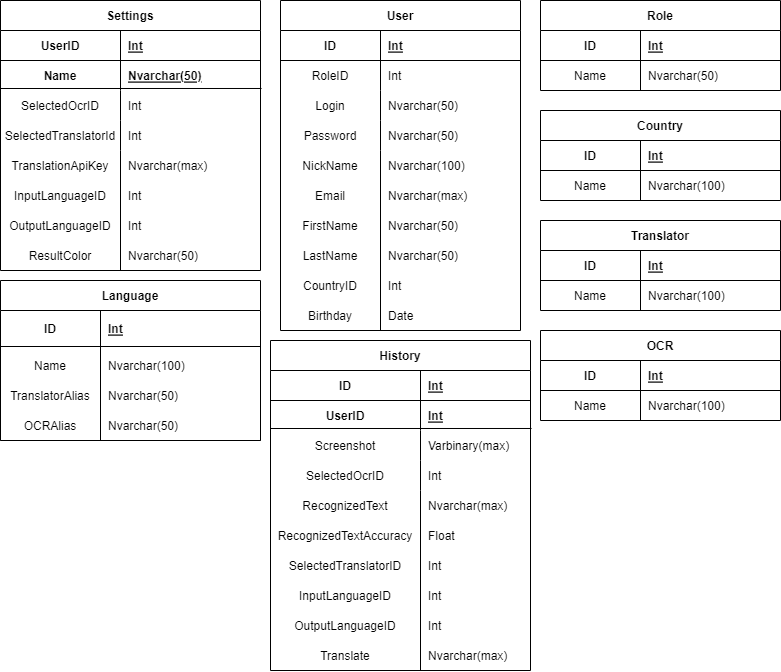


Рисунок 3 – Концептуальная модель БД

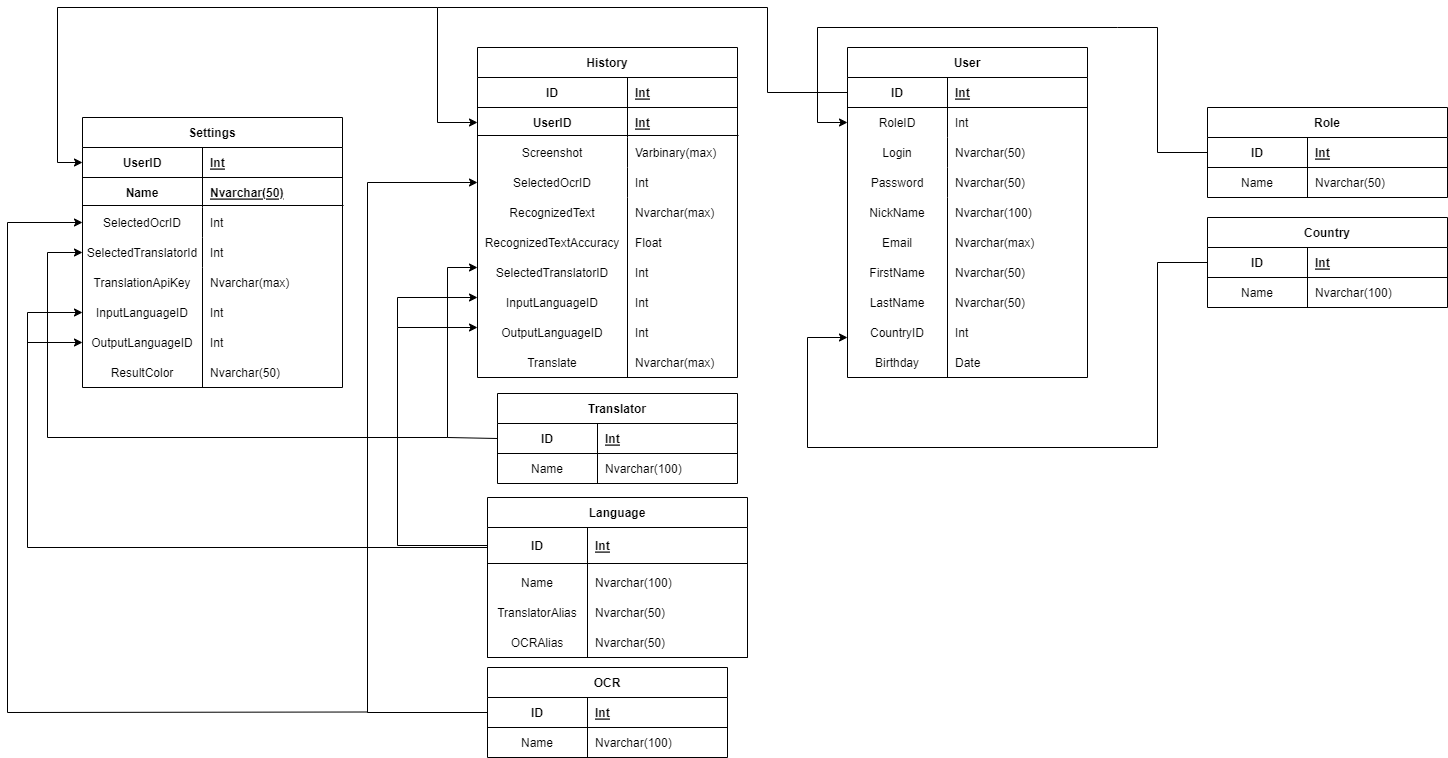


Рисунок 4 – Логическая модель БД

## **1.7 Выбор средств разработки**

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 2022.

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 [2].

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: прокачанной 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 [7].

Visual Studio 2022 является последней версией инструмента и дает   
64-разрядную версию Visual Studio, а также собственный набор преимуществ производительности для повышения продуктивности. Здесь 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 SQL Server Management Studio. 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 [7].

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 [8].

# **2 РАБОЧИЙ ПРОЕКТ**

## **2.1 Проектирование физической модели базы 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.

На основе логической модели, представленной в пункте 1.6, 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.

Физическая модель разработанной БД состоит из следующих таблиц:

* OCR (таблица 1);
* Translator (таблица 2);
* Country (таблица 3);
* Role (таблица 4);
* Language (таблица 5);
* User (таблица 6);
* Settings (таблица 7);
* History (таблица 8.

Таблица 1 – Структура таблицы OCR (Система распознавания текста – с.р.т.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ | Обязатель-ность |
| ID | Идентификатор | INT | PK | Да |
| Name | Название с.р.т. | NVARCHAR[100] | - | Да |

Таблица 2 – Структура таблицы Translator (Переводчик)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ | Обязатель-ность |
| ID | Идентификатор | INT | PK | Да |
| Name | Название переводчика | NVARCHAR[100] | - | Да |

Таблица 3 – Структура таблицы Country (Страна)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ | Обязатель-ность |
