UNIDADE 7

DISTRIBUCIÓN DE APLICACIÓNS

DESENVOLVEMENTO DE INTERFACES

CS DESENVOLVEMENTO DE APLICACIÓNS MULTIPLATAFORMA

Autor: Manuel Pacior Pérez

Índice

1 INTRODUCIÓN

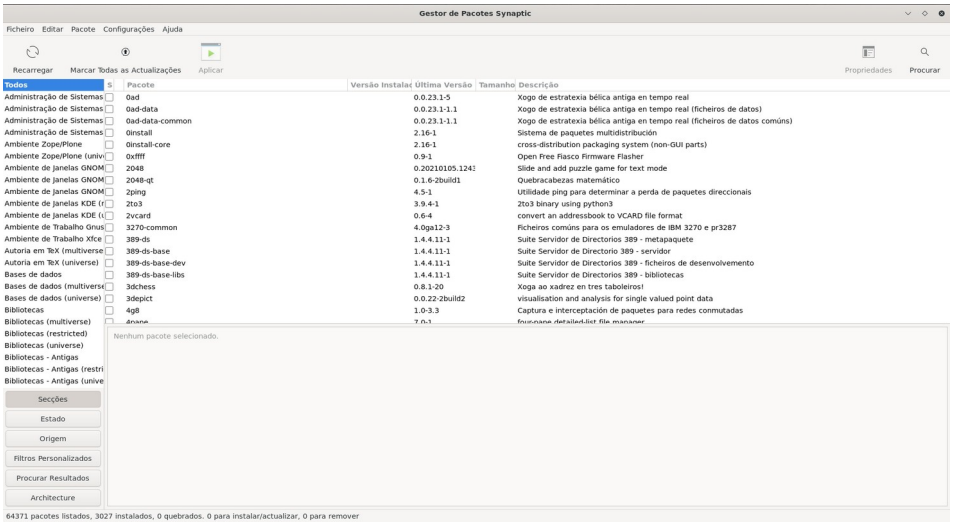
Unha distribución de software é un programa ou conxunto de programas específicos que se empaquetan e configurar para unha ou varias plataformas ou sistema operativo. No caso de que esa aplicación estea feita cun linguaxe de programación compilado, entón haberá que compilar previamente a aplicación para obter o binario correspondente.

2 SISTEMA DE XESTIÓN DE PAQUETES

Un sistema de xestión de paquetes é unha colección de ferramentas empregadas para automatizar o proceso de instalación, actualización, configuración e eliminación de paquetes de software.

Normalmente, este termo úsase para referirse a xestores de paquetes en sistemas Linux, que os usan como base principal para xestionar as aplicacións para instalar de xeito inmediato a través de repositorios oficiais da propia distribución, ou engadindo algúns de terceiros. Algúns exemplos destes sistemas son: ***rpm, deb, Snap ou Flatpak.***

Existen aplicacións gráficas para administrar e traballar con este tipo de sistemas, por exemplo, unha das máis coñecidas é ***Synaptic*** para o sistema de paquetes ***.deb***



Imaxe. Xestor de paquetes Synaptic

Nestes sistemas, o software distribúese en forma de paquetes, non é máis que un conxunto de ficheiros organizados en base a un determinado estándar. Se o descomprimimos atoparemos unha serie de recursos (ficheiros, imaxes, ...) que basicamente conterá a seguinte información:

- O propio software executable
- O nome completo do paquete
- Unha descrición da súa funcionalidade
- O número de versión
- O distribuidor de software
- Unha lista doutros paquetes necesarios para o correcto funcionamento do software (coñecidas como dependencias). Esta información normalmente introdúcese nunha base de datos de paquetes local.
- A icona visual que ten asignada a aplicación no sistema
- A suma de comprobación

3 INSTALADORES

Na maioría dos casos, un programa está composto por un conxunto de ficheiros. Normalmente, estes ficheiros deben copiarse a certos cartafoles ou directorios e, en moitos casos, deben darse de alta no rexistro ou ficheiros de configuración do sistema (Windows, Linux, ...). Os instaladores realizarán todas as operacións anteriores de forma transparente ao usuario.

