

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Informatyka (INF)
SPECJALNOŚĆ: Inżynieria systemów informatycznych (INS)

**Projektowanie systemów z dostępem w języku
naturalnym**

Opracowanie algorytmu określającego możliwe
zamienniki dla błędnych słów w tekstach w języku
polskim

AUTOR:
Radosław Taborski - 209347
Piotr Konieczny - 209174

PROWADZĄCY PROJEKT:
dr inż. Dariusz Banasiak

OCENA PROJEKTU:

Spis treści

Spis rysunków	3
Spis tabel	4
Spis listingów	5
1 Wstęp	6
1.1 Cele projektu	6
1.2 Założenia projektowe	6
1.3 Zakres projektu	6
2 Opis użytych algorytmów	7
3 Projekt i implementacja aplikacji	8
3.1 Funkcje aplikacji - diagram przypadków użycia	8
3.2 Interfejs aplikacji	9
3.3 Wpływ parametrów na wyniki wyszukiwania	10
3.4 Realizacja wybranych funkcjonalności	13
4 Optymalizacja	14
5 Wdrożenie i testowanie aplikacji	15
6 Podsumowanie	16
Literatura	17

Spis rysunków

3.1	Diagram przypadków użycia	8
3.2	Interfejs aplikacji	9
3.3	Przykładowy wynik wyszukiwania sugestii	10
3.4	Zwiększenie ilości wyświetlanych sugestii	10
3.5	Przykładowy wynik wyszukiwania sugestii dla małej ilości odpowiedzi	11
3.6	Zwiększenie odległości Levenhstaina	11
3.7	Zwiększenie ilości zmian	12
3.8	Wyszukiwanie sugestii z poziomu interfejsu	13

Spis tabel

Spis listingów

6.1	Przykład listingu	16
-----	-----------------------------	----

Rozdział 1

Wstęp

1.1 Cele projektu

Głównym celem projektu jest stworzenie aplikacji, która będzie umożliwiała wpisanie przez użytkownika dowolnego tekstu w języku polskim oraz pozwoli na jego sprawdzenie pod względem użycia poprawnych słów. Następnie użytkownik będzie miał możliwość wstawienia za każde z wychwyconych błędnych wyrazów, innego, jednego z tych, które będą sugerowane przez aplikację, bądź też zignorowanie sugestii. Aplikacja wyposażona będzie w algorytm sprawdzający czy wyraz istnieje w słowniku języka polskiego, gdy takiego nie znajdzie, wyszukiwane będą słowa o podobnej budowie, lecz nie koniecznie pasujące znaczeniowo do kontekstu zdania.

1.2 Założenia projektowe

W projekcie wykorzystany został język programowania C#, z wykorzystaniem technologii WPF, umożliwiającej tworzenie graficznego interfejsu użytkownika w oparciu o składnię XAML.

W projekcie wykorzystany został również słownik języka polskiego dostępny na stronie: <http://sjp.pl/slownik/growy>. Słownik ten zawiera podstawowe formy słów wraz ze wszystkimi możliwymi odmianami. Słownik ten przewiduje jedynie słowa o maksymalnej długości wynoszącej 15 znaków i nie zawiera żadnych nazw własnych tzn. imion, nazw krajów, miast itp.

1.3 Zakres projektu

Zakres projektu dotyczy zaprojektowania i implementacji aplikacji umożliwiającej pisanie własnych tekstów, które na bieżąco są sprawdzane pod względem poprawności ortograficznej.

Umożliwione użytkownikowi zostało również wklejanie gotowych tekstów, w celu ich sprawdzenia.

