Flowable w Docker Compose na docker01: Flowable UI, PostgreSQL i Traefik

Poziom: średnie

Dotyczy: flowable/flowable-ui:6.8.0, flowable/flowable-rest:8.0.0, postgres:16.15, Traefik v3.5 na docker01, restic 0.19.1

Czas: ~60 min

Uruchamiamy Flowable na `docker01` jako otwartą, samodzielnie hostowaną alternatywę dla Camundy: jeden kontener z modelerem, listą zadań i administracją w przeglądarce, baza PostgreSQL obok i wejście przez Traefika pod `https://flowable.zielona.example`. Na nim pani Halina z Olą modelują i uruchamiają pierwszy proces akceptacji faktury. Wracamy tu przy zmianie logowania, kopii i decyzji, czy zostajemy przy obrazie UI z 2022 roku, czy przechodzimy na silnik 8.0.

> W naszym środowisku dane aplikacji trzymamy w `/docker/apps/flowable`, a plik Compose w `/docker/compose/flowable/compose.yaml`. To konwencja, nie wymóg Dockera: można ją zmienić, byle w całej firmie była jedna. Katalog `/docker` to osobny wolumen `vgDocker/lvDocker`, żeby pełny dysk aplikacji nie zatrzymał systemu.

> Obraz `flowable/flowable-ui` ma ostatnią wersję 6.8.0 z grudnia 2022 roku. Od Flowable 7.0.0 (wrzesień 2023) aplikacje UI nie należą do otwartego projektu i nie dostają poprawek. Stawiamy go tylko w sieci wewnętrznej, za Traefikiem, bez wystawiania do internetu, a decyzję o dalszej drodze opisuje sekcja „Dwie drogi Flowable”.

Skróty użyte na tej stronie (13)

Skrót	Co znaczy
AD	Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą

Skrót	Co znaczy
BPMN	Business Process Model and Notation: notacja graficzna do rysowania procesów biznesowych, którą wykonuje silnik Camundy
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
CMMN	Case Management Model and Notation: notacja OMG do spraw, w których kolejność kroków nie jest z góry ustalona; wykonuje ją między innymi Flowable
DMN	Decision Model and Notation: notacja OMG do zapisu decyzji w tabelach, które silnik procesów sprawdza wiersz po wierszu, na przykład kto akceptuje fakturę według kwoty
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
GUID	Globally Unique Identifier: unikalny w skali globalnej identyfikator obiektu, tu konkretnego protektora klucza BitLockera
OAuth2	Open Authorization 2.0: standard, w którym użytkownik dostaje od zaufanego serwera tymczasowy token dostępu do usługi zamiast podawania jej hasła
OIDC	OpenID Connect: protokół logowania przez zewnętrznego dostawcę tożsamości, na którym opiera się funkcja SSO w Vaultwardenie
REST	sposób budowania API przez zwykłe żądania HTTP (GET, POST) zamiast osobnego protokołu
XML	eXtensible Markup Language: format tekstowy do zapisu ustrukturyzowanych danych w znacznikach, tutaj używany przez PowerShell do zapisu zrzutu obiektów poleceniem Export-Clixml
XOR	Exclusive OR: w BPMN bramka wyłączna, która z kilku dróg wybiera dokładnie jedną według warunku

Fiszka

Komendy wykonujemy z CLI bash jako użytkownik z uprawnieniami sudo na `docker01`, w katalogu `/docker/compose/flowable`, chyba że karta mówi inaczej.

□□ ZMIENIA SYSTEM

Założenie katalogów aplikacji: mkdir -p

```
sudo mkdir -p /docker/apps/flowable/postgres /docker/apps/flowable/kopie
/docker/compose/flowable
```

Katalog na dane bazy i na zrzuty pod `/docker/apps`, katalog na przepis pod `/docker/compose`, jak w każdej aplikacji Zielonej.

□□ **mkdir** = make directory: utwórz katalog

□□ ZMIENIA SYSTEM

Uruchomienie Flowable: docker compose up -d

```
docker compose up -d
```

Pobiera obrazy w przypiętych wersjach i uruchamia bazę, a po jej gotowości aplikację.

