



INTERFAZ GRÁFICA PARA OPENCV EN PYTHON

RAINER ARENCIBIA HERNÁNDEZ

PRESENTACIÓN TFG

ÍNDICE

1. Introducción: Motivación, Estado del Arte y Objetivos.
2. Software.
3. Diseño y Codificación – Diagramas.
4. Interfaz gráfica y Secuencia de uso.
5. Filtros digitales: Histograma, Canny y Difuminado Gaussiano.
6. Código - Repositorio.
7. Documentación del código: Paquete, Clase y Método.
8. Métricas.
9. Conclusiones: Mejoras, Aprendizaje y Experiencia.

INTRODUCCIÓN - MOTIVACIÓN

- Interfaz Gráfica Usuario: acceder a las funciones de OpenCV.
- Conocer el campo de la vision artificial.
- Motivaciones adicionales: Python.
- Poner en práctica los conocimientos aprendidos.
 - Diferentes paradigmas de programación.

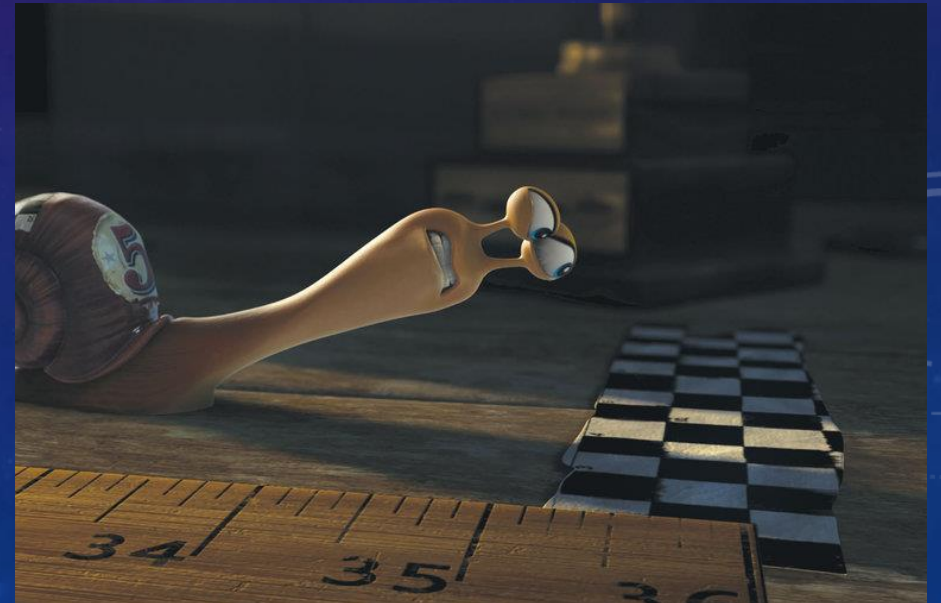


INTRODUCCIÓN – ESTADO DEL ARTE

- **Gimp:**
 - Motores GEGL y GTK+.
 - Permite tratar cada objeto de la imagen de forma independiente(Capas).
 - Precio: Gratis.
- **XnView:**
 - Categorizar y convertir imágenes.
 - Trabaja con 400 formatos.
 - Precio: Gratis.
- **Adobe PhotoShop:**
 - Motor gráfico de Adobe.
 - Permite tratar cada objeto de la imagen de forma independiente(Capas).
 - Precio: Alto. Solo para profesionales. “Estándar de facto”.

INTRODUCCIÓN – OBJETIVOS

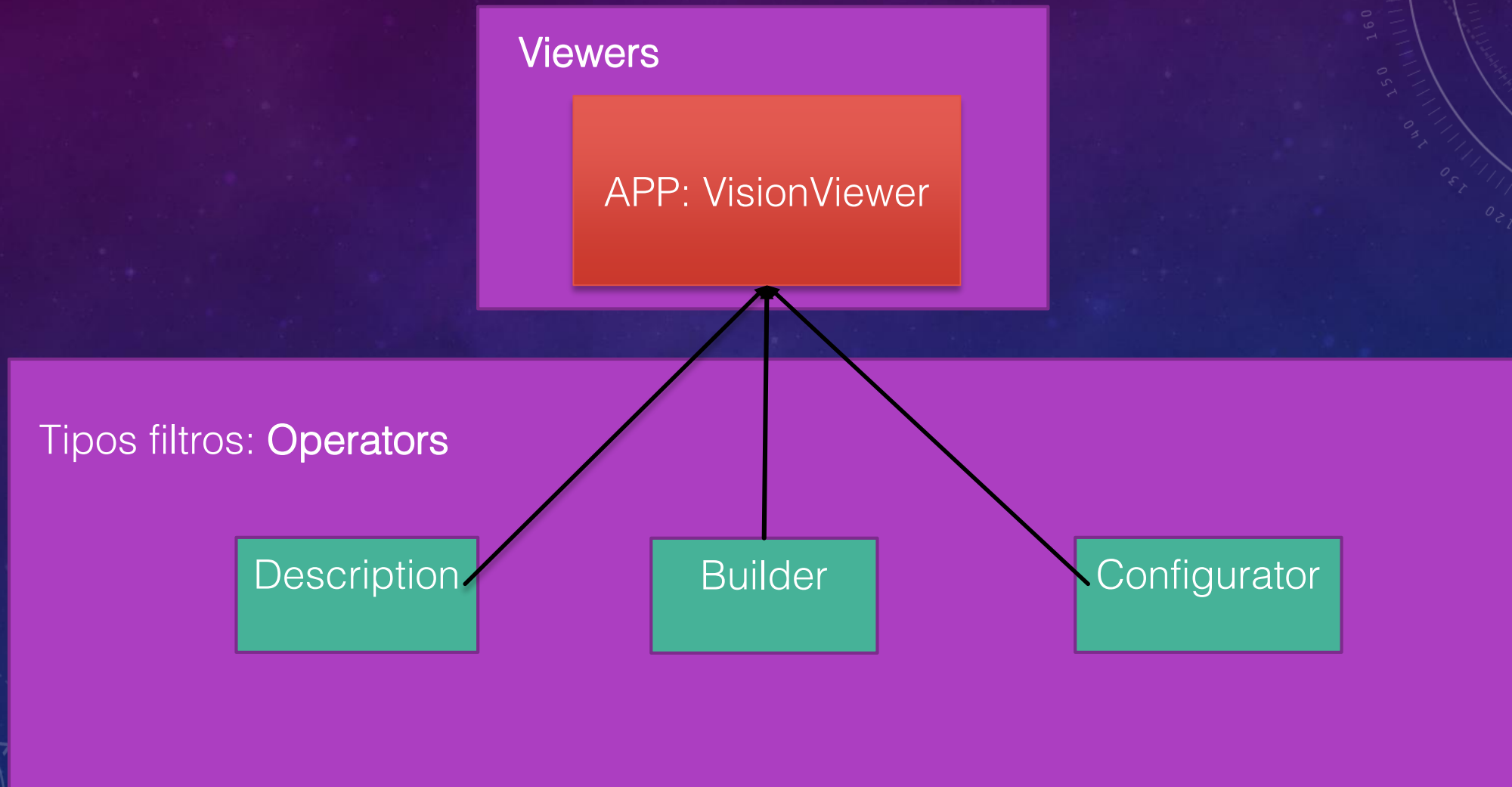
- Desarrollar una aplicación que procesa imágenes en tiempo real.
- Herramienta multiplataforma, para estudiantes y profesionales.
- Interfaz gráfica sencilla e intuitiva.
- Proyecto colaborativo e incremental.



