

Faculdade

XPe



RELATÓRIO

PROJETO
APLICADO

PÓS-GRADUAÇÃO

XP Educação
Relatório do Projeto Aplicado

Ferramenta de otimização de portfólio

Paulo Magno Oliveira Filho

Orientador(a): Silas Yunghwa Liu

2023



PAULO MAGNO OLIVEIRA FILHO

XP EDUCAÇÃO

RELATÓRIO DO PROJETO APLICADO

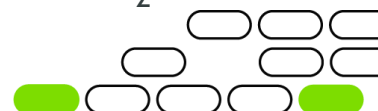
FERRAMENTA DE OTIMIZAÇÃO DE PORTFÓLIO

Relatório de Projeto Aplicado
desenvolvido para fins de conclusão do
curso de MBA em Ciência de Dados

Orientador (a): Silas Yunghwa Liu

SÃO PAULO

2023



Sumário

1. CANVAS do Projeto Aplicado	5
1.1 Desafio	6
1.1.1 Análise de Contexto	6
1.1.2 Personas	8
1.1.3 Benefícios e Justificativas	9
1.1.4 Hipóteses	11
1.2 Solução	12
1.2.1 Objetivo SMART	12
1.2.2 Premissas e Restrições	12
1.2.3 Backlog de Produto	14
2. Área de Experimentação	15
2.1 Sprint 1	15
2.1.1 Solução	15
• Evidência do planejamento:	15
• Evidência da execução de cada requisito:	16
• Evidência dos resultados:	21
2.1.2 Lições Aprendidas	24
2.2 Sprint 2	25
2.2.1 Solução	25
• Evidência do planejamento:	25
• Evidência da execução de cada requisito:	26
• Evidência dos resultados:	31
2.2.2 Lições Aprendidas	33
2.3 Sprint 3	34
2.3.1 Solução	34
• Evidência do planejamento:	34
• Evidência da execução de cada requisito:	35
• Evidência dos resultados:	38
2.3.2 Lições Aprendidas	40
3. Considerações Finais	41
3.1 Resultados	41
3.2 Contribuições	42





1. CANVAS do Projeto Aplicado

Figura conceitual, que representa todas as etapas do Projeto Aplicado definidas pelo curso de pós graduação em Ciência de dados.

Figura 1 - Canvas do Projeto



1.1 Desafio

1.1.1 Análise de Contexto

Segundo pesquisas da B3, bolsa de valores de São Paulo (<https://agenciabrasil.ebc.com.br/economia/noticia/2022-11/investimento-de-pessoa-fisica-em-renda-variavel-cresce-35>), e também da Associação Brasileira das Entidades dos Mercados Financeiro e de Capitais, a AMBIMA (https://www.ambima.com.br/pt_br/noticias/cresce-numero-de-investidores-brasileiros-em-2022-e-perspectiva-para-2023-e-de-novo-aumento.htm), vem crescendo a quantidade de brasileiros investindo de 2021 para 2022, e a perspectiva é continuar aumentando. Isto também mostrou que o brasileiro vem buscando diversificar seus investimentos, e, ainda, não só o número de pessoas físicas cresce, mas, o volume negociado.

Assim, obter informações e conhecimento para a tomada de decisão sobre como montar uma carteira diversificada se torna necessário para o investidor e analistas, os quais buscam orientações através de estudos, cálculos, métricas, mídias e notícias. Neste cenário ferramentas automatizadas e visuais são importantes para melhor compreensão, projeção dos investimentos, afim de tomar decisões melhores e mais embasadas.

A matriz CSD a seguir, mostra informações para um melhor conhecimento do contexto explorado buscando uma solução robusta para auxilio nos investimentos.

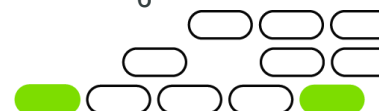


Figura 2 - Matriz CSD



Como complemento ao entendimento do contexto foi utilizado a ferramenta POEMS.

Figura 3 - POEMS



1.1.2 Personas

Visando uma análise profunda do problema e entender as reais necessidades e problemas buscamos definir Personas as quais correspondem ao perfil do usuário. As características e necessidades apresentadas foram embasadas em pesquisa e entrevista com possíveis usuários da solução.

Persona 1:

Mulher na faixa de 20 a 30 anos, com 5 anos de experiência no ramo bancário, especialista em investimentos com certificação CPA 20 (Certificação Profissional ANBIMA serie 20), tem como responsabilidade a orientação a clientes sobre informações e recomendações de oportunidades de investimento. Busca conhecimento contínuo, é pós graduando em investimento alta renda, se atualiza através de Moorning calls de corretoras e bancos de investimentos, palestras, e no cotidiano se atenta em realizar análises precisas e robustas para encontrar as melhores oportunidades de investimentos.

Os desafios que encontra são que, ainda há muitos clientes com receio do investimento diversificado, os valores mais expressivos se encontram em investimento na poupança, também conhecer o cliente para descrever seu perfil de investidor.

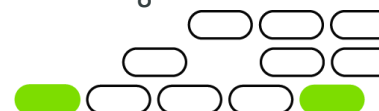
Uma das ferramentas mais utilizadas é o *Mais Retorno*, que possibilita comparação de ativos, simulação de rendimentos e outros serviços através de assinatura, mas que não possui funcionalidades de otimização.

As recomendações aos clientes são realizadas através da utilização destas ferramentas e, realizando cálculos, análises de índices e notícias. Alguns calculos realizados são: CAPM (Capital Asset Pricing Model), fronteira eficiente, SML (Security Market Line) e CML (Capital Market Line).

Busca soluções, ferramentas e técnicas que melhora a eficiência e tempo nas análises, permitindo visualizações, observar tendências, previsões de forma mais ágil, sem necessidade de recalcular e redigitar.

Persona 2:

Homem na faixa de 30 a 40 anos, administrador com experiencia em gestão e análise de frota de veículos, responsável por analisar custos e manutenções dos veículos da empresa. Busca a resolução de problemas realizando analises profundas das informações, estuda sobre finanças e investimento, e tem experiência e prática como investidor. Tem preferência em realizar investimentos através de estudos e análises próprias, se baseando também em informações de corretoras, mídia especializada, cartas de gestores, investidores experientes.



Assim, procura realizar análises mais visuais, e se basear em métricas, sem muitos cálculos mais elaborados, utilizando ferramentas prontas pagas no home broker da corretora.

Seus desafios se encontram em conhecimentos sem muita teoria, ferramentas pagas, considerado investidor de renda mais baixa.

A partir destas informações foi elaborado um mapa de empatia. O foco é identificar as necessidades, dores, problemas através do que sentem e pensam sobre os investimentos, procurando entender como agem e percebem diante do desafio de montar uma carteira de investimentos em renda variável.

Figura 4 - Mapa de empatia



1.1.3 Benefícios e Justificativas

Conforme o contexto observado, nota-se a necessidade de mais ferramentas para apoio a tomada de decisão sobre a formação de um portfólio, que contribuem para otimizar tempo e eficiência no processo, sem a realização repetida de cálculos.

Diante disto, a programação e a ciência de dados ajudar na extração de dados, automatização de tarefas e modelagem preditiva para encontrar soluções de portfólio baseado em dados e padrões, e em visualizações que auxiliam em uma absorção melhor das informações. E ainda, encapsular cálculos, códigos, e técnicas que não são tão

necessários se conhecer profundamente para realizar um investimento em renda variável, gráficos, tabelas e métrica podem resumir as principais informações. Abaixo, na Tabela 1, foi idealizado o mapeamento de ações do investidor utilizando a metodologia Business Design Blueprint.

