Создание робота по распознаванию текста с фотографий

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

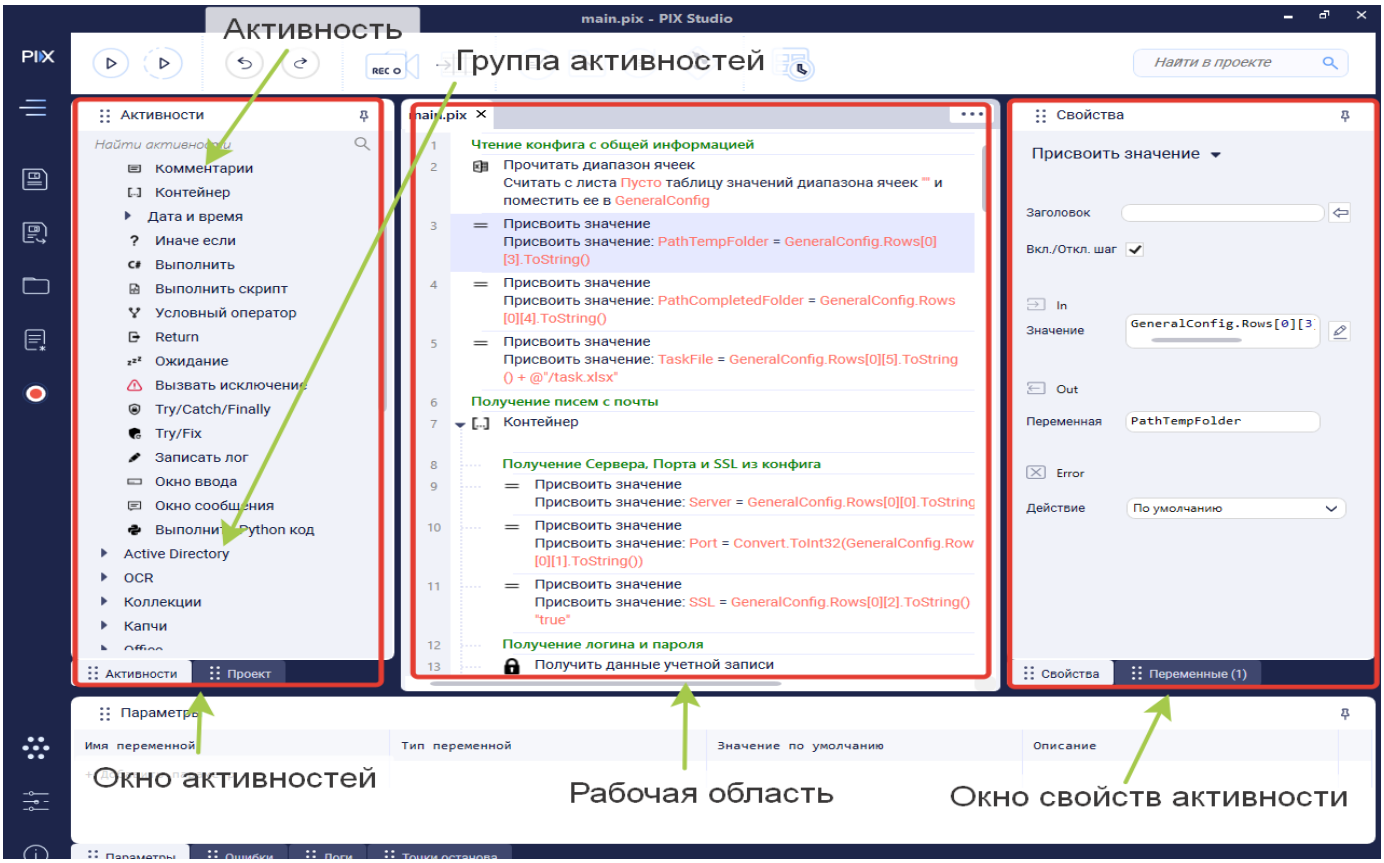


Рисунок 1.1 Окно редактора PIX Studio

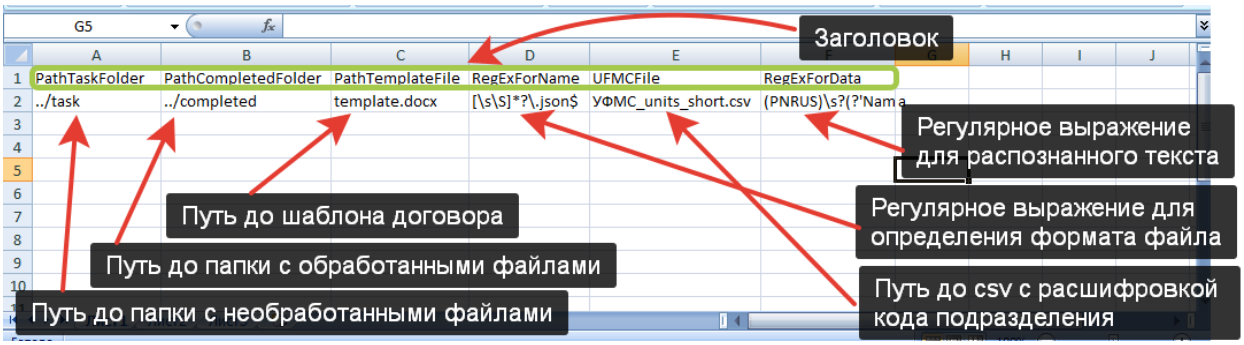
2. Постановка задачи

Робот должен проверить папку с необработанными файлами. Файл может быть либо изображение, либо JSON файл. В случае наличия необработанных файлов, робот должен считать текст из файлов и заполнить необходимые поля шаблона договора считанными данными.

3. Предварительные действия

Прежде чем приступить к созданию робота, желательно выполнить ряд подготовительных действий. 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.



3.1. Пример конфига для робота.

Помимо конфига необходимо убедится в наличие шаблона договора в формате docx. 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.

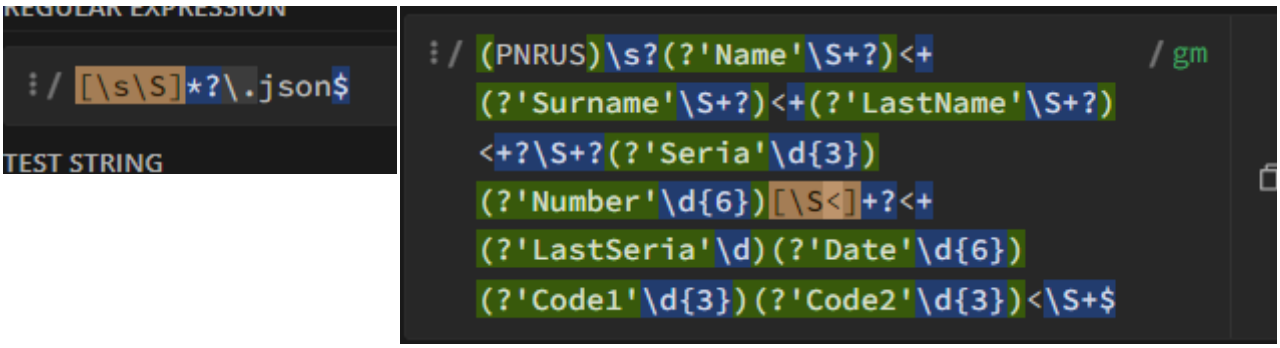


Рисунок 3.3 Пример регулярных выражений для распознавания формата файла (слева) и обработки распознанного текста (справа)

4. Работа с PIX Studio

Теперь необходимо запустить PIX Studio и создать новый 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.

