DOI 10.37882/2223-2966.2023.8-2.27

СИСТЕМА АВТОМАТИЗАЦИИ ПРОЦЕССОВ МАЛОЙ СЛОЖНОСТИ

AUTOMATION SYSTEM OF PROCESSES OF LOW COMPLEXITY

M. Polenok
S. Bondarenko
O. Yurkova
M. Morgunov

Summary. Software that uses artificial intelligence and machine learning to automate routine tasks that were previously performed by humans. RPA allows you to reduce the influence of the «human factor», as well as achieve a significant increase in work efficiency, which in turn makes it possible to reduce production costs.

Keywords: software, RPA — robots, software systems.

Поленок Максим Викторович
Брянский государственный инженерно-технологический университет
polenok.maksim.2001@mail.ru
Бондаренко Сергей Владимирович
Брянский государственный инженерно-технологический университет
Bondrenkoseregabondarenko576@gmail.com
Юркова Ольга Николаевна
к.э.н., Брянский государственный инженерно-технологический университет
yurkova_olga@mail.ru
Моргунов Михаил Валерьевич
к.т.н., Брянский государственный инженерно-технологический университет
5555@bk.ru

Аннотация. Программное обеспечение, которое использует искусственный интеллект и машинное обучение для автоматизации рутинных задач, которые прежде выполнялись человеком. RPA позволяет уменьшить влияние «человеческого фактора», а также достичь значительного повышения эффективности работы, что в свою очередь дает возможность уменьшения затрат производства.

Ключевые слова: программное обеспечение, RPA-роботы, программные комплексы.

Введение

Сегодня технологии развиваются опережающими темпами, в том числе и искусственный интеллект, который активно применяется во многих областях. Таким образом, возникает необходимость создания новых подходов к организации взаимодействия между человеком и машиной, а также развития новых компетенций, позволяющих участвовать в процессах, где ключевую роль играют технологии. Одной из таких областей является анализ данных.

Компьютерные системы способны успешно выполнять простые и повторяющиеся задачи. Таким образом, в различных сферах человеческой деятельности, где имеется большое количество однотипных задач является логичным внедрение RPA— роботов.

Объектом исследования являются системы автоматизации процессов малой сложности (RPA-роботы).

Предметом исследования выступает процесс создания программного комплекса по разработке и использованию RPA — роботов.

Цель исследования — создание программного комплекса по разработке и использованию RPA — роботов.

Задачи исследования:

1. Изучить объект исследования: системы автоматизации процессов малой сложности (RPA-роботы).
2. Выполнить анализ существующих программных решений и обосновать выбор средств разработки RPA-робота.
3. Выделить модули в разработке.
4. Реализовать модули программного комплекса.
5. Выполнить апробацию и тестирование программного решения.

Отсутствие отечественной системы автоматизации рутинных задач, осуществляемой с использованием роботов-программных агентов (RPA), обеспечивает **практическую значимость и актуальность** данной работы.

Понятие RPA программы, ее функции и классификации

RPA (RoboticProcessAutomation) — это программное обеспечение, которое использует искусственный интеллект и машинное обучение для автоматизации рутинных задач, которые ранее выполнялись человеком.

RPA — это одна из технологий автоматизации бизнес-процессов. Роботизацию применяют для рутинных задач, которые выполняются четко по инструкции (алгоритму).

Она может имитировать действия человека, такие как заполнение форм, обработка данных, отправка электронных писем и многое другое.

RPA-роботы бывают трёх видов:
— Attended RPA — программа, которая работает на компьютере пользователя и выполняет задачи по его запросу или с помощью горячих клавиш.
— Unattended RPA — программа, которая работает на сервере и выполняет задачи автоматически, без участия пользователя.
— Hybrid RPA — программа, которая сочетает в себе функции Attended и Unattended RPA.

Одним из главных преимуществ RPA является повышение эффективности работы. RPA программы могут выполнять задачи гораздо быстрее, чем человек, что позволяет сократить время выполнения задач и увеличить производительность. Кроме того, они могут работать круглосуточно, что также значительно повышает их эффективность.

