МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

Бизнес-проект «Программное обеспечение для захвата, распознавания и перевода текстового контента в реальном времени. Бэкенд-разработка»

Студент _____ Аргаткин Алексей Алексеевич
(подпись, дата) 01.06.2026 (ФИО полностью)

Руководитель _____ С.А. Зинкин
(подпись, дата) 01.06.2026 (фамилия, инициалы)

Нормоконтролёр _____ А.С. Бычков
(подпись, дата) 01.06.2026 (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 08 июня 2026 г. № 18)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
"07" 04 2026 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Аргаткин Алексей Алексеевич гр. 22ВВП1 факультета вычислительной техники

2.Тема работы: Бизнес-проект «Программное обеспечение для захвата, распознавания и перевода текстового контента в реальном времени. Бэкенд-разработка»

Тема утверждена приказом ПГУ № 347/о от "07" 04 2026 г.

3. Руководитель бакалаврской работы д.т.н., профессор, каф. ВТ ПГУ Зинкин С. А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать Бизнес-проект «Программное обеспечение для захвата, распознавания и перевода текстового контента в реальном времени. Бэкенд-разработка», позволяющее выбрать область экрана, получить изображение этой области, распознать текст и вывести перевод в отдельном окне или поверх исходного текста. В приложении должна быть интегрирована многоязычная системы перевода, обеспечивающая высокую точность и минимальную задержку обработки. Приложение должно обладать удобным клиентским интерфейсом, позволяющим пользователю управлять процессом перевода, настраивать языки и работать с выделенными элементами контента.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить существующие аналоги, выделить их достоинства и недостатки. Выбрать язык программирования, который подойдет для реализации Бэкенд-составляющей. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал библиотек для взаимодействия с изображением на экране и создания оконного приложения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	07.04.2026 – 13.04.2026	
2	Анализ требований	32	13.04.2026 – 20.04.2026	
3	Создание макета Бэкенд части	30	20.04.2026 – 26.04.2026	
4	Выбор технологий, применяемых при проектировании и разработке	40	26.04.2026 – 02.05.2026	
5	Реализация Бэкенд составляющей	66	02.05.2026 – 08.05.2026	
6	Разработка основной части программы	70	08.05.2026 – 15.05.2026	
7	Оформление пояснительной записки	54	15.05.2026 – 21.04.2026	

Дата выдачи задания "07" 04 2026 г.

Руководитель бакалаврской работы _____ Зинкин С. А.
(подпись, дата)

Задание к исполнению принял студент _____ Аргаткин А.А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)
05.06.2026

Реферат

Бакалаврская работа содержит 101 лист, 28 рисунков, 9 таблиц, 15 источников и 3 приложения.

БЭКЕНД-РАЗРАБОТКА, API, OCR, ПЕРЕВОД ТЕКСТА, FASTAPI, PYTHON, ОБРАБОТКА ИЗОБРАЖЕНИЙ, КЭШИРОВАНИЕ, JSON, АСИНХРОННАЯ ОБРАБОТКА

Цель работы – разработка бэкенд-части ПО для захвата, распознавания и перевода текстового контента в реальном времени.

В конечном итоге был разработан backend-модуль, который принимает изображение области экрана от клиентской части, выполняет предварительную обработку, распознает текст, переводит найденные фрагменты и возвращает результат в структурированном формате.

В качестве языка программирования был выбран Python с использованием фреймворка FastAPI. Программное обеспечение удовлетворяет поставленной задаче: реализованы API-методы для обработки изображения, валидация входных данных, предварительная подготовка кадра, запуск OCR, перевод текста, кэширование повторяющихся результатов и обработка ошибок. Клиентская часть может обращаться к backend через единый программный интерфейс и получать как полный текст перевода, так и данные для отображения результата.

Разработанное программное обеспечение может применяться при чтении иностранных интерфейсов, инструкций, субтитров, изображений с текстом и учебных материалов.

Содержание

Перечень сокращений и обозначений 6
Введение 7
1 Система бэкенд-разработки ПО экранного перевода 9
1.1 Анализ предметной области 9
1.1.1 Актуальность разработки 10
1.1.2 Анализ существующих аналогов 11
1.1.3 Обоснование актуальности разработки программы 23
1.1.4 Постановка задачи 24
1.2 Выбор технологий и средств разработки приложения 28
1.3 Проектирование архитектуры backend-сервиса 34
1.3.1 Общая архитектура серверной части 35
1.3.2 Граница ответственности бэкенд-разработки 36
1.3.3 Проектирование API взаимодействия с клиентской частью 36
1.3.4 Логические слои обработки данных 37
1.4 Разработка backend-сервиса 38
1.4.1 Реализация API и координатора обработки 39
1.4.2 Описание основных backend-классов с примерами реализации 40
1.4.3 Реализация предварительной обработки изображения 42
1.4.4 Реализация OCR-обработки 42
1.4.5 Реализация перевода и кэширования 43
1.4.6 Реализация формата ответа и обработки ошибок 44
1.4.7 Руководство по интеграции с клиентской частью 45
1.4.8 Тестирование и отладка 46
1.4.9 Последовательность взаимодействия компонентов 47
1.4.10 Анализ полученных результатов и развитие 48
2 Стартап-проект Screen Translate 50
2.1 Обоснование актуальности 50
2.2 Описание и обоснование выбора ВКР в формате ВКРС 52
2.3 Описание существующей проблемы на рынке 53

2.4 Описание решения проблемы и продуктового решения 55
2.4.1 Наименование продукта 55
2.4.2 Назначение и сфера применения 55
2.4.3 Пользовательский сценарий 56
2.4.4 Фронтенд-зона ответственности 56
2.4.5 Функциональные возможности реализованной программы ... 57
2.4.6 Конкурентоспособность и инновационность 58
2.4.7 Степень готовности продукта, поставка и обслуживание 59
2.5 Общая характеристика стартап-проекта и сферы деятельности ... 59
2.5.1 Основная бизнес-идея 59
2.5.2 Сфера деятельности 60
2.5.3 Целевая аудитория 60
2.5.4 Текущее состояние разработки 61
2.6 Объем, структура и развитие рынка 62
2.7 Анализ конкурентов и аналогов 64
2.8 Бизнес-модель и команда стартап-проекта 65
2.8.1 Бизнес-модель 65
2.8.2 Каналы продаж и продвижения 66
2.8.3 Команда стартап-проекта 67
2.9 Финансовая модель и необходимость привлечения инвестиций .. 68
2.9.1 Предпосылки финансовой модели 68
2.9.2 Расходы проекта 69
2.9.3 Потребность в финансировании 69
2.9.4 Точка безубыточности и инвестиционная привлекательность 70
2.10 Производственный и организационный план 71
2.10.1 Общие сведения о производстве цифрового продукта 71
2.10.2 Технологическая цепочка разработки 71
2.10.3 Кадровое обеспечение и организация работ 72
2.10.4 Материалы, оборудование, безопасность и экологичность .. 72
2.11 Описание развития бизнес-проекта 73
2.12 Описание рисков 74

2.13 Заключение по стартап-проекту 77
3 Охрана труда 79
3.1 Анализ опасных и вредных факторов при работе с ПЭВМ 79
3.2 Требования к рабочему месту 80
3.3 Освещение, микроклимат и режим труда 80
3.4 Электробезопасность и пожарная безопасность 81
Заключение 82
Список использованных источников 84
Приложение А – UML-диаграммы 86
Приложение Б – Сокращенный листинг программы 93
Приложение В – Презентация 100

Перечень сокращений и обозначений

В данной работе используются следующие обозначения:

Клиентская часть – часть приложения, отвечающая за экранные окна и взаимодействие с пользователем

ПО – программное обеспечение

API – программный интерфейс взаимодействия между клиентской и серверной частями приложения

Backend – серверная часть программного продукта, выполняющая обработку данных и предоставляющая API

EasyOCR – библиотека для оптического распознавания текста

FastAPI – Python-фреймворк для создания backend-сервисов и REST API

HTTP – протокол обмена запросами и ответами между компонентами

JSON – текстовый формат обмена структурированными данными

OCR – оптическое распознавание символов

Pydantic – библиотека валидации и описания схем входных и выходных данных

Pillow – библиотека Python для обработки изображений

UI – пользовательский интерфейс клиентской части

Введение

В современном информационном пространстве пользователя окружает значительная часть информации, представленная не в виде редактируемого текста, а как изображение на экране. Надпись может находиться в интерфейсе программы, в игре, в видеоролике, на скриншоте, в окне трансляции или в документе, где копирование текста ограничено. В таких случаях пользователь видит текст, но не может быстро передать его в переводчик стандартными средствами операционной системы.

Проблема особенно заметна при работе с иностранными программами и обучающими материалами. Пользователь вынужден делать снимок экрана, открывать сторонний сервис распознавания, вручную выделять фрагмент, копировать результат и только после этого переводить его. Такой порядок действий занимает время и не подходит для случаев, когда текст на экране меняется часто.

Разрабатываемое ПО предназначено для сокращения указанной цепочки действий. В совместном проекте клиентская часть отвечает за выбор области экрана и отображение результата, а backend принимает изображение этой области, распознает текст, выполняет перевод и возвращает данные клиентскому приложению.

Под бэкенд-разработкой в данном случае понимается разработка серверной и вычислительной части приложения: API-методов, схем входных и выходных данных, предварительной обработки изображений, запуска OCR, обращения к модулю перевода, кэширования, обработки ошибок и подготовки структурированного результата для клиентской части.

Актуальность темы связана с необходимостью создания устойчивого backend-сервиса для программ, работающих с экранным текстом в реальном времени. Даже удобный пользовательский интерфейс не решает задачу, если серверная часть медленно обрабатывает изображения, некорректно распознает текст или возвращает результат в неудобном для отображения виде.

Цель выпускной квалификационной работы – разработать бэкенд-часть ПО для захвата, распознавания и перевода текстового контента в реальном времени.

Для достижения цели необходимо решить следующие задачи:

- изучить предметную область экранного перевода;
- выполнить анализ существующих аналогов;
- сформулировать требования к backend-части;
- определить границы ответственности серверной части и клиентского приложения;
- выбрать технологии и средства разработки;
- спроектировать структуру API и внутреннего конвейера обработки;
- реализовать основные модули backend-сервиса;
- провести проверку API, OCR-обработки, перевода и устойчивости backend-части.

Объектом разработки является ПО для захвата, распознавания и перевода текстового контента. Предметом разработки является backend-часть данного ПО, обеспечивающая обработку изображений, распознавание, перевод и передачу результата клиентской части.

Практическая значимость работы заключается в создании серверной части приложения, которая позволяет клиентскому интерфейсу использовать распознавание и перевод экранного текста без сложных промежуточных действий. Разработанное ПО может применяться при чтении иностранных интерфейсов, инструкций, субтитров, изображений с текстом и учебных материалов.

Структура выпускной квалификационной работы включает введение, основной раздел с анализом предметной области, выбором технологий, проектированием и разработкой серверной части, раздел по охране труда, заключение, список использованных источников и приложения.

1 Система бэкенд-разработки ПО экранного перевода

Основная часть работы посвящена разработке серверной и вычислительной логики приложения Screen Translator. Backend-слой принимает данные от клиентской части, выполняет обработку изображения, распознавание текста, перевод и возвращает результат в формате, пригодном для отображения пользователю.

В отличие от клиентского интерфейса, backend-часть должна учитывать корректность входных данных, устойчивость OCR и перевода, повторную обработку похожих изображений, ограничение размера кадра, кэширование, обработку ошибок и скорость ответа. Поэтому проектирование начинается с анализа сценария обмена между клиентской частью и серверным модулем.

Структура первого раздела выстроена последовательно: сначала рассматривается предметная область и существующие решения, затем формулируется постановка задачи, обосновывается выбор средств разработки, описывается архитектура backend-сервиса и приводится реализация серверной части.

1.1 Анализ предметной области

Для того чтобы разработать устойчивое ПО, необходимо проанализировать не только действия пользователя, но и внутренний путь данных: получение изображения, подготовку кадра, распознавание, перевод и возвращение результата клиентской части.

Современный пользователь ожидает, что приложение будет работать поверх обычных окон операционной системы. Оно не должно требовать переноса текста в отдельный документ, ручного обрезания скриншота или длительной настройки перед каждым переводом. Поэтому важным требованием становится минимизация количества действий.

Предметная область экранного перевода находится на пересечении пользовательских интерфейсов, компьютерного зрения и машинного перевода. Для бэкенд-разработчика ключевой задачей является организация надежного конвейера обработки, который связывает изображение с OCR, переводчиком и форматом ответа.

На основе анализа было решено разработать ПО, в котором клиентская часть передает backend изображение выбранной области, исходный и целевой языки, а серверная часть возвращает распознанный текст, перевод, координаты фрагментов и диагностическую информацию о выполнении обработки.

Дополнительная особенность предметной области состоит в том, что пользователь может работать с динамическими окнами: видео, играми, прокручиваемыми страницами, меняющимися сообщениями. Для backend это означает необходимость быстро обрабатывать повторяющиеся запросы, использовать кэш и не выполнять лишний OCR при незначительном изменении кадра.

Важное значение имеет согласованность результата. Backend должен возвращать данные так, чтобы клиентская часть могла отобразить перевод поверх исходного текста или в отдельном окне, не разбирая внутренние особенности OCR и переводчика.

1.1.1 Актуальность разработки

Актуальность разработки обусловлена тем, что пользователю часто требуется быстро понять текст, который технически недоступен для копирования. Это может быть надпись в изображении, элемент интерфейса, субтитр, фрагмент окна игры или сообщение в окне программы.

Большинство стандартных переводчиков хорошо работают с обычным текстом, но хуже приспособлены к ситуации, когда текст находится только на экране. Пользователь вынужден использовать несколько отдельных

инструментов: скриншот, OCR-сервис и переводчик. Такой процесс разрывает контекст и снижает скорость работы.

Экранный перевод в реальном времени требует backend-сервиса, который не блокирует клиентскую часть и возвращает понятный статус обработки. Если каждый запрос выполняется непредсказуемо долго или ошибка не описана, пользовательский сценарий быстро становится неудобным.

Особую актуальность имеет структурированный ответ. Клиентской части требуется не только итоговая строка перевода, но и информация о распознанных блоках, координатах, времени выполнения, языке и возможных ошибках.

Разработка backend-части ПО является самостоятельной и значимой задачей. От нее зависит, сможет ли пользователь применять распознавание и перевод в реальной работе, а не только как экспериментальную функцию.

1.1.2 Анализ существующих аналогов

Для определения требований к разрабатываемому интерфейсу были рассмотрены аналоги, которые решают похожую задачу: перевод текста с изображения, распознавание текста на экране или перевод поверх другого окна.

Первый представленный аналог – Translumo.exe. К преимуществам данного приложения относятся: качественный интерфейс, использование горячих клавиш с возможностью переназначения и поддержание разных моделей для перевода, возможность перевода на множество языков.

На рисунке 1 изображен интерфейс приложения Translumo.exe.

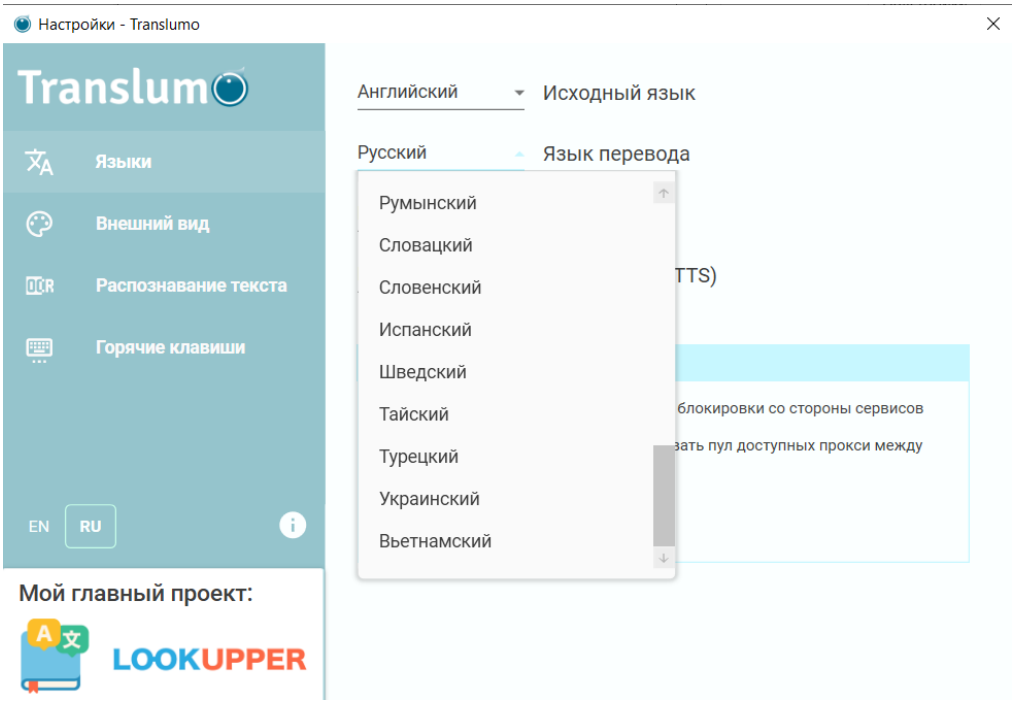


Рисунок 1 – Интерфейс приложения Translumo.exe.

Тем не менее у приложения имеются недостатки связанные как с переводом, так и с технической составляющей. Недостаток первый: у приложения всего один режим, в котором текст переводится подряд, без абзацев, переносов на новую строку, что усложняет восприятие переводимого текста.

На рисунке 2 изображена проблема формата переведенного текста.

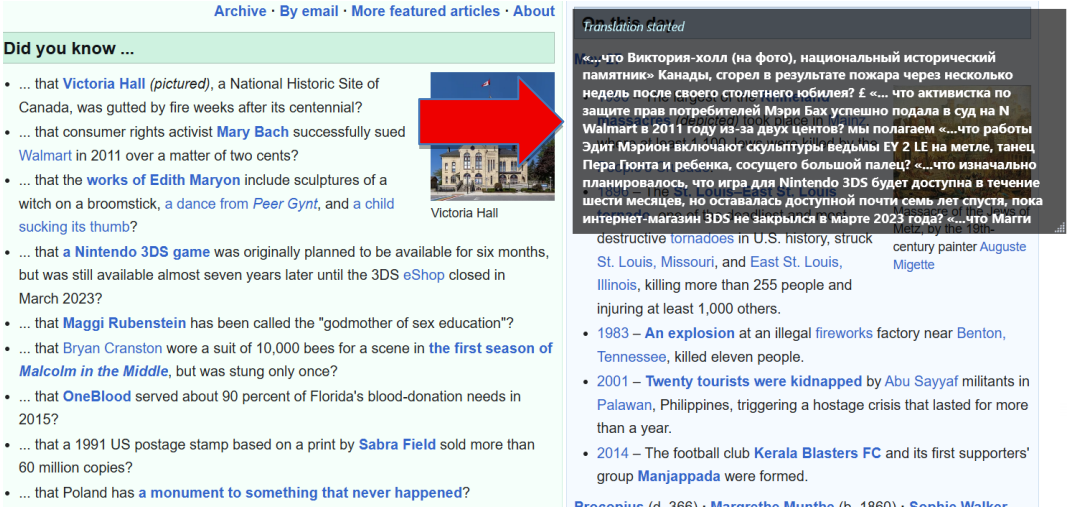


Рисунок 2 – Проблема формата переведенного текста.

В нашем приложении есть overlay-режим, в котором переведенный текст не просто сохраняет свой формат, но и накладывается на оригинал, что

делает восприятие текста очень простым и понятным. Таким образом, если пользователя не устраивает формат оконного режима, он может воспользоваться overlay-режимом.

На рисунке 3 изображено демонстрация удобства overlay режима.

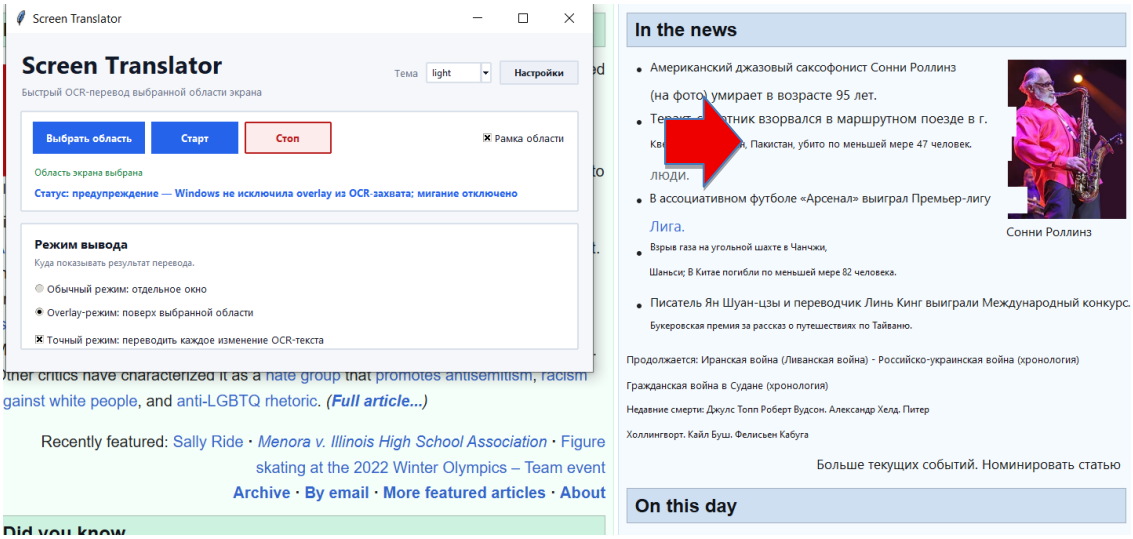


Рисунок 3 – Демонстрация удобства overlay режима.

Также у аналога имеются проблемы с некоторыми переводчиками, например переводчик DeepL работает неккоректно и часто вызывает ошибки.

На рисунке 4 изображено проблема работы переводчика DeepL.

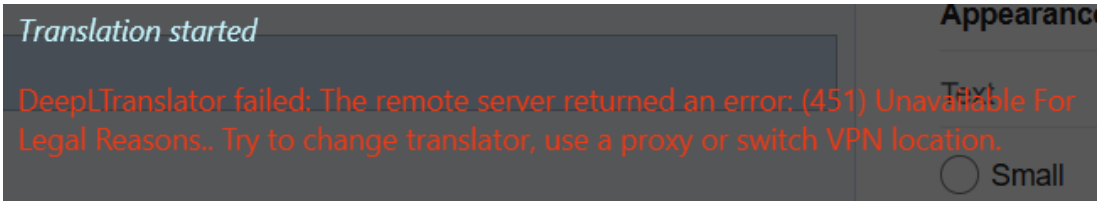


Рисунок 4 – Сообщение о проблеме работы переводчика DeepL.