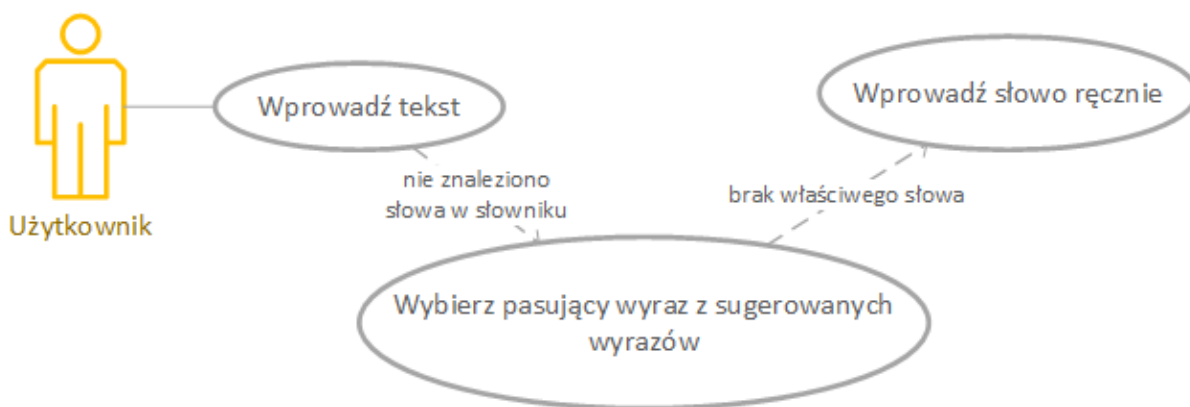
Rozdział 2

Opis użytych algorytmów

Rozdział 3

Projekt i implementacja aplikacji

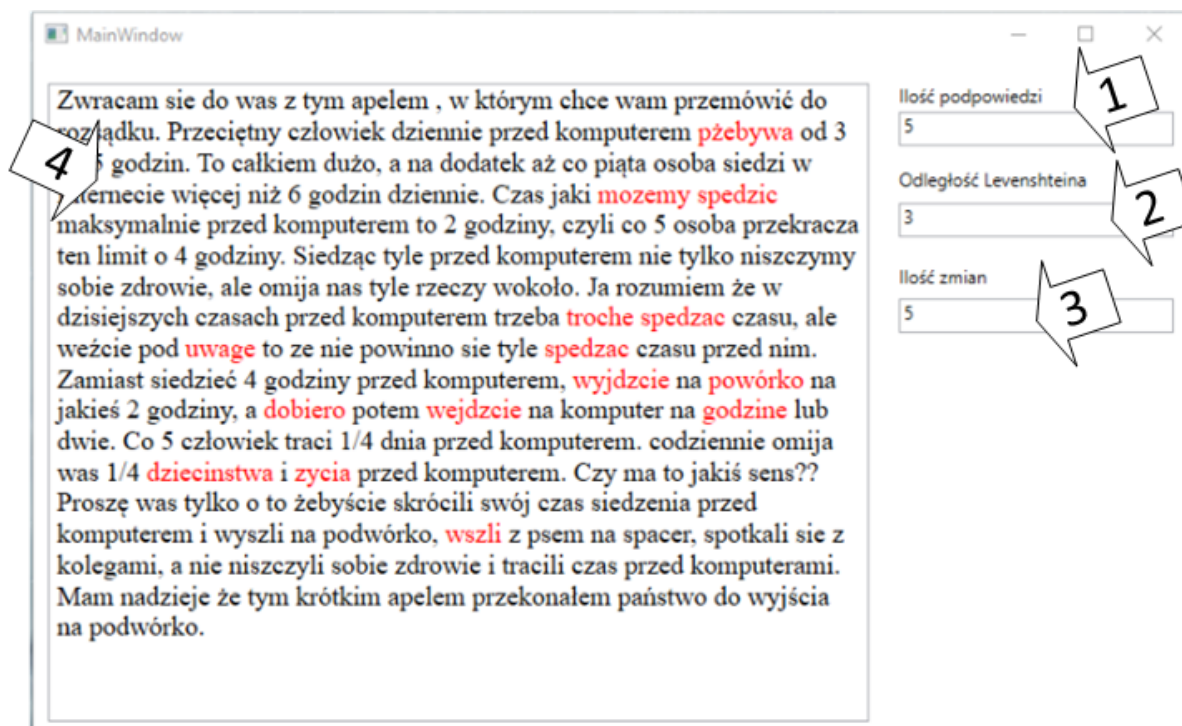
3.1 Funkcje aplikacji - diagram przypadków użycia



Rysunek 3.1 Diagram przypadków użycia

Głównym zadaniem aplikacji jest wyszukiwanie błędnych wyrazów w języku polskim oraz ich korekcja. Aby tego dokonać aplikacja porównuje wyrazy znajdujące się w tekście z wyrazami znajdującymi się w słowniku. Jeśli danego słowa nie znaleziono zostaje ono uznane za błędne. Aby dokonać korekcji błędnych wyrazów stosowane są algorytmy opisane w poprzednich rozdziałach. Dla optymalizacji działania aplikacji wyszukiwanie sugestii odbywa się w osobnych wątkach co znacząco przyspiesza działanie aplikacji.

