RSA Helper wersja 1.2.7

Aplikacja wsparcia dla procesów KSEF

Spis treści

---

1. Ogólne informacje

RSA Helper jest aplikacją webową (WebAPI) wspierającą procesy związane z obsługą e-faktur w systemie KSEF. Aplikacja udostępnia następujące funkcjonalności:

- szyfrowanie zapytań autoryzacyjnych za pomocą algorytmu RSA

- szyfrowanie faktur algorytmem AES-256-CBC
- szyfrowanie paczek faktur (ZIP) algorytmem AES-256-CBC
- generowanie danych kryptograficznych dla sesji interaktywnej KSEF 2.0
- konwertowanie plików e-faktur z formatu XML do HTML
- konwertowanie plików e-faktur z formatu XML do PDF
- walidowanie plików e-faktur w formacie XML
- generowanie kodów QR dla weryfikacji faktur
- obsługa trybu offline (praca bez dostępu do internetu)

---

2. Wymagania

2.1. Wymagania sprzętowe

Do prawidłowego działania aplikacji wymagany jest następujący sprzęt:

- procesor 4-rdzeniowy
- 2GB pamięci operacyjnej
- 500MB przestrzeni dyskowej

2.2. Wymagania oprogramowania

Aplikacja do prawidłowego działania wymaga:

- Serwer WWW:
  - Windows: IIS (rekomendowany) lub Apache
  - Linux: Apache lub Nginx (rekomendowany)
- ASP.NET Core Runtime w wersji 10.0
- Docker/Podman (opcjonalnie, dla konteneryzacji)
- Linux, instalacja tradycyjna (bez kontenera): **przeglądarka Chromium zainstalowana w systemie** — wymaga jej domyślny silnik generowania PDF (patrz sekcja 3.2). Obrazu Docker to nie dotyczy, niesie komplet zależności

2.3. Wymagania dodatkowe

> **UWAGA:** Poniższe wymagania dotyczące dostępu do zewnętrznych serwisów nie obowiązują w trybie offline (patrz sekcja 3.6).

Aplikacja do prawidłowego działania wymaga dostępu do zewnętrznych serwisów Ministerstwa Finansów:

- do generowania danych autoryzacyjnych i wysyłki faktur do KSeF:
  - https://api-demo.ksef.mf.gov.pl
  - https://api-test.ksef.mf.gov.pl
  - https://api.ksef.mf.gov.pl
- do walidacji i wizualizacji plików XML:
  - http://crd.gov.pl

Sprawdzenie dostępności adresów

Aby upewnić się, że serwery KSeF i CRD są dostępne z maszyny, na której zainstalowana jest aplikacja, można wykonać następujące testy:

Windows (PowerShell):

```
Test-NetConnection ksef-test.mf.gov.pl -Port 443
```

Windows (CMD):

```
curl -f -s -o nul -w "%{http_code}" https://ksef-test.mf.gov.pl
```

Linux:

```
curl -f -s -o /dev/null -w "%{http_code}\n" https://ksef-test.mf.gov.pl
```

W przypadku poprawnej dostępności:

- Windows PowerShell: wyświetli `TcpTestSucceeded : True`
- Windows CMD: wyświetli kod statusu HTTP (np. `200`, `301` lub `302`)
- Linux: wyświetli kod statusu HTTP (np. `200`, `301` lub `302`)

Jeśli serwer jest niedostępny, zostanie wyświetlony komunikat o błędzie połączenia.

---

3. Instalacja

3.1. RSA Helper downloads

W przypadku hostowania we własnej infrastrukturze aplikacja dostępna jest w dwóch wersjach:

- **Produkcyjna (KSEF 2.0):** 1.2.6 – dostępna dla windows tutaj
- **Produkcyjna (KSEF 2.0):** 1.2.6 – dostępna dla linux tutaj
- **Rozwój (KSEF 2.0):** 1.2.7 – dostępna dla windows tutaj
- **Rozwój (KSEF 2.0):** 1.2.7 – dostępna dla linux tutaj

> **UWAGA:** Pobranie aplikacji wymaga zalogowania do strefy klienta. Logowanie wymaga jedynie podania adresu e-mail oraz jednorazowego kodu wysłanego na niego — nie wymaga zakładania konta.

Dokumentacja (w języku polskim i angielskim) oraz pozostałe pliki do pobrania dostępne są na stronie z plikami.

3.2. Instalacja na systemach Linux

Instalacja tradycyjna (z serwerem WWW)

Aby uruchomić aplikację w systemie Linux należy zainstalować i skonfigurować serwer WWW. Rekomendowane serwery WWW to:

- **Nginx** – instrukcja instalacji dostępna tutaj
- **Apache** – instrukcja instalacji dostępna tutaj

Dodatkowo należy zainstalować środowisko uruchomieniowe ASP.NET Core w wersji 10. Instrukcja jak zainstalować i skonfigurować ASP.NET Core Runtime dla systemów Linux znajduje się tutaj.

Po zainstalowaniu i konfiguracji serwera WWW należy rozpakować otrzymaną paczkę `RSAHelper.WebAPI.zip` do katalogu głównego serwera WWW.

> **UWAGA:** W przypadku nowej instalacji, po rozpakowaniu paczki w tym samym katalogu należy utworzyć katalog `Data` i wgrać plik konfiguracyjny NIP (patrz sekcja 3.4)

Przeglądarka dla generowania PDF (wymagana)

Domyślny silnik konwersji HTML do PDF (`Syncfusion`) renderuje dokumenty przeglądarką Chromium. **Paczka dla systemów Linux nie zawiera przeglądarki** — jest instalowana w systemie i wskazywana aplikacji ustawieniem `PDFSettings:BlinkPath`. Bez tego kroku generowanie PDF kończy się błędem `Blink files are missing`.

Zalecany jest pakiet `google-chrome-stable` — ta sama instrukcja działa na Ubuntu i na Debianie, a `apt` sam dociąga wszystkie wymagane biblioteki systemowe:

```
cd /tmp
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo apt-get install -y ./google-chrome-stable_current_amd64.deb
google-chrome --version
```

Katalog do wskazania w konfiguracji to `/opt/google/chrome`. Ścieżkę podaje się w `appsettings.json`:

```
"PDFSettings": {
  "BlinkPath": "/opt/google/chrome"
}
```

albo zmienną środowiskową w pliku usługi systemd (nadpisuje `appsettings.json`):

```
Environment=PDFSettings__BlinkPath=/opt/google/chrome
```

Wariant z pakietem `chromium` z repozytorium dystrybucji. Silnik szuka w podanym katalogu pliku wykonywalnego o nazwie dokładnie `chrome`, a pakiet `chromium` instaluje `/usr/lib/chromium/chromium`. Potrzebny jest wtedy dowiązanie symboliczne, a w konfiguracji katalog `/usr/lib/chromium`:

```
sudo apt-get install -y chromium
sudo ln -s /usr/lib/chromium/chromium /usr/lib/chromium/chrome
```

> **UWAGA:** W Ubuntu pakiet `chromium-browser` instalowany przez `apt` to przekierowanie do wersji snap, która jest odcięta od plików spoza katalogu domowego i do tego zastosowania się nie nadaje. W Ubuntu należy użyć `google-chrome-stable`.

Przykładowy plik usługi systemd

Poniższy plik (`/etc/systemd/system/rsaHelper.service`) zawiera komplet ustawień wymaganych do generowania PDF:

```
[Unit]
Description=RSA Helper WebAPI
After=network.target

[Service]
WorkingDirectory=/var/www/html
ExecStart=/usr/bin/dotnet /var/www/html/RSAHelper.WebAPI.dll
Restart=always
RestartSec=10
SyslogIdentifier=rsaHelper
User=www-data
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=ASPNETCORE_URLS=http://127.0.0.1:5000
# Prywatny katalog tymczasowy dla usługi — katalog roboczy przeglądarki powstaje
# wtedy w przestrzeni usługi i nie koliduje z uruchomieniami spod innego użytkownika
PrivateTmp=yes
# Przeglądarka wymaga zapisywalnego katalogu domowego
Environment=HOME=/tmp
# Katalog przeglądarki — patrz punkt powyżej
Environment=PDFSettings__BlinkPath=/opt/google/chrome

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

```
sudo systemctl daemon-reload
sudo systemctl enable --now rsaHelper
```

Poprawność konfiguracji potwierdza pole `PdfGeneration` w odpowiedzi punktu końcowego **Health** (patrz sekcja 4.9) — powinno zawierać `"Available": true`. Wynik jest zapamiętywany na 5 minut, więc po poprawce konfiguracji należy zrestartować usługę, zamiast czekać na odświeżenie.

Instalacja z Docker

Wymagania:

- Docker Engine 20.10+
- Docker Compose (opcjonalnie)

Kroki instalacji:

1. Pobierz kod aplikacji lub obraz Docker
2. Utwórz katalog `Data` i plik konfiguracyjny NIP (patrz sekcja 3.4)
3. Uruchom kontener:

```
# Uruchomienie kontenera (werjsa produkcyjna)
docker run -d -p 5000:8080 -v ./Data:/app/Data:ro --name rsahelper-api rsahelper-webapi:latest
```

```
# Uruchomienie kontenera (werjsa rozwojowa)
docker run -d -p 5000:8080 -v ./Data:/app/Data:ro --name rsahelper-api rsahelper-webapi:develop
```

Lub z Docker Compose:

Przykładowy plik docker-compose.yml można pobrać na stronie z plikami

