README

Projekt dla międzynarodowego startup'u prowadzącego działalność w kilku krajach europejskich. Sektor przemysłu ciężkiego i logistyki. Aplikacja typu desktop, pierwsza wersja oddana na produkcję w Grudniu 2019, obecnie w użyciu wersja 4 (od 07/04/2020). Z uwagi na szereg danych (przetwarzanych i uwzględnionych) w kodzie (eg nazwy podmiotów oraz statystyki) stanowiących tajemnice przedsiębiorstwa, repozytorium na GitHub jest w trybie z ograniczonym dostępem.

I. Zakres projektu

Sponsorzy projektu: Prezes Zarządu Spółki, Treasurer Spółki oraz zewnętrzna księgowość (Główna Księgowa oraz osoba odpowiedzialna za spójność podatkową pomiędzy różnymi jurysdykcjami).

Tło projektu: Grupa prowadzi działalność handlową na terenie Europy obsługując znaczne i rozproszone strumienie płatnicze z różnych regionów. Treasurer jest odpowiedzialny za zebranie wszystkich dokumentów płatniczych z różnych obszarów gospodarczych (po stronie zakupu i sprzedaży) oraz generowanie okresowych statystyk ilościowych i jakościowych oraz finalnego zestawienia całego ruchu płatniczego do wyznaczonego dnia kolejnego miesiąca. Przed rozpoczęciem projektu, dokumenty były przetwarzane ręcznie w ilości ok **kilkuset sztuk miesięcznie** co konsumowało **ok 3/4 etatu**. Po uruchomieniu pierwszej wersji aplikacji ilość wymaganego czasu spadła do 1/2 etatu, **obecnie 1/4 etatu** (głównie wyrywkowa weryfikacja poprawności i konfrontacja z wyciągami z systemu F/K)

Założenia projektu: Automatyzacja przetwarzanych dokumentów:
- ujednolicone miejsce wymiany wiedzy,
- automatyczne pobieranie kluczowych danych z dokumentów płatniczych do zbiorczego raportu
- możliwość generowania raportów z edytowalnych dokumentów i skanów
- możliwość eksportu raportu do edytowalnego formatu tabelkowego (jak *.ods) i bazy danych (*.db)
- dowolna konfiguracja kontrahentów i zakresu czasowego raportu (ziarnistość jeden miesiąc)
- edytowalne ścieżki dostępu, zapisu i nazwy raportów

Założenia brzegowe: zadany draft formatu dokumentów wejściowych (edytowalnych i w formie zdjęcia / skanu)

