ООО «Амфител Плюс»

Программное обеспечение «Textora»

Руководство пользователя-администратора

Документация, содержащая информацию, необходимую для эксплуатации экземпляра программного обеспечения, предоставленного для проведения экспертной оценки в Экспертном совете при Минцифры России

г. Казань, 2026 г.

Введение

Настоящее руководство пользователя-администратора описывает порядок работы с программным обеспечением «Textora» (далее - ПО, Система) и предназначено для двух категорий пользователей:

- Пользователь API (интегратор) - разработчик или информационная система заказчика, осуществляющая интеграцию с Textora через REST API для передачи документов на обработку и получения результатов.
- Администратор системы - сотрудник ИТ-службы заказчика, ответственный за развёртывание, настройку, запуск, остановку и мониторинг системы на уровне хостовой операционной системы.

Программное обеспечение «Textora» предназначено для автоматизированного распознавания документов физических лиц и извлечения из них структурированных данных. ПО не имеет самостоятельного визуального пользовательского интерфейса и предоставляет функциональность через REST API.

Взаимодействие с системой осуществляется через REST API: клиентское приложение заказчика направляет HTTP-запросы к эндпоинтам системы, получая структурированные ответы в формате JSON. Документация API автоматически генерируется в составе работающего приложения и доступна в форматах Swagger UI и ReDoc.

Разграничение доступа

Система не содержит встроенных механизмов аутентификации. Все эндпоинты REST API доступны любому клиенту, имеющему сетевой доступ к порту 5434 на сервере развёртывания. Разграничение доступа обеспечивается средствами сетевой инфраструктуры заказчика: межсетевым экраном, VPN, сегментацией сети. Рекомендуется развёртывание в изолированном сетевом сегменте без прямого доступа из публичных сетей.

Роль	Доступ	Инструменты
Пользователь API (интегратор)	Все эндпоинты REST API (/api/tasks, /api/check, /api/verify/passport, /api/health, /api/stats, /api/archive, /api/config)	HTTP-клиент, Swagger UI, ReDoc
Администратор системы	Управление контейнером на хостовой ОС; все эндпоинты REST API	Docker CLI, командная строка хостовой ОС

Отдельного административного API или графического интерфейса администрирования не предусмотрено. Все операции администрирования выполняются через Docker CLI на хостовой ОС.

Для выполнения административных операций требуются права на выполнение команд Docker на хостовой ОС (членство в группе docker или права суперпользователя).

Рекомендуемые системные требования

Для развёртывания программного обеспечения необходима следующая конфигурация сервера:

Компонент	Минимальные требования	Рекомендуемые требования
Процессор (CPU)	2 ядра, поддержка инструкций AVX и AVX2	4 ядра и более, поддержка AVX/AVX2
Оперативная память (RAM)	2 ГБ	6 ГБ и более
Дисковое пространство (HDD/SSD)	40 ГБ	100 ГБ и более
Архитектура	x86-64	x86-64
Операционная система	Linux x86-64 (любой дистрибутив на ядре Linux)	Linux x86-64
Платформа контейнеризации	Docker, совместимый с Docker Compose	Docker, Docker Compose

Система поддерживает функционирование как на CPU, так и на GPU. При наличии совместимого видеоускорителя обеспечивается повышенная скорость обработки документов.

Для клиентского приложения, осуществляющего интеграцию через REST API, специальных требований к конфигурации не предъявляется - достаточно возможности отправки HTTP-запросов по протоколу TCP/IP.

Начало работы с системой

Запуск системы

После развёртывания Docker-образа (см. Инструкцию по установке) система запускается следующей командой:

```
docker-compose up -d
```

Система функционирует в фоновом режиме без участия оператора и готова принимать запросы после успешного запуска контейнера.

Проверка работоспособности

Для проверки состояния системы после запуска необходимо выполнить запрос к эндпоинту состояния:

```
curl http://<IP-адрес сервера>:5434/api/health
```

В параметре <IP-адрес сервера> необходимо указать IP-адрес или hostname сервера, на котором развёрнут Docker-контейнер. Значение определяется конфигурацией сетевой инфраструктуры заказчика при развёртывании.

Успешный ответ подтверждает готовность системы к приёму запросов.

Доступ к документации API

Интерактивная документация REST API доступна по следующим адресам:

- Swagger UI: http://<IP-адрес сервера>:5434/docs
- ReDoc: http://<IP-адрес сервера>:5434/redoc

Документация генерируется в формате OpenAPI автоматически в составе работающего приложения. В интерфейсе Swagger UI доступна возможность выполнения тестовых запросов непосредственно из браузера.

Остановка системы

Для остановки системы необходимо выполнить:

```
docker-compose down
```

Администрирование системы

Все операции администрирования выполняются администратором системы через командную строку хостовой ОС с использованием Docker CLI.

