**Руководство администратора**

**программного обеспечения**

**«Система автоматизированной обработки**

**материалов авиасъемки водных биологических ресурсов»**

**Содержание**

[1 Назначение, цели и задачи ПО 3](#_Toc66804899)

[2 Авторизация в системе 6](#_Toc66804900)

[3 Добавление исследований 7](#_Toc66804901)

[4 Просмотр результатов исследований 9](#_Toc66804902)

[5 Использование карты 11](#_Toc66804903)

# Назначение, цели и задачи ПО

Программное обеспечение «Система автоматизированной обработки материалов авиасъёмки водных биологических ресурсов и формирования отчетов исследований» (ПО САОМА ВБР) – предназначено для автоматизированной обработки полученных первичных материалов авиасъёмки различных видов водных биологических ресурсов с помощью инструментов цифрового зрения и искусственного интеллекта, заключающейся в распознавании образа исследуемого водного биологического ресурса, а также предоставлении формализованных отчетов о численности и пространственном распределении исследуемых водных биологических ресурсов.

Функционал пользователя программного обеспечения позволяет:

* Просматривать список проведенных исследований, проводить поиск, сортировку по списку;
* Добавлять новые исследования;
* получать формализованные отчеты результатов обработки о численности и пространственном распределении исследуемых водных биологических видов, в том числе получать отчеты уже проведенных ранее исследований;
* выгрузить сформированный отчет в нескольких форматах xlsx, csv, txt;
* увидеть пространственное распределение особей исследования на карте мира;
* увидеть результаты распознавания особей на исходных материалах авиасъемки с возможностью коррекции точности распознавания.



Рисунок 1 – Структурная схема инфраструктуры ПО САОМА ВБР

Таблица 1 – Описание блоков структурной схемы инфраструктуры ПО САОМА ВБР

|  |  |
| --- | --- |
| **Название** | **Описание** |
|  |  |
| Vniro.smrtch.com | Фронтэнд ПО САОМА ВБР. Использует статические файлы для сайта: css, html, js. |
| Vniro-api.smrtch.com | Публичное API проекта. Основное назначение – пересылка запросов ко всем микросервисам и агрегация ответов в удобные структуры для использования на фронтэнде. |
| BiologicalSystematics | Микросервис отвечающий за справочник биологическихэкземпляров и систематики. Основное назначение – создание, изменение, удаление и просмотр справочника. |
| Researches | Микросервис отвечающий за исследования. Основное назначение – создание, изменение, удаление и просмотр исследований. |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Branches | Микросервис отвечающий за справочник филиалов. Основное назначение – создание, изменение, удаление и просмотр справочника. |
| GeoObjects | Микросервис, отвечающий за справочник географических объектов. Основное назначение – создание, изменение, удаление и просмотр справочника. |
| Daemon | Постоянно запущенное приложение, которое мониторит FTP-сервер на наличие новых фотоматериалов, а также запрашивает список исследований, которые необходимо отправить на распознавание. Основное назначение – выполнение задач независимо от пользователя. |
| Vsftpd | FTP-сервер с поддержкой IPv6 и SSL. Основное назначение –временное хранилище для фотоматериалов. |
| Files | Микросервис отвечающий за фотоматериалы и дополнительные материалы. Основное назначение – создание, удаление и просмотр материалов. |
| Microsoft SQL Server2019 | Система управления реляционными базами данных (РСУБД).Основное назначение – хранение данных для микросервисов. |
| Recognize service | Сервис, отвечающий за обработку материалов исследований и вычислений координат и площади найденных особей. |
| VSTS Build agent | Агент, который при внесении изменений в git репозитории, забирает исходный код микросервиса, компилирует его, упаковывает в Docker контейнер и загрумжает его в приватный репозиторий. |

# Рекомендуемые системные требования

* 1. Сервер инфраструктуры ПО:
		+ Процессор: Intel Xeon E5 v4 с тактовой частотой 2.2 ГГц;
		+ Оперативная память: 128 Гб;
		+ Жесткий диск: >5 Тб (Примечание: одно исследование занимает около 100 Гб, чем больше жесткий диск, тем на больше исследований его хватит, одна территория исследования около 2 Тб).
	2. Виртуальная машина Database:
		+ Операционная система: Ubuntu 20.04 LTS;
		+ Место на диске: 500 Гб;
		+ Память: 32 Гб Количество ядер: 8;
		+ Дополнительное установленное ПО и компоненты: PostgreSql.
	3. Виртуальная машина Services:
		+ Операционная система: Ubuntu 20.04 LTS;
		+ Место на диске: >4 Тб;
		+ Память: 32 Гб;
		+ Количество ядер: 8;
		+ Дополнительное установленное ПО и компоненты: Ftp-сервер и Docker Compose (для микросервисов).
	4. Сервер обработки фотоматериалов:
		+ Процессор: Intel Core i7-7800X с тактовой частотой 3.50 ГГц;
		+ Оперативная память: 16 Гб;
		+ Жесткий диск: 200 Гб;
		+ Графический процессор: Nvidia Titan XP 12 Гб;
		+ Операционная система: Windows 10;
		+ Дополнительное установленное ПО и компоненты: Python 3.7, Cuda 10.1, CMake 3.18, Nvidia GeForce Experience, Microsoft Build Tools for Visual Studio (необходимо установить средства сборки С++).
	5. Перечень стороннего ПО и компонентов, используемых в проекте:
		+ Docker - https://www.docker.com/
		+ CUDA 10.1 - https://developer.nvidia.com/cuda-zone
		+ .NET Core 3.1 - https://dotnet.microsoft.com/download/dotnet-core/3.1
		+ Python 3.7 - https://www.python.org/
		+ Microsoft Sql Server 2019 - https://www.microsoft.com/ru-ru/sql-server/sql-server-2019
		+ Nginx - https://nginx.org/ru/

# Инструкция по сборке и установке ПО

Для сборки и настройки ПО необходимо:

* 1. Создание аккаунта в Microsoft Azure (бесплатный функционал).

