«Intellect Process Platform»

Руководство администратора приложений

ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ ТЕРМИНОВ

Термин	Описание
ИС (Платформа)	Intellect Process Platform
ПО	Программное обеспечение
БД	База Данных
СУБД	Система управления базами данных
УЗ	Учетная запись

ЛИСТ ИЗМЕНЕНИЙ

Дата	Версии	Описание	Автор

Содержание

1 ВВЕДЕНИЕ

1.1 Назначение документа

Настоящее Руководство предназначено для администраторов приложений системы Intellect Process Platform, которые будут инсталлировать / деинсталлировать, настраивать и сопровождать Intellect Process Platform.

1.2 Версия Intellect Process Platform

Данное Руководство относится к первой версии системы Intellect Process Platform в конфигурации «Кредитный конвейер». Возможны незначительные расхождения между Руководством и последующими версиями программы.

1.3 Актуальность документа

Документ актуален с момента согласования до внесения следующего изменения.

2 СИСТЕМНЫЕ ТРЕБОВАНИЯ

2.1 Требования к аппаратному обеспечению

Сервер приложений	
Процессор	CPU, содержащий не менее 8 – х ядер с рабочей тактовой частотой не ниже 2.4 GHz для всех ядер одновременно
Жесткий диск	2xHDD / SSD 500 Gb
Оперативная память	DDR4 суммарным объёмом не менее 32 Gb
RAID	RAID1 / RAID10
Сетевой адаптер	Ethernet 1000 Mbit/s
Сервер базы данных	
Процессор	CPU, содержащий не менее 4 – х ядер с рабочей тактовой частотой не ниже 2.4 GHz для всех ядер одновременно
Жесткий диск	2xHDD / SSD 500 Gb
Оперативная память	DDR4 суммарным объёмом не менее 16 Gb
RAID	RAID1 / RAID10
Сетевой адаптер	Ethernet 1000 Mbit/s
Сервер Kafka	
Процессор	CPU, содержащий не менее 4 – х ядер с рабочей тактовой частотой не ниже 2.4 GHz для всех ядер одновременно
Жесткий диск	2xHDD / SSD 1000 Gb
Оперативная память	DDR4 суммарным объёмом не менее 32 Gb

RAID	RAID1 / RAID10
Сетевой адаптер	Ethernet 1000 Mbit/s
Клиентские рабочие места	
Процессор	CPU, содержащий не менее 2 – х ядер с рабочей тактовой частотой не ниже 1.3 GHz для всех ядер одновременно
Оперативная память	DDR3 или выше, суммарным объёмом не менее 8 Gb
Жесткий диск	1xHDD / SSD 100 Gb
Монитор	Рекомендуемая диагональ – 21", разрешение Full HD – 1920x1080
Сетевой адаптер	Ethernet 100 Mbit/s

2.2 Требования к программному обеспечению

Тип ПО	Наименование ПО	Использование
Операционная система	Linux RHEL/CentOS	Все сервера
Брокер сообщений	Kafka 3.0.0 Confluent Schema Registry 7.2.0	Для сервера Kafka
СУБД	Postgresql 9.6-12	Для сервера БД
Офисное ПО	Google Chrome	Для рабочих станций
ПО сервера приложений	Java 11 Nginx >= 1.17 Keycloak >=12	Для сервера приложений

3 НАЗНАЧЕНИЕ СИСТЕМЫ

Основное назначение системы Intellect Process Platform – это, например, внедрение автоматизированной информационной системы (далее – ИС) для организации поточного кредитования клиентов различных банков, другими словами, кредитного конвейера, целями которого может являться:

- Автоматизация деятельности банка в части кредитных процессов, процесса выдачи средств клиенту (YTM – Yes To Money), а также в частности постановка задач и контроль исполнения процессов и подпроцессов кредитования;
- Обновление процесса разработки для различных продуктов банка (кредит и кредитная линия с лимитом выдачи и лимитом задолженности, банковская гарантия, аккредитив).

4 ИНСТАЛЛЯЦИЯ INTELLECT PROCESS PLATFORM

Под инсталляцией программного обеспечения подразумевается процесс сборки микросервисов и упаковка в rpm пакеты с помощью gitlab-runner с последующей автоматической установкой пакетов. Прежде чем приступить к инсталляции программного обеспечения, необходимо скорректировать конфигурацию микросервисов (настройка Intellect Process Platform). Подразумевается, что на момент развертывания ПО платформы выполнены установка и настройка всех необходимых компонентов (документ «Руководство системного администратора»).

4.1 Установка серверной части

Сборки и развертывание в Gitlab

Пример файла gitlab-ci.yaml

```
variables:
  SASS_BINARY_PATH: /home/gitlab-runner/linux-x64-83_binding.node
  CYPRESS_INSTALL_BINARY: /home/gitlab-runner/cypress.zip
  ANSIBLE_VAULT_PASSWORD_FILE: ~/pass_file
  GIT_STRATEGY: clone


stages:
  - build
  - package
  - deploy


build_frontend:
  stage: build
  only:
    variables:
      - $CI_PROJECT_NAME == "frontend"
  script:
  - echo Updating configuration for target environment...
  - ansible-playbook /etc/ansible/playbooks/update_files.yml --extra-vars "repository=$CI_PROJECT_NAME"
  - npm config set registry $NPM_REGISTRY
  - npm install -g @angular/cli --legacy-peer-deps
  - npm install --save-dev @angular-devkit/build-angular --legacy-peer-deps
  - npm run build --prod
  tags:
  - runner-1
  artifacts:
    name: $CI_COMMIT_TAG
    expire_in: 7 days
    paths:
      - ./dist
```

```
build_backend:
  stage: build
  only:
    variables:
      - $CI_PROJECT_NAME == "testserver-sale-platform"
  script:
  - echo Updating configuration for target environment...
  - ansible-playbook /etc/ansible/playbooks/update_files.yml --extra-vars "repository=$CI_PROJECT_NAME"
  - mvn clean package
  tags:
  - runner-1
  artifacts:
    name: $CI_COMMIT_TAG
    expire_in: 7 days
    paths:
      - ./sale-platform-modules/*/target/*.jar
      - ./sale-platform-modules/elib-service/*/target/*.jar
      - ./sale-platform-modules/integration/*/target/*.jar
      - ./sale-platform-modules/integration/*/*/target/*.jar
      - ./sale-platform-modules/task-manager/*/target/*.jar

build-rpm-backend:
 stage: package
 tags:
  - build
 needs:
  - build_backend_kk
 script:
  - rpmbuild -bb "$BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/testserverSalePlatform.spec"

build-rpm-frontend:
 stage: package
 tags:
  - build
 needs:
  - build_front_kk
 script:
  - rpmbuild -bb "$BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/testserverSalePlatform.spec"

upload_frontend_kk:
```

```
  stage: upload artifacts
  only:
    variables:
      - $CI_PROJECT_NAME == "frontend"
  script:
  - 'sed -i "s/"private": true/"private": false/g" package.json'
  - cp package.json dist
  - npm publish ./dist --registry https://okd-nexus-tst1.headoffice.testserver.local:5000/repository/npm-hosted/
  tags:
  - runner-1
  dependencies:
  - build_frontend_kk
  allow_failure: true



deploy_frontend:
 dependencies:
   - build_frontend-front
 only:
   variables:
   - $CI_PROJECT_NAME == "frontend"
  script:
   - echo 'Deploy job'
   - pushd "$BUILD_SCRIPTS_DIR/testserver-sale-platform/playbooks"
   - ansible-playbook -i inventories/uploadfront.yml

deploy-backend:
  stage: deploy
   dependencies:
   - build-rpm-backend
   only:
     variables:
       - $CI_PROJECT_NAME == "testserver-sale-platform"
  script:
   - echo 'Deploy job'
   - pushd "$BUILD_SCRIPTS_DIR/testserver-sale-platform/playbooks"
   - ansible-playbook -i inventories/upload.yml
```

