

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

Специальность 09.02.07 Информационные системы и программирование

Пояснительная записка
к производственной практике
ОПП.23.09.02.07.601.5.ПЗ

Студент

«__» _____ 2023 г.

П. Д. Левицкий

Руководитель

«__» _____ 2023 г.

Д. В. Муха

Томск 2023

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 12
2.2. Диаграмма вариантов использования	12
2.3. Схема базы данных	14
2.4. Словарь данных	15
2.5. Пользовательские сценарии	20
2.6. Прототипы интерфейсов	22
ЗАКЛЮЧЕНИЕ	28
ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	30
ПРИЛОЖЕНИЕ А. Листинг программы	32
ПРИЛОЖЕНИЕ Б. Инструкция пользователя	97
ПРИЛОЖЕНИЕ В. Диаграммы состояний	108

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"".

Сам проект в свою очередь призван упростить поступление абитуриентов в высшие учебные заведения, входящие в состав этого консорциума. На данный момент в состав консорциума входят ТУСУР, ТПУ, ТГПУ, ТГАСУ и ТГУ, также принимает участие СибГМУ.

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.

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 и в соответствии с выбором пользователя (потенциального абитуриента) отправляющего ее в чат. К такому сообщению должна быть прикреплена кнопка для обратной связи.

Задачи производственной практики:

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:

C# – объектно-ориентированный, простой и в то же время мощный язык программирования, позволяющий разработчикам создавать многофункциональные приложения;

C# относится к языкам компилируемого типа, поэтому он обладает всеми преимуществами таких языков;

C# объединяет лучшие идеи современных языков программирования Java, C++, Visual Basic и т.д;

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

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.

СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1. Описание требований к информационной системе

Информационная система предоставляет возможность сбора, хранения и выдачи данных о карточках специальности посредством опроса в телеграм-боте.

Администратор может 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, экспортировать данные по карточкам направлений и программам университетов.

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

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

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.

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
17. Установка URL сайта, с которого парсер будет собирать данные;

18. Установка токена бота, к которому будет подключаться ИС;
19. Установка пути для экспортирующихся отчетов;
20. Установка пути для журналов;
21. Очистка окна живого журнала;

2.2. Диаграмма вариантов использования

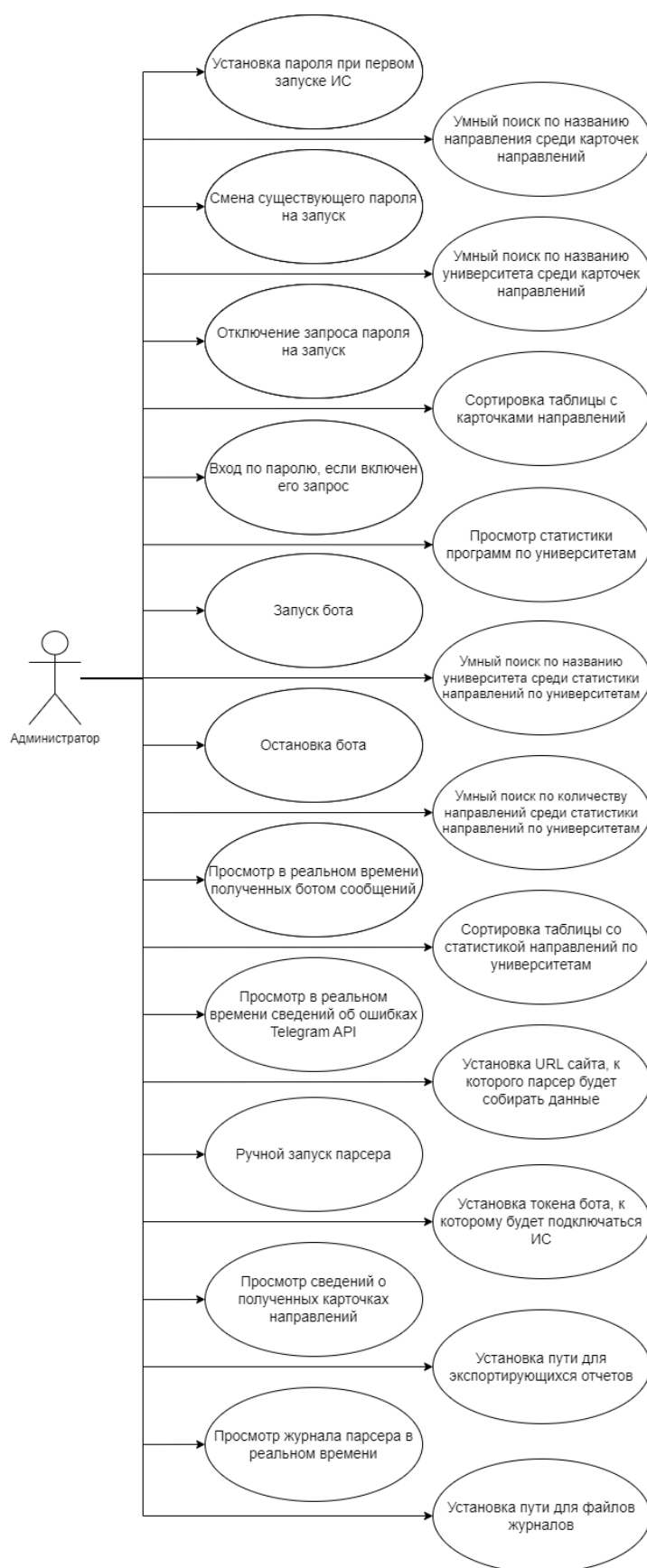


Рисунок 1 – Диаграмма вариантов использования для администратора

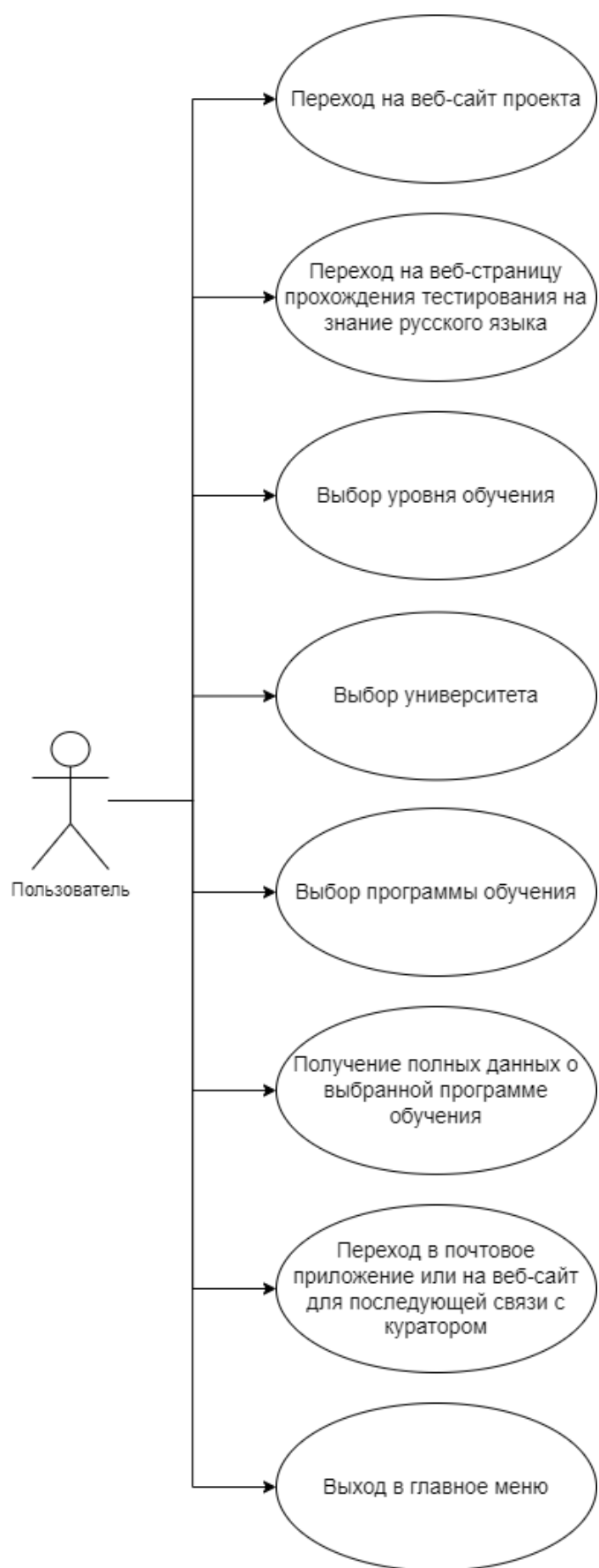


Рисунок 2 – Диаграмма вариантов использования для пользователя

2.3 Схема базы данных

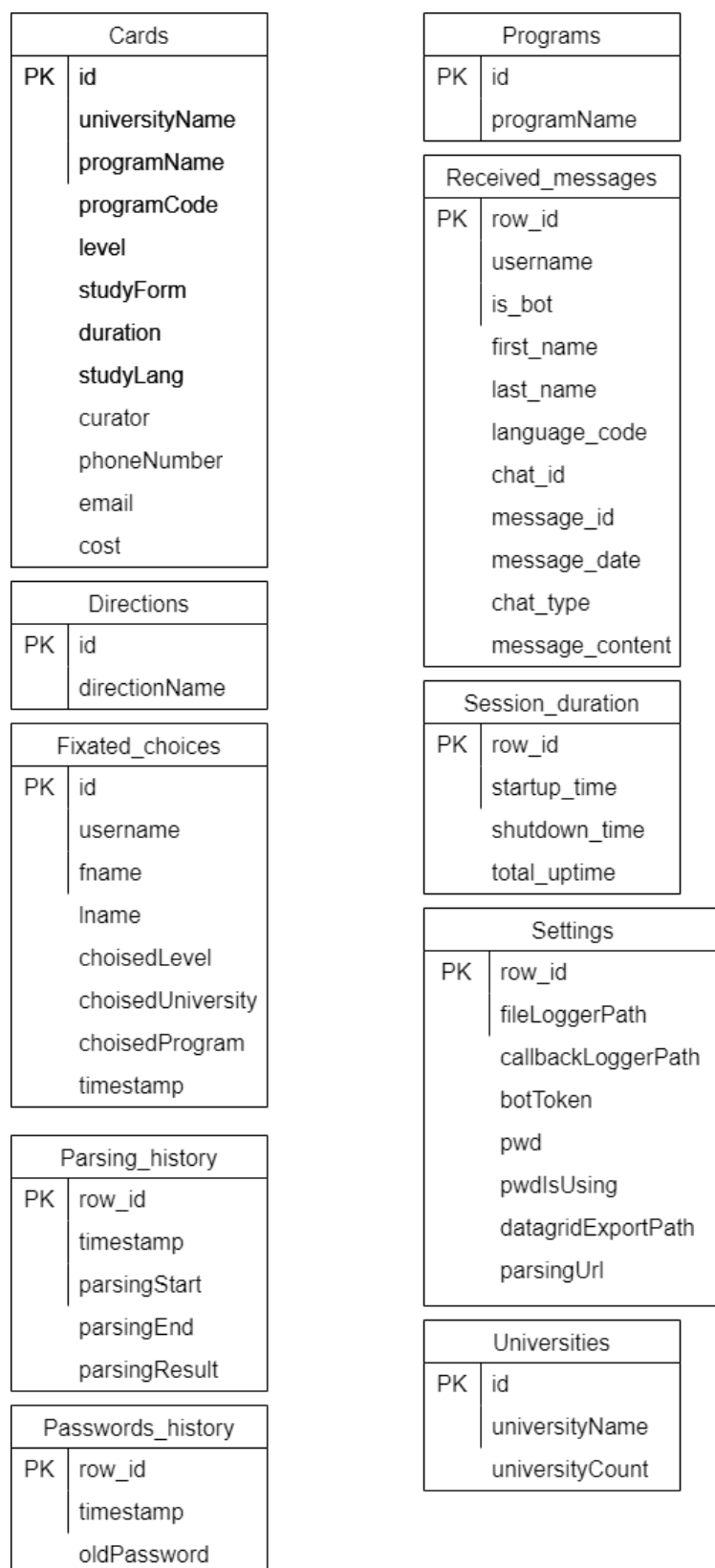


Рисунок 3 – Схема базы данных

2.4 Словарь данных

Таблица 1 – Cards (Карточки направлений)

№ п-п	Название поля	Тип данных	Описание
1	id (PK)	INT	Идентификатор карточки (уникальный)
2	universityName	TEXT	Название университета
3	programName	TEXT	Название программы
4	programCode	TEXT	Код программы
5	level	TEXT	Уровень обучения
6	studyForm	TEXT	Форма обучения
7	duration	TEXT	Длительность
8	studyLang	TEXT	Язык обучения
9	curator	TEXT	Куратор
10	phoneNumber	TEXT	Номер телефона
11	email	TEXT	Почта
12	cost	TEXT	Стоимость обучения

Таблица 2 – Directions (Направления)

№ п-п	Название поля	Тип данных	Описание
1	id (PK)	INT	Идентификатор направления (уникальный)
2	directionName	TEXT	Название направления (уникальное)

Таблица 3 – Fixated_choices (Зафиксированные выборы пользователей)

№ п-п	Название поля	Тип данных	Описание
1	id (PK)	INT	Идентификатор набора выборов (уникальный)
2	username	TEXT	Никнейм пользователя
3	fname	TEXT	Имя пользователя
4	lname	TEXT	Фамилия пользователя
5	choisedLevel	TEXT	Выбранный уровень обучения
6	choisedUniversity	TEXT	Выбранный университет
7	choisedProgram	TEXT	Выбранная программа
8	timestamp	TEXT	Время фиксации результата