```
# Uruchomienie
docker-compose up -d

# Sprawdzenie statusu
docker-compose ps

# Wyświetlanie logów
docker-compose logs -f rsahelper-api

# Zatrzymanie
docker-compose down
```

Sprawdzenie działania:

```
curl http://localhost:5000/health
```

Aby przetestować czy aplikacja została zainstalowana prawidłowo należy wywołać punkt końcowy **Health** (patrz sekcja 4.9).

Dokładna instrukcja instalacji z użyciem kontenerów dostępna jest na stronie z plikami.

3.3. Instalacja na systemach Windows

W systemie Windows należy skonfigurować serwer WWW. Rekomendowanym serwerem dla systemów Windows jest **IIS**. Możliwe jest też użycie serwera Apache – instrukcja instalacji i konfiguracji dostępna tutaj.

Dodatkowo należy zainstalować **ASP.NET Core Hosting Bundle** w wersji 10. Instrukcja instalacji .NET Hosting Bundle w wersji 10 znajduje się tutaj.

> **WAŻNE:** Należy zainstalować Hosting Bundle (nie samo Runtime), który zawiera zarówno ASP.NET Core Runtime jak i moduł IIS.

Po zainstalowaniu Hosting Bundle, w konfiguracji **Application Pool** w IIS należy ustawić **.NET CLR Version** na **"No Managed Code"**.

Po zainstalowaniu i konfiguracji serwera WWW należy rozpakować otrzymaną paczkę `RSAHelper.WebAPI.zip` do katalogu głównego serwera WWW.

Aby przetestować czy aplikacja została zainstalowana prawidłowo należy wywołać punkt końcowy **Health** (patrz sekcja 4.9).

> **UWAGA:** W przypadku nowej instalacji, po rozpakowaniu paczki w tym samym katalogu należy utworzyć katalog `Data` i wgrać plik konfiguracyjny NIP (patrz sekcja 3.4)

3.4. Pobranie pliku licencyjnego

Plik licencyjny można pobrać w strefie klienta (wymagane zalogowanie, patrz sekcja 3.1). Plik można załadować do aplikacji na dwa sposoby:

1. Poprzez bezpośrednie wgranie go do katalogu Data na serwerze WWW (w przypadku instalacji opartej o Docker do zamapowanego katalogu Data)
2. Poprzez wywołanie endpoint'a /api/Authorization/upload (patrz sekcja 4.1.4.)

3.5. Konfiguracja proxy

W pliku appsettings.json znajduje się sekcja odpowiedzialna za konfigurację proxy, którą należy uzupełnić aby aplikacja RSA Helper mogła komunikować się z zewnętrznymi serwisami poprzez proxy. Należy zmienić właściwość "Enabled" na true i w polach Address, Username i Password wprowadzić informacje o proxy (username i password są opcjonalne i powinny być tylko wypełnione w przypadku gdy proxy wymaga uwierzytelnienia)

```
"KSeFClientOptions": {
    ...
    "Proxy": {
      "Enabled": false,
      "Address": "http://proxy.mojafirma.pl:8080",
      "Username": "",
      "Password": ""
    }
  }
```

Pozostałe parametry sekcji `KSeFClientOptions`

Sekcja `KSeFClientOptions` konfiguruje bibliotekę kliencką KSeF.Client używaną wewnętrznie przez aplikację. Poza `Proxy` zawiera następujące parametry:

```
"KSeFClientOptions": {
  "customHeaders": {},
```

```
  "ResourcesPath": "Resources",
  "DefaultCulture": "pl-PL",
  "SupportedCultures": ["pl-PL", "en-US"],
  "SupportedUICultures": ["pl-PL", "en-US"]
}
```

Parametr	Opis	Domyślnie
`customHeaders`	Dodatkowe nagłówki HTTP dołączane do każdego żądania wysyłanego do API KSeF	`{}` (brak)
`ResourcesPath`	Ścieżka do katalogu z zasobami (plikami lokalizacyjnymi) biblioteki klienckiej KSeF.Client	`Resources`
`DefaultCulture`	Domyślna kultura używana przez bibliotekę kliencką (m.in. komunikaty błędów)	`pl-PL`
`SupportedCultures`	Lista kultur obsługiwanych przez bibliotekę kliencką	`["pl-PL", "en-US"]`
`SupportedUICultures`	Lista kultur interfejsu obsługiwanych przez bibliotekę kliencką	`["pl-PL", "en-US"]`

> **UWAGA:** Adres bazowy API KSeF (`BaseUrl`) nie jest ustawiany statycznie w tej sekcji — aplikacja dobiera go dynamicznie na podstawie parametru `serverType` przekazywanego w żądaniu (`prod` , `demo` , `test` , `test2`).

3.6. Konfiguracja trybu Offline

Tryb offline umożliwia działanie aplikacji bez dostępu do internetu. W tym trybie:

- Certyfikaty KSeF są pobierane z lokalnych plików zamiast z API
- Schematy XSD/XSL są ładowane z lokalnych folderów zamiast z crd.gov.pl

Tryby działania

Tryb	Enabled	FallbackToOnline	Opis
Online	`false`	-	Certyfikaty i schematy pobierane z internetu
Hybrid	`true`	`true`	Najpierw lokalne pliki, jeśli brak - pobieranie z sieci
Strict Offline	`true`	`false`	Tylko lokalne pliki, brak dostępu do sieci

Konfiguracja w appsettings.json

```
"OfflineMode": {
  "Enabled": true,
  "CertificatesPath": "Data/Certificates",
  "FallbackToOnline": true
}
```

Parametry:

- `Enabled` - włącza/wyłącza tryb offline (domyślnie: `false`)
- `CertificatesPath` - ścieżka do folderu z certyfikatami (domyślnie: `Data/Certificates`)
- `FallbackToOnline` - czy w przypadku braku lokalnych zasobów próbować pobrać z sieci (domyślnie: `true`)

Struktura plików certyfikatów

```
Data/
└── Certificates/
    ├── DEMO/
    │   └── certificates.json
    ├── TEST/
    │   └── certificates.json
    └── PROD/
        └── certificates.json
```

Format pliku certificates.json

```
[
  {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIID...\n-----END CERTIFICATE-----",
    "validFrom": "2024-01-01T00:00:00Z",
    "validTo": "2025-12-31T23:59:59Z",
    "usage": ["SymmetricKeyEncryption", "KsefTokenEncryption"]
  }
]
```

Pola:

- `certificate` - certyfikat w formacie PEM
- `validFrom` - data początku ważności (ISO 8601)
- `validTo` - data końca ważności (ISO 8601)
- `usage` - lista typów użycia: `SymmetricKeyEncryption`, `KsefTokenEncryption`

Lokalne schematy XML

Schematy XSD i szablony XSL są automatycznie dostarczane z aplikacją w folderach:

- `XmlSchemas/` - schematy XSD do walidacji
- `XslFiles/` - szablony XSLT do transformacji

W trybie offline aplikacja używa tych lokalnych plików zamiast pobierania z http://crd.gov.pl.

> **UWAGA:** W trybie Strict Offline (`FallbackToOnline: false`) upewnij się, że wszystkie wymagane certyfikaty i schematy są dostępne lokalnie. W przeciwnym razie operacje kryptograficzne i walidacja XML zakończą się błędem.

Narzędzie do pobierania certyfikatów (CertificateDownloader)

Do automatycznego pobierania certyfikatów KSeF i zapisywania ich w formacie wymaganym przez tryb offline dostępne jest narzędzie **RSAHelper.CertificateDownloader**.

Pobieranie: Narzędzie dostępne jest do pobrania na stronie z plikami (sekcja Narzędzia, bez wymogu logowania).

Użycie:

```
# Pobierz certyfikaty dla środowiska TEST
RSAHelper.CertificateDownloader -e TEST

# Pobierz certyfikaty dla środowiska PROD do domyślnego folderu
RSAHelper.CertificateDownloader -e PROD

# Pobierz certyfikaty dla wszystkich środowisk (DEMO, TEST, PROD)
RSAHelper.CertificateDownloader --all

# Pobierz certyfikaty do własnego folderu
RSAHelper.CertificateDownloader --all -o ./MojeCertyfikaty
```

Parametry:

Parametr	Skrót	Opis
`--environment`	`-e`	Środowisko KSeF: `DEMO`, `TEST` lub `PROD` (wymagany jeśli nie `--all`)
`--output`	`-o`	Ścieżka wyjściowa (domyślnie: `Data/Certificates`)
`--all`	`-a`	Pobierz certyfikaty dla wszystkich środowisk
`--help`	`-h`	Wyświetl pomoc

Przykład wyjścia:

```
=== RSAHelper - Certificate Downloader ===
Pobieranie certyfikatów ze środowiska: TEST
URL: https://api-test.ksef.mf.gov.pl
Łączenie z API KSeF...
Pobrano 2 certyfikatów
Zapisano certyfikaty do: Data/Certificates/TEST/certificates.json

Podsumowanie:
  - SymmetricKeyEncryption:
    Ważny od: 2024-01-01
    Ważny do: 2025-12-31
  - KsefTokenEncryption:
    Ważny od: 2024-01-01
    Ważny do: 2025-12-31
```

> **UWAGA:** Narzędzie wymaga dostępu do internetu w celu pobrania certyfikatów z API KSeF. Po pobraniu certyfikatów można uruchomić aplikację RSAHelper w trybie offline.

3.7. Konfiguracja domyślnego języka konwersji dokumentów

Aplikacja umożliwia konfigurację domyślnego języka dla konwersji dokumentów do HTML i PDF. Ustawienie to wpływa na endpointy ConvertToHtml i ConvertToPdf.

Konfiguracja w appsettings.json

```
"PDFSettings": {
  "DefaultLanguage": "pl"
}
```

Parametry:

- `DefaultLanguage` - domyślny język szablonów XSLT (domyślnie: `pl`)
  - Dostępne wartości: `pl` (polski), `en` (angielski)

Zasady działania

Przy wywołaniu endpointów konwersji:

- Jeśli parametr `lang` jest podany w zapytaniu, ma on priorytet
- Jeśli `lang` nie jest podany, używany jest skonfigurowany `DefaultLanguage`

Przykład:

```
# Używa skonfigurowanego domyślnego języka (np. "pl")
curl -X POST http://localhost:5000/api/ConvertToHtml/fromBase64 -d "..."