| ID | Идентификатор | INT | PK | Да |
| Name | Название страны | NVARCHAR[100] | - | Да |

Таблица 4 – Структура таблицы Role (Роль)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ | Обязатель-ность |
| ID | Идентификатор | INT | PK | Да |
| Name | Название роли | NVARCHAR[50] | - | Да |

Таблица 5 – Структура таблицы Language (Язык)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ | Обязатель-ность |
| ID | Идентификатор | INT | PK | Да |
| Name | Название роли | NVARCHAR[100] | - | Да |
| TranslatorAlias | Псевдоним для переводчика | NVARCHAR[50] | - | Нет |
| OCRAlias | Псевдоним для с.р.т. | NVARCHAR[50] | - | Нет |

Таблица 6 – Структура таблицы User (Пользователь)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ | Обязатель-ность |
| ID | Идентификатор | INT | PK | Да |
| RoleID | Идентификатор роли | INT | FK | Да |
| Login | Логин | NVARCHAR[50] | - | Да |
| Password | Пароль | NVARCHAR[50] | - | Да |
| NickName | Псевдоним | NVARCHAR[100] | - | Нет |
| Email | Почта | NVARCHAR[MAX] | - | Да |
| FirstName | Имя | NVARCHAR[50] | - | Нет |
| LastName | Фамилия | NVARCHAR[50] | - | Нет |
| CountryID | Идентификатор страны | INT | FK | Нет |
| Birhday | Дата рождения | Date | - | Нет |

Таблица 7 – Структура таблицы Settings (Настройки)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ | Обязатель-ность |
| UserID | Идентификатор | INT | PK | Да |
| Name | Название профиля настроек | NVARCHAR[100] | PK | Да |
| Selected-OCRID | Идентификатор выбранной с.р.т. | INT | FK | Нет |
| Selected-TranslatorID | Идентификатор выбранного переводчика | INT | FK | Нет |
| Translator-ApiKey | Ключ для Api переводчика | NVARCHAR[MAX] | - | Нет |
| Input-LanguageID | Идентификатор языка распознавания текста | INT | FK | Нет |
| Output-LanguageID | Идентификатор языка перевода | INT | FK | Нет |
| ResultColor | Цвет для вывода результата | NVARCHAR[50] | - | Нет |