Usar o instalador adoita ser moi sinxelo, e na actualidade é moi común que presenten ao usuario unha serie de formularios onde se mostran as indicacións pertinentes, limitando a acción do usuario a pequenas modificacións, ou directamente a premer o botón seguinte.

O instalador polo tanto, *copiará os ficheiros da aplicación que se instalará nos directorios apropiados, rexistrará a aplicación, creará os menús e os atallos no escritorio de xeito automatizado*.

3.1 Asistentes de instalación

Os asistentes de instalación son aplicacións que axudan ao usuario a personalizar a instalación do software. Cando usamos un instalador, normalmente ten asociado un asistente de instalación, que adoitan ter as seguintes características:

- Cada paso da instalación móstrase mediante un formulario coa información do paso que se está a dar.
- Permítennos escoller os directorios onde queremos que se instale a aplicación
- É común que permitan configurar o grupo de programas onde está integrada a aplicación no menú do escritorio
- Amosan información sobre a licenza, rexistro da aplicación, etc.

4 PAQUETES AUTOINSTALABLES

Cando finaliza o ciclo de desenvolvemento dunha aplicación, é hora de decidir a mellor forma de distribuíla, alén diso tamén debemos considerar engadir funcións adicionais, parches e revisións á aplicación.

Cando decides distribuír unha aplicación mediante un paquete autoinstalable, o que fas é empaquetar a aplicación nun único ficheiro, que conterá todos os ficheiros e directorios que compoñen a aplicación.

Este ficheiro será un ficheiro executable, que terá un formato diferente en función do SO e/ou distribución na que nos manexemos:

- *Windows*: extensión *.exe*
- *GNU/Linux*:
  - Distribucións baseadas en Debian (Debian, Ubuntu, etc.): ficheiro con extensión *.deb*
  - Distribucións ao estilo de Red Hat (Red Hat, Suse, etc.):.Un ficheiro con extensión *.rpm*

4.1 Windows

Se estamos en Windows, o paquete de autoinstalación será unha aplicación que, unha vez lanzada polo usuario, descomprimirá todos os ficheiros, creará as carpetas que a aplicación precisa, copiará os ficheiros nos directorios de destino, engadirá e/ou modificará as entradas no rexistro de Windows, e tamén as entradas no menú de aplicacións e mostrará atallos no escritorio.

Durante todo este proceso, o máis común é que o usuario poida interactuar co instalador: escollendo compoñentes para instalar, modificando os directorios de instalación, decidir se crea ou non accesos directos, ou escoller as opcións predeterminadas de instalación.

4.2 Linux

Se estamos distribuíndo unha aplicación para Ubuntu, por exemplo, o que se crea é un paquete deb. Este tipo de ficheiro contén todos os ficheiros e directorios da aplicación. Cando o usuario quere instalar o paquete, executa o "Ubuntu Software Center" ou "Gdebi" (en función de cal teñamos configurado), polo xeral xa fai todo de xeito automático, aínda que tamén pode mostrarnos certas confirmacións e realizar algunha consulta de configuración nalgún caso.

5 FERRAMENTAS PARA CREAR PAQUETES DE INSTALACIÓN

Hoxe en día existe unha ampla gama de ferramentas para crear paquetes de instalación. Algúns exemplos deste tipo de ferramentas nos diferentes sistemas operativos máis utilizados son:

- Para Linux:
  - Snaps
  - Flatpak
- Para Windows:
  - install4j
  - Windows Installer
- Multiplataforma (Linux, Windows e macOs):
  - Zero Install
  - PyInstaller

A instalación dunha aplicación é o primeiro contacto que terá un usuario coa aplicación, polo que se o proceso de instalación do software é lento, ou non remata correctamente, sentirase irritado e frustrado, o que producirá unha sensación de que a nosa aplicación non funciona ou é inestable, aínda que en realidade iso non sexa así. Porén, un instalador rápido e amigable debe ser unha parte esencial de calquera produto de software.

