

Instrukcja Instalacji Moduł „S”

PROJEKT

Budowa nowoczesnej platformy gromadzenia i analizy danych z Krajowego Rejestru Nowotworów oraz onkologicznych rejestrów narządowych, zintegrowanej z bazami świadczeniodawców leczących choroby onkologiczne (e-KRN+).

METRYKA DOKUMENTU

Stworzony na potrzeby	Projekt e-KRN+
Podstawa prawna	
Wersja	1.4
Ostatnia modyfikacja	
Autor	
Liczba stron	13

MODYFIKACJE

Wersja	Data	Zmodyfikował	Komentarz
1.0			
1.1	27.09	Kamil Jurczyk	Dodanie his.export.package.limit
1.2	28.10	Kamil Jurczyk	Końcowa wersja dokumentu
1.3	12.11.2024	M.Karpinski	Dodanie informacji o zawartości magazynów certyfikatów
1.4	16.01.2025	M.Karpiński	Doprecyzowanie dokumentu w zakresie przygotowania magazynów certyfikatów

Instrukcja

Kroki procedury

Struktura katalogów

Poniżej została umieszczona przykładowa struktura katalogów w formie drzewa w celu łatwiejszego zobrazowania miejsc umieszczenia artefaktów i plików konfiguracyjnych:

```
/DOCKER/
├── docker-compose.yml
├── logs
│   └── modul-s
└── modul-s
    ├── Dockerfile
    └── files
        ├── certs
        │   ├── keystore.jks
        │   └── truststore.jks
        ├── config
        │   ├── application.properties
        │   ├── clientSign.properties
        │   └── logback-modul-s.xml
        └── modul-s-1.0.1-SNAPSHOT.jar
```

Konfiguracja Aplikacji

1. W katalogu przeznaczonym na instalację modułu S np. DOCKER, należy stworzyć strukturę katalogów poleceniami

```
mkdir -p /DOCKER/modul-s/files/certs
mkdir -p /DOCKER/modul-s/files/config
mkdir -p /DOCKER/logs/modul-s
```

2. Należy stworzyć lub zmodyfikować plik docker-compose.yml tak aby zawierał następującą treść:
Plik docker-compose.yml powinien znajdować się w katalogu przeznaczonym na instalację modułu S np. w katalogu DOCKER

```
version: '3.0'
services:

#####
###                                MODUL-S                                ###
#####
modul-s:
  build: ./modul-s
  volumes:
    - ./logs/modul-s:/home/app/logs
  environment:
    - TZ=Europe/Warsaw
  ports:
    - "${IP_MODUL_S}:8080:8080"
  restart: unless-stopped
```

```
hostname: ${HOSTNAME}-modul-s
logging:
  driver: "json-file"
  options:
    max-size: "500M"
    max-file: "3"
```

3. Należy stworzyć lub zmodyfikować plik .env tak by zawierał następującą treść: Plik .env powinien znajdować się w katalogu przeznaczonym na instalację modułu S np. w katalogu DOCKER

```
HOSTNAME=nazwa_hosta
IP_MODUL_S=ip_kontenera
```

gdzie:

nazwa-hosta - nazwa hostname systemu

ip_kontenera - adres ip na którym będzie uruchomiony kontener z aplikacją

4. Należy stworzyć plik Dockerfile o następującej treści lub przekopiować do odpowiedniego katalogu dostarczony plik Dockerfile: Plik Dockerfile powinien znajdować się w katalogu modul-s przeznaczonym na instalację modułu S np. w katalogu /DOCKER/modul-s/

```
FROM openjdk:12.0.2
RUN mkdir -vp /home/app/logs
COPY files/certs/keystore.jks \
    files/certs/truststore.jks \
    files/config/clientSign.properties \
    files/config/logback-spring.xml \
    files/config/application.properties /home/app/
COPY files/modul-s-*.jar /home/app/modul-s.jar
WORKDIR /home/app
EXPOSE 8080
CMD ["java", "-jar", "modul-s.jar"]
```

