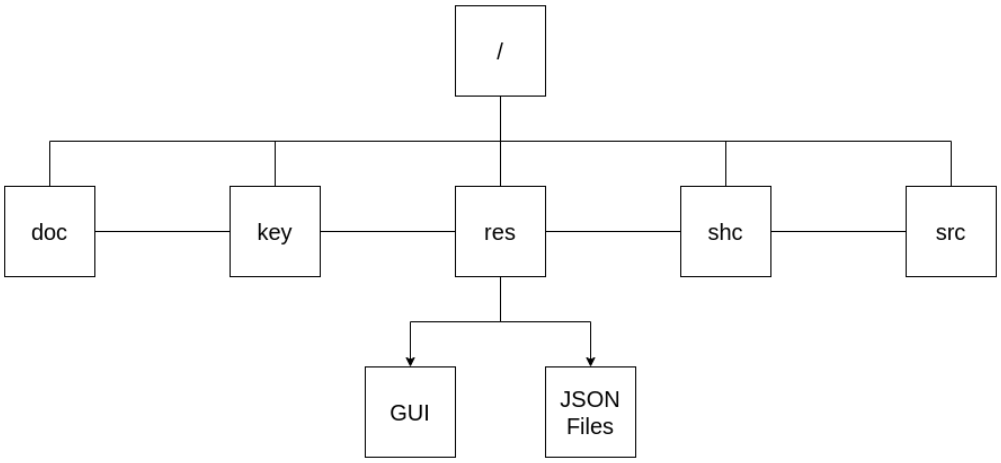
Manual Técnico de ScrapingUPF

Este documento es una explicación sobre como esta estructurado el código y como funciona el programa. Este manual está dirigido para usuarios con conocimiento previo del lenguaje de programación Python y un conocimiento básico del sistema de comunicación con un servidor mediante peticiones. Todo el proyecto y todo el código está escrito de cero por mi mismo.

1. Estructura de Carpetas



Siendo / el directorio principal del repositorio, tenemos 5 carpetas principales en el directorio, siendo estas sus descripciones:

- Documentation (doc): Este directorio solo contiene este archivo y los diagramas usados para la creación del archivo. La idea principal es guardar toda la documentación e información técnica del programa (El archivo README no se encuentra aquí).

- Authorization Keys (key): En este directorio tenemos los archivos de credenciales usados para usar la API de Google Calendar (credentials.json) y el que se usa para guardar la información de la cuenta de Google del usuario que ha iniciado sesión (token.pickle).

- External Resources (res): Aquí se encuentra todo el material externo usado para la aplicación. Tenemos unos archivos PNG y GIF que se usan para ilustrar el README y dos subcarpetas.
    - Graphic User Interface (GUI): Aquí encontramos los archivos de Qt5 usados para crear la interfaz gráfica, el logo de la aplicación en formato GIMP y en PNG, y un archivo de créditos donde hay la GUI en la que me he inspirado.
    - JSON Files (JSON Files): Aquí solo se encuentra un archivo JSON que de genera al descargar la información del horario seleccionado. A partir de este archivo se extrae la información para añadir las asignaturas a Google Calendar.

- Shell Compilation (shc): Encontramos archivos Shell que se usan **solo en Linux** para compilar los scripts de Python en un archivo binario. También tenemos toPython.sh, que nos permite transformar los archivos .ui a .py.

- Source Files (src): Todos los archivos de código fuente se encuentran aquí, luego se entrará en mayor detalle. (3. Archivos de Código Fuente)