Еще одним, не менее важным, преимуществом использование «роботов» является оптимизация затрат на персонал: один раз «научив» «робота» выполнять задачу, он сможет повторять ее сколько угодно раз, по-вашему, запросу.

Также использование RPA позволяет уменьшить влияние «человеческого фактора» — человек при выполнении задачи может оказаться «не в духе», что совершенно исключено, если этой же задачей будет заниматься «робот». RPA программы могут выполнять задачи без ошибок, что позволяет снизить вероятность ошибок и увеличить качество работы, что особенно важно для задач, связанных с финансами и бухгалтерией.

В целом, RPA является мощной технологией, которая позволяет автоматизировать рутинные задачи и повысить эффективность работы.

Проектирование компонентов RPA программы

Разрабатываемый программный RPA — комплекс, должен попадать в категорию HybridRPA — что означает наличие возможностей как для удаленного выполнения «сценария» автоматизированной работы, так и локального.

Также для автоматизации работы абсолютно разных программ в различных операционных системах, он должен будет иметь универсальный модуль для распознавания объектов на экране и взаимодействия с ними — для этих целей будет следует использовать компьютерное зрение и так называемый «искусственный интеллект» в качестве универсального «распознавателя».

Так как в настоящее время, так называемый «искусственный интеллект», пользуется большой популярностью было решено внедрить его в наш RPA-робот, в данном случае за словосочетанием «искусственный интеллект» скрывается нейросеть по определению типа объекта, а именно кнопок приложения и его полей ввода.

Программный комплекс будет состоять из следующих модулей:
1. Нейросеть распознавания объектов приложений на изображении.
2. RedRPA — фреймворк для разработки RPA-роботов.
3. Клиентское и серверное приложение на основе RedRPA.

Разрабатываемая нами нейронная сеть, будет ответственна за распознавание типа объекта (кнопка, поле ввода, остальное). НС будет реализована на языке Python, в среде GoogleCollaboratory, с помощью библиотеки TensorFlow.

Ознакомиться со структурой нейронной сети можно на рисунке 1.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 96, 96, 64)        256

 max_pooling2d_2 (MaxPooling  (None, 48, 48, 64)       0
 2D)

 dropout (Dropout)           (None, 48, 48, 64)        0

 max_pooling2d_3 (MaxPooling  (None, 24, 24, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 24, 24, 64)        0