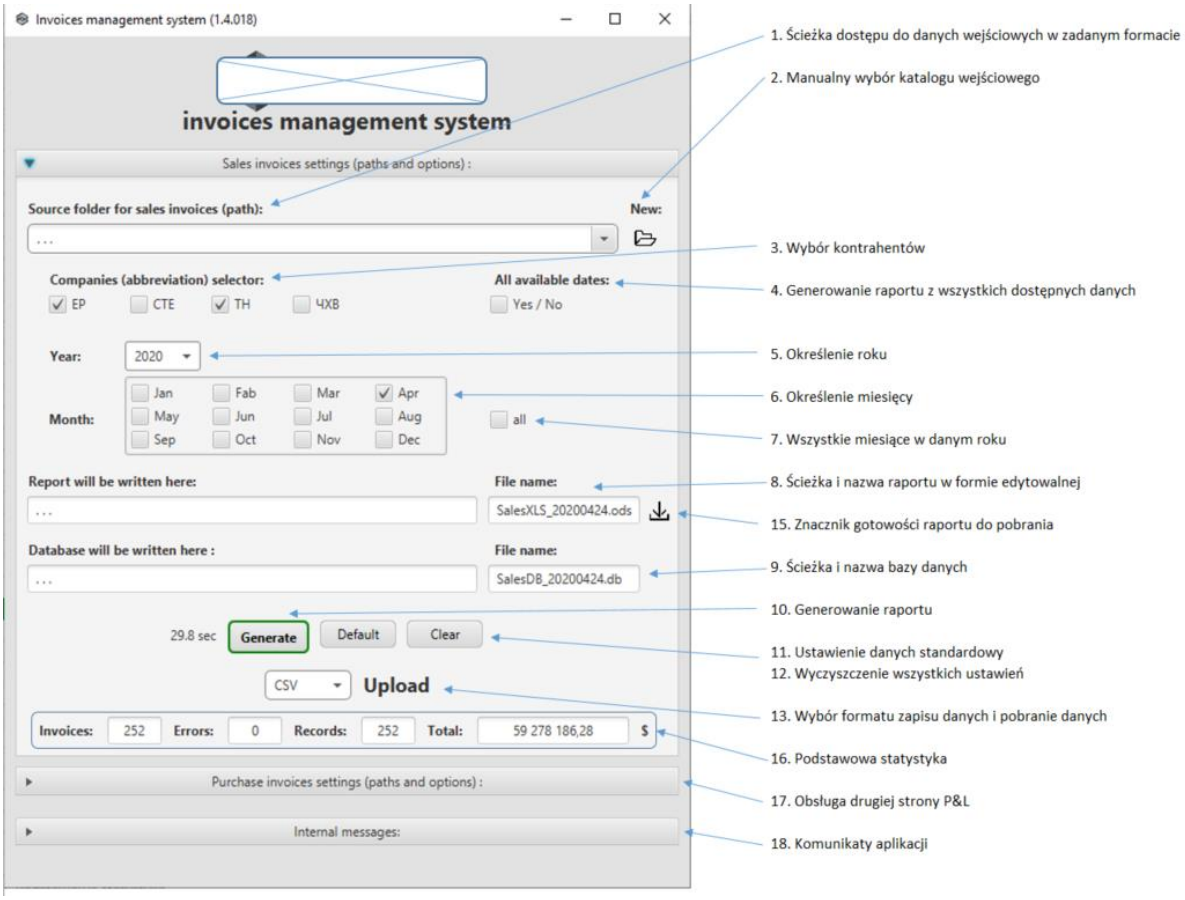
Czas wykonania projektu: 10/2019 – 01/2020 (pierwsza stabilna wersja)

II. Techniczny opis projektu

Pierwszy etap projektu: obsługa strumienia dokumentów edytowalnych

Technologie:
- silnik aplikacji (Java 12)
- struktura (Maven 3.8)
- graficzny interfejs użytkownika (JavaFX – openjfx 13)
- Apache Logger (obsługa raportów i informacji)
- Apache POI i PdfBox (obsługa edytowalnych dokumentów)
- środowisko (InteliJ IDEA)

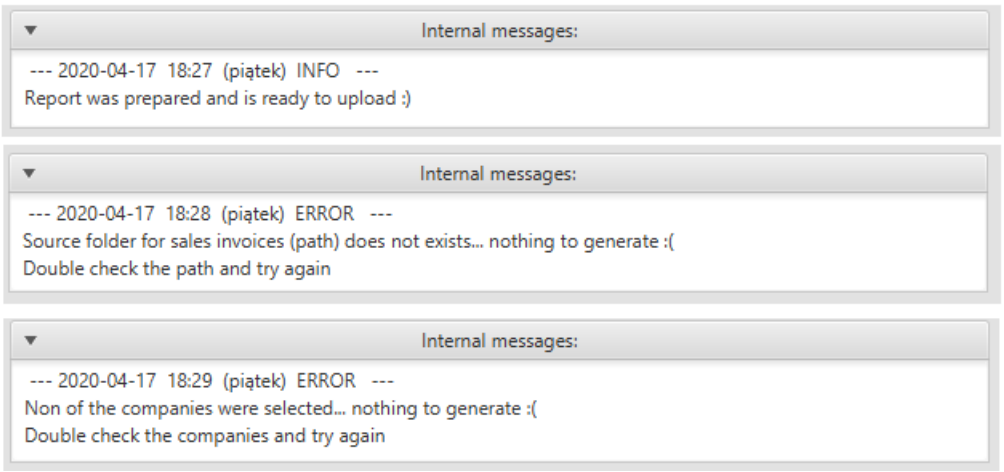
III. Graficzny interface:



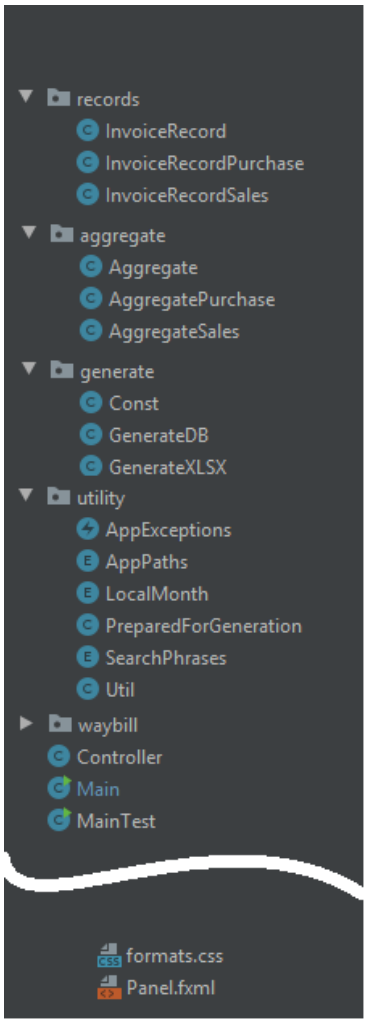
W pierwszej sekcji (od 1 do 9) użytkownik określa dane wejściowe i lokalizacje danych wyjściowych, w drugiej sekcji (od 10 do 14) użytkownik generuje dane i określa formaty danych i w trzeciej (16) otrzymuje podstawowe dane statystyczne.

Znacznik 15 pojawia się przy nazwach plików i określa gotowość otworzenia raportu w programie zewnętrznym, bezpośrednio z poziomu aplikacji.

Sekcja 17 dotyczy obsługi strony zakupowej, a w sekcji 18 dzieje się podstawowa komunikacja GUI do użytkownika:



IV. Struktura projektu



Projekt został zorganizowany w oparciu o kilka kluczowych obszarów:

A. Pakiet *records* zawiera klasę **InvoiceRecord** składającą się z 7 pól, które stanowią trzon każdego dokumentu płatniczego. Klasy InvoiceRecord**Sales** i InvoiceRecord**Purchase** dziedziczą po **InvoiceRecord** oraz wprowadzają zestaw pól charakterystycznych dla danego strumienia płatniczego.

B. Pakiet *aggregate* składa się z klas odpowiedzialnych za logikę zbierania danych z poszczególnych dokumentów płatniczych. Główna klasa **Aggregate** odpowiada za przygotowanie zestawu dokumentów (**List<File>**) do analizy dla danej specyfiki oraz core logiki zebrania danych do **LinkedHasMap <String, InvoiceRecord>**, które następnie są „uzupełniane" o specyfikę danego strumienia płatniczego w klasach dziedziczących: Aggregate**Sales** i Aggregate**Purchase**.

C. Pakiet *generate* odpowiada za przygotowanie raportu w zadanym formacie. GenerateXLSX odpowiada za logikę stworzenia sumarycznego raportu od odczytu w „edytorze tabelkowym" oraz klasa GenerateDB tworzy tabele i powiązania w relacyjnej bazie danych.

D. Pakiet *utility* posiada zestaw klas wspierających w procesie tworzenia, generowania i wydruku raportów.

E. Pakiet *waybill* to placecholder do kolejnej wersji obsługującej listy przewozowe (obecnie na etapie budowania strategicznych założeń).

F. Graficzny interfejs użytkownika jest oparty o FXML (zawartość) i CSS (wygląd).

V. Skład zespołu oraz role w projekcie:

- interesariusze po stronie firmy, wygenerowanie listy oczekiwań :
Prezes Zarządu, **Treasurer**, **Główny Księgowy** (outsourcing)

- opracowanie założeń, uzgodnienie struktury z interesariuszami, opracowanie dokumentacji, zaimplementowanie założeń, monitorowanie poprawności aplikacji, testy, bugfix, implementacja dodatkowych funkcjonalności, utrzymywanie aplikacji:
Rafał Sokołowski

VI. Link do repozytorium:

Kod zwiera dane stanowiące tajemnicę przedsiębiorstwa (m.in. dane kontrahentów, statystyki, thresholds, etc.) dlatego repozytorium jest w trybie zamkniętym, niemniej jest możliwość prezentacji kodu i funkcjonalności aplikacji podczas interview.