Návrh

Analýza lingvistických dát

Pavol Freivolt

Tamara Savková

Júlia Gablíková

Michal Knor

FMFI UK

*TIS*

**Obsah**

[1 Úvod 2](#_Toc529124869)

[2 Rozdelenie na časti 2](#_Toc529124870)

[2.1 Popis modulov 2](#_Toc529124871)

[2.1.1 Frontend webovej aplikácie 2](#_Toc529124872)

[2.1.2 Backend webovej aplikácie 2](#_Toc529124873)

[2.1.3 Spúšťací a ukončovací modul 2](#_Toc529124874)

[2.1.4 Čítací modul 3](#_Toc529124875)

[2.1.5 Upravovací modul 4](#_Toc529124876)

[2.1.6 Slabikovací modul 6](#_Toc529124877)

[2.1.7 Počítací modul 7](#_Toc529124878)

[2.1.8 Výsledkový modul 8](#_Toc529124879)

[2.1.9 Zápis naslabikovaného textu 9](#_Toc529124880)

[2.1.10 Zápis do excelu 10](#_Toc529124881)

[2.2 Súčasne bežiace procesy 11](#_Toc529124882)

[2.2.1 Moduly a vlákna 11](#_Toc529124883)

[2.2.2 Komunikácia medzi vláknami 11](#_Toc529124884)

[2.3 Dáta a formáty súborov 11](#_Toc529124885)

[2.3.1 Funkčné súbory aplikácie 11](#_Toc529124886)

[2.3.2 Konštanty 11](#_Toc529124887)

[2.3.3 Trieda Word 12](#_Toc529124888)

[2.3.4 Konfiguračné súbory jazyka 12](#_Toc529124889)

[2.3.5 Analyzovaný text 12](#_Toc529124890)

[2.3.6 Výsledné súbory modulu Zápis naslabikovaného textu 12](#_Toc529124891)

[2.3.7 Výsledné súbory modulu Zápis do excelu 12](#_Toc529124892)

[3 Používateľské rozhranie (obrázky) 13](#_Toc529124893)

[4 UML diagramy 15](#_Toc529124894)

[5 Využité technológie 15](#_Toc529124895)

# Úvod

Tento dokument je návrhom architektúry systému vyvíjaného na analýzu lingvistických dát. Je určený primárne pre vývojový tím a obsahuje všetky informácie potrebné na implementáciu softvéru. Názvy konštánt a atribútov v tomto návrhu nie sú záväzné, slúžia predovšetkým na zlepšenie predstavy o funkcionalite. Systém bude implementovaný v jazyku Python a niektoré časti návrhu sa môžu odkazovať na jeho funkcie a triedy.

# Rozdelenie na časti

## Popis modulov

### Frontend webovej aplikácie

Frontend modul tvorí vizuálnu časť programu a slúži na jednoduchšiu manipuláciu s programom.

Dostáva dáta priamo od používateľa. Ten ich vkladá pomocou súboru, ktorý nahrá alebo priamo vloží text do textového poľa. Tento modul ďalej obsahuje dva „selekty“ :

„encoding“ , z ktorého sa dá vybrať kódovanie so zoznamu podporovaných kódovaní

„language“ , z ktorého sa dá vybrať jazyk so zoznamu jazykov

Modul tieto dáta posiela do „backendu“ pomocou formulára po odkliknutí tlačidla „Submit“.

Implementácia:

Modul je zložený z dvoch stránok:

1.    stránka kde sa text nahrá pomocou súboru

2.    stránka kde sa text zadá cez textové pole

Obe stránky obsahujú nadpis „Analýza lingvistických dát“, navigáciu, pomocou ktorej sa dá prechádzať medzi stránkami, „selekt“ kde sa dá vybrať jazyk a „Submit“ tlačidlo a pätičku stránky.

### Backend webovej aplikácie

Backend modul pracuje s dátami ktoré sú mu zaslané z frontednu pomocou formuláru cez metódu $\_POST.

Implementácia:

Ak bol vstup zadaný cez textové pole tak tento vstup zapíše do súboru s názvom temp\_n (n je prirodzené číslo, ktoré slúži na odlíšenie dvoch dočasných súborov).

Ďalej tento modul spustí spúšťací a ukončovací modul, ktoré vygenerujú výstupné súbory. Po ukončení procesu modul zabalí výstupné súbory (.zip), vynúti stiahnutie súborov a vymaže všetky dočasné súbory.

### Spúšťací a ukončovací modul

Spúšťací a ukončovací modul je určený na inicializovanie objektov, spustenie modulov na vláknach a ukončenie modulov.

Argumenty:

* file\_name  - názov súboru s textom na spracovanie
* encoding - typ kódovania súboru
* language - jazyk v ktorom je súbor napísaný

Implementácia:

Modul načíta konfiguračný súbor (podľa typu jazyka) ako objekt a inicializuje:

condition\_read\_clean

condition\_read\_syll

condition\_syll\_count

condition\_count\_text

queue\_read\_clean

queue\_clean\_syll

queue\_syll\_count

queue\_count\_text

Thread1 s targetom Upravovací modul a atribútmi: queue\_read\_clean, condition\_read\_clean, queue\_clean\_syll, condition\_read\_syll

Thread2 s targetom Slabikovací modul a atribútmi: queue\_clean\_syll, condition\_read\_syll, queue\_syll\_count, condition\_syll\_count

Thread3 s targetom Počítací modul a atribútmi: queue\_syll\_count, condition\_syll\_count, queue\_count\_text, condition\_count\_text, objekt s konfiguračnými dátami, file\_name

Thread4 s targetom Zapisovací modul a atribútmi: file\_name, queue\_count\_text, condition\_count\_text

Spustí Thread1, Thread2, Thread3 a Thread4.