5. Należy stworzyć plik application.properties o następującej treści lub przekopiować do odpowiedniego katalogu dostarczony plik application.properties: Plik application.properties powinien znajdować się w katalogu config w katalogu przeznaczonym na instalację modułu S np. w katalogu /DOCKER/modul-s/files/config

```
spring.datasource.url=jdbc:h2:file:./data/db
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.open-in-view=false
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=true
```

```
spring.h2.console.enabled=true
spring.liquibase.url=jdbc:h2:file:./data/db
spring.liquibase.change-log=classpath:/db/liquibase-changelog.xml
spring.liquibase.enabled=true
spring.liquibase.contexts=test
spring.servlet.multipart.max-file-size=5MB
spring.servlet.multipart.max-request-size=6MB
logging.level.org.springframework.security=DEBUG
validation.password.minLength=6
validation.password.addComplexityRules=true
logging.pattern.console=%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint}
%clr(${LOG_LEVEL_PATTERN:%5p}) %clr(${PID:- }){magenta} %clr(---){faint}
%clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan} %clr(:){faint}
%clr(%X{user}){yellow} %m%n${LOG_EXCEPTION_CONVERSION_WORD:%wEx}
default.storage.days=30
default.his.overwrite=false
#Docelowy CRON codziennie o północy
#data.erase.cron=0 0 0 */1 * *
#testowy CRON co 15 sekund
data.erase.cron=*/15 * * * * *
client.ssl.key-store-path=./keystore.jks
client.ssl.key-store-password=password
client.ssl.trust-store-path=./truststore.jks
client.ssl.trust-store-password=password
client.ssl.debug-enabled=false
installation.id=${installation.id}
zpro.save.episodes.connectionTimeout=1000
zpro.save.episodes.receiveTimeout=1200000
#zpro.save.episodes.endpoint=http://10.204.0.202:8243/services/zapisEpizodo
w
zpro.save.episodes.endpoint=http://localhost:8290/services/zapisEpizodow
#zpro.save.episodes.endpoint=https://:8448/service/zapisEpizodow
db.config.oracle=jdbc:oracle:thin:@{address}:{port}/{database}
db.config.sqlserver=jdbc:sqlserver://{address}:{port};database={database}
db.config.db2=jdbc:db2://{address}:{port}/{database}
db.config.MYSQL=jdbc:mysql://{address}:{port}/{database}?useLegacyDatetimeC
ode=false&serverTimezone=UTC&useSSL=false
db.config.postgresql=jdbc:postgresql://{address}:{port}/{database}
db.config.h2=jdbc:h2:tcp://{address}:{port}/{database}
db.config.sybase=jdbc:jtds:sybase://{address}:{port}/{database}
db.config.test=jdbc:h2:mem:test
db.config.FAILTEST=jdbc:h2:memerror:test
spring.h2.console.settings.web-allow-others=true
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
logging.config=logback-modul-s.xml
his.export.package.limit=1000
```

gdzie:

his.export.package.limit – limit paczki wysyłanej do HIS

client.ssl.key-store-path – ścieżka do pliku keystore.jks

client.ssl.key-store-password – hasło do pliku keystore.jks

client.ssl.trust-store-path - ścieżka do pliku truststore.jks

client.ssl.trust-store-password - hasło do pliku truststore.jks

installation.id - identyfikator instalacji

6. Należy stworzyć plik logback-modul-s.xml o następującej treści lub przekopiować do odpowiedniego katalogu dostarczony plik logback-spring.xml: Plik logback-spring.xml powinien znajdować się w katalogu config w katalogu przeznaczonym na instalację modułu S np. w katalogu /DOCKER/modul-s/files/config

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="true">
  <property name="LOG_PATH" value="logs"/>
  <property name="APP_CODE" value="modul-s"/>
  <include resource="org/springframework/boot/logging/logback/defaults.xml" />
  <appender name="ROLLING-FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
      <fileNamePattern>${LOG_PATH}/${APP_CODE}_%d{yyyyMMdd}.%i.log</fileNamePattern>
      <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
        <maxFileSize>100MB</maxFileSize>
      </timeBasedFileNamingAndTriggeringPolicy>
      <maxHistory>2</maxHistory>
    </rollingPolicy>
    <encoder>
      <pattern>${FILE_LOG_PATTERN}</pattern>
      <charset>${FILE_LOG_CHARSET}</charset>
    </encoder>
  </appender>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>${CONSOLE_LOG_PATTERN}</pattern>
      <charset>${CONSOLE_LOG_CHARSET}</charset>
    </encoder>
  </appender>
  <logger name="pl.ekrn.moduls.ModulSApplication" level="INFO"/>
  <root level="INFO">
    <appender-ref ref="STDOUT"/>
  </root>
</configuration>
```

7. Należy stworzyć plik clientSign.properties o następującej treści lub przekopiować do odpowiedniego katalogu dostarczony plik clientSign.properties: Plik clientSign.properties powinien znajdować się w katalogu config w katalogu przeznaczonym na instalację modułu S np. w katalogu /DOCKER/modul-s/files/config