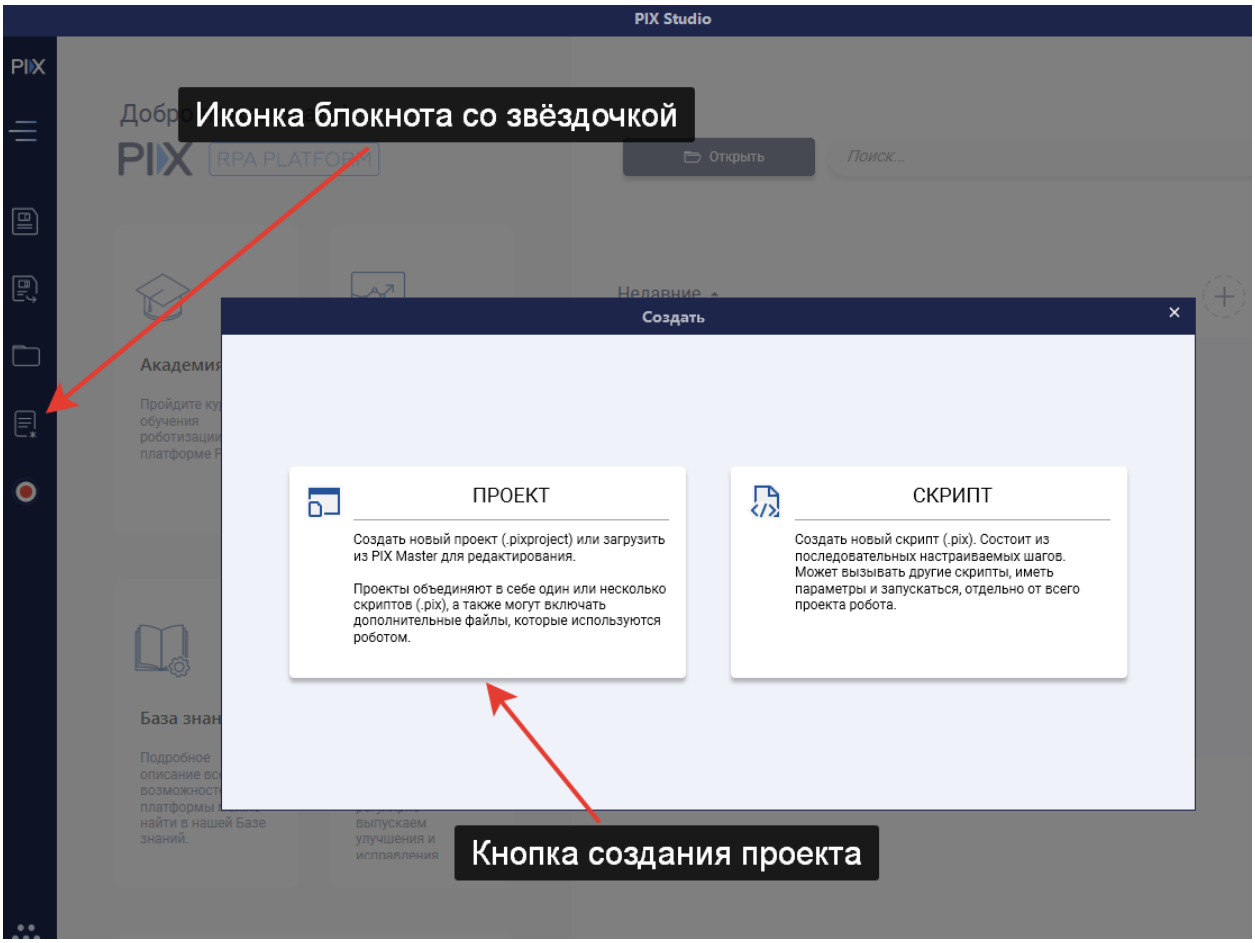


Рисунок 4.1. Создание проекта

4.1. Чтение конфига

В первую очередь необходимо прочитать конфиг. Для этого необходимо использовать активность *«Office/Excel/Прочитать диапазон ячеек»*. В окне свойств данной активности заполняется поля (рисунок 4.1.1):

- *Путь к файлу* – строка, содержащая полный или относительный путь до файла с конфигом.
- *Диапозон* – строка, которая указывает диапазон ячеек, которые необходимо считать, если необходимо прочитать все непустые ячейки с листа используется строка «"»».
- *Таблица* – Имя переменной, в которую будет помещён результат.