Zavolá čítací modul a atribútmi: queue\_read\_clean, condition\_read\_clean, názov vstupného súboru, kódovanie vstupného textu, objekt s konfiguračnými dátami.

Na každom z vlákien zavolá funkciu join().

### Čítací modul

Čítací modul je určený na vytváranie zoznamu objektov Word na základe vstupných argumentov.

Argumenty:

* Queue (queue\_read\_clean)  - určeny na komunikáciu s iným modulom
* Condition (condition\_read\_clean) – riadenie prístupu ku queue
* Názov vstupného textu
* Kódovanie vstupného textu
* Objekt s konfiguračnými dátami.

Špeciálnymi označeniami sú cifry, ktoré označujú úlohu symbolu v slove:

* ak symbolom sú písmená daného jazyku, potom na základe objektu s konfiguračnými dátami s dátami sú priradené do fonetických skupín:
  + všetky konsonantné písmená okrem sylabických (0),
  + sonórne písmena (1),
  + samohlásky a sylabické konsonantné písmená (2),
  + špeciálne symboly jazyka, ktoré pri slabikovaní sa pridávajú do predchádzajúcej slabiky (3),
  + špeciálne symboly jazyka, ktoré pri slabikovaní sa pridávajú do nasledujúcej slabiky (4),
* interpunkčne znamienka (5),
* prenos (6).

Funkcie modulu:

* Modul rozdeľuje vstupný text na “slova” - jednotlivé oddelené prázdnym symbolov postupnosti symbolov zo vstupného textu.
* Ak “slovo” je posledné v texte, modul nastaví boolean označenie pre zodpovedajúci Word na True, inak - na False.
* Tiež modul nastavuje špeciálne označenia pre zodpovedajúci Word každého “slova”:
  + Prenos (6)  - označenie pomlčky, použitej pre rozdelenie “slova” na riadky, aby sa dalo ich spracovať;
  + Interpunkčne znamienka (7) - označenie ostatných pomlčiek a znakov *. , : ; ? ! [ ] ( ) { } ⟨ ⟩ — ― ‹ › « » “ ” " "*
* Po jednom pridáva objekt Word do dátovej štruktúry queue. // todo

Implementácia:

* Modul vytvorí pomocné premenné:
  + zoznam pre Word objekty, vytvárané počas spracovania,
  + pracovný buffer pre text “slova”,
  + pracovný buffer pre špeciálne označenia “slova”,
  + boolean premenná, ktorá označuje či načítané “slovo” je prenesene na nový riadok cez pomlčku.
* Ďalej modul otvorí vstupný súbor na základe jeho mena a kódovania a číta ho celý po jednom symbole.
* Ak načítaný symbol nie je medzera, novy riadok alebo tabulátorová zarážka, potom pomocná boolean premenná sa nastaví na False a každý symbol sa snaží pridať do pracovných bufferov:
  + Ak symbolom je pomlčka, potom sa ošetria prípady prenášania textu na novy riadok a napísania slov cez pomlčku. Symbol pomlčky a špeciálne označenie prenosu sa pridajú do pracovných bufferov. Pomocná boolean premenná sa nastaví na True. Pokus o priradenie symbolu do pracovných bufferov skončí.
  + Ak symbol patrí do ostatných interpunkčných znamienok, tak symbol a špeciálne označenie interpunkčných znamienok sa pridajú do pomocných bufferov. Pokus o priradenie symbolu do pracovných bufferov skončí.
  + Ak proces pokračuje, symbol a špeciálne označenie sa pridajú do pomocných bufferov. Pokus o priradenie symbolu do pracovných bufferov skončí.
* Ak ďalší načítaný symbol je medzera, novy riadok alebo tabulátorová zarážka, začne sa spracovanie údajov v pracovných bufferoch:
  + Ak pracovný buffer nie je prázdny, potom:
    - Ak pomocná boolean premenná je nastavená na True a ďalší načítaný symbol nie je medzera, potom v pracovnom bufferi pre špeciálne označenia nastavenie poslednej hodnoty sa nastaví na označenie pre prenos.
  + Ďalej sa vytvorí objekt Word na základe obsahu pracovných bufferov a pridá sa do zoznamu pre získané objekty Word.
* Keď sa načíta cely text:
  + Ak v pracovných bufferoch ešte sú nejaké dáta, vytvorí sa objekt Word na základe obsahu pracovných bufferov a pridá sa do zoznamu pre získané objekty Word.
  + Poslednému objektu Word v zozname sa nastaví na True atribút, ktorý znamená či slovo je posledné. Defaultné nastavenie tohto atribútu je True.
  + Proces spracovania skonči.
  + Modul pošle objekt Word a objekt s konfiguračnými dátami do dátovej štruktúry queue.

### Upravovací modul

Upravovací modul je určený na prípravu k slabikovaniu slova, reprezentovaného objektom Word, na základe získaných z queue konfiguračných dát.