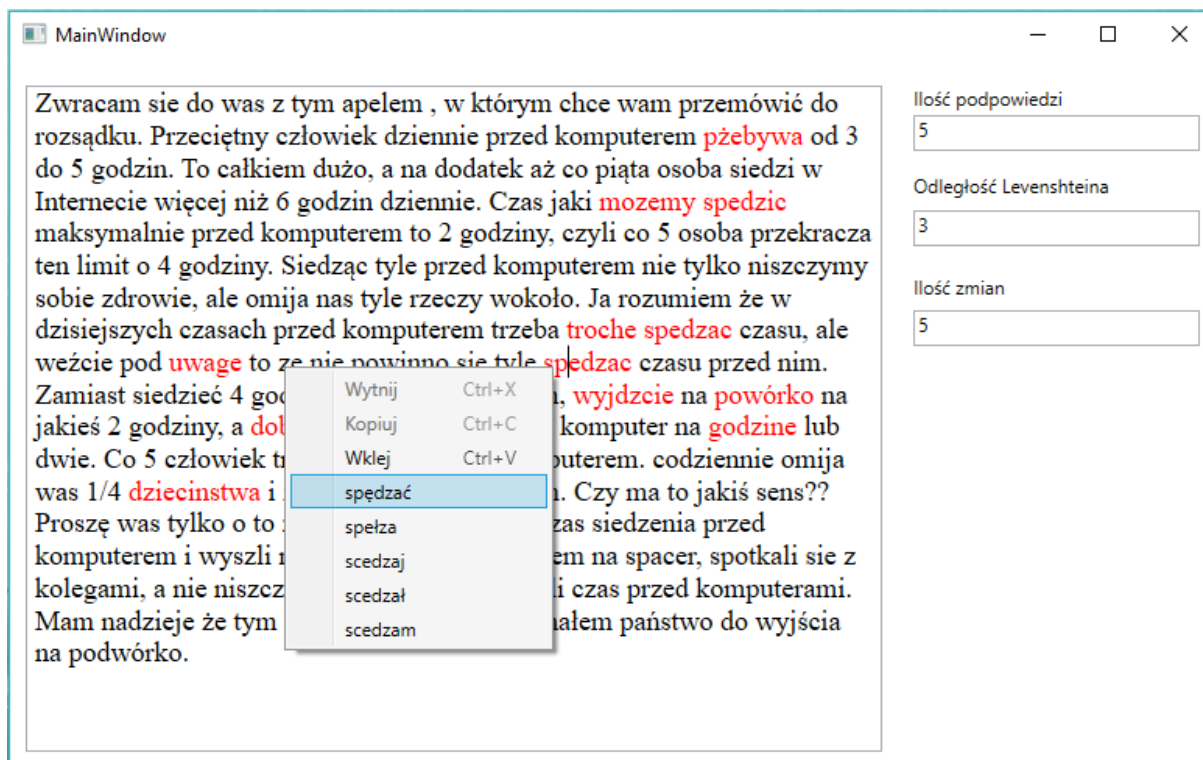
3.2 Interfejs aplikacji



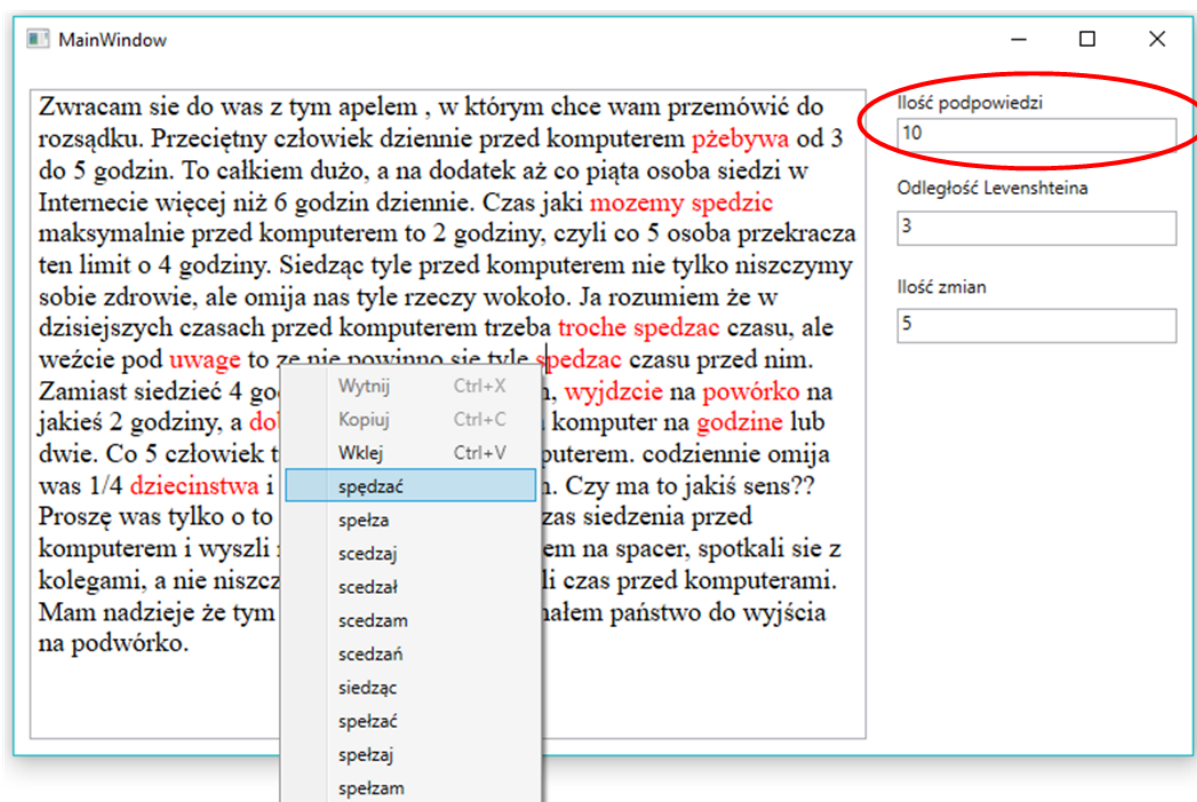
Rysunek 3.2 Interfejs aplikacji

1. Ilość podpowiedzi - pozwala na ustawienie maksymalnej ilości podpowiedzi jakie mają się pojawić po kliknięciu prawym przyciskiem myszy na błędnie napisane słowo.
2. Odległość Levenhteina - pozwala na ustawienie odległości jaka ma być brana pod uwagę w przypadku algorytmu Levenhstaina. Im większa liczba tym więcej podpowiedzi ale jednocześnie zmniejsza to szybkość działania aplikacji ze względu na dodatkowe obliczenia które muszą zostać wykonane.
3. Ilość zmian - pozwala na ustawienie parametru ilości zmian dla algorytmu podmieniającego znaki diakrytyczne. Im większa liczba tym więcej podpowiedzi oraz dłuższy czas wykonywania algorytmu.
4. Edytor tekstu - pozwala na edycję tekstu. Wyszukiwanie błędów działa w czasie rzeczywistym (funkcja sprawdzająca poprawność uruchamia się z każdym kliknięciem spacji). Jeśli dane słowo zostało uznane za błędne zostaje zaznaczone kolorem czerwonym. Aby je poprawić należy kliknąć na nie prawym przyciskiem myszki. Zostanie wyświetlone menu kontekstowe zawierające możliwe zamienniki. Aby podmienić słowo wystarczy kliknąć na zamiennik. Możliwe jest również poprawienie tekstu ręcznie wpisując w miejscu błędnego słowa poprawne.