□□ **docker** = narzędzie do kontenerów

□□ TYLKO ODCZYT

Śledzenie startu aplikacji: docker compose logs -f

```
docker compose logs -f server
```

Flowable przy pierwszym starcie zakłada kilkaset tabel w bazie, co trwa około minuty. Start kończy wpis Spring Boot o uruchomionej aplikacji.

Przykład (skrócony):

```
flowable  | ... Started FlowableUiApplication in 48.211 seconds
```

□□ **docker** = narzędzie do kontenerów

□□ TYLKO ODCZYT

Sprawdzenie odpowiedzi przez Traefika: curl -I

```
curl -sI https://flowable.zielona.example/flowable-ui/
```

Odpowiedź 200 albo 302 na stronę logowania potwierdza drogę od Traefika do kontenera. 502 albo 404 opisuje strona o częstych problemach.

□□ **curl** = client URL: pobieranie i wysyłanie przez HTTP z terminala

□□ ZMIENIA SYSTEM

Zrzut bazy Flowable: docker compose exec pg_dump

```
docker compose exec -T db pg_dump -U flowable -Fc flowable >
/docker/apps/flowable/kopie/flowable.dump
```

Spójny zrzut działającej bazy w formacie własnym `pg_dump`. Zabiera go potem restic razem z resztą `/docker/apps`.

docker = narzędzie do kontenerów

OSTROŻNIE · NIEODWRACALNE

Odtworzenie bazy ze zrzutu: pg_restore

```
docker compose exec -T db pg_restore -U flowable -d flowable --clean --if-exists <
/docker/apps/flowable/kopie/flowable.dump
```

Usuwa obiekty z bazy i zakłada je ze zrzutu. Wykonujemy po zatrzymaniu aplikacji (`docker compose stop server`), żeby Flowable nie pisał w trakcie.

docker = narzędzie do kontenerów

Z życia wzięte

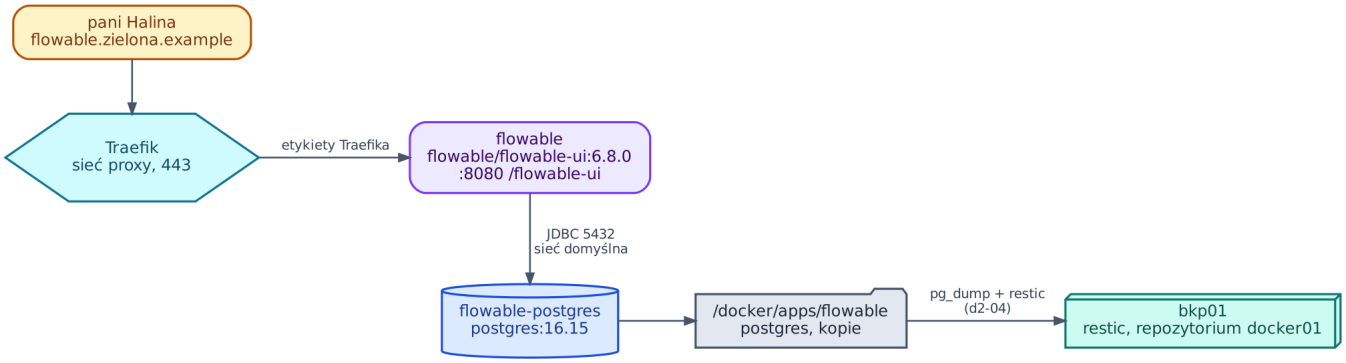
Z ŻYCIA WZIĘTE

Środa, 13:10. Pani Halina siada z Olą przy jednym monitorze i w modelerze Flowable rysuje obieg faktury kosztowej, a po dwudziestu minutach klika „Publish” i uruchamia pierwszy proces na fakturze za paliwo. Zadanie „Sprawdzenie formalne” pojawia się na jej liście zadań, a po zatwierdzeniu trafia do Marka, bo kategoria to transport. Pan Zbigniew zagląda przez ramię i pyta tylko, czy to jest ten bezpłatny wariant; jest, ale Ola od razu dodaje, że to wersja interfejsu z 2022 roku i do produkcji trzeba będzie zdecydować, co dalej.