Argumenty:

* Queue1 (queue\_read\_clean)  - obsahuje dvojicu objektov - pôvodný Word objekt
* Condition1 (condition\_read\_clean) – riadenie prístupu ku queue1 (queue\_read\_clean)
* Queue2 (queue\_clean\_syll) – obsahuje dvojicu objektov - upravený Word objekt
* Condition2 (condition\_clean\_syll) – riadenie prístupu ku queue2 (queue\_clean\_syll)
* Objekt s konfiguračnými dátami

Funkcie modulu:

* Získava z queue objekty Word a postupne vytvára z nich trojice (predchádzajúci, aktuálny a nasledujúci Word), spracováva trojicu, a aktualizuje trojicu po získaní ďalšieho slova (“predchádzajúci” dostáva hodnotu “aktuálneho”, “aktuálny” - “nasledujúceho”, a “nasledujúci” - nového slova”).
* Ak podľa objektu s konfiguračnými dátami slová, ktoré neobsahujú slabiky, sa spájajú s predchádzajúcim alebo nasledujúcim slovom, modul vykoná tieto operácie.
* Modul bude ignorovať slová, ktoré v strede obsahujú znaky, ktoré nepatria do jazyka.
* Modul bude ignorovať interpunkčné znaky, ktoré sú na začiatku alebo konci slova.
* Modul nastaví špeciálne označenia pre písmená textu slova, aby zodpovedali fonetickému typu:
  + 0 pre konsonantné písmena,
  + 1 pre sonorné písmena,
  + 2 pre samohlásky,
  + 3 pre špeciálne písmená, ktoré sa spájajú s predchádzajúcou slabikou,
  + 4 pre špeciálne písmená, ktoré sa spájajú s nasledujúcou slabikou.
* Výsledný objekt Word bude obsahovať v texte slova len povolene písmená (podľa objektu s konfiguračnými dátami) a len špeciálne označenia pre fonetický typ písmena.
* Ak podľa objektu s konfiguračnými dátami niektoré písmená zmenia svoj fonetickej typ, potom modul vykoná tieto zmeny.

Implementácia:

* Modul získava z queue objekty Word a postupne vytvára z nich trojice (predchádzajúci, aktuálny a nasledujúci Word).
* Spracováva trojicu, ak jej druhý element je objektom Word:
  + Nastaví pomocné premenne:
    - Pracovný buffer pre text slova,
    - Pracovný buffer pre špeciálne označenia slova,
    - Pomocná mapa, ktorá je určená na vykonanie zmeň fonetického typu písmen slova. Ukladá informácie o tom, či také zmeny treba bude robiť na konci spracovania objektu Word a pole s indexmi, ktoré označujú pre aké písmeno treba bude robiť zmeny.
  + Prechádza každým symbolom a špeciálnym označením objektu Word.
    - Ak buffer pre text slova nie je prázdny, špeciálne označenie  je ‘interpunkčné znamienko’ a zároveň tento symbol je *[ ( { ⟨ ‹ « ‘ “* alebo *'"*, potom tento symbol sa ignoruje a nepridáva do pracovných bufferov. Načíta ďalší symbol a špeciálne označenie.
    - Ak špeciálne označenie je ‘prenos’ a ďalší symbol je písmeno, potom tento symbol sa ignoruje a nepridáva do pracovných bufferov. Načíta ďalší symbol a špeciálne označenie.
    - Ak špeciálne označenie  je ‘interpunkčné znamienko’ a symbol nie je žiaden z - —, potom sa skontroluje či ďalšie symboly až po koniec slova, ci ďalšie špeciálne označenia sú tiež ‘interpunkčné znamienka’. Ak nie, vráti trojicu: “predchádzajúci” objekt, null objekt, “nasledujúci” objekt. Ak áno, vráti trojicu: “predchádzajúci” objekt, upravený objekt Word (text a označenia slova sú zmenené podľa obsahu pracovných bufferov), “nasledujúci” objekt
    - Ďalej symbol sa zmení na malý (lowercase)
    - Prebehne kontrola, ci symbol mení svoj foneticky typ (podľa objektu s konfiguračnými dátami). Ak áno, pomocná mapa uloží index tohto slova a informáciu o tom, že tieto zmeny treba bude vykonať.
    - Prebehne kontrola, ci symbol patrí do malých písmen jazyka (podľa objektu s konfiguračnými dátami). Ak áno, malý symbol sa pridá do pracovných bufferov. Tiež sa zisti fonetická reprezentácia písmena (podľa objektu s konfiguračnými dátami), a priradí sa do pracovných bufferov pre zvuky. Ak nie, vráti trojicu: “predchádzajúci” objekt, null objekt, “nasledujúci” objekt.
  + Na záver, ak dĺžka textu “aktívneho” objektu Word je 1 a tento text netvorí slabiky (podľa objektu s konfiguračnými dátami), potom:
    - Ak objekt sa pridáva k predchádzajúcemu objektu (podľa objektu s konfiguračnými dátami), potom text a špeciálne označenia “aktuálneho” objektu sa pridávajú na koniec textu a špeciálnych označení “predchádzajúceho” objektu.
    - Ak objekt sa pridáva k nasledujúcemu objektu (podľa objektu s konfiguračnými dátami), potom text a špeciálne označenia “aktuálneho” objektu sa pridávajú na začiatok textu a špeciálnych označení “nasledujúceho” objektu.
    - Vráti trojicu objektov: “predchádzajúci” objekt, null objekt, “nasledujúci” objekt.
  + Vráti trojicu objektov: “predchádzajúci” objekt, upravený objekt Word (text a označenia slova sú zmenené podľa obsahu pracovných bufferov), “nasledujúci” objekt
* Aktualizuje trojicu objektov po získaní ďalšieho slova (“predchádzajúci” dostáva hodnotu “aktuálneho”, “aktuálny” - “nasledujúceho”, a “nasledujúci” - nového slova”) a začína spracovávať ďalšiu trojicu.
* Ak z queue bol získaný objekt Word s nastaveným na True označením, že je posledné, potom:
  + Trojica objektov sa aktualizuje posledný krát,
  + Vykoná sa  posledné spracovanie trojice objektov Word,
  + Spracovaný objekt Word sa pridá do queue.