5.1 Windows

Nos sistemas Windows ou Mac OS, os programas que queremos instalar adoitan ser buscados en Internet e son principalmente en forma de instaladores executables Recentemente Microsoft incorporou na súa versión de Windows 10 a Microsoft Store,

desde a que se pode instalar aplicacións de xeito centralizado.

5.2 Linux

Se traballamos en Linux, necesitamos crear un paquete de instalación segundo a distribución de Linux onde queremos instalalo. Se nos atopamos Ubuntu Linux, o tipo de paquete será debian (.deb). O paquete de instalación de ***.deb*** empaquetará todos os ficheiros que precisa a nosa aplicación e debe configurarse para indicar ao "Centro de software Ubuntu", onde se deben copiar estes ficheiros e que opcións de configuración se deben modificar.

En sistemas de código aberto como Ubuntu GNU/Linux a maior parte do software está empaquetado en ficheiros .deb (.rpm en Red Hat), snap ou flatpak, que conteñen os programas e as bibliotecas que precisan.

Os repositorios son servidores que soportan o conxunto de paquetes e manéxanse con ferramentas como Synaptic. A través deles centralízase a instalación dos paquetes que se poden instalar, ofrecendo unha ampla gama de software.

En Ubuntu, por exemplo, teremos habilitados como mínimo os repositorios oficiais de Ubuntu (que poden incluír mesmo o CD de instalación), máis é bastante común ter activados outros repositorios de terceiros.

6 PERSONALIZACIÓN DA INSTALACIÓN

Despois de que un programador remate de desenvolver unha aplicación, debe decidir sobre o mecanismo a utilizar para a súa distribución:

- Se o produto desenvolvido debe instalarse en sistemas Windows, deberá crear instaladores para Windows (ficheiros .exe executables).
- Se o vai distribuír en Ubuntu Linux, o máis común é crear un paquete de instalación (paquete .deb).

Xeralmente, cando se crea un instalador para unha aplicación, este instalador amosará o logotipo da aplicación ou da empresa de desenvolvemento, terá a súa propia icona, as súas propias cores e formato da xanela, formularios nos que se amosan os acordos de licenza, a posibilidade para seleccionar o idioma de instalación, os directorios onde desexa copiar os ficheiros da aplicación, etc.

6.1 Logotipos

O logotipo é un elemento gráfico que identificará a empresa que desenvolveu o programa ou o propio programa. Os logotipos normalmente inclúen símbolos que están claramente asociados a quen representan.

O logotipo é un dos activos máis importante dun servizo/produto, pois utilízase como distintivo. É por iso que cando deseñamos o logotipo dunha aplicación, debemos ter en conta algúns conceptos. Para que un logotipo teña éxito, débese seguir o principio de deseño fundamental de "menos é máis", a sinxeleza permitirá que o logotipo sexa:

- Lexible
- Escalable
- Reproducible.
- Distinguible.
- Memorable.

Nun instalador pode inserir o logotipo arriba, abaixo, á dereita ou á esquerda do instalador. O tamaño do logotipo basearase no ancho / alto especificado, o ancho e alto do instalador e a fonte usada no instalador.

6.2 Fondos

Ao deseñar un instalador para unha aplicación de interface gráfica, a interface de instalador debe adquirir e presentar información de xeito coherente coa interface da aplicación á que serve. No caso dos fondos do instalador, organizaranse en función do deseño estándar que se mantén en todas as fiestras da aplicación.

Polo tanto, o fondo do instalador debería implementar as mesmas regras de deseño para manter a interacción durante toda a aplicación.

Está comprobado que os contidos cunha cor de fondo son interpretados polos usuarios como contido sen importancia; Normalmente trátase de contido publicitario.

Cando a cor de fondo é branca (ou, simplemente, non hai cor de fondo) o usuario interpreta esa información como relevante e o seu nivel de atención aumenta considerablemente.