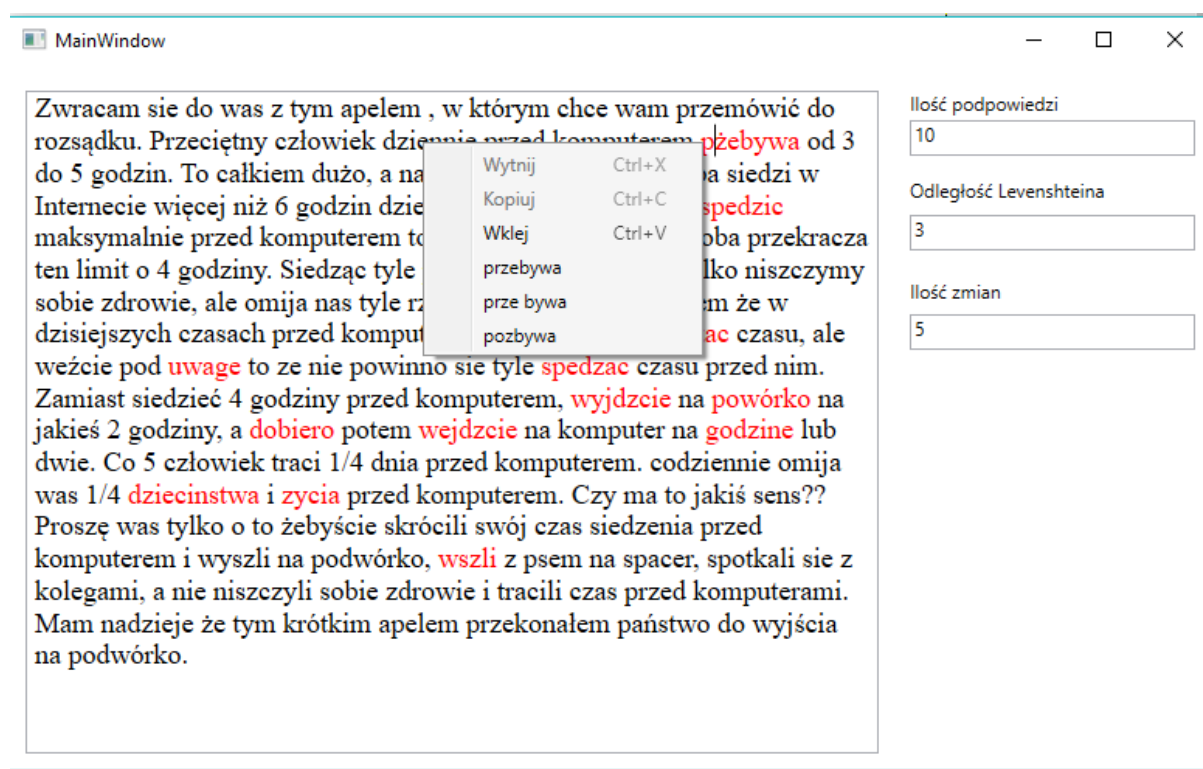
3.3 Wpływ parametrów na wyniki wyszukiwania



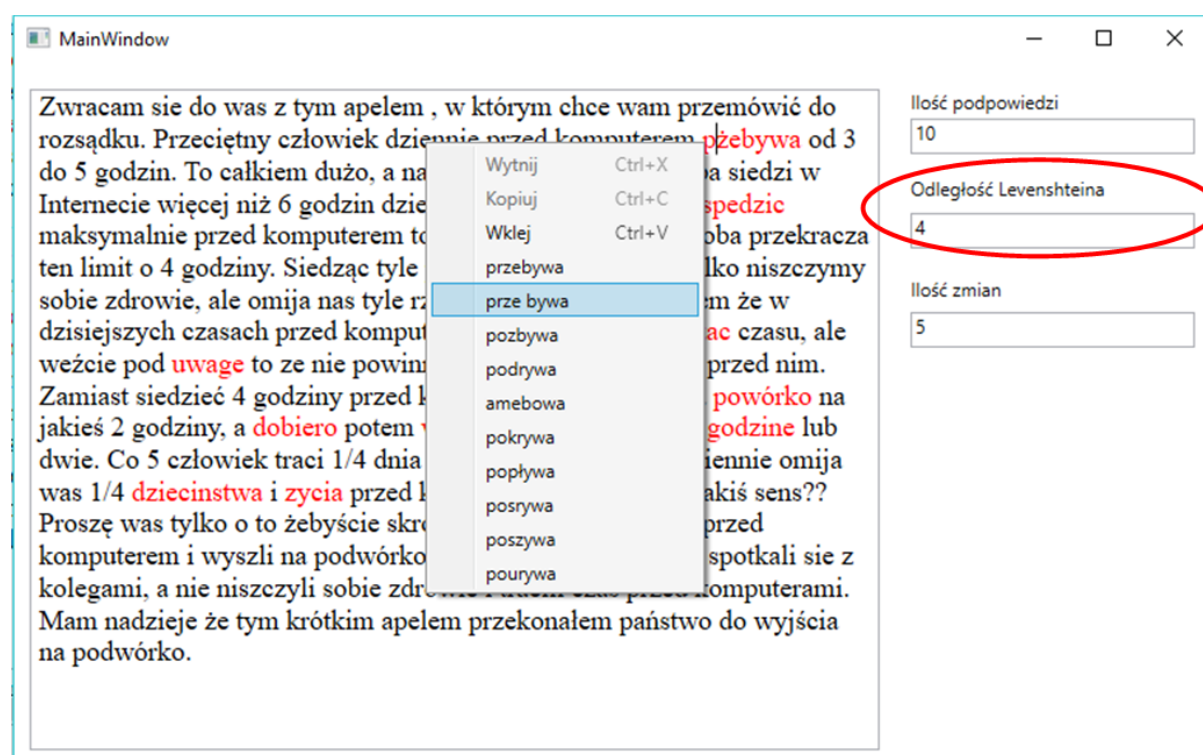
Rysunek 3.3 Przykładowy wynik wyszukiwania sugestii