Jak to działa

Rozwiń: jak to działa pod spodem

Jeden kontener, jedna baza, jeden adres



Rysunek 1. Flowable na docker01: pani Halina otwiera flowable.zielona.example, Traefik w sieci proxy kieruje ruch według etykiet do kontenera flowable (flowable-ui 6.8.0, port 8080, ścieżka /flowable-ui), który łączy się z bazą flowable-postgres (postgres 16.15) w sieci domyślnej projektu; dane bazy leżą w /docker/apps/flowable, a zrzut z restikiem idzie na bkp01 (kliknięcie otwiera pełny rozmiar)

Kontener `flowable` to aplikacja Spring Boot, w której są cztery części: modeler (procesy BPMN, tabele DMN, formularze, aplikacje), lista zadań, administracja silnika i konta (IDM). Aplikacja słucha na porcie 8080 pod ścieżką `/flowable-ui`, a bazę widzi po nazwie usługi `db` w sieci domyślnej projektu. Do Traefika dochodzi przez wspólną sieć `proxy`, opisaną na stronie Reverse proxy w Dockerze: Traefik przed wszystkimi aplikacjami na docker01.

> **Pojęcie: aplikacja Flowable (app definition).** Paczka w modelerze Flowable, która zbiera kilka modeli (procesy, decyzje, formularze) i publikuje je razem do silnika. Dopiero opublikowana aplikacja pojawia się na liście zadań jako kafelek, z którego użytkownik uruchamia proces.

> **Pojęcie: uprawnienie Flowable (privilege).** Prawo wejścia do części aplikacji: `access-modeler`, `access-task`, `access-admin`, `access-idm`, `access-rest-api`. Nadajemy je użytkownikom albo grupom w części IDM, a przy logowaniu OAuth2 wynikają z atrybutu tokenu.

Dwie drogi Flowable

Droga	Co dostajemy	Za	Przeciw
`flowable-ui:6.8.0` (ta strona)	modeler, lista zadań, administracja w przeglądarce	działa od razu, pani Halina sama modeluje	wersja z grudnia 2022, bez poprawek bezpieczeństwa, silnik 6.8
`flowable-rest:8.0.0` (krok 7)	aktualny silnik z REST API i Swaggerem	poprawki, Spring Boot 4	bez interfejsu: modelowanie gdzie indziej, lista zadań do napisania przez zespół Oli
produkty komercyjne Flowable	aktualne aplikacje i modeler	utrzymywane	licencja, warunki sprawdzamy u producenta

Twórcy Flowable piszą na forum, że modeler 6.x tworzy XML BPMN i CMMN zgodny z silnikiem 7, ale nie gwarantują tego na przyszłość. Rozsądny układ na pilotaż to więc modelowanie w `flowable-ui:6.8.0` i równoległe sprawdzenie tych samych plików na `flowable-rest:8.0.0`.

Przebieg

Zanim zaczniemy

1. Traefik działa na `docker01` z siecią `proxy` i certyfikatem `*.zielona.example`.
2. Rekord `flowable.zielona.example` w wewnętrznej strefie `zielona.example` na `dc01` wskazuje 10.20.10.42 (`Add-DnsServerResourceRecordA -ZoneName "zielona.example" -Name "flowable" -IPv4Address "10.20.10.42"`).
3. Dwa hasła wygenerowane w Vaultwardenie: do bazy i do konta `admin` Flowable.

1. Katalogi i plik .env

Katalogi zakładamy kartą z Fiszki. Plik `/docker/compose/flowable/.env`:

```
FLOWABLE_DB_HASLO=HasloBazyFlowable
FLOWABLE_ADMIN_HASLO=HasloAdminaFlowable
```

```
sudo chmod 600 /docker/compose/flowable/.env
```

Domyślne hasła z przykładów Flowable (`flowable` dla bazy, `test` dla konta `admin`) zastępujemy od pierwszego uruchomienia, bo zmiana hasła administratora po starcie wymaga już zmiany w aplikacji IDM.

