

Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki
Informatyka



Mateusz Stolecki

Konwerter danych pochodzących terminali kas fiskalnych.

projekt inżynierski

kierujący pracą: dr inż. Karolina Nurzyńska

Gliwice, 23 grudnia 2017

Spis treści

1	Wstęp	1
2	Analiza tematu	3
2.1	Wprowadzenie	3
2.1.1	Procesor danych z terminali kas fiskalnych Aloha	3
2.1.2	Umiejscowienie w systemie 360iQ	4
2.1.3	Dane wejściowe	4
2.1.4	Typy zdarzeń generowanych przez kasę Aloha	5
2.1.5	Etap wstępnego przetwarzania	7
2.1.6	Etap właściwego przetwarzania	7
2.1.7	Przypadki niestandardowe	8
2.2	Opis technologii i wykorzystanych narzędzi	10
2.2.1	Środowisko pracy i język programowania	10
2.2.2	Baza danych	10
2.2.3	Moduły wspierające	10
3	Wymagania	13
3.1	Wymagania funkcjonalne	13
3.1.1	Moduł wstępnego przetwarzania danych - PreParser	13
3.1.2	Moduł właściwego przetwarzania danych - Parser	13
3.2	Wymagania нефункционалне	14
3.2.1	Moduł wstępnego przetwarzania danych - PreParser	14
3.2.2	Moduł właściwego przetwarzania danych - Parser	14
3.3	Analiza przypadków użycia (diagramy UML)	15
4	Specyfikacja zewnętrzna	17
4.1	Wymagania sprzętowe i programowe	17
4.2	Instalacja	18
4.2.1	Instalacja PreParser-a na kontrolerze u klienta	18
4.2.2	Instalacja Parser-a po stronie administratora	20
4.3	Obsługa systemu i administrowanie nim	22
4.3.1	PreParser	22
4.3.2	Parser	23
4.4	Bezpieczeństwo	24
4.5	Przykład działania	24

5	Specyfikacja wewnętrzna	31
5.1	Architektura systemu	31
5.2	Organizacja i struktura baz danych	32
5.3	Wykorzystane biblioteki	32
5.4	Model klas	33
5.5	Szczegółowa analiza danych wejściowych	33
5.6	Algorytmy	34
5.6.1	Dodawanie brakujących przedmiotów będących częścią pro- mocji do transakcji	34
5.6.2	Rozszerzanie zdarzenia XML o rozszerzone nazwy przed- miotów i wartości podatków	35
6	Testowanie i uruchamianie	45
6.1	PreParser	45
6.2	Parser	45
7	Uwagi o przebiegu i wynikach prac	47
7.1	Stopień realizacji zagadnienia	47
7.2	Napotkane problemy	47
7.3	Dalszy rozwój	47
8	Podsumowanie	49

Rozdział 1

Wstęp

Celem pracy dyplomowej jest zaprojektowanie i implementacja systemu informatycznego, który umożliwiłby gromadzenie oraz przetwarzanie danych generowanych przez kasy sklepowe Aloha. System ten miałby na celu dostarczenie użytkownikowi informacji o funkcjonowaniu sklepu, w którym aplikacja została zainstalowana. Umożliwi to kompleksowy monitoring i analizę danych sprzedażowych. System będzie pobierał dane generowane przez kasę i poddawał je szczegółowej analizie, która pozwoli na tworzenie z nich struktur obrazujących rzeczywiste czynności wykonywane przy użyciu kas Aloha (szczególnie popularny model kasy w Stanach Zjednoczonych). Takie przetwarzanie przychodzących danych pozwoli użytkownikowi na wejście w dane szczegółowe konkretnej transakcji, zobaczenie materiału wideo z jej przebiegu oraz generowanie raportów i paragonów. Dzięki takim funkcjonalnościom system powinien umożliwić optymalizację sprzedaży oraz wprowadzanie oszczędności wynikających z usunięcia nieprawidłowości zaistniałych podczas pracy sklepu. Aplikacja będzie składała się z dwóch warstw:

- modułu wstępnego przetwarzania danych (PreParser-a),
- modułu właściwego przetwarzania danych (Parser-a).

PreParser będzie działał jako serwis po stronie użytkownika i wprowadzi konieczność jego instalacji na tzw. kontrolerze w sklepie użytkownika, czyli komputerze gromadzącym dane przychodzące z kas w danym sklepie i poddającym je etapowi wstępnej obróbki (preprasing) oraz wysyłającym dane wynikowe do bazy danych dostarczanej przez twórcę systemu, gdzie dalszym ich przetwarzaniem zajmie się Parser. Zadaniem modułu właściwego przetwarzania danych, będzie obróbka wstępnie przetworzonych informacji przez PreParser i zamiana ich na format mogący być analizowany przez system i zaprezentowany użytkownikowi. Pomysł na stworzenie takiego oprogramowania zrodził się z faktu zajmowania się tą tematyką zawodowo przez autora niniejszej pracy. Autor postanowił wykorzystać wiedzę i doświadczenie zdobyte podczas pracy zawodowej oraz studiów, by stworzyć system mający być częścią oprogramowania 360iQ dostarczanego przez firmę EZUniverse Inc. Dzięki regularnej pracy nad zagadnieniem podczas wykonywaniu obowiązków zawodowych, możliwe było dokładne dostosowanie systemu pod wymagania użytkownika oraz jego solidne przetestowanie przy działu z wykorzystaniem realnych danych i przypadków użycia.

Rozdział 2 niniejszej pracy zawiera zarys problemu oraz informacje opisujące wykorzystane technologie. Rozdział trzeci wprowadza w temat wymagań funkcjonalnych i niefunkcjonalnych oraz przypadków użycia. Następne rozdziały przybliżą tematykę specyfikacji zewnętrznej oraz wewnętrznej systemu. Poruszone zostaną zagadnienia związane z wykorzystanymi algorytmami oraz metodami radzenia sobie z problematycznymi danymi wejściowymi. Przedstawiony zostanie dodatkowo schemat bazy danych oraz nastąpi omówienie ważniejszych, ze względu na rolę i funkcjonalność klas. Dwa ostatnie rozdziały poruszają kwestię testowania aplikacji oraz omówiony zostanie przebieg prac i wyniki końcowe wraz z wizją dalszego rozwoju systemu.

Rozdział 2

Analiza tematu

Niniejszy rozdział omawia podstawowe zagadnienia związane z realizowanym projektem. Poruszona została kwestia umiejscowienia konwertera danych wygenerowanych przez kasy Aloha w całym systemie 360iQ oraz opisane zostały typy informacji przekazywane przez kasę.

2.1 Wprowadzenie

2.1.1 Procesor danych z terminali kas fiskalnych Aloha

Terminale kasowe Aloha (rys. 2.1) są niezwykle popularne w Stanach Zjednoczonych. Umożliwiają one kompleksową obsługę transakcji w sklepie, wpłat oraz wypłat do kasy. Jedną z dodatkowych możliwości terminala jest również dostarczanie informacji o zalogowaniu się pracownika i jego wylogowaniu, co pozwala ustalić intensywność pracy danej osoby, bądź też faktyczne godziny w jakich pracuje. W systemie 360iQ kasy Aloha wykorzystywane są do sklepach sieci Burger King. Terminal kasy przy każdym tzw. zdarzeniu, wysyła informację do nasłuchującego go kontrolera. Zdarzenia te są prostymi akcjami wykonywanymi podczas użytkowania kasy np. dodanie nowego produktu, potwierdzenie przyjęcia płatności. Dzięki gromadzeniu tych akcji po stronie kontrolera istnieje możliwość połączenia ich w zbiory, które będą reprezentować transakcje, wpłaty, wypłaty oraz obecności pracowników. Łączeniem tych zdarzeń w całość zajmuje się moduł aplikacji zainstalowany na kontrolerze zwany PreParserem, który jako część systemu 360iQ funkcjonuje pod nazwą AlohaPreParser. Przetwarza on zdarzenia uwzględniając zawarte w nich informacje dotyczące numeru kasy, rachunku etc. Po wstępnym przetworzeniu danych zostają one wysłane do bazy ulokowanej po stronie administratora, gdzie przechwytywane są one przez Parser i dokonywana jest właściwa analiza danych i ich przetwarzanie w celu dostarczenia klientowi potrzebnych informacji.



Rysunek 2.1: Przykładowy model kasy Aloha.

2.1.2 Umieszczenie w systemie 360iQ

AlohaPreParser oraz AlohaParser są bezpośrednimi składowymi systemu 360iQ dostarczanemu przez firmę EZUniverse. System ten składa się z całej infrastruktury nakierowanej na dostarczenie klientowi maksymalnej liczby informacji mogących wesprzeć funkcjonowanie biznesu klienta. Wspierane są duże ilości modeli kas, które mogą być zainstalowane w sklepie klienta. Dzięki wsparciu dla dużej ilości modeli kas oraz współpracy z największymi sieciami sklepów (np. Subway, Burger King) system 360iQ jest kompleksową platformą do wspomagania biznesu klienta.

2.1.3 Dane wejściowe

Danymi wejściowymi dostarczanych do kontrolera są pojedyncze zdarzenia zapisane w formacie XML. Format przychodzących danych omówiony zostanie na podstawie następującego zdarzenia (Rys. 2.2). Opisuje ono akcję dodania nowego przedmiotu do rachunku o numerze 3145860. Możemy na podstawie przedstawionego listingu ustalić również, że dane zdarzenie miało miejsce o godzinie 18:00:38 oraz przedmiot SM FRY został dodany do rachunku przez pracownika JAMES BOND. Na podstawie tak przedstawionych danych wejściowych moduł przetwarzania wstępnego jest w stanie budować całe zestawy transakcji, które zawierają szereg zdarzeń podobnych temu wyżej przedstawionemu. Dzięki tak pogrupowanym danym, zamiana zwykłych zdarzeń z terminala na struktury mogące być poddane analizie jest znacznie uproszczona. Terminal generuje również dodatkowe dane przechwytywane przez kontroler zwane słownikami. Słowniki są zestawami danych zapisanych w formacie XML niosącymi informacje o specyficznych cechach danego sklepu, np. promocjach, identyfikatorach produktów, bądź podatkach. Przykładowe słowniki przedstawiono na listingach w skróconej


```
<SpyMessage
  TerminalID="3"
  EventTime="18:00:38"
  EmployeeID="139"
  EmployeeName="JAMES BOND"
  ManagerID="0" ManagerName=""
  TableID="3132860"
  CheckID="3145860"
  TransactionTypeID="8"
  TransactionType="ADD_ITEM"
  Description="SM FRY"
  Amount="1.99"
  Quantity="1"
  Sender="192.168.0.101"
  ReceivedOn="2017/03/05 00:00:52.042"
/>
```

Rysunek 2.2: Przykładowe zdarzenie - dane wygenerowane przez kasę Aloha.

formie, gdyż całe pliki potrafią zawierać znaczne ilości informacji (: Rys. 2.3 - kategorie produktów, Rys. 2.4 - lista promocji:)

Słowniki te gromadzone są przez moduł wstępnego przetwarzania i wykorzystywane w analizie wspomnianych wcześniej danych wejściowych i algorytmach mających na celu dokonanie odpowiednich poprawek w danych transakcyjnych (szczegółowo ten temat został poruszony w rozdziale 5 traktującym o wykorzystanych algorytmach).

2.1.4 Typy zdarzeń generowanych przez kasę Aloha

Kasa Aloha jest zdolna do wygenerowania szeregu typów zdarzeń. Są to między innymi:

- Zalogowanie pracownika w kasie.
- Wylogowanie pracownika z kasy.
- Przerwa w pracy.
- Zakończenie przerwy w pracy.
- Otwarcie kasy.
- Dodanie nowego przedmiotu do transakcji.
- Anulowanie dodanego przedmiotu.
- Usunięcie przedmiotu z transakcji.
- Otwarcie nowego zamówienia.
- Drukowanie rachunku.
- Zamknięcie zamówienia.
- Ponowne otwarcie zamówienia.

```

1  <ItemCategoryMessage>
2    <Item ItemId="484" ItemName="WRP BLT CSP" Price="3.49">
3      <Category CategoryId="55" CategoryName="ALL ITEMS" />
4      <Category CategoryId="57" CategoryName="DISCOUNTABLE ITEMS
5        " />
6      <Category CategoryId="56" CategoryName="ALL ITEMS EXCEPT
7        CMB" />
8      <Category CategoryId="25" CategoryName="ALL SALADS" />
9      <Category CategoryId="1" CategoryName="FOOD" />
10   </Item>
11   <Item ItemId="5748" ItemName="AVOCADO RAN" Price="0">
12     <Category CategoryId="55" CategoryName="ALL ITEMS" />
13     <Category CategoryId="57" CategoryName="DISCOUNTABLE ITEMS
14       " />
15     <Category CategoryId="56" CategoryName="ALL ITEMS EXCEPT
16       CMB" />
17     <Category CategoryId="1" CategoryName="FOOD" />
18   </Item>
19 </ItemCategoryMessage>

```

Rysunek 2.3: Przykładowy słownik produktów w formacie XML - wersja skrócona.

- Przyjęcie płatności.
- Autoryzacja płatności.
- Odrzucenie płatności.
- Modyfikacja płatności.
- Usunięcie płatności.
- Dodaj promocje.
- Usuń promocje.
- Wartość podatku.
- Wartość netto transakcji.
- Wartość brutto transakcji.
- Dodatek do potrawy.