Spec файл для сервисов

```
Name: testserverSalePlatform
Version: 1.0.0
```

Release: 1

Summary: testserverSalePlatform

Group: Networking/Other

License: Public Joint-Stock Company Название организации/сервера PJSC

URL: https://www.testserver.ru

AutoReq: no

%description

testserverSalePlatform Services

%install

pushd $CI_PROJECT_DIR

mvn clean package

mkdir -p %{buildroot}/opt/testserver-sale-platform/

mkdir -p %{buildroot}/etc/nginx/conf.d/testserver-sale-platform/

mkdir -p %{buildroot}/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/sale-platform-modules/*/target/*.jar $RPM_BUILD_ROOT/opt/testserver-sale-platform/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/sale-platform-modules/elib-service/*/target/*.jar $RPM_BUILD_ROOT/opt/testserver-sale-platform/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/sale-platform-modules/integration/*/target/*.jar $RPM_BUILD_ROOT/opt/testserver-sale-platform/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/sale-platform-modules/integration/*/*/target/*.jar $RPM_BUILD_ROOT/opt/testserver-sale-platform/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/sale-platform-modules/task-manager/*/target/*.jar $RPM_BUILD_ROOT/opt/testserver-sale-platform/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/admin-service.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/camunda.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/core.service

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/elib-testserver-provider.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/spring-boot-admin.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/role-matrix.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/address-provider.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/credit-registry.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/Master Organization System.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/sap-hr.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/opendata.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/svetophor.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/messages.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/limit-server-integration.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/email-notification.service $RPM_BUILD_ROOT/etc/systemd/system/

cp $BUILD_SCRIPTS_DIR/testserver-sale-platform/templates/notification-generator.service $RPM_BUILD_ROOT/etc/systemd/system/

%files

/opt/testserver-sale-platform/

/etc/systemd/system/

/etc/nginx/conf.d/testserver-sale-platform/

%clean

rm -rf $RPM_BUILD_ROOT

Upload.yaml для установки пакетов и запуска сервисов

```
---
- name: Upload services
  gather_facts: no
  hosts: app_servers
  tasks:
    - name: Empty publish directory
      ansible.builtin.file:
        path: "{{pub_dir_path}}"
        state: absent
    - name: Make publish directory
      ansible.builtin.file:
        path: "{{pub_dir_path}}"
        state: directory
        mode: '0755'
    - name: Copy packages to publish directory
      ansible.builtin.copy:
        src: "{{item}}"
```

```
    dest: "{{pub_dir_path}}"
  with_fileglob:
    - /home/testserver-sale-platform/rpmbuild/RPMS/x86_64/testserverSalePlatform-*
  register: copy_outcome
- name: stop services
  ansible.builtin.command:
    cmd: "sudo systemctl stop {{item}}"
  args:
    warn: no
  loop: "{{pkg_list}}"
  changed_when: false
- name: Temporary job Remove package
  ansible.builtin.command:
    cmd: "sudo yum -y remove testserverSalePlatform"
  ignore_errors: yes
- name: Install packages
  ansible.builtin.command:
    cmd: "sudo yum -y install {{item.dest | basename}}"
    chdir: "{{ pub_dir_path }}"
  args:
    warn: no
  loop: "{{copy_outcome.results}}"
  loop_control:
    label: "{{item.dest}}"
- name: start services
  ansible.builtin.command:
    cmd: "sudo systemctl start {{item}}"
  args:
    warn: no
  loop: "{{pkg_list}}"
  changed_when: false
  ignore_errors: yes
- name: Wait for 10 seconds
  ansible.builtin.wait_for:
    timeout: 10
- name: Check services state
  ansible.builtin.command:
    cmd: "sudo systemctl status {{item}}"
  args:
    warn: no
  loop: "{{pkg_list}}"
```

```
    changed_when: false
    ignore_errors: yes
```

Пример сервиса для system

```
[Unit]
Description=testserver Sale Platform Core Service
After=NetworkManager.service

[Service]
WorkingDirectory=/opt/testserver-sale-platform/
ExecStart=/opt/testserver-sale-platform/
Restart=always
RestartSec=30
KillSignal=SIGINT
User=root
Group=root

[Install]
WantedBy=multi-user.target
```