Основные команды

Запуск системы:

```
docker-compose up -d
```

Остановка системы:

```
docker-compose down
```

Проверка состояния контейнера:

```
docker-compose ps
```

Просмотр журнала работы:

```
docker-compose logs -f
```

Изменение конфигурации

Конфигурация системы задаётся в файле config.yaml. Для применения изменений необходимо перезапустить контейнер:

```
docker-compose down
docker-compose up -d
```

Основные параметры конфигурации:

Параметр	Описание	Значение по умолчанию
worker.num_workers	Количество потоков обработки задач	1
worker.executor_timeout	Таймаут обработки одной задачи (секунды)	120
queue.max_size	Максимальный размер очереди задач	1000
upload.max_file_size_mb	Максимальный размер входного файла (МБ)	50
ml.use_gpu	Использование GPU для инференса нейросетевых моделей	false

Для включения режима GPU необходимо установить параметр ml.use_gpu: true. Использование GPU требует наличия совместимого видеоускорителя на сервере развёртывания.

Мониторинг состояния системы

Текущее состояние системы и метрики:

```
curl "http://<IP-адрес сервера>:5434/api/health"
```

Статистика обработки задач:

```
curl "http://<IP-адрес сервера>:5434/api/stats"
```

Журнал контейнера:

```
docker-compose logs -f
```

Перечень эндпоинтов REST API

Все эндпоинты доступны без аутентификации для любого клиента с сетевым доступом к порту 5434.

Метод	Путь	Описание
POST	/api/tasks	Создание задачи распознавания документа
GET	/api/tasks/{id}	Получение статуса и результатов задачи распознавания
POST	/api/check	Создание задачи проверки качества изображения
GET	/api/check/{id}	Получение статуса и результатов проверки качества
POST	/api/verify/passport	Верификация паспортных данных без обработки изображения
GET	/api/verify/{id}	Получение результата верификации
GET	/api/health	Проверка состояния системы и метрики
GET	/api/stats	Статистика обработки задач
GET	/api/archive	История задач
GET	/api/config	Конфигурация для клиентского приложения

Распознавание документов

Создание задачи распознавания

Для распознавания документа необходимо отправить POST-запрос с файлом документа на эндпоинт /api/tasks. Поддерживаемые форматы входных файлов: JPEG, PNG, TIFF, BMP, WEBP, PDF, ZIP-архив (для пакетной обработки нескольких изображений). Максимальный размер файла - 50 МБ.

Пример запроса:

```
curl -X POST "http://<IP-адрес сервера>:5434/api/tasks" \
  -F "file=@document.jpg"
```

Ответ содержит идентификатор созданной задачи и её начальный статус:

```
{
```

```
  "task_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "pending",
  "file_type": "image",
  "total_pages": 1
}
```

Получение результатов распознавания

Обработка задачи выполняется асинхронно. Для получения статуса и результатов необходимо выполнить GET-запрос с идентификатором задачи:

```
curl "http://<IP-адрес сервера>:5434/api/tasks/550e8400-e29b-41d4-a716-446655440000"
```

Статусы задачи

Статус	Описание
pending	Задача создана, ожидает обработки в очереди
processing	Задача выполняется
completed	Задача выполнена успешно, результаты доступны
partial	Часть страниц обработана успешно, часть завершилась с ошибками
failed	Задача завершилась с ошибкой

Структура ответа

Поле	Тип	Описание
task_id	string	UUID задачи
status	string	Статус задачи
file_type	string	Тип файла: image, pdf, zip
total_pages	int	Общее количество страниц или изображений
processed_pages	int	Количество обработанных страниц
created_at	string	Время создания задачи (ISO 8601)
updated_at	string	Время последнего обновления (ISO 8601)
page_statuses	object	Статусы обработки каждой страницы
page_results	object	Результаты для каждой страницы (ключ - строковый индекс страницы)

Поле	Тип	Описание
processing_time	float	Общее время обработки (секунды)

Результаты обработки каждой страницы находятся в поле page_results. Ключи в объектах page_statuses и page_results являются строками: "0", "1", "2" и т.д.

Извлекаемые поля документов

Паспорт РФ - страница 2

Поле	Описание
passport_series_number_page2	Серия и номер (формат: XX XX XXXXXX)
issue_date	Дата выдачи (формат: ДД.ММ.ГГГГ)
division_code	Код подразделения (формат: XXX-XXX)
who_issued	Орган выдачи (Кем выдан)
document_source	Источник изображения: photo, scan, photo_of_scan, photo_of_screen, selphi

Паспорт РФ - страница 3

Поле	Описание
passport_series_number_page3	Серия и номер (формат: XX XX XXXXXX)
last_name	Фамилия
first_name	Имя
patronymic	Отчество
birthday	Дата рождения (формат: ДД.ММ.ГГГГ)
place_of_birth	Место рождения
gender	Пол (МУЖ. / ЖЕН.)
mrz1	MRZ - строка 1 (44 символа)
mrz2	MRZ - строка 2 (44 символа)

