Projeto Integrador 2020
Indicadores de Qualidade e Negócio

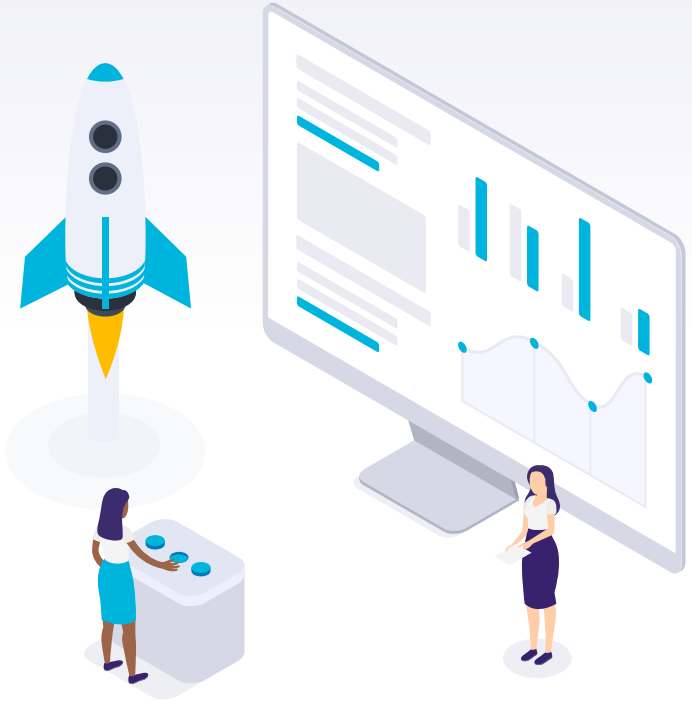
Integrantes:

André Luis dos Santos Fagundes
José Danrley da Silva
Luis Guilherme Belem de Sousa
Matheus Amauri de Jesus Campos
Washington Henrique Fernandes de Sousa

1

Ideia Inicial

Objetivo inicial do time



Objetivo inicial

- Construir uma aplicação Web que realiza o seguinte processo: recebe os lotes de dados, analisa, trata e exibe os indicadores correspondentes;
- Manipular a analisar os dados diretamente de um banco de dados em SQL, integrado à aplicação construída.

Motivos de escolha do objetivo inicial

- Observamos que os dados enviados para nós foram extraídos de um banco de dados utilizando uma Query em SQL, que estava disponível no documento;
- A interface Web é ampla no mercado e facilitaria o acesso aos dados pelos membros da organização.

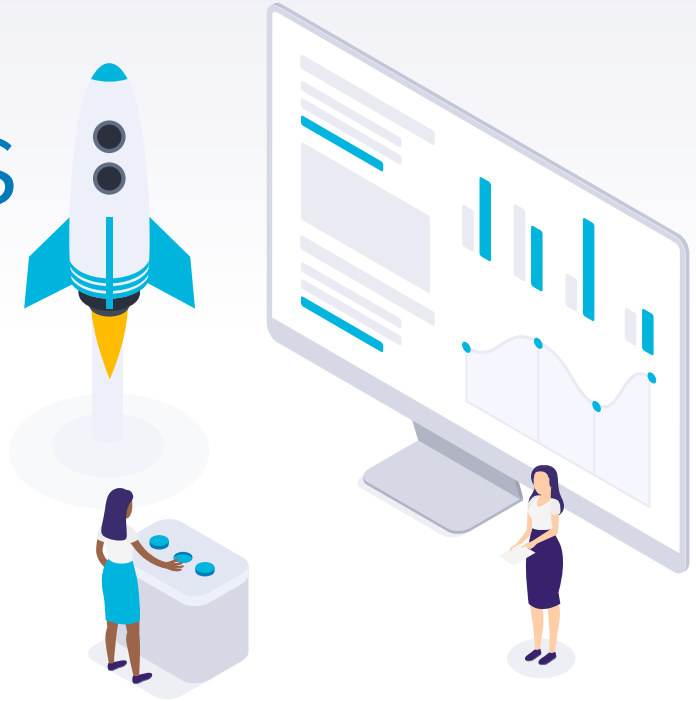
Desenvolvimento da ideia

- Desenvolvemos funções em Python para extrair alguns requisitos dos indicadores solicitados pelo nosso cliente;
- Os dados foram convertidos para um banco de dados em MySQL;
- Na IDE PyCharm, o nosso programa se conecta ao banco de dados com as tabelas convertidas e extrai parâmetros para a construção de indicadores.