Пример скрипта создания БД и ролей на сервере Postgres

```
#!/bin/bash
psql -v ON_ERROR_STOP=1 --username postgres <<-EOSQL

 CREATE ROLE core WITH
 LOGIN
 NOSUPERUSER
 INHERIT
 NOCREATEDB
 NOCREATEROLE
 NOREPLICATION
 PASSWORD 'пароль для роли core';

 CREATE DATABASE база_core
  WITH OWNER = core
      ENCODING = 'UTF8'
      TABLESPACE = pg_default
      LC_COLLATE = 'en_US.UTF8'
      LC_CTYPE = 'en_US.UTF8'
      CONNECTION LIMIT = -1;
 GRANT CONNECT, TEMPORARY ON DATABASE база_core TO public;
```

```
  GRANT ALL ON DATABASE база_core TO core;

  create role core_r with
    nologin
    nosuperuser
    inherit
    nocreatedb
    nocreaterole
    noreplication;

  create role core_rw with
    nologin
    nosuperuser
    inherit
    nocreatedb
    nocreaterole
    noreplication;

EOSQL

psql -v ON_ERROR_STOP=1 --username core база_core <<-EOSQL

  CREATE SCHEMA sp
      AUTHORIZATION core;

  GRANT ALL ON SCHEMA sp TO core;
  GRANT USAGE ON SCHEMA sp TO core_r;
  GRANT USAGE ON SCHEMA sp TO core_rw;

  alter default privileges in schema sp
      grant select, insert, update, delete on tables to core_rw;
  alter default privileges in schema sp
      grant select on tables to core_r;
EOSQL

psql -v ON_ERROR_STOP=1 --username postgres база_core <<-EOSQL
  CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
EOSQL

# camunda-db
```

```
psql -v ON_ERROR_STOP=1 --username postgres <<-EOSQL

  CREATE ROLE роль_camunda WITH
    LOGIN
    NOSUPERUSER
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'пароль для роли camunda';

  CREATE DATABASE camunda_db
    WITH OWNER = camunda
    ENCODING = 'UTF8'
    TABLESPACE = pg_default
    LC_COLLATE = 'en_US.UTF8'
    LC_CTYPE = 'en_US.UTF8'
    CONNECTION LIMIT = -1;

EOSQL

psql -v ON_ERROR_STOP=1 --username postgres camunda_db <<-EOSQL
  CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
EOSQL

# keycloak-db
psql -v ON_ERROR_STOP=1 --username postgres <<-EOSQL

  CREATE ROLE keycloak WITH
    LOGIN
    NOSUPERUSER
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'пароль роли keycloak';

  CREATE DATABASE keycloak_db
    WITH OWNER = keycloak
    ENCODING = 'UTF8'
    TABLESPACE = pg_default
```

```
    LC_COLLATE = 'en_US.UTF8'
    LC_CTYPE = 'en_US.UTF8'
    CONNECTION LIMIT = -1;

  GRANT ALL ON DATABASE keycloak_db TO keycloak;

EOSQL

psql -v ON_ERROR_STOP=1 --username postgres keycloak_db <<-EOSQL
  CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
EOSQL

# msg_db
psql -v ON_ERROR_STOP=1 --username postgres <<-EOSQL

  CREATE ROLE msg_u WITH
    LOGIN
    NOSUPERUSER
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'пароль роли msg_u';

  CREATE DATABASE msg_db
    WITH OWNER = msg_u
    ENCODING = 'UTF8'
    TABLESPACE = pg_default
    LC_COLLATE = 'en_US.UTF8'
    LC_CTYPE = 'en_US.UTF8'
    CONNECTION LIMIT = -1;

  GRANT CONNECT, TEMPORARY ON DATABASE msg_db TO public;

  GRANT ALL ON DATABASE msg_db TO msg_u;
EOSQL

psql -v ON_ERROR_STOP=1 --username msg_u msg_db <<-EOSQL
  CREATE SCHEMA msg AUTHORIZATION msg_u;

  GRANT ALL ON SCHEMA msg TO msg_u;
```

```
EOSQL

# company schema
psql -v ON_ERROR_STOP=1 --username postgres <<-EOSQL

  CREATE ROLE company_rw WITH
    LOGIN
    NOSUPERUSER
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'пароль роли company_rw';

EOSQL

psql -v ON_ERROR_STOP=1 --username postgres <<-EOSQL
  GRANT company_rw TO core;
EOSQL

psql -v ON_ERROR_STOP=1 --username postgres база_core <<-EOSQL

  CREATE SCHEMA company
      AUTHORIZATION core;

  GRANT ALL ON SCHEMA company TO core;

  GRANT USAGE ON SCHEMA company TO company_rw;

  alter default privileges in schema company
      grant select, insert, update, delete on tables to company_rw;

EOSQL

# company-assistant schema
psql -v ON_ERROR_STOP=1 --username core база_core <<-EOSQL

  CREATE SCHEMA company_assistant
      AUTHORIZATION core;

  GRANT ALL ON SCHEMA company_assistant TO core;
```

```
  GRANT USAGE ON SCHEMA company_assistant TO core_r;

  GRANT USAGE ON SCHEMA company_assistant TO core_rw;

  alter default privileges in schema company_assistant
     grant select, insert, update, delete on tables to core_rw;

  alter default privileges in schema company_assistant
     grant select on tables to core_r;
EOSQL

# tm schema
psql -v ON_ERROR_STOP=1 --username postgres <<-EOSQL

  CREATE ROLE task_manager WITH
    LOGIN
    NOSUPERUSER
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'пароль роли manager';

EOSQL

psql -v ON_ERROR_STOP=1 --username postgres база_core <<-EOSQL

  CREATE SCHEMA tm
     AUTHORIZATION core;

  GRANT ALL ON SCHEMA tm TO core;
  alter default privileges in schema tm
     grant select, insert, update, delete on tables to core;

EOSQL

# role-matrix schema
psql -v ON_ERROR_STOP=1 --username core база_core <<-EOSQL

  CREATE SCHEMA role_matrix
```

```
    AUTHORIZATION core;

  GRANT ALL ON SCHEMA role_matrix TO core;

  GRANT USAGE ON SCHEMA role_matrix TO core_r;

  GRANT USAGE ON SCHEMA role_matrix TO core_rw;

  alter default privileges in schema role_matrix
    grant select, insert, update, delete on tables to core_rw;

  alter default privileges in schema role_matrix
    grant select on tables to core_r;
EOSQL
```

4.2 Установка клиентской части

Дополнительное программное обеспечение на клиентских машинах не требуется.

5 НАСТРОЙКА INTELLECT PROCESS PLATFORM

5.1 Настройка серверной части

Настройка конфигурации микросервисов

Auth

```
auth.rolematrix.domain.url: ${AUTH_ROLEMATRIX_DOMAIN_URL:http://адрес:порт_сервиса_role-matrix}
spring:
  security:
    oauth2:
      resourceserver:
        jwt:
          issuer-uri: ${REST_SECURITY_ISSUER_URI:http://адрес:порт_keycloak/auth/realms/angular-web}
      client:
        registration:
          keycloak:
            client-id: ${REST_SECURITY_CLIENT_ID:клиент в реалме angular-web}
            client-secret: ${REST_SECURITY_CLIENT_SECRET:42ffe6bd-9d8f-4874-b4b5-d41b09ceac00} (id клиента)
            authorization-grant-type: password
            username: ${REST_SECURITY_USERNAME:системный_пользватель}
            password: ${REST_SECURITY_PASSWORD:пароль_системного_пользователя}
        provider:
          keycloak:
```

token-uri: ${REST_SECURITY_TOKEN_URI:http://**адрес:порт_keycloak**/auth/realms/**angular-web**/protocol/openid-connect/token}

Address Provider

```
spring:
  application:
    name: address-provider-service
  main:
    web-application-type: servlet
  boot:
    admin:
      client:
        instance:
          name: address-provider-service
          service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-service
:${server.port:порт_admin-service}}
admin-service
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_address-provider}

springdoc:
  api-docs:
    enabled: true
  swagger-ui:
    path: swagger-ui

logging:
  level:
    ru.testserver.integration.dadata: debug

info:
  app:
    version: @buildNumber@

dadata:
  client:
    token: ${DADATA_CLIENT_TOKEN:токен_клиента_dadata}
    base-url: ${DADATA_CLIENT_URL:https://suggestions.dadata.ru/suggestions/api/4_1/rs}
    timeout: 3s
    proxy:
      host: ${DADATA_PROXY_HOST:}
      port: ${DADATA_PROXY_PORT:}
      login: ${DADATA_PROXY_LOGIN:}
      password: ${DADATA_PROXY_PASSWORD:}

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always
```

Admin Service

```
server:
  port: порт_сервиса
logging:
  level:
    ru.testserver.saleplatform: DEBUG
    org.springframework: INFO
```

spring:
  boot:
    admin:
      client:
        instance:
          name: admin-service
          service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http:// **адрес_admin-service**:${server.port:**порт_admin-service**}}
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://**адрес:порт_admin-service** }
  application:
    name: admin-service
  kafka:
    producer:
      bootstrap-servers: ${SPRING_KAFKA_PRODUCER_BOOTSTRAP_SERVERS:**адрес:порт kafka producer**}
      user-update-keycloak-topic: ${SPRING_KAFKA_PRODUCER_USER_UPDATE_KEYCLOAK_TOPIC:**user-update-keycloak-topic**}

