# Waluty

## I. Założenia

a) Program powinien być łatwy do zmodyfikowania, by czerpać dane z różnych źródeł. (Wszelkie konfiguracje łączności są zawarte w config.properties.)
b) Użytkownik powinien mieć dostęp do czytania bazy danych z poziomu aplikacji.
c) Program winien być wydajny, tj. nie zapisywać do bazy danych niepotrzebnych danych.
d) Program ma być w przyszłości łatwy do konfiguracji, tj. wszelkie dane w bazie danych powinny być albo typu VARCHAR (String) albo Integer (int).
e) Dane powinny być skatalogowane hierarchicznie, tj. każda encja waluty posiada zdefiniowany dokument z którego pochodzi, natomiast dokument posiada informacje dotyczące aktualizacji (HistoryEntity).

## II. Użyte technologie

a) Spring framework - szybkie tworzenie aplikacji internetowej. Dzięki tej technologii przy tworzeniu aplikacji mogę skupić się głównie na logice programu.
b) Widoki są skonfigurowane na *Java Servlet Pages*, dzięki czemu nie potrzebuję wielu widoków, by wyświetlać konkretne dane.
c) Spring boot - szybkie "postawienie" serwera.
d) Baza danych: postgreSQL
e) Spring MVC - dzięki niemu korzystam z JSP.
f) Rome / Rome Fetcher - służy ograniczeniu "ręcznego przeglądania" dokumentów XML. Dzięki tej technologii ograniczyłem przeglądanie głównego pliku z RSS.
g) JFreeChart - wyświetlanie wykresów za pomocą "czystej" Javy, tj. rysuję po stronie serwera, a następnie wyświetlam zdjęcia (w formacie .PNG).
h) Maven - proste budowanie zależności aplikacji.

## III. Jak przebiega aktualizacja?

1) Aktualizacja jest przeprowadzana automatycznie co 10 sekund (10000 milisekund). Timer ustawiony za pomocą adnotacji z wykorzystaniem *fixedDelay* zamiast *fixedRate.* Powodem jest potencjalna sytuacja: pobieranie dużej aktualizacji, która zajmuje ponad 10 sekund. W takiej sytuacji wątki zaczną na siebie nachodzić, co jest zbędne i niebezpieczne. Dzięki *fixedDelay* kolejne sprawdzenie aktualizacji rozpocznie się dopiero 10 sekund po zakończeniu poprzedniego sprawdzania.
   Klasa: AutoUpdate
   Metoda: autoUpdate()
   a) Jest możliwość ręcznej aktualizacji za pomocą strony "/document/update"

2) Następnie program przechodzi do metody checkDateOfMainXMLBuild() w klasie XMLDistributorService. Zostaje pobrana data dokumentu (z RSS) i przekazana do metody *saveHistoryEntity(String dateOfLastBuild)* w klasie HistoryService.

3) Metoda *saveHistoryEntity(String dateOfLastBuild)* na początku pobiera wszelkie linki z separatorów "enclosure" za pomocą metody *getLinksFromMainXML()* (Main czyli taki, którego adres jest zdefiniowany z Config.properties).

4) Sprawdzany jest warunek, czy aktualizacja jest potrzebna. By baza danych została zaktualizowana muszą być spełnione dwa warunki, czyli : żadna encja dokumentu (DocumentEntity) nie może posiadać linku odebranego w poprzednim punkcie oraz data stworzenia dokumentu (z którego pobierane są linki) musi być różna od tych, które już znajdują się w bazie danych.

a) Zakładam, że każda zmiana na stronie np.: http://rss.nbp.pl/kursy/TabelaA.xml sprawia, że wartość w separatorze <lastBuildDate> jest zmieniana.
b) Nawet gdyby data została zmieniona, ale nie byłoby nowego linku, aktualizacja by nie przebiegła.

5) Dane "historyEntity" są uzupełniane oraz każdy link jest wykorzystywany do stworzenia encji dokumentu. Wszystkie linki są przekazywane, nie jedynie "nowe". (Mój błąd podczas tworzenia rozwiązania).

6) W pętli są tworzone dokumenty dzięki metodzie *saveDocument(String link, HistoryEntity historyEntity)* w klasie DocumentEntityService. Dokument zostanie utworzony, jeżeli nie istnieje jeszcze taki, który posiada identyczny internalCode (<numer_tabeli> z dokumentu, np.: http://rss.nbp.pl/kursy/xml2/2017/a/17a094.xml).

7) Jeżeli dokument może zostać zapisany, jego dane są uzupełniane a następnie wydzielone waluty i zapis walut w klasie CurrencyService, metoda *saveCurrencyFromDocument(String link, DocumentEntity document)*

8) Dane waluty są wyszukane dzięki NodeList - ustawione ręcznie.

9) Jeżeli wszystko przebiegnie pomyślnie HistoryEntity zostanie utworzone wraz z swoimi "dziećmi": dokumentami, które posiadają informacje o walucie.

# IV. Zarządzanie dokumentami

Dokumenty są wyświetlane w porządku chronologicznym. Dzięki jednemu kliknięciu można przejść do wszelkich informacji o walutach z tegoż dokumentu.

Strony: /document/showAll - pokazuje wszystkie dokumenty
/document/{id}/show - pokazuje informacje o dokumencie o podanym id
/history/last - pokazuje ostatnią aktualizację
/history/all - pokazuje wszelkie aktualizacje

# X. TODO

1) Animowane wykresy;
2) Wartość waluty względem dnia poprzedniego;
3) Zmienienie config.properties by z tego pliku była pobierana nazwa separatora waluty, aktualnie jest ustawiona ręcznie.