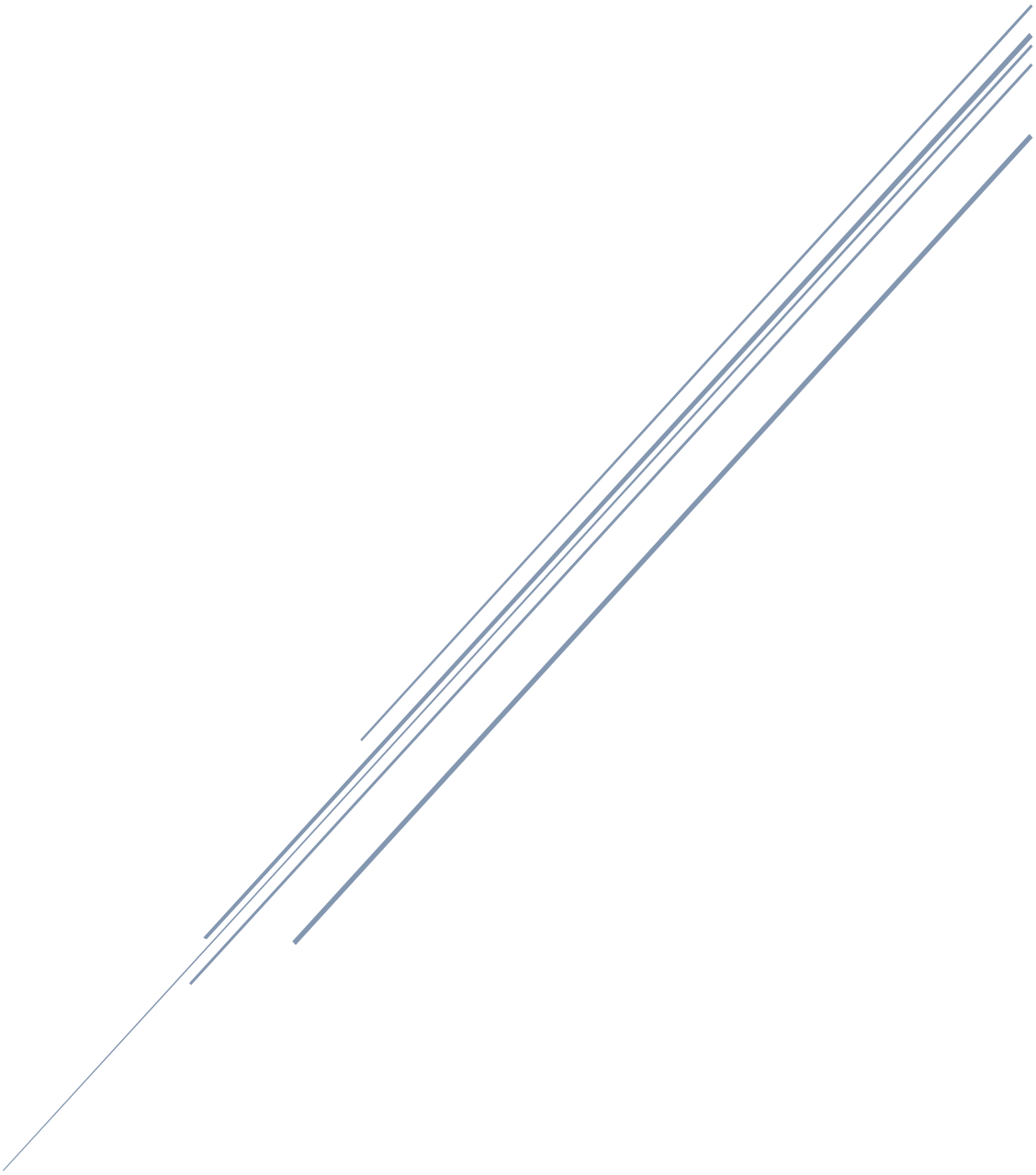
TECHNICKÁ DOKUMENTÁCIA

Analýza lingvistických dát

Pavol Freivolt
Tamara Savková
Júlia Gablíková
Michal Knor



FMFI UK
TIS

Obsah

1 Katalóg požiadaviek

1.1 Úvod

1.1.1 Predmet špecifikácie

V tejto špecifikácii požiadaviek sa popisujú používateľské, funkčné a parametrické požiadavky na aplikáciu, ktorá rozdeľuje slová na slabiky. Špecifikácia je určená pre zadávateľa a tím, ktorý bude aplikáciu implementovať. Zároveň je súčasťou dohody medzi objednávateľom a dodávateľom. Špecifikácia bude slúžiť ako východisko pre vyhodnocovanie správnosti softvéru.

1.1.2 Rozsah projektu

Výsledná aplikácia bude mať dve časti, ktoré sa budú dopĺňať - konzolovú aplikáciu a webové rozhranie. Konzolová aplikácia bude kompatibilná s operačnými systémami typu Microsoft Windows, Ubuntu a macOS. Používateľ bude môcť odoslať do aplikácie text v jednom z podporovaných slovanských jazykov. Aplikácia mu okrem iného vygeneruje ako výstup súbor, v ktorom je upravený vstupný text rozdelený na slabiky a ďalšie súbory, ktoré rozanalyzujú text vzhľadom na dĺžky a početnosti vyskytujúcich sa slabík. Úprava vstupného textu spočíva v tom, že v texte sú ponechané len samotné slová (neobsahuje napr. interpunkciu) a všetky písmená sú zmenené na malé. Aplikácia nenavracia ako výstup pôvodný text rozdelený na slabiky, vracia výlučne jeho upravenú verziu.

1.1.3 Slovník pojmov

1. Podporovaný slovanský jazyk – čeština, poľština, slovinčina, ruština, bieloruština, ukrajinčina, macedónčina, chorvátčina, bulharčina, horná a dolná lužická srbčina, srbčina
2. Konfiguračný súbor jazyka – súbor, ktorý popisuje jednotlivé jazyky a pravidlá, podľa ktorých sa v týchto jazykoch delia slová na slabiky, pričom formát tohto súboru je pevne daný a pripravený tak, aby s ním aplikácia vedela priamo pracovať

1.1.4 Odkazy

1. product_specification_2018_10_12.docx - súbor od zadávateľa so všeobecnými pravidlami delenia slov na slabiky
2. language_year_month_day.docx - 12 súborov od zadávateľa so špecifikáciou jednotlivých jazykov (názvy súborov zodpovedajú uvedenej šablóne)

1.1.5 Prehľad nasledujúcich kapitol

Druhá kapitola dokumentu popisuje perspektívu a funkcionalitu produktu, charakterizuje používateľov a uvádza tiež obmedzenia kladené na produkt. Tretia kapitola podrobne popisuje jednotlivé požiadavky zadávateľa na vizuálnu i na funkcionálnu časť produktu, obsahuje detailnejší opis vstupných a výstupných súborov, priložených príručiek a rozoberá tiež proces spracovania súborov.

1.2 Všeobecný popis

1.2.1 Perspektíva produktu

Produktom je aplikácia slúžiaca ako nástroj pre analýzu lingvistických dát. Aplikácia dostane vstup od používateľa, spracuje ho a ponúkne výstup na stiahnutie. Produkt rozdeľuje vstupný text v jednom z podporovaných slovanských (neskôr prípadne aj ďalších) jazykov na slabiky podľa pravidiel jazyka tohto

textu a spraví tiež analýzu výsledku. Vstup určený na spracovanie je možné vložiť do textového okna alebo nahrať ako textový súbor. Text vkladaný do textového okna musí byť v kódovaní UTF-8. Pri nahrávaní textového súboru treba vybrať jeho jazyk a kódovanie z ponuky. Po spracovaní vstupu aplikácia vráti niekoľko výstupov: textový súbor rozdelený na slabiky a excelové súbory s analýzami vstupu — tabuľka s početnosťou a dĺžkou slabík, frekvenčná tabuľka dĺžok slabík berúc do úvahy všetky výskyty slabík, frekvenčná tabuľka dĺžok slabík berúc do úvahy výskyt slabík v texte iba raz a postupnosť dĺžok slabík. Výstupné súbory sa dajú stiahnuť.

1.2.2 Funkcie produktu

Hlavnou funkciou produktu je rozdelenie slov textu na slabiky podľa pravidiel príslušného jazyka. Aplikácia pred spracovaním upraví vstup - vyčisti text od zbytočných znakov a zmení veľkosti písmen na malé. Následne rozdelí slová na slabiky, zistí početnosti i dĺžky slabík, frekvencie výskytov dĺžok slabík vrátane, aj bez opakovaných výskytov jednotlivých slabík a výsledky vráti vo forme výstupných súborov, ktoré si bude používateľ môcť stiahnuť.

1.2.3 Charakteristika používateľov

Stránka bude mat používateľov dvoch typov.
Návštevník – bude vedieť vkladať text alebo súbor, zvoliť vybrané vlastnosti textu a stiahnuť výstup po spracovaní. Aplikácia sa bude používať vedcami pre lingvistickú analýzu textov na základe rozdelenia slov na slabiky. Návštevníkovi postačujú začiatočnícke skúsenosti s prácou s počítačom.
Správca systému – udržuje systém v prevádzke, inštaluje na server a pridáva nové jazyky vrátane vytvorenia ich konfiguračných súborov. Časť správcovských úkonov je sprístupnená lokálne aj používateľovi, ktorý sa ich využívaním stáva správcom svojho systému.

1.2.4 Všeobecné obmedzenia

Aplikácia nevie rozpoznať jazyk ani kódovanie vstupu.

1.2.5 Predpoklady a závislosti

Aplikácia vráti správny výstup len v prípade správne zvoleného jazyka a kódovania vstupu, pretože spôsob spracovania textu závisí na týchto údajoch.

1.3 Špecifikácia požiadaviek

1.3.1 Používateľské rozhranie

Aplikácia bude používateľovi poskytovať možnosť vložiť vstupný text, vybrať jazyk a kódovanie vloženého textu a stiahnuť výstupné súbory.

1.3.1.1 Vkladanie textu priamo

Aplikácia bude obsahovať textové okno, do ktorého sa bude dať vložiť text určený na spracovanie.

1.3.1.2 Vkladanie textu vo forme súboru

Aplikácia bude ponúkať možnosť nahrať textový súbor obsahujúci text určený na spracovanie.

1.3.1.3 Výber jazyka

Pred spracovaním textu (nech už je vložený akokoľvek) si používateľ bude musieť vybrať jazyk textu, ktorý vložil. Pri výbere nesprávneho jazyka bude výstup chybný. Ponuka bude obsahovať zoznam podporovaných slovanských jazykov, pričom neskôr môžu byť pridané aj ďalšie podporované jazyky.

1.3.1.4 Výber kódovania

V prípade vloženia textu vo forme súboru si používateľ bude musieť vybrať jeho kódovanie. Pri zvolení nesprávneho kódovania nie je zaručená správnosť výstupu. V prípade vloženia textu do textového okna je nutné, aby text bol v kódovaní UTF-8.

1.3.1.5 Získanie výstupu

Aplikácia po spracovaní vstupu umožní používateľovi stiahnuť výstupné súbory jednotlivo alebo spolu.

1.3.2 Vstupný text

Hlavným vstupom aplikácie bude text (priamo vložený používateľom alebo získaný z textového súboru) ľubovoľnej dĺžky napísaný v cyrilike alebo latinke. Okrem znakov abecedy vybraného jazyka môže obsahovať aj iné znaky (napr. číslice, interpunkciu a podobne).

1.3.3 Spracovanie textu

Pred vyprodukovaním výstupu bude aplikácia musieť najprv upraviť vstupný text a následne upravený text spracovať rôznymi spôsobmi.

1.3.3.1 Vyčistenie textu od nadbytočných znakov

Aplikácia odstráni z textu všetky znaky, ktoré podľa súborov poskytnutých zadávateľom nepatria do abecedy vybraného jazyka. Okrem písmen abecedy zostanú iba medzery.

1.3.3.2 Zmena veľkosti písmen

Aplikácia zmení všetky veľké písmená na malé.

1.3.3.3 Rozdelenie textu na slabiky

Podľa pravidiel špecifických pre jazyk textu (zvolený používateľom) aplikácia rozdelí slová na slabiky. Pravidlá pre jednotlivé jazyky sú dané v súboroch poskytnutých zadávateľom a tie sú následne upravené do konfiguračných súborov pripravených pre spracovanie aplikáciou.

1.3.3.4 Zistenie početnosti slabík

Pre jednotlivé slabiky vyskytujúce sa v texte aplikácia zistí ich početnosť.

1.3.3.5 Zistenie dĺžky slabík

Pre jednotlivé slabiky vyskytujúce sa v texte aplikácia zistí ich dĺžku. Pravidlá na zisťovanie dĺžky slabiky v jednotlivých jazykoch sú definované v súboroch poskytnutých zadávateľom a konfiguračných súboroch z toho vyhotovených.