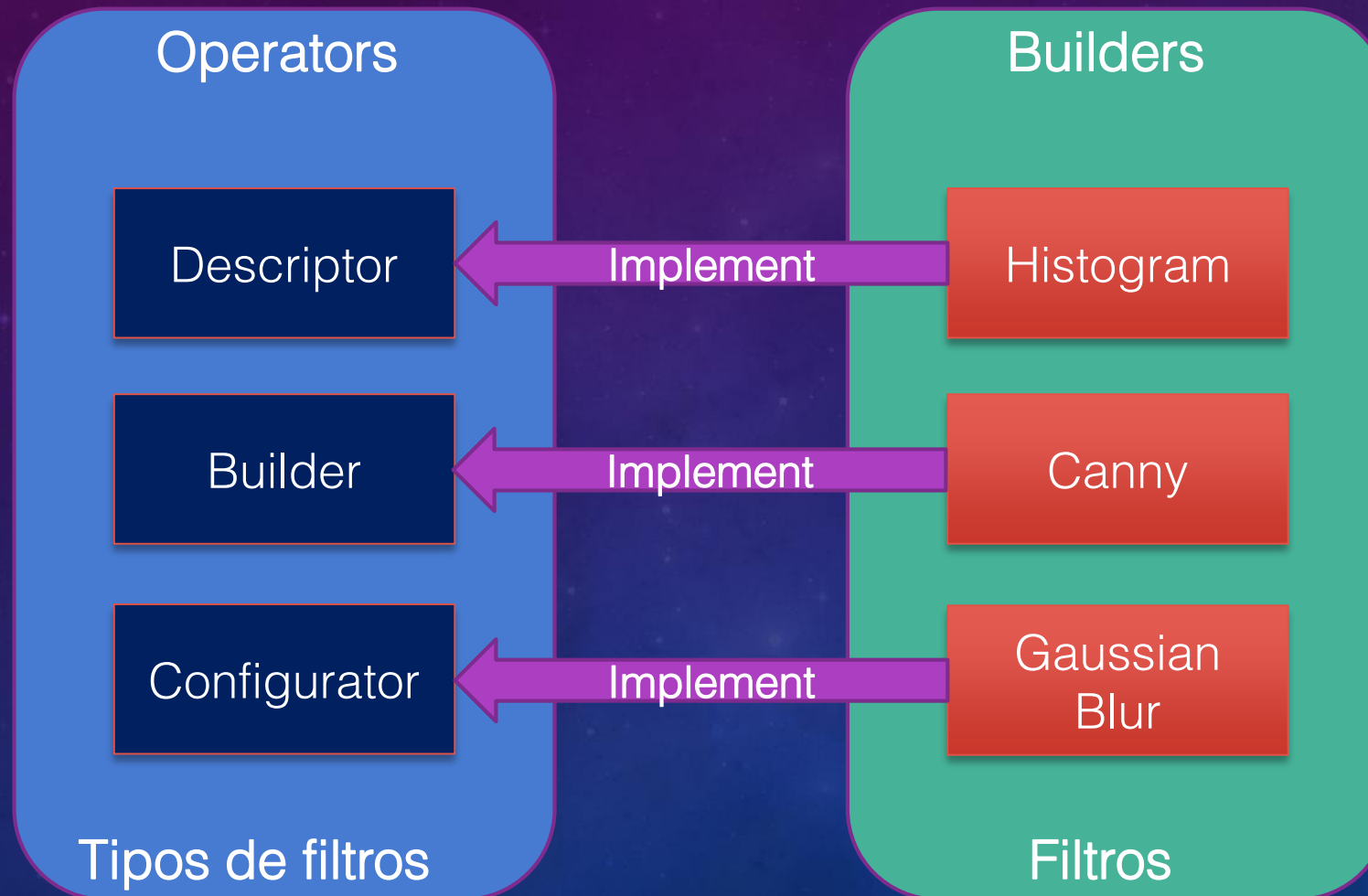
SOFTWARE



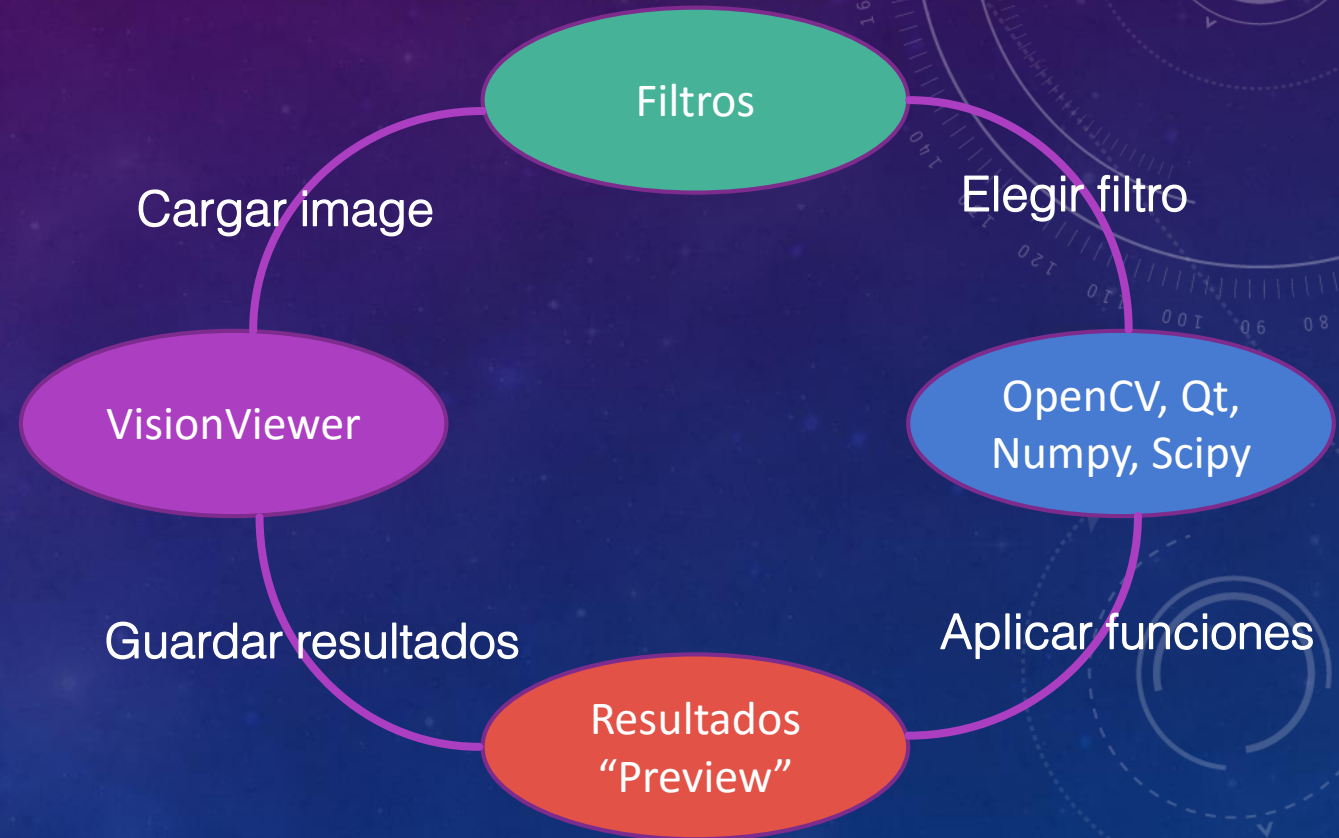