2. Plik compose.yaml

Plik `/docker/compose/flowable/compose.yaml`, oparty na przykładzie `docker/config/ui-postgres.yml` z wydania 6.8.0 silnika Flowable:

```
services:
  server:
    image: flowable/flowable-ui:6.8.0
    container_name: flowable
    restart: unless-stopped
    depends_on:
      db:
        condition: service_healthy
    environment:
      - SERVER_PORT=8080
      - SERVER_FORWARD_HEADERS_STRATEGY=native
      - SPRING_DATASOURCE_DRIVER-CLASS-NAME=org.postgresql.Driver
      - SPRING_DATASOURCE_URL=jdbc:postgresql://db:5432/flowable
      - SPRING_DATASOURCE_USERNAME=flowable
      - SPRING_DATASOURCE_PASSWORD=${FLOWABLE_DB_HASLO}
      - FLOWABLE_COMMON_APP_IDM-ADMIN_USER=admin
      - FLOWABLE_COMMON_APP_IDM-ADMIN_PASSWORD=${FLOWABLE_ADMIN_HASLO}
    networks:
      - default
      - proxy
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.flowable.rule=Host(`flowable.zielona.example`)"
      - "traefik.http.routers.flowable.entrypoints=websecure"
      - "traefik.http.routers.flowable.tls=true"
      - "traefik.http.routers.flowable.middlewares=flowable-start"
      - "traefik.http.middlewares.flowable-
start.redirectregex.regex=^https://flowable.zielona.example/$$"
      - "traefik.http.middlewares.flowable-
start.redirectregex.replacement=https://flowable.zielona.example/flowable-ui/"
      - "traefik.http.services.flowable.loadbalancer.server.port=8080"


  db:
    image: postgres:16.15
```

```
    container_name: flowable-postgres
    restart: unless-stopped
    environment:
      - POSTGRES_DB=flowable
      - POSTGRES_USER=flowable
      - POSTGRES_PASSWORD=${FLOWABLE_DB_HASLO}
    volumes:
      - /docker/apps/flowable/postgres:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U flowable -d flowable"]
      interval: 10s
      timeout: 5s
      retries: 6

networks:
  default:
  proxy:
    external: true
```

Fragment	Dlaczego
`flowable/flowable-ui:6.8.0`	ostatni obraz UI; tag przypięty, bez `latest`, choć tu `latest` też wskazuje 6.8.0
`postgres:16.15`	ta sama główna wersja co `pg01` (PostgreSQL 16), najnowsza poprawka z 1 listopada 2026; przykład Flowable używał 9.6, której nie stawiamy
`depends_on` z `service_healthy`	zastępuje skrypt `wait-for-something.sh` z przykładu Flowable: aplikacja startuje dopiero, gdy baza odpowiada
`SERVER_FORWARD_HEADERS_STRATEGY=native`	Spring Boot bierze schemat i host z nagłówków `X-Forwarded-*` od Traefika, więc przekierowania wskazują `https://flowable.zielona.example`, a nie `http://`
brak `ports`	ruch wchodzi tylko przez Traefika; port 8080 nie jest osiągalny z sieci
`$$` w `redirectregex`	Compose traktuje pojedynczy `$` jako zmienną, a `$$` przekazuje znak dolara do Traefika

Przekierowanie z `/` na `/flowable-ui/` jest wygodą: bez niego pod samym adresem Flowable odpowiada 404, jak każda aplikacja ze ścieżką kontekstową.

3. Pierwsze uruchomienie

```
cd /docker/compose/flowable
docker compose config --quiet
docker compose up -d
docker compose logs -f server
docker compose ps
curl -sI https://flowable.zielona.example/flowable-ui/
```

`docker compose config --quiet` bez wyniku znaczy, że plik i zmienne z `.env` są poprawne. Logujemy się w przeglądarce kontem `admin` z hasłem z `.env`.