1.3.3.6 Zistenie frekvencie výskytu dĺžok slabík vrátane opakovaných výskytov

Aplikácia vypočíta počet výskytov dĺžok slabík, pričom zarátava aj opakované výskyty konkrétnej slabiky s danou dĺžkou. Príklad: „mama“ obsahuje dvakrát slabiku „ma“, a teda do počtu výskytov slabík s dĺžkou 2 prispieva dvoma výskytmi.

1.3.3.7 Zistenie frekvencie výskytu dĺžok slabík bez opakovaných výskytov

Aplikácia vypočíta počet výskytov dĺžok slabík, pričom sa nebudú zarátavať opakované výskyty konkrétnej slabiky s danou dĺžkou – teda každá slabika prispeje len raz. Príklad: „mama“ obsahuje dvakrát slabiku „ma“, a teda do počtu výskytov slabík s dĺžkou 2 prispieva iba jedným výskytom.

1.3.4 Výstupné súbory

Výsledky spracovania textu budú prezentované používateľovi formou výstupných súborov.

1.3.4.1 Text rozdelený na slabiky

Aplikácia vytvorí textový súbor, v ktorom bude vstupný text už po rozdelení na slabiky, pričom slabiky budú od seba oddelené pomlčkami.

1.3.4.2 Tabuľka s početnosťou a dĺžkou slabík

Aplikácia vytvorí excelový súbor, kde v prvom stĺpci budú slabiky, ktoré sa v texte vyskytli, v druhom stĺpci ich početnosti a v treťom ich dĺžky. Slabiky budú zoradené podľa početnosti zostupne.

1.3.4.3 Frekvenčná tabuľka dĺžok slabík berúc do úvahy všetky výskyty slabík

Aplikácia vytvorí excelový súbor, pričom v prvom stĺpci bude uvedené, aké dĺžky slabík sa vo vstupnom texte vyskytli (usporiadané vzostupne). Druhý stĺpec bude obsahovať počet výskytov týchto dĺžok (vrátane opakovaných výskytov).

1.3.4.4 Frekvenčná tabuľka dĺžok slabík berúc do úvahy výskyt slabík v texte iba raz

Aplikácia vytvorí excelový súbor, pričom v prvom stĺpci bude uvedené, aké dĺžky slabík sa vo vstupnom texte vyskytli (usporiadané vzostupne). Druhý stĺpec bude obsahovať počet výskytov týchto dĺžok (bez opakovaných výskytov).

1.3.4.5 Postupnosť dĺžok slabík

Aplikácia vytvorí textový súbor podobný textu rozdelenému na slabiky, avšak namiesto jednotlivých slabík budú v tomto súbore napísané ich dĺžky. Čísla budú oddelené pomlčkami. Príklad: „mama“ z pôvodného textu bude nahradené postupnosťou „2-2“.

1.3.5 Používateľská príručka

K aplikácii bude priložený textový súbor obsahujúci návody určené pre používateľa aplikácie.

1.3.5.1 Lokálne použitie konzolovej aplikácie

Príručka bude obsahovať podrobný návod, ako sfunkčniť konzolovú aplikáciu na počítači používateľa, ako ju spustiť a ďalej obsluhovať.

1.3.5.2 Lokálne pridanie nového jazyka

Používateľ bude môcť lokálne rozšíriť konzolovú aplikáciu o nový jazyk, ktorý bude následne zaradený medzi podporované jazyky. Návod bude okrem iného obsahovať aj popis štruktúry konfiguračného súboru jazyka, ktorý bude potreba pri pridávaní vytvoriť. Návod neobsahuje úpravu webového rozhrania.

1.3.6 Správcovská príručka

K aplikácii bude priložený textový súbor obsahujúci návody určené pre správcu aplikácie.

1.3.6.1 Inštalácia aplikácie na webový server

Príručka bude obsahovať podrobný návod, ako sfunkčniť aplikáciu na webovom serveri.

1.3.6.2 Pridanie nového jazyka

Správca bude môcť rozšíriť celú aplikáciu o nový jazyk, ktorý bude následne zaradený medzi podporované jazyky. Návod bude okrem iného obsahovať aj popis štruktúry konfiguračného súboru jazyka, ktorý bude potreba pri pridávaní vytvoriť. Súčasťou návodu je úprava webového rozhrania aplikácie.

1.3.7 Ostatné požiadavky

1.3.7.1 Jednoduchosť

Užívateľské prostredie bude prehľadné a ľahko ovládateľné. Aplikácia sa bude dať spustiť bez použitia iných aplikácií.

1.3.7.2 Voľná dostupnosť

Aplikácia bude vytvorená pomocou voľne dostupného legálneho softvéru.

2 Návrh

2.1 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.

2.2 Rozdelenie na časti

2.2.1 Popis modulov

2.2.1.1 Frontend webovej aplikácie

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

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, „Submit“ tlačidlo a pätičku stránky
- používateľ vkladá dáta pomocou súboru, ktorý nahrá alebo priamo vloží text do textového poľa
- modul obsahuje dva „selekty“ (prvý je prístupný len pri nahrávaní súboru, druhý vždy):
    1. „encoding“ , z ktorého sa dá vybrať kódovanie zo zoznamu podporovaných kódovaní
    2. „language“ , z ktorého sa dá vybrať jazyk zo zoznamu jazykov
- modul dáta posiela do „backendu“ pomocou formulára po odkliknutí tlačidla „Submit“.

2.2.1.2 Backend webovej aplikácie

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

Implementácia:

- ak bol súbor nahratý cez formulár, tak ho modul uloží do priečinku temp_files/*random_name*
- ak bol vstup zadaný cez textové pole tak tento vstup zapíše do súboru, tomuto súboru dá meno podľa prvého slova textu a následne tento súbor uloží do priečinku temp_files*/random_name*
- modul spustí Spúšťací a ukončovací modul, ktorý vygeneruje výstupné súbory
- po ukončení procesu modul zabalí výstupné súbory (.zip), vynúti stiahnutie súborov a vymaže všetky dočasné súbory

2.2.1.3 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
- file_path – cesta k vygenerovanému priečinku temp_files/*random_name*
- encoding - typ kódovania súboru
- language - jazyk v ktorom je súbor napísaný

Postup

Vytvoria sa objekty:
- data - objekt typu ConfigData, ktorému poskytne názov konfiguračného súboru zvoleného jazyka
- pipe_read_clean - objekt typu Pipe
- pipe_clean_sound - objekt typu Pipe
- pipe_sound_syll - objekt typu Pipe
- pipe_syll_count - objekt typu Pipe
- pipe_count_txt - objekt typu Pipe

Inicializujú sa moduly:
- Upravovací modul s atribútmi: pole obsahujúce pipe_read_clean a pipe_clean_syll
- Zvukový modul s atribútmi: pole obsahujúce pipe_clean_sound a pipe_sound_syll
- Slabikovací modul s atribútmi: pole obsahujúce pipe_sound_syll a pipe_syll_count
- Počítací modul s atribútmi: pole obsahujúce pipe_syll_count a pipe_count_txt, data, file_path + file_name
- modul Zápis naslabikovaného textu s atribútmi: pole obsahujúce pipe_count_txt, file_path

Spustenie práce modulov a ich ukončenie:
- na každom module sa zavolá funkcia start().
- zavolá Čítací modul a atribútmi: pole obsahujúce pipe_read_clean, ďalej atribúty file_path + file_name, kódovanie vstupného textu, Data.
- na každom module sa zavolá funkcia join().

2.2.1.4 Čítací modul

Čítací modul je určený na vytváranie zoznamu objektov tried TextPunctuation a End na základe vstupných argumentov.

Argumenty:
- Pipes – pole s jedným prvkom typu Pipe (pipe_read_clean);
- Adresa vstupného textu (file_path);
- Kódovanie vstupného textu (encoding);
- Objekt triedy ConfigData s konfiguračnými dátami (data).

Modul je objektom triedy ReadModule, ktorá je podtriedou triedy Module.
Daný modul vytvára objekty triedy TextPunctuation (podtrieda triedy Word), ktoré:
- reprezentujú “slovo” – jednotlivú oddelenú prázdnym symbolov postupnosť symbolov zo vstupného textu,
- majú atribúty:
    - text “slova” – zoznam symbolov slova (_text),
    - zoznam špeciálnych označení pre interpunkčné znamienka – zoznam označení priradených ku každému symbolu (_punct).

Tiež modul vytvára objekt triedy End, ktorý označuje, že text, ktorý je v súbore s názvom vstupného textu, bol prečítaný a modul skončil svoju prácu.

Špeciálnymi označeniami sú cifry, ktoré označujú úlohu symbolu v slove:
interpunkčne znamienka (konštanta PUNCT - 5),
prenos (konštanta HYPHEN - 6).

Funkcie modulu:
- Modul rozdeľuje vstupný text na “slova” - jednotlivé oddelené prázdnym symbolov postupnosti symbolov zo vstupného textu.
- Tiež modul nastavuje špeciálne označenia pre zodpovedajúci objekt triedy TextPunctuation každého “slova”:

- Interpunkčne znamienka - označenie ostatných pomlčiek a znakov . , : ; ? ! [ ] ( ) { } ⟨ ⟩ — — ‹ › « » “ ” ″ ″;
- Prenos - označenie pomlčky, použitej pre rozdelenie “slova” na riadky, aby sa dalo ich spracovať.

Po jednom pridáva objekty triedy TextPunctuation alebo End do dátovej štruktúry queue. Ak “slovo” je posledné v texte, modul pridá do dátovej štruktúry queue objekt triedy End, inak objekt triedy TextPunctuation.

Implementácia:

- Modul vytvorí pomocné premenné:
  - zoznam pre objekty triedy TextPunctuation 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 triedy TextPunctuation na základe obsahu pracovných bufferov a pridá sa do zoznamu pre získané objekty triedy TextPunctuation.

Keď sa načíta cely text:

- Ak v pracovných bufferoch ešte sú nejaké dáta, vytvorí sa objekt triedy TextPunctuation na základe obsahu pracovných bufferov a pridá sa do zoznamu pre získané objekty triedy TextPunctuation.
- Modul pošle objekt triedy End do dátovej štruktúry queue.

2.2.1.5 Upravovací modul

Upravovací modul je určený na prípravu slova na slabikovanie. Pomocou konfiguračných vytvára zo slova typu TextPunctuation získaného z pipe_read_clean slovo typu Text.

Argumenty:

1. Pole s dvoma prvkami typu Pype pre vstup a výstup (pipe_read_clean, pipe_clean_phono) – pipes;
2. Objekt s konfiguračnými dátami triedy ConfigData – data.

Modul je objektom triedy CleanModule, ktorá je podtriedou triedy ThreadModule.

- modul dedí metódu run(), ktorú je nutné implementovať;
- modul dedí metódu __init__(self, pipes), ktorá okrem iného vytvára thread s targetom run();
- modul dedí metódy start() a join(), ktoré slúžia na prácu s threadom modulu.

Funkcie modulu:

- Získava z pipe pre vstup objekty triedy TextPunctuation (podtrieda triedy Word), a postupne z nich vytvára dvojice (aktuálny a nasledujúci objekt), spracováva dvojicu, a aktualizuje dvojicu po získaní ďalšieho slova (“aktuálny” dostáva hodnotu “nasledujúceho” a “nasledujúci” hodnotu nového slova).
- Ak sa podľa objektu s konfiguračnými dátami slová, ktoré neobsahujú slabiky, spájajú s predchádzajúcim alebo nasledujúcim slovom, modul vykoná tieto operácie.
- Modul bude ignorovať slová, ktoré vo vnútri obsahujú znaky nepatriace do jazyka.
- Modul bude ignorovať interpunkčné znaky, ktoré sú na začiatku alebo konci slova.
- Výsledný objekt triedy Text bude obsahovať v texte slova len povolene písmena (podľa objektu s konfiguračnými dátami).