Tabela 1 - Detalhamento do problema (Blueprint). Fonte: elaborador pelo autor

ITEM	DETALHAMENTO
Ações do cliente	Montar portfolio
Objetivos	Melhor arranjo do valor investido, maior retorno, menor risco
Atividades	Coleta dados dos ativos, visualiza gráficos, faz cálculos
Questões	Quais os melhores ativos para montar o portfólio? Quantos ativos? Qual o melhor peso para cada ativo? Quanto se pode ganhar e em quanto tempo?
Barreiras	Incertezas do mercado, cálculos incorretos, falta de dados, flutuações dos valores, acompanhamento periódicos dos ativos
Saída desejável	Otimização da carteira com melhores pesos para cada ativo de acordo com o objetivo (maior retorno, menor risco, outra métrica)
Funcionalidades	Modelos de otimização de portfólio, gráfico com projeções da carteira
Interação	Insere o código dos ativos e valor de investimento. Visualiza gráfico e tabelas
Mensagem	Percentual de cada ativo de cada tipo de otimização, rendimento final em um período de cada otimização, gráfico com projeções.
Onde ocorre	Local que possui um dispositivo com acesso a internet (computador, notebook, tablet, celular)
Tarefas aparentes	Gera percentuais, gráficos, tabelas, previsões
Tarefas escondidas	Cálculos automatizados em script, algoritmo de aprendizado de máquina, códigos de programação
Processos de suporte	Página interativa em Streamlit e API em Flask

Através destas informações foi idealizada uma proposta de solução criada na figura abaixo utilizando a ferramenta Canvas de Proposta de valor.

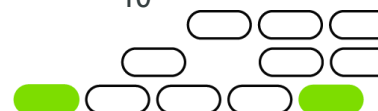
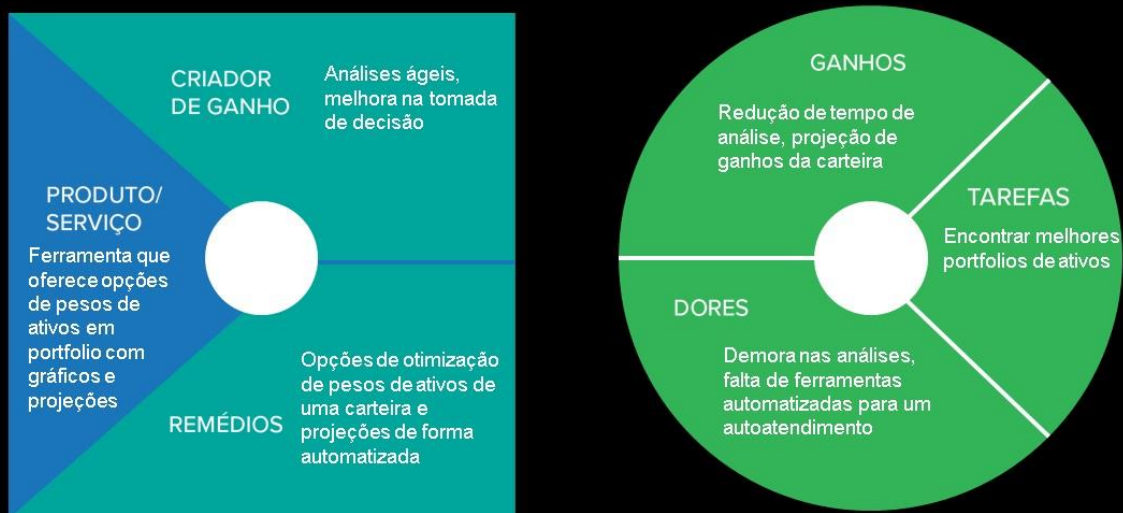


Figura 5 - Canvas de proposta de valor.

XPe EXPLICAÇÃO DE PROPOSIÇÃO DE VALOR



1.1.4 Hipóteses

Diante das pesquisas e análises realizadas pode-se notar algumas questões onde surge as ideias e prioridades para se chegar em uma solução.

Segue, na Tabela 2, um resumo destas observações e hipóteses encontradas.

Tabela 2 - Observações e hipóteses. Fonte: elaborado pelo autor

Observações	Hipóteses
Pessoas tem medo de investir em renda variável	Pouco conhecimento do mercado e pouco tempo para se aprofundar no estudo, falta de ferramenta auxiliadora
Analistas tem muitas opções de formação de carteira e muitos cálculos a realizar	Muitos ativos disponíveis, uso de ferramentas limitadas, cálculos muito demorados
Investidores não conhecem técnicas de diversificação de carteira	Dependência de terceiros/analistas, falta de ferramenta prática e intuitiva.

Com estas informações foi listado abaixo as ideias para solucionar o problema e com a ajuda da Matriz BASICO verificar sua relevância e necessidade de priorização.

Tabela 3 - Priorização das ideias - dimensões

Escala	B - Benefícios	A - Abrangência	S - Satisfação	I - Investimentos	C - Cliente	O - Operacionalidade
5	De vital importância	Total (de 70 a 100%)	Muito grande	Pouquíssimo investimento	Nenhum impacto	Muito fácil
4	Significativo	Muito grande (de 40 a 70%)	Grande	Algum investimento	Impacto pequeno	Fácil
3	Razoável	Razoável (de 20 a 40%)	Média	Médio investimento	Médio impacto	Média facilidade
2	Poucos benefícios	Pequena (de 5 a 20%)	Pequena	Alto investimento	Impacto grande	Difícil
1	Algum benefício	Muito pequena	Quase não é notada	Altíssimo investimento	Impacto muito grande no cliente	Muito difícil

Fonte: <https://engenhariaexercicios.com.br/gestao-de-qualidade/matriz-gut-basico-conceito-aplicacao-das-matrizes-priorizacao/>. Acesso em 05 de jun. de 2023.

Tabela 4 - Matriz BASICO. Fonte: elaborada pelo autor

Ideias	B	A	S	I	C	O	Total	Priorização
Disponibilizar mais de um tipo de otimização de carteira	5	5	5	5	1	2	23	1
Simular rendimentos das carteiras mediante um valor	5	4	5	5	2	2	23	1
Projetar investimento com gráfico	4	4	3	5	3	4	23	1
Disponibilizar gráficos com análises	4	3	3	5	3	3	21	2
Possibilitar baixar as imagens e tabelas	3	3	2	5	3	4	20	3

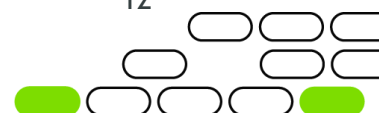
1.2 Solução

1.2.1 Objetivo SMART

O objetivo deste trabalho é desenvolver uma ferramenta que informa os pesos dos ativos através de métodos de otimização de portfólio e suas previsões de rendimento utilizando programação e aprendizado de máquina.

1.2.2 Premissas e Restrições

Premissas do projeto:



- Programação na linguagem Python.
- Otimização de portfólio utilizando a fronteira eficiente e Hierarchical Risk Parity (HRP).
- Fonte de dados da API Yahoo Finance íntegras.
- Métricas: Índice Sharpe, Risco, Retorno.
- Interface desenvolvida com Streamlit.