Таблица 8 – Структура таблицы History (История)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ | Обязатель-ность |
| ID | Идентификатор | INT | PK | Да |
| UserID | Идентификатор пользователя | NVARCHAR[100] | PK | Да |
| Screenshot | Снимок экрана | VARBINARY | - | Нет |
| Selected-OCRID | Идентификатор выбранной с.р.т. | INT | FK | Нет |
| Recognized-Text | Распознанный текст | NVARCHAR[MAX] | - | Нет |
| Recognized-TextAccuracy | Точность распознанного текста | FLOAT | - | Нет |
| Selected-TranslatorID | Идентификатор выбранного переводчика | INT | FK | Нет |
| Input-LanguageID | Идентификатор языка распознавания текста | INT | FK | Нет |
| Output-LanguageID | Идентификатор языка перевода | INT | FK | Нет |
| Translate | Переведённый текст | NVARCHAR[MAX] | - | Нет |

## **2.2 Функциональное взаимодействие модулей программы**

На основе разработанного технического проекта было написано приложение для перевода иностранного текста с графического изображения. 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.Linq – предоставляет классы и интерфейсы, поддерживающие запросы с использованием LINQ;
* 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;
* Tesseract – система распознания текста;
* Microsoft.EntityFrameworkCore – библиотека для работы с базой данных;
* GlobalHotKeys – библиотека для создания глобальных горячих клавиш [6].

Также были разработаны следующие элементы для взаимодействия пользователя с программой:

* SignInWindow.xaml.сs (окно авторизации) – осуществляет функции авторизации в приложении и переход на главное окно;
* SignUpWindow.xaml.сs (окно регистрации) – осуществляет функции регистрации в приложении;
* MainWindow.xaml.cs (главное окно программы) – осуществляет переход между страницами приложения, а также обеспечивает вывод руководства по использованию программы;
* HistoryAndTranslatePage.xaml.cs (страница истории переводов и загрузки изображения) – предоставляет возможность посмотреть пользователю историю своих переводов, а также на этой странице можно загрузить изображения и получить перевод текста с него;
* ScreenshotWindow.xaml.cs (окно для создания скриншота) – осуществляет функцию создания скриншота части экрана;
* SettingsPage.xaml.cs (страница настроек) – обеспечивает настройку параметров приложения и профиля.

Для реализации функциональности описанного пользовательского интерфейса были разработаны следующие модули, взаимодействие которых представлено на рисунке 5:

* 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;
* Resources (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;
* ViewModel – модуль, являющейся связующим звеном между интерфейсом и логикой программы.

Для реализации функциональности модуля Command были написаны следующие классы:

* RelayCommand.cs – абстрактный класс, реализующий паттерн команды;
* CommandManager.cs – класс, инкапсулирующий в себе логику создания комманд.

Методы класса RelayCommand.cs:

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

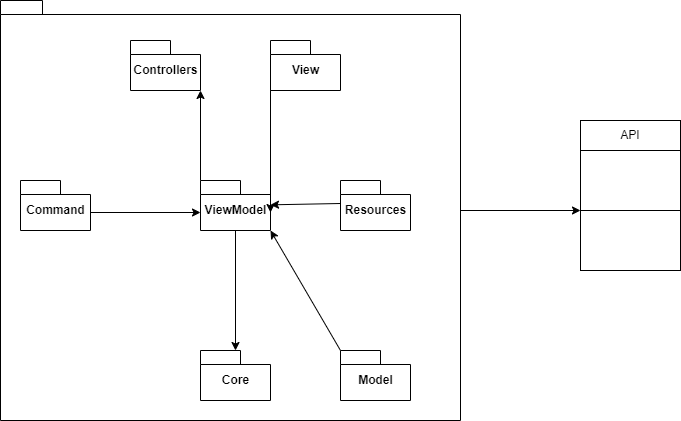


Рисунок 5 – Схема функционального взаимодействия модулей

Для реализации функциональности модуля Controllers был написан класс UniversalController<T>.cs, реализующий универсальный контроллер для работы с API. Диаграмма классов модуля Controllers представлена на рисунке 6.

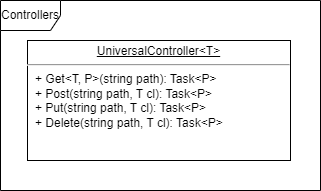


Рисунок 6 – Диаграмма классов модуля Controllers

Для реализации функциональности модуля Resources было создано специальное окно MessageResultWindow.xaml для вывода результатов.