2

Alterações após a 1ª reunião com o cliente

Adaptação do projeto frente às novas solicitações do SPC Brasil



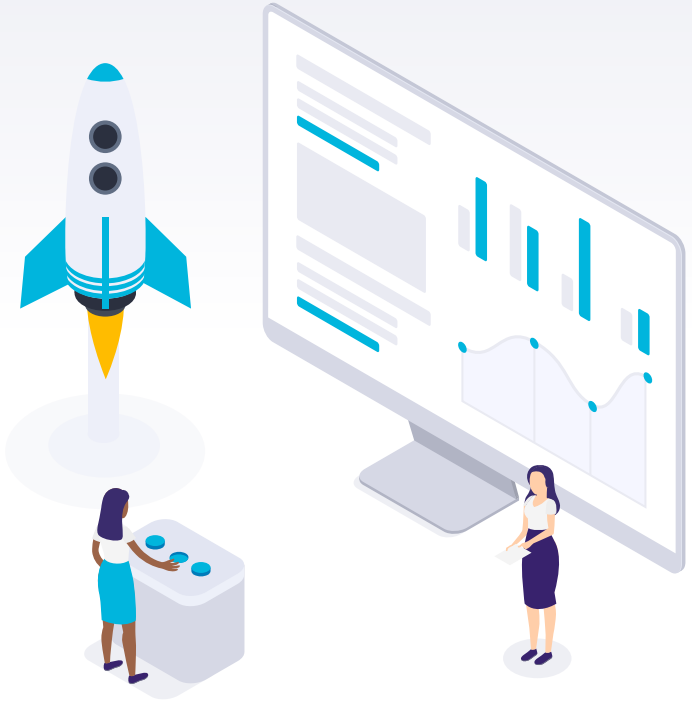
Adaptação de Interface

- Com base nas ressalvas e dificuldades de aprovação e implementação de uma aplicação Web;
- Para a criação dos dashboards solicitados pelo cliente, a ferramenta Microsoft Power BI está sendo estudada para aplicar os algoritmos criados e geração dos relatórios.

3

Funções desenvolvidas até o momento

Código desenvolvido para analisar as tabelas recebidas e estruturar os primeiros indicadores.



Funções desenvolvidas

Ajuste de dígitos no CNPJ

- Corrige os complementos do CNPJ que podem estar com menos de 6 dígitos, aplicando zeros para viabilizar a contagem de dígitos no dado.

programacao.py

```
# Funções:

# Função: Verifica a quantidade de digitos do CNPJ:
def digitos_cnpj(lista):
    CNPJ = []
    for z in lista:
        if len(z[1]) == 8:
            if len(z[2]) <= 6:
                complemento = str(z[2])
                while len(complemento) < 6:
                    complemento = '0' + complemento
            cnpj_pessoa = str(z[0]) + "|" + str(z[1]) + "-" + complemento
            CNPJ.append(cnpj_pessoa)
    return CNPJ
```

Funções desenvolvidas

Filtro e contagem de CNPJ's duplicados

- Cria uma lista dos documentos duplicados para facilitar a remoção e contagem destes itens.

programacao.py

```
# Função: Contabilizando os duplicados (Função especifica para o CNPJ):
def duplicados_cnpj(lista):
    duplicado_id = []
    correto_id = []
    correto = []

    for z in lista:
        CNPJ = z[4:]
        if CNPJ not in correto:
            correto.append(CNPJ)
            correto_id.append(z)
        else:
            duplicado_id.append(z)
    return duplicado_id
```

Funções desenvolvidas

Função genérica para retornar os itens duplicados

- Mecanismo genérico para contabilizar e separar valores duplicados, que pode ser usadas em diferentes tabelas e colunas.