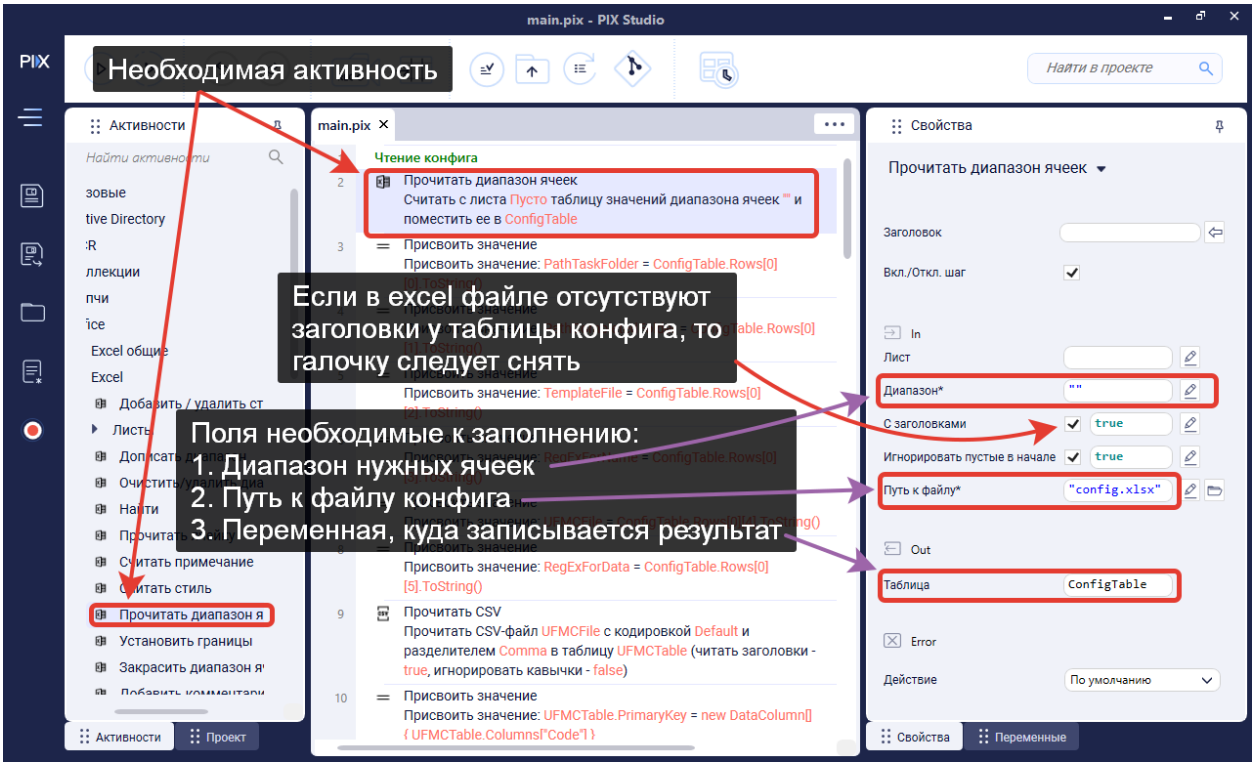


Рисунок 4.1.1. Считывание конфига.

Затем с помощью активностей *«Базовые/Присвоить значение»* данные из таблицы конфига помещаются в отдельные переменные. Для доступа к соответствующей ячейке таблицы используется следующие C# выражение: *«Переменная_с_таблицей.Rows[Индекс_Строки][Индекс_Столбца].ToString()»*. Причём нумерация строк и столбцов начинается с 0.

В окне свойств данной активности заполняются поля (Рисунок 4.1.2):

- *Значение* – значение, которое необходимо сохранить в переменной.

- *Переменная* – имя переменной, куда необходимо сохранить результат.

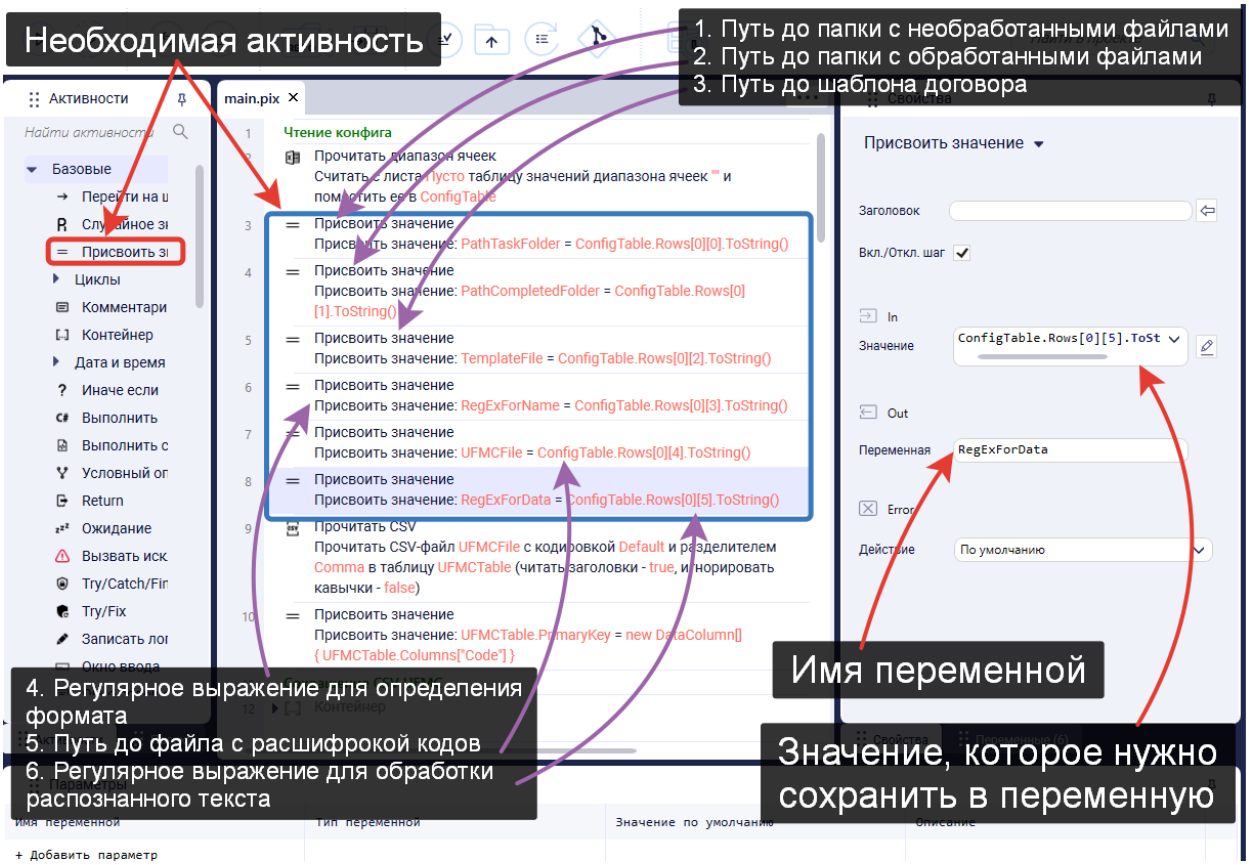


Рисунок 4.1.2. Сохранение значений в отдельные переменные.

Затем считываются данные из csv файла и заносятся в таблицу с помощью активности *«CSV/Прочитать CSV»*. В свойствах активности необходимо заполнить:

- *Читать заголовки?* – присутствуют ли заголовки в csv файле. Если присутствуют, то значение должно быть true.
- *Путь к CSV* – полный или относительный путь к csv файлу.
- *Разделитель* – разделитель, который используется в csv файле. По умолчанию comma (запятая).
- *Таблица* – переменная для хранения результирующей таблицы.

Помимо этого, в результирующей таблице необходимо задать первичный ключ, по которому будет осуществляться поиск в дальнейшем. Для этого используется активность «Базовые/Присвоить значение», которой реализуется следующий C# код:

Название_таблицы.PrimaryKey = new DataColumn[] {Название_таблицы.Column[“Название_столбца_первичного_ключа”]}

Рисунок 4.1.3. Чтение CSV.

4.2. Получение путей файлов и нужных дат

Затем с помощью активности *«Файлы/Получить пути к файлам/каталогам»* получаются пути к необработанным файлам. В свойствах активности указывается: *Путь* – путь до папки, в которой хранятся необработанные файлы, *Список* – переменная, в которую будут сохранены пути ко всем необработанным файлам.

