Акционерное общество «Башкирский регистр социальных карт»

Техническая документация к ИС ЦУМ «Аналитика»
Руководство администратора

2023

1 Общая характеристика

Для реализации функций визуального представления данных в сфере Услуг ЖКХ, последующих анализа и управления данными, разработана информационная система «Центр Управления Муниципалитетом «Аналитика»» (далее – ИС ЦУМ «Аналитика»).

Дашборды ИС ЦУМ «Аналитика» обеспечивают:

- визуализацию ключевых показателей по плановым и аварийным отключениям от Услуг;
- визуализацию ключевых показателей об объектах, предоставляющих Услуги;
- визуализацию ключевых показателей об объектах, потребляющих Услуги;
- возможности дополнительной детализации данных после нажатия на выбранный компонент (круговая диаграмма, интерактивная карта, выпадающий список и т.п.);
- ввод и ведение БД по результатам обследования дымоходов и вентканалов, хранение сопутствующих документов.

2 Системно-технические характеристики

- Операционная система Debian 10,
- СУБД PostgreSQL,
- Платформа Node.js,
- HTTP-сервер Nginx,
- Платформа контроля версий Gitlab,
- Платформа развёртывания ПО Gitlab CI/CD,
- Frontend framework Vue.js 2,
- Мониторинг обновления журнала через Zabbix.

3 Установка и настройка ПО для backend

3.1 Подготовка системы

Установка утилиты sudo

```
apt install sudo
```

Обновление имеющихся пакетов

```
sudo apt update && sudo apt upgrade -y
```

Установка необходимы пакетов для возможности установки и работы Postgresql 14

```
sudo apt -y install gnupg2 wget vim make gcc
```

3.2 Установка Postgresql 14

Добавление репозитория Postgresql 14 в пакетный менеджер apt

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

Импорт gpg ключа для добавленного репозитрия

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

Обновление индексации пакетов apt

```
sudo apt -y update
```

Установка Postgresql 14

```
sudo apt install postgresql-14
```

Устанавливаем дополнительные библиотеки

```
apt install postgresql-server-dev-14
```

3.3 Создание БД

Добавление ролей arm_deployer, arm_api из файла Роли.sql

```
sudo -u postgres psql -f ~/Роли.sql
```

Создание схем gar, vlast таблиц, комментариев, ограничений, функций, триггеров из файла Схема.sql

```
sudo -u postgres psql -f ~/Схема.sql
```

Наполнение таблиц словарей, типов из файла Словари.sql

```
psql -h localhost -d vlast -U pir_deployer -f ~/Словари.sql
```

3.4 Node.js

Установка необходимых пакетов для установки и работы Node.js

```
sudo apt -y install curl
```

Установка менеджера версий Node.js

```
curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash
```

Установка Node.js 16

```
nvm install 16
```

3.5 HTTP-сервер Nginx

Процесс установки Nginx в зависимости от операционной системы описывается здесь:

https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-open-source/.

В частности, для Debian 10 необходимо выполнить следующие команды:

Устанавливаются требуемые системные зависимости:

```
sudo apt install curl gnupg2 ca-certificates lsb-release debian-archive-keyring
```

Затем устанавливается ключ верификации пакета:

```
curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
    | sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null
```

В список источников *apt* добавляется путь к пакету Nginx:

```
echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] \
http://nginx.org/packages/mainline/debian `lsb_release -cs` nginx" \
    | sudo tee /etc/apt/sources.list.d/nginx.list
```

После сохранения требуется удалить, при наличии, установленный из репозитория Debian пакет *nginxcommon*, обновить реестр пакетов и выполнить установку Nginx:

```
sudo apt-get remove nginx-common
sudo apt-get update
sudo apt-get install nginx
```

