

ДЕПАРТАМЕНТ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
ТОМСКОЙ ОБЛАСТИ  
ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
«ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Специальность 09.02.07 Информационные системы и программирование

УТВЕРЖДАЮ  
Заместитель директора по  
учебно-методической работе  
Е.А. Родзик  
«\_\_» \_\_\_\_\_ 2023 г.

РАЗРАБОТКА TELEGRAM-БОТА С МОДУЛЕМ НАСТРОЙКИ ДЛЯ ВЕБ-  
САЙТА ОБЪЕДИНЕНИЯ ТОМСКИХ ВУЗОВ

Пояснительная записка  
к дипломному проекту  
ДП.23.09.02.07.603.05.ПЗ

Студент		
«__» _____ 2023 г.	_____	П.Д. Левицкий
Руководитель		
«__» _____ 2023 г.	_____	Д.В. Муха
Консультант по специальности		
«__» _____ 2023 г.	_____	Д.В. Смоляков
Консультант по экономической части		
«__» _____ 2023 г.	_____	О.М. Керб
Нормоконтролёр		
«__» _____ 2023 г.	_____	А.Ю. Маюнова

ДОПУСТИТЬ К ЗАЩИТЕ

Председатель ПЦК		
«__» _____ 2023 г.	_____	А.Ю. Маюнова

Томск 2023

**ГРАФИК**

выполнения дипломного проекта

Группа 603

Дипломант Левицкий П. Д.

Срок сдачи проекта с отзывом руководителя	<b>01.06.2023</b>
---	-------------------

Наименование разделов проекта	Объем работы в %	Срок исполнения по плану	% выполнения на 20.04.2023	% выполнения на 15.05.2023	% выполнения на 25.05.2023
Общая часть	7%	20.04.2023	100%		
Специальная часть	10%	20.04.2023	100%	100%	
Организационно- экономическая часть	18%	15.05.2023	15%	60%	100%
Графическая часть	20%	16.05.2023	80%	100%	
Экспериментальная часть	20%	21.05.2023	75%	100%	
Оформление пояснительной записки	25%	25.05.2023	50%	90%	100%
Сдача проекта в готовом виде руководителю	100%	31.05.2023	100%	100%	100%

Дни консультаций	
Дата	время
20.04.2023	8:30
25.04.2023	8:30
29.04.2023	8:30
13.05.2023	8:30
20.05.2023	8:30
27.05.2023	8:30
31.05.2023	17:50

Выполнение проекта		
дата	%	подпись руководителя
07.05.2023	50	
15.05.2023	70	
20.05.2023	90	
30.05.2023	100	

**ПРИМЕЧАНИЕ:** Составление пояснительной записки производится параллельно с выполнением соответствующих разделов проекта.

Дипломант: Левицкий П. Д.

«\_\_» \_\_\_\_\_ 2023 г.

Руководитель: Муха Д. В.

«\_\_» \_\_\_\_\_ 2023 г.

ДЕПАРТАМЕНТ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
ТОМСКОЙ ОБЛАСТИ  
ОГБПОУ «ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

*Специальность 09.02.07 – «Информационные системы и программирование»*

СОГЛАСОВАНО  
заведующий отделением ПО  
\_\_\_\_\_ В.Е. Курочкин  
«\_\_» \_\_\_\_\_ 2023 г.

**ЗАДАНИЕ**

на дипломное проектирование

студенту группы 603

Левицкий Павел Дмитриевич  
(Фамилия, Имя, Отчество)

Дата выдачи: 10 апреля 2023 г.

Дата окончания: 01 июня 2023 г.

**I ТЕМА ДИПЛОМНОГО ПРОЕКТА**

Разработка telegram-бота с модулем настройки для веб-сайта объединения  
ТОМСКИХ ВУЗОВ.

**II ТЕХНИЧЕСКИЕ УСЛОВИЯ**

Процессор 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz, оперативная память 16гб  
DDR4 SDRAM, операционная система Windows 11 Домашняя для одного  
языка

### III СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

- 1) Общая часть
- 2) Специальная часть
- 3) Экономическая часть
- 4) Заключение
- 5) Перечень используемых источников
- 6) Приложение А. Листинг кода
- 7) Приложение Б. Инструкция пользователя
- 8) Приложение В. Диаграммы состояний

### IV ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ ПРОЕКТА

C# v10, .NET v6.0, Microsoft Visual studio 2022, DB Browser for SQLite, draw.io, HtmlAgilityPack v1.11.46, MaterialDesignThemes v4.8.0, Microsoft.Data.Sqlite v7.0.5, Telegram.Bot v18.0.0.

### V ЭКОНОМИЧЕСКАЯ ЧАСТЬ ПРОЕКТА

Расчет затрат на разработку программы и решение задачи на ЭВМ, Расчет экономического эффекта и определение срока окупаемости

### VI ЛИТЕРАТУРА

Рихтер, Д. CLR VIA C# — Текст: электронный [электронная книга]. — 2022 (дата обращения: 26.04.2023). Мартин, Р. Чистый код. Создание, анализ и рефакторинг — Текст: электронный [электронная книга]. — 2022 (дата обращения: 20.05.2023).

## VII СРОКИ ВЫПОЛНЕНИЯ ПРОЕКТА

В соответствии с графиком выполнения дипломного проекта.

Руководитель дипломного проекта \_\_\_\_\_ / Д.В. Муха  
(подпись) (расшифровка подписи)

Дипломное задание рассмотрено на заседании предметной цикловой комиссии

Протокол № \_\_\_\_ от « \_\_\_\_ » \_\_\_\_\_ 2023 г.

Председатель предметной цикловой комиссии \_\_\_\_\_ А.Ю. Маюнова  
(подпись)

Задание к исполнению получил(а) студент \_\_\_\_\_ / П. Д. Левицкий  
(подпись) (расшифровка подписи)

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
1 ОБЩАЯ ЧАСТЬ .....	5
1.1 Анализ предметной области .....	5
1.2 Средства и среда разработки.....	7
2 СПЕЦИАЛЬНАЯ ЧАСТЬ.....	9
2.1 Описание требований к информационной системе.....	9
2.2 Диаграмма вариантов использования .....	12
2.3 Схема базы данных .....	14
2.4 Словарь данных .....	15
2.5 Пользовательские сценарии .....	19
2.6 Прототипы интерфейсов .....	21
3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ.....	29
3.1 Расчет затрат на разработку программы и решение задачи на ЭВМ.....	29
3.2 Расчет экономического эффекта и определение срока окупаемости.....	35
ЗАКЛЮЧЕНИЕ .....	39
ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ .....	40
ПРИЛОЖЕНИЕ А. Листинг программы .....	42
ПРИЛОЖЕНИЕ Б. Инструкция пользователя .....	96
ПРИЛОЖЕНИЕ В. Диаграммы состояний.....	111

## ВВЕДЕНИЕ

ТУСУР - Томский государственный университет систем управления и радиоэлектроники, признанный лидер в сфере подготовки квалифицированных кадров для высокотехнологичных отраслей экономики, аэрокосмического и оборонного комплексов страны, внедряющий инновационные образовательные и исследовательские программы, прикладные разработки новой техники, аппаратуры и систем управления. Университет уверенно держит первенство в реализации программ инновационного развития, выпускники ТУСУРа составляют кадровую основу многих предприятий как в России, так и за рубежом. Руководство университета участвовало в создании проекта "Консорциум "Объединение томских вузов"".

В состав данного консорциума на настоящий момент входят ТУСУР, ТПУ, ТГПУ, ТГАСУ и ТГУ.

Проект призван упростить поступление абитуриентов в определенный перечень высших учебных заведений.

Основное предназначение сайта проекта - подбор направлений обучения и подача заявки на поступление на выбранное направление.

Направления обучения содержатся только от тех университетов, которые входят в состав консорциума.

Стоит сказать, что сам проект ориентирован также и на иностранных абитуриентов, по этой причине на веб-сайте присутствует страница с прохождением тестирования на знание русского языка.

Среднестатистический студент в РФ имеет возраст от 18 до 23 лет. Логично предположить, что люди в таком возрастном диапазоне активно используют мессенджеры, либо социальные сети.

Как руководитель проекта, университет стремится к увеличению охвата аудитории в лице потенциальных абитуриентов, по этой причине ему необходимо получить готовое решение, выполняющее данную задачу.

Было предложено несколько возможных вариантов реализации программного продукта, выполняющего задачу расширения аудитории, из них были следующие: создание чат-бота на готовой платформе, позволяющей провести интеграцию с Telegram и непосредственное написание Telegram-бота, используя стандартные средства разработки.

Первый вариант не подходит по нескольким причинам - подобные платформы не предполагают написание программного кода, но лишь использование визуального конструктора с весьма ограниченным функционалом, и они не позволяют использовать динамические источники данных.



# 1 ОБЩАЯ ЧАСТЬ

## 1.1 Анализ предметной области

Работа сервиса напрямую зависит от актуальности данных, которыми располагает сервис. В противном случае использование сервиса пользователями будет не только бесполезным, но и вредным. По этой причине продукт должен собирать данные с веб-сайта проекта при каждом запуске и иметь возможность вручную обновить собранные данные.

Основная атомарная единица, хранящаяся на ресурсе – карточка направления.

В свою очередь карточка направления содержит в себе такие данные, как название университета, название направления, уровень обучения, форма обучения, код программы, продолжительность, квалификация, язык обучения, ФИО куратора, телефон, почта и стоимость за год обучения.

Также на веб-сайте проекта доступна форма выбора направления по уровню обучения (квалификации) и направлению, результатом является набор карточек направлений, при нажатии кнопки «Поступить» на одной из них открывается страница, позволяющая оставить заявку на поступление.

Исходя из описанного выше необходимо написать Telegram-бота, генерирующего карточки направлений, отбирающиеся из общего числа полученных карточек в соответствии с выбором потенциального абитуриента и отправляющего ее в чат. К такому сообщению должна быть прикреплена кнопка для обратной связи.

Задачи дипломного проекта:

- 1) Исследование технологий по созданию ботов;
- 2) Обзор решений для создания чат-ботов;
- 3) Анализ конкурентов;
- 4) Формулирование сценария работы бота;
- 5) Выбор платформы и языка программирования;
- 6) Исследование доступных библиотек;
- 7) Анализ возможностей доступных библиотек;
- 8) Реализация программного кода;
- 9) Исправление ошибок логики и ошибок интерфейса;
- 10) Составление документации.

## 1.2 Средства и среда разработки

На этапе проектирования продукта были использовано средство draw.io – ресурс для отрисовки диаграмм, обладает всем необходимым функционалом, таким, как обширная коллекция фигур и возможностью экспорта диаграмм в .png, прост в использовании и обладает кроссплатформенностью, применялся для отрисовки логической модели базы данных и построении остальных диаграмм.

На этапе разработки программного кода были использованы следующие средства:

Microsoft Visual studio 2022 - интегрированная среда разработки, позволяющая написание программного кода, предоставляющая средства отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для повышения качества процесса разработки.

Среда разработки располагает редактируемым дизайном, огромным количеством расширений и приятным UI.

В качестве языка программирования был выбран C#, являющийся объектно- и компонентно-ориентированным языком программирования. Обладает рядом положительных качеств:

- 1) C# – объектно-ориентированный, простой и в то же время мощный язык программирования, позволяющий разработчикам создавать многофункциональные приложения;
- 2) C# относится к языкам компилируемого типа, поэтому он обладает всеми преимуществами таких языков;
- 3) C# объединяет лучшие идеи современных языков программирования Java, C++, Visual Basic и т.д;

4) Из-за большого разнообразия синтаксических конструкций и возможности работать с платформой .Net, С# позволяет быстрее, чем любой другой язык, разрабатывать программные решения;

5) С# отличается надежностью и наличием большого количества синтаксических конструкций.

Nuget-пакет HtmlAgilityPack v1.11.46 – библиотека, необходимая для работы парсера HTML-страницы средствами С#. Поддерживает ХРАТН, необходимый для парсинга HTML-документа.

SQLite —компактная встраиваемая СУБД. Удобно использовать в случаях, когда не требуется разделенное хранение данных по типу клиент-сервер, так как движок является не отдельно работающим процессом, а библиотекой, с которой выполняется построение программы. В свою очередь это позволяет время отклика и упрощает программу в целом.

Nuget-пакет Telegram.Bot v18.0.0 – предоставляет возможность работы с Telegram Bot API и непосредственного написания логики бота и подключения к боту по токену.

Nuget-пакет Microsoft.Data.Sqlite v7.0.4 – легковесный ADO.NET провайдер, предоставляет возможность работы с SQLite.

## 2 СПЕЦИАЛЬНАЯ ЧАСТЬ

### 2.1 Описание требований к информационной системе

Telegram-бот должен иметь возможность настройки посредством модуля настройки, реализованного в виде настольного приложения.

Информационная система должна предоставлять возможность сбора, хранения и выдачи данных о карточках специальности посредством опроса потенциального абитуриента посредством Telegram -бота.

Администратор должен иметь возможность запускать и останавливать бота, вручную запускать парсер, просматривать различную статистику по работе бота, просматривать генерируемые ИС файлы журналов, просматривать полученные ботом сообщения в реальном времени, устанавливать URL сайта, с которого собираются данные, устанавливать, изменять и отключать пароль, изменять настройки ИС (пути к файлам журналов, токен бота, URL страницы), просматривать статистику и осуществлять поиск по программам университетов, просматривать и осуществлять поиск по карточкам направлений, просматривать статистику по длительности сессий, а так же экспортировать данные по карточкам направлений и программам университетов.

Пользователями информационной системы (Далее - ИС) являются: администратор, пользователь.

Функционал администратора должен быть реализован непосредственно на стороне ИС, функционал пользователя - в виде чата с Telegram-ботом.

Требуемый функционал пользователя:

- 1) Переход на веб-сайт проекта;
- 2) Переход на веб страницу прохождения тестирования на знание русского языка;
- 3) Выбор уровня обучения;
- 4) Выбор университета;
- 5) Выбор программы обучения;
- 6) Получение полных данных о выбранной программе обучения;
- 7) Переход в почтовое приложение или на веб-сайт для последующей связи с куратором по почте, указанным в карточке;
- 8) Выход в главное меню.

Требуемый функционал администратора:

- 1) Установка пароля, если это первый запуск приложения;
- 2) Смена существующего пароля;
- 3) Отключение установленного пароля;
- 4) Вход по паролю, если включен;
- 5) Запуск бота;
- 6) Остановка бота;
- 7) Просмотр полученных ботом сообщений;
- 8) Просмотр сведений об ошибках Telegram API;
- 9) Ручной запуск парсера;
- 10) Просмотр сведений о полученных карточках направлений;
- 11) Умный поиск по названию направления среди карточек направлений;
- 12) Умный поиск по названию университета среди карточек направлений;
- 13) Сортировка таблицы с карточками направлений;

- 14) Просмотр статистики направлений по университетам
- 15) Умный поиск по названию университета среди статистики программ по университетам;
- 16) Умный поиск по количеству направлений среди статистики программ по университетам;
- 17) Сортировка таблицы со статистикой направлений по университетам
- 18) Установка URL сайта, с которого парсер будет собирать данные;
- 19) Установка токена бота, к которому будет подключаться ИС;
- 20) Установка пути для экспортирующихся отчетов;
- 21) Установка пути для журналов;
- 22) Очистка окна живого журнала.

## 2.2 Диаграмма вариантов использования

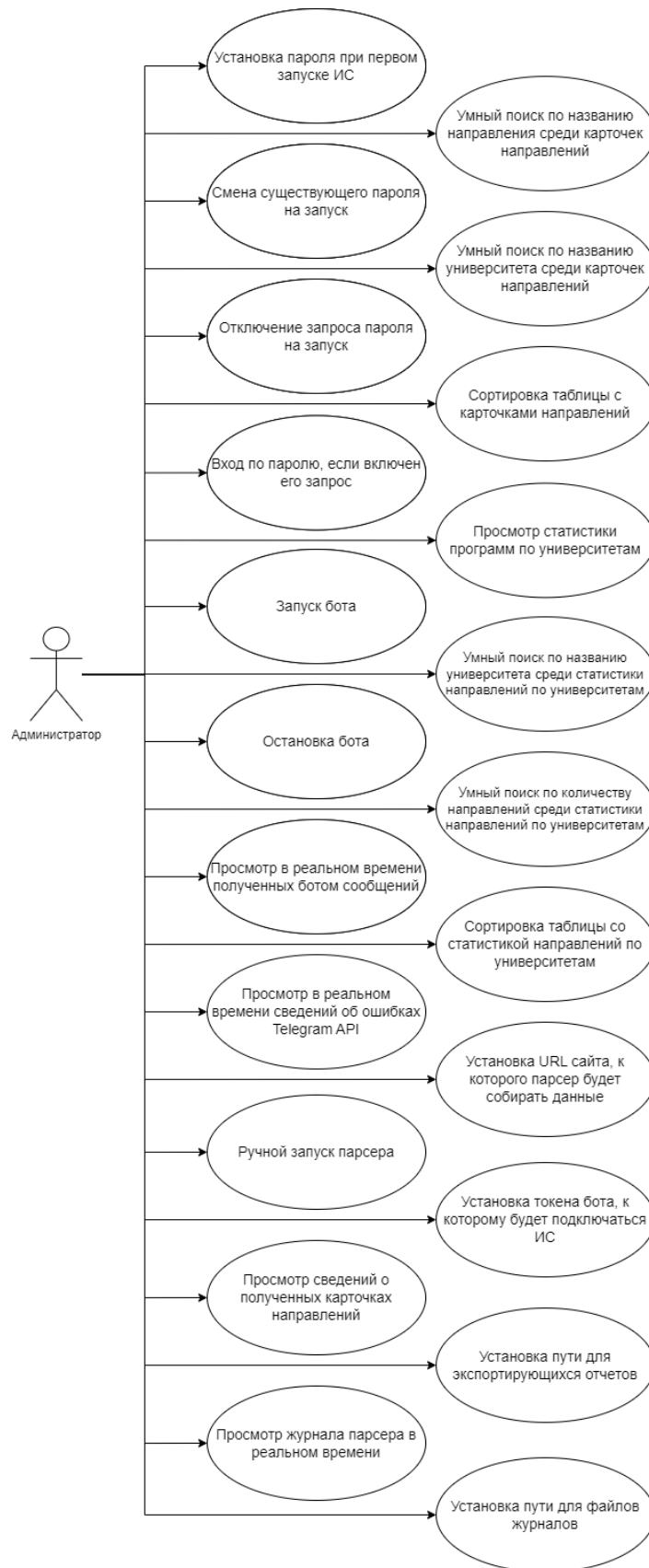


Рисунок 1 – Диаграмма вариантов использования для администратора



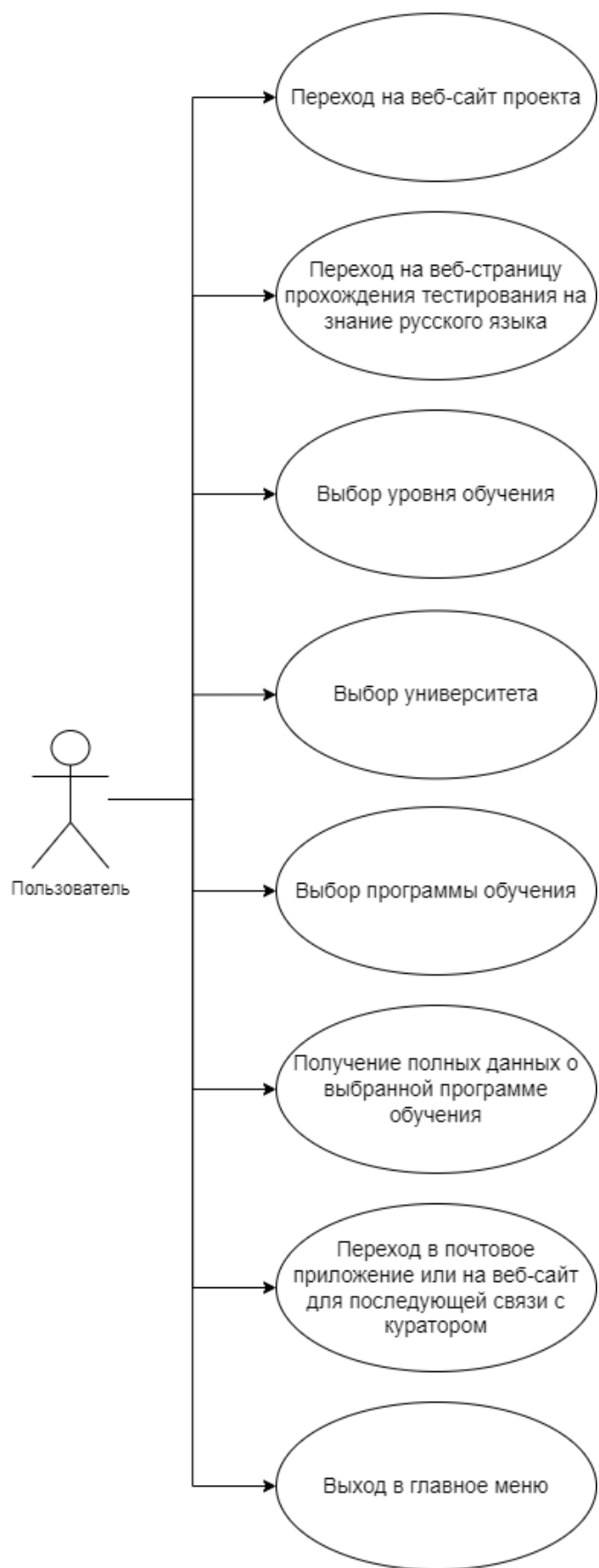


Рисунок 2 – Диаграмма вариантов использования для пользователя

## 2.3 Схема базы данных

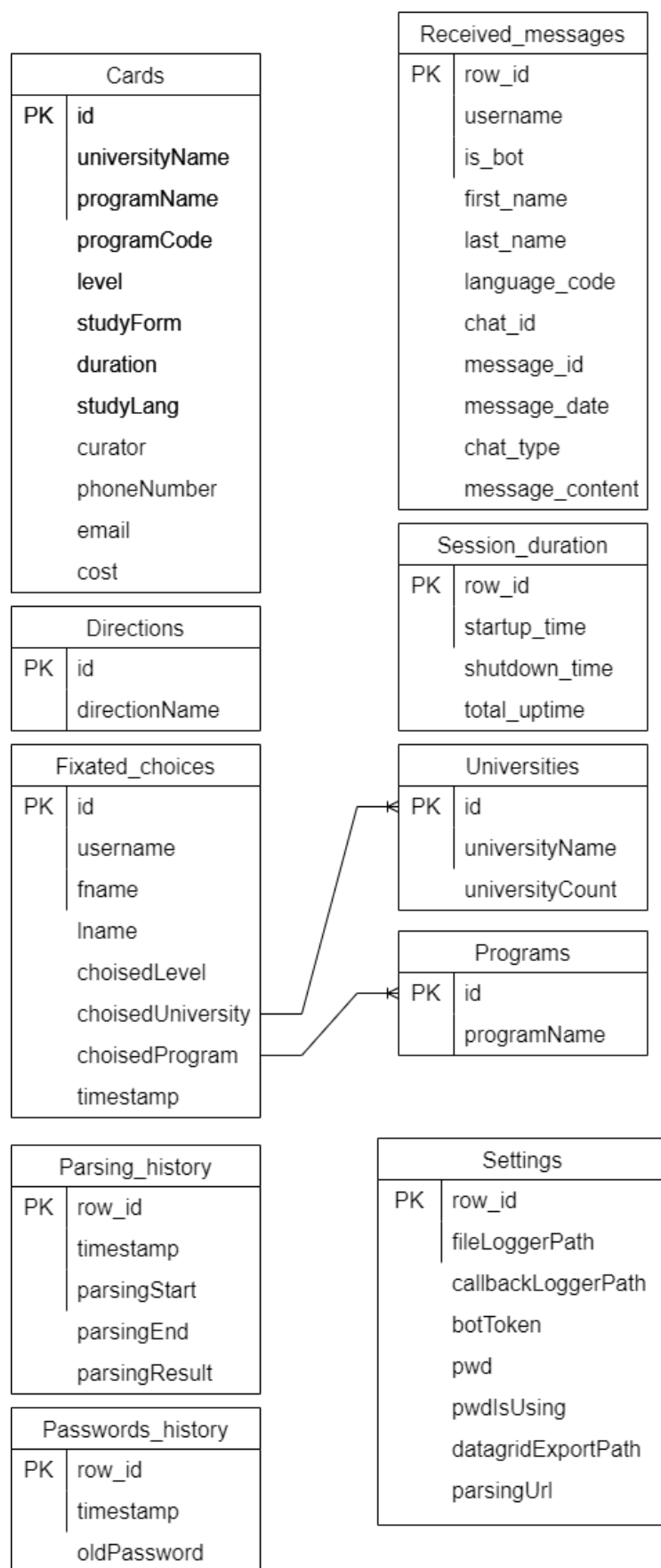


Рисунок 3 – Схема базы данных

## 2.4 Словарь данных

Таблица 1 – Cards (Карточки направлений)