# Wymusza język angielski
curl -X POST "http://localhost:5000/api/ConvertToHtml/fromBase64?lang=en" -d "..."
```

Konfiguracja w Docker

Aby zmienić domyślny język w środowisku Docker, utwórz lub zaktualizuj plik `Data/appsettings.additional.json`:

```
{
  "PDFSettings": {
    "DefaultLanguage": "en"
  }
}
```

Następnie zrestartuj kontener:

```
docker-compose restart
```

3.8. Konfiguracja wydruków PDF

Aplikacja umożliwia konfigurację parametrów generowania dokumentów PDF: wybór silnika konwersji, marginesów strony oraz orientacji wydruku. Wszystkie ustawienia znajdują się w sekcji `PDFSettings` pliku `appsettings.json`.

Konfiguracja w appsettings.json

```
"PDFSettings": {
  "PdfEngine": "Syncfusion",
  "PdfOrientation": "Portrait",
  "DefaultFontSize": 26,
  "DefaultQrCodeSize": 350,
  "BlinkPath": "",
  "PdfMargins": {
    "Top": 20,
    "Bottom": 20,
    "Left": 20,
    "Right": 20
  }
}
```

Silnik konwersji (`PdfEngine`)

Aplikacja obsługuje dwa silniki konwersji HTML do PDF:

Wartość	Opis	Wymagania
`Syncfusion`	Silnik oparty na przeglądarce Chromium. Najlepsza jakość renderowania	Windows i Docker: brak dodatkowych kroków. Linux bez kontenera: przeglądarka zainstalowana w systemie i ustawienie `BlinkPath` (patrz sekcja 3.2 oraz niżej)
`WkHtmlToPdf`	Silnik oparty na WebKit	Linux: wymagane biblioteki natywne (szczegóły poniżej)

WkHtmlToPdf na serwerze Linux (Debian/Ubuntu, SLES, Fedora):

- Wymagane biblioteki natywne: `libjpeg62`, `libxrender1`, `libfontconfig1`, `libx11-6`, `libxcb1`, `libxext6`
- Dodatkowo wymagana jest biblioteka OpenSSL zgodna z dystrybucją i binarką wkhtmltopdf (np. `libssl3` na Ubuntu 24.04 / Debian 12 lub `libssl1.1` na starszych systemach)
- Przykładowa instalacja:

```
sudo apt-get update
sudo apt-get install -y --no-install-recommends \
  libgdiplus zlib1g libfontconfig1 libfreetype6 libx11-6 libxext6 libxrender1 libssl3
```

Domyślna wartość: `Syncfusion`

Katalog przeglądarki dla silnika Syncfusion (`BlinkPath`)

Katalog z przeglądarką Chromium, którą silnik `Syncfusion` renderuje dokumenty. Ustawienie dotyczy wyłącznie instalacji na systemach Linux **bez kontenera** — paczka dla Linuksa nie zawiera przeglądarki. Instalację przeglądarki opisuje sekcja

3.2.

Parametr	Opis	Domyślnie
`BlinkPath`	Katalog z przeglądarką dla silnika `Syncfusion`	puste

Sposób instalacji	Wartość `BlinkPath`
`google-chrome-stable` (zalecany, Ubuntu i Debian)	`/opt/google/chrome`
pakiet `chromium` z repozytorium dystrybucji	`/usr/lib/chromium` (wymaga dowiązania `chrome` — patrz sekcja 3.2)
obraz Docker	puste — przeglądarka jest częścią obrazu
Windows (IIS)	puste — przeglądarka jest częścią paczki

Wartość można też podać zmienną środowiskową `PDFSettings__BlinkPath`, która nadpisuje `appsettings.json` — wygodne, gdy ten sam plik konfiguracyjny obsługuje kilka środowisk. Zmiana wymaga restartu aplikacji.

Gdy ustawienie jest puste, silnik szuka przeglądarki w katalogu `runtimes` obok aplikacji. Na systemie Linux bez kontenera skutkuje to błędem `Blink files are missing at .../runtimes/linux/native` widocznym w polu `PdfGeneration` punktu końcowego **Health** (patrz sekcja 3.12).

Orientacja strony (`PdfOrientation`)

Wartość	Opis
`Portrait`	Orientacja pionowa
`Landscape`	Orientacja pozioma

Domyślna wartość: `Portrait`

Marginesy strony (`PdfMargins`)

Marginesy podawane są w milimetrach. Ustawienie dotyczy wszystkich silników konwersji.

Parametr	Opis	Domyślnie
`Top`	Margines górny	`0`
`Bottom`	Margines dolny	`0`
`Left`	Margines lewy	`0`
`Right`	Margines prawy	`0`

Rozmiar czcionki dokumentu (`DefaultFontSize`)

Bazowy rozmiar czcionki (w pikselach) korzenia dokumentu dla szablonów obsługujących parametr `baseFontSize` (obecnie szablon alternatywny **FA(3)**, PL i EN). Pozostałe rozmiary w szablonie wyrażone są w jednostkach `rem`, więc zmiana tej wartości skaluje cały dokument proporcjonalnie.

Parametr	Opis	Domyślnie
`DefaultFontSize`	Bazowy rozmiar czcionki dokumentu w px	`26`

Priorytet ustalania rozmiaru:

1. Parametr `fontSize` przekazany w żądaniu (query lub pole `FontSize` w body `withQrCodes`)
2. `PDFSettings.DefaultFontSize` z `appsettings.json`
3. Wartość domyślna z szablonu (`26` px), gdy powyższe nie są ustawione

Wartość `null` lub niedodatnia oznacza użycie wartości domyślnej z szablonu. Szablony, które nie deklarują parametru `baseFontSize`, ignorują to ustawienie. Zmiana w `appsettings.json` wymaga restartu aplikacji.

Rozmiar kodów QR w wizualizacji (`DefaultQrCodeSize`)

Rozmiar (szerokość w pikselach) kodów QR wyświetlanych na końcu wizualizacji faktury, gdy są przekazane przez endpointy `withQrCodes`. Rozmiar dotyczy jednakowo obu kodów QR i jest niezależny od użytego szablonu wizualizacji.

Parametr	Opis	Domyślnie
`DefaultQrCodeSize`	Szerokość wyświetlanych kodów QR w px	`350` (tak jak w dostarczanym `appsettings.json`)

Priorytet ustalania rozmiaru:

1. Parametr `qrCodeSize` przekazany w żądaniu (pole `QrCodeSize` w body `withQrCodes`)
2. `PDFSettings.DefaultQrCodeSize` z `appsettings.json` (domyślnie `350`)
3. Wartość wbudowana w kod (`207` px) — używana tylko, gdy `DefaultQrCodeSize` zostanie całkowicie usunięte z `appsettings.json`

Wartość `null` lub niedodatnia oznacza użycie wartości domyślnej. Zmiana w `appsettings.json` wymaga restartu aplikacji.

3.9. Konfiguracja renderera i czcionek etykiet kodów QR

Aplikacja umożliwia konfigurację renderera oraz czcionki używanej do renderowania etykiet wyświetlanych pod kodami QR (np. numer KSeF).

Konfiguracja w appsettings.json

```
"QrCode": {
  "Renderer": "SkiaSharp",
  "FontFamily": "<embedded>",
  "DefaultFontSize": 14,
  "EmbeddedFontPath": "Fonts/DejaVuSans.ttf",
```

```
  "BmpBitsPerPixel": 1
}
```

Parametry:

- `Renderer` - renderer używany do generowania obrazu kodu QR i etykiet. Dostępne wartości: `SkiaSharp` (domyślny) oraz `ImageSharp` (nowy renderer)
- `FontFamily` - nazwa czcionki używanej do etykiet (domyślnie: `<embedded>`). Użyj `<embedded>` aby użyć czcionki wbudowanej w aplikację (DejaVu Sans z katalogu `Fonts`), lub podaj nazwę czcionki systemowej (np. `Arial`) - czcionka musi być wtedy zainstalowana w systemie operacyjnym
- `DefaultFontSize` - rozmiar czcionki etykiet w pikselach (domyślnie: `14`)
- `EmbeddedFontPath` - ścieżka względna do pliku czcionki wbudowanej (domyślnie: `Fonts/DejaVuSans.ttf`). Używane tylko gdy `FontFamily` jest ustawione na `<embedded>`
- `BmpBitsPerPixel` - głębia kolorów obrazu BMP: `1` = monochromatyczny (mały rozmiar pliku), `24` = pełne kolory z antyaliasingiem tekstu (lepsza jakość etykiet). Domyślnie: `1`

3.10. Konfiguracja uwierzytelniania certyfikatem KSeF (KsefAuth)

Aplikacja umożliwia uwierzytelnianie w KSeF 2.0 certyfikatem klienta oraz przechowywanie poświadczeń w lokalnym, zaszyfrowanym magazynie na serwerze. Ustawienia znajdują się w sekcji `KsefAuth` pliku `appsettings.json` i dotyczą endpointów `api/authorization-certificates` (uzyskiwanie/odświeżanie access tokenu) oraz `api/certificates` (zapis, aktualizacja, odczyt metadanych i usuwanie lokalnie przechowywanych poświadczeń).

Konfiguracja w appsettings.json

```
"KsefAuth": {
  "CredentialStoragePath": "Data/KsefCredentials",
  "MasterKey": "",
  "KeyVersion": "v1",
  "AllowInlineCredentials": true,
  "AllowStoredCredentials": true,
  "AuthenticationTimeoutSeconds": 120
}
```

Parametry:

Parametr	Opis	Domyślnie
`CredentialStoragePath`	Ścieżka do katalogu z lokalnym magazynem poświadczeń	`Data/KsefCredentials`
`MasterKey`	Główny klucz ochrony danych po stronie RSAHelper, używany do szyfrowania poświadczeń zapisywanych lokalnie. Zalecany format: Base64	brak (pusty)
`KeyVersion`	Wersja klucza szyfrującego używana przy zapisie nowych rekordów poświadczeń	`v1`

`AllowInlineCredentials`	Czy dozwolone jest przekazywanie poświadczeń (np. certyfikatu) bezpośrednio w treści żądania, bez zapisu w magazynie lokalnym	`true`
`AllowStoredCredentials`	Czy dozwolone jest użycie poświadczeń zapisanych wcześniej w lokalnym magazynie	`true`
`AuthenticationTimeoutSeconds`	Maksymalny czas oczekiwania (w sekundach) na zakończenie procesu autoryzacji w KSeF	`120`

> **UWAGA:** `MasterKey` chroni poświadczenia zapisane w `CredentialStoragePath`. Wartość tę należy ustawić na silny, unikalny sekret przed uruchomieniem na środowisku produkcyjnym i przechowywać poza repozytorium kodu (np. w zmiennej środowiskowej lub menedżerze sekretów). Wyłączenie zarówno `AllowInlineCredentials`, jak i `AllowStoredCredentials` uniemożliwi uwierzytelnianie certyfikatem KSeF.

3.11. Konfiguracja logów

Aplikacja wykorzystuje bibliotekę **NLog**, konfigurowaną przez plik `nlog.config` znajdujący się w katalogu aplikacji (obok pliku `RSAHelper.WebAPI.dll`). Plik jest wczytywany z atrybutem `autoReload="true"`, dzięki czemu zmiany zapisane w tym pliku są stosowane od razu, bez potrzeby restartu aplikacji ani usługi.

Domyślne cele logowania

Plik	Zawartość	Minimalny poziom	Przechowywanie
`logs/log-main.log`	Wszystkie logi aplikacji	`Warning`	7 dni
`logs/log-errors.log`	Tylko błędy	`Error`	7 dni
`logs/log-requests.log`	Szczegóły przychodzących żądań HTTP (patrz niżej)	`Error` (domyślnie)	2 dni

Niezależnie od powyższych plików, logi na poziomie `Info` i wyższym są dodatkowo wypisywane na konsolę.

Logowanie szczegółów żądań HTTP (`RequestLoggingMiddleware`)

Klasa `RSAHelper.WebAPI.Middleware.RequestLoggingMiddleware` loguje każde przychodzące żądanie HTTP na samym początku potoku (przed routingiem i wykonaniem kontrolera) — metodę, pełny adres, protokół, typ i długość zawartości, wszystkie nagłówki oraz treść (body) żądania. Middleware jest zarejestrowany globalnie i działa we wszystkich środowiskach, również produkcyjnym.

Wpisy te są logowane na poziomie `Information`, a ich docelowy plik kontroluje osobna reguła w `nlog.config`:

```
<logger name="RSAHelper.WebAPI.Middleware.RequestLoggingMiddleware" minlevel="Error" writeTo="requestLogfile
```

Domyślnie `minlevel` tej reguły jest ustawiony na `Error`. Ponieważ middleware loguje wyłącznie na poziomie `Information`, żaden wpis nie spełnia domyślnie tego warunku i nic nie trafia do `logs/log-requests.log`. Reguła ma dodatkowo atrybut `final="true"`, który zatrzymuje przetwarzanie kolejnych reguł dla tego loggera już przez samo

dopasowanie nazwy — niezależnie od poziomu. Dlatego żądania nie pojawiają się domyślnie również na konsoli ani w `log-main.log`, mimo że pasowałyby do ogólnej reguły `<logger name="*" minlevel="Info" writeTo="console" />`.

Aby zobaczyć dokładne wywołania API (pełne żądania wraz z nagłówkami i treścią), zmień w pliku `nlog.config` wartość `minlevel` tej reguły z `Error` na `Info`:

```
<logger name="RSAHelper.WebAPI.Middleware.RequestLoggingMiddleware" minlevel="Info" writeTo="requestLogfile"
```

Po zapisaniu pliku zmiana zostanie wczytana automatycznie (dzięki `autoReload="true"`), a kolejne żądania zaczną być zapisywane w `logs/log-requests.log`, np.:

```
2026-08-18 10:15:32.1234 | Info | HTTP Request started: GET http://localhost:5000/api/Authorization/57322960
```

> **UWAGA:** Przy `minlevel="Info"` w logu zapisywana jest pełna treść żądania, w tym wszystkie nagłówki oraz body (np. treść przesyłanej faktury XML). Włączaj ten poziom tylko tymczasowo, na czas diagnostyki, a po jej zakończeniu przywróć `minlevel="Error"`. Katalog `logs` powinien mieć ograniczony dostęp, ponieważ zapisywane żądania mogą zawierać dane wrażliwe.

Konfiguracja w Docker

Plik `nlog.config` jest zapisywany na stałe wewnątrz obrazu podczas budowania i domyślnie nie jest montowany jako wolumin w `docker-compose.yml`. Aby zmiana `minlevel` przetrwała ponowne utworzenie kontenera, zamontuj własny plik `nlog.config` analogicznie do katalogu `Data`:

```
volumes:
  - ./Data:/app/Data:ro
  - ./nlog.config:/app/nlog.config:ro
```

Do szybkiej, tymczasowej zmiany w już działającym kontenerze (do czasu jego ponownego utworzenia) wystarczy podmienić plik wewnątrz kontenera — dzięki `autoReload="true"` zmiana powinna zostać wykryta automatycznie; jeśli tak się nie stanie, zrestartuj kontener:

```
docker cp nlog.config rsahelper-api:/app/nlog.config
docker restart rsahelper-api  # jeśli zmiana nie zostanie wykryta automatycznie
```

3.12. Troubleshooting

Błędy generowania PDF na systemach Linux (instalacja bez kontenera)

Stan generowania PDF pokazuje pole `PdfGeneration` w odpowiedzi punktu końcowego **Health** (sekcja 4.9). Gdy `"Available"` ma wartość `false`, pole `Error` zawiera komunikat, po którym rozpoznaje się przyczynę:

Komunikat w polu `Error`	Przyczyna	Rozwiązanie
`Blink files are missing at .../runtimes/linux/native`	nie ustawiono `PDFSettings:BlinkPath` albo wskazany katalog nie zawiera pliku `chrome`	sekcja 3.2 — instalacja przeglądarki i ustawienie ścieżki
`Failed to launch Base!` oraz `chrome_crashpad_handler: --database is required` w logach	użytkownik usługi nie ma zapisywalnego katalogu domowego	`Environment=HOME=/tmp` w pliku usługi
`Failed to launch Base!` bez wzmianki o `crashpad`	brak bibliotek systemowych przeglądarki	`ldd /opt/google/chrome/chrome | grep "not found"` i doinstalowanie brakujących pakietów
`Failed to create connection`	przeglądarka uruchamia się, ale zrywa połączenie sterujące	sprawdzić, czy uruchamia się samodzielnie: `sudo -u www-data /opt/google/chrome/chrome --headless=new --no-sandbox --dump-dom about:blank`
`Access to the path '/tmp/chromium-rsahelper/...' is denied`	katalog roboczy przeglądarki w `/tmp` należy do innego użytkownika, bo aplikacja była wcześniej uruchamiana z innego konta	`PrivateTmp=yes` w sekcji `[Service]` pliku usługi albo jednorazowo `sudo rm -rf /tmp/chromium-rsahelper`

Wynik sprawdzenia jest zapamiętywany na 5 minut — po poprawce konfiguracji należy zrestartować usługę, zamiast czekać na odświeżenie.

Błąd `503 Service Unavailable` przy konfiguracji Apache + .NET

W tej konfiguracji błąd `503 Service Unavailable` najczęściej oznacza, że Apache działa poprawnie, ale aplikacja .NET nie odpowiada. Apache próbuje przekazać ruch do aplikacji (np. na port `5000`), ale backend nie nasłuchuje albo zakończył pracę z błędem.

1. Sprawdzenie statusu usługi

Najpierw sprawdź, czy usługa systemowa aplikacji w ogóle działa:

```
sudo systemctl status <nazwa-uslugi>.service
```

- jeśli widzisz `Active: failed`, aplikacja nie uruchomiła się poprawnie
- jeśli widzisz `Active: active (running)`, aplikacja działa, ale może nasłuchiwać na innym porcie niż zakłada konfiguracja Apache

2. Sprawdzenie logów aplikacji

Jeśli usługa nie działa poprawnie, sprawdź ostatnie logi systemd:

```
sudo journalctl -u <nazwa-uslugi>.service -n 50 --no-pager
```

Szukaj wpisów `Critical` lub `Error`. Częste przyczyny:

- błąd w `appsettings.json`
- brak wymaganej biblioteki systemowej
- błędna ścieżka do plików aplikacji lub brak uprawnień

3. Sprawdzenie nasłuchiwania na porcie

Sprawdź, czy aplikacja rzeczywiście nasłuchuje na porcie oczekiwanym przez Apache:

```
sudo ss -tulpn | grep 5000
```

Jeśli polecenie nie zwraca wyniku, aplikacja nie nasłuchuje na porcie `5000`. Może działać na innym porcie, np. `5001`, `8080` albo nie uruchomiła się wcale.

4. Najczęstsze przyczyny i rozwiązania

Błędna ścieżka w pliku `.service`

Upewnij się, że `ExecStart` zawiera pełną ścieżkę do `dotnet` (najczęściej `/usr/bin/dotnet`) oraz poprawną ścieżkę do pliku `.dll` aplikacji.

Brak uprawnień do katalogu aplikacji

Jeżeli usługa działa jako `www-data`, upewnij się, że użytkownik ma dostęp do katalogu aplikacji:

```
sudo chown -R www-data:www-data /var/www/rsahelper
```

Problem środowiskowy lub konfiguracyjny

Jeśli logi wskazują na problem z konfiguracją albo zależnościami, tymczasowo ustaw środowisko developerskie w pliku `.service`:

```
Environment=ASPNETCORE_ENVIRONMENT=Development
```

Po każdej zmianie pliku `.service` wykonaj:

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart <nazwa-uslugi>.service
```

5. Szybki test ręczny

Jeżeli chcesz zobaczyć błąd bezpośrednio w konsoli, zatrzymaj usługę i uruchom aplikację ręcznie:

```
cd /var/www/rsahelper
dotnet RSAHelper.WebAPI.dll
```

Jeśli aplikacja wypisze komunikat podobny do:

```
Now listening on: http://localhost:5000
```

a w przeglądarce nadal pojawia się `503`, problem najprawdopodobniej leży po stronie konfiguracji Apache lub reverse proxy.

---

4. Użytkowanie

Aplikacja umożliwia dostęp do wszystkich funkcji poprzez interfejs **Swagger**, gdzie wszystkie dostępne funkcje można wywołać w dowolnej przeglądarce internetowej.

Dostęp do interfejsu Swagger

Serwer ogólnodostępny:

- Serwer produkcyjny: `https://rsahelper.unitsoftware.com.pl/swagger`
- Serwer rozwojowy: `https://testrsahelper.unitsoftware.com.pl/swagger`

Serwer lokalny:

- Format adresu: `http://<adresIPSerwera>:<port>/swagger`
- Gdzie:
  - `adresIPSerwera` - IP lub nazwa komputera gdzie zainstalowana jest aplikacja
  - `port` - port na którym dostępna jest aplikacja (domyślnie 80 lub 8080 dla Docker)

Docker:

- `http://localhost:5000/swagger`

> **UWAGA:** Opis działania poszczególnych funkcji wraz z parametrami dla aktualnie zainstalowanej wersji dostępny jest poprzez interfejs Swagger i może się różnić od informacji zawartych w tej instrukcji (instrukcja zawiera zawsze opis najnowszej wersji aplikacji).

4.1. Authorization

Ścieżka bazowa: `/api/Authorization`

Kontroler odpowiedzialny za zarządzanie listą autoryzowanych numerów NIP oraz operacje na plikach konfiguracyjnych.