◫◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: strona startowa Flowable UI pod `https://flowable.zielona.example/flowable-ui/` po zalogowaniu, widoczne cztery kafelki: Task app, Modeler app, Admin app, IDM app.

4. Konta i grupy

W części IDM zakładamy konta i grupy dla pilotażu. Konta lokalne Flowable to najprostsza droga; logowanie kontem firmowym opisuje krok 5.

```
IDM app → Users → Create user: halina.kowalska, marek.zawadzki, ola.wisniewska
IDM app → Groups → Create group: ksiegowosc (pani Halina), magazyn (Marek), procesy (Ola)
IDM app → Privileges → access-modeler: grupa procesy; access-task: ksiegowosc, magazyn,
procesy
```

Nazwy kont są przykładowe, w Zielonej bierzemy je z AD. Konto `admin` zostaje tylko do administracji, nikt na nim nie pracuje.

5. Logowanie kontem firmowym

Flowable UI 6.8 umie logować przez OAuth2 (OIDC) według Spring Security. Przykład Flowable `ui-keycloak-postgres.yml` z wydania 6.8.0 pokazuje Keycloak; dla drogi bez Microsoftu to gotowy wzór.

Zmienna z przykładu Flowable	Co ustawia
`FLOWABLE.COMMON.APP.SECURITY.TYPE=OAUTH2`	logowanie przez dostawcę OIDC zamiast kont lokalnych
`SPRING.SECURITY.OAUTH2.CLIENT.REGISTRATION.KEYCLOAK.CLIENT-ID`, `CLIENT-SECRET`, `SCOPE`	rejestracja klienta u dostawcy
`SPRING.SECURITY.OAUTH2.CLIENT.PROVIDER.KEYCLOAK.ISSUER-URI` i `SPRING.SECURITY.OAUTH2.RESOURCESERVER.JWT.ISSUER-URI`	adres wystawcy (realm Keycloaka)

Zmienna z przykładu Flowable	Co ustawia
`SPRING.SECURITY.OAUTH2.CLIENT.PROVIDER.KEYCLOAK.USER-NAME-ATTRIBUTE=preferred_username`	które roszczenie jest nazwą użytkownika
`FLOWABLE.COMMON.APP.SECURITY.OAUTH2.AUTHORITIES-ATTRIBUTE=groups`, `...GROUPS-ATTRIBUTE=userGroups`	z których roszczeń brać uprawnienia i grupy Flowable

Wariant z Microsoft Entra ID to ta sama konfiguracja Spring Security z inną nazwą rejestracji i adresem wystawcy `https://login.microsoftonline.com/IdentyfikatorDzierzawy/v2.0`, rejestracja w Entra z adresem przekierowania Spring Security i role aplikacji (`access-task`, `access-modeler`) zamiast grup, bo grupy przychodzą z Entra jako identyfikatory GUID. Flowable tego wariantu nie opisuje, więc przed wdrożeniem sprawdzamy go w laboratorium (do potwierdzenia), a do tego czasu pilotaż działa na kontach lokalnych.

6. Pierwszy proces z modelera

Proces akceptacji faktury kosztowej w wersji na pilotaż:

```
Modeler app → Processes → Create Process: nazwa „Akceptacja faktury”, klucz akceptacja-faktury
```

1. Zdarzenie startowe z formularzem: numer faktury, kontrahent, kwota, kategoria.
2. Zadanie użytkownika „Sprawdzenie formalne”, grupa kandydatów `ksiegowosc`.
3. Bramka XOR z warunkami w wyrażeniach Flowable: `${kategoria == 'transport'}` do zadania „Akceptacja: magazyn” (grupa `magazyn`), `${kwota <= 5000}` do „Akceptacja: kierownik”, w pozostałych przypadkach „Akceptacja: zarząd”.
4. Na zadaniach akceptacji timer na brzegu, nieprzerywający, `P3D`, z zadaniem „Przypomnienie”.
5. Zdarzenie końcowe „Zaakceptowano”.

```
Modeler app → Apps → Create App: „Faktury” → Edit included models: Akceptacja faktury → Publish
Task app → Faktury → Start process → Akceptacja faktury
```