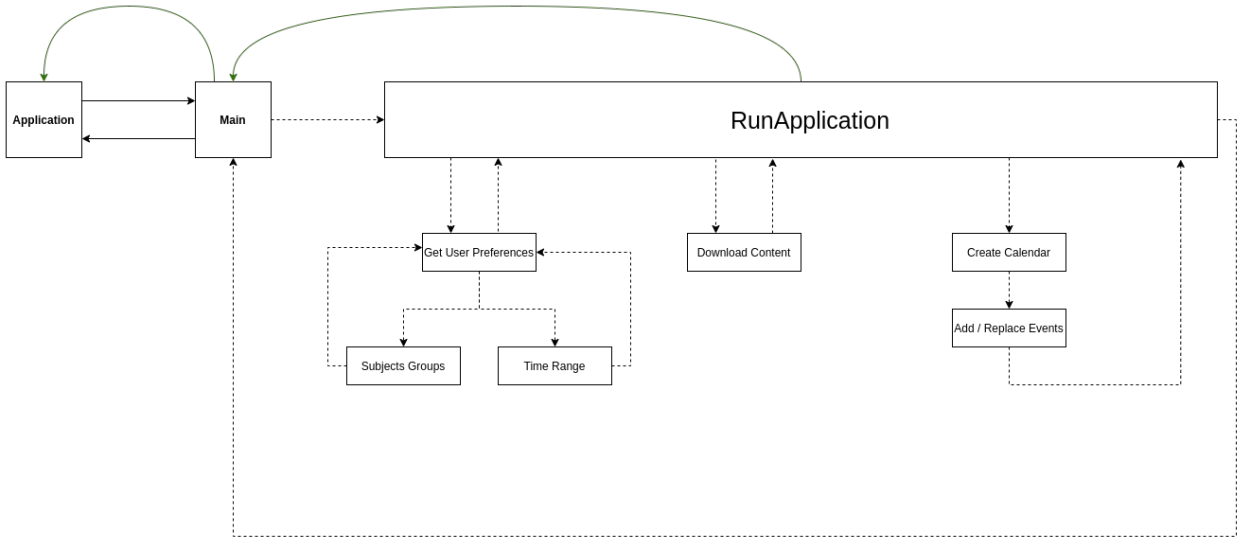
2. Archivos de Requisitos

En el directorio raíz tenemos tres archivos de requisitos, el CoreRequirements, DeveloperRequirements y GraphicRequirements. Todos estos archivo se pueden instalar con la herramienta PIP.

- CoreRequirements: Este archivo contiene las mínimas librerías para usar el programa sin interfaz gráfica, es el menos pesado de todos ya que no instala niguna dependencia de PyQt5 ni permite compilar los scripts.
- GraphicRequirements: Es una versión idéntica a los CoreRequirements pero en este caso se han añadido las dependencias para PyQt5 y para poder ejecutar la interfaz gráfica. Esto hace que los requisitios sean más pesados que el anterior caso.
- DeveloperRequirements: Esta versión contiene todas las librerías necesarias para el desarrollo del programa, no es necesario usar esto si no se quiere programar nada de esta aplicación. En estos requisitos tenemos las librerías para compilar la aplicación.

3. Archivos del Núcleo del Programa (src)

Aquí es donde empieza lo interesante, vamos a hablar sobre como esta estructurado el código del proyecto. Todo el proyecto está hecho en Python 3.8 desde Debian 11.1.
Primero se va a explicar como funciona el núcleo del programa y luego se entrará en detalles de como funciona la interfaz gráfica. Es recomendable ir leyendo esto con los archivos de src en pantalla.



Este diagrama no sigue ninguna convención de diagramas, es simplemente un esquema que tengo para planificar la idea principal de la aplicación. Las líneas discontínuas indican llamadas a funciones, las contínuas son llamadas a otros archivos y las verdes son el estado de carga de la aplicación (Luego entro en más detalles).

Application: Este es el archivo principal, lo único que hace es llamar a la función de ejecutar el programa con el deleteMode desactivado. La ejecución se hace mediante un iterador ya que esta función es un generador que va devolviendo el estado de la carga del programa. Esto solo se usa para la barra de carga de la GUI.
Veremos que dentro de la función principal (RunApplication) tenemos una instrucción yield que se va ejecutando después de terminar correctamente una fase del programa. Esto nos sirve para poder ir mandando información del estado del programa a nuestra función principal, estos valores se van actualizando en la barra de carga. Para más información sobre los generadores ver "Los Generadores en Python".