В случае, если терминация TLS осуществляется нижестоящим сервером, а процесс сервиса API сконфигурирован на ожидание запросов по порту TCP 8084, конфигурация веб-сервера задаётся файлом *conf.d* в папке */etc/nginx/conf.d* со следующим содержанием:

```
server {
    listen      10.130.0.6:443;
    server_name arm-api.oktema.ru ;
    root        /home/vadim/web/arm-api.oktema.ru/public_html;
    index       index.php index.html index.htm;
    access_log  /var/log/nginx/domains/arm-api.oktema.ru.log combined;
    access_log  /var/log/nginx/domains/arm-api.oktema.ru.bytes bytes;
    error_log   /var/log/nginx/domains/arm-api.oktema.ru.error.log error;

    ssl         on;
    ssl_certificate      /home/vadim/conf/web/ssl.arm-api.oktema.ru.pem;
    ssl_certificate_key  /home/vadim/conf/web/ssl.arm-api.oktema.ru.key;

    location / {
                proxy_buffering off;
                proxy_pass http://127.0.0.1:4000/;
                proxy_intercept_errors off;
                proxy_set_header Host $host;
                client_max_body_size 50M;
```

```
  }

    include      /home/vadim/conf/web/snginx.arm-api.oktema.ru.conf*;
}
```

3.6 Сервисы

Запуск и поддержка работы сервисов осуществляется менеджером процессов pm2. Для создания процесса в менеджере используется команда

```
pm2 start <исполняемый_файл> --name <название_сервиса>
```

Для перезапуска сервиса вызывается команда pm2 reload <название_сервиса>

Удаленное развертывание сервисов производится с помощью инструмента Gitlab CI. В корне репозитория сервиса описывается файл .gitlab-ci.yml который содержит инструкцию по развертыванию сервиса при его первичном развертывании или обновлении.

Для описания конфигурации системы используются переменные, которые используются при развертывании

3.6.1 Сервис пользовательского интерфейса

Переменные

Для интеграции сервиса пользовательского интерфейса необходимо создать следующие переменные для каждого сервера на который будет разворачиваться система:

- TARGET_HOST – Адрес сервера, пример
- TARGET_PATH– Путь, по которому будет располагаться директория с сервисом
- TARGET_USER– Имя пользователя для подключения к серверу
- YURTADOM_SSH_KEY – SSH ключ для сервера на который производится развертывание

Непрерывная интеграция

Непрерывная интеграция сервиса API состоит из 2 этапов: сборки(build) и развертывании(deploy).
На стадии сборки происходит:

- Загрузка npm пакетов

На стадии развертывания, для каждого указанного развертывания

- Собранные файлы копируются на рабочий сервер, адрес сервера указывается в переменных окружения GitLab
- Перезапускается сервис

Файл .gitlab-ci.yml выглядит следующим образом:

```
default:
  image: node:16-alpine
  tags:
    - node


build:
  stage: build
```

```
  only:
    - master
  before_script:
    - npm ci --cache .npm --prefer-offline
  script:
    - npm run build
  artifacts:
    paths:
      - dist/*
  cache:
    key: $CI_COMMIT_REF_SLUG
    paths:
      - .npm/

publish:
  stage: deploy
  only:
    - master
  before_script:
    - 'command -v ssh-agent >/dev/null || ( apk update && apk add --no-
cache openssh-client)'
    - eval $(ssh-agent -s)
    - echo "$YURTADOM_SSH_KEY" | tr -d '\r' | ssh-add -
    - mkdir -p ~/.ssh
    - chmod 700 ~/.ssh
    - ssh-keyscan $TARGET_HOST >> ~/.ssh/known_hosts
    - chmod 644 ~/.ssh/known_hosts
    - apk add --no-cache rsync
  script:
    - rsync -avz --delete-after ./dist/*
$TARGET_USER@$TARGET_HOST:$TARGET_PATH
  environment:
    name: production
    url: https://$TARGET_HOST/
```

3.6.2 Сервис API

Переменные

Для интеграции сервиса API необходимо создать следующие переменные для каждого сервера на который будет разворачиваться система:

- TARGET_HOST – Адрес сервера

- TARGET_PATH– Путь, по которому будет располагаться директория с сервисом
- TARGET_USER– Имя пользователя для подключения к серверу
- DOTENV – Будет скопирована в файл .env. и должна содержать следующие значения:

```
# Номер порта
PORT=4000

# Секретное значение для создания и проверки паролей
APP_SECRET="vadim"

# Переменные для подключения к сервису UDS
UDS_HOST="uds.uk-rb.ru"
UDS_TOKEN_KEY="HTTP_X_TOKEN_UDS"
UDS_TOKEN="ca73f686823967a9f9767edd8c6edbdc3e66e3dd"

# Строка для подключения Prisma ORM к базе данных. https://www.prisma.io/docs/reference/database-reference/connection-urls
DATABASE_URL="postgresql://vlast:Vlast$$$@localhost:5432/vlast_dev?schema=vlast&connection_limit=10&pool_timeout=15"

# Пути нахождения директорий. Пути к директории кроме static указываются относительно директории static
DIRECTORY_STATIC="~/projects/ARM-VLAST/static"
DIRECTORY_INSPECT_ACTS="ventilation_repairs/inspect_acts"   # Акты обследования
DIRECTORY_ESTIMATES="ventilation_repairs/estimates"        # Сметы
DIRECTORY_REPAIR_ACTS="ventilation_repairs/repair_acts"    # Акты ремонта
DIRECTORY_TMP="tmp"                                        # Временные файлы
DIRECTORY_OVERHAUL_REGISTERS="overhaul_registers"          # Реестры по капитальному ремонту

# Токен для обращений к сервису dadata
DADATA_TOKEN="63766d42d28fb7d0f84448ae4eaa076e24b81bb9"

# Overhaul registers
OVERHAUL_REGISTERS="/home/vadim/projects/ARM-VLAST/static/overhaul_registers"
```

Непрерывная интеграция

Непрерывная интеграция сервиса API состоит из 2 этапов: сборки(build) и развертывании(deploy).
На стадии сборки происходит:

- Загрузка npm пакетов
- Генерирация клиента используемой Prisma ORM
- Транспилирование исходного кода написанного на языке TypeScript в код на языке JavaScript

На стадии развертывания, для каждого указанного развертывания

- Собранные файлы копируются на рабочий сервер, адрес сервера указывается в переменных окружения GitLab
- Перезапускается сервис с помощью менеджера процессов pm2

Файл .gitlab-ci.yml выглядит следующим образом:

```
default:
  image: 'node:16'
  tags:
    - node
build:
```

```
  stage: build
  only:
    - master
  before_script:
    - NPM_GITLAB_TOKEN=$NPM_GITLAB_TOKEN && export NPM_GITLAB_TOKEN
    - npm ci --cache .npm --prefer-offline --force
    - ls -a
    - echo $DOTENV > .env
  script:
    - npx prisma generate
  artifacts:
    paths:
      - node_modules/*
  cache:
    key: $CI_COMMIT_REF_SLUG
    paths:
      - .npm/
publish:
  stage: deploy
  only:
    - master
  before_script:
    - >-
      command -v ssh-agent >/dev/null || ( apt update && apt install
      openssh-client)
    - eval $(ssh-agent -s)
    - echo "$GITLAB_RUNNER_SSH_KEY" | tr -d '\r' | ssh-add -
    - mkdir -p ~/.ssh
    - chmod 700 ~/.ssh
    - ssh-keyscan dev.oktema.ru >> ~/.ssh/known_hosts
    - chmod 644 ~/.ssh/known_hosts
    - apt update && apt install rsync -y
    - echo $DOTENV > .env
  script:
    - >-
      rsync -avz --delete-after --exclude .git ./
      $TARGET_USER@$TARGET_HOST:$TARGET_PATH
    - 'echo "$TARGET_USER@$TARGET_HOST:$TARGET_PATH"'
    - |
```

```
      ssh $TARGET_USER@$TARGET_HOST "
          cd $TARGET_PATH
          if grep 'not found' <(pm2 reload arm 2>&1) 1> /dev/null ; then
pm2 start ; fi
          pm2 save
      "
  environment:
    name: production
    url: 'https://$TARGET_HOST.ru/'
```

4 Установка и настройка ПО для frontend

Установка NodeJS и NPM https://nodejs.org/en/
Установка Vue CLI https://cli.vuejs.org/ru/guide/installation.html

```
vue add vuetify apollo
npm install --save vue-meta axios vue-axios vue-moment apexcharts vue-
apexcharts leaflet
```

Compiles and hot-reloads tor development

```
npm run serve
```

Compiles and minifies for production

```
npm run build
```

Lints and fixes files

```
npm run lint
```

5 Мониторинг обновления журнала ошибок через Zabbix

Каждый сервис формирует журнал событий и журнал ошибок. Журнал ошибок содержит сведения о возникших исключительных ситуациях и часто требует внимания разработчиков. Поэтому для ряда сервисов настроен мониторинг обновления журнала ошибок через Zabbix, а также отправка сообщений об ошибках соответствующим разработчикам в систему мгновенного обмена сообщениями Telegram и в электронную почту.

