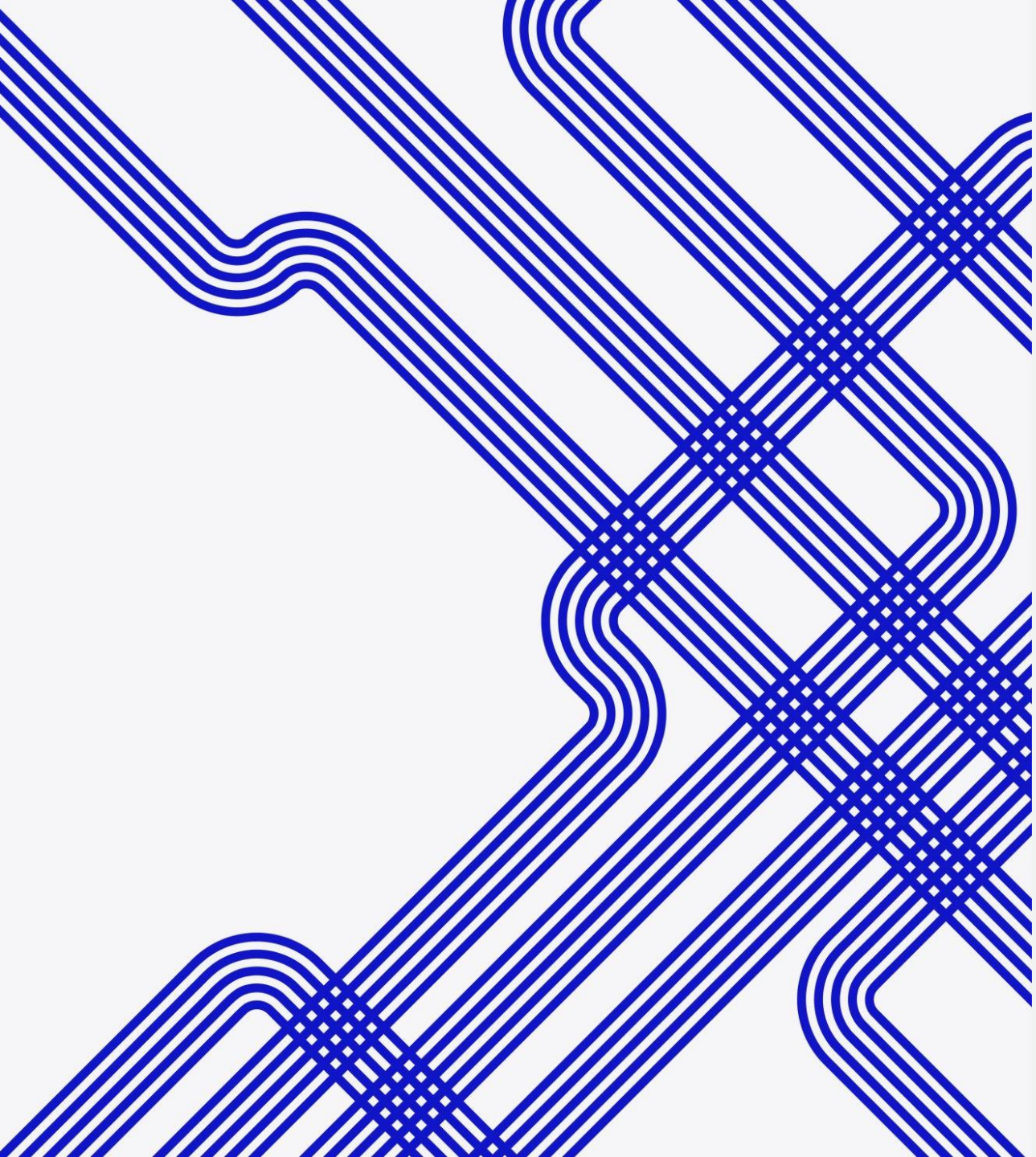


Desafio Data Machina - Solucionário

Ian Lucas Ramos de Carvalho dos Santos - Ciências Moleculares – Universidade de São Paulo

Problema 1 – API Sequência de Fibonacci

Enunciado – Criar uma API que recebe um valor inteiro n e retorna o n-ésimo termo da sequências de Fibonacci.