Таблица 4 – Parsing_history (История парсинга)

№ п-п	Название поля	Тип данных	Описание
1	row_id (PK)	INT	Идентификатор строки (уникальный)
2	timestamp	TEXT	Время, затраченное на парсинг
3	parsingStart	TEXT	Время начала парсинга
4	parsingEnd	TEXT	Время конца парсинга
5	parsingResult	INT	Результат парсинга (количество полученных карточек)

Таблица 5 – Passwords_history (История установленных паролей)

№ п-п	Название поля	Тип данных	Описание
1	row_id (PK)	INT	Идентификатор строки (уникальный)
2	timestamp	TEXT	Дата смены пароля
3	oldPassword	TEXT	Неактуальный на момент записи пароль

Таблица 6 – Programs (Программы обучения)

№ п-п	Название поля	Тип данных	Описание
1	id (PK)	INT	Идентификатор программы обучения (уникальный)
2	programName	TEXT	Названия программы

Таблица 7 – Received_messages (Принятые ботом сообщения)

№ п-п	Название поля	Тип данных	Описание
1	row_id (PK)	INT	Идентификатор строки (уникальный)
2	username	TEXT	Никнейм пользователя
3	is_bot	TEXT	Пользователь – бот?
4	first_name	TEXT	Имя пользователя
5	last_name	TEXT	Фамилия пользователя
6	language_code	TEXT	Код языка пользователя
7	chat_id	TEXT	Идентификатор чата

Продолжение таблицы 8

8	message_id	TEXT	Идентификатор принятого сообщения
9	message_date	TEXT	Дата получения ботом сообщения
10	chat_type	TEXT	Тип чата с пользователем
11	message_content	TEXT	Содержимое принятого сообщения

Таблица 8 – Session_duration (Длительность сессий)

№ п-п	Название поля	Тип данных	Описание
1	row_id (PK)	INT	Идентификатор строки (уникальный)
2	startup_time	TEXT	Дата и время запуска бота
3	shutdown_time	TEXT	Дата и время остановки бота
4	total_uptime	TEXT	Интервал работы бота

Таблица 9 – Settings (Настройки)

№ п-п	Название поля	Тип данных	Описание
1	row_id (PK)	INT	Идентификатор строки (уникальный)
2	fileLoggerPath	TEXT	Путь к журналу принятых сообщений
3	callbackLoggerPath	TEXT	Путь к журналу выборов пользователей
4	botToken	TEXT	Токен telegram-бота

Продолжение таблицы 9

5	pwd	TEXT	Пароль на вход в GUI
6	pwdIsUsing	TEXT	Вход по паролю активирован?
7	datagridExportPath	TEXT	Путь экспорта данных из таблицы карточек направлений
8	parsingUrl	TEXT	URL, требуемый парсеру

Таблица 10 – Universities (Университеты)

№ п-п	Название поля	Тип данных	Описание
1	Id (PK)	INT	Идентификатор университета (уникальный)
2	universityName	TEXT	Название университета
3	universityCount	INT	Количество направлений

2.5 Пользовательские сценарии

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

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 "Settings" установить флажок "Use this password", 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 «Set password», после этого откроется окно ввода старого пароля. Для смены пароля необходимо указать старый пароль и нажать кнопку «Next», 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 "Parser". Живой журнал даст знать об окончании процедуры парсинга, также есть возможность ручного перезапуска парсера.

Далее перейти на вкладку "Main" и нажать кнопку "Start", 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, запустить экземпляр командной строки или ставить живой журнал на паузу, очищать окно вывода или экспортировать его содержимое.

На вкладке "Parsed cards" выводятся данные по всем полученным парсером карточкам направлений, также расположено два поля умного поиска - по университету и по названию направления.

На вкладке "Parsed universities" 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.

2.6 Прототипы интерфейсов

Окно установки пароля (Рисунок 4)

The diagram shows a rectangular window with a title bar. Inside, the text "Set a new password:" is centered at the top. Below the title are two identical rectangular input fields, each labeled "PasswordBox" in its center. At the bottom of the window, there are two buttons: "Cancel" on the left and "Next" on the right. Both buttons are light gray with black text.

Рисунок 4 – Окно установки пароля

Окно входа (Рисунок 5)

The diagram shows a rectangular window with a title bar. Inside, the text "Your password:" is centered at the top. Below the title is a single rectangular input field labeled "PasswordBox" in its center. At the bottom of the window, there are two buttons: "Cancel" on the left and "Next" on the right. Both buttons are light gray with black text.

Рисунок 5 – Окно входа

Вкладка «Main» окна botserver_standard (Рисунок 6)



Рисунок 6 – Вкладка «Main» окна botserver_standard

Вкладка «Parser» окна botserver_standard (Рисунок 7)

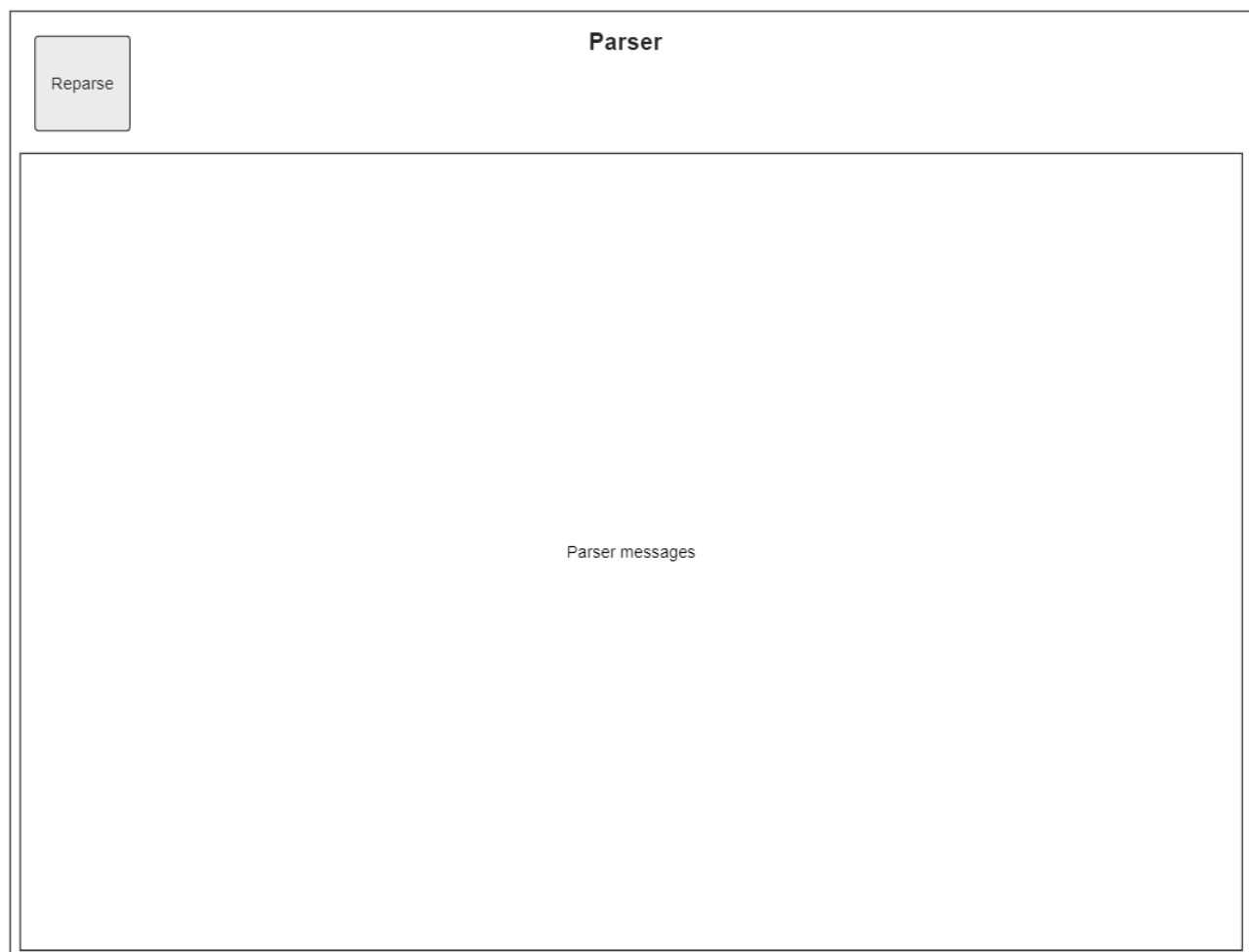


Рисунок 7 – Вкладка «Parser» окна botserver_standard

Вкладка «Parsed cards» окна botserver_standard (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 – Вкладка «Parsed cards» окна botserver_standard

Вкладка «Parsed universities» окна «botserver_standard» (Рисунок 9)



Рисунок 9 – Вкладка «Parsed universities» окна «botserver_standard»

Вкладка «Settings» окна «botserver_standard» (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 choises 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 – Вкладка «Settings» окна «botserver_standard»

Заключение

По итогу прохождения производственной практики была успешно разработана информационная система со следующим функционалом:

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.

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
17. Установка URL сайта, с которого парсер будет собирать данные;
18. Установка токена бота, к которому будет подключаться ИС;
19. Установка пути для экспортирующихся отчетов;
20. Установка пути для журналов;
21. Очистка окна живого журнала;

Перечень используемых источников

Ниже перечислены источники, использовавшиеся при составлении документации для данного проекта:

1. Хабр. Использование диаграммы вариантов использования UML при проектировании программного обеспечения // [Электронный ресурс] – Режим доступа: <https://habr.com/ru/post/566218/>. Дата обращения: 28.02.2023.
2. Worldskills. Проектирование диаграммы состояний UML (statechart diagram) // [Электронный ресурс] – Режим доступа: <https://nationalteam.worldskills.ru/skills/proektirovanie-diagrammy-sostoyaniy-uml-statechart-diagram/>. Дата обращения: 5.03.2023.
3. IT-GOST.RU. Теория и практика UML. Диаграмма состояний // [Электронный ресурс] – Режим доступа: http://it-gost.ru/articles/view_articles/97. Дата обращения: 5.03.2023
4. METANIT.COM. Полное руководство по языку программирования C# 10 и платформе .NET 6 // [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/tutorial/> . Дата обращения: 5.03.2023
5. Microsoft Learn. Общие сведения о WPF // [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8> . Дата обращения: 5.03.2023.

6. METANIT.COM. Введение в WPF // [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/wpf/1.php> . Дата обращения: 5.03.2023.
7. ИНТУИТ. Практикум 9: Пример технического задания для рецензирования // [Электронный ресурс] – Режим доступа: <https://intuit.ru/studies/courses/2195/55/lecture/15050?page=2> . Дата обращения: 1.04.2023.

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";

            //восстановление структуры бд при необходимости
            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

```

<Window x:Class="botserver_standard.AskingPassword"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:botserver_standard"
    mc:Ignorable="d"
    Width="450"
    Height="150"
    Background="#FF4A2E98"
    ResizeMode="NoResize"
    WindowStyle="None"
    WindowStartupLocation="CenterScreen">
<Grid>
    <Label
        Margin="0,10,0,0"
        HorizontalAlignment="Center"
        VerticalAlignment="Top"
        Content="Your password:"
        FontFamily="Cascadia Mono"
        FontSize="15"
        Foreground="#FFB4B4B4" />
    <PasswordBox
        x:Name="EnterPwdBox"
        HorizontalAlignment="Center"
        VerticalContentAlignment="Center"
        Margin="0,42,0,0"
        VerticalAlignment="Top"
        Width="410" Height="22"
        Background="#FF5F3CC0"
        Foreground="#FFB4B4B4"/>
    <Button
        x:Name="CancelBtn"
        Background="#FF5F3CC0"
        Content="Cancel"
        FontFamily="Cascadia Mono"
        Foreground="#FFB4B4B4"
        Margin="20,93,0,0"
        Height="22"
        HorizontalAlignment="Left"
        Width="155"
        VerticalAlignment="Top" IsCancel="True" />
    <Button
        x:Name="NextBtn"
        Background="#FF5F3CC0"
        Click="NextBtn_Click"
        Content="Next"
        FontFamily="Cascadia Mono"

```

```

Foreground="#FFB4B4B4"
Margin="275,93,0,0"
Height="22"
HorizontalAlignment="Left"
Width="155"
VerticalAlignment="Top" />

```

```

</Grid>
</Window>

```

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). Нечитабельно при отладке(?)
```

```
/*
Всего строк: 6001
Строк на одну карточку: 12
Карточек: 500
*/
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:

```
<Window x:Class="botserver_standard.ChangeDefaultPwd"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:local="clr-namespace:botserver_standard"
```

```

mc:Ignorable="d"
Width="450"
Height="150"
Background="#FF4A2E98"
ResizeMode="NoResize"
WindowStyle="None"
WindowStartupLocation="CenterScreen">
<Grid>
  <Label
    Margin="0,10,0,0"
    HorizontalAlignment="Center"
    VerticalAlignment="Top"
    Content="Set a new password:"
    FontFamily="Cascadia Mono"
    FontSize="15"
    Foreground="#FFB4B4B4" />
  <PasswordBox
    x:Name="EnterPwdBox"
    HorizontalAlignment="Center"
    VerticalContentAlignment="Center"
    Margin="0,42,0,0"
    VerticalAlignment="Top"
    Width="410" Height="22"
    Background="#FF5F3CC0"
    Foreground="#FFB4B4B4"/>
  <Button
    x:Name="CancelBtn"
    Background="#FF5F3CC0"
    Content="Cancel"
    FontFamily="Cascadia Mono"
    Foreground="#FFB4B4B4"
    Margin="20,104,0,0"
    Height="22"
    HorizontalAlignment="Left"
    Width="155"
    VerticalAlignment="Top" IsCancel="True" />
  <Button
    x:Name="NextBtn"
    Background="#FF5F3CC0"
    Click="NextBtn_Click"
    Content="Next"
    FontFamily="Cascadia Mono"
    Foreground="#FFB4B4B4"
    Margin="275,104,0,0"
    Height="22"
    HorizontalAlignment="Left"
    Width="155"
    VerticalAlignment="Top" />
  <PasswordBox
    x:Name="EnterPwdBox_Repeated"
    HorizontalAlignment="Center"
    VerticalContentAlignment="Center"