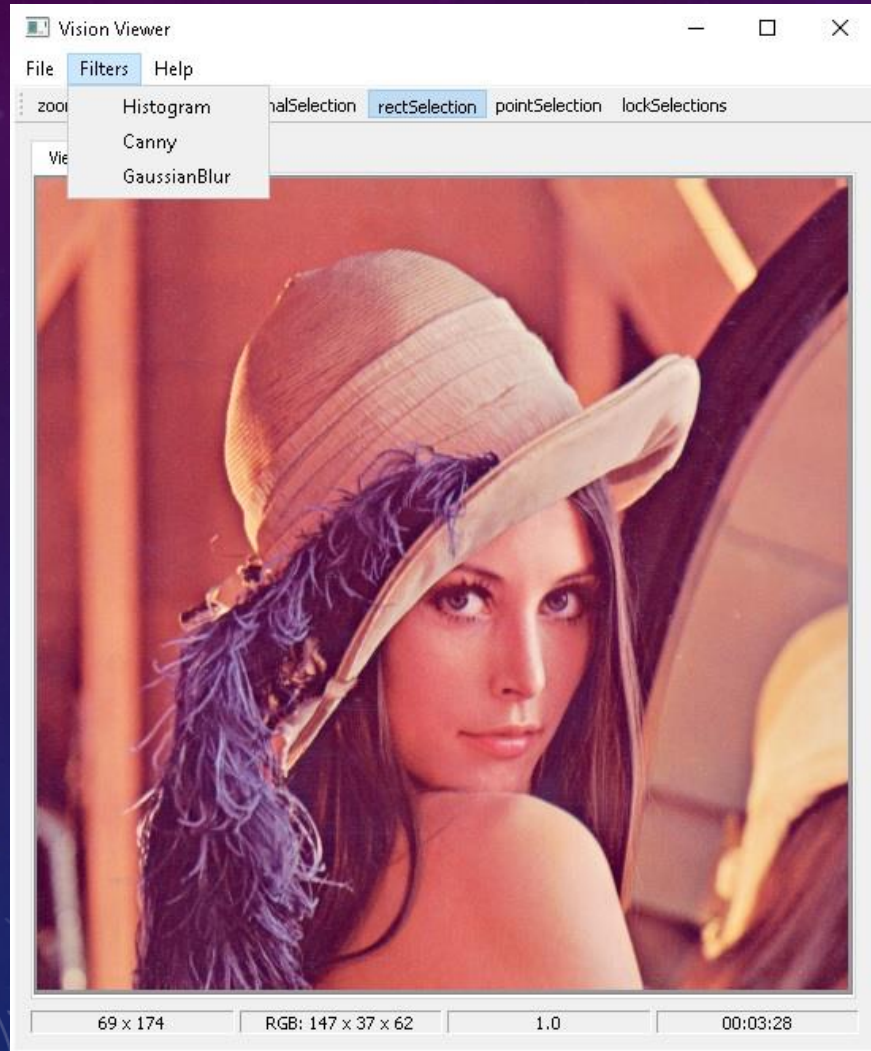
DISEÑO Y CODIFICACIÓN - DIAGRAMA