Solução - API Sequência de Fibonacci

- Solução:

$$
\begin{aligned}
F_0 &= 0 \\
F_1 &= 1 \\
F_2 &= F_1 + F_0 \\
F_3 &= F_2 + F_1 \\
F_4 &= F_3 + F_2
\end{aligned}
$$

```
F_n_1 = 1
F_n_2 = 0

if n == 0: return F_n_2
if n == 1: return F_n_1

for i in range(n-2):

    F_n_1_novo = F_n_1 + F_n_2
    F_n_2_novo = F_n_1

    F_n_1 = F_n_1_novo
    F_n_2 = F_n_2_novo

F_n = F_n_1 + F_n_2
```

Solução - API Sequência de Fibonacci

Criando a API:

- web framework flask
- modulo fibo.py

Requests:

- Via Postman
- Método POST (HTTP)
- Body (JSON) : {"n": numero}
- Resposta da API (JSON) : {"term": n_term}

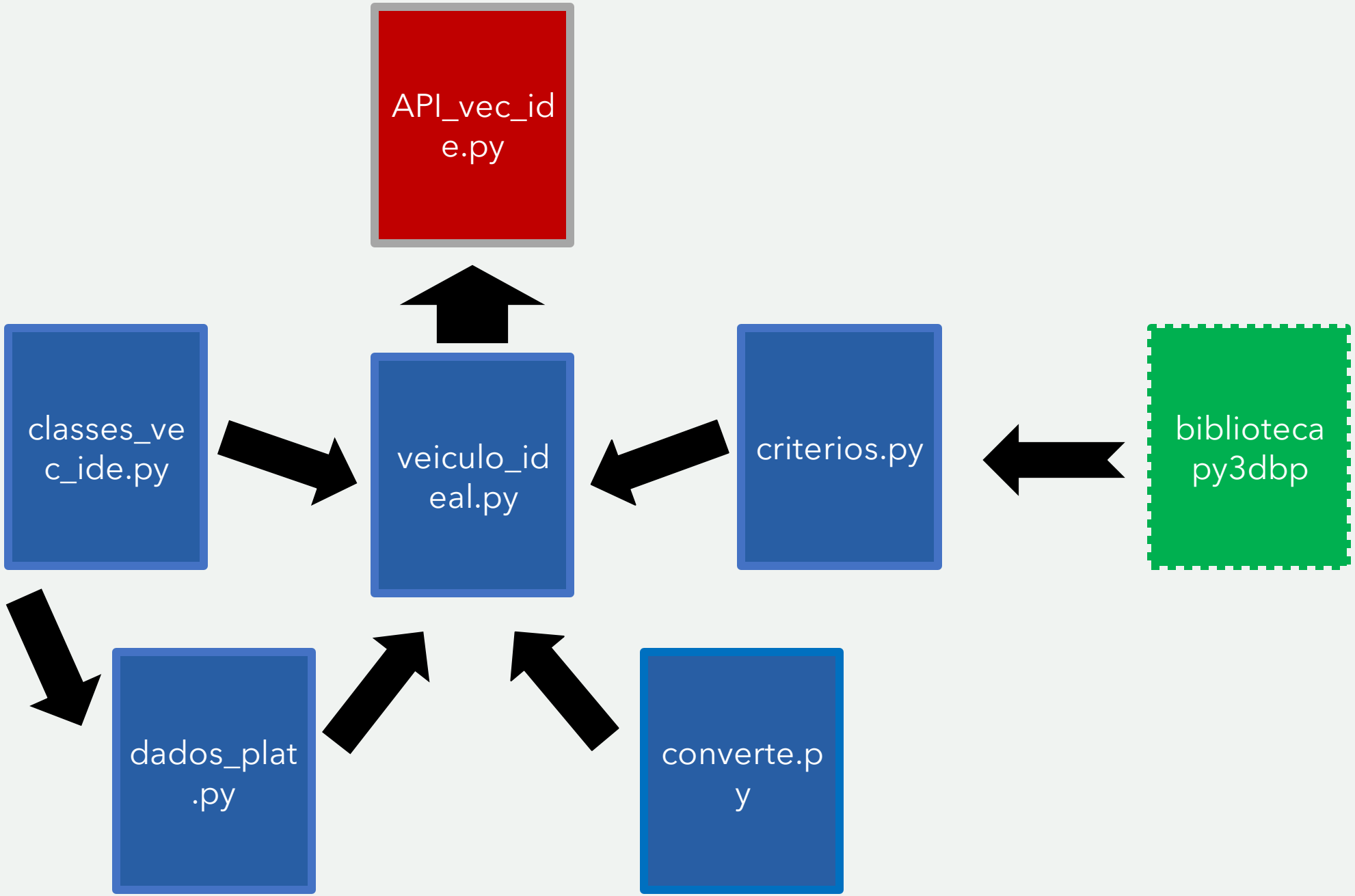
Solução - API Sequência de Fibonacci

```python
from flask import Flask, request
from fibo import Fibo

app = Flask("Fibonacci")

@app.route('/fibonacci', methods= ['POST'])
def fibonacci():

    number = request.get_json()

    n = number['n']

    n_termo = Fibo(n)

    resposta = dict(termo = n_termo)

    resposta_json = jsonify(resposta)

    return resposta_json

app.run()
```

