Micro Commander

Dokumentacja deweloperska

Język programowania

Projekt został wykonany w języku Python 3.5, wykorzystując biblioteki:

- pytest
- pytest-mock
- sphinx
- sphinx-rtd-theme
- pynput
- PyQt5
- python-xlib
- SpeechRecognition
- PyAudio

oraz googlowskim modułem przetwarzania sygnału głosowego na stringi

Struktura projektu

Cały kod źródłowy znajduje się w folderze src. Można tam znaleźć wszystkie moduły funkcjonalne oraz serce aplikacji – app.py, gdzie klasa Application odpowiada za załadowanie wszystkich modułów, inicjalizację GUI oraz uruchomienie.

Kod wydzielony jest na moduły odpowiadające za różne funkcjonalności. Każdy moduł jest traktowany jako osobna paczka pythonowa importowana do głównej aplikacji. Wyróżniamy moduły:

Speech

Moduł odpowiedzialny za przetwarzanie sygnału głosowego na stringi (Google API)

Converter - Nasłuchuje sygnału wejściowego

Converter_thread - klasa AudioManager zapewnia wykonywanie operacji na osobnym wątku

Recorder - Nagrywa dźwięk i umieszcza go w kolejce

Response queue - Kolejkuje otrzymane odpowiedzi, dba o to, by najwcześniejszy request był na samej górze

Analyser

Analyser – odbiera string wychwycony przez moduł speech i dobiera do niego odpowiedni serwis, w zależności od pierwszego wyłapanego słowa. Następne słowo traktowane jest jako argument.

Do mapowania serwisów wykorzystywany jest słownik znajdujący się w settings pod kluczem ‘servicemapping’. Defaultowo wygląda on następująco:

```
'servicemapping': {
        'komenda': 'CommandService',
        'uruchom': 'ExecutionService',
        'usuń': 'BackspaceService',
        'ustawienia': 'SettingsService'
    }
```

Moduł został napisany w taki sposób, aby można go było dowolnie rozszerzać o kolejne serwisy.

Tworzenie nowego serwisu:

1. Dodajemy plik ./src/analyser/nazwaservisu.py
2. W stworzonym pliku implementujemy klasę NazwaServisu, która musi implementować następujący interfejs:
   - Musi dziedziczyć po InsertService
   - Musi udostępniać publiczną metodę ‘process(text)’, gdzie ‘text’ to string przekazany z modułu rozpoznawania mowy, który będzie poddany analizie
   - W konstruktorze przekazywana jest referencja do klasy AnalyserSetting (omawiana poniżej).
3. W pliku ./src/conf/default_config.py do słownika 'servicemapping' dodajemy mapowanie - które słowo triggeruje nasz nowo stworzony serwis (np. ‘serwis’ : ‘NazwaServisu’) - w ten sposób, gdy użytkownik rozpocznie zdanie od słowa ‘serwis’, będzie ono przetwarzane przy pomocy nowego serwisu

Serwisy zaimplementowane podczas tworzenia aplikacji:

InsertService - jest to serwis używany domyślnie, tzn. gdy pierwsze słowo nie pasuje do żadnego z kluczy w słowniku ‘servicemapping’. Służy on głównie do pisania na klawiaturze

Commandservice - serwis używany, gdy pierwsze słowo to ‘komenda’. Służy do wykonywania zdefiniowanych wcześniej komend znajdujących się w słownikach ‘commands’ dla danych aplikacji.

Przykładowa konfiguracja komend dla przeglądarek:

```
'browser': {

    'commands': {

        'nowa karta': [(Key.ctrl, 't')],

        'zamknij kartę': [(Key.ctrl, 'w')],

        'odśwież': [Key.f5],

        'incognito': [(Key.ctrl, Key.shift, 'n')]

    }

}
```

Jest to lista klawiszy, które mają zostać naciśnięte w odpowiedniej kolejności. Jeżeli elementem listy jest tupla, to przyciski w niej zawarte mają zostać naciśnięte jednocześnie.

ExecutionService – serwis triggerowany słowem ‘uruchom’. Po tym słowie kluczowym należy podać klucz znajdujący się w słowniku 'execute'.

Podstawowy słownik wygląda następująco:

```
'execute': {
    'edytor kodu': ['/snap/bin/code'],
    'przeglądarka': ['chromium-browser'],
    'youtube' : ['chromium-browser', 'youtube.com']
}
```

Pierwszym elementem listy jest program, który ma być uruchomiony, natomiast kolejne elementy listy to argumenty, które będą użyte przy wykonaniu programu. Argumenty można również przekazywać mówiąc je bezpośrednio.

BackspaceService - triggerowany słowem ‘usuń’. Serwis służący do usuwania znaków (dokładne działanie opisane w dokumentacji użytkownika)

AnalyserSettings – pozwala na włączenie/wyłączenie autospacji, ustawienie automatycznych, dużych lub małych liter, oraz włączanie i wyłączanie mikrofonu (dokładne działanie również opisane w dokumentacji użytkownika)

GUI

Moduł odpowiedzialny za graficzny interfejs użytkownika zaimplementowanym w bibliotece PyQt5. Wyróżniamy w nim komponenty:

Add_mode_window – Okno odpowiedzialne za dodawanie nowych settingsow

Main_window – Okno główne, w którym można przejrzeć i edytować komendy, po zatwierdzeniu uruchamiany jest walidator, który za pomocą parsera sprawdza poprawność edytowanych ustawień.

Key Input

Moduł odpowiedzialny za symulowanie inputu klawiaturowego

Diacritic mapping - Zapewnia funkcjonalność prawego altu (np. Polskie znaki)

Input simulator - Obsługuje kombinacje klawiszy (np. Ctrl+s), symuluje input klawiatury

Window manager

Odpowiedzialny za rozpoznawanie okna, w którym się obecnie znajdujemy. W zależności od wykrytego programu dobiera do niego odpowiednie settingsy (załadowuje odpowiedni słownik)

*Można dodawać własne komendy w app mapping

Xorg – implementacja windowmanagera w systemie linux

Ustawienia

Wszystkie moduły korzystają z jednolitych ustawień, opartych na wzorcu projektowym Singleton oraz Observer. Klasa Settings dziedziczy po UserDict, przez co może być importowana jako słownik.

Debugowanie

W celu debugowania wystarczy wyświetlić pożądane informacje na konsoli. W tym celu należy zaimportować moduł logger, wywołać odpowiednią metodę dla poziomu debugu (info, exception, error) i przesłać w argumencie wiadomość, która ma się wyświetlić.

Przykład:

```
import logger

logger.info("Example log")

try:

      raise KeyError()

except:

      logger.exception("Opening configuration failed"
```

Dalszy rozwój projektu

Projekt można rozwijać na wiele sposobów. Dzięki uniwersalności googlowskiego modułu, jego zakres działania można rozszerzyć o urządzenia mobilne oraz inne systemy operacyjne. W przyszłości użytkownik mógłby sam „nagrywać” akcje, np. Klikając myszką wywołując odpowiednie zdarzenia jakie mają się wykonywać w systemie i podpinać pod nie własne komendy. Nic nie stoi również na przeszkodzie, by wspierać inne rodzaje klawiatur lub inne języki mowy.