 flatten_1 (Flatten)         (None, 36864)             0

 dense_6 (Dense)             (None, 3)                 110595

=================================================================
Total params: 110,851
Trainable params: 110,851
Non-trainable params: 0
_________________________________________________________________
```

Рис. 1. Структура нейронной сети

По рисунку 1 можно заметить, что модель использует такие «слои» как:
— **Dense** — полно-связный слой, где каждый «нейрон» соединен с каждым.
— **MaxPooling2D** — слой свёртки, реализующий её по максимальному значению.
— **Dropout** — слой, устанавливающий случайные веса в 0, используется для предотвращения переобучения модели.
— **Flatten** — нормализующий слой, реализующий трансформацию многомерного массива в одномерный.

Проектирование фреймворка RedRPA

Проектирование библиотек и фреймворков совсем не тривиальная задача и далеко не каждый программист сталкивается с ней. При разработке данного типа программного обеспечения следует учитывать большое количество различных деталей, лишь косвенно относящихся к основному функционалу.

Так как, одним из основных требований к фреймворку заявлена гибкость и расширяемость, архитектура RedRPA будет модульная. Ментальная модель библиотеки будет выглядеть следующим образом: Ядро (Core) + Внешние модули (ExternalModules).

Для обеспечения расширяемости и гибкости проектируемой библиотеки в ядре был использован такие возможности объектно-ориентированного программирования, как абстрактные классы и наследование.

Абстракции будут реализованы в виде внутреннего модуля ядра **Abstract**. Фреймворк должен соответствовать по возможностям программам класса HybridRPA.

Во время работы робота необходимым элементом будут средства для формирования «сценария» автоматизации работы. Мы спроектируем и разработаем язык сценариев RSL (RedScenarioLanguage).