programacao.py

```
54  # Função para verificar os duplicados:
55  def duplicados(lista):
56      duplicado = []
57      correto = []
58
59      for z in lista:
60          if z not in correto:
61              correto.append(z)
62          else:
63              duplicado.append(z)
64      return duplicado
65
```

Funções desenvolvidas

Função para retornar uma lista dos campos nulos

programacao.py

```
68  # Função para verificar se o campo esta nulo:
69  def nulo(lista):
70      nulo = []
71
72      for z in lista:
73          for y in z:
74
75              # Programação para a tabela fonte, modalidade e pagamento:
76              if x == fonte or x == modalidade or x == pagamento:
77                  if y == "":
78                      nulo.append(str(z[0]))
79
80              # Programação para a tabela movimento:
81              if x == movimento:
82                  if y == "" and not y[9] == "E01" and not y[9] == "D01":
83                      nulo.append(str(z[0]))
84
85              # Programação para a tabela operação:
86              if x == operacao:
87                  if y == "" and not y[7] == "C01":
88                      nulo.append(str(z[0]))
89      return nulo
```

Funções desenvolvidas

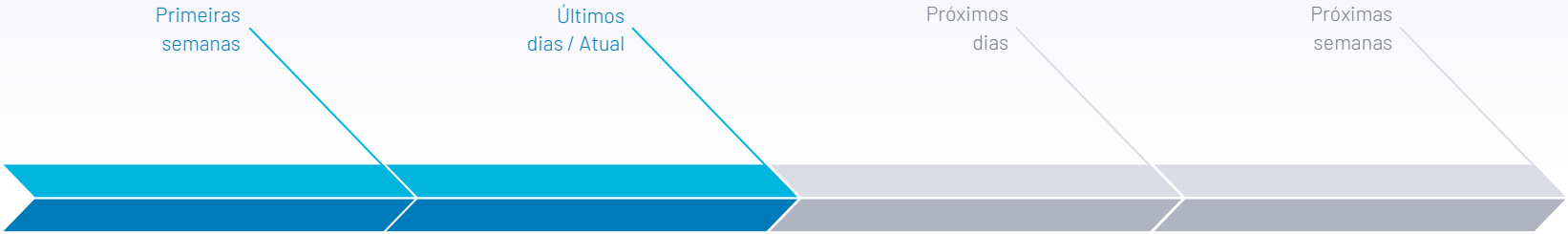
Função de cálculo dos valores nas tabelas movimento e de operação.

- Mecanismo que após criar um dicionário com o valor de cada modalidade, separa o valor movimentado para cadastrados e não cadastrados no Cadastro Positivo.

programacao.py

```
    # Função para conta os valores da tabela movimento e operação:
def contar_quantidade(lista):
    modalidade_dicionario = {
    'A01' : [0, 0],
    'A02' : [0, 0],
    'A04' : [0, 0],
    'A05' : [0, 0],
    'A99' : [0, 0],
    'B01' : [0, 0],
    'B02' : [0, 0],
    'B03' : [0, 0],
    'B04' : [0, 0],
    'B05' : [0, 0],
    'B06' : [0, 0],
    'B07' : [0, 0],
    'B99' : [0, 0],
    'C01' : [0, 0],
    'D01' : [0, 0],
    'E01' : [0, 0],
    'E02' : [0, 0],
    'F01' : [0, 0],
    'G01' : [0, 0],
    }
    if x == operacao:
        for y in lista:
            if modalidade_dicionario[y[7]]:
                modalidade_dicionario[y[7]][0] += int(y[4])
                modalidade_dicionario[y[7]][1] += int(y[5])
    return modalidade_dicionario
```

Resumindo o processo e passos futuros



Convertemos os dados em banco de dados SQL

Criamos uma conexão com a IDE para manipulação dos dados via nosso algoritmo em Python

Adequar o código a todos os requisitos de tratamento e levantamento dos indicadores

Em andamento

Aplicar os resultados em ferramentas de análise de dados para exportação de relatórios

Em breve

Obrigado pela atenção