6.3 Botóns

Cando se fai un instalador, os botóns que se implementan aceptan ou rexeitan o acordo de licenza. Os botóns que normalmente aparecen son os seguintes, anteriores e finais.

No proceso de instalación gráfica dunha aplicación, o que se nos presenta é un conxunto de fiestras, nas que o usuario toma algunhas decisións. Os botóns de cada unha das fiestras permiten aceptar ou cancelar os valores ofrecidos polo instalador así como avanzar ou retroceder nas fiestras de instalación.

Temos que ser coherentes co deseño gráfico dos botóns do instalador e o deseño gráfico da aplicación á que serve, é dicir, sempre hai que ser coherente con todos os elementos que forman parte dunha aplicación, incluído o seu instalador.

Manter a coherencia implica que o formato das fiestras, as cores, as iconas, as imaxes, os botóns e outros compoñentes gráficos manteñen o mesmo formato, tanto en cores, tipos de letra e tamaños.

A forma máis común dos botóns gráficos é rectangular, aínda que ás veces poden ter esquinas redondeadas. Outras formas comúns son: circulares, elípticas e trapezoidais.

6.4 Idioma

A maioría das aplicacións que se distribúen están dispoñibles para calquera usuario nunha páxina web ou páxinas de descarga. Esta disposición global das aplicacións implica que calquera usuario de calquera parte do mundo pode ter acceso á aplicación e pode descargala e instalala.

Polo tanto, se unha aplicación quere ser distribuída en calquera lugar, debe ter a interface implementada, como mínimo, en inglés. Tamén é común traducir a interface a varios idiomas, e que o usuario final elixa durante o proceso de instalación cal desexa usar para a súa instalación.

O programador debe ter en conta o seu deseño, a posibilidade de que a súa aplicación poida ser distribuída en varios idiomas. Se esta posibilidade está presente, o programa de instalación tamén debe presentarse en diferentes idiomas e nel pódese escoller o idioma de instalación final.

7 PARÁMETROS DE INSTALACIÓN

Os parámetros de instalación personalizarán a aplicación, dándolle ao usuario que realiza a instalación a opción de escoller entre diferentes alternativas e opcións, para axustar a instalación da aplicación ás súas necesidades.

Dado que a maioría das aplicacións actuais distribúense a través de repositorios aloxados en sitios web, as aplicacións son accesibles para calquera usuario desde calquera parte do mundo.

7.1 Idioma

As aplicacións que teñen unha maior difusión adoitan estar dispoñibles en diferentes idiomas. Isto forma parte dun proceso máis complexo que se denomina internacionalización do software, no que as aplicacións se adaptan á lingua, cultura e peculiaridade dos diferentes grupos de usuarios que a van usar a aplicación. Un dos primeiros pasos da instalación soe ser a selección do idioma.

7.2 Aceptación da licenza

Todas as aplicacións distribúense baixo diferentes contratos de licenza, polo que o usuario debe configurar a aceptación ou rexeitamento do contrato de licenza. Se o usuario non acepta os termos da licenza, a instalación será interrompida.

7.3 Ruta de instalación

Unha vez que o usuario acepta os termos da licenza, o seguinte parámetro a configurar adoita ser a ruta de instalación dos ficheiros da aplicación. Normalmente o instalador ofrece un camiño predeterminado que o usuario pode modificar se o desexa.

7.4 Atallos da aplicación

Unha vez escollidos os cartafoles onde se van copiar os ficheiros da aplicación necesarios, o seguinte paso soe ser a creación de atallos no menú de inicio do sistema operativo así como o atallo no escritorio. En Windows normalmente dáselle ao usuario a posibilidade de crear ou non un atallo no escritorio, así como no inicio rápido.

7.5 Outros parámetros

Outra característica común nos procesos de instalación actuais é que inclúen aplicacións ou barras de ferramentas adicionais para navegadores web. Estas instalacións son opcionais, polo que o usuario final decide se quere ou non que se realice a súa

instalación.