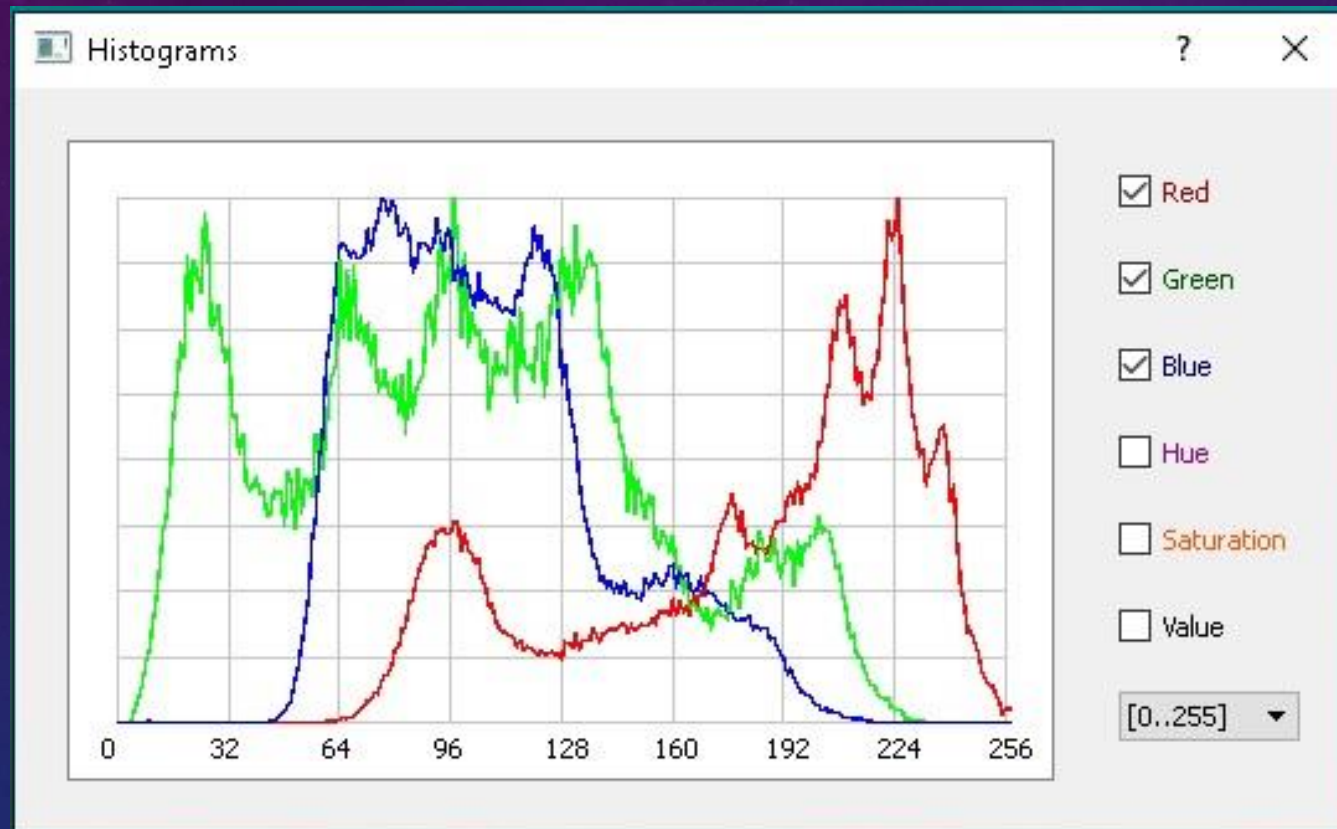
DISEÑO Y CODIFICACIÓN - DIAGRAMA



GUI – SECUENCIA DE USO

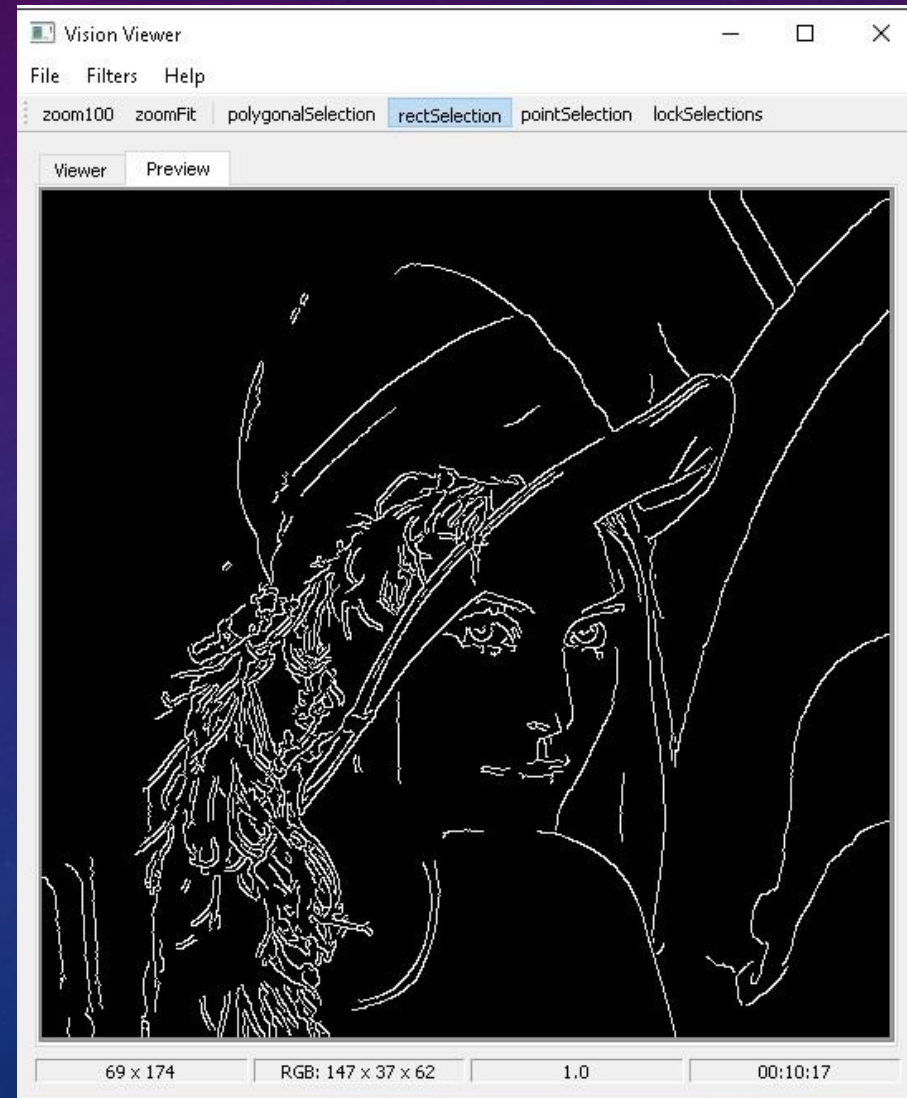