Po starcie pani Halina widzi zadanie „Sprawdzenie formalne” w Task app, a po jego zamknięciu zadanie przechodzi do właściwej grupy. W Admin app widać egzemplarz procesu i jego zmienne. Plik `.bpmn` eksportujemy z modelera do repozytorium w Gitea, żeby wersje procesu były też poza bazą.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: Modeler app z otwartym procesem „Akceptacja faktury”, widoczne zadanie „Sprawdzenie formalne”, bramka XOR z trzema wyjściami i timer na brzegu zadania akceptacji.

7. Wariant: silnik 8.0 bez interfejsu

Aktualny silnik stawiamy jako osobny projekt z własną bazą, nigdy na bazie Flowable UI 6.8, bo różne wersje silnika mają różne schematy tabel. Plik `/docker/compose/flowable-rest/compose.yaml` (katalogi `/docker/apps/flowable-rest/postgres` i `/docker/compose/flowable-rest` zakładamy jak w kroku 1, w `.env` hasła `FLOWABLE_REST_DB_HASLO` i `FLOWABLE_REST_ADMIN_HASLO`):

```
services:
  server:
    image: flowable/flowable-rest:8.0.0
    container_name: flowable-rest
    restart: unless-stopped
    depends_on:
      db:
        condition: service_healthy
    environment:
      - SERVER_PORT=8080
      - SERVER_FORWARD_HEADERS_STRATEGY=native
      - SPRING_DATASOURCE_DRIVER-CLASS-NAME=org.postgresql.Driver
      - SPRING_DATASOURCE_URL=jdbc:postgresql://db:5432/flowable
      - SPRING_DATASOURCE_USERNAME=flowable
      - SPRING_DATASOURCE_PASSWORD=${FLOWABLE_REST_DB_HASLO}
      - FLOWABLE_REST_APP_ADMIN_PASSWORD=${FLOWABLE_REST_ADMIN_HASLO}
    networks:
      - default
      - proxy
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.flowable-rest.rule=Host(`flowable.zielona.example`) &&
PathPrefix(`/flowable-rest`)"
      - "traefik.http.routers.flowable-rest.entrypoints=websecure"
      - "traefik.http.routers.flowable-rest.tls=true"
      - "traefik.http.services.flowable-rest.loadbalancer.server.port=8080"


  db:
    image: postgres:16.15
    container_name: flowable-rest-postgres
    restart: unless-stopped
    environment:
      - POSTGRES_DB=flowable
```

```
      - POSTGRES_USER=flowable
      - POSTGRES_PASSWORD=${FLOWABLE_REST_DB_HASLO}
    volumes:
      - /docker/apps/flowable-rest/postgres:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U flowable -d flowable"]
      interval: 10s
      timeout: 5s
      retries: 6

networks:
  default:
  proxy:
    external: true
```

Aplikacja ma ścieżkę `/flowable-rest`, API silnika procesów pod `/service`, dokumentację Swagger pod `/flowable-rest/docs/` i konto `rest-admin` (domyślnie z hasłem `test`, które zmieniamy zmienną `FLOWABLE_REST_APP_ADMIN_PASSWORD`, odpowiednikiem właściwości `flowable.rest.app.admin.password`). Reguła z `PathPrefix` jest dłuższa niż sama reguła hosta, więc Traefik wybiera ją przed regułą Flowable UI. Plik z modelera wdrażamy kartą „Wdrożenie pliku BPMN do Flowable REST” ze strony BPMN po ludzku i wybór silnika: Camunda 8, Flowable i reszta.