### Slabikovací modul

Slabikovací modul je určený na rozdelenie slov na slabiky.

Argumenty:

1. queue\_clean\_syll - obsahuje objekty typu Word
2. condition\_clean\_syll – obmedzuje prístup ku queue\_clean\_syll
3. queue\_syll\_count - obsahuje objekty typu tuple(pole stringov, pole polí integerov, Word.end)
4. condition\_syll\_count - obmedzuje prístup ku queue\_syll\_count

Modul vyberie z queue\_clean\_syll za použitia condition\_clean\_syll objekt typu Word. Slovo uložené v obdržanom objekte pod atribútom text bude rozdelené na slabiky vzhľadom na atribút signs, ktorý definuje sonoritu jednotlivých písmen. Následne budú slabiky tohto slova uložené do poľa, ktoré bude vložené do queue\_syll\_count za použitia condition\_syll\_count spolu s poľom obsahujúcim Word.signs rozdelené podľa slabík a Word.end.

Postup

Inicializácia:

- modul si najprv do premennej run vloží hodnotu True

- následne modul spustí cyklus podmienený hodnotou premennej run (while run), v ktorom beží zvyšok programu

Získanie objektu Word z queue\_clean\_syll :

- modul sa najprv pokúsi získať condition\_clean\_syll

- po získaní condition\_clean\_syll sa modul pokúsi vybrať z queue\_clean\_syll prvok. Ak sa mu to nepodarí, čaká (condition.wait()). Ak sa mu to podarí, uvoľní condition\_clean\_syll a pokračuje

Rozdelenie slova Word.text na slabiky (1. iterácia):

- modul si vytvorí premennú A, ktorá bude poľom polí integerov

- modul si vytvorí premennú B – pole integerov

- postupne prechádza pole Word.signs a vkladá tieto prvky do B

- ak modul narazil pri prechádzaní na konštantu s hodnotou 2, vloží ju do B, celé pole B vloží do A, pole B vyprázdni a pokračuje

- po prečítaní celého Word.signs pokračuje 2. iteráciou

Príklad: Word.signs = [2,1,0,2,1], z toho A = [[2],[1,0,2],[1]]

Rozdelenie slova Word.text na slabiky (2. iterácia):

- modul prechádza prvky poľa A počnúc A[1] (budeme tieto prvky (polia integerov) označovať písmenom A[i])

- ak nastane situácia, že nejaký prvok v A[i] má vyššiu hodnotu ako prvok, ktorý za ním nasleduje, tento prvok a všetky pred ním sú vybraté z A[i] a vložené na koniec poľa A[i-1]

- ak má prvok A[i][j] hodnotu konštanty SPEC\_PREC alebo SPEC\_FOLL, teda je to špeciálny znak, neporovnáva sa, namiesto toho sa vykoná operácia uvedená nižšie a prvok A[i][j-1] sa porovnáva s prvkom A[i][j+1] (teda A[i][j] sa v porovnávaní preskočil)

- ak prechádzame posledný prvok poľa A a tento prvok A[i] obsahuje len prvky s hodnotou 0, A[i] zrušíme a všetky jeho prvky presunieme do A[i-1]

Príklad: A = [[2],[1,0,2],[1]], z toho A = [[2,1], [0,2,1]]

Práca so špeciálnymi znakmi SPEC\_PREC a SPEC\_FOLL:

- ak má znak hodnotu SPEC\_PREC, skontroluje sa, či sa pred ním v slabike niečo nachádza, ak nie, tento znak sa vyberie zo slabiky a pripojí sa na koniec predchádzajúcej slabiky (vloží sa na koniec predchádzajúce poľa)

- ak má znak hodnotu SPEC\_FOLL, netreba robiť nič, pretože týmto znakom určite nekončí slabika (SPEC\_SUC určite nemá hodnotu 2)

Rozdelenie slova Word.text na slabiky (3. iterácia):

- modul vytvorí výsledné pole D, ktoré bude obsahovať stringy (slabiky)

- podľa dĺžok prvkov poľa A rozkúskuje slovo vo Word.text a tieto substringy vloží do poľa D

Príklad: Word.text = “intel”, A = [[2,1],[0,2,1]], teda A[0].length = 2, A[1].length = 3, z toho D = [“in”, “tel”]

Vloženie slova do queue\_syll\_count:

- modul sa najprv pokúsi získať condition\_syll\_count

- po získaní condition\_syll\_count modul vloží do queue\_syll\_count tuple(D, A, Word.end)

- modul urobí condition\_syll\_count.notify()

- modul uvoľní condition\_syll\_count

Ukončenie behu modulu:

- ak modul vo fáze získania objektu Word z queue\_clean\_syll zistí, že Word.end je True, nastaví premennú run na False, a teda po spracovaní tohto slova ukončí svoj beh

### Počítací modul

Počítací modul je určený na zistenie dĺžok slabík a počítanie ich výskytov.