4.1.1. Pobierz listę wszystkich NIP

Endpoint: `GET /api/Authorization`

Opis: Zwraca listę wszystkich numerów NIP obsługiwanych przez aplikację w formacie JSON.

Parametry: Brak

Odpowiedź:

```
{
  "nips": [
    "5732296089",
    "9721121810",
    "1234567890"
  ]
}
```

Przykład wywołania:

```
GET /api/Authorization
```

4.1.2. Pobierz plik tekstowy z listą NIP

Endpoint: `GET /api/Authorization/NipListFile`

Opis: Zwraca listę NIP w formacie tekstowym (jeden NIP na linię), przydatne do eksportu.

Parametry: Brak

Odpowiedź:

- Content-Type: `application/txt`
- Plik: `NipList.txt`
- Format: Jeden NIP w każdej linii

Przykład odpowiedzi:

```
5732296089
9721121810
1234567890
```

4.1.3. Sprawdź czy NIP jest obsługiwany

Endpoint: `GET /api/Authorization/{nip}`

Opis: Sprawdza czy podany numer NIP znajduje się na liście obsługiwanych przez aplikację.

Parametry:

- `nip` (path, wymagany) - Numer NIP do sprawdzenia (10 cyfr)

Odpowiedź:

- Format: `boolean`
- `true` - NIP jest obsługiwany
- `false` - NIP nie jest obsługiwany

Przykład wywołania:

```
GET /api/Authorization/5732296089
```

Odpowiedź:

```
true
```

4.1.4. Upload pliku z NIP

Endpoint: `POST /api/Authorization/upload`

Opis: Aktualizuje plik konfiguracyjny z dozwolonymi numerami NIP. Plik musi zawierać poprawny hash HMAC-SHA256.

Parametry:

- Body (JSON):

```
{
  "nips": ["5732296089", "9721121810"],
  "version": "1.0",
  "createdAt": "2024-12-08T12:00:00Z",
  "description": "Production NIP list",
  "hash": "valid-hmac-sha256-hash-base64"
}
```

Pola:

- `nips` (array, wymagany) - Lista numerów NIP
- `version` (string, wymagany) - Wersja pliku
- `createdAt` (datetime, opcjonalny) - Data utworzenia
- `description` (string, opcjonalny) - Opis
- `hash` (string, wymagany) - Hash HMAC-SHA256 (obliczony z użyciem klucza skonfigurowanego w aplikacji)

Odpowiedź:

```
{
  "message": "Plik został pomyślnie załadowany.",
  "nipsCount": 2,
  "version": "1.0"
}
```

Kody błędów:

- `400 Bad Request` - Niepoprawny format pliku lub błędny hash
- `500 Internal Server Error` - Błąd serwera podczas zapisu pliku

4.2. Calculate

Ścieżka bazowa: `/api/Calculate`

Kontroler odpowiedzialny za szyfrowanie tokenów autoryzacyjnych algorytmem RSA.

4.2.1. Szyfrowanie tokena RSA

Endpoint: `GET /api/Calculate/{serverType}/{inputString}`

Opis: Metoda szyfrująca ciąg znaków algorytmem RSA przy użyciu klucza publicznego KSeF. Używana do szyfrowania tokenów przy autoryzacji w KSEF.

Parametry:

- `serverType` (path, wymagany) - Typ środowiska KSeF:
  - `prod` - Produkcyjne
  - `demo` - Demonstracyjne
  - `test` - Testowe
- `inputString` (path, wymagany) - Ciąg znaków do zaszyfrowania (token)
- `nip` (query, wymagany) - NIP podmiotu (parametr autoryzacyjny)

Mechanizm działania:

- Dla 'prod', `test` i `demo`: Pobiera dynamicznie klucze publiczne z API KSeF za pomocą `CryptographyService`

Odpowiedź:

- Status: `200 OK`
- Format: `string` (Base64)

- Treść: Zaszyfrowany ciąg znaków zakodowany w Base64

Kody błędów:

- `401 Unauthorized` - Brak parametru NIP
- `401 Unauthorized` - NIP nie jest obsługiwany przez aplikację
- `500 Internal Server Error` - Błąd serwera

Przykład wywołania:

```
GET /api/Calculate/test/mojToken123?nip=5732296089
```

Odpowiedź:

```
SGVsbG8gV29ybGQhIFRoaXMgaXMgYW4gZXhhbXBsZSBvZiBSU0EgZW5jcnlwdGVkIGRhdGE=
```

4.3. Invoices

Ścieżka bazowa: `/api/Invoices`

Kontroler odpowiedzialny za generowanie danych kryptograficznych oraz szyfrowanie faktur dla KSEF 2.0 (sesja interaktywna).

4.3.1. Generowanie danych szyfrowania

Endpoint: `GET /api/Invoices/GenerateEncryptionData/{serverType}`

Opis: Generuje klucz AES-256, wektor inicjalizujący (IV) oraz zaszyfrowany klucz symetryczny potrzebny do:

1. Inicjalizacji sesji interaktywnej w KSEF 2.0
2. Szyfrowania faktury przed wysyłką

Parametry:

- `serverType` (path, wymagany) - Typ środowiska: `prod`, `demo`, `test`

Odpowiedź:

- Status: `200 OK`
- Format: JSON

```
{
  "encryptedSymmetricKey": "e+SdWZTeqd3aeSNyMay16Tag/JTQRjNxXyJm3TRxTTHGu1dUcL9MoqZ3TZazsCBc...",
  "initializationVector": "9CrrDfN8jEdZHAbLpodNw==",
```

```
  "symmetricKey": "ZAb5UNVeDh6DoZQvuPmMlDFfB39IqREEbFywQ/VUjSM="
}
```

Opis pól odpowiedzi:

- `encryptedSymmetricKey` - Zaszyfrowany klucz symetryczny (RSA). Należy przekazać w żądaniu inicjalizacji sesji interaktywnej KSEF.
- `initializationVector` - Wektor inicjalizujący AES (IV). Należy użyć przy szyfrowaniu faktury oraz przekazać w inicjalizacji sesji.
- `symmetricKey` - Klucz symetryczny AES-256 w czystej postaci. Należy użyć przy szyfrowaniu faktury.

Przykład wywołania:

```
GET /api/Invoices/GenerateEncryptionData/test
```

Przepływ użycia:

1. Wywołaj ten endpoint aby otrzymać dane kryptograficzne
2. Użyj `encryptedSymmetricKey` i `initializationVector` do inicjalizacji sesji KSEF
3. Użyj `symmetricKey` i `initializationVector` w endpoint `fromBase64` do zaszyfrowania faktury

4.3.2. Szyfrowanie faktury i generowanie metadanych

Endpoint: `POST /api/Invoices/fromBase64/{serverType}`

Opis: Szyfruje fakturę w formacie XML algorytmem AES-256-CBC i zwraca metadane wymagane do wysyłki do KSeF (hasze, rozmiary, zaszyfrowana treść).

Parametry:

- `serverType` (path, wymagany) - Typ środowiska: `prod`, `demo`, `test`
- `nip` (query, wymagany) - NIP podmiotu (parametr autoryzacyjny)
- `cipherKey` (query, wymagany) - Klucz AES zakodowany w Base64 (pole `symmetricKey` z `GenerateEncryptionData`)
- `cipherIv` (query, wymagany) - Wektor IV zakodowany w Base64 (pole `initializationVector` z `GenerateEncryptionData`)

Body:

- Content-Type: `text/plain`
- Treść: Plik XML faktury zakodowany w Base64

Odpowiedź:

- Status: `200 OK`
- Format: JSON

```
{
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "invoiceSize": 1234,
  "encryptedInvoiceHash": "d7iTpRuPx1IS/JPTVXTgIwmQRtYvXCoIqNM2rNOzgMw=",
  "encryptedInvoiceSize": 1456,
  "encryptedInvoiceContent": "HpGGgaC0JS21I643ZALb3gA=..."
}
```

Opis pól odpowiedzi:

- `invoiceHash` - Hash SHA-256 oryginalnej (niezaszyfrowanej) faktury
- `invoiceSize` - Rozmiar oryginalnej faktury w bajtach
- `encryptedInvoiceHash` - Hash SHA-256 zaszyfrowanej faktury
- `encryptedInvoiceSize` - Rozmiar zaszyfrowanej faktury w bajtach
- `encryptedInvoiceContent` - Zaszyfrowana faktura zakodowana w Base64

Kody błędów:

- `400 Bad Request` - Brak parametru NIP lub niepoprawny format danych wejściowych
- `403 Forbidden` - NIP nie jest obsługiwany przez aplikację
- `500 Internal Server Error` - Błąd przetwarzania faktury

Funkcje specjalne:

- Automatyczne usuwanie BOM (Byte Order Mark) z pliku XML
- Obsługa różnych kodowań UTF-8

Przykład wywołania:

```
POST /api/Invoices/fromBase64/test?nip=5732296089&cipherKey=ZAb5UNVeD...&cipherIv=9CrrDfN8jEdZH...
Content-Type: text/plain

PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPEZha3R1cmE+Li4uPC9GYWt0dXJhPg==
```

Przepływ pełnego procesu wysyłki faktury KSEF 2.0:

1. Wywołaj `GET /api/Invoices/GenerateEncryptionData/{serverType}` → otrzymasz dane kryptograficzne
2. Zainicjuj sesję interaktywną w KSEF używając `encryptedSymmetricKey` i `initializationVector`
3. Wywołaj `POST /api/Invoices/fromBase64/{serverType}` z kluczami z kroku 1 → otrzymasz metadane
4. Wyślij zaszyfrowaną fakturę do KSEF używając danych z kroku 3

4.3.3. Szyfrowanie paczki ZIP i generowanie metadanych

Endpoint: `POST /api/Invoices/package/{serverType}`

Opis: Szyfruje paczkę ZIP algorytmem AES-256-CBC i zwraca metadane wymagane do wysyłki paczki faktur do KSeF (hasze, rozmiary, zaszyfrowana treść). Używane do wysyłania wielu faktur w jednej paczce.

