QtCon BR

Estendendo aplicações C++ com PyQt

Eliakin Costa– KDE
eliakim170@gmail.com

whoami

Eliakin Costa de Almeida
IFBA - KDE - GSoC - PIBIC

Agenda

1. Estender Código C++ com Python
2. SIP
3. PyQt
4. Krita e PyQt

Gratidão

Qt

1.

Estender Código C++ com Python

Por que estender?

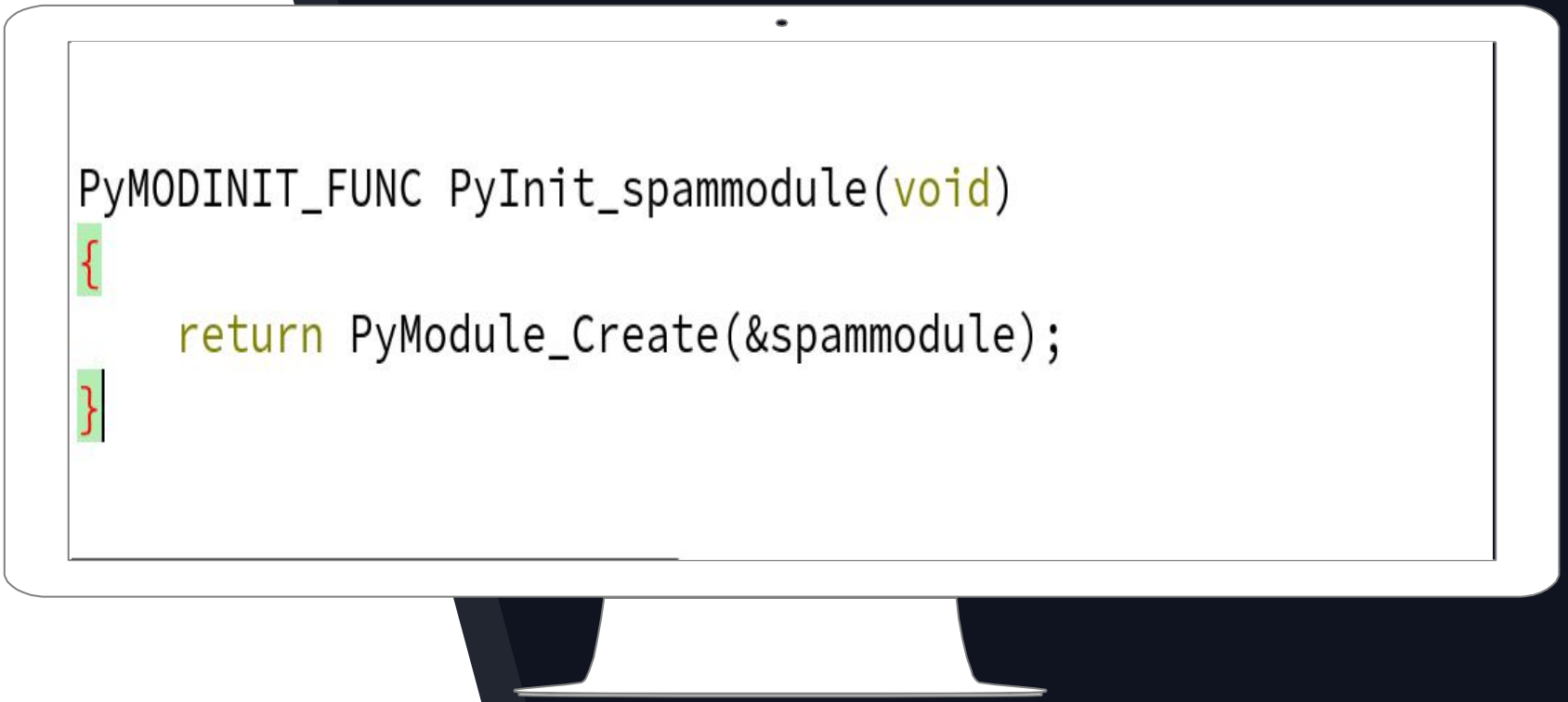
1. Python é mais fácil
2. Quando programando com Qt, podemos utilizar o PyQt
3. C++ pode ser usado apenas quando a performance é um ponto crítico
4. Tornar o seu software acessível a scripts python em run-time

