

# IT-IT Identificação e Transcrição de Imagens em Texto

OCR uma abordagem com Grafos

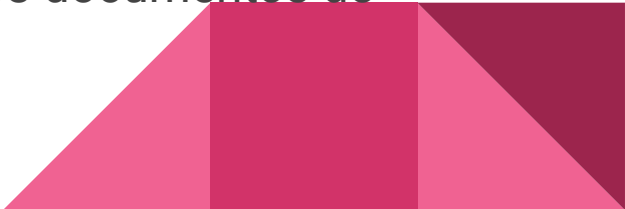
# Membros

- Carlos Humberto Martins Júnior
- Lucas Gabriel Teodoro Araújo
- Paulo Henrique Alves Santos



# Problema

Empresas diferentes utilizam templates diferentes utilizam modelos diferentes de notas, também é comum que uma mesma empresa altere o layout de suas notas com o passar do tempo. Criar modelos de OCR eficazes para cada uma desses modelos é um trabalho exaustivo e que consome muito tempo, além disso esses modelos perdem a precisão mesmo com pequenas alterações nas notas. Reconhecimento de padrões de texto em imagens é um trabalho amplamente abordado na literatura e existem diversas abordagens para esse problema. Utilizar essas técnicas evita redigitação de informações e aumenta a estabilidade do sistema permitindo a extração das informações direto dos documentos de imagem.

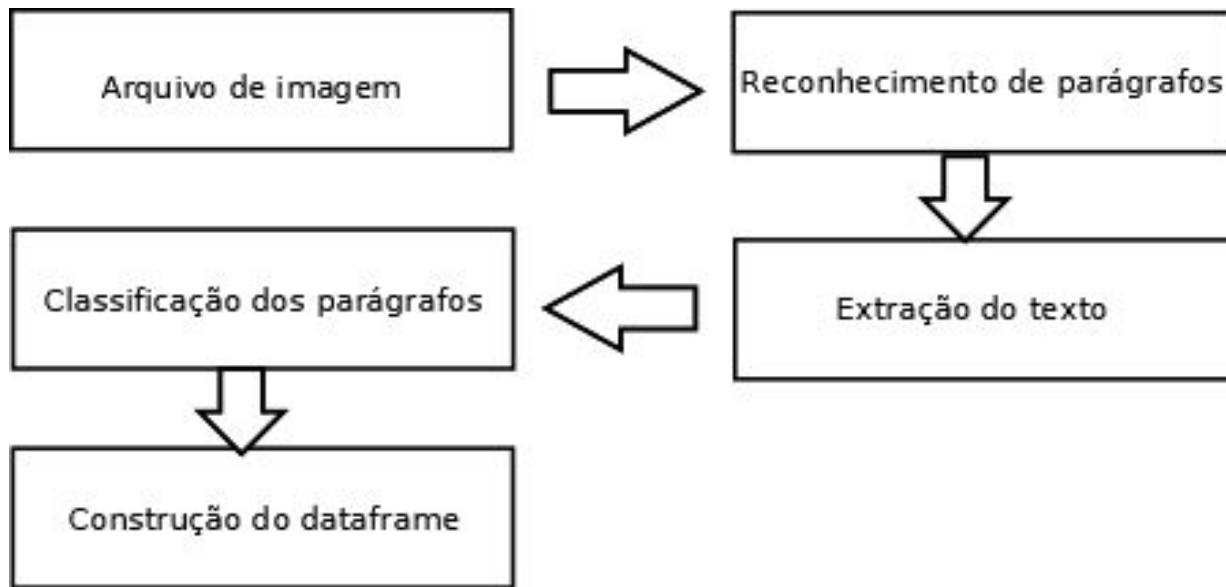


## Solução Proposta

- Dado uma imagem de um formulário o sistema reconhecerá cada bloco, os isolará e rotulara baseado nas informações extraídas do mesmo. Utilizando grafos e reconhecimento de padrões, para a extração dos dados necessários.



# Desenho da Arquitetura



# Bibliotecas Utilizadas

- PyX - Python graphics package
  - Canvas, Path, Color
- Scikit-Learn
- Math
- Pillow(PIL)
  - image
- Pytesseract



# Features Utilizadas

- Python 3.9
- Visual Studio Code
- Poetry
- Scikit-Learn
- Git
- GitHub
- Pyenv



# Imagens da Solução do Problema - Pytesseract

```
>>> import pytesseract
>>> pytesseract.image_to_string('Captura de tela_2021-10-26_15-34-57.png')
'Byer\n\nPoetry\n\x0c'
>>> for i in pytesseract.image_to_boxes('Captura de tela_2021-10-26_15-34-57.png')
'.split('\n'):
...     i = i.split()
...     if len(i)>0:
...         i.pop(0)
...     print(i)
```

```
...
'B'
['0', '107', '28', '139', '0']
'y'
['59', '105', '68', '116', '0']
'e'
['71', '108', '76', '116', '0']
'r'
['81', '108', '97', '116', '0']
'p'
['55', '54', '57', '64', '0']
```



# Imagens da Solução do Problema - Skeleton

```
36
37 def edge1(p,q):
38     c.stroke(path.line(p[0],p[1],q[0],q[1]),
39             [color.rgb.black])
40
41 def edge2(p,q):
42     c.stroke(path.line(p[0],p[1],q[0],q[1]),
43             [color.rgb.blue])
44
45 def point(p):
46     c.fill(path.circle(p[0],p[1],radius),[color.rgb.red])
```

```
for p in points:
    for q in points:
        if p < q:
            theta = sharp(p,q)
            if theta < theta1:
                edge1(p,q)
            elif theta < theta2:
                edge2(p,q)

for p in points:
    point(p)

#retorna imagem do grafo
c.writePDFfile("Beta-skeleton")

#retorna os pontos de ligação do grafo
print(points)
```