FILTROS DIGITALES - HISTOGRAMA

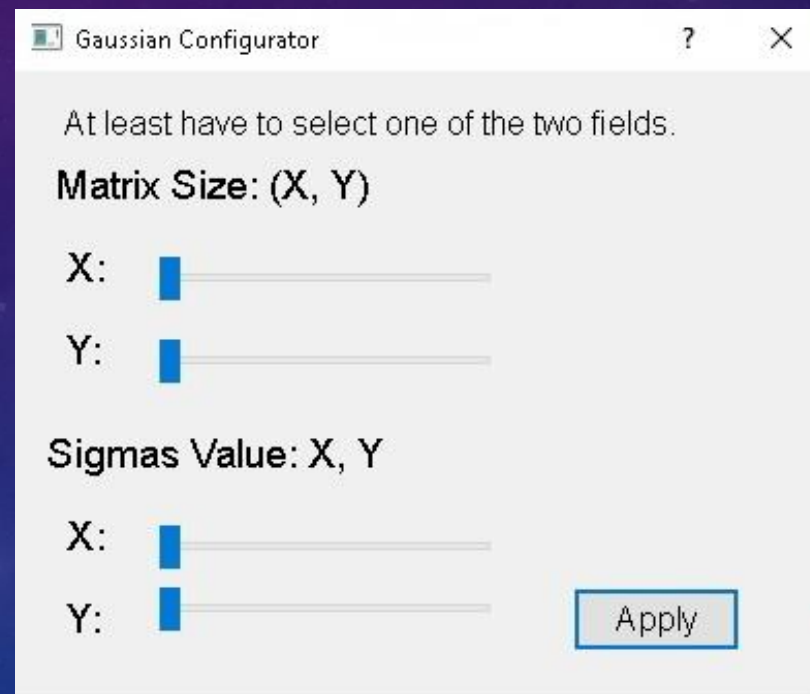
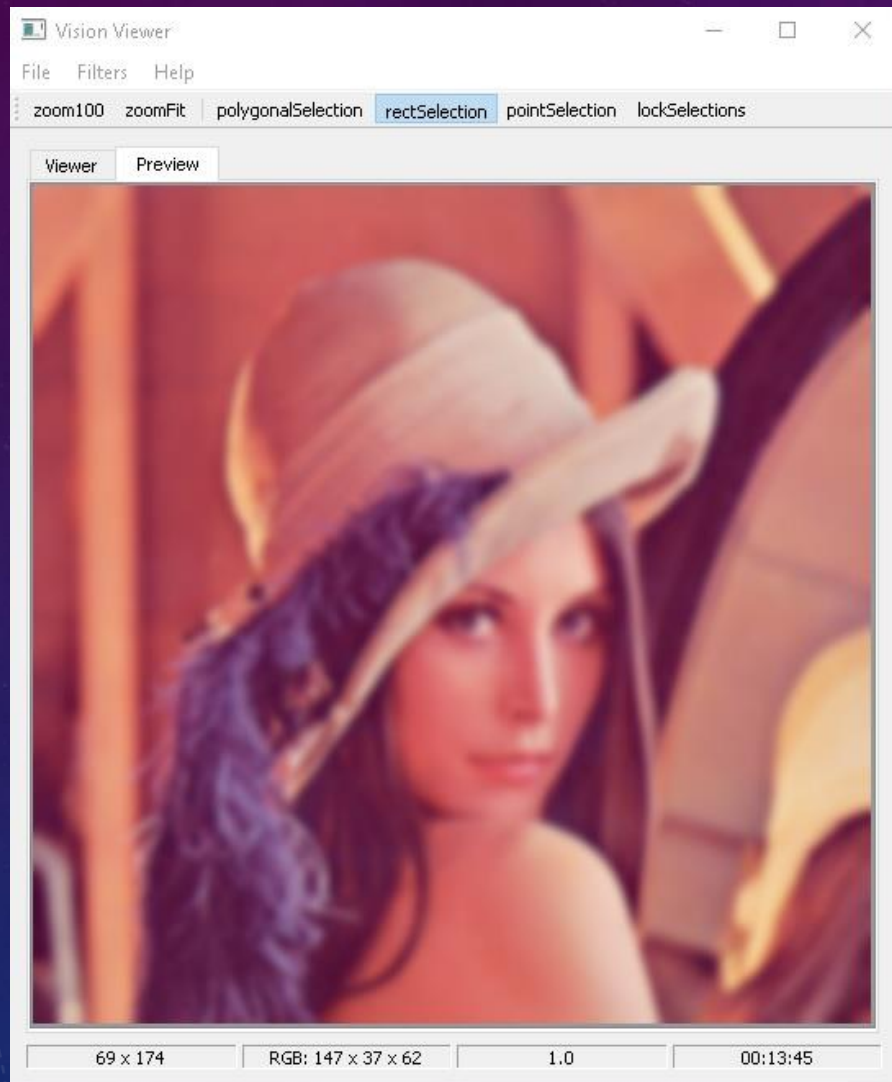