Интерфейс аналога также имеет недостатки: окно перевода открывается при запуске приложения и закрыть его возможно только через диспетчер задач.

Для смены модели перевода, переназначения горячих клавиш, смены языка перевода требуется постоянное открытие и закрытие основного окна, так как перевод не может начаться с открытым окном приложения, что усложняет работу с интерфейсом приложения.

На рисунке 5 изображено невозможность перевода при открытом приложении.

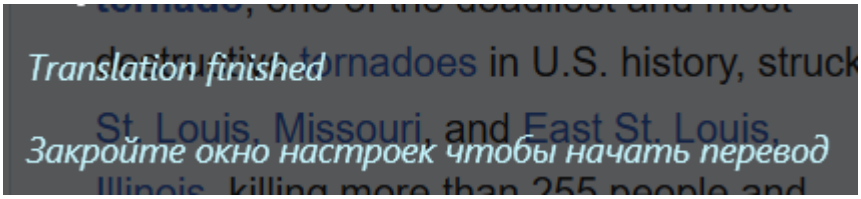


Рисунок 5 – Невозможность перевода при открытом приложении.

В нашем приложении такая проблема отсутствует. Переводить текст, менять языки и другие настройки можно без закрытия окна приложения.

Ещё одним преимуществом нашего приложения является возможность делать скриншоты переведенной области, аналог не поддерживает такую возможность. В настройках нашего приложения можно отключить скрытие overlay при создании скриншотов.

На рисунке 6 изображено возможность управлять видимостью overlay.

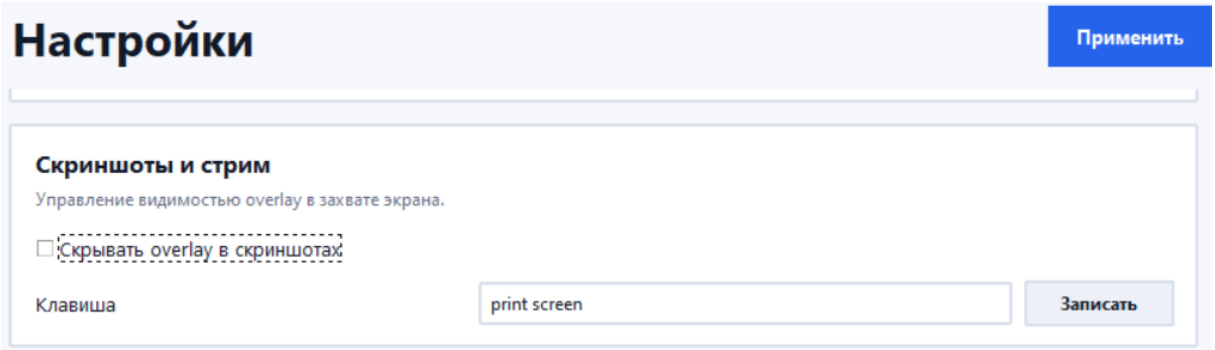


Рисунок 6 – Возможность управлять видимостью overlay.

У приложения аналога есть также и преимущества и главное из них – скорость перевода. Перевод текста в данном приложении занимает от 1 до 2 секунд, в то время как перевод в нашем приложении занимает больше времени – около 5-10 секунд.

Еще одним преимуществом приложения аналога является поддержания интерфейса на двух языках – русский и английский. Это делает интерфейс приложения доступным и понятным не только для русскоязычного сегмента.

На рисунке 7 изображено возможность работы с приложением на двух языках.

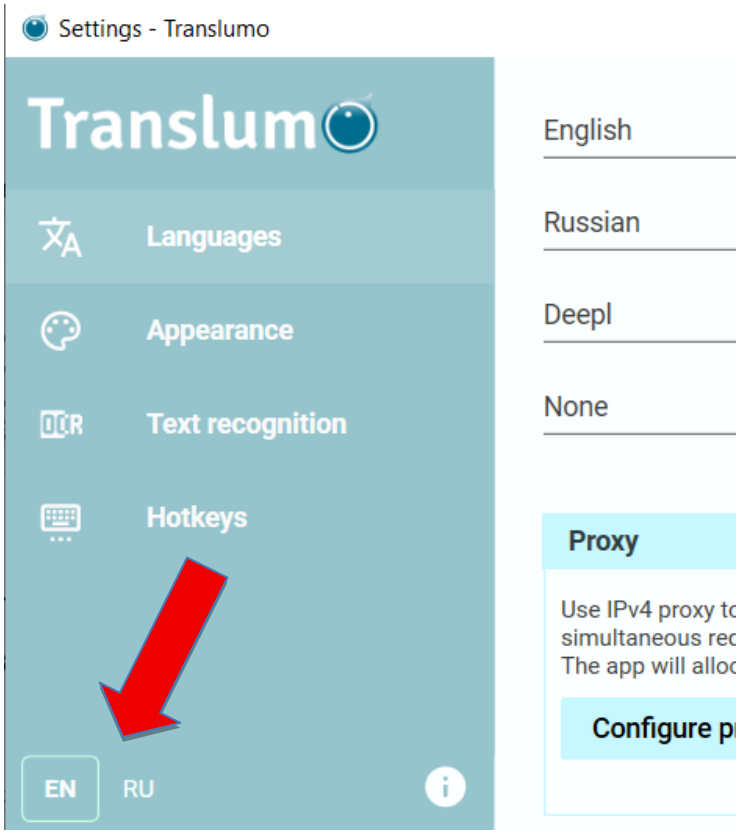


Рисунок 7 – Возможность работы с приложением на двух языках.

Также значимым преимуществом приложения аналога является поддержка разных моделей OCR, что даёт пользователю возможность получать более точные переводы и работать со сложными текстами.

На рисунке 8 изображено поддержка разных OCR под конкретные задачи.

Рисунок 8 – Поддержка разных OCR под конкретные задачи.

У приложения нет обозначения процесса перевода, поэтому в случае ошибки или бага невозможно понять, процесс перевода запущен или же перевода не произошло вообще. Выявлены программные ошибки, связанные с этим: в процессе игры при нажатии запуска перевода ничего не происходит, хотя область экрана выбрана и горячая клавиша нажата.

Второй представленный аналог – ScreTran.exe.

На рисунке 9 изображено интерфейс приложения ScreTran.

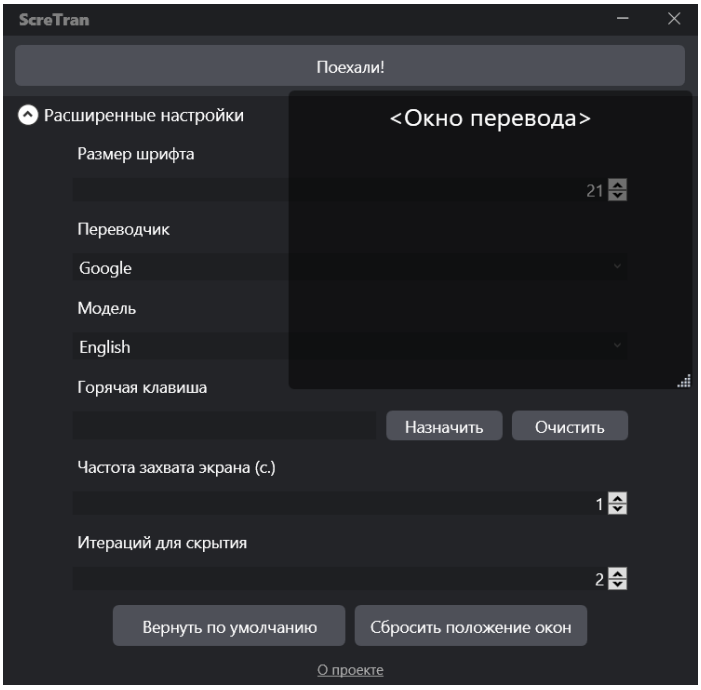


Рисунок 9 – Интерфейс приложения ScreTran.

Первый недостаток приложения который видно сразу – неудобный интерфейс. При открытии приложения появляется основное окно с настройками, окно перевода и рамка, которая является областью захвата экрана. Область захвата нужно двигать и настраивать по ширине и длине самому. В нашем же приложении захват экрана происходит по нажатию на кнопку и пользователь быстро может захватить необходимую область экрана за 1 клик.

Также у приложения аналога имеется проблема с окном переведенного текста – если область захвата большая и текста много, пользователь не может прокрутить полный переведенный текст колёсиком мыши в маленьком окне.

Чтобы увидеть весь текст – нужно обязательно растягивать окно перевода до больших размеров. Таким образом область захвата и окно перевода будут занимать слишком много рабочего пространства на экране, такое отсутствие гибкости делает приложение неудобным для выполнения более узких и конкретных задач.

Следующим недостатком приложения аналога является малое количество настроек: нет возможности менять цвет текста, прозрачность окна и его цвет, что делает приложение не очень удобным для использования и восприятия переведенного текста.

На рисунке 10 изображено малое количество настроек для пользователя.

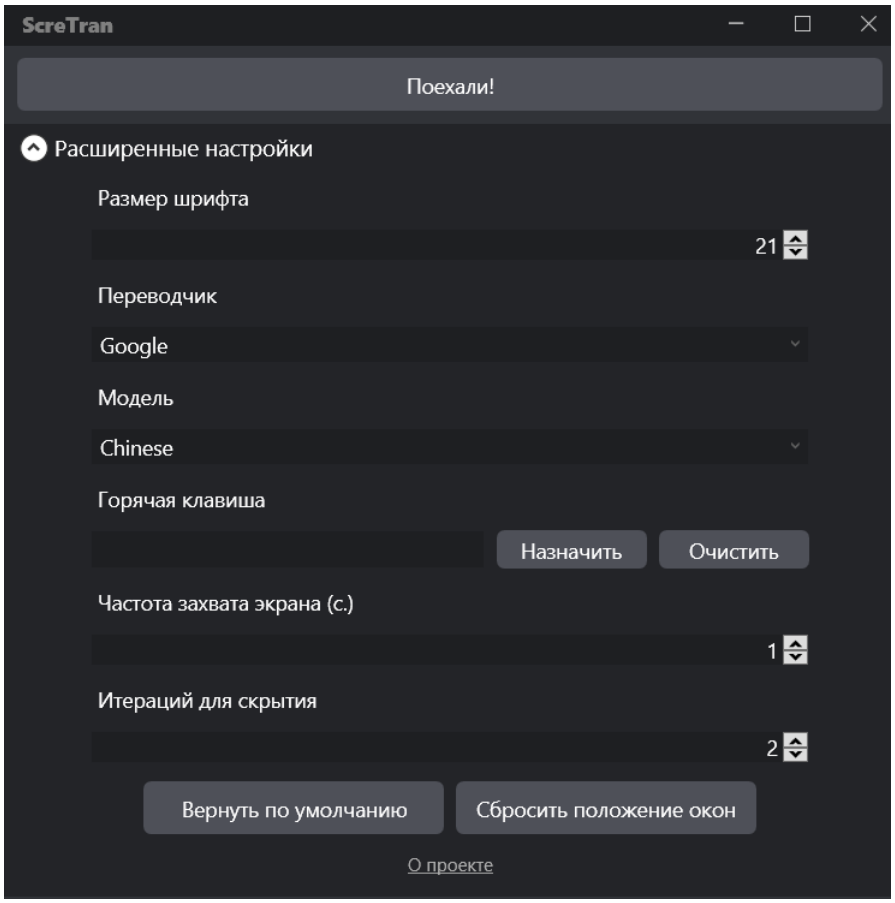


Рисунок 10 – Малое количество настроек для пользователя.

Еще одной проблемой приложения аналога является перевод всего на 4 языка, исходный язык – только английский. Это делает приложение очень узконаправленным, возможно переводить только англоязычный контент.

Наше приложение обладает большим арсеналом языков и переводит не только английский, но и другой иностранный контент, перевод с русского на другие языки также поддерживается, что позволяет охватывать гораздо больший контингент пользователей.

На рисунке 11 изображено работа ScreTran в игре.

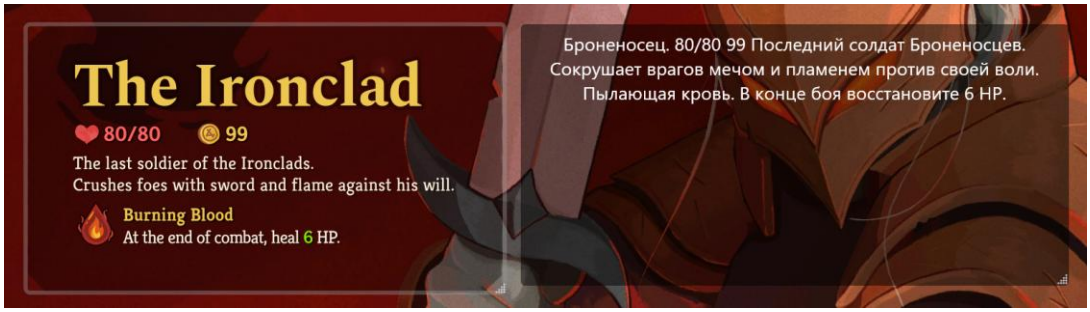


Рисунок 11 – Работа ScreTran в игре.

С данным приложением неудобно играть в игры, потому что область захвата текста находится поверх экрана, следовательно данная область недоступна для взаимодействия с игровым контентом.

Преимущество приложения аналога – скорость перевода, около 2-3 секунд. Наше – 5-10 секунд.

Следующий аналог – переводчик «Ёлочка».

Нестандартный формат приложения, полноценного оконного интерфейса не имеет и настраивается из панели «Скрытых значков»

На рисунке 12 изображено приложение «Ёлочка».

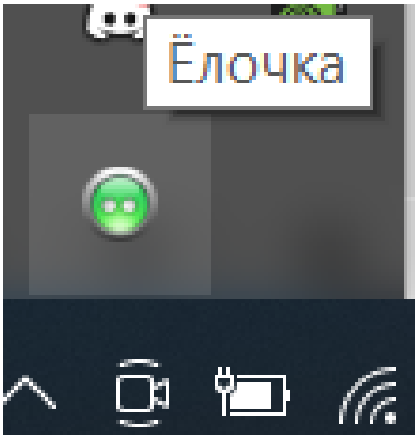


Рисунок 12 – Приложение «Ёлочка».

К преимуществам данного приложения можно отнести максимальную простоту в использовании, нужно всего лишь нажать на букву «Ё» и выделить область экрана.

При своей простоте приложение хорошо сохраняет формат переведенного текста, имеются отступы, перенос на новую строку, текст хорошо воспринимается.

На рисунке 13 изображено формат перевода приложения «Ёлочка».

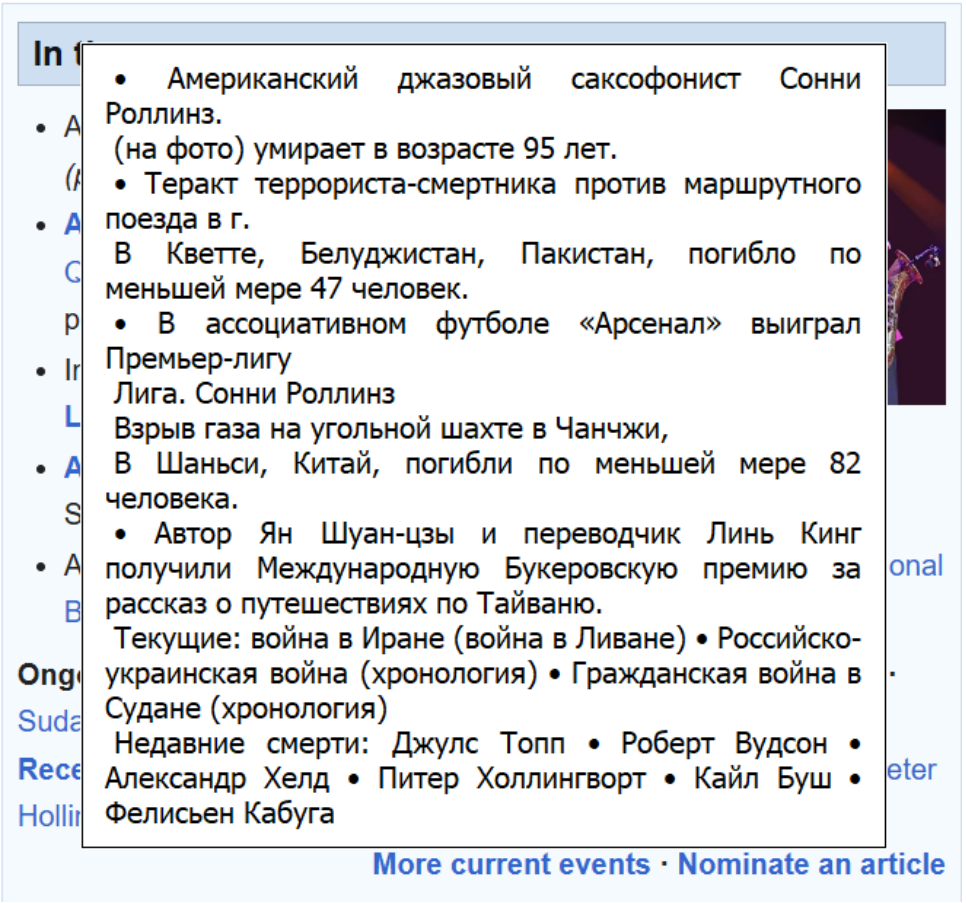


Рисунок 13 – Формат перевода приложения «Ёлочка».

К недостаткам приложение можно отнести малое количество настроек, возможность переводить только с английского на русский.

На рисунке 14 изображено малое количество настроек приложения аналога.

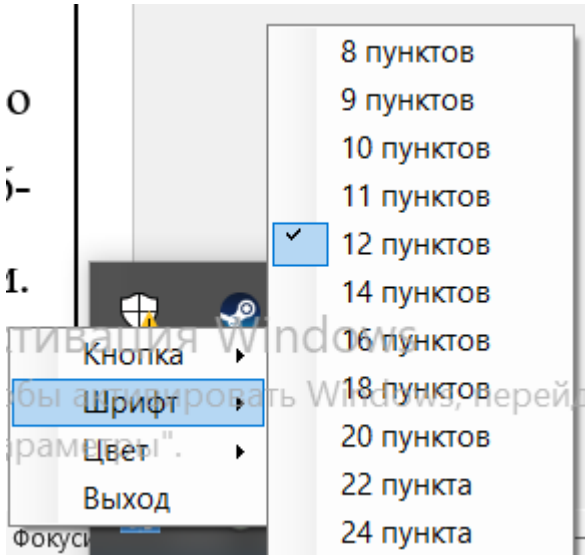


Рисунок 14 – Малое количество настроек приложения аналога.

Данное приложение подойдёт только русскоязычному сегменту, который переводит исключительно англоязычный контент.

Преимуществом данного приложения является высокая скорость перевода, почти моментальная – около 0,5 секунд. При этом переведенный текст не сплошной, а с сохранением формата оригинала, что делает приложение максимально простым для восприятия и использования.

На рисунке 15 изображено работа переводчика «Ёлочка» в игре.

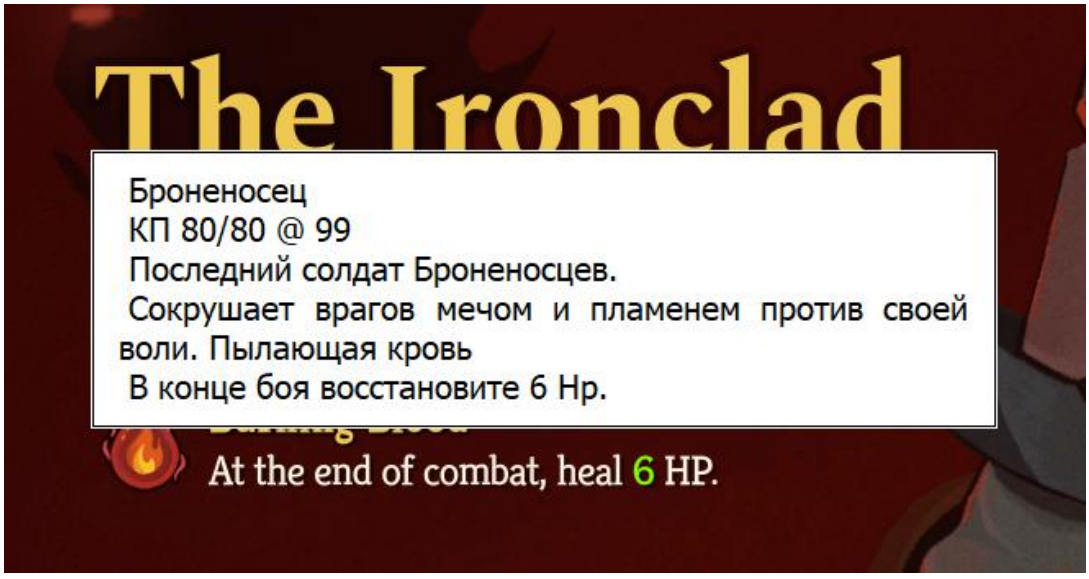


Рисунок 15 – Работа переводчика «Ёлочка» в игре.

Также приложение не переводит в реальном времени, а переводит текст на экране одноразово, выделяемая область экрана считывается один раз, а не постоянно в фоновом режиме.

Наше приложение уступает в скорости и простоте использования, но имеет гораздо больший функционал и ориентирован на более широкий сегмент пользователей, что делает наше приложение универсальным по сравнению с данным аналогом.

В таблице 1 приведены сведения по теме «анализ существующих аналогов».

Таблица 1 – Анализ существующих аналогов

Аналог	Основные возможности	Преимущества	Недостатки для данной задачи
Translumo	Перевод текста в оконном режиме	Скорость перевода, интерфейс на двух языках, поддержка разных моделей OCR	Программные ошибки в определенных кейсах, потребность в постоянном открытии/закрытии приложения, отсутствие overlay
ScreTran	Перевод текста в оконном режиме	Скорость перевода	Громоздкий интерфейс с тремя окнами, отсутствие overlay, малое количество настроек
«Ёлочка»	Одноразовый перевод текста при захвате экрана	Удобно для сайтов, пошаговых игр	Малое количество настроек, перевод только с английского на русский, отсутствие overlay
Разрабатываемое приложение	выбор области, overlay, окно перевода, горячие клавиши	ориентировано на быстрый пользовательский сценарий	требует дальнейшего развития и упаковки

1.1.3 Обоснование актуальности разработки программы

Проведенный анализ показывает, что пользовательская потребность остается актуальной: требуется инструмент, который переводит не абстрактный текстовый документ, а именно тот фрагмент экрана, который видит пользователь в текущий момент.

Существующие решения часто решают только часть задачи. Одни хорошо распознают текст, но требуют загрузки изображения. Другие переводят веб-страницы, но не работают с графическими интерфейсами сторонних программ. Третьи обладают нужной функциональностью, но перегружены настройками.