Jest to lista wszystkich obsługiwanych typów zdarzeń. Są one identyfikowane przez PreParser na podstawie przypisanych im unikalnych identyfikatorów. Istnieje jeden typ zdarzenia, który nie został uwzględniony w wyżej wymienionej liście, mianowicie zdarzenie nie będące elementem transakcji zakończonej sprzedażą. Takie przypadki to tzw. transakcje typu NO_SALE i są one analizowane z wykorzystaniem szczególnego podejścia. Jest ono szerzej opisane w rozdziale 5.

```
1  <PromotionMessage>
2    <Item ItemId="7592" ItemName="Items" Price="0">
3      <Promotion PromotionId="6485" PromotionVersion="1941"
4        PromotionName="6485*$5C'wich Ml 2" />
5      <Promotion PromotionId="118" PromotionVersion="999"
6        PromotionName="Local Combos" />
7      <Promotion PromotionId="139" PromotionVersion="2655"
8        PromotionName="Bacon King" />
9      <Promotion PromotionId="101" PromotionVersion="689"
10     PromotionName="Whopper Combo" />
11     <Promotion PromotionId="108" PromotionVersion="693"
12     PromotionName="Orig. Chicken Combo" />
13     <Promotion PromotionId="221" PromotionVersion="93"
14     PromotionName="Crois'wich Combo" />
15   </Item>
16 </PromotionMessage>
```

Rysunek 2.4: Przykładowy słownik listy promocji w formacie XML - wersja skrócona.

2.1.5 Etap wstępnego przetwarzania

Każdy sklep korzystający z oprogramowania będącego przedmiotem niniejszej pracy wymaga instalacji specjalnego modułu działającego jako serwis na komputerze zwanym kontrolerem. Urządzenie to gromadzi zdarzenia generowane przez kasy działające w obrębie omawianego sklepu. Odebrane informacje są przechwytywane przez działający w systemie moduł wstępnego przetwarzania, który poddaje je wstępnej analizie. Możemy wyodrębnić kluczowe etapy składające się na przeprowadzenie wspomnianego procesu:

- Sprawdzanie danych pod kątem duplikatów i ich odpowiednie odfiltrowanie.
- Analiza możliwości wystąpienia niezamkniętych lub błędnych transakcji i ich odpowiednia obsługa.
- Grupowanie zdarzeń przychodzących z kasy w kolekcje związane z jedną transakcją.
- Przeszukiwanie transakcji, które zostały pogrupowane i sprawdzanie czy wymagana jest korekcja cen produktów lub płatności.
- Przetwarzanie całych grup transakcji, rozbudowa zdarzeń z wykorzystaniem algorytmów analizy promocji oraz kategorii produktów.
- Konwersja zgrupowanych zdarzeń na obiekty XML (transakcje, wpłaty / wypłaty z kasy, obecności pracowników).
- Wysyłanie XMLi do bazy danych w celu ich przetworzenia przez moduł właściwego przetwarzania.

2.1.6 Etap właściwego przetwarzania

Po wstępnym przetworzeniu danych, gdy moduł wstępnego analizowania informacji przekaże je do bazy danych udostępnianej przez administratora sys-

temu, są one przechwytywane przez działający w środowisku dostarczyciela usługi moduł właściwego przetwarzania danych. Pobiera on z tabel dane w formacie XML, które dzięki wstępnej pracy wykonanej przez PreParser są już odpowiednio pogrupowane i oznaczone. Umożliwia to Parserowi wejście na poziom wyżej w analizie danych, podczas gdy PreParser analizował zdarzenia pod kątem grupowania ich w struktury, Parser może analizować same zdarzenia i wyciągać z nich informacje, które są konieczne do zbudowania wyczerpującego opisu transakcji, bądź innego zdarzenia, które miało miejsce na kasie. Moduł ten analizuje każde zdarzenie i krok po kroku tworzy obiekt, który na sam koniec przetwarzania danych zostanie umieszczony w wynikowej bazie. Udostępnia ona bezpośrednio dane mogące być analizowane przez klienta za pośrednictwem specjalnej aplikacji webowej będącej częścią systemu 360iQ. Schemat analizy danych przez moduł właściwego przetwarzania wygląda następująco:

- Sprawdzanie danych pod kątem duplikatów i ich odpowiednie odfiltrowanie.
- Wykrywanie typu zdarzenia jakie zostało zgrupowane w wejściowy XML: (transakcja, wpłata/wypłata, obecność pracownika).
- Uruchamianie odpowiedniego schematu przetwarzania zależnie od wykrytego typu wiadomości.
- Analiza danych w formacie XML krok po kroku i tworzenie cyfrowego modelu będącego odpowiednikiem realnej czynności wykonanej przez pracownika sklepu.
- Sprawdzanie poprawności stworzonych modeli i dokonywanie koniecznych poprawek, jak również oznaczenie tych, które zawierają wartości odbiegające od normy (ten fakt jest później odpowiednio prezentowany użytkownikowi w aplikacji webowej).
- Generowanie paragonów do wglądu przez klienta.
- Wysyłanie przetworzonych modeli do odpowiednich tabel wynikowej bazy danych w zależności od typu przetwarzanych wiadomości.

2.1.7 Przypadki niestandardowe

Terminale kasowe Aloha są dość bogate w przypadki, które można uznać za niestandardowe. Do najczęstszych takich sytuacji możemy zaliczyć:

- Wadliwe dane wejściowe,
- Niezamknięte transakcje,
- Dodawanie przedmiotów do rachunku z użyciem ich pełnej nazwy oraz późniejsze usuwanie z wykorzystaniem jej skróconej wersji,
- Dodawanie promocji bez uwzględnienia odpowiednich przedmiotów, które powinna dodana promocja dodawać,
- W regionach innych niż Stany Zjednoczone częstym problemem jest brak informacji o podatkach.

Wadliwe dane wejściowe

Z racji faktu, że moduł wstępnego przetwarzania otrzymuje dane z terminali kasowych Aloha w formacie XML, często zdarza się sytuacja, że jest on wadliwy (np. ucięty) lub też występują jego duplikaty. W takich przypadkach PreParser ustawia odpowiednie statusy tych wiadomości (mogą być one potem zweryfikowane przez użytkownika) i pomija je w procedurze wstępnej analizy.

Niezamknięte transakcje

Jednym z najrzadziej występujących przypadków niestandardowych jest sytuacja, gdy mamy do czynienia z niezamkniętą transakcją. Dzieje się to wtedy, gdy otwierana jest transakcja, lecz nie przychodzi zdarzenie oznaczające jej zamknięcie (w Aloha jest to: CLOSE_CHECK). W takiej sytuacji moduł wstępnego przetwarzania ustawia taką transakcję w stan oczekiwania, który trwa 24 godziny. Jeśli do tego czasu przyjdzie zdarzenie zamykające daną transakcję, to zostanie ona zatwierdzona i przesłana do dalszej analizy, w przeciwnym razie nastąpi jej usunięcie z kolejki oczekujących i oznaczenie jako błędnej danej wejściowej, by użytkownik mógł samodzielnie się jej przyjrzeć.

Dodawanie przedmiotów do rachunku z użyciem ich pełnej nazwy oraz późniejsze usuwanie, przy wykorzystaniu ich skróconej nazwy

Przedmioty dodawane przez kasę Aloha do rachunków niestety często charakteryzują się niejednolitym nazewnictwem. Ten sam przedmiot może być nazywany z wykorzystaniem:

- nazwy skróconej,
- pełnej nazwy przedmiotu,
- ID przedmiotu, bądź jego oznaczenia kodowego.

Powoduje to problemy przy usuwaniu z rachunku już dodanych przedmiotów. Przedmiot dodany do transakcji pod nazwą X może być z niej usuwany pod nazwą Y. W celu uzyskania spójności i powiązania ze sobą przedmiotów wykorzystywany jest algorytm rozszerzania danych przekazywanych przez zdarzenie. Korzysta on ze słownika kategorii przedmiotów, który jest dostarczany raz dziennie przez właściciela sklepu jako specjalne zdarzenie terminala Aloha i przechowywany przez kontroler. Dzięki temu słownikowi możliwe jest powiązanie ze sobą tych samych przedmiotów występujących pod różną nazwą, wykorzystując fakt, że każdy przedmiot ma swoje unikalne ID, które jest udostępniane za pośrednictwem słownika kategorii produktów. Takie rozwiązanie gwarantuje, że w wynikowej transakcji przedmioty będą powiązane i usunięte w sposób prawidłowy niezależnie od zastosowania pełnej, bądź jego skróconej nazwy.

Dodawanie promocji bez uwzględnienia odpowiednich przedmiotów, które określona promocja powinna dodawać

Niezbyt przyjazną cechą terminala Aloha jest obsługa dodawania promocji bez uwzględnienia przedmiotów wchodzących w jej skład. W takim wypadku

konieczne jest dodanie przedmiotów do transakcji, których dotyczy dana promocja, dzięki czemu zachowana zostanie zgodność danych z realnie sprzedanymi przedmiotami po stronie sklepu. Dodawanie przedmiotów przez moduł wstępnego przetwarzania realizowane jest przy użyciu słownika promocji dostarczanego przez klienta dla każdego sklepu. Pozwala on zidentyfikować nazwę promocji i rozszerzyć XML z transakcją o potrzebne przedmioty.

W regionach innych niż Stany Zjednoczone częstym problemem jest brak informacji o podatkach

Problem ten objawia się sytuacją, gdy w zdarzeniu wystawianym przez terminal mającym typ "TAX" pojawia się zerowa wartość podatku. W tej sytuacji, by uzyskać prawidłową cenę końcową z wliczoną już wartością podatku, konieczne jest wczytanie dodatkowego słownika przechowującego wartości podatków dla każdego z przedmiotów. Takie rozwiązanie pojawia się w sytuacji, gdy oprogramowanie działa w krajach gdzie system podatkowy znacząco różni się od tego, który obowiązuje w Stanach Zjednoczonych.

2.2 Opis technologii i wykorzystanych narzędzi

2.2.1 Środowisko pracy i język programowania

Projekt został zrealizowany przy użyciu technologii .NET oraz języka C#. Wielokrotnie podczas przeglądania kodu źródłowego projektu można również napotkać skrypty pisane w języku T-SQL, w celu komunikacji z bazą danych. Całość projektu została napisana w środowisku Visual Studio 2017, przy czym korzystano również z Visual Studio Code w celu analizy danych XML oraz szybkiego przeglądania plików tekstowych. Do administrowania i zarządzania bazą danych wykorzystano Microsoft SQL Server Management Studio 2016.

2.2.2 Baza danych

Serwerem bazy danych był Microsoft SQL Server 2016 w wersji Express zainstalowany na lokalnym komputerze. Dokonano połączenia, na lokalnej bazie developera, tabel koniecznych do pracy modułu wstępnego przetwarzania i tych niezbędnych do funkcjonowania jednostki realizującej właściwe procesowanie danych. Dzięki takiemu rozwiązaniu można było testować działanie całego systemu na lokalnym komputerze bez konieczności korzystania z osobnego komputera, który miałby pełnić funkcję kontrolera. Dane wykorzystywane w niniejszym projekcie są kopią zapasową rzeczywistych danych źródłowych zaczerpniętych z codziennej pracy systemu u klienta. Pozwala to testować i przedstawić pracę niniejszego projektu w jego naturalnych warunkach i w starciu z prawdziwymi danymi generowanymi przez sklepy w Stanach Zjednoczonych.

2.2.3 Moduły wspierające

Opisywana w niniejszej pracy aplikacja jest wspierana przez następujące oprogramowanie, które jest częścią systemu 360iQ i zostało opracowane przez firmę EZUniverse:

- AlohaSpyRelay,
- EZ360DataInterface.

AlohaSpyRelay

AlohaSpyRelay jest aplikacją zapewniającą transmisję danych z kasy Aloha do kontrolera gdzie zainstalowany jest moduł wstępnego przetwarzania. Pobiera ona dane bezpośrednio z terminala, zamienia ją na zdarzenia zapisane w formacie XML i przekazuje do kontrolera.

EZ360DataInterface

EZ360DataInterface jest łącznikiem pomiędzy modułem wstępnego przetwarzania i częścią odpowiadającą za właściwe przetwarzanie. Po wstępnym przetworzeniu danych przez PreParser istnieje konieczność wysłania ich do bazy danych dostarczanej przez administratora systemu, tym właśnie zajmuje się opisywana aplikacja. Moduł wstępnego przetwarzania przygotowuje gotową paczkę danych do wysłania i powiadamia EZ360DataInterface, że są dane do wysyłki. Aplikacja przechwytytuje dane od PreParsera i wysyła je zgodnie ze skonfigurowanymi w jej ustawieniach wytycznymi.

Rozdział 3

Wymagania

Rozdział ten opisuje wymagania konieczne do uruchomienia oprogramowania po stronie kontrolera jak i administratora aplikacji. Przedstawione zostały również przypadki użycia oprogramowania przez użytkownika wraz ze stosownym komentarzem.