Argumenty:

1. queue\_syll\_count – obsahuje objekty typu tuple(pole stringov, pole polí integerov, boolean)
2. condition\_syll\_count – obmedzuje prístup ku queue\_syll\_count
3. queue\_count\_txt – obsahuje objekty typu tuple(pole stringov, pole integerov, boolean)
4. condition\_count\_txt – obmedzuje prístup ku queue\_count\_txt
5. data – pravidlá pre jazyk textu
6. file\_name – názov analyzovaného súboru

Modul vyberie z queue\_syll\_count pole a každú slabiku samostatne spracuje – zistí a zaznamená jej dĺžku a zaráta jej výskyt.

Postup

Inicializácia:

- modul si vytvorí premenné *map\_len* (obsahuje prvky typu string: integer, kde string je slabika a integer je jej dĺžka) a *map\_freq* (obsahuje prvky typu string: integer, kde string je slabika a integer jej počet výskytov)

- modul si do premennej *run* vloží hodnotu True

- následne modul spustí cyklus podmienený hodnotou premennej *run* (while *run*), v ktorom beží zvyšok programu (okrem zavolania nasledujúceho modulu)

Získanie objektu z queue\_syll\_count:

- modul sa najprv pokúsi získať condition\_syll\_count

- po získaní condition\_syll\_count sa modul pokúsi vybrať z queue\_syll\_count prvok *p*. Ak sa mu to nepodarí, čaká (condition.wait()). Ak sa mu to podarí, uvoľní condition\_syll\_count a pokračuje

Spracovanie slabík

- modul prechádza slabiky poľa *p[0]*

- modul si vytvorí pomocné pole A

- zistí, či sa už slabika nachádza v map\_freq, ak áno, zvýši hodnotu pod týmto kľúčom o 1, ak nie, pridá ju do map\_freq s hodnotou 1

- ak sa slabika nenachádzala v map\_freq, nenachádza sa ani v map\_len a teda modul musí zistiť jej dĺžku podľa postupu nižšie

- modul pridá do poľa A dĺžku spracovávanej slabiky, ktorú zistí z map\_len

- po prečítaní poslednej slabiky v *p[0]* vloží modul pole A do queue\_count\_txt podľa postupu nižšie

Zistenie a evidencia dĺžky slabiky

- modul prechádza znaky (písmená) v slabike

- modul si vytvorí premennú *syl\_len*, ktorej počiatočná hodnota je 0

- modul zo štruktúry data zistí dĺžku daného znaku vzhľadom na okolité znaky a túto dĺžku pripočíta do *syl\_len*

- modul pridá do map\_len slabiku (kľúč) s jej dĺžkou v premennej *syl\_len* (hodnota)

Vloženie prvku do queue\_count\_txt:

- modul sa najprv pokúsi získať condition\_count\_txt

- po získaní condition\_count\_txt modul vloží do queue\_count\_txt tuple(*p[0]*, A, *p[2]*)

- modul urobí condition\_count\_txt.notify()

- modul uvoľní condition\_count\_txt

Ukončenie práce

- ak modul vo fáze získania prvku *p* z queue\_syll\_count zistí, že *p[2]* je True, nastaví premennú run na False, a teda po spracovaní tohto slova skončí cyklus

- po ukončení cyklu modul zavolá Výsledkový modul s argumentami map\_len, map\_freq, file\_name

### Výsledkový modul

Výsledkový modul vytvára dve nové mapy za pomoci map\_len a map\_freq.

Argumenty:

1. map\_len – obsahuje prvky typu string: integer, kde string je slabika a integer je jej dĺžka
2. map\_freq – obsahuje prvky typu string: integer, kde string je slabika a integer jej počet výskytov
3. file\_name – názov analyzovaného súboru

Modul vytvorí map\_with\_rep a map\_wout\_rep za pomoci map\_len a map\_freq, kde kľúčom v map\_with\_rep je dĺžka slabiky a hodnotou je počet unikátnych výskytov danej dĺžky slabiky, zatiaľ čo v mape3 je kľúčom dĺžka slabiky a hodnotou počet všetkých výskytov danej dĺžky slabiky.

Postup

Inicializácia:

* modul je volaný modulom ‘Počítací modul’

Vytvorenie premennej map\_with\_rep:

* modul vytvorí map\_with\_rep
* modul prejde všetky hodnoty (dĺžka slabiky) v map\_len
* ak hodnota ešte nie je kľúčom v map\_with\_rep, tak ju vytvorí kľúč (dĺžka slabiky) a hodnotu nastaví na 1, inak hodnotu zvýši o 1

Vytvorenie premennej map\_wout\_rep:

* modul vytvorí map\_wout\_rep
* modul prejde postupne všetky prvky v map\_freq
* modul za pomoci map\_len zistí dĺžku daného prvku v map\_freq
* ak sa kľúč s danou dĺžkou v map\_wout\_rep nenachádza, tak vytvorí kľúč s danou dĺžkou, ktorého hodnota je hodnota daného prvku z map\_freq
* ak sa kľúč s danou dĺžkou v map\_wout\_rep už nachádza, tak k nemu pripočíta hodnotu z map\_freq

Ukončenie práce

- ak modul zapíše posledný prvok do map\_wout\_rep, tak zavolá modul ‘Zápis do excelu’ s argumentami: file\_name, map\_len, map\_freq, map\_with\_rep a map\_wout\_rep

### Zápis naslabikovaného textu

Modul ‘Zápis naslabikovaného textu‘ slúži na zapisovanie naslabikovaného textu do súboru.