O último parámetro ao que o usuario adoita acceder é a opción de executar a aplicación unha vez instalada.

8 XERACIÓN DE PAQUETES DE INSTALACIÓN

Para xerar paquetes de instalación dunha aplicación, temos varias alternativas:

- Usar contornos de desenvolvemento.
- Fai uso de ferramentas externas.
- Instalar en modo desatendido.

8.1 Contorna de desenvolvemento

A primeira alternativa é empregar as ferramentas de xeración de paquetes incorporadas nas propias contornas de desenvolvemento. Normalmente contan con sistemas de xeración de paquetes, aínda que non soen xerar instaladores demasiado "amigables". Un exemplo disto son os IDE Netbeans ou Eclipse.

8.2 Ferramentas externas

Hai moitas ferramentas externas aos contornos de desenvolvemento que nos permiten crear paquetes de instalación para aplicacións. Algunhas delas xa as enumeramos no apartado 5:

- Para Linux:
  - Snaps
  - Flatpak
- Para Windows:
  - install4j
  - Windows Installer
- Multiplataforma (Linux, Windows e macOs):
  - Zero Install
  - PyInstaller

8.3 Modo desatendido

Unha instalación en modo desatendido é aquela na que o usuario non ten que tomar ningún tipo de decisión: onde instalar a aplicación, idioma, outras características, etc

O uso de instalacións desatendidas é útil cando os programas deben instalarse nun gran número de ordenadores, por exemplo, é moi común na instalación de sistemas operativos de xeito masivo en determinados contornas.

9 DESCARGA E EXECUCIÓN DE APLICACIÓNS WEB

Cando se implementa un paquete de software para a súa distribución, existe a posibilidade de aloxalo nun servidor web para que sexa accesible a un grupo de usuarios. Estes usuarios poden instalar o paquete directamente no seu ordenador, simplemente premendo nunha ligazón. Este xeito de distribuír software é moi común en distribucións Linux (como Ubuntu).

As principais vantaxes deste mecanismo de distribución de aplicacións é a de centralizar e facilitar a instalación. Para instalar aplicacións desde un servidor, o único que fai o usuario é facer clic na ligazón onde aparece o paquete que quere instalar.

Un exemplo onde podemos comprobar isto é a instalación de Firefox en Ubuntu. Para realizala é preciso ter apturl. Trátase dunha aplicación gráfica para instalar programas desde o repositorio onde se atopan. Para distribuír aplicacións mediante esta ferramenta, edítase unha páxina web onde normalmente se dá unha descrición do programa que se vai instalar e engádese a seguinte ligazón:

- Imaxe que mostra un monitor de ordenador activado.

```
<a href="apt:package"> Nome do paquete a instalar </a>
```

- Se queremos distribuír varios paquetes na mesma ligazón, a sintaxe sería a seguinte:

```
<a href="apt:package1,package2,package3"> Nome da aplicación </a>
```

Un exemplo de instalación deste xeito sería:

```
apturl apt:pidgin,pidgin-plugin-pack
```

Onde se instalaría a aplicación Pidgin e Pidgin Plugin Pack.

10 DISTRIBUCIÓN DE APLICACIÓNS EN PYTHON

Para distribuír as aplicacións de Python precisamos usar o intérprete de Python, pois ao contrario do que acontece coas linguaxes compiladas, onde se xera un executable binario para a distribución ou plataforma na que desexamos executalo, nunha linguaxe interpretada, o código da aplicación é executada a través do seu intérprete.

O uso dun intérprete facilita o desenvolvemento de aplicacións multiplataforma (compatibles con varios SO), sempre que se implemente correctamente (sistemas de rutas, formato dos ficheiros, ...), aínda que para crear o empaquetado debemos botar man de técnicas e ferramentas diferentes aos linguaxes compilados.

10.1 PyInstaller

Para as aplicacións implementadas usando a linguaxe Python, temos a ferramenta PyInstaller. Trátase dun programa que empaqueta programas Python en executables autónomos para diferentes sistemas operativos, entre os que destacan: Windows, Linux, Mac OS X, FreeBSD e Solaris.