Также необходимо с помощью активности *«Базовые/Дата и время/Получить дату и время»* получить текущую дату. В свойствах активности заполняется: *Результат* – переменная, в которой хранится текущая дата и время.

Помимо этого с помощью активностей *«Базовые/Присвоить значение»*, *«Базовые/Циклы/Цикл пока…»* и C# функции *Переменная.AddDays(Количество_дней)* получается первое число следующего месяца. (Рисунок 4.2.1).

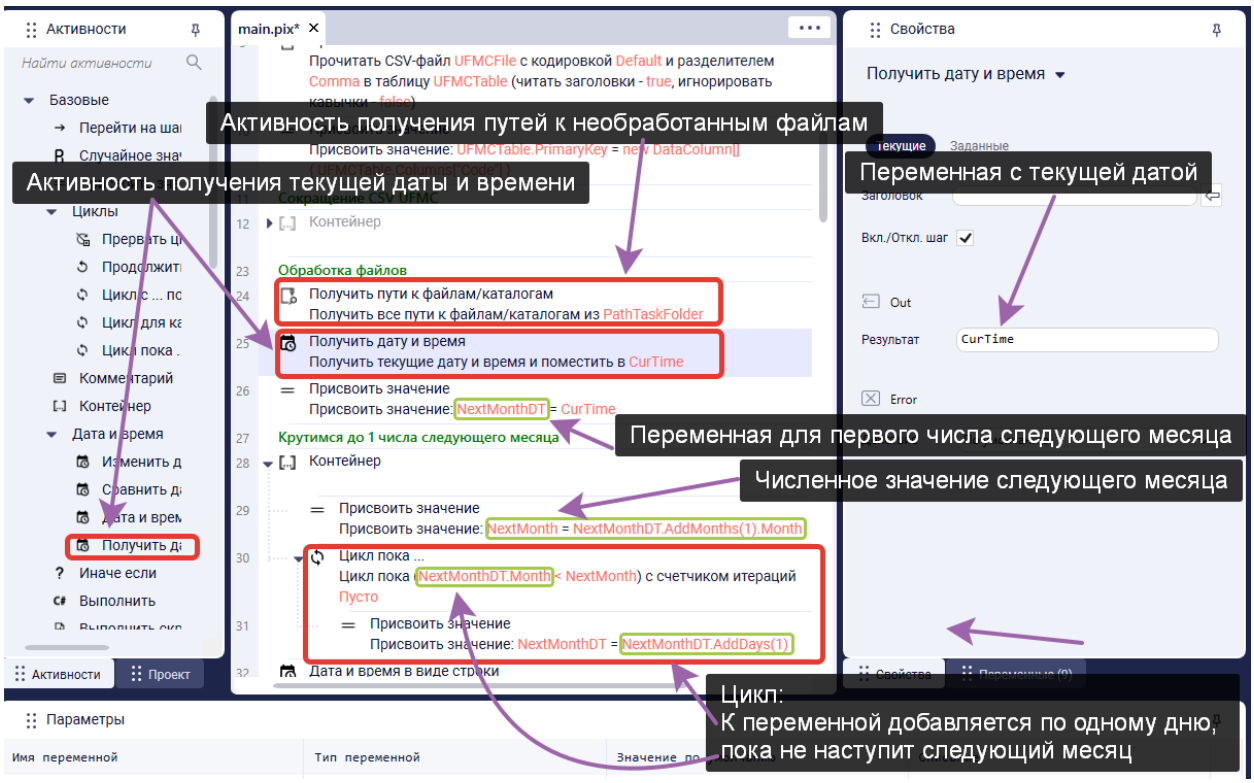
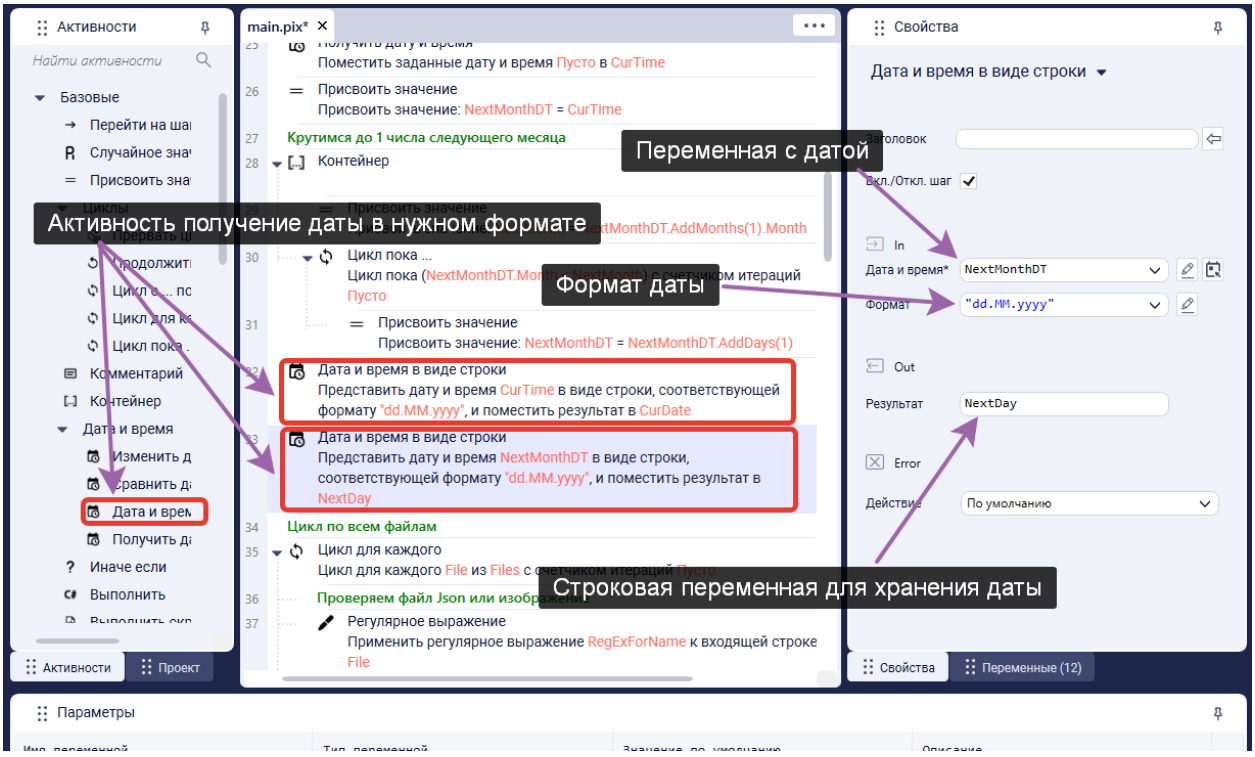


Рисунок 4.2.1. Получение нужных дат и путей к файлам

Затем с помощью активности *«Базовые/Дата и время/Дата и время в виде строки»* полученные даты приводятся в строковый тип с нужным форматом записи. В свойствах активности указывается: *Дата и время* – переменная, в которой хранится дата, *Формат* – формат даты, *Результат*.