Паспорт РФ - страницы 5-12 (адреса регистрации)

Для каждой страницы регистрации возвращается массив блоков. Каждый блок содержит:

Поле	Описание
page	Номер страницы паспорта
index	Индекс блока на странице (0 - верхний блок)
block_type	Тип блока: printed (печатный) или handwritten (рукописный)
region	Регион (для печатных блоков)
district	Район (для печатных блоков)
point	Населённый пункт (для печатных блоков)
street	Улица (для печатных блоков)
house	Дом (для печатных блоков)
building	Корпус (для печатных блоков)
flat	Квартира (для печатных блоков)
date	Дата регистрации (для печатных блоков)
department	Подразделение (для печатных блоков)
division_code	Код подразделения (для печатных блоков)
status	Статус регистрации (для печатных блоков)

СНИЛС

Поле	Описание
number	Страховой номер индивидуального лицевого счёта (формат: XXX-XXX-XXX XX)
lastname	Фамилия
firstname	Имя
patronymic	Отчество
date_of_birth	Дата рождения
gender	Пол
place_of_birth	Место рождения
date_of_registration	Дата регистрации в ПФР

Проверка качества изображения

Создание задачи проверки качества

Для проверки качества входного изображения или PDF-файла необходимо отправить POST-запрос на эндпоинт /api/check. Поддерживаемые форматы: изображения (JPEG, PNG,

BMP, WEBP) и PDF. Проверка выполняется асинхронно и независимо от задачи распознавания.

Пример запроса:

```
curl -X POST "http://<IP-адрес сервера>:5434/api/check" \
  -F "file=@image.jpg"
```

Поля результата проверки качества

Поле	Тип	Описание
is_grayscale	bool	Изображение в оттенках серого
source	string	Источник изображения: photo, photo_of_scan, photo_of_screen, scan, selphi
documents_count	int	Количество найденных документов
documents	object	Словарь документов: ключ - тип документа, значение - объект с полями или null
documents[type].fields	object	Словарь полей: {field_name: {"detected": bool}}
documents[type].detected_count	int	Количество найденных полей
documents[type].total_count	int	Общее количество полей модели
climbs_out	bool	Документ выходит за границы изображения
is_low_resolution	bool	Низкое разрешение документа
is_blurred	bool	Документ размыт
has_fingers	bool	Обнаружены пальцы, перекрывающие поля документа
has_flare	bool	Обнаружены засветы на документе
fingers_affected_fields	array	Список полей, перекрытых пальцами
flare_affected_fields	array	Список полей, перекрытых засветами
image_shape	array	Размеры изображения [height, width]

Детекция полей (documents[type].fields) поддерживается для типов: passport_page2, passport_page3, snils. Для остальных типов документов значение поля documents[type] будет null.

Верификация паспортных данных

Создание задачи верификации

Верификация паспортных данных выполняется синхронно без обработки изображения. Запрос принимает JSON-объект с полями паспорта и возвращает детальный отчёт по каждой из 33 проверок.

Пример запроса:

```
curl -X POST "http://<IP-адрес сервера>:5434/api/verify/passport" \
  -H "Content-Type: application/json" \
  -d '{
    "last_name": "ИВАНОВ",
    "first_name": "ИВАН",
    "patronymic": "ИВАНОВИЧ",
    "birthday": "01.01.1990",
    "issue_date": "15.06.2010",
    "division_code": "770-012",
    "passport_series_number": "92 16 147936"
  }'
```

Входные поля верификации

Поле	Тип	Обязательное	Описание
last_name	string	Да	Фамилия
first_name	string	Да	Имя
patronymic	string	Да	Отчество
birthday	string	Да	Дата рождения (ДД.ММ.ГГГГ)
issue_date	string	Да	Дата выдачи (ДД.ММ.ГГГГ)
division_code	string	Да	Код подразделения (XXX-XXX)
passport_series_number	string	Да	Серия и номер (XX XX XXXXXX)
who_issued	string	Нет	Орган выдачи
gender	string	Нет	Пол (МУЖ. / ЖЕН.)
mrz1	string	Нет	MRZ - строка 1 (44 символа)
mrz2	string	Нет	MRZ - строка 2 (44 символа)

Группы проверок и статусы

Верификация включает 33 проверки, сгруппированные по следующим категориям: required_fields (обязательные поля), full_name (ФИО), birthday (дата рождения), issue_date (дата выдачи), cross_dates (перекрёстные проверки дат), passport_series (серия и номер паспорта), division_code (код подразделения), mrz (машиночитаемая зона).

Статус проверки	Описание
passed	Проверка пройдена
error	Проверка не пройдена; поле message содержит описание ошибки
skipped	Проверка пропущена при недостаточности входных данных

Техническая поддержка

В случае возникновения затруднений при использовании системы обращайтесь в службу технической поддержки ООО «Амфител Плюс»:

по телефону: 89013470518

по электронной почте: info@amfitel.ru