Название поля	Тип данных	Описание
id (PK)	INT	Идентификатор карточки (уникальный)
universityName	TEXT	Название университета
programName	TEXT	Название программы
programCode	TEXT	Код программы
level	TEXT	Уровень обучения
studyForm	TEXT	Форма обучения
duration	TEXT	Длительность
studyLang	TEXT	Язык обучения
curator	TEXT	Куратор
phoneNumber	TEXT	Номер телефона
email	TEXT	Почта
cost	TEXT	Стоимость обучения

Таблица 2 – Directions (Направления)

Название поля	Тип данных	Описание
id (PK)	INT	Идентификатор направления (уникальный)
directionName	TEXT	Название направления (уникальное)

Таблица 3 – Fixated\_choices (Зафиксированные выборы пользователей)

Название поля	Тип данных	Описание
id (PK)	INT	Идентификатор набора выборов (уникальный)
username	TEXT	Никнейм пользователя
fname	TEXT	Имя пользователя
lname	TEXT	Фамилия пользователя
choisedLevel	TEXT	Выбранный уровень обучения
choisedUniversity	TEXT	Выбранный университет
choisedProgram	TEXT	Выбранная программа
timestamp	TEXT	Время фиксации результата

Таблица 4 – Parsing\_history (История парсинга)

Название поля	Тип данных	Описание
row_id (PK)	INT	Идентификатор строки (уникальный)
timestamp	TEXT	Время, затраченное на парсинг
parsingStart	TEXT	Время начала парсинга
parsingEnd	TEXT	Время конца парсинга
parsingResult	INT	Результат парсинга (количество полученных карточек)

Таблица 5 – Passwords\_history (История установленных паролей)

Название поля	Тип данных	Описание
row_id (PK)	INT	Идентификатор строки (уникальный)
timestamp	TEXT	Дата смены пароля
oldPassword	TEXT	Неактуальный на момент записи пароль

Таблица 6 – Programs (Программы обучения)

Название поля	Тип данных	Описание
id (PK)	INT	Идентификатор программы обучения (уникальный)
programName	TEXT	Названия программы

Таблица 7 – Received\_messages (Принятые ботом сообщения)

Название поля	Тип данных	Описание
row_id (PK)	INT	Идентификатор строки (уникальный)
username	TEXT	Никнейм пользователя
is_bot	TEXT	Пользователь – бот?
first_name	TEXT	Имя пользователя
last_name	TEXT	Фамилия пользователя
language_code	TEXT	Код языка пользователя
chat_id	TEXT	Идентификатор чата
message_id	TEXT	Идентификатор принятого сообщения
message_date	TEXT	Дата получения ботом сообщения
chat_type	TEXT	Тип чата с пользователем
message_content	TEXT	Содержимое принятого сообщения

Таблица 8 – Session\_duration (Длительность сессий)

Название поля	Тип данных	Описание
row_id (PK)	INT	Идентификатор строки (уникальный)
startup_time	TEXT	Дата и время запуска бота
shutdown_time	TEXT	Дата и время остановки бота
total_uptime	TEXT	Интервал работы бота

Таблица 9 – Settings (Настройки)

Название поля	Тип данных	Описание
row_id (PK)	INT	Идентификатор строки (уникальный)
fileLoggerPath	TEXT	Путь к журналу принятых сообщений
callbackLoggerPath	TEXT	Путь к журналу выборов пользователей
botToken	TEXT	Токен telegram-бота
pwd	TEXT	Пароль на вход в GUI
pwdIsUsing	TEXT	Вход по паролю активирован?
datagridExportPath	TEXT	Путь экспорта данных из таблицы карточек направлений
parsingUrl	TEXT	URL, требуемый парсеру

Таблица 10 – Universities (Университеты)

Название поля	Тип данных	Описание
Id (PK)	INT	Идентификатор университета (уникальный)
universityName	TEXT	Название университета
universityCount	INT	Количество направлений

## 2.5 Пользовательские сценарии

### Модуль настройки:

После первого запуска программы откроется окно, предлагающее установить пароль. Для этого необходимо ввести пароль и повторить, после чего нажать кнопку «Продолжить».

Если пароли совпадают между собой, длина более трех символов и такой пароль не был использован ранее - пароль устанавливается, после чего открывается главное окно приложения.

Для использования входа в приложение необходимо на вкладке «Параметры» установить флажок «Использовать этот пароль», и при следующем перезапуске приложение запросит установленный ранее пароль.

Существуют поля для смены пароля. Для смены пароля необходимо ввести новый пароль дважды в соответствующие поля и нажать кнопку «Установить пароль», после этого откроется окно ввода старого пароля. Для смены пароля необходимо указать старый пароль и нажать кнопку «Далее», вследствие чего будет открыто окно настроек с уведомлением об успешной смене пароля на новый.

Также на данной вкладке при первом запуске необходимо настроить параметры, необходимые для работы бота. Их можно установить посредством ввода в соответствующие поля и нажатия соответствующих кнопок записи.

Бота возможно запустить с главной вкладки, но не ранее окончания работы парсера. Для контроля окончания работы парсера необходимо перейти на вкладку «Загрузка данных». Живой журнал даст знать об окончании процедуры парсинга, также есть возможность ручного перезапуска парсера.

Далее перейти на вкладку «Домашняя» и нажать кнопку «Старт», после чего отобразится сообщение о прослушивании бота с информацией о боте и дате старта прослушивания.

При необходимости можно остановить бота, остановить бота и выйти из приложения, очистить окно живого журнала или экспортировать его содержимое.

На вкладке «Загруженные карточки» выводятся данные по всем полученным парсером карточкам направлений, также расположено два поля умного поиска - по университету и по названию направления.

На вкладке «Загруженные университеты» данные по всем полученным парсером университетам и количеством направлений по каждому из них. Здесь также расположено два поля для умного поиска - по названию университета и по количеству направлений.

Telegram-бот:

После перехода в чат с ботом необходимо нажать кнопку «Запустить». Далее бот отправит перечень возможных действий – подбор программы обучения, посещение веб-сайта проекта и переход на страницу проекта.

Для начала подбора программы обучения необходимо нажать на кнопку «Выбор программы обучения». Далее пользователю будут последовательно заданы вопросы, в результате которых запишется его выбор уровня обучения, учебного заведения и программы обучения.

Результатом работы бота является сообщение с исчерпывающей информацией о выбранной программе обучения и кнопка для связи с приемной комиссией, отвечающей за поступление на данную программу обучения.



## 2.6 Прототипы интерфейсов

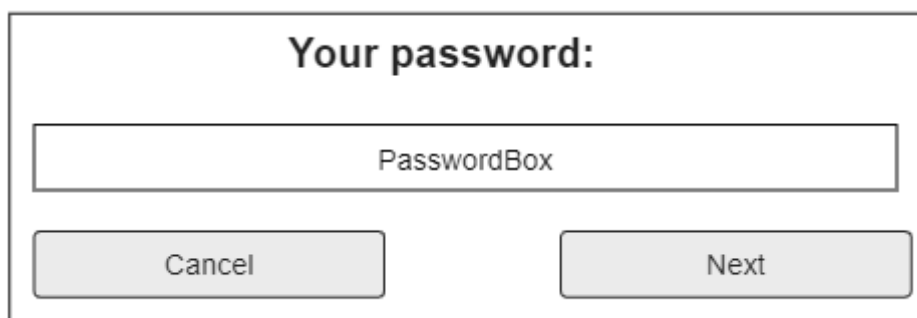
Окно установки пароля (Рисунок 4).



The image shows a rectangular window titled "Set a new password:". Inside the window, there are two identical text input fields, each labeled "PasswordBox", stacked vertically. Below these fields, there are two buttons: "Cancel" on the left and "Next" on the right. The buttons are light gray with rounded corners.

Рисунок 4 – Окно установки пароля

Окно входа (Рисунок 5).



The image shows a rectangular window titled "Your password:". Inside the window, there is a single text input field labeled "PasswordBox". Below this field, there are two buttons: "Cancel" on the left and "Next" on the right. The buttons are light gray with rounded corners.

Рисунок 5 – Окно входа

Вкладка «Домашняя» основного окна (Рисунок 6).

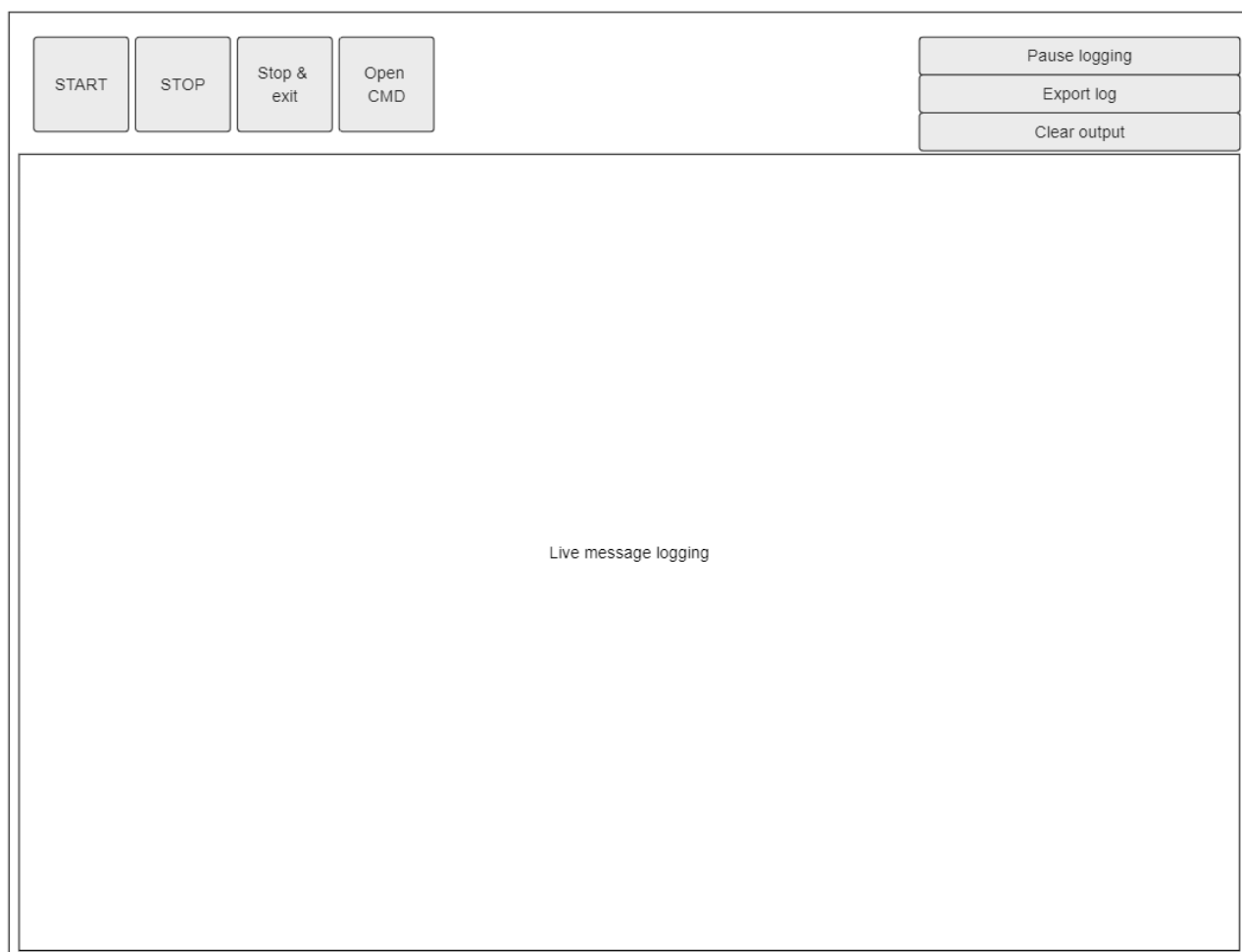


Рисунок 6 – Общий вид вкладки «Домашняя»

Вкладка «Загрузка данных» основного окна (Рисунок 7).

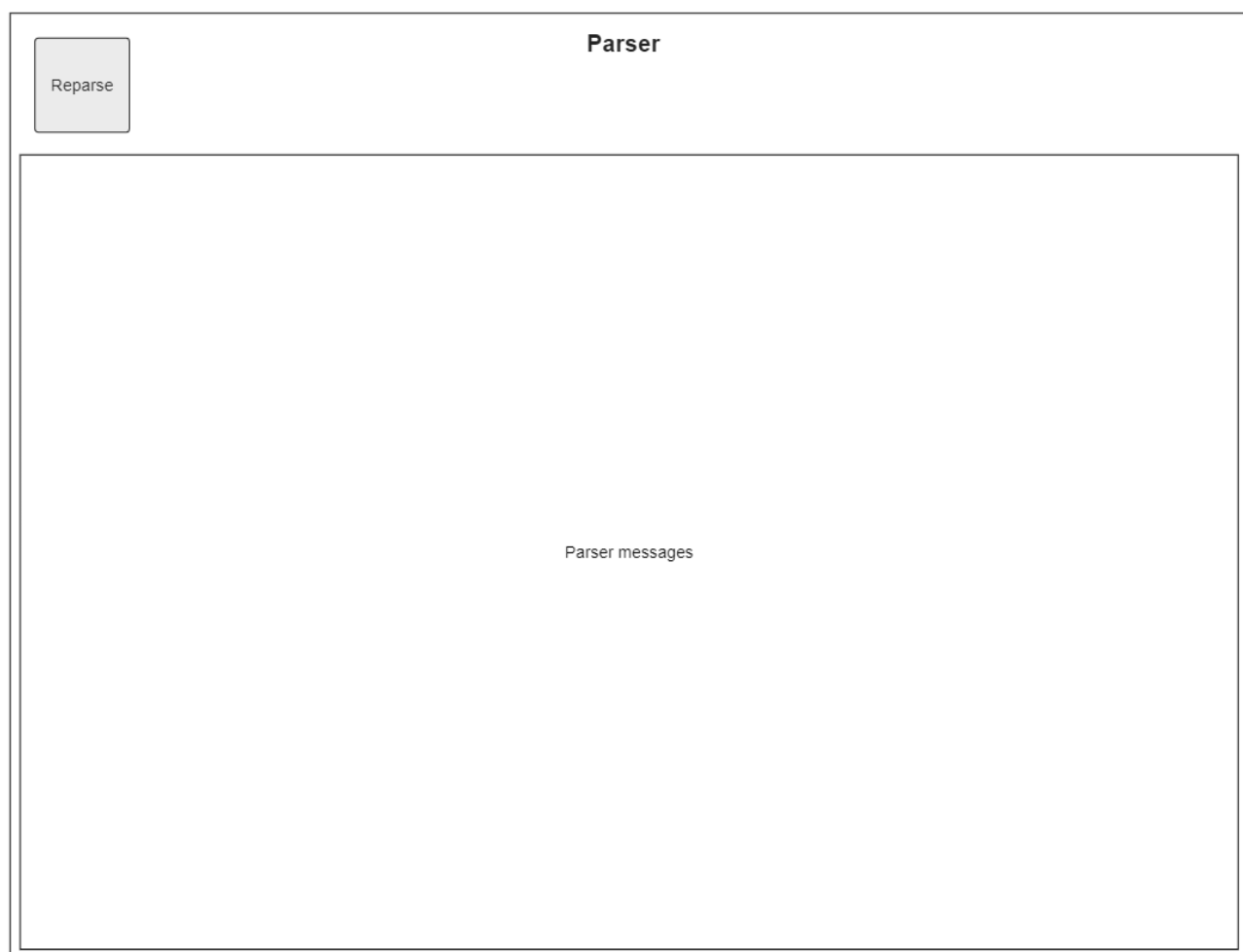


Рисунок 7 – Общий вид вкладки «Загрузка данных»

Вкладка «Загруженные карточки» основного окна (Рисунок 8).

Поиск по названию программы :

Поиск по университету :

Название университета	Название программы	Код программы	Уровень обучения	Форма обучения	Срок обучения	Язык обучения	Куратор	Телефон	Почта	Стоимость
DataGrid										

Рисунок 8 – Общий вид вкладки «Загруженные карточки»

Вкладка «Загруженные университеты» основного окна (Рисунок 9).

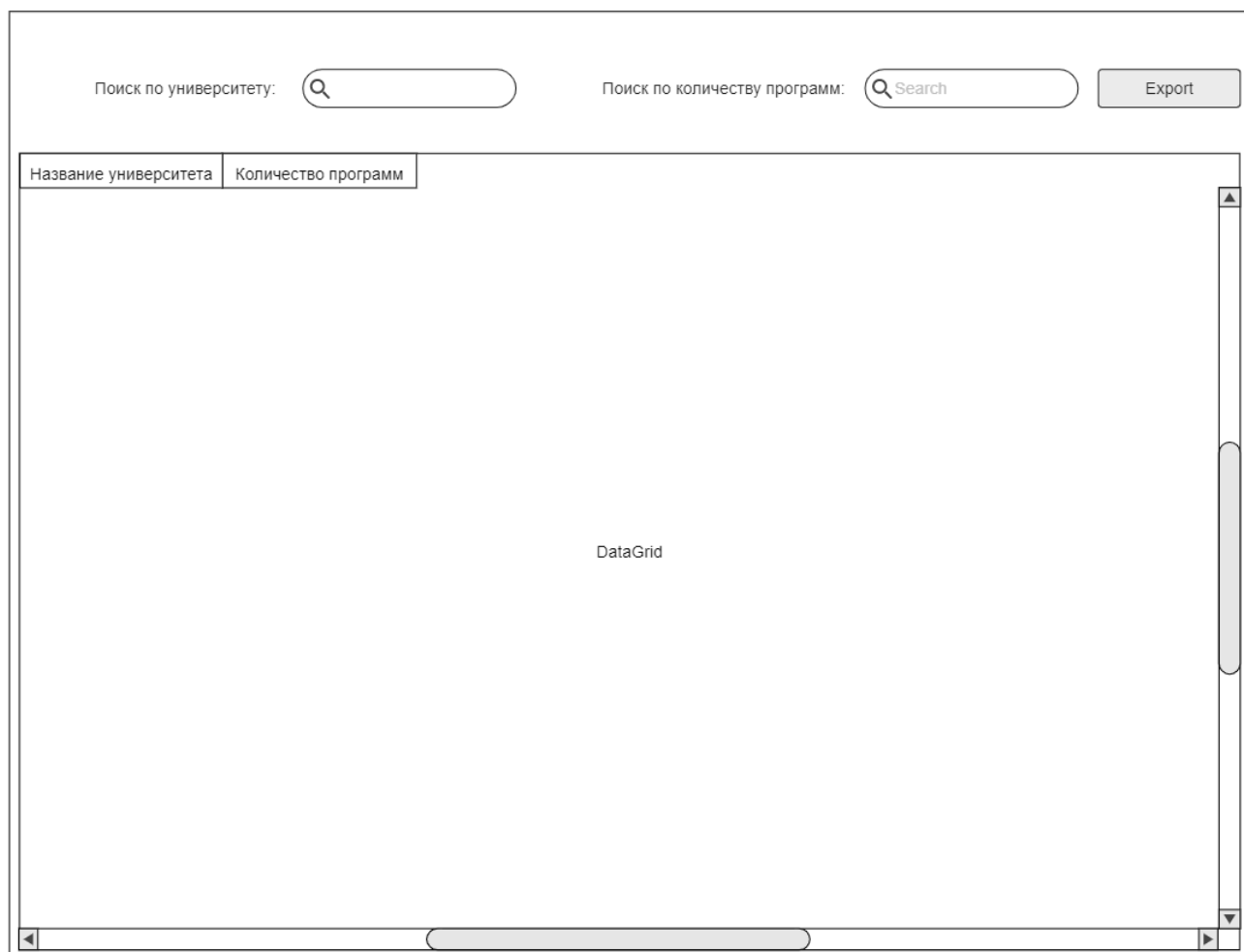


Рисунок 9 – Общий вид вкладки «Загруженные университеты»

Вкладка «Параметры» основного окна (Рисунок 10).

**Settings of the BotServer**

Bot token:	<input type="text" value="TextBox"/>	<input type="button" value="Set bot token"/>
Message log path:	<input type="text" value="TextBox"/>	<input type="button" value="Set message log path"/>
Cards export path:	<input type="text" value="TextBox"/>	<input type="button" value="Set export path"/>
Choices log path:	<input type="text" value="TextBox"/>	<input type="button" value="Set choices log path"/>
Parsing URL:	<input type="text" value="TextBox"/>	<input type="button" value="Set parsing URL"/>
Enter password:	<input type="text" value="TextBox"/>	<input checked="" type="checkbox"/> Use this password
Repeat password:	<input type="text" value="TextBox"/>	<input type="button" value="Set password"/>

Рисунок 10 – Общий вид вкладки «Параметры»

Главное меню Telegram-бота (Рисунок 11).

**Приветствие**

**Выбрать программу обучения**

Посетить веб-сайт проекта	Пройти тест на знание русского языка
---------------------------	--------------------------------------

Рисунок 11 – сообщение бота с предложением действий

Выбор уровня подготовки (Рисунок 12).

Выберите желаемый уровень подготовки:		
Уровень 1	Уровень 2	Уровень 3
Домой		

Рисунок 12 - Сообщение бота с выбором уровня подготовки

Выбор университета (Рисунок 13).

Выберите университет:			
Университет 1	Университет 2	Университет 3	Университет 4
Университет 5	Университет 6	Университет 7	Университет 8
Домой			

Рисунок 13 – Сообщение бота с выбором университета

Выбор направления (Рисунок 14).

Подобранные программы обучения:			
Направление 1	Направление 2	Направление 3	Направление 4
Направление 5	Направление 6	Направление 7	Направление 8
Домой			

Рисунок 14 – Сообщение бота с выбором программы обучения

Финальное сообщение, содержащее информацию о выбранном направлении (Рисунок 15).

Мы подобрали для вас следующее направление:
Университет
Программа
Код программы
Уровень образования
Форма обучения
Длительность обучения
Язык обучения
Куратор
Номер телефона
Почта
Стоимость обучения
Связаться
Домой

Рисунок 15 – Сообщение бота с информацией о направлении



### 3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

#### 3.1 Расчет затрат на разработку программы и решение задачи на ЭВМ

##### 3.1.1 Расчет затрат на разработку программы

Основными компонентами затрат на разработку программ и решение задачи на ЭВМ являются затраты, связанные с оплатой труда специалистов на разработку программы, обслуживание и эксплуатацию ЭВМ в период отладки программы и решения задачи, то есть рассчитываются прямые и косвенные затраты.

При определении полной себестоимости программы учтены все материальные расходы, расходы по заработной плате, отчисления в социальные статьи и составлена калькуляция затрат в следующей последовательности:

- 1) основная заработная плата персонала;
- 2) дополнительная заработная плата персонала;
- 3) отчисления в социальные статьи;
- 4) стоимость работ на ЭВМ;
- 5) расчет стоимости материалов;
- 6) расчет косвенных затрат на разработку программы.

Определим стоимость часа работы руководителя и программиста.

Стоимость часа работы руководителя  $C_{т.ч.р.}$ , руб., определяется по следующей формуле

$$C_{т.ч.р.} = \frac{O_{клад.р.}}{K_{р.ч.}}, \quad (1)$$

где  $O_{клад.р.}$  – оклад руководителя, руб.;

$K_{р.ч.}$  – количество рабочих часов в месяце, ч.