4.2.2. Перевод даты в строковый формат

4.3 Получение и обработка текста из файлов

Затем с помощью активности *«Базовые/Циклы/Цикл для каждого»* запускается цикл по списку путей необработанных файлов, полученному активностью *«Файлы Получить пути к файлам/каталогам»*. Далее будут описываться действия для одного пути к файлу, которые одинаковые для каждого пути.

В первую очередь, полный путь к файлу проверяется регулярным выражением на принадлежность файла к формату JSON с помощью активности *«Строки/Регулярное выражение»*. В окне свойств активности заполняются поля: *Входная строка* – переменная с полным путём к файлу в формате строки, *Регулярное выражение* – переменная, в которую записано регулярное выражение в строковом формате, *Результат* – булева переменная, куда будет записан результат проверки. А также заводится переменная для хранения текста из обработанного документа с помощью активности *«Базовые/Присвоить значение»*. (Рисунок 4.3.1)

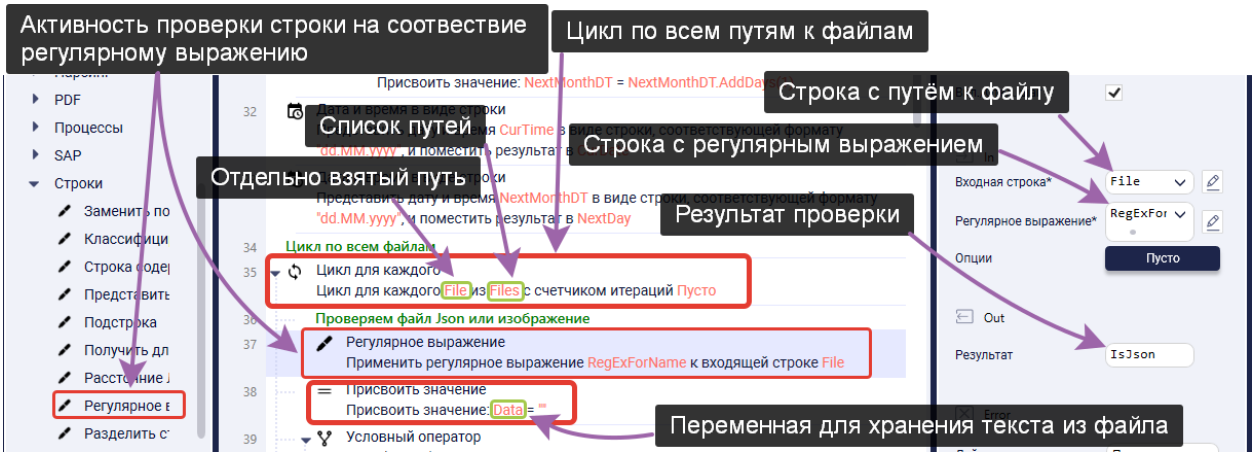


Рисунок 4.3.1. Начало обработки файлов

Затем с помощью активности *«Базовые/Условный оператор»* в зависимости от результатов проверки на соответствие регулярному выражению забирается вся информация из файла. В первом случае из файла формата JSON.

С помощью активности *«Файлы/Прочитать файл»* считывается текстовое содержимое файла. В свойствах указывается: *Путь к файлу* –

переменная со строкой пути к файлу, *Содержимое файла* – переменная, в которую будет записан текст из файла.

Следующим шагом, активностью *«Парсинг/JSON»* строка с текстом преобразуется в JSON объект. В свойствах указывается: *Строка JSON* – переменная со строкой содержимого файла, полученная на предыдущем шаге, *Результат* – переменная с JSON объектом.

Далее активностью *«Базовые/Цикл/Цикл для каждого»* и C# функцией *JSON_Object.EnumerateArray()* получается цикл по всем элементам JSON объекта. И в каждом элементе с помощью активности *«Парсинг/Получить свойство JSON элемента»* забирается свойство со значением текста и активностью *«Базовые/Присвоить значение»* прибавляется в переменную с текстом файла. Свойства активности *«Парсинг/Получить свойство JSON элемента»* заполняются следующим образом: *Элемент JSON* – переменная содержащая элемент JSON, *Имя свойства* – имя свойства, которое необходимо забрать, *Значение* – переменная, в которую будет записано значение необходимого свойства. (Рисунок 4.2.4)

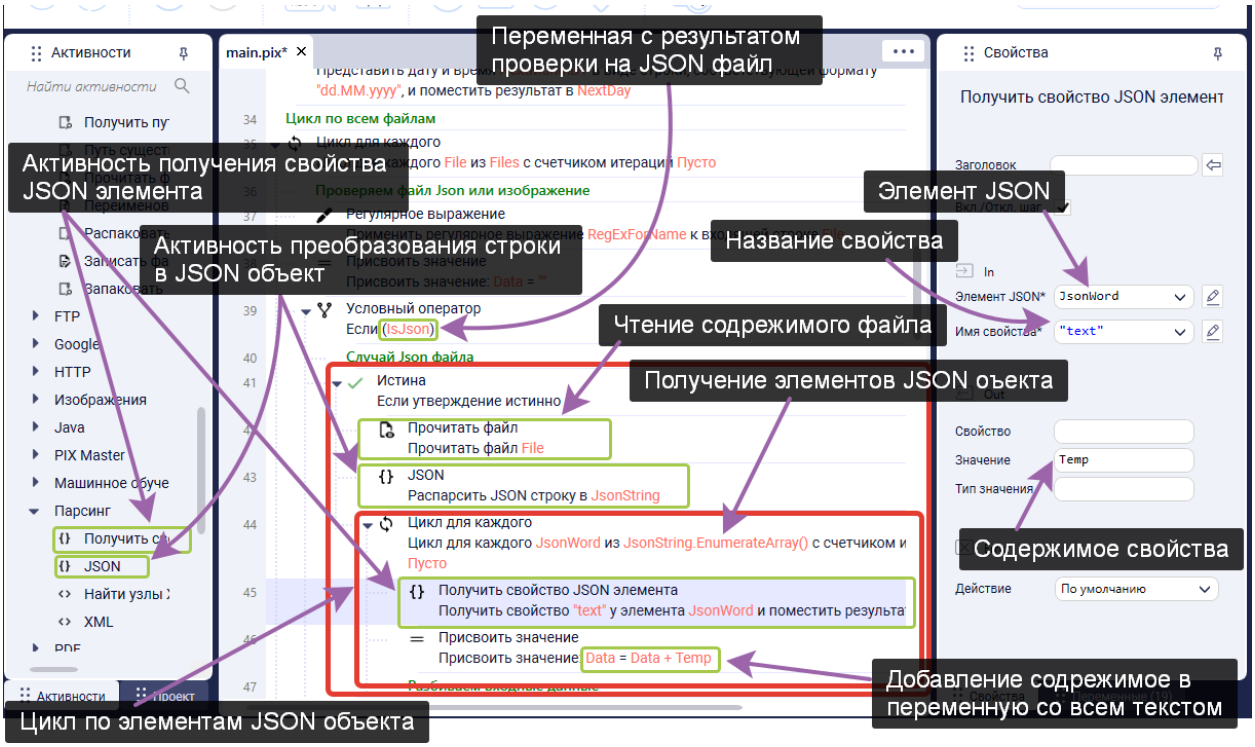


Рисунок 4.3.2.Чтение JSON файла.