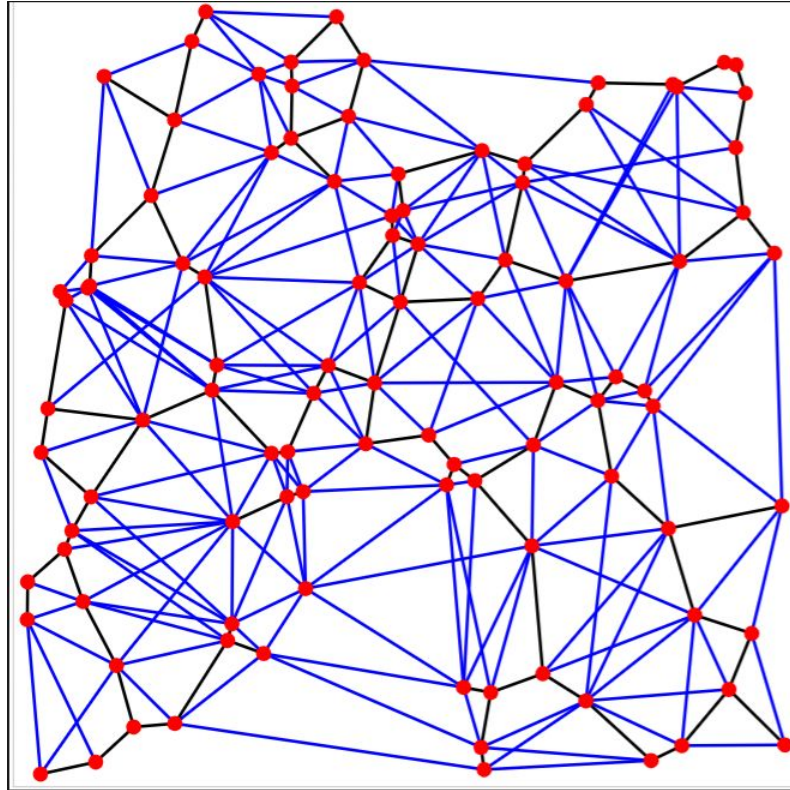
# Saída do Skeleton

```
[(4.4512924254286155, 4.513819260572453), (1.4932689307388558, 3.9473682075673535), (3.1447456548395487, 1.460413279919564), (3.169383191832870576), (2.759155357276774, 1.8879846380288712), (3.791161785983001, 2.8077656366727597), (4.042391211686017, 4.339199529137364), (0.6844570360694824, 1.3454588069091766), (2.852786180733087, 0.7897399039929837), (2.9492475221244687, 1.708947684675986), (0.2128915330497, 1.576603941219662), (1.915418192146368, 4.646012164140974), (1.4176746292576974, 0.09673239865857886), (3.764394692418653, 4.055488512857207), (4.484225082224722, 2.8646127916723083), (4.37557194013243, 2.4675155736330243), (4.202417888935718, 2.3236125430803973), (1.2816792448011516, 1.6714342283602945), (2.670472958616682, 4.706873556383267), (3.8419643878299063, 0.2181900792162006), (3.463738286975895, 1.2041947797520351), (1.0454240275896964, 3.8037158938938624), (1.599125963832555, 3.063257051563969), (1.9077692695308262, 3.711807410863703), (3.0542634204193666, 0.664552977474101), (4.351316980782726, 1.661467799779552), (4.25237752417884, 3.516440413178128), (3.508586060868657, 2.7943417461110496), (3.6590952393781606, 3.962507311581958), (1.4993456622246755, 4.6996009356679895), (3.624609948364448, 2.565758676686138), (4.165301971251619, 0.31581430175127045), (3.769260649486535, 2.2864808021654124), (4.886527014564709, 1.3059896706705114), (2.165901215821733, 3.395966009934452), (1.1588326243099345, 0.1116120743066501), (4.023445862789431, 2.747299214488069), (0.8459761110747777, 0.48239977408532697), (2.1814258619590987, 3.462789295233586), (3.75227999743092, 2.8791536337624137), (4.8635648135586305, 1.0101540489530332), (3.9197743056270036, 3.0574837113572)]
```

-



# Grafo de Saída



# Possíveis Melhoramentos

- Parte do classificador feito em HardCode pode ser mudado para SoftCode.
- Otimização do desempenho.
- Existe casos patológicos onde é preciso rodar em  $O(n^2)$ , no caso de interseção das caixas, sendo possível melhorias.
- Foi notado que o Pytesseract demora um tempo considerável para retornar o resultado.



# Referências

- Beta-skeleton: <https://en.wikipedia.org/wiki/File:Beta-skeleton.svg>
- Pytesseract 0.3.8: <https://pypi.org/project/pytesseract/>
- <https://scikit-learn.org/stable/>
- <https://pyx-project.org/>
- <https://pillow.readthedocs.io/en/stable/>
- [https://www.w3schools.com/python/module\\_math.asp](https://www.w3schools.com/python/module_math.asp)
- <https://docs.python.org/3/library/math.html>
- <https://github.com/pyenv/pyenv>



# GitHub

<https://github.com/newGabriel/TaikaiAmbev>