Для реализации функциональности модуля Core были написаны следующие классы:

* ImagePreparation.cs – реализует различные преобразования изображения, например из Image в ImageSource;
* MainWindowManager.cs – реализует переопределение главного окна приложения;
* ProgramElementFinder.cs – реализует поиск какого-либо объекта в программе по названию, например поиск страницы или класса по его названию;
* RegisterGlobalHotkey.cs – реализует добавление глобальных горячих клавиш для приложения;
* ScreenCapture.cs – реализует работу с библиотекой WinApi, требуется для создания скриншота экрана;
* SettingsProfile.cs – реализует получение и сохранение настроек приложения пользователя в БД.

Для реализации функциональности модуля ViewModel был написан класс BaseViewModel.cs, реализующий логику взаимодействия приложения с интерфейсом. Схема класса представлена на рисунке 7.

Метод класса BaseViewModel.cs protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null) вызывается при установке нового значения у элемента интерфейса для его обновления.

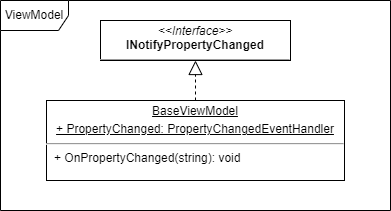


Рисунок 7 – Диаграмма классов модуля ViewModel

## **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 таблицах в БД.

Входными данными в модуле SignInWindow.xaml.cs являются данные из таблицы «Пользователи». При авторизации пользователя входные и выходные данные хранятся в статическом классе ConnectedUserSingleton.cs в следующих полях:

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

Входными данными для модуля SignUpWindow.xaml.cs являются необходимые данные для регистрации нового аккаунта, а выходными – новый созданный аккаунт пользователя.

Входными данными для модуля ScreenshotWindow.xaml.cs является снимок экрана, а выходными является новая фотография выделенной области на снимке экрана.

Входными данными для модуля SettingsPage.xaml.cs являются данные из таблицы «Settings», а выходными – данные для работы приложения.

Входными данными для модуля HistoryAndTranslatePage.xaml.cs является фотография с текстом, а выходными – распознанный текст с фотографии и его перевод, а также история переводов авторизованного пользователя.

## **2.4 Установка и настройка программы**

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

* папки:
* Resources;
* runtimes;
* ru;
* libraries;
* ScreenRecognition.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 и иметь не менее 1 ядра;
* стабильное Ethernet-соединение.
* оперативная память не менее 1 Гб;
* дисплей с разрешением не менее 1024x768 точек;
* 100 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 «ScreenRecognition.exe», двойным кликом. В результате открывается главное меню приложения, в котором по умолчанию открыта страница «Настройки» (рисунок 8).

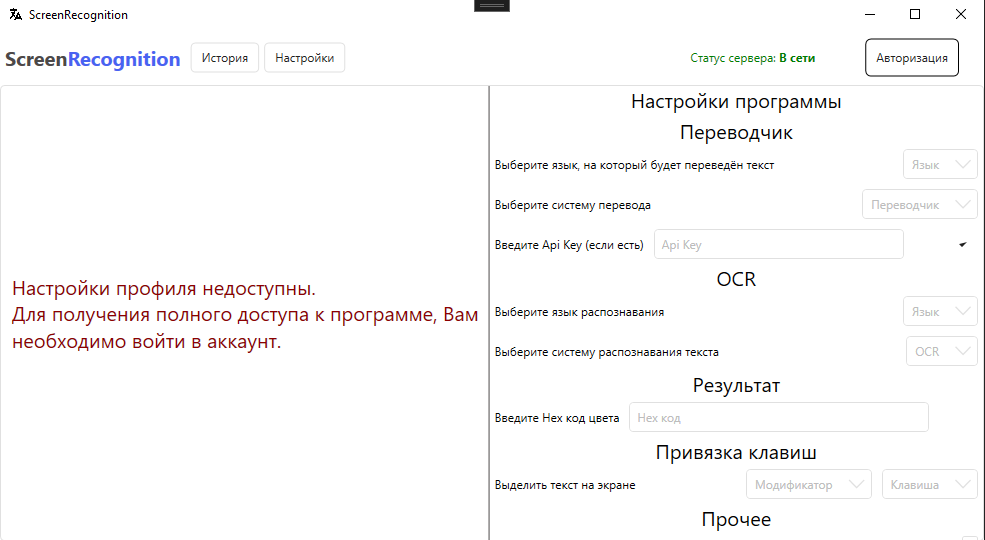


Рисунок 8 – Главное меню (гость)

В левом углу находится навигационная панель приложения, а при запуске приложения откроется страница с настройками приложения.