Implementácia:

- Modul získava z pipe pre vstup objekty triedy TextPunctuation a postupne vytvára z nich dvojice (aktuálny a nasledujúci objekt).
- Pomocou funkcie spracováva dvojicu, ak jej prvý element je objektom triedy TextPunctuation:
    - Nastaví pomocné premenné – pracovný buffer pre text a interpunkčné znamienka slova.
    - Ak text slova obsahuje viac ako jeden symbol a text obsahuje len veľké písmena, potom sa dané slovo ignoruje a funkcia vráti dvojicu: null a „aktuálny“ objekt.
    - Prechádza každým symbolom textu a k nemu prislúchajúcim špeciálnym označením.
        - Ak symbolom je bodka, „nasledujúci“ objekt je objektom triedy TextPunctuation a jeho text nezačína veľkým písmenom, potom dané slovo sa ignoruje a funkcia vráti dvojicu: null a „nasledujúci“ objekt.
        - Ak buffer pre text slova je prázdny, špeciálne označenie je ‘interpunkčné znamienko’, potom:
            - Ak symbol patri do symbolov ., : ; ? ! [ ] ( ) { } ⟨ ⟩ ‹ › « »“ ” „ ’ ‘ , potom sa načíta ďalší symbol;
            - Ak symbol patri do symbolov — —, potom funkcia vráti dvojicu: null a „nasledujúci“ objekt.
        - Ak špeciálne označenie je ‘prenos’ a ďalší symbol je písmeno, potom sa tento symbol ignoruje a nepridáva do pracovných bufferov. Načíta ďalší symbol a špeciálne označenie. Ak ďalší symbol nie je písmenom, potom funkcia vráti dvojicu: null a „nasledujúci“ objekt.
        - Ak špeciálne označenie je ‘interpunkčné znamienko’ a symbol nepatrí do — —, potom sa pre všetky ďalšie symboly slova kontroluje, či ich označenia sú tiež ‘interpunkčné znamienka’. Ak nie, vráti dvojicu: null a “nasledujúci” objekt. Ak áno, vráti dvojicu: nový objekt vytvorený na základe pomocných bufferov TextPunctuation a “nasledujúci” objekt.
        - Ďalej sa symbol zmení na malý (lowercase)
        - Prebehne kontrola, či symbol patrí do malých písmen jazyka (podľa objektu s konfiguračnými dátami). Ak áno, malý symbol sa pridá do pracovných bufferov. Ak nie, vráti dvojicu: null a “nasledujúci” objekt.
    - Na záver, ak nasledujúci objekt patri do triedy TextPunctuation a dĺžka jeho textu je 1, vytvorí sa jeho lowercase verzia – objekt triedy TextPunctuation. Ak je jeho text 0-slabičné slovo jazyka (podľa objektu s konfiguračnými dátami), potom:
        - Ak sa objekt pridáva k predchádzajúcemu objektu (podľa objektu s konfiguračnými dátami), potom sa k pomocným bufferom pridávajú text a špeciálne označenia daného objektu a „nasledujúci“ objekt sa nastavuje na null.
    - Ak nasledujúci objekt patri do triedy TextPunctuation, dĺžka „aktuálneho“ objektu je 1 a jeho text je 0-slabičné slovo jazyka (podľa objektu s konfiguračnými dátami), potom:
        - Ak objekt sa pridáva k nasledujúcemu objektu (podľa objektu s konfiguračnými dátami), potom na začiatok textu “nasledujúceho” objektu sa pridávajú text a špeciálne označenia „aktuálneho“ objektu.
        - Funkcia vráti dvojicu: null a „nasledujúci“ objekt.
    - Funkcia na konci vráti dvojicu: nový objekt vytvorený na základe pomocných bufferov TextPunctuation a “nasledujúci” objekt.
- Aktualizuje dvojicu objektov po získaní ďalšieho slova (“aktuálny” dostáva hodnotu “nasledujúceho ”, a “nasledujúci” hodnotu nového slova) a začína spracovávať ďalšiu dvojicu.

- Ak z queue pre vstup bol získaný objekt triedy End, potom:
    - Trojica objektov sa aktualizuje posledný krát,
    - Vykoná sa posledné spracovanie trojice objektov triedy Text,
    - Spracovaný objekt triedy Text sa pridá do queue pre výstup,
    - Objekt End sa pridá do queue pre výstup,
    - Práca modulu skončí.

2.2.1.6 Zvukový modul

Modul je určený na priradenie fonotypov písmenám podľa pravidiel jazyka.

Argumenty:
1. pipes – pole dĺžky 2 obsahujúce objekty typu Pipe v nasledujúcom poradí [pipe_clean_sound, pipe_sound_syll]
2. data - ConfigData objekt

Dedenie:
- modul dedí metódu run(), ktorú je nutné implementovať
- modul dedí metódu __init__(self, pipes), ktorá okrem iného vytvára thread s targetom run()
- modul dedí metódy start() a join(), ktoré slúžia na prácu s threadom modulu

Metóda run() – inicializácia:
- modul vytvorí dve premenné pipe_in, ktorá odkazuje na pipes[0] a pipe_out, ktorá odkazuje na pipes[1]
- modul si do premennej running vloží hodnotu True
- následne modul spustí cyklus podmienený hodnotou premennej running (while running), v ktorom beží zvyšok programu

Implementácia:
Modul vyberie z pipe_in (= pipe[0]) objekt typu Text. Do pipe_out (= pipe[1]) sa bude vkladať objekt typu TextPhonotypes, ktorého atribút _text obsahuje Text._text a atribút _phonotypes je pole konštánt. K písmenu na indexe *i* v slove _text prislúcha fonotyp v poli _phonotypes na indexe *i* označený niektorou z konštánt *SONOR, CON, VOWEL, SPEC, SUBUNIT*. Fonotyp je určený pomocou štruktúry data, pričom sú zohľadnené všetky výnimky, ktoré môžu fonotyp daného písmena ovplyvniť.

Ukončenie behu modulu:
- ak modul vo fáze získania objektu z pipe_in zistí, že získaný objekt je typu End, nastaví premennú running na False, a tento objekt vloží do pipe_out podobne ako vkladá TextPhonotypes

2.2.1.7 Slabikovací modul

Slabikovací modul je určený na rozdelenie slov na slabiky.

Argumenty:
1. pipes – pole dĺžky 2 obsahujúce objekty typu Pipe v nasledujúcom poradí [pipe_sound_syll, pipe_syll_count]

Modul vyberie z pipe_in (= pipe[0]) objekt typu TextPhonotypes. Slovo uložené v obdržanom objekte pod atribútom _text bude rozdelené na slabiky vzhľadom na atribút _phonotypes, ktorý definuje sonoritu jednotlivých písmen. Následne bude vytvorené slovo SyllablesPhonotypes, ktoré bude vložené do pipe_out (= pipe[1]).

Postup

Dedenie:
- modul dedí metódu run(), ktorú je nutné implementovať

- modul dedí metódu __init__(self, pipes), ktorá okrem iného vytvára thread s targetom run()
- modul dedí metódy start() a join(), ktoré slúžia na prácu s threadom modulu

Metóda run() a inicializácia:
- modul vytvorí dve premenné pipe_in, ktorá odkazuje na pipes[0] a pipe_out, ktorá odkazuje na pipes[1]
- modul si do premennej running vloží hodnotu True
- následne modul spustí cyklus podmienený hodnotou premennej running (while running), v ktorom beží zvyšok programu

Získanie objektu TextPhonotypes z pipe_in :
- modul sa najprv pokúsi získať pipe_in (pipe_in.acquire())
- po získaní pipe_in sa modul pokúsi vybrať z pipe_in prvok. Ak sa mu to nepodarí, čaká (pipe_in.wait()). Ak sa mu to podarí, uvoľní pipe_in a pokračuje.

Rozdelenie slova TextPhonotypes._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 TextPhonotypes._phonotypes 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 TextPhonotypes._phonotypes pokračuje 2. iteráciou

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

Rozdelenie slova TextPhonotypes._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 a SUBUNIT:
- ak má znak hodnotu SPEC, 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 SUBUNIT, netreba robiť nič, pretože týmto znakom určite nekončí slabika (SUBUNIT nemá hodnotu 2)

Rozdelenie slova TextPhonotypes._text na slabiky (3. iterácia):
- modul vytvorí výsledné pole D, ktoré bude obsahovať stringy (slabiky)
- podľa dĺžok prvkov poľa A rozkúskuje slovo vo TextPhonotypes._text a tieto substringy vloží do poľa D

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

Vloženie slova SyllablesPhonotypes do pipe_out:
- modul sa najprv pokúsi získať pipe_out (pipe_out.acquire())
- po získaní pipe_out modul vloží do pipe_out objekt typu SyllablesPhonotypes s argumentami (D, A)
- modul urobí pipe_out.notify()
- modul uvoľní pipe_out

Ukončenie behu modulu:

- ak modul vo fáze získania objektu z pipe_in zistí, že získaný objekt je typu End, nastaví premennú running na False, a tento objekt vloží do pipe_out podobne ako vkladá SyllablesPhonotypes

2.2.1.8 Počítací modul

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

Argumenty:
1. pipes – pole dĺžky 2 obsahujúce objekty typu Pipe v nasledujúcom poradí [pipe_syll_count, pipe_count_txt]
2. data – objekt typu ConfigData
3. file_path – cesta k vstupnému súboru (string)

Modul vyberie z pipe_in [= pipes[0]] pole a každú slabiku samostatne spracuje – zistí a zaznamená jej dĺžku a zaráta jej výskyt.

Postup

Dedenie:
- modul dedí metódu run(), ktorú je nutné implementovať
- modul dedí metódu __init__(self, pipes), ktorá okrem iného vytvára thread s targetom run()
- modul dedí metódy start() a join(), ktoré slúžia na prácu s threadom modulu

Metóda run() a inicializácia:
- modul vytvorí dve premenné pipe_in, ktorá odkazuje na pipes[0] a pipe_out, ktorá odkazuje na pipes[1]
- 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 *running* vloží hodnotu True
- následne modul spustí cyklus podmienený hodnotou premennej *running* (while *running*), v ktorom beží zvyšok programu (okrem zavolania nasledujúceho modulu)

Získanie objektu SyllablesPhonotypes z pipe_in:
- modul sa najprv pokúsi získať pipe_in (pipe_in.acquire())
- po získaní pipe_in sa modul pokúsi vybrať z pipe_in prvok *p* typu SyllablesPhonotypes. Ak sa mu to nepodarí, čaká (pipe_in.wait()). Ak sa mu to podarí, uvoľní pipe_in a pokračuje.

Spracovanie slabík:
- modul prechádza slabiky poľa *p._syllables*
- 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._syllables* vloží modul novovytvorený prvok typu SyllablesLengths s argumentom (A, p._syllables) do pipe_out 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 SyllablesLengths do pipe_out:

- modul sa najprv pokúsi získať pipe_out (pipe_out.acquire())
- po získaní pipe_out modul vloží do pipe_out prvok typu SyllablesLengths s argumentom (A, p._syllables)
- modul urobí pipe_out.notify()
- modul uvoľní pipe_out

Ukončenie práce:

- ak modul vo fáze získania prvku *p* z pipe_out zistí, že *p* je typu End, nastaví premennú running na False, a teda po spracovaní tohto slova skončí cyklus
- získaný prvok p typu End modul vloží do pipe_out podobne ako SyllablesLengths
- po ukončení cyklu modul zavolá Výsledkový modul s argumentami map_len, map_freq, file_path

2.2.1.9 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_path – cesta k vstupnému súboru (string)

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 map_wout_rep je kľúčom dĺžka slabiky a hodnotou počet všetkých výskytov danej dĺžky slabiky.

Postup

Dedenie:

- modul dedí metódu run(), ktorú je nutné implementovať
- modul dedí metódu __init__(self), ktorá okrem iného vytvára thread s targetom run()

Metóda run() – inicializácia:

- modul si do premennej run vloží hodnotu True
- následne modul spustí cyklus podmienený hodnotou premennej run (while run), v ktorom beží zvyšok programu

Vytvorenie a naplnenie 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 a naplnenie 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 tabuliek’ s argumentami: map_len, map_freq, map_with_rep, map_wout_rep, file_path

2.2.1.10 Zápis naslabikovaného textu

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

Argumenty:

1. pipes – pole dĺžky 1 obsahujúce objekty typu Pipe [pipe_count_text]
2. file_path – cesta k vstupnému súboru (string)

Modul vyberá z pipe_in (= pipe[0]) object typu SyllablesPhonotypes. Slabiky v obdržanom objekte pod atribútom _syllables sa uložia do poľa A a ich sonorita pod atribútom _phonotypes do poľa B. Ak pole A dosiahne dĺžku ‘n’, tak vytvorí/otvorí súbor ‘syllable_text.txt’ a uloží tam obsah pola A. Ak pole B dosiahne dĺžku ‘n’, tak vytvorí/otvorí súbor ‘syllable_ lengths_text.txt’, kam uloží obsah pola B.