“

Expor C/C++ para Python pode ser árduo e enfadonho.

```c
#include <Python.h>
#include "word.h"

/*
 * Function to be called from Python
 */
static PyObject* reverse_string(PyObject* self, PyObject* args)
{
  char *input;
  char *result;
  PyObject *ret;

  // parse arguments
  if (!PyArg_ParseTuple(args, "s", &input)) {
    return NULL;
  }

  result = reverse(input);

  // build the resulting string into a Python object.
  ret = Py_BuildValue("s", input);
```

```c
  return ret;
}

/*
 * Bind Python function names to our C functions
 */
static PyMethodDef myModule_methods[] = {
  {"reverse_string", reverse_string, METH_VARARGS},
  {NULL, NULL}
};

/*
 * Python calls this to let us initialize our module
 */
static struct PyModuleDef spammodule = {
   PyModuleDef_HEAD_INIT,
   "spammodule",   /* name of module */
   "", /* module documentation, may be NULL */
   -1,       /* size of per-interpreter state of the module, or -1 if the module kee
   myModule_methods
};
```


PyMODINIT_FUNC PyInit_spammodule(void)
{
return PyModule_Create(&spammodule);
}

```
from distutils.core import setup, Extension

module1 = Extension('spammodule',
                    sources = ['spammodule.c'])

setup (name = 'spammodule',
       version = '1.0',
       description = 'This is a example package',
       ext_modules = [module1])
```

Quais os principais problemas dessa abordagem?

Difícil de manter
Alterações feitas na API acessada podem gerar muito impacto na implementação.

Maior Complexidade
Quem implementa o *wrapper* precisa conhecer a API Python/C e gerenciar memória quando necessário.

Muito Repetitivo
Na maior parte dos casos o código se resume a cast dos tipos/objetos do C++ para Python.