Пример сообщения:

«ПРОБЛЕМА!!!: Error_dolg
Проблема началась в 04:03:13 on 2023.05.25
Название Проблемы: Error_dolg
Host: ServicePIR.it.brsc.ru
Важность: High
Эксплуатационные данные: [25.05.2023, 04:02:13] pg: connect ETIMEDOUT 10.3.26.121:5432

ID Проблемы: 12423500

Zabbix-new»

Настройка осуществляется системным администратором по заявке разработчика или аналитика.

5.1 Установка Zabbix

Debian:
wget http://repo.zabbix.com/zabbix/5.0/debian/pool/main/z/zabbix-release/zabbix-release_5.0-1+stretch_all.deb
//Debian
dpkg -i zabbix-release_5.0-1+stretch_all.deb

Ubuntu:
wget https://repo.zabbix.com/zabbix/5.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_5.0-1+xenial_all.deb //Ubuntu22.04
dpkg -i zabbix-release_5.0-1+xenial_all.deb

wget https://repo.zabbix.com/zabbix/5.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_5.0-1+focal_all.deb
//Ubuntu20.04
dpkg -i zabbix-release_5.0-1+focal_all.deb

sudo apt-get update
sudo apt-get install zabbix-agent

systemctl status zabbix-agent
systemctl stop zabbix-agent
systemctl is-enabled zabbix-agent
systemctl enable zabbix-agent
systemctl restart zabbix-agent
systemctl status zabbix-agent
service zabbix-agent restart

wget стандартно отправляет файл в корневой каталог пользователя. Если это пользователь root -> Каталог root.

Если wget не скачивает файл, его можно скачать в браузере и через WinSCP закинуть вручную на сервер в каталог пользователя. И дальше продолжаем с команды dpkg

5.2 Настройка Zabbix

На сервере, который добавляем на Zabbix, делаем следующее:

Заходим на /etc/zabbix/zabbix_agentd.conf

Добавляем в пунктах и ServerActive, адрес сервера Zabbix

Server=ИП_АДРЕС

ServerActive=ИП_АДРЕС

Hostname=ИМЯ_ХОСТА

На веб-интерфейсе Zabbix необходимо:

Перейти в Configuration -> Hosts, в правом верхнем углу нажать Create host.

В разделе Host указать:

Host name: ИМЯ ХОСТА

Grous: нажимаем Select и выбираем к какой группе будет относиться host

Interfaces: прописываем IP Address или DNS. Выбираем один из двух

В разделе Templates:

нажимаем Select, выбираем темплейт (Template OS Linux by Zabbix agent) -> Select.

5.3 Удаление Zabbix

sudo apt-get remove zabbix-agent

sudo apt-get purge --auto-remove zabbix-agent

rm -rf /etc/zabbix/*

6 Требования к персоналу

Требования к системному администратору и администратору БД

Знания и опыт практической работы:

- OS Debian 9 и выше.
- DNS, основных протоколов TCP
- Построение отказоустойчивых кластеров
- PostgreSQL:
  а) установка настройка СУБД
  б) создание, настройка безопасности БД
  в) высокая доступность БД: зеркалирование БД, доставка журналов транзакций, настройка и обслуживание AlwaysON High Awailiability обслуживание индексов (поиск необходимых и создание при необходимости)
  г) секционирование таблиц, распределение секций по файловым группам
  д) резервное копирование БД, целиком и на уровне Файловых групп, создание заданий резервного копирования БД.

Требования к разработчикам

Разработчик должен обладать следующими навыками:

- Знания: Node.js, Nginx, PostgreSQL, HTML, JavaScript, Vue;
- Опыт работы с jQuery
- Опыт разработки API для веб приложений;
- Отличные знания: JavaScript, Vue;
- Опыт работы с базами данных (MongoDB, Microsoft SQL Server, PostgreSQL);
- Глубокое понимание принципов устройства баз данных «изнутри» (индексы, материализованные представления, оптимизация запросов, транзакции, процедуры, работа с планировщиком запросов и пр.);
- Умение писать unit-тесты;
- Уровень английского для чтения технической документации;
- Умение работать с системой контроля версий (gitlab).
- Понимание жизненного цикла разработки ПО.