В случае, если пользователь не авторизован, то функционал приложения будет сильно ограничен. Пользователь сможет только изменять настройки приложения, а также сможет сделать скриншот экрана, выделить на нём нужную область с текстом и получить перевод этого текста (рисунок 9). Причём цвет текста перевода зависит от настроек пользователя.



Рисунок 9 – Результат перевода с экрана

Для получения полного доступа необходимо авторизоваться.

В правом углу отображается текущее состояние сервера, а также кнопка авторизации. По её нажатию открывается окно авторизации (рисунок 10).

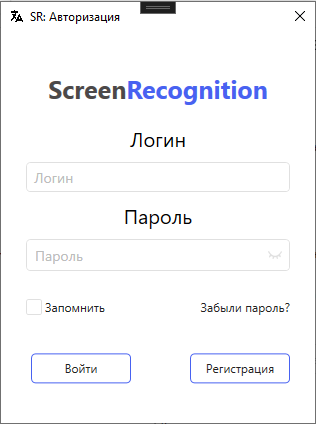


Рисунок 10 – Окно авторизации

Здесь пользователю нужно ввести данные для входа, а если аккаунта нет, то его можно создать. Сделать это можно, нажав на кнопку «Регистрация» (рисунок 11).

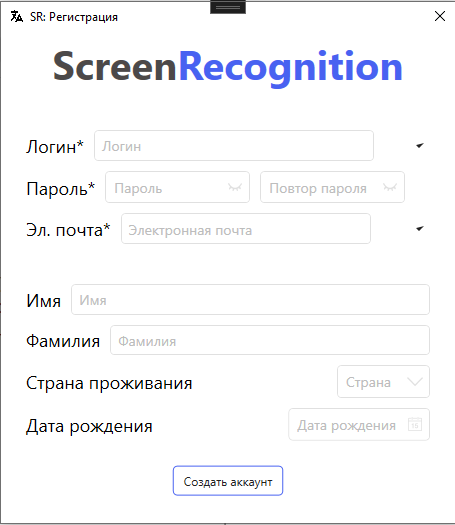


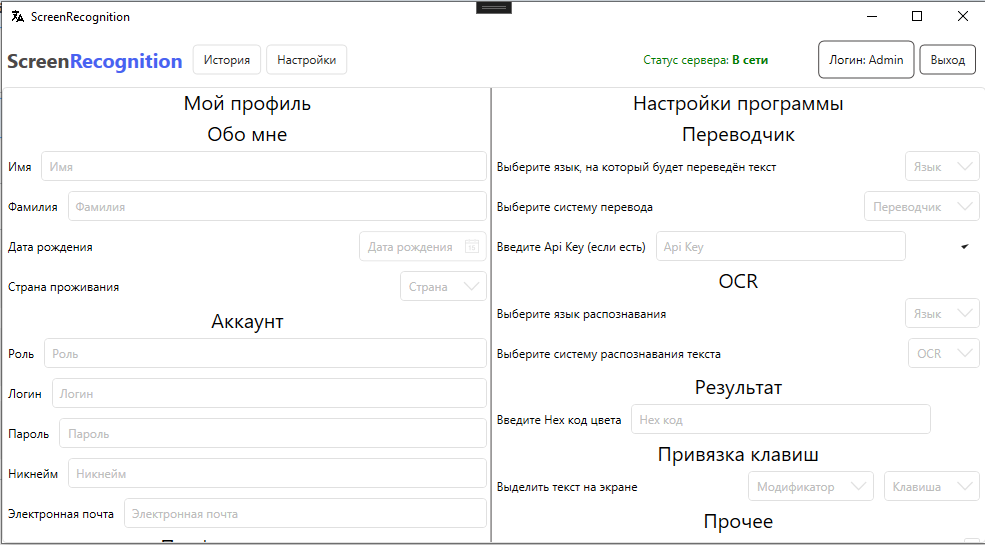
Рисунок 11 – Окно регистрации

После заполнения всех обязательных полей, при отсутствии проблем, будет создан новый аккаунт, и пользователь сможет авторизоваться.

После входа в аккаунт у пользователя будут доступны все функции программы (рисунок 12).

Теперь пользователь сможет настроить свой профиль, а также появляется доступ ко вкладке «История» (рисунок 13).

На этой странице пользователь может посмотреть свою историю прошлых переводов с дополнительной информацией.