Postup

Dedenie:

- modul dedí metódu run(), ktorú je nutné implementovať
- modul dedí metódu __init__(self, pipes, file_path), ktorá okrem iného vytvára thread s targetom run()

Metóda run() – inicializácia:

- modul si vytvorí pomocné pole A
- modul si vytvorí pomocné pole B
- modul si do premennej run vloží hodnotu True
- následne modul spustí cyklus podmienený hodnotou premennej run (while run), v ktorom beží zvyšok programu

Získanie objektu SyllablesPhonotypes z premennej pipe_in:

- modul sa najprv pokúsi získať pipe_in (pipe_in.acquire())
- po získaní pipe_in sa modul pokúsi vybrať z pipe_in prvok; ak sa mu to nepodarí, čaká (pipe_in.wait()). Ak sa mu to podarí, uvoľní pipe_in a pokračuje

Zapísanie naslabikovaného textu do súboru:

- modul postupne prechádza pole SyllablesPhonotypes._syllables a vkladá tieto prvky do poľa A
- ak pole A presiahne dĺžku ‘n’, tak obsah pola A zapíše do súboru ‘syllable_text.txt’, pričom slabiky jednotlivých slov sú oddelené pomlčkou a pole A vynuluje

Zapísanie naslabikovaného textu do súboru vo forme čísel:

- modul postupne prechádza pole SyllablesPhonotypes._phonotypes a vkladá tieto prvky do poľa B
- ak pole B presiahne dĺžku ‘n’, tak obsah pola B zapíše do súboru ‘syllable_lengths_text.txt’, kde slabiky sú nahradené ich dĺžkou a oddelené pomlčkou a pole B vynuluje

Ukončenie behu modulu:

- ak modul vo fáze získania objektu z pipe_in zistí, že získaný objekt je typu End(), nastaví premennú run na False a teda po spracovaní tohto slova ukončí svoj beh

2.2.1.11 Zápis tabuliek

Modul má na starosti vytvorenie troch tabuľkových súborov .xls.

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. map_with_rep – obsahuje prvky typu integer1: integer2, kde integer1 je dĺžka slabiky a integer2 je počet všetkých slabík majúcich takúto dĺžku
4. map_wout_rep – obsahuje prvky typu integer1: integer2, kde integer1 je dĺžka slabiky a integer2 je počet unikátnych slabík majúcich takúto dĺžku
5. file_path – cesta k vstupnému súboru (string)

Modul vytvorí v priečinku file_path 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

Dedenie:

- modul dedí metódu run(), ktorú je nutné implementovať
- modul dedí metódu __init__(self), ktorá okrem iného vytvára thread s targetom run()

Metóda run() – inicializácia:

- modul si do premennej run vloží hodnotu True
- následne modul spustí cyklus podmienený hodnotou premennej run (while run), v ktorom beží zvyšok programu

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_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_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_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_without_repetition.xls' v priečinku file_name
- zapíše do tabuľky prvky poľa A, ktorej stĺpce sú: dĺžka slabiky, početnosť, percentuálny výskyt

2.2.2 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.

2.2.2.1 Moduly a vlákna

- ***Main thread***: Spúšťací a ukončovací modul -> Čítací modul (-> Spúšťací a ukončovací modul)
- ***Thread1***: Upravovací modul
- ***Thread2***: Zvukový modul
- ***Thread3***: Slabikovací modul
- ***Thread4***: Počítací modul -> Výsledkový modul -> Zápis tabuliek
- ***Thread5***: Zápis naslabikovaného textu

2.2.2.2 Komunikácia medzi vláknami

- ***Spúšťací a ukončovací modul***: Od modulu Backend webovej aplikácie získava dáta, na základe ktorých vytvára všetky štruktúry potrebné na komunikáciu, spúšťa moduly 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 pipe_read_clean.
- ***Thread1 – Thread2***: Upravovací modul komunikuje so Zvukovým modulom pomocou pipe_clean_sound.
- ***Thread2 – Thread3***: Zvukový modul komunikuje so Slabikovacím modulom pomocou pipe_sound_syll.
- ***Thread3 – Thread4***: Slabikovací modul komunikuje s Počítacím modulom pomocou pipe_syll_count.
- ***Thread4 – Thread5***: Počítací modul komunikuje s modulom Zápis naslabikovaného textu pomocou pipe_count_txt.

2.2.3 Dáta a formáty súborov

2.2.3.1 Funkčné súbory aplikácie

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

2.2.3.2 Konfiguračné súbory jazyka

Súbory, v ktorých sú uložené pravidlá pre prácu s jazykom analyzovaného textu, budú vo formáte **json**.

Štruktúra:

```
{
  "language": {
      "name": "language_name",
      "writing_system": "latin/cyrillic"
   },
   "lowercase": {
       "lowercase_letter1": {
           "phoneme-length": length or -1,
```

```
          "sign": "SONOR/CONS/VOWEL/SPEC",
          "subunit": 0/1
      },
      "lowercase_letter2": {
          "phoneme-length": length or -1,
          "sign": "SONOR/CONS/VOWEL/SPEC",
          "subunit": 0/1
      },
       ...
},
"clusters": {
    "letter1": {
        "letter1": {
          "except_in_subword ": ["subword1", "subword2", ...],
          "only_in_subword": ["subword1", "subword2", ...],
          "length": length,
          "phonotype": "SONOR/CONS/VOWEL"
        }
      },
      ...
},
"phonotype_changes": {
    "letter1": {
       "preceding": "SONOR/CONS/VOWEL/NONE",
       "following": "SONOR/CONS/VOWEL/NONE",
       "becomes": "SONOR/CONS/VOWEL/NONE",
       "only_in_word": ["word1", "word2", ...]
    },
    "letter2": {
       "preceding": "SONOR/CONS/VOWEL/NONE",
       "following": "SONOR/CONS/VOWEL/NONE",
       "becomes": "SONOR/CONS/VOWEL/NONE",
       "only_in_word ": ["word1", "word2", ...]
    },
    ...
},
"text_changes": {
    "text1": {
       "becomes": "resultText"
    },
    "text2": {
       "becomes": "resultText"
    },
    ...
},
"zero-syllable_words": {
    "word1": "to_following/to_preceding",
    "word2": "to_following/to_preceding",
    ...
},
"special_sound_length": {
    "iu": {
        "0": [
          {
            "preceding": {
               "signs": ["NONE"/"CONS"/"VOWEL"/"SPEC", ...],
               "letters": ["letter1", "letter2", ...]
          },
           "following": {
               "signs": ["NONE"/"CONS"/"VOWEL"/"SPEC", ...],
               "letters": ["letter1", "letter2", ...]
           }
          },
          ...
        ],
        "1": [
          {
            "preceding": {
```

```
                "signs": ["NONE"/"CONS"/"VOWEL"/"SPEC", ...],
                "letters": ["letter1", "letter2", ...]
             },
              "following": {
                "signs": ["NONE"/"CONS"/"VOWEL"/"SPEC", ...],
                "letters": ["letter1", "letter2", ...]
             }
            },
            ...
           ],
            "2": [
            {
              "preceding": {
                "signs": ["NONE"/"CONS"/"VOWEL"/"SPEC", ...],
                "letters": ["letter1", "letter2", ...]
             },
              "following": {
                "signs": ["NONE"/"CONS"/"VOWEL"/"SPEC", ...],
                "letters": ["letter1", "letter2", ...]
             }
            },
            ...
           ]
      }
  }
}
```

Popis:

- *language*: obsahuje meno jazyka v angličtine, a písmo, ktoré sa v konfiguračnom súbore používa
- *lowercase*: obsahuje malé písmená jazyka, pričom pre každé písmeno uvádza zvukovú dĺžku phoneme-length (ak je uvedené -1, znamená to, že dĺžka môže byť rôzna a treba nazerať do *special_sound_length*), označenie fonotypu sign (môže byť SONOR, CONS, VOWEL, SPEC - pre špeciálne znaky jazyka) a informáciu o tom, či písmeno môže byť subunit (0 – nemôže, 1 – môže). Ak písmeno môže byť subunit, treba nazrieť do *clusters* a pokiaľ skúmané písmeno spĺňa podmienku tam uvedenú, viac s ním nepracovať ako so samostatným písmenom (informácie uvedené v *lowercase* sú neplatné).
- *clusters:* obsahuje písmená, ktoré môžu byť prvým písmenom zvukovo nedeliteľnej dvojice písmen. Každé uvedené písmeno obsahuje zoznam „druhých" písmen, ktoré s ním môžu tvoriť takúto dvojicu. Ak túto dvojicu tvoria vždy okrem prípadov, kedy sa nachádzajú v určitých podslovách, zoznam týchto výnimiek je uvedený v except_in_subword. Ak túto dvojicu naopak netvoria nikdy okrem prípadov, kedy sa nachádzajú v určitých podslovách, zoznam týchto podslov je uvedený v only_in_subword. Táto dvojica je považovaná za jedno písmeno a teda je uvedená jej dĺžka aj fonotyp (jednotlivé písmená tvoriace dvojicu sa neposudzujú nikdy zvlášť). Ak je dĺžka -1, treba túto dvojicu tvoriacu jedno písmeno hľadať v special_sound_length.
- *phonotype_changes*: obsahuje písmená, ktorých fonotyp sa môže zmeniť v závislosti od okolitých písmen. Preceding spolu s following tvoria jednu podmienku, ktorá musí byť kompletne splnená, aby sa uplatnila zmena fonotypu. V preceding a following môže byť uvedený niektorý z fonotypov (SONOR, CONS, VOWEL), čo znamená, že túto časť podmienky spĺňa ktorékoľvek písmeno danej sonority, alebo NONE, čo znamená, že na danom mieste (pred/za) nesmie byť žiadne písmeno. Iná podmienka (nezávislá od predchádzajúcej) na zmenu fonotypu je, že sa písmeno nachádza v konkrétnom celom slove (neplatí pre podslová) – tieto slová sú uvedené v only_in_word. Ak je ktorákoľvek z týchto troch podmienok splnená, za fonotyp písmena sa nepovažuje to, čo bolo uvedené v *lowercase*, ale to, čo je uvedené v becomes.
- *text_changes:* obsahuje textové reťazce (v texte sa môžu nachádzať vo vnútri slova), pričom je pri každom z nich uvedené, na akú postupnosť znakov má byť reťazec prepísaný.
- *zero-syllable_words*: obsahuje slová, ktoré neobsahujú slabiku, pričom pre každé z nich je uvedené, či sa majú pri slabikovaní a analyzovaní pripojiť k slovu predchádzajúcemu (to_preceding) alebo nasledujúcemu (to_following).
- *special_sound_length*: obsahuje písmená, ktoré menia zvukovú dĺžku v závislosti od okolitých písmen. Pre každé písmeno je pre každú z dĺžok (0, 1, 2) uvedený zoznam postačujúcich podmienok, pri ktorých splnení túto dĺžku písmeno nadobúda. Každá podmienka sa skladá

z dvoch „podpodmienok“ – jedna hovorí o tom, aké znaky pripadne fonotypy musia nasledovať, aby vyššia podmienka bola splnená, a druhá o tom, aké musia predchádzať skúmanému písmenu. Hlavná podmienka je potom splnená, ak predchádzajúce písmeno je buď v zozname letters „podpodmienky“ preceding alebo je jeho fonotyp v zozname signs tejto „podpodmienky“ a zároveň je nasledujúce písmeno (resp. fonotyp) v zozname letters (resp. signs) „podpodmienky“ following. Signs môže okrem SONOR, CONS, VOWEL obsahovať aj NONE, čo značí, že sa na danom mieste sa nenachádza žiadne písmeno (teda je to prvý alebo posledný znak slova).

2.2.3.3 Konfiguračné súbory webovej časti aplikácie

Každý z týchto súborov bude vo formáte **json**.

- *languages.json*: obsahuje priradenie jazyka zobrazeného užívateľovi ku konfiguračnému súboru tohto jazyka pomocou nasledujúcej štruktúry:

```
{
	"languages": [
		{
			"option": "Belarusian",
			"value": "conf_be_cyr"
		},
		...
	]
}
```

- *python_path_origin.json*: obsahuje príklad cesty k *python.exe*. Tento súbor je pre chod aplikácie potrebné skopírovať, premenovať na *python_path.json* a doň uviesť svoju cestu. Štruktúra súboru je nasledovná:

```
{
	"path" : "C:/Users/tomas/AppData/Local/Programs/Python/Python37/python.exe"
}
```

- *encodings*: obsahuje priradenie kódovania zobrazeného užívateľovi k hodnote používanej aplikáciou pomocou nasledujúcej štruktúry:

```
{
    "encodings": [
		{
			"option": "UTF-8",
			"value": "UTF-8-sig"
		},
		...
	]
}
```

2.2.3.4 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.

2.2.3.5 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_lengths_text.txt: Bude obsahovať iba číslice oddelené medzerami a pomlčkami.