В случае, если файл является изображением, файл считывается с устройства с помощью активности *«Изображение/Получить изображение»*. В свойствах указывается: *Путь к картинке* – переменная со строкой пути к файлу, *Изображение* – переменная, куда будет сохранено изображение.

Затем к полученному изображению применяется активность *«OCR/Tesseract»*, которая получает текст из изображения. В свойствах активности указывается: *Изображение* – переменная с изображением, полученная на предыдущем шаге, *Язык* – язык текста на изображение, в случае паспортов лучше указать «eng», так как нужны нижние строчки, которая записаны английскими буквами, *Результат* – переменная, в которую сохраняется полученный текст. (Рисунок 4.3.3)

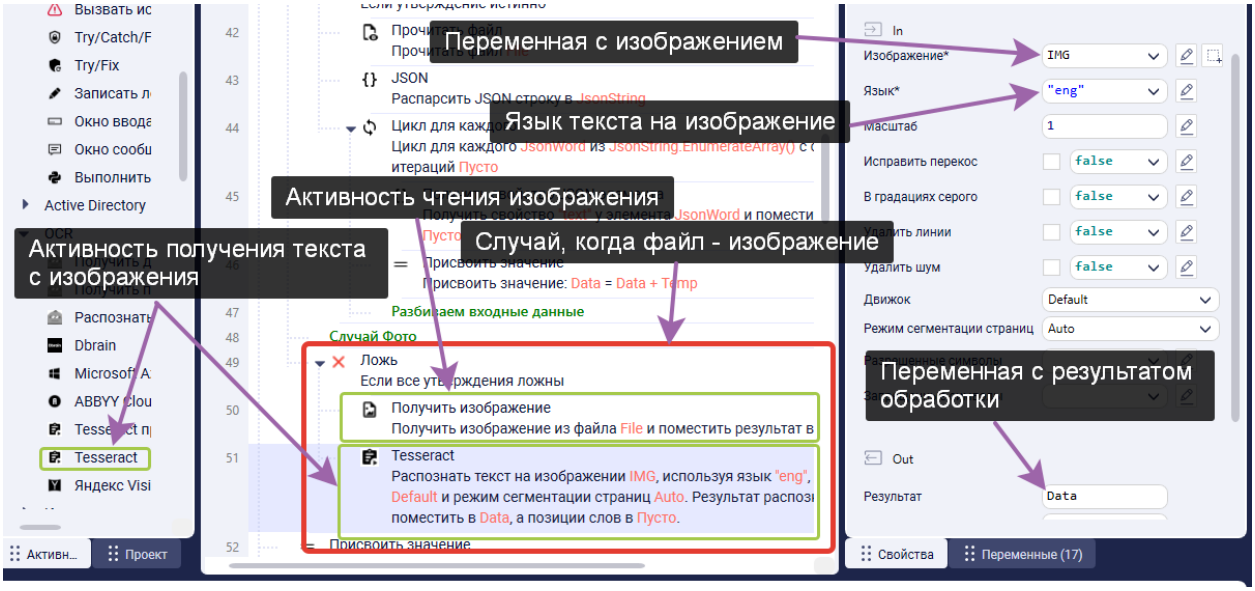


Рисунок 4.3.3. Чтение текста с изображения.

Затем с помощью активностей *«Базовые/Присвоить значение»* и C# функций *Строка.Replace(Текст_под_замену, Заменяющий_Текст)* и *System.Text.RegularExpression.Regex.Match(Текст, Регулярное_Выражение, Опции_Регулярного_Выражения)* текст разбивается на необходимые части. (Рисунок 4.3.4)

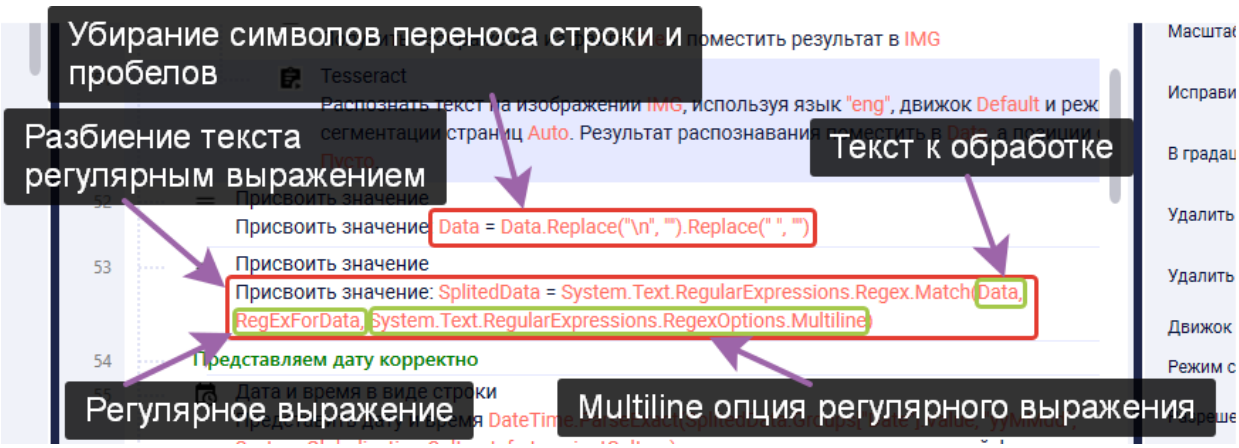


Рисунок 4.3.4. Разбиение текста на необходимые данные

Далее из результатов применения регулярного выражения достаются необходимые значения.

С помощью активности *«Базовые/Дата и время/Дата и время в виде строки»* и C# функций *DateTime.ParseExact(Строка_с_Датой, Формат_даты_в_строке)*, *Результат_Регулярного_Выражения.Groups[“Название_группы”].Value* (в дальнейшем просто *Groups[]*) получается дата выдачи паспорта.