Рисунок 12 – Главное меню (пользователь)

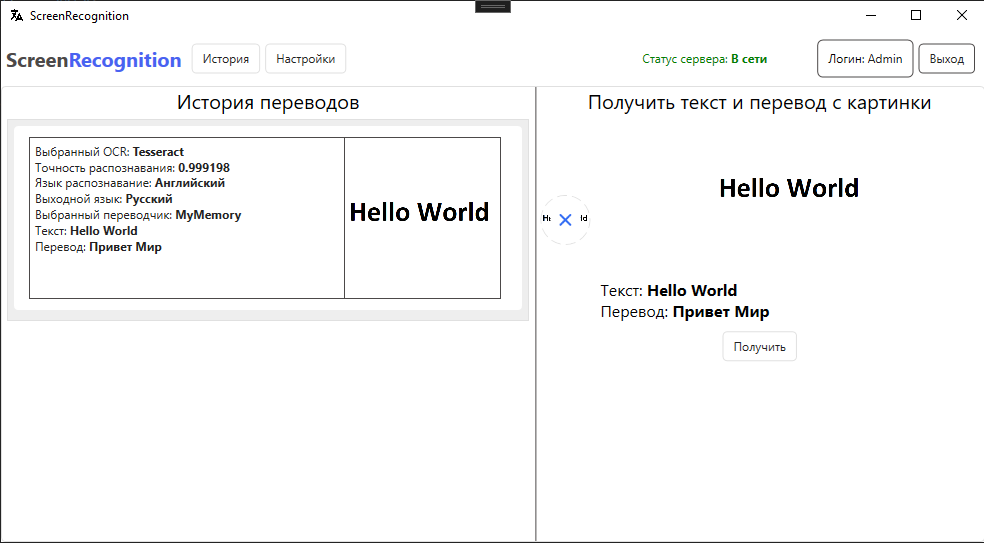


Рисунок 13 – Окно с историей переводов и загрузка фотографии для перевода

На правой стороне страницы пользователь может выбрать фотографию, на текст которой будет распознан и переведён на выбранный на странице «Настройки» язык (см. рисунок 12).

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта было разработано клиент-серверное приложение для перевода иностранного текста с графического изображения. Также было разработано документальное сопровождение программы, включающее в себя технический и рабочий проекты.

В ходе выполнения курсового 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 SQL Server;
* 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 таблицам, которые логически связаны между собой. Основываясь на логической связи таблиц, 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:

* регистрацию и авторизацию пользователей;
* доступ к окнам и функциям системы через основное меню;
* внесение информации;
* редактирование информации;
* перевод выбранного текста на выбранный язык;
* работу с информационной БД.

Основные достоинства разработанного приложения:

* низкая требовательность к ресурсам системы;
* простота установки и настройки;
* удобство эксплуатации.

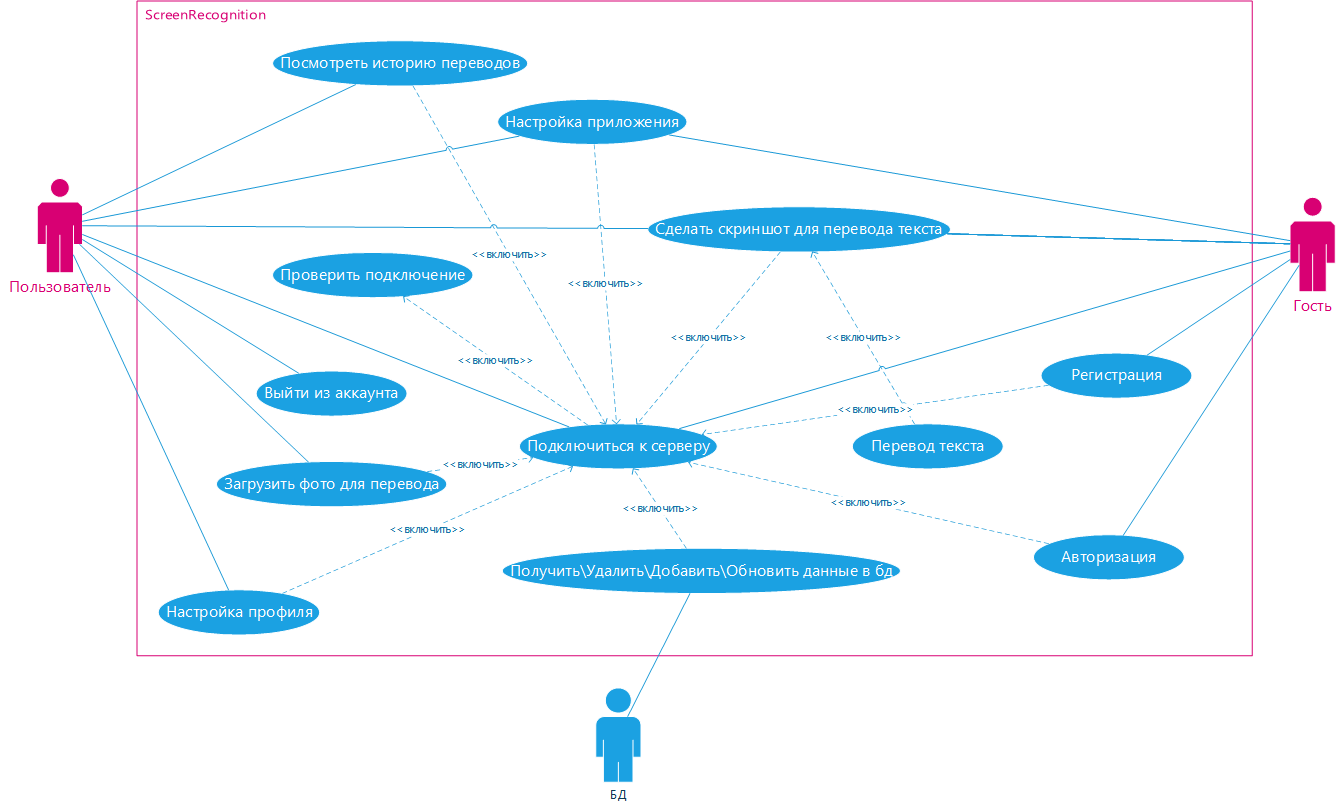
К недостаткам данной программы можно отнести то, что вся информация в БД хранится в незашифрованном виде.