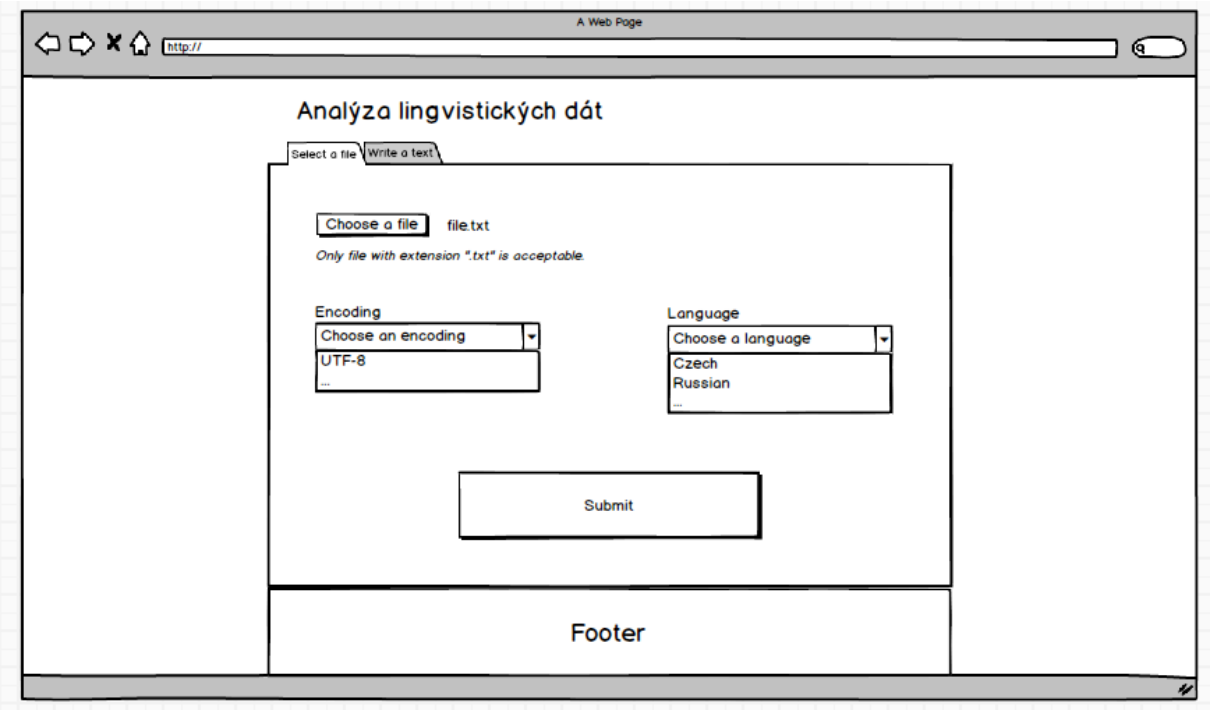
2.2.3.6 Výsledné súbory modulu Zápis tabuliek

Súbory budú vo formáte **xls**.

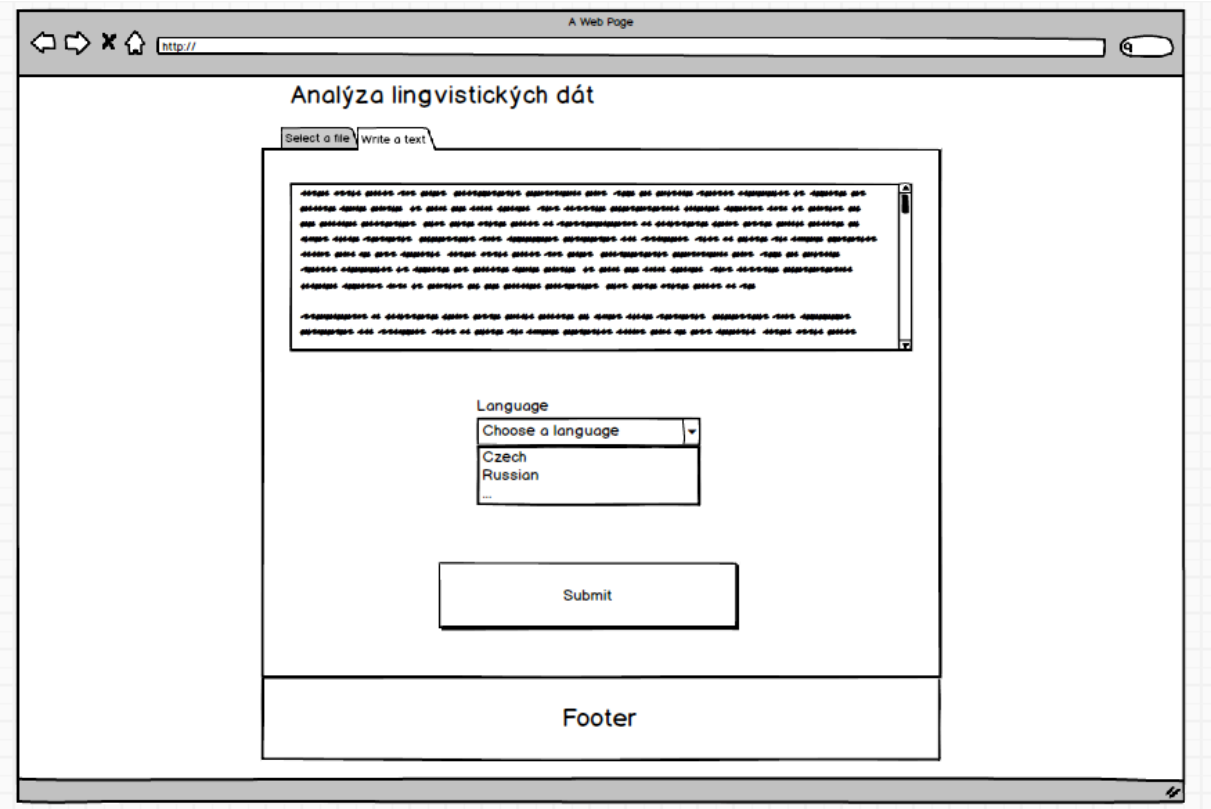
syllables_multiplicity.xls: Tabuľka, v ktorej prvý stĺpec obsahuje text slabiky, druhý stĺpec obsahuje dĺžku slabiky a tretí stĺpec obsahuje počet výskytov slabiky v texte. Slabiky sú usporiadané podľa početnosti zostupne.
count_of_length_of_syllables_with_repetition.xls*:* Tabuľka, v ktorej prvý stĺpec je dĺžka slabík a druhý stĺpec je počet slabík tejto dĺžky s opakovaním. Dĺžky sú usporiadané vzostupne.
count_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.