Но что есть язык — просто набор синтаксически корректных единиц (слов). Слова не умеют выполнять сценарии автоматизации, следовательно, наша задача научить их это делать, для этого нам потребуется компилятор — «переводчик» языка RSL на язык, что сможет выполнить машина. Кроме этого, нам потребуется собственная виртуальная машина.

Все вышесказанное подводит к созданию первого модуля абстракций: **SDK** (ScenarioDevelopmentKit) — набор средств разработки сценариев.

Также наш RPA-робот должен иметь возможности для универсальной оптимизации с помощью, так называемого «искусственного интеллекта». По уже сложившейся традиции, для него нужно спроектировать абстракцию, представленную модулем **ObjectScanning**. Формально процесс поиска объекта на изображении можно разбить на 3 этапа:

1. Поиск «шаблонов» объекта на изображении.
2. Определение класса «шаблона».
3. Присвоение идентификатора распознанному объекту.

С первыми двумя этапами не возникает сложностей в понимании, стоит обратить внимание на 3 пункт, в нашем случае «Присвоение идентификатор…» будет являться распознаванием текста с полученного объекта.

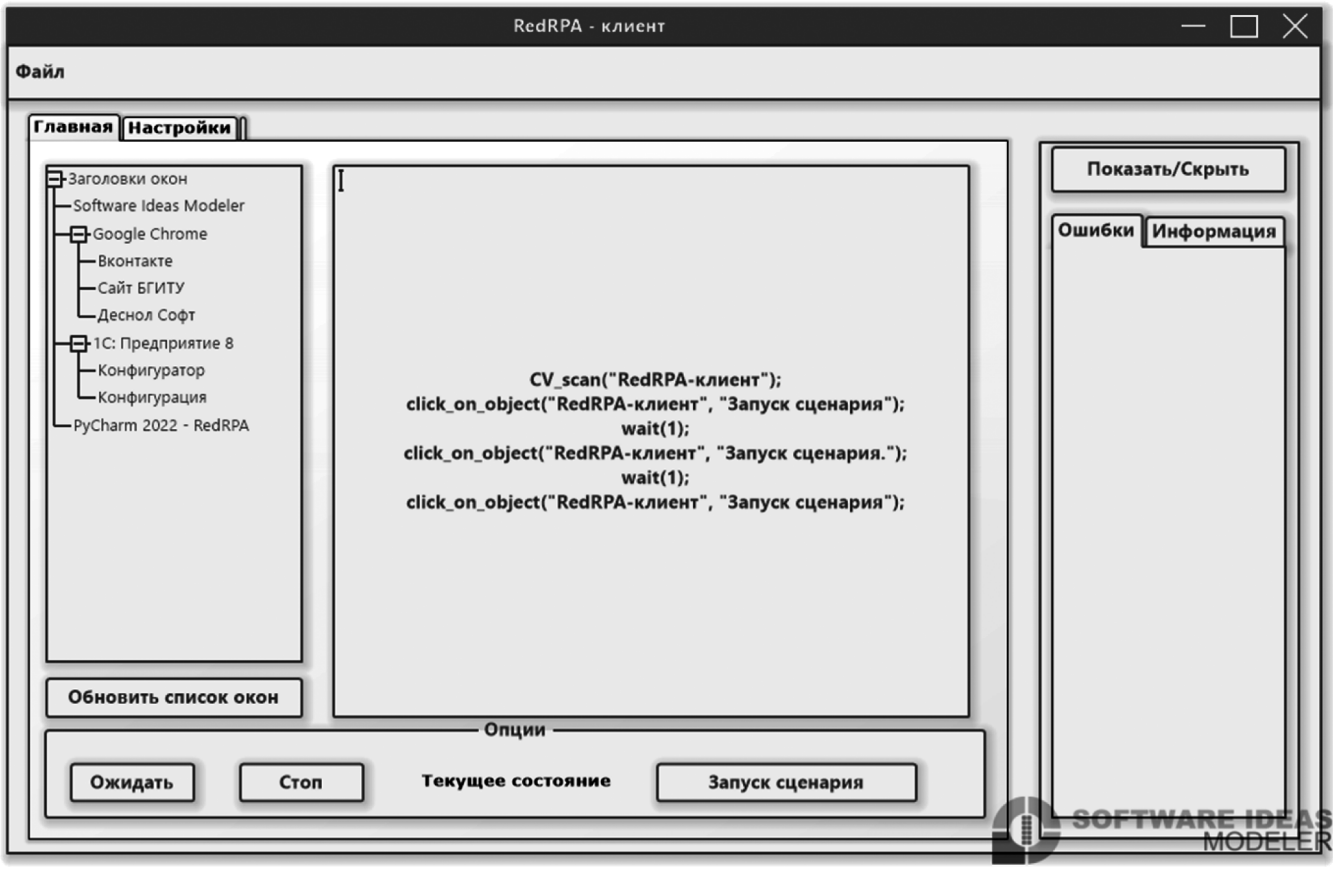

Рис. 2. Вид окна клиентского приложения

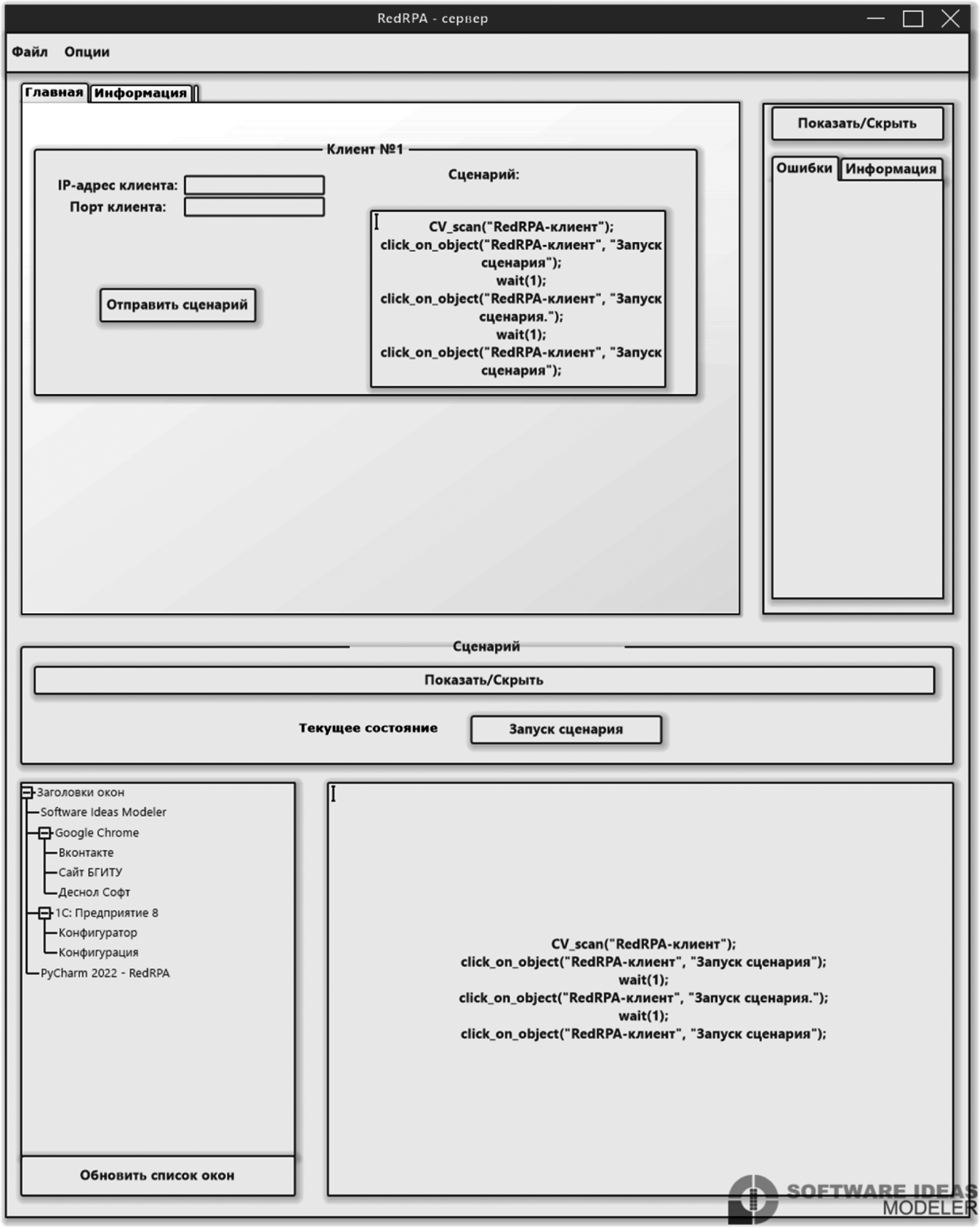

Рис. 3. Вид окна серверного приложения

Кроме всего выше сказанного, данной программе потребуется клиентское и серверное приложения.

И серверное, и клиентское приложения будут спроектированы и разработаны в архитектуре MVC (Model–View–Controller). В качестве фронтенда данных приложений будет выступать графический фреймворк Qt под лицензией PySide2. В роли бекенда будет выступать спроектированный нами ранее фреймворк RedRPA.

Интерфейс клиентского приложения должен содержать:

1. Редактор сценариев.
2. Список окон приложений.