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always

springdoc:
  api-docs:
    enabled: true
    path: api-docs
  swagger-ui:
    enabled: true
    path: swagger-ui

keycloak:
  username: ${KEYCLOAK_USERNAME:**логин системного пользователя keycloak**}
  password: ${KEYCLOAK_PASSWORD:**пароль системного пользователя keycloak** }
  token-url: ${KEYCLOAK_TOKEN_URL:http://**адрес:порт_keycloak**/auth/realms/**angular-web** /protocol/openid-connect/token}
  users-url: ${KEYCLOAK_USERS_URL:http://**адрес:порт_keycloak**/auth/admin/realms/ **angular-web**/users} (**angular-web – реалм созданный в keycloak**)
  page-size: ${KEYCLOAK_PAGE_SIZE:3}
  db-url: ${KEYCLOAK_DB_URL:jdbc:postgresql://**адрес:порт_сервера_БД/db_keycloak**}
  db-user: ${KEYCLOAK_DB_USER:**имя_УЗ_бд_keycloak**}
  db-password: ${KEYCLOAK_DB_PASSWORD:**пароль_УЗ_db_keycloak**}
core:
  client:
    url: ${CORE_URL:http://**адрес:порт_сервиса_core**}
Master Organization System-integration:
  client:
    url: ${Master Organization System_INTEGRATION_URL:http://**адрес:порт_сервиса_Master Organization System_integration**}

Master Organization System Integration

server:
  port: **порт_сервиса**
sap:
  security:
    username: ${SAP_SECURITY_USERNAME:root}
    password: ${SAP_SECURITY_PASSWORD:admin}

```
spring:
  boot:
    admin:
      client:
        instance:
          name: Master Organization System-integration-service
          service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http:// адрес_admin-service
:${server.port:порт_admin-service}}
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_Master Organization System_integration}
  cloud:
    stream:
      function:
        definition: proces-
sAc-
coun-
tRestrictions;processAccountSearch;processClientManagement;processClientDataOpenCard;processClientDataListXml
Result;processClientDataList
      bindings:
        processAccountRestrictions-in-0:
          destination: account-restrictions-request
        processAccountRestrictions-out-0:
          destination: account-restrictions-response

        processAccountSearch-in-0:
          destination: account-search-request
        processAccountSearch-out-0:
          destination: account-search-response

        processClientManagement-in-0:
          destination: client-management-request
        processClientManagement-out-0:
          destination: client-management-response

        processClientDataOpenCard-in-0:
          destination: client-data-open-card-request
        processClientDataOpenCard-out-0:
          destination: client-data-list-xml

        processClientDataListXmlResult-in-0:
          destination: client-data-list-xml
        processClientDataListXmlResult-out-0:
          destination: client-data-list

        processClientDataList-in-0:
          destination: client-data-list-request
        processClientDataList-out-0:
          destination: client-data-list-result-request
      kafka:
        streams:
          binder:

            configuration:
              schema.registry.url: http://localhost:8081
              default.key.serde: org.apache.kafka.common.serialization.Serdes$StringSerde
              default.value.serde: io.confluent.kafka.streams.serdes.avro.SpecificAvroSerde

management:
  endpoint:
    health:
      enabled: true
      show-details: always
      probes:
```

```
      enabled: true
    bindings:
      enabled: true
  endpoints:
    web:
      exposure:
        include: health,info,prometheus

role-matrix:
  client:
    url: ${ROLE_MATRIX_URL:http://адрес:порт_сервиса_role_matrix }

springdoc:
  api-docs:
    path: api-docs
  swagger-ui:
    path: swagger-ui
sap:
  security:
    username: логин_УЗ_SAP_PO
    password: пароль_УЗ_SAP_PO
  Master Organization System:
    account-restrictions:
      uri: http://sap-po-dev testserv-
er.ru:8000/XISOAPAdapter/MessageServlet?senderParty=&senderService=BS_LOANPIPEKB_DEV&receiverParty=
&receiverService=&interface=si_so_FileRestricrtionClientAccountGet&interfaceNamespace=urn:testserver.ru:PipeKB
Loan
    accountsearch:
      uri: http://sap-po-
dev.testserver.ru:8000/XISOAPAdapter/MessageServlet?senderParty=&senderService=BS_LOANPIPEKB_DEV&rec
eiverParty=&receiverService=&interface=si_so_AccountSearch&interfaceNamespace=urn:testserver.ru:AccountSearch
    client-management:
      uri: http://sap-po-
dev.testserver.ru:8000/XISOAPAdapter/MessageServlet?senderParty=&senderService=BS_LOANPIPEKB_DEV&rec
eiverPar-
ty=&receiverService=&interface=si_so_ClientDataManagement&interfaceNamespace=urn:testserver.ru:ClientDataMa
nagement
    client-data-list:
      uri: http://sap-po-
dev.testserver.ru:8000/XISOAPAdapter/MessageServlet?senderParty=&senderService=BS_LOANPIPEKB_DEV&rec
eiverPar-
ty=&receiverService=&interface=si_ao_ClientDataListKKKB&interfaceNamespace=urn:testserver.ru:ClientDataMana
gement
    client-data-open-card:
      uri: http://sap-po-
dev.testserver.ru:8000/XISOAPAdapter/MessageServlet?senderParty=&senderService=BS_LOANPIPEKB_DEV&rec
eiverPar-
ty=&receiverService=&interface=si_so_ClientDataOpenCardGet&interfaceNamespace=urn:testserver.ru:ClientDataMa
nagement

spring:
  cloud:
    stream:
      kafka:
        streams:
          binder:
            configuration:
              schema.registry.url: ${KAF-
KA_STREAMS_SCHEMA_REGISTRY_URL:http://адрес:порт_schema_registry}
              default.key.serde: org.apache.kafka.common.serialization.Serdes$StringSerde
              default.value.serde: io.confluent.kafka.streams.serdes.avro.SpecificAvroSerde
            brokers: ${KAFKA_STREAMS_BROKERS:адрес:порт_kafka_producer}
```