Разрабатываемая программа отличается тем, что объединяет несколько backend-функций: прием изображения через API, предварительную обработку кадра, OCR, перевод, кэширование результатов и возврат структурированного ответа клиентской части.

С точки зрения бэкенд-разработки актуальность состоит в необходимости создать надежный слой обработки, который будет понятен клиентской части через стабильный программный интерфейс. Клиентская часть должна передать изображение и параметры, а backend должен вернуть результат без ручной настройки внутренних модулей.

Отдельным фактором является возможность настройки параметров обработки. Для разных изображений могут требоваться разные языковые наборы OCR, порог уверенности, масштабирование и режим группировки строк.

Собственная разработка позволяет сосредоточиться на устойчивости обработки и границах backend-части, что соответствует теме выпускной квалификационной работы.

1.1.4 Постановка задачи

После анализа предметной области задача разработки была сформулирована как создание backend-сервиса, который принимает изображение выбранной области экрана, выполняет OCR и перевод, а затем

возвращает клиентской части понятный структурированный результат без ручного копирования текста в сторонние сервисы.

Разрабатываемое backend ПО относится к серверной части настольного приложения. Его особенность состоит в том, что оно работает не с обычным текстовым документом, а с изображением, полученным клиентской частью с экрана. Поэтому требования целесообразно разделить на две группы: функциональные возможности API и технические свойства обработки.

В работе не разрабатывается собственная модель OCR и не создается отдельная модель машинного перевода. Эти возможности используются как технологические модули. Основная задача заключается в проектировании backend-слоя, который корректно принимает данные, подготавливает изображение, вызывает OCR и переводчик, обрабатывает ошибки и возвращает результат клиентской части.

В таблице 2 приведены сведения по теме «функциональные требования к программе».

Таблица 2 – Функциональные требования к программе

Код	Функциональное требование к backend-сервису	Критерий выполнения
FR-01	Клиентская часть передает изображение области экрана.	Backend принимает файл или base64-строку, декодирует изображение и проверяет его размер.
FR-02	Клиентская часть передает исходный и целевой язык перевода.	Языковые значения преобразуются во внутренние коды и проверяются по списку поддерживаемых языков.
FR-03	Backend выполняет предварительную обработку изображения.	Кадр переводится в оттенки серого, улучшается контраст и при необходимости уменьшается размер.
Код	Функциональное требование к	Критерий выполнения

	backend-сервису	
FR-04	Backend запускает OCR для подготовленного изображения.	Сервис возвращает найденные строки, координаты блоков и среднюю уверенность распознавания.
FR-05	Backend переводит распознанные строки.	Каждая строка переводится на целевой язык, а итоговый текст формируется в порядке исходных блоков.
FR-06	Backend кэширует повторяющиеся результаты.	При повторном изображении или повторной строке используется ранее полученный результат.
FR-07	Backend возвращает структурированный ответ клиентской части.	Ответ содержит исходный текст, перевод, координаты, статусы, время обработки и сообщения об ошибках.
FR-08	Backend предоставляет служебные методы состояния.	Клиентская часть может проверить доступность сервиса и получить список поддерживаемых языков.

На рисунке 16 изображено диаграмма вариантов использования backend-сервиса.

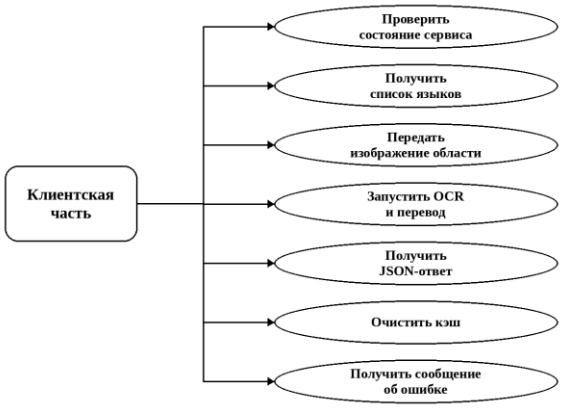


Рисунок 16 – Диаграмма вариантов использования backend-сервиса.

Нефункциональные требования сформулированы как проверяемые технические условия. Они задают свойства работы backend-сервиса: скорость ответа, устойчивость, ограничения размера изображения, кэширование и безопасность обработки данных.

В таблице 3 приведены сведения по теме «нефункциональные требования к программе».

Таблица 3 – Нефункциональные требования к программе

Код	Нефункциональное требование	Как проверить выполнение
NFR-01	Backend не должен блокировать клиентскую часть во время OCR и перевода.	Запрос обрабатывается в отдельном потоке или контролируемом обработчике, клиент получает ответ или ошибку без зависания интерфейса.
NFR-02	Изображение должно проходить проверку размера и формата.	При пустом, поврежденном или слишком большом изображении возвращается ошибка валидации.
NFR-03	Изображение перед OCR ограничивается по размеру.	Кадр масштабируется так, чтобы количество пикселей не превышало 2 200 000, а длинная сторона не превышала 1800 пикселей.
NFR-04	Повторная обработка одинакового изображения должна сокращаться.	При малом изменении сигнатуры кадра используется кэш результата, порог повторного использования составляет 0,7 процента.
NFR-05	API должен возвращать понятные ошибки.	Ошибки отсутствия изображения, неподдерживаемого языка и пустого OCR возвращаются в JSON-ответе.

Код	Нефункциональное требование	Как проверить выполнение
NFR-06	Сервис должен контролировать частоту тяжелых операций.	Повторные запросы используют кэш, а длительные операции не запускаются без проверки входных данных.
NFR-07	Клиентская часть должна получать состояние обработки.	Ответ содержит статус выполнения, время обработки и диагностические поля.
NFR-08	Backend не должен сохранять снимки экрана на диск в штатном сценарии.	Изображение обрабатывается в памяти и удаляется после формирования ответа.
NFR-09	После обработки должны очищаться временные данные.	Очищаются промежуточные изображения, служебные флаги и временные переменные запроса.
NFR-10	Ответ должен быть пригоден для разных режимов отображения.	Возвращаются и полный перевод, и список фрагментов с координатами для клиентского вывода поверх исходного фрагмента или отдельного окна результата.

Такая постановка задачи задает последовательную логику изложения: сначала фиксируются API-функции и скрытые технические ограничения, затем выбираются средства разработки, после этого проектируется архитектура и описывается реализация backend-классов.

1.2 Выбор технологий и средств разработки приложения

Выбор технологий выполнялся с учетом условий задачи: backend должен принимать изображения от клиентской части, выполнять предварительную обработку, запускать OCR и перевод, возвращать структурированный ответ и оставаться удобным для интеграции с настольным приложением.

Python позволяет быстро создавать прикладные backend-сервисы, использовать готовые библиотеки для обработки изображений, распознавания текста и перевода. Для дипломного проекта это важно, потому что основное внимание уделяется объединению этих модулей в единый конвейер обработки, а не низкоуровневой реализации каждого алгоритма.

В качестве веб-фреймворка используется FastAPI. Он позволяет описывать REST API, принимать запросы с изображениями и параметрами перевода, выполнять валидацию данных через Pydantic и возвращать ответы в формате JSON.

Для обработки изображений используется Pillow, для распознавания текста – EasyOCR, для перевода – deep-translator. Дополнительно применяются стандартные средства Python для потоковой обработки, кэширования и контроля ошибок.

Выбранный стек технологий соответствует backend-характеру проекта и позволяет реализовать все функции обработки, необходимые для совместной работы с клиентской частью.

1.2.1 Выбор языка программирования

Для реализации backend-сервиса рассматривались языки Python, C#, Java и C++. Каждый из них может применяться для серверной обработки данных, однако отличается сложностью разработки и доступностью библиотек OCR и машинного перевода.

C# удобен для разработки Windows-приложений и имеет развитые средства построения пользовательских решений, однако для задач OCR, предварительной обработки изображений и подключения готовых библиотек машинного распознавания он требует дополнительных оберток и более сложной настройки зависимостей. Python в этом отношении проще: большинство необходимых инструментов доступно как готовые пакеты и может быть подключено напрямую.

Java является кроссплатформенным языком, но интеграция с OCR-библиотеками и быстрый прототип backend API потребовали бы более

громоздкой структуры проекта. C++ обеспечивает высокую производительность, однако заметно усложняет работу с памятью, зависимостями и внешними библиотеками.

Python выбран как язык, который лучше всего подходит для объединения API, обработки изображений, OCR и перевода в одном прикладном backend-модуле. Для данной работы это важно, поскольку разработка сосредоточена на надежной обработке запроса: принять изображение, распознать текст, перевести строки и вернуть результат клиентской части.

Дополнительным преимуществом Python является читаемость исходного кода и высокая скорость внесения изменений. Это позволяет быстрее дорабатывать обработчики API, проверять сценарии интеграции и изменять внутреннюю логику без значительного усложнения проекта.

1.2.2 Выбор веб-фреймворка для backend API

FastAPI является современным фреймворком Python для разработки backend-сервисов. Его возможностей достаточно для реализации API-методов, приема изображений, описания схем запросов и ответов, автоматической валидации и формирования JSON-документации.

Главное преимущество FastAPI для данного проекта – удобство интеграции. Клиентская часть может обращаться к backend через HTTP-запросы, не зная внутренней структуры OCR и переводчика.

В проекте FastAPI используется для создания основных точек входа: проверки состояния сервиса, получения списка доступных языков и выполнения запроса на распознавание и перевод изображения.

Pydantic применяется для описания входных и выходных моделей. Благодаря этому backend явно фиксирует, какие поля требуются от клиентской части и какие данные будут возвращены после обработки.

Ограничением такого подхода является необходимость аккуратно контролировать размер передаваемых изображений и время выполнения

OCR. В рамках работы это решается валидацией, предварительной обработкой и кэшированием результатов.

1.2.3 Выбор средств обработки изображений

Для подготовки изображения используется библиотека Pillow. Она позволяет декодировать кадр, переводить его в оттенки серого, повышать контрастность, изменять размер и передавать результат в OCR-модуль.

Важным требованием является возможность обработки не всего экрана, а изображения конкретной области, которую передает клиентская часть. Такой подход снижает объем данных, поступающих в OCR, и ускоряет обработку.

Перед распознаванием изображение дополнительно обрабатывается: выполняется перевод в оттенки серого, автоконтраст и повышение контрастности. Эти операции относятся к backend-логике, потому что именно серверная часть подготавливает данные для OCR.

Дополнительно backend ограничивает размер изображения. Это защищает сервис от слишком тяжелых запросов и делает время обработки более предсказуемым.

Таким образом, средства обработки изображения выбраны с учетом требований реального времени и стабильного взаимодействия с клиентской частью.

1.2.4 Выбор средств распознавания и перевода

Для распознавания текста используется EasyOCR. Библиотека поддерживает несколько языков и позволяет получать не только текст, но и координаты распознанных блоков. Координаты необходимы клиентской части для размещения перевода около соответствующего фрагмента.

В программе предусмотрены языковые наборы для русского, английского, немецкого, польского, итальянского и французского языков. Для каждого выбранного языка подбирается соответствующий набор OCR-языков.

Для перевода используется deep-translator с обращением к GoogleTranslator. Для backend-части важно, что значения языков, полученные из API-запроса, преобразуются во внутренние языковые коды и передаются переводчику в корректном формате.

Так как OCR и перевод могут выполняться заметное время, backend должен поддерживать контролируемую обработку запроса, возвращать понятные ошибки и не выполнять лишние вычисления при повторной отправке одинакового изображения.

В рамках данной ВКР EasyOCR и переводчик рассматриваются как технологические модули, с которыми взаимодействует backend. Основной акцент остается на том, как сервис принимает данные, запускает обработку и возвращает результат клиентской части.

1.2.5 Выбор среды разработки

В качестве среды разработки может использоваться Visual Studio Code. Она удобна для работы с Python-файлами, поддерживает подсветку синтаксиса, встроенный терминал, расширения для анализа кода и запуск программы из рабочего каталога.

Для проекта, состоящего из одного основного Python-файла и внешних зависимостей, легкая среда разработки удобнее, чем тяжелая интегрированная IDE. Разработчик может быстро открывать файл, проверять изменения и запускать приложение.

Дополнительным преимуществом является наличие встроенного терминала, через который можно устанавливать зависимости, запускать скрипты и проверять синтаксис программы.

Среда разработки не является частью конечного продукта, но влияет на скорость реализации и отладки backend-сервиса. В дипломной работе она рассматривается как одно из средств разработки.

Использование Visual Studio Code соответствует характеру проекта: разработка выполняется на Python, а основная логика сосредоточена в API, обработке изображений, OCR и переводе.

1.2.6 Обмен данными между компонентами

В приложении используется обмен данными между клиентской частью и backend-сервисом. Клиент передает изображение выбранной области, исходный язык, целевой язык и параметры обработки, а сервер возвращает JSON-ответ с распознанным текстом, переводом, координатами фрагментов и служебной информацией.

Backend хранит настройки обработки, список поддерживаемых языков, кэш переводов, сигнатуры последних изображений и состояние выполняющихся задач. Эти данные не должны смешиваться с логикой отображения окон.

Изображение передается в запросе как файл или строка base64. Такой формат подходит для локальной интеграции с настольным клиентом и не привязывает backend к конкретному способу захвата экрана.

Результаты OCR передаются в виде распознанных текстовых блоков и координат. Клиентская часть может использовать эти координаты для вывода перевода поверх исходного фрагмента или для формирования отдельного текстового окна.

Для предотвращения повторной обработки одинакового изображения используется кэш и сигнатуры. Это помогает уменьшить лишние OCR-запуски и повышает стабильность ответа backend-сервиса.

В таблице 4 приведены сведения по теме «используемые технологии и средства разработки».

Таблица 4 – Используемые технологии и средства разработки

Средство	Назначение	Использование в проекте
Python	основной язык программы	связывает API, обработку изображения, OCR и перевод
FastAPI	backend API	методы состояния, списка языков и обработки изображения
Pydantic	валидация	модели запросов и ответов сервиса

	данных	
Средство	Назначение	Использование в проекте
Pillow	подготовка изображения	контраст, оттенки серого, масштабирование
EasyOCR	распознавание текста	получение текста, координат блоков и уверенности
deep-translator	машинный перевод	перевод распознанных строк
threading / asyncio	управление обработкой	разделение длительных операций и API-слоя
hashlib / cache	кэширование	сигнатуры изображений и повторное использование переводов

1.3 Проектирование архитектуры backend-сервиса

Проектирование начинается с общей архитектуры, потому что один endpoint сам по себе не решает задачу экранного перевода. Backend должен принять запрос от клиентской части, проверить данные, подготовить изображение, выполнить OCR, перевести текст и вернуть результат в согласованном формате.

В данной программе backend выделен как отдельный логический слой приложения. Он может запускаться локально на компьютере пользователя и взаимодействовать с клиентской частью через API, что упрощает разделение ответственности между разработкой интерфейса и разработкой обработки данных.

На рисунке 17 изображено архитектура программы с backend-сервисом.

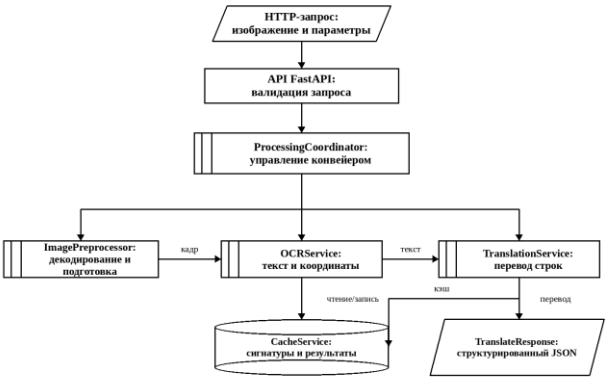


Рисунок 17 – Архитектура программы с backend-сервисом.

1.3.1 Общая архитектура серверной части

Архитектуру backend можно разделить на слой API, слой координации, слой обработки изображения, слой OCR, слой перевода и служебный слой кэширования. Слой API принимает запросы и возвращает ответы. Координатор определяет порядок вызова внутренних модулей.

Внутренний backend включает декодирование изображения, предобработку, распознавание текста, группировку найденных фрагментов, перевод строк, кэширование результатов и очистку временных данных. Этот слой скрыт от пользователя, но именно он выполняет основную техническую работу.

Backend возвращает клиентской части не абстрактный статус выполнения, а конкретный результат: исходный текст, перевод, координаты блоков, среднюю уверенность OCR, время обработки и сообщения об ошибках. Поэтому граница между слоями проходит по ответственности компонентов.

В таблице 5 приведены сведения по теме «состав архитектурных слоев программы».

Таблица 5 – Состав архитектурных слоев программы

Слой	Состав	Назначение
API	FastAPI endpoints, Pydantic-схемы	принимает запросы и возвращает JSON-ответы
Координатор	ProcessingCoordinator, параметры, состояние	связывает API и внутренние сервисы обработки
Backend обработки	ImagePreprocessor, OCRService, TranslationService	подготавливает кадр, распознает и переводит текст
Служебная логика	кэш, пороги, сигнатуры, очистка памяти	снижает лишние OCR-запуски и поддерживает устойчивость
Слой ответа	TranslateResponse, FragmentResult	возвращает данные клиентской части для

		отображения

1.3.2 Граница ответственности бэкенд-разработки

Зона бэкенд-разработки определяется обработкой данных и API. К этой части относятся прием изображения, проверка параметров, декодирование кадра, предобработка, запуск OCR, группировка фрагментов, перевод, кэширование, формирование ответа и обработка ошибок.

К клиентской части относятся экранные окна, выбор области, управляющие клавиши, визуальное отображение результата и настройки внешнего вида. Backend не реализует эти элементы, но предоставляет данные, необходимые для их работы: текст, перевод, координаты и служебные статусы.

Такое разграничение важно для корректного описания результата. Разработанная часть не заменяет EasyOCR или переводчик и не занимается интерфейсом, а делает обработку доступной через стабильный backend-сервис.

1.3.3 Проектирование API взаимодействия с клиентской частью

API строится вокруг нескольких запросов. Клиентская часть может проверить состояние backend, получить список поддерживаемых языков и отправить изображение для распознавания и перевода. Основной запрос содержит изображение, исходный язык, целевой язык и параметры обработки.

Такое разделение сделано для того, чтобы клиентское приложение не зависело от внутренней структуры OCR и переводчика. Если в будущем будет заменена библиотека распознавания или сервис перевода, контракт API может остаться прежним.

Диаграмма компонентов показывает, какие backend-модули участвуют в обработке запроса и как они связаны с координатором сервиса.

На рисунке 18 изображено компонентная структура backend-части.

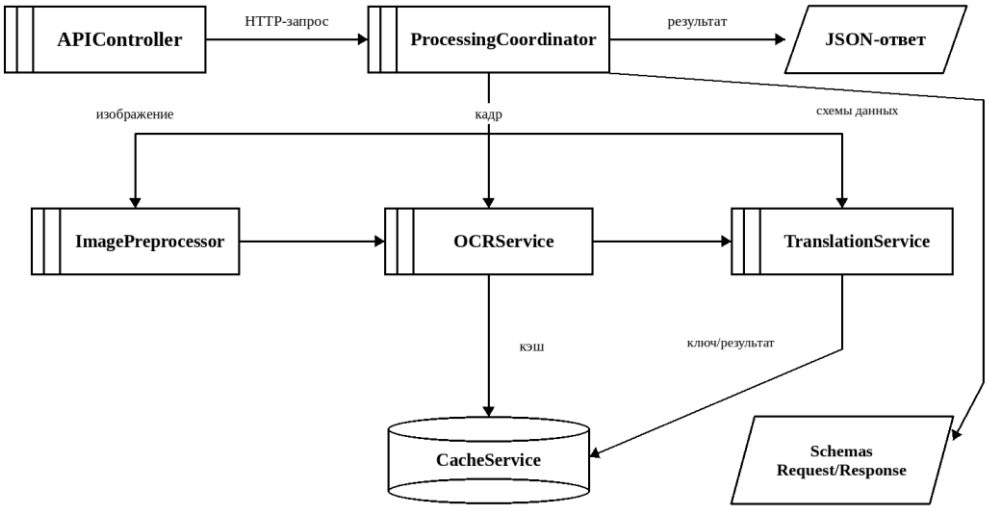


Рисунок 18 – Компонентная структура backend-части.

1.3.4 Логические слои обработки данных

Для анализа backend можно представить как набор логических слоев. Первый слой отвечает за прием и валидацию запроса. Второй слой декодирует и подготавливает изображение. Третий слой выполняет OCR. Четвертый слой переводит найденные строки. Пятый слой формирует ответ и сохраняет результат в кэше.

На рисунке 19 изображены логические слои backend-сервиса.

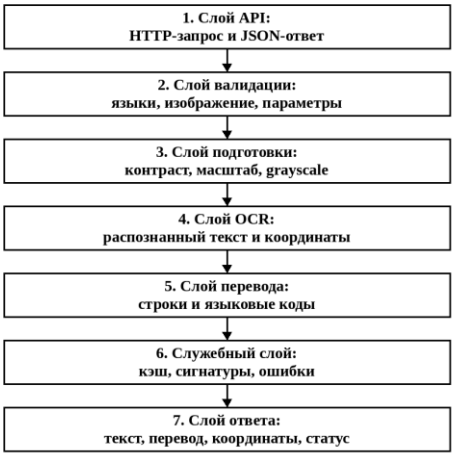


Рисунок 19 – Логические слои backend-сервиса.

Такое представление помогает объяснить, почему в программе используется отдельный класс-координатор. Запрос проходит через несколько этапов, и backend должен сохранять параметры языка, ограничения размера изображения, кэш переводов и данные последнего результата.

1.4 Разработка backend-сервиса

Разработка выполнялась после определения требований и архитектуры. Основная задача заключалась в том, чтобы реализовать не разрозненные функции OCR и перевода, а единый backend-конвейер: прием запроса, подготовка изображения, распознавание, перевод, кэширование и возврат ответа.

В коде используются отдельные классы для обработки изображения, OCR, перевода, кэша и координации запроса. Такой вариант проще сопровождать и тестировать, потому что назначение каждого класса соответствует конкретному этапу backend-сценария.

На рисунке 20 изображено диаграмма деятельности основного backend-сценария.

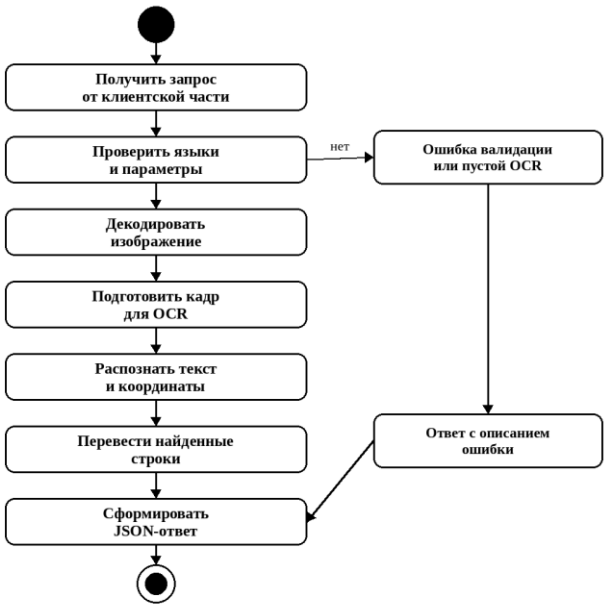


Рисунок 20 – Диаграмма деятельности основного backend-сценария.

