Содержание

Создание телеграмм бота по сбору данных паспорта

1. Базовые понятия

В данном разделе определяется ряд базовых понятий, которые будут использоваться в дальнейшем (рисунок 1.1):

- *Рабочая область* – центральное окно в программе PIX Studio после создания проекта
- *Окно активностей* – левое окно в программе PIX Studio после создания проекта
- *Активности* – базовые функции, из которых конструируется робот
- *Группа активностей* – активности, объединённые единой идей своего функционала. В дальнейшем для ясности активности будут именоваться следующим образом «Группа активностей/Название активности»
- *Использовать/применить активность* – перетащить активность из окна активностей в рабочую область. Для выполнения данного действия требуется нажать ЛКМ на активность и, зажав ЛКМ, перенести курсор в рабочую область, а затем отпустить ЛКМ

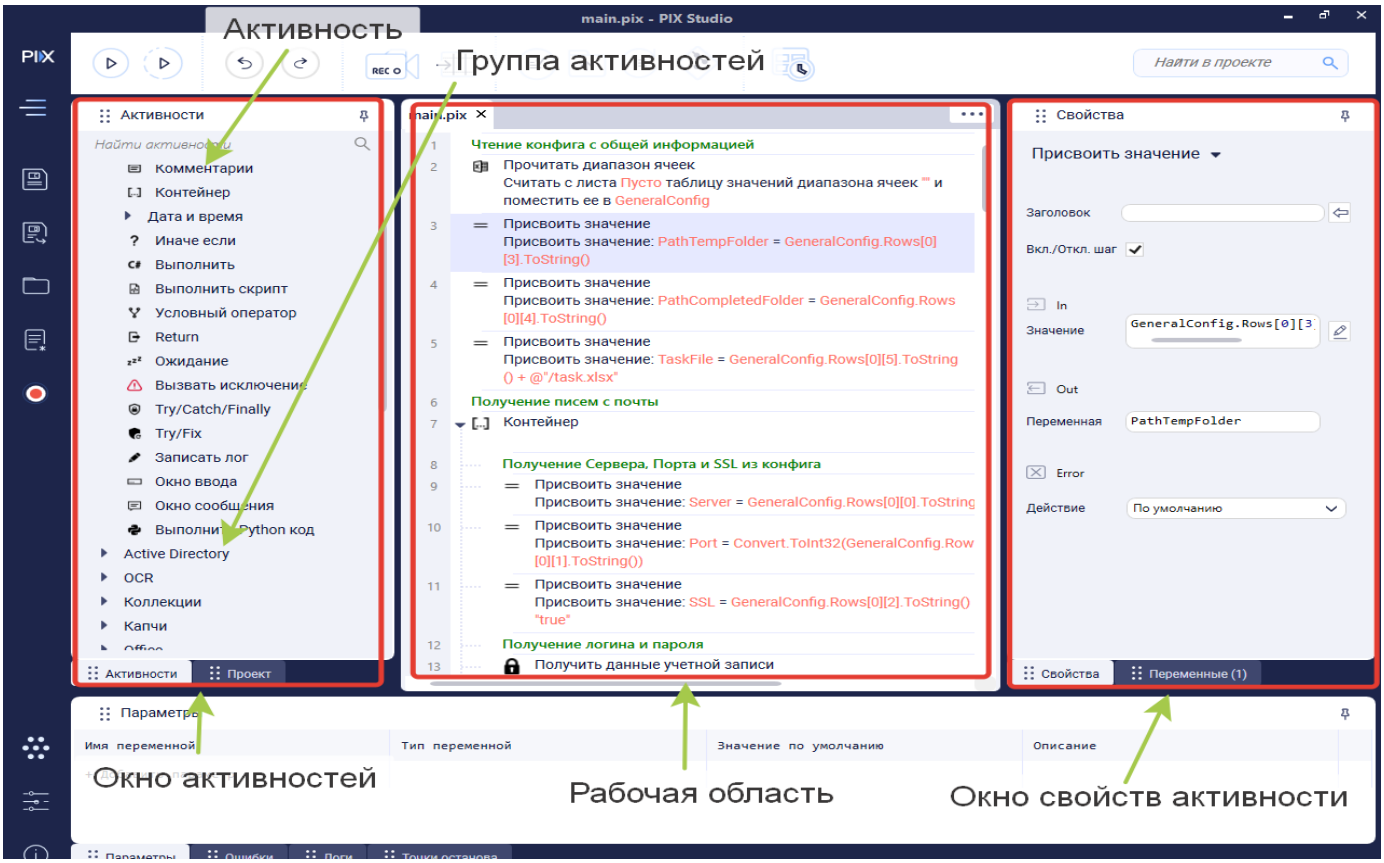


Рисунок 1.1 Окно редактора PIX Studio

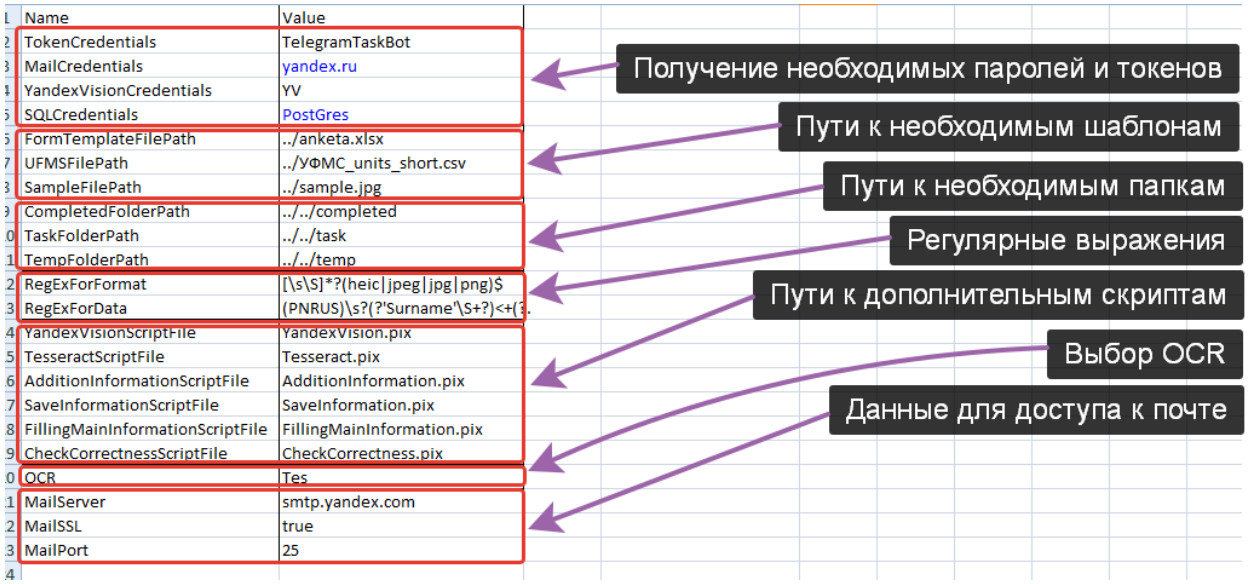
2. Постановка задачи

Робот является телеграмм ботом, который должен у пользователя запросить фотографию паспорта, распознать необходимые данные и отправить их пользователю обратно на проверку. Затем пользователь может изменить получившиеся данные, следом бот запрашивает данные прописки. После их получения робот заносит данные в excel файл и отправляет полученный файл по почте.

3. Предварительные действия

В рамках данной задачи создаётся два робота. Первый является телеграмм ботом, второй обрабатывает полученные данные и отправляет их по почте. Однако в первую очередь, создаётся дополнительный excel документ, который в дальнейшем будет именоваться, как «*конфиг*». В данные документ заносится информация, которая необходима для работы робота, но может быть изменена сторонними действиями. Для данной задачи конфиг может содержать следующую информацию (Рисунок 3.1):

- Относительные пути к необходимым папкам и файлам (путь к папке с необработанными файлами, путь к папке с результатом обработки и т.д.)
- Регулярные выражения для определения формата файла и обработки распознанного текста.
- Данные для доступа к почте и различным паролям



3.1. Пример конфига для робота.

А также на отдельной странице заполняются данные для записи полученных данных (Рисунок 3.2).

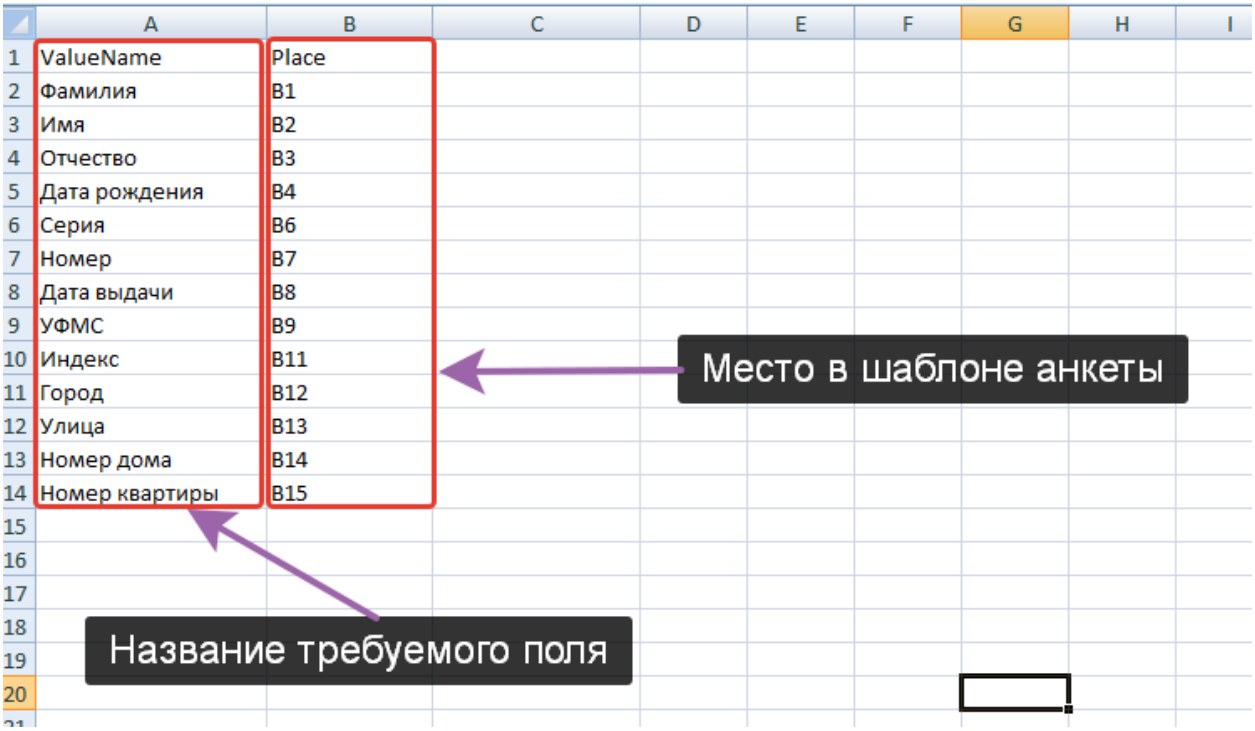


Рисунок 3.2. Пример данных для записи результата

Помимо конфига необходимо убедится в наличие шаблона с примером фотографии паспорта, шаблона анкеты. А также в наличие csv файла с названиями УФМС и их кодами.

Так же стоит написать регулярные выражения для обработки распознанного текста и определения формата файла. Примеры регулярных выражений для данных целей представлены на рисунке 3.2.

Для написания регулярных выражений можно использовать сайт «regex101.com» или какой-нибудь подобный.

```
/ [\s\S]*?(heic|jpeg|jpg|png)$
```

```
/ (PNRUS)\s?(?'Name'\S+?)<+
(?'Surname'\S+?)<+(?'LastName'\S+?)
<+?\S+?(?'Seria'\d{3})
(?'Number'\d{6})[\S<]+?<+
(?'LastSeria'\d)(?'Date'\d{6})
(?'Code1'\d{3})(?'Code2'\d{3})<\S+$
```

Рисунок 3.3 Пример регулярных выражений для распознавания формата файла (слева) и обработки распознанного текста (справа)

Так же в Windows Credentials нужно записать логин и пароль к почте, токен подключения к OCR, токен подключения к боту. И создать профиль самого бота используя бот @BotFather (https://t.me/BotFather).

4. Работа с PIX Studio

Теперь необходимо запустить PIX Studio и создать два новых проект. Для этого сначала необходимо нажать на панели слева на иконку блокнота со звёздочкой в левом нижнем углу. В появившемся окне выбирается пункт проект (Рисунок 4.1). PIX Studio попросит ввести название проекта и его расположение. Также для одного из проектов необходимо создать 6 дополнительных скриптов. Делается это аналогичным образом, но в окне, появляющемся после нажатия на иконку блокнота выбирается пункт скрипт.