3. Поле для отображения информационных сообщений.
4. Поле для отображения сообщений ошибок компиляции.
5. Кнопки для обеспечения взаимодействия.

На основе этих требований можно спроектировать обобщенный вид клиентского приложения (рисунок 2).

В нижней части окна будут содержаться основные опции взаимодействия с приложением, боковая панель с информацией будет выдвижной (на этот факт намекает кнопка «Показать/Скрыть»).

Так как разрабатываемое приложение должно относиться к классу HybridRPA, то серверное приложение должно иметь не только функции для взаимодействия с клиентскими, но и в какой-то мере дублировать их функционал (рисунок 3).

Как можно заметить, основная часть клиентского функционала «перекочевала» в выдвижную панель в нижней части окна, а на ее месте появилось поле для взаимодействия с клиентами, оно будет поддерживать взаимодействие сразу с несколькими клиентами путем создания нового окна в специально отведенной для этого области по запросу.

Реализация программы, апробация и тестирование программного решения

Первым этапом разработки системы является разработка нейронной сети и сбор наборов данных для обучения и тестирования модели.

Набор данных был собран вручную из более чем 100 программ в различных областях деятельности (1С: Предприятие, Telegram, EpicGames, Проводник и т.д.).

Для реализации нейронной сети будет использовать библиотека TensorFlow и среда GoogleCollab, предоставляющая мощности GPU для обучения моделей.

На рисунке 4 можно ознакомиться с процессом реализации модели нейронной сети.

После успешной реализации модели можно приступить к обучения нейронной сети (рисунок 5).

На последнем этапе обучения точность разрабатываемой модели составляет 78 %, что является приемлемым результатом.

Реализация модуля компьютерного зрения проходила в 3 этапа:

▼ Собираем модель