Рассчитаем по формуле (1) стоимость часа работы руководителя:

$$C_{\text{т.ч.р.}} = \frac{50000}{168} = 298 \text{ руб/ч.}$$

Стоимость часа работы программиста  $C_{\text{т.ч.п.}}$ , руб., определяется по следующей формуле

$$C_{\text{т.ч.п.}} = \frac{O_{\text{клад.п.}}}{K_{\text{р.ч.}}}, \quad (2)$$

где  $O_{\text{клад.п.}}$  – оклад программиста, руб.;

$K_{\text{р.ч.}}$  – количество рабочих часов в месяце, ч.

Рассчитаем по формуле (2) стоимость часа работы программиста:

$$C_{\text{т.ч.п.}} = \frac{20000}{168} = 119 \text{ руб/ч.}$$

При расчете всех экономических показателей была составлена таблица 1, в которой указаны все этапы работы по разработке программы и решению задачи, исполнитель каждого этапа, трудоемкость и стоимость исполнения.

Стоимость каждого этапа определена, исходя из оклада исполнителей и времени выполнения этапа.

Количество рабочих часов в месяце равно 168 часов, то есть 21 рабочий день в месяце по 8 часов.

Таблица 1 – Этапы разработки

Наименование этапов работ	Исполнитель	Трудоемкость, ч	Плата за час, руб/ч	Стоимость исполнения, руб.
Постановка задачи	Руководитель	8	298	2384
	Программист	8	119	952
Изучение литературы	Программист	4	119	476
Технический проект	Программист	17	119	2023
	Руководитель	9	298	2682

Продолжение таблицы 1

Наименование этапов работ	Исполнитель	Трудоемкость, ч	Плата за час, руб/ч	Стоимость исполнения, руб.
Эскизный проект	Программист	15	119	1785
	Руководитель	4	298	1192
Кодирование	Программист	48	119	5712
Отладка программы Тестирование	Программист	12	119	1428
Оптимизация программы	Программист	5	119	595
Оформление сопроводительной документации	Программист	9	119	1071
	Руководитель	4	298	1192
Итого	Программист	118	119	14042
	Руководитель	25	298	7450

Основная заработная плата персонала ЗП, руб., рассчитывается по следующей формуле

$$ЗП = C_{\text{ти}} \times РК, \quad (3)$$

где  $C_{\text{ти}}$  – стоимость исполнения для каждого исполнителя (из таблицы 1), руб.;

РК – районный коэффициент (1,3).

Рассчитаем по формуле (3) основную заработную плату руководителя:

$$ЗП_{\text{рук.}} = 7450 \times 1,3 = 9685 \text{ руб.}$$

Рассчитаем по формуле (3) основную заработную плату программиста:

$$ЗП_{\text{пр.}} = 14042 \times 1,3 = 18255 \text{ руб.}$$

Дополнительная заработная плата персонала  $ЗП_{\text{доп}}$ , руб., включает различные виды доплат и составляет 10% от основной заработной платы, рассчитывается по следующей формуле

$$ЗП_{\text{доп}} = ЗП \times 0,10, \quad (4)$$

где  $ЗП$  – основная заработная плата персонала, руб.

Рассчитаем по формуле (4) дополнительную заработную плату руководителя:

$$ЗП_{\text{доп}} = 7450 \times 0,1 = 745 \text{ руб.}$$

Рассчитаем по формуле (4) дополнительную заработную плату программиста:

$$ЗП_{\text{доп}} = 14042 \times 0,1 = 1404 \text{ руб.}$$

Отчисления на социальные нужды  $О_{\text{сн}}$ , руб., составляют 30% и рассчитываются по формуле

$$О_{\text{сн}} = (ЗП + ЗП_{\text{доп}}) \times 0,30, \quad (5)$$

где  $ЗП_{\text{доп}}$  – дополнительная заработная плата, руб.;

$ЗП$  – основная заработная плата, руб.

Рассчитаем по формуле (5) отчисления на социальные нужды руководителя:

$$О_{\text{сн.р.}} = (9685 + 745) \times 0,30 = 3129 \text{ руб.}$$

Рассчитаем по формуле (5) отчисления на социальные нужды программиста:

$$О_{\text{сн.п.}} = (18255 + 1404) \times 0,30 = 5898 \text{ руб.}$$

Стоимость работ на ЭВМ  $C_{pm}$ , руб., рассчитывается по следующей формуле

$$C_{pm} = C_{мч} \times T_m, \quad (6)$$

где  $C_{мч}$  – стоимость машинного часа, руб.

$T_m$  – общее время работы ЭВМ, ч.

Рассчитаем по формуле (6) стоимость работ на ЭВМ:

$$C_{pm} = 5 \times 143 = 715 \text{ руб.}$$

Расчет стоимости материалов приведен в таблице 2.

Таблица 2 – Материальные расходы

Наименование материалов	Единица измерения	Количество	Цена за 1 ед., руб.	Стоимость, руб.
Бумага (А4, 500 л, белая)	уп.	2	300	600
Ручка	шт.	2	30	60
Корректор	шт.	1	70	70
Итого				730

Косвенные расходы на разработку программы  $K_p$ , руб., рассчитываются по следующей формуле

$$K_p = ЗП \times K_{нр}, \quad (7)$$

где  $ЗП$  – основная заработная плата персонала, руб.;

$K_{нр}$  – коэффициент накладных расходов (5-10%).

Рассчитаем по формуле (7) косвенные расходы на разработку программы:

$$K_p = 27940 \times 0,05 = 1397 \text{ руб.}$$

Полная себестоимость программы приведена в таблице 3.

Таблица 3 – Смета затрат на разработку

Наименование статей расходов		Стоимость работ, руб.
Основная заработная плата	программиста	18255
	руководителя	9685
Дополнительная заработная плата	программиста	1404
	руководителя	745
Отчисления на социальные нужды	программиста	5898
	руководителя	3129
Стоимость работ на ЭВМ		715
Стоимость материалов		730
Косвенные расходы		1397
Полная себестоимость		41958

### 3.1.2 Расчет годовых затрат на эксплуатацию программы

Стоимость одного непосредственного решения на ЭВМ  $C_{р.м}$ , руб., определяется по формуле

$$C_{р.м} = C_{мч} \times T_p + ЗП_{о.п.} \times Q \times K_p \times K_{кр}, \quad (8)$$

где  $C_{мч}$  – стоимость работы на ЭВМ за час, руб/ч;

$T_p$  – время решения задачи на ЭВМ, ч;

$Q$  – трудоемкость исполнителя, ч;

$K_p$  – районный коэффициент (1,3);

$K_{кр}$  – коэффициент косвенных расходов (1,05);

$ЗП_{о.п.}$  – заработная плата за час работника, руб/ч.

Рассчитаем по формуле (8) стоимость одного непосредственного решения на ЭВМ:

$$C_{р.м} = 5 \times 0,3 + 119 \times 1 \times 1,3 \times 1,05 = 164 \text{ руб.}$$

Расчет годовых затрат на разработку программы необходимо провести для последующего анализа эффективности данного программного продукта.

Годовые затраты на эксплуатацию программы  $C_{р.м.год}$ , руб., рассчитываются по следующей формуле

$$C_{р.м.год} = N \times C_{р.м} + E_n \times C, \quad (9)$$

где  $N$  – плотность потока заявок в год, шт.;

$C_{р.м.год}$  – годовые затраты на эксплуатацию программы, руб.;

$C_{р.м}$  – стоимость одного непосредственного решения на ЭВМ, руб.;

$E_n$  – нормальный коэффициент сложности (0,2-0,6);

$C$  – себестоимость разработки программы, руб. (итог таблицы 3).

Рассчитаем по формуле (9) годовые затраты на разработку программы:

$$C_{р.м.год} = 100 \times 164 + 0,2 \times 41958 = 24720 \text{ руб.}$$

### 3.2 Расчет экономического эффекта и определение срока окупаемости

Экономический эффект достигается при эксплуатации и характеризуется экономией живого и овеществленного труда в общественном производстве, выраженной в денежной форме (прибыль предприятия), а также снижением затрат.

Социальный эффект заключается в обеспечении комфортных условий жизни населения и развития экономики страны.

Для того чтобы определить экономическую эффективность проекта необходимо рассчитать затраты на эксплуатацию ранее употреблявшимся образом.

Расходы на выполнение работ ранее употреблявшимся способом  $C_{р.сп}$ , руб., рассчитываются по следующей формуле

$$C_{р.сп} = 3П_{сп} \times T_{сп} \times K_{кр} \times K_p, \quad (10)$$

где  $3П_{сп}$  – заработная плата специалиста за час, руб/ч;

$T_{\text{сп}}$  – затраты времени специалиста на выполнение работ ранее употреблявшимся способом, ч;

$K_p$  – районный коэффициент (1,3);

$K_{\text{кр}}$  – коэффициент косвенных расходов (1,05).

Рассчитаем по формуле (10) расходы на выполнение работ ранее употреблявшимся способом:

$$C_{\text{р.сп}} = 119 \times 4,5 \times 1,05 \times 1,3 = 731 \text{ руб.}$$

Зная стоимость всех работ по выполнению одной задачи, определим годовые расходы ранее употреблявшимся способом  $C_{\text{р.сп.год}}$ , руб., по следующей формуле

$$C_{\text{р.сп.год}} = N \times C_{\text{р.сп}}, \quad (11)$$

где  $N$  – плотность потока заявок в год, шт.;

$C_{\text{р.сп.}}$  – расходы на выполнение работ ранее употреблявшимся способом, руб.

Рассчитаем годовые затраты на выполнение работ ранее употреблявшимся способом используя формулу (11):

$$C_{\text{р.сп.год}} = 100 \times 731 = 73100 \text{ руб.}$$

### 3.2.1 Экономический эффект и срок окупаемости

Экономическая эффективность  $\mathcal{E}_{\text{год}}$ , руб., рассчитывается по следующей формуле

$$\mathcal{E}_{\text{год}} = C_{\text{р.сп.год}} - C_{\text{р.м.год}}, \quad (12)$$

где  $C_{\text{р.сп.год}}$  – годовые затраты на выполнение работ ранее употреблявшимся способом, руб.;

$C_{\text{р.м.год}}$  – годовые затраты на эксплуатацию программы, руб.



Рассчитаем по формуле (12) экономию, связанную с использованием разработки:

$$\Delta_{\text{год}} = 73100 - 24720 = 48380 \text{ руб.}$$

### 3.2.2 Определение коэффициента экономической эффективности программы

Коэффициент экономической эффективности показывает сколько на один рубль вложенных затрат в разработку и эксплуатацию, получаем экономии. Чем больше данное значение, тем эффективнее проект.

Коэффициент экономической эффективности  $E_p$ , рассчитывается по формуле

$$E_p = \frac{\Delta_{\text{год}}}{(C + C_{\text{р.м.год}})}, \quad (13)$$

где  $\Delta_{\text{год}}$  – экономия, связанная с использованием разработки, руб.;

$C_{\text{р.м.год}}$  – годовые затраты на эксплуатацию программы, руб.;

$C$  – себестоимость разработки программы, руб.

Рассчитаем по формуле (13) экономическую эффективность программы:

$$E_p = \frac{48380}{41958 + 24720} = 0.72$$

Экономический эффект показывает, что на один вложенный рубль в разработку и эксплуатацию программы, получаем 72 копейки экономии. Так как проект не предполагает коммерциализации, мы не можем посчитать его коммерческую эффективность, но в результате внедрения программы облегчается труд специалиста, снижаются затраты времени на решение задач.

Срок окупаемости программы  $T_{\text{ок}}$ , год рассчитываем исходя из экономии. То есть благодаря экономии за какой период времени окупятся затраты на разработку и внедрение программы.

$$T_{\text{ок}} = \frac{1}{E_p}, \quad (14)$$

Рассчитаем по формуле (14) срок окупаемости:

$$T_{\text{ок}} = \frac{1}{0.72} = 1.4 \text{ года}$$

Таким образом, программа окупится через 17 месяцев.

На основании проведенных расчетов себестоимости и экономического эффекта можно сделать следующие выводы:

- 1) результаты технико-экономического обоснования свидетельствуют об экономической эффективности проекта;
- 2) за счет снижения эксплуатационных затрат проект окупится через 17 месяцев.

## ЗАКЛЮЧЕНИЕ

По итогу выполнения дипломного проекта были выполнены следующие задачи:

- 1) Исследование технологий по созданию ботов;
- 2) Обзор решений для создания чат-ботов;
- 3) Анализ конкурентов;
- 4) Формулирование сценария работы бота;
- 5) Выбор платформы и языка программирования;
- 6) Исследование доступных библиотек;
- 7) Анализ возможностей доступных библиотек;
- 8) Реализация программного кода;
- 9) Исправление ошибок логики и ошибок интерфейса;
- 10) Составление документации.

Также был успешно разработан Telegram-бот для консорциума томских вузов, успешно и без ошибок выполняющий свои функции.

Разработанный бот позволяет существенно расширить количество потенциальных абитуриентов, ознакомленных с интересующими их направлениями обучения в высших учебных заведениях, входящих в консорциум. Исходя из результатов работы можно сделать вывод о успешном достижении целей дипломного проекта.

## ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

- 1) Типичный студент: как выглядит учащийся вуза с точки зрения статистики — Текст : электронный // ТИНЬКОФФ ЖУРНАЛ : [сайт]. — URL: <https://journal.tinkoff.ru/students-stat/> (Дата обращения: 01.02.2023).
- 2) Использование диаграммы вариантов использования UML при проектировании программного обеспечения — Текст : электронный // habr : [сайт]. — <https://habr.com/ru/post/566218/>. Дата обращения: 28.02.2023.
- 3) Проектирование диаграммы состояний UML (statechart diagram) — Текст : электронный // Worldskills: [сайт]. — <https://nationalteam.worldskills.ru/skills/proektirovanie-diagrammy-sostoyaniy-uml-statechart-diagram/>. Дата обращения: 28.02.2023.
- 4) Теория и практика UML. Диаграмма состояний — Текст : электронный // IT-GOST.RU: [сайт]. — [http://it-gost.ru/articles/view\\_articles/97](http://it-gost.ru/articles/view_articles/97). Дата обращения: 5.03.2023
- 5) Полное руководство по языку программирования C# 10 и платформе .NET 6 — Текст: электронный // METANIT.COM [сайт]. — URL: <https://metanit.com/sharp/tutorial/> Дата обращения: 12.04.2023.
- 6) Стилмен, Э., Грин, Д. Head First. Изучаем C#. 4-е издание — Текст: электронный [электронная книга]. — 2022 (дата обращения: 20.04.2023).
- 7) Общие сведения о WPF — Текст : электронный // Microsoft Learn: [сайт]. — <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8>. Дата обращения: 5.03.2023
- 8) Введение в WPF — Текст : электронный // METANIT.COM: [сайт]. — <https://metanit.com/sharp/wpf/1.php>. Дата обращения: 5.03.2023
- 9) Васильев, А. Н. Программирование на C# для начинающих — Текст: электронный [электронная книга]. — 2022 (дата обращения: 27.03.2023).

- 10) Просто о списках, словарях и множествах или ТОП 5 структур данных — Текст : электронный // habr : [сайт]. — URL: <https://habr.com/ru/post/232009/> (Дата обращения: 25.04.2023).
- 11) Евдокимов, П. В. С# на примерах. 4-е издание — Текст: электронный [электронная книга]. — 2019 (дата обращения: 30.03.2023).
- 12) Практикум 9: Пример технического задания для рецензирования — Текст : электронный // materialdesigninxml : [сайт]. — URL: <https://intuit.ru/studies/courses/2195/55/lecture/15050?page=2> (Дата обращения: 01.04.2023).
- 13) Material Design In XAML — Текст : электронный // ИНТУИТ : [сайт]. — URL: <http://materialdesigninxml.net/home#home> (Дата обращения: 06.05.2023).

## Листинг кода приложения

Содержимое файла App.xaml.cs:

```
using Microsoft.Data.Sqlite;
using System.Windows;

namespace botserver_standard
{
    public partial class App : Application
    {
        protected override void OnStartup(StartupEventArgs e)
        {
            base.OnStartup(e);
            // OnStartup code next:

            Stats.StartupTimeFixator();
            Settings.connString = "Data Source = appDB.db";

            try
            {
                DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
DbWorker.dbStructureRessurrection);
            }

            catch
            {
                return;
            }

            finally
            {
                //чтение настроек
```

```

        using SqliteDataReader reader =
        DbWorker.SettingsReader(DbWorker.readSettings, DbWorker.sqliteConn);
    }

}

}
}

```

Содержимое файла AskingPassword.xaml.cs:

```

using System.Windows;

namespace botserver_standard
{
    public partial class AskingPassword : Window
    {
        public AskingPassword()
        {
            InitializeComponent();
        }

        private void NextBtn_Click(object sender, RoutedEventArgs e)
        {
            if (EnterPwdBox.Password == Settings.pwd) //если введенный пароль корректен
            {
                this.DialogResult = true;
            }
        }

        public string Password
        {
            get { return EnterPwdBox.Password; }
        }
    }
}

```

Содержимое файла Card.cs:

```
using System.Collections.Generic;

namespace botserver_standard
{

    public class Card
    {
        public static List<Card> cards = new(); // упорядоченный набор карточек
(экземпляров классов)

        public int Id { get; set; }
        public string UniversityName { get; set; }
        public string ProgramName { get; set; }
        public string Level { get; set; }
        public string ProgramCode { get; set; }
        public string StudyForm { get; set; }
        public string Duration { get; set; }
        public string StudyLang { get; set; }
        public string Curator { get; set; }
        public string PhoneNumber { get; set; }
        public string Email { get; set; }
        public string Cost { get; set; }

        public Card(int id, string universityName, string programName, string level, string
studyForm, string programCode, string duration, string studyLang, string curator, string
phoneNumber, string email, string cost)
        {
            this.Id = id;
            this.UniversityName = universityName;
            this.ProgramName = programName;
            this.Level = level.ToLower();
            this.StudyForm = studyForm.ToLower();
```



```

        this.ProgramCode = programCode;
        this.Duration = duration.ToLower();
        this.StudyLang = studyLang.ToLower();
        this.Curator = curator;
        this.PhoneNumber = phoneNumber;
        this.Email = email;
        this.Cost = cost;

    }
}

```

Содержимое файла ChangeDefaultPwd.xaml.cs:

```

using Microsoft.Data.Sqlite;
using System;
using System.Windows;

namespace botserver_standard
{
    public partial class ChangeDefaultPwd : Window
    {
        public ChangeDefaultPwd()
        {
            InitializeComponent();
            EnterPwdBox.MaxLength = 50;
            EnterPwdBox_Repeated.MaxLength = 50;
        }

        int rowsChanged;
        private void NextBtn_Click(object sender, RoutedEventArgs e)
        {
            string changeDefaultPwdQuery = $"UPDATE Settings SET pwd =
'{{EnterPwdBox.Password}}'";

```

```

string setCheckboxQuery = $"UPDATE Settings SET pwdIsUsing = 'True';";
DateTime updateMoment;
string previousPwdWrite = $"INSERT INTO Passwords_history (timestamp,
oldPassword) VALUES ('{updateMoment = DateTime.Now}', '{Settings.pwd}');"; //установка
пароля по умолчанию и отключение его запроса при старте

```

```

if (EnterPwdBox.Password == EnterPwdBox_Repeated.Password &&
EnterPwdBox_Repeated.Password != Settings.pwd) // если юзер не ошибся и пароль не равен
предыдущему

```

```

{
    rowsChanged = DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
changeDefaultPwdQuery); //смена дефолтного пароля
    //IsSetted?
    if (rowsChanged is 1) //если удачно
    {
        DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
previousPwdWrite); //запись истории паролей

```

```

        DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
setCheckboxQuery); //установка галки на использование пароля на старте

```

```

using SqlDataReader reader =
DbWorker.SettingsReader(DbWorker.readSettings, DbWorker.sqliteConn); //обновление
настроек приложения из бд

```

```

MessageBox.Show("Пароль успешно установлен. На вкладке \"Settings\"
вы можете отключить его использование.", "Notice");

```

```

this.DialogResult = true;

```

```

}

```

```

else

```

```

        { MessageBox.Show("Непредвиденная ошибка", "Error"); }
    }
    else
    {
        MessageBox.Show("Вы пытаетесь установить пароль по умолчанию, либо
пароли не совпадают", "Notice");
    }

}

}

}

```

Содержимое файла ConsoleWorker.cs:

```

using System;
using System.IO;

namespace botserver_standard
{
    internal class ConsoleWorker
    {
        public static void CardOutputter()
        {
            MainWindow.AllocConsole();

            TextWriter stdOutWriter = new StreamWriter(Console.OpenStandardOutput(),
Console.OutputEncoding) { AutoFlush = true };
            TextWriter stdErrWriter = new StreamWriter(Console.OpenStandardError(),
Console.OutputEncoding) { AutoFlush = true };
            TextReader strInReader = new StreamReader(Console.OpenStandardInput(),
Console.InputEncoding);

            Console.SetOut(stdOutWriter);
            Console.SetError(stdErrWriter);

```

```

        Console.SetIn(strInReader);

        foreach (var item in Card.cards)
        {
            Console.WriteLine($"{item.Id}          |          {item.UniversityName}          |
{item.ProgramName} | {item.Level} | " +
            $"{item.ProgramCode}    |    {item.StudyForm}    |    {item.Duration}    |
{item.StudyLang} | " +
            $"{item.Curator} | {item.PhoneNumber} | {item.Email} | {item.Cost}");

        }

        Console.ReadKey();

        MainWindow.FreeConsole();
    }
}

```

Содержимое файла DatagridControls.cs:

```

using System.Collections.Generic;
using System.IO;
using System.Windows;
using System.Windows.Input;

namespace botserver_standard
{
    public partial class MainWindow : Window
    {
        //parsedCards
        private void SearchByProgramName_KeyUp(object sender, KeyEventArgs e)
        {
            List<Card> searchResult = new();

```

```

string programNameFrag = SearchByProgramName.Text;
foreach (var item in cardsView)
{
    if (item.ProgramName.ToLower().Contains(programNameFrag))
    {
        searchResult.Add(item);
    }
    else { continue; }
}
parsedCardsGrid.ItemsSource = searchResult;
}

private void SearchByUniversity_KeyUp(object sender, KeyEventArgs e)
{
    List<Card> searchResult = new();
    string universityNameFrag = SearchByUniversity.Text;
    foreach (var item in cardsView)
    {
        if (item.UniversityName.ToLower().Contains(universityNameFrag))
        {
            searchResult.Add(item);
        }
        else { continue; }
    }
    parsedCardsGrid.ItemsSource = searchResult;
}

private void CardsExportBtn_Click(object sender, RoutedEventArgs e)
{
    StreamWriter parsedCardsExport = new(Settings.datagridExportPath);
    parsedCardsExport.WriteLine("Id      Название      университета\tНазвание
программы\tКод      программы\tУровень      обучения\tФорма      обучения\tДлительность
обучения\tЯзык обучения\tКуратор\tНомер телефона\tПочта\tСтоимость");
    foreach (var item in cardsView)

```

```

{

parsedCardsExport.WriteLine($"{item.Id}\t{item.UniversityName}\t{item.ProgramName}\t{ite
m.ProgramCode}\t{item.Level}\t{item.StudyForm}\t{item.Duration}\t{item.StudyLang}\t{item
.Curator}\t{item.PhoneNumber}\t{item.Email}\t{item.Cost}");

}
parsedCardsExport.Close();
}

//parsedUniversities
private void SearchByUniversityName_KeyUp(object sender, KeyEventArgs e)
{
    List<UniversityEntryFreq> searchResult = new();
    string universityNameFrag = SearchByUniversityName.Text;
    foreach (var item in UniversityEntryFreq.universitiesFreqList)
    {
        if (item.UniversityName.ToLower().Contains(universityNameFrag))
        {
            searchResult.Add(item);
        }
        else { continue; }
    }
    parsedUniversitiesGrid.ItemsSource = searchResult;
}

private void SearchByUniversityFreq_KeyUp(object sender, KeyEventArgs e)
{
    List<UniversityEntryFreq> searchResult = new();
    string universityNameFrag = SearchByUniversityFreq.Text;
    foreach (var item in UniversityEntryFreq.universitiesFreqList)
    {
        if (item.Count.ToString().Contains(universityNameFrag))

```

```

        {
            searchResult.Add(item);
        }
        else { continue; }
    }
    parsedUniversitiesGrid.ItemsSource = searchResult;
}

private void UniversitiesExportBtn_Click(object sender, RoutedEventArgs e)
{
    StreamWriter parsedUniversitiesExport = new("exportUniversities.txt");
    parsedUniversitiesExport.WriteLine("Id  Название  университета\tКоличество
программ");
    foreach (var item in UniversityEntryFreq.universitiesFreqList)
    {
        parsedUniversitiesExport.WriteLine($"{item.UniversityName}\t{item.Count}");
    }
    parsedUniversitiesExport.Close();
}
}
}