```

```

Margin="0,69,0,0"
VerticalAlignment="Top"
Width="410" Height="22"
Background="#FF5F3CC0"
Foreground="#FFB4B4"/>

```

```

</Grid>
</Window>

```

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 SqliteDataReader 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? choisedLevel;
        static string? choisedUniversity;
        static string? choisedProgram;
    }
}

```

```

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.ToLower() == "/start") //if recieved Message update type

```

```
{

```

```
var 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); // логирование в файл

```

```
DbWorker.DbQuerySilentSender(DbWorker.sqliteConn,
recievedMessageToDbQuery);

```

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

```

```
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, cancellationToken: cancellationToken);

```

```

        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,
        cancellationToken: cancellationToken);

```

```

    }

```

Eight:

```

    ///главное меню (выбор программы) -

```

```

    ///выбор уровня (фиксированный ответ)

```

```

    ///выбор вуза(фиксированный ответ)

```

```

    ///выбор направления(доступные ответы генерятся на основе предыдущих
    записанных callbackData)

```

```

    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!";

```

```

            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);

```

```

        }

```

```

    ///отправка клавиатуры выбора уровня

```

```

        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);

        }

        //отправка клавиатуры выбора университета

        if (update.CallbackQuery.Data.Contains("_level")) //если ответ содержал в себе
level, то изменить сообщение на следующее...

        {

            choisedLevel = update.CallbackQuery.Data.Replace("_level", string.Empty) as
string;

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

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

        }

        //отправка клавиатуры выбора программы

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

        {

            string telegramMessage = "Подобранные программы обучения:";

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

            // фильтрация карточек на основании выборов юзера

```

```

List<Card> filteredCardsByClient = new();

foreach (var item in cardsView)
{
    if (choisedLevel == item.Level && choisedUniversity ==
item.UniversityName)
        filteredCardsByClient.Add(item);
}

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

foreach (var item in cardsView)
{
    if (item.Level == choisedLevel && item.UniversityName ==
choisedUniversity)

filteredUniversitiesButtons.Add(InlineKeyboardButton.WithCallbackData(text:
item.ProgramName, callbackData: Convert.ToString(item.Id)));
}

var dynamicProgramChoosingKeypad = new
InlineKeyboardMarkup(filteredUniversitiesButtons);

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

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

```


ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А

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

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: "Связаться", url:
$"mailto:{finalSelectedCard.Email}"),
    },
    // second row
    new[]
    {
        InlineKeyboardButton.WithCallbackData(text: "На главную", 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;
    }

}

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)
{
    ConsoleWorker.CardOutputter();
}

#region other right stackpanel buttons
private void OutputPauseBtn_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show("Not implemented");
}

```

```

    }

    private void ExportBtn_Click(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Not implemented");
    }

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

    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 {choisedLevel}, университет {choisedUniversity} 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("-----
-----");
    }

    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 {choisedLevel}, университет {choisedUniversity} и программу
{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}', '{choisedLevel}',
'{choisedUniversity}', '{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}",
            cancellation_token: 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?",
        cancellation_token: cancellationToken);
}
}
}

```

Содержимое файла MainWindow.xaml:

```

<Window
    x:Class="botserver_standard.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:botserver_standard"
    Title="botserver_standard"

```

```

Width="1050"
Height="800"
Background="#FF4A2E98"
WindowStartupLocation="CenterScreen"
mc:Ignorable="d">
<Grid>
  <TabControl x:Name="tabControl">
    <TabItem
      x:Name="mainTab"
      Background="#FF4A2E98"
      FontFamily="Cascadia Mono"
      Foreground="#FFB4B4B4"
      Header="Main">

      <Grid Background="#FF4A2E98">
        <TextBox
          x:Name="LiveLogOutput"
          Margin="10,126,10,10"
          Background="Black"
          FontFamily="Cascadia Mono"
          Foreground="White"
          IsReadOnly="True"
          TextWrapping="Wrap" />

        <StackPanel HorizontalAlignment="Right" Width="355">
          <Button
            x:Name="OutputPauseBtn"
            Height="40"
            Background="#FF5F3CC0"
            Click="OutputPauseBtn_Click"
            Content="Pause logging"
            FontFamily="Cascadia Mono"

```



```

Foreground="#FFB4B4B4" />
<Button
  x:Name="ExportBtn"
  Height="40"
  Background="#FF5F3CC0"
  Click="ExportBtn_Click"
  Content="Export log"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4B4" />
<Button
  x:Name="OutputClrBtn"
  Height="40"
  Background="#FF5F3CC0"
  Click="OutputClrBtn_Click"
  Content="Clear output"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4B4" />
</StackPanel>

<Button
  x:Name="CmdOpenBtn"
  Background="#FF5F3CC0"
  Click="CmdOpenBtn_Click"
  Content="Open cmd"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4B4"
  Margin="265,10,0,0"
  Width="80"
  HorizontalAlignment="Left"
  Height="80"
  VerticalAlignment="Top" />
<Button

```

```

x:Name="BotLogoutBtn"
Background="#FF5F3CC0"
Click="StopBotBtn_Click"
Content="STOP"
FontFamily="Cascadia Mono"
Foreground="#FFB4B4B4"
Margin="95,10,0,0"
HorizontalAlignment="Left"
Width="80"
Height="80"
VerticalAlignment="Top" />
<Button
x:Name="BotStartBtn"
Background="#FF5F3CC0"
Click="BotStartBtn_Click"
Content="START"
FontFamily="Cascadia Mono"
Foreground="#FFB4B4B4"
Margin="10,10,0,0"
HorizontalAlignment="Left"
Width="80"
IsDefault="True"
Height="80"
VerticalAlignment="Top" />
<Button
x:Name="StopExitBotBtn"
Background="#FF5F3CC0"
Click="StopExitBotBtn_Click"
Content="Stop & exit"
FontFamily="Cascadia Mono"
Foreground="#FFB4B4B4"
Margin="180,10,0,0"

```

```

HorizontalAlignment="Left"
Width="80"
IsCancel="True"
Height="80"
VerticalAlignment="Top" />
</Grid>
</TabItem>

<TabItem
    x:Name="parserTab"
    Background="#FF4A2E98"
    FontFamily="Cascadia Mono"
    Foreground="#FFB4B4B4"
    Header="Parser">
<Grid Background="#FF4A2E98">
    <TextBox
        x:Name="ParserLogOutput"
        Margin="10,127,10,9"
        Background="Black"
        FontFamily="Cascadia Mono"
        Foreground="White"
        IsReadOnly="True"
        TextWrapping="Wrap" />
    <Button
        x:Name="ReparseBtn"
        Background="#FF5F3CC0"
        Click="ParserPeparseBtn_Click"
        Content="Reparse"
        FontFamily="Cascadia Mono"
        Foreground="#FFB4B4B4" Margin="10,42,0,0"
        HorizontalAlignment="Left"
        Width="80"

```

```

        IsDefault="True"
        Height="80"
        VerticalAlignment="Top" />
<Label
    Margin="0,10,0,0"
    HorizontalAlignment="Center"
    VerticalAlignment="Top"
    Content="Parser"
    FontFamily="Cascadia Mono"
    FontSize="15"
    Foreground="#FFB4B4B4" Width="62" />

</Grid>
</TabItem>
<TabItem
    x:Name="parsedCardsTab"
    Background="#FF4A2E98"
    FontFamily="Cascadia Mono"
    Foreground="#FFB4B4B4"
    Header="Parsed cards">
    <Grid Background="#FF4A2E98">

        <DataGrid x:Name="parsedCardsGrid" AutoGenerateColumns="False"
            CanUserAddRows="False" CanUserDeleteRows="False" CanUserResizeRows="False"
            IsReadOnly="True" RenderTransformOrigin="0.504,1.97" Foreground="Black"
            BorderBrush="#FFABADB3" Background="#FF5F3CC0" Margin="0,42,0,0">

            <DataGrid.Columns>

                <!--<DataGridTextColumn Header="Id:" Binding="{Binding
cardsView.Id}" />-->

                <DataGridTextColumn Header="Название университета" Binding="{Binding
UniversityName}" />

                <DataGridTextColumn Header="Название программы" Binding="{Binding
ProgramName}" />

```

```

        <DataGridTextColumn Header="Код программы" Binding="{Binding
ProgramCode}"/>
        <DataGridTextColumn Header="Уровень обучения" Binding="{Binding
Level}"/>
        <DataGridTextColumn Header="Форма обучения" Binding="{Binding
StudyForm}"/>
        <DataGridTextColumn Header="Срок обучения" Binding="{Binding
Duration}"/>
        <DataGridTextColumn Header="Язык обучения" Binding="{Binding
StudyLang}"/>
        <DataGridTextColumn Header="Куратор" Binding="{Binding Curator}"/>
        <DataGridTextColumn Header="Телефон" Binding="{Binding
PhoneNumber}"/>
        <DataGridTextColumn Header="Почта" Binding="{Binding Email}"/>
        <DataGridTextColumn Header="Стоимость" Binding="{Binding Cost}"/>

```

```

    </DataGrid.Columns>

```

```

</DataGrid>

```

```

<TextBox

```

```

    x:Name = "SearchByProgramName"

```

```

    HorizontalAlignment = "Left"

```

```

    Margin="264,10,0,0"

```

```

    TextWrapping="Wrap"

```

```

    VerticalContentAlignment="Center"

```

```

    VerticalAlignment="Top"

```

```

    Width="199"

```

```

    Height="27"

```

```

    KeyUp="SearchByProgramName_KeyUp"

```

```

    Background="#FF5F3CC0"

```

```

    Foreground="#FFABADB3"/>

```

```

<TextBox

```

```

    x:Name = "SearchByUniversity"

```

```

HorizontalAlignment="Left"
Margin="709,10,0,0"
TextWrapping="Wrap"
VerticalContentAlignment="Center"
VerticalAlignment="Top"
Width="199" Height="27"
KeyUp="SearchByUniversity_KeyUp"
Background="#FF5F3CC0"
Foreground="#FFABADB3"/>
<Label
    Margin="10,10,0,0"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    Content="Поиск по названию программы:"
    FontFamily="Cascadia Mono"
    FontSize="15"
    Foreground="#FFB4B4B4" Width="254" />
<Label
    Margin="497,10,0,0"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    Content="Поиск по университету:"
    FontFamily="Cascadia Mono"
    FontSize="15"
    Foreground="#FFB4B4B4" Width="207" />
    <Button x:Name="CardsExportBtn" Content="Export" HorizontalAlignment="Left"
Margin="913,10,0,0" VerticalAlignment="Top" Height="27" Width="71"
Background="#FF5F3CC0" Foreground="#FFB4B4B4" Click="CardsExportBtn_Click"/>

</Grid>
</TabItem>

```

```

<TabItem
    x:Name="parsedUniversitiesTab"
    Background="#FF4A2E98"
    FontFamily="Cascadia Mono"
    Foreground="#FFB4B4B4"
    Header="Parsed universities">
    <Grid Background="#FF4A2E98">

        <!--<DataGrid
            x:Name = "parsedUniversitiesGrid"
            Margin="0,42,0,0"
            IsReadOnly="True"
            Foreground="Black"
            BorderBrush="#FFABADB3"
            Background="#FF5F3CC0"/>-->

            <DataGrid x:Name="parsedUniversitiesGrid" AutoGenerateColumns="False"
                CanUserAddRows="False" CanUserDeleteRows="False" CanUserResizeRows="False"
                IsReadOnly="True" RenderTransformOrigin="0.504,1.97" Foreground="Black"
                BorderBrush="#FFABADB3" Background="#FF5F3CC0" Margin="0,42,0,0">

                <DataGrid.Columns>

                    <!--<DataGridTextColumn Header="Id:" Binding="{Binding
cardsView.Id}"/>-->

                    <DataGridTextColumn Header="Название университета" Binding="{Binding
UniversityName}"/>

                    <DataGridTextColumn Header="Количество программ" Binding="{Binding
Count}"/>

                </DataGrid.Columns>

            </DataGrid>

        <TextBox
            x:Name = "SearchByUniversityName"
            HorizontalAlignment = "Left"
            Margin="220,10,0,0"

```

```

TextWrapping="Wrap"
VerticalContentAlignment="Center"
VerticalAlignment="Top"
Width="199"
Height="27"
KeyUp="SearchByUniversityName_KeyUp"
Background="#FF5F3CC0"
Foreground="#FFABADB3"/>
<TextBox
  x:Name = "SearchByUniversityFreq"
  HorizontalAlignment = "Left"
  Margin="769,10,0,0"
  TextWrapping="Wrap"
  VerticalContentAlignment="Center"
  VerticalAlignment="Top"
  Width="112" Height="27"
  KeyUp="SearchByUniversityFreq_KeyUp"
  Background="#FF5F3CC0"
  Foreground="#FFABADB3"/>
<Label
  Margin="10,10,0,0"
  HorizontalAlignment="Left"
  VerticalAlignment="Top"
  Content="Поиск по университету:"
  FontFamily="Cascadia Mono"
  FontSize="15"
  Foreground="#FFB4B4B4" Width="254" />
<Label
  Margin="497,10,0,0"
  HorizontalAlignment="Left"
  VerticalAlignment="Top"
  Content="Поиск по количеству программ:"

```