Argumenty:

1. file\_name
2. queue\_count\_txt – obsahuje objekty typu tuple (pole stringov, pole integerov, Word.end)
3. condition\_count\_txt – obmedzuje prístup ku queue\_count\_txt

Modul vytvorí priečinok file\_name a v ňom súbory ‘syllable\_text.txt’ a ‘syllable\_number\_text.txt’. Následne vyberie z queue\_count\_txt za použitia condition\_count\_txt všetky prvky z pola stringov a pridá ich do lokálneho poľa A a všetky prvky z pola integerov do poľa B. Ak pole A dosiahne dĺžku ‘n’, tak otvorí súbor ‘syllable\_text.txt’ a uloží tam obsah pola A. Ak pole B dosiahne dĺžku ‘n’, tak otvorí súbor ‘syllable\_number\_text.txt’, kam uloží obsah pola B.

Postup

Inicializácia:

- modul si najprv do premennej run vloží hodnotu True

- následne modul spustí cyklus podmienený hodnotou premennej run (while run), v ktorom beží zvyšok program

Vytvorenie súborov:

* vytvorí sa priečinok file\_name
* v priečinku sa vytvoria súbory ‘syllable\_text.txt’ a ‘syllable\_number\_text.txt’

Získanie prvkov z queue\_count\_txt:

* modul sa najprv pokúsi získať condition\_count\_txt
* po získaní condition\_count\_txt sa modul pokúsi vybrať z queue\_count\_txt prvok. Ak sa mu to nepodarí, čaká (condition.wait()). Ak sa mu to podarí, uvoľní condition\_count\_txt a pokračuje

Zapísanie naslabikovaného textu do súboru:

* modul si vytvorí pomocné pole A
* modul ukladá prvky z pola stringov z queue\_count\_txt do poľa A
* ak pole A presiahne dĺžku ‘n’, tak obsah poľa A zapíše do súboru ‘syllable\_text.txt’. pričom slabiky jednotlivých slov sú oddelené pomlčkou

Zapísanie naslabikovaného textu do súboru vo forme čísel:

* modul si vytvorí pomocné pole B
* modul ukladá prvky z pola integerov z queue\_count\_txt do poľa B
* ak pole B presiahne dĺžku ‘n’, tak obsah pola B zapíše do súboru ‘syllable\_number\_text.txt’, kde slabiky sú nahradené ich dĺžkou a oddelené pomlčkou

Ukončenie behu modulu:

- ak modul vo fáze získania prvkov z premennej queue\_count\_txt zistí, že Word.end je True, nastaví premennú run na False, a teda po spracovaní tohto slova ukončí svoj beh

### Zápis do excelu

Modul má na starosti vytvorenie troch tabuľkových súborov .xls.

Argumenty:

1.    file\_name – názov analyzovaného súboru

2.    map\_len – obsahuje prvky typu string: integer, kde string je slabika a integer je jej dĺžka

3.    map\_freq – obsahuje prvky typu string: integer, kde string je slabika a integer jej počet výskytov

4.    map\_with\_rep – obsahuje prvky typu integer1: integer2, kde integer1 je dĺžka slabiky a integer2 je počet unikátnych slabík majúcich takúto dĺžku

5.    map\_wout\_rep – obsahuje prvky typu integer1: integer2, kde integer1 je dĺžka slabiky a integer2 je počet všetkých slabík majúcich takúto dĺžku

Modul vytvorí v priečinku file\_name súbory:

- ‘syllables\_multiplicity.xls‘ - tabuľka s početnosťou a dĺžkou slabík

- ‘number\_of\_length\_of\_syllables\_with\_repetition.xls‘ – frekvenčná tabuľka dĺžok slabík berúc do úvahy všetky výskyty slabík

- ‘number\_of\_length\_of\_syllables\_without\_repetition.xls’ - frekvenčná tabuľka dĺžok slabík berúc do úvahy výskyt slabík v texte iba raz

Postup

Inicializácia:

- modul je volaný modulom ‘Výsledkový modul’

Vytvorenie tabuľky s početnosťou a dĺžkou slabík:

-       modul vytvorí pomocné pole A, ktorého prvkami budú polia o veľkosti 3

-       modul prejde všetky hodnoty (početnosť) v map\_freq zostupne

-       zistí aký je kľúč (slabika) danej hodnoty

-       na základe kľúča zistí jeho dĺžku (dĺžka slabiky) z map\_len

-       zistené údaje zapíše do poľa A, kde prvkami poľa sú polia formátu [slabika, početnosť, dĺžka slabiky]

-       vytvorí súbor ‘syllables\_multiplicity.xls‘ v priečinku file\_name

-       zapíše do tabuľky prvky poľa A, ktorej stĺpce sú: slabika, početnosť, dĺžka slabiky

Vytvorenie tabuľky frekvenčnej tabuľky dĺžok slabík berúc do úvahy všetky výskyty slabík:

-       modul vytvorí pomocné pole A, ktorého prvkami budú polia o veľkosti 3

-       modul prejde všetky hodnoty (početnosť) v map\_wout\_rep a ich súčet zapíše do pomocnej premennej S

-       prejde sa každý kľúč  (dĺžka slabiky) map\_wout\_rep

-       pre každý kľúč sa zistí jeho početnosť

-       vypočíta sa jeho percentuálny výskyt ako početnosť/S\*100

-       zistené údaje zapíše do poľa A, kde prvkami poľa sú polia formátu [dĺžka slabiky, početnosť, percentuálny výskyt]