```
[ ]  #@title Собираем модель
     from keras.models import Sequential
     from keras.layers import Dense, Conv2D, Flatten, AvgPool2D, MaxPool2D, BatchNormalization, Dropout
     from keras.optimizers import Adam
     from keras import regularizers

     model=Sequential([
         Dense(64, input_shape=(img_width, img_height, 3)),
         MaxPool2D((2, 2), strides=2),
         Dropout(0.2),
         MaxPool2D((2, 2), strides=2),
         Dropout(0.2),
         Flatten(),
         Dense(3, activation='softmax')
     ])

     model.compile(optimizer=Adam(0.01),loss='sparse_categorical_crossentropy',metrics=['accuracy'])
     model.build()
     model.summary()
```

Рис. 4. Реализация модели нейронной сети

▾ **Обучаем модель**

```
[ ]  #@title Обучаем модель
     from keras.callbacks import ModelCheckpoint, History
     from keras.optimizers import Adam

     hist = History()
     epochs = 15
     batch_size = 16
     #model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])
     hist_final = model.fit(x_train, y_train, validation_split=0.2,
                            epochs=epochs, batch_size=batch_size,
                            callbacks=[hist], verbose=1)
```

```
Epoch 1/15
85/85 [==============================] - 14s 163ms/step - loss: 2.0627 - accuracy: 0.7196 - val_loss: 0.6511 - val_accuracy: 0.7882
Epoch 2/15
85/85 [==============================] - 14s 164ms/step - loss: 0.5469 - accuracy: 0.8308 - val_loss: 0.6398 - val_accuracy: 0.8441
Epoch 3/15
85/85 [==============================] - 15s 174ms/step - loss: 0.4798 - accuracy: 0.8249 - val_loss: 0.9224 - val_accuracy: 0.7971
Epoch 4/15