FILTROS DIGITALES - CANNY

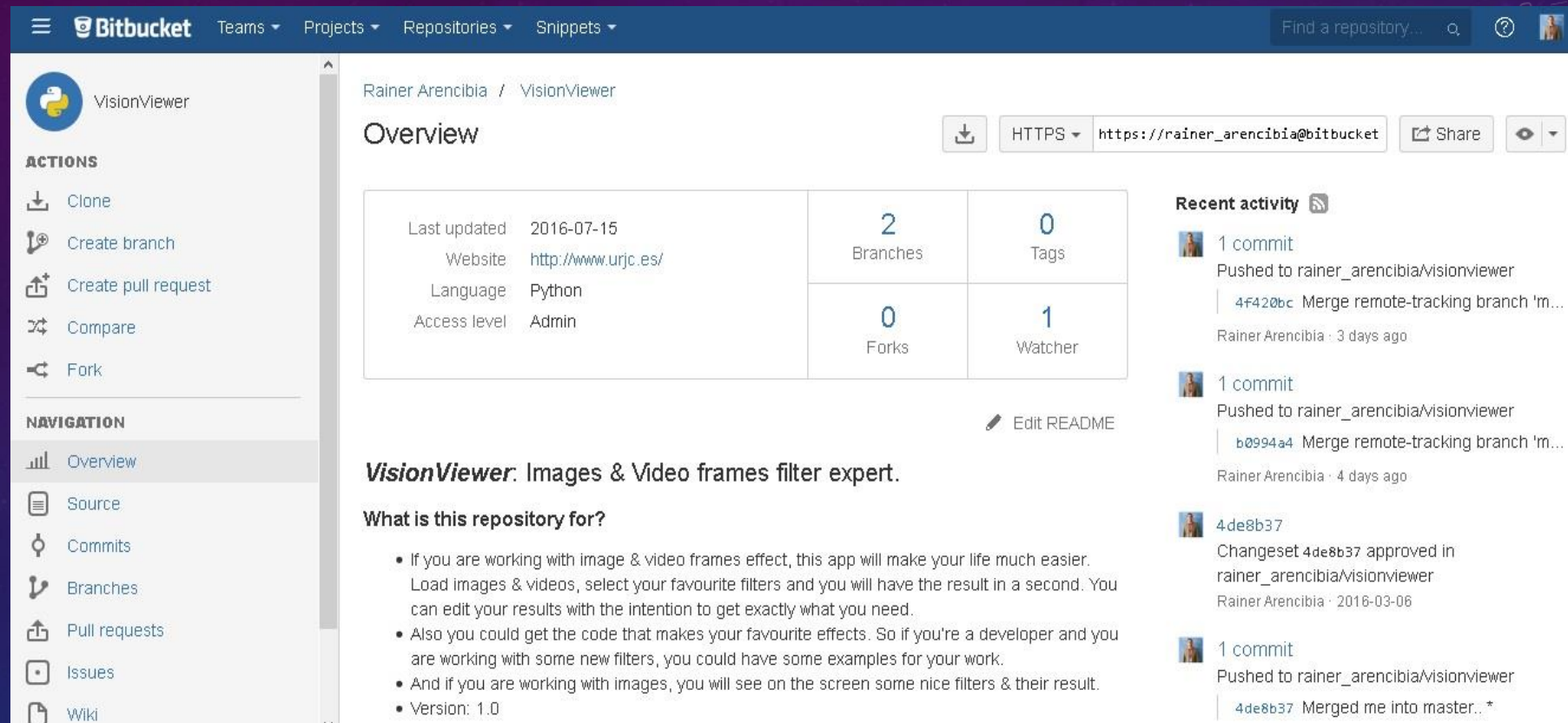
Canny es compatible
con imágenes en
grises.



FILTROS DIGITALES – GAUSSIAN BLUR



CÓDIGO



The screenshot shows the Bitbucket web interface for a repository named 'VisionViewer' by user 'Rainer Arencibia'. The interface includes a top navigation bar with links to Teams, Projects, Repositories, and Snippets. A search bar on the right says 'Find a repository...'. On the left, there's a sidebar with 'ACTIONS' (Clone, Create branch, Create pull request, Compare, Fork) and 'NAVIGATION' (Overview, Source, Commits, Branches, Pull requests, Issues, Wiki). The main content area is titled 'Overview' and shows repository statistics: Last updated 2016-07-15, Website http://www.urjc.es/, Language Python, Access level Admin, 2 Branches, 0 Tags, 0 Forks, and 1 Watcher. Below this, there's a description of 'VisionViewer' as an 'Images & Video frames filter expert' and a list of features. On the right, a 'Recent activity' section shows three commits by Rainer Arencibia, including a merge of a remote-tracking branch and a merge into master.

Bitbucket Teams Projects Repositories Snippets Find a repository...