С помощью активности «*Базовые/Присвоить значение*» и C# функции *Groups[]* получается ФИО владельца паспорта английским буквами, и затем с помощью C# функции *Строка.Replace()* английские буквы заменяются на русские. И с помощью C# функции *System.Globalization.CultureInfo.CurrentCulture.TextInfo.ToTitleCase(Строка_в_нижнем_регистре)*. ФИО приводится к формату: первые буквы в верхнем регистре, остальные в нижнем

Следующим шагом с помощью активности «*Базовые/Присвоить значение*» и C# функции *Groups[]* получаются серия и номер паспорта. (Рисунок 4.3.5).

Код подразделения получается аналогичным образом.

Место выдачи получается с помощью активности «*Базовые/Присвоить значение*» и C# функции *Таблица.Rows.Find(Значение_для_поиска)*. (Рисунок 4.3.6)

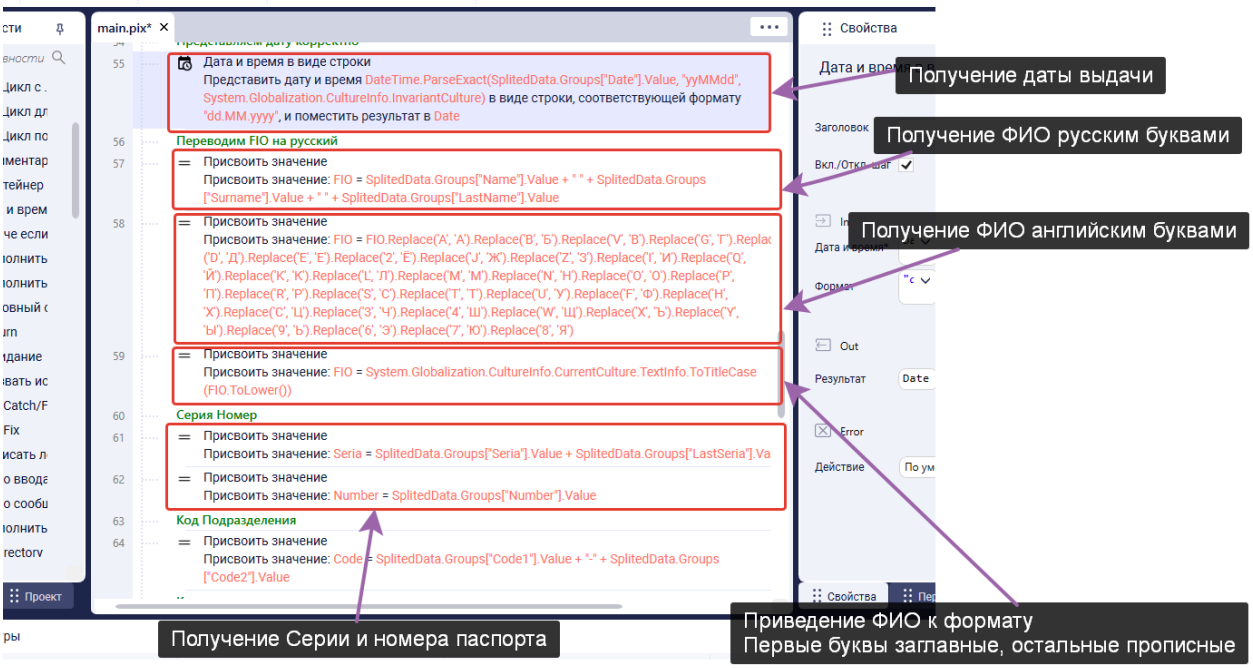


Рисунок 4.3.5. Получение информации

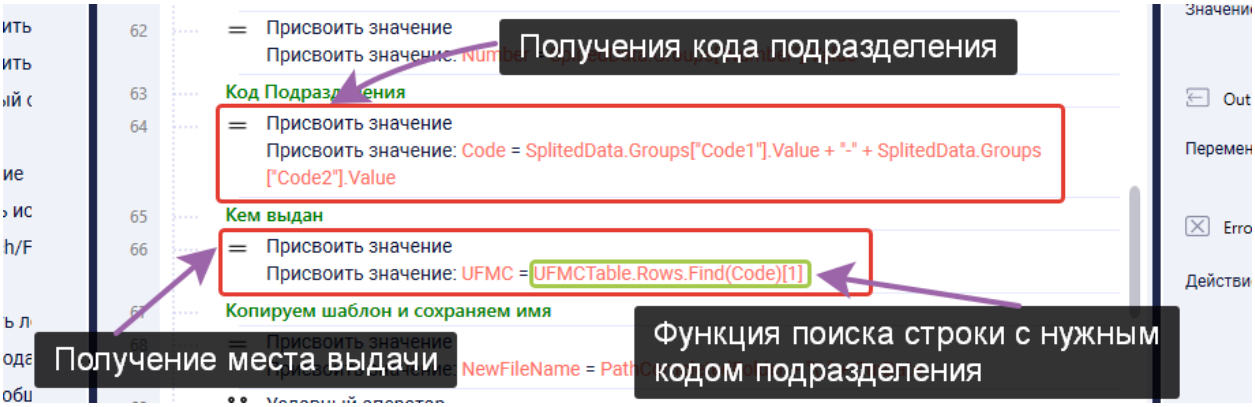


Рисунок 4.3.6. Получение информации

4.4 Сохранение полученных данных

В первую очередь с помощью активностей «*Базовые/Присвоить значение*» и «*Базовые/Условный оператор*», в переменную сохраняется путь до файла с обработанным договором.

Затем с помощью активности «*Файлы/Копировать файл/папку*» файл с шаблоном договора копируется в результирующее место. (Рисунок 4.4.1). В свойствах активности указывается: *Путь откуда* – путь до файла шаблона, *Путь куда* – путь до результирующего файла.