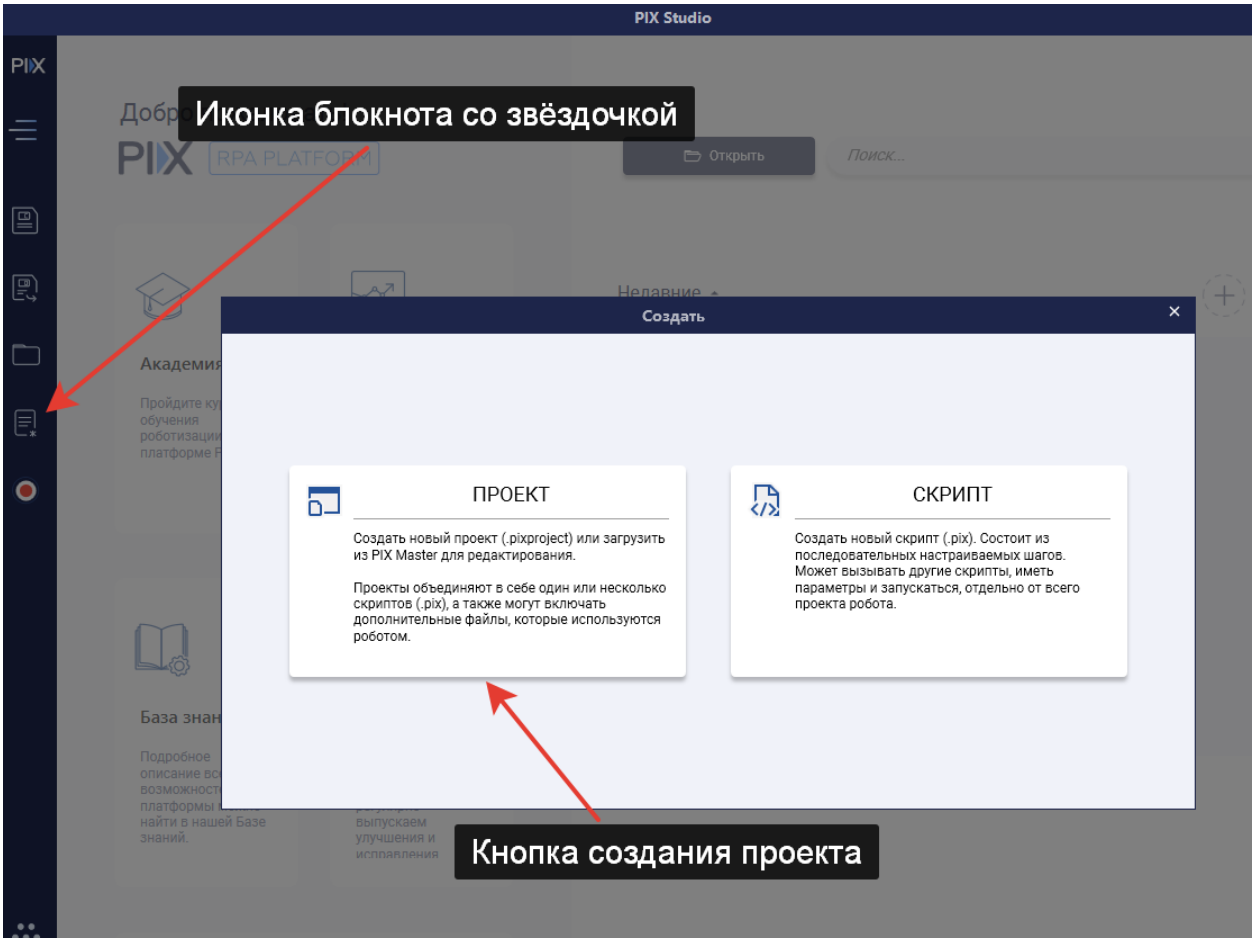


Рисунок 4.1. Создание проекта

5. Проект Telegram бота

В данном проекте присутствует 7 скриптов:

- *Main.pix* – Отвечает за запуск бота и начальную работу
- *FillingMainInformation.pix* – Заполнение основных паспортных данных.
- *Tesseract.pix* – Распознаёт содержимое фото с помощью OCR Tesseract.
- *CheckCorrectness.pix* – Проверка информации на корректность.
- *AdditionInformation.pix* – Заполнение дополнительной информации о пользователе.
- *SaveInformation.pix* – Сохранение полученной информации.

5.1. Скрипт Main.pix

В первую очередь с помощью активности *«Базовые/Присвоить значения»* и C# функции *new* инициализируются важные переменные. (Рисунок 5.1.1)

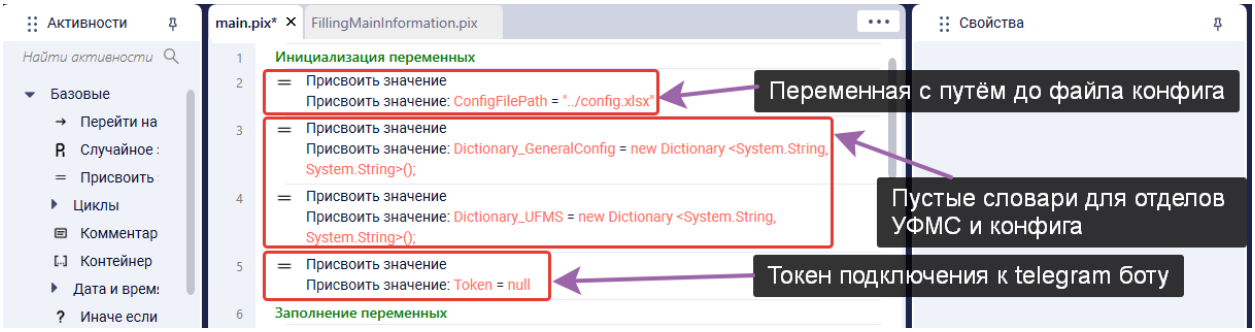


Рисунок 5.1.1. Инициализация переменных

Затем с помощью активности *«Office/Excel/Прочитать диапазон ячеек»* считывается содержимое конфига и заносится в таблицу. В свойствах активности заполняется: *Лист* – название листа с конфигом, *Диапазон* – указывается «""» для прочтения всего диапазона, *С заголовками* – ставится галочка в случае использования строки заголовков в конфиге, *Путь к файлу* – путь до файла с конфигом, *Таблица* – переменная с результирующей таблицей. Затем с помощью активности *«Базовые/Присвоить значение»* и C# функции *Таблица.AsEnumerable().ToDictionary<DataRow, Формат_ключа,*

Формат_значения>(row => row.Field<Формат_ключа>(Столбец_с_ключами), row => row.Field<Формат_значения>(Столбец_со_значениями)) таблица конфига переводится в словарь. (Рисунок 5.1.2)

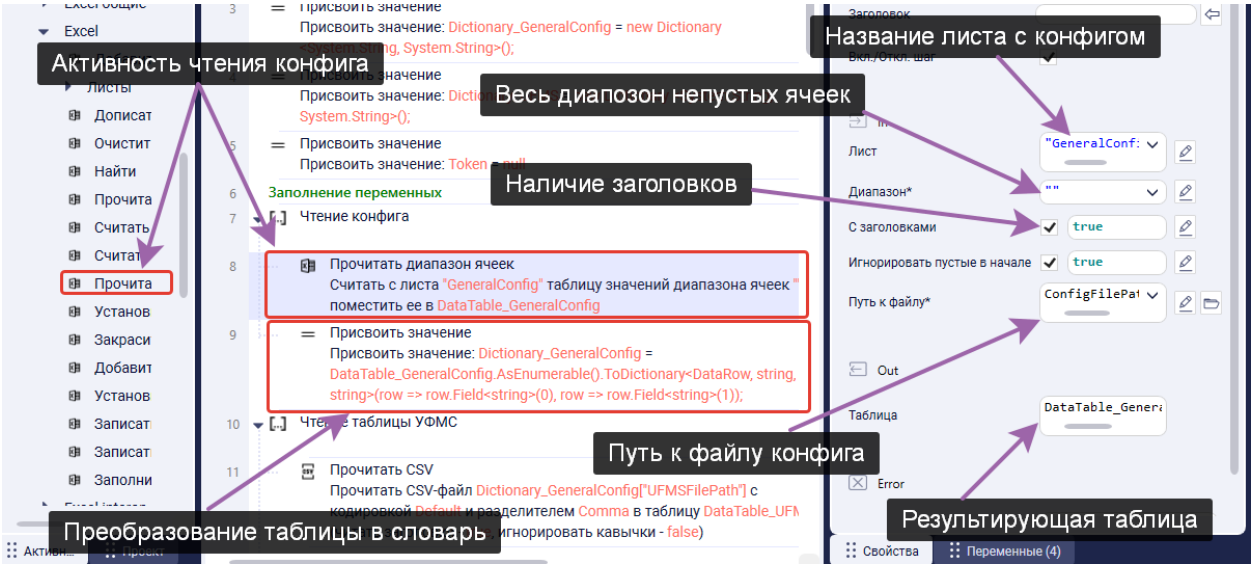


Рисунок 5.1.2. Чтение конфига

Далее с помощью активности *«CSV/Прочитать CSV»* считывается таблица с кодами подразделения и отделами УФМС. В свойствах активности указывается: *Читать заголовки?* – наличие заголовков в CSV файлов, *Путь к CSV-файлу* – путь к файлу с отделами УФМС, *Разделитель* – символ разделения в CSV файле, *Таблица* – результирующая таблица. После таблица аналогичным образом преобразуется в словарь.

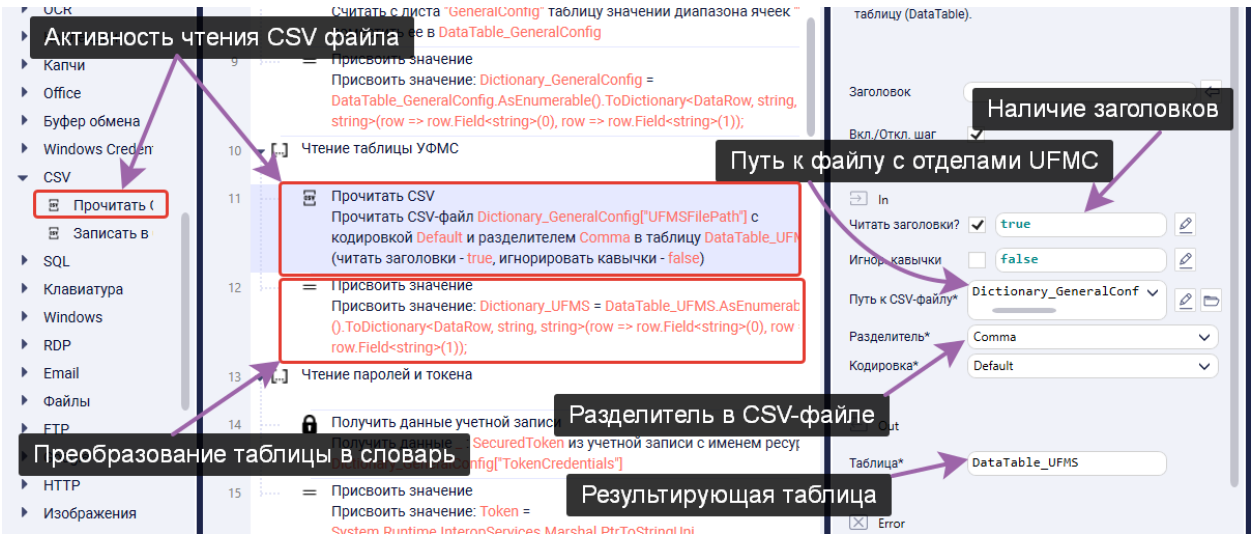


Рисунок 5.1.3. Чтение файла с отделами УФМС

Следующим шагом с помощью активности *«Windows Credentials/Получить данные учетной записи»* считывается значение токена для бота Telegram. В свойствах заполняется: *Имя ресурса* – наименование под которым сохранено значение, *Логин* – переменная, в которую сохранится строка с токеном, если значение сохранялось не в защищённом виде, *Пароль* – переменная с защищённой строкой, если значение сохранялось в защищённом виде. В случае, если токен хранится в защищённом виде, то с помощью активности *«Базовые/Присвоить значения»* и C# функции *System.Runtime.InteropServices.Marshal.PtrToStringUni(System.Runtime.InteropService.Marshal.SecureStringToGlobalAllocUnicode(Защищённый_токен))* токен приводится к строке. (Рисунок 5.1.4)

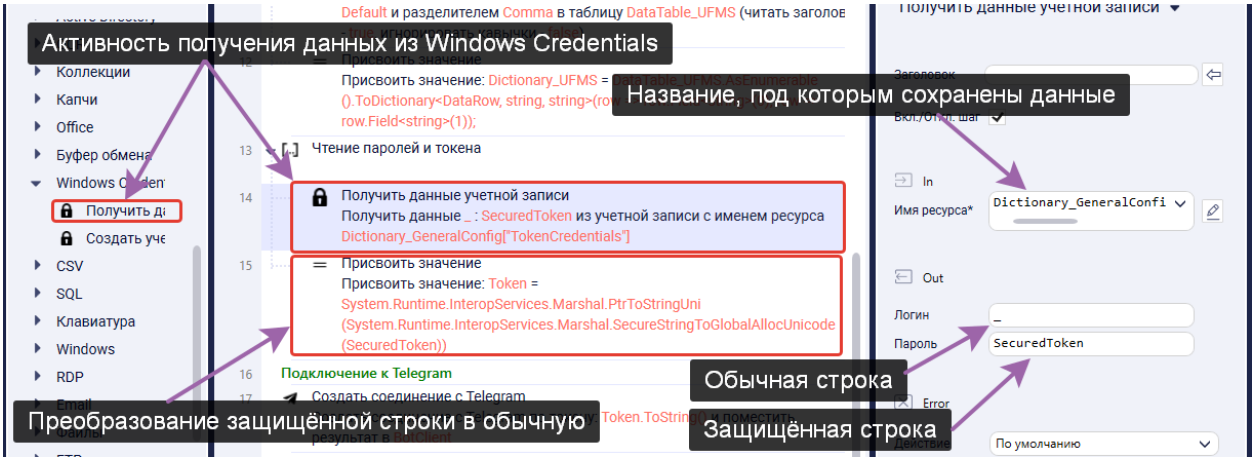


Рисунок 5.1.4. Считывание токена для telegram

Следующим шагом с помощью активности *«Мессенджеры/Telegram/Создать соединение с Telegram»* создаётся подключение к телеграмм боту. В свойствах указывается: *Токен* – токен подключения к телеграмм боту, *Соединение* – переменная с соединением к телеграмму. После с помощью активности *«Мессенджеры/Telegram/Менеджер Telegram»* запускается бесконечный цикл считывания входящих сообщений. В свойствах заполняется: *Соединение* – переменная с соединением к телеграмму. (Рисунок 5.1.5)