Main: Aquí empieza el verdadero código, la función principal de este script es RunApplication, que durante su ejecución va a ir llamando a otras funciones y indicando que porcentaje de carga lleva. En el momento que nuestra aplicación se encuentre con algun error, va a devolver False y va a terminar la ejecución.
- Lo primero que hacemos es establecer el estado de carga al 0% y hacer el primer yield para indicar que hemos empezado.
- A continuación, vamos a verificar que el usuario use la versión de Python mínima requerida (Actualmente 3.8.0). Podemos encontrar la versión necesaria en el archivo Constants.py (PYTHON_VERSION), esta verificación implica situar el estado al 3%.
- Ahora verificaremos que exista el archivo de preferencias, ya que sin el no podemos hacer nada, en caso de que exista lo guardaremos en userPreferences. Lo que hemos guardado es un ConfigParser, que es una estrucutra que nos va a facilitar leer y escribir en este archivo. Este proceso aumenta el estado de la carga al 9%.
- En este punto ya hemos verificado que el entorno sea el correcto para nuestro programa, así que vamos a leer la informacinó de userPreferences para saber si el usuario usa el modo automático o el modo manual. Si usa el modo automático, localizaremos la ubicación del archivo HTML con las asignaturas y extraemos la información, en caso de que use el modo manual, leemos la información de asignaturas del UserPreferences.ini.
Al terminar este punto tenemos el estado de carga al 14% y las variables:

· fromGroups: Es un conjunto con los distintos grupos de teoria que se usan.
· fromSubjects: El código de cada una de las asignaturas. (Lista)
· userSubjectsGroups: Para cada asignatura, su grupo de teoría. (Lista)
· pGroups: Para cada asignatura, su grupo de prácticas. (Lista)
· sGroups: Para cada asignatura, su grupo de teoría. (Lista)

(fromGroups es necesario para la petición al servidor, ya que pide un listado de todos los grupos distintos, para todo los $i \in fromGroups$ y $j \in fromGroups \rightarrow i \neq j$)
- La siguiente operación es muy sencilla, consiste en generar un diccionario con asignaturas como claves y colores como valores. Asignamos un color a cada asignatura para luego añadirlas a Google Calendar con ese color. Esto hace que el estado de carga ascienda hasta el 16%.
- A continuación extraemos la información principal y la información de las fechas.
La información principal (basicInformation) es un diccionario con claves:

· PlanEstudio · IdiomaPais
· Trimestre · PlanDocente
· CodigoCentro · CodigoEstudio · Curso

Una vez tenemos la información básica, obtenemos una tupla con la fecha inicial del calendario y la fecha final (DD/MM/YY, DD/MM/YY).
Para la petición tenemos que transformar esa fecha en UNIX time format, así que usamos unas funciones de las librerías time y datetime.
Al terminar este proceso, la carga llega al 21%.
- Al llegar a esta fase, estamos seguros de que el los datos son correctos y que hemos extraído los datos del usuario, así que vamos a generar la petición y a extraer los datos. Vamos a llamar a la función downloadContent que recibe los datos para generar la petición, y los headers. Esta función nos va a devolver el archivo JSON con la información de las asignaturas. Esta función es una de las más importantes del programa ya que es la que hace los engaños al servidor de la UPF para obtener el horario.
Para hacer esto se usan tres URLs distintas:
URL, URL_RND y URL_JSON (Sus valores se pueden ver en Imports.py)
A continuación vamos a ver para que sirve cada una de ellas.

Primero haremos una petición GET a URL, no le mandamos ninguna información sobre el usuario, simplemente obtenemos la sesión (SESSION) sobre la que vamos a trabajar. En esta petición solo mandamos los headers, no necesitamos nada más.
Una vez ya tengamos la sesión establecida vamos a necesitar extraer un número RND, esto es valor que creo que se usa para añadir algun tipo de seguridad o evitar que se haga scraping pero no estoy muy seguro de ello. Así que hay que extraer un valor llamado RND que se va a tener que mandar junto con el rango de tiempo del horario que queremos consultar.
A URL_RND le vamos a hacer una petición POST, le mandamos los datos del usuario y nos devuelve una página en la que se encuentra el número RND que es necesario para recibir el JSON.
Una vez hecha la anterior petición, extraemos mediante la librería BeautifulSoup el valor de RND. Con el valor RND ya podemos hacer una petición POST a URL_JSON y de esta forma extraemos el archivo JSON. Esta última petición la hacemos mandando el contenido:

- RND: Número entero que asocia la petición con la información de asignaturas y grupos del usuario.
- fromDate: Fecha en formato UNIX a partir de la que queremos ver el calendario.
- toDate: Fecha en formato UNIX hasta la que queremos ver el calendario.