Parametry:

- `serverType` (path, wymagany) - Typ środowiska: `prod`, `demo`, `test`

Body:

- Content-Type: `application/json`

```
{
  "inputZipData": "UEsDBBQAAAAIAA...",
  "nip": "5732296089",
  "cipherKey": "ZAb5UNVeDh6DoZQvuPmMlDFfB39IqREEbFywQ/VUjSM=",
  "cipherIv": "9CrrDfN8jEdZHAbLpodNw=="
}
```

Pola body:

- `inputZipData` (string, wymagany) - Paczka ZIP zakodowana w Base64
- `nip` (string, wymagany) - NIP podmiotu (parametr autoryzacyjny)
- `cipherKey` (string, wymagany) - Klucz AES zakodowany w Base64 (pole `symmetricKey` z `GenerateEncryptionData`)
- `cipherIv` (string, wymagany) - Wektor IV zakodowany w Base64 (pole `initializationVector` z `GenerateEncryptionData`)

Odpowiedź:

- Status: `200 OK`
- Format: JSON

```
{
  "packageHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "packageSize": 45678,
  "encryptedPackageHash": "d7iTpRuPx1IS/JPTVXTgIwmQRtYvXCoIqNM2rNOzgMw=",
  "encryptedPackageSize": 45696,
  "encryptedPackageContent": "HpGGgaC0JS21I643ZALb3gA=..."
}
```

Opis pól odpowiedzi:

- `packageHash` - Hash SHA-256 oryginalnej (niezaszyfrowanej) paczki ZIP
- `packageSize` - Rozmiar oryginalnej paczki w bajtach
- `encryptedPackageHash` - Hash SHA-256 zaszyfrowanej paczki
- `encryptedPackageSize` - Rozmiar zaszyfrowanej paczki w bajtach
- `encryptedPackageContent` - Zaszyfrowana paczka zakodowana w Base64

Kody błędów:

- `400 Bad Request` - Brak parametru NIP lub niepoprawny format danych wejściowych
- `403 Forbidden` - NIP nie jest obsługiwany przez aplikację
- `500 Internal Server Error` - Błąd przetwarzania paczki

Przykład wywołania:

```
POST /api/Invoices/package/test
Content-Type: application/json

{
  "inputZipData": "UEsDBBQAAAAIAA...",
  "nip": "5732296089",
  "cipherKey": "ZAb5UNVeDh6DoZQvuPmMlDFfB39IqREEbFywQ/VUjSM=",
  "cipherIv": "9CrrDfN8jEdZHAbLpodNw=="
}
```

Przepływ wysyłki paczki faktur KSEF 2.0:

1. Wywołaj `GET /api/Invoices/GenerateEncryptionData/{serverType}` → otrzymasz dane kryptograficzne
2. Zainicjuj sesję interaktywną w KSEF używając `encryptedSymmetricKey` i `initializationVector`
3. Przygotuj paczkę ZIP zawierającą faktury XML
4. Wywołaj `POST /api/Invoices/package/{serverType}` z kluczami z kroku 1 → otrzymasz metadane
5. Wyślij zaszyfrowaną paczkę do KSEF używając danych z kroku 4

4.4. ValidateXml

Ścieżka bazowa: `/api/ValidateXml`

Kontroler odpowiedzialny za walidację dokumentów XML względem schematów XSD JPK.

4.4.1. Walidacja XML (z Base64)

Endpoint: `POST /api/ValidateXml/fromBase64`

Opis: Waliduje dokument XML zakodowany w Base64 względem odpowiedniego schematu XSD JPK. Schemat jest automatycznie wykrywany na podstawie zawartości dokumentu.

Parametry:

- Body (text/plain): Dokument XML zakodowany w Base64

Content-Type: `text/plain`

Odpowiedź:

- Status: `200 OK`
- Format: JSON

```
{
  "isValid": true,
  "messages": [
    {
      "severity": "Warning",
      "message": "Ostrzeżenie o niekompletnych danych"
    }
  ]
}
```

Struktura odpowiedzi:

- `isValid` (boolean) - `true` jeśli dokument jest poprawny, `false` w przeciwnym wypadku
- `messages` (array) - Lista komunikatów walidacji
  - `severity` - Poziom powagi: `Error`, `Warning`
  - `message` - Treść komunikatu z numerem linii (jeśli dostępny)

Przykład wywołania:

```
POST /api/ValidateXml/fromBase64
Content-Type: text/plain

PD94bWwgdmVyc2lvbj0iMS4wIj8+CjxGYWt0dXJhPi4uLjwvRmFrdHVyYT4=
```

4.4.2. Walidacja XML (bezpośrednio)

Endpoint: `POST /api/ValidateXml`

Opis: Waliduje dokument XML względem odpowiedniego schematu XSD JPK. Schemat jest automatycznie wykrywany na podstawie zawartości dokumentu.

Parametry:

- Body (application/xml): Dokument XML w czystej postaci

Content-Type: `application/xml`

Odpowiedź: Identyczna jak w 4.4.1

Przykład wywołania:

```
POST /api/ValidateXml
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<Faktura xmlns="http://crd.gov.pl/wzor/2023/06/29/12648/">
```

```
  ...
</Faktura>
```

Obsługiwane struktury:

- FA(1)_v1-0E - Faktura ustrukturyzowana wersja 1
- FA(2)_v1-0E - Faktura ustrukturyzowana wersja 2
- FA(3)_v1-0E - Faktura ustrukturyzowana wersja 3
- JPK_V7M(2) - JPK VAT z deklaracją

Mechanizm wykrywania schematu: Aplikacja automatycznie wykrywa odpowiedni schemat XSD na podstawie:

1. Kodu formularza
2. Kodu systemowego
3. Wersji schematu

4.5. ConvertToHtml

Ścieżka bazowa: `/api/ConvertToHtml`

Kontroler odpowiedzialny za konwersję dokumentów e-faktur XML do formatu HTML przy użyciu transformacji XSLT.

4.5.1. Konwersja XML do HTML (z Base64)

Endpoint: `POST /api/ConvertToHtml/fromBase64`

Opis: Konwertuje dokument e-faktury w formacie XML (zakodowany Base64) na reprezentację HTML. Transformacja XSLT jest automatycznie wybierana na podstawie typu dokumentu.

Parametry:

- `lang` (query, opcjonalny) - Język transformaty XSLT
  - Wartość domyślna: `pl`
  - Dostępne wartości: `pl` (polski), `en` (angielski)
- `fontSize` (query, opcjonalny) - Bazowy rozmiar czcionki dokumentu w px (dotyczy szablonu alternatywnego FA(3))
  - Gdy pominięty, używana jest wartość `PDFSettings.DefaultFontSize`, a następnie wartość domyślna szablonu (`26`)
  - Wartość niedodatnia jest ignorowana
- Body (text/plain): Dokument XML zakodowany w Base64

Content-Type: `text/plain`

Odpowiedź:

- Status: `200 OK`
- Content-Type: `text/plain`
- Treść: Dokument HTML zakodowany w Base64

Przykład wywołania:

```
POST /api/ConvertToHtml/fromBase64?lang=en
Content-Type: text/plain

PD94bWwgdmVyc2lvbj0iMS4wIj8+CjxGYWt0dXJhPi4uLjwvRmFrdHVyYT4=
```

4.5.2. Konwersja XML do HTML (bezpośrednio)

Endpoint: `POST /api/ConvertToHtml`

Opis: Konwertuje dokument e-faktury w formacie XML na reprezentację HTML.

Parametry:

- `lang` (query, opcjonalny) - Język transformaty XSLT
  - Wartość domyślna: `pl`
  - Dostępne wartości: `pl` (polski), `en` (angielski)
- `fontSize` (query, opcjonalny) - Bazowy rozmiar czcionki dokumentu w px (dotyczy szablonu alternatywnego FA(3))
  - Gdy pominięty, używana jest wartość `PDFSettings.DefaultFontSize`, a następnie wartość domyślna szablonu (`26`)
  - Wartość niedodatnia jest ignorowana
- Body (application/xml): Dokument XML w czystej postaci

Content-Type: `application/xml`

Odpowiedź:

- Status: `200 OK`
- Content-Type: `text/html`
- Treść: Dokument HTML

Przykład wywołania:

```
POST /api/ConvertToHtml?lang=en
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<Faktura xmlns="http://crd.gov.pl/wzor/2023/06/29/12648/">
  ...
</Faktura>
```

Obsługiwane typy dokumentów:

- FA(1) - Faktura ustrukturyzowana wersja 1
- FA(2) - Faktura ustrukturyzowana wersja 2
- FA(3) - Faktura ustrukturyzowana wersja 3

- JPK_V7M(2) - JPK VAT z deklaracją

Mechanizm działania:

1. Wykrycie typu dokumentu
2. Wybór odpowiedniego szablonu XSLT
   - Dla `lang=pl`: używany szablon bez sufiksu (np. `FA(3)_v1-0E.xsl`)
   - Dla `lang=en`: używany szablon z sufiksem `_en` (np. `FA(3)_v1-0E_en.xsl`)
3. Transformacja XML → HTML

4.6. ConvertToPdf

Ścieżka bazowa: `/api/ConvertToPdf`

Kontroler odpowiedzialny za konwersję dokumentów e-faktur XML do formatu PDF.

4.6.1. Konwersja XML do PDF (z Base64)

Endpoint: `POST /api/ConvertToPdf/fromBase64`

Opis: Konwertuje dokument e-faktury w formacie XML (zakodowany Base64) na dokument PDF. Konwersja odbywa się przez transformację XML → HTML → PDF.

Parametry:

- `fileName` (query, opcjonalny) - Nazwa zwracanego pliku PDF
  - Domyślna wartość: `eInvoice.pdf`
- `lang` (query, opcjonalny) - Język transformaty XSLT
  - Wartość domyślna: `pl`
  - Dostępne wartości: `pl` (polski), `en` (angielski)
- `fontSize` (query, opcjonalny) - Bazowy rozmiar czcionki dokumentu w px (dotyczy szablonu alternatywnego FA(3))
  - Gdy pominięty, używana jest wartość `PDFSettings.DefaultFontSize`, a następnie wartość domyślna szablonu (`26`)
  - Wartość niedodatnia jest ignorowana
- Body (application/xml): Dokument XML zakodowany w Base64

Content-Type: `application/xml`

Odpowiedź:

- Status: `200 OK`
- Content-Type: `application/pdf`
- Headers: `Content-Disposition: attachment; filename="<fileName>"`
- Treść: Plik PDF

Przykład wywołania:

```
POST /api/ConvertToPdf/fromBase64?fileName=faktura_2024.pdf&lang=en
Content-Type: application/xml

PD94bWwgdmVyc2lvbj0iMS4wIj8+CjxGYWt0dXJhPi4uLjwvRmFrdHVyYT4=
```

4.6.2. Konwersja XML do PDF (bezpośrednio)

Endpoint: `POST /api/ConvertToPdf`

Opis: Konwertuje dokument e-faktury w formacie XML na dokument PDF.