1.4.1 Реализация API и координатора обработки

API является точкой входа в backend-сервис. Через него клиентская часть передает изображение, исходный язык, целевой язык и параметры обработки. Внутри программы запрос обрабатывается координатором, который последовательно вызывает предобработку, OCR, перевод и формирование ответа.

Координатор хранит служебное состояние backend: список поддерживаемых языков, кэш переводов, параметры ограничения изображения, порог уверенности OCR и статистику обработки. Эти данные используются только внутри сервиса и не относятся к визуальному интерфейсу.

При поступлении запроса backend проверяет корректность языков, наличие изображения и допустимый размер данных. После этого запускается конвейер обработки, а результат возвращается клиентской части в виде JSON-ответа.

```
@app.post("/api/translate", response_model=TranslateResponse)
async def translate(request: TranslateRequest):
    if request.source_lang not in TRANSLATOR_LANGUAGES.values():
        raise HTTPException(status_code=400,
detail="Неподдерживаемый исходный язык")
    if request.target_lang not in TRANSLATOR_LANGUAGES.values():
        raise HTTPException(status_code=400,
detail="Неподдерживаемый язык перевода")
    try:
        return coordinator.process_request(request)
    except ImageDecodeError as error:
        raise HTTPException(status_code=400, detail=str(error))
    except OcrEmptyResult:
        raise HTTPException(status_code=422, detail="Текст на
изображении не найден")
    except Exception as error:
```

```
        raise HTTPException(status_code=500, detail=f"Ошибка
backend-обработки: {error}")
```

1.4.2 Описание основных backend-классов с примерами реализации

Класс APIController отвечает за внешние точки входа backend-сервиса. Он принимает HTTP-запросы, передает данные координатору и возвращает клиентской части готовый JSON-ответ.

Класс ProcessingCoordinator реализует основной сценарий обработки. Он получает проверенный запрос, обращается к предобработчику изображения, OCR-сервису, переводчику и кэшу, а затем собирает итоговый ответ.

Класс ImagePreprocessor выполняет декодирование изображения, проверку размера, перевод в оттенки серого, повышение контраста и масштабирование перед передачей в OCR.

Класс OCRService отвечает за инициализацию EasyOCR, запуск распознавания и группировку найденных фрагментов текста.

Класс TranslationService переводит строки и использует кэш, чтобы не отправлять повторяющиеся фрагменты переводчику несколько раз.

```
class TranslateRequest(BaseModel):
    image_base64: str
    source_lang: str
    target_lang: str
    min_confidence: int = 20
    ocr_scale: int = 1


class FragmentResult(BaseModel):
    text: str
    translation: str
    x: int
    y: int
    width: int
    height: int
    confidence: float
```

```
class TranslateResponse(BaseModel):
    source_text: str
    translated_text: str
    fragments: list[FragmentResult]
    elapsed_ms: int
    cached: bool = False
```

В таблице 6 приведены сведения по теме «основные классы приложения».

Таблица 6 – Основные классы приложения

Класс	Назначение	Основные методы
APIController	прием запросов backend API и возврат ответов	health, languages, translate
ProcessingCoordinator	координация этапов обработки запроса	process_request, build_response
ImagePreprocessor	декодирование и подготовка изображения	decode_image, preprocess, image_signature
OCRService	распознавание текста и координат	get_reader, run_easyocr, group_fragments
TranslationService	перевод строк и работа с кэшем	translate_lines, get_cached_translation

1.4.3 Реализация предварительной обработки изображения

Предварительная обработка реализована как отдельный этап backend-конвейера. Клиентская часть передает изображение выбранной области экрана, а сервер декодирует его, проверяет размер и приводит к формату, удобному для OCR.

Если изображение слишком маленькое или повреждено, backend возвращает ошибку валидации и не запускает распознавание. Это предотвращает лишнюю нагрузку на OCR и делает поведение сервиса предсказуемым.

После успешной проверки изображение переводится в оттенки серого, улучшается контрастность и при необходимости уменьшается размер.

Благодаря этому OCR получает более стабильный входной кадр независимо от особенностей клиентского интерфейса.

1.4.4 Реализация OCR-обработки

После запуска обработки backend получает подготовленное изображение, формирует его сигнатуру, проверяет возможность повторного использования результата и при необходимости запускает EasyOCR. Такой подход снижает количество лишних операций при неизменном тексте.

Перед передачей изображения в EasyOCR выполняется предварительная обработка. Изображение переводится в оттенки серого, улучшается контраст и ограничивается размер. Это снижает нагрузку на OCR и делает поведение программы более предсказуемым.

Распознанные фрагменты очищаются от лишних пробелов и группируются в строки. Для каждой строки сохраняются координаты, исходный текст и уровень уверенности, после чего данные передаются модулю перевода.

```
def run_easyocr(self, image: Image.Image, source_name: str,
scale: int, min_conf: int):
    reader = self.get_reader(source_name)
    processed, actual_scale = self.preprocessor.prepare(image,
scale=scale)
    raw_results = reader.readtext(
        np.array(processed.convert("RGB")),
        detail=1,
        paragraph=EASYOCR_PARAGRAPH_MODE,
        decoder="greedy",
        batch_size=1,
        workers=0,
    )
    fragments = []
    for box, text, confidence in raw_results:
        text = clean_text(str(text))
        if text and confidence * 100 >= min_conf:
```

```
            fragments.append(FragmentResult(box=box, text=text,
confidence=confidence * 100))
    return self.group_fragments_into_lines(fragments), fragments
```

1.4.5 Реализация перевода и кэширования

Модуль перевода получает список распознанных строк и языковые коды. Каждая строка передается переводчику, а результат сохраняется в кэше по ключу, который включает исходный текст, исходный язык и целевой язык.

Кэширование особенно важно при работе с динамическими окнами, где часть текста может не меняться между запросами. Если backend уже переводил такую строку, он возвращает готовый результат без повторного обращения к переводчику.

После перевода координатор объединяет исходные строки, переведенные строки и координаты OCR в единый ответ. Клиентская часть может использовать эти данные для вывода поверх исходного фрагмента или отдельного окна результата.

1.4.6 Реализация формата ответа и обработки ошибок

Ответ backend-сервиса содержит полный распознанный текст, полный перевод, список фрагментов с координатами, среднюю уверенность OCR, время обработки и флаг использования кэша. Такой формат удобен для клиентской части и для последующего тестирования.

Ошибки также возвращаются в структурированном виде. Если изображение отсутствует, язык не поддерживается или OCR не найден, клиентская часть получает понятное сообщение и может показать его пользователю.

Для устойчивости используется обработка исключений на уровне API и внутренних сервисов. Это снижает риск аварийного завершения backend и упрощает отладку интеграции.

В таблице 7 приведены сведения по теме «основные API-методы backend-сервиса».

Таблица 7 – Основные API-методы backend-сервиса

Метод	Назначение
GET /api/health	проверка доступности backend-сервиса
Метод	Назначение
GET /api/languages	получение списка поддерживаемых языков
POST /api/translate	прием изображения, запуск OCR и перевода
POST /api/ocr	распознавание текста без перевода для отладки
POST /api/translate/text	перевод готового текста без OCR
DELETE /api/cache	очистка кэша переводов и сигнатур
GET /api/stats	получение служебной статистики обработки

1.4.7 Руководство по интеграции с клиентской частью

Для начала работы клиентская часть запускает backend-сервис и проверяет его доступность через метод состояния. После этого можно получить список поддерживаемых языков и сформировать запрос на обработку изображения.

Клиентское приложение выделяет область экрана и передает backend изображение этой области. В запрос также добавляются исходный язык, целевой язык, порог уверенности OCR и дополнительные параметры обработки.

После получения ответа клиентская часть отображает перевод выбранным способом: поверх исходного фрагмента или в отдельном окне. Backend при этом не управляет окнами, а только возвращает данные, необходимые для отображения.

При ошибке клиентская часть должна использовать поле сообщения из ответа backend. Такой подход позволяет отделить обработку данных от пользовательского интерфейса и упростить дальнейшее развитие продукта.

В таблице 8 приведены сведения по теме «порядок взаимодействия клиентской части с backend-сервисом».

Таблица 8 – Порядок взаимодействия клиентской части с backend-сервисом

Шаг	Действие клиентской части	Результат backend-сервиса
1	Проверить /api/health	сервис сообщает, что готов принимать запросы
2	Получить список языков	клиент получает допустимые языковые коды
3	Передать изображение области экрана	backend декодирует и проверяет изображение
4	Запустить /api/translate	выполняются OCR, перевод и кэширование
5	Получить JSON-ответ	клиент получает текст, перевод, координаты и статус
6	Обработать ошибку или повторить запрос	backend возвращает понятное сообщение или кэшированный результат

1.4.8 Тестирование и отладка

Тестирование выполнялось по основным backend-сценариям. Проверялись запуск сервиса, доступность API, валидация языков, обработка корректного изображения, реакция на поврежденные данные, OCR, перевод, кэширование и формат ответа.

Отдельно проверялась устойчивость при повторных запросах. Клиентская часть может несколько раз отправлять похожие изображения, менять языки и запускать обработку заново. В этих ситуациях backend не должен возвращать устаревший результат и не должен выполнять лишнюю тяжелую обработку.

Проверялись ошибки обработки: отсутствие изображения, неподдерживаемый язык, пустой результат OCR и исключения переводчика.

В каждом случае сервис должен вернуть понятное сообщение без аварийного завершения.

Результаты проверки показывают, что основные backend-сценарии выполняются корректно. Ограничения текущей версии связаны с зависимостью от OCR-библиотеки, скорости перевода и дальнейшей оптимизацией очереди обработки.

В таблице 9 приведены сведения по теме «проверка основных backend-сценариев».

Таблица 9 – Проверка основных backend-сценариев

Проверка	Ожидаемый результат	Итог
Запуск backend-сервиса	метод /api/health возвращает статус готовности	Выполнено
Проверка языков	неподдерживаемые языки отклоняются с понятной ошибкой	Выполнено
Обработка изображения	корректный кадр проходит предобработку и OCR	Выполнено
Поврежденные данные	backend возвращает ошибку валидации	Выполнено
Перевод строк	распознанный текст переводится на целевой язык	Выполнено
Кэширование	повторный запрос использует сохраненный результат	Выполнено
Формат ответа	JSON содержит текст, перевод, координаты и служебные поля	Выполнено
Повторный запуск	новый запрос не ломает состояние сервиса	Выполнено

1.4.9 Последовательность взаимодействия компонентов

Последовательность взаимодействия показывает, как запрос проходит через backend. Сначала клиентская часть формирует изображение и параметры перевода. Затем API принимает запрос, координатор обращается к модулю предобработки, передает изображение в OCR, отправляет найденные строки переводчику и формирует итоговый JSON-ответ.

Важный момент состоит в том, что сценарий не должен завершаться на внутреннем компоненте. Для клиентской части результатом является не факт вызова OCR или переводчика, а готовый набор данных, который можно отобразить пользователю.

Если во время обработки возникает ошибка, координатор не завершает приложение аварийно, а формирует структурированный ответ с описанием проблемы. Такой подход делает поведение backend понятным для клиентской части.

На рисунке 21 изображено диаграмма последовательности основного backend-сценария.

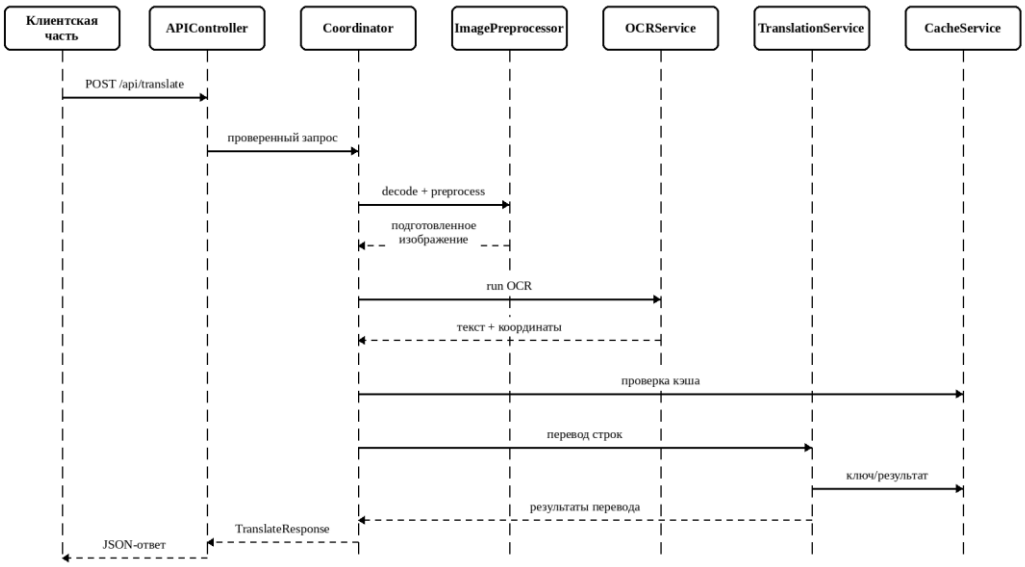


Рисунок 21 – Диаграмма последовательности основного backend-сценария.

1.4.10 Анализ полученных результатов и развитие

В результате разработки была создана backend-часть для экранного перевода. Она закрывает ключевую техническую проблему предметной области: изображение с текстом можно отправить на обработку и получить

распознанный текст, перевод и координаты фрагментов без ручного использования сторонних OCR-сервисов.

К сильным сторонам реализации можно отнести наличие единого API, предварительной обработки изображения, OCR с координатами, перевода строк, кэширования, проверки входных данных и структурированного формата ответа. Эти элементы относятся именно к backend-части и обеспечивают основу для работы клиентского приложения.

Ограничением текущей версии остается зависимость от внешних библиотек распознавания и перевода. При сложном фоне, низком качестве текста или отсутствии доступа к переводчику результат может быть хуже. Для дальнейшего развития целесообразно добавить очередь задач, историю запросов, расширенное логирование, поддержку нескольких OCR-моделей и упаковку backend-сервиса вместе с клиентским приложением.

Поставленная цель достигнута: разработана backend-часть приложения, которая принимает данные от клиентской части, выполняет обработку изображения, OCR и перевод, а затем возвращает результат в удобном для отображения формате.

2 Стартап-проект Screen Translate

2.1 Обоснование актуальности

В современном мире информации пользователи часто сталкиваются с информацией на иностранном языке. Такой контент присутствует в интерфейсах программ, веб-сервисах, технической документации, учебных материалах, презентациях, PDF-файлах, видеокурсах, компьютерных играх, субтитрах и изображениях. Значительная часть этой информации отображается на экране как изображение или как элемент динамического интерфейса, поэтому ее нельзя быстро выделить мышью, скопировать и вставить в обычный переводчик.

Из-за этого у пользователя возникает повторяющаяся проблема: текст виден на экране, но технически недоступен для быстрого перевода. Чтобы понять фрагмент, приходится делать скриншот, открывать отдельный OCR-сервис, распознавать текст, копировать результат, переходить в переводчик и только после этого возвращаться к исходному окну. Такой путь занимает лишнее время, разрушает рабочую ситуацию и делает перевод неудобным в тех случаях, когда информация нужна сразу.

Актуальность проекта усиливается глобализацией цифрового контента. По данным исследования DeepL, менее 20% мирового населения свободно владеет английским языком, хотя английский часто используется как универсальный язык бизнеса, образования и программного обеспечения. Это означает, что большая часть пользователей нуждается в переводе при работе с иностранными интерфейсами, учебными материалами и профессиональной документацией.

Коммерческая значимость локализации подтверждается исследованием CSA Research: 76% онлайн-покупателей предпочитают получать информацию о товарах на родном языке, а 40% не готовы совершать покупку на сайте, если информация представлена на чужом языке. Для программных продуктов и цифровых сервисов это означает, что язык интерфейса влияет не только на удобство, но и на готовность пользователя работать с продуктом.

Для рынка программного обеспечения языковой барьер особенно заметен. Многие профессиональные инструменты, обучающие платформы, технические каталоги, CRM-системы, библиотеки и документация создаются на английском языке. Даже если пользователь в целом понимает английский, постоянное переключение между окнами ради перевода небольших фрагментов снижает скорость работы и увеличивает когнитивную нагрузку.

Развитие искусственного интеллекта, OCR-технологий и нейросетевого машинного перевода делает возможным создание более удобного инструмента. Пользователь уже привык к автоматическому переводу, но большинство доступных решений предполагает, что текст уже находится в копируемом виде. В проекте Screen Translate решается более широкий сценарий: сначала программа получает текст из изображения экрана, а затем переводит его.

Проект Screen Translate направлен на создание программного обеспечения для захвата, распознавания и перевода текстового материала в реальном времени. Пользователь выбирает область экрана, программа анализирует изображение, распознает текст, переводит его на выбранный язык и показывает результат в отдельном окне или поверх исходного фрагмента. За счет этого перевод становится частью рабочего процесса, а не отдельной последовательностью ручных операций.

Таким образом, актуальность стартап-проекта определяется ростом объема иностранного цифрового контента, регулярной потребностью пользователей в переводе, недостаточностью обычных переводчиков для экранного текста и возможностью объединить OCR и машинный перевод в один удобный продукт.

В рамках данной работы особое значение имеет фронтенд-разработка. Именно через интерфейс пользователь воспринимает ценность продукта: насколько быстро можно выбрать область, насколько понятно отображается статус, насколько читаем overlay и не мешает ли программа исходному окну.

2.2 Описание и обоснование выбора ВКР в формате ВКРС

Выбор формата «Стартап как диплом» для данного проекта является обоснованным, поскольку работа имеет выраженный продуктовый и коммерческий характер. Она не ограничивается теоретическим исследованием технологии распознавания или перевода, а направлена на создание прикладного программного продукта с понятной целевой аудиторией и перспективой коммерциализации.

Проект относится к технологическим стартапам в сфере информационных и коммуникационных технологий. Он основывается на программной разработке, использовании библиотек захвата экрана, OCR-распознавания, машинного перевода, обработки изображений, кэширования и пользовательского взаимодействия. В результате формируется продукт, который может применяться в учебе, профессиональной деятельности, работе с иностранным программным обеспечением и повседневном использовании.

Формат ВКРС позволяет рассмотреть проект не только как техническую разработку, но и как бизнес-проект. Для стартап-части важно показать существующую проблему, предлагаемое решение, целевой рынок, конкурентов, бизнес-модель, финансовые предпосылки, команду, развитие и риски. Screen Translate соответствует этой логике, поскольку решает конкретную боль пользователей и имеет несколько возможных моделей монетизации.

Выбор ВКРС также позволяет разделить общий проект на зоны ответственности. Продукт включает клиентскую часть, отвечающую за пользовательский сценарий и отображение результата, и вычислительную часть, отвечающую за захват, OCR, перевод, кэширование, производительность и стабильность. Такое разделение делает проект удобным для командной разработки и позволяет каждому участнику показать собственный вклад.

В отличие от обычной дипломной работы, формат стартапа требует не только описать алгоритмы, но и показать, зачем продукт нужен рынку, кто

будет его использовать, какие функции являются коммерчески значимыми и какие риски необходимо контролировать. Это делает работу более приближенной к реальному процессу создания программного продукта.

Таким образом, выбор формата ВКРС является целесообразным, потому что проект Screen Translate имеет практическую значимость, рыночный потенциал, рабочую техническую реализацию и понятную траекторию дальнейшего развития.

В рамках темы «Фронтенд-разработка» основной акцент сделан на клиентской части продукта. Для данного проекта фронтенд - это не только внешний вид главного окна, но и весь пользовательский путь: выбор области, настройка языков, запуск и остановка перевода, визуальный статус, overlay-отображение, работа горячих клавиш и понятная реакция программы на ошибки.

Фронтенд-часть особенно важна для стартапа, потому что именно она определяет, будет ли пользователь воспринимать продукт как быстрый и удобный. Если интерфейс перегружен, перевод может появиться где угодно или overlay плохо читается, даже качественный OCR и перевод не превращаются в полезный продукт.

2.3 Описание существующей проблемы на рынке

Основная проблема рынка заключается в том, что существующие инструменты перевода плохо работают в ситуациях, когда текст нельзя быстро скопировать. Пользователь может видеть иностранную надпись в программе, игре, видео, PDF-документе, презентации или изображении, но обычный переводчик не может напрямую обработать этот текст без дополнительного этапа распознавания.

На практике пользователь вынужден выполнять длинную последовательность действий: сделать скриншот, сохранить или передать его в OCR-сервис, распознать текст, проверить ошибки распознавания, скопировать результат, открыть переводчик, вставить текст, получить перевод и вернуться обратно. Даже если каждый шаг кажется простым,

суммарно процесс занимает от одной до двух минут и отвлекает от основной задачи.

Проведенные интервью среди студентов университета показывают наличие у пользователей проблемы. Из 30 опрошенных респондентов 18 сталкиваются с текстом без возможности копирования не менее трех раз в неделю. 16 из 30 предпочли бы видеть перевод поверх исходного места, 20 из 30 отметили важность обработки только выбранной области экрана, а 14 из 30 готовы рассматривать платную подписку при условии стабильного OCR и удобного сценария использования.

Проблема наиболее заметна в образовательном сценарии. Студент или школьник смотрит иностранный видеокурс, читает презентацию, научную статью или PDF-файл. Часто важная информация находится не в копируемом тексте, а на слайде, в видеокадре или в защищенном документе. Ручной перевод фрагментов мешает обучению, заставляет ставить видео на паузу и переключаться между приложениями.

В профессиональной сфере проблема проявляется у IT-специалистов, инженеров, сотрудников компаний и исследователей. Иностранные интерфейсы, сообщения об ошибках, технические инструкции, API-документация, CRM-системы, каталоги и веб-сервисы часто требуют понимания прямо в момент работы. Пользователю нужен быстрый перевод, который не разрушает рабочий контекст.

В бытовом и развлекательном сценарии проблема возникает в играх без локализации, видео с субтитрами, изображениях, скриншотах, уведомлениях и меню программ. В таких случаях пользователь чаще всего не хочет заниматься полноценной локализацией, ему нужно просто быстро понять конкретный фрагмент текста.

Существующие онлайн-переводчики решают задачу перевода, но не решают задачу получения текста из экрана. Браузерные переводчики ограничены веб-страницами и не работают в большинстве десктопных приложений. OCR-утилиты извлекают текст, но часто не дают мгновенного

перевода и не возвращают результат в исходный контекст. Поэтому на рынке остается ниша для инструмента screen translation.