Restrições do projeto:

- Protótipo em 2 semanas.
- Ativos limitados à API utilizada.
- Serviço on-premise inicialmente

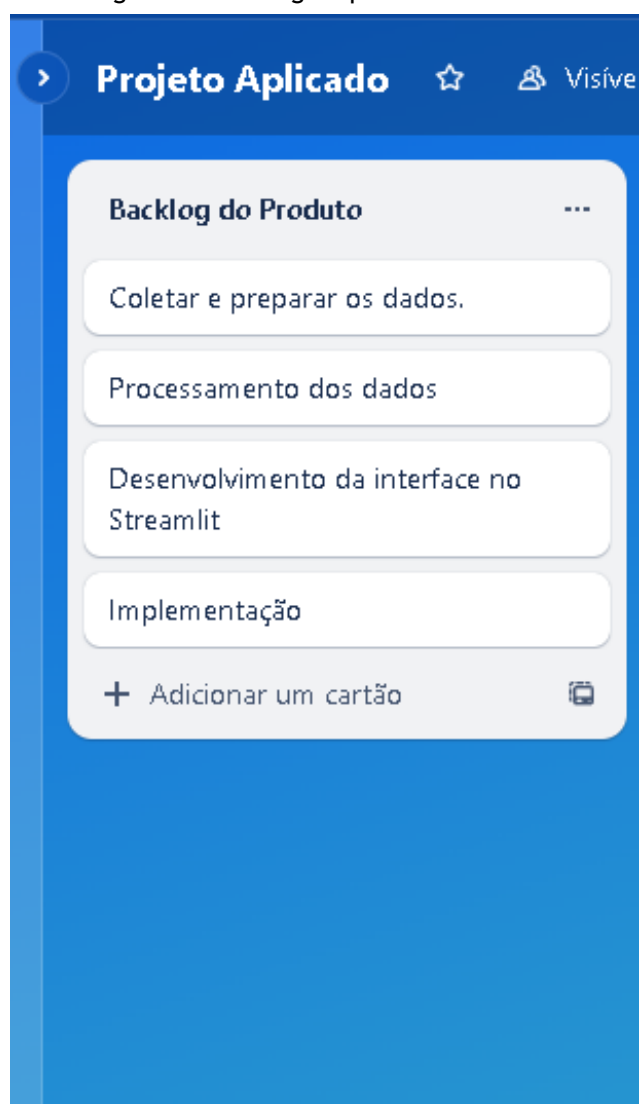
Para que o problema seja sanado de forma eficiente deve-se identificar e gerenciar os riscos e obstáculos para que se possa agir de forma proativa procurando mitiga-los. Na tabela 5 foi elaborada a matriz de riscos mapeando os riscos e as ações para contorná-los.

Riscos identificados	Impacto potencial	Ações Preventivas	Ações Corretivas
Erro de coleta de dados	Falha na ferramenta	Testes periódicos	Troca de API
Limitação de arquitetura	Dificuldade de processamento e manutenção	Padrão de qualidade, revisões códigos, aumento de processamento	Refatoração código, mudança para arquitetura escalável
Bugs na ferramenta	Não atinge a expectativa do usuário	Testes periódicos	Corrigir erros, melhorar qualidade



1.2.3 Backlog de Produto

Figura 6 - Backlog do produto.



2. Área de Experimentação

Esta seção apresenta as etapas de desenvolvimento do projeto, colocando evidências do planejamento estruturado no Backlog de Produto

2.1 Sprint 1

2.1.1 Solução

- Evidência do planejamento:

Foi definida na Sprint 1 a coleta e preparação dos dados e o processamento, segue o planejamento das atividades para cumprir os requisitos:

Figura 7 - Planejamento Sprint 1.



Figura 8 - Sprint 1: Coleta e preparação.

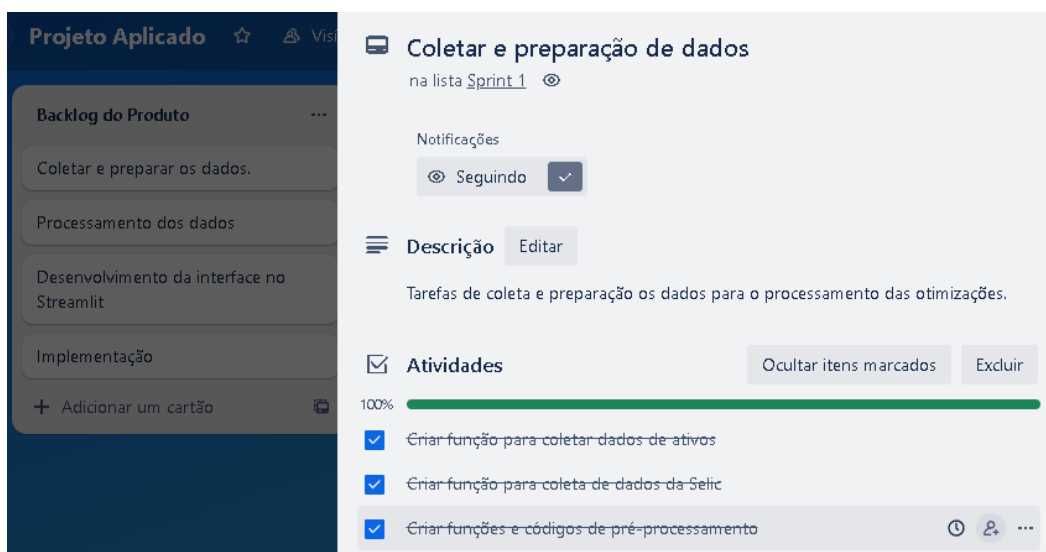
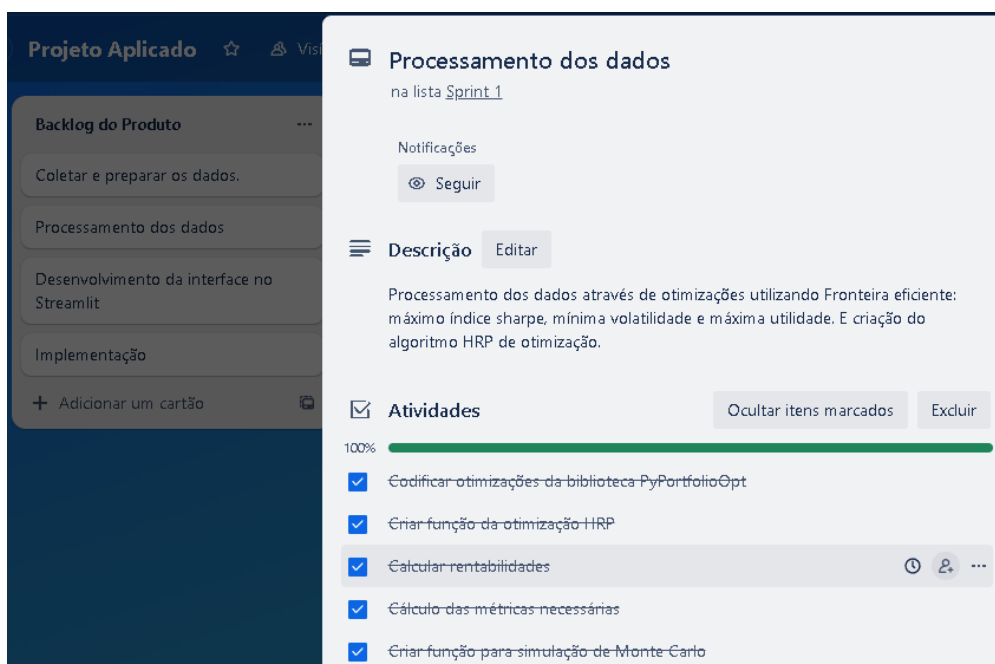


Figura 9 - Sprint 1: Processamento.