3.1 Wymagania funkcjonalne

3.1.1 Moduł wstępnego przetwarzania danych - PreParser

Wymagania funkcjonalne odnośnie modułu PreParsera składają się z następujących założeń:

- Serwis powinien być w stanie odczytywać pobrane dane z lokalnej bazy danych kontrolera, gdzie przechowywane są informacje o zdarzeniach, które nadesłał terminal Aloha,
- Powinna istnieć możliwość przetwarzania danych na etapie wstępnej analizy w celu eliminacji duplikatów i wadliwych wiadomości.
- Kluczową kwestią jest zastosowanie algorytmów dodających brakujące produkty z promocji oraz rozszerzające nazwy wspomnianych wcześniej produktów, by można było je łatwo ze sobą powiązać.
- Serwis powinien informować użytkownika na bieżąco o stanie swojej aktywności za pośrednictwem komunikatów w pliku logów.
- Serwis powinien być zdolny grupować wiadomości w kolekcje i konwertować je na odpowiednio zdefiniowane XMLe, a następnie wysyłać je do bazy danych po stronie administratora, by mogły być one dalej analizowane.

3.1.2 Moduł właściwego przetwarzania danych - Parser

Podczas projektowania modułu Parsera powinny zostać uwzględnione następujące wymagania funkcjonalne:

- Aplikacja powinna móc odczytywać dane z odpowiednich tabel w bazie danych, gdzie w sposób ciągły pojawiają się nowe dane przysyłane z bliżej

nie określonej liczby kontrolerów, gdzie zostały one wstępnie obrobione przez moduł wstępnego przetwarzania.

- Oprogramowanie musi wspierać różne typy przychodzących wiadomości (transakcje, wpłaty/wypłaty, obecności pracowników) i stosować dla nich właściwe formy analizy i przetwarzania.
- Wynikowe dane powinny zostać zapisane w bazie danych, z której może korzystać aplikacja webowa dostarczana jako element systemu 360iQ i opracowana przez firmę EZUniverse i prezentująca wyniki użytkownikowi w przyjaznej dla niego formie. Rezultat przetwarzania powinien być możliwie jak najdokładniejszym odwzorowaniem zdarzenia jakie zaszło w danej chwili czasu na kasie w sklepie użytkownika. Ważne jest zachowanie odpowiednich stref czasowych i synchronizacja czasu wideo,
- W przypadku wystąpienia błędu podczas analizy danych, aplikacja powinna zamieścić odpowiednią informację na ekranie interfejsu użytkownika.

3.2 Wymagania niefunkcjonalne

3.2.1 Moduł wstępnego przetwarzania danych - PreParser

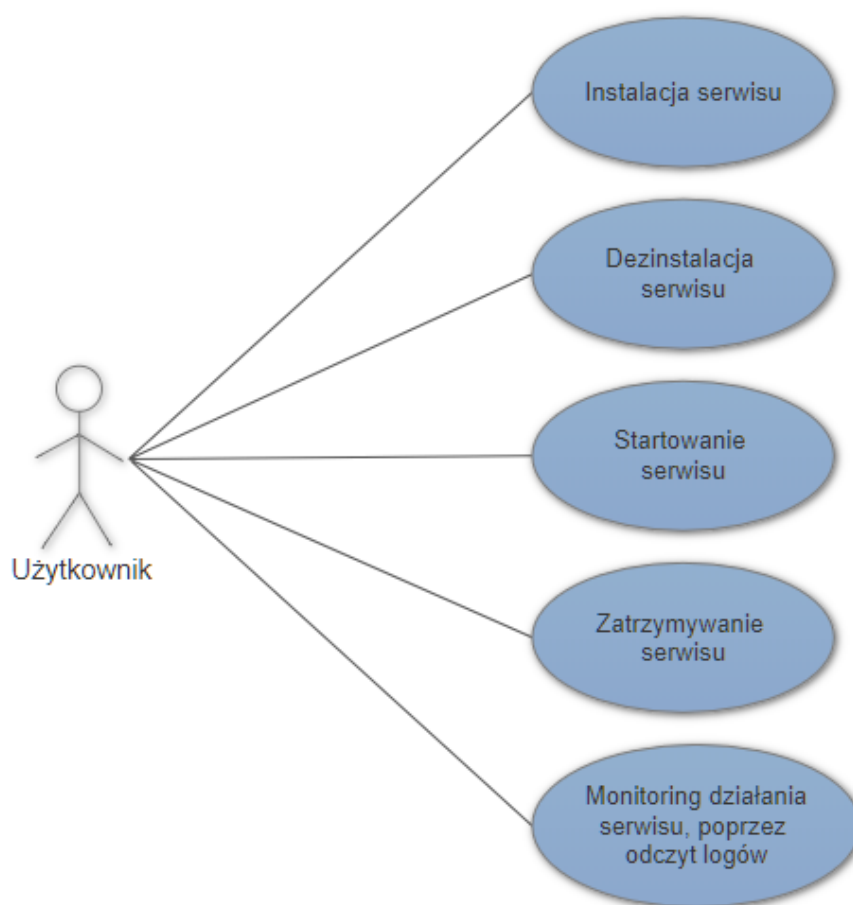
Wymagania niefunkcjonalne odnośnie modułu PreParsera składają się z następujących założeń:

- Stabilnie działający serwis po stronie kontrolera.
- Personalizacja za pomocą jednego pliku konfiguracyjnego.
- Prosty sposób instalacji oraz włączania serwisu.
- Szczegółowy system logów pomagający w diagnostyce usterek i monitoringu działania aplikacji.
- Niskie wymagania sprzętowe, serwis powinien się uruchamiać i pracować bez widocznych strat na wydajności będąc zainstalowanym na stosunkowo taniej i mało wydajnej maszynie.

3.2.2 Moduł właściwego przetwarzania danych - Parser

Podczas projektowania modułu Parsera powinny zostać uwzględnione następujące wymagania niefunkcjonalne:

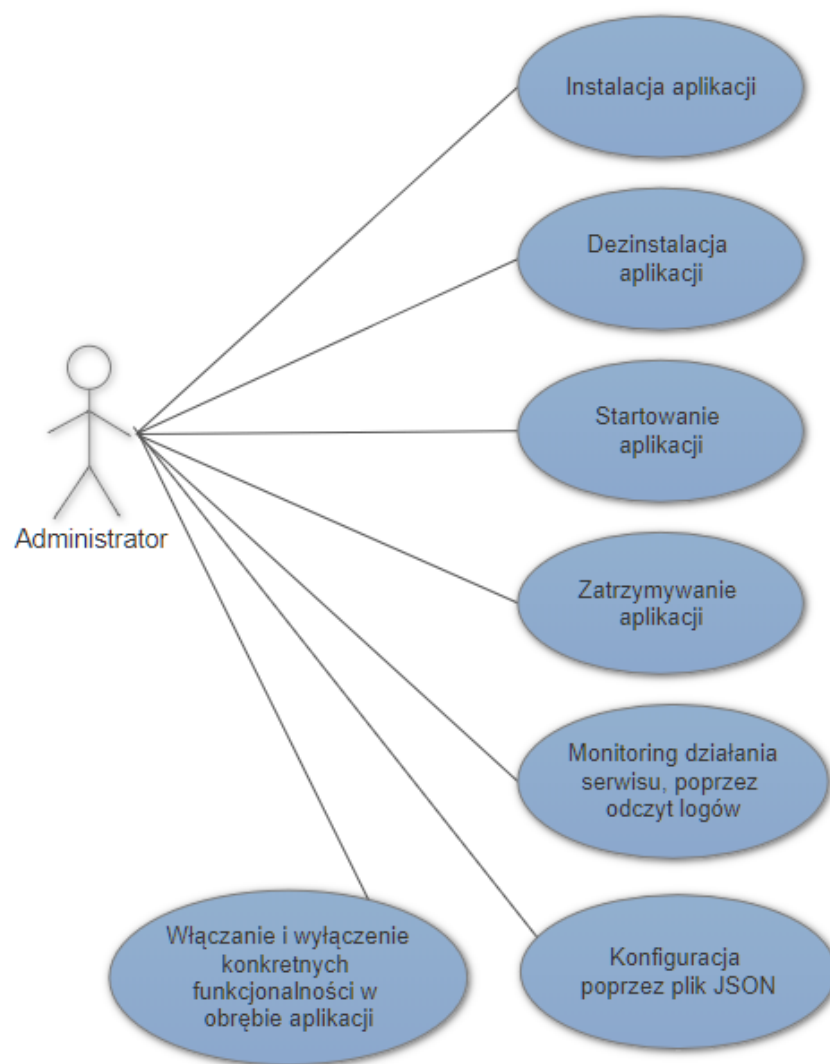
- Stabilnie działająca aplikacja desktopowa z intuicyjnym interfejsem użytkownika.
- Pełne wsparcie dla przetwarzania danych z kas Aloha oraz integracja z pozostałymi parserami z innych kas działającymi w systemie.
- Prosty i intuicyjny interfejs użytkownika.
- Łatwość wystartowania i zatrzymania aplikacji.
- Dążenie do jak najniższego wykorzystania zasobów systemu, w którym aplikacja jest zainstalowana.



Rysunek 3.1: Przypadki użycia - moduł wstępnego przetwarzania.

3.3 Analiza przypadków użycia (diagramy UML)

W niniejszej sekcji zaprezentowano przypadki użycia dla modułu wstępnego przetwarzania (rys. 3.1). Schemat dla modułu PreParsera obrazuje czynności możliwe do wykonania przez użytkownika aplikacji w celu sterowania jej pracą. Natomiast diagram dla części odpowiadającej za właściwe przetwarzanie (rys. 3.2) pokazuje akcje i czynności, jakie może podjąć administrator, by kontrolować pracę Parsera.



Rysunek 3.2: Przypadki użycia - moduł właściwego przetwarzania.

Rozdział 4

Specyfikacja zewnętrzna

Rozdział ten opisuje wymagania sprzętowe i programowe, które należy spełnić, by zapewnić poprawne funkcjonowanie aplikacji. Poruszone zostały kwestie instalacji i administracji poszczególnymi modułami oprogramowania oraz temat zabezpieczeń.

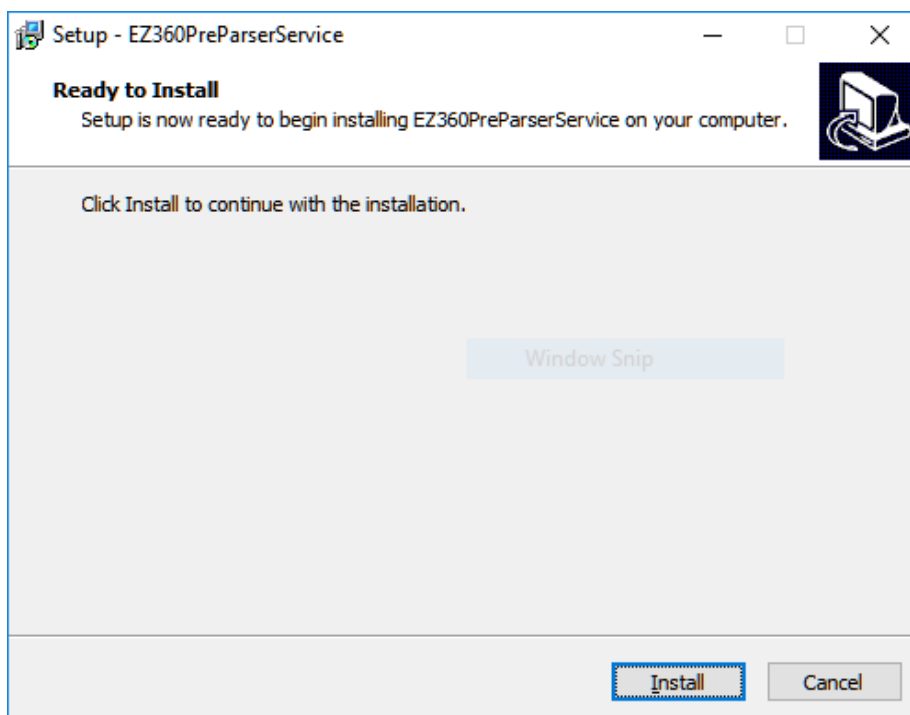
4.1 Wymagania sprzętowe i programowe

Opisywana aplikacja została przetestowana, na dwóch konfiguracjach sprzętowych:

- Komputer stacjonarny:
 - Procesor - i5 4690K @ 4.5 GHz, 4.4GHz Uncore,
 - Pamięć RAM - 8GB DDR3 @ 2133MHz CL9.11.12 T1,
 - Dysk - SSD 128 GB TLC,
 - System operacyjny - Windows 10 PRO.
- Laptop:
 - Procesor - i5 7200U @ 2.5 GHz,
 - Pamięć RAM - 16GB DDR4 @ 2133MHz CL15,
 - Dysk - SSD 256 GB MLC,
 - System operacyjny - Windows 10 PRO.

Obydwie konfiguracje gwarantowały satysfakcjonującą płynność i szybkość działania aplikacji. Na podstawie testów z użyciem wyżej wymienionych zestawów zdefiniowano przybliżone wymagania sprzętowe oraz programowe:

- Minimalne wymagania sprzętowe:
 - Procesor - i3 2 generacji Intel Core lub odpowiednik AMD,
 - Pamięć RAM - 4GB pamięci operacyjnej,
 - Dysk - 100MB wolnej przestrzeni dyskowej,
 - Stały dostęp do internetu.



Rysunek 4.1: Instalacja modułu wstępnego przetwarzania - początek instalacji.