```

FontFamily="Cascadia Mono"
FontSize="15"
Foreground="#FFB4B4B4" Width="267" />

```

```

<Button
  x:Name="UniversitiesExportBtn"
  Content="Export"
  HorizontalAlignment="Left"
  Margin="913,10,0,0"
  VerticalAlignment="Top"
  Height="27"
  Width="71"
  Background="#FF5F3CC0"
  Foreground="#FFB4B4B4"
  Click="UniversitiesExportBtn_Click"/>

```

```

</Grid>

```

```

</TabItem>

```

```

<TabItem
  x:Name="settingsTab"
  Background="#FF4A2E98"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4B4"
  Header="Settings">
  <Grid Background="#FF4A2E98">
    <TextBox
      x:Name="SettingsTokenInput"
      Width="495"
      Height="22"
      Margin="230,71,0,0"
      HorizontalAlignment="Left"
      VerticalAlignment="Top"

```

```

VerticalContentAlignment="Center"
Background="#FF5F3CC0"
FontFamily="Cascadia Mono"
Foreground="#FFB4B4B4"
TextWrapping="NoWrap" />
<Label
    Width="169"
    Height="22"
    Margin="56,71,0,0"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    HorizontalContentAlignment="Right"
    Content="Bot token:"
    FontFamily="Cascadia Mono"
    FontWeight="Bold"
    Foreground="#FFB4B4B4" />
<Label
    Width="169"
    Height="25"
    Margin="56,101,0,0"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    HorizontalContentAlignment="Right"
    Content="Messages log path:"
    FontFamily="Cascadia Mono"
    FontWeight="Bold"
    Foreground="#FFB4B4B4" />
<Label
    Width="169"
    Height="26"
    Margin="56,130,0,0"
    HorizontalAlignment="Left"

```

```

VerticalAlignment="Top"
HorizontalContentAlignment="Right"
Content="Cards export path:"
FontFamily="Cascadia Mono"
FontWeight="Bold"
Foreground="#FFB4B4" />
<TextBox
  x:Name="MessagesLogRootInput"
  Width="495"
  Height="22"
  Margin="230,101,0,0"
  HorizontalAlignment="Left"
  VerticalAlignment="Top"
  VerticalContentAlignment="Center"
  Background="#FF5F3CC0"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4"
  TextWrapping="NoWrap" />
<TextBox
  x:Name="datagridExportPath"
  Width="495"
  Height="22"
  Margin="230,131,0,0"
  HorizontalAlignment="Left"
  VerticalAlignment="Top"
  VerticalContentAlignment="Center"
  Background="#FF5F3CC0"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4"
  TextWrapping="NoWrap" />
<Label
  Margin="0,10,0,0"

```

```

HorizontalAlignment="Center"
VerticalAlignment="Top"
Content="Settings of the BotServer"
FontFamily="Cascadia Mono"
FontSize="15"
Foreground="#FFB4B4" />
<Button
  x:Name="SettingsFilePathSetBtn"
  Background="#FF5F3CC0"
  Click="ChoicesLogRootInputSetBtn_Click"
  Content="Set choices log path"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4"
  Margin="0,160,66,0"
  Height="22"
  VerticalAlignment="Top" HorizontalAlignment="Right" Width="155" />
<Button
  x:Name="SetTokenBtn"
  Background="#FF5F3CC0"
  Click="SetTokenBtn_Click"
  Content="Set bot token"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4"
  Margin="0,71,66,0"
  Height="22"
  VerticalAlignment="Top" HorizontalAlignment="Right" Width="155" />
<Button
  x:Name="SetLogPathBtn"
  Background="#FF5F3CC0"
  Click="SetMessageLogPathBtn_Click"
  Content="Set message log path"
  FontFamily="Cascadia Mono"

```

```

Foreground="#FFB4B4B4"
Margin="0,101,66,0"
Height="22"
VerticalAlignment="Top" HorizontalAlignment="Right" Width="155" />
<Button
  x:Name="SetDbPathBtn"
  Background="#FF5F3CC0"
  Click="SetDatagridExportPathBtn_Click"
  Content="Set export path"
  FontFamily="Cascadia Mono"
  Foreground="#FFB4B4B4"
  Margin="0,131,66,0"
  Height="22"
  VerticalAlignment="Top" HorizontalAlignment="Right" Width="155" />
<Label
  Width="169"
  Height="24"
  Margin="56,161,0,0"
  HorizontalAlignment="Left"
  VerticalAlignment="Top"
  HorizontalContentAlignment="Right"
  Content="Choices log path:"
  FontFamily="Cascadia Mono"
  FontWeight="Bold"
  Foreground="#FFB4B4B4" />
<TextBox
  x:Name="ChoicesLogRootInput"
  Width="495"
  Height="22"
  Margin="230,161,0,0"
  HorizontalAlignment="Left"
  VerticalAlignment="Top"

```

```

VerticalContentAlignment="Center"
Background="#FF5F3CC0"
FontFamily="Cascadia Mono"
Foreground="#FFB4B4B4"
TextWrapping="NoWrap" />
<Button
    x:Name="ParsingUrlSetBtn"
    Background="#FF5F3CC0"
    Content="Set parsing URL"
    FontFamily="Cascadia Mono"
    Foreground="#FFB4B4B4"
    Margin="0,189,66,0"
    Height="22"
    VerticalAlignment="Top"
    Click="ParsingUrlSetBtn_Click" HorizontalAlignment="Right" Width="155" />
<Label
    Width="175"
    Height="24"
    Margin="50,190,0,0"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    HorizontalContentAlignment="Right"
    Content="Parsing URL:"
    FontFamily="Cascadia Mono"
    FontWeight="Bold"
    Foreground="#FFB4B4B4" FontSize="11" />
<TextBox
    x:Name="UrlSet"
    Width="495"
    Height="22"
    Margin="230,191,0,0"
    HorizontalAlignment="Left"

```

```

VerticalAlignment="Top"
VerticalContentAlignment="Center"
Background="#FF5F3CC0"
FontFamily="Cascadia Mono"
Foreground="#FFB4B4B4"
TextWrapping="NoWrap" />
<Label
    Width="169"
    Height="24"
    Margin="56,260,0,0"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    HorizontalContentAlignment="Right"
    Content="Enter password:"
    FontFamily="Cascadia Mono"
    FontWeight="Bold"
    Foreground="#FFB4B4B4" />
<Label
    Width="169"
    Height="24"
    Margin="56,291,0,0"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    HorizontalContentAlignment="Right"
    Content="Repeat password:"
    FontFamily="Cascadia Mono"
    FontWeight="Bold"
    Foreground="#FFB4B4B4" />
<Button
    x:Name="SetPwdBtn"
    Background="#FF5F3CC0"
    Click="SetPwdBtn_Click"

```

```

Content="Set password"
FontFamily="Cascadia Mono"
Foreground="#FFB4B4B4"
Margin="0,292,66,0"
Height="22"
VerticalAlignment="Top" HorizontalAlignment="Right" Width="155" />
<PasswordBox
  x:Name="SetPwdBox"
  HorizontalAlignment="Left"
  VerticalContentAlignment="Center"
  Margin="230,261,0,0"
  VerticalAlignment="Top"
  Width="495" Height="22"
  Background="#FF5F3CC0"
  Foreground="#FFB4B4B4"/>
<PasswordBox
  x:Name="SetRepeatedPwdBox"
  HorizontalAlignment="Left"
  VerticalContentAlignment="Center"
  Margin="230,292,0,0"
  VerticalAlignment="Top"
  Width="495" Height="22"
  Background="#FF5F3CC0"
  Foreground="#FFB4B4B4"/>
<CheckBox
  x:Name="UseThisPwdCheckbox"
  Content="Use this pwd"
  Margin="0,261,66,0"
  VerticalAlignment="Top"
  Background="#FF5F3CC0"
  Foreground="#FFB4B4B4"
  Height="22"

```



```

        Checked="UseThisPwdCheckbox_Checked"
        Unchecked="UseThisPwdCheckbox_Unchecked" HorizontalAlignment="Right"
Width="154"/>
    <Label x:Name="PwdChangeNotifier"
        Width="411"
        Height="24"
        Margin="230,319,0,0"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        HorizontalContentAlignment="Center"
        Content=""
        FontFamily="Cascadia Mono"
        FontWeight="Bold"
        Foreground="#FFB4B4B4" />
</Grid>
</TabItem>
<TabItem
    x:Name="aboutTab"
    Background="#FF4A2E98"
    FontFamily="Cascadia Mono"
    Foreground="#FFB4B4B4"
    Header="About">
    <Grid Background="#FF4A2E98" />
</TabItem>
</TabControl>

</Grid>
</Window>

```

Содержимое файла MainWindow.xaml.cs:

```
using System;
```

```

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

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 в
окне
                }
                else
                {
                    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
        }
    }
}

```

```

    }
}

```

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();
        public async void CardParser(SqliteConnection sqliteConn)
        {

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

            Dispatcher.Invoke(() =>
            {
                ParserLogOutput.Text += $"{DateTime.Now} | Getting data from web...\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} | Selecting nodes...\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} | Processing received data...\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 подряд
cardsList = noTabsDoc.Split('\n').ToList(); //построчная запись данных (в том числе и
мусора)

//удаление мусора из листа
Dispatcher.Invoke(() =>
{
    ParserLogOutput.Text += $"{DateTime.Now} | Cleaning data...\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

/*
Всего строк: 6501
Строк на одну карточку: 12
Карточек: 500
*/

//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} | Writing data...\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

```

```

};

SqliteDataReader 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;
}

//ТУСУР - 78 раз
//ТПУ - 203 раз
//ТГПУ - 52 раз
//ТГАСУ - 45 раз
//ТГУ - 122 раз

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
    };
    SqliteDataReader freqReader = freqCommand.ExecuteReader();

    List < UniversityEntryFreq > universityFreqListView= new();
    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} | Parsing is done.\n{Card.cards.Count}
cards have been added;\n{universityFreqListView.Count} universities have been added.\n";

    ParserLogOutput.Text += "-----\n";

});
}

}
}

```

Содержимое файла ParserTab.cs:

```

using System;
using System.Threading.Tasks;
using System.Windows;

namespace botserver_standard
{
    public partial class MainWindow : Window
    {

```

```

private void ParserPeparseBtn_Click(object sender, RoutedEventArgs e)
{
    Card.cards.Clear();
    Dispatcher.Invoke(() =>
    {
        ParserLogOutput.Text += $"{DateTime.Now} | Reparsing...\n";

    });
    Task.Factory.StartNew(() => CardParser(DbWorker.sqliteConn)); //ok
}
}
}

```

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

    }
}

```

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 = "Unforseen error";
}
}

private void SetMessageLogPathBtn_Click(object sender, RoutedEventArgs e) //ok
{
    string query = $"UPDATE Settings SET logPath = '{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 SqliteDataReader reader = DbWorker.SettingsReader(DbWorker.readSettings,
DbWorker.sqliteConn);

        datagridExportPath.Text = $"file path has been setted.";
    }
    else

```

```

        {
            datagridExportPath.Clear();
            datagridExportPath.Text += "Unforeseen 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 = "Unforseen 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**:

```

<Window x:Class="botserver_standard.StupidWall"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:botserver_standard"
    mc:Ignorable="d"
    Width="450"
    Height="150"
    Background="#FF4A2E98"
    ResizeMode="NoResize"
    WindowStyle="None"
    WindowStartupLocation="CenterScreen">
<Grid>
    <Label
        Margin="0,10,0,0"
        HorizontalAlignment="Center"
        VerticalAlignment="Top"
        Content="Your password:"
        FontFamily="Cascadia Mono"
        FontSize="15"
        Foreground="#FFB4B4B4" />
    <PasswordBox
        x:Name="EnterPwdBox"
        HorizontalAlignment="Center"
        VerticalContentAlignment="Center"
        Margin="0,42,0,0"
        VerticalAlignment="Top"
        Width="410" Height="22"
        Background="#FF5F3CC0"
        Foreground="#FFB4B4B4"/>
    <Button
        x:Name="CancelBtn"
        Background="#FF5F3CC0"
        Content="Cancel"
        FontFamily="Cascadia Mono"

```

```

        Foreground="#FFB4B4B4"
        Margin="20,93,0,0"
        Height="22"
        HorizontalAlignment="Left"
        Width="155"
        VerticalAlignment="Top" IsCancel="True" />
<Button
    x:Name="NextBtn"
    Background="#FF5F3CC0"
    Click="NextBtn_Click"
    Content="Next"
    FontFamily="Cascadia Mono"
    Foreground="#FFB4B4B4"
    Margin="275,93,0,0"
    Height="22"
    HorizontalAlignment="Left"
    Width="155"
    VerticalAlignment="Top" />

</Grid>
</Window>

```

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[]
            {
                // first row
                new[]
                {
                    InlineKeyboardButton.WithCallbackData(text: "Выбрать программу обучения",
callbackData: "programChoose"),
                    InlineKeyboardButton.WithUrl(text: "Проверить знание русского языка", url:
"https://studyintomsk.2i.tusur.ru/"),
                },
                // second row
                new[]
                {
                    InlineKeyboardButton.WithUrl(text: "Посетить веб-сайт проекта", url:
"https://studyintomsk.ru/"),
                    //InlineKeyboardButton.WithCallbackData(text: "Сменить язык", callbackData:
"langSwitch"),
                },
            },

```

```

});

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: "На главную", callbackData:
"toHome"),
    },
});