Camunda BPM

```
spring:
  application:
    name: camundabpm
  task:
    execution:
      pool:
        core-size: ${TASK_EXECUTION_POOL_CORE_SIZE:8}
        max-size: ${TASK_EXECUTION_POOL_MAX_SIZE:32}
  datasource:
    url: ${POSTGRES_ADDRESS:jdbc:postgresql://адрес:порт_сервера_БД/db_camunda}
    username: ${POSTGRES_USER:имя_УЗ_бд_camunda}
    password: ${POSTGRES_PASSWORD:пароль_ УЗ_бд_camunda}
  jpa:
    generate-ddl: false
    properties:
      hibernate:
        jdbc:
          lob:
            non_contenxtual_creation: true
  kafka:
    bootstrap-servers: ${KAFKA_BOOTSTRAP_SERVER:адрес:порт_kafka_producer}
    consumer:
      group-id: ${KAFKA_CONSUMER_GROUP_ID:group_id_camunda_topics}
      process:
        topic: ${KAFKA_CONSUMER_PROCESS_TOPIC:process_consumer_topic}
    producer:
      common:
        topic: ${KAFKA_PRODUCER_COMMON_TOPIC:common_topic}
      process:
        topic: ${KAFKA_PRODUCER_PROCESS_TOPIC:process_topic}
  boot:
    admin:
      client:
        instance:
          name: camunda-bpm
          service-base-url:          service-base-url:
${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-service
:${server.port:порт_admin-service}}
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_camunda_bpm}
multipart:
  maxFileSize: ${MULTIPART_MAX_FILE_SIZE:100Mb}
  maxRequestSize: ${MULTIPART_MAX_REQUEST_SIZE:100Mb}

springdoc:
  api-docs:
    enabled: true
    path: api-docs
  swagger-ui:
    enabled: true
    path: swagger-ui

camunda:
  process:
    name: ${CAMUNDA_PROCESS_NAME:stage_01_initialization}
  bpm:
    webapp:
      index-redirect-enabled: true
      admin-user:
        id: ${CAMUNDA_BPM_ADMIN_USER_ID:логин_УЗ_администратора}
        password: ${CAMUNDA_BPM_ADMIN_USER_PASSWORD:пароль_УЗ_администратора}
```

```
feign:
  client:
    config:
      role-matrix:
        url: ${ROLE_MATRIX_SERVICE:http://адрес_сервиса_role-matrix/api}
      core:
        url: ${CORE_SERVICE:http://адрес:порт_сервиса_core}

logging:
  level:
    root: info
    org.springframework: info
    ru.testserver: debug

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always

schema:
  registry:
    url: ${SCHEMA_REGISTRY_URL:http://адрес:порт_schema_registry}

info:
  app:
    version: ^project.version^

svetophor-integration:
  client:
    url: ${SVETOPHOR_INTEGRATION_URL:http://адрес:порт_сервиса_svetofor}
```

Core

```
server:
  port: порт_сервиса

lombok:
  addLombokGeneratedAnnotation: true
  anyConstructor:
    addConstructorProperties: true

app:
  liquibase:
    run-after-app-start: true

feign:
  client:
    config:
      camunda-bpm:
        url: ${CAMUNDA_BPM_SERVICE:http://адрес:порт_сервиса_camunda-bpm}
      credit-registry-integration:
        url: ${CREDIT_REGISTRY_INTEGRATION:http://адрес:порт_сервиса_credit_registry}

spring:
  servlet:
    multipart:
      maxFileSize: 200MB
```

```
    maxRequestSize: 200MB
  mvc:
    throw-exception-if-no-handler-found: true
  application:
    name: sp-core
  datasource:
    url: ${POSTGRES_ADDRESS:jdbc:postgresql://адрес:порт_сервера_БД/db_core}
    username: ${POSTGRES_USER: имя_УЗ_db_core}
    password: ${POSTGRES_PASSWORD: пароль_УЗ_db_core}
    driver-class-name: org.postgresql.Driver
  jpa:
    properties:
      hibernate:
        dialect: ru.testserver.saleplatform.core.hibernate.dialect.PostgreSQL10DialectCustom
        jdbc.lob.non_contenxtual_creation: true
        temp.use_jdbc_metadata_defaults: false
        validator.apply_to_ddl: false
        default_schema: sp
        cache:
          use_second_level_cache: false
          use_query_cache: false
#       show_sql: true
#       use_sql_comments: true
#       format_sql: true
    hibernate:
      ddl-auto: none
  session:
    store-type: jdbc
    jdbc:
      initialize-schema: always
      schema: classpath:session-jdbc.sql
      #org/springframework/session/jdbc/schema-postgresql.sql
      table-name: SPRING_SESSION
  liquibase:
    change-log: ${LIQUIBASE_CHANGELOG_FILEPATH:changelog/db.master-develop-data-changelog.yaml}
    enabled: ${LIQUIBASE_ENABLED:true}
  boot:
    admin:
      client:
        instance:
          name: core
            service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-
service:${server.port:порт_admin-service}}
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_сервиса_core}
  kafka:
    producer:
      bootstrap-servers:
${SPRING_KAFKA_PRODUCER_BOOTSTRAP_SERVERS:http://адрес:порт_kafka_producer}
      key-serializer: org.apache.kafka.common.serialization.StringSerializer
      value-serializer: io.confluent.kafka.serializers.KafkaAvroSerializer
    consumer:
      bootstrap-servers:
${SPRING_KAFKA_CONSUMER_BOOTSTRAP_SERVERS:http://адрес:порт_kafka_consumer}
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      group-id: ${SPRING_KAFKA_CONSUMER_CORE_GROUP_ID:sp_core_group}
      auto-offset-reset: earliest
      calendar-topic: ${SPRING_KAFKA_CONSUMER_CALENDAR_TOPIC:calendar_topic}
      limit-server-topic: ${SPRING_KAFKA_CONSUMER_LIMIT_SERVER_TOPIC:limit_server_topic}

springdoc:
  api-docs:
    enabled: true
```

```
    path: api-docs
  swagger-ui:
    enabled: true
    path: swagger-ui

logging:
  level:
    root: ${LOGGING_LEVEL_ROOT:info}
    org.springframework: ${LOGGING_LEVEL_ORG_SPRINGFRAMEWORK:info}
    org.keycloak.authorization: ${LOGGING_LEVEL_ORG_KEYCLOAK_AUTHORIZATION:DEBUG}
    org.keycloak.adapters: ${LOGGING_LEVEL_ORG_KEYCLOAK_ADAPTERS:DEBUG}
    ru.testserver.saleplatform: ${LOGGING_LEVEL_RU_TESTSERVER_SALEPLATFORM:DEBUG}

elib-service:
  client:
    url: ${ELIB_SERVICE_URL:http://адрес_сервиса_электронного_досье}
    documents-url: ${ELIB_DOCUMENTS_URL:http://адрес_сервиса_электронного_досье/document}

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always

opendata:
  integration:
    url: ${OPENDATA_INTEGRATION_URL:http://адрес:порт_сервиса_opendata_integration}

schema:
  registry:
    url: ${SCHEMA_REGISTRY_URL:http://адрес:порт_schema_registry }
loanApplication:
  id:
    topic: ${LOAN_APPLICATION_ID_TOPIC:loanApplication}

role-matrix:
  find-user-by-id: ${ROLE_MATRIX_FIND_USER_BY_ID:http://адрес:порт_сервиса_role-matrix/api/user/}
  get-role-list-by-user-id: ${ROLE_MATRIX_GET_ROLE_LIST_BY_USER_ID:http://адрес:порт_сервиса_role-
matrix/api/role/user/}
  client:
    url: ${ROLE_MATRIX_URL:http://адрес:порт_сервиса_role-matrix }

info:
  app:
    version: ^project.version^

Master Organization System-integration:
  client:
    url: ${ Master Organization System_INTEGRATION_URL:http://адрес:порт_Master Organization System-
integration }

applications:
  mask:
    loan: ${LOAN_APPLICATION_MASK:КБ-setLim-}
    credit-product: ${CREDIT_PRODUCT_APPLICATION_MASK:КБ-credit-}
    tranche: ${TRANCHE_APPLICATION_MASK:КБ-tranche-}
    loan-change: ${LOAN_CHANGE_APPLICATION_MASK:КБ-chLim-}
    tech: ${TECH_APPLICATION_MASK:КБ-tech-}

sla:
```

```
  work-time:
    start-time: ${WORK_START_TIME:10:00:00}
    end-time: ${WORK_END_TIME:18:00:00}

bki:
  permission-document-nomenclature-id: ${PERMISSION_DOCUMENT_NOMENCLATURE_ID:c4e4f7fb-700b-
4698-861f-72f6e6add942}
  reports-url-template: ${REPORTS_TEMPLATE_URL:https://spine2/credit_registry/showReport.html?reportCode=}
```