Проблема формулируется как языковой барьер любого видимого текста. Пользователю нужен единый сценарий: выбрать область экрана, получить распознавание и перевод, увидеть результат там, где он был нужен, и продолжить работу без лишних переключений.

2.4 Описание решения проблемы и продуктового решения

2.4.1 Наименование продукта

Наименование продукта - Screen Translate. Это программное обеспечение для захвата, распознавания и перевода текстового контента в реальном времени. В рамках данной выпускной квалификационной работы рассматривается направление «Фронтенд-разработка».

2.4.2 Назначение и сфера применения

Продукт предназначен для пользователей, которым необходимо быстро переводить текст, видимый на экране. Основная особенность заключается в том, что программа работает не только с копируемым текстом, но и с изображениями, интерфейсами программ, PDF, видео, презентациями и играми.

Сфера применения включает учебные материалы, веб-сайты, PDF-документы, презентации, зарубежное программное обеспечение, техническую документацию, игры, видео и изображения. На первом этапе продукт ориентирован на Windows-сценарий, потому что именно в среде настольных приложений часто требуется перевод произвольной области экрана.

2.4.3 Пользовательский сценарий

Базовый пользовательский сценарий состоит из нескольких последовательных действий. Сначала пользователь открывает иностранный интерфейс, видеоурок, PDF-файл, игру или другое окно с текстом. Затем он

нажимает горячую клавишу или кнопку выбора области и выделяет прямоугольник на экране.

После этого система распознает текст, переводит его на выбранный язык и показывает результат. В обычном режиме перевод выводится в отдельном окне. В overlay-режиме перевод отображается поверх исходного текста или рядом с ним, сохраняя визуальный контекст. Пользователь может остановить перевод, перезапустить его, скрыть или показать рамку области, изменить режим вывода и скопировать результат.

2.4.4 Фронтенд-зона ответственности

Фронтенд-разработка в проекте включает несколько ключевых блоков. Первый блок - интерфейс выбора области. Он отвечает за полупрозрачное окно выделения, рамку выбора, обработку мыши, отмену по ESC, проверку минимального размера области и сохранение координат.

Второй блок - главное окно приложения. Оно содержит основные элементы управления: выбор режима вывода, выбор языков в настройках, кнопку выбора области, старт, стоп, включение рамки и переход к настройкам. Важная задача интерфейса - не перегружать пользователя техническими параметрами, оставляя на первом экране только основные действия.

Третий блок - окно результата в обычном режиме. Оно должно быть поверх остальных окон, позволять выделять и копировать текст, менять размер и положение, настраивать шрифт, цвет фона, цвет текста и прозрачность. При этом окно не должно выглядеть как обычный редактор: ручное изменение текста блокируется, но копирование остается доступным.

Четвертый блок - overlay результата. Это наиболее важная часть фронтенда, потому что она отличает продукт от обычной связки OCR и переводчика. Overlay должен быть читаемым, не мешать исходному контенту, учитывать цвет фона и текста, подбирать размер шрифта, избегать наложений и сохранять ощущение, что перевод встроен в исходное окно.

Пятый блок - визуальная обратная связь. Пользователь должен понимать, что программа делает в данный момент: ожидает выбора области, выполняет перевод, показывает результат, остановлена или очищает память. Для этого используются статусы, индикатор загрузки, надпись «Перевод...» и состояния кнопок.

2.4.5 Функциональные возможности реализованной программы

На текущем этапе разработана рабочая версия приложения Screen Translator на языке Python. Интерфейс реализован с использованием Tkinter. В коде присутствуют отдельные окна и классы для выбора области, рамки области, overlay-отображения и окна перевода. Это позволяет разделить пользовательские задачи и поддерживать более понятную структуру приложения.

Реализованы два режима вывода. Первый режим - отдельное окно перевода, которое отображается поверх остальных окон, может перемещаться пользователем, содержит настройки шрифта, цвета и прозрачности. Второй режим - overlay поверх выбранной области, где перевод отображается прямо в контексте исходного текста.

Программа поддерживает горячие клавиши: Print Screen используется для режима скриншота, F6 - для выбора области, F7 - для перезапуска перевода, F8 - для остановки и очистки, F9 - для включения или скрытия рамки, клавиши 1 и 2 - для переключения обычного и overlay-режима. Для защиты от случайных повторных нажатий реализован cooldown горячих клавиш.

В overlay-режиме фронтенд получает от OCR координаты фрагментов, оценивает высоту и ширину текстовых блоков, подбирает шрифт, перенос строк, цвет текста и цвет подложки. Для лучшей читаемости используются функции оценки цветов исходного изображения, проверки контраста и уменьшения шрифта при возможных пересечениях.

Особенность фронтенда состоит в том, что overlay остается видимым для пользователя, но может быть исключен из захвата экрана Windows, чтобы программа не распознавала собственный перевод повторно. При необходимости пользователь может временно разрешить попадание overlay в скриншот.

2.4.6 Конкурентоспособность и инновационность

Конкурентоспособность продукта определяется тем, что он объединяет несколько функций в одном пользовательском сценарии. Обычный онлайн-переводчик требует копируемого текста. OCR-утилита извлекает текст, но не всегда переводит его и не возвращает результат в исходный контекст. Screen Translate объединяет выбор области, распознавание, перевод и отображение результата.

Инновационность фронтенд-части заключается в overlay-сценарии и минимизации пользовательских действий. Пользователь получает перевод прямо там, где возникла проблема, а не в отдельном сервисе после нескольких ручных шагов. Это делает продукт ближе к реальному поведению человека при работе с иностранным экранным контентом.

Дополнительным преимуществом является privacy-by-design: пользователь выбирает только нужную область экрана. Визуальная рамка, статусы и настройки помогают показать, что программа не анализирует весь экран, а работает с ограниченным фрагментом.

2.4.7 Степень готовности продукта, поставка и обслуживание

Степень готовности продукта можно определить как рабочий прототип, близкий к MVP. В программе реализованы основные пользовательские функции: выбор области экрана, OCR-распознавание, перевод, обычный режим, overlay-режим, горячие клавиши, рамка выбранной области, настройки окна перевода и остановка с очисткой результата.

Для коммерческого запуска требуется доработка стабильности, тестирование на разных типах экранов, упаковка приложения, подготовка инструкции, настройка установщика, улучшение обработки ошибок и

юридическая подготовка политики обработки данных. Поставка продукта предполагается в цифровом виде: установочный файл, архив приложения или корпоративная версия с настройками.

Сервисное обслуживание должно включать исправление ошибок, улучшение overlay-режима, обновление OCR/MT-интеграций, добавление языков, поддержку новых сценариев и консультации для B2B-клиентов.

2.5 Общая характеристика стартап-проекта и сферы деятельности

2.5.1 Основная бизнес-идея

Основная бизнес-идея проекта заключается в создании продукта, который позволяет переводить любой видимый текст на экране без ручного копирования. Пользователь получает инструмент, который сокращает путь от возникновения языкового барьера до понимания текста. Вместо последовательности «скриншот - OCR - переводчик - возврат в окно» предлагается единая цепочка внутри одного приложения.

Ценностное предложение формулируется следующим образом: Screen Translate помогает быстро понимать иностранный текст в любом окне Windows, сохраняя контекст работы и уменьшая количество ручных действий. Для частного пользователя это экономия времени, для студента - ускорение обучения, для IT-специалиста - более удобная работа с документацией и интерфейсами, для организации - повышение продуктивности сотрудников.

На первом этапе продукт ориентирован на персональные компьютеры под Windows, поскольку именно в этой среде часто встречаются игры, PDF-просмотрщики, десктопные программы, корпоративные системы и технические утилиты, где встроенного перевода нет.

2.5.2 Сфера деятельности

Сфера деятельности проекта относится к информационным и коммуникационным технологиям, разработке программного обеспечения,

инструментам продуктивности, OCR-решениям и машинному переводу. По своей природе продукт находится на стыке нескольких рынков: языковые сервисы, распознавание текста, переводческие технологии и desktop productivity software.

Проект не требует физического производства или складской логистики. Его результатом является цифровой продукт, который может распространяться в виде установочного файла, подписки, лицензии для организаций или индивидуальной версии для конечного пользователя.

2.5.3 Целевая аудитория

Целевая аудитория продукта включает студентов и школьников, использующих иностранные образовательные ресурсы, видеолекции, презентации, онлайн-курсы и PDF-материалы. Для них ключевая ценность состоит в ускорении обучения и снижении количества ручных действий.

Вторая группа - IT-специалисты, инженеры и технические сотрудники. Они регулярно сталкиваются с англоязычными сообщениями об ошибках, документацией, интерфейсами программ, настройками серверов, API-инструкциями и обучающими материалами. Для этой группы важны скорость, точность и возможность перевода прямо в рабочем контексте.

Третья группа - сотрудники компаний, которые работают с зарубежным программным обеспечением, CRM-системами, техническими каталогами, инструкциями и обучающими материалами. В этом сегменте возможны B2B-лицензии и пилотные внедрения.

Четвертая группа - частные пользователи, геймеры, исследователи и люди с недостаточной языковой подготовкой. Для них продукт полезен при работе с играми, видео, скриншотами, уведомлениями и иностранными интерфейсами.

2.5.4 Текущее состояние разработки

На текущем этапе создан рабочий прототип приложения Screen Translator на языке Python. Программа демонстрирует полный пользовательский сценарий: выбор области экрана, захват изображения,

распознавание текста, перевод и вывод результата. Реализованы обычный режим отображения в отдельном окне и overlay-режим поверх выбранной области.

Прототип поддерживает русский, английский, немецкий, польский, итальянский и французский языки. Для распознавания используется EasyOCR, для перевода - GoogleTranslator через библиотеку deep_translator. Захват экрана выполняется с помощью mss, обработка изображений - через PIL, NumPy и OpenCV. Интерфейс реализован на Tkinter.

Продукт находится на стадии MVP-прототипа. Основные функции уже работают, но для коммерческого запуска требуется дополнительное тестирование на разных типах экранов, улучшение стабильности, упаковка приложения, подготовка инструкции и проверка юридических аспектов обработки данных.

2.6 Объем, структура и развитие рынка

Целевой рынок проекта можно рассматривать как пересечение нескольких направлений: языковые сервисы, OCR-технологии, машинный перевод, инструменты продуктивности и программное обеспечение для работы с экранным контентом. Это не один узкий рынок, а ниша на стыке технологий, где пользовательская потребность формируется из необходимости быстро понимать иностранный цифровой контент.

Рынок языковых сервисов является крупным и растущим. Оценка language services составляет около 75,5 млрд долларов в 2026 году, а дальнейший рост поддерживается глобализацией контента, дистанционным обучением, международной работой, развитием ИИ-перевода и увеличением количества мультиязычных цифровых продуктов.

Отдельно развивается рынок OCR. Для проекта он важен, потому что без распознавания изображение экрана не может быть превращено в переводимый текст. В прогнозах по OCR указывается потенциал роста до 32,9 млрд долларов к 2030 году. OCR применяется в документообороте,

автоматизации бизнеса, распознавании изображений, мобильных приложениях и инструментах доступности.

Рынок машинного перевода также растет благодаря развитию нейросетевых моделей. Прогнозируемая оценка рынка машинного перевода составляет около 2,7 млрд долларов к 2030 году. Пользователи уже привыкли к автоматическому переводу, но большинство инструментов предполагает, что исходный текст уже доступен в цифровом виде. Screen Translate дополняет этот рынок этапом извлечения текста из изображения экрана.

Структура рынка применительно к проекту состоит из нескольких групп решений. Первая группа - онлайн-переводчики, такие как Google Translate, DeepL и Яндекс Переводчик. Они сильны качеством перевода, но требуют копируемого текста. Вторая группа - OCR-утилиты, которые извлекают текст, но часто не обеспечивают удобный перевод и вывод результата в контексте. Третья группа - браузерные переводчики, ограниченные веб-страницами. Четвертая группа - приложения экранного перевода, наиболее близкие к проекту, но часто ограниченные по скорости, языкам, настройкам или удобству overlay-сценария.

Потенциальная емкость аудитории достаточно широка. Продукт может быть полезен всем пользователям, которые работают с иностранным экранным контентом. Однако на начальном этапе целесообразно сфокусироваться на более узких сегментах: студенты, IT-специалисты, пользователи зарубежного ПО, сотрудники организаций и геймеры. Эти группы чаще сталкиваются с текстом, который нельзя скопировать, и быстрее понимают ценность продукта.

Первые три года развития проекта могут быть ориентированы на постепенное подтверждение спроса. На первом этапе важно довести MVP до устойчивой версии и собрать beta-пользователей. На втором этапе - получить первых платных пользователей и B2B-пилоты. На третьем этапе - масштабировать подписку и лицензии, расширить языки и улучшить стабильность продукта.

Таким образом, рынок для проекта является достаточно большим, но входить в него нужно через узкую и понятную нишу: перевод любого видимого текста на экране персонального компьютера. В этой нише конкурентным фактором будет не только качество перевода, но и скорость сценария, приватность, стабильность и удобство использования.

Для фронтенд-разработки рыночная возможность связана с качеством пользовательского опыта. Если продукт будет восприниматься как легкий и быстрый инструмент, он сможет конкурировать даже с более известными переводчиками, потому что решает другой сценарий - перевод прямо поверх исходного контента.

2.7 Анализ конкурентов и аналогов

В рамках анализа конкурентов можно выделить несколько типов решений. Первый тип - универсальные онлайн-переводчики. Они обладают высоким качеством перевода и известным брендом, но требуют, чтобы пользователь уже имел текст в копируемом виде. Если текст находится в изображении, игре, видео или desktop-интерфейсе, пользователь должен сначала извлечь его отдельно.

Второй тип - Google Lens и аналогичные визуальные переводчики. Они подтверждают существование спроса на перевод изображения, а не только копируемого текста. Однако такие решения чаще ориентированы на мобильные сценарии и не всегда удобны для постоянной работы на ПК. Кроме того, пользовательский сценарий может требовать переключения устройства или отдельного приложения.

Третий тип - OCR-утилиты и программы извлечения текста, например Text Grab и похожие инструменты. Они быстро распознают текст, но часто завершают задачу на этапе копирования. Пользователю все равно нужно отдельно вставить текст в переводчик и затем вернуться к исходному окну. Screen Translate отличается тем, что объединяет OCR, перевод и вывод результата в одной цепочке.

Четвертый тип - браузерные переводчики и автоматические субтитры. Они удобны для веб-страниц и отдельных видеосервисов, но не работают с произвольными окнами Windows, играми, PDF-просмотрщиками, приложениями и скриншотами. Их ограничение связано не только с качеством перевода, но и с областью применения.

Пятый тип - существующие screen translator-приложения. Они наиболее близки к проекту по идее, но могут иметь ограничения по языкам, скорости, качеству OCR, приватности, понятности настроек или стабильности работы на разных типах экранов. Для нового продукта важно конкурировать не только набором функций, но и простотой сценария.

Конкурентная стратегия проекта заключается в фокусе на Windows-сценарии, выбранной области экрана, быстром переводе, overlay-выводе, горячих клавишах и privacy-by-design. Вместо попытки конкурировать со всеми переводчиками сразу проект занимает конкретную нишу: мгновенный перевод экранного текста без ручного копирования.

Возможные конкурентные действия крупных игроков включают добавление более удобного экранного перевода в существующие продукты. Ответом проекта должен стать узкий фокус, быстрая доработка под реальные сценарии пользователей, поддержка B2B-пилотов, гибкость настроек и развитие тех функций, которые крупные универсальные сервисы часто не делают приоритетом.

С точки зрения фронтенда ключевые критерии сравнения включают количество действий пользователя, удобство выбора области, понятность режимов, читаемость overlay, наличие горячих клавиш, возможность настройки внешнего вида и сохранение исходного контекста. Именно эти параметры должны стать основой конкурентного преимущества продукта.

2.8 Бизнес-модель и команда стартап-проекта

2.8.1 Бизнес-модель

Бизнес-модель проекта строится на комбинированном подходе B2C и B2B. Для конечных пользователей целесообразна freemium-модель: базовая

версия дает возможность проверить основной сценарий, а платная подписка открывает расширенные возможности, повышенную стабильность, историю переводов, горячие клавиши, расширенные языки и дополнительные настройки.

Базовый тариф Free может включать ограничение по количеству минут, символов или переводов, а также базовый OCR и перевод. Такой тариф нужен для снижения барьера входа и проверки пользовательской ценности. Pro-тариф по цене 299 рублей в месяц может включать отсутствие лимитов базового сценария, overlay, историю и горячие клавиши. Pro+ по цене 499 рублей в месяц может включать расширенные языки, экспорт, приоритетную обработку и дополнительные настройки. Team-тариф по цене 799 рублей за пользователя в месяц может включать общие настройки, управление пользователями и документы для оплаты.

B2B-модель предполагает продажу лицензий организациям. Для компаний продукт может быть полезен как инструмент работы с иностранным ПО, документацией, техническими каталогами и обучающими материалами. Корпоративная лицензия может начинаться от 49 000 рублей в год и включать поддержку, пилотное внедрение, настройку под требования организации и консультации по использованию.

Логика дохода выглядит следующим образом: бесплатный вход в продукт позволяет набрать пользователей и получить обратную связь, затем часть пользователей переходит в Pro, а корпоративные сценарии развиваются в Team и B2B-лицензии. Для проекта особенно важно подтвердить, что overlay, скорость и приватность являются функциями, за которые пользователь готов платить.

2.8.2 Каналы продаж и продвижения

На начальном этапе продвижение целесообразно строить через демонстрацию конкретных сценариев. Потенциальный пользователь должен увидеть, что программа переводит текст из видео, игры, PDF, программы или технической документации без ручного копирования. Для этого подходят

короткие демонстрационные видео, публикации в тематических сообществах, материалы для студентов и IT-специалистов, а также описание кейсов использования.

Для B2C-сегмента каналами могут быть тематические форумы, социальные сети, студенческие сообщества, публикации на площадках для разработчиков и каталоги программ. Для IT-аудитории важно показать работу с документацией, ошибками, интерфейсами и техническими статьями.

Для B2B-сегмента каналами могут стать прямые обращения в образовательные организации, IT-компании, центры технической поддержки, команды внедрения зарубежного ПО и организации, где сотрудники регулярно работают с иностранными интерфейсами. В этом сегменте важны пилоты, понятная политика приватности и возможность настройки под требования компании.

2.8.3 Команда стартап-проекта

Команда проекта может быть представлена несколькими ролями. Первая роль - продуктовая и проектная. Она отвечает за формулировку ценностного предложения, интервью с пользователями, проверку гипотез, подготовку beta-теста, работу с B2B-пилотами, финансовую модель и упаковку продукта.

Вторая роль - разработка клиентской части. Она отвечает за интерфейс, пользовательский сценарий, настройку языков, обработку действий пользователя, отображение результата, понятные статусы, визуальную обратную связь и удобство работы с приложением.

Третья роль - разработка вычислительной части. Она отвечает за захват изображения, OCR, предобработку, машинный перевод, кэширование, оптимизацию задержек, управление памятью, устойчивость к ошибкам и поддержку языковых моделей.

На раннем этапе часть ролей может выполняться одним участником, но для масштабирования требуется распределение ответственности. Команда должна обладать компетенциями в Python-разработке, интерфейсах, OCR и

машинном переводе, тестировании пользовательских сценариев, продуктовой аналитике и коммуникации с потенциальными клиентами.

В данной ВКР индивидуальная роль связана с фронтенд-разработкой. Зона ответственности включает проектирование пользовательского сценария, реализацию окон, overlay, статусов, настроек, горячих клавиш и визуальной обратной связи. Результат этой роли должен сделать техническую цепочку OCR и перевода понятной и удобной для пользователя.

2.9 Финансовая модель и необходимость привлечения инвестиций

2.9.1 Предпосылки финансовой модели

Финансовая модель проекта должна быть осторожной, потому что продукт находится на стадии рабочего прототипа и требует проверки спроса. Для первых лет нецелесообразно закладывать чрезмерно высокие показатели. Основной целью является не мгновенный массовый рост, а подтверждение готовности пользователей регулярно использовать продукт и платить за расширенные функции.

В качестве источников выручки рассматриваются Pro-подписки, доработки и настройки под клиентов, а также B2B-лицензии. На первый год закладывается выручка около 220 000 рублей. Она может состоять из 60 000 рублей от Pro-подписок, 60 000 рублей от доработок и настройки и 100 000 рублей от первых B2B-лицензий или пилотов.

На второй год плановая выручка может вырасти до 450 000 рублей: 140 000 рублей от Pro-подписок, 90 000 рублей от доработок и 220 000 рублей от B2B. На третий год целевой показатель составляет около 750 000 рублей: 260 000 рублей от подписок, 110 000 рублей от доработок и 380 000 рублей от B2B-лицензий.

Такая модель выглядит реалистично для учебного стартап-проекта, потому что она не предполагает огромной пользовательской базы с первого года. Рост происходит постепенно: сначала проверяется MVP и собирается

обратная связь, затем формируются первые платные пользователи, затем появляются корпоративные сценарии.

2.9.2 Расходы проекта

Основные расходы проекта связаны с разработкой, тестированием, поддержкой, OCR/MT-интеграциями, маркетингом и юридической подготовкой. На старте значительная часть работы выполняется командой проекта, поэтому прямые денежные расходы могут быть относительно небольшими. Однако для коммерческого запуска требуется время на доработку, проверку стабильности и подготовку продукта к распространению.

Расходы на OCR и машинный перевод могут возникать при переходе от демонстрационной версии к коммерческой. В текущей реализации используется EasyOCR и GoogleTranslator через библиотеку deep_translator, но для стабильного коммерческого продукта может потребоваться надежный платный переводческий провайдер, резервный OCR-движок или гибридная архитектура.

Маркетинговые расходы включают подготовку демонстрационных видео, лендинга, публикаций, тестовых рекламных кампаний и материалов для B2B-пилотов. Юридические расходы могут включать регистрацию программного обеспечения, подготовку пользовательского соглашения, политики обработки данных и договоров с организациями.

2.9.3 Потребность в финансировании

Минимальная потребность в финансировании на стартовом этапе оценивается в 140 000 рублей. Эта сумма может быть направлена на завершение разработки, тестирование, упаковку продукта, юридическую подготовку и стартовый маркетинг. Основная часть средств должна быть направлена на доработку продукта, часть - на тестирование и исследование пользовательского опыта, часть - на продвижение и юридическое оформление.

Привлечение финансирования может осуществляться в форме гранта, партнерского финансирования, предоплаты от B2B-клиента или участия в программе поддержки студенческих стартапов. Для проекта важно не столько крупное финансирование, сколько возможность довести MVP до состояния, пригодного для beta-тестирования и первых пилотов.