```

```

public static readonly InlineKeyboardMarkup universityChoosingKeypad = new(
// keyboard
new[]
{
    // first row
    new[]
    {
        InlineKeyboardButton.WithCallbackData(text: "ТГАСУ", callbackData:
"ТГАСУ_university"),
        InlineKeyboardButton.WithCallbackData(text: "ТГПУ", callbackData:
"ТГПУ_university"),
        InlineKeyboardButton.WithCallbackData(text: "ТГУ", callbackData:
"ТГУ_university"),
    },
    // second row
    new[]
    {
        InlineKeyboardButton.WithCallbackData(text: "ТПУ", callbackData:
"ТПУ_university"),
        InlineKeyboardButton.WithCallbackData(text: "ТУСУР", callbackData:
"ТУСУР_university"),
        //InlineKeyboardButton.WithCallbackData(text: "ФГБОУ ВО СибГМУ",
callbackData: "sixth_university"),
    },
});

```

```

        // third row
        new[]
        {
            InlineKeyboardButton.WithCallbackData(text: "На главную", 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;
        }
    }
}

```

```

    }
}
}

```

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;
        }
    }
}

```


После первого запуска приложения откроется окно установки нового пароля (Рисунок 22).

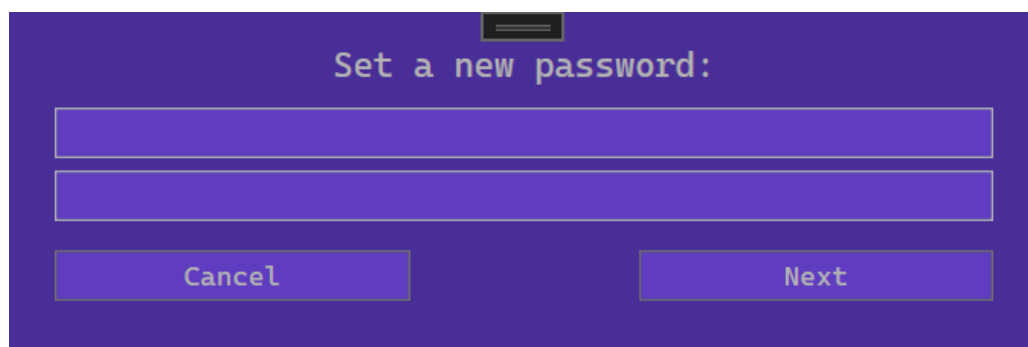


Рисунок 22 – Окно установки нового пароля

Далее необходимо ввести в соответствующие поля пароль и повторить его, затем нажать кнопку «Next» (Рисунок 23), после чего появится уведомление (Рисунок 24), откроется главное окно и во вкладке «Settings» флажок «Use this password» будет установлен (Рисунок 25).

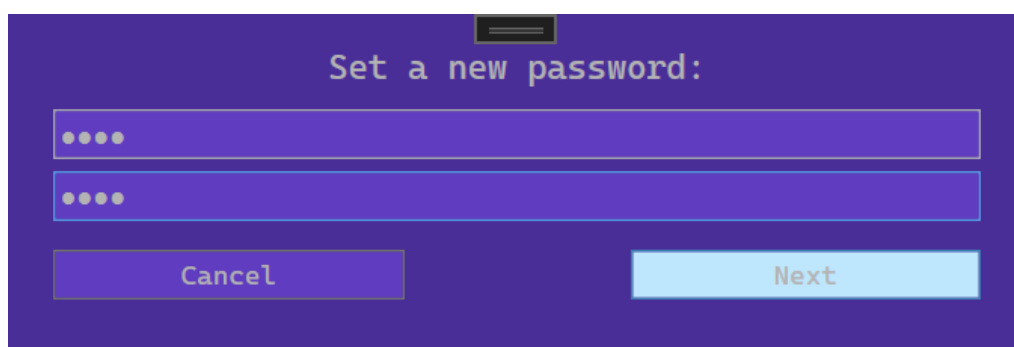


Рисунок 23 – Ввод пароля в окно установки нового пароля

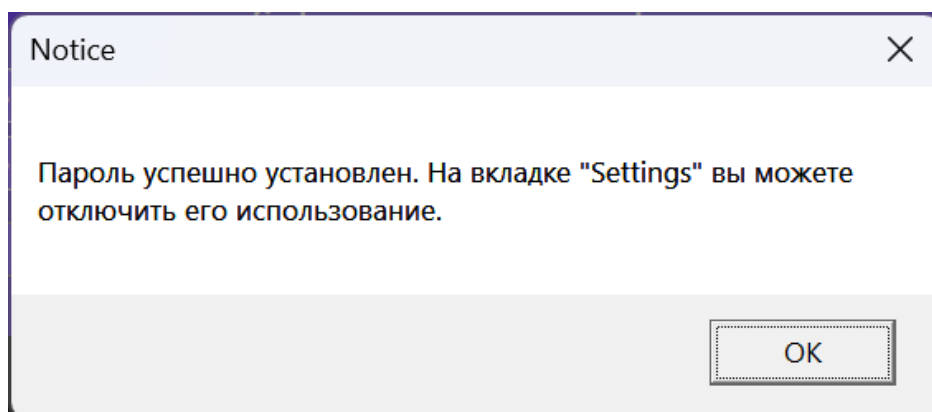


Рисунок 24 – Окно уведомления

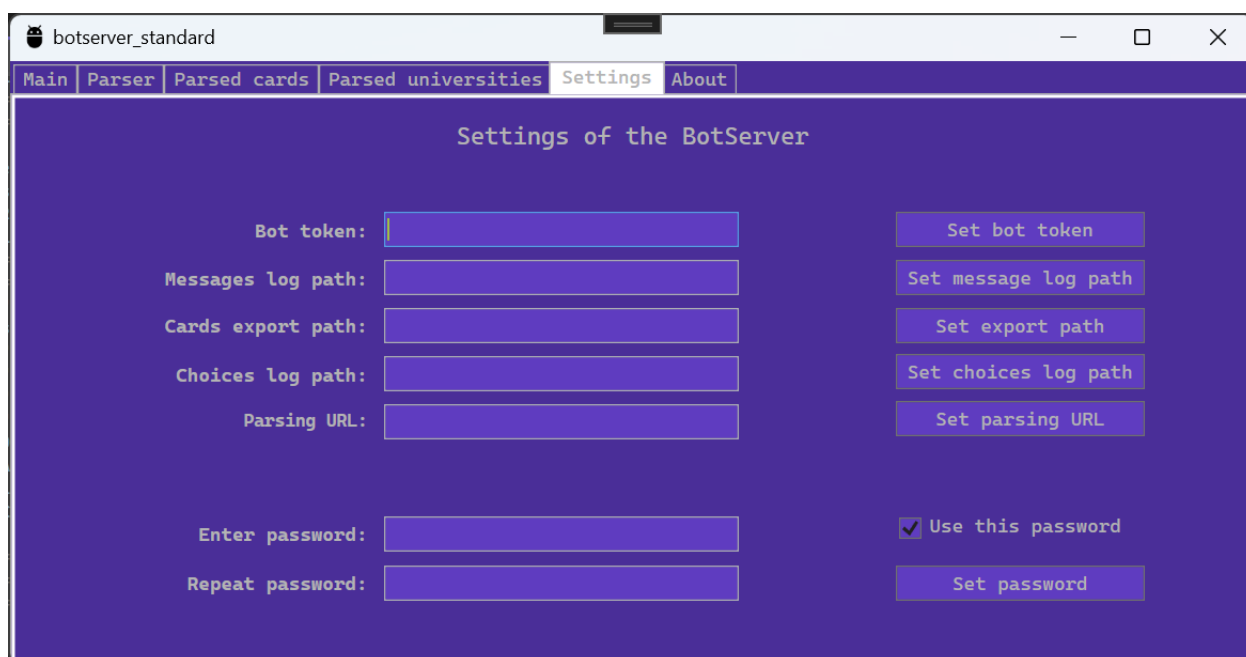


Рисунок 25 – Установленный флажок «Use this password» во вкладке «Settings»

Для запуска бота необходимо дождаться завершения работы парсера.
Отслеживать журнал работы парсера можно во вкладке «Parser» (Рисунок 26)

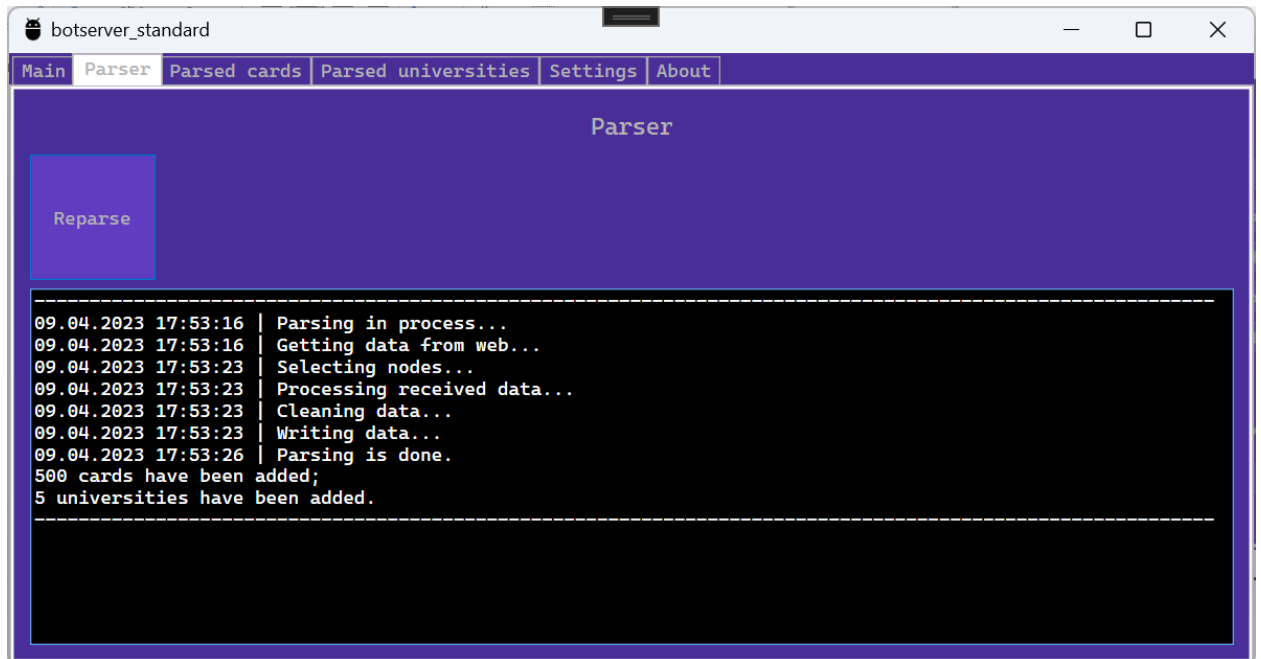


Рисунок 26 – Журнал парсера, сообщающий об окончании его работы

Также в этой вкладке можно перезапустить парсер для сбора свежих данных о карточках направлений. Для этого на вкладке присутствует кнопка «Reparsing». Это обновит полученные ранее данные карточек (Рисунок 27)

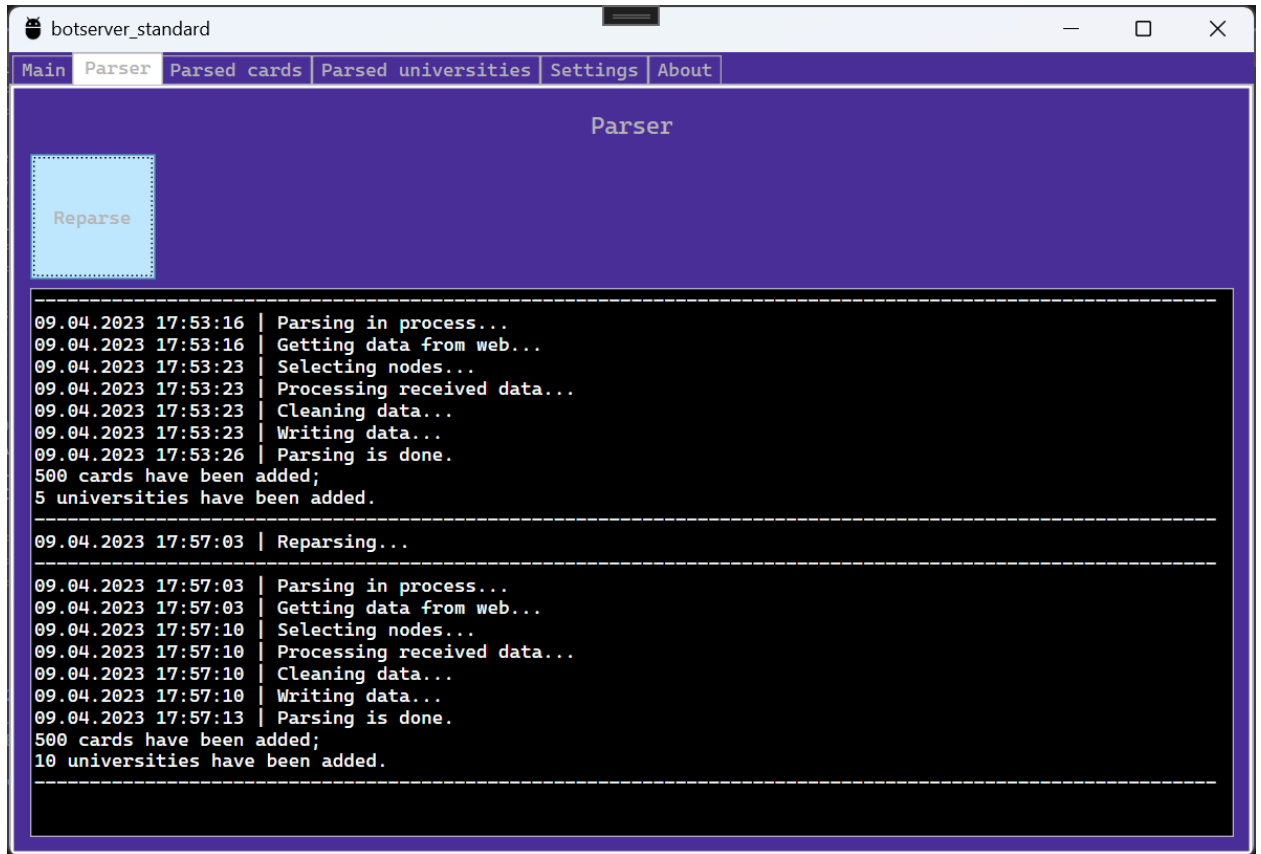


Рисунок 27 – Результат перезапуска парсера

После окончания работы парсера можно переходить к запуску бота. Для этого во вкладке «Main» присутствует кнопка «Start» (Рисунок 28).

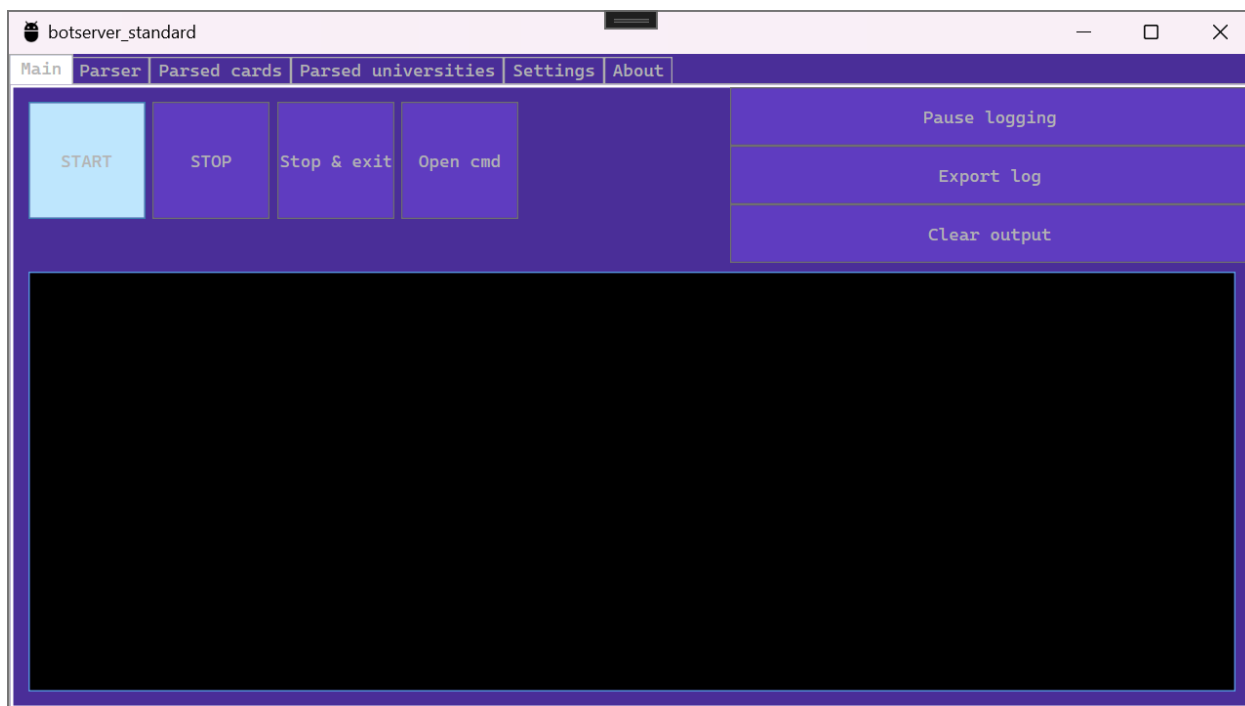


Рисунок 28 – Вкладка «Main»

Также на этой вкладке есть кнопки «STOP», «Stop&exit», «Open cmd», в правой части кнопки «Pause logging», «Export log» и «Clear output».

- «STOP» - позволяет остановить прием ботом сообщений без закрытия GUI;
- «Stop&exit» - останавливает бота и закрывает GUI;
- «Open cmd» - позволяет открыть экземпляр консоли;
- «Pause logging» - позволяет приостановить вывод живого журнала принятых ботом сообщений;
- «Export log» - позволяет экспортировать текущий живой журнал;
- «Clear output» - позволяет очистить вывод;

Для работы всего функционала после первого запуска бота необходимо настроить. Настройка производится в рассмотренной ранее вкладке «Settings» (Рисунок 25). В соответствующие поля вводятся необходимые данные и нажимаются кнопки записи для записи или обновления данных. По результату записи данных выведутся уведомления (Рисунок 29, 30)

botserver_standard

Main Parser Parsed cards Parsed universities Settings About

Settings of the BotServer

Bot token: 5969998133:AAF3nNDLYNfryOulNHKtsxlhuGo_ro Set bot token

Messages log path: messageLog.txt Set message log path

