Mariusz Sadowski

Sebastian Lisowski

**Projekt Goldman Sachs – Lost and Found Application**

**Dokumentacja**

1. **Główny cel projektu**

Aplikacja ma na celu zbieranie danych z różnych stron źródłowych o niejednolitym formacie (m.in. pliki HTML, XLS i PDF) i wyświetlenie ich użytkownikowi w przystępnej i wspólnej formie. Dane zbierane przez aplikację dotyczą rzeczy zgubionych w różnych miejscach w Polsce i dostarczonych do odpowiedniego biura rzeczy znalezionych (m.in. nazwa, data znalezienia, miejsce znalezienia).

1. **Struktura projektu (moduły Maven)**

Projekt składa się z 8 modułów Maven:

- application (element wejściowy aplikacji)

- service (kontrolowanie pobierania i wysyłania danych)

- dao (dostęp do bazy danych)

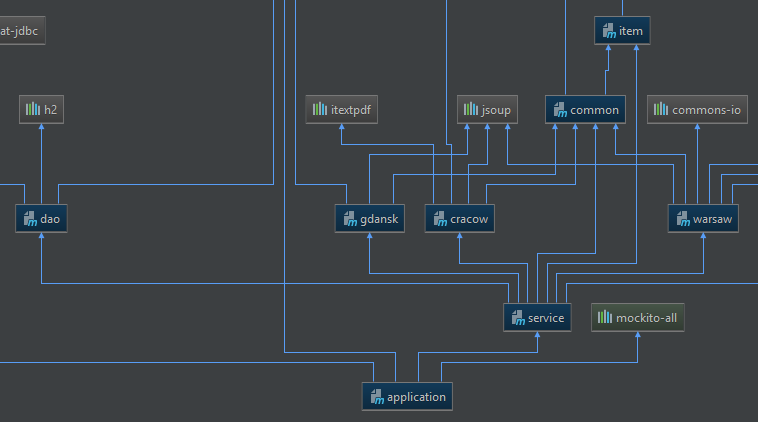
- gdansk (parser zbierający dane z formatu HTML)

- cracow (parser zbierający dane z formatu PDF)

- warsaw (parser zbierający dane z formatu XLS)

- common (posiada interfejs parserów)

- item (ujednolicona forma danych zebranych przez parsery)



Szczegółowa struktura danych znajduje się w dołączonym pliku o nazwie project\_structure.png.

1. **Wykorzystane technologie i biblioteki**

* AngularJS/HTML/CSS do frontendu
* Apache Maven jako szkielet aplikacji
* Framework Spring boot
* JUnit oraz Mockito do testów jednostkowych
* JDBC do łączenia z bazą danych (pośrednio wykonywana przez Spring)
* Embedded H2 jako slinik bazy danych
* Jsoup do przetwarzania stron HTML
* Biblioteka iText do przetwarznia plików PDF
* Biblioteka Apache POI do przetwarzania plików XLS

1. **Szczegółowy opis modułów i zależności między nimi**

* Moduł item

Jest to JavaBean służący jako jednolity model danych zebranych z parserów

* Moduł common

Zawiera interfejs Parser, z którego korzystają wszystkie parsery. Interfejs Parser posiada dwie deklaracje metod:

- void parseData() – ma na celu pobierania danych ze strony i zapisuje w postaci lokalnego modelu w postaci niejednolitej zależnej od parsera

- List<Item> getParsedData() – przetwarza zebrane dane i zwraca je w postaci jednolitego modelu Item

* Moduły warsaw, gdansk, cracow

Zajmują się przeszukiwaniem oraz wydobywaniem danych z odpowiedniego źródła. Parsery warsaw i cracow tworzą tymczasowe dokumenty w odpowiednim formacie, które są przetwarzane w celu wyciągniecia danych. Każdy parser posiada swoją własną strukturę dancyh (DataSource), która później jest konwertowana na Item. Każdy z nich implementuje metody z interfejsu Parser.

* Moduł service

Wykorzystuje wszystkie parsery do zbierania danych, DAO do komunikacji z bazą danych oraz lokalny konwerter do przejścia z Item do LostItem, które jest powiązane z bazą danych. Klasa Service jest naszym kontrolerem z MVC. Zawiera dwie metody:

- public void DownloadData() – uruchamiana cyklicznie za pomocą Schedulera (jako adnotacja ze Springa wykorzystująca wyrażenie Cron). Aktualizuje dane w bazie danych

- public List<LostItem> getAllItems() – metoda zmapowana na /items; przez protokół HTTP zwraca wszystkie przedmioty znajdujące się w bazie danych

* Moduł dao

Posiada Interfejs LostItemDAO. Interfejs ten posiada 3 metody:

- ArrayList<LostItem> getListOfLostItem – zwraca całą listę LostItem z bazy danych

- void deleteListOfLostItem() – usuwa wszystkie rekordy z bazy danych

- void insertListOfLostItem() – wstawia listę LostItem do bazy danych

Klasa LostItemDAOimpl korzysta z interfejsu LostItemDAO. Połączenie z bazą danych w poszczególnych implementacjach metod jest zrealizowana za pomocą JDBC (pośrednio przez Spring).

Moduł dao posiada również klasę LostItem, która reprezentuje encję w bazie danych.

* Moduł application

Jest to punkt wejściowy aplikacji. Definiuje funkcje main rozpoczynającą pracę frameworka Spring boot. W tym samym module znajduje się konfiguracja Spring boota (m.in. definiująca połączenie z bazą danych H2) oraz frontend. Główna strona używa HTML w połączeniu z CSS, natomiast zapytania HTTP do kontrolera są wysyłane za pomocą AngularJS. AngularJS jest także wykorzystywany do wyświetlania na stronie informacji na temat przedmiotów.

1. **Testowanie**

Poprawność działania elementów jest zapewniona dzięki zastosowaniu testów jednostkowych, wykorzystujących narzędzia JUnit oraz Mockito. Testowanie każdego z parserów polega na porównywaniu danych z lokalnych plików z danymi oczekiwanymi. Maven przy każdym instalowaniu aplikacji wymaga testowania, co znacząco ułatwia sprawdzanie poprawności działania poszczególnych elementów aplikacji.