Credit Registry Integration

```
server:
  port: порт_сервиса

spring:
  boot:
    admin:
      client:
        instance:
          name: credit-registry-integration-service
            service-base-url:          service-base-url:
${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-
service:${server.port:порт_admin-service}}
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_credit-registry-ingegration
credit-registry:
  url: ${CREDIT_REGISTRY_URL:http://cre-
test01.headoffice.testserver.local:8080/credit_registry/xservices/ConnectorService}
(адрес сервиса CreditRegestry)
  auth:
    username: ${CREDIT_REGISTRY_AUTH_USERNAME:логин_УЗ_CreditRegestry }
    password: ${CREDIT_REGISTRY_AUTH_PASSWORD:пароль_УЗ_CreditRegestry }

logging:
  level:
    root: info
    org.springframework: info
    ru.testserver.saleplatform: debug

springdoc:
  api-docs:
    enabled: true
    path: api-docs
  swagger-ui:
    enabled: true
    path: swagger-ui

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always
```

Elib testserver Provider

```
spring:
  servlet:
    multipart:
      maxFileSize: 200MB
      maxRequestSize: 200MB
  application:
```

```
    name: elib-provider-service
  main:
    web-application-type: servlet
  datasource:
    driverClassName: org.postgresql.Driver
    url: jdbc:postgresql://${POSTGRES_ADDRESS:адрес}:порт/${POSTGRES_DB:имя_бд_elib_provider}
    username: ${POSTGRES_USER:имя_УЗ_elib_provider }
    password: ${POSTGRES_PASSWORD: пароль_УЗ_elib_provider }
  jpa:
    hibernate:
      ddl-auto: validate
    properties:
      hibernate:
        dialect: org.hibernate.dialect.PostgreSQL10Dialect
        jdbc.lob.non_contenxtual_creation: true
        temp.use_jdbc_metadata_defaults: false
        validator.apply_to_ddl: false #disable Java Bean constraint propagation to DDL.
        default_schema: elib
        cache.use_second_level_cache: false
        cache.use_query_cache: false
        format_sql: true
  boot:
    admin:
      client:
        instance:
          name: elib-testserver-provider-service
            service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-
service:${server.port:порт_admin-service}}
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_elib-testserver-provider
  liquibase:
    change-log: classpath:/changelog/db.master-changelog.yaml

springdoc:
  api-docs:
    enabled: true
    path: api-docs
  swagger-ui:
    path: swagger-ui

feign:
  client:
    config:
      default:
        loggerLevel: basic
      elib:
        url: ${ELIB_SERVICE_URL:http://адрес_сервиса_электронного_досье/elib/api/service}
        urlAuth: ${ELIB_SERVICE_AUTH_URL:http://адрес_сервиса_электронного_досье/elib/api}
        username: ${ELIB_SERVICE_USERNAME:логин УЗ электронного досье }
        password: ${ELIB_SERVICE_PASSWORD:пароль УЗ электронного досье }
logging:
  level:
    ru:
      testserver: debug

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always
```

SAP HR Integration

```
logging:
  level:
    ru.testserver.saleplatform: DEBUG
    org.springframework: INFO

spring:
  boot:
    admin:
      client:
        instance:
          name: integration-sap_hr
          service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-service:${server.port:порт_admin-service}}
admin-service
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_sap-hr-integration}
  application:
    name: sap-hr-service
  kafka:
    producer:
      bootstrap-servers: ${SPRING_KAFKA_PRODUCER_BOOTSTRAP_SERVERS:адрес:порт kafka producer}
      user-update-topic: ${SPRING_KAFKA_PRODUCER_USER_UPDATE_TOPIC:user_update_topic}
      calendar-topic: ${SPRING_KAFKA_PRODUCER_CALENDAR_TOPIC:calendar_topic}
      department-topic: ${SPRING_KAFKA_PRODUCER_DEPARTMENT_TOPIC:department_topic}

management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always
```

Limit Server Integration

```
logging:
  level:
    ru.testserver.saleplatform: DEBUG
    org.springframework: INFO

spring:
  boot:
    admin:
      client:
        instance:
          name: integration-limit-server
          service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-service:${server.port:порт_admin-service}}
admin-service
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_limit-server-integration}
  application:
    name: limit-server-service
  kafka:
    producer:
      bootstrap-servers: ${SPRING_KAFKA_PRODUCER_BOOTSTRAP_SERVERS:адрес:порт kafka producer}
      limit-server-topic: ${SPRING_KAFKA_PRODUCER_LIMIT_SERVER_TOPIC:limit_server_topic}

management:
  endpoints:
    web:
      exposure:
```

```
      include: "*"
  endpoint:
    health:
      show-details: always
```

Messages

```
server:
  port: порт_сервиса

spring:
  jpa:
    database: postgresql
    generate-ddl: false
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        jdbc.lob.non_contenxtual_creation: true
        validator.apply_to_ddl: false
        default_schema: msg (схема в бд сервиса messages)
  datasource:
    url: jdbc:postgresql://${POSTGRES_ADDRESS:адрес}:порт/${POSTGRES_DB: бд_сервиса_messages }
    username: ${POSTGRES_USER:логин пароль пользователя бд messages }
    password: ${POSTGRES_PASSWORD:пароль пользователя бд messages}
  liquibase:
    change-log: ${LIQUIBASE_CHANGELOG_FILEPATH:changelog/db.master-develop-data-changelog.yaml}
    enabled: ${LIQUIBASE_ENABLED:true}

app:
  liquibase:
    run-after-app-start: true

springdoc:
  api-docs:
    enabled: true
    path: api-docs
  swagger-ui:
    enabled: true
    path: swagger-ui

logging:
  level:
    root: ${LOGGING_LEVEL_ROOT:info}
    org.springframework: ${LOGGING_LEVEL_ORG_SPRINGFRAMEWORK:info}
    ru.testserver.saleplatform: ${LOGGING_LEVEL_RU_TESTSERVER_SALEPLATFORM:DEBUG}

role-matrix:
  find-user-by-id:
${ROLE_MATRIX_FIND_USER_BY_ID:http://адрес:порт_сервиса_role-matrix/api/user/}
  get-role-list-by-user-id: ${ROLE_MATRIX_GET_ROLE_LIST_BY_USER_ID:http://адрес:порт_сервиса_role-
matrix/api/role/user/}
  client:
    url: ${ROLE_MATRIX_URL:http://адрес:порт_сервиса_role-matrix }
```