- **Evidência da execução de cada requisito:**

Para realização destas atividades foi utilizado a ferramenta Jupyter notebook criando arquivos .ipynb e py. Primeiramente, foi realizado testes nas APIs de coleta de ativos e da taxa Selic para verificar os seus retornos e quais tratamentos seriam necessários para preparar os dados para os cálculos.

ITEM1: coletar dados de ativos

Figura 10 - Função para coletar ativos

Jupyter asset_download.py 12/06/2023

```

1 def stocks_dataframe(start, end = 0, *stocks_id:list):
2     import datetime as dt
3     import pandas as pd
4     import yfinance as yf
5
6     if end == 0:
7         end = dt.date.today()
8
9
10    # convert string date in datetime
11    if type(start) != dt.date:
12        start = dt.datetime.strptime(start, '%d/%m/%Y')
13
14    if type(end) != dt.date:
15        end = dt.datetime.strptime(end, '%d/%m/%Y')
16
17    stock_data = pd.DataFrame()
18
19
20
21    # iterate stocks args
22    for i in stocks_id:
23
24        for asset in i:
25
26            try:
27                # get close data stocks in yahoo finance API
28                close = yf.download(asset, start=start, end=end)['Adj Close']
29
30                # add adj close value
31

```

ITEM 2: coletar dados da Selic

Figura 11 - Função de coletar dados da Selic

Jupyter SELIC_tax.py Último sábado às 19:44

```

6 def calculate_average_selic_annual(year1:int, year2 = 0):
7
8     if year2 == 0:
9         year2 = dt.date.today().year
10
11    if year1 > year2:
12
13        raise ValueError("Primeiro ano deve ser menor que o segundo ano.")
14
15    try:
16
17        # api extract selic
18        serie = 432
19        url = f'https://api.bcb.gov.br/dados/serie/bcdata.sgs.{serie}/dados?formato=json'
20        data = pd.read_json(url)
21
22        # convert to datetime
23        data['data'] = pd.to_datetime(data['data'], dayfirst = True)
24
25        # year column, grouping by year and aggregate average
26        data['data'] = data['data'].dt.year
27        data = data.rename(columns = {'valor': 'selic'})
28        data = data.loc[(data['data'] >= year1) & (data['data'] <= year2)]
29
30    # another API if error
31    except ValueError:
32
33
34        data = sgs.get(('selic', 432), start = dt.datetime.strptime('01/01/'+str(year1), '%d/%m/%Y'), end = dt.date.today())
35        data = data.rename_axis('data').reset_index()
36        data['data'] = data['data'].dt.year

```

ITEM 3: Funções e códigos pré-processamento

Figura 12 - Tratamento pré-processamento

```
def replaceZeroes(data):
    min_nonzero = np.min(data[np.nonzero(data)])
    data[data == 0] = min_nonzero
    return data

data = pd.DataFrame(columns = columns, data = data, index = index)
returns = data.pct_change().apply(lambda x: np.log1p(x)).fillna(0, axis = 1)
returns.replace([np.inf, -np.inf, np.nan], 0, inplace=True)

# annual returns
mean = returns.mean() * 252

# deviation
sigma = returns.std(axis = 0) * np.sqrt(252)

# covariance matrix
cov_matrix = returns.cov() * 252
cov_matrix
```

ITEM 4: Codificar otimizações utilizando a biblioteca PyPortfolioOpt

Figura 13 - biblioteca PyPortfolioOpt

Otimização max sharpe ratio

```
expected_r = expected_returns.capm_return(data)
estimative = risk_models.CovarianceShrinkage(data).ledoit_wolf()
ef1 = EfficientFrontier(expected_r, estimative)
ef1.max_sharpe()
ret_tangent, std_tangent, _ = ef1.portfolio_performance(verbose = True)
```

Expected annual return: 11.2%
Annual volatility: 21.4%
Sharpe Ratio: 0.43

Otimização CLA mínima volatilidade

```
ef2 = CLA(expected_r, estimative)
ef2.min_volatility()
ef2.portfolio_performance(verbose = True, risk_free_rate=tax_w_risk)
```

Expected annual return: 8.2%
Annual volatility: 17.3%
Sharpe Ratio: -0.02
(0.08162723074021115, 0.17278344683015753, -0.02183524712004041)

Otimização máxima utilidade

```
ef3 = EfficientFrontier(expected, estimative)
ef3.max_quadratic_utility()
ef3.portfolio_performance(verbose = True, risk_free_rate=tax_w_risk)
```

Expected annual return: 43.0%
Annual volatility: 41.2%
Sharpe Ratio: 0.84
(0.42973286380445724, 0.41249717104069955, 0.8374187462497159)



ITEM 5: otimização HRP

Figura 14 - algoritmo HRP

```
: class HRP():
    def __init__(self, returns):
        self.returns = returns

    def matrix_seriation(self):

        self.matrix_cov = self.returns.cov()

        dendrogram = sns.clustermap(self.matrix_cov, method='ward', metric='euclidean')

        seriation = dendrogram.dendrogram_col.reordered_ind
        self.columns_seriation = returns.columns[seriation]

        return (self.matrix_cov, self.columns_seriation)

    def calcula_w_hrp(self):

        self.matrix_cov, self.columns_seriation = matrix_seriation(self.returns)
        # Inicialização de pesos
        self.pesos = pd.Series(1, index=self.colunas_seriation)
        paridades = [self.colunas_seriation]

        while len(paridades) > 0:
            # Instanciação de clusters
            paridades = [cluster[inicio:fim]
                          for cluster in paridades
                          for inicio, fim in ((0, len(cluster) // 2), (len(cluster) // 2, len(cluster)))
                          if len(cluster) > 1]

            # Iteração entre paridades
            for subcluster in range(0, len(paridades), 2):

                cluster_esquerdo = paridades[subcluster]
                cluster_direito = paridades[subcluster + 1]

                matriz_cov_esquerda = matriz_cov[cluster_esquerdo].loc[cluster_esquerdo]
                inversa_diagonal = 1 / np.diag(matriz_cov_esquerda.values)
```

ITEM 6: Calcular rentabilidades

Figura 15 - rentabilidades de cada otimização

Adicionando ao dataframe rendimento dos portfolios diários

```
returns['sharpe_otm'] = 0
returns['cla_otm'] = 0
returns['max_sqrt_otm'] = 0
returns['hrp_otm'] = 0

for i in range(len(data)):
    sum_all_sharpe = 0
    sum_all_risk = 0
    sum_all_return = 0
    sum_all_hrp = 0

    for x in range(len(values_sharpe)):

        # sum returns sharpe weights
        sharpe = (1+(returns.iloc[i,x]))* values_sharpe[x]
        sum_all_sharpe = sum_all_sharpe + sharpe
        values_sharpe[x] = sharpe

        # sum returns risk weights
        risk = (1+(returns.iloc[i,x]))* values_risk[x]
        sum_all_risk = sum_all_risk + risk
        values_risk[x] = risk

        # sum returns return weights
        return_ = (1+(returns.iloc[i,x]))* values_return[x]
        sum_all_return = sum_all_return + return_
        values_return[x] = return_
```