8. Kopia zapasowa

Flowable trzyma wszystko w bazie: modele, wdrożenia, egzemplarze, historię i konta. Kopia to więc zrzut bazy plus katalog `/docker/compose/flowable`. Do skryptu ze strony Kopia całego docker01: /docker/apps, /docker/compose i bazy w kontenerach dopisujemy zrzut i wykluczenie plików bazy:

```
docker compose -f /docker/compose/flowable/compose.yaml exec -T db \
  pg_dump -U flowable flowable > "${ZRZUTY}/flowable.sql"
```

oraz w poleceniu `restic backup` opcję `--exclude=/docker/apps/flowable/postgres` (tak samo dla `flowable-rest`, jeśli stoi). Kopia trafia wtedy do repozytorium `docker01` z poziomem 90 dni. Historia akceptacji faktur może być dowodem dla pana Wojtka, więc autor może zdecydować o przeniesieniu Flowable do repozytorium `bazy` (180 dni) albo dłuższego.

9. Aktualizacja

Dla `flowable-ui` aktualizacji nie ma: 6.8.0 jest ostatnia. Aktualizujemy tylko obraz bazy w obrębie wersji 16 (zmiana tagu, zrzut przed, `docker compose pull` i `up -d`). Dla `flowable-rest` czytamy informacje o wydaniu na GitHubie, robimy zrzut i zmieniamy tag; silnik przy starcie sam podnosi schemat bazy, czego nie da się cofnąć bez zrzutu.

Weryfikacja

```
cd /docker/compose/flowable
docker compose ps
docker compose images
curl -sI https://flowable.zielona.example/
curl -sI https://flowable.zielona.example/flowable-ui/
ls -l /docker/apps/flowable/kopie/
```

Oczekujemy kontenerów `flowable` i `flowable-postgres` w stanie `running` (baza `healthy`), obrazów `6.8.0` i `16.15`, przekierowania z `/` na `/flowable-ui/`, odpowiedzi strony logowania pod `/flowable-ui/` i świeżego zrzutu bazy po nocnej kopii. W Task app widać egzemplarz procesu z kroku 6.

Wycofanie zmian

```
cd /docker/compose/flowable
docker compose down
```

`down` usuwa kontenery i sieć domyślną projektu, a dane zostają w `/docker/apps/flowable`. Całość usuwamy dopiero po zrzucie i decyzji, że modele nie są potrzebne: katalogi `/docker/apps/flowable` i `/docker/compose/flowable`, rekord `flowable` w DNS i wpisy w skrypcie kopii.

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
Kontener `flowable` czeka i nie startuje	baza nie przechodzi `healthcheck`	`docker compose logs db`, uprawnienia katalogu `/docker/apps/flowable/postgres`
W logu `password authentication failed for user "flowable"`	baza założona wcześniej z innym hasłem, `POSTGRES_PASSWORD` działa tylko przy pierwszym starcie	hasło z pierwszego startu albo zmiana hasła w bazie `ALTER ROLE`

Objaw	Przyczyna	Co robimy
404 pod `https://flowable.zielona.example/`	brak przekierowania, aplikacja jest pod `/flowable-ui`	middleware `flowable-start` albo adres z `/flowable-ui/`
502 od Traefika	kontener poza siecią `proxy` albo zły port w etykiecie	sieć `proxy` w `compose.yaml`, port 8080 w `loadbalancer.server.port`
Po zalogowaniu przekierowanie na `http://` albo na adres kontenera	Spring Boot nie czyta nagłówków `X-Forwarded-*`	`SERVER_FORWARD_HEADERS_STRATEGY=native`
Pani Halina nie widzi Modeler app	brak uprawnienia `access-modeler`	IDM app → Privileges
Proces nie startuje z błędem wyrażenia	brak zmiennej użytej w warunku bramki (na przykład `kwota`)	pole formularza startowego o tym samym identyfikatorze co zmienna
`flowable-rest` odpowiada 401 dla `rest-admin`	hasło inne niż w `.env` albo baza z poprzedniego startu	hasło z pierwszego startu, konto w bazie

Ćwiczenia

1. Dlaczego w etykiecie `redirectregex` piszemy `$$` zamiast `$`?

Odpowiedź

Docker Compose podstawia zmienne zapisane jako `$NAZWA` albo `${NAZWA}`. Pojedynczy `$` na końcu wyrażenia zostałby potraktowany jako początek zmiennej, a `$$` przekazuje do Traefika dosłowny znak `$`, który w wyrażeniu regularnym oznacza koniec adresu.

