Нов Български Университет

Бакалавърски факултет

Департамент “Информатика”

Курсов проект

Курс:

CITB579 Практика по програмиране и интернет технологии

Тема:

Проект за интелигентен личен асистент реализиран с wxPython

Изготвил:
Тихомир Младенов

Проверил:
гл. ас. д-р Филип Андонов

Специалност:
Бизнес Информатика
Курс: Четвърти
Фак. Н: F-61480

НБУ София, юли 2019 г.

Съдържание

1. Описание на заданието

Целта на заданието е да се разработи интелигентен личен асистент с помощта на wxPython, което е обвивка написана на Python върху wxWidgets. Така решението ще може да работи под Линукс, Уиндоус, Мак ОС с минимални усилия.

Решението трябва да съдържа в себе си функционалност позволяваща на потребителя да чете новини, да получава и чете имейли, да разглежда социалните мрежи, да съобщава текущото метеорологично време, текущата дата и час, да има аларма и да възпроизвежда говор от текста в него без да ползва онлайн източници.

Интерфейсът трябва да е интуитивен и лесен за навигиране, а решението трябва да бъде конструирано в модули, които позволяват лесното добавяне на допълнителни функционалности

2. Описание на аритектурното решение

Модули и класове

Решението се базира на модулната архитектура за развитие на софтуер. Модулите се съдържат в Python класове, където се намират техните атрибути и методи. Приложението има и помощни класове, в които се съдържа допълнителна функционалност на някои от модулите.

Архитектурата и йерархията на приложението е както следва:

(По-малкото число значи по-важен модул. Подчертаните модули за помощни класове)

Клас	Описание
1 wxPythonApp.py	Главният клас, който задвижва цялата апликация. Съдържа инстанция на MainFrame() в себе си.
2 mainFrame.py 2a weather_location_class.py 2b alarm_frame.py 2c about_frame.py	В mainFrame.py се инициализират всички модули на апликацията. Това са модулите за новини (REST), новини (RSS), имейл, социални мрежи. В mainFrame се съдържа навигацията на апликацията и меню плота, в който се разполага час, мет. време, локация на потребителя, а също така и достъпът до неговите аларми. Класът предоставя лесна възможност за включване на допълнителни модули и контрол за техния достъп. В него се извършва намиране на локацията на потребителя по неговото публично IP, както и времето за тази локация. Това става с инстанция на клас

	WeatherLocation(weather_location_class.py). От този панел може да се достъпи модула за контрол на алармите на потребителя, който се намира в инстанция на Alarm(alarm_frame.py). Алармите са записани в лакална база данни. Те са визуализирани в списък, от където потребителя може да избере коя да е активната аларма. В модулът той може да добавя и махне аларми. В модулът има цикъл, който на всяка секунда проверява дали алармата отговаря на настоящото време, и ако е така, включва алармата.
3 news_panel_class.py 4 NewsApiCall.py 4 ReadNewsFrame.py 5 TTS_Reader.py	В класът news_panel_class.py се инициализира панелът, в който ще се заредят новините, които са изтеглени от RESTful API-то на The Guardian. Преди това да стане обаче, потребителят трябва да зададе някакъв критерий на какви точно новини иска да му се покажат. След това се инициализира инстанция на ReadNews (NewsApiCal.py), която връща резултат статус + JSON, с JSON ако са успешно изтеглени данните. След това нужните данни се изтеглят от JSON и добавят в масиви за всеки тип запис. След това масивите се обхождат,

	за да се попълва информацията за всеки ред от списъка с новините, като накрая новият ред се добавя към въпросния списък с новини. Ако потребителят желае да прочете дадена новина, и натисне бутона за това, се инициализира инстанциця на ReadNewsFrame (ReadNewsFrame.py), където се показва подробно информацията от статията. При нужда, главният текст може да се възпроизведе с Text-To-Speech генератор чрез инстанция на TextToSpeech (TTS_Reader.py)
3 news_rss_panel_class.py	В класът news_rss_panel_class.py се инициализира панелът за зареждане на новините от RSS услугите на някои сайтове за новини. След като потребителят избере новинарска агенция, се създава връзка до нейната RSS услуга и се изтегля информацията от нея. Върнатият отговор е масив с JSON елементи. Масивът се обхожда и се изважда нужната информация от всеки елемент, като след това се попълва в шаблон за ред, който след това се добавя към списъка с редове с новини. Когато потребителят иска да прочете дадена новина, той натиска на бутона й и му се отваря инстанция на

	UrlDialog с хиперлинк, който той може да натисне, за да отвори статията в браузъра си.
3 email_panel_class.py 4 GmailApiCall.py 4 TTS_Reader.py	В този клас се инициализира имейл панелът, който е интегриран с Gmail API. Потребителят натиска бутонът за търсене, за да инициализира зареждане на имейлите. В този момент се създава инстанция на GmailApi (GmailApiCall.py), който проверява дали потребителят е оторизиран да ползва апликацията с Gmail API. Ако не е, му се отваря oauth2 dance browser прозорец, където той позволява на апликацията да достъпи информацията му, и със създадения token.json може отново да опита да зареди имейлите си. При успех, GmailApi връща масив от JSON обекти, които след това биват обходени, за да се изтеглят имейл данните от тях и съхранят в обекти от тип EmailData, които се добавят в масив. Освен обичайните имейл данни се изтегля и запазва в локална база данни и id на състоянието на пощата на потребителя. Масивите с EmailData обекти се обхождат и се изтеглят данните за всеки обект, като се добавят в даден имейл ред, който се добавя към списъка с имейл редове.