Parametry:

- `fileName` (query, opcjonalny) - Nazwa zwracanego pliku PDF
  - Domyślna wartość: `eInvoice.pdf`
- `lang` (query, opcjonalny) - Język transformaty XSLT
  - Wartość domyślna: `pl`
  - Dostępne wartości: `pl` (polski), `en` (angielski)
- `fontSize` (query, opcjonalny) - Bazowy rozmiar czcionki dokumentu w px (dotyczy szablonu alternatywnego FA(3))
  - Gdy pominięty, używana jest wartość `PDFSettings.DefaultFontSize`, a następnie wartość domyślna szablonu (`26`)
  - Wartość niedodatnia jest ignorowana
- Body (application/xml): Dokument XML w czystej postaci

Content-Type: `application/xml`

Odpowiedź: Identyczna jak w 4.6.1

Przykład wywołania:

```
POST /api/ConvertToPdf?fileName=invoice.pdf&lang=en
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<Faktura xmlns="http://crd.gov.pl/wzor/2023/06/29/12648/">
  ...
</Faktura>
```

Obsługiwane typy dokumentów:

- FA(1) - Faktura ustrukturyzowana wersja 1
- FA(2) - Faktura ustrukturyzowana wersja 2
- FA(3) - Faktura ustrukturyzowana wersja 3
- JPK_V7M(2) - JPK VAT z deklaracją

Mechanizm działania:

1. Wykrycie typu dokumentu
2. Transformacja XML → HTML (XSLT)
3. Konwersja HTML → PDF

Wymagania licencyjne: Konwersja PDF wykorzystuje bibliotekę Syncfusion.HtmlToPdfConverter, która wymaga licencji (klucz licencyjny jest wbudowany w aplikację).

4.7. QrCode

Ścieżka bazowa: `/api/QrCode`

Kontroler odpowiedzialny za generowanie kodów QR dla weryfikacji faktur w systemie KSEF.

Informacje ogólne o kodach QR

Aplikacja generuje **dwa typy kodów QR** zgodnie ze specyfikacją KSEF:

1. **Kod QR #1 (Weryfikacja online)** - Link do weryfikacji faktury w systemie KSEF
   - Endpointy: `get`, `post`, `postContent`
   - Zawiera: URL weryfikacyjny z NIP, datą i hashem faktury
   - Użycie: Weryfikacja faktury przez odbiorców
2. **Kod QR #2 (Weryfikacja z certyfikatem)** - Link z podpisem cyfrowym
   - Endpointy: `postVer`, `postVerContent`
   - Zawiera: URL + podpis ECDSA z certyfikatu
   - Użycie: Zaawansowana weryfikacja z certyfikatem

Parametry techniczne:

- Format obrazu: BMP (domyślnie)
- Rozmiar: 100-2000 pikseli (domyślnie 270). Dla ostrych kodów zalecana jest wielokrotność 45; w przeciwnym razie rozmiar roboczy jest zmniejszany do najbliższej niższej wielokrotności 45
- ECC Level: M (Medium Error Correction)
- Kodowanie: UTF-8
- Renderer: `SkiaSharp` (domyślny) lub `ImageSharp`; wybór realizowany przez ustawienie `QrCode.Renderer` w `appsettings.json`

4.7.1. Generowanie kodu QR #1 (GET)

Endpoint: `GET /api/QrCode/get/{serverType}/{nip}/{issueDate}`

Opis: Generuje kod QR zawierający link do weryfikacji faktury online w systemie KSEF.

Parametry:

- `serverType` (path, wymagany) - Typ środowiska: `prod`, `demo`, `test`

- `nip` (path, wymagany) - NIP wystawcy faktury (10 cyfr)
- `issueDate` (path, wymagany) - Data wystawienia faktury w formacie `dd-MM-yyyy`
- `invoiceHash` (query, wymagany) - Hash SHA-256 faktury (z endpoint Invoices, pole `invoiceHash`)
- `ksefNo` (query, opcjonalny) - Numer KSeF do wyświetlenia pod kodem QR jako etykieta, może to być też dowolny inny tekst, np. OFFLINE
- `size` (query, opcjonalny) - Rozmiar kodu w pikselach
  - Domyślnie: 270
  - Zakres: 100-2000
  - Rekomendacja: wielokrotność 45 dla ostrych krawędzi
  - Gdy nie jest wielokrotnością 45: rozmiar roboczy jest zmniejszany do najbliższej niższej wielokrotności 45

Odpowiedź:

- Status: `200 OK`
- Content-Type: `image/bmp`
- Treść: Obraz kodu QR w formacie BMP

Kody błędów:

- `400 Bad Request` - Rozmiar kodu QR poza zakresem 100-2000

Przykład wywołania:

```
GET /api/QrCode/get/test/5732296089/15-12-2024?invoiceHash=abc123def456...&ksefNo=KSeF-123456&size=400
```

4.7.2. Generowanie kodu QR #1 (POST)

Endpoint: `POST /api/QrCode/post/{serverType}`

Opis: Generuje kod QR #1 (weryfikacja online). Alternatywa dla metody GET, używana gdy parametry są zbyt długie lub preferowany jest POST.

Parametry:

- `serverType` (path, wymagany) - Typ środowiska: `prod`, `demo`, `test`
- Body (JSON):

```
{
  "nip": "5732296089",
  "issueDate": "15-12-2024",
  "invoiceHash": "abc123def456...",
  "ksefNo": "KSeF-123456",
  "size": 400
}
```

Pola body:

- `nip` (string, wymagany) - NIP wystawcy faktury (10 cyfr)
- `issueDate` (string, wymagany) - Data wystawienia w formacie `dd-MM-yyyy`
- `invoiceHash` (string, wymagany) - Hash SHA-256 faktury
- `ksefNo` (string, opcjonalny) - Numer KSeF do wyświetlenia pod kodem QR jako etykieta, może to być też dowolny inny tekst, np. OFFLINE
- `size` (integer, opcjonalny) - Rozmiar w pikselach (100-2000, domyślnie 270); dla ostrych krawędzi zalecana jest wielokrotność 45, inaczej rozmiar roboczy jest zmniejszany do najbliższej niższej wielokrotności 45

Odpowiedź:

- Status: `200 OK`
- Content-Type: `image/bmp`
- Treść: Obraz kodu QR

Przykład wywołania:

```
POST /api/QrCode/post/test
Content-Type: application/json

{
  "nip": "5732296089",
  "issueDate": "15-12-2024",
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "ksefNo": "KSeF-123456-789",
  "size": 400
}
```

4.7.3. Generowanie zawartości kodu QR #1

Endpoint: `POST /api/QrCode/postContent/{serverType}`

Opis: Zwraca ciąg znaków (URL), który powinien zostać umieszczony w kodzie QR #1. Przydatne gdy chcesz wygenerować kod QR we własnym systemie.

Parametry:

- `serverType` (path, wymagany) - Typ środowiska
- Body (JSON): Identyczny jak w 4.7.2

Odpowiedź:

- Status: `200 OK`
- Content-Type: `text/plain`
- Treść: URL weryfikacyjny

Przykład odpowiedzi:

```
https://ksef-test.mf.gov.pl/web/verify/5732296089/2024-12-15/QSS1NTy1DruETws1yrWJJt%2FjlTFFJ%2FDP2%2FFasvuAqH70%
```

Przykład wywołania:

```
POST /api/QrCode/postContent/test
Content-Type: application/json

{
  "nip": "5732296089",
  "issueDate": "15-12-2024",
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70="
}
```

4.7.4. Generowanie kodu QR #2 (z certyfikatem)

Endpoint: `POST /api/QrCode/postVer/{serverType}`

Opis: Generuje kod QR #2 zawierający link weryfikacyjny z podpisem cyfrowym (ECDSA) opartym na certyfikacie. Zapewnia wyższy poziom bezpieczeństwa.

Parametry:

- `serverType` (path, wymagany) - Typ środowiska: `prod`, `demo`, `test`
- Body (JSON):

Content-Type: `application/json`

Pola JSON:

- `nip` (string, wymagany) - NIP wystawcy faktury (10 cyfr)
- `invoiceHash` (string, wymagany) - Hash faktury (Base64)
- `certificate` (string, wymagany) - Certyfikat X.509 w formacie PEM, zakodowany w Base64
- `password` (string, wymagany) - Hasło do klucza prywatnego w certyfikacie, zakodowany w Base64
- `privateKey` (string, opcjonalny) - Klucz prywatny ECDSA w formacie PEM, zakodowany w Base64 (jeśli nie jest wbudowany w certyfikat)
- `size` (integer, opcjonalny) - Rozmiar kodu QR w pikselach (100-2000, domyślnie 270); dla ostrych krawędzi zalecana jest wielokrotność 45, inaczej rozmiar roboczy jest zmniejszany do najbliższej niższej wielokrotności 45

Odpowiedź:

- Status: `200 OK`
- Content-Type: `image/bmp`
- Treść: Kod QR z etykietą “CERTYFIKAT”

Przykład wywołania:

```
POST /api/QrCode/postVer/test
Content-Type: application/json

{
  "nip": "5732296089",
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "certificate": "MIIDXTCCAkW...",
  "password": "aGFzbG8xMjM=",
  "size": 400
}
```

4.7.5. Generowanie zawartości kodu QR #2

Endpoint: `POST /api/QrCode/postVerContent/{serverType}`

Opis: Zwraca ciąg znaków (URL z podpisem), który powinien zostać umieszczony w kodzie QR #2. Przydatne gdy chcesz wygenerować kod QR we własnym systemie.

Parametry:

- `serverType` (path, wymagany) - Typ środowiska: `prod`, `demo`, `test`
- Body (JSON): Identyczny jak w 4.7.4

Content-Type: `application/json`

Odpowiedź:

- Status: `200 OK`
- Content-Type: `text/plain`
- Treść: URL weryfikacyjny z podpisem ECDSA

Przykład wywołania:

```
POST /api/QrCode/postVerContent/test
Content-Type: application/json

{
  "nip": "5732296089",
  "invoiceHash": "QSS1NTy1DruETws1yrWJJt/jlTFFJ/DP2/FasvuAqH70=",
  "certificate": "MIIDXTCCAkW...",
  "password": "aGFzbG8xMjM="
}
```

4.8. Version

Ścieżka bazowa: `/api/Version`

Kontroler odpowiedzialny za informacje o wersji aplikacji.