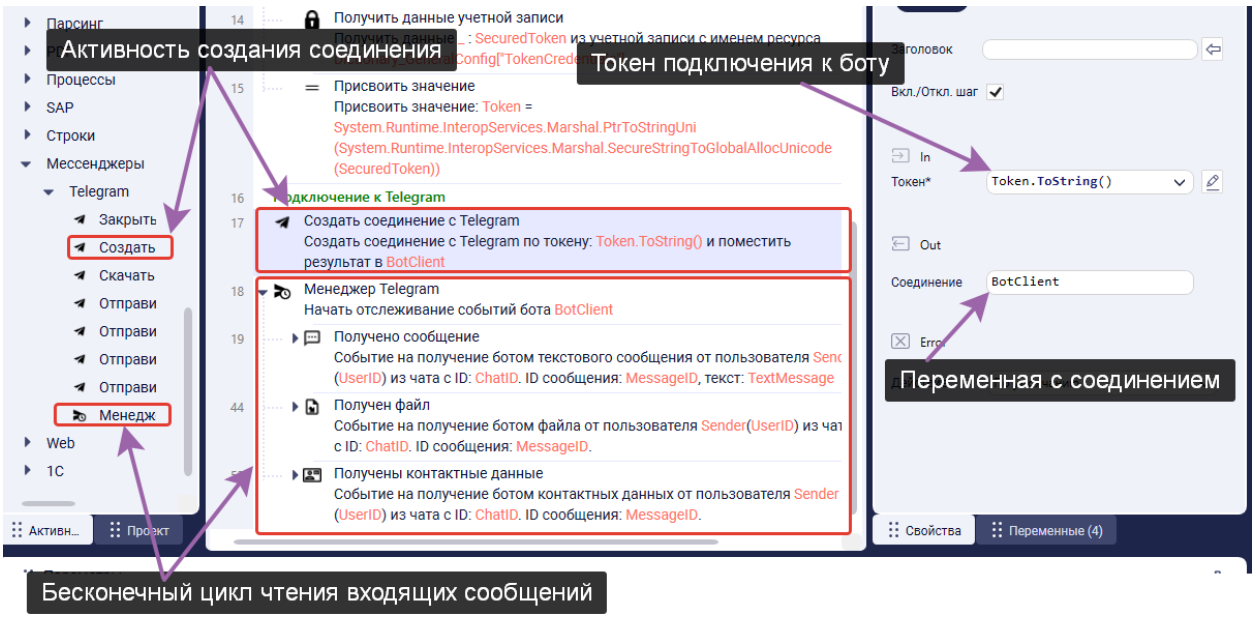


Рисунок 5.1.5. Подключение к телеграмм

В случае, если пришёл контакт другого пользователя, то с помощью активности *«Мессенджеры/Telegram/Отправить фото в чат»* в чат отправляется шаблон скана паспорта с подписью, что надо отправить скан паспорта. В свойствах активности указывается: *Фото* – путь к шаблону фото скана, *Подпись* – необходимый текст, *Соединение* – соединение с телеграмм, *ID чата* – ID чата, которое инициализируется в соответствующем разделе активности *«Мессенджеры/Telegram/Менеджер Telegram»*. (Рисунок 5.1.6)

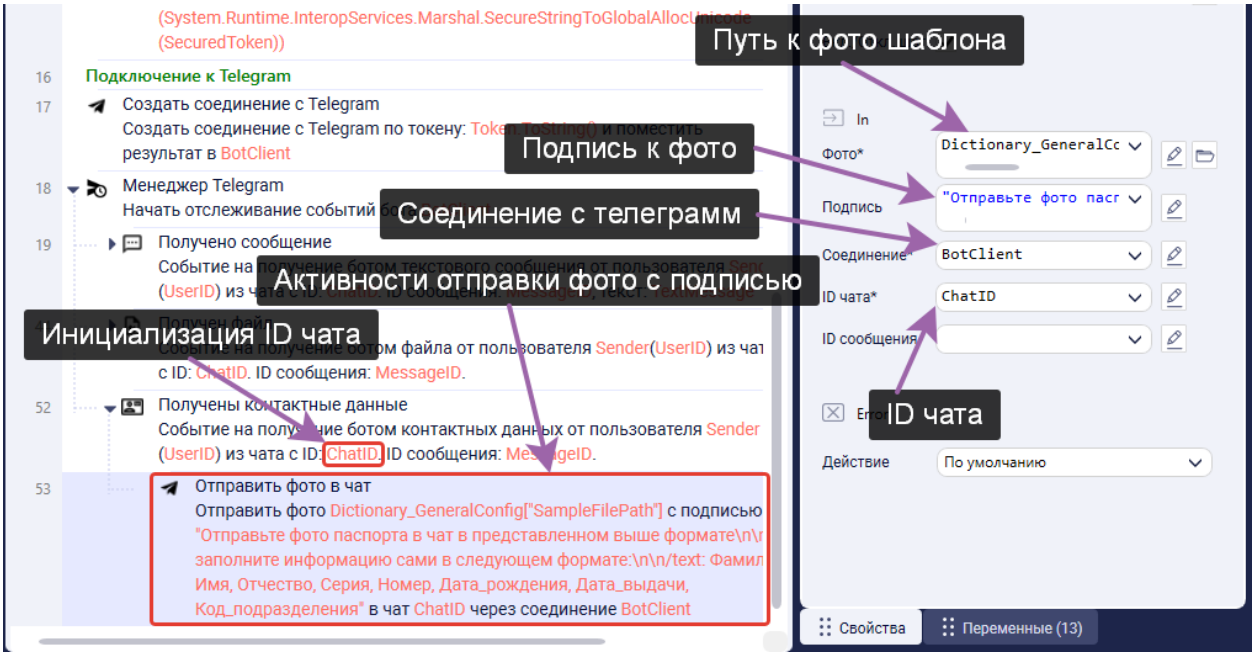


Рисунок 5.1.6. Отправка шаблона входящего сообщения

В случае, если получен файл, формат файла с помощью активности *«Строки/Регулярное выражение»* проверяется на соответствие фото. В свойствах указывается: *Входная строка* – строка с форматом файла, инициализируется в соответствующем разделе активности *«Мессенджеры/Telegram/Менеджер Telegram»*, *Регулярное выражение* – регулярное выражение проверки формата, *Опции* – опции регулярного выражения, *Результат* – булева переменная с результатами проверки. Затем с помощью активности *«Базовые/Условный оператор»* в зависимости от результатов проверки либо как в предыдущем случае отправляется шаблон входящего сообщения, либо вызывается активность *«Базовые/Выполнить скрипт»*, которая запускает скрипт *FillingMainInformation.pix*. (Рисунок 5.1.7)

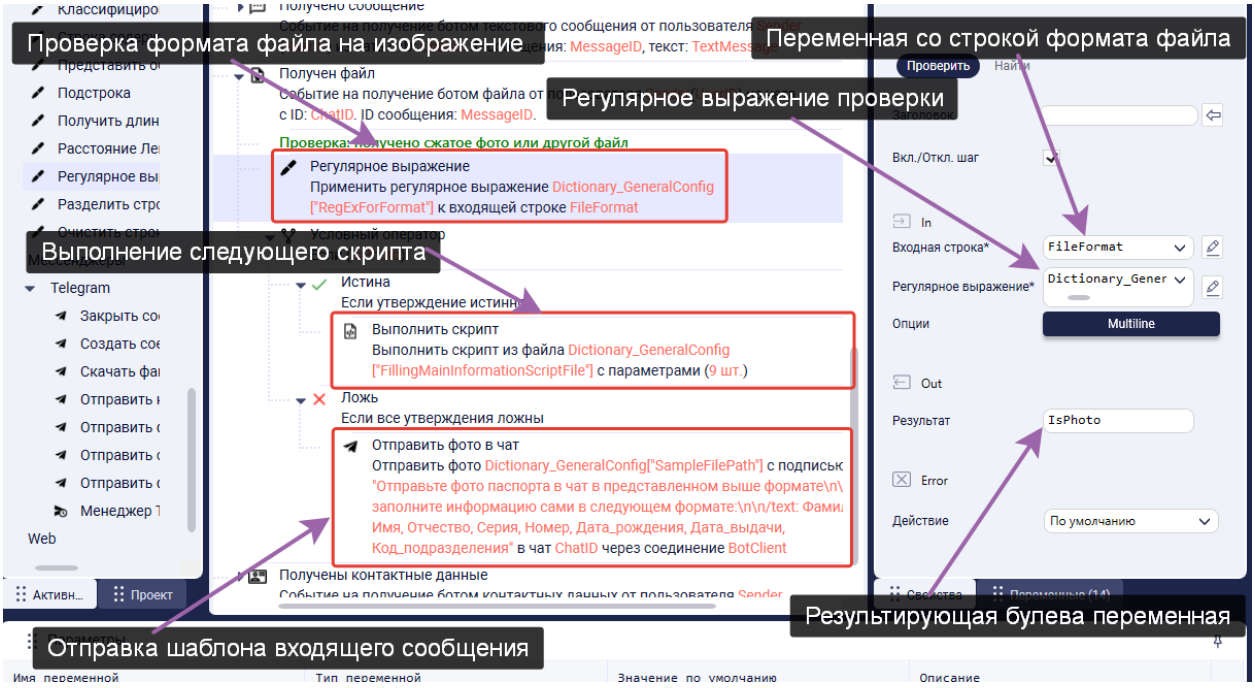


Рисунок 5.1.7. Проверка на получение фото

В случае, если входящее сообщение – это текст, то в соответствующем разделе активности *«Мессенджеры/Telegram/Менеджер Telegram»* он сохраняется в переменную, затем данная переменная с помощью активности *«Базовые/Условный оператор»* и C# функции *Строка.Equals(Строка_для_сравнения)* текст сообщения сравнивается на соответствие одной из команд. Если текст является командой выключения бота, то с помощью активности *«Мессенджеры/Telegram/Закрыть*

соединение Telegram» обрывается соединение с телеграммом. Если текст совпадает с командой ручного ввода паспортных данных, то вызывается активность *«Базовые/Выполнить скрипт»*, которая запускает скрипт *FillingMainInformation.pix*. Если текст – это команда удаления анкеты, то с помощью активности *«Файлы/Путь существует?»* проверяется существование файла с анкетой, в случае, когда он существует, то он удаляется с помощью активности *«Файлы/Удалить файл/каталог»* и с помощью активности *Мессенджеры/Telegram/Отправить сообщение в чат»* отправляется сообщение об успешном удаление, в ином случае той же активностью отправляется сообщение об отсутствие файла. Если полученное сообщение не совпадает ни с одной командой, отправляется шаблон входящего сообщения. (Рисунок 5.1.8)

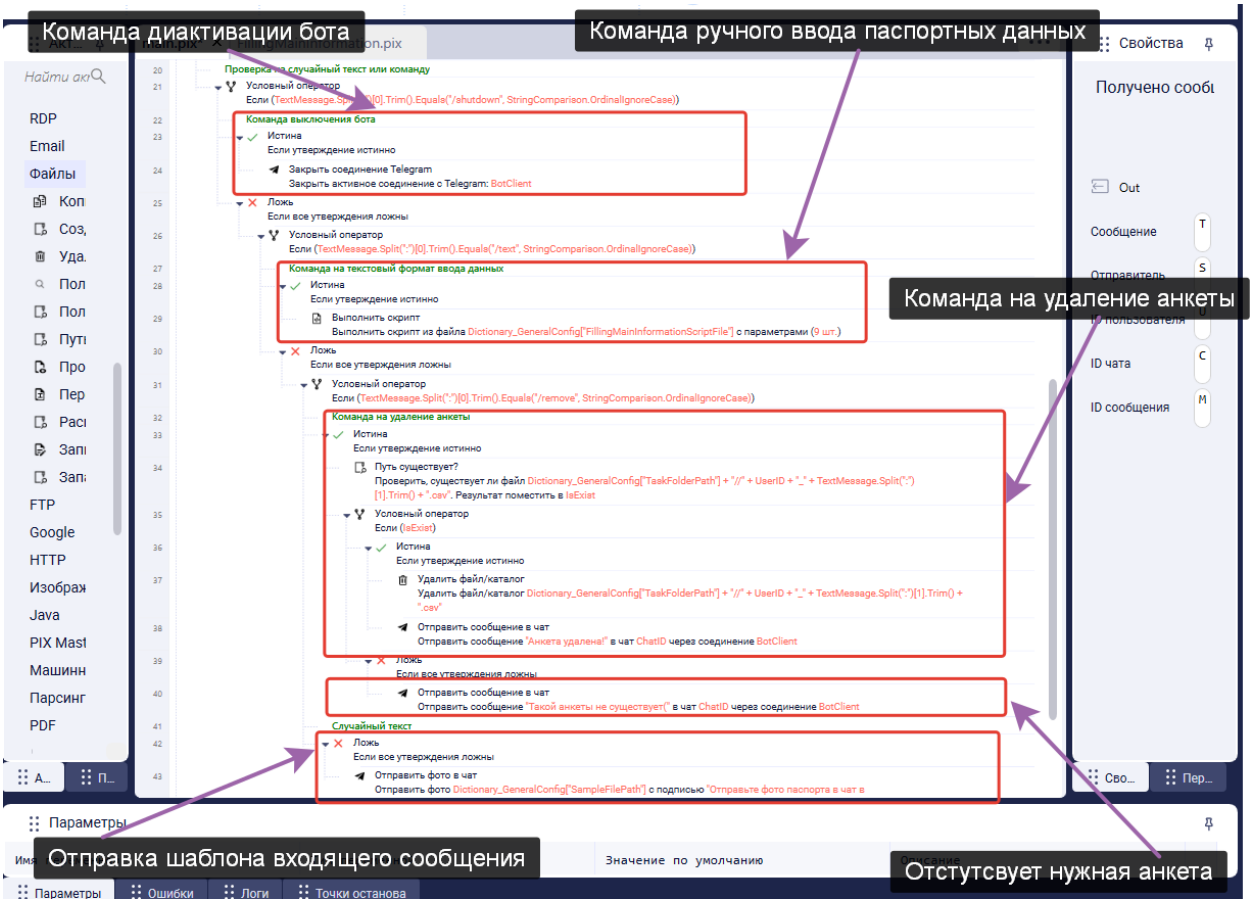


Рисунок 5.1.8. Проверка полученного текста.

