Акционерное общество «Башкирский регистр социальных карт»

Техническая документация к информационной системе «Претензионно-исковая работа»

Руководство по установке, развертыванию и администрированию системы

2024

Оглавление

Определения и сокращения

Термин, аббревиатура, сокращение, обозначение	Пояснение, определение, расшифровка
API	application programming interface – программный интерфейс приложения.
БД	База данных
Системный администратор	Подразделение/ Служба, отвечающая за системное администрирование и администрирование БД.
Разработчик	Подразделение/ Служба, отвечающая за разработку, адаптацию, доработку, внедрение программного обеспечения и сопровождение Системы
ИС ПИР, ИС, Программа	Информационная система «Претензионно-исковая работа»
ПО	Программное обеспечение – совокупность программ систем обработки информации

1 Общая характеристика программы

Информационная система «Претензионно-исковая работа» (ИС «ПИР») обеспечивает возможность автоматического массового формирования полного пакета документов по ведению претензионно-исковой работы в отношении собственников, являющихся физическими и юридическими лицами, имеющих задолженность по уплате взносов на капитальный ремонт и пени за неполную и (или) несвоевременную уплату взносов на капитальный ремонт в соответствии с региональными требованиями.

Функционал ИС «ПИР» доступен Пользователям с Ролью «Региональный оператор (РО)» и «Администратор», и состоит из следующих разделов:

- Мой профиль: редактирование персональных данных и смена пароля;
- Администрирование: редактирование данных Пользователя, изменение Ролей и добавление новых Пользователей;
- Должники: просмотр данных должников, формирование соглашений о рассрочке, формирование уведомлений о задолженности; формирование дел (предварительный пакет заявлений в суд); ручной ввод сведений о делах сформированных вне Программы «ПИР»; формирование пакета заявлений в суд, обновление информации о деле; отображение списка сформированных пакетов документов с возможностью выгрузки; удаление дела или событий по делу.
- Реестр госпошлины: импорт и экспорт реестра, информация по статьям НК РФ о размерах государственной пошлины;
- Отчеты: формирование и выгрузка отчетов.

2 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: Python, C#, ADO.NET Entity Framework, Net Core, PostgreSQL, HTML, JavaScript;
- Опыт работы с jQuery
- Опыт разработки API для веб приложений;
- Отличные знания. NET, C#, Python;
- 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.

3 Системно-технические характеристики

- Операционная система Debian 10, Windows 10,
- СУБД PostgreSQL 15,
- HTTP-сервер Nginx,
- Платформа Node.JS 16,
- NPM 8.1,
- Платформа Python 3.11,
- Менеджер процессов PM2,
- Платформа контроля версий Gitlab,
- Платформа развёртывания ПО Gitlab CI/CD,
- Frontend framework Vue.js 2,
- КриптоПро,
- Stunnel,
- сервис для создания, тестирования, документирования, публикации и обслуживания API Postman,
- Мониторинг обновления журнала ошибок через Zabbix.

3.1 Технические требования

Технические требования для серверов представлены в таблице 1.

Таблица 1

Сервер БД	**Требования**
Количество процессоров/ ядер	8
Тактовая частота процессора	от 2.3 ГГц
Количество выделяемых вычислительных потоков	8
Необходимый объем оперативной памяти	16 Гб
Необходимый объем дискового пространства	100+ Гб
Количество серверов БД	от 1
СУБД	PostgreSQL, версия от 14 и выше
Доступы	10.3.24.9 для получения адресной структуры ФИАС
Сервер сервисов	**Требования**
Количество процессоров/ ядер	4
Тактовая частота процессора	от 2.3 ГГц
Количество выделяемых вычислительных потоков	4