2.3 Používateľské rozhranie (obrázky)

Obr. 1: Mobilné rozhranie

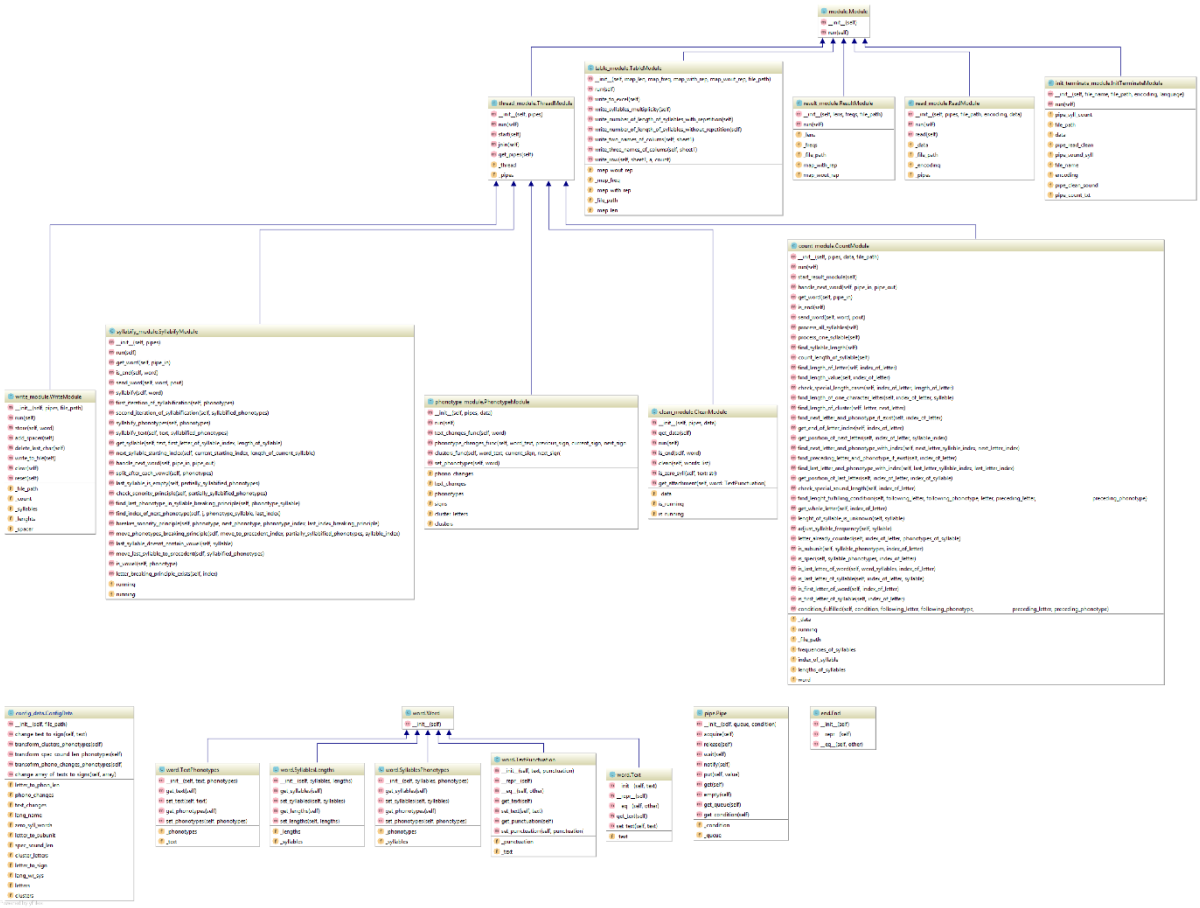


Obr. 2: Počítačové rozhranie - karta1

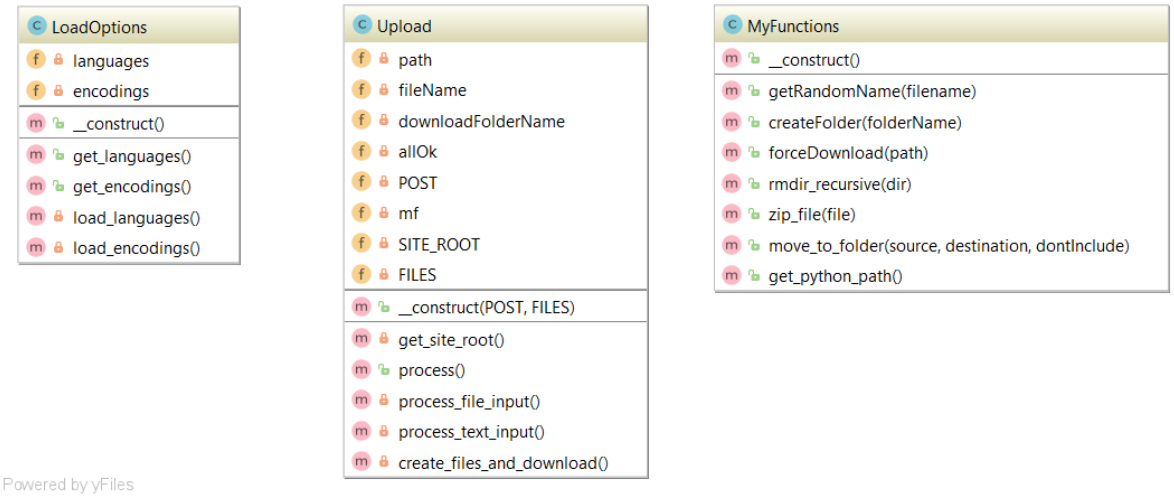


Obr. 3: Počítačové rozhranie - karta2

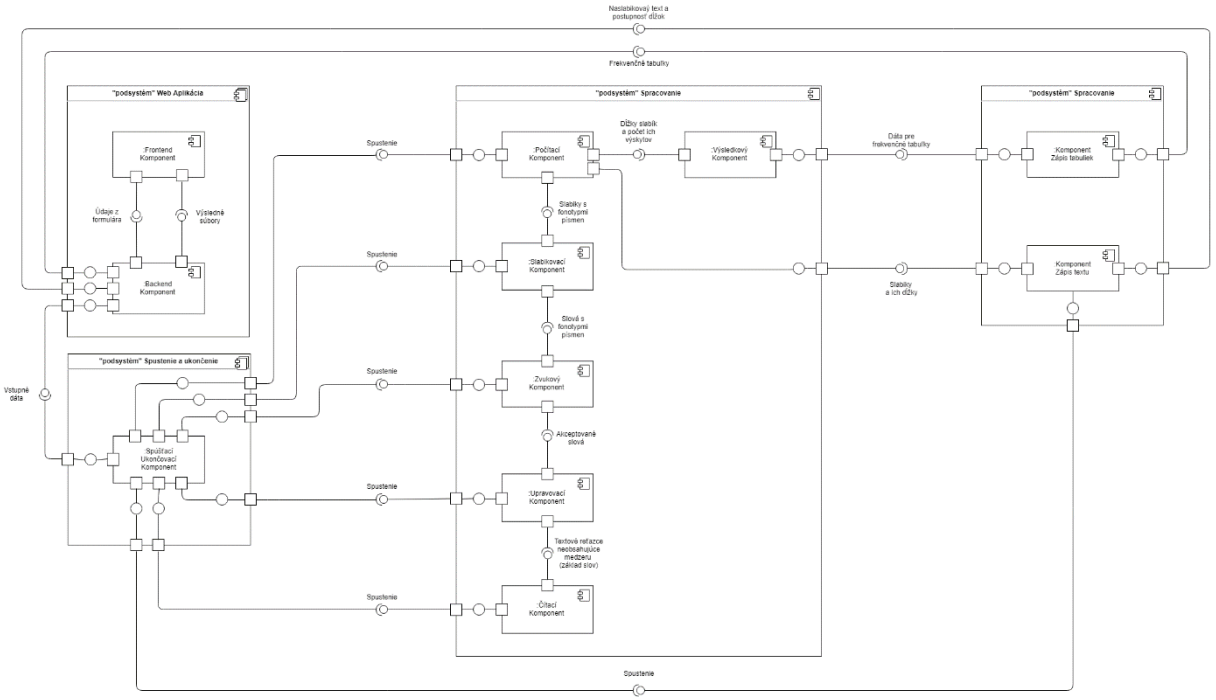
2.4 UML diagramy



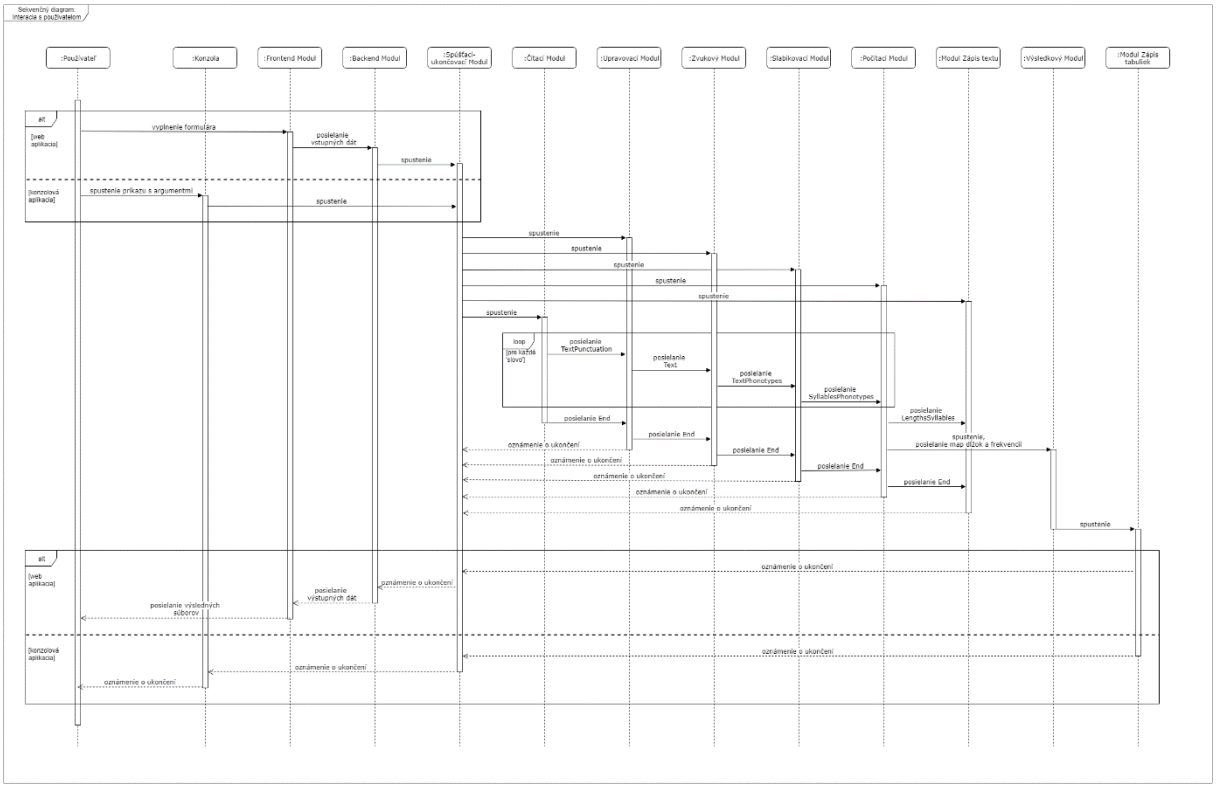
Obr. 4: Class diagram (triedy základu aplikácie)



Obr. 5: Class diagram (triedy webovej časti aplikácie)



Obr. 6: Component diagram



Obr. 7: Sequence diagram

2.5 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**
  - knižnica slúži na vytváranie a zapisovanie súborov s príponou .xls
  - zoznam hlavných príkazov:
    - “wb = Workbook()” - vytvorí sa lokálna tabuľka
    - “sheet1 = wb.add_sheet(‘Sheet 1’)” – pomenuje sa list tabuľky
    - “sheet1.write(row, column, ‘text’)” – zápis textu do bunky listu
    - “sheet1.write(row, column, Formula(‘Sum(A1:A10)’)” – zápis vzorca do listu
    - “wb.save(‘name.xls’, encoding=’UTF-8’)” – vytvorí/prepíše a uloží .xls súbor

3 Testovacia scenáre

3.1 Frontend modul

3.1.1 Vloženie súboru

Akcia: otvorenie aplikácie vo webovom prehliadači
Reakcia: Zobrazenie stránky

Akcia: kliknutie na tlacidlo "Choose a file"
Reakcia: Otvorenie okna, ktoré umožní nahrať súbor určený na spracovanie

Akcia: výber súboru a stlačenie tlačidla "Open"
Reakcia: cesta k súboru sa uloží do formuára a zobrazí sa

Akcia: vybratie kódovania pomocou selektu
Reakcia: vybraté kódovanie sa uloží do formulára

Akcia: vybratie jazyka pomocou selektu
Reakcia: vybratý jazyk sa uloží do formulára

Akcia: stlačenie tlačidla "Submit"
Reakcia: stiahnu sa zipnuté výsledné súbory

3.1.2 Vloženie textu

Akcia: Otvorenie aplikácie vo webovom prehliadači.
Reakcia: Zobrazenie stránky.

Akcia: Kliknutie na tlačidlo "Write a text".
Reakcia: Zobrazenie stránky kde sa text vkladá manuálne.

Akcia: Vloženie textu do textového poľa.
Reakcia: Text sa uloží do formulára.

Akcia: Vybratie jazyka pomocou selektu.
Reakcia: Vybratý jazyk sa uloží do formulára.

Akcia: Stlačenie tlačidla "Submit".
Reakcia: Stiahnu sa zipnuté výsledné súbory.

3.2 Backend modul

Akcia: Backendu prídu dáta pomocou metódy $_POST.
Reakcia: Modul spustí python skript, počká na jeho vykonanie, zozipuje súbory, ktoré boli vytvorené python skriptom a vynúti stiahnutie tohoto zip súboru.

3.3 Spúšťací a ukončovací modul

Akcia: Dostane dáta od backend modulu .
Reakcia: Inicializuje moduly a pipe-y, spustí ich a počká na ich úspešné ukončenie.

3.4 Čítací modul

3.4.1 Neexistujúci súbor

Akcia: Modul dostane: pole s prvkom typu Pipe pre výstup (pipe_out), adresu vstupného textu, jeho kódovanie, objekt triedy ConfigData, avšak súbor na danej adrese neexistuje.
Reakcia: Modul vloží do pipe_out objekt triedy End.

3.4.2 Čítanie textu v bieloruštine

Akcia: Modul dostane: pole s prvkom typu Pipe pre výstup (pipe_out), adresu vstupného textu, jeho kódovanie, objekt triedy ConfigData, pričom cstupný text súboru na danej adrese obsahuje text:
„ У беларускай мове зычныя могуць адрозніваццa даўжынёй гучання, якая пака-звае на стык марфем... Пераважная ‚колькасць‘ гукаў утвараюцца ў цэнтры ротавай поласці пры высокім агульным пад'ёме языка.
Вялікае Ducatus Lithuaniae знаходзілася ў дынастычнай уніі — з Польскім кара-леўствам!“
a vstupný objekt triedy ConfigData zodpovedá jazyku daného textu.
Reakcia: Modul postupne vloží do pipe_out objekty triedy TextPunctuation s hodnotami atribútov:

- _text:

  "У", "беларускай", "мове", "зычныя", "могуць", "адрозніваццa", "даўжынёй", "гучання,", "якая", "пака-звае", "на", "стык", "марфем...", "Пераважная", ‚колькасць‘", "гукаў", "утвараюцца", "ў", "цэнтры", "ротавай", "поласці", "пры", "высокім", "агульным", "пад'ёме", "языка.", "Вялікае", "Ducatus", " Lithuaniae", "знаходзілася", "ў", "дынастычнай", "уніі", "—", "з", "Польскім", "кара-леўствам!"

- _punctuation:

  Všeobecne k uvedeným hodnotám atribútov _text budú priradene hodnoty _punct, ktoré sú polom objektov None, a veľkosť poľa zodpovedá počtu symbolov.
  Okrem výnimiek, kde objektom s uvedenou hodnotou atribútu _text budú priradene takéto hodnoty atribútu _punct:
  "гучання," – [None, None, None, None, None, None, None, PUNCT];
  "пака-звае" – [None, None, None, None, HYPHEN, None, None, None, None];
  "марфем..." – [None, None, None, None, None, None, PUNCT, PUNCT, PUNCT];
  ‚колькасць‘" – [PUNCT, None, None, None, None, None, None, None, None, None, PUNCT];
  "языка." – [None, None, None, None, None, PUNCT];
  "—" – [PUNCT];
  "кара-леўствам!" – [None, None, None, None, HYPHEN, None, None, None, None, None, None, None, None, PUNCT].

Po skončení čítania textu súboru modul vloží do pipe_out objekt triedy End.

3.5 Upravovací modul

3.5.1 Objekt End na vstupe

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData
Modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.2 Úprava slova obsahujúceho veľké písmeno

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.

Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = "Київ“ a _punctuation = [None, None, None, None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul postupne vloží do pipe_out objekt triedy Text s atribútom _text = „київ“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.3 Úprava slova obsahujúceho písmená nepatriace abecede

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = '' foreign“ a _punctuation = [None, None, None, None, None, None, None]. V danom prípade písmená textu nepatria do jazyka uvedeného v objekte triedy ConfigData. Ďalej modul vyberie z pipe_in objekt triedy End
Reakcia: Modul slovo ignoruje. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.4 Úprava slova obsahujúce len interpunkčné znamienko

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = "—" a _punctuation = [PUNCT]. Ďalej modul vyberie z pipe_in objekt triedy End
Reakcia: Modul slovo ignoruje. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.5 Úprava slova s interpunkčným znamienkom

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = " слово!" a _punctuation = [None, None, None, None, None, PUNCT]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „слово“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.6 Úprava slova s interpunkčným znamienkom v strede

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = " сло!во" a _punctuation = [None, None, None, constants.PUNCT, None, None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul slovo ignoruje. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.7 Úprava slova v úvodzovkách

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = " «слово»" a _punctuation = [constants.PUNCT, None, None, None, None, None, constants.PUNCT]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „слово“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.8 Úprava slova so spojovníkom v strede

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = " сло-во" a _punctuation = [None, None, None, constants.HYPHEN, None, None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „слово“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.9 Úprava slova so spojovníkom na konci

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = " сло-" a _punctuation = [None, None, None, constants.HYPHEN]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul slovo ignoruje. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.10 Úprava slova so spojovníkom a interpunkčným znamienkom

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = "сло-во!" a _punctuation = [None, None, None, constants.HYPHEN, None, None, constants.PUNCT]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „слово“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.11 Úprava slova, ktoré nemá slabiku a sa spája s predchádzajúcim slovom

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in dva objekty triedy TextPunct. Prvý objekt má atribúty: _text = " українського" a _punctuation = [None, None, None, None, None, None, None, None, None, None, None, None]; druhý objekt má atribúty: _text = "ж" a _punctuation = [None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „українськогож“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.12 Úprava slova, ktoré nemá slabiku, je prvé v texte a sa spája s predchádzajúcim slovom

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = "ж" a _punctuation = [None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „ж“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.13 Úprava slova, ktoré nemá slabiku, začína veľkým písmenom a sa spája s predchádzajúcim slovom