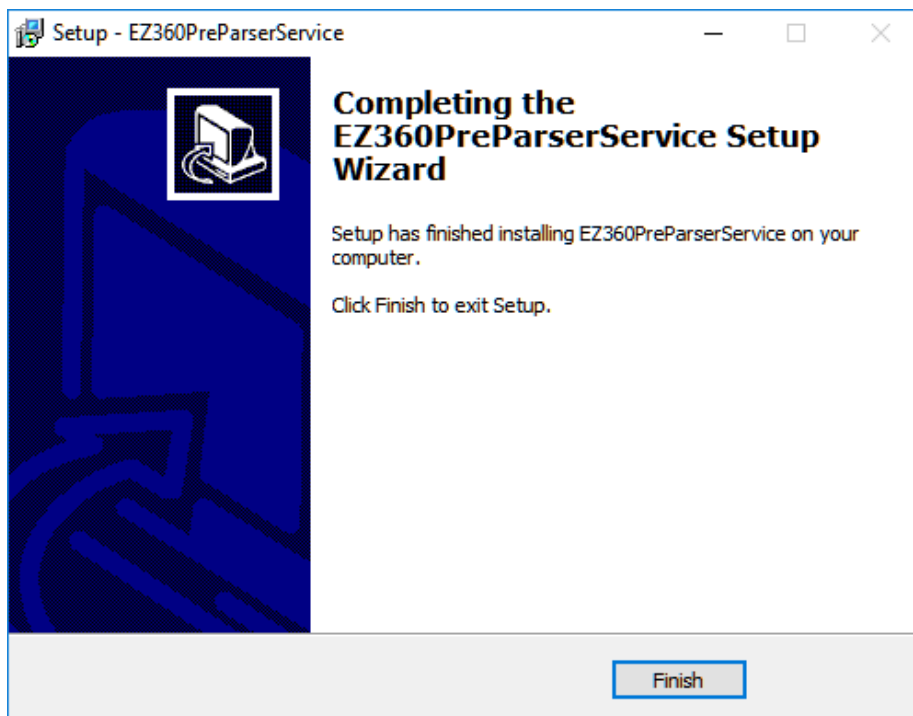
- Minimalne wymagania programowe:
 - System operacyjny - Windows w wersji 7 lub nowszej,
 - Microsoft SQL Server Management Studio oraz MS SQL Server w wersji 2012 lub nowszej,
 - .NET Framework w wersji 3.5 lub nowszej.

4.2 Instalacja

4.2.1 Instalacja PreParser-a na kontrolerze u klienta

W celu zainstalowania modułu wstępnego przetwarzania danych należy uruchomić plik `EZ360PreParserService.Setup.v.VERSION.exe`, gdzie `VERSION` jest numerem wersji. Uruchomiony zostanie instalator serwisu (rys. 4.1). W momencie wciśnięcia przycisku `Install`, instalator rozpocznie wgrywanie `PreParser` na kontroler. Po zakończeniu procesu powinien zostać wyświetlony ekran informujący o poprawnie zakończonej instalacji serwisu (rys. 4.2). Następnie należy dokonać konfiguracji zainstalowanego serwisu. W tym celu otwieramy plik `configuration.json`. Powinien on mieć format przedstawiony na rys. 4.3. Skrócony opis poszczególnych parametrów:

- `ProcessID` - id procesu działającego na danym kontrolerze.
- `ParserType` - typ parsera komunikującego się z kontrolerem.



Rysunek 4.2: Instalacja modułu wstępnego przetwarzania - koniec instalacji.

- ObjectsConnectionString - parametry połączenia do tabeli EZ360Objects w bazie danych.
- CommunicationConnectionString - parametry połączenia do tabeli EZ360Communication w bazie danych.
- TransactionWebservice - adres url do webservice-u obsługującego transakcje.
- ActionWebservice - adres url do webservice-u obsługującego akcje.
- CashOperationWebservice - adres url do webservice-u obsługującego wpłaty/wypłaty.
- TimeClockWebservice - adres url do webservice-u obsługującego obecności pracowników.
- DataWebservice - adres url do webservice-u obsługującego dane dodatkowe.
- TimerInterval - czas pomiędzy kolejnymi sprawdzeniami czy nie przyszły nowe zdarzenia (w milisekundach).
- DatabaseTimeout - maksymalny czas w jakim może zostać wykonywane zapytanie do bazy danych (w sekundach).
- DatabaseTimeoutInterval - przerwa pomiędzy kolejnymi zapytaniem do bazy danych (w sekundach).
- DelayAfterDatabaseError - czas do ponownej próby połączenia z bazą danych po wystąpieniu błędu.

```
1  {
2    "ProcessID": 636298774975,
3    "ParserType": "ALOHA",
4    "ObjectsConnectionString": "Data Source = source; User ID = sa
5    ; Password = sa; Initial Catalog = EZ360Objects;",
6    "CommunicationConnectionString": "Data Source = source; User
7    ID = sa; Password = sa; Initial Catalog =
8    EZ360Communication;",
9    "TransactionWebservice": "http://localhost:65059/api/
10   Transactions",
11    "ActionWebservice": "http://localhost:65059/api/Actions",
12    "CashOperationWebservice": "http://localhost:65059/api/
13   CashOperations",
14    "TimeClockWebservice": "http://localhost:65059/api/TimeClocks"
15   ,
16    "DataWebservice": "http://localhost:65059/api/Data",
17    "TimerInterval": 500,
18    "DatabaseTimeout": 30,
19    "DatabaseTimeoutInterval": 10,
20    "DelayAfterDatabaseError": 10,
21    "DelayDeleteNotServedTransaction": 40,
22    "Rows": 500,
23    "SenderTimerInterval": 1000,
24    "MaxRowsToSend": 25,
25    "MaxSenderQueues": 250,
26    "CompressWebRequest": true,
27    "ControllerWebURL": "http://127.0.0.1:81/"
28  }
```

Rysunek 4.3: Zawartość pliku konfiguracyjnego - PreParser.

- DelayDeleteNotServedTransaction - czas po jakim nieobsługiwana transakcja jest usuwana.
- Rows - maksymalna liczba wierszy możliwa do procesowania.
- SenderTimerInterval - interwał z jakim wysyłane są dane do bazy.
- MaxRowsToSend - maksymalna liczba wierszy przesłana w jednym zapytaniu.
- MaxSenderQueues - limit liczby obiektów w kolejce do wysłania.
- CompressWebRequest - kompresowanie wysyłanych danych.
- ControllerWebURL - adres URL kontrolera.

4.2.2 Instalacja Parser-a po stronie administratora

Poprawna instalacja modułu właściwego przetwarzania danych na komputerze administratora wymaga ręcznego rozpakowania archiwum AlohaParser_VERSION.zip, gdzie VERSION jest numerem wersji. Po rozpakowaniu archiwum w miejscu wybranym przez administratora, powinny być dostępne pliki potrzebne do uruchomienia modułu właściwego przetwarzania danych. Aplikację można uruchomić otwierając plik EZ360ParserUI.exe. Następnie należy dokonać konfiguracji zainstalowanej aplikacji. W tym celu otwieramy plik configuration.json. Powinien on mieć format przedstawiony na listingu 4.4. Skrócony opis poszczególnych parametrów:


```

1  "ProcessID": 444,
2  "ParserType": "ALOHA",
3  "Enabled": false,
4  "DBConnections": {
5      "AccessConnectionString": "Data Source = source; User ID =
        sa; Password = sa; Initial Catalog = EZ360Access",
6      "CommunicationConnectionString": "Data Source = source; User
        ID = sa; Password = sa; Initial Catalog =
        EZ360Communication",
7      "LaborConnectionString": "Data Source = source; User ID = sa
        ; Password = sa; Initial Catalog = EZ360Labor",
8      "ObjectsConnectionString": "Data Source = source; User ID =
        sa; Password = sa; Initial Catalog = EZ360Objects",
9      "SalesConnectionString": "Data Source = source; User ID = sa
        ; Password = sa; Initial Catalog = EZ360Sales",
10     "ReportsConnectionString": "Data Source = source; User ID =
        sa; Password = sa; Initial Catalog = EZ360Reports",
11     "ApplicationsConnectionString": "Data Source = source; User
        ID = sa; Password = sa; Initial Catalog =
        EZ360Applications"
12 },
13 "TimerInterval": 1000,
14 "DatabaseTimeout": 10,
15 "DatabaseTimeoutInterval": 10,
16 "DelayAfterDatabaseError": 5000,
17 "EnablePreserveSavedTransactions": false,
18 "DeletePreservedTransactionsAfter": 24,
19 "VideoTimeOffset": -10,
20 "IdentificationPrefix" : "",
21 "IdentificationNumber" : "",
22 "Actions": {
23     "Enabled": false,
24     "Rows": 100
25 },
26 "CashOperations": { "Enabled": true, "Rows": 100 },
27 "Transactions": { "Enabled": true, "Rows": 100 },
28 "TimeClocks": { "Enabled": true, "Rows": 100 },
29 "Data": { "Enabled": true, "Rows": 100 }

```

Rysunek 4.4: Zawartość pliku konfiguracyjnego - Parser - wycinek dotyczący kasy Aloha.

- ProcessID - id procesu działającego na komputerze administratora.
- ParserType - typ parsera, którego dotyczy konfiguracja.
- AccessConnectionString - parametry połączenia do tabeli EZ360Access w bazie danych.
- CommunicationConnectionString - parametry połączenia do tabeli EZ360Communication w bazie danych.
- LaborConnectionString - parametry połączenia do tabeli EZ360Labor w bazie danych.
- ObjectsConnectionString - parametry połączenia do tabeli EZ360Objects w bazie danych.
- SalesConnectionString - parametry połączenia do tabeli EZ360Sales w bazie danych.

- ReportsConnectionString - parametry połączenia do tabeli EZ360Reports w bazie danych.
- ApplicationsConnectionString - parametry połączenia do tabeli EZ360Applications w bazie danych.
- TimerInterval - czas pomiędzy kolejnymi sprawdzeniami czy nie przyszedły nowe dane (w milisekundach).
- DatabaseTimeout - maksymalny czas w jakim może zostać wykonywane zapytanie do bazy danych (w sekundach).
- DatabaseTimeoutInterval - przerwa pomiędzy kolejnymi zapytaniami do bazy danych (w sekundach).
- DelayAfterDatabaseError - czas do ponownej próby połączenia z bazą danych po wystąpieniu błędu.
- EnablePreserveSaveTransactions - definiuje czy włączyć funkcjonalność zachowywania transakcji.
- DeletePreserveTransactionsAfter - definiuje po jakim czasie powinny być usuwane zachowane transakcje (podane w godzinach).
- Rows - maksymalna liczba wierszy możliwa do procesowania.
- VideoTimeOffset - przesunięcie w minutach uwzględniane przy wyliczaniu czasu wideo.
- IdentificationPrefix - prefix pomagający zidentyfikować dane.
- IdentificationNumer - numer pomagający zidentyfikować dane.
- Dalsze ustawienia definiują, które typy danych powinny być przetwarzane.

4.3 Obsługa systemu i administrowanie nim

4.3.1 PreParser

Startowanie serwisu

Startowanie serwisu realizowane jest z wykorzystaniem systemu operacyjnego Windows. Poprawne wystartowanie modułu wstępnego przetwarzania (po jego odpowiedniej konfiguracji) sprowadza się do uruchomienia serwisu za pośrednictwem menadżera serwisów systemu Windows (proces został zaprezentowany na rys. 4.5)

Zatrzymywanie serwisu

Zatrzymywanie preparsera realizowane jest podobnie jak jego startowanie. Wykorzystuje się do tego również menadżer serwisów Windows.

Dezinstalacja serwisu

W celu usunięcia serwisu z systemu operacyjnego należy uruchomić wiersz poleceń systemu Windows, a następnie przejść do katalogu gdzie został zainstalowany serwis. Po dokonaniu wyżej opisanych czynności należy dokonać dezinstalacji poleceniem:

`EZ360PreParserService.exe -uninstall`

Spowoduje one usunięcie serwisu z systemu.

Diagnostyka i logowanie

Serwis podczas pracy regularnie generuje szereg komunikatów o statusie swojego działania i aktywności. Są one zapisywane w plikach logów, które przechowywane są w folderze logs. Przechowywanych jest do 30 plików logów, które następnie po pojawieniu się następnych logów są usuwane. Wyróżniamy następujące typy wiadomości składowanych w logach:

- INFO - informuję użytkownika o statusie działa serwisu.
- WARN - komunikuje pojawienie się problemu podczas analizy danych, co w większości przypadków oznacza brak wsparcia dla konkretnego typu danych i nie powoduje błędnego działania systemu.
- ERROR - zawiadamia o wystąpieniu poważnego problemu, który powinien zostać zgłoszony do administratora w celu jego rozwiązania.

4.3.2 Parser

Uruchamianie aplikacji

W celu uruchomienia aplikacji należy otworzyć plik `EZ360ParserUI.exe`, następnie dokonać inicjalizacji poprzez kliknięcie odpowiedniego przycisku (rys. 4.7). Po poprawnej inicjalizacji modułu, wystarczy wcisnąć przycisk start, by rozpocząć proces przetwarzania danych (rys. 4.8).

Zatrzymywanie aplikacji

Administrator może w każdej chwili zatrzymać moduł właściwego przetwarzania danych. W celu dokonania tego, należy kliknąć przycisk STOP (rys. 4.9). Aplikacja dokończy obecny proces przetwarzania i wyśle dane, następnie zatrzyma się (trwa to około kilka sekund, zależnie od ilości przetwarzanych danych).