В активности *«Базовые/Выполнить скрипт»* при вызове заполняется окно *Параметры запуска скрипта*. В случае вызова скрипта

FillingMainInformation.pix. Заполняются следующие параметры (Рисунок 5.1.9)

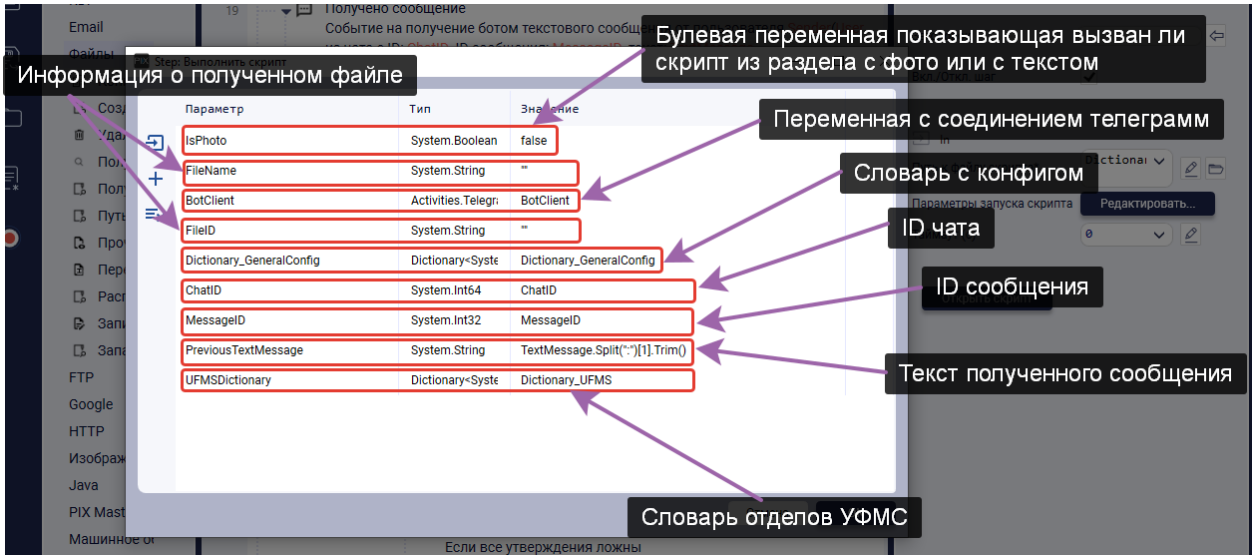


Рисунок 5.1.9. Вызов следующего скрипта

5.2. Скрипт FillingMainInformation.pix

Первым делом с помощью активности *«Коллекции/Словарь/Создать словарь»* создаётся словарь для хранения паспортных данных и данных прописки. Затем с помощью активности *«Базовые/Условный оператор»* рассматриваются случаи получения паспортных данных в виде фото или в текстовом формате.

Если данные получены в текстовом формате, то с помощью активности *«Файлы/Путь существует?»* проверяется существование папки для хранения изображений, и в случае её отсутствия, она создаётся с помощью активности *«Файлы/Создать папку»*.

Также создаётся переменная, в которую помещается путь до изображения. (Рисунок 5.2.1)

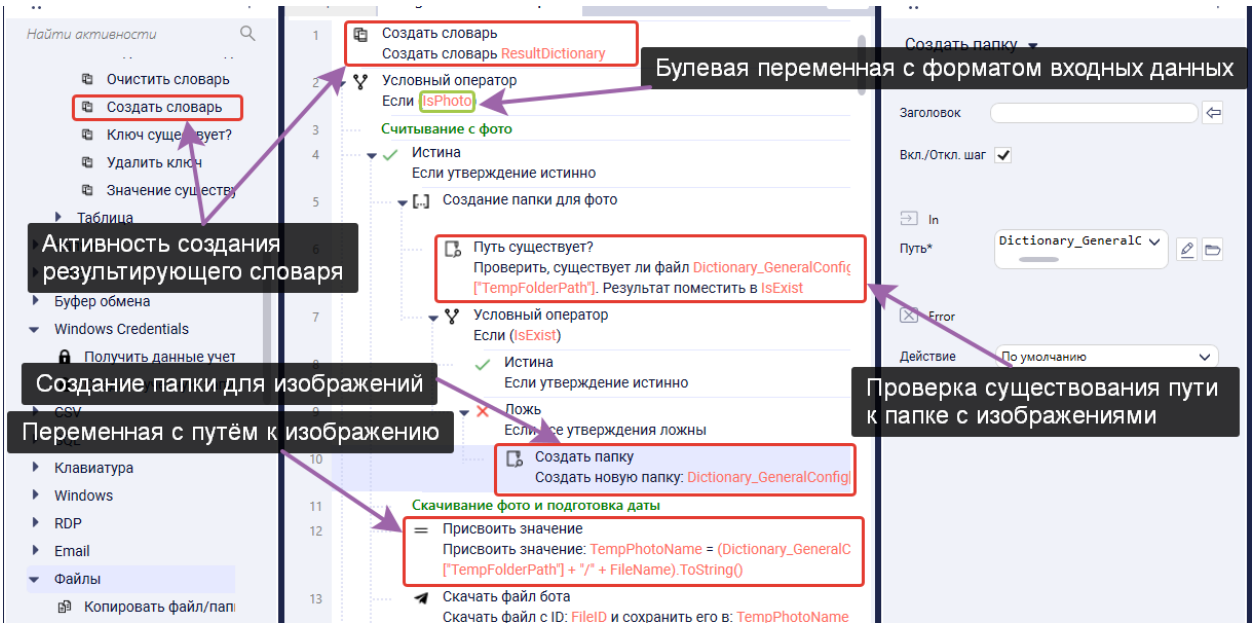


Рисунок 5.2.1. Предварительные действия по обработке сообщений

Следующим делом с помощью активности *«Мессенджеры/Telegram/Скачать файл бота»* скачивается изображение со сканом паспорта. В свойствах указывается: *Соединение* – соединение с телеграммом, *ID файла* – переменная указывающая, какой файл надо скачать, *Путь* – путь, куда сохранить файл.

Затем с помощью активности *«Базовые/Присвоить значение»* создаётся переменная для хранения текста с изображения. И с помощью активностей *«Базовые/Условный оператор»* и *«Базовые/Выполнить скрипт»* вызывается скрип получения текста с изображения с использованием выбранного в конфиге OCR, результат сохраняется в созданную раннее переменную. (Рисунок 5.2.2)

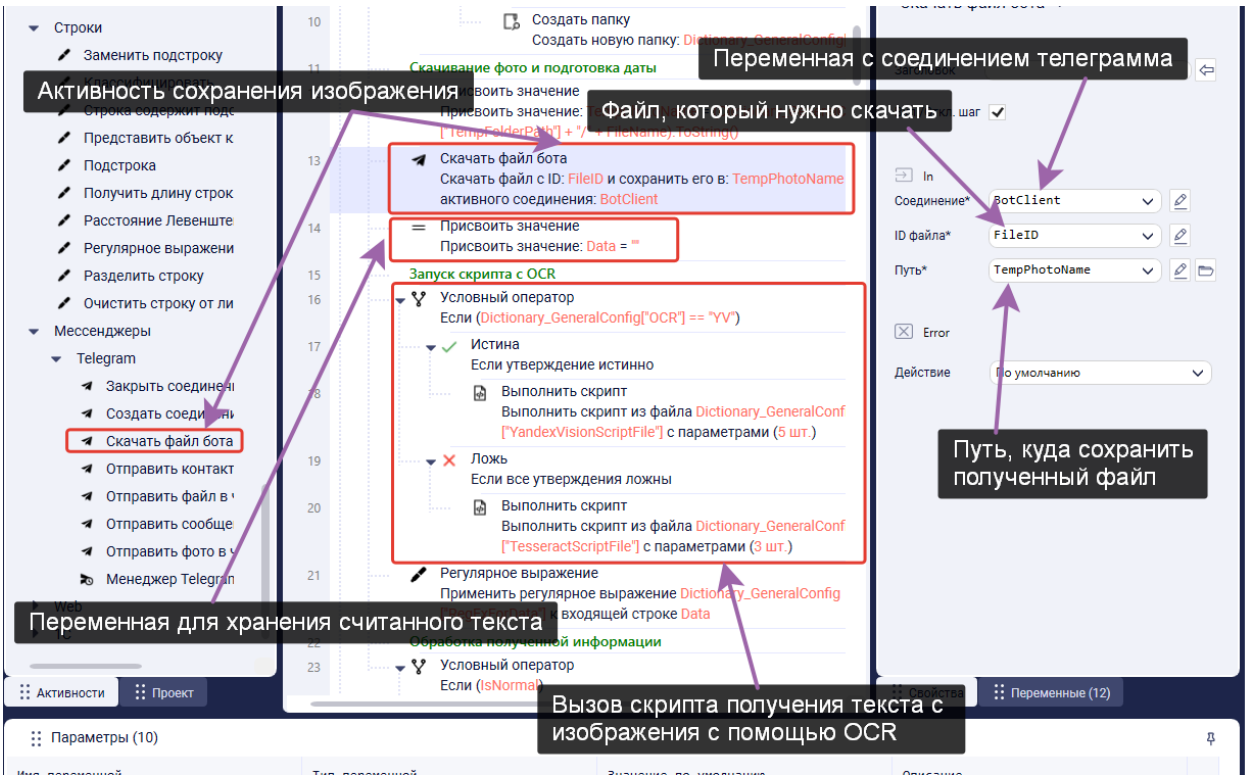


Рисунок 5.2.2. Сохранение и обработка фото

Затем с помощью активности *«Строки/Регулярное выражение»* проверяется применимость регулярного выражения к полученному тексту. Дальше с помощью активности *«Базовые/Условный оператор»*, если регулярное выражение возможно применить считанному тексту, то при помощи активности *«Базовые/Присвоить значение»* и C# функции *System.Text.Regular.Expressions.Regex.Match(Текст, Регулярное_выражение)* регулярное выражение применяется к полученному тексту. Потом с использованием активности *«Коллекции/Словарь/Задать значение для ключа»* и C# функции *Результат_регулярного_выражения.Groups(Названия)* полученные значения заносятся в результирующий словарь (Рисунок 5.2.3).

Также в случае ФИО результаты обрабатываются с помощью C# функции *Строка.Replace(Текст_к_замене, Заменяющий_текст)*, заменяя английские буквы на русские.

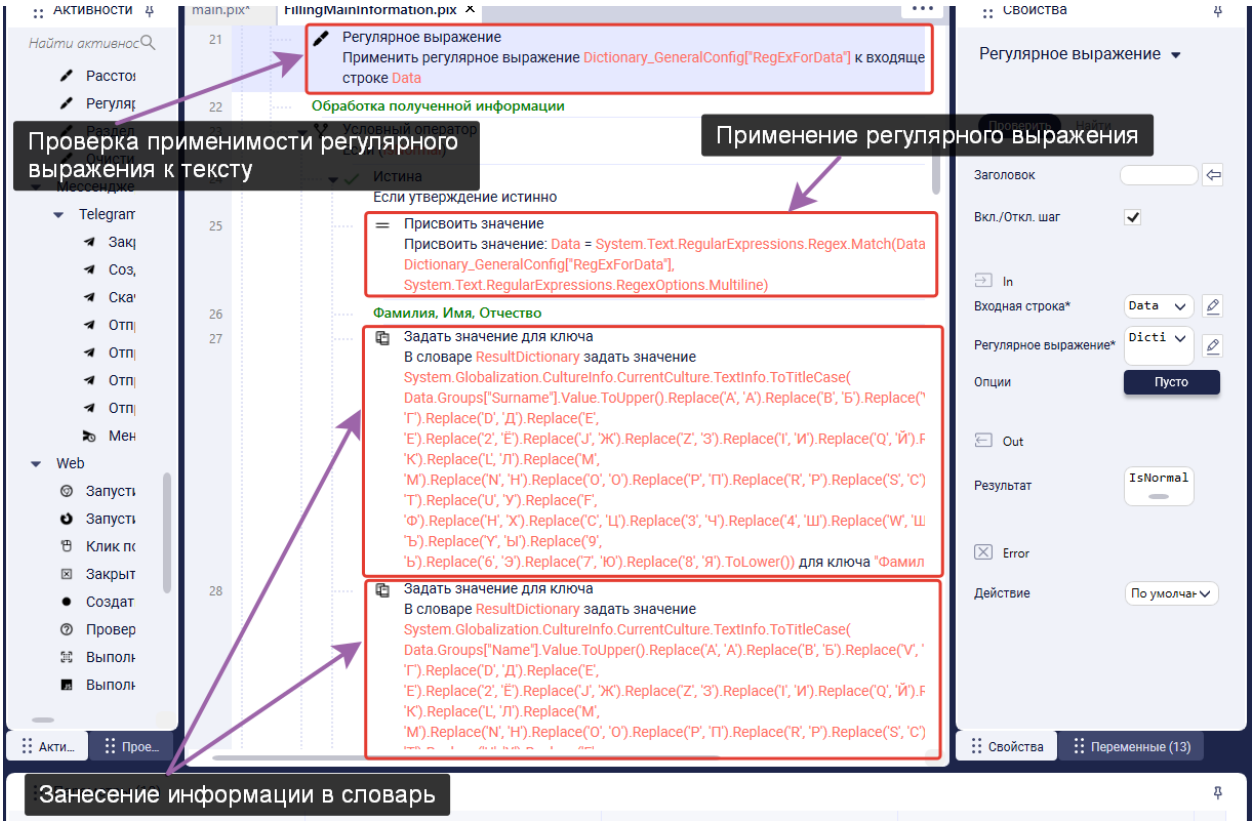


Рисунок 5.2.3. Разбор считанного текста

В случае если к тексту не применимо регулярное выражение, то с помощью активности *«Мессенджеры/Telegram/Отправить сообщение в чат»* отправляется соответствующее сообщение пользователю и происходит возврат к скрипту Main.pix при помощи скрипта *«Базовые/Return»*. (Рисунок 5.2.4)