```

Содержимое файла DbWorker.cs:

```

using Microsoft.Data.Sqlite;
using System;

namespace botserver_standard
{
    internal class DbWorker
    {
        public static bool pwdSetResult;
        public static SqliteConnection sqliteConn = new(Settings.connString);
    }
}

```

```
//восстановление структуры БД, если файл не найден
```

```
public static readonly string dbStructureRessurrection =
    "CREATE TABLE IF NOT EXISTS Received_messages (username TEXT, is_bot
INTEGER, first_name TEXT, last_name TEXT, language_code TEXT, chat_id INTEGER,
message_id INTEGER, message_date TEXT, chat_type TEXT, message_content BLOB);" +
    "\r\nCREATE TABLE IF NOT EXISTS Settings (logPath TEXT, connString
TEXT, botToken TEXT, pwd TEXT, pwdIsUsing TEXT, prsFilePath TEXT);" +
    "\r\nCREATE TABLE IF NOT EXISTS Cards (id INTEGER NOT NULL
UNIQUE, universityName TEXT, programName TEXT, programCode TEXT, level TEXT,
studyForm TEXT, duration TEXT, studyLang TEXT, curator TEXT, phoneNumber TEXT, email
TEXT, cost TEXT, PRIMARY KEY(id)) WITHOUT ROWID" +
    "\r\nCREATE TABLE IF NOT EXISTS Session_duration (startup_time TEXT,
shutdown_time TEXT, total_uptime TEXT);" +
    "\r\nCREATE TABLE IF NOT EXISTS Universities (id INTEGER NOT NULL,
universityName TEXT);" +
    "\r\nCREATE TABLE IF NOT EXISTS Directions (id INTEGER NOT NULL,
directionName TEXT);" +
    "\r\nCREATE TABLE IF NOT EXISTS Programs (id INTEGER NOT NULL,
programName TEXT);" +
    "\r\nCREATE TABLE IF NOT EXISTS Parsing_history (timestamp TEXT,
parsingStart TEXT, parsingEnd TEXT, parsingResult INTEGER);" +
    "\r\nCREATE TABLE IF NOT EXISTS Passwords_history (timestamp TEXT,
oldPassword TEXT);";

public static string readTokenFromDb = "SELECT botToken FROM Settings";

public static readonly string received_messagesConsoleOutput = "SELECT * FROM
Received_messages";

public static readonly string readSettings = "SELECT * FROM Settings";
```



```

public static int DbQuerySilentSender(SqliteConnection sqliteConn, string
queryText) //no feedback
{
    sqliteConn.Open();
    SqliteCommand command = new()
    {
        Connection = sqliteConn, //соединение для выполнения запроса
        CommandText = queryText //текст запроса
    };
    int rowsChanged = command.ExecuteNonQuery(); //выполнение запроса и
возврат количества измененных строк
    sqliteConn.Close(); //закрытие соединения
    return rowsChanged;
}

public static SqliteDataReader SettingsReader(string queryText, SqliteConnection
sqliteConn) //оновление настроек из бд
{
    sqliteConn.Open(); //открытие соединения
    SqliteCommand command = new() //инициализация экземпляра SqliteCommand
    {
        Connection = sqliteConn, //соединение для выполнения запроса
        CommandText = queryText //текст запроса
    };
    SqliteDataReader reader = command.ExecuteReader();

    if (reader.HasRows) // если есть строки
    {
        while (reader.Read()) // построчное чтение данных
        {
            Settings.fileLoggerPath = Convert.ToString(reader["fileLoggerPath"]);
            Settings.callbackLoggerPath
            =
            Convert.ToString(reader["callbackLoggerPath"]);
            Settings.botToken = Convert.ToString(reader["botToken"]);

```

```

        Settings.pwd = Convert.ToString(reader["pwd"]);
        Settings.pwdIsUsing = Convert.ToBoolean(reader["pwdIsUsing"]);
        Settings.datagridExportPath
Convert.ToString(reader["datagridExportPath"]);
        Settings.parsingUrl = Convert.ToString(reader["parsingUrl"]);

    }
}
sqliteConn.Close();
return reader;
}

}
}

```

Содержимое файла MainTab.cs:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Threading;
using System.Windows;
using Telegram.Bot.Exceptions;
using Telegram.Bot.Types;
using Telegram.Bot;
using Telegram.Bot.Polling;
using Telegram.Bot.Types.ReplyMarkups;
using System.Linq;

namespace botserver_standard
{
    public partial class MainWindow : Window

```

```

{
    //maintab methods

    static string? selectedLevel;
    static string? selectedUniversity;
    static string? selectedProgram;
    string? firstname;
    private async void BotStartBtn_Click(object sender, RoutedEventArgs e)
    {
        LiveLogOutput.Clear();

        using CancellationTokenSource OnBotLoadCts = await OnBotLoadMsg();

        // отправка запроса отмены для остановки
        OnBotLoadCts.Cancel();

        var receiverOptions = new ReceiverOptions
        {
            AllowedUpdates = { }, // receive all update types
        };

        await Task.Factory.StartNew(() =>
            TgBot.botClient.StartReceiving(updateHandler: HandleUpdateAsync,
                                           pollingErrorHandler: HandleErrorAsync,
                                           cancellationToken:
            TgBot.MainBotCts.Token, receiverOptions: receiverOptions)); //ok

        async Task HandleUpdateAsync(ITelegramBotClient botClient, Update update,
            CancellationToken cancellationToken)
        {
            Message message = update.Message;
            if (message is null) { goto Eight; }

            #region sqlQueries править запросы на запись в бд

```

```

//запись принятых сообщений в бд
string recievedMessageToDbQuery = $"INSERT INTO
Received_messages(username, is_bot, first_name, last_name, language_code, chat_id,
message_id, message_date, chat_type, message_content) " +
    $"VALUES('@{message.Chat.Username}', '0', '{message.Chat.FirstName}',
'{message.Chat.LastName}', 'ru', '{message.Chat.Id}', '{message.MessageId}',
'{DateTime.Now}', '{message.Chat.Type}', '{message.Text}')";

//запись принятых фотографий в бд
string recievedPhotoMessageToDbQuery = $"INSERT INTO
Received_messages(username, is_bot, first_name, last_name, language_code, chat_id,
message_id, message_date, chat_type, message_content) " +
    $"VALUES('@{message.Chat.Username}', '0', '{message.Chat.FirstName}',
'{message.Chat.LastName}', 'ru', '{message.Chat.Id}', '{message.MessageId}',
'{DateTime.Now}', '{message.Chat.Type}', '{message.Photo}')";

#endregion

if (update.Type is Telegram.Bot.Types.Enums.UpdateType.Message &&
message.Text is null) //suggestion if recieved not text message
{
    await botClient.SendTextMessageAsync(chatId: message.Chat.Id, text:
$"Пожалуйста, выберите один из доступных вариантов:", replyMarkup:
TelegramBotKeypads.mainMenuKeypad, cancellationToken: cancellationToken);

    LiveLogger_message(message); // живой лог
    FileLogger_message(message, message.Text, message.Chat.Id,
Settings.fileLoggerPath); // логгирование в файл
    return;
}

if (update.Type is Telegram.Bot.Types.Enums.UpdateType.Message &&
message.Text.ToLower() == "/start") //if recieved Message update type
{
    firstname = update.Message.Chat.FirstName;

```

```

        if (message.Text.ToLower() == "/start") //if recieved this text
        {
            LiveLogger_message(message); // живой лог
            FileLogger_message(message, message.Text, message.Chat.Id,
Settings.fileLoggerPath); // логгирование в файл
            DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
recievedMessageToDbQuery);

            await botClient.SendTextMessageAsync(chatId: message.Chat.Id, text:
$"Добро пожаловать, {firstname}!", replyMarkup: TelegramBotKeypads.mainMenuKeypad,
cancellationTokens: cancellationTokens);

            return;
        }
        else if (message.Text is not null && message.Text.ToLower() is not "/start")
        {
            await botClient.SendTextMessageAsync(chatId: message.Chat.Id, text:
$"Пожалуйста, выберите один из доступных вариантов:", replyMarkup:
TelegramBotKeypads.mainMenuKeypad, cancellationTokens: cancellationTokens);

            LiveLogger_message(message); // живой лог
            FileLogger_message(message, message.Text, message.Chat.Id,
Settings.fileLoggerPath); // логгирование в файл
            DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
recievedMessageToDbQuery);

            return;
        }

        await botClient.SendTextMessageAsync(chatId: message.Chat.Id, text:
$"Добро пожаловать, {firstname}!", replyMarkup: TelegramBotKeypads.mainMenuKeypad,
cancellationTokens: cancellationTokens);
    }

```

Eight:

```
if (update.Type is Telegram.Bot.Types.Enums.UpdateType.CallbackQuery)
```

```

        {
            if (update.CallbackQuery.Data is "toHome") //действия при нажатии На
главную
            {
                string telegramMessage = $"Добро пожаловать, {firstname}!";
                await
botClient.EditMessageTextAsync(update.CallbackQuery.Message.Chat.Id,
update.CallbackQuery.Message.MessageId,          telegramMessage,          replyMarkup:
TelegramBotKeypads.mainMenuKeypad,                parseMode:
Telegram.Bot.Types.Enums.ParseMode.Html, cancellationToken: cancellationToken);
            }

            if (update.CallbackQuery.Data is "programChoose") //если ответ был
programChoose, то изменить сообщение на следующее...
            {
                string telegramMessage = "Выберите желаемый уровень подготовки.";
                await
botClient.EditMessageTextAsync(update.CallbackQuery.Message.Chat.Id,
update.CallbackQuery.Message.MessageId,          telegramMessage,          replyMarkup:
TelegramBotKeypads.levelChoosingKeypad,            parseMode:
Telegram.Bot.Types.Enums.ParseMode.Html, cancellationToken: cancellationToken);
            }

            if (update.CallbackQuery.Data.Contains("_level")) //если ответ содержал в
себе level, то изменить сообщение на следующее...
            {
                selectedLevel      =      update.CallbackQuery.Data.Replace("_level",
string.Empty) as string;
                //UniversityEntryFreq.universitiesFreqList;
                List<InlineKeyboardButton> parsedUniversitiesButtons = new(); //

                foreach (var item in UniversityEntryFreq.universitiesFreqList)
                {

```

```

parsedUniversitiesButtons.Add(InlineKeyboardButton.WithCallbackData(text:
item.UniversityName, callbackData: Convert.ToString(item.UniversityName) + "_university"));
    }

    parsedUniversitiesButtons.Add(InlineKeyboardButton.WithCallbackData(text: "🏠",
callbackData: "toHome"));

    var dynamicUniversityChoosingKeypad = new
    InlineKeyboardMarkup(parsedUniversitiesButtons);

    string telegramMessage = "Пожалуйста, выберите необходимый
университет:";

    await
    botClient.EditMessageTextAsync(update.CallbackQuery.Message.Chat.Id,
    update.CallbackQuery.Message.MessageId, telegramMessage, replyMarkup:
    dynamicUniversityChoosingKeypad, parseMode: Telegram.Bot.Types.Enums.ParseMode.Html,
    cancellationToken: cancellationToken);

    }

    if (update.CallbackQuery.Data.Contains("_university"))
    {

        selectedUniversity = update.CallbackQuery.Data.Replace("_university",
string.Empty) as string;

        string telegramMessage = "Подобранные программы обучения:\n\n";

        // фильтрация карточек на основании выборов абитуриента
        List<Card> filteredCardsByEnrollee = new();
        foreach (var item in cardsView) //переписать цикл на фор для
нормальной нумерации направлений
    {

```

## ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А

```

        if (selectedLevel == item.Level && selectedUniversity ==
item.UniversityName)
            filteredCardsByEnrollee.Add(item);
    }

    //составление сообщения с номерами направлений
    foreach (var item in filteredCardsByEnrollee)
    {
        telegramMessage += $"{item.Id}:\t{item.ProgramName}\n";
    }

    //извлечение идентификаторов из отфильтрованных направлений
    List<string> cardIds = new();
    foreach (var item in filteredCardsByEnrollee)
    {
        cardIds.Add(Convert.ToString(item.Id));
    }

    //генерация кнопок на основе отфильтрованных карточек
    var filteredUniversitiesButtons = new
List<List<InlineKeyboardButton>>>();

    for (int i = 0; i < cardIds.Count; i += 3)
        filteredUniversitiesButtons.Add(new
List<InlineKeyboardButton>(cardIds.Skip(i).Take(3).Select(id
=>
InlineKeyboardButton.WithCallbackData(id)))));

    var dynamicProgramChoosingKeypad = new
InlineKeyboardMarkup(filteredUniversitiesButtons);

    //отправка сообщения
    await
botClient.EditMessageTextAsync(update.CallbackQuery.Message.Chat.Id,
update.CallbackQuery.Message.MessageId, telegramMessage, replyMarkup:

```



```

dynamicProgramChoosingKeypad, parseMode: Telegram.Bot.Types.Enums.ParseMode.Html,
cancellationToken: cancellationToken);
    }

    //отправка финального сообщения с данными о выбранном направлении
    if (int.TryParse(update.CallbackQuery.Data, out int integerValue) is true)
    {
        selectedProgram = update.CallbackQuery.Data;

        Card finalSelectedCard = cardsView[Convert.ToInt32(selectedProgram)];

        string telegramMessage = $"Мы подобрали для вас следующее
направление:\n" +
                                $"Университет:\t{finalSelectedCard.UniversityName}\n"
+
                                $"Программа:\t{finalSelectedCard.ProgramName}\n" +
                                $"Код программы:\t{finalSelectedCard.ProgramCode}\n"
+
                                $"Уровень образования:\t{finalSelectedCard.Level}\n" +
                                $"Форма обучения:\t{finalSelectedCard.StudyForm}\n" +
                                $"Длительность
обучения:\t{finalSelectedCard.Duration}\n" +
                                $"Язык обучения:\t{finalSelectedCard.StudyLang}\n" +
                                $"Куратор:\t{finalSelectedCard.Curator}\n" +
                                $"Номер телефона:\t{finalSelectedCard.PhoneNumber}\n"
+
                                $"Почта:\t{finalSelectedCard.Email}\n" +
                                $"Стоимость обучения:\t{finalSelectedCard.Cost}";

        InlineKeyboardMarkup lastButtonsKeypad = new(
new[]
{
    // first row
new[]

```

```

        {
            InlineKeyboardButton.WithUrl(text: "✉", url:
$"mailto:{finalSelectedCard.Email}"),
        },
        // second row
        new[]
        {
            InlineKeyboardButton.WithCallbackData(text: "🔙", callbackData:
"toHome"),
        },
    });

    await
botClient.EditMessageTextAsync(update.CallbackQuery.Message.Chat.Id,
update.CallbackQuery.Message.MessageId, telegramMessage, replyMarkup: lastButtonsKeypad,
parseMode: Telegram.Bot.Types.Enums.ParseMode.Html, cancellationToken:
cancellationToken);

    LiveLogger_callBack(update.CallbackQuery, finalSelectedCard);
    ChoicesToDb(update.CallbackQuery, finalSelectedCard);
    FileLogger_callBack(update.CallbackQuery, Settings.callbackLoggerPath,
finalSelectedCard);
    }
    }
    }

    Task HandleErrorAsync(ITelegramBotClient botClient, Exception exception,
CancellationToken cancellationToken) //обработчик ошибок API
    {
        var ErrorMessage = exception switch
        {

```

```

        ApiRequestException apiRequestException
            => $"Telegram API
Error:\n[{apiRequestException.ErrorCode}]\n{apiRequestException.Message}\nTimestamp:
{DateTime.Now}",
        _ => exception.ToString()
    };

    Dispatcher.Invoke(() =>
    {
        return LiveLogOutput.Text += $"{ErrorMessage}\n" + "-----
-----\n";
    });
    return Task.CompletedTask;
}

}

public async Task<CancellationTokenSource> OnBotLoadMsg()
{
    CancellationTokenSource OnBotLoadCts = new();
    User me = await TgBot.botClient.GetMeAsync();
    LiveLogOutput.Text += $"Начато прослушивание бота @{me.Username} с
именем {me.FirstName} в {DateTime.Now}\n";
    LiveLogOutput.Text += "-----
-----\n";
    return OnBotLoadCts;
}

public void LiveLogger_message(Message? message)
{
    Dispatcher.Invoke(() =>
    {

```

```

        return LiveLogOutput.Text += $"Получено сообщение '{message.Text}' от
пользователя @{message.Chat.Username} так же известного, как {message.Chat.FirstName}
{message.Chat.LastName} в чате {message.Chat.Id} в {DateTime.Now}.\n" +
        "-----\n";
    });
}

public void LiveLogger_callBack(CallbackQuery callbackQuery, Card card)
{
    Dispatcher.Invoke(() =>
    {
        return LiveLogOutput.Text += $"Пользователь
@{callbackQuery.From.Username}, так же известный, как {callbackQuery.From.FirstName}
{callbackQuery.From.LastName} выбрал уровень {selectedLevel}, университет
{selectedUniversity} и программу {card.ProgramName} в {DateTime.Now}.\n" +
        "-----\n";
    });
}

public static async void FileLogger_message(Message message, string messageText,
long chatId, string logPath) //логгирование полученных сообщений в файл
{
    using StreamWriter logWriter = new(logPath, true); //инициализация экземпляра
StreamWriter

    await logWriter.WriteLineAsync($"Получено сообщение '{messageText}' от
пользователя @{message.Chat.Username}, так же известного, как {message.Chat.FirstName}
{message.Chat.LastName} в чате {chatId} в {DateTime.Now}."); //эхо
    await logWriter.WriteLineAsync("-----\n");
}

```

## ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А

```
public static async void FileLogger_callBack(CallbackQuery callbackQuery, string
logPath, Card card) //логгирование callback в файл
{
    using StreamWriter logWriter = new(logPath, true); //инициализация экземпляра
StreamWriter

    await logWriter.WriteLineAsync($"Пользователь
@{callbackQuery.From.Username}, так же известный, как {callbackQuery.From.FirstName}
{callbackQuery.From.LastName} выбрал уровень {selectedLevel}, университет
{selectedUniversity} и программы {card.ProgramName} в {DateTime.Now}");

    await logWriter.WriteLineAsync("-----
-----");

}

public void ChoicesToDb(CallbackQuery callbackQuery, Card card)
{
    string query = $"INSERT INTO Fixated_choices (username, fname, lname,
choisedLevel, choisedUniversity, choisedProgram, timestamp) " +
        $"VALUES ('@{callbackQuery.From.Username}',
'{callbackQuery.From.FirstName}', '{callbackQuery.From.LastName}', '{selectedLevel}',
'{selectedUniversity}', '{card.ProgramName}', '{DateTime.Now}');
    DbWorker.DbQuerySilentSender(DbWorker.sqliteConn, query); //запись
истории паролей

}

public static async Task PreparedMessageSender(ITelegramBotClient botClient,
string sendingMessage, long chatId, CancellationToken cancellationToken)
{
    await botClient.SendTextMessageAsync(chatId: chatId,
text: sendingMessage,
cancellationToken: cancellationToken);

}
```

```

public static async Task<Task> ParrotedMessageSender(ITelegramBotClient
botClient, Message? message, long? chatId, CancellationToken cancellationToken) //отправка
пользователю текста его сообщения

```

```

{
    if (message is not null)
    {
        await botClient.SendTextMessageAsync(
            chatId: chatId,
            text: $"I received the following message:\n{message.Text}",
            cancellationToken: cancellationToken);
    }

    else await ErrorToChatSender(botClient, chatId, cancellationToken);
    return Task.CompletedTask;
}

```

```

public static async Task ErrorToChatSender(ITelegramBotClient botClient, long?
chatId, CancellationToken cancellationToken)

```

```

{
    await botClient.SendTextMessageAsync(
        chatId: chatId,
        text: $"botserver_standard error. message.Text is null?",
        cancellationToken: cancellationToken);
}

```

```

#region кнопки

```

```

private void StopBotBtn_Click(object sender, RoutedEventArgs e)

```

```

{
    Stats.ShutdownTimeFixator();
    Stats.UpTimeWriter();
    TgBot.MainBotCts.Cancel();
    LiveLogOutput.Clear();
    LiveLogOutput.Text = "Бот был остановлен.";
}

```

```

    }

    private void StopExitBotBtn_Click(object sender, RoutedEventArgs e)
    {
        Stats.ShutdownTimeFixator();
        Stats.UpTimeWriter();
        TgBot.MainBotCts.Cancel();
        Environment.Exit(0);
    }

    private void CmdOpenBtn_Click(object sender, RoutedEventArgs e)
    {
        Task.Factory.StartNew(() => ConsoleWorker.CardOutputter());
    }

    private void LogExportBtn_Click(object sender, RoutedEventArgs e)
    {
        string pathPart = $"livelog_{DateTime.Now}.txt".Replace(":", "_");
        string parserOutPath = Settings.baseLogPath + pathPart;
        StreamWriter parserExport = new(parserOutPath);
        parserExport.WriteLine(LiveLogOutput.Text);
        parserExport.Close();
    }

    private void LogClearBtn_Click(object sender, RoutedEventArgs e)
    {
        LiveLogOutput.Clear();
    }
    #endregion
}
}

```

Содержимое файла MainWindow.xaml.cs:

```

using System;
using System.IO;
using System.Runtime.InteropServices;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Media;

namespace botserver_standard
{
    public partial class MainWindow : Window
    {
        //импорт библиотек для запуска консоли
        [DllImport("kernel32.dll")]
        public static extern bool AllocConsole();
        [DllImport("kernel32.dll")]
        public static extern bool FreeConsole();

        public MainWindow()
        {
            InitializeComponent();

            if (Settings.pwdIsUsing is true)
            {
                AskingPassword askPwd = new();
                if (askPwd.ShowDialog() == true)
                {
                    MessageBox.Show("Добро пожаловать!");
                    UseThisPwdCheckbox.IsChecked = false; //обновление состояния чекбокса
                }
            }
            else
            {

```

В ОКНЕ



```

        MessageBox.Show("Операция отменена.");
        Environment.Exit(0);
    }
}

UseThisPwdCheckbox.IsChecked = Settings.pwdIsUsing;

if (Settings.pwd is
"YtcPoTIZPA0WpUdc~SMCaTjL7Kvt#ne3k*{Tb|H2Kx4t227gXy") // setting new pwd if now
setted default
{
    ChangeDefaultPwd changeDefaultPwd = new();
    changeDefaultPwd.ShowDialog();
    if (this.DialogResult is true)
    {
        UseThisPwdCheckbox.IsChecked = true;
    }
}
SetPwdBox.MaxLength = 50;
SetRepeatedPwdBox.MaxLength = 50;
Task.Factory.StartNew(() => CardParser(DbWorker.sqliteConn)); //ok
}

private void TabControl_SelectionChanged(object sender,
System.Windows.Controls.SelectionChangedEventArgs e)
{
    if (mainTab.IsSelected)
    {
        HomePic.Foreground = Brushes.RoyalBlue;
        ParserPic.Foreground = Brushes.LightGray;
        ParsedCardsPic.Foreground = Brushes.LightGray;
        ParsedUnivsPic.Foreground = Brushes.LightGray;
        SettingsPic.Foreground = Brushes.LightGray;
        AboutPic.Foreground = Brushes.LightGray;
    }
}

```