Opendata Integration

```
opendata:
  url: ${OPENDATA_URL:
https://wsopendata.headoffice.testserver.local/OD/ODataBasic.svc}
  basic-auth:
    username: ${OPENDATA_USERNAME:логин УЗ Opendata}
    password: ${OPENDATA_PASSWORD:пароль УЗ Opendata}
  auto-complete-inn-result-limit: ${OPENDATA_AUTOCOMPLETE_INN_RESULT_LIMIT:5}
```

```
logging:
  level:
    root: info
    org.springframework: info
    ru.testserver.saleplatform: debug

springdoc:
  api-docs:
    path: api-docs
  swagger-ui:
    path: swagger-ui

spring:
  boot:
    admin:
      client:
        instance:
          name: opendata-integration-service
          service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-service:${server.port:порт_admin-service}}
admin-service
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_opendata-integration}
management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    health:
      show-details: always

server:
  port: ${SERVER_PORT:порт_сервиса}
```

Role Matrix

```
server:
  port: порт_сервиса
springdoc:
  api-docs:
    enabled: true
    path: api-docs
  swagger-ui:
    enabled: true
    path: swagger-ui

lombok:
  addLombokGeneratedAnnotation: true
  anyConstructor:
    addConstructorProperties: true

spring:
  datasource:
    url: ${POSTGRES_ADDRESS:jdbc:postgresql://адрес:порт_сервера_БД/db_core }
    username: ${POSTGRES_USER: имя_УЗ_db_core}
    password: ${POSTGRES_PASSWORD: пароль_УЗ_db_core}
    driver-class-name: org.postgresql.Driver
  jpa:
    properties:
      hibernate:
        dialect: ru.testserver.saleplatform.rolematrix.hibernate.dialect.PostgreSQL10DialectCustom
        jdbc.lob.non_contenxtual_creation: true
```

```
      temp.use_jdbc_metadata_defaults: false
      validator.apply_to_ddl: false #disable Java Bean constraint propagation to DDL.
      default_schema: role_matrix
      cache:
        use_second_level_cache: false
        use_query_cache: false
      format_sql: true
    hibernate:
      ddl-auto: validate
  liquibase:
    change-log: ${LIQUIBASE_CHANGELOG_FILEPATH:classpath:/changelog/db.changelog-master.yaml}
    enabled: ${LIQUIBASE_ENABLED:true}
    default-schema: role_matrix
    liquibase-schema: role_matrix
    user: ${LIQUIBASE_USER:имя_УЗ_db_core }
    password: ${LIQUIBASE_PASSWORD:пароль_УЗ_db_core }

app:
  liquibase:
    run-after-app-start: true

feign:
  client:
    config:
      company-assistant:
        url: ${COMPANY_ASSISTANT_SERVICE:http://localhost:8085/api}
```

Spring Boot Admin Service

```
server:
  port: порт_сервиса

spring:
  application:
    name: spring-boot-admin
```

Svephofor Integration Service

```
server:
  port: ${SERVER_PORT:порт_сервиса}

svetophor:
  url:
${SAP_SVETOPHOR_URI:http://localhost:8080/XISOAPAdapter/MessageServlet?senderParty=&senderService=BS_LOANPIPEKB_DEV&receiverParty=&receiverService=&interface=si_so_TaskCheck&interfaceNamespace=urn:testserver.ru:SecurityClearance}
  user: ${SAP_SVETOPHOR_USER:root}
  password: ${SAP_SVETOPHOR_PASSWORD:admin}

spring:
  cloud:
    stream:
      function:
        definition: processCheckTasks
      bindings:
        processCheckTasks-in-0:
          destination: check-task-request
        processCheckTasks-out-0:
          destination: check-task-response

      kafka:
        streams:
          binder:
            configuration:
```

```
        schema.registry.url:
${SCHEMA_REGISTRY_URL:адрес:порт_schema_registry}
        default.key.serde: org.apache.kafka.common.serialization.Serdes$StringSerde
        default.value.serde: io.confluent.kafka.streams.serdes.avro.SpecificAvroSerde
springdoc:
  api-docs:
    enabled: true
    path: api-docs
  swagger-ui:
    enabled: true
    path: swagger-ui
```

Email Notification

```
server:
  port: порт_сервиса

logging:
  level:
    ru.testserver.saleplatform: debug

spring:
  application:
    name: sp-email-notification-service
  boot:
    admin:
      client:
        instance:
          name: email-notification
          service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-
service:${server.port:порт_admin-service}}
admin-service
        url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_email-notification}
  kafka:
    consumer:
      bootstrap-servers: ${SPRING_KAFKA_CONSUMER_BOOTSTRAP_SERVERS:http://адрес:порт_kafka}
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      group-id: ${SPRING_KAFKA_CONSUMER_NOTIFICATION_GROUP_ID:sp_email_notification_group}
      auto-offset-reset: earliest
      email-notification-topic: ${SPRING_KAFKA_CONSUMER_NOTIFICATION_TOPIC:email_notification_topic}

email:
  enabled: ${EMAIL_NOTIFICATION_ENABLED:true}
  smtpHost: ${EMAIL_SMTP_HOST:адрес_smtp_сервера}
  smtpPort: ${EMAIL_SMTP_PORT:порт smtp}
  username: ${EMAIL_USERNAME:почтовый адрес}
  password: ${EMAIL_PASSWORD:пароль ящика рассылки}
```

Notification-generator

```
server:
  port: порт_сервиса

logging:
  level:
    org.springframework: info
    ru.testserver.saleplatform: debug

spring:
  application:
    name: sp-notification-generator-service
  boot:
    admin:
```

```
    client:
      instance:
        name: notification-generator
        service-base-url: ${SPRING_BOOT_ADMIN_CLIENT_SERVICE_BASE_URL:http://адрес_admin-
service:${server.port:порт_admin-service}}
admin-service
      url: ${SPRING_BOOT_ADMIN_CLIENT_URL:http://адрес:порт_notification_generator}
  kafka:
    consumer:
      bootstrap-servers: ${SPRING_KAFKA_CONSUMER_BOOTSTRAP_SERVERS:http://адрес:порт_kafka }
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      group-id: ${SPRING_KAFKA_CONSUMER_NOTIFICATION_GROUP_ID:sp_notification_generator_group}
      auto-offset-reset: earliest
      application-start-topic:
${SPRING_KAFKA_CONSUMER_APPLICATION_START_TOPIC:application_start_topic}
      task-resolving-topic: ${SPRING_KAFKA_CONSUMER_TASK_RESOLVING_TOPIC:task_resolving_topic}
      stage-finished-topic: ${SPRING_KAFKA_CONSUMER_STAGE_FINISHED_TOPIC:stage_finished_topic}
      task-assignment-topic:
${SPRING_KAFKA_CONSUMER_TASK_ASSIGNMENT_TOPIC:task_assignment_topic}
      application-rejection-topic:
${SPRING_KAFKA_CONSUMER_APPLICATION_REJECTION_TOPIC:application_rejection_topic}
    producer:
      bootstrap-servers:  ${SPRING_KAFKA_PRODUCER_BOOTSTRAP_SERVERS:http://адрес:порт_kafka }
      email-notification-topic:
${SPRING_KAFKA_PRODUCER_EMAIL_NOTIFICATION_TOPIC:email_notification_topic}

role-matrix:
  client:
    url: ${ROLE_MATRIX_URL:http://адрес:порт_role_matrix}

core:
  client:
    url: ${CORE_CLIENT_URL:http://адрес:порт_core }

deeplink:
  application: ${APPLICATION_DEEP_LINK:http://адрес_фронтенда/api/application/{applicationId}}
  task: ${TASK_DEEP_LINK:http://адрес_фронтенда/order/{applicationId}/1/{taskId}}
  company-group: ${COMPANY_GROUP_DEEP_LINK: http://адрес_фронтенда/client/gc/{companyGroupId}}
```

5.2 Настройка клиентской части

Дополнительные настройки на клиентских машинах не требуются.