2.9.4 Точка безубыточности и инвестиционная привлекательность

Поскольку проект является цифровым продуктом, он не требует аренды производственных площадей, закупки оборудования или хранения складских запасов. Основные затраты связаны с трудом разработчиков, тестированием, поддержкой и внешними сервисами. Это снижает точку безубыточности и делает проект устойчивым при небольшом количестве платных пользователей.

Инвестиционная привлекательность проекта заключается в сочетании растущего рынка, понятной пользовательской боли, уже реализованного рабочего прототипа и низких стартовых расходов. При успешной проверке гипотез продукт может масштабироваться за счет подписочной модели и B2B-лицензий.

Для фронтенд-части приоритетными расходами являются доработка overlay, улучшение читаемости, тестирование интерфейса на разных сценариях, исправление UX-ошибок, подготовка установочного пакета и пользовательской инструкции. Качество фронтенда влияет на конверсию, удержание и готовность пользователя платить за Pro-функции.

2.10 Производственный и организационный план

2.10.1 Общие сведения о производстве цифрового продукта

Производство в данном проекте означает разработку, тестирование, упаковку и сопровождение программного продукта. Так как продукт цифровой, не требуется производственная площадка, сырье, склад или физическая логистика. Основным ресурсом является труд разработчиков, время на тестирование и программная инфраструктура.

Для фронтенд-части производственный процесс связан с проектированием пользовательского сценария, реализацией окон и overlay, настройкой горячих клавиш, проверкой удобства интерфейса, тестированием читаемости результата и устранением визуальных ошибок.

2.10.2 Технологическая цепочка разработки

Технологическая цепочка начинается с анализа пользовательских сценариев. Нужно определить, как пользователь выбирает область, какие действия выполняет чаще всего, когда ему нужен overlay, когда достаточно отдельного окна и какие настройки не должны мешать базовому сценарию.

Следующий этап - разработка интерфейса. На текущем этапе используется Tkinter: главное окно, окно настроек, окно выбора области, окно перевода, рамка выбранной области и overlay. Для каждого окна определяются состояния, обработчики событий, реакции на горячие клавиши и правила отображения.

Отдельный этап - разработка overlay. Здесь требуется совместить координаты OCR-фрагментов с визуальным слоем перевода, подобрать шрифт и цвет, стереть исходный текст в пределах OCR-блоков, разместить перевод и не допустить наложений. Для этого фронтенд должен учитывать ширину строк, высоту блоков, фон, контраст и границы выбранной области.

Затем выполняется тестирование UX. Проверяются сценарии выбора области, отмены выбора, слишком маленькой области, остановки перевода, перезапуска, переключения режимов, копирования результата, изменения прозрачности и поведения overlay при скриншоте.

Последний этап - упаковка и подготовка к распространению. Для фронтенда важно, чтобы пользователь мог запустить приложение без настройки среды разработки, быстро понять интерфейс и получить результат без чтения сложной инструкции.

2.10.3 Кадровое обеспечение и организация работ

На начальном этапе фронтенд-часть может выполняться одним разработчиком. Он отвечает за интерфейс, пользовательский сценарий,

визуальные состояния, горячие клавиши, настройки и тестирование. При расширении проекта может потребоваться UX/UI-специалист, тестировщик и разработчик desktop-приложений.

Организация работ должна строиться итерационно. Сначала реализуется базовый сценарий, затем проверяется на пользователях, после этого дорабатываются спорные элементы интерфейса. Такой подход особенно важен для overlay, где реальные проблемы видны только при тестировании на разных приложениях и фонах.

2.10.4 Материалы, оборудование, безопасность и экологичность

Потребность в физических материалах отсутствует. Используемые инструменты включают компьютер разработчика, операционную систему Windows, Python, Tkinter, mss, PIL, EasyOCR, deep_translator, NumPy, PyTorch, OpenCV и вспомогательные библиотеки. Основные расходы связаны не с материалами, а со временем разработки и тестирования.

Экологическая нагрузка минимальна, поскольку продукт цифровой. Основной вопрос безопасности связан не с физическим производством, а с информационной приватностью. Для фронтенда важно явно показывать выбранную область, не создавать у пользователя ощущения скрытого захвата экрана и давать понятное управление запуском и остановкой перевода.

2.11 Описание развития бизнес-проекта

Развитие проекта должно происходить поэтапно. На начальном этапе главная цель - довести рабочий прототип до стабильного MVP. Для этого необходимо устранить выявленные ошибки, протестировать продукт на разных типах экранов, проверить работу с несколькими языками, улучшить обработку сложных изображений и подготовить понятную инструкцию для пользователя.

Первый этап - ближайшее время. На этом этапе выполняется исправление ошибок, улучшение стабильности, настройка поведения горячих клавиш, подготовка инструкции и демонстрационных материалов. Ожидаемый результат - MVP, который можно показывать пользователям и

использовать в типовых сценариях: PDF, видеокурс, интерфейс программы, игра, скриншот, техническая документация.

Второй этап - beta-тестирование. Целесообразно привлечь 30-50 пользователей из основных целевых групп: студенты, IT-специалисты, пользователи зарубежного ПО, геймеры и сотрудники организаций. В ходе beta-теста нужно измерять точность OCR, задержку перевода, удобство выбора области, понятность статусов, частоту повторного использования и готовность платить за расширенные функции.

Третий этап - запуск первых платных функций. После beta-теста можно сформировать Free/Pro-модель. Бесплатная версия позволяет проверить базовый сценарий, а платная открывает регулярное использование без лимитов, overlay, историю, горячие клавиши, расширенные языки и дополнительные настройки. В этот же период целесообразно начать первые B2B-пилоты.

Четвертый этап - развитие на горизонте одного-трех лет. На этом этапе продукт расширяет языки, улучшает OCR, добавляет пресеты областей, историю переводов, экспорт, корпоративные настройки, управление пользователями и поддержку разных провайдеров OCR/MT. При успешном росте возможно создание отдельной версии для организаций.

Ключевые метрики развития должны оставаться измеримыми. Для продукта важны количество успешно переведенных экранных фрагментов, средняя задержка от выбора области до вывода результата, доля повторных пользователей, количество активных Pro-подписок, число B2B-пилотов, количество обращений в поддержку и конверсия из Free в Pro.

Проект способен масштабироваться, потому что его сценарий не привязан к одному типу контента. Один и тот же продукт можно использовать в учебе, работе, играх, документации и корпоративных системах. Модульная архитектура позволяет развивать клиентскую часть и вычислительное ядро независимо друг от друга.

Для фронтенд-направления ключевые задачи развития - повышение читаемости overlay, добавление пресетов областей, улучшение окна настроек,

упрощение первого запуска, сохранение пользовательских параметров и более понятная обработка ошибок. Чем меньше действий выполняет пользователь, тем выше ценность продукта.

2.12 Описание рисков

Любой стартап-проект связан с рисками. Для Screen Translate наиболее значимыми являются технологические, пользовательские, рыночные, юридические, финансовые и кадровые риски. Важно не только перечислить их, но и заранее определить меры снижения.

Первый риск - недостаточное качество OCR на сложных экранах. Нестандартные шрифты, низкое качество изображения, динамический фон, прозрачные интерфейсы, игры и видео могут ухудшать распознавание. Этот риск имеет среднюю вероятность и высокое влияние, потому что без распознавания текста продукт не сможет выполнить основную функцию. Меры снижения включают предобработку изображений, тестовый набор типовых экранов, поддержку нескольких OCR-подходов и улучшение фильтрации распознанных фрагментов.

Второй риск - задержка перевода. Если результат появляется слишком поздно, пользователь вернется к ручному переводчику или перестанет пользоваться продуктом. Для снижения риска используются обработка только выбранной области, ограничение размера изображения, кэширование, управление частотой OCR, оптимизация количества потоков и повторное использование результатов, если текст не изменился.

Третий риск - опасения пользователей по поводу приватности. Приложение работает с экраном, поэтому пользователь может бояться, что анализируется весь рабочий стол или чувствительные данные. Меры снижения: обработка только выбранной области, понятная визуальная рамка, прозрачные настройки, локальная обработка там, где это возможно, и отсутствие хранения лишних изображений.

Четвертый риск - стоимость внешних API и снижение маржинальности. Если при масштабировании потребуется платный OCR или переводчик,

расходы могут расти вместе с количеством пользователей. Меры снижения включают лимиты бесплатного тарифа, оптимизацию запросов, кэширование, пакетную обработку и использование локальных моделей для части сценариев.

Пятый риск - конкуренция со стороны крупных сервисов. Крупные переводчики могут добавить более удобные функции экранного перевода. Ответ проекта должен быть в узком фокусе на Windows-сценарии, скорости, выбранной области, overlay, горячих клавишах, приватности и B2B-пилотах.

Шестой риск - недостаточный спрос на платную версию. Пользователи могут использовать только бесплатный функционал или бесплатные аналоги. Для снижения риска платные функции должны иметь очевидную ценность: стабильный overlay, отсутствие лимитов, история, горячие клавиши, экспорт, приоритетная обработка и корпоративные настройки.

Седьмой риск - кадровая зависимость. На старте значительная часть знаний сосредоточена у разработчика. Для снижения риска нужно документировать код, хранить проект в репозитории, описывать архитектуру, формировать тест-кейсы и постепенно разделять роли внутри команды.

Восьмой риск - юридические и лицензионные ограничения библиотек и сервисов. При коммерческом развитии необходимо проверить условия использования OCR-библиотек, переводческих сервисов, моделей и зависимостей. Для снижения риска требуется выбрать правомерную модель распространения, подготовить пользовательское соглашение и описать обработку данных.

В целом риски проекта являются управляемыми. Главные зоны контроля - качество OCR, скорость, приватность, стоимость обработки, конкурентная стратегия и поддерживаемость кода. По каждому направлению предусмотрены меры снижения, а дальнейшее тестирование позволит уточнить вероятность и влияние рисков.

Для фронтенд-разработки отдельным риском является плохая читаемость overlay. Перевод может быть слишком мелким, перекрывать важные элементы или плохо смотреться на сложном фоне. Меры снижения

включают настройку прозрачности, подбор цветов, адаптивный размер шрифта, возможность скрыть рамку, отдельный режим окна и тестирование на разных типах экранов.

Еще один фронтенд-риск - перегрузка интерфейса настройками. Если пользователь не понимает, что нажимать, продукт теряет преимущество простоты. Для снижения риска необходимо оставлять основной экран минимальным, а расширенные параметры переносить в отдельный раздел настроек.

2.13 Заключение по стартап-проекту

Стартап-проект Screen Translate представляет собой программное обеспечение для захвата, распознавания и перевода текстового контента в реальном времени. Его основная ценность заключается в решении конкретной пользовательской боли: необходимости быстро понимать иностранный текст, который виден на экране, но не копируется обычными средствами.

В рамках темы «Фронтенд-разработка» ключевым результатом является проработка и реализация пользовательского сценария. Пользователь выбирает область, запускает распознавание, получает перевод в отдельном окне или overlay и продолжает работу в исходном контексте. Именно фронтенд делает цепочку OCR и машинного перевода понятной, быстрой и удобной.

Анализ проблемы показывает, что потребность в screen translation существует у разных групп пользователей: студентов, IT-специалистов, сотрудников компаний, исследователей, геймеров и обычных пользователей. Интервью подтверждают, что людям важны скорость, overlay, горячие клавиши и приватность выбранной области.

Текущая реализация программы уже включает важные элементы продукта: выбор области экрана, OCR через EasyOCR, перевод через GoogleTranslator, два режима вывода, горячие клавиши, настройки интерфейса, overlay, кэширование, стабилизацию результата и очистку

памяти. Это позволяет рассматривать проект не только как учебную разработку, но и как основу для MVP.

Бизнес-модель проекта основана на freemium-подходе, Pro-подписках, командных тарифах и B2B-лицензиях. Финансовая модель выбрана осторожной и предполагает постепенный рост выручки: 220 000 рублей в первый год, 450 000 рублей во второй год и 750 000 рублей в третий год. Минимальная потребность в финансировании составляет 140 000 рублей и направлена на доработку, тестирование, упаковку и стартовое продвижение продукта.

Главные риски проекта связаны с качеством OCR, задержкой перевода, читаемостью overlay, приватностью, конкуренцией и стоимостью внешних сервисов. Для каждого риска предложены меры снижения. При дальнейшей доработке продукт может выйти на beta-тестирование, подтвердить спрос и развиваться как инструмент для частных пользователей и организаций.

Таким образом, проект является практически значимым стартапом с понятной проблемой, рабочим прототипом, рыночной логикой и перспективами развития. Фронтенд-разработка в данном проекте играет ключевую роль, потому что именно она превращает техническую возможность OCR и перевода в удобный пользовательский продукт.

3 Охрана труда

Разработка и эксплуатация ПО выполняется с использованием персональной электронно-вычислительной машины. Поэтому в выпускной квалификационной работе необходимо рассмотреть вопросы охраны труда и безопасной организации рабочего места пользователя.

Работа с приложением Screen Translator связана с длительным наблюдением за экраном, использованием клавиатуры и мыши, а также возможной работой с несколькими окнами одновременно. Такие условия требуют соблюдения правил эргономики, освещения и режима труда.

В разделе рассматриваются основные опасные и вредные факторы, требования к рабочему месту, режим труда и отдыха, а также вопросы электробезопасности и пожарной безопасности.

3.1 Анализ опасных и вредных факторов при работе с ПЭВМ

При работе с персональным компьютером на пользователя воздействуют факторы зрительного напряжения, статической нагрузки, монотонности, неудобной рабочей позы и психоэмоционального напряжения.

Зрительное напряжение возникает из-за постоянного наблюдения за экраном. В случае работы с переводчиком экранного текста пользователь может долго рассматривать мелкие символы, переведенные надписи и элементы интерфейса.

Статическая нагрузка связана с длительным положением тела за рабочим столом. При неправильной высоте стула или монитора повышается нагрузка на шею, спину и плечевой пояс.

Монотонность возникает при повторяющихся действиях: выборе области, запуске перевода, чтении результата и переключении между окнами. Для снижения утомления необходимо делать регулярные перерывы.

Психоэмоциональное напряжение может возникать при работе с большим количеством информации или при необходимости быстро понять иностранный текст. Удобный интерфейс частично снижает такую нагрузку, так как уменьшает количество ручных действий.

3.2 Требования к рабочему месту

Рабочее место пользователя должно быть организовано таким образом, чтобы экран находился на удобном расстоянии, а верхняя граница монитора располагалась примерно на уровне глаз.

Клавиатура и мышь должны находиться на одной поверхности и позволять удерживать кисти без чрезмерного изгиба. Это особенно важно при использовании горячих клавиш и частом управлении программой.

Стул должен обеспечивать поддержку спины, а высота сиденья должна позволять пользователю ставить стопы на пол или на подставку. Неправильная посадка быстро приводит к утомлению.

На рабочем столе следует оставить достаточно места для свободного перемещения мыши. Так как в программе используется выделение области экрана, точность движения мыши влияет на удобство работы.

Окружающие предметы не должны закрывать экран или создавать блики. Для программ экранного перевода это особенно важно, так как пользователь должен хорошо различать исходный текст и перевод.

3.3 Освещение, микроклимат и режим труда

Освещение рабочего места должно быть равномерным и не создавать бликов на экране. Слишком яркий источник света за спиной пользователя ухудшает видимость текста и повышает нагрузку на зрение.

При работе в темное время суток не рекомендуется использовать монитор как единственный источник света. Лучше применять мягкое фоновое освещение, которое снижает контраст между экраном и окружающим пространством.

Температура и влажность воздуха должны соответствовать нормальным условиям работы в помещении. Перегрев или сухой воздух усиливают утомляемость и снижают концентрацию.

При длительной работе за компьютером рекомендуется делать короткие перерывы. Во время перерыва следует отвести взгляд от экрана, выполнить простые упражнения для глаз и сменить положение тела.

В клиентской части разработанного приложения предусмотрены темная и светлая темы. Это позволяет снизить зрительный дискомфорт в разных условиях освещения.

3.4 Электробезопасность и пожарная безопасность

Персональный компьютер и периферийные устройства должны подключаться к исправной электрической сети. Не допускается использование поврежденных кабелей, перегруженных удлинителей и нестабильных розеток.

Перед началом работы следует убедиться, что провода не находятся в местах, где их можно случайно задеть ногой или повредить мебелью. Кабели должны быть уложены так, чтобы не создавать препятствий для пользователя.

Пожарная безопасность обеспечивается исправностью оборудования, отсутствием перегрева и соблюдением правил эксплуатации электроприборов. Не следует закрывать вентиляционные отверстия системного блока или ноутбука.

В рабочем помещении должен быть свободный доступ к средствам пожаротушения и путям эвакуации. Пользователь должен знать порядок действий при обнаружении запаха гари или неисправности оборудования.

Соблюдение перечисленных требований обеспечивает безопасную работу как при разработке программы, так и при ее дальнейшей эксплуатации.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана backend-часть программного обеспечения для захвата, распознавания и перевода текстового контента в реальном времени. Работа была направлена на решение практической задачи: обработать изображение с текстом, полученное от клиентской части, распознать содержимое, перевести его и вернуть результат в структурированном виде.

В первой части работы была рассмотрена предметная область экранного перевода, выполнен анализ аналогов и сформулирована постановка задачи. Функциональные требования были описаны с позиции backend-сервиса: принять изображение, проверить параметры, выполнить OCR, перевести текст, вернуть координаты и итоговый перевод. Нефункциональные требования были связаны с устойчивостью, ограничением размера изображения, кэшированием, временем обработки и обработкой ошибок.

На этапе проектирования была сформирована архитектура программы. Backend в данном проекте рассматривается как серверный слой, который выполняет прием запроса, предобработку изображения, OCR, перевод, кэширование и подготовку JSON-ответа. Пользовательские окна, управляющие клавиши и визуальный вывод результата относятся к клиентской части и отделены от серверной логики.

В процессе разработки реализованы основные backend-компоненты: APIController, ProcessingCoordinator, ImagePreprocessor, OCRService, TranslationService и CacheService. Каждый компонент выполняет отдельную роль, поэтому структура программы остается последовательной и понятной. В основной части приведены примеры реализации, а в приложении размещен сокращенный листинг ключевых частей программы.

Проверка ПО показала, что backend корректно выполняет основной сценарий: принимает изображение и параметры перевода, проводит предварительную обработку, распознает текст, переводит найденные строки,

использует кэш при повторных запросах и возвращает клиентской части структурированный результат.

Дальнейшее развитие программы может быть связано с очередью фоновых задач, расширенным логированием, поддержкой нескольких OCR-моделей, историей запросов, профилированием производительности и упаковкой backend-сервиса вместе с клиентским приложением. При этом базовая структура уже позволяет развивать продукт без изменения основной идеи разделения клиентской и серверной частей.

Список использованных источников

1. ГОСТ 7.32-2017. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. (дата обращения: 03.01.2026).

2. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. (дата обращения: 13.04.2026).

3. ГОСТ 2.105-2019. Единая система конструкторской документации. Общие требования к текстовым документам. (дата обращения: 03.03.2026).

4. Python Software Foundation. Документация Python. URL: https://docs.python.org/3/ (дата обращения: 05.02.2026).

5. FastAPI. FastAPI documentation. URL: https://fastapi.tiangolo.com/ (дата обращения: 10.03.2026).

6. Pydantic documentation. URL: https://docs.pydantic.dev/ (дата обращения: 11.04.2026).

7. Pillow. Pillow documentation. URL: https://pillow.readthedocs.io/ (дата обращения: 12.03.2026).

8. NumPy documentation. URL: https://numpy.org/doc/ (дата обращения: 03.06.2026).

9. Jaided AI. EasyOCR documentation. URL: https://www.jaided.ai/easyocr/documentation/ (дата обращения: 03.06.2026).

10. deep-translator documentation. URL: https://deep-translator.readthedocs.io/ (дата обращения: 07.02.2026).