```
org.apache.ws.security.crypto.provider=org.apache.ws.security.components.crypto.Merlin
org.apache.ws.security.crypto.merlin.keystore.type=jks
org.apache.ws.security.crypto.merlin.keystore.alias=sign
org.apache.ws.security.crypto.merlin.keystore.password=
org.apache.ws.security.crypto.merlin.keystore.private.password=
org.apache.ws.security.crypto.merlin.keystore.file=
ws-security.self-sign-saml-assertion=true
```

gdzie:

org.apache.ws.security.crypto.merlin.keystore.alias – nazwa aliasu do certyfikatu wss oraz klucza prywatnego podmiotu zainstalowanego zgodnie z instrukcją opisaną w InstrukcjaModułuS.docx

Wypełnienie poniższych parametrów nie jest wymagane, natomiast są niezbędne z punktu widzenia działania biblioteki merlin.

```
org.apache.ws.security.crypto.merlin.keystore.password
org.apache.ws.security.crypto.merlin.keystore.private.password
org.apache.ws.security.crypto.merlin.keystore.file
```

8. Należy przekopiować na serwer otrzymany plik *.jar z aplikacją i umieścić go w katalogu files przeznaczonym na instalację modułu S np. w katalogu /DOCKER/modul-s/files
9. Należy przekopiować na serwer otrzymane certyfikaty i umieścić je w katalogu certs w katalogu przeznaczonym na instalację modułu S np. w katalogu /DOCKER/modul-s/files/certs. Przygotowanie keystore.jks i trustore.jks opisane w sekcji poniżej – Przygotowanie magazynów certyfikatów.

Przygotowanie magazynów certyfikatów

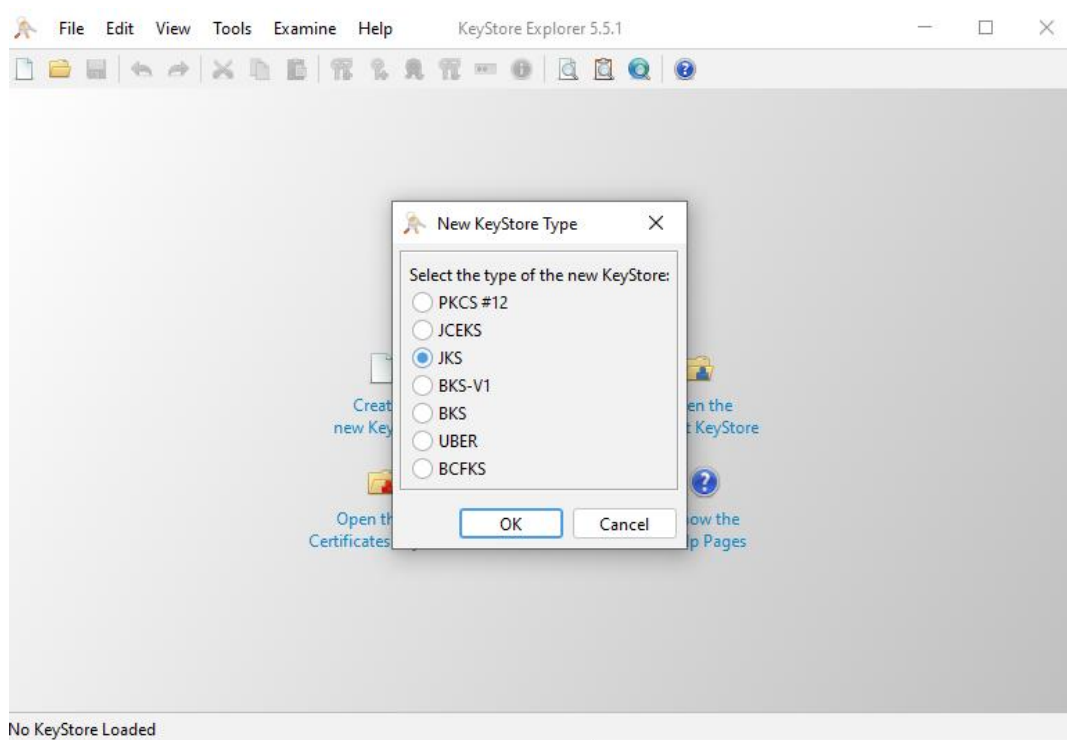
Przygotowanie magazynu certyfikatów trustore.jks

Uwaga: W trustore.jks umieszczamy następujące certyfikaty:

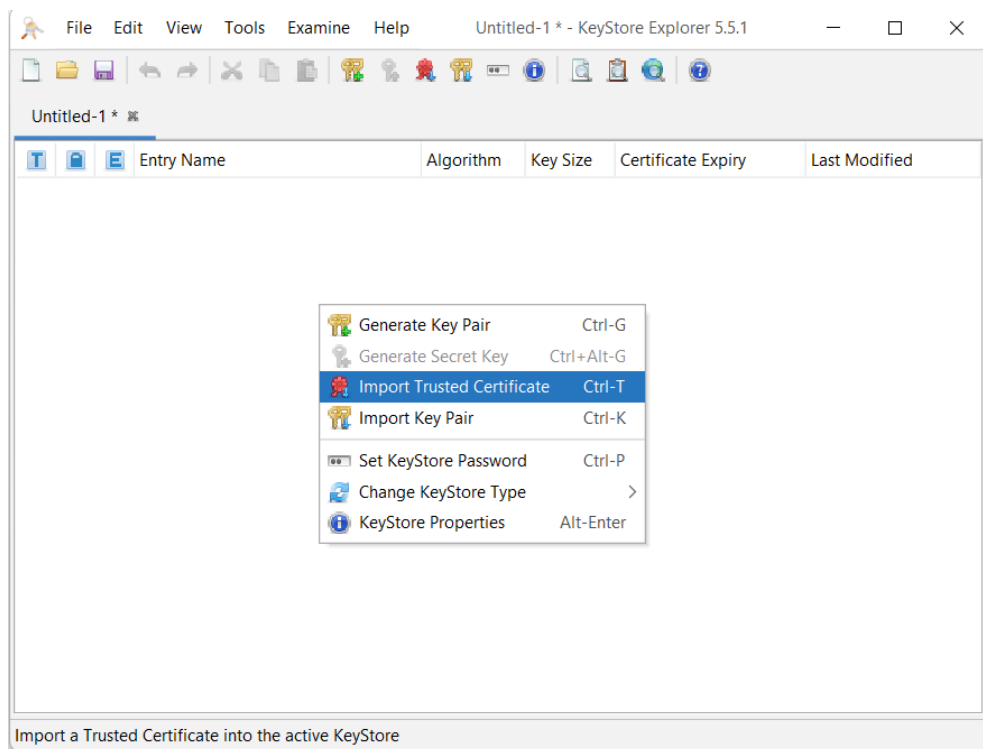
- Certyfikat SSL systemu eKRN wraz z kompletem certyfikatów wystawcy certyfikatu SSL - RootCA, SubCA wystawcy – niezbędnych do weryfikacji systemu eKRN i zestawienia połączenia SSL z systemem eKRN (kroki realizowane w przypadku kiedy moduł S komunikuje się bezpośrednio z systemem eKRN)

Do przygotowania można użyć dowolnego narzędzia: keytool, openssl lub graficznego keystore

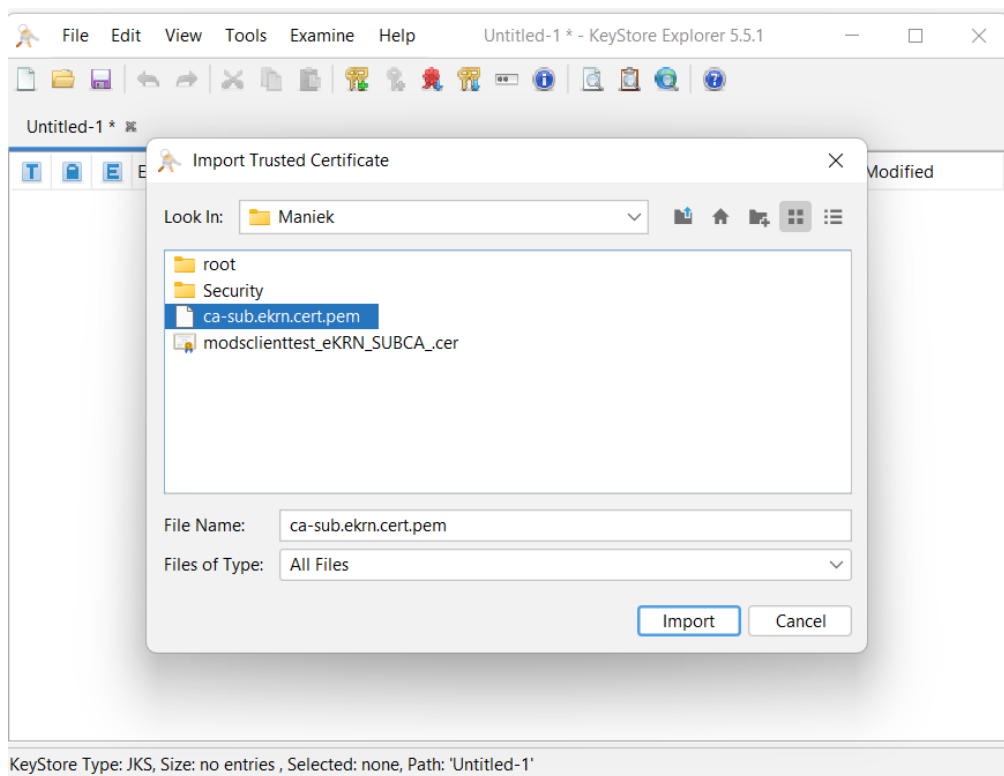