Rainer Arencibia / VisionViewer

Overview

Last updated 2016-07-15
Website <http://www.urjc.es/>
Language Python
Access level Admin

2 Branches
0 Tags
0 Forks
1 Watcher

Recent activity

1 commit
Pushed to rainer_arencibia/visionviewer
4f420bc Merge remote-tracking branch 'm...
Rainer Arencibia · 3 days ago

1 commit
Pushed to rainer_arencibia/visionviewer
b0994a4 Merge remote-tracking branch 'm...
Rainer Arencibia · 4 days ago

4de8b37
Changeset 4de8b37 approved in
rainer_arencibia/visionviewer
Rainer Arencibia · 2016-03-06

1 commit
Pushed to rainer_arencibia/visionviewer
4de8b37 Merged me into master.. *
Rainer Arencibia · 2016-03-06

VisionViewer: Images & Video frames filter expert.

What is this repository for?

- If you are working with image & video frames effect, this app will make your life much easier. Load images & videos, select your favourite filters and you will have the result in a second. You can edit your results with the intention to get exactly what you need.
- Also you could get the code that makes your favourite effects. So if you're a developer and you are working with some new filters, you could have some examples for your work.
- And if you are working with images, you will see on the screen some nice filters & their result.
- Version: 1.0

El código se ha compartido en un repositorio llamado bitbucket.

DOCUMENTACIÓN - PAQUETE

```
__init__.py x
# -*- coding: utf-8 -*-
__author__ = 'Rainer Arencibia'

""" We have here 4 python packages to organize and structure the project.
1 - Builders
2 - Operators
3 - Viewer
4 - Listeners

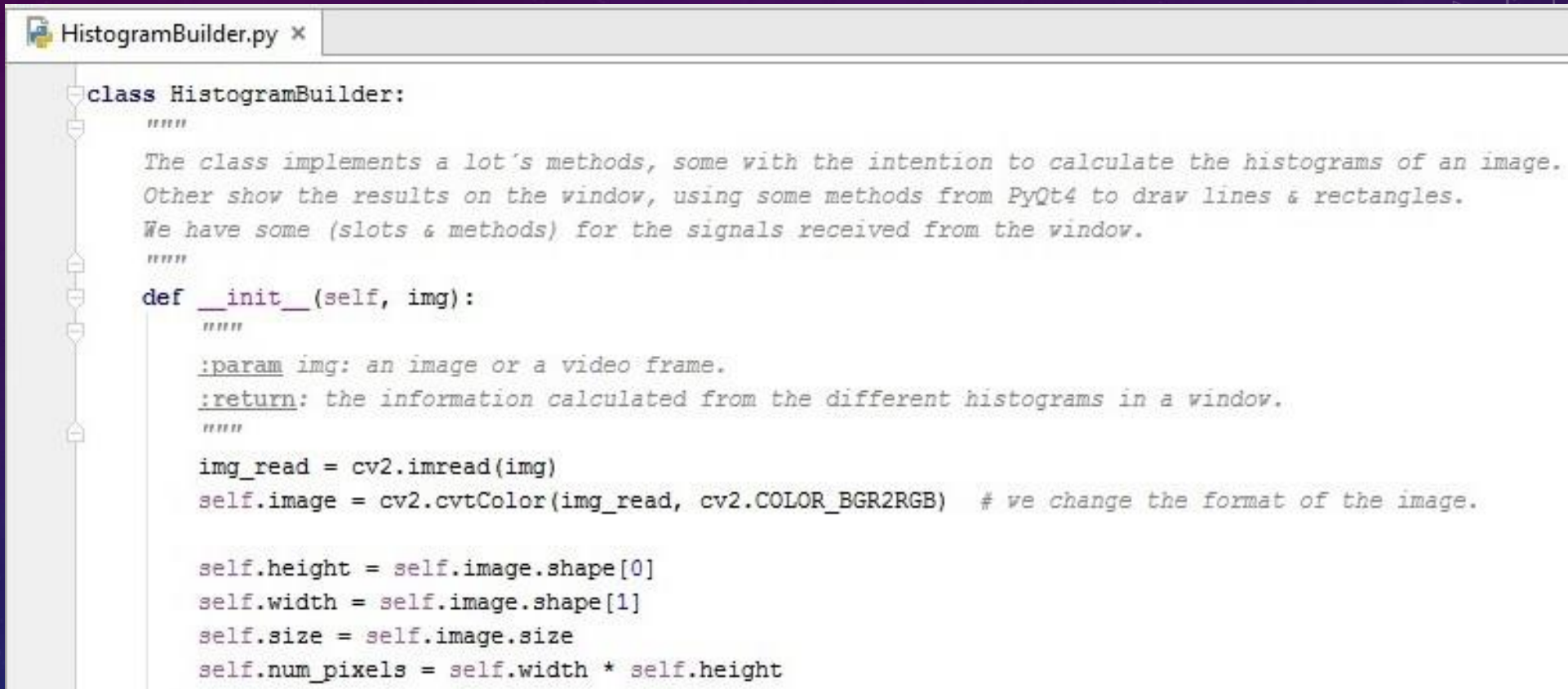
1) We use Builder so save our filters, each one of this filters is saved in different packages by Name.
For example: In the Histogram package we have a Window(Qt Designer - interface.ui) to show the information of the
histograms, we have the code translated in python of the interface, generated by the pyuic4 program & we have
the HistogramBuilder, which received a image and do the process of calculated the histograms, draw each
| histogram & connect the objects("checkbox") of the window to their respectively slots, by his signals.

NOTE: Every developer will leave their code in a python package on this folder.

2) On the Operators package we have 3 categories to organize each filter on the project.
| 1 - Builder
| 2 - Configurator
| 3 - Descriptor
```

El proyecto completo se ha documentado en Inglés.
Siguiendo el “PEP 0257 – Docstring Convention”.