Cards export path: cardsExport.txt Set export path

Choices log path: choicesLog.txt Set choices log path

Parsing URL: https://studyintomsk.ru/programs-main/?le Set parsing URL

Enter password: Use this password ☒

Repeat password: Set password

Рисунок 29 – Заполнение полей данными, необходимыми для работы бота

botserver_standard

Main Parser Parsed cards Parsed universities Settings About

Settings of the BotServer

Bot token: Token has been changed to 5969998133: Set bot token

Messages log path: file path has been setted. Set message log path

Cards export path: file path has been setted. Set export path

Choices log path: Choices log path has been setted. Set choices log path

Parsing URL: Parsing URL has been setted. Set parsing URL

Enter password: Use this password ☒

Repeat password: Set password

Рисунок 30 – Результат записи данных

На вкладке «Parsed cards» отображается список всех карточек направлений, которые смог собрать парсер. Эти данные можно экспортировать посредством нажатия кнопки «Export» (Рисунок 31).

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 32, Рисунок 33).

Main / Парсы / Parsed cards / Парсы universities / Settings / About /									
Поиск по названию программы:		Поиск по университету:		Export					
Название университета	Название программы	Код программы	Уровень обучения	Форма обучения	Срок обучения	Язык обучения	Куратор	Телефон	
ТУСУР	Оптические системы и сети связи	11.03.02	бакалавриат	очная	4 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Системы беспроводной связи и "Интернета вещей"	11.03.02	бакалавриат	очная	4 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Видеоинформационные технологии	11.03.02	бакалавриат	очная	4 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Проектирование и технологии радиоэлектронных средств	11.03.03	бакалавриат	очная	4 года	русский	Денисова Татьяна Владимировна	+7 3822 900 121	...
ТУСУР	Проектирование и технологии электроно-вычислительных средств	11.03.03	бакалавриат	очная	4 года	русский	Денисова Татьяна Владимировна	+7 3822 900 121	...
ТУСУР	Технология электронных средств	11.03.03	бакалавриат	очная	4 года	русский	Денисова Татьяна Владимировна	+7 3822 900 121	...
ТУСУР	Квантовая и оптическая электроника	11.03.04	бакалавриат	очная	4 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Промышленная электроника	11.03.04	бакалавриат	очная	4 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Микроэлектроника и твердотельная электроника	11.03.04	бакалавриат	очная	4 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Волонка нелинейных, волноводных и периодических структур	12.03.03	бакалавриат	очная	4 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Электромагнитная совместимость	11.03.01	бакалавриат	очная	4 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Системы мобильной связи	11.03.02	бакалавриат	очная	4 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Завешенные системы и сети связи	11.03.02	бакалавриат	очная	4 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Автоматизированное управление бизнес-процессами и финансами	09.03.01	бакалавриат	очная	4 года	русский	Хабидулина Надежда Вьевна	+7 3822 900 131	...
ТУСУР	Аналитические информационные системы	09.03.02	бакалавриат	очная	4 года	русский	Хабидулина Надежда Вьевна	+7 3822 900 131	...
ТУСУР	Прикладная информатика в экономике	09.03.03	бакалавриат	очная	4 года	русский	Вадим Ярослав Петрович	+7 3822 900 122	...
ТУСУР	Индустриальная разработка программных продуктов	09.03.04	бакалавриат	очная	4 года	русский	Вадим Ярослав Петрович	+7 3822 900 122	...