Dezinstalacja aplikacji

Dezinstalacja aplikacji sprowadza się do usunięcia zawartości folderu, w którym został zainstalowany moduł. Należy przy tym pamiętać, by wcześniej zatrzymać Parser i zamknąć aplikację.

Diagnostyka i logowanie

Podobnie jak w przypadku modułu wstępnego przetwarzania, tutaj również zaimplementowano system logowania komunikatów. Komunikaty informujące o stanie działania aplikacji przechowywane są w folderze logs oraz wyświetlane w interfejsie użytkownika, który udostępnia oprogramowanie. W tym wypadku typy komunikatów wyświetlane użytkownikowi są określone przez odpowiedni kolor czcionki:

- INFO - czarny,

- WARN - zielony,
- ERROR -czerwony.

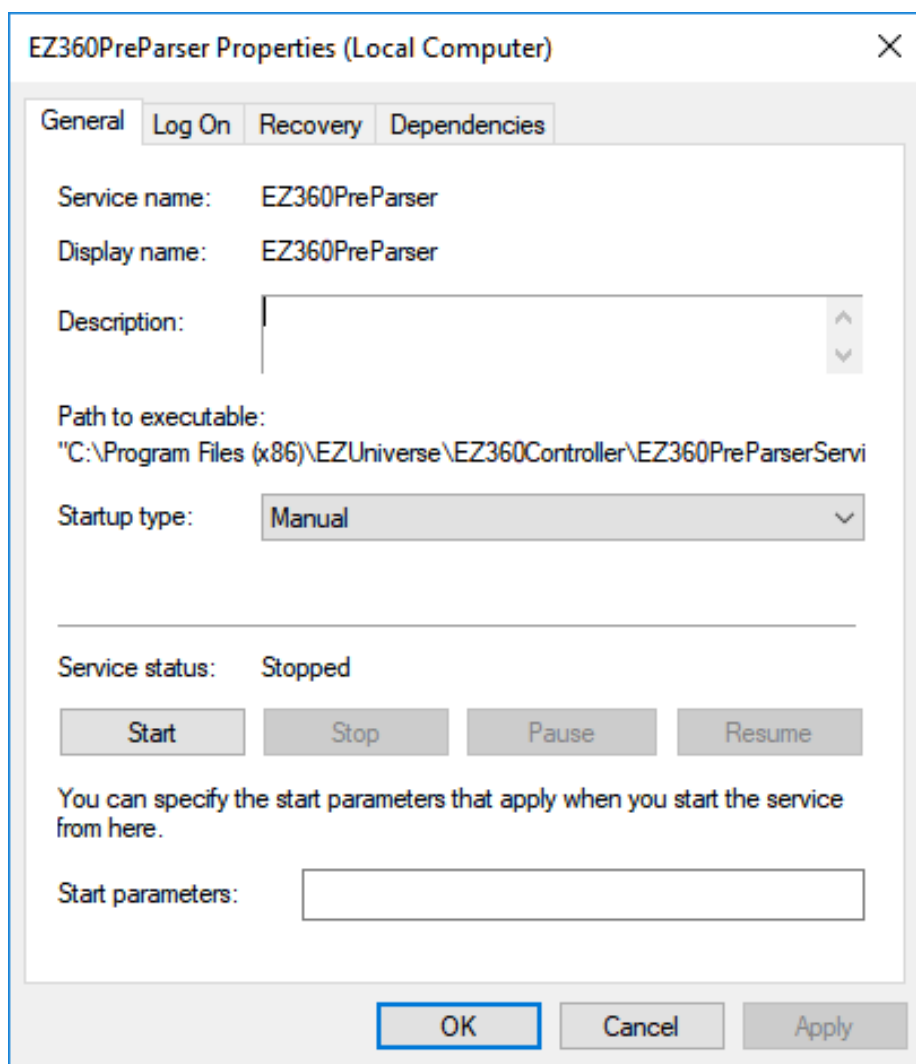
4.4 Bezpieczeństwo

Ważną kwestią uwzględnianą podczas procesu tworzenia oprogramowania będącego przedmiotem niniejszej pracy jest bezpieczeństwo. W celu zapewnienia uwarunkowań sprzyjających bezpieczeństwu przetwarzanych danych moduł właściwego przetwarzania oraz wszystkie bazy przechowujące informacje przychodzące od strony użytkownika składowane są po stronie administratora systemu. Zapewnia on szereg zabezpieczeń i ciągłe wsparcie techniczne. Po stronie użytkownika zainstalowany jest tylko serwis modułu wstępnego przetwarzania na kontrolerze, który tylko przetwarza lokalne dane i wysyła je do głównego centrum danych, więc nie ma możliwości dostępu do informacji zgromadzonych przez innych użytkowników.

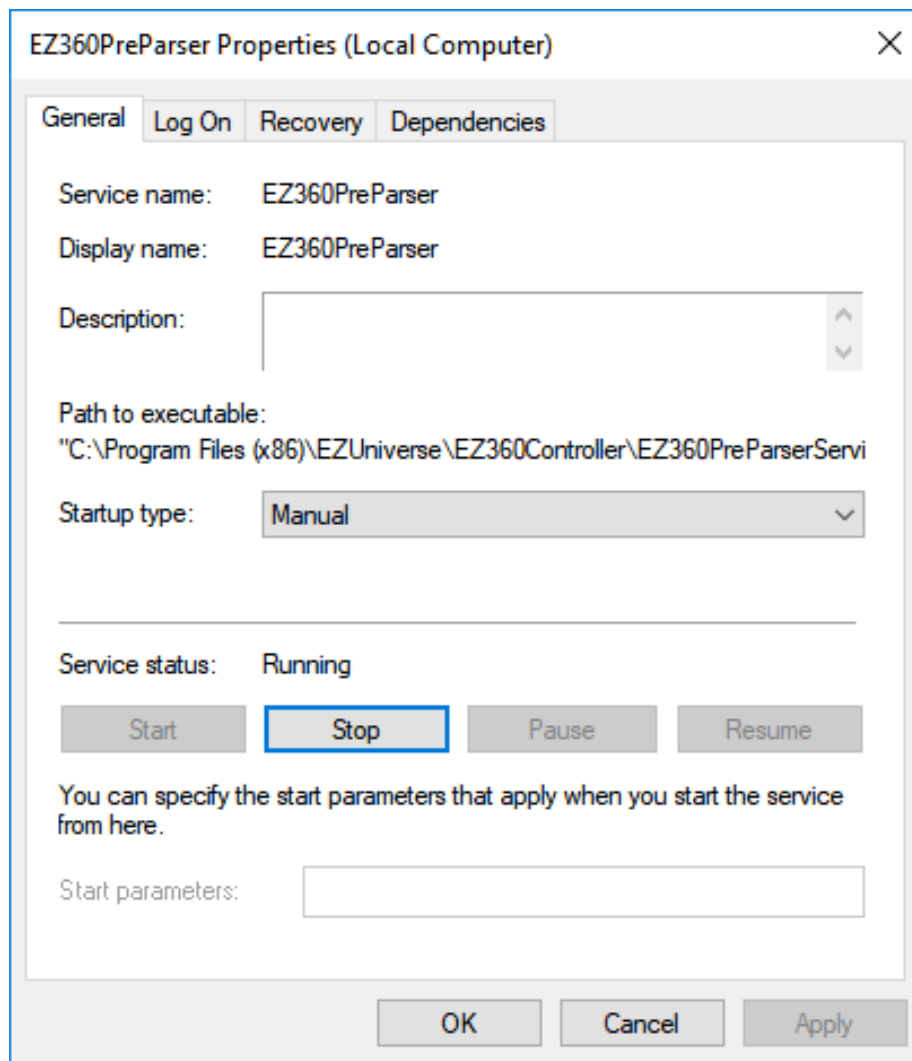
4.5 Przykład działania

Przykłady działania aplikacji i jej obsługi:

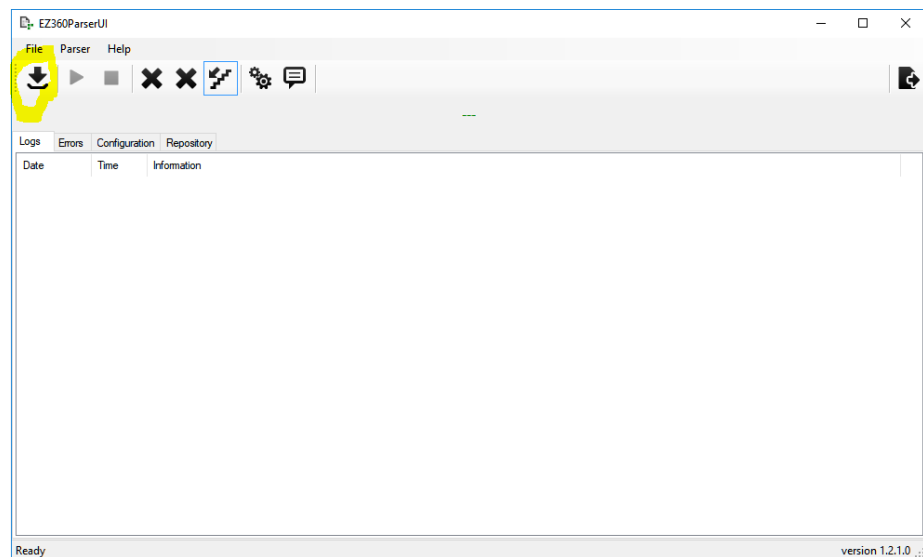
- Startowanie preparsera (rys. 4.5).
- Zatrzymywanie preparsera (rys. 4.6).
- Startowanie aplikacji (rys. 4.8).
- Zatrzymywanie aplikacji (rys. 4.9).
- Logi modułu wstępnego przetwarzania (rys. 4.10).
- Logi modułu właściwego przetwarzania (rys. 4.11).



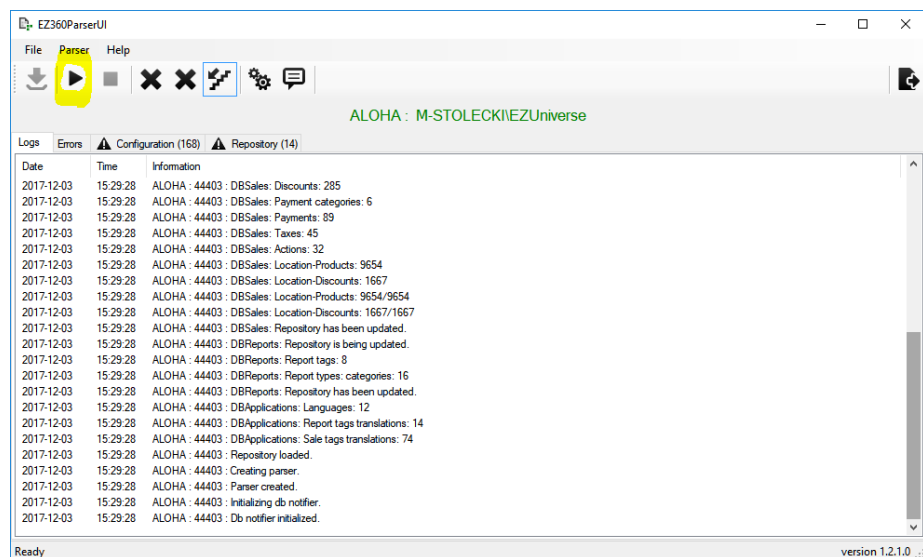
Rysunek 4.5: Przykład startowania serwisu.



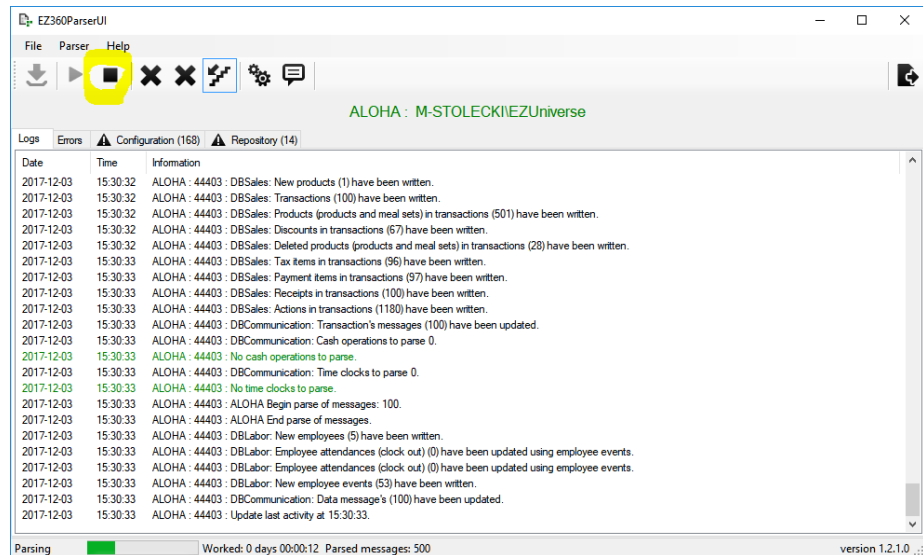
Rysunek 4.6: Przykład zatrzymywania serwisu.