85/85 [==============================] - 14s 162ms/step - loss: 0.3768 - accuracy: 0.8536 - val_loss: 0.6020 - val_accuracy: 0.8676
Epoch 5/15
85/85 [==============================] - 16s 186ms/step - loss: 0.4320 - accuracy: 0.8433 - val_loss: 1.0771 - val_accuracy: 0.7529
```

Рис. 5. Обучение нейронной сети

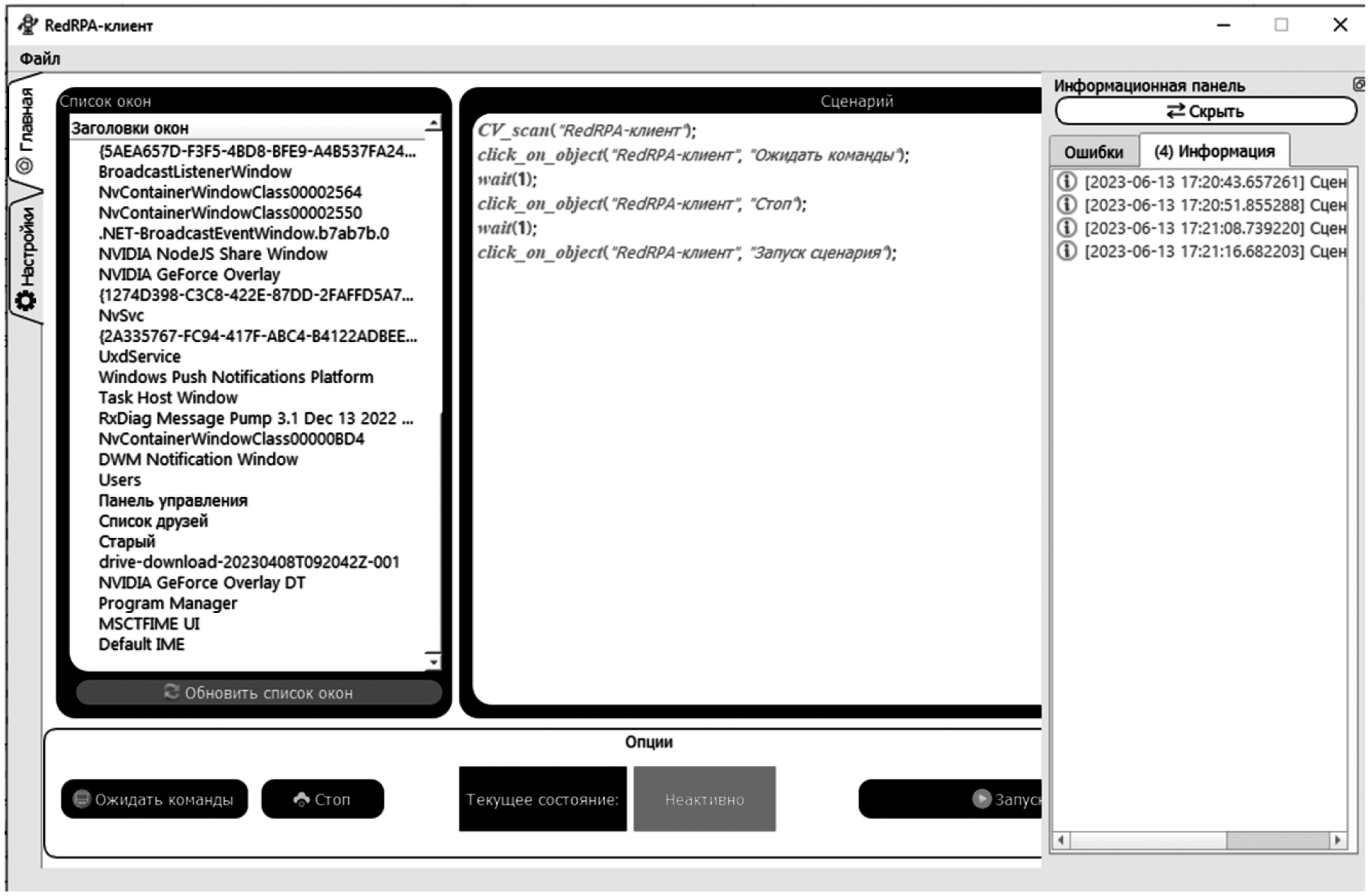

Рис. 6. Дизайн клиентского приложения

1. Для процесса сопоставления шаблонов была использована библиотека OpenCV.
2. Для определения типа объекта пригодилась разработанная ранее нейронная модель.
3. Для распознавания текста использовалась библиотека tesseract-OCR.

Для минимизации ошибок на третьем этапе потребовалось использовать алгоритм нечеткого сравнения строк Дамерау — Левенштайна в реализации библиотеки FuzzyWuzzy. В связи с этим в ядро был добавлен модуль Algorithms, в данный момент реализующий лишь алгоритм нечеткого сравнения строк.

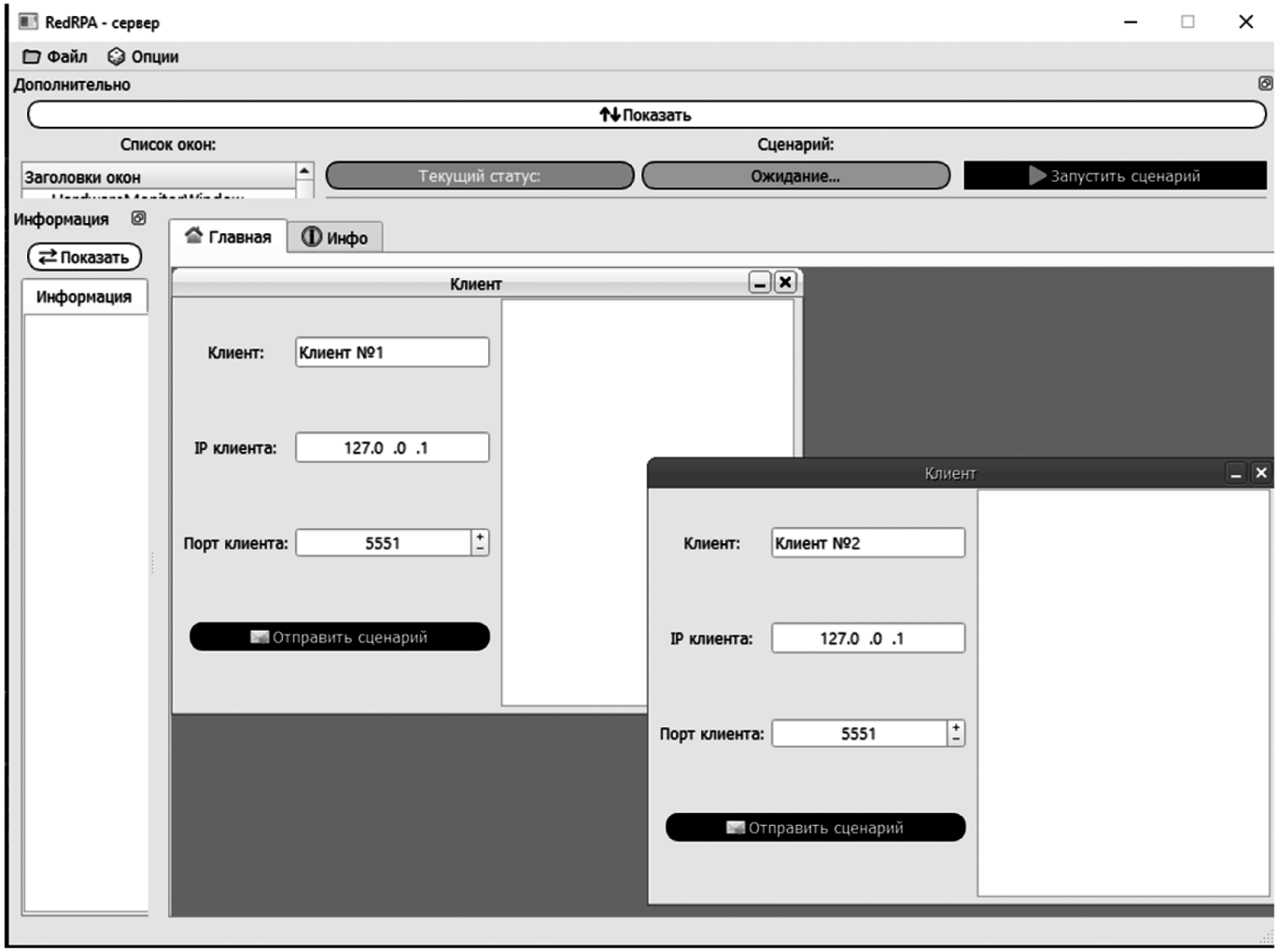

Рис. 7. Пример работы серверного приложения

С помощью графического фреймворка Qt в реализации PySide2, а также графического редактора QtDesigner были реализованы серверное и клиентское приложение на основе фреймворка RedRPA.

На рисунке 6 можно ознакомиться с дизайном клиентского приложения и примером его работы.

Боковая панель является выдвижным элементом интерфейса, так же для удобства она может быть перемещена прямо во время работы программы.

Так как, данное программное решение должно относиться к классу HybridRPA, серверное приложение должно располагать не только возможностями для контроля исполнения сценариев, но и само должно «уметь» автоматизировать работу, то было решено реализовать в форме выдвижного модуля, по аналогии с информационной панелью (рисунок 7).