# **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. Документация Microsoft. URL: https://docs.microsoft.com/ru-ru/documentation (дата обращения: 28.10.2022).
8. Официальный сайт Inno Setup. URL: https://jrsoftware.org/isinfo.php (дата обращения: 28.11.2022).
9. 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: 28.10.2022).
10. Официальный сайт TesseracOCR. URL: https://tesseract.patagames.com (дата обращения: 30.10.2022).

# **ПРИЛОЖЕНИЕ А**

Use Case диаграмма проектируемой информационной системы



# **ПРИЛОЖЕНИЕ Б**

Код основных классов

Класс UniversalController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http.Headers;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using System.Security.Permissions;

using System.Net.Http.Json;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Runtime.Serialization;

using System.Text.Json;

using System.Reflection.Metadata;

namespace ScreenRecognition.Desktop.Controllers

{

public class UniversalController

{

private readonly string \_webPath = "http://localhost:80/api/";

public UniversalController(string webPath = "")

{

if (!string.IsNullOrEmpty(webPath))

{

\_webPath = webPath;

}

}

public async Task<P?> Get<T, P>(string path)

{

var result = await ControllerOperations<T, P>("get", path);

return result;

}

public async Task<P?> Post<T, P>(string path, T? cl)

{

var result = await ControllerOperations<T, P>("post", path, cl);

return result;

}

private async Task<P?> ControllerOperations<T, P>(string type, string path, T? cl = default(T))

{

HttpClient client = new HttpClient();

P? result;

client.BaseAddress = new Uri($"{\_webPath}");

client.DefaultRequestHeaders.Accept.Clear();

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

HttpResponseMessage response = new HttpResponseMessage();

try

{

if (type == "get")

response = await client.GetAsync(path);

else if (type == "post")

{

response = await client.PostAsJsonAsync($"{path}", cl);

response.EnsureSuccessStatusCode();

}

result = await response.Content.ReadFromJsonAsync<P>();

}

catch

{

return default(P);

}

return result;

}

}

}

Класс ImagePreparation.cs

using System;

using System.Collections.Generic;

using System.Drawing.Imaging;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media.Imaging;

using System.Windows.Media;

namespace ScreenRecognition.Desktop.Core

{

public class ImagePreparation

{

public ImageSource ImageToImageSource(System.Drawing.Image image)

{

BitmapImage bitmap = new BitmapImage();

using (MemoryStream stream = new MemoryStream())

{

image.Save(stream, ImageFormat.Png);

stream.Seek(0, SeekOrigin.Begin);

bitmap.BeginInit();

bitmap.StreamSource = stream;

bitmap.CacheOption = BitmapCacheOption.OnLoad;

bitmap.EndInit();

}

return bitmap;

}

public System.Drawing.Image? Crop(System.Drawing.Image image, System.Drawing.Rectangle selection)

{

System.Drawing.Bitmap? bmp = image as System.Drawing.Bitmap;

System.Drawing.Bitmap? cropBmp = bmp?.Clone(selection, bmp.PixelFormat);

image.Dispose();

return cropBmp;