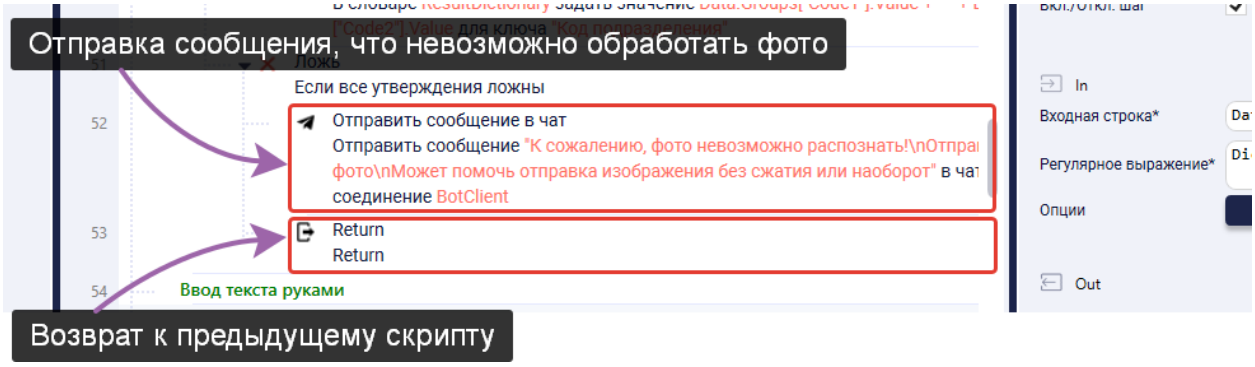


Рисунок 5.2.4. Случай невозможности обработки изображения

В случае же если получен текст вместо фотографии, то с помощью активности *«Базовые/Присвоить значение»* и C# функции *Строка.Split(Разделитель)* создаётся список подстрок с паспортными данными. С помощью активности *«Базовые/Условный оператор»* и C#

функции *Список.Length* список проверяется на соответствие нужному количеству подстрок. Если ответ положительный, то активностью *«Базовые/Циклы/Цикл для каждого»* реализуется цикл по всем подстрокам и внутри цикла с помощью активности *«Базовые/Выполнить»* и C# функции *switch(Переменная) {case Условие_1: …; break; case Условие_2: …; break;…},* которая в зависимости от итерации цикла выдаёт значение ключа, и активности *«Базовые/Задать значение для ключа»* полученные значения заносятся в результирующий словарь с нужным ключом. (Рисунок 5.2.5)

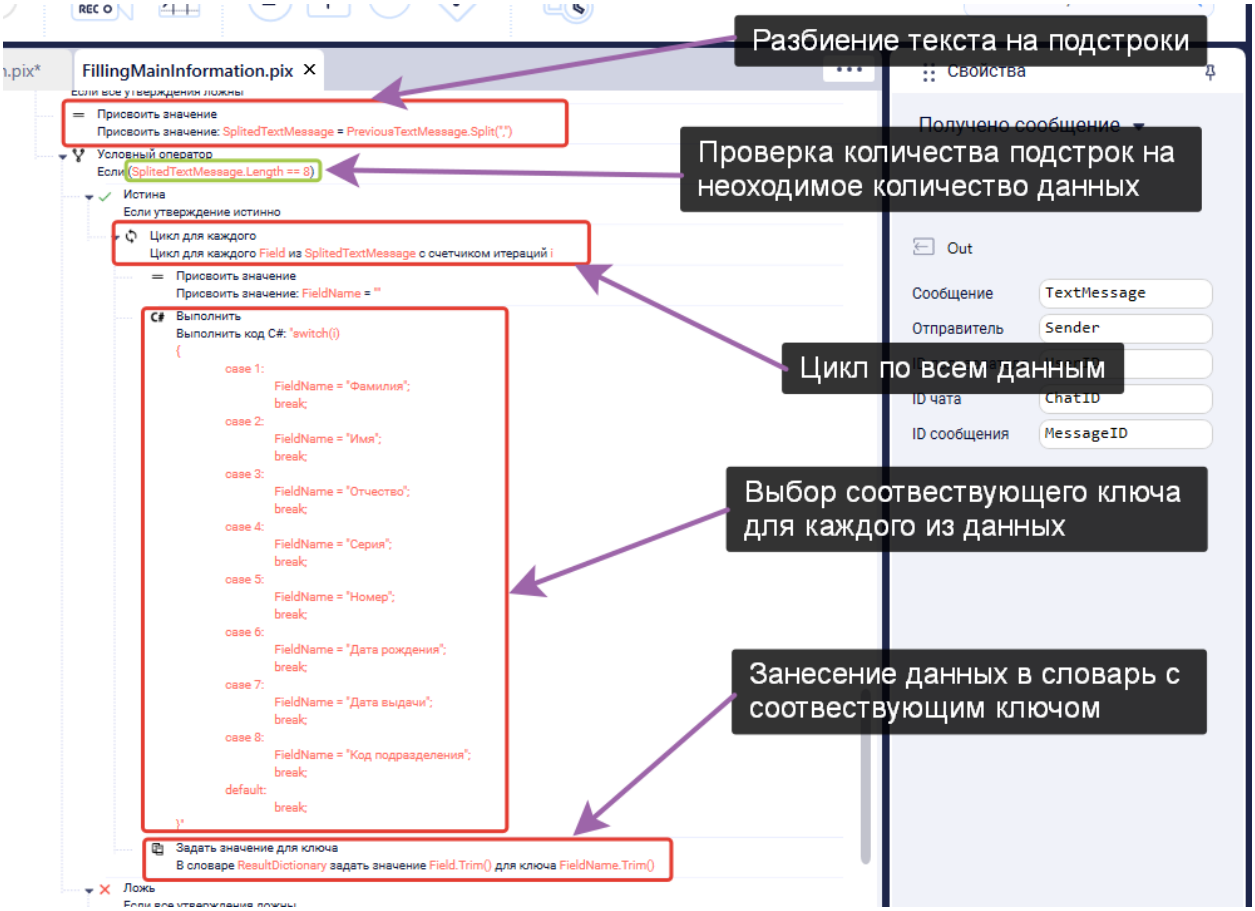


Рисунок 5.2.5. Занесение текстовых данных

В случае если количество подстрок не совпадает с требуемым, то делаются действия аналогичные рисунку 5.2.4.

Следующим шагом с помощью активности *«Базовые/Присвоить значение»* создаётся переменная для вывода информации. После с помощью активностей *«Базовые/Циклы/Цикл для каждого»* и *«Базовые/Присвоить значение»* и C# функции *string.Format(Строка_с_пропуском_значений,*

Значение_1, Значение_2, ...) значения из словаря заносятся в созданную переменную.

После значение переменной выводится в чат с помощью активности *«Мессенджеры/Telegram/Отправить сообщение в чат»* с просьбой проверить данные и предложением их изменить. (Рисунок 5.2.6)

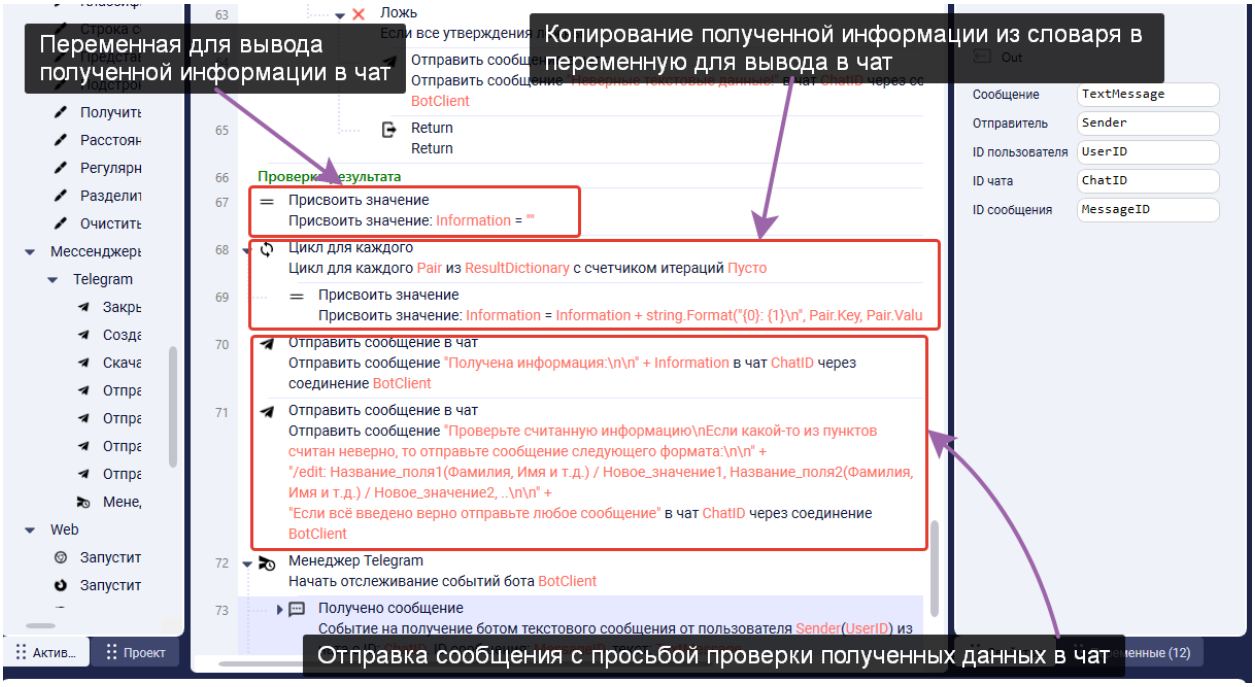


Рисунок 5.2.6. Отправка данных на проверку

Далее запускается бесконечный цикл получения входящих сообщений с помощью активности *«Мессенджеры/Telegram/Менеджер Telegram»*. В случаях, когда получен в ответ файл или контакты другого пользователя, то считается, что полученная информация верна и с помощью активности *«Базовые/Выполнить скрипт»* запускается скрипт *CheckCorrectness.pix*, который проверяет полученную информацию на корректность. Предварительно активностью *«Базовые/Присвоить значение»* создаётся переменная для хранения результатов проверки. Если проверка не была пройдена, то ожидаются исправленные данные, если же проверка прошла успешно активностью *«Базовые/Выполнить скрипт»* запускается скрипт *AdditionInformation.pix*, который требует данные прописки, после выполнения скрипт применяется активность *«Базовые/Return»*, возвращает пользователя к скрипту *Main.pix*. (Рисунок 5.2.7)

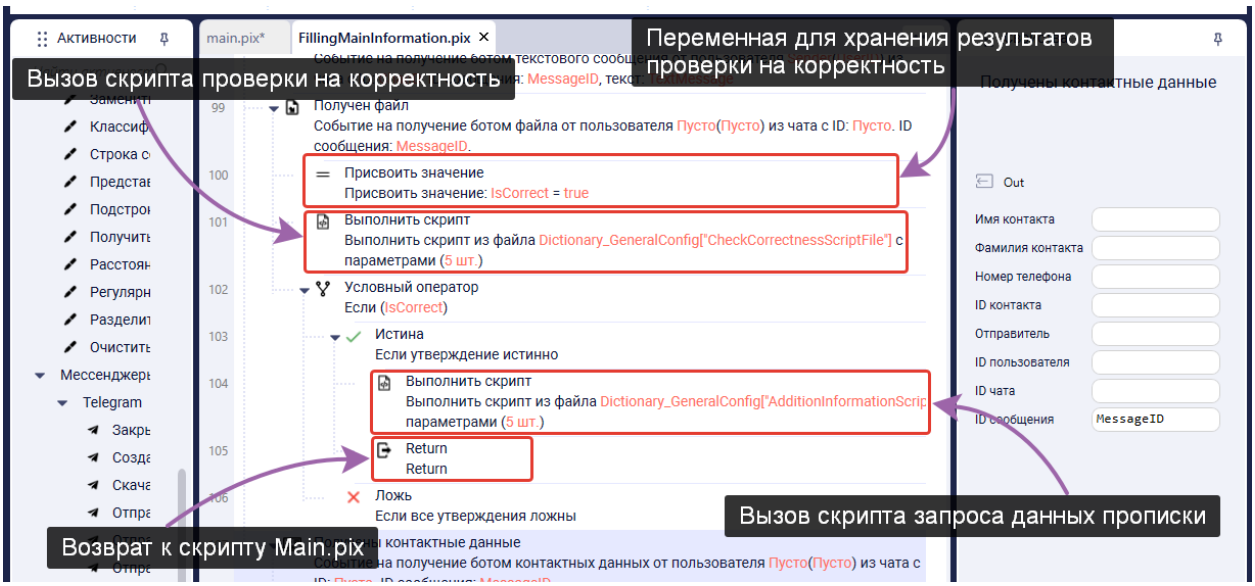


Рисунок 5.2.7. Случай получения файла или контакта

В случае получения запроса на изменение данных, полученный текст обрабатывается и добавляется в словарь аналогично рисунку 5.2.5, а затем происходят действия из рисунка 5.2.7. Если же получен случайный текст, то сразу применяются действия из рисунка 5.2.7.