ITEM 7: Cálculo das métricas necessárias

Figura 16 - Cálculo dos betas de cada ativo

Beta dos ativos

```
dict_betas = {}
|
# benchmark returns
df_ibovr = df_ibov.pct_change()
df_ibovr.dropna(inplace = True)

for i in data.columns:

    # assets returns
    asset_r = data[i].pct_change()
    asset_r.dropna(inplace = True)

    # treatment for NaN and inf values
    asset_r.replace([np.inf, -np.inf, np.nan], 0, inplace=True)

    # linear regression to calculate beta
    model = LinearRegression()
    reg = model.fit(df_ibovr.values.reshape(-1, 1), asset_r.values.reshape(-1, 1))

    # beta
    beta_asset = reg.coef_[0][0]

    # add beta to dict
    dict_betas[i] = round(beta_asset, 2)

dict_betas
```

Figura 17 - função para calcular CAPM

CAPM

```
# capm calculates function for apply in every assets
def capm_calc(columns: list, betas: dict, weights: dict, risk_market, tax_free_risk):

    sum_betas = 0

    for i in columns:

        sum_betas += (betas[i] * weights[i])

    capm = tax_free_risk + sum_betas * (risk_market - tax_free_risk)
    return capm
```

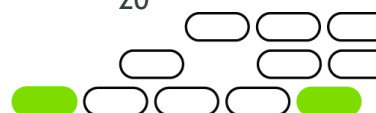


Figura 18 - função para plotar o gráfico da fronteira eficiente

```
def ploty_efficient_frontier(std, ret, expected_returns, cov_matrix, n_assets, n_samples):

    plt.scatter(std, ret, marker="*", s=100, c="r", label="Portfolio")

    n_samples = n_samples
    w = np.random.dirichlet(np.ones(n_assets), n_samples)
    rets_s = w.dot(expected_returns)
    stds_s = np.sqrt(np.diag(w @ cov_matrix @ w.T))
    sharpes = rets_s / stds_s
    plt.scatter(stds_s, rets_s, marker=".", c=sharpes, cmap="viridis_r")
    plt.colorbar(label = 'Sharpe ratio')
    plt.title("Fronteira Eficiente")
    plt.legend()
    plt.tight_layout();
    plt.clf()
    return rets_s, stds_s, sharpes
```

ITEM 8: simulação de Monte Carlo

Figura 19 - função para o algoritmo de Monte Carlo

```
6 def monte_carlo_projection(data_returns: pd.DataFrame, amount):
7
8     # returns percentage
9     returns_pct = np.log1p(data_returns[['sharpe_otm', 'cla_otm', 'max_sqrt_otm', 'hrp_otm']].pct_change())
10    returns_pct.fillna(0, inplace=True)
11
12    # calculates statistics
13    mean_mc = returns_pct.mean()
14    var_mc = returns_pct.var()
15    drift = mean_mc - (0.5 * var_mc)
16    std_mc = returns_pct.std()
17
18
19    # simulation params
20    years = 3
21    days = 252 * years
22    simulation = 1000
23
24    # random volatility by multidimensional normal distribution
25    np.random.seed(0)
26    Z = stats.norm.ppf(np.random.rand(days, simulation))
27    daily_sharpe = np.exp(drift[0] + std_mc[0] * Z)
28
29    np.random.seed(1)
30    Z = stats.norm.ppf(np.random.rand(days, simulation))
31    daily_risk = np.exp(drift[1] + std_mc[1] * Z)
32
33    np.random.seed(2)
34    Z = stats.norm.ppf(np.random.rand(days, simulation))
35    daily_mqu = np.exp(drift[2] + std_mc[2] * Z)
36
37    np.random.seed(3)
38    Z = stats.norm.ppf(np.random.rand(days, simulation))
39    daily_hrp = np.exp(drift[3] + std_mc[3] * Z)
40
41    # the last row every column to predict
42    pred_sharpe = np.zeros_like(daily_sharpe)
43    pred_sharpe[0] = amount #data_returns['sharpe_otm'][-1]
44
45    pred_risk = np.zeros_like(daily_risk)
46    pred_risk[0] = amount #data_returns['cla_otm'][-1]
```

- Evidência dos resultados:

Nesta Sprint foram criadas funções para coletar e tratar dados de ativos e da taxa Selic. Após foram desenvolvidas classes e funções para realizar os cálculos e otimizações de carteira de ativos, retornando pesos, valores, históricos e métricas para serem exibidos posteriormente.

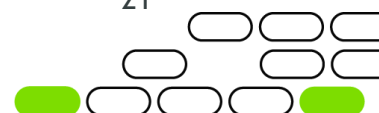


Figura 20 - Exemplo de Dataframe com rendimentos dos ativos e de cada otimização

```
In [550]: returns.head()
```

```
Out[550]:
```

	ABEV3.SA	PETR3.SA	MGLU3.SA	ITUB3.SA	ALPA3.SA	VIVT3.SA	WEGE3.SA	VALE3.SA	BBAS3.SA	sharpe otm	cla otm	max_sqrt otm	hrp otm
Date													
2020-06-16	-0.012739	0.040447	0.002275	0.020791	0.000874	0.017612	0.011008	0.027640	0.012919	1013.404940	1014.138474	1015.089227	1013.62544
2020-06-17	0.037740	0.002250	0.017870	0.009013	0.060162	0.005466	0.030799	0.014518	0.029414	1036.616152	1032.101252	1038.106638	1033.69027
2020-06-18	-0.006557	0.015608	0.036676	0.025369	0.001233	0.019197	-0.000650	-0.000534	-0.002903	1046.605858	1041.713589	1052.760308	1043.86324
2020-06-19	0.011628	-0.014260	0.024098	0.048127	0.018312	0.009069	0.001733	-0.017964	-0.013463	1054.507163	1049.366447	1060.817455	1054.39162
2020-06-22	0.019321	-0.018574	-0.007027	-0.041786	0.007232	-0.023031	-0.013288	-0.003086	-0.020840	1042.503341	1036.207872	1046.523173	1038.36496

Figura 21 - plot dos histórico de rendimentos de cada portfólio e o índice Ibov

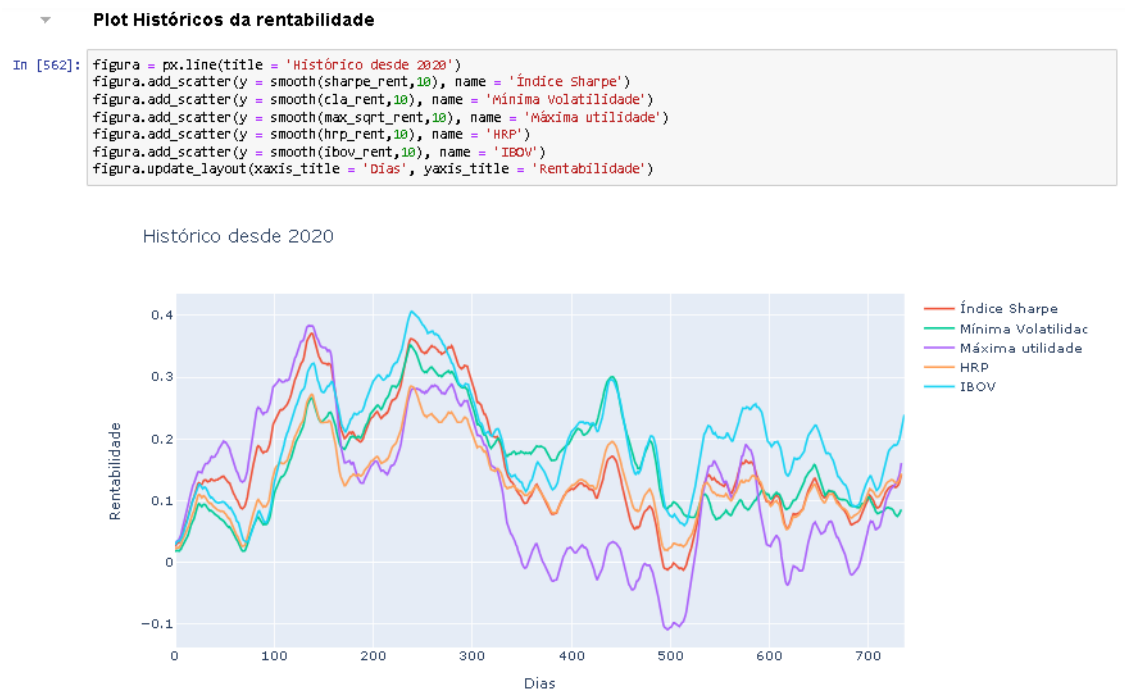


Figura 22 - plot das projeções de Monte Carlo, exemplo do portfólio pelo índice sharpe