```
}
```

```
if (parserTab.IsSelected)
```

```
{
```

```
    HomePic.Foreground = Brushes.LightGray;
```

```
    ParserPic.Foreground = Brushes.RoyalBlue;
```

```
    ParsedCardsPic.Foreground = Brushes.LightGray;
```

```
    ParsedUnivsPic.Foreground = Brushes.LightGray;
```

```
    SettingsPic.Foreground = Brushes.LightGray;
```

```
    AboutPic.Foreground = Brushes.LightGray;
```

```
}
```

```
if (parsedCardsTab.IsSelected)
```

```
{
```

```
    HomePic.Foreground = Brushes.LightGray;
```

```
    ParserPic.Foreground = Brushes.LightGray;
```

```
    ParsedCardsPic.Foreground = Brushes.RoyalBlue;
```

```
    ParsedUnivsPic.Foreground = Brushes.LightGray;
```

```
    SettingsPic.Foreground = Brushes.LightGray;
```

```
    AboutPic.Foreground = Brushes.LightGray;
```

```
}
```

```
if (parsedUniversitiesTab.IsSelected)
```

```
{
```

```
    HomePic.Foreground = Brushes.LightGray;
```

```
    ParserPic.Foreground = Brushes.LightGray;
```

```
    ParsedCardsPic.Foreground = Brushes.LightGray;
```

```
    ParsedUnivsPic.Foreground = Brushes.RoyalBlue;
```

```
    SettingsPic.Foreground = Brushes.LightGray;
```

```
    AboutPic.Foreground = Brushes.LightGray;
```

```
}
```

```
if (settingsTab.IsSelected)
```

```
{
```

```
HomePic.Foreground = Brushes.LightGray;
ParserPic.Foreground = Brushes.LightGray;
ParsedCardsPic.Foreground = Brushes.LightGray;
ParsedUnivsPic.Foreground = Brushes.LightGray;
SettingsPic.Foreground = Brushes.RoyalBlue;
AboutPic.Foreground = Brushes.LightGray;
}

if (aboutTab.IsSelected)
{
    HomePic.Foreground = Brushes.LightGray;
    ParserPic.Foreground = Brushes.LightGray;
    ParsedCardsPic.Foreground = Brushes.LightGray;
    ParsedUnivsPic.Foreground = Brushes.LightGray;
    SettingsPic.Foreground = Brushes.LightGray;
    AboutPic.Foreground = Brushes.RoyalBlue;
}

}

}
```

Содержимое файла Parser.cs:

```
using HtmlAgilityPack;
using Microsoft.Data.Sqlite;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Threading;

namespace botserver_standard
{
    public partial class MainWindow : Window
    {
        List<Card> cardsView = new();
        List<UniversityEntryFreq> universityFreqListView = new();

        public async void CardParser(SqliteConnection sqliteConn)
        {

            Dispatcher.Invoke(() =>
            {
                ParserLogOutput.AppendText("-----\n");
                ParserLogOutput.Text += $"{DateTime.Now} | Парсер запущен...\n";
            });

            Dispatcher.Invoke(() =>
            {
                ParserLogOutput.Text += $"{DateTime.Now} | Получение данных из
сети...\n";

            });

            var parsingUrl = "https://studyintomsk.ru/programs-main/";
```

```

var web = new HtmlWeb();
HtmlDocument document;

document = web.Load(parsingUrl); //loading html
/*
/html/body/div[2]/div/div[3]/div[5]/select - программы подготовки
/html/body/div[3]/div[3] - карточки со сдвигами
/html/body/div[2]/div/div[3]/div[3]/select - вузы
/html/body/div[2]/div/div[3]/div[1]/select - уровни
/html/body/div[2]/div/div[3]/div[4]/select - языки
*/

Dispatcher.Invoke(() =>
{
    ParserLogOutput.Text += $"{DateTime.Now} | Выбор узлов...\n";
});
if (document is null)
{
    return;
}

var cardsValue =
document.DocumentNode.SelectNodes("/html/body/div[3]/div[3]/div/div/div/div/div");

string noTabsDoc = string.Empty; //первичная строка с сырыми данными

Dispatcher.Invoke(() =>
{
    ParserLogOutput.Text += $"{DateTime.Now} | Обработка полученных
данных...\n";
});
foreach (var item in cardsValue)
{
    noTabsDoc += item.InnerText; //node is single row?
}

```

```

noTabsDoc = noTabsDoc.Replace("\t", "\n"); //замена табуляций

List<string> cardsList = new(); //лист с правильными данными, идущими
по ряд
cardsList = noTabsDoc.Split('\n').ToList(); //построчная запись данных (в том
числе и мусора)

//удаление мусора из листа
Dispatcher.Invoke(() =>
{
    ParserLogOutput.Text += $"{DateTime.Now} | Очистка полученных
данных...\n";
});
#region data cleaning

string itemToRemove = "Уровень обучения";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Форма обучения";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Код программы ";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Продолжительность";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Степень или квалификация";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Язык обучения";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Куратор";

```

```

cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "за год обучения";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Поступить";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Нет подходящей программы?";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Напишите нам об этом и мы придумаем для вас
индивидуальное решение.";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Получить решение";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "Основная программа О программе";
cardsList.RemoveAll(x => x == itemToRemove);

//symbols
itemToRemove = "\n";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = "";
cardsList.RemoveAll(x => x == itemToRemove);

itemToRemove = string.Empty;
cardsList.RemoveAll(x => x == itemToRemove);
#endregion

//строки для передачи в атрибуты экземпляра класса
int Id = 0; // идентификатор экземпляра класса. Задать в бд?

```

```

//для прыжков по строкам всех карточек в листе
int row0 = 0;
int row1 = 1;
int row2 = 2;
int row3 = 3;
int row4 = 4;
int row5 = 5;
//int row6 = 6; //пропуск повторяющегося атрибута
int row7 = 7;
int row8 = 8;
int row9 = 9;
int row10 = 10;
int row11 = 11;

int cardCounter = 0;
int cardsTotalRows = cardsList.Count / 12;

Dispatcher.Invoke(() =>
{
    ParserLogOutput.Text += $"{DateTime.Now} | Запись данных...\n";
});
foreach (string line in cardsList)
{
    Card.cards.Add(new Card(Id, cardsList[row0], cardsList[row1],
cardsList[row2],
cardsList[row3], cardsList[row4], cardsList[row5],
cardsList[row7], cardsList[row8], cardsList[row9],
cardsList[row10], cardsList[row11]));
    Id++;

    //прыжок на строки следующей карточки
    row0 += 12;
    row1 += 12;

```



```

row2 += 12;
row3 += 12;
row4 += 12;
row5 += 12;
//row6 += 12; //пропуск повторяющегося атрибута
row7 += 12;
row8 += 12;
row9 += 12;
row10 += 12;
row11 += 12;

cardCounter++;
if (cardCounter == cardsTotalRows)
    break;
}

string clearCardsDb = "DELETE FROM Cards;";
DbWorker.DbQuerySilentSender(DbWorker.sqliteConn, clearCardsDb);

//запись полученных карточек в бд
foreach (var item in Card.cards)
{
    string cardsToDb = $"INSERT INTO Cards(id, universityName, programName,
programCode, level, studyForm, duration, studyLang, curator, phoneNumber, email, cost) " +
        $"VALUES('{item.Id}', '{item.UniversityName}', '{item.ProgramName}',
'{item.ProgramCode}', '{item.Level}', '{item.StudyForm}', '{item.Duration}',
'{item.StudyLang}', '{item.Curator}', '{item.PhoneNumber}', '{item.Email}', '{item.Cost}')";
    DbWorker.DbQuerySilentSender(DbWorker.sqliteConn, cardsToDb);
}

//Вывод данных из бд на вкладку карточек
int id;
string? universityName;

```

```

string? programName;
string? level;
string? programCode;
string? studyForm;
string? duration;
string? studyLang;
string? curator;
string? phoneNumber;
string? email;
string? cost;
string queryText = "SELECT * FROM Cards";

sqliteConn.Open(); //открытие соединения
SqlCommand command = new() //инициализация экземпляра SqlCommand
{
    Connection = sqliteConn, //соединение для выполнения запроса
    CommandText = queryText //текст запроса
};
SqlDataReader reader = command.ExecuteReader();

if (reader.HasRows) // если есть строки
{
    while (reader.Read()) // построчное чтение данных
    {
        //public Card(int id, string universityName, string programName, string level,
        string studyForm, string programCode, string duration, string studyLang, string curator, string
        phoneNumber, string email, string cost)

        id = Convert.ToInt32(reader["Id"]);
        universityName = Convert.ToString(reader["universityName"]);
        programName = Convert.ToString(reader["programName"]);
        programCode = Convert.ToString(reader["programCode"]);
        level = Convert.ToString(reader["level"]);
        studyForm = Convert.ToString(reader["studyForm"]);
    }
}

```

```

duration = Convert.ToString(reader["duration"]);
studyLang = Convert.ToString(reader["studyLang"]);
curator = Convert.ToString(reader["curator"]);
phoneNumber = Convert.ToString(reader["phoneNumber"]);
email = Convert.ToString(reader["email"]);
cost = Convert.ToString(reader["cost"]);

cardsView.Add(new Card(id, universityName, programName, level,
studyForm, programCode, duration, studyLang, curator, phoneNumber, email, cost));
    }
}
sqliteConn.Close();

//выделение уникальных вузов
List<string> universitiesList = new();
int universityRow = 0;
int rowCounter = 0;
foreach (string line in cardsList)
{

    universitiesList.Add(cardsList[universityRow]);
    universityRow += 12;
    rowCounter++;
    if (rowCounter >= cardsTotalRows)
        break;
}

foreach (string item in universitiesList.Distinct())
{
    UniversityEntryFreq.universitiesFreqList.Add(new UniversityEntryFreq(item,
universitiesList.Where(x => x == item).Count()));
}

string clearUniversitiesFreqDb = "DELETE FROM Universities;";

```

```

        DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
clearUniversitiesFreqDb);
        id = 0;
        //запись полученных карточек в бд
        foreach (var item in UniversityEntryFreq.universitiesFreqList)
        {
            string universitiesToDb = $"INSERT INTO Universities(id, universityName,
universityCount) " +
                $"VALUES('{id}', '{item.UniversityName}', '{item.Count}')";
            DbWorker.DbQuerySilentSender(DbWorker.sqliteConn, universitiesToDb);
            id++;
        }

        //вывод данных из бд на вкладку карточек

        string? freqUniversityName;
        int freqUniversityCount;

        queryText = "SELECT * FROM Universities";

        sqliteConn.Open(); //открытие соединения
        SqliteCommand freqCommand = new() //инициализация экземпляра
        SqliteCommand
        {
            Connection = sqliteConn, //соединение для выполнения запроса
            CommandText = queryText //текст запроса
        };
        SqlDataReader freqReader = freqCommand.ExecuteReader();

        if (freqReader.HasRows) // если есть строки
        {
            while (freqReader.Read()) // построчное чтение данных
            {
                freqUniversityName = Convert.ToString(freqReader["universityName"]);
            }
        }
    }
}

```

```

freqUniversityCount = Convert.ToInt32(freqReader["universityCount"]);

        universityFreqListView.Add(new UniversityEntryFreq(freqUniversityName,
freqUniversityCount));
    }
}
sqliteConn.Close();

Dispatcher.Invoke(() =>
{
    parsedCardsGrid.ItemsSource = cardsView;
});
Dispatcher.Invoke(() =>
{
    parsedUniversitiesGrid.ItemsSource = universityFreqListView;
});
Dispatcher.Invoke(() =>
{
    ParserLogOutput.Text += $"{{DateTime.Now}} | Парсинг
окончен.\n{{Card.cards.Count}} карточек было добавлено;\n{{universityFreqListView.Count}}
университетов было добавлено.\n";
    ParserLogOutput.Text += "-----
-----\n";

});
}
}
}

```

Содержимое файла ParserTab.cs:

```
using System;
using System.IO;
using System.Threading.Tasks;
using System.Windows;

namespace botserver_standard
{
    public partial class MainWindow : Window
    {
        private void ParserPeparseBtn_Click(object sender, RoutedEventArgs e)
        {
            universityFreqListView.Clear();
            Card.cards.Clear();
            Dispatcher.Invoke(() =>
            {
                ParserLogOutput.Text += $"{DateTime.Now} | Reparsing...\n";
            });
            Task.Factory.StartNew(() => CardParser(DbWorker.sqliteConn)); //ok
        }

        private void ParserExportBtn_Click(object sender, RoutedEventArgs e)
        {
            //string parserOutPath = Settings.baseLogPath + $"parser_{DateTime.Now}.txt";
            string pathPart = $"parser_{DateTime.Now}.txt".Replace(":", "_");
            string parserOutPath = Settings.baseLogPath + pathPart;
            StreamWriter parserExport = new(parserOutPath);
            parserExport.WriteLine(ParserLogOutput.Text);
            parserExport.Close();
        }
    }
}
```

Содержимое файла Settings.cs:

```
namespace botserver_standard
{
    internal class Settings
    {
        public static string? fileLoggerPath = null; // путь к логу (добавить изменение пути
лога в интерфейсе?)
        public static string? callbackLoggerPath = null;
        public static string? connString = null; //путь к бд. setted in app!
        public static string? botToken = null; //токен бота
        public static string? pwd; //пароль на запуск. Может быть отключен, см. ниже
        public static bool pwdIsUsing = false; //пароль используется(чекбокс)?
        public static string? datagridExportPath = null;
        public static string? parsingUrl = null; //URL страницы, подлежащей парсингу

        public static string? baseLogPath =
"C:\\Users\\creat\\source\\repos\\botserver_standard\\bin\\Debug\\net6.0-windows\\logs\\";
    }
}
```

Содержимое файла SettingsTab.cs:

```
using Microsoft.Data.Sqlite;
using System;
using System.Windows;

namespace botserver_standard
{
    public partial class MainWindow : Window
    {
        private void SetTokenBtn_Click(object sender, RoutedEventArgs e) //ok
        {
```

```

        string query = $"UPDATE Settings SET botToken =
'{SettingsTokenInput.Text.Trim(' ')}';";
        int rowsChanged = DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
query);

        //IsChanged?
        if (rowsChanged is 1)
        {
            using SqliteDataReader reader =
DbWorker.SettingsReader(DbWorker.readSettings, DbWorker.sqliteConn); //from db to fields

            string tokenLeftPart = SettingsTokenInput.Text[..^35]; //
:AAF3nNDIYNfryOulNHKtsxlhuGo_roxXYXI
            SettingsTokenInput.Text = $"Token has been changed to {tokenLeftPart}";
        }
        else
        {
            SettingsTokenInput.Text = "Unforeseen error";
        }
    }

    private void SetMessageLogPathBtn_Click(object sender, RoutedEventArgs e) //ok
    {
        string query = $"UPDATE Settings SET fileLoggerPath =
'{MessagesLogRootInput.Text}';";
        int rowsChanged = DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
query);

        //IsChanged?
        if (rowsChanged is 1)
        {
            using SqliteDataReader reader =
DbWorker.SettingsReader(DbWorker.readSettings, DbWorker.sqliteConn);

```



```

        MessagesLogRootInput.Text = $"file path has been setted.";
    }
    else
    {
        MessagesLogRootInput.Text = "Unforseen error";
    }
}

private void SetDatagridExportPathBtn_Click(object sender, RoutedEventArgs e)
{
    string query = $"UPDATE Settings SET datagridExportPath =
'{datagridExportPath.Text}';";
    int rowsChanged = DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
query);

    //IsChanged?
    if (rowsChanged is 1)
    {
        using SqlDataReader reader =
DbWorker.SettingsReader(DbWorker.readSettings, DbWorker.sqliteConn);

        datagridExportPath.Text = $"file path has been setted.";
    }
    else
    {
        datagridExportPath.Clear();
        datagridExportPath.Text += "Unforseen error";
    }
}

private void ChoicesLogRootInputSetBtn_Click(object sender, RoutedEventArgs e)
{

```

```

        string query = $"UPDATE Settings SET callbackLoggerPath =
'{ChoicesLogRootInput.Text}';";
        int rowsChanged = DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
query);

        //IsChanged?
        if (rowsChanged is 1)
        {
            using SqlDataReader reader =
DbWorker.SettingsReader(DbWorker.readSettings, DbWorker.sqliteConn);

            ChoicesLogRootInput.Text = $"Choices log path has been setted.";
        }
        else
        {
            ChoicesLogRootInput.Text = "Unforeseen error";
        }
    }

    private void ParsingUrlSetBtn_Click(object sender, RoutedEventArgs e)
    {
        string query = $"UPDATE Settings SET parsingUrl = '{UrlSet.Text}';";
        int rowsChanged = DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
query);

        //IsChanged?
        if (rowsChanged is 1)
        {
            using SqlDataReader reader =
DbWorker.SettingsReader(DbWorker.readSettings, DbWorker.sqliteConn);

            ChoicesLogRootInput.Text = $"Parsing URL has been setted.";
        }
        else

```

```

        {
            ChoicesLogRootInput.Text = "Unforeseen error";
        }
    }

    private void SetPwdBtn_Click(object sender, RoutedEventArgs e)
    {
        if (SetPwdBox.Password.Length > 3 && SetRepeatedPwdBox.Password.Length
        > 3 && SetPwdBox.Password == SetRepeatedPwdBox.Password)
        {
            int rowsChanged;
            StupidWall stupidWall = new();

            if (stupidWall.ShowDialog() == true) // if checking is success
            {
                if (stupidWall.EnterPwdBox.Password == Settings.pwd)
                {
                    MessageBox.Show("Авторизация пройдена");

                    string updatePwdQuery = $"UPDATE Settings SET pwd =
                    '{SetPwdBox.Password}';";

                    rowsChanged = DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
                    updatePwdQuery); // обновление бд

                    //IsChanged?
                    if (rowsChanged is 1)
                    {
                        using SqliteDataReader reader =
                        DbWorker.SettingsReader(DbWorker.readSettings, DbWorker.sqliteConn); //обновление
                        локальных настроек из бд

                        PwdChangeNotifier.Content = $"Your password has been changed at
                        {DateTime.Now}.";
                    }
                }
            }
        }
    }

```

```

    }
    else { PwdChangeNotifier.Content = "Unforeseen error. Use old password.";
}

    }

    if (stupidWall.EnterPwdBox.Password != Settings.pwd)
    {
        MessageBox.Show("Неверный пароль"); //if checking is failed
    }
}

else
{
    MessageBox.Show("Cancelled"); // if cancelled
}
}

else
{
    MessageBox.Show("Passwords are not the same, try again", "Error");
    SetRepeatedPwdBox.Clear();
}
}

public void UseThisPwdCheckbox_Checked(object sender, RoutedEventArgs e)
{
    string setCheckboxQuery = $"UPDATE Settings SET pwdIsUsing = 'True'";
    DbWorker.DbQuerySilentSender(DbWorker.sqliteConn, setCheckboxQuery); //
обновление бд

    using          SqlDataReader          reader          =
    DbWorker.SettingsReader(DbWorker.readSettings,    DbWorker.sqliteConn);    //обновление
    локальных настроек из бд

}

```

```

private void UseThisPwdCheckbox_Unchecked(object sender, RoutedEventArgs e)
{
    StupidWall stupidWall = new();

    if (stupidWall.ShowDialog() == true)
    {

        MessageBox.Show("Запрос пароля отключен.");
        UseThisPwdCheckbox.IsChecked = false; //обновление состояния чекбокса в
окне

        string updatePwdIsUsingQuery = $"UPDATE Settings SET pwdIsUsing =
'False';";

        DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
updatePwdIsUsingQuery); //обновление состояния чекбокса в бд
    }
    else
    {
        MessageBox.Show("Операция отменена.");
        UseThisPwdCheckbox.IsChecked = true;
    }
}
}
}

```

Содержимое файла Stats.cs:

```

using System;
namespace botserver_standard
{
    public static class Stats
    {
        static DateTime startupTime;
    }
}

```

```

static DateTime shutdownTime;
public static void StartupTimeFixator()
{
    startupTime = DateTime.Now;
}

public static void ShutdownTimeFixator()
{
    shutdownTime = DateTime.Now;
}
public static void UpTimeWriter()
{
    TimeSpan upTime = shutdownTime - startupTime;
    string query = $"INSERT INTO Session_duration(startup_time, shutdown_time,
total_uptime)" +
        $"VALUES('{startupTime}', '{shutdownTime}', '{upTime}')";
    DbWorker.DbQuerySilentSender(DbWorker.sqliteConn, query);
}
}
}

```

Содержимое файла StupidWall.xaml.cs:

```

using System.Windows;

namespace botserver_standard
{
    public partial class StupidWall : Window
    {
        public StupidWall()
        {
            InitializeComponent();
        }
    }
}

```

```

private void NextBtn_Click(object sender, RoutedEventArgs e)
{

    if (EnterPwdBox.Password == Settings.pwd) //если введенный пароль корректен
    {
        this.DialogResult = true;
    }
}

public string Password
{
    get { return EnterPwdBox.Password; }
}
}

```

Содержимое файла TelegramBot.cs:

```

using System.Threading;
using Telegram.Bot;

namespace botserver_standard
{
    internal class TgBot
    {
        public static TelegramBotClient botClient = new(Settings.botToken);
//инициализация клиента
        public static CancellationTokenSource MainBotCts = new();
    }
}

```

Содержимое файла TelegramBotKeypads.cs:

```
using Telegram.Bot.Types.ReplyMarkups;

namespace botserver_standard
{
    internal class TelegramBotKeypads
    {
        public static readonly InlineKeyboardMarkup mainMenuKeypad = new(
            new[]
            {
                // second row
                new[]
                {
                    InlineKeyboardButton.WithCallbackData(text: "Выбрать программу
обучения", callbackData: "programChoose"),
                    //InlineKeyboardButton.WithCallbackData(text: "Сменить язык",
callbackData: "langSwitch"),
                },
                // first row
                new[]
                {
                    InlineKeyboardButton.WithUrl(text: "Посетить веб-сайт проекта", url:
"https://studyintomsk.ru/"),
                    InlineKeyboardButton.WithUrl(text: "Проверить знание русского языка", url:
"https://studyintomsk.2i.tusur.ru/"),
                },
            });

        public static readonly InlineKeyboardMarkup levelChoosingKeypad = new(
            // keyboard
            new[]
            {
                // first row
```



```

        new[]
        {
            InlineKeyboardButton.WithCallbackData(text: "Бакалавриат", callbackData:
"бакалавриат_level"),
            InlineKeyboardButton.WithCallbackData(text: "Магистратура", callbackData:
"магистратура_level"),
            InlineKeyboardButton.WithCallbackData(text: "Специалитет", callbackData:
"специалитет_level"),
        },
        // second row
        new[]
        {
            InlineKeyboardButton.WithCallbackData(text: "🏠", callbackData: "toHome"),
        },

    });

    //old universities keypad
    public static readonly InlineKeyboardMarkup universityChoosingKeypad = new(
//MUST BE PARSED!!!
    // keyboard
    new[]
    {
        new[]
        {
            InlineKeyboardButton.WithCallbackData(text: "ТГАСУ", callbackData:
"ТГАСУ_university"),
            InlineKeyboardButton.WithCallbackData(text: "ТГПУ", callbackData:
"ТГПУ_university"),
            InlineKeyboardButton.WithCallbackData(text: "ТГУ", callbackData:
"ТГУ_university"),
        },
        new[]

```

```

        {
            InlineKeyboardButton.WithCallbackData(text: "ТПУ", callbackData:
"ТПУ_university"),
            InlineKeyboardButton.WithCallbackData(text: "ТУСУР", callbackData:
"ТУСУР_university"),
        },
        // third row
        new[]
        {
            InlineKeyboardButton.WithCallbackData(text: "🏠", callbackData: "toHome"),
        },
    });
}
}

```

Содержимое файла UniversityEntryFreq.cs:

```

using System.Collections.Generic;

namespace botserver_standard
{
    internal class UniversityEntryFreq
    {
        public static List<UniversityEntryFreq> universitiesFreqList = new();

        public string UniversityName { get; set; }
        public int Count { get; set; }

        public UniversityEntryFreq(string universityName, int count)
        {
            this.UniversityName = universityName;
            this.Count = count;
        }
    }
}

```

```
}
```

Содержимое файла UniversityProgramButton.cs:

```
namespace botserver_standard
{
    public class UniversityProgramButton
    {

        public int Id { get; set; } //buttonID = cardId (callbackData)
        public string Text { get; set; } //programName, buttonText

        public UniversityProgramButton(string programName, int id)
        {
            this.Id = id;
            this.Text = programName;
        }
    }
}
```

## Инструкция пользователя

После первого запуска информационной системы откроется окно установки нового пароля (Рисунок 16).

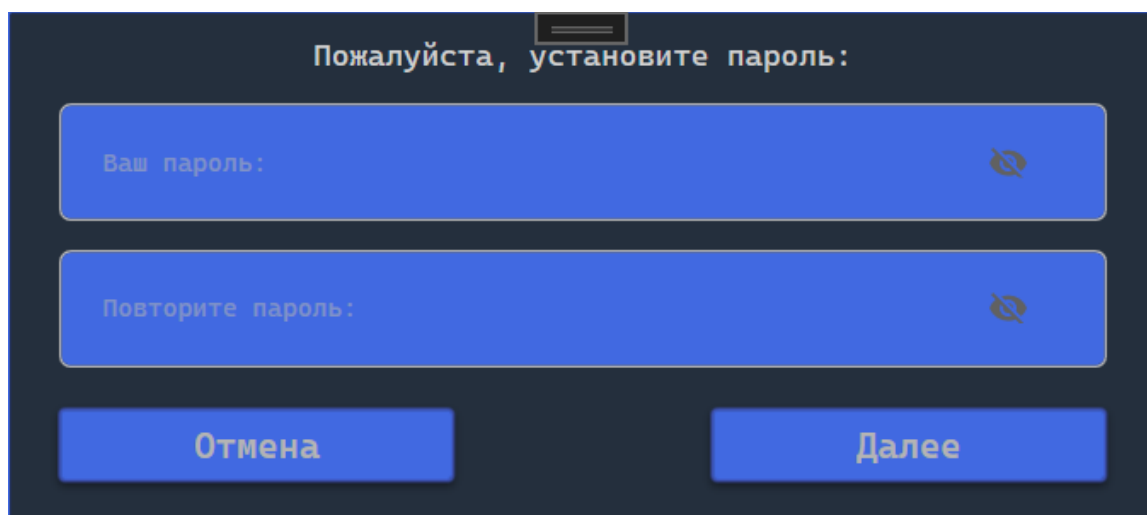


Рисунок 16 – Окно установки нового пароля

Далее необходимо ввести в соответствующие поля пароль и повторить, затем нажать кнопку «Далее» (Рисунок 17), после чего появится уведомление (Рисунок 18), откроется главное окно и во вкладке «Settings» флажок «Использовать этот пароль» будет установлен (Рисунок 19).

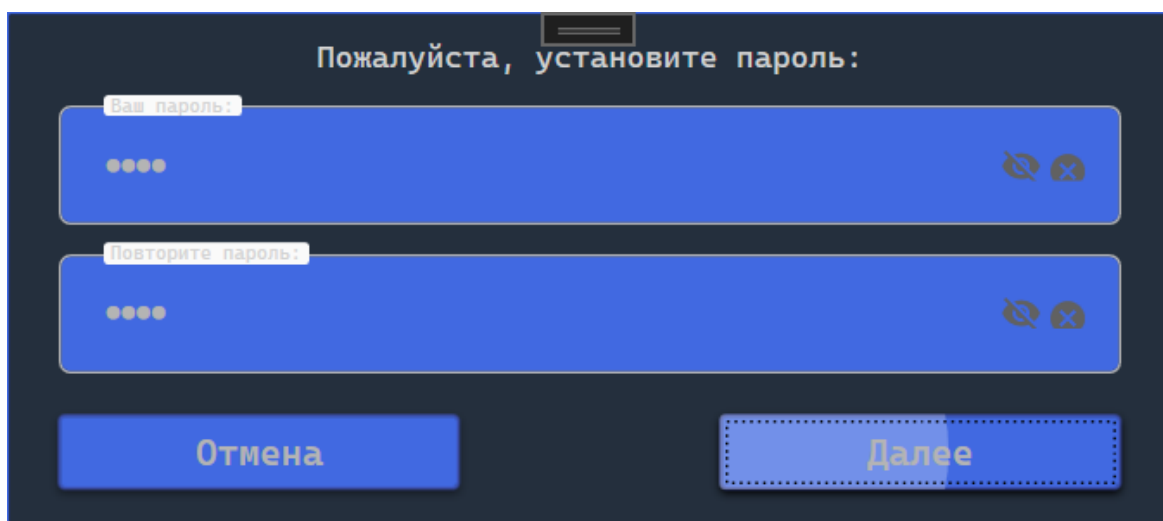


Рисунок 17 – Ввод пароля в окно установки нового пароля

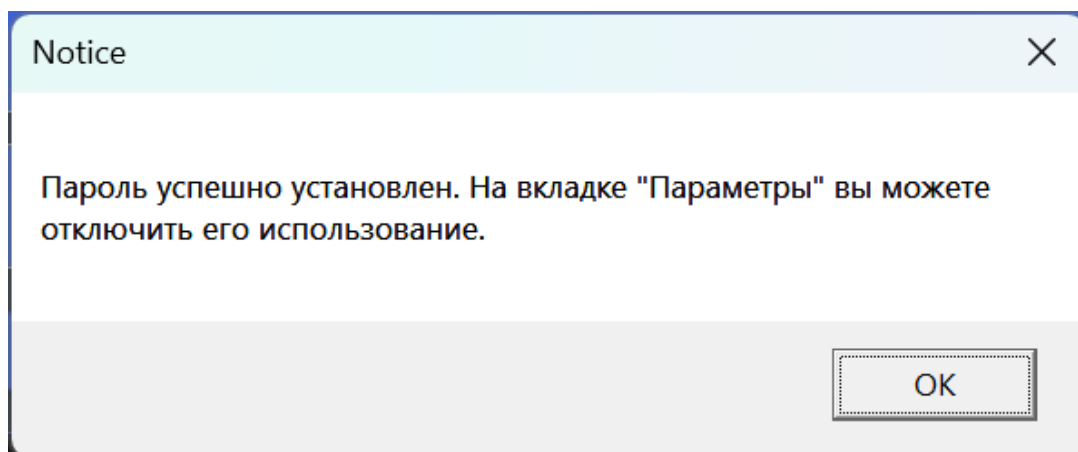


Рисунок 18 – Окно уведомления

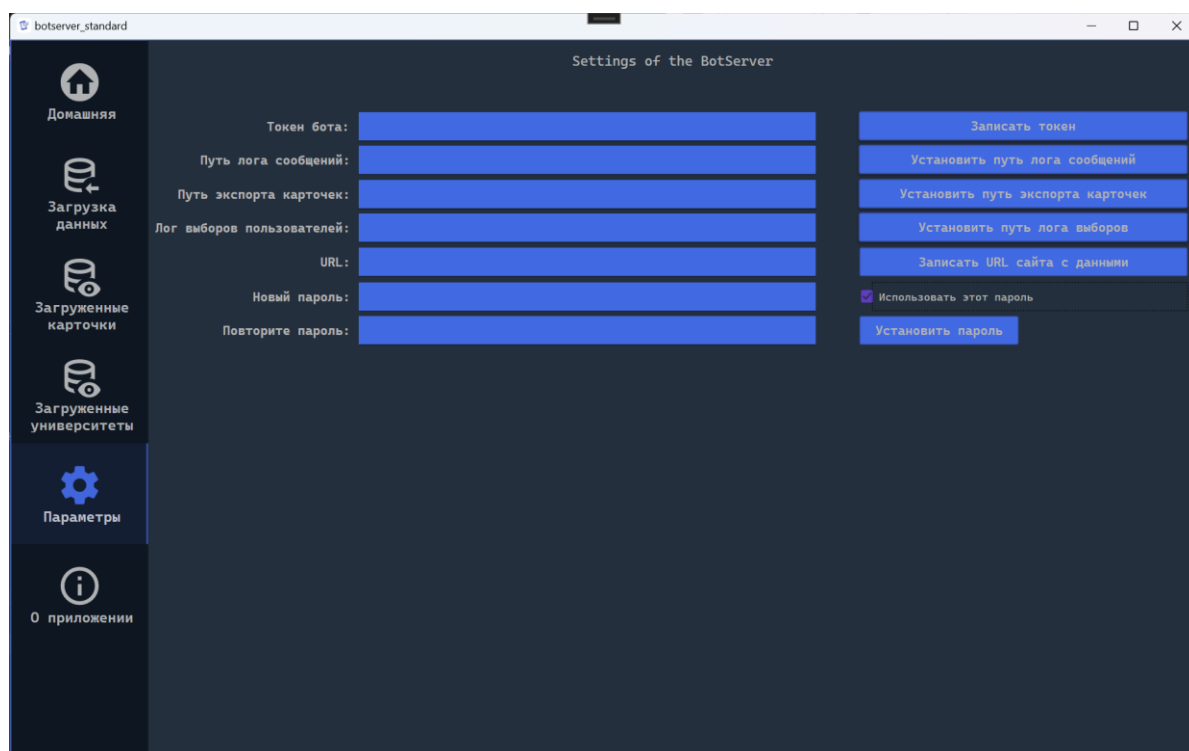


Рисунок 19 – Установленный флажок «Использовать этот пароль» во вкладке «Параметры»

Для запуска бота необходимо дождаться завершения работы парсера. Отслеживать журнал работы парсера можно во вкладке «Загрузка данных» (Рисунок 20).

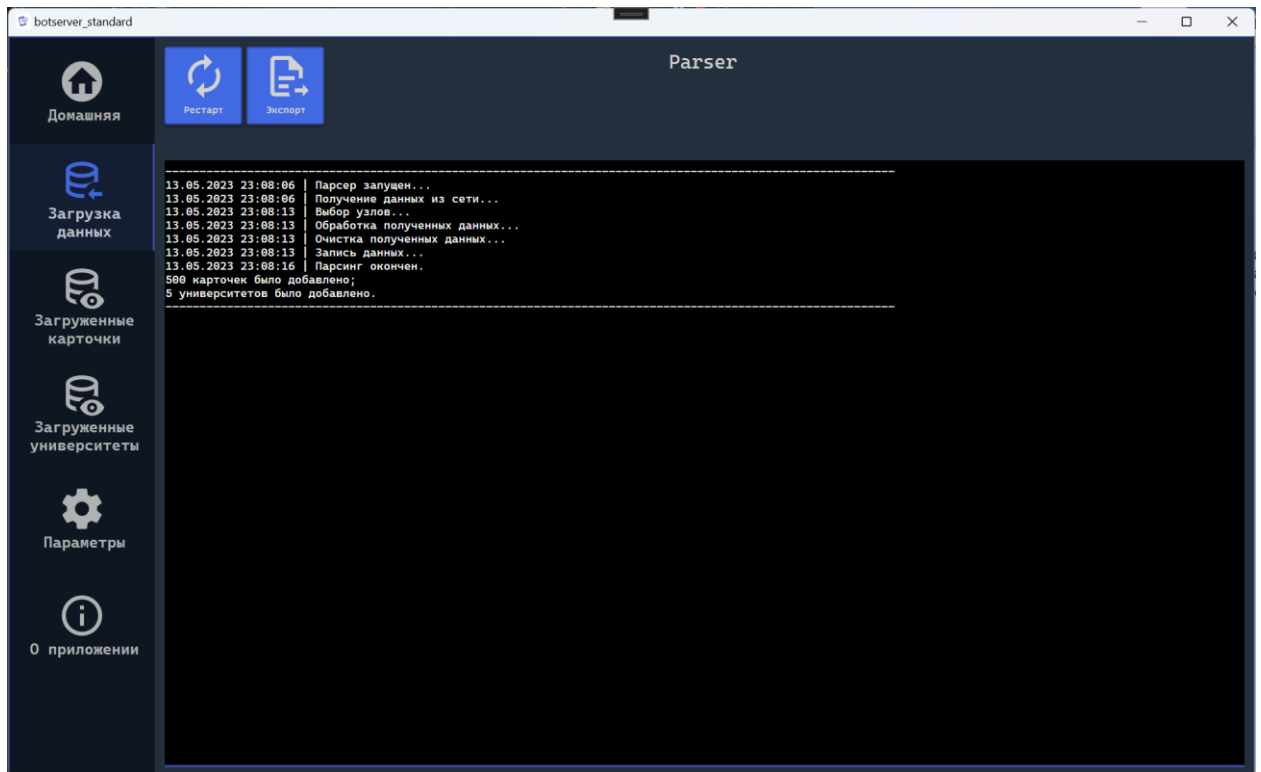


Рисунок 20 – Журнал парсера, сообщающий об окончании его работы

Также на этой вкладке можно перезапустить парсер для сбора свежих данных о карточках направлений, не перезапуская бота и экспортировать журнал парсера. Для обновления данных на вкладке присутствует кнопка «Рестарт». Это обновит полученные ранее данные карточек (Рисунок 21).

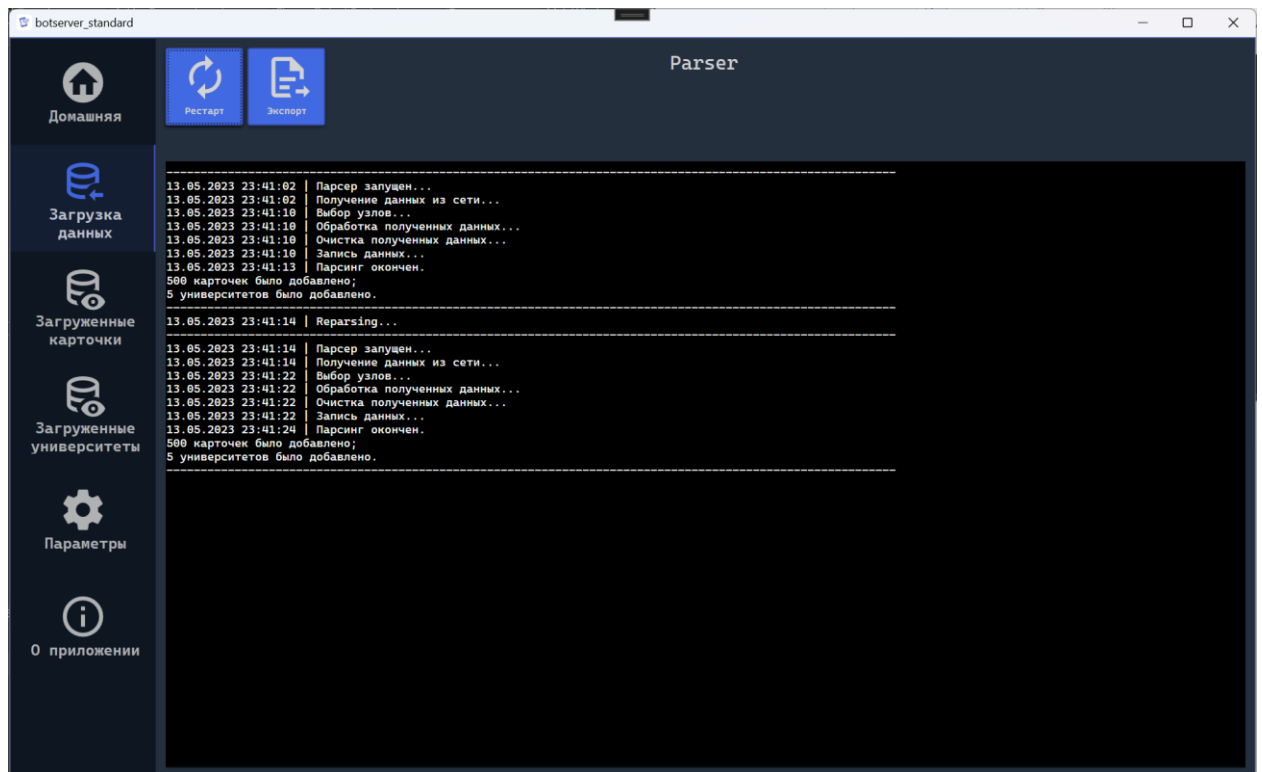


Рисунок 21 – Результат перезапуска парсера

После окончания работы парсера можно переходить к запуску бота. Для этого во вкладке «Домашняя» присутствует кнопка «Старт» (Рисунок 22).

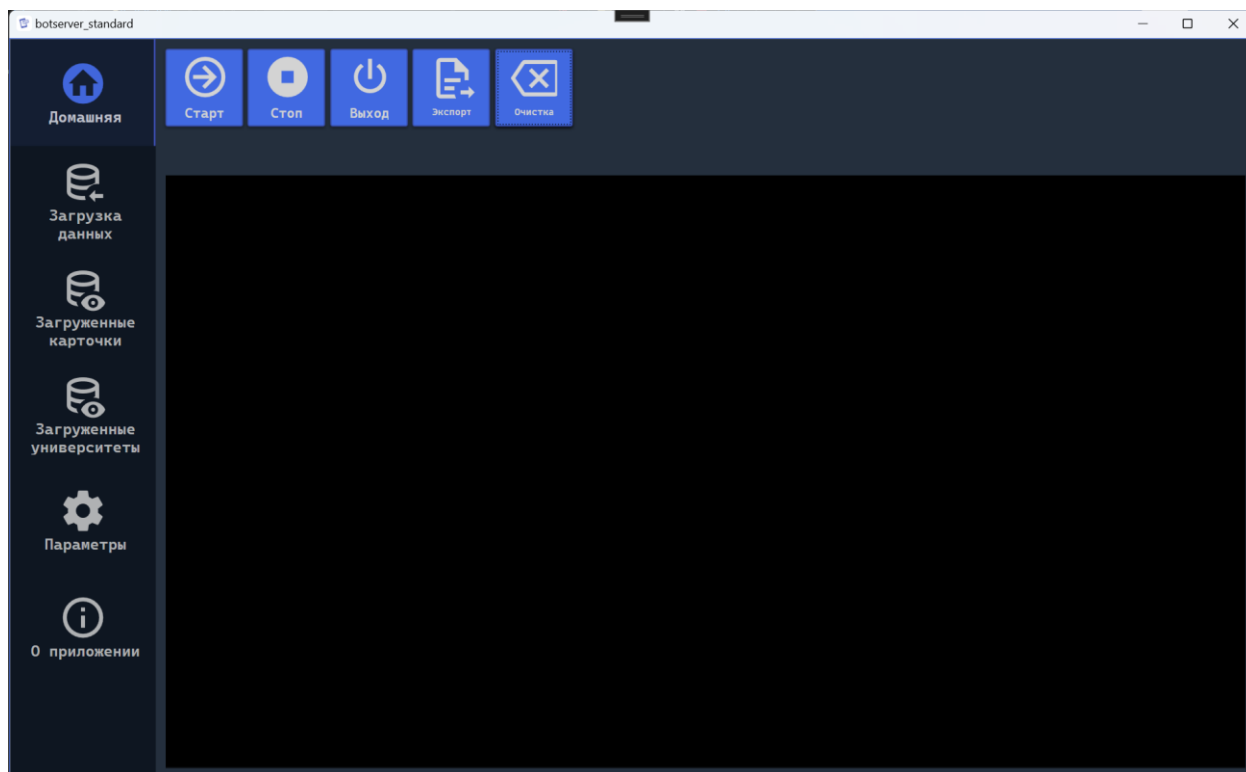


Рисунок 22 – Вкладка «Домашняя»

Также на этой вкладке есть кнопки «Стоп», «Выход», «Экспорт» и «Очистка»:

- «Стоп» - позволяет остановить прием ботом сообщений без закрытия GUI;
- «Выход» - останавливает бота и закрывает GUI;
- «Экспорт» - позволяет экспортировать текущий живой журнал;
- «Очистка» - позволяет очистить вывод.

Для работы всего функционала после первого запуска бота необходимо настроить. Настройка производится в рассмотренной ранее вкладке «Параметры» (Рисунок 23). В соответствующие поля вводятся необходимые данные и нажимаются кнопки записи для записи или обновления данных. По результату записи данных выведутся уведомления (Рисунок 24).



The screenshot shows the 'Settings of the BotServer' window. On the left is a sidebar with icons and labels: 'Домашняя' (Home), 'Загрузка данных' (Load data), 'Загруженные карточки' (Loaded cards), 'Загруженные университеты' (Loaded universities), 'Параметры' (Parameters) - which is highlighted with a blue gear icon, and '0 приложения' (0 applications) with an information icon. The main area contains several input fields and buttons:

- Токен бота:** A text field containing a long string of 'x' characters.
- Путь лога сообщений:** A text field containing 'messages.txt'.
- Путь экспорта карточек:** A text field containing 'cards.txt'.
- Лог выборов пользователей:** A text field containing 'choices.txt'.
- URL:** A text field containing 'https://studyintomsk.ru/programs-main/?level=card-item&direction=card-i'.
- Новый пароль:** An empty text field.
- Повторите пароль:** An empty text field.

On the right side of the main area, there are several buttons:

- Записать токен** (Save token)
- Установить путь лога сообщений** (Set message log path)
- Установить путь экспорта карточек** (Set card export path)
- Установить путь лога выборов** (Set user choice log path)
- Записать URL сайта с данными** (Save website URL with data)
- ☐ **Использовать этот пароль** (Use this password)
- Установить пароль** (Set password)

Рисунок 23 – Заполнение полей данными, необходимыми для работы бота

This screenshot shows the same 'Settings of the BotServer' window after the data has been saved. The input fields now contain confirmation messages:

- Токен бота:** 'Токен сменен на 000000000000' (Token changed to 000000000000).
- Путь лога сообщений:** 'Путь установлен.' (Path set).
- Путь экспорта карточек:** 'Путь установлен.' (Path set).
- Лог выборов пользователей:** 'Путь установлен.' (Path set).
- URL:** 'URL записан.' (URL saved).
- Новый пароль:** An empty text field.
- Повторите пароль:** An empty text field.

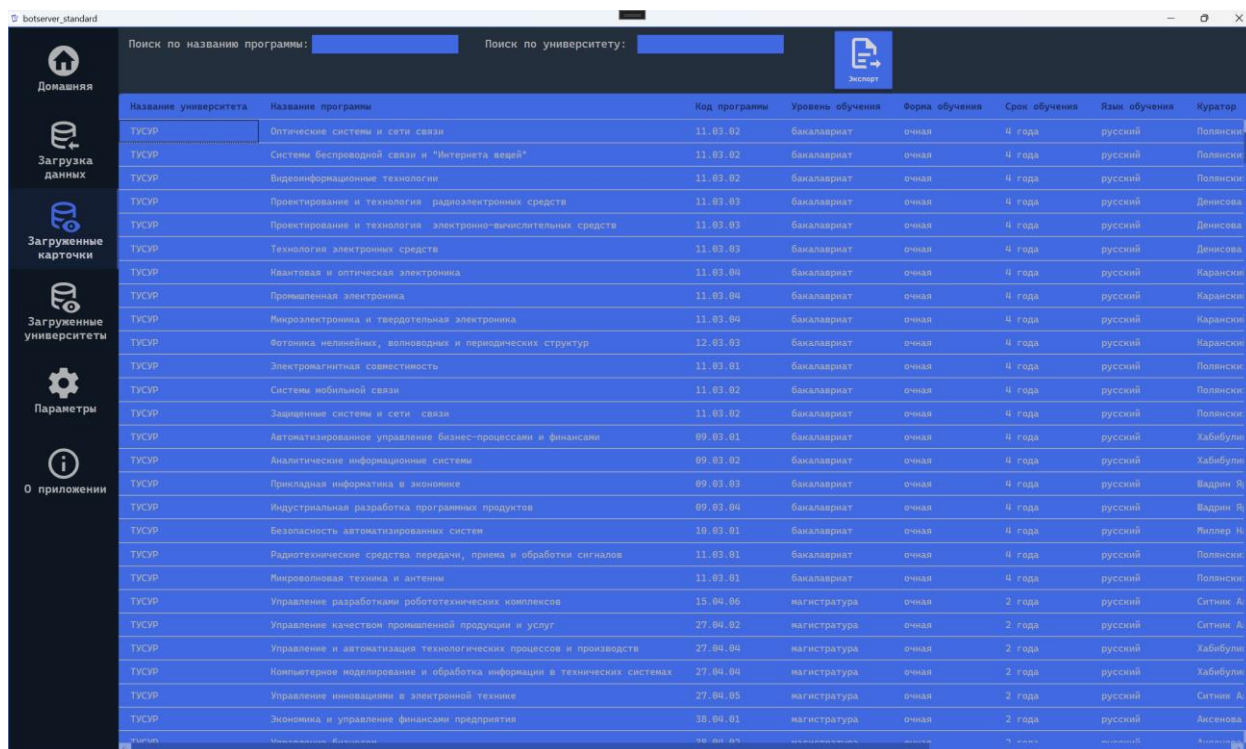
The buttons on the right remain the same as in the previous screenshot.

Рисунок 24 – Результат записи данных

На вкладке «Загруженные карточки» отображается список всех карточек направлений, которые получил парсер. Эти данные можно экспортировать посредством нажатия кнопки «Экспорт» (Рисунок 25).

Для поиска по карточкам присутствует два поля – «Поиск по названию программы» и «Поиск по университету».