Необходимый объем оперативной памяти	8 Гб
Объем дискового пространства	от 200 Гб
Доступы	– к серверу БД, – к сервису dadata https://support.dadata.ru/knowledge-bases/4/articles/2026-spisok-ip-adresov-i-domenov, – к серверу с сервисом подписания запросов – к ГИС ЖХК.
ПО	– NodeJS 16.13, – npm 8.1, – Python 3.1, – libreoffice-base, – libreoffice-calc, – libreoffice-core, – libreoffice-math, – libreoffice- pdfimport, – libreoffice-writer
Сервер для сервиса формирования и подписания xml запросов для ГИС ЖКХ	
Количество процессоров/ ядер	1
Тактовая частота процессора	от 2 ГГц
Необходимый объем оперативной памяти	4 ГБ
Объем дискового пространства	50 Гб
ПО	– Windows 10, – КриптоПро 5, – КриптоПро .NET, – Stunnel
Другие требования	**Требования**
Параметры для мониторинга	доступность серверов по сети (ping); наличие свободного места на дисках серверов не менее 10% от объёма; доступность сайтов на серверах приложений;
Требования к ЗИ	– система обнаружения вторжений; – защита среды виртуализации; – СЗИ от несанкционированного доступа.

4 Установка и настройка ПО

4.1 Сервер БД

4.1.1 Подготовка системы

Установить утилиты sudo:

```
apt install sudo
```

Обновить имеющиеся пакеты:

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

Установить необходимые пакеты для возможности установки и работы Postgresql 14:

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

4.1.2 Установка Postgresql 14

Добавить репозиторий 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 репозитория:

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

Обновить индексации пакетов apt:

```
sudo apt -y update
```

Установить Postgresql 14:

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

Установить дополнительные библиотеки:

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

4.1.3 Установка расширения pg_rational

Требуется скачать с https://github.com/begriffs/pg_rational репозиторий удобным способом в любую директорию на сервер и из этой директории установить расширение:

```
make
sudo make install
```

4.1.4 Создание БД

Добавить роли pir_deployer, pir_api, gis_gkh и pir_import из файла **pir_roles.sql**:

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

Создать схемы gar, gis_gkh, import, public таблиц, комментариев, ограничений, функций, триггеров из файла **pir_schema.sql**:

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

Наполнить таблицы словарей, типов из файла **pir_dictionaries.sql**:

```
psql -h localhost -d instance_preprod -U pir_deployer -f
/home/vadim/pir_dictionaries.sql
```

4.2 Сервер сервисов

4.2.1 Подготовка системы

Установить утилиты sudo:

```
apt install sudo
```

Обновить имеющиеся пакеты:

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

Установить пакеты LibreOffice для работы сервисов генерации документов:

```
sudo apt install -y libreoffice-core libreoffice-base libreoffice-calc
libreoffice-core libreoffice-math libreoffice-pdfimport libreoffice-writer
```

4.2.2 Node.JS

Установить необходимые пакеты для установки и работы Node.JS:

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

Установить менеджер версий Node.JS:

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

Установить Node.JS 16:

```
nvm install 16
```

4.2.3 Python

Установить необходимые пакеты для установки и работы Python:

```
sudo apt install wget build-essential libncursesw5-dev libssl-dev
libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev libffi-dev zlib1g-
dev
```

Загрузить исходный код Python:

```
wget https://www.python.org/ftp/python/3.11.3/Python-3.11.3.tgz
```

Разархивировать его в систему:

```
tar xzf Python-3.11.3.tgz
```

Затем подготовить код к компиляции. Для этого необходимо перейти в директорию Python 3.11.3 и воспользоваться **./configure:**

```
cd Python-3.11.3
./configure --enable-optimizations
```

Скомпилировать готовый код. С помощью параметра altinstall запретить компилятору переопределить версию Python:

```
make altinstall
```

4.2.4 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:

Установить требуемые системные зависимости:

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

Затем установить 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* добавить 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:

```
user  pir;
worker_processes  1;

error_log  /var/log/nginx/error.log crit;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
     use                 epoll;
     multi_accept        on;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
"$request_filename" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    log_format  bytes   '$body_bytes_sent';

    #access_log  /var/log/nginx/access.log  main;
    access_log off;

    sendfile                        on;
    tcp_nopush                      on;
     client_header_timeout          1m;
    client_body_timeout            1m;
    client_header_buffer_size      2k;
    client_body_buffer_size        256k;
    client_max_body_size           256m;
    large_client_header_buffers    4   8k;
    send_timeout                   30;
    keepalive_timeout              60 60;
    reset_timedout_connection      on;
    server_tokens                  off;
    server_name_in_redirect        off;
    server_names_hash_max_size     512;
    server_names_hash_bucket_size  512;

    # Compression
    gzip               on;
    gzip_comp_level    5;
    gzip_min_length    512;
    gzip_buffers       8 64k;
    gzip_types         text/plain text/css text/javascript text/js
text/xml application/json application/javascript application/x-javascript
application/xml application/xml+rss application/x-font-ttf image/svg+xml
font/opentype;
    gzip_proxied       any;
```

```
    gzip_disable        "MSIE [1-6]\.";

    # Proxy settings
    proxy_redirect      off;
    proxy_set_header    Host            $host;
    proxy_set_header    X-Real-IP       $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass_header   Set-Cookie;
    proxy_connect_timeout   90;
    proxy_send_timeout  90;
    proxy_read_timeout  90;
    proxy_buffers       32 4k;

     # Cache settings
    proxy_cache_path /var/cache/nginx levels=2 keys_zone=cache:10m
inactive=60m max_size=1024m;
    proxy_cache_key "$host$request_uri $cookie_user";
    proxy_temp_path  /var/cache/nginx/temp;
    proxy_ignore_headers Expires Cache-Control;
    proxy_cache_use_stale error timeout invalid_header http_502;
    proxy_cache_valid any 1d;

    # Cache bypass
    map $http_cookie $no_cache {
        default 0;
        ~SESS 1;
        ~wordpress_logged_in 1;
    }

    # File cache settings
    open_file_cache          max=10000 inactive=30s;
    open_file_cache_valid    60s;
    open_file_cache_min_uses 2;
    open_file_cache_errors   off;

    include /etc/nginx/conf.d/*.conf;
}
```

4.3 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.

4.3.1 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: 10.3.2.101);
- TARGET_PATH– Путь, по которому будет располагаться директория с сервисом (пример: /home/pir/pir-backend/api2);
- TARGET_USER– Имя пользователя для подключения к серверу (пример: pir).

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 с помощью менеджера процессов pm2.

Файл .gitlab-ci.yml выглядит следующим образом:

```
stages:
  - build
  - deploy

build:
  stage: build
  only:
    - api_v2
    - dev
  before_script:
    - nvm install 12
    - npm ci --cache .npm --prefer-offline
  script:
    - if [ "$CI_COMMIT_REF_NAME" == "dev" ]; then npm run build-dev; else
npm run build; fi
  artifacts:
    paths:
      - dist/*
  cache:
    key: $CI_COMMIT_REF_SLUG
    paths:
      - .npm/
```

```
.deploy_template: &deploy_template
  stage: deploy
  before_script:
    - eval $(ssh-agent -s)
    - echo "$PIR_SSH_KEY" | tr -d '\r' | ssh-add -
    - ssh-keyscan $TARGET_HOST >> ~/.ssh/known_hosts
  script:
    - rsync -avz --delete-after ./dist/*
$TARGET_USER@$TARGET_HOST:$TARGET_PATH

deploy nofrb:
  only:
    - api_v2
  <<: *deploy_template
  environment:
    name: production/nofrb
    url: https://pir.brsc.ru/
```

4.3.2 Сервис API

Переменные

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

- TARGET_HOST – Адрес сервера (пример: 10.3.2.101);
- TARGET_PATH– Путь, по которому будет располагаться директория с сервисом, (пример: /home/pir/pir-backend/api2);
- TARGET_USER– Имя пользователя для подключения к серверу (пример: pir);
- DOTENV – Будет скопирована в файл .env. и должна содержать следующие значения:
  1) Строка для подключения Prisma ORM к базе данных: https://www.prisma.io/docs/reference/database-reference/connection-urls DATABASE_URL="postgresql://pir_api:pass@localhost:5432/instance_preprod?schema=public" # dev-new-life
  2) Путь к директории static: STATIC="/home/pir/pir-backend/static/"
  3) Значение секретного ключа для генерации и проверки паролей: SECRET="secret"
  4) Токен Gitlab, необходим для скачивания npm пакетов из приватного репозитория: NPM_GITLAB_TOKEN="glpat-9_jxQP1wciZ57cgyypYa"
  5) Токен для обращения к сервису dadata: DADATA_TOKEN="1234542d28fb7d0f84448vf4bny076e24b56ab9"
  6) Данные об организации: ACCOUNTS_PROVIDER_NAME="НОФ Региональный оператор РБ"

Непрерывная интеграция

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

На стадии сборки происходит:

- Загрузка npm пакетов;
- Генерация 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:

```
stages:
  - build
  - deploy

cache:
  untracked: true
  key: $CI_COMMIT_REF_SLUG
  paths:
    - node_modules/

build:
  stage: build
  only:
    - master
    - /^dev($|\/.)/
  before_script:
    - nvm install 16
    - |
      {
        echo "@ao-brsc:registry=https://gitlab.com/api/v4/packages/npm/"
        echo
"//gitlab.com/api/v4/packages/npm/:_authToken=$BRSC_GITLAB_NPM_TOKEN"
      } >> .npmrc
    - echo "$DOTENV" > .env
  script:
    - npm i --omit-dev
    - npx prisma generate
    - npm run build
  artifacts:
    paths:
      - dist/

.deploy_template: &deploy_template
  stage: deploy
  before_script:
    - eval $(ssh-agent -s)
    - echo "$PIR_SSH_KEY" | tr -d '\r' | ssh-add -
    - ssh-keyscan $TARGET_HOST >> ~/.ssh/known_hosts
    - echo "$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 pir-api 2>&1) 1> /dev/null ;
then pm2 start ; fi
          pm2 save
      "

deploy nofrb:
  only:
    - master
  <<: *deploy_template
  environment:
    name: production/nofrb
    url: https://pir-api.brsc.ru/
```

4.3.3 Сервис формирования пакета документов

Сервис формирования пакета документов состоит из нескольких подсервисов генерации следующих документов:

- Пакет досудебных претензий,
- Пакет предварительных документов в суд,
- Пакет документов в суд,
- Отчеты по различным формам.

Для каждого подсервиса запускается отдельный процесс в pm2.

Переменные

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

- DOTENV, содержит переменные окружения: HOST-адрес хоста для API сервиса, PORT – порт по которому работает API сервиса;
- TARGET_HOST – Адрес сервера (пример: 10.3.2.101);
- TARGET_PATH– Путь, по которому будет располагаться директория с сервисом (пример: /home/pir/create_docx_python);
- TARGET_USER– Имя пользователя для подключения к серверу (пример: pir).

Непрерывная интеграция

Непрерывная интеграция сервиса формирования пакета документов состоит из 2 этапов: сборки (build) и развертывании (deploy).

На стадии сборки происходит:

- Загрузка pip пакетов

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:

```
stages:
  - build
  - deploy

cache:
  untracked: true
  key: $CI_COMMIT_REF_SLUG
  paths:
    - env/
    - app/assets/

build:
  stage: build
  only:
    - master
    - dev
  before_script:
    - python3 -m pip install --user --upgrade pip
    - python3 -m pip install --user virtualenv
    - python3 -m venv env
    - source env/bin/activate
    - which python
    - pip3 install --upgrade wheel
    - pip3 install --upgrade setuptools
  script:
    - pip3 install -r requirements.txt
    # Патч jinja2
    # https://gitlab.brsc.ru/pir/backend/docs-export-service/-
/blob/master/templater.py#L3
    - sed -i 's/ctx = self\.new_context(dict(\*args, \*\*kwargs))/ctx =
self.new_context(vars= args[0], shared=True)/g' env/lib/python$( python3 -V
| sed -E 's/^.* ([0-9]+\.[0-9]+).*/\1/' )/site-
packages/jinja2/environment.py
    - python3 ./app/utils/template_variables_from_comments.py

.deploy_template: &deploy_template
  stage: deploy
  before_script:
    - eval $(ssh-agent -s)
    - echo "$PIR_SSH_KEY" | tr -d '\r' | ssh-add -
    - ssh-keyscan $TARGET_HOST >> ~/.ssh/known_hosts
    - echo "$DOTENV" > .env
  script:
    - "rsync -avz --exclude .git ./
$TARGET_USER@$TARGET_HOST:/home/pir/create_docx_python"
    - |
        ssh $TARGET_USER@$TARGET_HOST <<"EOF"
            cd /home/pir/create_docx_python
            python3.11 -m venv env
            source env/bin/activate
            which python3.11
```

```
            declare -A services

            services[pdf-service]=create_pdf.py
            services[docx-service]=create_docx.py
            services[lawsuit-claim-
service]=lawsuit_claim_generator_service.py
            services[notification-
service]=notifications_generator_service.py
            services[executions-
service]=create_court_decisions_executions.py
            services[report-service]=report_service.py
            services[template-api-service]=api_service.py

            for service in "${!services[@]}"; do
              echo "start or restart ${service} ${services[${service}]}"
              if grep 'not found' <(pm2 reload "${service}" --kill-timeout
8500 2>&1) 1> /dev/null ; then pm2 start python3.11 --name "${service}" --
"${services[${service}]}" ; fi
            done

            pm2 save
        EOF

deploy nofrb:
  only:
    - master
  <<: *deploy_template
  environment:
    name: production/nofrb
    url: https://pir-api.brsc.ru/
```