Problema 2 – API Veículo Ideal

Enunciado – Criar uma API que recebe um conjunto de itens, com suas dimensões e peço, e determina quais são os veículos ideais para a entrega deles pela plataforma Lala e Ogi.

Solução - Esquema



Solução - Classes

- **Item**
  1. Dimensões
  2. Peso
  3. Volume
  4. Nome

- **Veiculo**
  1. Dimensões
  2. Volume
  3. Peso máximo
  4. Plataforma
  5. Tipo

Solução - Classes

- **Entrega**
  1. Lista de Item
  2. Número de itens
  3. Peso Total
  4. Volume Total

- **Plataforma**
  1. Nome
  2. Lista de Veículo
  3. Número de veículos

Solução - **Veículo Ideal**

1. Instância objetos Item -> Instância objeto Entrega (converte.py)
2. Instância objeto Veiculo -> Instância objeto Plataforma (dados_plat.py)
3. Critério de Peso
4. Critério de Volume
5. Critério Disposição
6. Criterio Capacidade
7. Retorna resposta (converte.py)

Solução - Critérios de Eliminação

1. **Peso**: Elimina veículos cujo peso máximo é inferior ao peso total dos itens
2. **Volume**: Elimina veículos cujo volume é inferior ao volume total dos itens
3. **Disposição**: Elimina os veículos que não conseguem armazenar todos itens
4. **Capacidade**: Elimina os veículos de alta capacidade

Solução - **Critério de disposição e Kanpsack Problems**

- Descobrir o número máximo de itens que podem ser armazenados num container é um tipo de problema chamado Knapsack .
- Sendo esse problema um tipo NP-Hard, as soluções mais eficazes são conseguidas geralmente por algoritmos heurísticos
- Para resolver esse tipo de problema é usada a biblioteca py3dbp, que implementa uma heurística desenvolvida por Dube e Kanavathy (2006). Essa biblioteca preenche containers e fala que itens ficaram de fora

Solução - **API Veículo Ideal**

- Uma vez que a API está funcionando, ela recebe requests POST com body em formato JSON com dados escritos na forma genérica:

[{ "item": "café", "largura":35, "altura":20, "espessura": 30, "peso":1 }, { "item": "leite", "largura":35, "altura":20, "espessura": 30, "peso":1 }]

Solução - **API Veículo Ideal**

- A resposta dada pela API para o usuário também é em formato JSON, podendo ser, por exemplo:

{ "Lala": "Fiorino", "Ogi": "Não há veiculo adequado" }

Extra - **Autenticação**

- Ideias que poderiam ser implementadas
    1. Basic Auth: usuário e senha, facilmente implementado no Flask

Referência

Dube, Erick, Leon R. Kanavathy, and Phoenix Woodview. "Optimizing Three-Dimensional Bin Packing Through Simulation." *Sixth IASTED International Conference Modelling, Simulation, and Optimization*. 2006.