Умный поиск был реализован с целью помочь администратору при необходимости найти определенную карточку и получить о ней исчерпывающую информацию. (Рисунок 26, Рисунок 27).

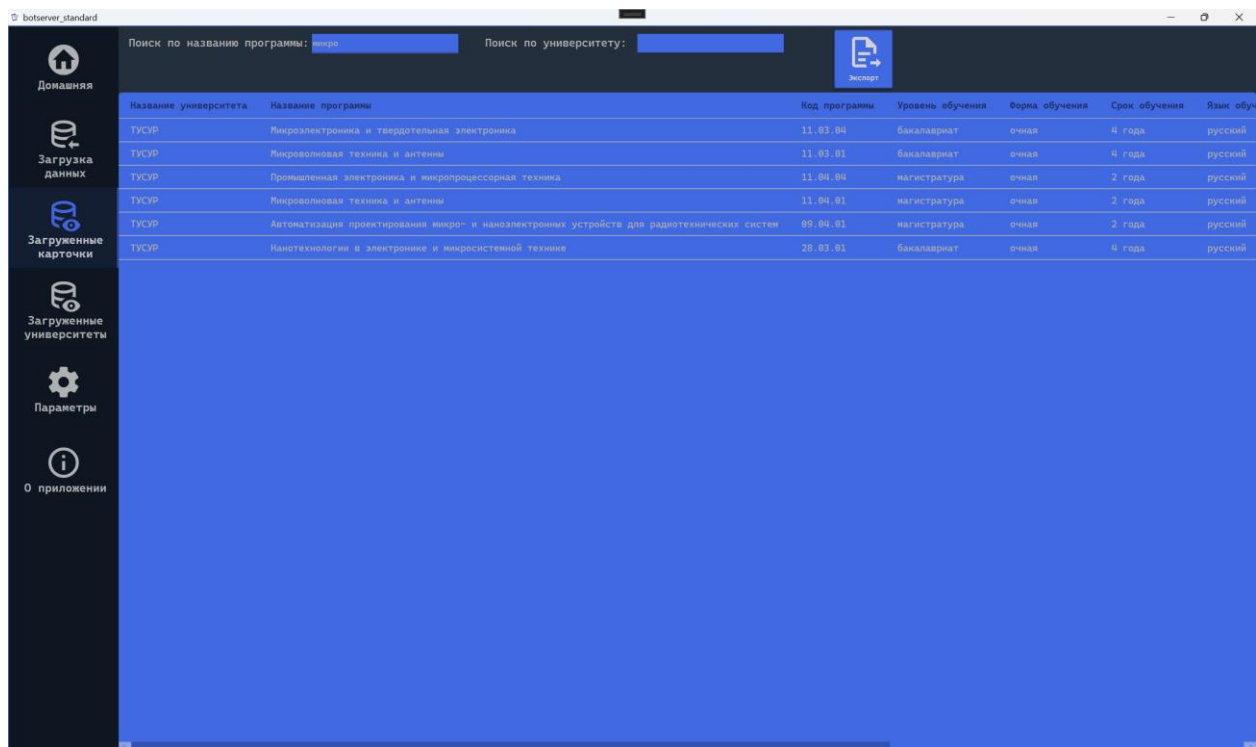


Поиск по названию программы:  Поиск по университету:  Экспорт

Название университета	Название программы	Код программы	Уровень обучения	Форма обучения	Срок обучения	Язык обучения	Куратор
ТУСУР	Оптические системы и сети связи	11.03.02	бакалавриат	очная	4 года	русский	Полански
ТУСУР	Системы беспроводной связи и "Интернета вещей"	11.03.02	бакалавриат	очная	4 года	русский	Полански
ТУСУР	Видеоинформационные технологии	11.03.02	бакалавриат	очная	4 года	русский	Полански
ТУСУР	Проектирование и технология радиоэлектронных средств	11.03.03	бакалавриат	очная	4 года	русский	Денисова
ТУСУР	Проектирование и технология электронно-вычислительных средств	11.03.03	бакалавриат	очная	4 года	русский	Денисова
ТУСУР	Технология электронных средств	11.03.03	бакалавриат	очная	4 года	русский	Денисова
ТУСУР	Квантовая и оптическая электроника	11.03.04	бакалавриат	очная	4 года	русский	Каранский
ТУСУР	Промышленная электроника	11.03.04	бакалавриат	очная	4 года	русский	Каранский
ТУСУР	Микроэлектроника и твердотельная электроника	11.03.04	бакалавриат	очная	4 года	русский	Каранский
ТУСУР	Фотоника нелинейных, волноводных и периодических структур	12.03.03	бакалавриат	очная	4 года	русский	Каранский
ТУСУР	Электромагнитная совместимость	11.03.01	бакалавриат	очная	4 года	русский	Полански
ТУСУР	Системы мобильной связи	11.03.02	бакалавриат	очная	4 года	русский	Полански
ТУСУР	Защитные системы и сети связи	11.03.02	бакалавриат	очная	4 года	русский	Полански
ТУСУР	Автоматизированное управление бизнес-процессами и финансами	09.03.01	бакалавриат	очная	4 года	русский	Хабидуллин
ТУСУР	Аналитические информационные системы	09.03.02	бакалавриат	очная	4 года	русский	Хабидуллин
ТУСУР	Прикладная информатика в экономике	09.03.03	бакалавриат	очная	4 года	русский	Вадим Я.
ТУСУР	Индустриальная разработка программных продуктов	09.03.04	бакалавриат	очная	4 года	русский	Вадим Я.
ТУСУР	Безопасность автоматизированных систем	10.03.01	бакалавриат	очная	4 года	русский	Миллер Н.
ТУСУР	Радиотехнические средства передачи, приема и обработки сигналов	11.03.01	бакалавриат	очная	4 года	русский	Полански
ТУСУР	Микроволновая техника и антенны	11.03.01	бакалавриат	очная	4 года	русский	Полански
ТУСУР	Управление разработкой робототехнических комплексов	15.04.06	магистратура	очная	2 года	русский	Ситник А.
ТУСУР	Управление качеством промышленной продукции и услуг	27.04.02	магистратура	очная	2 года	русский	Ситник А.
ТУСУР	Управление и автоматизация технологических процессов и производств	27.04.04	магистратура	очная	2 года	русский	Хабидуллин
ТУСУР	Компьютерное моделирование и обработка информации в технических системах	27.04.04	магистратура	очная	2 года	русский	Хабидуллин
ТУСУР	Управление инновациями в электронной технике	27.04.05	магистратура	очная	2 года	русский	Ситник А.
ТУСУР	Экономика и управление финансами предприятия	38.04.01	магистратура	очная	2 года	русский	Аксенова
ТУСУР	Математическая экономика	38.04.01	магистратура	очная	2 года	русский	Аксенова

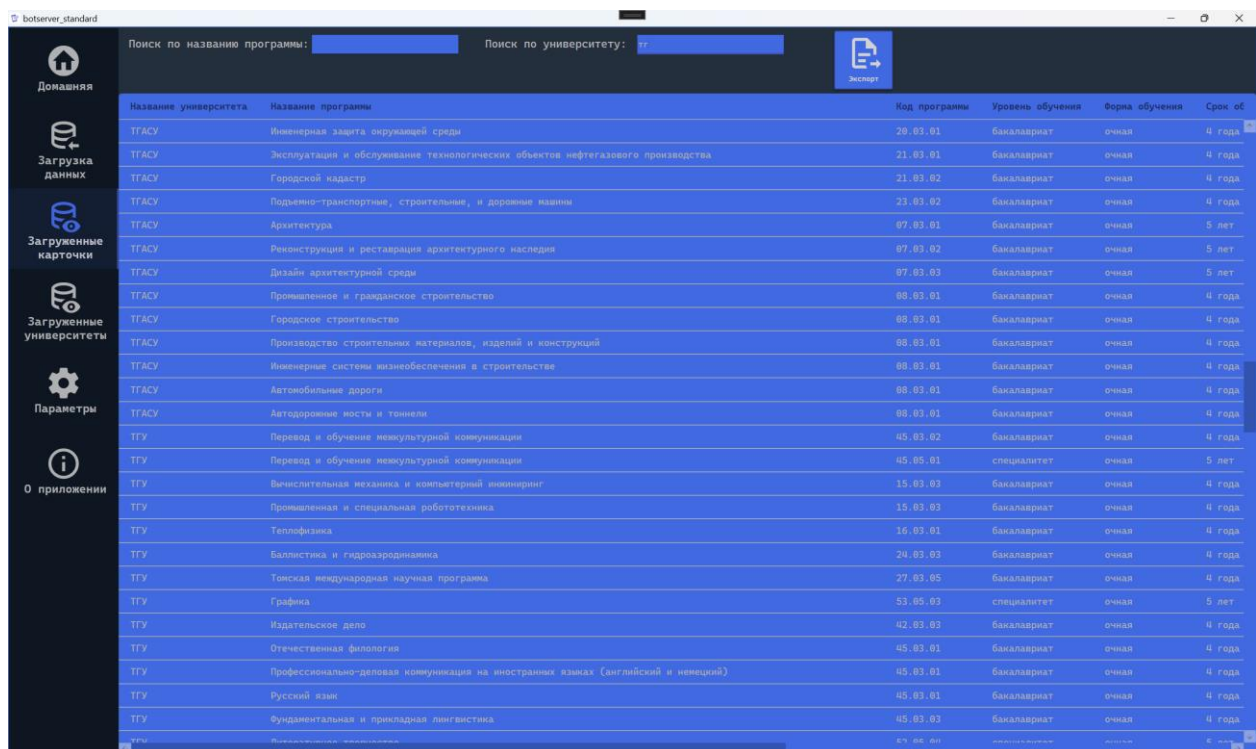
Рисунок 25 – общий вид окна «Загруженные карточки»

## ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б



Название университета	Название программы	Код программы	Уровень обучения	Форма обучения	Срок обучения	Язык обучения
TUSUR	Микроэлектроника и твердотельная электроника	11.03.04	бакалавриат	очная	4 года	русский
TUSUR	Микроволновая техника и антенны	11.03.01	бакалавриат	очная	4 года	русский
TUSUR	Промышленная электроника и микропроцессорная техника	11.04.04	магистратура	очная	2 года	русский
TUSUR	Микроволновая техника и антенны	11.04.01	магистратура	очная	2 года	русский
TUSUR	Автоматизация проектирования микро- и нанoeлектронных устройств для радиотехнических систем	09.04.01	магистратура	очная	2 года	русский
TUSUR	Нанотехнологии в электронике и микросистемной технике	28.03.01	бакалавриат	очная	4 года	русский

Рисунок 26 – Результат работы умного поиска по названию программы



Название университета	Название программы	Код программы	Уровень обучения	Форма обучения	Срок обучения	Язык обучения
TIGASU	Инженерная защита окружающей среды	28.03.01	бакалавриат	очная	4 года	русский
TIGASU	Эксплуатация и обслуживание технологических объектов нефтегазового производства	21.03.01	бакалавриат	очная	4 года	русский
TIGASU	Городской кадастр	21.03.02	бакалавриат	очная	4 года	русский
TIGASU	Подъемно-транспортные, строительные, и дорожные машины	23.03.02	бакалавриат	очная	4 года	русский
TIGASU	Архитектура	07.03.01	бакалавриат	очная	5 лет	русский
TIGASU	Реконструкция и реставрация архитектурного наследия	07.03.02	бакалавриат	очная	5 лет	русский
TIGASU	Дизайн архитектурной среды	07.03.03	бакалавриат	очная	5 лет	русский
TIGASU	Промышленное и гражданское строительство	08.03.01	бакалавриат	очная	4 года	русский
TIGASU	Городское строительство	08.03.01	бакалавриат	очная	4 года	русский
TIGASU	Производство строительных материалов, изделий и конструкций	08.03.01	бакалавриат	очная	4 года	русский