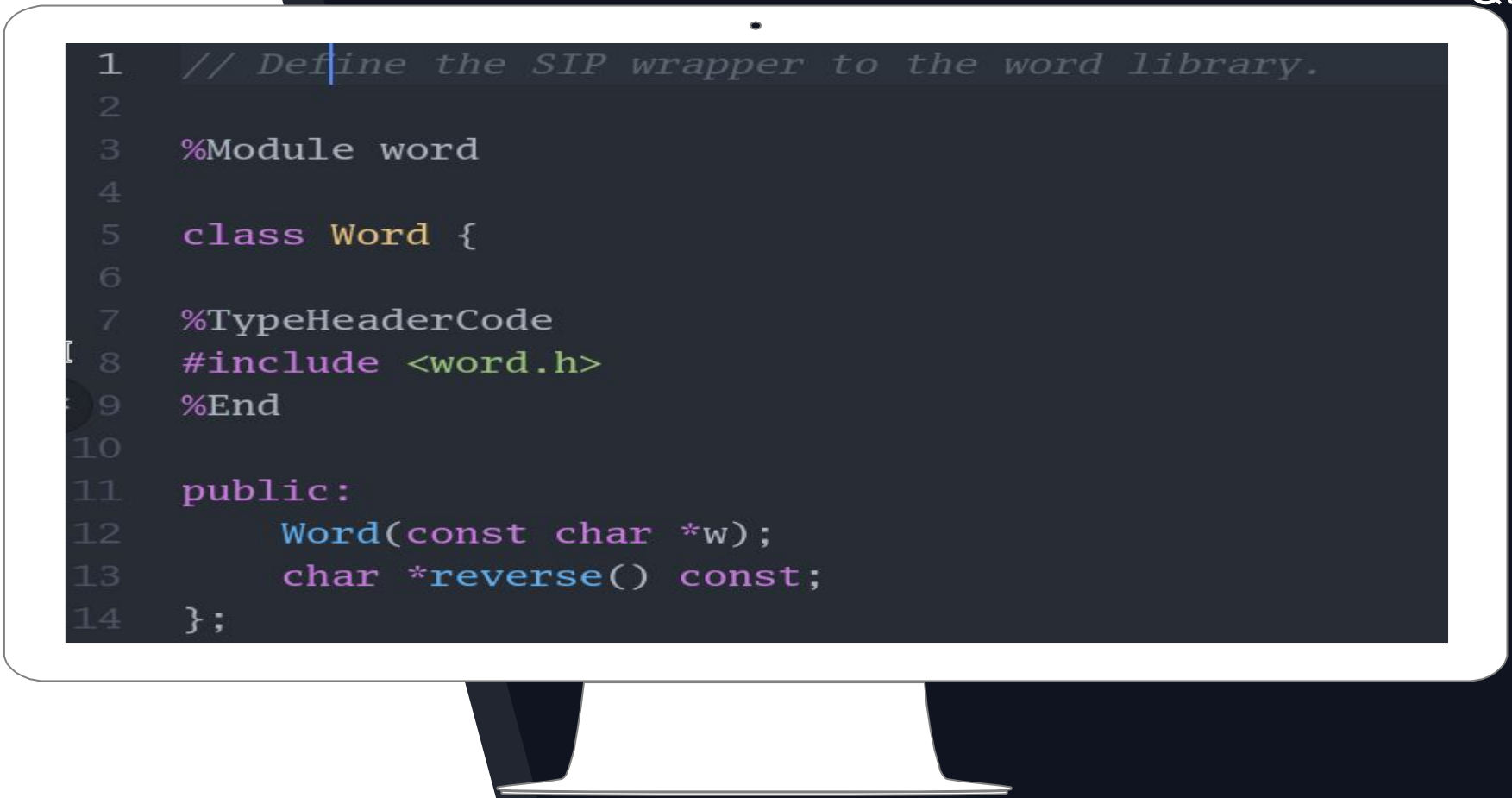
2.

SIP

“

É uma ferramenta para gerar automaticamente *bindings* Python para bibliotecas C e C++.

ESTENDER API C++ COM SIP É FÁCIL


// Define the SIP wrapper to the word library.
%Module word
class Word {
%TypeHeaderCode
#include <word.h>
%End
public:
Word(const char *w);
char *reverse() const;
};


import os
import sipconfig
# The name of the SIP build file generated by SIP and used by the build
# system.
build_file = "word.sbf"
# Get the SIP configuration information.
config = sipconfig.Configuration()
# Run SIP to generate the code.
os.system(" ".join([config.sip_bin, "-c", ".", "-b", build_file, "word.sip"]))
# Create the Makefile.
makefile = sipconfig.SIPModuleMakefile(config, build_file)
# Add the library we are wrapping. The name doesn't include any platform
# specific prefixes or extensions (e.g. the "lib" prefix on UNIX, or the
# ".dll" extension on Windows).
makefile.extra_libs = ["word"]
# Generate the Makefile itself.
makefile.generate()

Quais as vantagens?

1. Todo o código C++ é gerado automaticamente
2. Suporte a geração do MakeFile
3. Suporta Qt, incluindo signals e slots
4. .sip simples de declarar

3.

PyQt

“

PyQt é um conjunto de bindings para Qt que roda em todas as plataformas suportadas pelo Qt, incluindo Windows, OS X, Linux, iOS e Android.