11. Starlette documentation. URL: https://www.starlette.io/ (дата обращения: 04.04.2026).

12. Uvicorn. ASGI web server implementation. URL: https://www.uvicorn.org/ (дата обращения: 02.03.2026).

13. Requests: HTTP for Humans. URL: https://requests.readthedocs.io/ (дата обращения: 03.06.2026).

14. OpenCV documentation. URL: https://docs.opencv.org/ (дата обращения: 21.03.2026).

15. PyTorch documentation. URL: https://pytorch.org/docs/ (дата обращения: 27.02.2026).

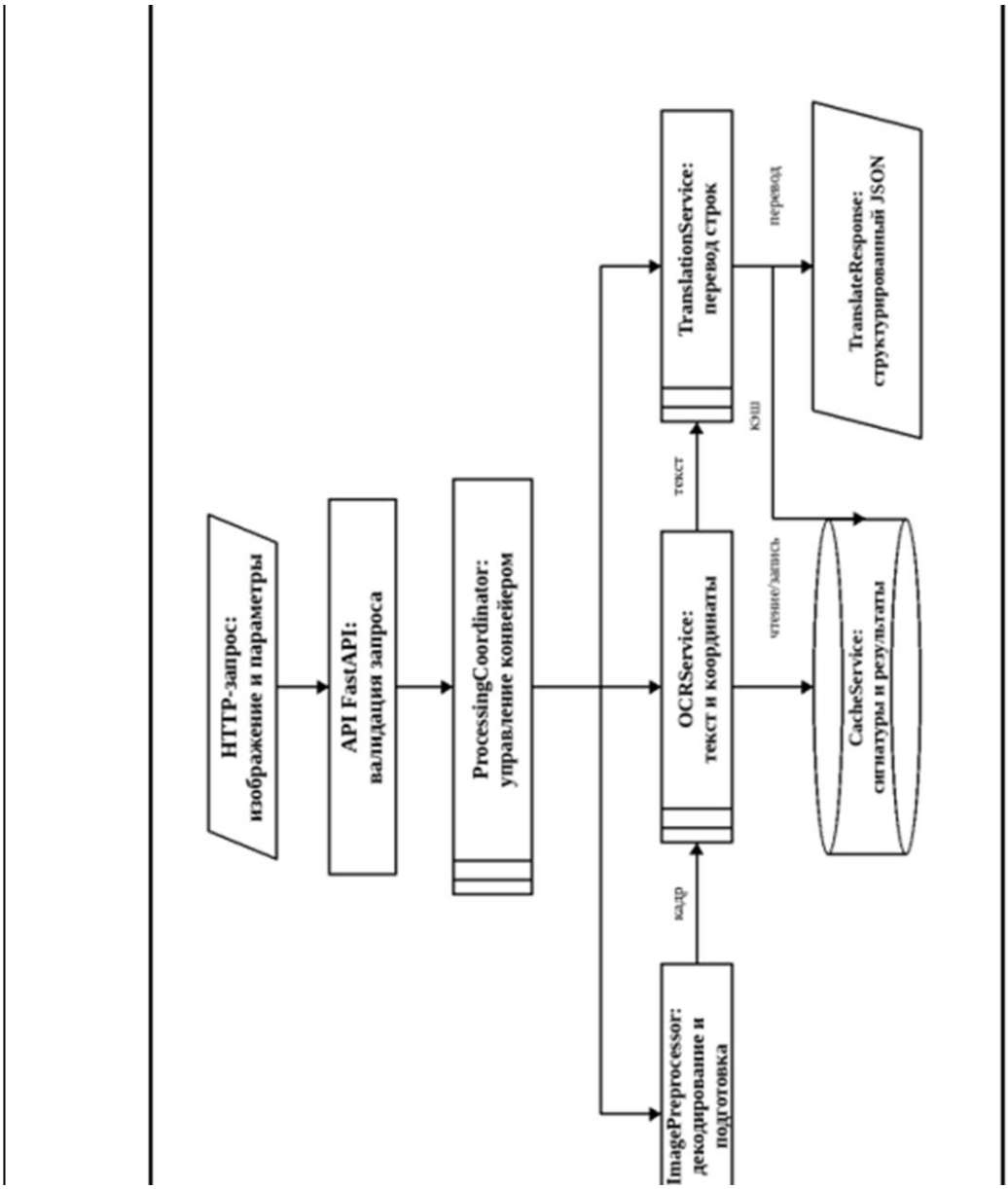
Приложение А

(обязательное)

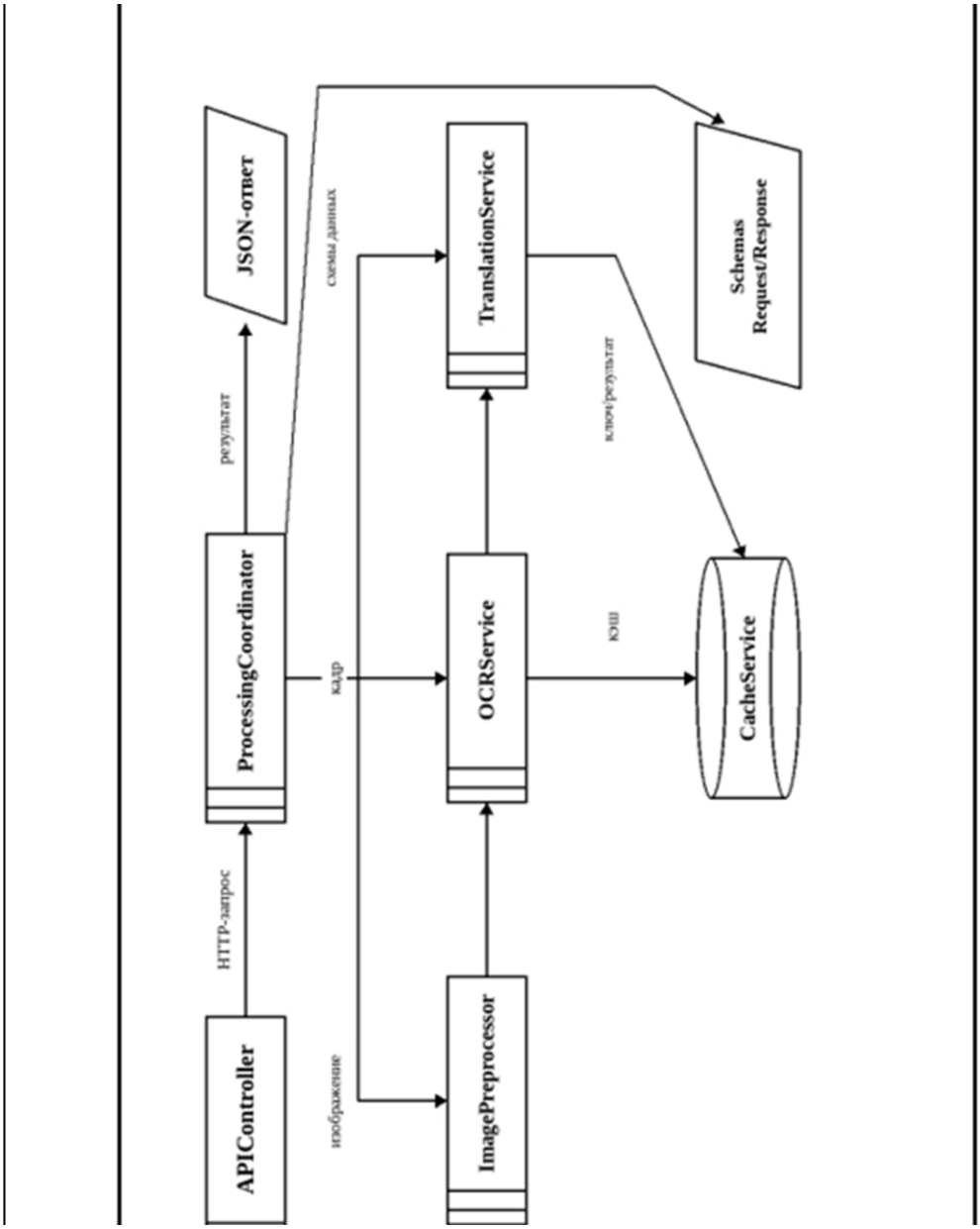
UML-диаграммы

ПО для захвата, распознавания и перевода текстового контента в реальном времени. Бэкенд-разработка

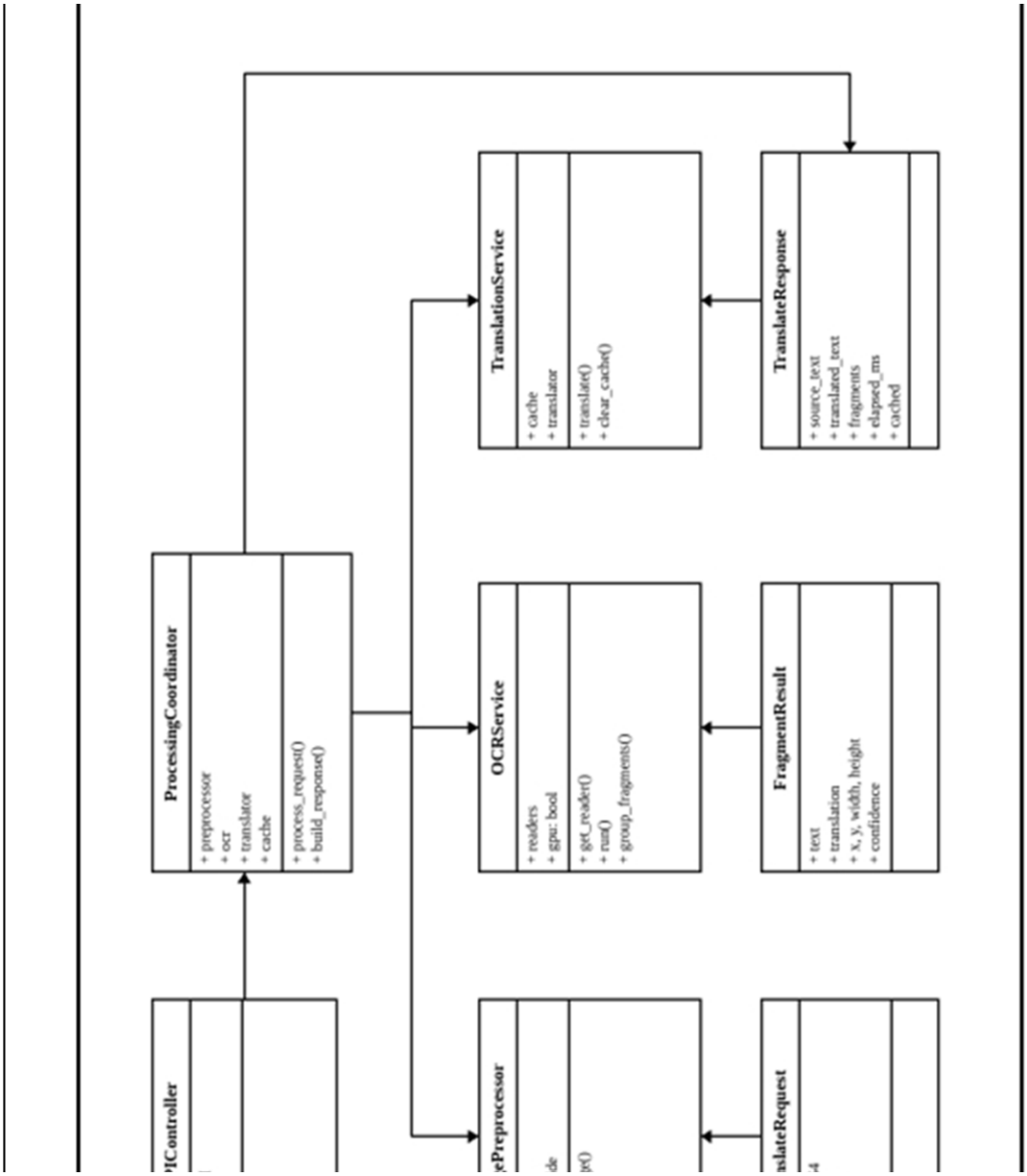
Приложение А.1 - Диаграмма компонентов backend-части



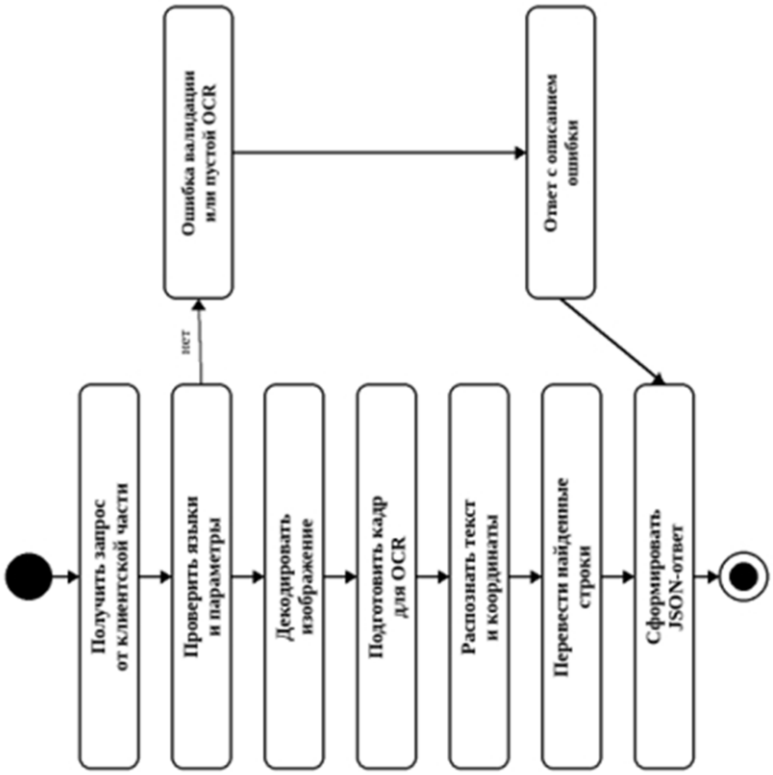
Приложение А.2 - Архитектура программы с backend-сервисом



Приложение А.3 - Диаграмма классов backend-части



Приложение А.4 - Диаграмма деятельности основного backend-сценария.


Получить запрос от клиентской части
Проверить языки и параметры
Ошибка валидации или пустой OCR
нет
Ответ с описанием ошибки
Декодировать изображение
Подготовить кадр для OCR
Распознать текст и координаты
Перевести найденные строки
Сформировать JSON-ответ

Приложение Б

(обязательное)

Сокращенный листинг программы

В приложении приведен сокращенный листинг ключевых фрагментов backend-сервиса. Второстепенные методы логирования, настройки окружения, вспомогательная сериализация и обработка частных ошибок исключены из листинга, чтобы оставить основной конвейер обработки запроса.

```
import base64
import hashlib
import time
from io import BytesIO
from typing import List

import numpy as np
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel, Field
from PIL import Image, ImageOps, ImageEnhance
from deep_translator import GoogleTranslator

try:
    import easyocr
except ImportError:
    easyocr = None

app = FastAPI(title="Screen Translator Backend")

TRANSLATOR_LANGUAGES = {
    "русский": "ru",
    "английский": "en",
    "немецкий": "de",
    "польский": "pl",
    "итальянский": "it",
    "французский": "fr",
}

EASYOCR_LANG_BY_CODE = {
```

```
    "ru": ["ru", "en"],
    "en": ["en"],
    "de": ["de", "en"],
    "pl": ["pl", "en"],
    "it": ["it", "en"],
    "fr": ["fr", "en"],
}

EASYOCR_GPU = False
EASYOCR_PARAGRAPH_MODE = False
EASYOCR_MAX_OCR_PIXELS = 2_200_000
EASYOCR_MAX_OCR_LONG_SIDE = 1800

class TranslateRequest(BaseModel):
    image_base64: str = Field(..., description="Изображение
области экрана в base64")
    source_lang: str = Field(..., description="Код исходного
языка")
    target_lang: str = Field(..., description="Код языка
перевода")
    min_confidence: int = 20
    ocr_scale: int = 1

class FragmentResult(BaseModel):
    text: str
    translation: str
    x: int
    y: int
    width: int
    height: int
    confidence: float

class TranslateResponse(BaseModel):
    source_text: str
    translated_text: str
    fragments: List[FragmentResult]
```

```
    elapsed_ms: int
    cached: bool = False

class ImageDecodeError(Exception):
    pass

class OcrEmptyResult(Exception):
    pass

def clean_text(text: str) -> str:
    return " ".join(text.strip().split())

class ImagePreprocessor:
    def decode_image(self, image_base64: str) -> Image.Image:
        try:
            raw = base64.b64decode(image_base64)
            image = Image.open(BytesIO(raw)).convert("RGB")
            return image
        except Exception as error:
            raise ImageDecodeError("Не удалось декодировать
изображение") from error

    def signature(self, image: Image.Image) -> str:
        return hashlib.sha256(image.tobytes()).hexdigest()

    def prepare(self, image: Image.Image, scale: int = 1):
        original_width, original_height = image.size
        target_width = max(1, int(original_width * scale))
        target_height = max(1, int(original_height * scale))

        if target_width * target_height >
EASYOCR_MAX_OCR_PIXELS:
            factor = (EASYOCR_MAX_OCR_PIXELS / max(1,
target_width * target_height)) ** 0.5
            target_width = max(1, int(target_width * factor))
```

```
            target_height = max(1, int(target_height * factor))

        longest = max(target_width, target_height)
        if longest > EASYOCR_MAX_OCR_LONG_SIDE:
            factor = EASYOCR_MAX_OCR_LONG_SIDE / longest
            target_width = max(1, int(target_width * factor))
            target_height = max(1, int(target_height * factor))

        image = ImageOps.grayscale(image)
        image = ImageOps.autocontrast(image)
        image = ImageEnhance.Contrast(image).enhance(1.6)

        if (target_width, target_height) != image.size:
            image = image.resize((target_width, target_height),
Image.Resampling.BILINEAR)
        return image

class OCRService:
    def __init__(self):
        self.readers = {}

    def get_reader(self, source_lang: str):
        if easyocr is None:
            raise RuntimeError("EasyOCR не установлен")
        langs = tuple(EASYOCR_LANG_BY_CODE[source_lang])
        if langs not in self.readers:
            self.readers[langs] = easyocr.Reader(list(langs),
gpu=EASYOCR_GPU)
        return self.readers[langs]

    def run(self, image: Image.Image, source_lang: str,
min_confidence: int):
        reader = self.get_reader(source_lang)
        raw_results = reader.readtext(
            np.array(image.convert("RGB")),
```

```
            detail=1,
            paragraph=EASYOCR_PARAGRAPH_MODE,
            decoder="greedy",
            batch_size=1,
            workers=0,
        )
        fragments = []
        for box, text, confidence in raw_results:
            cleaned = clean_text(str(text))
            if cleaned and confidence * 100 >= min_confidence:
                xs = [point[0] for point in box]
                ys = [point[1] for point in box]
                fragments.append({
                    "text": cleaned,
                    "box": box,
                    "x": int(min(xs)),
                    "y": int(min(ys)),
                    "width": int(max(xs) - min(xs)),
                    "height": int(max(ys) - min(ys)),
                    "confidence": confidence * 100,
                })
        if not fragments:
            raise OcrEmptyResult()
        return fragments


class TranslationService:
    def __init__(self):
        self.cache = {}


    def translate(self, text: str, source_lang: str,
target_lang: str) -> str:
        key = (text, source_lang, target_lang)
        if key not in self.cache:
            translator = GoogleTranslator(source=source_lang,
target=target_lang)
            self.cache[key] = translator.translate(text)
```

```
        return self.cache[key]


class ProcessingCoordinator:
    def __init__(self):
        self.preprocessor = ImagePreprocessor()
        self.ocr = OCRService()
        self.translator = TranslationService()
        self.image_cache = {}

    def process_request(self, request: TranslateRequest) ->
TranslateResponse:
        started_at = time.perf_counter()
        image =
self.preprocessor.decode_image(request.image_base64)
        signature = self.preprocessor.signature(image)
        cache_key = (signature, request.source_lang,
request.target_lang)

        if cache_key in self.image_cache:
            response = self.image_cache[cache_key]
            response.cached = True
            return response

        prepared = self.preprocessor.prepare(image,
scale=request.ocr_scale)
        fragments = self.ocr.run(prepared, request.source_lang,
request.min_confidence)

        result_fragments = []
        source_lines = []
        translated_lines = []
        for fragment in fragments:
            translation = self.translator.translate(
                fragment["text"], request.source_lang,
request.target_lang
```

```
            )
            source_lines.append(fragment["text"])
            translated_lines.append(translation)
            result_fragments.append(FragmentResult(
                text=fragment["text"],
                translation=translation,
                x=fragment["x"],
                y=fragment["y"],
                width=fragment["width"],
                height=fragment["height"],
                confidence=fragment["confidence"],
            ))

        elapsed_ms = int((time.perf_counter() - started_at) *
1000)
        response = TranslateResponse(
            source_text="\n".join(source_lines),
            translated_text="\n".join(translated_lines),
            fragments=result_fragments,
            elapsed_ms=elapsed_ms,
            cached=False,
        )
        self.image_cache[cache_key] = response
        return response

coordinator = ProcessingCoordinator()

@app.get("/api/health")
def health():
    return {"status": "ok"}

@app.get("/api/languages")
def languages():
    return TRANSLATOR_LANGUAGES
```

```
@app.post("/api/translate", response_model=TranslateResponse)
def translate(request: TranslateRequest):
    if request.source_lang not in EASYOCR_LANG_BY_CODE:
        raise HTTPException(status_code=400,
detail="Неподдерживаемый исходный язык")
    if request.target_lang not in EASYOCR_LANG_BY_CODE:
        raise HTTPException(status_code=400,
detail="Неподдерживаемый язык перевода")
    try:
        return coordinator.process_request(request)
    except ImageDecodeError as error:
        raise HTTPException(status_code=400, detail=str(error))
    except OcrEmptyResult:
        raise HTTPException(status_code=422, detail="Текст на
изображении не найден")
    except Exception as error:
        raise HTTPException(status_code=500, detail=f"Ошибка
backend-обработки: {error}")

@app.delete("/api/cache")
def clear_cache():
    coordinator.image_cache.clear()
    coordinator.translator.cache.clear()
    return {"status": "cache cleared"}
```

Приложение В

(обязательное)

Презентация

ПРОЕКТ

SCREEN TRANSLATE

Бизнес-проект «Программное обеспечение для захвата, распознавания и перевода текстового контента в реальном времени.

ПО захватывает текст с экрана, распознаёт его и показывает перевод в реальном времени. Фокус дипломной части — пользовательский интерфейс, overlay, сценарии взаимодействия и качество фронтенд-опыта.

«Видит текст. Понимает смысл. Переводит мгновенно.»

Реализовали:
Аргаткин А.А.
Коробкин В.Ю.

ПРОБЛЕМА

Языковой барьер любого видимого текста

Текст из видео, игр, интерфейсов, изображений и PDF часто нельзя скопировать напрямую. Пользователь тратит около 1-2 минут на скриншот, OCR, перевод и возврат в исходное окно; из-за переключения между окнами теряется контекст.

Учёба
видео, PDF, презентации без ручного копирования

Работа
документация и зарубежное ПО в контексте окна

Развлечения
игры, субтитры, скриншоты перевод поверх экрана

Фронтенд-боль
много окон, нет overlay, нет быстрого выбора области

Вывод: нужен перевод любого видимого текста прямо на экране - без потери контекста и ручного копирования.

2

ЦЕЛЬ

Разработать программноу обеспечению для захвата, распознавания и перевода текстового контента в реальном времени

ОБЪЕКТ

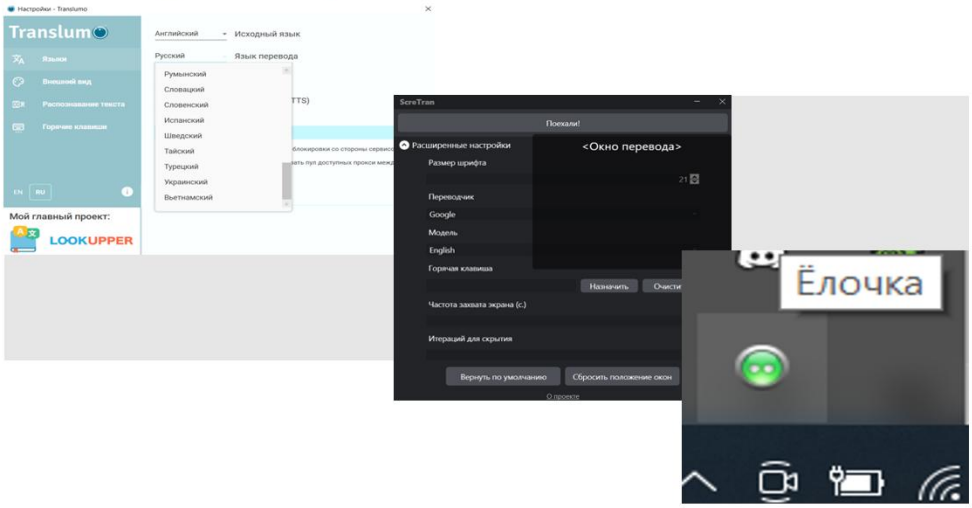
Программное обеспечение для захвата, распознавания и перевода текстового контента

ЗАДАЧИ

- изучить предметную область экранного перевода;
- выполнить анализ существующих аналогов;
- сформулировать требования к фронтенд-части и бекенд;
- определить границы ответственности разрабатываемого интерфейса;
- выбрать технологии и средства разработки;
- спроектировать структуру окон и пользовательских сценариев;
- реализовать основные элементы интерфейса;
- провести проверку работы интерфейса и пользовательских режимов.