5.3. Скрипт Tesseract.pix

При вызове скрипта необходимо передать ряд параметров в окне *«Параметры запуска скрипта»*. (Рисунок 5.3.1)

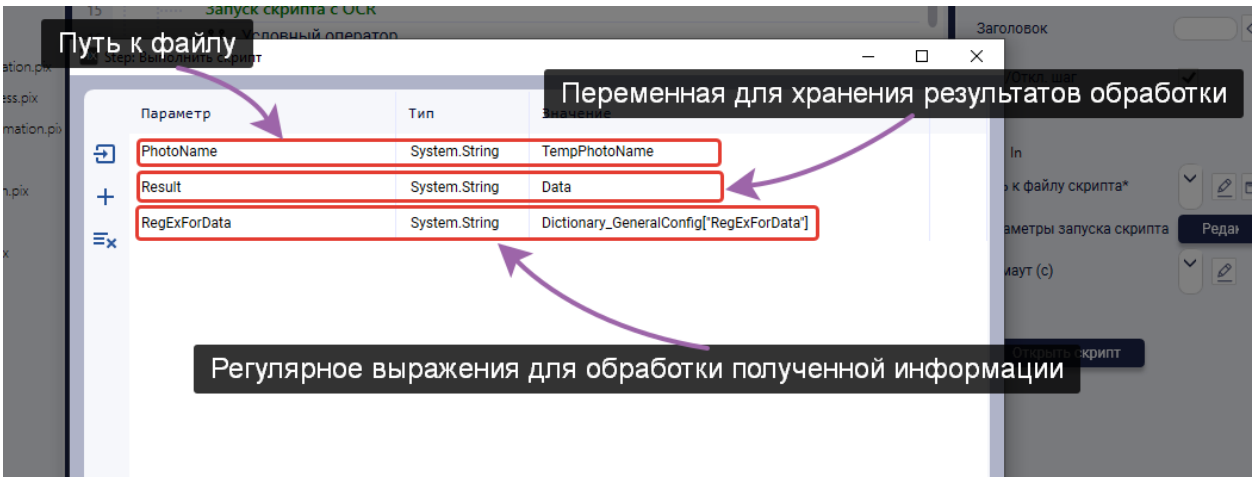


Рисунок 5.3.1. Передача параметров в скрипт

Первым делом скрипт считывает изображение с помощью активности *«Изображения/Получить изображения»*. В свойствах указывается: *Путь к картинке* – путь до изображения, *Изображение* – переменная для хранения изображения.

Затем применяется активность *«OCR/Tesseract»*, в свойствах которой заполняется: *Изображение* – переменная с изображением, *Язык* – язык текста на изображение, в случае для паспорта «eng», *Результат* – переменная, куда будет сохранён текст.

В полученном тексте с помощью активности *«Базовые/Присвоить значение»* и C# функции *Строка.Replace()* удаляются все переносы строк и пробелы. После чего полученная строка проверяется на соответствие регулярному выражению с помощью активности «*Строка/Регулярное выражение»*. (Рисунок 5.3.2)

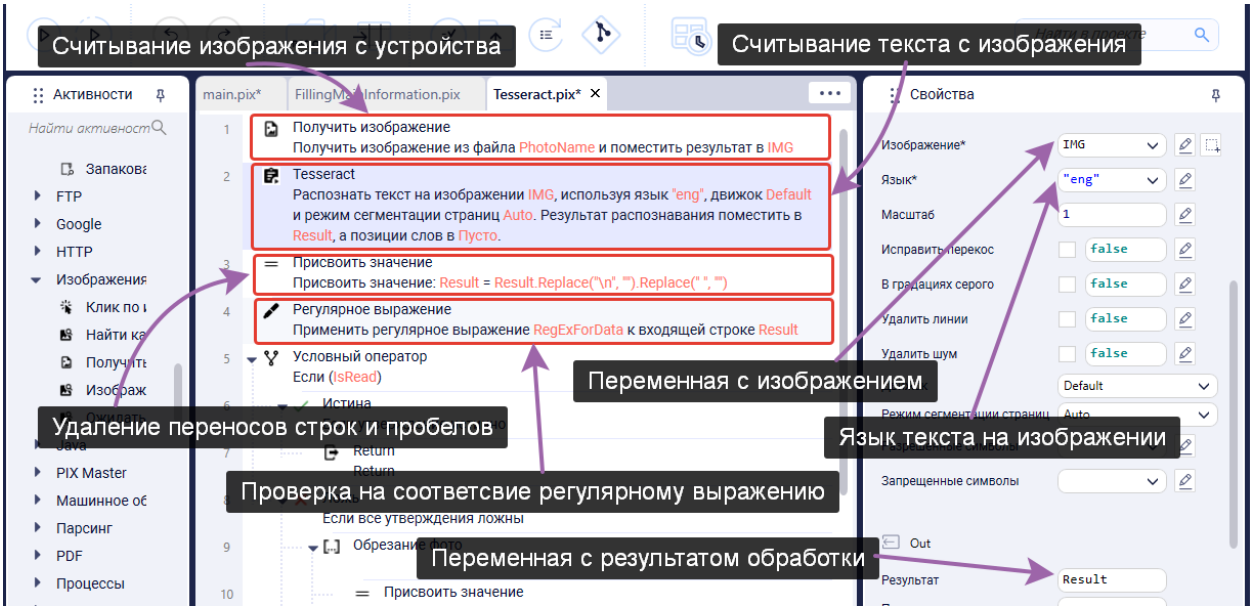


Рисунок 5.3.2. Считывание текста с изображения

Далее если удалось применить регулярное выражение, то с помощью активности «Базовые/Return» происходит возврат к скрипту FillingMainInformation. В ином случае с помощью активности «Базовые/Присвоить значение» и C# функций Изображение.Bitmap и Изображение_в_формате_Bitmap.Clone(new System.Drawing.Rectangle(Точки_прямоугольника_на_изображение), Цветой_формат) изображение обрезается в 8 раз. (Рисунок 5.3.3)

Затем производятся аналогичные действия по обработке полученного изображения и возврат в скрипт FillingMainInformation.

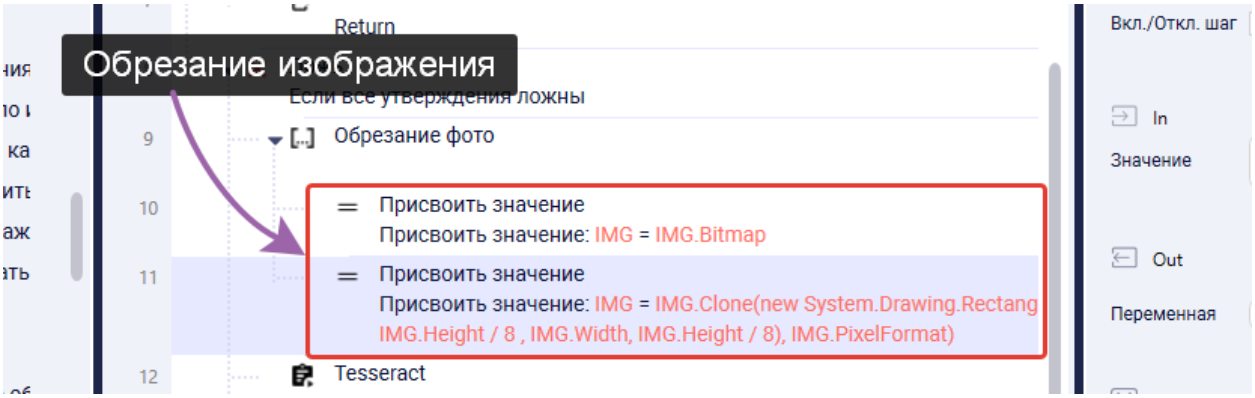


Рисунок 5.3.3. Обрезание фото

5.4. Скрипт CheckCorrectness.pix

При вызове скрипта необходимо передать ряд параметров в окне *«Параметры запуска скрипта»*. (Рисунок 5.4.1)

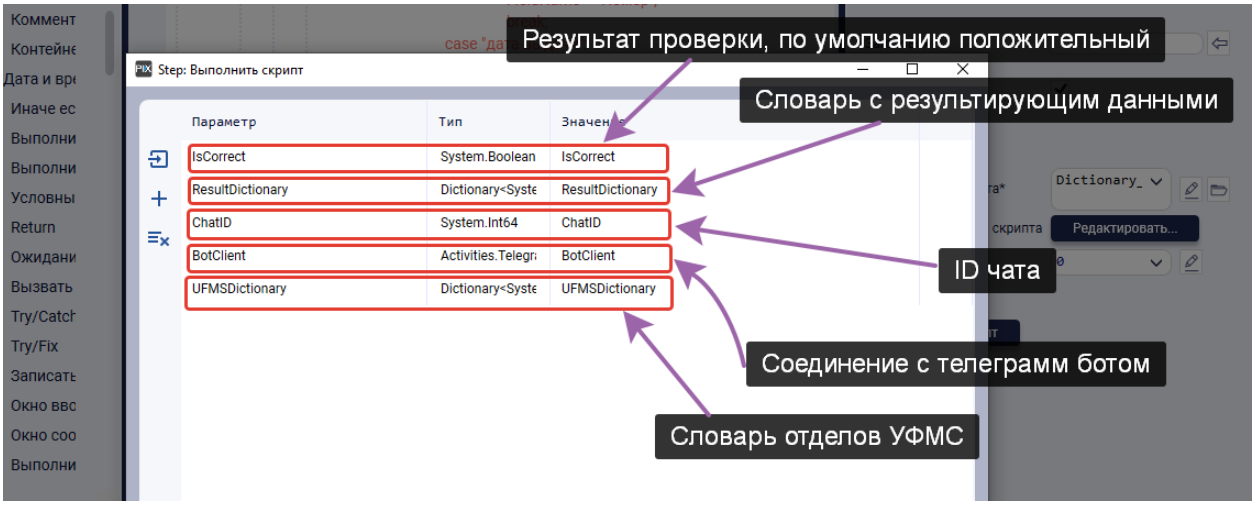


Рисунок 5.4.1. Передача параметров в скрипт

В данном скрипте осуществляется проверка входных данных на корректность. Для этого используется активности *«Базовые/Try/Catch/Finally»* и *«Базовые/Присвоить значение»* и C# функции *Conver.ToInt32(Строка_с_числами)*, *DateTime.ParseExact(Дата_в_виде_строки, Формат_даты)*. В блоке *Try* принимается попытка преобразовать строку, в блоке *Catch* в случае появлении ошибки булева переменная проверки приравнивается отрицательному значению, в блоке *Finally* в случае если возникла ошибка активностью *«Мессенджеры/Telegram/Отправить сообщение в чат»* отправляется соответствующее сообщение о некорректности введённых

данных и применяется активность *«Базовые/Return»*, которая возвращается робота в скрипт *FillingMainInformation*. (Рисунок 5.4.2)

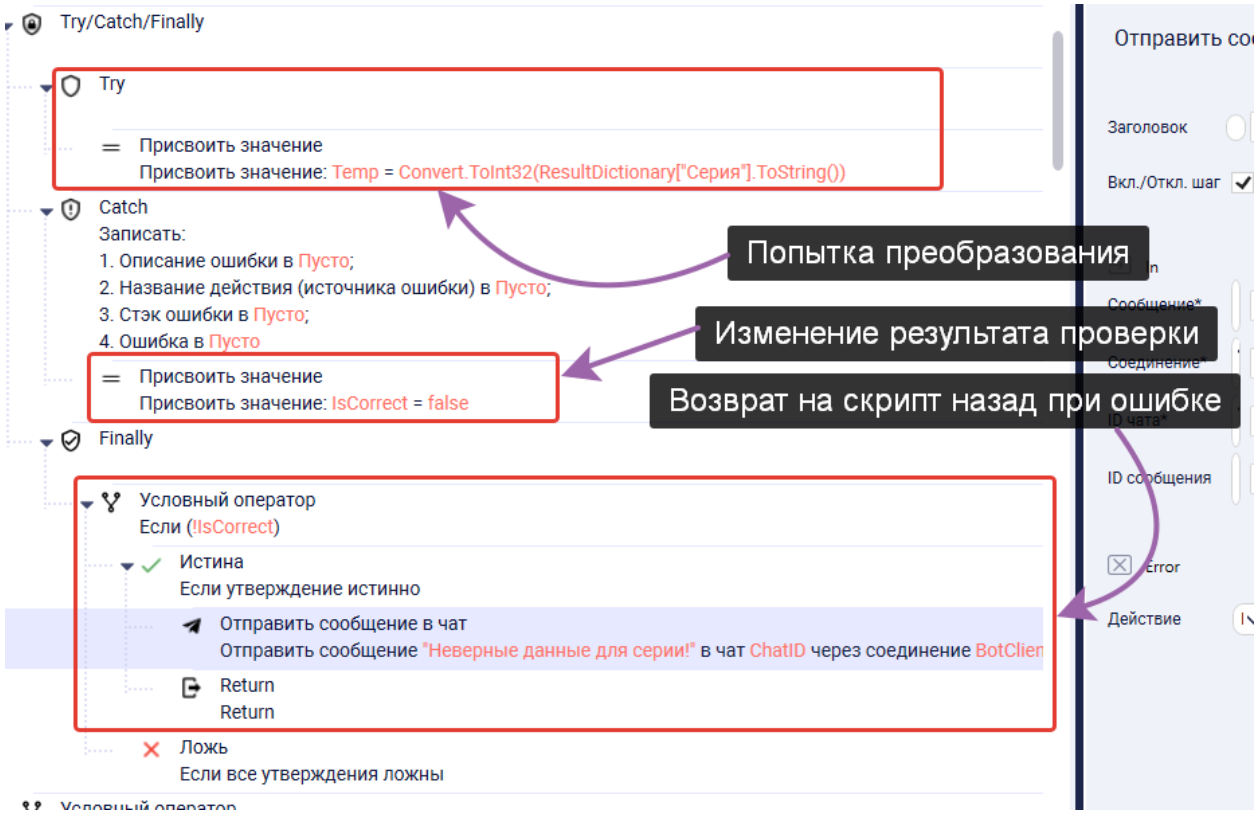


Рисунок 5.4.2. Пример блока Try/Catch/Finally

В некоторых случаях данные проверяются с помощью активности «Базовые/Условный оператор» и C# функции Строка.Length на соотвествие длине.

В случае с кодом подразделения с помощью C# функции Словарь.TryGetValue(Ключ_для_проверки) проверяется его существование с словаре отделов УФМС.