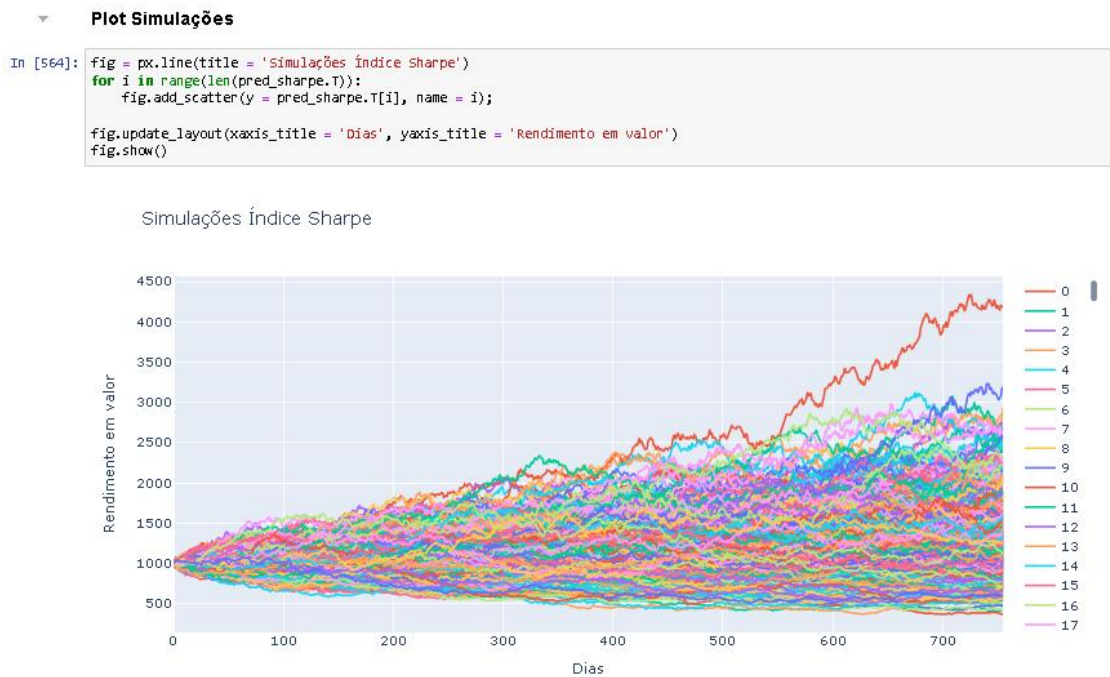
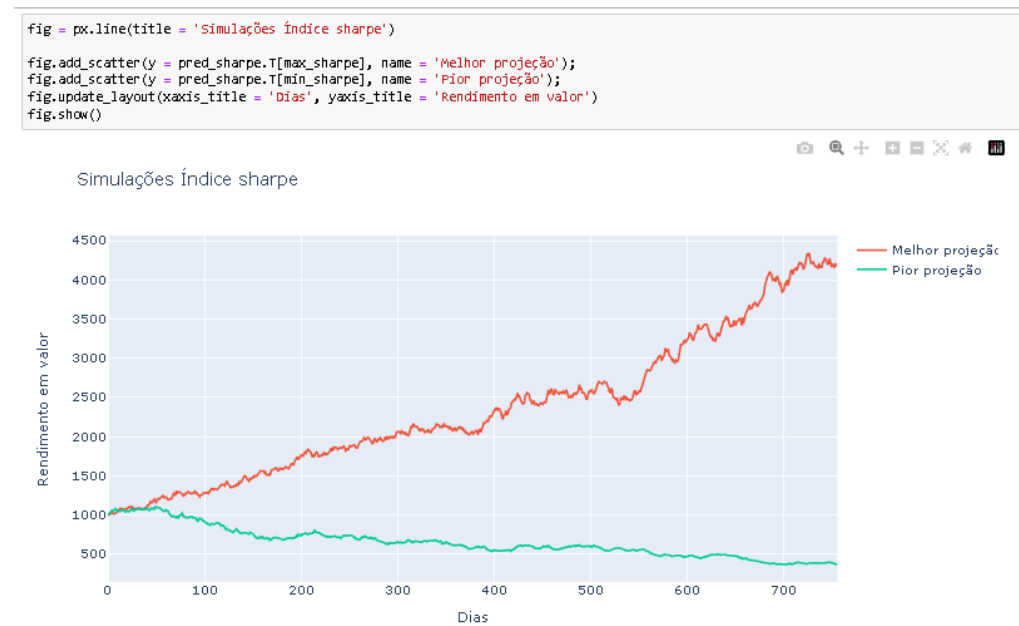


Figura 23 - plot da melhor e pior simulação pelo índice sharpe



2.1.2 Lições Aprendidas

Após o desenvolvimento da Sprint 1, algumas novas questões e desafios foram encontrados.

A API da Selic algumas vezes não retorna valores; talvez, algumas barreiras quando se acessa por um mesmo IP; necessário um tratamento de erros e colocar outra API que é exclusiva para o python (bcb), mas, não sendo uma API oficial do governo. Em testes, alguns dados são diferentes em algumas casas decimais, mas, não compromete o resultado final.

A biblioteca PyPortfolioOpt já fornece algumas métricas através de métodos e atributos, já para o algoritmo HRP deve-se calcular individualmente.

Para uma melhor exibição dos gráficos dos portfólios, necessário suavizar os dados para uma visualização mais limpa das linhas temporais conforme o período de observação se torna maior.

No final da sprint, em alguns testes, foi verificado que alguns ativos estrangeiros apresentam alguns raros valores de fechamento zerado o que gerou erro para cálculos de retornos e riscos na biblioteca pypfopt, a qual não possui tratamento em dados.