-       vytvorí súbor ‘number\_of\_length\_of\_syllables\_with\_repetition.xls‘ v priečinku file\_name

-       zapíše do tabuľky prvky poľa, ktorej stĺpce sú: dĺžka slabiky, početnosť, percentuálny výskyt

Vytvorenie tabuľky frekvenčnej tabuľky dĺžok slabík berúc do úvahy výskyt slabík v texte iba raz:

-       modul vytvorí pomocné pole A, ktorého prvkami budú polia o veľkosti 3

-       modul prejde všetky hodnoty (početnosť) v map\_with\_rep a ich súčet zapíše do pomocnej premennej S

-       prejde sa každý kľúč  (dĺžka slabiky) map\_with\_rep

-       pre každý kľúč sa zistí jeho početnosť

-       vypočíta sa jeho percentuálny výskyt ako početnosť/S\*100

-       zistené údaje zapíše do poľa A, kde prvkami poľa sú polia formátu [dĺžka slabiky, početnosť, percentuálny výskyt]

-       vytvorí súbor ‘number\_of\_length\_of\_syllables\_without\_repetition.xls‘ v priečinku file\_name

-       zapíše do tabuľky prvky poľa A, ktorej stĺpce sú: dĺžka slabiky, početnosť, percentuálny výskyt

## Súčasne bežiace procesy

Niektoré moduly bežia v samostatných vláknach, a teda aplikácia môže spracovávať viacero slov v rôznych úrovniach naraz. Zoznam vlákien a modulov, ktoré sú v týchto vláknach spúšťané, je uvedený v časti Moduly a vlákna. Znakom „->“ je vyjadrená následnosť, teda postupné spúšťanie modulov v týchto vláknach. Okrem Main threadu sú všetky vlákna robené podľa architektúry Pipes and filters a to v rámci seba, aj medzi sebou. Posúvanie dát medzi vláknami je opísané v časti Komunikácia medzi vláknami.

### Moduly a vlákna

* ***Main thread***: Spúšťací a ukončovací modul -> Čítací modul (-> Spúšťací a ukončovací modul)
* ***Thread1***: Upravovací modul
* ***Thread2***: Slabikovací modul
* ***Thread3***: Počítací modul -> Výsledkový modul -> Zápis do excelu
* ***Thread4***: Zápis naslabikovaného textu

### Komunikácia medzi vláknami

* ***Spúšťací a ukončovací modul***: Od modulu Backend webovej aplikácie získava dáta, na základe ktorých vytvára všetky štruktúry potrebné na komunikáciu, spúšťa vlákna a volá Čítací modul. Štruktúry a potrebné dáta dáva modulom k dispozícii pri ich spustení/zavolaní a viac s nimi nemanipuluje. Čaká, kým všetky vlákna skončia, aby mohol ukončiť aj svoj beh.
* ***Main thread – Thread1***: Čítací modul komunikuje s Upravovacím modulom pomocou queue\_read\_clean, ku ktorému je prístup riadený pomocou condition\_read\_clean.
* ***Thread1 – Thread2***: Upravovací modul komunikuje so Slabikovacím modulom pomocou queue\_clean\_syll, ku ktorému je prístup riadený pomocou condition\_clean\_syll.
* ***Thread2 – Thread3***: Slabikovací modul komunikuje s Počítacím modulom pomocou queue\_syll\_count, ku ktorému je prístup riadený pomocou condition\_syll\_count.
* ***Thread3 – Thread4***: Počítací modul komunikuje s modulom Zápis naslabikovaného textu pomocou queue\_count\_txt, ku ktorému je prístup riadený pomocou condition\_count\_txt.

## Dáta a formáty súborov

### Funkčné súbory aplikácie

Aplikácia bude implementovaná v jazyku Python, a teda všetky jej fuknčné obsahové súčasti musia byť vo formáte **py**.

### Konštanty

Súbor vo formáte **py** obsahuje nasledovné konštanty potrebné pre prácu modulov a ich hodnoty (za znakom „##“ sú vysvetlivky):

* CONS = 0 ## consonant
* SONOR = 1 ## sonorant
* VOWEL = 2 ## vowel
* SPEC\_PREC = 3 ## special sign attached to precedent character
* SPEC\_FOLL = 4 ## special sign attached to following character
* PUNCT = 5 ## punctiation
* CARRY = 6 ## carry

### Trieda Word

Súbor vo formáte **py** s definíciou triedy Word, ktorá reprezentuje slovo textu. Táto trieda má atribúty:

* text – text slova ako postupnosť symbolov neobsahujúca medzeru
* signs – pole konštánt z časti 2.3.2, ktoré každému znaku z Word.text priraďuje označenie
* end – hodnota typu boolean, ktorá hovorí o tom, či je toto slovo posledným slovom textu

### Konfiguračné súbory jazyka

Súbory, v ktorých sú uložené pravidlá pre prácu s jazykom analyzovaného textu, budú vo formáte **json**. Popis presnej štruktúry súborov bude priložený neskôr.

### Analyzovaný text

Vstupný text musí byť vo formáte **txt** a v jednom z kódovaní: **ASCII**, **Windows-1250**, **Windows-1251**, **UTF-8**, **UTF-16**, **UTF-32**. Jazyk súboru musí byť podporovaný.