Suporta Python 2 e 3

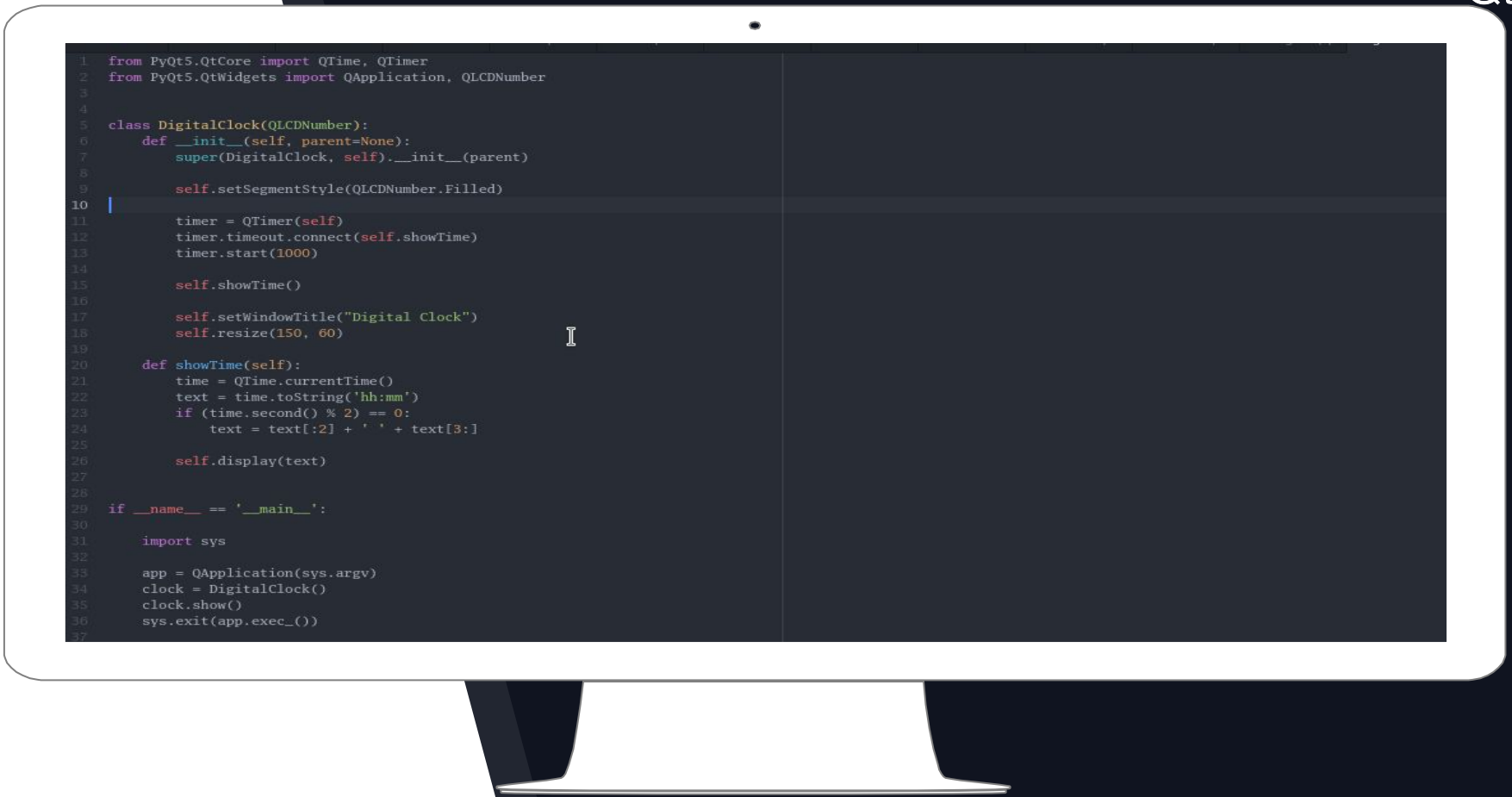
>1000

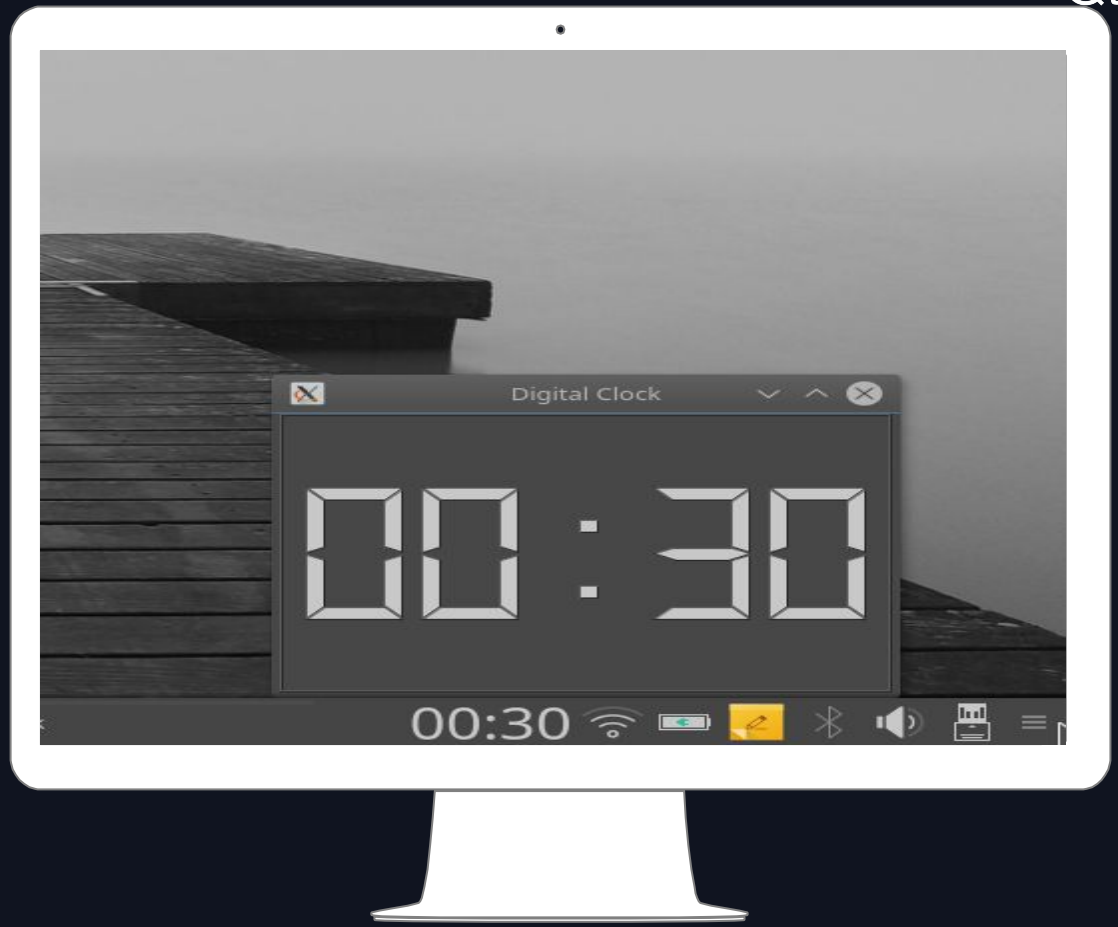
classes implementadas do Qt.

Qt

Qt + Python = <3

Um programador tem todo o poder do Qt, mas pode explorar isso com a simplicidade do Python.


from PyQt5.QtCore import QTime, QTimer
from PyQt5.QtWidgets import QApplication, QLCDNumber
class DigitalClock(QLCDNumber):
def __init__(self, parent=None):
super(DigitalClock, self).__init__(parent)
self.setSegmentStyle(QLCDNumber.Filled)
timer = QTimer(self)
timer.timeout.connect(self.showTime)
timer.start(1000)
self.showTime()
self.setWindowTitle("Digital Clock")
self.resize(150, 60)
def showTime(self):
time = QTime.currentTime()
text = time.toString('hh:mm')
if (time.second() % 2) == 0:
text = text[:2] + ' ' + text[3:]
self.display(text)
if __name__ == '__main__':
import sys
app = QApplication(sys.argv)
clock = DigitalClock()
clock.show()
sys.exit(app.exec_())


Digital Clock
00 : 30
00:30

4.