4.8.1. Pobierz wersję aplikacji

Endpoint: `GET /api/Version`

Opis: Zwraca informacje o wersji aplikacji oraz dacie kompilacji. Jest to endpoint informacyjny. Do sprawdzania stanu aplikacji należy używać endpointu **Health** (sekcja 4.9).

Parametry: Brak

Odpowiedź:

- Status: `200 OK`
- Format: JSON

```
{
  "versionId": "1.2.0",
  "buildDate": "2024-12-08T12:30:23.705232"
}
```

Pola odpowiedzi:

- `versionId` - Wersja aplikacji (Semantic Versioning)
- `buildDate` - Data i czas kompilacji aplikacji (ISO 8601)

Przykład wywołania:

```
GET /api/Version
```

Adresy dla różnych środowisk:

- Testowy: `https://testrsahelper.unitsoftware.com.pl/api/Version`
- Lokalny: `http://localhost:8080/api/Version`

4.9. Health

Ścieżka bazowa: `/health`

Kontroler odpowiedzialny za monitorowanie stanu aplikacji (używany przez Docker HEALTHCHECK).

4.9.1. Health check

Endpoint: `GET /health`

Opis: Endpoint zwraca szczegółowy stan aplikacji i jej zależności. Używany przez mechanizm Docker HEALTHCHECK do monitorowania dostępności kontenera. Kod HTTP pozostaje `200 OK`, a stan logiczny należy odczytywać z pola `Status`.

Parametry: Brak

Odpowiedź:

- Status: `200 OK`
- Format: JSON


```
{
  "Status": "Healthy",
  "Timestamp": "2026-04-16T12:00:00.0000000Z",
  "OfflineMode": false,
  "MinistryServers": [
    {
      "Url": "https://api-demo.ksef.mf.gov.pl/api/v2",
      "Available": true,
      "StatusCode": 200,
      "ResponseTimeMs": 124,
      "Error": null
    }
  ],
  "LocalCertificates": [],
  "AllowedNips": {
    "Available": true,
    "CheckedAt": "2026-04-16T12:00:00.0000000Z",
    "Count": 42,
    "Error": null
  },
  "PdfGeneration": {
    "Available": true,
    "CheckedAt": "2026-04-16T12:00:00.0000000Z",
    "Error": null
  },
  "QrGeneration": {
    "Available": true,
    "CheckedAt": "2026-04-16T12:00:00.0000000Z",
    "Error": null
  }
}
```


Pola odpowiedzi:

- `Status` - Stan aplikacji: `Healthy` albo `Degradated`
- `Timestamp` - Czas wygenerowania odpowiedzi w UTC (ISO 8601)
- `OfflineMode` - Informacja czy aplikacja działa w trybie offline
- `MinistryServers` - Lista statusów zewnętrznych serwerów sprawdzanych przez aplikację
- `MinistryServers[].Url` - Adres sprawdzanego serwera
- `MinistryServers[].Available` - Czy serwer odpowiedział poprawnie
- `MinistryServers[].StatusCode` - Kod HTTP zwrócony przez serwer, jeśli odpowiedź została otrzymana
- `MinistryServers[].ResponseTimeMs` - Czas odpowiedzi w milisekundach
- `MinistryServers[].Error` - Komunikat błędu lub `null`

- `LocalCertificates` - Status lokalnych certyfikatów dla środowisk `DEMO`, `TEST`, `PROD`; w trybie online tablica jest zwykle pusta
- `LocalCertificates[].Environment` - Nazwa środowiska
- `LocalCertificates[].Available` - Czy plik certyfikatów jest dostępny i możliwy do odczytania
- `LocalCertificates[].HasValidCertificates` - Czy dla środowiska istnieje co najmniej jeden ważny certyfikat
- `LocalCertificates[].TotalCount` - Łączna liczba certyfikatów w pliku
- `LocalCertificates[].ValidCount` - Liczba aktualnie ważnych certyfikatów
- `LocalCertificates[].Error` - Kod/problem diagnostyczny lub `null`
- `AllowedNips` - Status załadowanej listy dozwolonych NIP-ów
- `AllowedNips.Available` - Czy lista NIP-ów jest dostępna i nie jest pusta
- `AllowedNips.CheckedAt` - Czas wykonania sprawdzenia
- `AllowedNips.Count` - Liczba wpisów na liście
- `AllowedNips.Error` - Komunikat błędu lub `null`
- `PdfGeneration` - Status mechanizmu generowania PDF
- `PdfGeneration.Available` - Czy test generowania PDF zakończył się powodzeniem
- `PdfGeneration.CheckedAt` - Czas ostatniego sprawdzenia
- `PdfGeneration.Error` - Komunikat błędu lub `null`
- `QrGeneration` - Status mechanizmu generowania kodów QR
- `QrGeneration.Available` - Czy test generowania kodów QR zakończył się powodzeniem
- `QrGeneration.CheckedAt` - Czas ostatniego sprawdzenia
- `QrGeneration.Error` - Komunikat błędu lub `null`

Uwagi:

- W trybie online na wynik `Status` wpływa dostępność serwerów ministerstwa, lista NIP-ów, generowanie PDF oraz generowanie QR
- W trybie offline na wynik `Status` wpływa ważność lokalnych certyfikatów, lista NIP-ów, generowanie PDF oraz generowanie QR
- Informacje o `MinistryServers` są zwracane także w trybie offline, ale nie decydują wtedy o stanie końcowym

Przykład wywołania:

```
GET /health
```

Konfiguracja Docker HEALTHCHECK:

```
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
  CMD curl -f http://localhost:8080/health || exit 1
```

Użycie z Docker:

```
# Sprawdzenie stanu kontenera
docker ps

# Sprawdzenie logów health check
docker inspect --format='{{.State.Health.Status}}' rsahelper-api
```

---

5. Wersje aplikacji

Historia wersji

Wersja 1.2.7

- Dodano konfigurowalny rozmiar wyświetlanych kodów QR (parametr `qrCodeSize` oraz `PDFSettings.DefaultQrCodeSize`)
- Dodano wyświetlanie etykiety numeru KSeF w języku dokumentu ("Numer KSeF" / "KSeF Number" zależnie od `lang`)
- Wprowadzono nową konfigurację logów (NLog) z dzienną rotacją plików i przechowywaniem logów z ostatnich 7 dni
- Dodano możliwość logowania szczegółów wywołań WebAPI (`RequestLoggingMiddleware`)

Wersja 1.2.6

- Dodano uwierzytelnianie certyfikatem klienta w KSeF 2.0
- Dodano konfigurowalny rozmiar czcionki wydruku dla alternatywnego szablonu FA(3) (parametr `fontSize` oraz `PDFSettings.DefaultFontSize`)
- Dodano nowy renderer kodów QR oparty o ImageSharp jako alternatywę dla SkiaSharp (`QrCode.Renderer`)
- Rozszerzono endpoint `/health` o szczegółowe testy gotowości generowania PDF, kodów QR oraz listy dozwolonych NIP-ów
- Dodano obsługę wyświetlania numeru KSeF na fakturze w alternatywnym szablonie
- Naprawiono generowanie PDF i kodów QR w kontenerach Linux/Docker (domyślny silnik PDF zmieniony na Syncfusion, poprawiona zgodność wersji bibliotek natywnych)
- Naprawiono brakujące pola JST/GV w alternatywnym szablonie FA(3)

Wersja 1.2.5

- Dodano nowy silnik do generowania plików PDF
- Dodatkowe ustawienia przy generowaniu PDF
- Dodatkowe ustawienia dla generowania kodów QR (rodzina i rozmiar czcionek)

Wersja 1.2.4

- Dodano tryb offline umożliwiający pracę bez dostępu do internetu
- Trzy tryby działania: Online, Hybrid (z fallback do sieci), Strict Offline
- Obsługa lokalnych certyfikatów KSeF (pobieranie z plików JSON zamiast z API)
- Lokalne rozwiązywanie schematów XSD/XSL (bez konieczności dostępu do crd.gov.pl)
- Obsługa domyślnego języka dla wizualizacji

Wersja 1.2.3

- Dodano endpoint do szyfrowania paczek faktur ZIP (`POST /api/Invoices/package/{serverType}`)
- Umożliwienie wysyłania wielu faktur w jednej paczce do KSeF

Wersja 1.2.2

- Naprawiono generowanie PDF na systemach Linux
- Dodano automatyczne wykrywanie systemowego Chromium
- Dodano obsługę multiplatformową (Windows/Linux) dla konwersji PDF
- Zaktualizowano dokumentację o wymagania Chromium

Wersja 1.2.1

- Aktualizacja bibliotek
- Implementacja proxy

Wersja 1.2.0

- Zaktualizowana dokumentacja API
- Dodanie nowych endpointów QR Code (nowa struktura)
- Wsparcie dla Docker
- Upgrade do .NET 10.0
- Nowe endpointy Authorization: `upload` , `NipListFile`
- Dodanie endpoint `/health` dla Docker HEALTHCHECK

Wersja 1.1.0

- Wsparcie dla KSEF 2.0 (nowa grupa endpointów Invoices)
- Szyfrowanie AES-256-CBC dla faktur
- Generowanie danych kryptograficznych (GenerateEncryptionData)
- Walidacja FA(3)_v1-0E

Wersja 1.0.6

- Dodane metody do generowania kodów QR (getHash, getImage)
- Wsparcie dla ECCLevel M w kodach QR

Wersja 1.0.5

- Dodanie wizualizacji e-faktury w formacie PDF
- Usunięcie parametru `documentType` z metody ConvertToHtml (automatyczne wykrywanie)
- Integracja z Syncfusion HtmlToPdfConverter

Wersja 1.0.4

- Pierwsza wersja produkcyjna
- Obsługa algorytmu RSA (autoryzacja w systemie KSEF)
- Konwersja XML do HTML

- Walidacja plików FA(1)_v1-0E oraz FA(2)_v1-0E

Licencje

Aplikacja wykorzystuje następujące biblioteki komercyjne:

- **Syncfusion HtmlToPdfConverter** - Licencja wbudowana w aplikację

Data ostatniej aktualizacji dokumentacji: 2026-08-15

Wersja dokumentacji: 1.4

Zgodność z wersją aplikacji: 1.2.7+