1. Uruchamiamy keystoreExplorer
2. Tworzymy nowy keystore i wybieramy typ jks



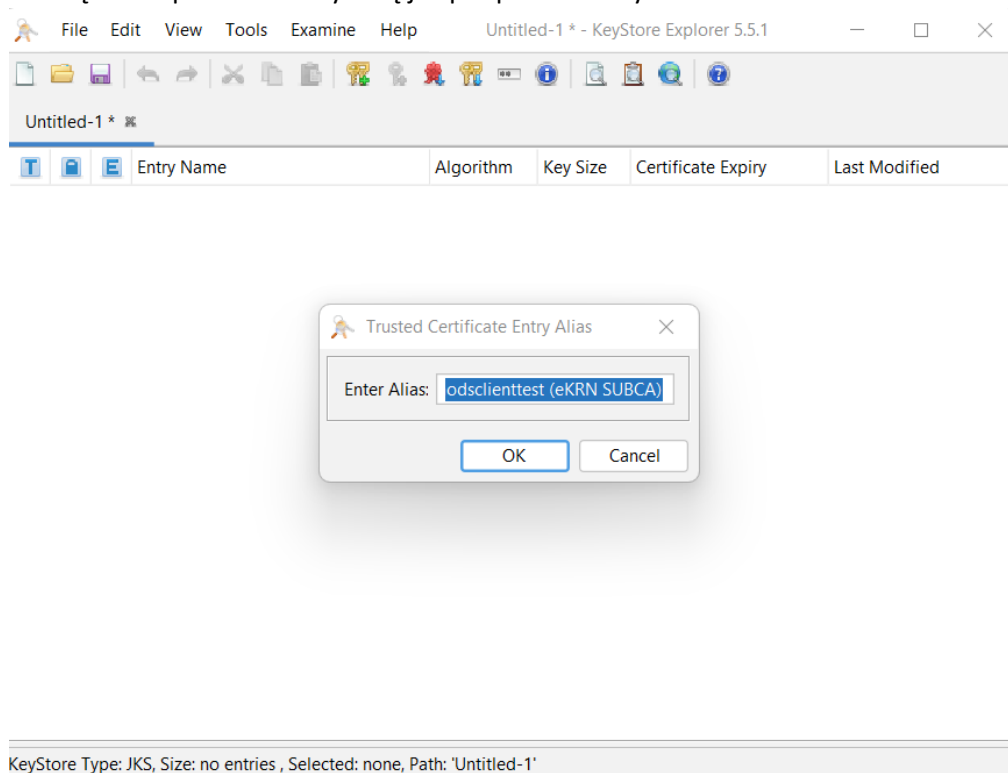
3. Klikamy prawy klawisz myszy i z menu wybieramy opcję importu zaufanego certyfikatu



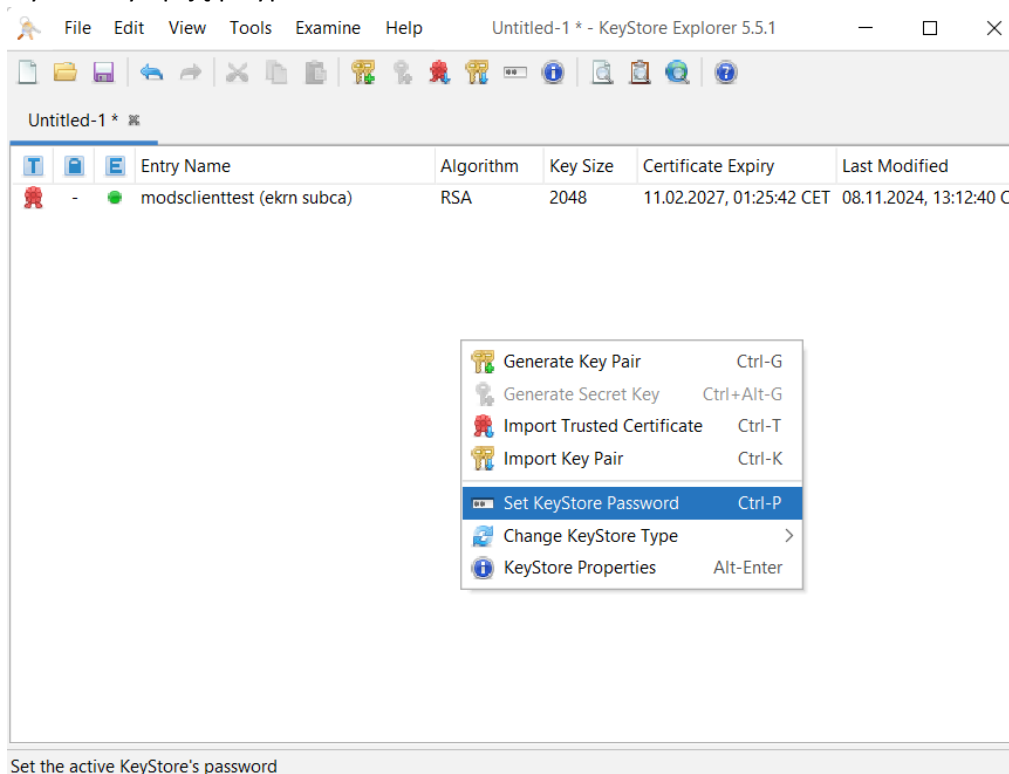
4. Wskazujemy ścieżkę do certyfikatu



4. Nazwę aliasu pozostawiamy taką jak podpowiada keystore



- Kolejny krok to przypisanie hasła do truststora - klikamy prawy klawisz myszy i z menu wybieramy opcję przypisania hasła



- Po poprawnym imporcie certyfikatów zapisujemy truststore na dysku (UWAGA: tu należy pamiętać by odpowiednio skonfigurować wartość parametru: `client.ssl.trust-store-password` oraz `client.ssl.trust-store-path` w pliku `application.properties`)

Przygotowanie magazynu certyfikatów `keystore.jks`

Uwaga: W `keystore.jks` umieszczamy:

- Certyfikat klienta SSL oraz klucz prywatny wystawiony dla podmiotu wystawiony przez system P1

W tym przypadku postępujemy zgodnie instrukcją zawartą w dokumencie „InstrukcjaModułuS.docx – Instalacja nowego klucza i certyfikatu WSS Podmiotu”

Jedyną różnicą jest nazwa aliasu – w przypadku `keystora.jks` zawierającego certyfikat klienta SSL Podmiotu – alias definiujemy jako: `localhost`

Budowanie i uruchomienie aplikacji

- Należy przejść do katalogu w którym umieszczony jest plik `docker-compose.yml` i wykonać budowanie aplikacji:

`docker-compose build modul-s`

- Należy uruchomić `modul-s`, by działał w tle:

`docker-compose up -d modul-s`

3. Do zatrzymania działania kontenera używamy:

`docker-compose stop`

4. Do ponownego uruchomienia kontenera używamy:

`docker-compose stop && docker system prune -a && docker-compose build modul-s && docker-compose up -d modul-s`

Weryfikacja uruchomienia aplikacji

1. Należy zweryfikować czy kontener z aplikacją uruchomił się, w tym celu wykonać polecenie:

`docker ps`

3. Należy zweryfikować logi i sprawdzić czy aplikacja się uruchomiła poprawnie za pomocą poniższego polecenia:

`docker-compose logs -f --tail=2000 modul-s`