4.3.4 Адаптер взаимодействия с внешними системами

Адаптер взаимодействия с внешними системами отслеживает наличие новых архивов с данными о задолженностях по лицевым счетам и истории начислений, обрабатывает эти данные и обновляет их в базе данных.

Переменные

Для интеграции адаптера и обработки выписок необходимо создать следующие переменные для каждого сервера на который будет разворачиваться система:

- TARGET_HOST – Адрес сервера (пример: 10.3.2.101);
- DOTENV, содержит переменные окружения: HOST-адрес хоста для API сервиса, PORT – порт по которому работает API сервиса;
- APPCONFIG – содержит конфигурацию подключения к базе данных и расположение архивов с реестрами данных. Пример значения:

```
{
    "postgres": {
    "host":     "10.3.26.121",
    "port":     "5432",
```

```
            "db":       "instance_nofrb",
            "login":    "pir_deployer",
            "password": "x7iOK9rfzS"
          },
            "reesters_dir": "opt/work/ObmenPIR/"
        }
```

Непрерывная интеграция

Непрерывная интеграция состоит из 2 этапов: сборки (build) и развертывании (deploy).

На стадии сборки происходит:

– Загрузка pip пакетов.

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:

```
stages:
  - build
  - deploy

cache:
  untracked: true
  key: $CI_COMMIT_REF_SLUG
  paths:
    - ./.venv/

build:
  stage: build
  only:
    - master
    - /^dev(-.+)?$/
  before_script:
    - python3 -m pip install --user --upgrade pip
    - python3 -m pip install --user poetry
    - poetry shell
    - which python
  script:
    - poetry install

deploy production:
  stage: deploy
  only:
    - master
  script:
    - echo "$DOTENV_PRODUCTION" > .env
    - "rsync -avz --exclude .git ./
pir@10.3.2.119:/home/pir/reester_parser"
    - |
        ssh pir@10.3.2.119 "
            cd /home/pir/reester_parser
```

```
            poetry shell
            which python3.11
        "
  environment:
    name: production
    url: https://pir-api.brsc.ru/
```

4.4 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.

4.4.1 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 – необходимо скачать в браузере и через WinSCP перенести вручную на сервер в каталог пользователя. И дальше продолжать с команды dpkg.

4.4.2 Настройка Zabbix

На сервере, который добавляем на Zabbix, необходимо сделать следующее:

1. Зайти на /etc/zabbix/zabbix_agentd.conf

2. Добавить в пунктах и ServerActive, адрес сервера Zabbix:

Server=ИП_АДРЕС,

ServerActive=ИП_АДРЕС,

Hostname=ИМЯ_ХОСТА,

3. 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: нажать Select и выбрать к какой группе будет относиться host,

Interfaces: прописать IP Address или DNS. Выбрать один из двух.

- В разделе Templates:

нажать Select, выбрать темплейт (Template OS Linux by Zabbix agent) -> Select.

4.4.3 Удаление Zabbix

sudo apt-get remove zabbix-agent
sudo apt-get purge --auto-remove zabbix-agent
rm -rf /etc/zabbix/*

5 Установка программы ПИР

Установка Программы «Претензионно-исковая работа» осуществляется следующим образом:

1. GitLab хранит исходный код.
2. Во время разворачивания GitLab Runner осуществляет скачивание из интернета или из подготовленной папки зависимости/библиотеки (пакет).
3. GitLab Runner собирает в проект исходный код и пакет.
4. GitLab Runner переносит проект на рабочий сервер.

Сборка и разворачивание проекта осуществляется по сценарию описанном в .gitlab-ci.yml.