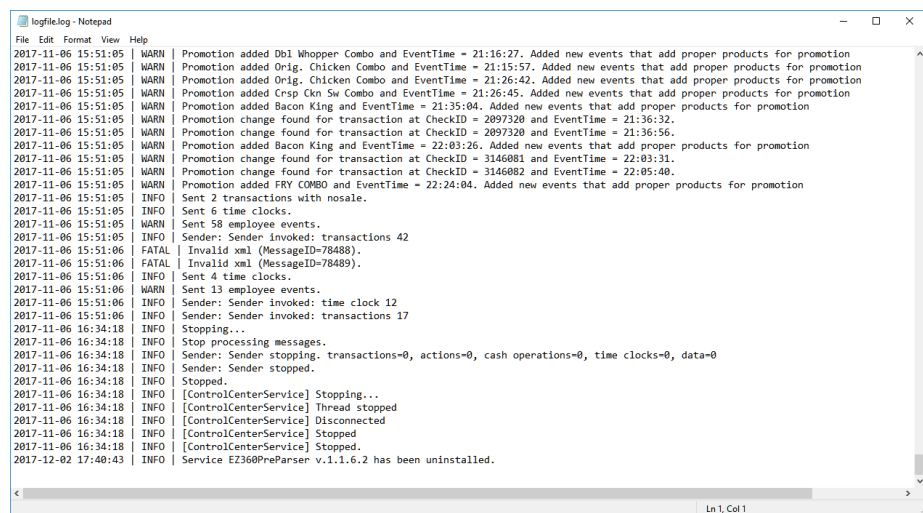
Rysunek 4.7: Inicjalizacja aplikacji - należy kliknąć na ikonę zaznaczoną żółtym kolorem.



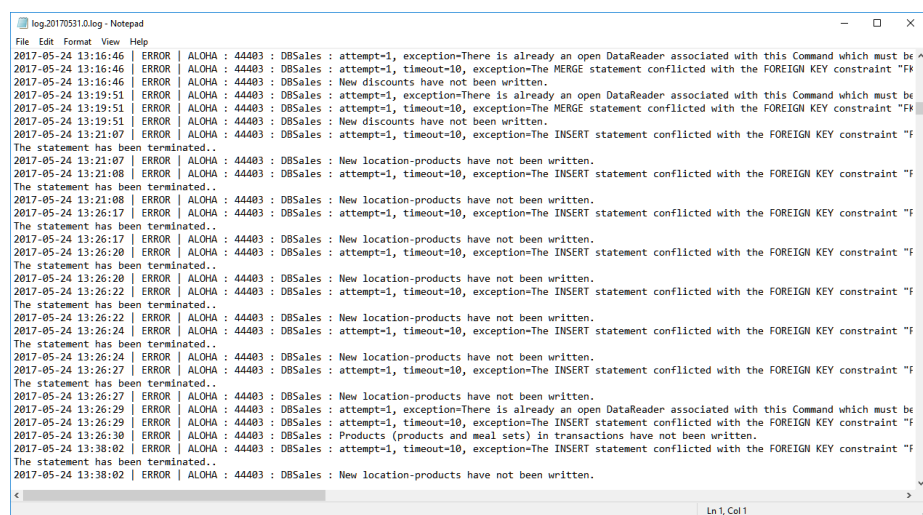
Rysunek 4.8: Startowanie aplikacji - należy kliknąć na ikonę zaznaczoną żółtym kolorem.



Rysunek 4.9: Zatrzymywanie aplikacji - należy kliknąć na ikonę zaznaczoną żółtym kolorem.



Rysunek 4.10: Logi PreParsera.



```
log-20170531.0.log - Notepad
File Edit Format View Help
2017-05-24 13:16:46 | ERROR | ALOHA : 44403 : DBSales : attempt=1, exception=There is already an open DataReader associated with this Command which must be
2017-05-24 13:16:46 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The MERGE statement conflicted with the FOREIGN KEY constraint "FK
2017-05-24 13:16:46 | ERROR | ALOHA : 44403 : DBSales : New discounts have not been written.
2017-05-24 13:19:51 | ERROR | ALOHA : 44403 : DBSales : attempt=1, exception=There is already an open DataReader associated with this Command which must be
2017-05-24 13:19:51 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The MERGE statement conflicted with the FOREIGN KEY constraint "FK
2017-05-24 13:19:51 | ERROR | ALOHA : 44403 : DBSales : New discounts have not been written.
2017-05-24 13:21:07 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
The statement has been terminated..
2017-05-24 13:21:07 | ERROR | ALOHA : 44403 : DBSales : New location-products have not been written.
2017-05-24 13:21:08 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
The statement has been terminated..
2017-05-24 13:21:08 | ERROR | ALOHA : 44403 : DBSales : New location-products have not been written.
2017-05-24 13:26:17 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
The statement has been terminated..
2017-05-24 13:26:17 | ERROR | ALOHA : 44403 : DBSales : New location-products have not been written.
2017-05-24 13:26:20 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
The statement has been terminated..
2017-05-24 13:26:20 | ERROR | ALOHA : 44403 : DBSales : New location-products have not been written.
2017-05-24 13:26:22 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
The statement has been terminated..
2017-05-24 13:26:22 | ERROR | ALOHA : 44403 : DBSales : New location-products have not been written.
2017-05-24 13:26:24 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
The statement has been terminated..
2017-05-24 13:26:24 | ERROR | ALOHA : 44403 : DBSales : New location-products have not been written.
2017-05-24 13:26:27 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
The statement has been terminated..
2017-05-24 13:26:27 | ERROR | ALOHA : 44403 : DBSales : New location-products have not been written.
2017-05-24 13:26:29 | ERROR | ALOHA : 44403 : DBSales : attempt=1, exception=There is already an open DataReader associated with this Command which must be
2017-05-24 13:26:29 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
2017-05-24 13:26:30 | ERROR | ALOHA : 44403 : DBSales : Products (products and meal sets) in transactions have not been written.
2017-05-24 13:38:02 | ERROR | ALOHA : 44403 : DBSales : attempt=1, timeout=10, exception=The INSERT statement conflicted with the FOREIGN KEY constraint "f
The statement has been terminated..
2017-05-24 13:38:02 | ERROR | ALOHA : 44403 : DBSales : New location-products have not been written.
```

Rysunek 4.11: Logi Parsera.

Rozdział 5

Specyfikacja wewnętrzna

W niniejszym rozdziale podjęta została kwestia architektury oprogramowania. Przedstawione zostały schematy baz danych oraz zakres wykorzystywanych bibliotek. Rozpisano również model ważniejszych klas oraz szczegółowo omówione zostały dane wejściowe wraz z algorytmami koniecznymi do ich odpowiedniej analizy.

5.1 Architektura systemu

Rysunek (rys. 5.1) przedstawia przepływ danych w systemie realizującym przetwarzanie informacji z kas Aloha. Jak już wielokrotnie wspomniano w niniejszej pracy, dane wychodzące z terminala kasy Aloha mają pewną ścieżkę, którą muszą pokonać zanim zostaną one przekształcone w model, mogący być prezentowany użytkownikowi i analizowany przez niego. Wspominany proces wygląda następująco:

1. Terminal kasy Aloha wysyła komunikat o zdarzeniu.
2. AlohaSpyRelay przechwytuje informację przychodzącą z terminala i przetwarza ją na format rozpoznawalny przez moduł wstępnego przetwarzania.
3. Moduł wstępnego przetwarzania zainstalowany na kontrolerze odczytuje dane przychodzące z terminala i opracowane przez AlohaSpyRelay, a następnie dokonuje ich wstępnej analizy.
4. Dane wynikowe z PreParser przekazywane są do EZ360DataInterface, który przesyła je do bazy danych ulokowanej po stronie administratora systemu.
5. Moduł właściwego przetwarzania danych odczytuje z bazy wiadomości i przetwarza je na format mogący być analizowany i oglądany przez użytkownika.
6. Dane wyjściowe z Parsersa prezentowane są w aplikacji webowej, będącej częścią systemu 360iQ dostarczanego przez firmę EZUniverse.



Rysunek 5.1: Schemat przepływu danych.

5.2 Organizacja i struktura baz danych

Moduł wstępnego przetwarzania danych:

- _RAWDATA,
- ALOHA_Transactions,
- ALOHA_CashOperations,
- ALOHA_TimeClocks,
- ALOHA_Data,
- ALOHA_Actions.

Schemat bazy danych - (rys. 5.2)

Moduł właściwego przetwarzania danych:

- Transactions,
- TransactionProducts,
- TransactionPayments,
- TransactionDiscounts,
- TransactionReceipts,
- TransactioActions.

Schemat bazy danych - (rys. 5.3)

5.3 Model klas

Niniejsza sekcja opisuje najważniejsze klasy wykorzystywane w projekcie. W zależności od modułu są to:

- Moduł wstępnego przetwarzania danych (schemat 5.4),
PreParser,
AlohaPreParser,
TransactionPreParser,
TimeClockPreParser,
CashDropPreParser,.
- Moduł właściwego przetwarzania danych (schemat 5.5).
Parser,
AlohaParser,
TransactionParser,
TimeClockParser,
CashDropParser,
DataParser.

5.4 Szczegółowa analiza danych wejściowych

Na podstawie przykładowych danych wejściowych przedstawionych na rys. 5.6. Każde zdarzenie w formacie XML zawiera następujące atrybuty:

- TerminalID - numer terminala kasowego.
- EventTime - czas wystąpienia zdarzenia.
- EmployeeID - ID pracownika.
- EmployeeName - imię i nazwisko pracownika.
- ManagerID - ID managera.
- ManagerName - imię i nazwisko managera.
- TableID - ID stanowiska, na którym ulokowany jest terminal.
- CheckID - numer transakcji.
- TransactionTypeID - ID typu zdarzenia.
- TransactionType - nazwa typu zdarzenia.
- Description - opis zdarzenia.
- Amount - wartość pieniężna związana ze zdarzeniem.
- Quantity - ilość sztuk (w przypadku produktów i płatności).
- Sender - adres IP urządzenia wysyłającego.
- ReceivedOn - data otrzymania zdarzenia po stronie kontrolera.

5.5 Algorytmy

W systemie wykorzystane zostały następujące algorytmy przetwarzające i rozszerzające dane wejściowe w formacie XML:

- Dodawanie brakujących przedmiotów będących częścią promocji do transakcji (listing został podzielony na części - kolejno: 5.7, 5.8, 5.9)
- Rozszerzanie zdarzenia XML o rozszerzone nazwy przedmiotów i wartości podatków (listing został podzielony na części - kolejno: 5.10, 5.11, 5.12)

5.5.1 Dodawanie brakujących przedmiotów będących częścią promocji do transakcji

Raz w ciągu dnia, otrzymywana jest wiadomość zawierająca PromotionXML. Za każdym razem, gdy taka informacja jest odbierana, zapisana zostaje do pliku lub aktualizowana, jeśli już istnieje. PreParser zawiera również obiekt, który przechowuje dane pochodzące ze słownika PromotionsXML. Wspomniana wiadomość jest konwertowana na instancję klasy PromotionBundle, która przechowuje informacje o promocjach i elementach w niej zawartych. Metoda bezpośrednio zaangażowana w przetwarzanie promocji nazywa się ProcessAddPromo. Zapewnia funkcjonalność, która wykrywa, czy nastąpiła zmiana promocji i wstawia produkty, których brakuje. To, co dokładnie dzieje się w wykrywaniu algorytmów promocji, można podsumować w prostych krokach:

- Przechowywana jest historia przetworzonych zdarzeń, co pozwala algorytmowi powrócić do przeszłych akcji, aby wykryć pożądany schemat,
- Szukany jest zestaw 2 instrukcji (jest to wartość minimum - zostaje ona zwiększona, jeśli wykryte są zdarzenia, których zawartość pola TransactionType zaczyna się od 6 lub 7), które poprzedzały bieżące zdarzenie,
- Jeśli znaleziony zostanie odpowiedni zestaw zdarzeń, który jest rozpoznawany jako zmiana promocji to dodawane lub usuwane są odpowiednie wydarzenia na podstawie wyników wyszukiwania odpowiednich schematów zdarzeń.

5.5.2 Rozszerzanie zdarzenia XML o rozszerzone nazwy przedmiotów i wartości podatków

Od czasu do czasu pojawia się komunikat zawierający ItemCategoryXML. Dostarcza on rozszerzonych informacji o przedmiotach i ich kategoriach. W Aloha ten sam przedmiot może być nazwany z wykorzystaniem różnych formatów:


- ItemShortName,
- ItemLongName,
- ItemLongName2,
- ItemChitName,
- ItemName.

Biorąc to pod uwagę, opracowano funkcjonalność rozszerzającą przychodzące zdarzenie XML z danych wejściowych. Dodaje ona nowe atrybuty do istniejącego pliku XML z wymienionymi wyżej nazwami elementów. Nazwa przedmiotu jest identyfikowana za pomocą tego algorytmu:


- Nazwa przedmiotu (lub jej odmiany) jest wyodrębniona z atrybutu Description,
- Uzyskana nazwa porównywana jest z nazwami produktu pochodzącymi z ItemCategoryXML,
- Gdy tylko znaleziona zostanie pasująca nazwa, znajdowane są wszystkie jej odmiany i rozszerzana jest wiadomość XML o dodatkowe pola z odpowiednimi wartościami,
- Od tego momentu zdarzenie zawiera wszystkie potrzebne informacje o przedmiocie, które mogą być wykorzystane podczas dalszego przetwarzania.