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in dva obekty triedy TextPunct. Prvý objekt má atribúty: _text = " українського" a _punctuation = [None, None, None, None, None, None, None, None, None, None, None, None]; druhý objekt má atribúty: _text = "Ж" a _punctuation = [None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „українськогож“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.14 Úprava slova, ktoré nemá slabiku a sa spája s nasledujúcim slovom

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in dva obekty triedy TextPunct. Prvý objekt má atribúty: _text = "з" a _punctuation = [None]; druhý objekt má atribúty: _text = "собою" a _punctuation = [None, None, None, None, None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „зсобою“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.15 Úprava slova, ktoré nemá slabiku, je posledné v texte a sa spája s nasledujúcim slovom

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in objekt triedy TextPunct, ktorý má atribúty: _text = "з" a _punctuation = [None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „з“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.16 Úprava slova, ktoré nemá slabiku, začína veľkým písmenom a sa spája s nasledujúcim slovom

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in dva obekty triedy TextPunct. Prvý objekt má atribúty: _text = "З" a _punctuation = [None]; druhý objekt má atribúty: _text = "собою" a _punctuation = [None, None, None, None, None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „зсобою“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.17 Kontrola pre slovo, ktoré má symbol 'APOSTROPHE' (U+0027)

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in obekt triedy TextPunct, ktorý má atribúty: _text = "в'ю" a _punctuation = [None, None, None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „в'ю“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.18 Kontrola pre slovo, ktoré má symbol 'RIGHT SINGLE QUOTATION MARK' (U+2019)

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in obekt triedy TextPunct, ktorý má atribúty: _text = "в’ю" a _punctuation = [None, None, None]. Ďalej modul vyberie z pipe_in objekt triedy End.
Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „в’ю“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.5.19 Kontrola pre slovo, ktoré má symbol 'MODIFIER LETTER APOSTROPHE' (U+02BC)

Akcia: Modul dostane pole s dvoma prvkami typu Pipe pre vstup (pipe_in) a výstup (pipe_out), objekt triedy ConfigData.
Modul vyberie z pipe_in obekt triedy TextPunct, ktorý má atribúty: _text = " в’ю" a _punctuation = [None, None, None]. Ďalej modul vyberie z pipe_in objekt triedy End.

Reakcia: Modul vloží do pipe_out objekt triedy Text s atribútom _text = „в’ю“. Ďalej modul vloží do pipe_out objekt triedy End a ukončí svoju prácu.

3.6 Zvukový modul

3.6.1 Testovanie deliteľnosti „ou“ (except v ConfigData clusters)

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "pouze".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "pouze" a atribút p_phonotypes obsahujúci [CONS, SUBUNIT, VOWEL, CONS, VOWEL].

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "použil".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "použil" a atribút p_phonotypes obsahujúci [CONS, VOWEL, VOWEL, CONS, VOWEL, SONOR].

3.6.2 Testovanie deliteľnosti „eu“ (only v CofingData clusters)

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "farmaceutický".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "farmaceutický" a atribút p_phonotypes obsahujúci [CONS, VOWEL, SONOR, SONOR, VOWEL, CONS, SUBUNIT, VOWEL, CONS, VOWEL, CONS, CONS, VOWEL].

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "neurčitý".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "neurčitý" a atribút p_phonotypes obsahujúci [SONOR, VOWEL, VOWEL, SONOR, CONS, VOWEL, CONS, VOWEL].

3.6.3 Testovanie nedeliteľnosti „au“

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "automatický".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "automatický" a atribút p_phonotypes obsahujúci [SUBUNIT, VOWEL, CONS, VOWEL, SONOR, VOWEL, CONS, VOWEL, COSN, CONS, VOWEL].

3.6.4 Testovanie zmeny fonotypu (phono_changes v ConfigData)

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "vlna".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "vlna" a atribút p_phonotypes obsahujúci [CONS, VOWEL, SONOR, VOWEL].

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "osm".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "osm" a atribút p_phonotypes obsahujúci [VOWEL, CONS, VOWEL].

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "wrobl".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "wrobl" a atribút p_phonotypes obsahujúci [SONOR, SONOR, VOWEL, CONS, VOWEL].

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "rž".
Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "rž" a atribút p_phonotypes obsahujúci [VOWEL, CONS].

3.6.5 Testovanie slova po zmene textu (text_changes v ConfigData)

Akcia: Modul dostane objekt typu Text, ktorého atribút _text obsahuje string "biologie ".

Reakcia: Modul vráti objekt typu TextPhonotypes, ktorého atribút _text obsahuje string "bijologije" a atribút p_phonotypes obsahujúci [CONS, VOWEL, SONOR, VOWEL, SONOR, VOWEL, CONS, VOWEL, SONOR, VOWEL].

3.7 Slabikovací modul

3.7.1 Testovanie deliteľnosti „ou“ (except v ConfigData clusters)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _text = “pouze”, _phonotypes = [CONS, SUBUNIT, VOWEL, CONS, VOWEL].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['pou', 'ze'], _phonotypes = [[CONS, SUBUNIT, VOWEL], [CONS, VOWEL]].

Akcia: Modul vyberie z pipe_in objekt s atribútmi _text = “použil”, _phonotypes = [CONS, VOWEL, VOWEL, CONS, VOWEL, SONOR].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['po', 'u', 'žil'], _phonotypes = [[CONS, VOWEL], [VOWEL], [CONS, VOWEL, SONOR]].

3.7.2 Testovanie deliteľnosti „eu“ (only v CofingData clusters)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _text = “farmaceutický”, _phonotypes = [CONS, VOWEL, SONOR, SONOR, VOWEL, CONS, SUBUNIT, VOWEL, CONS, VOWEL, CONS, CONS, VOWEL].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['fa', 'rma', 'ceu', 'ti', 'cký'], _phonotypes = [[CONS, VOWEL], [SONOR, SONOR, VOWEL], [CONS, SUBUNIT, VOWEL], [CONS, VOWEL], [CONS, CONS, VOWEL]].

Akcia: Modul vyberie z pipe_in objekt s atribútmi _text = “neurčitý”, _phonotypes = [SONOR, VOWEL, VOWEL, SONOR, CONS, VOWEL, CONS, VOWEL].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['ne', 'ur', 'či', 'tý'], _phonotypes = [[SONOR, VOWEL], [VOWEL, SONOR], [CONS, VOWEL], [CONS, VOWEL]].

3.7.3 Testovanie nedeliteľnosti „au“

Akcia: Modul vyberie z pipe_in objekt s atribútmi _text = automatický “”, _phonotypes = [SUBUNIT, VOWEL, CONS, VOWEL, SONOR, VOWEL, CONS, VOWEL, CONS, CONS, VOWEL].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['au', 'to', 'ma', 'ti', 'cký'], _phonotypes = [[SUBUNIT, VOWEL], [CONS, VOWEL], [SONOR, VOWEL], [CONS, VOWEL], [CONS, CONS, VOWEL]].

3.7.4 Testovanie zmeny fonotypu (phono_changes v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _text = “vlna”, _phonotypes = [CONS, VOWEL, SONOR, VOWEL].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['vl', 'na'], _phonotypes = [[CONS, VOWEL], [SONOR, VOWEL]].

3.7.5 Testovanie slova po zmene textu (text_changes v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _text = “bijologije”, _phonotypes = [CONS, VOWEL, SONOR, VOWEL, SONOR, VOWEL, CONS, VOWEL, SONOR, VOWEL].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['bi', 'jo', 'lo', 'gi', 'je'], _phonotypes = [[CONS, VOWEL], [SONOR, VOWEL], [SONOR, VOWEL], [CONS, VOWEL], [SONOR, VOWEL]].

3.7.6 Testovanie objektu typu End

Akcia: Modul vyberie z pipe_in objekt typu End.
Reakcia: Modul vloží do pipe_out získaný objekt a ukončí svoju prácu.

3.8 Počítací modul

3.8.1 Testovanie dĺžky „ou“ (except v ConfigData clusters)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['pou', 'ze'], _phonotypes = [[CONS, SUBUNIT, VOWEL], [CONS, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['pou', 'ze'], _lengths = [2, 2].

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['po', 'u', 'žil'], _phonotypes = [[CONS, VOWEL], [VOWEL], [CONS, VOWEL, SONOR]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['po', 'u', 'žil'], _lengths = [2, 1, 3].

3.8.2 Testovanie dĺžky „eu“ (only v CofingData clusters)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['fa', 'rma', 'ceu', 'ti', 'cký'], _phonotypes = [[CONS, VOWEL], [SONOR, SONOR, VOWEL], [CONS, SUBUNIT, VOWEL], [CONS, VOWEL], [CONS, CONS, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['fa', 'rma', 'ceu', 'ti', 'cký'], _lengths = [2, 3, 2, 2, 3].

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['ne', 'ur', 'či', 'tý'], _phonotypes = [[SONOR, VOWEL], [VOWEL, SONOR], [CONS, VOWEL], [CONS, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['ne', 'ur', 'či', 'tý'], _lengths = [2, 2, 2, 2].

3.8.3 Testovanie dĺžky „au“

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['au', 'to', 'ma', 'ti', 'cký'], _phonotypes = [[SUBUNIT, VOWEL], [CONS, VOWEL], [SONOR, VOWEL], [CONS, VOWEL], [CONS, CONS, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['au', 'to', 'ma', 'ti', 'cký'], _lengths = [1, 2, 2, 2, 3].

3.8.4 Testovanie dĺžky po zmene fonotypu (phono changes v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['vl', 'na'], _phonotypes = [[CONS, VOWEL], [SONOR, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['vl', 'na'], _lengths = [2, 2].

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['o', 'sm'], _phonotypes = [[VOWEL], [CONS, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['o', 'sm'], _lengths = [1, 2].

3.8.5 Testovanie dĺžky slova po zmene textu (text_changes v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['bi', 'jo', 'lo', 'gi', 'je'], _phonotypes = [[CONS, VOWEL], [SONOR, VOWEL], [SONOR, VOWEL], [CONS, VOWEL], [SONOR, VOWEL]].

Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['bi', 'jo', 'lo', 'gi', 'je'], _lengths = [2, 2, 2, 2, 2].

3.8.6 Testovanie slova obsahujúceho 0-slabičné slova (zero_syll_words v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['sro', 'zpa', 'dem'], _phonotypes = [[CONS, SONOR, VOWEL], [CONS, CONS, VOWEL], [CONS, VOWEL, SONOR]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['sro', 'zpa', 'dem'], _lengths = [3, 3, 3].

3.8.7 Testovanie špeciálnych prípadov dĺžky písmen (spec_sound_len v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['vě', 'do', 'mí'], _phonotypes = [[CONS, VOWEL], [CONS, VOWEL], [SONOR, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['vě', 'do', 'mí'], _lengths = [3, 2, 2].

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['sex'], _phonotypes = [[CONS, VOWEL, CONS]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['sex'], _lengths = [4].

3.8.8 Testovanie vytvorenia máp frekvencie a dĺžky slabík a získania objektu typu End

Akcia: Modul vyberie z pipe_in postupne všetky objekty uvedené v testoch 8.1 - 8.7 a následne objekt typu End.
Reakcia: Modul ukončí svoju prácu a zavolá Výsledkový modul s agumentami map_len = {'au': 1, 'bi': 2, 'ceu': 2, 'cký': 3, 'dem': 3, 'do': 2, 'fa': 2, 'gi': 2, 'je': 2, 'jo': 2, 'lo': 2, 'ma': 2, 'mí': 2, 'na': 2, 'ne': 2, 'o': 1, 'po': 2, 'pou': 2, 'rma': 3, 'sex': 4, 'sm': 2, 'sro': 3, 'ti': 2, 'to': 2, 'tý': 2, 'u': 1, 'ur': 2, 'vl': 2, 'vě': 3, 'ze': 2, 'či': 2, 'zpa': 3, 'žil': 3}, map_freq = {'au': 1, 'bi': 1, 'ceu': 1, 'cký': 2, 'dem': 1, 'do': 1, 'fa': 1, 'gi': 1, 'je': 1, 'jo': 1, 'lo': 1, 'ma': 1, 'mí': 1, 'na': 1, 'ne': 1, 'o': 1, 'po': 1, 'pou': 1, 'rma': 1, 'sex': 1, 'sm': 1, 'sro': 1, 'ti': 2, 'to': 1, 'tý': 1, 'u': 1, 'ur': 1, 'vl': 1, 'vě': 1, 'ze': 1, 'či': 1, 'zpa': 1, 'žil': 1}, file_path (posunie argument, ktorý dostal pri vytvorení).

3.8.9 Testovanie zamlčania písmena (spec_sound_len v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi atribútmi _syllables = ['sněh'], _phonotypes = [[CONS, SONOR, VOWEL, CONS]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['sněh'], _lengths = [3].

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['sa', 'hać'], _phonotypes = [[CONS, VOWEL,], [CONS, VOWEL, CONS]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['sa', 'hać'], _lengths = [2, 3].