1. Pan Zbigniew pyta, czy skoro Flowable UI jest bezpłatny, możemy na nim od razu prowadzić wszystkie faktury. Co odpowiadamy?

Odpowiedź

Licencja silnika na to pozwala, ale obraz UI ma wersję z grudnia 2022 roku i nie dostaje poprawek bezpieczeństwa. Do pilotażu w sieci wewnętrznej wystarcza; do produkcji trzeba wybrać: aktualny silnik 8.0 z własnym interfejsem od zespołu Oli, komercyjne aplikacje Flowable albo Camundę z licencją.

1. Dlaczego `flowable-rest:8.0.0` dostaje własną bazę zamiast bazy Flowable UI?

Odpowiedź

Bo silnik przy starcie dopasowuje schemat bazy do swojej wersji. Silnik 8.0 podniósłby schemat bazy używanej przez 6.8, a Flowable UI 6.8 nie umiałby już na niej pracować. Osobne bazy pozwalają porównać oba silniki bez ryzyka.

1. Gdzie leżą modele procesów narysowane w Flowable UI i co to znaczy dla kopii?

Odpowiedź

W bazie PostgreSQL, razem z wdrożeniami, egzemplarzami i kontami. Kopia katalogu `/docker/compose/flowable` bez zrzutu bazy nie zawiera ani jednego procesu, dlatego zrzut `pg_dump` jest obowiązkową częścią kopii, a eksport plików `.bpmn` do Gitea to dodatkowe zabezpieczenie i historia wersji.

Powiązane: Silniki procesów: od workflow i BPEL do Camundy 8 i Flowable · Jak trzymamy aplikacje w Dockerze: konwencje Zielonej · Reverse proxy w Dockerze: Traefik przed wszystkimi aplikacjami na docker01 · Kopia całego docker01: /docker/apps, /docker/compose i bazy w kontenerach · Hasła i sekrety poza repozytorium: .env, Docker secrets i Vaultwarden · Częste problemy z silnikami procesów: Traefik 502, Zeebe, PVC i OIDC

Źródła (odczyt 1 listopada 2026): github.com/flowable/flowable-engine, tag `flowable-6.8.0`: `docker/README.md` (adres `/flowable-ui/`, konto `admin`), `docker/config/ui-postgres.yml` i `docker/config/ui-keycloak-postgres.yml` (zmienne środowiskowe, OAuth2), `modules/flowable-ui/flowable-ui-app/src/main/resources/flowable-default.properties` (ścieżka `/flowable-ui`, `flowable.common.app.idm-admin.*`); tag `flowable-8.0.0`: `docker/README.md` i `docker/config/rest-postgres.yml` (adres `/flowable-rest/`, Swagger `/docs/`, `rest-admin`), `modules/flowable-app-rest/src/main/resources/flowable-default.properties` (`flowable.process.servlet.path=/service`, `flowable.rest.app.admin.*`); hub.docker.com, tagi `flowable/flowable-ui` (6.8.0 z 23 grudnia 2022), `flowable/flowable-rest` (8.0.0 z 27 lutego 2026) i `library/postgres` (16.15 z 1 listopada 2026); forum.flowable.org, „Flowable UI app maven repository does not have 7.0.0 support” (usunięcie aplikacji UI w 7.0, zgodność modelera 6.x); github.com/flowable/flowable-engine/releases (8.0.0: Spring Boot 4). Nazwy uprawnień `access-*` podajemy według aplikacji IDM w 6.8 i sprawdzamy w interfejsie.

Revision #2
Created 2026-11-01 18:02:14 UTC by Tomasz Konieczny
Updated 2026-11-01 18:03:39 UTC by Tomasz Konieczny