Metoda implementująca tę funkcjonalność to: ExtendMessageWithAdditionalItemInfo. Słownik ItemCategoryXML jest przechowywany zarówno na dysku, jak i jako obiekt XElement po stronie serwisu. Jest aktualizowany za każdym razem, gdy przychodzi nowa wiadomość z nową wersją ItemCategoryXML.


RAWDATA (IN)

 INRawDataID
Type
Source
Port
Data
Status
ProcessID
IsSent
SentTimeDiff
UTC
CreatedOn
CreatedBy
ModifiedBy
ModifiedOn


ALOHA_Actions (IN)

 ALOHAActionID
LocationID
ControllerID
DeviceID
Number
Body
ProcessID
Status
CreatedOn
CreatedBy
ModifiedOn
ModifiedBy


ALOHA_Transactions (IN)

 ALOHATransactionID
LocationID
ControllerID
DeviceID
Number
Body
ProcessID
Status
CreatedOn
CreatedBy
ModifiedOn
ModifiedBy


ALOHA_CashOperations (IN)

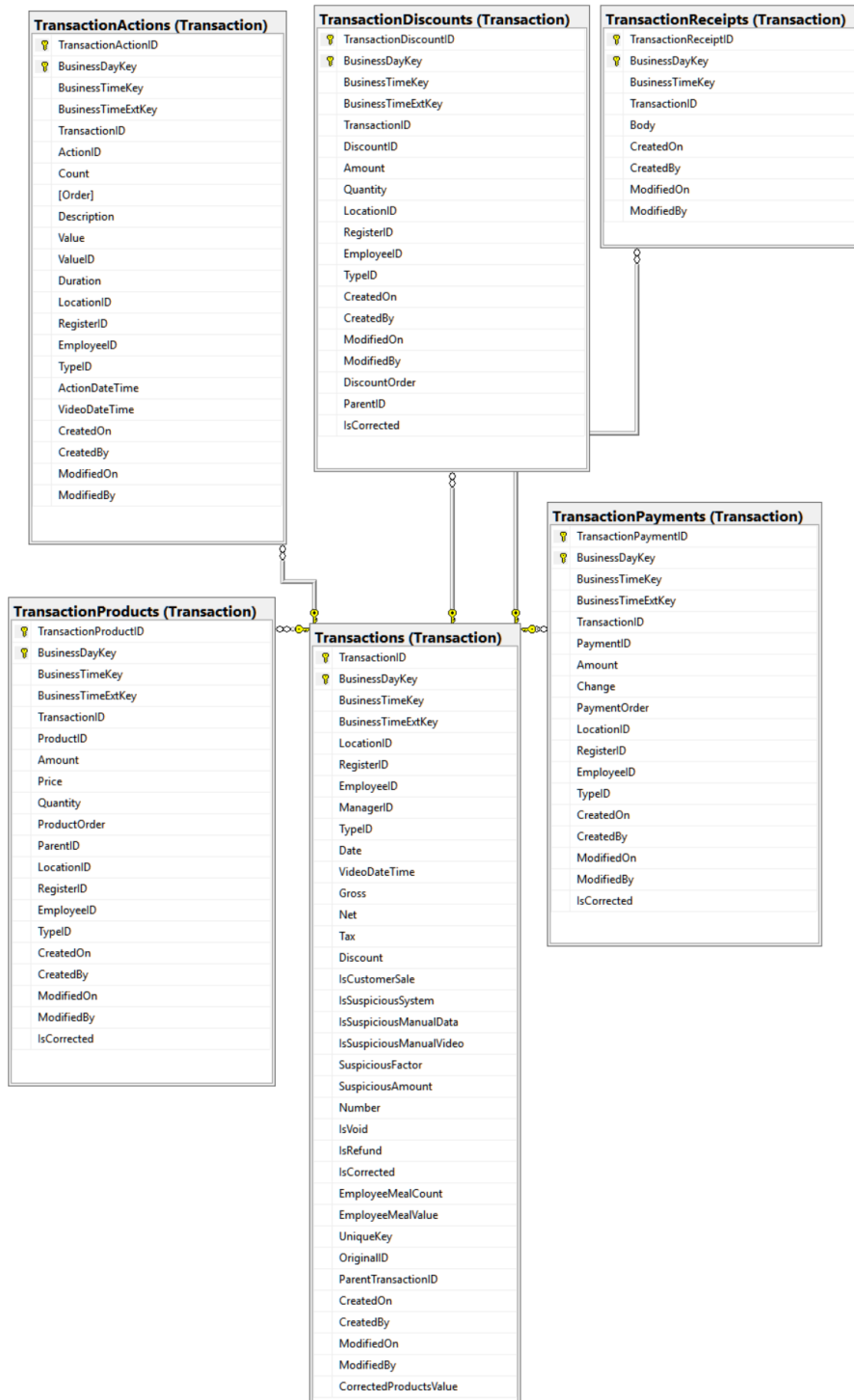
 ALOHACashOperationID
LocationID
ControllerID
DeviceID
Number
Body
ProcessID
Status
CreatedOn
CreatedBy
ModifiedOn
ModifiedBy

ALOHA_Data (IN)

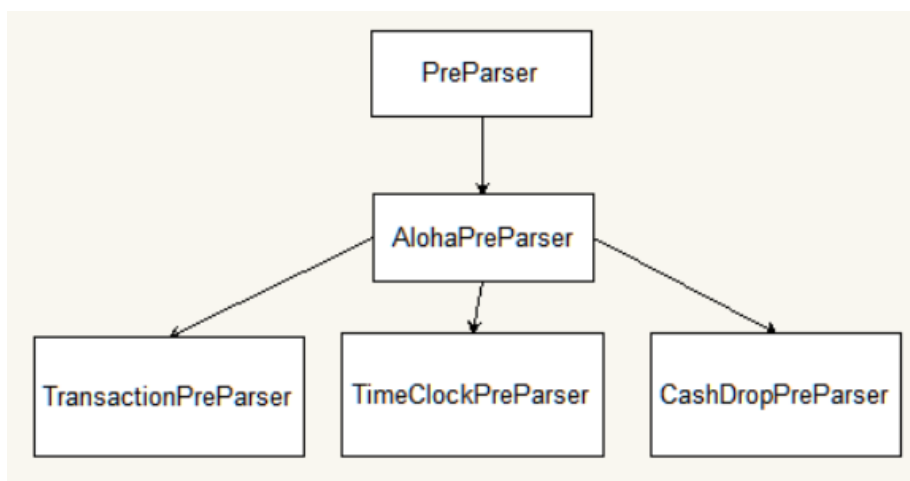
 ALOHADataID
LocationID
ControllerID
DeviceID
Number
Body
ProcessID
Status
CreatedOn
CreatedBy
ModifiedOn
ModifiedBy

ALOHA_TimeClocks (IN)

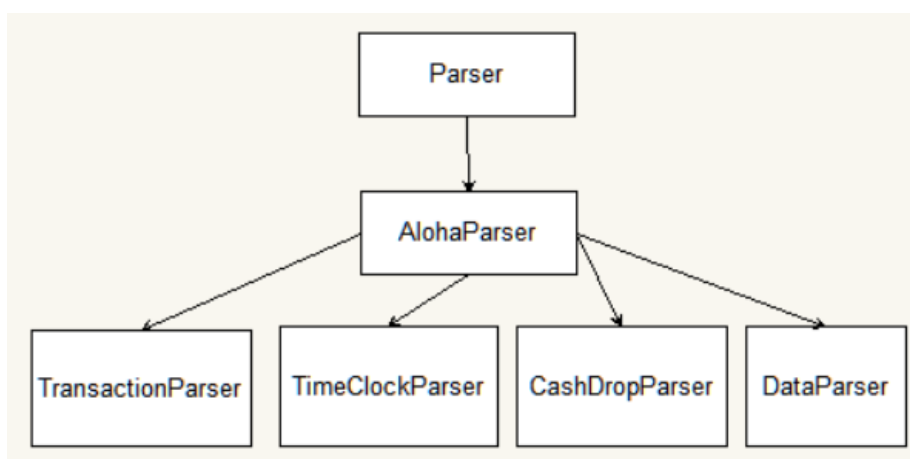
 ALOHATimeClockID
LocationID
ControllerID
DeviceID
Number
Body
ProcessID
Status
CreatedOn
CreatedBy
ModifiedOn
ModifiedBy



Rysunek 5.3: Schemat bazy danych dla modułu właściwego przetwarzania.



Rysunek 5.4: Diagram przedstawiający najważniejsze klasy - PreParser.



Rysunek 5.5: Diagram przedstawiający najważniejsze klasy - Parser.

```
1 <SpyMessage TerminalID="4" EventTime="23:59:57" EmployeeID="94205" EmployeeName="AHMET B?RL?K" ManagerID="0" ManagerName="" TableID="4194420" CheckID="40116" TransactionTypeID="59" TransactionType="TOTAL" Description="" Amount="18.75" Quantity="0" Sender="192.168.100.4" ReceivedOn="2017.11.30 00:00:08.204" />
```

Rysunek 5.6: Przykładowe zdarzenie - dane wejściowe Aloha.

```
1 private void ProcessAddPromo(XElement action, XElement
    lastAction, List<SpyMessage> actions,
2 List<PromotionBundle> promotionBundles, List<string>
    warningMessages, List<SpyMessage> result)
3 {
4     var listOfActions = actions.Select(x => x.Body).ToList();
5     var lastActionTransactionType = lastAction?.Attribute("
        TransactionType")?.Value ?? string.Empty;
6     if (lastActionTransactionType == "DELETE_PROMO")
7     {
8         var actionsOffset = 2;
9         var cancelItemActionAppeared = false;
10        var addItemActionAppeared = false;
11        for (var i = 1; i < actionsOffset + 1; i++)
12        {
13            var indexToMove = listOfActions.IndexOf(lastAction) - i;
14            if (indexToMove >= 0)
15            {
16                var currentAction = actions.ElementAt(indexToMove).Body;
17                var currentActionName = currentAction?.Attribute("
                    TransactionType")?.Value ?? string.Empty;
18                if (currentActionName.StartsWith("6") || currentActionName.
                    StartsWith("7"))
19                {
20                    actionsOffset++;
21                }
22
23                if (currentActionName == "ADD_ITEM" &&
                    cancelItemActionAppeared)
24                {
25                    addItemActionAppeared = true;
26                    break;
27                }
28
29                if (currentActionName == "CANCEL_ITEM" && !
                    addItemActionAppeared)
30                {
31                    cancelItemActionAppeared = true;
32                }
33            }
34            else
35            {
36                break;
37            }
38        }
```

Rysunek 5.7: Dodawanie brakujących przedmiotów będących częścią promocji do transakcji - część pierwsza

```

1  if (addItemActionAppeared)
2  {
3      IsPromoChanged = true;
4      AreItemsAlreadyAdded = true;
5      warningMessages.Add(
6          $"Promotion change found for transaction at CheckID = {action
              ?.Attribute("CheckID")?.Value ?? string.Empty} and
              EventTime = {action?.Attribute("EventTime")?.Value ??
              string.Empty}.";
7      }
8  }
9  if (!AreItemsAlreadyAdded)
10 {
11     var promoName = action?.Attribute("Description")?.Value ??
        string.Empty;
12     var itemsMultiplier = 1;
13     if (promoName.StartsWith("2 for"))
14     {
15         itemsMultiplier = 2;
16     }
17     var bundle =
18         promotionBundles.FirstOrDefault(x => x.PromotionName.
                ToUpperInvariant() ==
19             promoName.ToUpperInvariant());
20     var count = 1;
21     var spyMessage = actions.FirstOrDefault(x => x.Body == action)
        ;
22     if (bundle != null)
23     {
24         foreach (var product in bundle.ProductsInPromotion)
25         {
26             var element = new XElement(action);
27             element.SetAttributeValue("TransactionType", "ADD_ITEM");
28             element.SetAttributeValue("TransactionTypeID", "8");
29             element.SetAttributeValue("Quantity", "1");
30             element.SetAttributeValue("Description", product.Name);
31             element.SetAttributeValue("Amount", product.DisplayAs);
32             element.Add(new XAttribute("ItemId", product.PLU));
33             element.Add(new XAttribute("ItemName", product.Name));
34             var message = new SpyMessage
35             {
36                 Body = element,
37                 CheckID = long.Parse(element?.Attribute("CheckID")?.Value),
38                 RawDataID = -1,
39                 TransactionType = element?.Attribute("TransactionType")?.Value
                    ?? string.Empty,
40                 LastActionDate = actions.ElementAt(actions.IndexOf(spyMessage)
                    ).LastActionDate,
41                 UtcTime = actions.ElementAt(actions.IndexOf(spyMessage)).
                    UtcTime
42             };

```

Rysunek 5.8: Dodawanie brakujących przedmiotów będących częścią promocji do transakcji - część druga

```
1  for (var i = 0; i < itemsMultiplier; i++)
2  {
3      result.Add(message);
4      count++;
5  }
6  }
7  warningMessages.Add(
8      $"Promotion added {promoName} and EventTime = {action?.
        Attribute("EventTime")?.Value ?? string.Empty}. Added new
        events that add proper products for promotion");
9  }
10 else
11 {
12     warningMessages.Add(
13         $"Not found bundle: {promoName} in bundle collection for
            action's CheckID: {action?.Attribute("CheckID")?.Value ??
            string.Empty}.");
14 }
15 }
16
17 AreItemsAlreadyAdded = false;
18 }
```