5.5. Скрипт AdditionalInformation.pix

При вызове скрипта необходимо передать ряд параметров в окне *«Параметры запуска скрипта»*. (Рисунок 5.5.1)

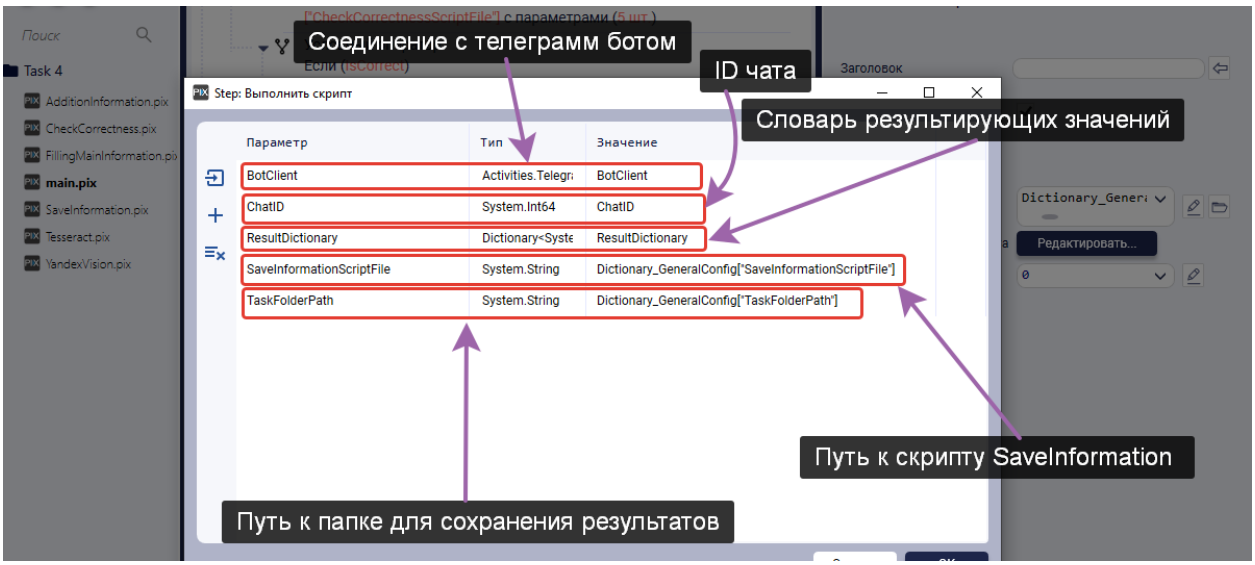


Рисунок 5.5.1. Передача параметров в скрипт

В начале скрипта активностью *Мессенджеры/Telegram/Отправить сообщение в чат»* отправляется сообщение в чат с просьбой заполнить данные прописки и пример их написания.

Затем аналогично рисунку 5.2.5 и 5.2.4 (только в данном случае без активности *«Базовые/Return»*) активностью *«Мессенджеры/Telegram/Менеджер Telegram»* ожидается получение данных прописки. В случае получения неверной информации активностью *«Мессенджеры/Telegram/Отправить сообщение в чат»* повторяется сообщение отправленное в начале скрипта. При получение нужной информации она заносится в результирующий словарь и вызывает скрипт *SaveInformation.pix* активностью *«Базовые/Выполнить скрипт»*.

5.6. Скрипт SaveInformation.pix

При вызове скрипта необходимо передать ряд параметров в окне *«Параметры запуска скрипта»*. (Рисунок 5.6.1)

Рисунок 5.6.1. Передача параметров скрипта

Сначала активностью *«Мессенджеры/Telegram/Отправить сообщение в чат»* пользователю отправляется благодарность за заполнение анкеты. А также выводится записанная информация аналогично рисунку 5.2.6.

Затем с помощью активности *«Файлы/Путь существует?»* проверяется наличие папки для записи результатов, и в случае её отсутствия она создаётся с помощью активности *«Файлы/Создать папку»*.

Далее активностью *«Получить пути к файлам/каталогам»* собираются пути ко всем файлам в папке и с помощью активности *«Базовые/Присвоить значения»* формируется имя файла с текущими результатами на основе количества файлов в папке и ID пользователя. (Рисунок 5.6.2)

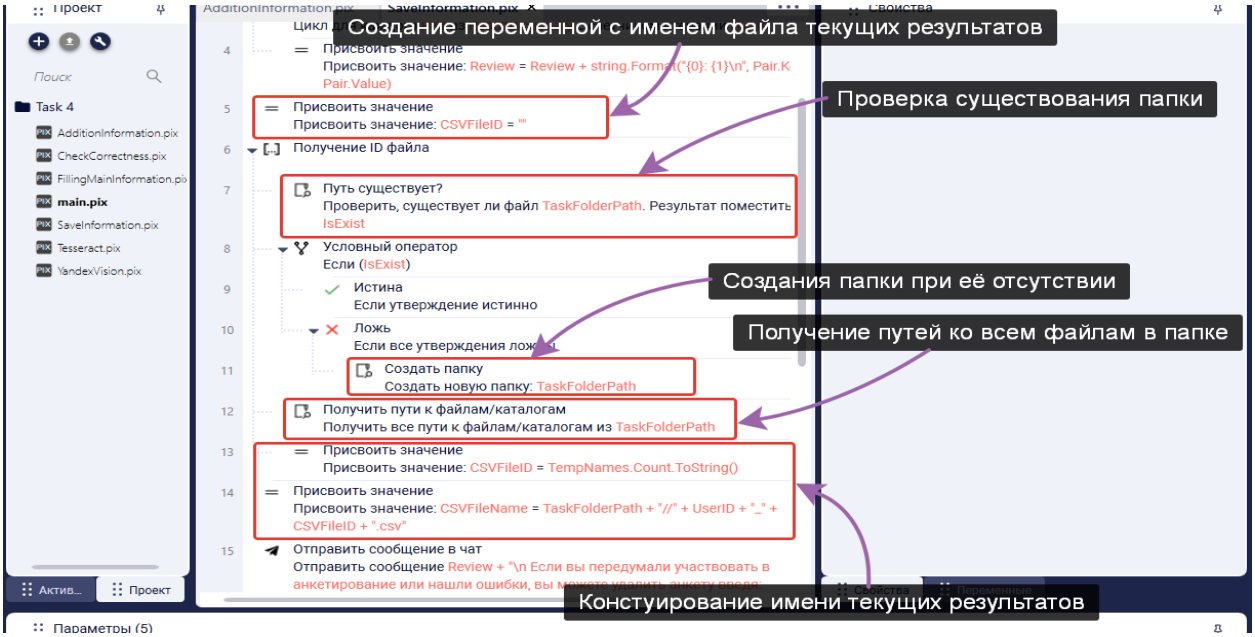


Рисунок 5.6.2. Создание результирующего файла

Затем с помощью активности *«Коллекции/Таблица/Создать таблицу»* создаётся таблица для записи результатов в CSV-файл. В свойствах активности необходимо в свойстве *«Структура таблицы»* создать два столбца. Потом с помощью активностей *«Базовые/Циклы/Цикл для каждого»*, *«Коллекции/Словарь/Создать словарь»*, *«Коллекции/Словарь/Задать значение для ключа», «Коллекции/Таблица/Добавить строку»* данные из результирующего словаря построчно переносятся в созданную таблицу. На каждой итерации цикла создаётся временный словарь, куда заносится ключ и значение из результирующего словаря с ключами названия соответствующих столбцов. В конце итерации временный словарь записывается в созданную таблицу.

После цикла таблица активностью *«CSV/Записать в CSV»* записывает в CSV-файл. В свойствах активности указывается: *Таблица* – созданная таблица, *Путь CSV-файлу* – переменная с созданным раннее путём к результирующему файлу, *Добавлять заголовки* – переносить ли названия столбцов в CSV файл, *Разделитель* – символ означающий разделение данных.

В конце используется активность «Базовые/Return» для возврата в скрипт AdditionInformation.pix. (Рисунок 5.6.3)

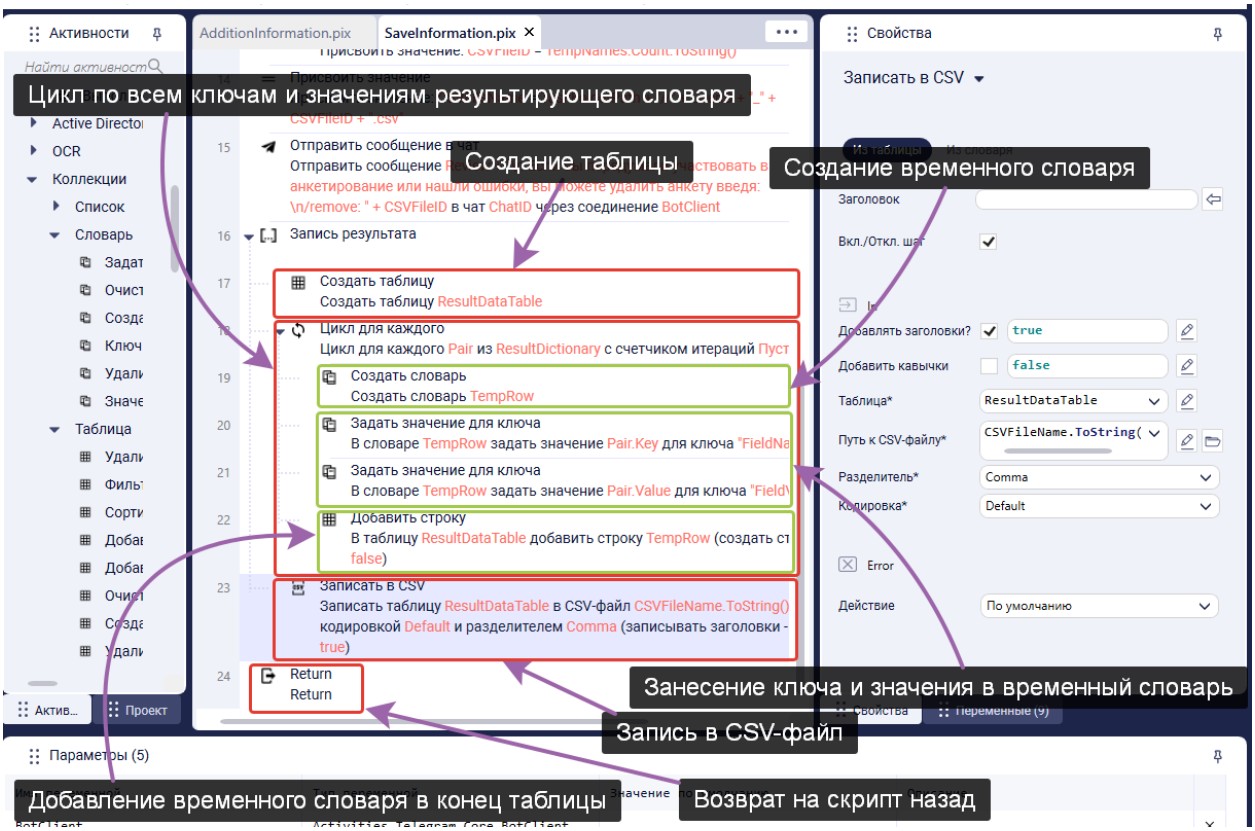


Рисунок 5.6.3. Сохранение результатов

6. Проект по обработке сохранённой информации

Первым делом производятся действия аналогичные рисункам 5.1.1, 5.1.2, 5.1.3, 5.1.4, а именно считывается вся информация из конфига, считывается таблица с отделами УФМС, считывается из Windows Credentials данные для доступа к почте и базе данных.

Затем проверяется существование необходимых папок аналогично рисункам 5.2.1, 5.6.3.

Следующим шагом с помощью активности *«Файлы/Получить пути к файлам/каталогам»* получаются пути ко всем файлам с полученными данными. А также с помощью активности *«SQL/Создать подключение к БД»* осуществляется подключение к базе данных. В свойствах заполняется: *Строка соединения с БД*, *Провайдер* – в зависимости от используемой базы данных имеют различное значение, примеры строк легко ищутся в интернете и в документации к активности, желательно заранее сохранить данные в Windows Credentials, *Подключение* – переменная с подключением к базе данных. А после с помощью активности *«SQL/Выполнить команду»*

отправляется SQL запрос *«CREATE TABLE IF EXISTS Название (Имя_столбца_1 тип_данных_1, имя_столбца_2, тип_данных_2, ...)»* на создание таблицы для записи результатов, если она не существует. (Рисунок 6.1)

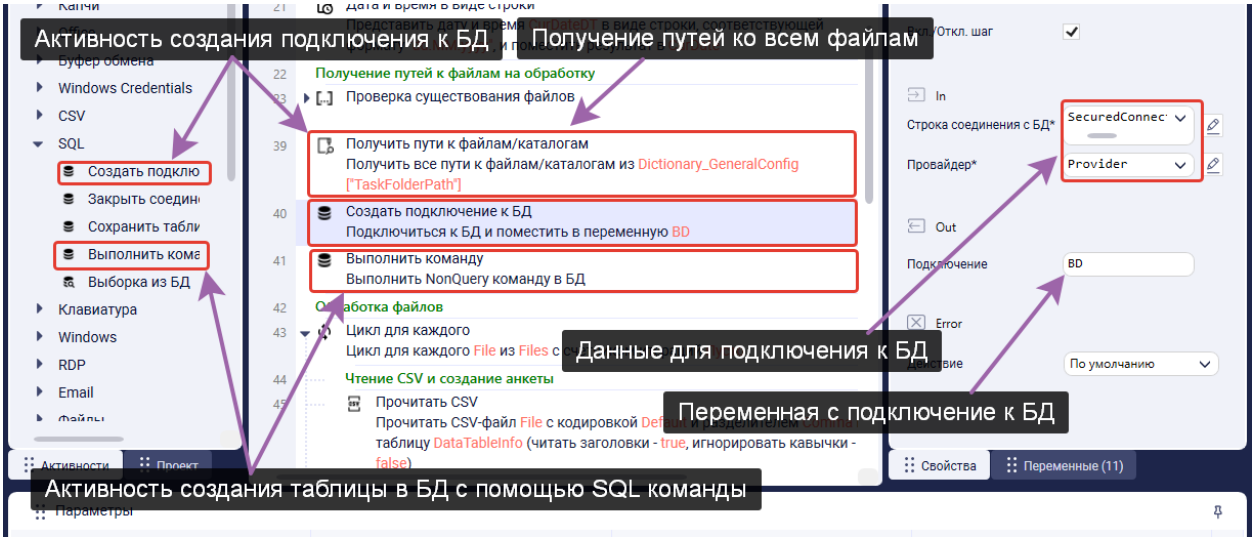


Рисунок 6.1. Подключение к БД

Затем запускается цикл по всем путям активностью *«Базовые/Циклы/Цикл для каждого»*.