Finalmente, recibimos un archivo JSON con toda la información requerida.
Ahora que ya tenemos la información de las asignaturas, las vamos a transformar a bloques. Esto es simplemente leer el archivo JSON y transformarlo a una estructura de datos llamada SubjectBlock. En este punto nos aseguramos de que se haya descargado almenos una asignatura. El estado de carga es del 31%.
- Ahora vamos a crear el objecto calendario de Google Calendar, si el usuario ha seleccionado algun ID de un calendario ya existente se va a usar ese, de lo contrario, se usa el calendario principal. Esto establece el estado de carga al 33%.
- Finalmente, vamos a añadir los eventos a Google Calendar dependiendo de varias condiciones:

- Replace Mode: En este caso, si una asignatura ya existe en el calendario, se actualiza evitando que se añadan duplicados. Primero eliminamos todos los eventos y luego los añadimos, nada complicado y al mismo tiempo nada eficiente. :)
- Delete Mode: En este caso se eliminan todas las asignaturas añadidas por el programa que cumplan con las condiciones de UserPreferences.ini.
- Else: Aquí simplemente se añaden las asignaturas con riesgo de que esten duplicadas.

En este momento el programa ha terminado y el estado de carga es del 100%.

Configuration: Este script contiene funciones que se usan para extraer y escribir información en el archivo de preferencias, la mayoría son muy similares y simplemente extraen un valor de un diccionario o añaden un valor a un diccionario. Todas ellas tienen sus parámetros explicados en el código.

Constants: Este script se puede ver a simple vista lo que es, son todas aquellas variables globales o constantes que se usan en todo el programa. La idea es centralizarlas en un solo archivo para facilitar su modificación

Imports: Este script esta directamente relacionado con el anterior, contiene todos los imports necesarios para ejecutar el programa. También importa todos los elementos del archivo de constantes.

Google Calendar: En este script creamos una clase Calendar que va a almacenar la identificación del usuario (key/tocken.pickle) y que nos va a servir para añadir eventos, eliminar eventos, leer un evento o leer todos los eventos de una forma sencilla. Es simplemente el sitio donde se usa la API de Google Calendar.

Network Requests: Encontramos una primera función, que para una URL con contenido id=1&class=5..., nos devuelve un diccionario tal que dict[id]=1, dict[class]=5...
La clase Request nos permite hacer peticiones POST y GET manteniendo la sesión y los headers, es una modificación del request habitual pero acomodado para este programa. Las funciones son bastante descriptivas y el el código se encuentra comentado.
La clase HTML_Localfile sirve para extraer la información de las asignaturas de nuestro archivo HTML.

Schedule Blocks: Contiene una clase sencilla para guardar información sobre una asignatura y contiene una función para transformar a SubjectBlock los elementos de un JSON file.

(A partir de este punto vamos a entrar en detalles sobre la interfaz gráfica)

4. Archivos de PyQt5 y la Interfaz Gráfica (src)

Lo único que hace la interfaz gráfica es escribir y leer en el archivo de UserPreferences y ejecutar RunApplication en distintos modos dependiendo de lo que el usuario elija.