Заключение

В ходе выполнения данного исследования был разработан программный комплекс по разработке и использованию RPA-роботов. Были изучены системы автоматизации процессов малой сложности (RPA-роботы), был выполнен анализ существующих программных решений, были реализованы модули программного комплекса, также было проведено тестирование программного решения.

Из написанного выше следует, что полностью достигнуты как цели данной работы, так и выполнены все сопутствующие задачи. Также стоит подчеркнуть, что практическая значимость данного исследования заключается в отсутствии отечественной системы автоматизации однотипных рутинных задач, осуществляемых с использованием роботов — программных агентов (RPA), что безусловно указывает на важность данного исследования.

ЛИТЕРАТУРА

1. Альфред В. Ахо, Моника С. Лам Компиляторы. Принципы, технологии и инструментарий — Второе издание, 2008.
2. Документация pyWin32. [Электронный источник] — URL: http://timgolden.me.uk/pywin32-docs/win32_modules.html
3. Как HTTPS обеспечивает безопасность соединения: что должен знать каждый Web-разработчик. [Электронный ресурс] — URL: https://habr.com/ru/articles/188042
4. Константин Владимиров. Базовый курс C++ (MIPT, ILab). Lecture 10. Языки и грамматики. [Электронный ресурс] — URL: https://youtu.be/93eipfA32G8
5. Олег Казимиров — Стрибог. [Электронный источник] — URL: https://github.com/okazymyrov/stribog/tree/master
6. Основы RPA: программные роботы и зачем они нужны. [Электронный ресурс] — URL: https://habr.com/ru/ companies/uipath/articles/574342.

Журнал «Современная наука: актуальные проблемы теории и практики»