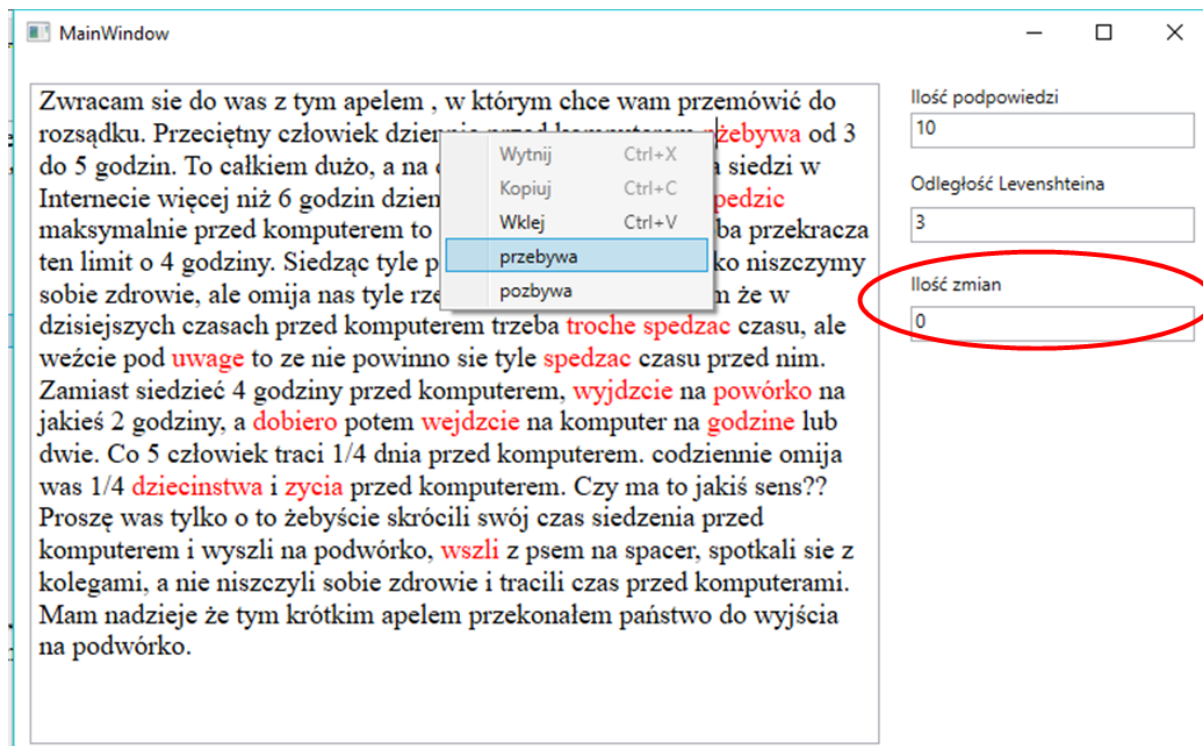
Rysunek 3.4 Zwiększenie ilości wyświetlanych sugestii