10.1.1 Primeira aplicación

Para instalalo tan so debemos usar o xestor de paquetes de python *pip* do seguinte xeito:

```
$ pip instalar pyinstaller
```

Para probalo crearemos un primeiro programa moi sinxelo en Python chamado *ola.py*:

```
# -*- coding: utf-8 -*-
print ("Ola mundo!")
```

Agora imos crear un executable a partir del, polo que debemos facer o seguinte:

```
$ pyinstaller ola.py
```

Unha vez rematado o proceso, temos o seuginte:

- Crearanse varios directorio: *build* e *dist*.
- O que nos interesa é *dist*, dentro atoparemos un cartafol co nome do programa, que contén unha chea de ficheiro e o executable

- No caso de estar sobre Linux: Será un ficheiro executable co nome de ***ola.***
- No caso de estar sobre Windows: será un ficheiro executable co nome de ***ola.exe***.

Como é un programa de terminal, para executalo teño que abrir o terminal nese directorio e executalo manualmente.

10.1.2 Executable con interface

Agora imos facer outro exemplo con un sinxelo programa usando PySide2, para elo creamos un arquivo chamado ***eventos.qt.py*** co seguinte contido:

```
import sys
from PySide2.QtWidgets import QApplication, QLabel, QLineEdit, QMainWindow

class MainWindow(QMainWindow):
    '''Esta clase representa a xanela principal da nosa aplicación, contén
       un elemento de tipo QLabel que vai amosando información sobre os
       diferentes eventos que suceden no sistema'''

    def __init__(self):
        '''Constructor da clase QMainWindow, o primeiro que facemos é chamar
           á función da clase pai  través do método 'super()'. A partires de
           aí, engadimos o código preciso para personalizar o funcionamento
           da nosa xanela'''
        super().__init__()
        self.label = QLabel("Preme nesta xanela co rato")
        self.line_edit = QLineEdit()
        self.setMouseTracking(True)
        self.label.setMouseTracking(True)
        self.setCentralWidget(self.label)

    def mouseMoveEvent(self, e):
        '''Evento que salta cando movemos o rato sobre o Label ao mesmo tempo
           que temos o botón premido'''
        super().mouseMoveEvent(e)
        text_label = "O rato móvese x: {0} | y: {1}".format(
            e.pos().x(), e.pos().y())
        self.label.setText(text_label)

    def mousePressEvent(self, e):
        '''Evento que salta cando se fai un click co rato sobre o Label'''
        super().mousePressEvent(e)
        self.label.setText(" Preme o botón do rato")

    def mouseReleaseEvent(self, e):
        '''Evento que salta cando despois de premer o botón do rato, deixamos
```

```
            de premelo'''
        super().mouseReleaseEvent(e)
        self.label.setText(" Solta o botón do rato")

    def mouseDoubleClickEvent(self, e):
        '''Evento que salta cando se fai un duplo click co rato sobre o
           Label'''
        super().mouseDoubleClickEvent(e)
        self.label.setText(" O rato fai duplo click")


if __name__ == "__main__":
    # Agora que temos definida a nosa xanela principal. Comezamos definindo a
    # instancia
    app = QApplication(sys.argv)
    # da aplicación principal Qt e pasamos a lista de argumentos do sistema.
    # Definimos un obxecto da clase definida para xerar a xanela principal da
    # aplicación
    window = MainWindow()
    # Amosamos a nosa xanela principal
    window.show()
    # Inicia o ciclo de eventos até que se pecha a aplicación
    sys.exit(app.exec_())
```

Agora podemos volver a xerar o executable tal como fixemos antes:

```
$ pyinstaller eventos.qt.py
```

Nesta ocasión, se executamos o programa cun dobre clic, funcionará ben, o problema será que o terminal móstrase no fondo.