Завести новый или использовать существующий аккаунт Azure DevOps по ссылке:

<https://azure.microsoft.com/en-us/services/devops/?nav=min>,

с возможностью создавать проекты и собирать их.

* 1. Создание проекта в Azure.

После авторизации на портале Azure DevOps необходимо создать новый проект.

* 1. Создание Git-репозитория.

Далее, в созданном проекте необходимо создать Git-репозиторий для хранения исходных кодов из архивов FrontEnd.zip и BackEnd.zip.

* 1. Автоматическая сборка проектов.

Необходимо создать и настроить виртуальную машину с агентом для автоматический сборки проектов.

Описание процесса создания и настройки можно найти по ссылке:

https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure- devops&tabs=browser.

* 1. Создание Docker Hub.

Далее необходимо пройти по ссылке: <https://hub.docker.com/>, и завести новый или использовать существующий аккаунт на Docker Hub.

* 1. Создание репозитория.

Для выкладки контейнеров на Docker Hub необходимо создать приватный репозиторий. Для этого вводим название репозитория и выбраем тип - private. Описание процесса создания приватного репозитория можно посмотреть по ссылке: https://hub.docker.com/repository/create.

* 1. Загрузка архивов в репозиторий.

Далее содержимое архивов должно быть извлечено и загружено в Git-репозиторий с помощью импорта на портале Azure DevOps, либо с помощью pull-запроса Git-клиентом.

* 1. Сборка контейнеров для микросервисов.

Далее необходимо настроить сборку контейнеров (Pipelines) для каждого микросервиса (FrontEnd, FtpDaemon, Operator.ApiGw, BiologicalSystematics.API, Branches.API, Files.API, GeoObjects.API, Researches.API), а также обеспечить их выкладку в Docker Hub с соответствующими именами.

* 1. Работа с базами данных:
		1. В проектах микросервисов (BiologicalSystematics.API, Branches.API, Files.API, GeoObjects.API, Researches.API) необходимо изменить подключения к базе данных отредактировав файлы appsettings.json.
		2. Для микросервисов (BiologicalSystematics.API, Branches.API, Files.API, GeoObjects.API, Researches.API) необходимо применить команду Update-Database для создания базы данных (выбрать Startup проект API, а default project - Infrastucture).
		3. Далее, необходимо создать docker-compose.yml файл для сервера с файлами.

*Пример:*

*version: "3.2" services:*

*nginx:*

*image: nginx:latest container\_name: nginx volumes:*

*- ./nginx:/etc/nginx/*

*- ./certs:/etc/ssl/private*

*- /etc/letsencrypt/:/etc/letsencrypt/ ports:*

*- 80:80*

*- 443:443*

*files:*

*image: "smrtch/smart\_tech:files-api" volumes:*

*- ./filescontent:/data/content expose:*

*- "80"*

* + 1. После чего необходимо создать docker-compose.yml файл для сервера с микросервисами.

Пример:

*version: "3.2" services:*

*nginx:*

*image: nginx:latest container\_name: nginx volumes:*

*- ./nginx:/etc/nginx/*

*- ./certs:/etc/ssl/private*

*- /etc/letsencrypt/:/etc/letsencrypt/ ports:*

*- 80:80*

*- 443:443*

*- 8081:8081*

*- 8082:8082*

*- 8083:8083*

*- 8084:8084*

*api:*

*image: "smrtch/smart\_tech:operator-apigw-dev" expose:*

*- "80"*

*environment:*

*- \_FileService=http://192.168.11.200:80*

*- \_BiologicalSystematicService=biologicalsystematic:80*

*- \_BranchService=branch:80*

*- \_GeoObjectService=geoobject:80*

*- \_ResearchService=research:80 biologicalsystematic:*

*image: "smrtch/smart\_tech:biologicalsystematics-api" expose:*

*- 80*

*branch:*

*image: "smrtch/smart\_tech:branches-api" expose:*

*- 80*

*geoobject:*

*image: "smrtch/smart\_tech:geoobjects-api" expose:*

*- 80*

*research:*

*image: "smrtch/smart\_tech:researches-api" expose:*

*- 80*

*ftpdaemon:*

*image: "smrtch/smart\_tech:ftpdaemon" environment:*

*- ASPNETCORE\_Ftp Server=192.168.11.200*

*- ASPNETCORE\_Ftp Port=21*

*- ASPNETCORE\_Ftp User=ftp*

*- ASPNETCORE\_Ftp Password=ftp*

*- ASPNETCORE\_WaitMinutesToDownloadFile=30*

*apihtml:*

*image: "smrtch/smart\_tech:operator-html" expose:*

*- 80*

* 1. Запуск сервиса распознавания.

Для запуска сервиса распознавания необходимо распаковать архив Recognize.zip и запустить файл requirements.bat для установки всех необходимых библиотек. Для начала работы сервиса запускаем файл service.py.

В случае успешного старта сервиса начинается «прослушивание» всех IP-адресов текущего хоста, что подтверждается выводом в консоли сообщения со следующим содержанием:

*\* Running on http://0.0.0.0:8090/ (Press CTRL+C to quit)*

Сервис Daemon отправляет сформированный JSON-файл на сервер распознавания. Сервер распознавания, в свою очередь, видит задание на обработку, забирает материалы и обрабатывает. После обработки формируется JSON-файл и передает его в микросервис исследований, который формирует отчет.