Rysunek 5.9: Dodawanie brakujących przedmiotów będących częścią promocji do transakcji - część trzecia

```

1  private void ExtendMessegeWithAdditionalItemInfo(XElement
      action, XElement itemCategoryXML, List<TaxInformation>
      taxInformations, List<PromotionBundle> promotionBundles)
2  {
3      var itemName = action?.Attribute("Description")?.Value ??
      string.Empty;
4      if (itemCategoryXML != null)
5      {
6          var itemNodes = itemCategoryXML.Elements();
7          action.Add(new XAttribute("ItemId", string.Empty));
8          action.Add(new XAttribute("ItemShortName", string.Empty));
9          action.Add(new XAttribute("ItemLongName", string.Empty));
10         action.Add(new XAttribute("ItemLongName2", string.Empty));
11         action.Add(new XAttribute("ItemChitName", string.Empty));
12         action.Add(new XAttribute("ItemName", string.Empty));
13         action.Add(new XAttribute("Tax", "0"));
14         foreach (var item in itemNodes)
15         {
16             var itemShortName = item?.Attribute("ItemShortName")?.Value ??
            string.Empty;
17             var itemLongName = item?.Attribute("ItemLongName")?.Value ??
            string.Empty;
18             var itemLongName2 = item?.Attribute("ItemLongName2")?.Value ??
            string.Empty;
19             var itemChitName = item?.Attribute("ItemChitName")?.Value ??
            string.Empty;
20             var itemNameAttr = item?.Attribute("ItemName")?.Value ??
            string.Empty;
21             if (!string.IsNullOrEmpty(itemShortName) && itemShortName ==
                itemName
22             || !string.IsNullOrEmpty(itemLongName) && itemLongName ==
                itemName
23             || !string.IsNullOrEmpty(itemLongName2) && itemLongName2 ==
                itemName
24             || !string.IsNullOrEmpty(itemChitName) && itemChitName ==
                itemName
25             || !string.IsNullOrEmpty(itemNameAttr) && itemNameAttr ==
                itemName)
26             {
27                 action.SetAttributeValue("ItemId", item?.Attribute("ItemId")?.
                    Value ?? string.Empty);
28                 action.SetAttributeValue("ItemShortName", itemShortName);
29                 action.SetAttributeValue("ItemLongName", itemLongName);
30                 action.SetAttributeValue("ItemLongName2", itemLongName2);
31                 action.SetAttributeValue("ItemChitName", itemChitName);
32                 action.SetAttributeValue("ItemName", itemNameAttr);

```

Rysunek 5.10: Rozszerzanie zdarzenia XML o rozszerzone nazwy przedmiotów i wartości podatków. - część pierwsza

```
1
2  if (taxInformations != null)
3  {
4      var taxId = item?.Attribute("ItemTaxId")?.Value ?? "0";
5      var tax = taxInformations.FirstOrDefault(x => x.TaxId == taxId
6      );
7      if (tax != null)
8      {
9          if (decimal.TryParse(
10             action?.Attribute("Amount")?.Value ?? "0",
11             NumberStyles.Any,
12             CultureInfo.InvariantCulture,
13             out var amountValue))
14          {
15              var taxValue = amountValue * tax.TaxRate;
16              action.SetAttributeValue("Tax", taxValue.ToString(CultureInfo.
17                  InvariantCulture));
18          }
19      }
20  }
21  }
22  else
23  {
24      var itemsInPromotions = promotionBundles.Select(x => x.
25          ProductsInPromotion).ToList();
26      var items = new List<Product>();
27      foreach (var itemsInPromotion in itemsInPromotions)
28      {
29          foreach (var itemInPromotion in itemsInPromotion)
30          {
31              if (!items.Any(x => x.PLU == itemInPromotion.PLU))
32              {
33                  items.Add(itemInPromotion);
34              }
35          }
36      }
37      action.Add(new XAttribute("ItemId", string.Empty));
38      action.Add(new XAttribute("ItemShortName", string.Empty));
39      action.Add(new XAttribute("ItemLongName", string.Empty));
40      action.Add(new XAttribute("ItemLongName2", string.Empty));
41      action.Add(new XAttribute("ItemChitName", string.Empty));
42      action.Add(new XAttribute("ItemName", string.Empty));
43      action.Add(new XAttribute("Tax", "0"));
```

Rysunek 5.11: Rozszerzanie zdarzenia XML o rozszerzone nazwy przedmiotów i wartości podatków. - część druga

```
1  foreach (var item in items)
2  {
3  if (!string.IsNullOrEmpty(item.ShortName) && item.ShortName ==
    itemName
4  || !string.IsNullOrEmpty(item.LongName) && item.LongName ==
    itemName
5  || !string.IsNullOrEmpty(item.Name) && item.Name == itemName)
6  {
7  action.SetAttributeValue("ItemId", item.PLU);
8  action.SetAttributeValue("ItemShortName", item.ShortName);
9  action.SetAttributeValue("ItemLongName", item.LongName);
10 action.SetAttributeValue("ItemLongName2", string.Empty);
11 action.SetAttributeValue("ItemChitName", string.Empty);
12 action.SetAttributeValue("ItemName", item.Name);
13 }
14 }
15 }
16 }
```

Rysunek 5.12: Rozszerzanie zdarzenia XML o rozszerzone nazwy przedmiotów i wartości podatków. - część trzecia

Rozdział 6

Testowanie i uruchamianie

W niniejszym rozdziale zostaną opisane przeprowadzone testy zarówno dla modułu wstępnego przetwarzania danych jak i części odpowiedzialnej za właściwe przetwarzanie informacji. Dane testowe zostały zaimportowane w formie backup-u bazy danych z lokalizacji, na której codziennie działa opisane w tej pracy rozwiązanie. Testowane przypadki obejmują dane zawarte w zrzucie z bazy danych (03-11-2017) z lokalizacji położonej na terenie Stanów Zjednoczonych.

6.1 PreParser

Tabela 6.1 przedstawia wyniki procedury testowej modułu wstępnego przetwarzania. Dane wejściowe zgromadzone są w tabeli `_RAWDATA` i wyniki ich analizy powinny zostać umieszczone w tabelach: `Aloha_Transactions`, `Aloha_TimeClocks`, `Aloha_CashOperations`. Całość tabel znajduje się bazie `EZ360Communication`.

6.2 Parser

Tabela 6.2 przedstawia wyniki procedury testowej modułu właściwego przetwarzania. Dane wejściowe zgromadzone są w tabelach: `Aloha_Transactions`, `Aloha_TimeClocks`, `Aloha_CashOperations` i wyniki ich analizy powinny zostać umieszczone w tabelach: `Transaction.Transactions`, `Transaction.Taxes`, `Transaction.Receipts`, `Transaction.Products`, `Transaction.Payments`, `Transaction.Discounts` (`EZ360Sales`) oraz `Employee.Employees`, `Employee.EmployeeAttendances` (`EZ360Labor`).

Tablica 6.1: Testy modułu wstępnego przetwarzania danych.

Testowany przypadek	Dane wejściowe	Dane wyjściowe
Procesowanie całości danych wejściowych z testowanej paczki.	Zawartość tabeli _RAWDATA. Ilość zdarzeń do przetworzenia - 78490	Liczba przetworzonych danych: Aloha.Transactions - 3700, Aloha.TimeClocks - 232, Aloha.CashOperations - 34
Przypadek pustej tabeli z danymi wejściowymi.	Zawartość tabeli _RAWDATA. Ilość zdarzeń do przetworzenia - 0	Brak danych do przetworzenia. PreParser oczekuje na przyjscie nowych danych do przetwarzania.
Przypadek tabeli z danymi wejściowymi zawierającymi jedną niepełną transakcję.	Zawartość tabeli _RAWDATA. Ilość zdarzeń do przetworzenia - 9. Brak zdarzenia kończącego transakcje.	Dane zostają przetworzone, jednakże transakcja nie zostaje wysłana, ponieważ PreParser czeka na przyjscie zdarzenia kończącego transakcje.
Przypadek tabeli z danymi wejściowymi zawierającymi jedną pełną transakcję.	Zawartość tabeli _RAWDATA. Ilość zdarzeń do przetworzenia - 10. Występuje zdarzenie kończące transakcje.	Dane zostają przetworzone, transakcja zostaje wysłana.

Tablica 6.2: Testy modułu właściwego przetwarzania danych.

Testowany przypadek	Dane wejściowe	Dane wyjściowe
Procesowanie transakcji	Zawartość tabeli Aloha.Transactions. Ilość zdarzeń do przetworzenia - 3700	Dodano 3700 nowych transakcji do systemu.
Procesowanie wpłat/wypłat z kasy	Zawartość tabeli Aloha.CashOperations. Ilość zdarzeń do przetworzenia - 34	Dodano 34 nowych wpłat/wypłat do systemu.
Procesowanie wpłat/wypłat z kasy	Zawartość tabeli Aloha.TimeClocks. Ilość zdarzeń do przetworzenia - 232	Dodano 232 nowych informacji o obecnościach pracowników do systemu.
Brak danych do przetworzenia	Pusta tabela Aloha.Transactions.	Parser czeka na nadejście danych do przetworzenia.
Błędne dane wejściowe	Tabela Aloha.Transactions z transakcjami zawierającymi błędne XML-e.	Parser oznacza nieprawidłowe dane odpowiednim statusem i kontynuuje pracę.

Rozdział 7

Uwagi o przebiegu i wynikach prac

7.1 Stopień realizacji zagadnienia

Opisywane w poprzednich rozdziałach zagadnienie zostało zrealizowane w stopniu całkowitym. Aplikacja przeszła pomyślnie proces projektowania, tworzenia oraz testów. Jest ona wdrożona jako część systemu 360iQ udostępnianego przez firmę EZUniverse i widnieje w nim pod nazwami AlohaPreParser oraz AlohaParser. Oprogramowanie stabilnie działa na dużej liczbie kontrolerów i dostarcza wysoce wiarygodne dane, mogące być analizowane przez użytkownika systemu.

7.2 Napotkane problemy

Głównymi problemami, które generuje terminal kasowy Aloha, są rozdrobnione i niespójne dane. Każda transakcja dostarczana jest jako seria zdarzeń, które niekoniecznie zostają wysyłane do kontrolera w należytych porządku. Wymagało to implementacji funkcjonalności łączącej zdarzenia w grupy, który można opisywać jako transakcje lub inne zdarzenia zachodzące na kasie. Dodatkowymi problemami były nieścisłości związane z produktami zawartymi w promocji oraz wiele nazw, które opisywały ten sam przedmiot. Rozwiązano te problemy wykorzystując opisane wcześniej w niniejszej pracy algorytmy rozszerzające i modyfikujące dane.

7.3 Dalszy rozwój

Z racji faktu ciągłej pracy systemu, konieczne jest jego stałe wspieranie oraz aktualizowanie. Rozwiązywane są również zgłaszane przez klienta usterki i błędy w przetwarzaniu danych. Konieczność ciągłej rozbudowy aplikacji wynika z wdrażania systemu na coraz większą ilość lokalizacji co wymaga specjalnych poprawek lub optymalizacji wynikających z uwarunkowań regionalnych.

Rozdział 8

Podsumowanie

W niniejszej pracy opisano proces realizacji zagadnienia, którym autor zajmuje się regularnie w ramach pracy zawodowej. Wiedza zdobyta podczas pracy z tym typem danych oraz urządzeń, pozwoliła zbudować od samych podstaw własny moduł realizujący kompleksowe przetwarzanie danych. Nieocenionym czynnikiem były również umiejętności wypracowane podczas studiów inżynierskich, które znacznie poszerzyły horyzonty myślowe autora oraz zapewniły podstawowe umiejętności, które następnie mogły być rozwijane dzięki samodzielnej pracy autora. Aplikacja przeszła szereg wymagających testów sprawdzających jakość przetwarzanych danych oraz ich poprawność. Oprogramowanie zostało wdrożone w sklepach na całym świecie (największa ilość wspieranych sklepów znajduje się na terenie Stanów Zjednoczonych) i jest ono na bieżąco monitorowane oraz rozwijane przez autora. Z racji faktu przeprowadzenia znacznej ilości testów oraz analizy przypadków testowych, autor niniejszej pracy ograniczył się do umieszczenia tylko tych najbardziej podstawowych, ponieważ całościowy opis i analiza testowanych przypadków nie był możliwy z racji zbyt ich dużego stopnia skomplikowania oraz obszerności.

Bibliografia

- [1] T-SQL Reference. <https://docs.microsoft.com/en-us/sql/t-sql/language-reference/>. [data dostępu: 12 grudnia 2017].
 - [2] Jon Bentley. *Perelki programowania. Wydanie II*. Helion, Gliwice, 2012.
 - [3] Ralph Johnson John Vlissides Erich Gamma, Richard Helm. *Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku*. Helion, Gliwice, 2010.
 - [4] Adam Freeman. *ASP.NET MVC 5. Zaawansowane programowanie*. Helion, Gliwice, 2015.
 - [5] Johnson M. Hart. *Programowanie w systemie Windows. Wydanie IV*. Helion, Gliwice, 2010.
 - [6] Andrew Stellman Jennifer Greene. *Rusz głową! C#*. Helion, Gliwice, 2014.
 - [7] Piotr Wróblewski. *Algorytmy, struktury danych i techniki programowania. Wydanie IV*. Helion, Gliwice, 2010.
- [1] [6] [3] [7] [4] [5] [2]