### Výsledné súbory modulu Zápis naslabikovaného textu

Súbory budú vo formáte **txt** a v kódovaní **UTF-8**.

***syllable\_text.txt***: Bude obsahovať iba znaky abecedy uvedené v konfiguračnom súbore jazyka textu, medzery a pomlčky, pričom medzi pomlčkami a medzerami musí byť vždy minimálne jeden znak abecedy.

***syllable\_number\_text.txt***: Bude obsahovať iba číslice oddelené medzerami a pomlčkami.

### Výsledné súbory modulu Zápis do excelu

Súbory budú vo formáte **xls**.

***syllables\_multiplicity.xls***: Tabuľka, v ktorej prvý stĺpec obsahuje text slabiky, druhý stĺpec obsahuje dĺžku slabiky a tretí stĺpec obsahuje počet výskytov slabiky v texte. Slabiky sú usporiadané podľa početnosti zostupne.

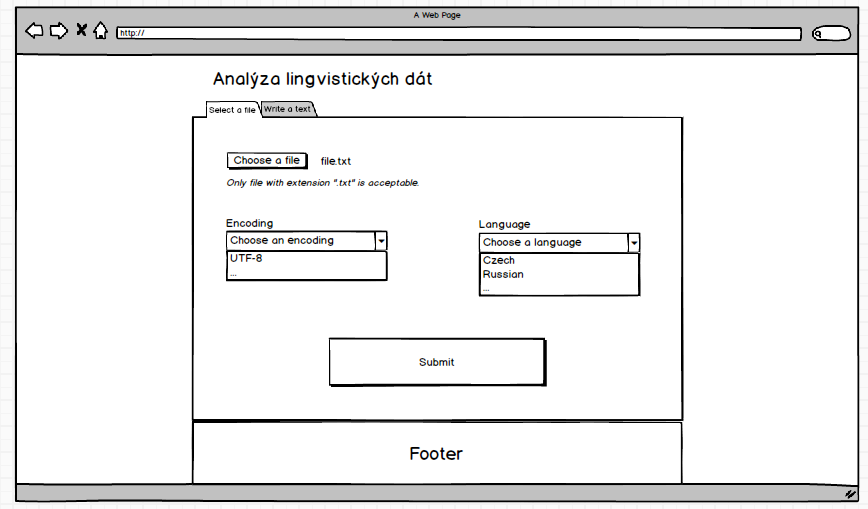
***number\_of\_length\_of\_syllables\_with\_repetition.xls****:* Tabuľka, v ktorej prvý stĺpec je dĺžka slabík a druhý stĺpec je počet slabík tejto dĺžky s opakovaním. Dĺžky sú usporiadané vzostupne.

***number\_of\_length\_of\_syllables\_without\_repetition.xls****:* Tabuľka, v ktorej prvý stĺpec je dĺžka slabík a druhý stĺpec je počet slabík tejto dĺžky s opakovaním. Dĺžky sú usporiadané vzostupne.

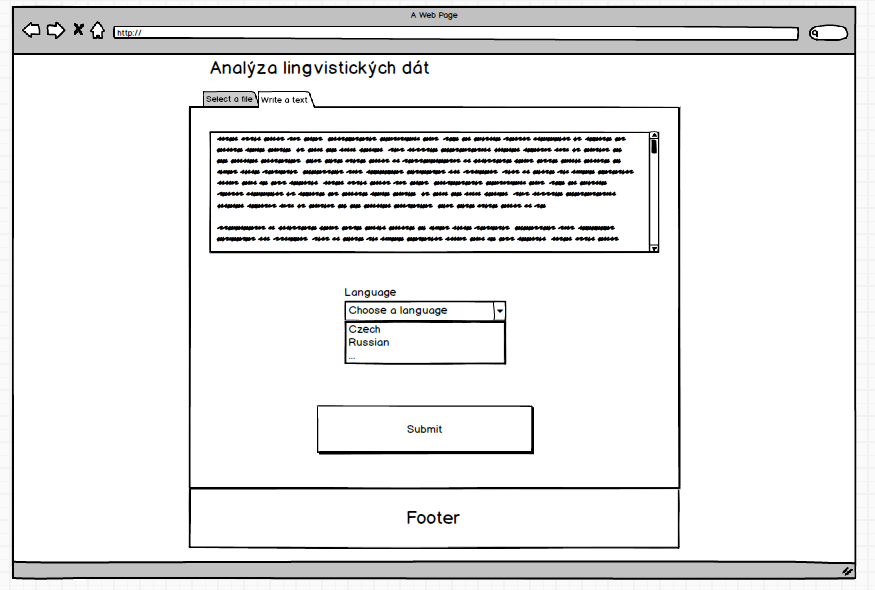
# Používateľské rozhranie (obrázky)



Obr. 1: Mobilné rozhranie



Obr. 2: Počítačové rozhranie - karta1



Obr. 3: Počítačové rozhranie - karta2

# UML diagramy

Budú pridané neskôr.

# Využité technológie

* **CSS3** – umožňuje rozmiestnenie html objektov po stránke a úpravu celkového vizuálu stránky
* **HTML5** – umožňuje vytvoriť základnú kostru stránky
* **PHP** – umožňuje spracovať údaje z formuláru a spustiť py skript
* **JSON** – umožňuje, aby načítanie konfiguračných súborov vytvorilo objekt, s ktorým aplikácia dokáže pracovať
* **XLWT** – umožňuje zápis do excelových súborov