TIGASU	Инженерные системы жизнеобеспечения в строительстве	08.03.01	бакалавриат	очная	4 года	русский
TIGASU	Автомобильные дороги	08.03.01	бакалавриат	очная	4 года	русский
TIGASU	Автотранспортные средства и тоннели	08.03.01	бакалавриат	очная	4 года	русский
TGU	Перевод и обучение межкультурной коммуникации	45.03.02	бакалавриат	очная	4 года	русский
TGU	Перевод и обучение межкультурной коммуникации	45.03.01	специалитет	очная	5 лет	русский
TGU	Вычислительная механика и компьютерный моделирование	15.03.03	бакалавриат	очная	4 года	русский
TGU	Промышленная и специальная робототехника	15.03.03	бакалавриат	очная	4 года	русский
TGU	Теплофизика	16.03.01	бакалавриат	очная	4 года	русский
TGU	Баллистика и гидроаэродинамика	24.03.03	бакалавриат	очная	4 года	русский
TGU	Томская международная научная программа	27.03.05	бакалавриат	очная	4 года	русский
TGU	Графика	53.05.03	специалитет	очная	5 лет	русский
TGU	Издательское дело	42.03.03	бакалавриат	очная	4 года	русский
TGU	Отечественная филология	45.03.01	бакалавриат	очная	4 года	русский
TGU	Профессионально-деловая коммуникация на иностранных языках (английский и немецкий)	45.03.01	бакалавриат	очная	4 года	русский
TGU	Русский язык	45.03.01	бакалавриат	очная	4 года	русский
TGU	Фундаментальная и прикладная лингвистика	45.03.03	бакалавриат	очная	4 года	русский
TGU	Информационные технологии	07.06.01	бакалавриат	очная	4 года	русский

Рисунок 27 – Результат работы умного поиска по названию университета

На вкладке «Загруженные университеты» отображается список всех университетов с количеством направлений в каждом. (Рисунок 28).

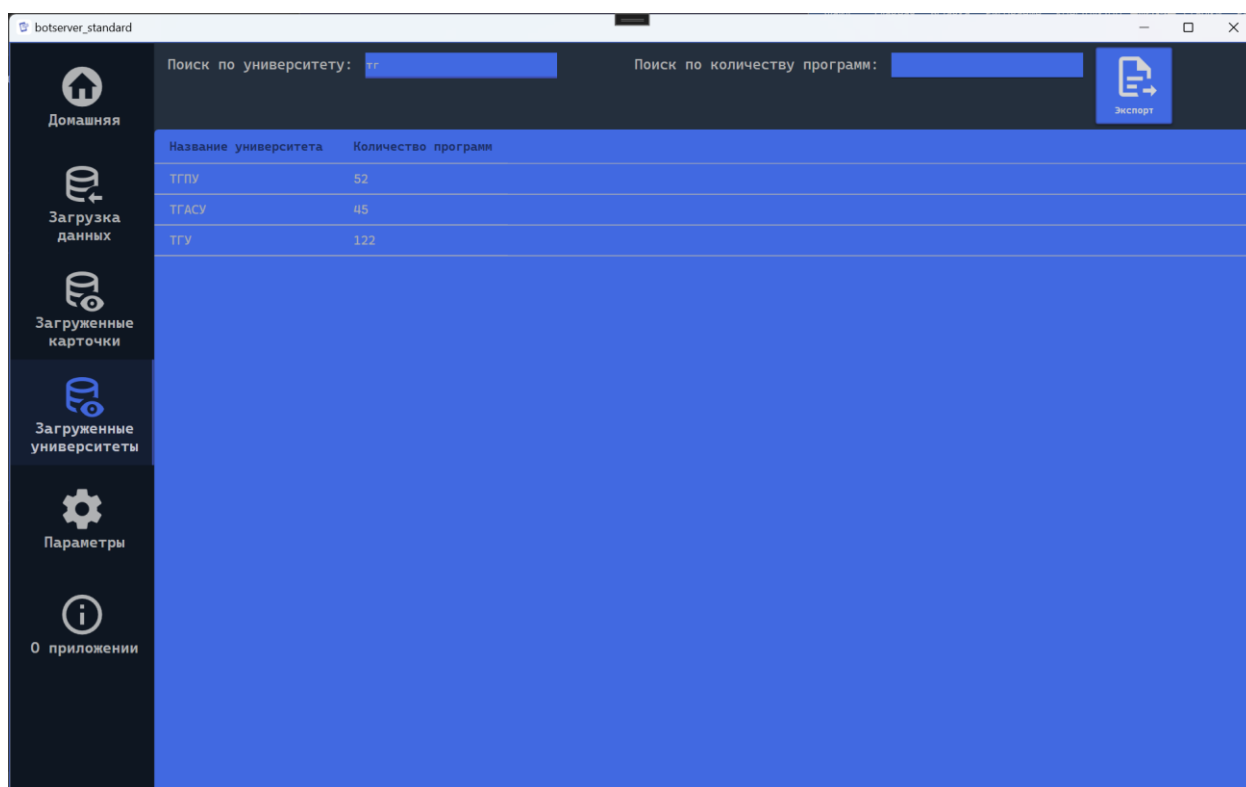
Для поиска по карточкам присутствует два поля – «Поиск по университету» и «Поиск по количеству программ» (Рисунок 29, Рисунок 30).

Умный поиск работает по тому же принципу, что и на вкладке, рассмотренной выше за исключением тех данных, по которым он ищет.



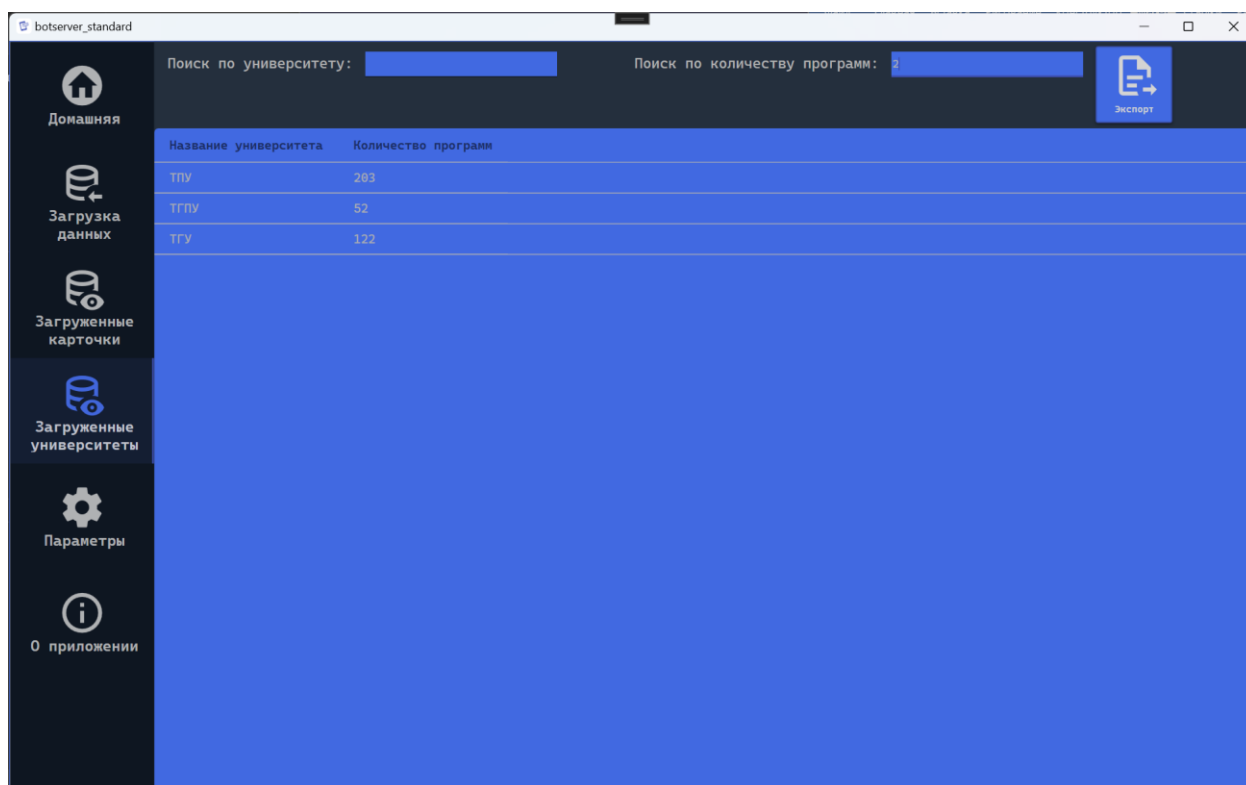
Название университета	Количество программ
ТУСУР	78
ТПУ	203
ТГПУ	52
ТГАСУ	45
ТГУ	122

Рисунок 28 – Общий вид вкладки «Загруженные университеты»



Название университета	Количество программ
ТГПУ	52
ТГАСУ	45
ТГУ	122

Рисунок 29 - Результат работы умного поиска по университету



Название университета	Количество программ
ТПУ	203
ТГПУ	52
ТГУ	122

Рисунок 30 - Результат работы умного поиска по количеству программ

На вкладке «Параметры» осталась последняя нерассмотренная возможность – смена пароля. Для ее использования необходимо ввести в соответствующие поля новый пароль и повторить его, затем нажать кнопку «Установить пароль» (Рисунок 31).

Появится окно, куда необходимо ввести старый пароль для подтверждения установки нового (Рисунок 32). При вводе корректного пароля и подтверждении окно закроется, и на вкладке «Параметры» будет отображено уведомление о смене пароля (Рисунок 33).

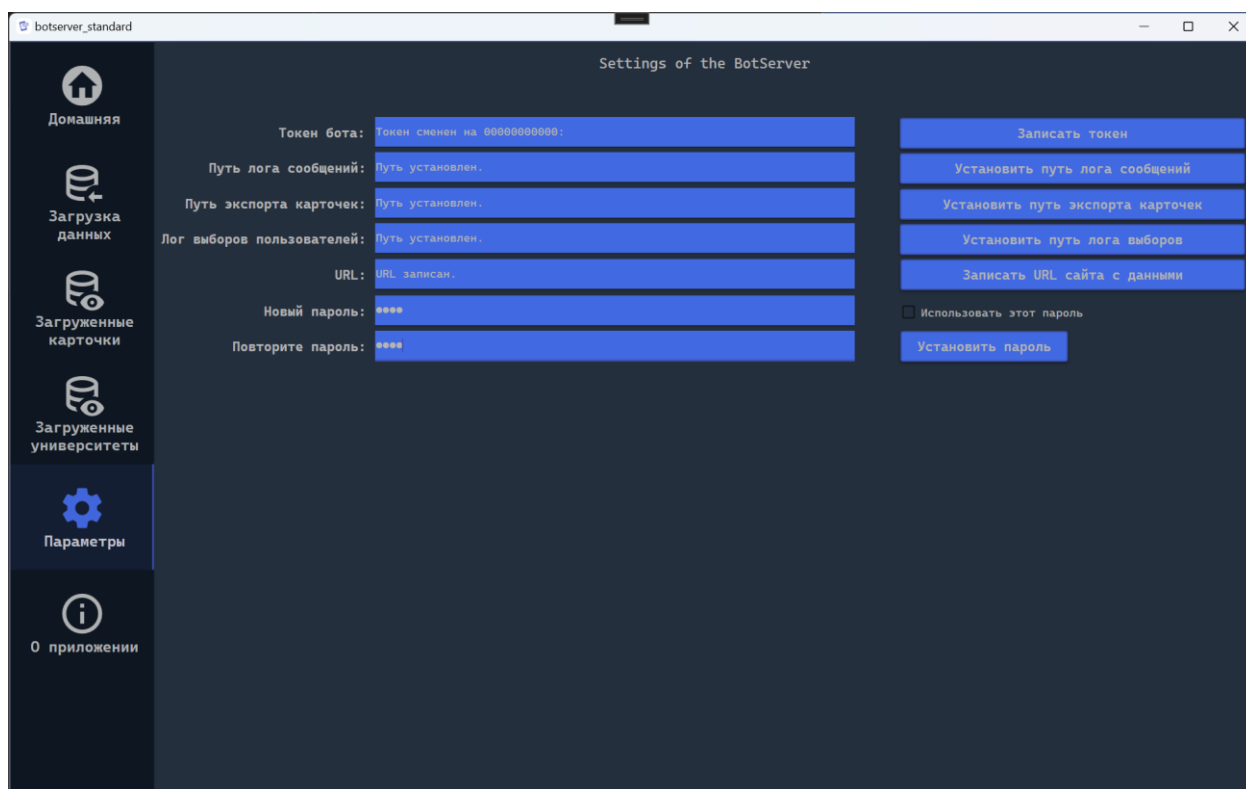


Рисунок 31 – Ввод нового пароля



Рисунок 32 – Окно ввода пароля для подтверждения его смены

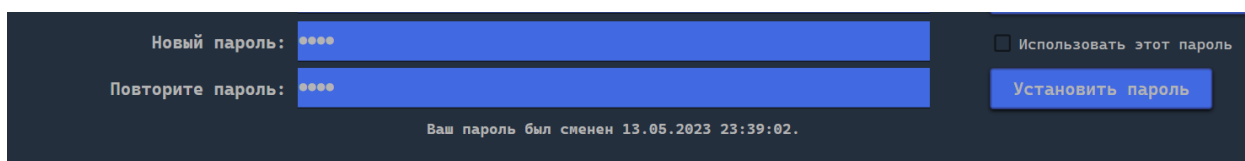


Рисунок 33 – Уведомление об успешной смене пароля

Пользовательский функционал информационной системы представлен Telegram-ботом. Для начала использования бота необходимо перейти по ссылке на бота, токен которого был предварительно указан в модуле настройки и нажать кнопку «Запустить» (Рисунок 34).



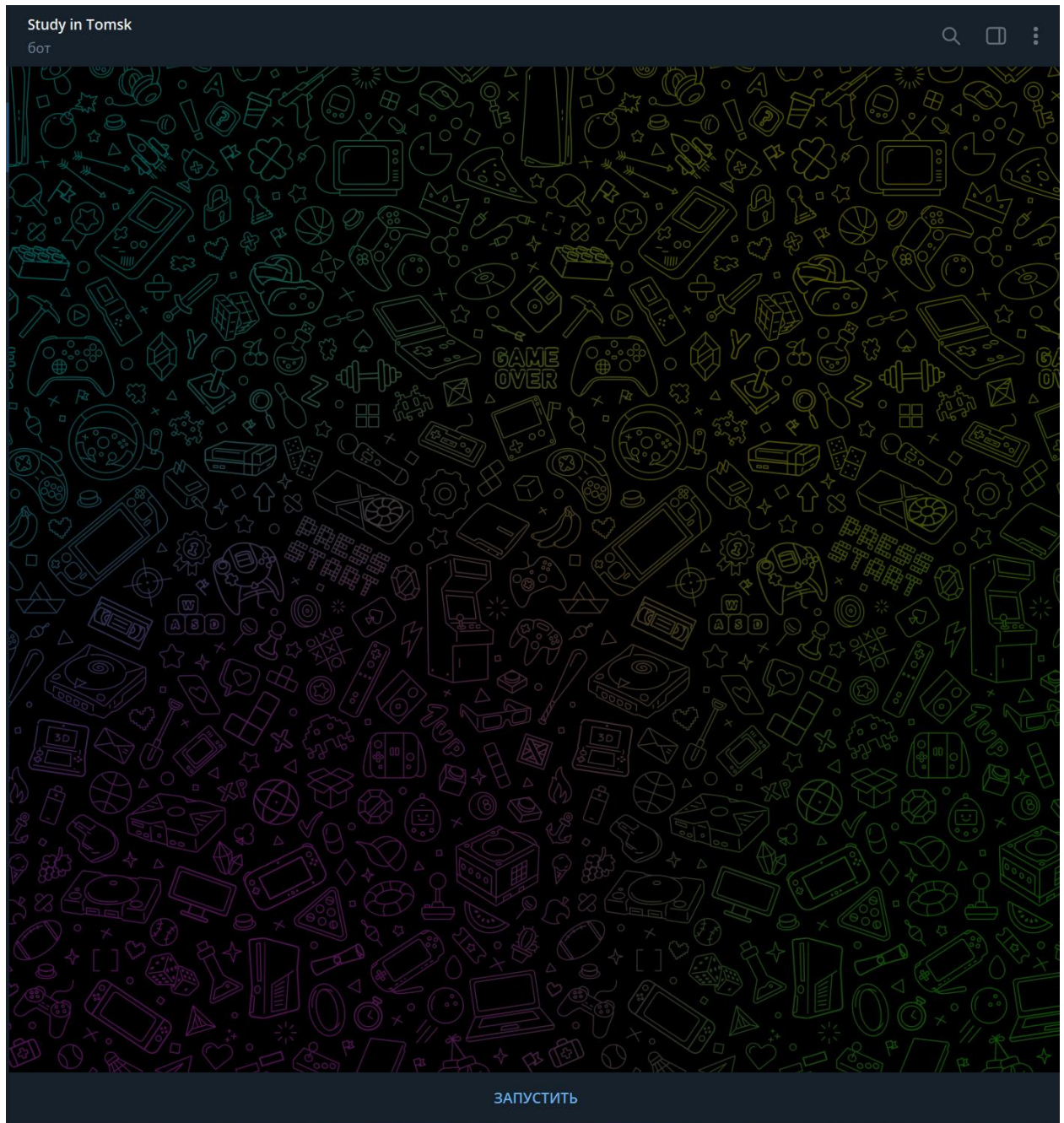


Рисунок 34 – Начальное состояние Telegram-бота

После нажатия кнопки «Запустить» пользователь получит от бота сообщение с предложением выбора действия – главное меню (Рисунок 35).

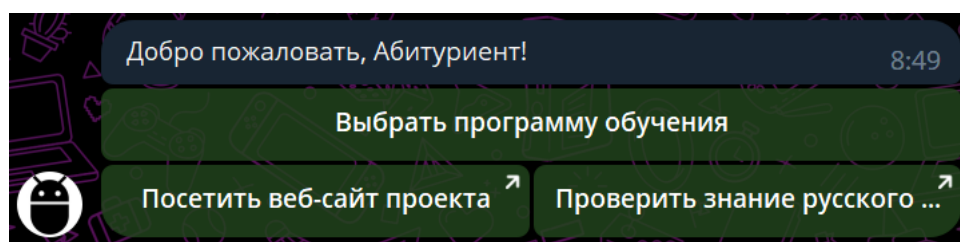


Рисунок 35 – Главное меню бота



Для начала выбора программы обучения пользователю необходимо нажать кнопку «Выбрать программу обучения». Далее начнется последовательный опрос пользователя (Рисунок 36, 37, 38).

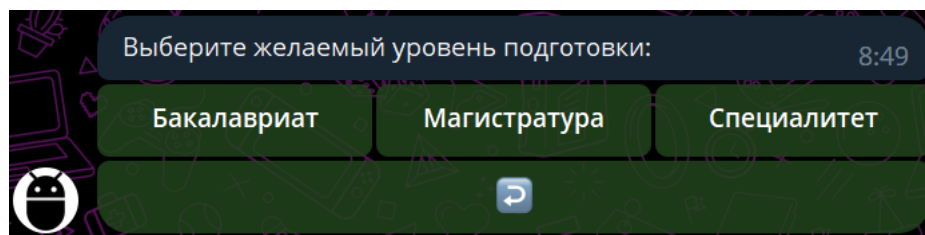


Рисунок 36 – Выбор уровня подготовки

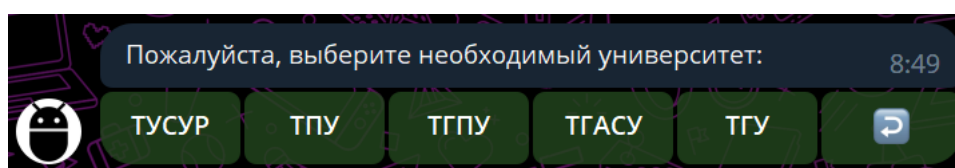


Рисунок 37 – Выбор требуемого университета

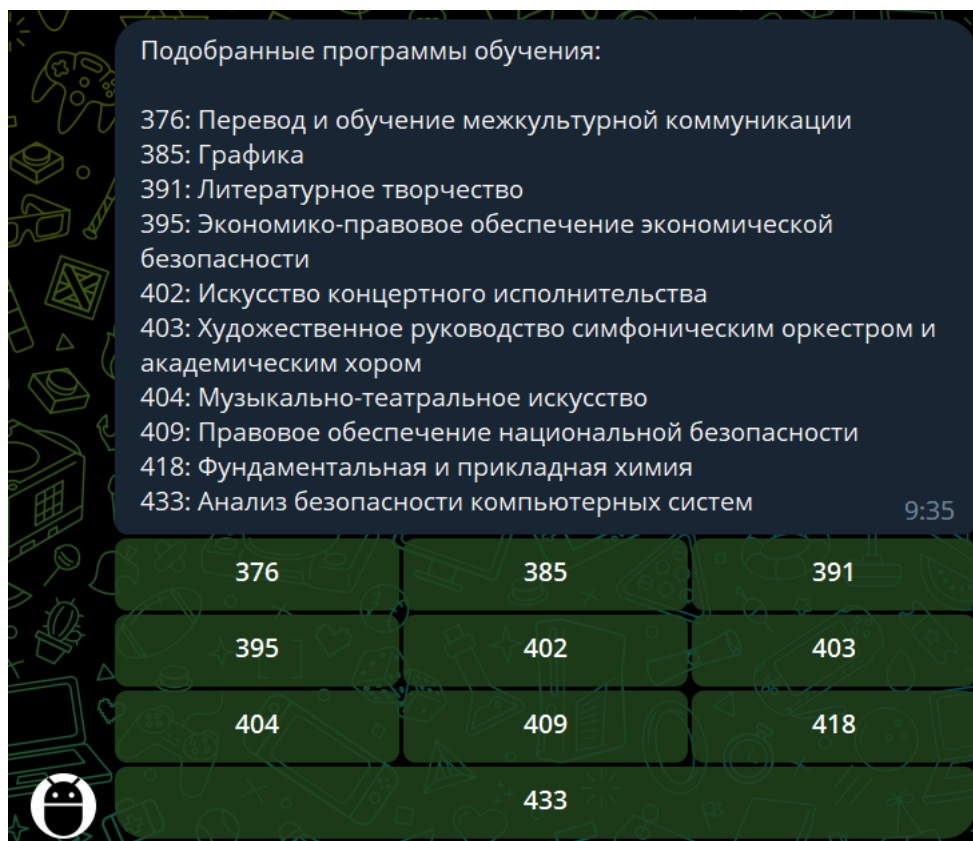


Рисунок 38 – Выбор программы из подобранных

## ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б

По окончании опроса ботом будет предоставлено единственное направление, соответствующее выбору пользователя. Под сообщением расположены кнопки для связи с приемной комиссией для составления заявления и возврата в главное меню (Рисунок 39).

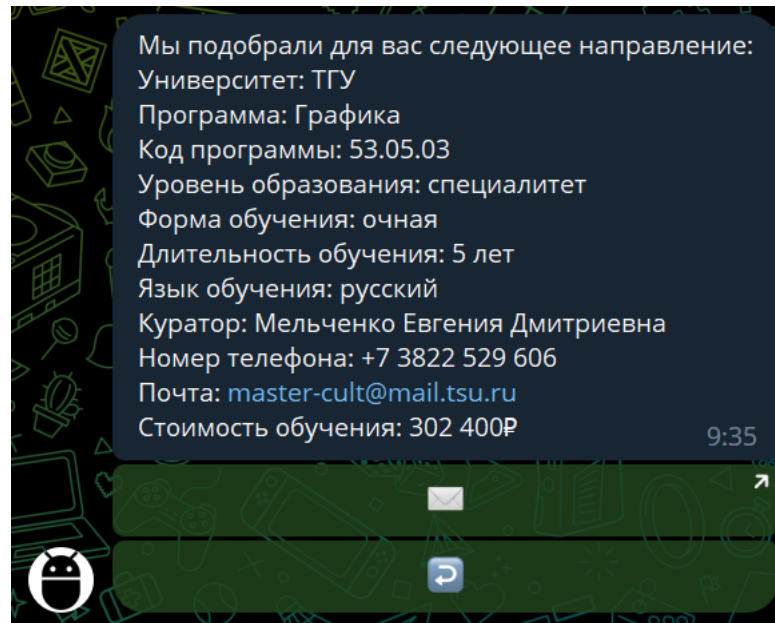


Рисунок 39 – Результат работы бота

Диаграммы состояний

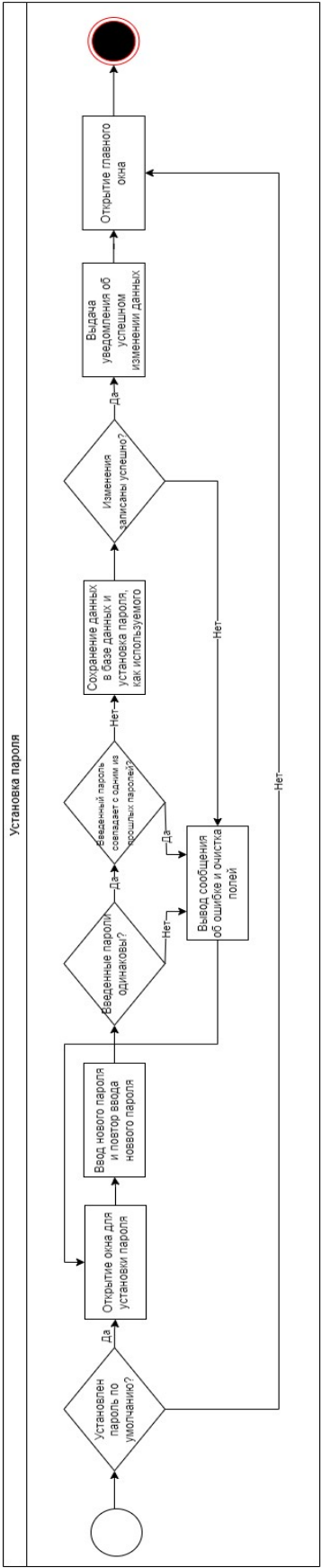


Рисунок 40 – Диаграмма состояния «Установка пароля»

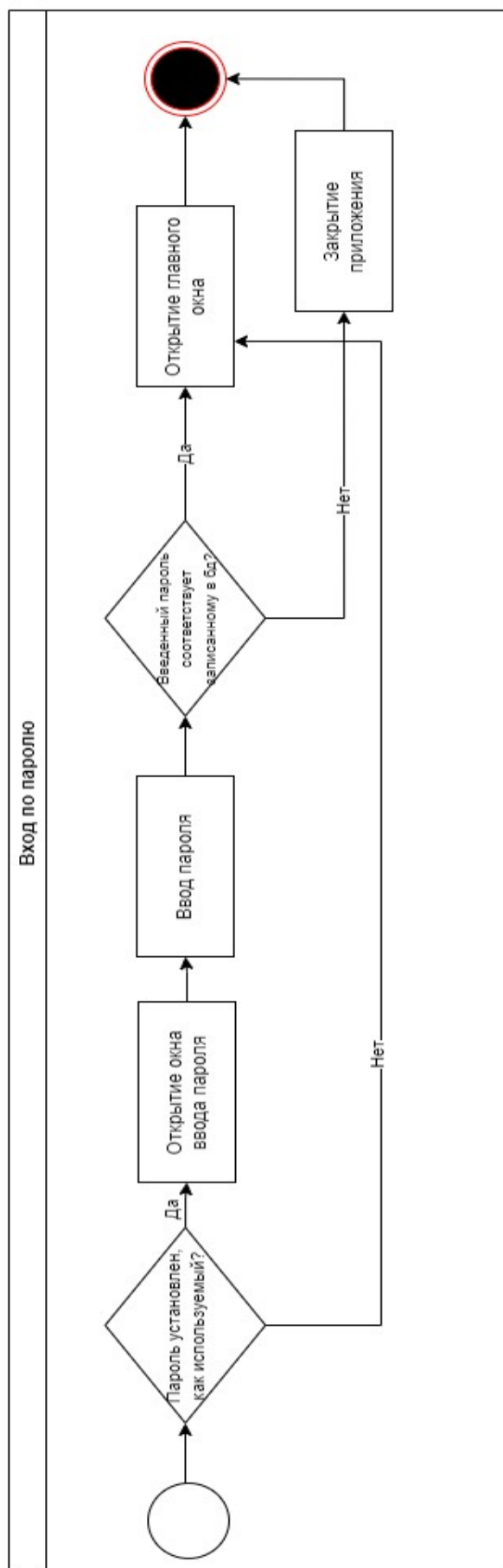


Рисунок 41 – Диаграмма состояния «Вход по паролю»

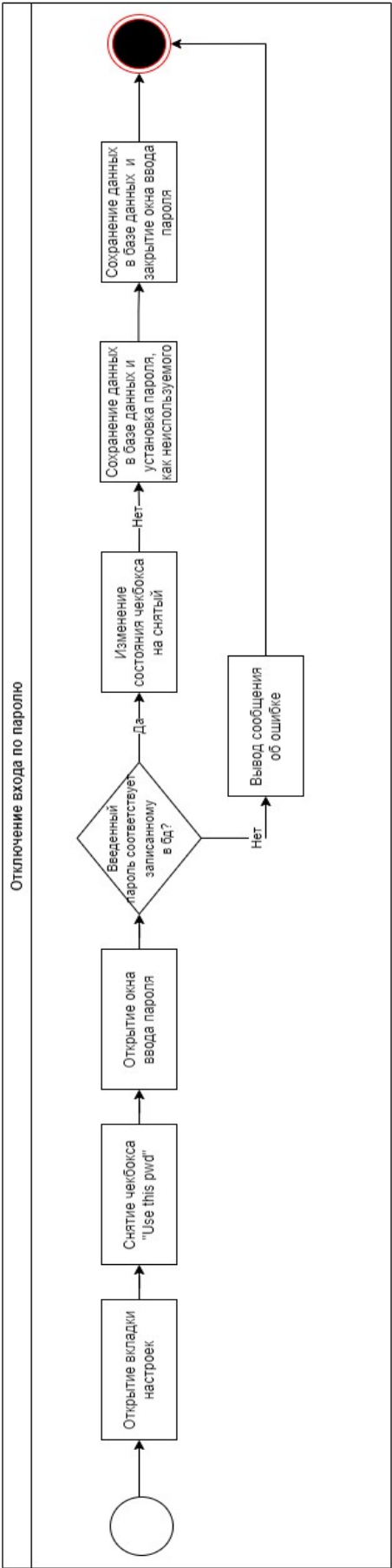


Рисунок 42 – Диаграмма состояния «Отключение входа по паролю»

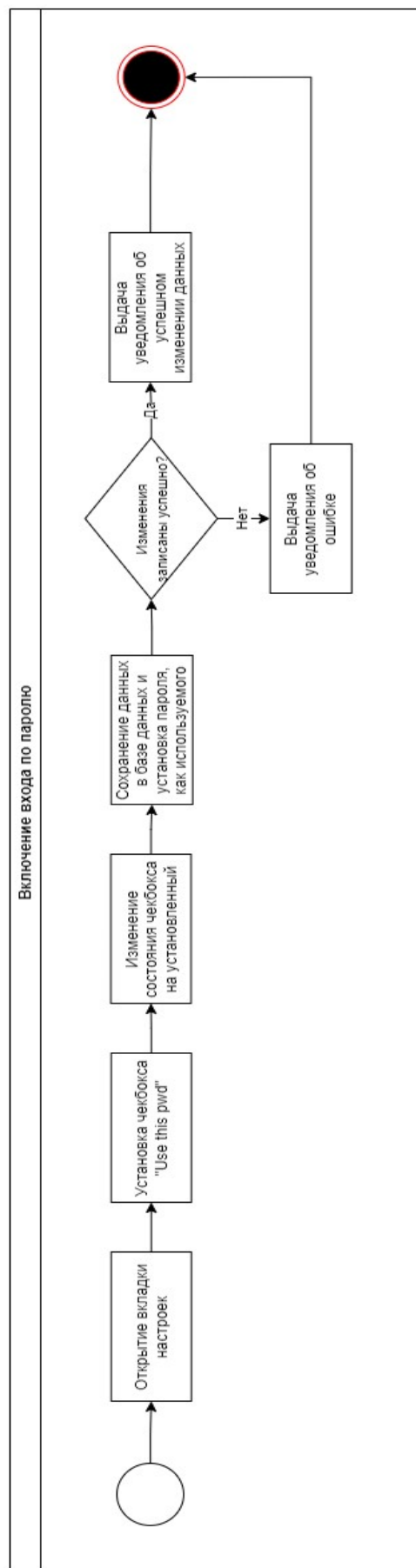


Рисунок 43 – Диаграмма состояния «Включение входа по паролю»

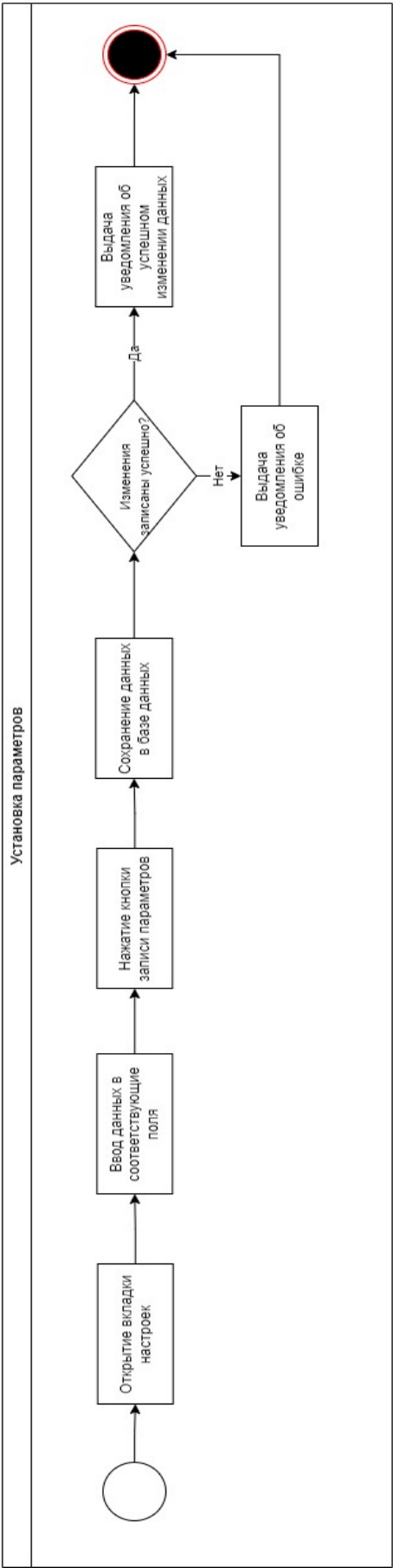


Рисунок 44 – Диаграмма состояния «Установка параметров»

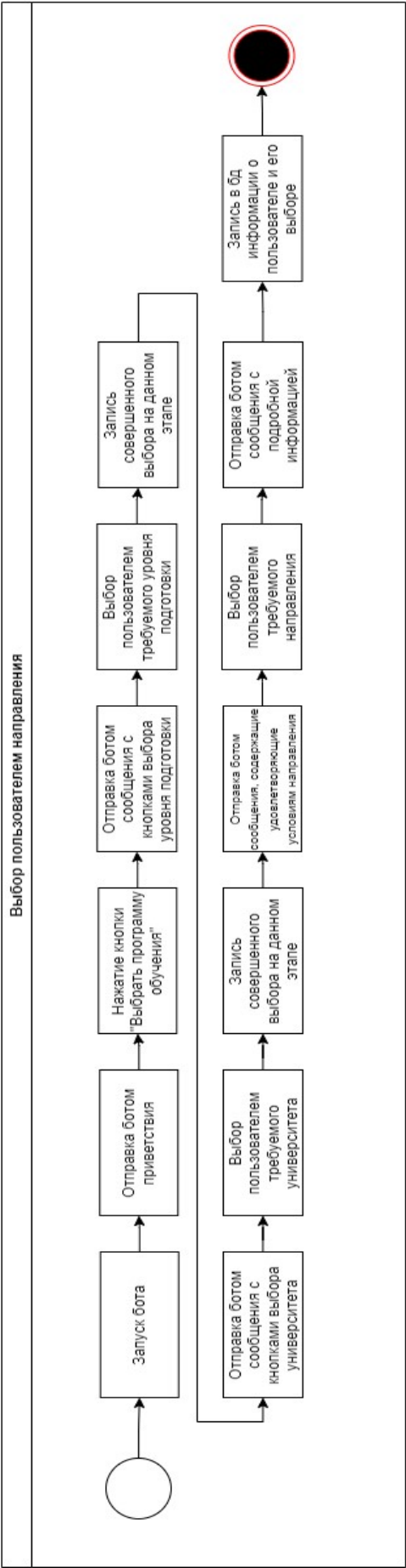


Рисунок 45 – Диаграмма состояния «Выбор пользователем направления обучения»