Krita e PyQt

KRITA É UMA FERRAMENTA PARA PINTURA DIGITAL GRATUITA E OPEN SOURCE

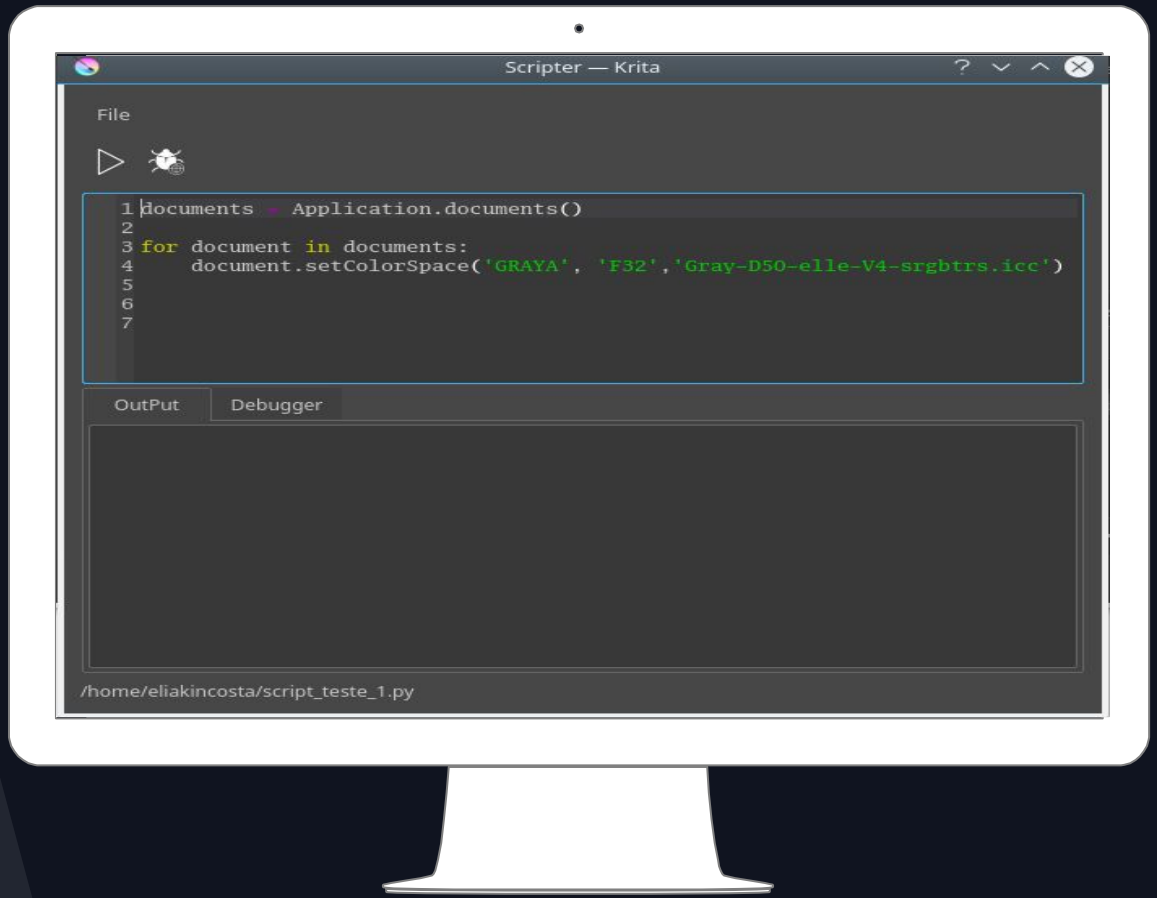
QUAL O PROBLEMA QUE TÍNHAMOS?

Muitas tarefas repetitivas (exportar layers, ajustar tamanho de documento, etc) dentro da ferramenta precisavam ser automatizadas.