6 ДЕИНСТАЛЛЯЦИЯ INTELLECT PROCESS PLATFORM

Выполните команду на сервера приложений:

```
yum -y remove testserverSalePlatform
```

будут удалены jar файлы в /opt/testserver-sale-platform/

файлы сервисов платформы в /etc/systemd/system/

7 ЭКСПЛУАТАЦИЯ INTELLECT PROCESS PLATFORM

7.1 Регламент обслуживания Intellect Process Platform

Название процедуры	Время	Комментарий
Процедура резервного копирования	С переодичностью раз в сутки.	См. Раздел 7.3.1
Очистка (VACUUM) СУБД PostgreSQL	Периодичность выбирается на усмотрение лица ответственного за эксплуатацию Intellect Process Platform. Рекомендуется выполнение в периоды наименьшей нагрузки на СУБД Платформы. Процедура очистки выполняется не реже 1 раза в месяц. Для базы объемом ~100 Gb может занимать до 6 часов.	VACUUM высвобождает пространство, занимаемое «мёртвыми» кортежами. При обычных операциях Postgres Pro кортежи, удалённые или устаревшие в результате обновления, физически не удаляются из таблицы; они сохраняются в ней, пока не будет выполнена команда VACUUM. Таким образом, периодически необходимо выполнять VACUUM, особенно для часто изменяемых таблиц.
Обновление статистики планировщика (ANALYZE)	Выполняется автоматически после процедуры очистки. Допускается более частое выполнение обновления статистики при общем замедлении выполнения поисковых запросов су таблицам сущностей имеющим долгий отклик поиска.	Планировщик запросов в Postgres Pro, выбирая эффективные планы запросов, полагается на статистическую информацию о содержимом таблиц. Эта статистика собирается командой ANALYZE, которая может вызываться сама по себе или как дополнительное действие команды VACUUM. Статистика должна быть достаточно точной, так как в противном случае неудачно выбранные планы запросов могут снизить производительность базы данных.

7.2 Требования к рабочему месту

Клиентские рабочие места	
Процессор	CPU, содержащий не менее 2 – х ядер с рабочей тактовой частотой не ниже 1.3 GHz для всех ядер одновременно
Оперативная память	DDR3 или выше, суммарным объёмом не менее 8 Gb
Жесткий диск	1xHDD / SSD 100 Gb
Монитор	Рекомендуемая диагональ – 21", разрешение Full HD – 1920x1080
Сетевой адаптер	Ethernet 100 Mbit/s
ПО	Установленные pgAdmin WinSCP на машинах с ОС семейства Windows, или пакет ssh для unixlike ОС.

7.3 Порядок штатных действий

7.3.1 Бекапирование баз данных платформы

Выполните команды на сервере БД Postgres:

sudo -u postgres pg_dump core_db > core_db.sql

sudo -u postgres pg_dump camunda_db > camunda_db.sql

sudo -u postgres pg_dump keycloak_db > keycloak_db.sql

sudo -u postgres pg_dump msg_db > keycloak_db.sql

7.3.2 Восстановление баз данных платформы

Выполните команды на сервере БД Postgres:

sudo -u postgres psql core_db < core_db.sql

sudo -u postgres psql camunda_db < camunda_db.sql

sudo -u postgres psql keycloak_db < keycloak_db.sql

sudo -u postgres psql msg_db < keycloak_db.sql

7.3.3 Идентификация правильности выполнения работ

Название процедуры	Статус	Комментарий
Процедура резервного копирования	Отсутствие ошибок в stderr, в списке процессов, на сервере остсутствует процесс с именем pg_dump	
Очистка (VACUUM) СУБД PostgreSQL	Отсутствие ошибок в stderr. Пример ошибки: backend> vacuum; < 2019-11-06 14:27:47.556 UTC > ERROR: catalog is missing 3 attribute(s) for relid xxxxx < 2019-11-06 14:27:47.556 UTC > STATEMENT: vacuum;	
Обновление статистики планировщика (ANALYZE)	Отсутствие ошибок в stderr.	

7.4 Порядок действий при нештатных ситуациях

7.4.1 При регламентных процедурах

Во время выполнения процедур резервного копирования, очистки, и обновления статистики в нештатной ситуации требуется:

1. Остановить штатными средствами ОС процесс СУБД
2. Запустить СУБД в однопользовательском режиме (ключ –single -D)
3. Повторить процедуру.

7.4.2 Идентификация правильности выполнения работ

Название процедуры	Статус	Комментарий
Запуск платформы	Проверить корректность работы можно путем входа зарегистрированным в платформе пользователем. Вход выполнен без ошибок, отображаемая пользовательская информация выводится полном объеме, функционал отображается в соответствии с пользовательскими инструкциями. Для проверки интеграционных сервисов требуется выполнить запросы поиска клиентов, и создания ЮЛ (роли описаны в Карте доступа). При поиске клиента (Холдинга/ Группы компаний) на модельном окне должны отобразится результаты поиска в инстансах и платформе. Проверка интеграции с OD (OpenData) производится путем перехода на форму создания ЮЛ в холдинге, ответ приходит при заполнении 10 символов ИНН в соответствующем поле карточки. Успешный результат обращения характеризуется заполнением всех полей карточки ЮЛ. Проверка интеграции с сервисами получения остатков по счетам и лимитам производится переходом в соответствующий раздел на карточке ЮЛ (см. инструкцию пользователя). Данные совпадают с Лимит Сервером и Мастер системой организации (в части остатков). Внимание в системе предусмотрено кеширование. Загрузка изменений (при переходе на карточку ЮЛ) доступна раз в час для ЮЛ. Проверка загрузки данных по новым сотрудникам без выгрузки со стороны через SAP HR силами пользователей платформы. Проверка интеграции с СИАМ Светофор осуществляется в момент подачи новой заявки. Заявку можно отменить сразу после получения данным по проверке.	

7.4.3 Уведомления бизнес пользователям

Кому и в каком порядке пользователям из бизнес-подразделений необходимо знать, что операция не прошла успешно.

ФИО	Должность	Комментарий

7.4.4 Порядок эскалации

Список (а не одну фамилию) горизонтальной эскалации (кого из инженеров подключать к решению проблемы и в какой последовательности), а так же список иерархической эскалации (т.е. стоит ли подключать руководство к решению - если вопрос не решается оперативно, и если "ДА", то кого и когда).

№ п./п.	ФИО/Подразделение/Роль	Должность	Комментарий
1.			
2.			

8 ПРИЛОЖЕНИЕ 1. ЛИСТ СОГЛАСОВАНИЯ

Лист согласования (т.е. фактически, кто на себя берет ответственность, за корректность той информации, которая содержится в документе.)

ФИО	Должность	Описание