ТУСУР	Безопасность автоматизированных систем	10.03.01	бакалавриат	очная	4 года	русский	Миллер Наталья Георгиевна	+7 3822 900 128	...
ТУСУР	Радиотехнические средства передачи, приема и обработки сигналов	11.03.01	бакалавриат	очная	4 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Микроволновая техника и антенны	11.03.01	бакалавриат	очная	4 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Управление разработками робототехнических комплексов	15.04.06	магистратура	очная	2 года	русский	Ситник Александр Анатольевич	+7 3822 900 138	...
ТУСУР	Управление качеством промышленной продукции и услуг	27.04.02	магистратура	очная	2 года	русский	Ситник Александр Анатольевич	+7 3822 900 138	...
ТУСУР	Управление и автоматизация технологических процессов и производств	27.04.04	магистратура	очная	2 года	русский	Хабидулина Надежда Вьевна	+7 3822 900 131	...
ТУСУР	Компьютерное моделирование и обработка информации в технических системах	27.04.05	магистратура	очная	2 года	русский	Хабидулина Надежда Вьевна	+7 3822 900 131	...
ТУСУР	Управление инновациями в электронной технике	38.04.01	магистратура	очная	2 года	русский	Ситник Александр Анатольевич	+7 3822 900 138	...
ТУСУР	Экономика и управление финансами предприятия	38.04.01	магистратура	очная	2 года	русский	Аксенова Янина Николаевна	+7 3822 900 123	...
ТУСУР	Управление бизнесом	38.04.02	магистратура	очная	2 года	русский	Аксенова Янина Николаевна	+7 3822 900 123	...
ТУСУР	Управление проектами	38.04.02	магистратура	очная	2 года	русский	Аксенова Янина Николаевна	+7 3822 900 123	...
ТУСУР	Управление персоналом организации	38.04.03	магистратура	очная	2 года	русский	Аксенова Янина Николаевна	+7 3822 900 123	...
ТУСУР	Цифровое государство и управление	38.04.04	магистратура	очная	2 года	русский	Вадим Ярослав Петрович	+7 3822 900 122	...
ТУСУР	Оптические системы связи и обработки информации	11.04.02	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Радиоэлектронные системы передачи информации	11.04.02	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Конструирование и производство бортовой космической радиоаппаратуры	11.04.04	магистратура	очная	2 года	русский	Денисова Татьяна Владимировна	+7 3822 900 121	...
ТУСУР	Приборы, технологии контроля качества и диагностики	11.04.04	магистратура	очная	2 года	русский	Денисова Татьяна Владимировна	+7 3822 900 121	...
ТУСУР	Твердотельная электроника	11.04.04	магистратура	очная	2 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Квантовая и оптическая электроника	11.04.04	магистратура	очная	2 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Промышленная электроника и микропроцессорная техника	11.04.04	магистратура	очная	2 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Электронные приборы и устройства сбора, обработки и отображения информации	11.04.04	магистратура	очная	2 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Волонка нелинейных и периодических структур	12.04.03	магистратура	очная	2 года	русский	Каранский Виталий Владиславович	+7 3822 900 125	...
ТУСУР	Информационное и программное обеспечение автоматизированных систем	09.04.01	магистратура	очная	2 года	русский	Хабидулина Надежда Вьевна	+7 3822 900 131	...
ТУСУР	Методы и технологии индустриального проектирования программного обеспечения	09.04.04	магистратура	очная	2 года	русский	Вадим Ярослав Петрович	+7 3822 900 122	...