Qt

Python Scripting Support

QtCon BR

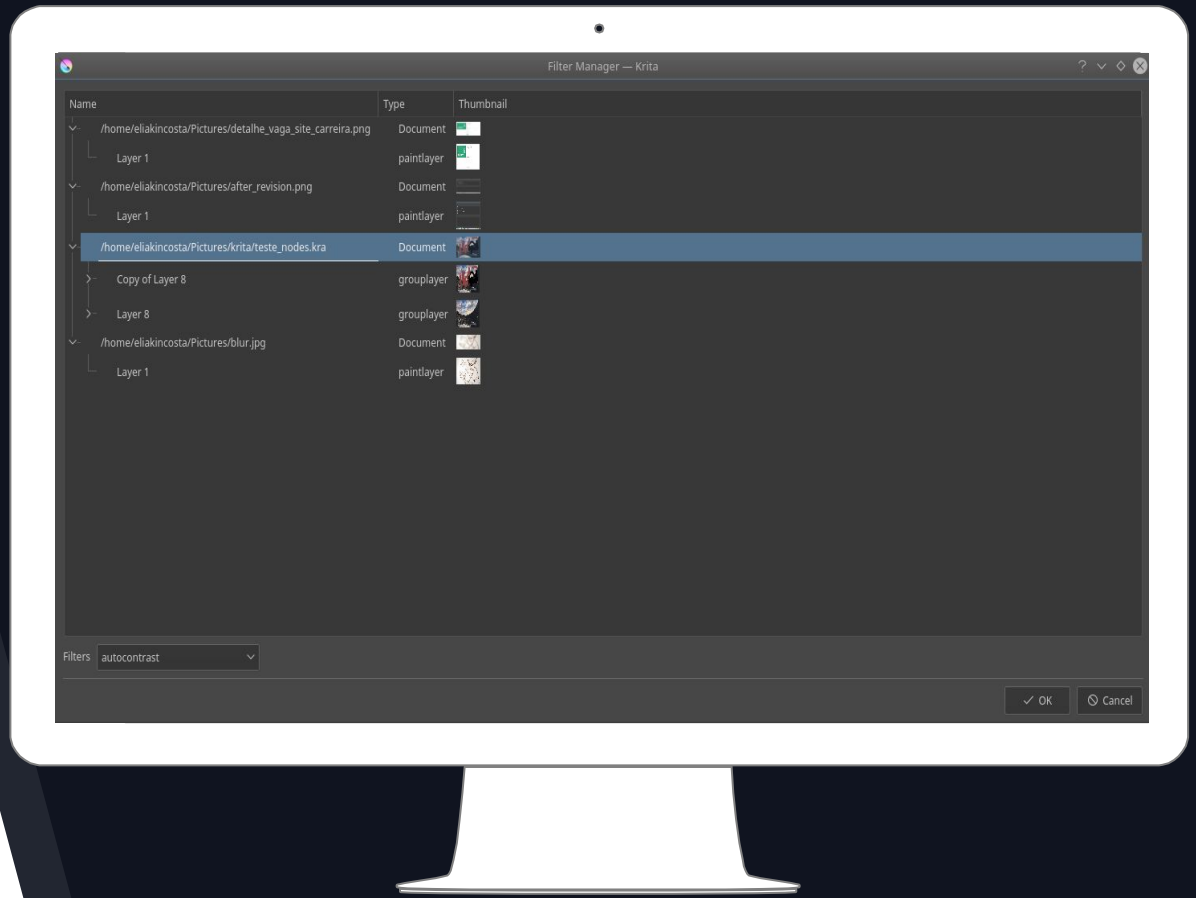


Qt

PYTHON SCRIPTER

QtCon BR

FILTER MANAGER



Considerações finais

1. O PyQt torna possível desenvolver suas aplicações Qt em Python
2. É necessário adquirir a licença comercial do PyQt para distribuir como software proprietário
3. O repositório do PyQt tem muitos exemplos disponíveis

QtCon BR

OBRIGADO!

Perguntas?

eliakim170@gmail.com
github.com/eliakincosta
@eliakin_costa

Qt