3.8.10 Testovanie zmeny dĺžky dvojice vzhľadom na predchádzajúce písmeno (spec_sound_len pre clusters v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['ma', 'ria'], _phonotypes = [[SONOR, VOWEL], [SONOR, SUBUNIT, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['ma', 'ria'], _lengths = [2, 3].

3.8.11 Testovanie zmeny dĺžky dvojice vzhľadom na nasledujúcu dvojicu (spec_sound_len pre clusters v ConfigData)

Akcia: Modul vyberie z pipe_in objekt s atribútmi _syllables = ['mię', 'dzy'], _phonotypes = [[SONOR, SUBUNIT, VOWEL], [SUBUNIT, CONS, VOWEL]].
Reakcia: Modul vloží do pipe_out objekt s atribútmi _syllables = ['mię', 'dzy'], _lengths = [2, 2]

3.9 Výsledkový modul

Akcia: Modul vyberie zo vstupných argumentov objekty
map_len = { 'au': 1, 'bi': 2, 'ceu': 2, 'cký': 3, 'dem': 3, 'do': 2, 'fa': 2, 'gi': 2, 'je': 2, 'jo': 2, 'lo': 2, 'ma': 2, 'mí': 2, 'na': 2, 'ne': 2, 'o': 1, 'po': 2, 'pou': 2, 'rma': 3, 'sex': 4, 'sm': 2, 'sro': 3, 'ti': 2, 'to': 2, 'tý': 2, 'u': 1, 'ur': 2, 'vl': 2, 'vě': 3, 'ze': 2, 'či': 2, 'zpa': 3, 'žil': 3},
map_freq = {'au': 1, 'bi': 1, 'ceu': 1, 'cký': 2, 'dem': 1, 'do': 1, 'fa': 1, 'gi': 1, 'je': 1, 'jo': 1, 'lo': 1, 'ma': 1, 'mí': 1, 'na': 1, 'ne': 1, 'o': 1, 'po': 1, 'pou': 1, 'rma': 1, 'sex': 1, 'sm': 1, 'sro': 1, 'ti': 2, 'to': 1, 'tý': 1, 'u': 1, 'ur': 1, 'vl': 1, 'vě': 1, 'ze': 1, 'či': 1, 'zpa': 1, 'žil': 1}.
Reakcia: Modul zavolá modul Zápis tabuliek s argumentmi:
map_len = { 'au': 1, 'bi': 2, 'ceu': 2, 'cký': 3, 'dem': 3, 'do': 2, 'fa': 2, 'gi': 2, 'je': 2, 'jo': 2, 'lo': 2, 'ma': 2, 'mí': 2, 'na': 2, 'ne': 2, 'o': 1, 'po': 2, 'pou': 2, 'rma': 3, 'sex': 4, 'sm': 2, 'sro': 3, 'ti': 2, 'to': 2, 'tý': 2, 'u': 1, 'ur': 2, 'vl': 2, 'vě': 3, 'ze': 2, 'či': 2, 'zpa': 3, 'žil': 3},
map_freq = {'au': 1, 'bi': 1, 'ceu': 1, 'cký': 2, 'dem': 1, 'do': 1, 'fa': 1, 'gi': 1, 'je': 1, 'jo': 1, 'lo': 1, 'ma': 1, 'mí': 1, 'na': 1, 'ne': 1, 'o': 1, 'po': 1, 'pou': 1, 'rma': 1, 'sex': 1, 'sm': 1, 'sro': 1, 'ti': 2, 'to': 1, 'tý': 1, 'u': 1, 'ur': 1, 'vl': 1, 'vě': 1, 'ze': 1, 'či': 1, 'zpa': 1, 'žil': 1},
map_with_rep = {1: 3, 2: 23, 3: 8, 4: 1},
map_wout_rep = {1: 3, 2: 22, 3: 7, 4: 1},
file_path (posunie argument, ktorý dostal pri vytvorení).

3.10 Zápis tabuliek

Akcia: Modul vyberie zo vstupných argumentov objekty:
map_len = {'au': 1, 'bi': 2, 'ceu': 2, 'cký': 3, 'dem': 3, 'do': 2, 'fa': 2, 'gi': 2, 'je': 2, 'jo': 2, 'lo': 2, 'ma': 2, 'mí': 2, 'na': 2, 'ne': 2, 'o': 1, 'po': 2, 'pou': 2, 'rma': 3, 'sex': 4, 'sm': 2, 'sro': 3, 'ti': 2, 'to': 2, 'tý': 2, 'u': 1, 'ur': 2, 'vl': 2, 'vě': 3, 'ze': 2, 'či': 2, 'zpa': 3, 'žil': 3},
map_freq = {'au': 1, 'bi': 1, 'ceu': 1, 'cký': 2, 'dem': 1, 'do': 1, 'fa': 1, 'gi': 1, 'je': 1, 'jo': 1, 'lo': 1, 'ma': 1, 'mí': 1, 'na': 1, 'ne': 1, 'o': 1, 'po': 1, 'pou': 1, 'rma': 1, 'sex': 1, 'sm': 1, 'sro': 1, 'ti': 2, 'to': 1, 'tý': 1, 'u': 1, 'ur': 1, 'vl': 1, 'vě': 1, 'ze': 1, 'či': 1, 'zpa': 1, 'žil': 1},
map_with_rep = {1: 3, 2: 23, 3: 8, 4: 1},
map_wout_rep = {1: 3, 2: 22, 3: 7, 4: 1},
file_path = 'outputs/'.
Reakcia: Vytvoril sa súbor 'syllables_multiplicity.xls' v priečinku 'outputs/' a jeho obsah je:

syllable	multiplicity	length of syllable
cký	2	3
ti	2	2
au	1	1
bi	1	2
ceu	1	2
dem	1	3
do	1	2
fa	1	2
gi	1	2
je	1	2
jo	1	2
lo	1	2
ma	1	2
mí	1	2
na	1	2
ne	1	2

o	1	1
po	1	2
pou	1	2
rma	1	3
sex	1	4
sm	1	2
sro	1	3
to	1	2
tý	1	2
u	1	1
ur	1	2
vl	1	2
vě	1	3
ze	1	2
zpa	1	3
či	1	2
žil	1	3

Vytvoril sa súbor 'number_of_length_of_syllables_with_repetition.xls' v priečinku 'outputs/' a jeho obsah je:

length of syllable	multiplicity
1	3
2	23
3	8
4	1

Vytvoril sa súbor 'number_of_length_of_syllables_without_repetition.xls' v priečinku 'outputs/' a jeho obsah je:

length of syllable	multiplicity
2	22
3	7
1	3
4	1

3.11 Zápis textu

3.11.1 Testovanie získania samotného slova (pred ním nebolo získané nič)

Akcia: Modul získa zo vstupných argumentov objekt file_path = 'outputs/'. Modul vyberie z pipe_in jediný objekt s atribútmi _syllables = [[vl, na]], _lengths = [[2, 2]].
Reakcia: Nenastala žiadna reakcia.

3.11.2 Testovanie získania slov a následne objektu typu End

Akcia: Modul získa zo vstupných argumentov objekt file_path = 'outputs/'. Modul vyberie z pipe_in 3 objekty:
1. má atribúty _syllables = ['pou', 'ze'] a _lengths = [2, 2],
2. má atribúty _syllables = ['fa', 'rma', 'ceu', 'ti', 'cký'] a _lengths = [2, 3, 2, 2, 3],
3. je objekt typu End (nemá žiadne atribúty)
Reakcia: Vytvoril sa súbor 'syllable_text.txt' v priečinku 'outputs/' a jeho obsah je:
pou-ze fa-rma-ceu-ti-cký
Vytvoril sa súbor 'syllable_lengths_text.txt' v priečinku 'outputs/' a jeho obsah je:
2-2 2-3-2-2-3

3.11.3 Testovanie získania 1000-ceho slova

Akcia: Modul získa zo vstupných argumentov objekt file_path = 'outputs/'. Modul vyberie z pipe_in 1000 objektov, pričom každý z nich má atribúty _syllables = [[sněh]], _phonotypes = [[3]].
Reakcia: Vytvoril sa súbor 'syllable_text.txt' v priečinku 'outputs/' a jeho obsah je (1000 krát slovo 'sněh'):
sněh sněh ... sněh
Vytvoril sa súbor 'syllable_lengths_text.txt' v priečinku 'outputs/' a jeho obsah je (1000 krát znak '3'):
3 3 ... 3

3.12 Celkový test aplikácie

3.12.1 Test vstupu cez web

Akcia: Používateľ pomocou webového rozhrania nahrá súbor s obsahom:
Republika Hrvatska je europska država, u geopolitičkom smislu srednjoeuropska i sredozemna država, a zemljopisno smještena u južnom dijelu Srednje Europe te u sjevernom dijelu Sredozemlja. Na sjeveru graniči sa Slovenijom i Mađarskom, na istoku sa Srbijom i Bosnom i Hercegovinom, na jugu s Crnom Gorom, dok na zapadu s Italijom ima morsku granicu.
Reakcia: Používateľ bude môcť stiahnuť zip súbor, ktorý bude okrem frekvenčných tabuliek obsahovať aj textové súbory

- ‘syllable_text.txt‘ s obsahom:
  re-pu-bli-ka hr-va-tska je e-u-ro-pska dr-ža-va u ge-o-po-li-ti-čkom smi-slu sre-dnjo-e-u-ro-pska i sre-do-ze-mna dr-ža-va a ze-mljo-pi-sno smje-šte-na u ju-žnom di-je-lu sre-dnje e-u-ro-pe te u sje-ve-rnom di-je-lu sre-do-ze-mlja na sje-ve-ru gra-ni-či sa slo-ve-ni-jom i ma-đar-skom na i-sto-ku sa sr-bi-jom i bo-snom i her-ce-go-vi-nom na ju-gu scr-nom go-rom dok na za-pa-du si-ta-li-jom i-ma mor-sku gra-ni-cu
- a ‘syllable_lengths_text.txt‘ s obsahom:
  2-2-3-2 2-2-4 2 1-1-2-4 2-2-2 1 2-1-2-2-2-4 3-3 3-3-1-1-2-4 1 3-2-2-3 2-2-2 1 2-3-2-3 4-3-2 1 2-4 2-2-2 3-3 1-1-2-2 2 1 3-2-4 2-2-2 3-2-2-3 1 3-2-2 3-2-2 1 3-2-2-3 1 2-3-4 1 1-3-2 2 2-2-3 1 2-4 1 3-2-2-2-3 2 2-2 3-3 2-3 3 2 2-2-2 2-2-2-3 1-2 3-3 3-2-2

3.12.2 Test vstupu cez konzolu

Akcia: Používateľ pomocou konzoly dá spracovať súbor s obsahom:
Россия, официально также Российская Федерация — государство в Восточной Европе, Центральной и Северной Азии. Территория России в рамках её конституционного устройства составляет 17 125 191 км²; население страны (в пределах её заявленной территории) составляет 146 880 432 [чел. (2018). Занимает первое место в мире по территории, шестое — по объёму ВВП по ППС и девятое — по численности населения. Столица — Москва. Государственный язык — русский.
Reakcia: V používateľom zadanom priečinku (pri spúšťaní) sa vytvoria okrem frekvenčných tabuliek aj textové súbory

- ‘syllable_text.txt‘ s obsahom:
  ро-сси-я о-фи-ци-а-льно та-кже ро-ссий-ска-я фе-де-ра-ци-я го-су-дар-ство вво-сто-чной е-вро-пе цен-тра-льной и се-ве-рной а-зи-и те-рри-то-ри-я ро-сси-и врам-ках е-ё кон-сти-ту-ци-о-нно-го у-строй-ства со-ста-вля-ет на-се-ле-ни-е стра-ны впре-де-лах е-ё за-я-вле-нной те-рри-то-ри-и со-ста-вля-ет за-ни-ма-ет пер-во-е ме-сто вми-ре по те-рри-то-ри-и ше-сто-е по о-бъё-му по и де-вя-то-е по чи-сле-нно-сти на-се-ле-ни-я сто-ли-ца мо-сква го-су-дар-стве-нный я-зык ру-сский
- a ‘syllable_lengths_text.txt‘ s obsahom:
  2-3-2 1-2-2-1-3 2-3 2-4-3-2 2-2-2-2-2 2-2-3-4 3-3-4 2-3-2 3-3-4 1 2-2-4 1-2-1 2-3-2-2-2 2-3-1 4-3 2-2 3-3-2-2-1-3-2 1-5-4 2-3-3-3 2-2-2-2-2 4-2 4-2-3 2-2 2-2-3-4 2-3-2-2-1 2-3-3-3 2-2-2-3 3-2-2 2-3 3-2 2 2-3-2-2-1 2-3-2 2 1-3-2 2 1 2-2-2-2 2 2-3-3-3 2-2-2-2-2 3-2-2 2-4 2-2-3-4-4 2-3 2-5