	Когато се натисне бутонът за прочитане на даден имейл, цялата информация се зарежда в пространството в панела, отредено за детайлно визуализиране на имейл съдържанието, където се добавя отново функционалност за четене на имейлите от TextToSpeech (TTS_Reader.py) инстанция. В същото време на всяка една минута се изтегля id на състоянието на имейла на потребителя в Gmail API и се сравнява с локално запазената стойност. Ако не съответстват, апликацията информира потребителя за нуждата от синхронизация и при желание от потребителя, интерфейсът се презарежда с новата информация
3 social_panel_class.py 4 social_panel_m_facebook.py 5 FbApiCall.py 4 social_panel_m_twitter.py 5 TweeterApiCall.py	Това е главен родителски панел, в който се зареждат социални мрежи. Също като manFrame, този панел е базиран на модулния принцип и много лесно може да се добавят допълнителни модули и контроли за други социални мрежи освен Facebook и Twitter. В него се инициализират инстанции на FB_panel(social_panel_m_facebook.py) и Twitter_panel(social_panel_m_twitter.py). FB_panel зарежда и визуализира информацията за потребителя от

	фейсбук Graph API като използва инстанция на FbApi(FbApiCall.py), която връща dict, в който срещу всеки ключ стой JSON обект. Речникът се обхожда, извличат се нужните данни, ако са налични, и се добавят директно върху потребителския интерфейс или ако са от сложен тип page, comment, post - в инстанции на FB_page, FB_post, FB_comment, които на свой ред се добавят в масиви. Масивите на стой ред биват обходени и информацията за всеки обект в тях се извлича и добавя върху потребителския интерфейс. В случая на постове, в ред за всеки пост, който се добавя в списък от редове. Ако потребителя иска да прочете целия пост, включително и коментари ако има такива, той натиска върху бутона на поста. Цялата информация за него се извлича от масива с постове и се предава за визуализиране в отреденото за това поле на Фейсбук панела. Добавяне на информация в Graph API на FB през апликацията не се разрешава, защото апликацията не е удобрена от FB за тази дейност. --- Twitter_panel зарежда и визуализира информацията за потребителя, който я е

	оторизирал. Оторизирането е статично и трябва ръчно като се променят потребителските token-и за достъп в кода, за да се достъпи информацията на потребителя в Туитър. За (пре)зареждане на Tweeter информацията на потребителя, се създава инстанция на Tweeter(TweeterApiCall.py). С нейната референция се взима информация за потребителя от Twitter API, като брой последвани, следвани, брой туитове, туитове. Информацията за върнатите сложни обекти, туитове и ретуитове, се запазва в инстанции на класа Tweet, след това всеки обект се запазва в масив за туитове на потребителя. Почти същата логика следва и зареждането на туитове на хора и страници, които потребителя следва. Там има допълнителна стъпка за проверяване дали туитът е ретуийт, за да се събере допълнителната информация за него, и след това обекта на туита от стената се добавя в специален масив за този тип обекти. След това масивът с туитове на потребителя се обхожда и за всеки туит се извлича информацията и се добавя в потребителския интерфейс. Същата

	логика следва и зареждането на туитове от стената. Ако потребителят желае да прочете целия туит от стената в сайта на Twitter, той натиска бутона Read за съответния туит, апликацията намира съобщението с това ID и кой го е създал, създава URL за достъп и го отваря в browser-а на потребителя. Ако потребителя иска да потърси хора или страници, може да използва полето за търсене. С въведената от него информация се създава линк, който се отваря автоматично в браузъра му. Така лесно може да последва някой в социалната мрежа. От Twitter панелът може да се цитира чуждо съобшение или да се пише ново от потребителя. За да се отбележат промените, всяка една от тези две функционалности кара целия Twitter панел да бъде презареден.

Използвани допълнителни библиотеки

Приложението е написано с Python 3. Поради това трябва да се инсталира на машината, на която то ще работи, специална версия на wxPython: **wxpython-phoenix.** Стандартната версия на wxPython е написана за Python 2 и няма да сработи. Wxpython-phoenix се инсталира с командата:

```
pip install -U \
-f https://extras.wxpython.org/wxPython4/extras/linux/gtk3/ubuntu-16.04 \
wxPython
```

Също така трябва да бъдат инсталирани следните допълнителни библиотеки:

- Python Text To Speech:

  ```
  pip3 install pyttsx3
  ```

- Gmail API python library:

  ```
  pip install --upgrade google-api-python-client \
  google-auth-httplib2 google-auth-oauthlib
  ```

- Python facebook-sdk:

  ```
  pip install facebook-sdk
  ```

- Python twitter-sdk:

  ```
  pip install python-twitter
  ```

3. Идеи за бъдещо развитие на апликацията

В апликацията може да се добавят още доставчици на новини, както RESTful, така и с RSS. Добра идея е да се добави функционалност, която да отваря pop-up frame за оторизиране на апликацията за достъп до профила на потребителя, например в Gmail, Facebook, Twitter, тъй наречения “oauth dance”, за да става това автоматично. Може да се добавят още социални мрежи като LinkedIn например. Добре е да се добави и persistance към апликацията. Например, заредената информация в апликацията да се запази в локална база данни при затваряне. Реализира се лесно с обхождане на масивите при затваряне.

4. Използвана литература

Python 3's official documentation	https://docs.python.org/3/
wxPython Phoenix official documentation	https://wxpython.org/Phoenix/docs/html/
The Guardian RESTful API documentation	https://open-platform.theguardian.com/documentation/
Gmail API documentation	https://developers.google.com/gmail/api/v1/reference/
Facebook's Graph API documentation	https://developers.facebook.com/docs/graph-api/
Python facebook-sdk documentation	https://facebook-sdk.readthedocs.io/en/latest/api.html
Twiter's API focumentation	https://developer.twitter.com/en/docs.html
Python twitter-sdk documentation	https://python-twitter.readthedocs.io/en/latest/installation.html