Rysunek 3.5 Przykładowy wynik wyszukiwania sugestii dla małej ilości podpowiedzi

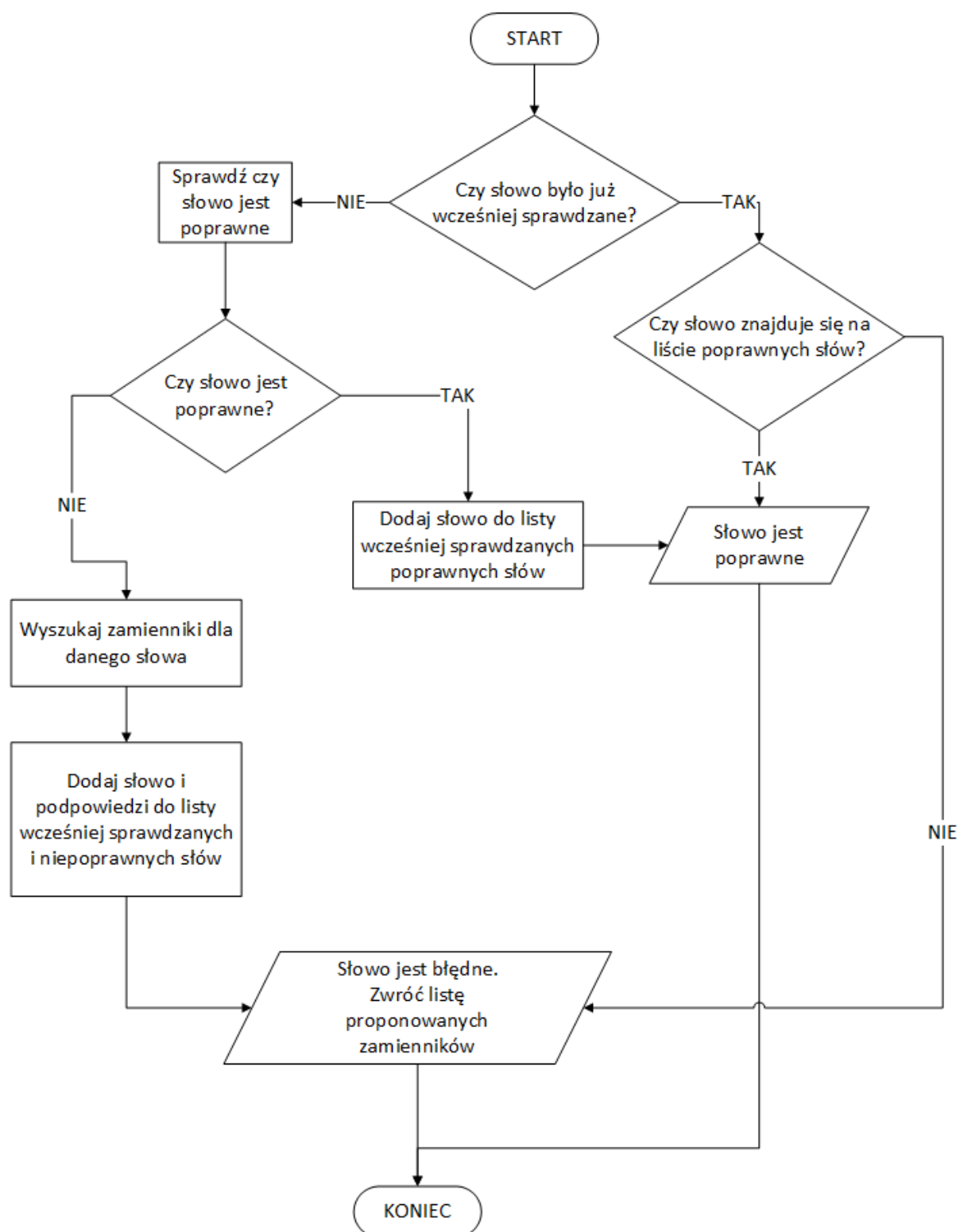


Rysunek 3.6 Zwiększenie odległości Levenhstaina



Rysunek 3.7 Zwiększenie ilości zmian

3.4 Realizacja wybranych funkcjonalności



Rysunek 3.8 Korekcja tekstu z poziomu interfejsu

Rozdział 4

Optymalizacja

Rozdział 5

Wdrożenie i testowanie aplikacji

Rozdział 6

Podsumowanie

```
1 | przykład
2 | listingu
3 | pamietaj zmieniac nazwy labelek!!! :D
```

Listing 6.1 Przykład listingu

Literatura

- [1] Anonim G. i in., *Kronika polska*, T.1: *Kronika i czyny książąt czyli władców polskich*, Gniezno 1115