}

public System.Drawing.Image ImageWpfToGDI(ImageSource image)

{

MemoryStream ms = new MemoryStream();

var encoder = new BmpBitmapEncoder();

encoder.Frames.Add(BitmapFrame.Create(image as BitmapSource));

encoder.Save(ms);

ms.Flush();

return System.Drawing.Image.FromStream(ms);

}

public ImageSource ByteToImage(byte[] imageData)

{

BitmapImage biImg = new BitmapImage();

MemoryStream ms = new MemoryStream(imageData);

biImg.BeginInit();

biImg.StreamSource = ms;

biImg.EndInit();

ImageSource imgSrc = biImg as ImageSource;

return imgSrc;

}

}

}

Класс MainWindowManager.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace ScreenRecognition.Desktop.Core

{

public static class MainWindowManager

{

public static void Set(Window window)

{

var app = App.Current.MainWindow;

App.Current.MainWindow = window;

app.Close();

window.Show();

}

}

}

Класс ProgramElementFinder.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace ScreenRecognition.Modules.Modules

{

public class ProgramElementFinder

{

public static T? FindByName<T>(string name, string assemblyName)

{

T? result = default(T);

try

{

if (name == null || assemblyName == null)

return result;

var assemblyElement = Assembly.Load(assemblyName);

var typeElement = assemblyElement.GetExportedTypes()

.FirstOrDefault(type => type.Name == name);

if (typeElement == null)

return result;

result = (T?)Activator.CreateInstance(assemblyName, typeElement.FullName)?.Unwrap();

}

catch { }

return result;

}

}

}

Класс RegisterGlobalHotkey.cs

using System.Runtime.InteropServices;

using System.Windows.Interop;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Globalization;

using System.Windows.Input;

using HandyControl.Controls;

using System.Reflection.PortableExecutable;

using GlobalHotKeys;

using System.Reactive.Linq;

using System.Threading;

namespace ScreenRecognition.Desktop.Core

{

/// <summary>

/// Это класс для добавления глобального хоткея

/// </summary>

public class RegisterGlobalHotkey

{

private static IDisposable? s\_hotkey;

private static IDisposable? s\_subscription;

private static HotKeyManager? s\_hotKeyManager;

public RegisterGlobalHotkey(GlobalHotKeys.Native.Types.VirtualKeyCode key, GlobalHotKeys.Native.Types.Modifiers modifiers, Action func)

{

s\_hotKeyManager = new HotKeyManager();

s\_hotkey = s\_hotKeyManager?.Register(GlobalHotKeys.Native.Types.VirtualKeyCode.KEY\_Q, GlobalHotKeys.Native.Types.Modifiers.Control);

s\_subscription = s\_hotKeyManager?.HotKeyPressed

.ObserveOn(SynchronizationContext.Current)

.Subscribe(hotkey => func());

}

public static bool HotkeyEnabledStatus()

{

if (s\_hotKeyManager != null)

return true;

return false;

}

public static void Dispose()

{

s\_hotkey?.Dispose();

s\_subscription?.Dispose();

s\_hotKeyManager?.Dispose();

}

}

}

Класс ScreenCapture.cs

using ScreenRecognition.Desktop.View.Windows;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Imaging;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace ScreenRecognition.Desktop.Core

{

public class ScreenCapture

{

[DllImport("user32.dll")]

private static extern IntPtr GetForegroundWindow();

[DllImport("user32.dll", CharSet = CharSet.Auto, ExactSpelling = true)]

public static extern IntPtr GetDesktopWindow();

[StructLayout(LayoutKind.Sequential)]

private struct Rect

{

public int Left;

public int Top;

public int Right;

public int Bottom;

}

[DllImport("user32.dll")]

private static extern IntPtr GetWindowRect(IntPtr hWnd, ref Rect rect);

public static Image CaptureDesktop()

{

return CaptureWindow(GetDesktopWindow());

}

public static Bitmap CaptureActiveWindow()

{

return CaptureWindow(GetForegroundWindow());

}

public static Bitmap CaptureWindow(IntPtr handle)

{

var rect = new Rect();

GetWindowRect(handle, ref rect);

var bounds = new Rectangle(rect.Left, rect.Top, rect.Right - rect.Left, rect.Bottom - rect.Top);

var result = new Bitmap(bounds.Width, bounds.Height);

using (var graphics = Graphics.FromImage(result))

{

graphics.CopyFromScreen(new System.Drawing.Point(bounds.Left, bounds.Top), System.Drawing.Point.Empty, bounds.Size);

}

return result;

}

}

}