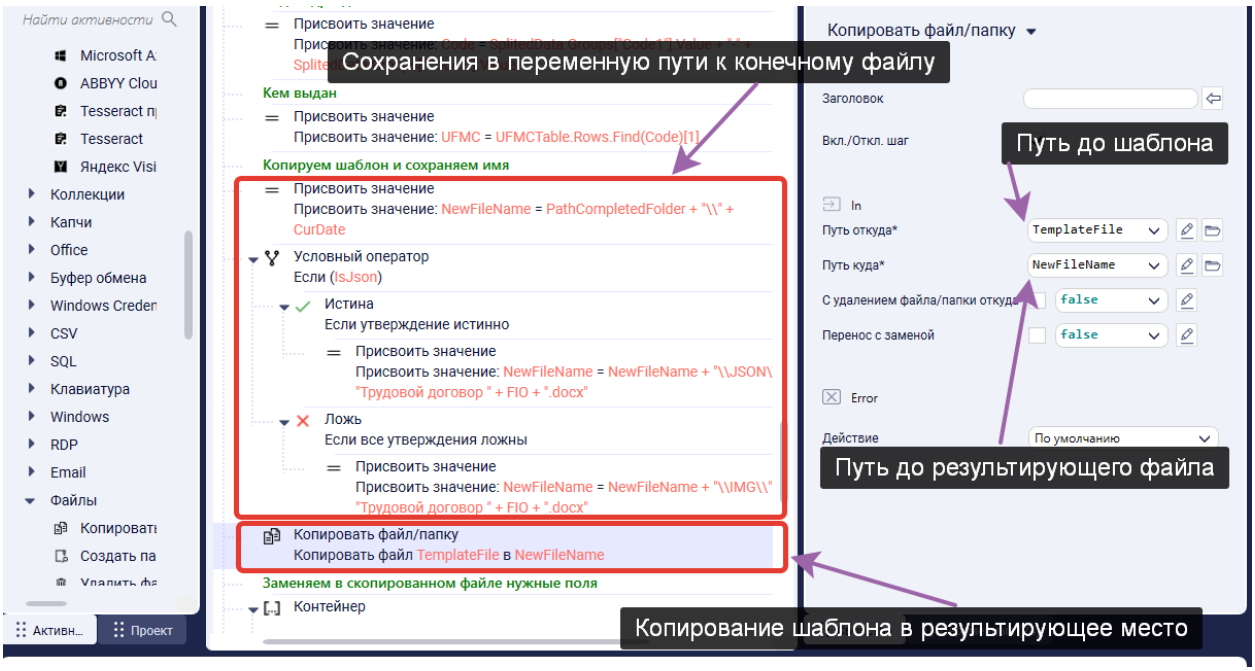


Рисунок 4.4.1. Создание результирующего файла.

Затем с помощью активностей «Office/Word/Заменить текст в Word» все необходимые значения заносятся в результирующий файл. В свойствах указывается: Заменить – строка в шаблоне, которую надо заменить, Заменить на – значение, которое нужно вписать, Учитывать регистр – ставится галочка с целью избежать ненужной замены, путь к файлу – путь к результирующему файлу. (Рисунок 4.4.2)

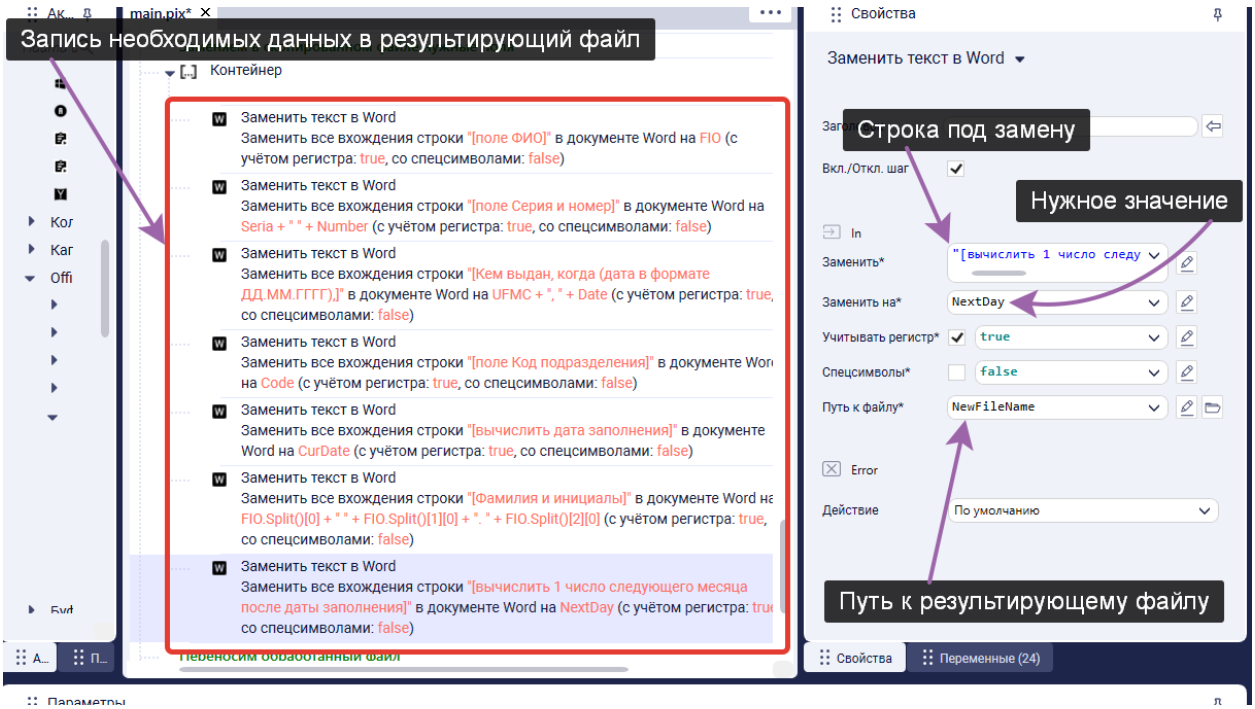


Рисунок 4.4.2. Запись результатов.

Литература

1. https://knowledgebase.pixrpa.ru/actions/Excel/ReadRange
2. https://knowledgebase.pixrpa.ru/actions/Base/Assign
3. https://knowledgebase.pixrpa.ru/actions/CSV/ReadCSV
4. https://knowledgebase.pixrpa.ru/actions/Files/GetListFilesOrCatalogs
5. https://knowledgebase.pixrpa.ru/actions/Base/GetDateTime
6. https://knowledgebase.pixrpa.ru/actions/Base/LoopWhile
7. https://knowledgebase.pixrpa.ru/actions/Base/DateTimeToString
8. https://knowledgebase.pixrpa.ru/actions/Base/If
9. https://knowledgebase.pixrpa.ru/actions/Base/LoopForEach
10. https://knowledgebase.pixrpa.ru/actions/Strings/RegEx
11. https://knowledgebase.pixrpa.ru/actions/Files/ReadFile
12. https://knowledgebase.pixrpa.ru/actions/Parsing/JSONParsing
13. https://knowledgebase.pixrpa.ru/actions/Parsing/JSONElementGetProperty
14. https://knowledgebase.pixrpa.ru/actions/Image/GetImage
15. https://knowledgebase.pixrpa.ru/actions/Tesseract/TesseractOCR
16. https://knowledgebase.pixrpa.ru/actions/Word/ReplaceText
17. https://knowledgebase.pixrpa.ru/actions/Files/CopyFileCatalog
18. https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.match?view=net-7.0
19. https://learn.microsoft.com/ru-ru/dotnet/api/system.data.datatable?view=net-7.0
20. https://learn.microsoft.com/ru-ru/dotnet/api/system.globalization.textinfo.totitlecase?view=net-7.0
21. https://docs.microsoft.com/ru-ru/dotnet/api/system.json.jsonobject?view=dotnet-plat-ext-7.0