Para facer desaparecer ese terminal, temos que indicar que é unha aplicación con xanelas cando xeramos o executable. Para iso facemos:

```
$ pyinstaller --windowed eventos.qt.py
```

10.1.3 Outras opcións

Nos anteriores exemplo vemos que por defecto PyInstaller crea un directorio con moitos ficheiros. Alén diso, podemos xerar un único ficheiro executable que o conteña todo, porén ocupará moito máis. Teríamos que facelo deste xeito:

```
$ pyinstaller --windowed --onefile eventos.qt.py
```

Tamén podemos cambiar a icona predeterminada do executable, para iso precisamos unha imaxe en formato *.ico*.

```
$ pyinstaller --windowed --onefile --icon =./logo.ico eventos.qt.py
```

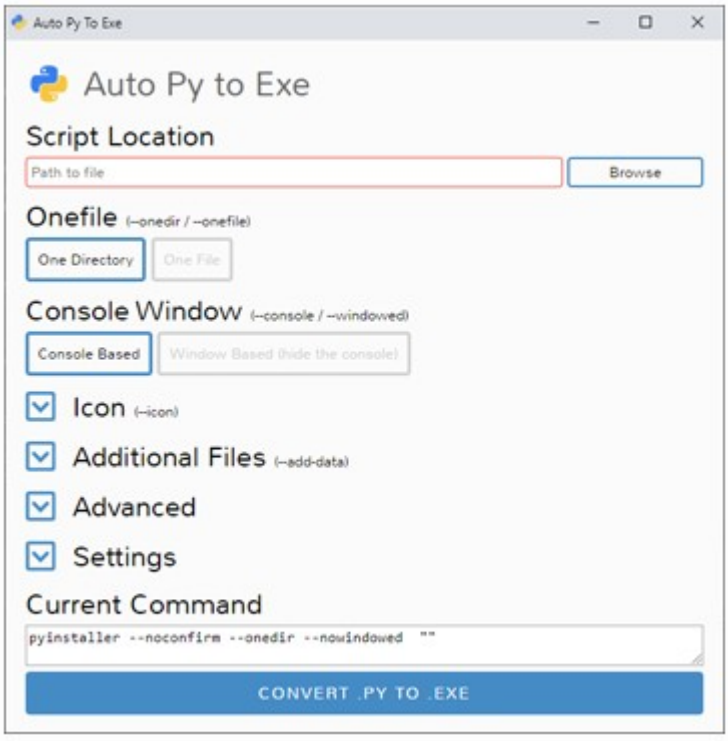
Nota: Nos sistemas Windows, se a icona non cambia ao facer isto e volver executar a aplicación, entón podemos mudar o nome do executable, pois ás veces o sistema de caché do sistema provoca estas cousas.

10.1.4 Limitacións

O gran problema con PyInstaller son as dependencias, pois se a nosa aplicación usa módulos da biblioteca estándar non teremos ningún problema, porén á hora de usar módulos externos, pode que non sexa compatible e no se atopen soportados. Na seguinte ligazón hai información sobre os módulos soportados.

10.1.5 Xestión gráfica: auto-py-to-exe

A aplicación *auto-py-to-exe* permite xerar executables das aplicacións python usando PyInstaller a través dunha interface gráfica de usuario.



Imaxe. Aplicación auto-py-to-exe

10.2 Contornas virtuais: virtualenv

A aplicación virtualenv é é unha ferramenta de desenvolvemento Python, escrita por Ian Bicking, que se usa para crear contornas illados en Python, nas que se poden instalar paquetes sen interferir con outras contornas virtuais en virtualenv, nin cos paquetes Python do sistema.

Esta ferramenta foi deseñada para poder desenvolver proxectos de Python que traballan con diferentes versións de paquetes, que doutro xeito entrarían en conflito. Por exemplo, se un programador quere desenvolver dous proxectos con versións diferentes de Django, non é posible ter ambas versións instaladas no directorio da biblioteca Python do sistema. Aquí é onde podemos empregar unha ferramenta como virtualenv., xerando unha contorna para cada proxecto, nas que instalaría a versión compatible con cada unha.