Внутри цикла активностью *«CSV/Прочитать CSV»* считываются полученные данные из анкеты и записываются во временную таблицу. Затем активностью *«Базовые/Присвоить значение»* и C# функции *Таблица.AsEnumerable().ToDictionary* таблица преобразовывается в словарь. А также той же активностью создаётся имя результирующего файла анкеты и активностью *«Файлы/Копировать файл/папку»* файл шаблона копируется в нужной место. (Рисунок 6.2)

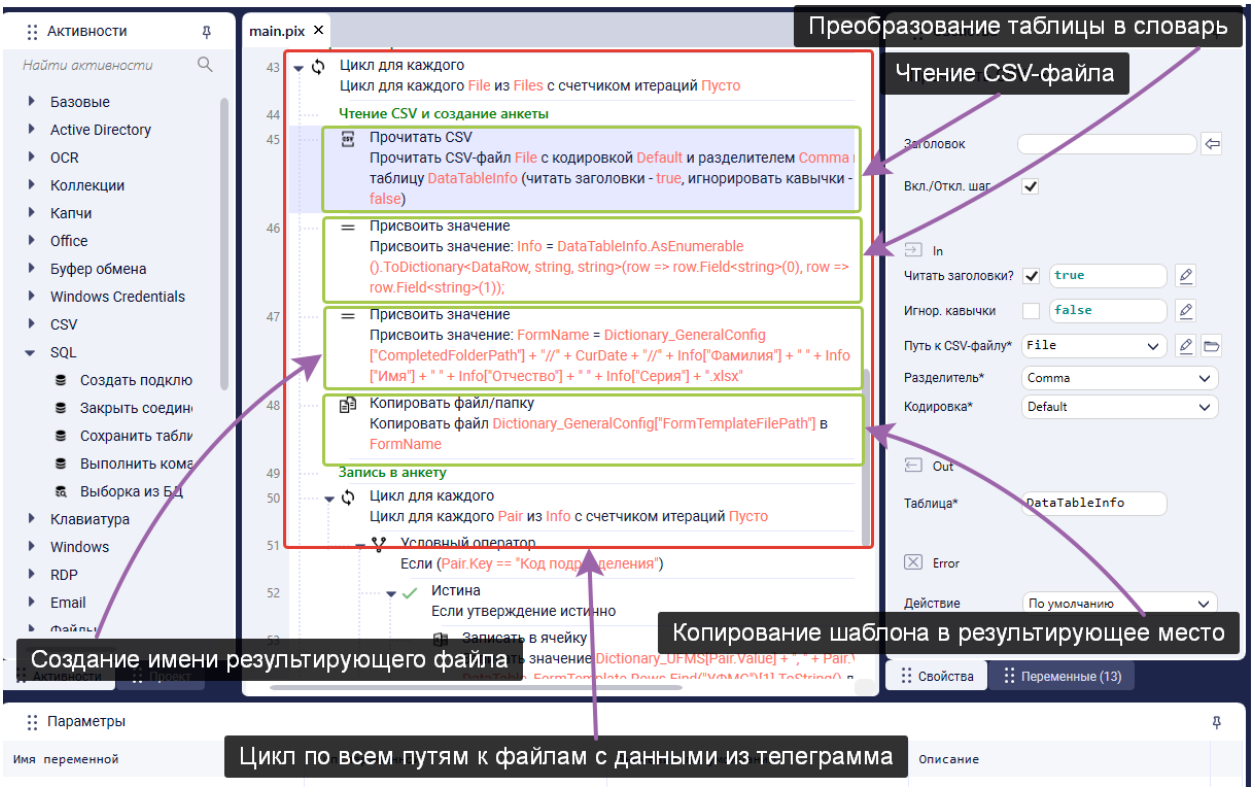


Рисунок 6.2. Создание результирующего файла

Далее активностью *«Базовые/Циклы/Цикл для каждого»* запускается цикл по всем парам ключ/значение в словаре. И с помощью активность *«Office/Excel/Записать в ячейку»* данные записываются в соответствующее место в результирующем файле. В случае ключа код подразделения также записывается ещё название соответствующего отдела УФМС. (Рисунок 6.3)

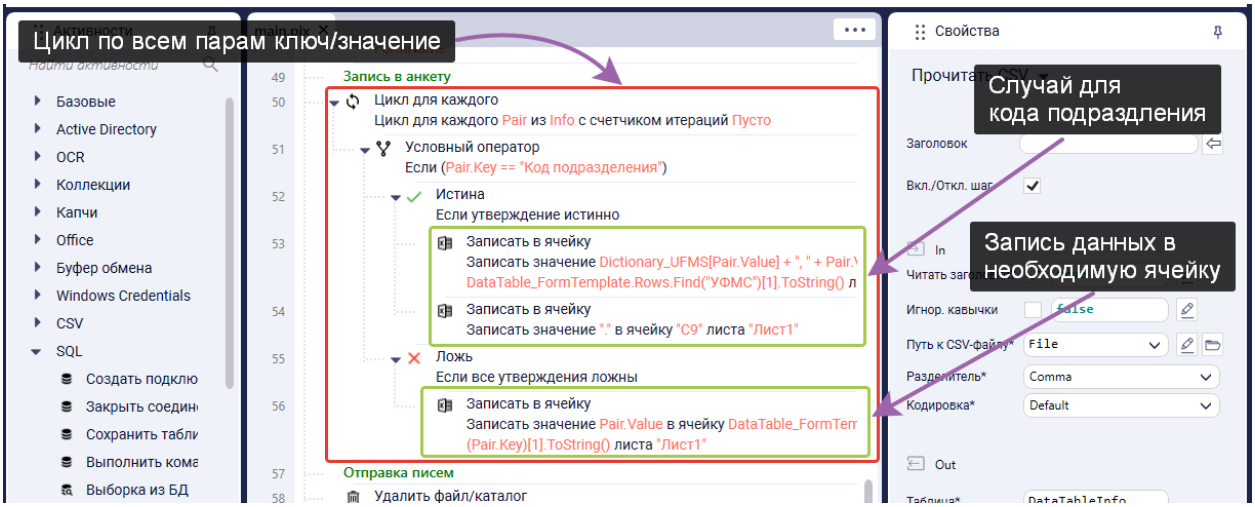


Рисунок 6.3. Запись в результирующий файл

После активностью *«Файлы/Удалить файл/каталог»* удаляется CSV-файл.

Затем активностью *«Email/Отправить Email (SMTP)»* получившаяся анкета отправляется по почте. В свойствах активности заполняются: *Тема* – тема письма, *Порт* – порт SMTP сервера, *Сервер* – сервер SMTP, *Адрес электронной почты* – логин почты, с которой отправляется письмо, *Безопасный строковый пароль* – пароль от почты, с которой отправляется письмо, *Кому* – почта на которую необходимо отправить письмо, *Вложения* – путь к файлу, который необходимо отправить.

Следующим шагом активностью *«SQL/Выполнить команду»* отправляется SQL-запрос *«INSERT INTO Название_таблицы VALUES (Значения)»* к базе данных на добавление полученных данных.

Когда цикл обработает все файлы необходимо активностью *«SQL/Закрыть соединение с БД»* закрыть подключение к БД. (Рисунок 6.4)

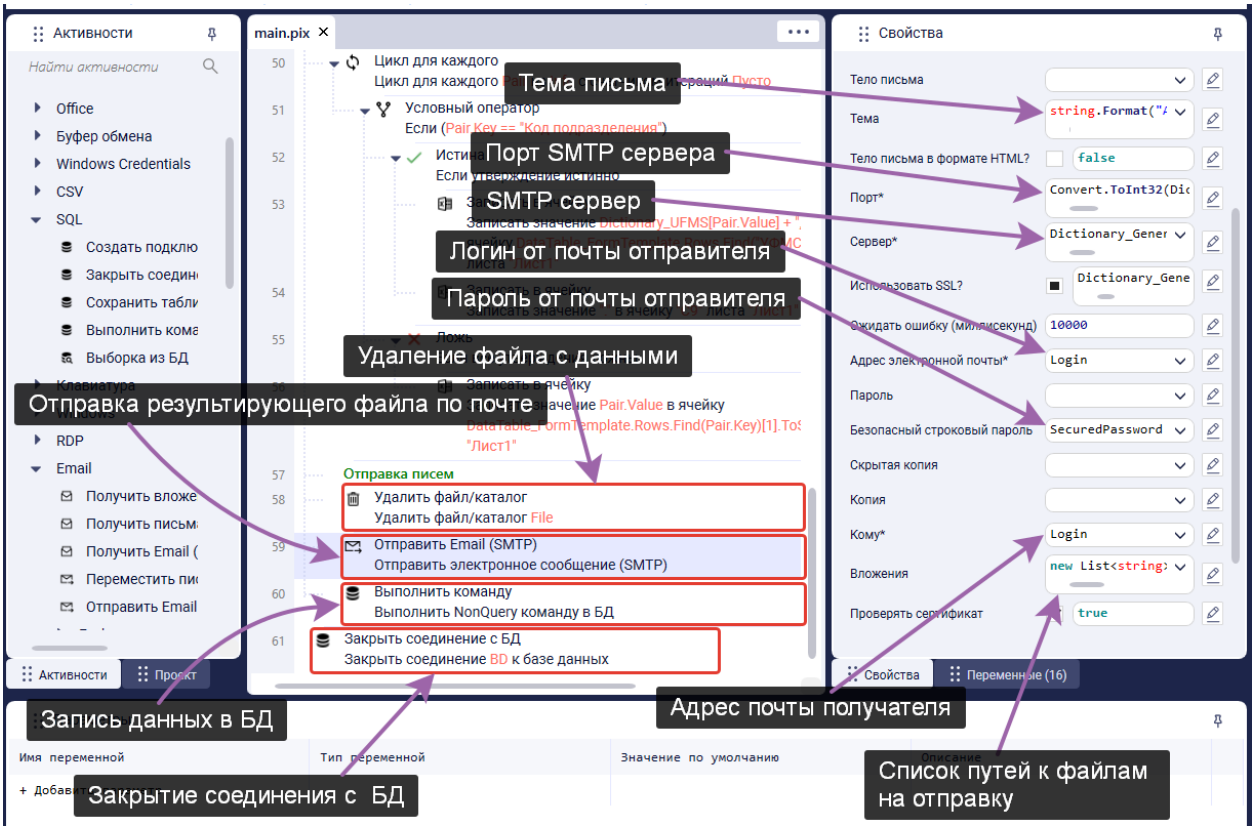


Рисунок 6.4. Отправка файла по почте и занесение данных в БД

Литература

1. https://regex101.com/
2. http://2sql.ru/
3. https://core.telegram.org/
4. https://knowledgebase.pixrpa.ru/actions/Excel/ReadRange
5. https://knowledgebase.pixrpa.ru/actions/Credentials/GetCredentials
6. https://knowledgebase.pixrpa.ru/actions/Base/DateTimeToString
7. https://knowledgebase.pixrpa.ru/actions/Base/GetDateTime
8. https://knowledgebase.pixrpa.ru/actions/Files/CreateFolder
9. https://knowledgebase.pixrpa.ru/actions/Files/PathExist
10. https://knowledgebase.pixrpa.ru/actions/Files/GetListFilesOrCatalogs
11. https://knowledgebase.pixrpa.ru/actions/DataBaseSQL/CreateConnection
12. https://knowledgebase.pixrpa.ru/actions/CSV/ReadCSV
13. https://knowledgebase.pixrpa.ru/actions/Base/Assign
14. https://knowledgebase.pixrpa.ru/actions/Files/CopyFileCatalog
15. https://knowledgebase.pixrpa.ru/actions/DataBaseSQL/RemoveConnection
16. https://knowledgebase.pixrpa.ru/actions/DataBaseSQL/SqlExecuteNonQuery
17. https://knowledgebase.pixrpa.ru/actions/Email/SendSMTPMailMessage
18. https://knowledgebase.pixrpa.ru/actions/Files/DeleteFile
19. https://knowledgebase.pixrpa.ru/actions/Base/LoopForEach
20. https://knowledgebase.pixrpa.ru/actions/Base/If
21. https://knowledgebase.pixrpa.ru/actions/Excel/WriteCell
22. https://knowledgebase.pixrpa.ru/actions/Telegram/CreateConnection
23. https://knowledgebase.pixrpa.ru/actions/Telegram/TelegramManager

24. https://knowledgebase.pixrpa.ru/actions/Base/TryCatch
25. https://knowledgebase.pixrpa.ru/actions/Telegram/SendMessage
26. https://knowledgebase.pixrpa.ru/actions/Strings/RegEx
27. https://knowledgebase.pixrpa.ru/actions/Base/ExecuteScript
28. https://knowledgebase.pixrpa.ru/actions/Telegram/SendPhoto
29. https://knowledgebase.pixrpa.ru/actions/Base/Return
30. https://knowledgebase.pixrpa.ru/actions/Tesseract/TesseractOCR
31. https://knowledgebase.pixrpa.ru/actions/Image/GetImage
32. http://knowledgebase.pixrpa.ru/actions/base/executescript
33. http://knowledgebase.pixrpa.ru/actions/base/executecscode
34. https://learn.microsoft.com/ru-ru/dotnet/api/system.security.securestring?cid=kerryherger&view=net-7.0
35. https://learn.microsoft.com/ru-ru/dotnet/api/system.data.datatable?view=net-7.0
36. https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.dictionary-2?view=net-5.0
37. https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.image?view=windowsdesktop-7.0
38. https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.bitmap?view=windowsdesktop-7.0
39. https://learn.microsoft.com/ru-RU/dotnet/api/system.datetime?view=net-5.0
40. https://learn.microsoft.com/en-us/dotnet/api/system.string?view=net-7.0
41. https://learn.microsoft.com/ru-ru/dotnet/api/system.runtime.interopservices.marshal.ptrtostringuni?view=net-7.0