2.2 Sprint 2

2.2.1 Solução

- **Evidência do planejamento:**

Foi definida na Sprint 2 a inserção de uma otimização simples para se utilizar como baseline, e também a criação da interface utilizando o framework Streamlit no editor Visual Studio Code, segue o planejamento das atividades para cumprir os requisitos:

Figura 24 - Planejamento Sprint 2

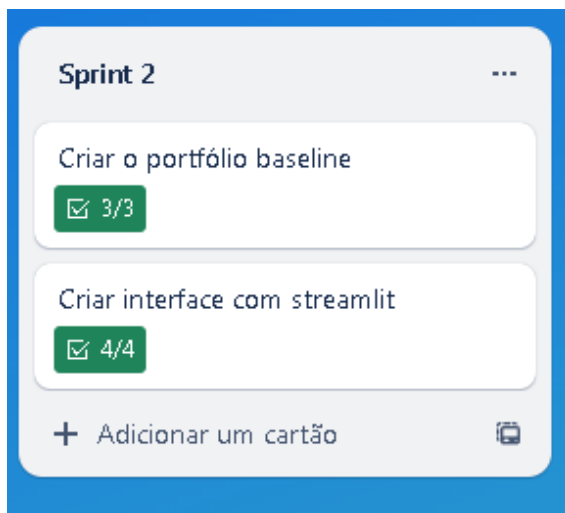


Figura 25 - Sprint 2: Criar o portfólio baseline

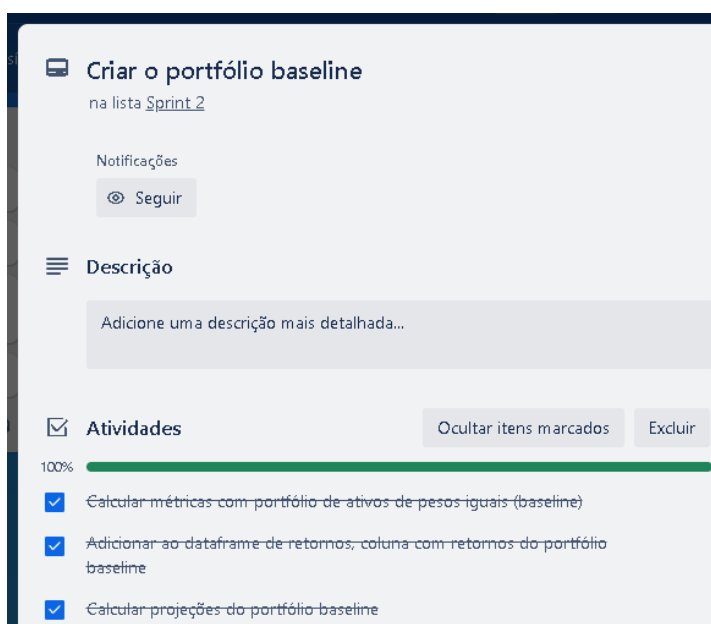
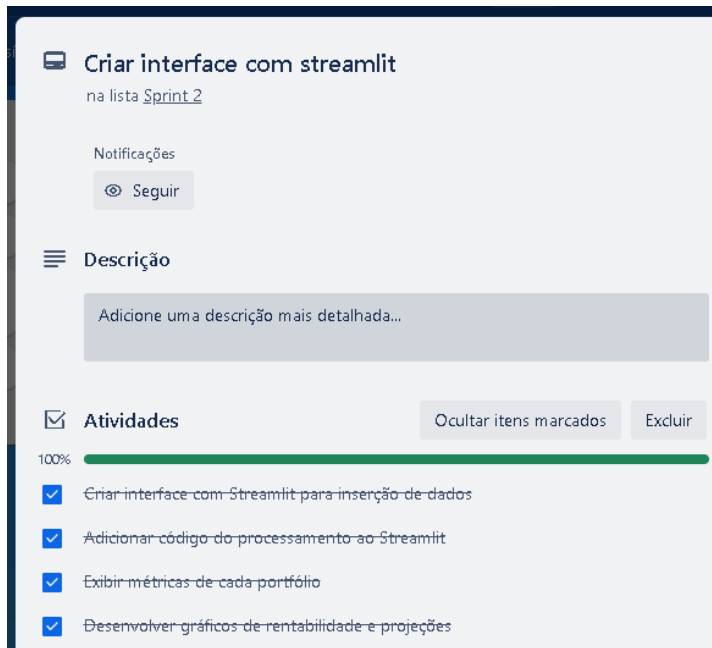


Figura 26 – Sprint 2: Criar interface com Streamlit



- Evidência da execução de cada requisito:

Para realização destas atividades foi utilizado o Jupyter notebook e o framework Streamlit do Python na ferramenta Visual Studio Code e realizando testes com scripts no Windows Power Shell. Primeiramente, foi desenvolvido no notebook uma otimização simples baseada em portfólio de pesos iguais para servir de baseline, referência e se incluir no processamento. Após, se idealizou um visual básico para inserção dos dados pelo usuário, a inclusão do código do processamento já desenvolvido e adaptações deste ao Streamlit.

ITEM 1: desenvolver métricas do portfólio baseline

Figura 27 – Cálculo das métricas

Portfólio sem otimização, pesos iguais

```
assets_len = len(data.columns)

# weights for baseline portfolio
weights_equals = np.full(assets_len, (1 / assets_len))

# risk for baseline portfolio
sigma_equals = cov_portfolio(weights_equals, cov_matrix) * 100

# returns for baseline portfolio
ret_equals = return_portfolio(weights_equals, mean).sum() * 100

# sharpe ratio for baseline portfolio
sharp_equals = (ret_equals - tax_w_risk) / sigma_equals

values_equals = np.dot(weights_equals, 1000)
```

ITEM 2: adicionar retornos do portfólio baseline ao dataframe

Figura 28 - cálculo dos rendimentos do portfólio

```
# sum returns hrp weights
hrp = (1+(returns.iloc[i,x]))* values_hrp[x]
sum_all_hrp = sum_all_hrp + hrp
values_hrp[x] = hrp

# sum returns equals weights
equals = (1+(returns.iloc[i,x]))* values_equals[x]
sum_all_equals = sum_all_equals + equals
values_equals[x] = equals

returns['sharpe otm'].iloc[i] = sum_all_sharpe
returns['cla otm'].iloc[i] = sum_all_risk
returns['max_sqrt otm'].iloc[i] = sum_all_return
returns['hrp otm'].iloc[i] = sum_all_hrp
returns['equals_portf'].iloc[i] = sum_all_equals
```

ITEM 3: adicionar o portfólio baseline a função da projeção de Monte Carlo

Figura 29 - incremento na função

```
# the last row every column to predict
pred_sharpe = np.zeros_like(daily_sharpe)
pred_sharpe[0] = amount

pred_risk = np.zeros_like(daily_risk)
pred_risk[0] = amount

pred_mqu = np.zeros_like(daily_mqu)
pred_mqu[0] = amount

pred_hrp = np.zeros_like(daily_hrp)
pred_hrp[0] = amount

pred_equals = np.zeros_like(daily_equals)
pred_equals[0] = amount
|
# calculating projections every day
for day in range(1, days):

    pred_sharpe[day] = pred_sharpe[day - 1] * daily_sharpe[day]
    pred_risk[day] = pred_risk[day - 1] * daily_risk[day]
    pred_mqu[day] = pred_mqu[day - 1] * daily_mqu[day]
    pred_hrp[day] = pred_hrp[day - 1] * daily_hrp[day]
    pred_equals[day] = pred_equals[day - 1] * daily_equals[day]
```



ITEM 4: Criar interface no Streamlit para inserção de dados

Figura 30 - código do streamlit

```
Otimização_Carteira.py
pages > Otimização_Carteira.py > ...
37 # title page
38 st.title(':white[Otimização da carteira]')
39
40 # assets typing area
41 keywords = st_tags(
42     label=' Digite os código dos ativos*',
43     text='Pressione enter para adicionar mais',
44     maxtags = 15,
45     key='1')
46
47 # website for asset code information
48 link = 'https://finance.yahoo.com/'
49 st.markdown(f'<i>* Código dos ativos de acordo com o site: {link}</i>', unsafe_allow_html=True)
50
51
52 # min and max date from actual day
53 day = 365 * 3
54 min_data = dt.date.today() - dt.timedelta(days = day)
55 max_data = dt.date.today()
56
57 #st.write(min_data)
58 #st.write(max_data)
59
60 # schedule exhibition and choice
61 start_date = st.date_input('Data inicial:', value = min_data, max_value = max_data )
62
63 #start_date = dt.datetime.strptime(str(start_date),"%d/%m/%Y")
64
65 # initial amount input
66 initial_inv = st.number_input("Digite o valor")
```