DOCUMENTACIÓN – CLASE Y MÉTODO



The screenshot shows a code editor window titled "HistogramBuilder.py". The code defines a class `HistogramBuilder` and its `__init__` method. The class docstring describes its purpose: calculating histograms and displaying them using PyQt4. The `__init__` method docstring specifies its parameters and return value. The method body shows the initialization of the image, its dimensions, and the total number of pixels.

```
class HistogramBuilder:
    """
    The class implements a lot's methods, some with the intention to calculate the histograms of an image.
    Other show the results on the window, using some methods from PyQt4 to draw lines & rectangles.
    We have some (slots & methods) for the signals received from the window.
    """

    def __init__(self, img):
        """
        :param img: an image or a video frame.
        :return: the information calculated from the different histograms in a window.
        """

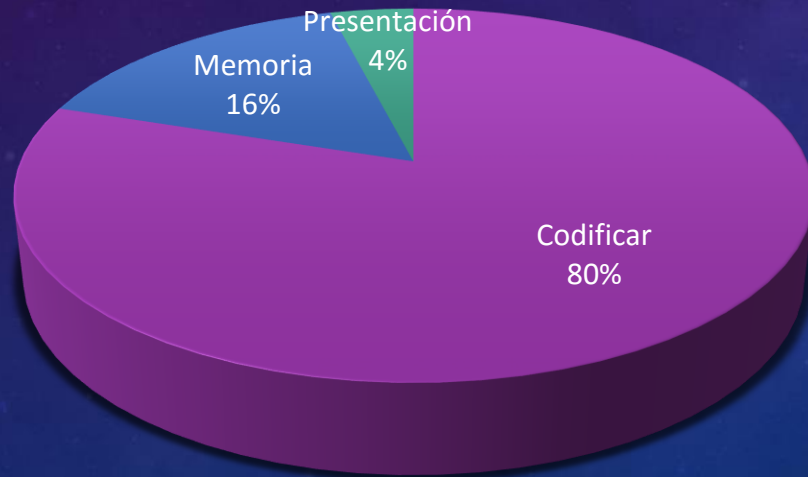
        img_read = cv2.imread(img)
        self.image = cv2.cvtColor(img_read, cv2.COLOR_BGR2RGB) # we change the format of the image.

        self.height = self.image.shape[0]
        self.width = self.image.shape[1]
        self.size = self.image.size
        self.num_pixels = self.width * self.height
```

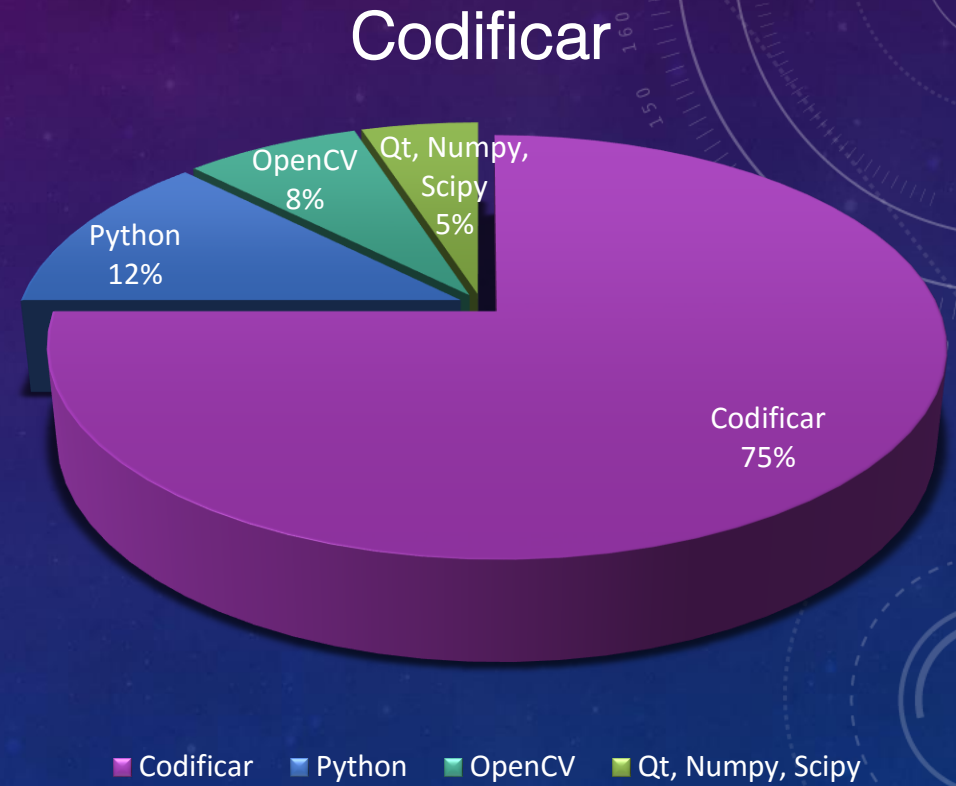
Ejemplo de la documentación de una clase y un método.
Siguiendo el “[PEP 0257 – Docstring Convention](#)”.

MÉTRICAS

- Ficheros creados: 20.
- Numero de clases: 14.
- Líneas de código: 1276.
- Horas de trabajo: $400 + 80 + 10$.



■ Codificar ■ Memoria ■ Presentación



■ Codificar ■ Python ■ OpenCV ■ Qt, Numpy, Scipy

CONCLUSIONES - MEJORAS

- Filtros+.
- Ventana con historial “Navigator”.
- Objeto “Listener”. Filtros compatibles.
- Módulo de idiomas.

CONCLUSIONES - APRENDIZAJE

- Conceptos y definiciones de Visión Artificial.
- Lenguaje Python.
- Librerías OpenCV, Qt, Numpy y Scipy.
- Documentación de un proyecto, "PEP-0257".
- Investigación y corrección de errores.

CONCLUSIONES - EXPERIENCIA

- Toma decisiones: Análisis, Diseño y Codificación.
- Programación con librerías de terceros.
- Superado los obstáculos.
- Cumplido con los objetivos presentados.
- Añadido nuevos elementos:
 - Filtros.
 - Ventana “About us”.
 - Log para errores.

FIN

- Rainer Arencibia Hernández
- +34 663 73 79 51
- rainer85ah@gmail.com
- <https://www.linkedin.com/in/rainerarencibia>



Muchas gracias! ☺