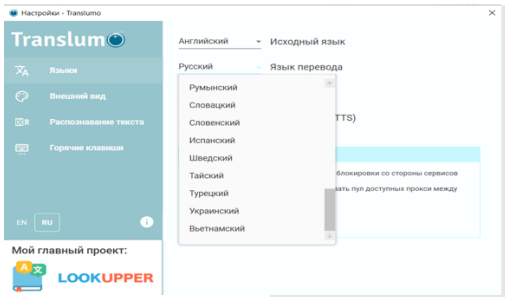
3

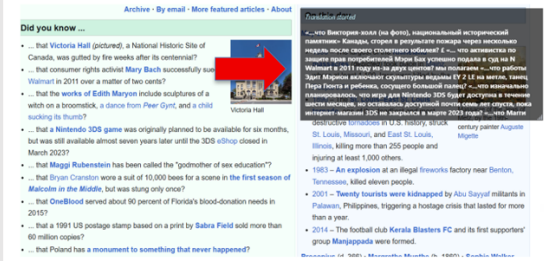
АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ

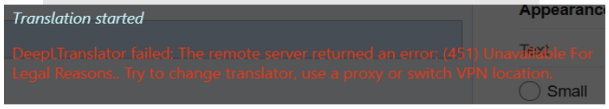


4

АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ

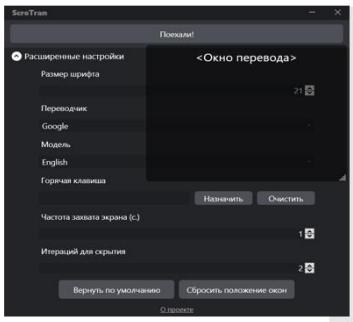


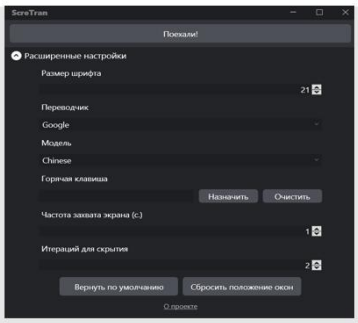




5

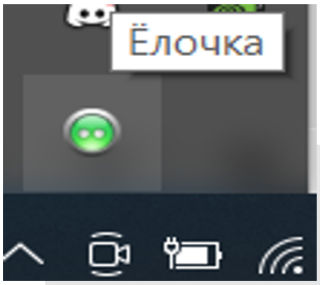
АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ

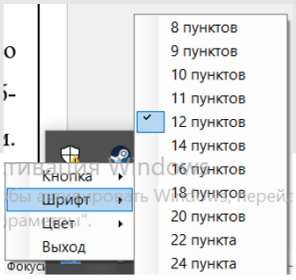


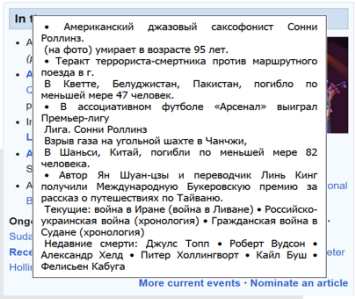


6

АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ







7

АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ

Аналог	Основные возможности	Преимущества	Недостатки для данной задачи
Translumo	Перевод текста в оконном режиме	Скорость перевода, интерфейс на двух языках, поддержка разных моделей OCR	Программные ошибки в определенных кейсах, потребность в постоянном открытии/закрытии приложения, отсутствие overlay
ScreTran	Перевод текста в оконном режиме	Скорость перевода	Громоздкий интерфейс с тремя окнами, отсутствие overlay, малое количество настроек
«Ёлочка»	Одноразовый перевод текста при захвате экрана	Удобно для сайтов, пошаговых игр, скорость перевода	Малое количество настроек, перевод только с английского на русский, отсутствие overlay

SCREEN TRANSLATE: узкий фокус на Windows-сценарии, выбор области, быстрый overlay, окно перевода, горячие клавиши, удобный пользовательский режим.

8

РЕШЕНИЕ

Пользовательский сценарий в интерфейсе

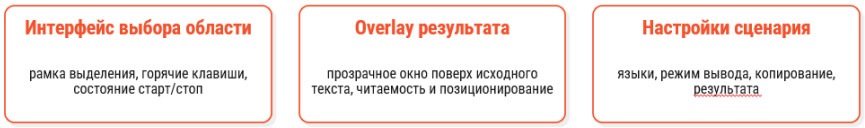
Сценарий: пользователь открывает иностранный интерфейс, видеоурок или PDF и не может выделить текст.

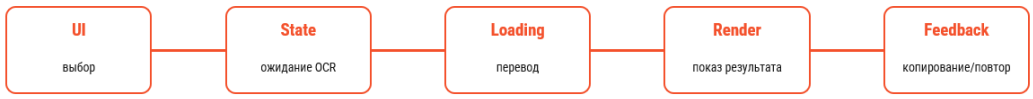
Целевой эффект: перевод фрагмента за 5-10 секунд вместо 1-2 минут ручных действий. Ключевая ценность - быстрый и понятный путь: выбрать область → увидеть перевод → продолжить работу.

9

ФРОНТЕНД-РАЗРАБОТКА

Зона ответственности дипломной части





Итог фронтенд-разработки: пользователь получает быстрый, понятный и ненавязчивый экранный перевод без перехода в сторонние окна.

10

БЭКЕНД-ЧАСТЬ

Зона ответственности дипломной части

Технологический стек

Python / Desktop UI, OCR-модуль, переводчик, локальные настройки, хранение кэша

Приватность

Обрабатывается только выбранный фрагмент

Масштабирование

очередь задач, лимиты Free, контроль затрат API, B2B-профили настроек

Цель backend-слоя: скорость <20 сек., стабильность OCR на типовых экранах и предсказуемая стоимость одного перевода.

11

СТЕК ТЕХНОЛОГИЙ

Python
Язык программирования

Pillow
Подготовка изображения

VS Code
Среда разработки

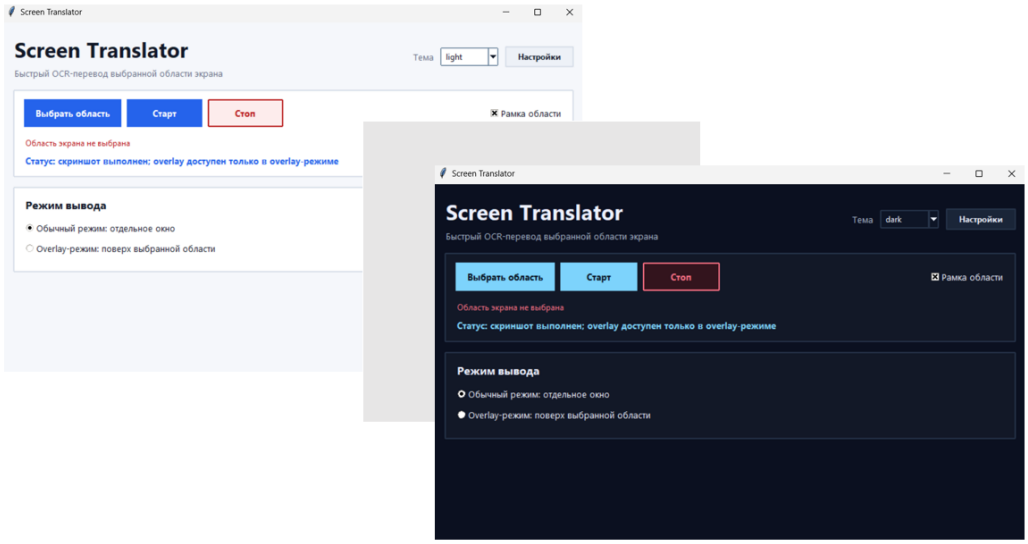
Deep-translator
Машинный перевод

Tkinter
Графический интерфейс

EasyOCR
Распознавание текста

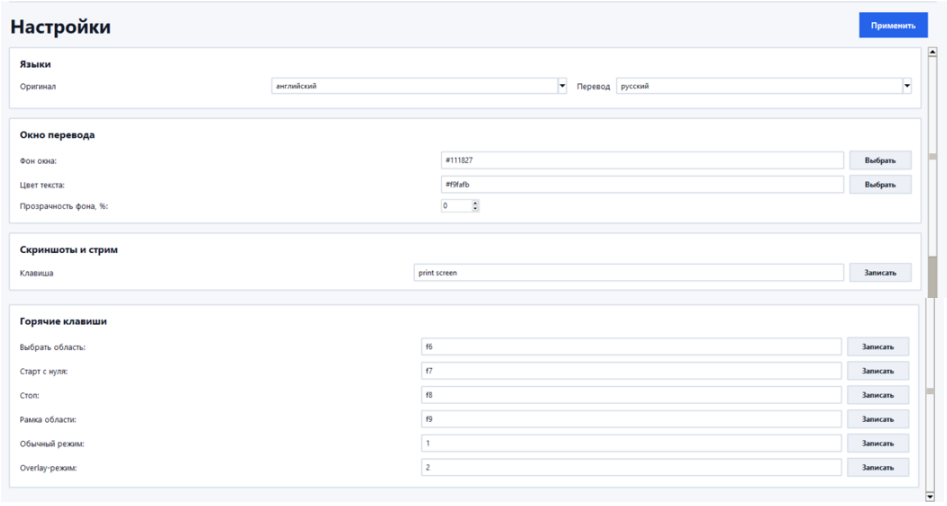
12

ОПИСАНИЕ ПРОГРАММЫ



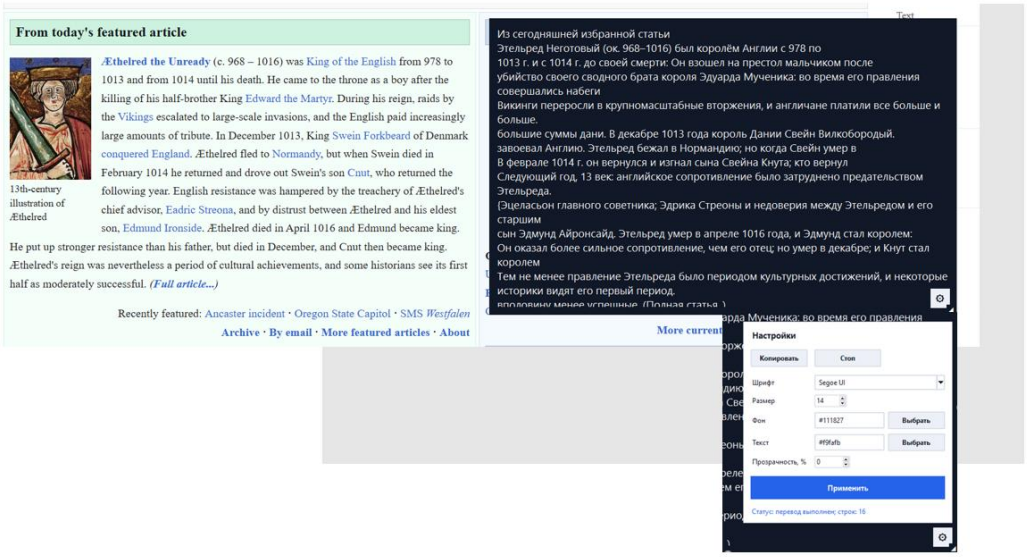
13

ОПИСАНИЕ ПРОГРАММЫ

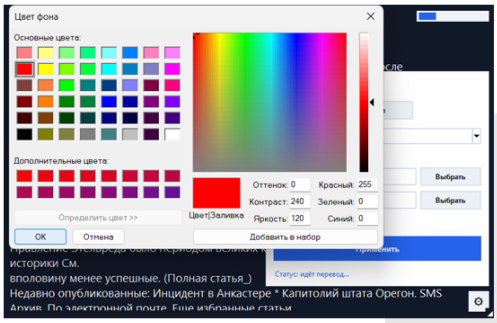


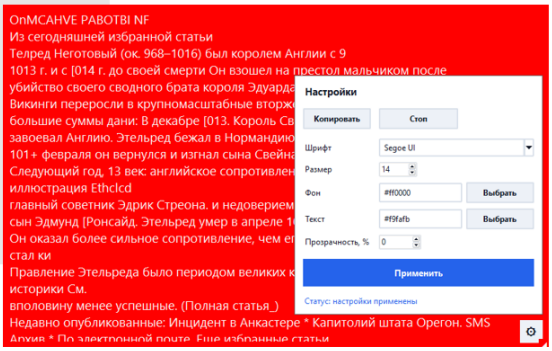
14

ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

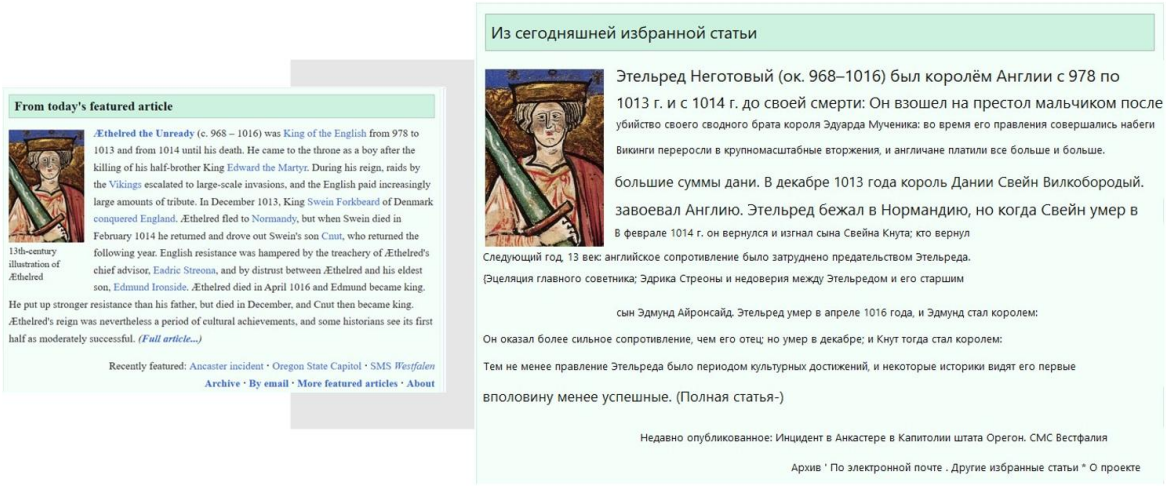


ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ





ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ



From today's featured article

13th-century illustration of Æthelred

Æthelred the Unready (c. 968 – 1016) was King of the English from 978 to 1013 and from 1014 until his death. He came to the throne as a boy after the killing of his half-brother King Edward the Martyr. During his reign, raids by the Vikings escalated to large-scale invasions, and the English paid increasingly large amounts of tribute. In December 1013, King Swein Forkbeard of Denmark conquered England. Æthelred fled to Normandy, but when Swein died in February 1014 he returned and drove out Swein's son Cnut, who returned the following year. English resistance was hampered by the treachery of Æthelred's chief advisor, Eadric Streona, and by distrust between Æthelred and his eldest son, Edmund Ironside. Æthelred died in April 1016 and Edmund became king. He put up stronger resistance than his father, but died in December, and Cnut then became king. Æthelred's reign was nevertheless a period of cultural achievements, and some historians see its first half as moderately successful. ***(Full article...)***

Recently featured: Ancaster incident · Oregon State Capitol · SMS *Westfalen*

Archive · By email · More featured articles · About

Из сегодняшней избранной статьи

Этельред Неготовый (ок. 968–1016) был королём Англии с 978 по 1013 г. и с 1014 г. до своей смерти: Он взошел на престол мальчиком после убийство своего сводного брата короля Эдуарда Мученика: во время его правления совершались набеги Викинги переросли в крупномасштабные вторжения, и англичане платили все больше и больше. большие суммы дани. В декабре 1013 года король Дании Свейн Вилкобородый. завоевал Англию. Этельред бежал в Нормандию, но когда Свейн умер в В феврале 1014 г. он вернулся и изгнал сына Свейна Кнута; кто вернул Следующий год. 13 век: английское сопротивление было затруднено предательством Этельреда. (Эцеляция главного советника; Эдрика Стреоны и недоверия между Этельредом и его старшим

сын Эдмунд Айронсайд. Этельред умер в апреле 1016 года, и Эдмунд стал королем:

Он оказал более сильное сопротивление, чем его отец; но умер в декабре; и Кнут тогда стал королем:

Тем не менее правление Этельреда было периодом культурных достижений, и некоторые историки видят его первые

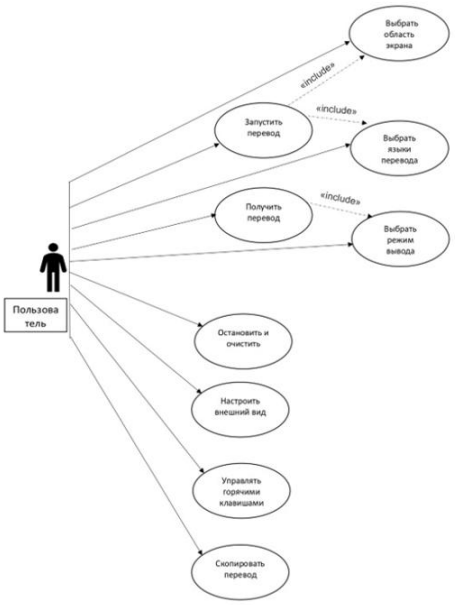
вполовину менее успешные. (Полная статья-)

Недавно опубликованное: Инцидент в Анкастере в Капитолии штата Орегон. СМС Вестфалия

Архив ' По электронной почте . Другие избранные статьи " О проекте

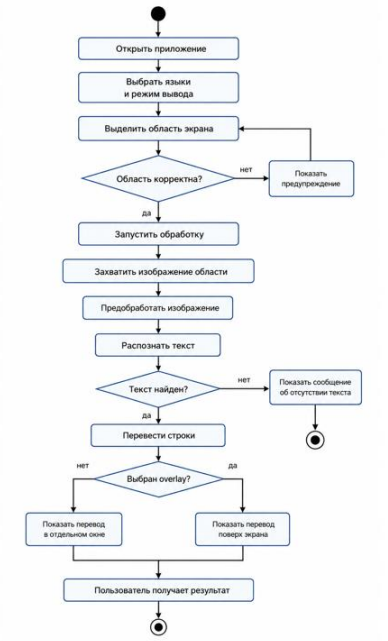
17

ДИАГРАММА ПРЕЦЕДЕНТОВ



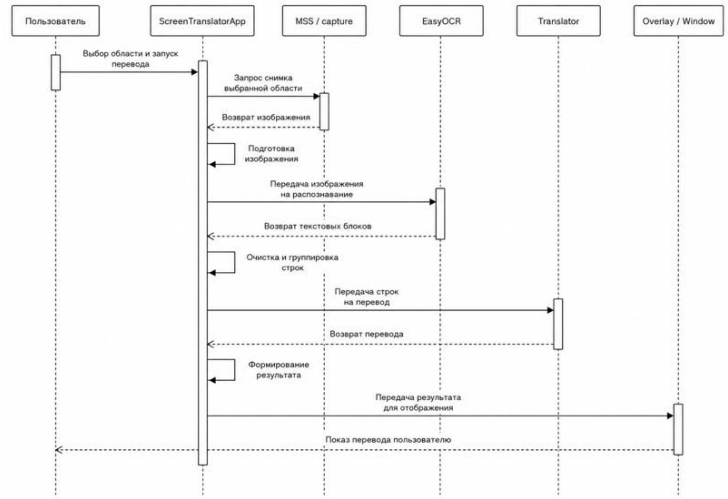
18

ДИАГРАММА ДЕЯТЕЛЬНОСТИ



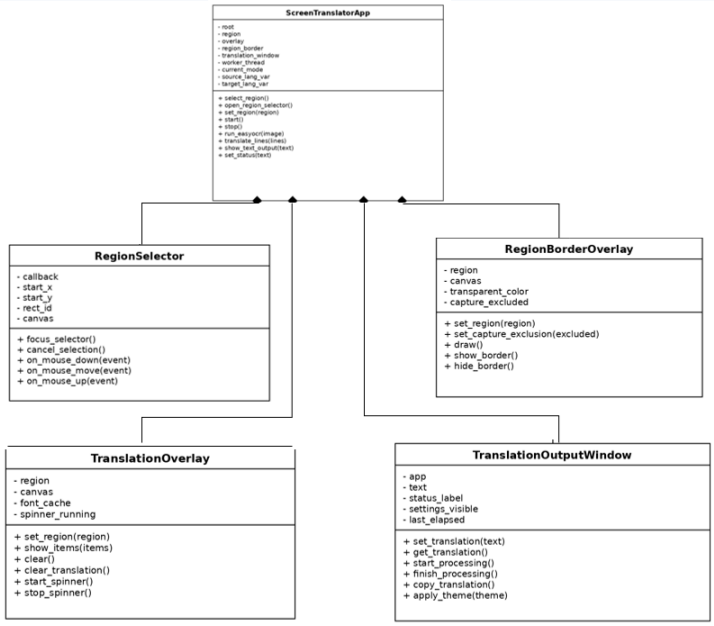
19

ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ



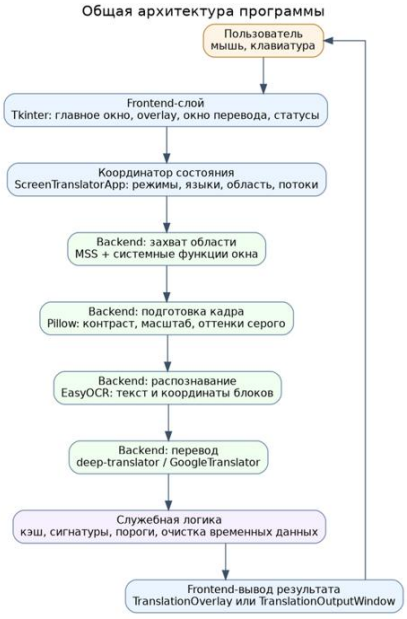
20

ДИАГРАММА КЛАССОВ



21

АРХИТЕКТУРА ПРОГРАММЫ



22

ФИНАНСОВЫЕ ПОКАЗАТЕЛИ

Подписка + лицензии

Показатель	1 год	2 год	3 год
Pro-подписки	60 000 ₽	140 000 ₽	260 000 ₽
Доработки/настройка	60 000 ₽	90 000 ₽	110 000 ₽
B2B-лицензии	100 000 ₽	220 000 ₽	380 000 ₽
Итого выручка	220 000 ₽	450 000 ₽	750 000 ₽

23

ЗАКЛЮЧЕНИЕ

• **На данный момент:** создана рабочая версия приложения. Реализованы выбор области экрана, окно результата и overlay-отображение перевода. Интерфейс понятен и готов к дальнейшему тестированию.

• **Планы на ближайшее время:** доработать внешний вид overlay, добавить пресеты областей, провести тестирование на 50-100 пользователях.

• **Дальнейшее развитие (год 1-3):** выпустить стабильную версию для Windows, добавить историю переводов, расширить настройки интерфейса и подготовить продукт к продаже по подписке и B2B-лицензиям.

24

БЛАГОДАРИМ ЗА ВНИМАНИЕ