Temos outra aplicación chamada virtualenvwrapper (deseñado por Dough Hellmann), trátase dun complemento moi interesante, pois facilita moito o uso de virtualenv. desde a consola.

Ademais destas dúas ferramentas, a partir da versión 3.3 de Python, a biblioteca estándar inclúe venv como módulo incorporado, o cal implementa unha API similar á de virtualenv.

A continuación imos detallar os pasos necesarios para crear unha contorna virtual Python para o noso proxecto:

1. **Situarse dentro do noso proxecto**: Entramos na carpeta do noso proxecto (sustitúe ruta_ao_proxecto pola do teu proxecto):

   ```
   $ cd ruta_ao_proxecto
   ```

2. **Crear a nosa contorna virtual**: Unha vez dentro crearemos a nosa contorna virtual do seguinte xeito

   ```
   $ virtualenv --python={executable_python} {carpeta_repositorio}
   ```

   Onde:

   - *{executable_python}*: Representa o executable de Python que usaremos (por exemplo: python2, python3, ...). Python debe estar instalado no noso sistema e o binario indicado aquí debe ser accesible (estar no noso PATH de binarios do sistema).
   - *{carpeta_repositorio}*: Representa o nome que queremos dar á carpeta dentro do noso proxecto onde se instalarán todas as dependencias de

python que imos usar. En Linux é unha boa práctica facer que sexa unha carpeta oculta poñendo diante do nome un punto ('.'). Tamén é recomendable dar un nome que sexa característico e defina ben o que é (por exemplo: *.env*), pero podemos darlle o nome que queiramos. Por exemplo poderíamos facelo deste xeito:

Tendo en conta todo isto, procedemos a crear a nosa contorna:

```
$ virtualenv --python=python3 .env
```

3. **Activar a contorna virtual**: Agora xa temos creada a nosa contorna de execución, pero para que as librarías que instalemos con pip ou a versión de python utilizada por defecto sexa a da nosa contorna virtual, primeiro debemos activala:

```
$ source .env/bin/activate
```

4. **Entendendo como traballa coa nosa contorna**: A partires dese momento estamos traballando coa contorna creada para este proxecto en particular. Agora na nosa terminal aparecerá o nome *(.env)* diante do símbolo do sistema no terminal. Todas as execucións de python que fagamos baixo esta contorna, farán referencia á versión de python configurada.

   Porén, todos os módulos (dependencias) que instalemos para o noso proxecto con *pip*, farano dentro do directorio *.env* que se creou no apartado 2. Isto é así se temos activado a nosa contorna virtual (apartado 3), en caso contrario a instalación realizarase no sistema en vez de facelo neste directorio.

   Compre dicir tamén, que a carpeta *.env* que contén as dependencias, déixase fora dos repositorios de código, xa que en realidade o ficheiros e configuracións que ten aí fan referencia ao sistema no que se está traballando. En *git*, por exemplo, engádese ao *.gitignore* do raíz do proxecto, para evitar que sincronice esa carpeta co repositorio.

5. **Crear ficheiro coas dependencias da aplicación**: Con isto xa temos a nosa contorna virtual configurada. Outra boa práctica é ter nun ficheiro o listado de dependencias necesarias do noso proxecto. Crearemos para elo un ficheiro de texto baleiro no que engadiremos esa información. Por exemplo, *requirements.txt*. e para mantelo temos dúas opcións:

   - Engadir as dependencias unha a unha de xeito manual editando o ficheiro.
   - Podemos engadir todas as dependencias instaladas na nosa contorna co

seguinte comando:

```
$ pip freeze > requirements.txt
```

6. **Instalar as dependencias de xeito masivo desde o ficheiro**: Por último indicamos o xeito de instalar masivamente todas as dependencias contidas dentro do ficheiro *requirements.txt*, se por exemplo queremos executar a nosa aplicación noutro equipo:

```
$ pip install -r requirements.txt
```