ТУСУР	Радиоэлектронные устройства передачи информации	11.04.01	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Микроволновая техника и антенны	11.04.01	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Видеоинформационные технологии и цифровое телевидение	11.04.01	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Защита от электромагнитного терроризма	11.04.01	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Информационные системы беспроводного широкополосного доступа	11.04.02	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Электромагнитная совместимость радиоэлектронной аппаратуры	11.04.02	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Электронная совместимость в топливно-энергетическом комплексе	11.04.02	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...
ТУСУР	Техническая эксплуатация радиоэлектронного оборудования воздушных судов и аэропортов	25.05.03	специалитет	очная	5,5 лет	русский	Денисова Татьяна Владимировна	+7 3822 900 121	...
ТУСУР	Экономико-правовое обеспечение экономической безопасности	38.05.01	специалитет	очная	5 лет	русский	Миллер Наталья Георгиевна	+7 3822 900 128	...
ТУСУР	Компьютерное моделирование в задачах экологии и техносферной безопасности	01.04.02	магистратура	очная	2 года	русский	Денисова Татьяна Владимировна	+7 3822 900 121	...
ТУСУР	Автоматизация проектирования микро- и нанoeлектронных устройств для радиотехнических систем	09.04.01	магистратура	очная	2 года	русский	Поланских Петр Андреевич	+7 3822 900 132	...

Рисунок 31 – общий вид окна «Parsed cards»

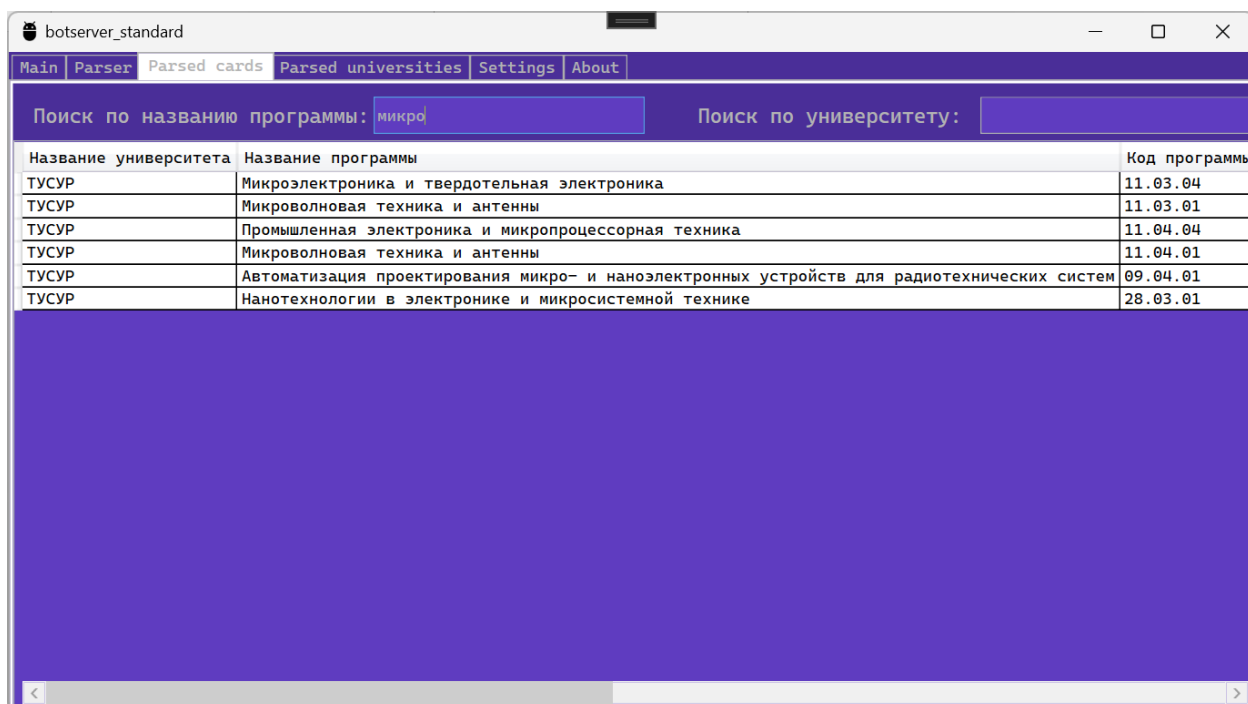


Рисунок 32 – Результат работы умного поиска по названию программы

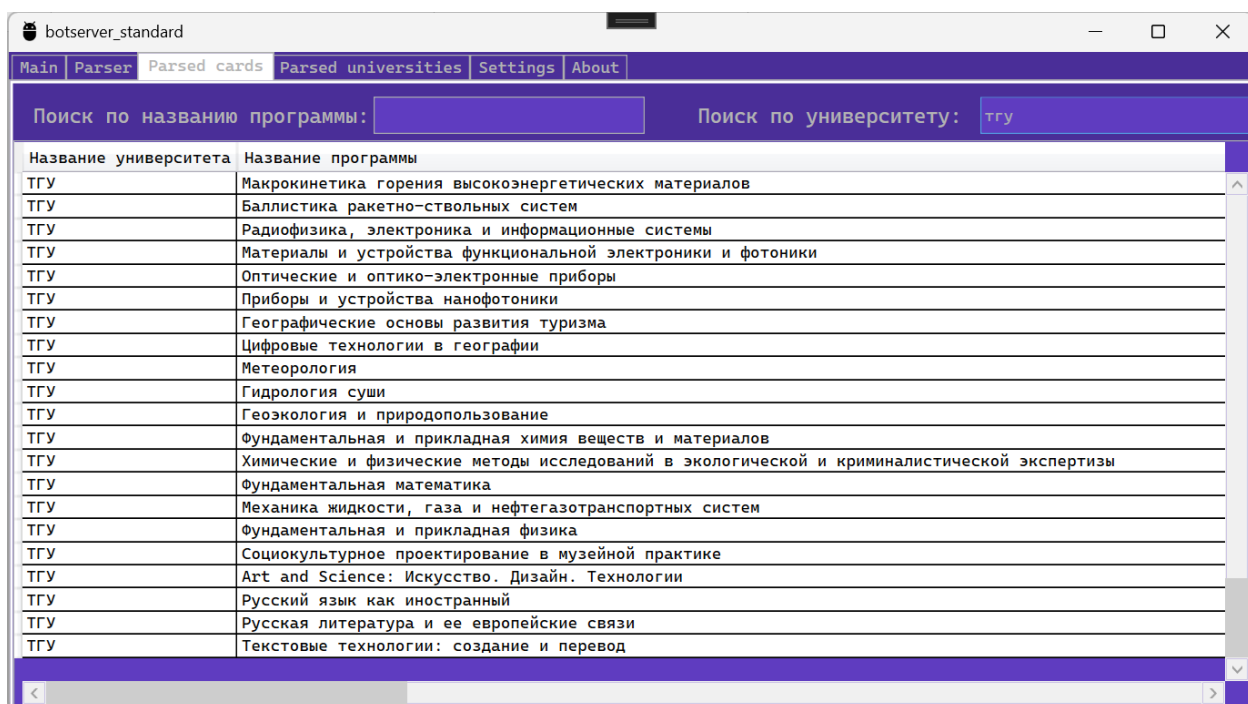


Рисунок 33 – Результат работы умного поиска по названию программы

На вкладке «Parsed universities» отображается список всех университетов с количеством направлений в каждом. (Рисунок 34).

Для поиска по карточкам присутствует два поля – «Поиск по университету» и «Поиск по количеству программ» (Рисунок 35, Рисунок 36).

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

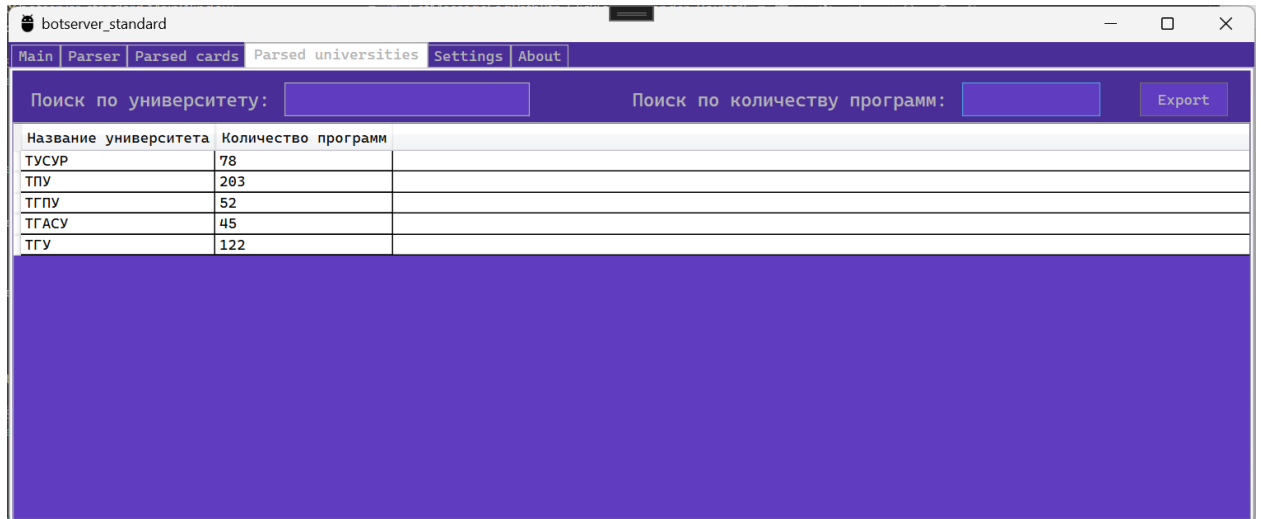


Рисунок 34 – Общий вид вкладки «Parsed universities»

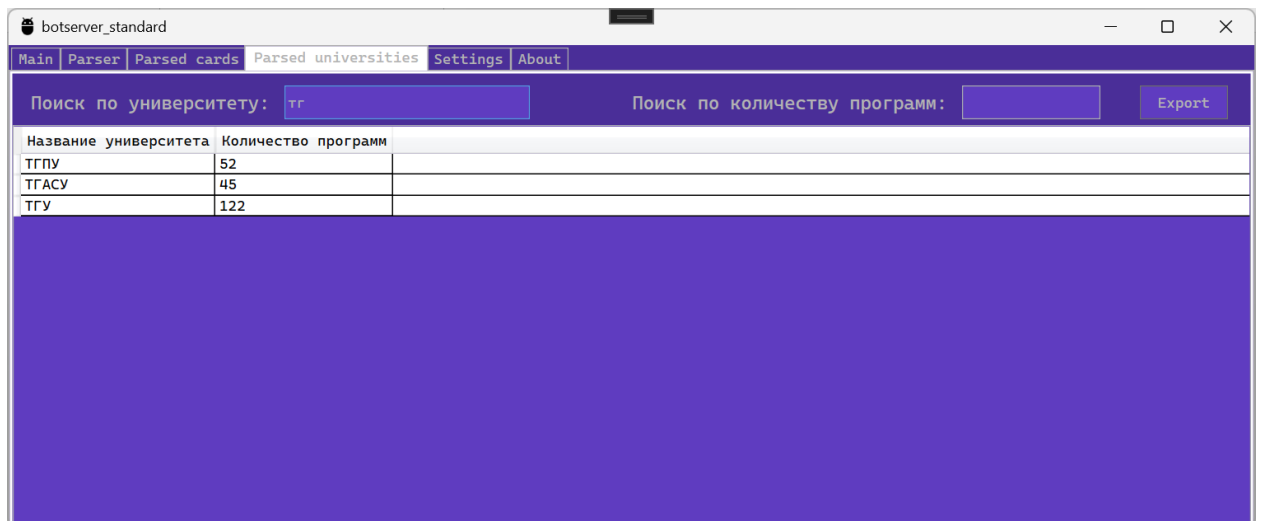


Рисунок 35 - Результат работы умного поиска по университету

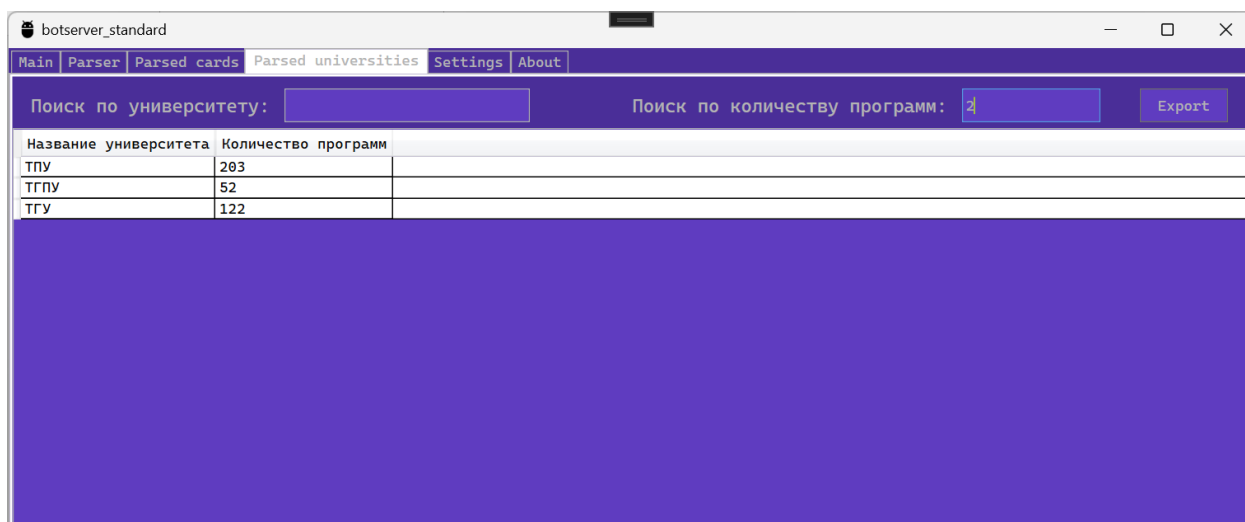


Рисунок 36 - Результат работы умного поиска по количеству программ

На вкладке «Settings» осталась последняя нерассмотренная возможность – смена пароля. Для ее использования необходимо ввести в соответствующие поля новый пароль и повторить его, затем нажать кнопку «Set password» (Рисунок 37).

Появится окно, куда необходимо ввести старый пароль для подтверждения его смены (Рисунок 38). При вводе корректного пароля и подтверждении окно закроется, и на вкладке «Settings» будет отображено уведомление о смене пароля (Рисунок 39)

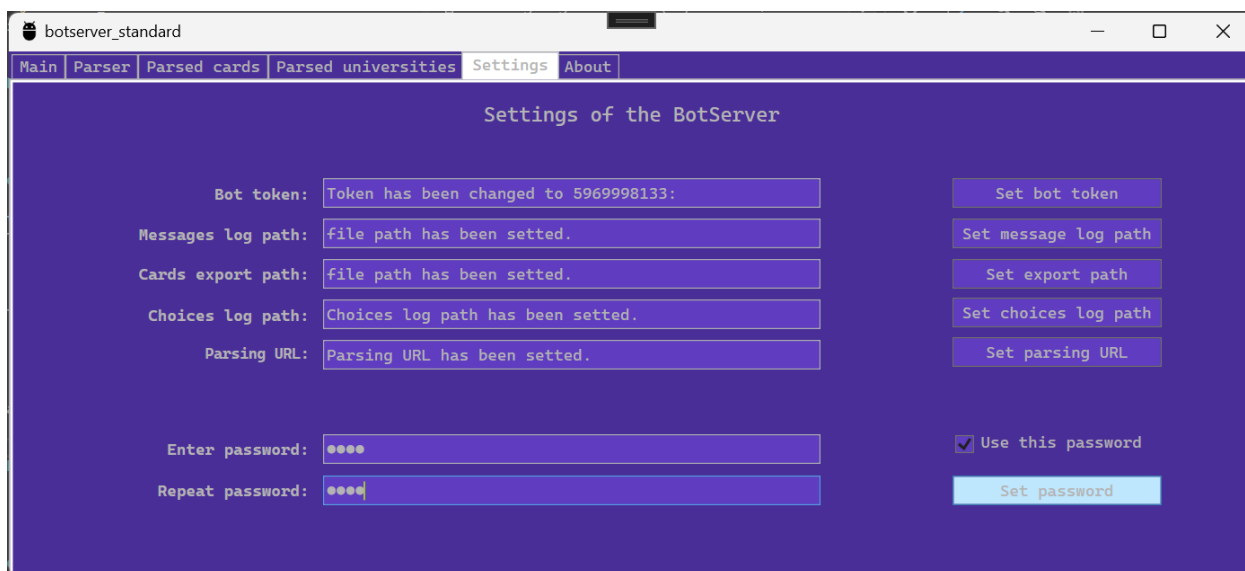


Рисунок 37 – Ввод нового пароля

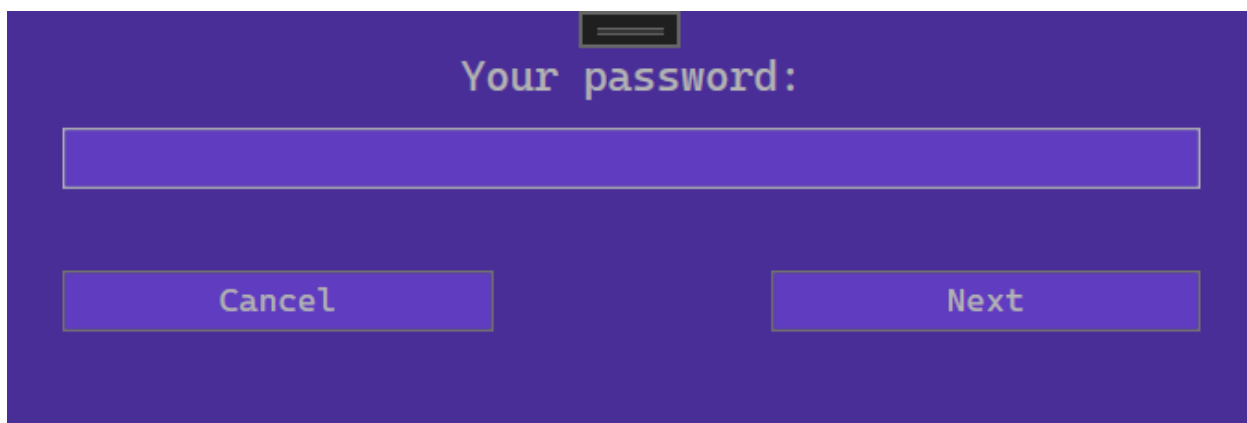


Рисунок 38 – Окно ввода пароля для подтверждения его смены

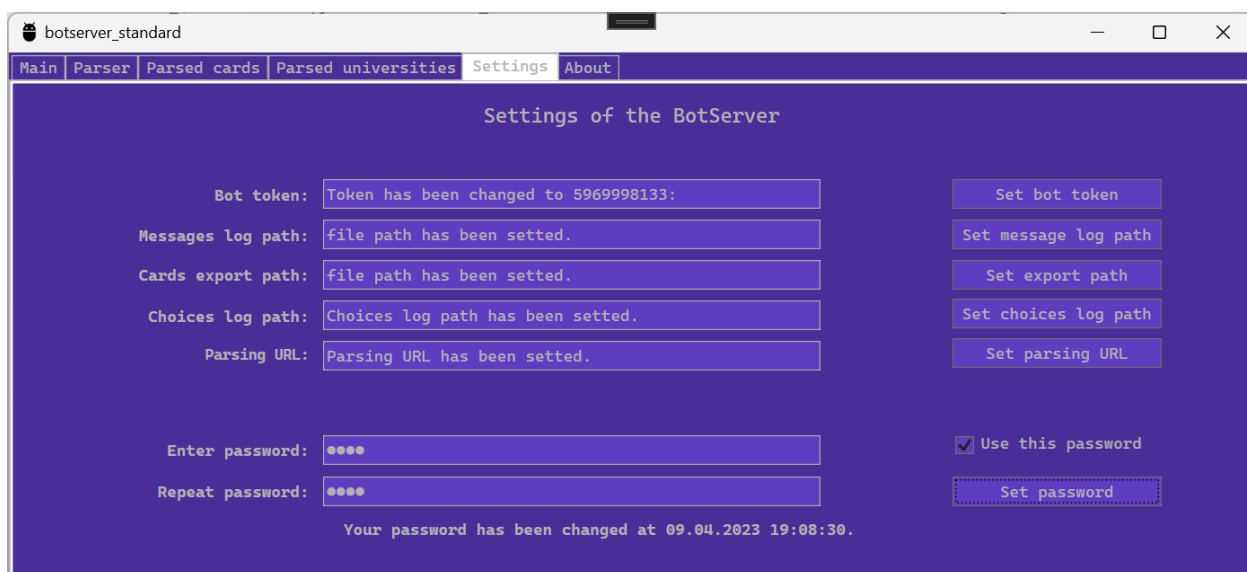


Рисунок 39 – Уведомление об успешной смене пароля

Диаграмма состояния «Установка пароля»

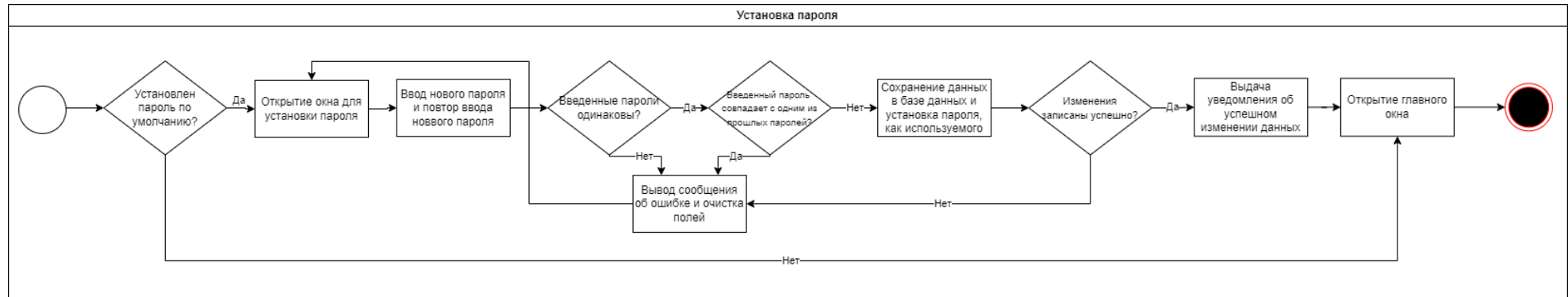


Рисунок 40 – Диаграмма состояния «Установка пароля»

Диаграмма состояния «Вход по паролю»

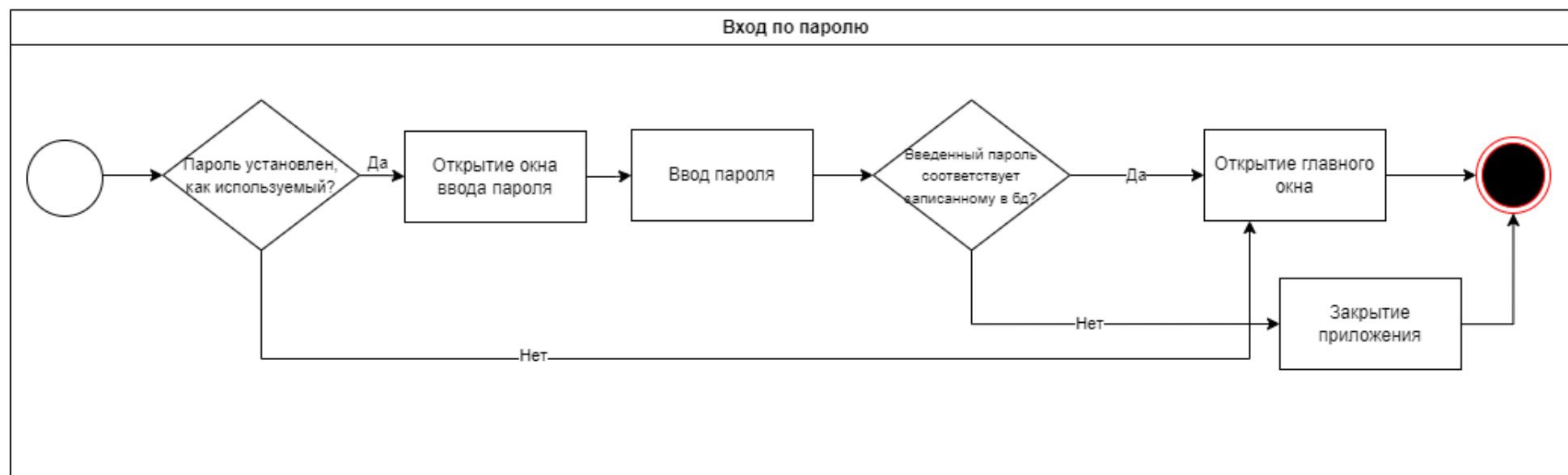


Рисунок 41 – Диаграмма состояния «Вход по паролю»

Диаграмма состояния «Отключение входа по паролю»



Рисунок 42 – Диаграмма состояния «Отключение входа по паролю»

Диаграмма состояния «Включение входа по паролю»

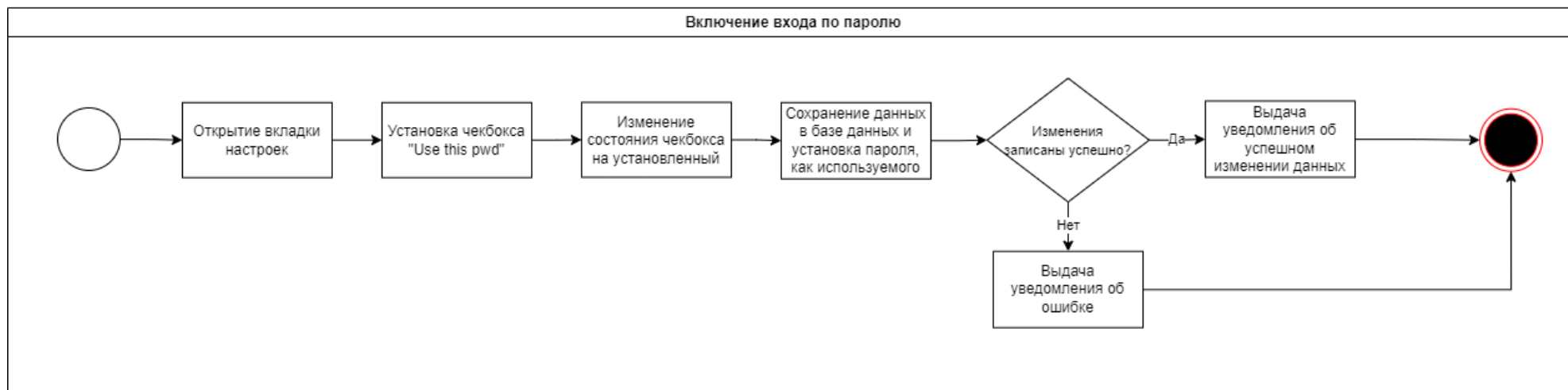


Рисунок 43 – Диаграмма состояния «Включение входа по паролю»

Диаграмма состояния «Установка параметров»



Рисунок 44 – Диаграмма состояния «Установка параметров»

Диаграмма состояния «Выбор пользователем направления обучения»

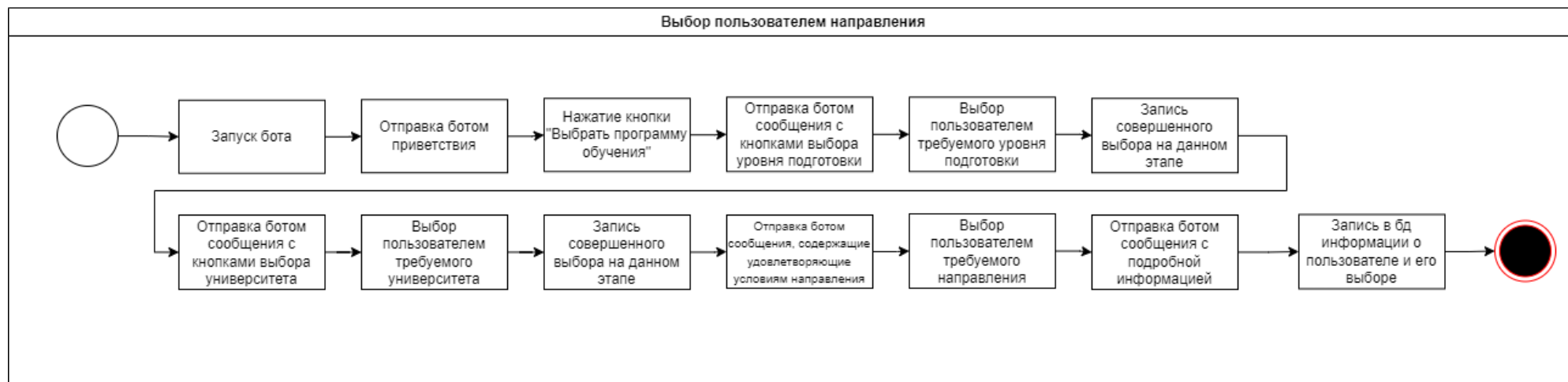


Рисунок 45 – Диаграмма состояния «Выбор пользователем направления обучения»