MainWindow: Este archivo contiene el código en Python necesario para ejecutar la aplicación, a partir de la función runOnInit ha sido generado por Pyuic5, que es una herramienta de PyQt para transformar un archivo UI (res/GUI/MainWindow.ui) a un archivo Python y poder modificar el funcionamiento de diversos elementos. Todo ese código es implemente para crear un botón y añadir texto, y otras cosas visuales no importantes. Si se quiere modificar la interfaz, lo mejor será abrir MainWindow.ui con Qt Designet.
La función runOnInit va a ser lo primero que se va a ejecutar después de generar la interfaz, va a leer la información de UserPreferences y la va a meter en las cajas de texto.
Las otras funciones:

- clickSaveButton: Escribe la información de los campos de texto en el archivo de preferencias.
- clickAddButton: Esto nos añade una asignatura nueva (Sin eliminar lo existente) al archivo de preferencias. Es la manera manual de añadir asignaturas.
- clickUseButton: Esto nos guarda la información de la ruta del archivo HTML de las asignaturas.
- clickRemoveSubjectsButton: Ejecuta el programa en modo eliminar.
- clickAddSubjectsButton: Ejecuta el programa en modo reemplazar.
- clickVerificationButton: Verifica que nuestro archivo HTML sea correcto.

PyQt5 Imports: Contiene las librerías necesarias para ejecutar todo lo relacionado con Qt5. Es lo mismo que el Imports.py pero es una versión con todo lo relacionado con Qt5.

QtPalette: Contiene una paleta para generar el modo oscuro, aún no está terminada. Pero los valores que hay que modificar son los que se pueden ver en el archivo, así que puedes jugar con ellos.

5. Archivo Log.txt

Durante todo el programa, se va generando un texto que va informando de lo que se ha hecho y de la situación del programa. Toda esta información se escribe también en un archivo llamado Logs.txt. Esto se hace en Main.addLogInformation donde para cada línea escrita, se le añade los milisegundos que hace que se ha iniciado el programa.

6. GNU General Public License 3.0

Como todos sabemos que nadie (Excepto gente como Richard Stallman) se lee las licencias, voy a explicar de forma resumida que implica usar esta licencia en este proyecto:

1. Todo el mundo puede copiar, modificar y distribuir este programa.
2. Para toda versión de este programa, la licencia debe seguir siendo GPL 3.
3. Puedes usar este programa de forma privada y de forma comercial.
4. Si usas este programa de forma comercial, debe seguir siendo open source.
5. Todas las modificaciones de código deben ser indicadas.
6. Este programa no viene con ninguna garantía.
7. No me hago responsible de los daños que pueda causar este programa.

7. ¿Qué puedes contribuir?

Yo voy a intentar seguir manteniendo este programa almenos durante este año y el siguiente, si alguien quiere ayudar reportando errores, dando ideas o mejorando cosas, todo se puede hacer mediante Pull Requests o Issues en el repositorio de GitHub.
Me gustaría crear binarios ejecutables tanto para MacOS como para Windows, pero debido a que no uso ninguno de estos dos sistemas operativos, solo hago binarios para las distribuciones Debian, Fedora, Arch, Gentoo y todas aquellas basadas en las anteriores. La aplicación ha sido testeada en esas distribuciones y en Pop!_OS, Manjaro, Windows 10 y MacOS.

· Si alguien se quisiera encargar de compilar para Windows y MacOS, solo tendria que usar PyInstaller, que es extremadamente sencillo de usar.
· La versión actual requiere Python 3.8 o superior, hay la "oldpython" branch que no usa el operador Walrus, pero no esta para nada actualizada. Si alguien quiere mantenerla, solo que hay que ejecutar el programa actual en Python 3.6 y ver que cosas no funcionan y reemplazarlas por lo equivalente en versiones anteriores a 3.8.
· Otra cosa que se debería gestionar es el hecho de que el usuario tenga que introducir tanta información me gustaria que solo con el archivo HTML se pudiera hacer todo.
· El sistema que informa del progreso de carga se podria modifiar y usar un thread paralelo en vez de generadores, ya que los generadores pueden ser demasiado complejos.
· Finalmente, me gustaría que la GUI pareciese un poco más profesional pero no tengo suficientes habilidades gráficas para ello.

8. Contactar

Si necesitas comentarme lo que sea puede ser mediante e-mail o Discord:
joan.gracia01@estudiant.upf.edu
Sklyvan#3902

Si el proyecto te interesa o te ha ayudado en algo, se agradece una estrella en GitHub!

14/12/2021