ITEM 5: Adicionar código do processamento ao Streamlit

Figura 31 - código do streamlit, botão de processamento

```
# extract values assets function
data = stocks_dataframe(start_date,0, keywords)

returns = data.pct_change().apply(lambda x: np.log1p(x)).fillna(0, axis = 1)
returns.replace([np.inf, -np.inf, np.nan],0 , inplace=True)

mean = returns.mean() *252

# deviation
sigma = returns.std(axis = 0) * np.sqrt(252)

#covariance
cov_matrix = returns.cov() * 252

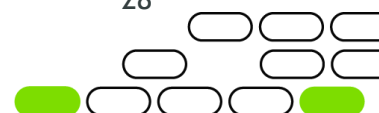
# calculates annual tax free risk
tax_w_risk = calculate_average_selic_annual(start_date.year)
expected_r = expected_returns.capm_return(data, risk_free_rate = tax_w_risk, log_returns = True)

estimative = risk_models.sample_cov(data, frequency = 252)
S = risk_models.CovarianceShrinkage(data).ledoit_wolf()

# pypfot is not adjusted for inf, -inf, and nan values in returns and risk calculation
# we use an conditional treatment to calculate log returns and covariance matrix and replace them in the
if np.isnan(expected_r.values).any() or ~np.isfinite(expected_r.values).any():

    expected_r = mean

    S = cov_matrix
```



ITEM 6: Exibir métricas de cada portfólio

Figura 32 - exibição das métricas com tabelas no Streamlit

```
# beta of the portfolio min vol
sum_betas = 0
for i in data.columns:
    sum_betas += (dict_betas[i] * ef3_weights2.loc[ef3_weights2['Ativos'] == i, "Pesos"].values[0])

capm_value = capm_calc(list(data.columns), dict_betas, ef3_weights, ibov_mean, tax_w_risk)

# add beta portfolio to metrics min vol dataframe
max_sqrt_df.loc['Beta do portfólio'] = f'{round(sum_betas, 2)}'

# add capm value to metrics dataframe
max_sqrt_df.loc['CAPM'] = f'{round(capm_value * 100, 2)}%'

# show max quadratic utility dataframes
col1.write('Métricas:')
col1.data_editor(max_sqrt_df)
col2.write('Pesos de cada ativo:')
col2.data_editor(ef3_weights2, hide_index = True)

# weights from every optimization in dictionary
w_sharpe = dict(ef1.clean_weights())
w_min_vol = dict(ef2.clean_weights())
w_max_q = dict(ef3.clean_weights())
```

ITEM 7: Desenvolver gráfico da fronteira eficiente

Figura 33 - código do gráfico usando a biblioteca plotly

```
# efficient frontier graphic
#
fig = px.scatter(df_aux, x="Risco", y="Retorno", color="Índice Sharpe",
                title="Fronteira Eficiente")

# max sharpe plot
fig.add_scatter(x = np.array(std_tangent), y = np.array(ret_tangent),
                name = 'Maior Sharpe', mode = 'markers', marker_symbol = 'star',
                marker = dict(size=15, color = 'Red', line=dict(width = 2,color='Black')))

# min vol plot
fig.add_scatter(x = np.array(std_tangent2), y = np.array(ret_tangent2),
                name = 'Menor Volatilidade', mode = 'markers',
                marker_symbol = 'x', marker = dict(size=15, color = 'Yellow',
                line=dict(width = 2,color='Black')))

# max quadratic utility plot
fig.add_scatter(x = np.array(std_tangent3), y = np.array(ret_tangent3),
                name = 'Máximo Retorno', mode = 'markers', marker_symbol = 'diamond',
                marker = dict(size=15, color = 'Green', line=dict(width = 2,color='Black')))

# equals weights portfolio plot
fig.add_scatter(x = np.array(sigma_equals/100), y = np.array(ret_equals/100),
                name = 'Pesos Iguais', mode = 'markers', marker_symbol = 'star-diamond',
                marker = dict(size=15, color = 'Orange', line=dict(width = 2,color='Black')))

# hrp plot
fig.add_scatter(x = np.array(sigma_hrp/100), y = np.array(ret_hrp/100),
                name = 'HRP', mode = 'markers', marker_symbol = 'circle',
                marker = dict(size=15, color = 'Purple', line=dict(width = 2,color='Black')))
```



ITEM 8: Desenvolver gráfico da rentabilidade

Figura 34 - gráfico das rentabilidade histórica de cada portfólio

```
st.plotly_chart(fig)

# portfolios historics
sharpe_rent = returns['sharpe otp'] / returns['sharpe otp'][0] -1
cla_rent = returns['cla otp'] / returns['cla otp'][0] -1
max_sqrt_rent = returns['max_sqrt otp'] / returns['max_sqrt otp'][0] -1
hrp_rent = returns['hrp otp'] / returns['hrp otp'][0] -1
equals_rent = returns['equals_portf'] / returns['equals_portf'][0] -1
ibov_rent = df_ibov / df_ibov[0] -1

figura = px.line(title = f'Retorno dos portfólios desde {start_date.year} X IBOV')
figura.add_scatter(y = smooth(sharpe_rent,10), name = 'Índice Sharpe')
figura.add_scatter(y = smooth(cla_rent,10), name = 'Mínima Volatilidade')
figura.add_scatter(y = smooth(max_sqrt_rent,10), name = 'Máximo Retorno')
figura.add_scatter(y = smooth(hrp_rent,10), name = 'HRP')
figura.add_scatter(y = smooth(equals_rent,10), name = 'Pesos Iguais')
figura.add_scatter(y = smooth(ibov_rent,10), name = 'IBOV')
figura.update_layout(width = 900, xaxis_title = 'Dias', yaxis_title = 'Rentabilidade')
st.plotly_chart(figura, width = 900)

df_portfolios = pd.DataFrame((returns.iloc[-1,-5:] / initial_inv -1) *100 )
df_portfolios.columns = ['Rentabilidade no Período']
df_portfolios.index = ['Índice Sharpe','Mínima Volatilidade', 'Máximo Retorno', 'HRP', 'Pesos Iguais']
df_portfolios['Rentabilidade no Período'] = list(map(lambda x: f'{round(x,2)}%',df_portfolios.values.reshape(-1)))

st.write(df_portfolios.T )
```

ITEM 9: Desenvolver gráficos das projeções

Figura 35 - gráfico com projeção de um portfólio

```
# Min Vol projection

fig = px.line(title = 'Portfólio Min Vol')

# calculating VaR min vol portfolio
last_risk = pred_risk[-1]
Var = np.percentile(last_risk, alpha)
# average values
avg_risk_monte_carlo = pred_risk.mean(axis = 1)

# plots plotly express
# min risk
fig.add_scatter(x = list(range(0,252*years_proj)),y = pred_risk.T[min_risk], name = 'Minimo',
| marker = dict(color = '#a020f0'))
# max risk
fig.add_trace(go.Scatter(x = list(range(0,252*years_proj)), y= pred_risk.T[max_risk],
| fill = 'tonexty', name = 'Intervalo', marker = dict(color = '#a020f0'), opacity = 0.1))
# average
fig.add_scatter(y = avg_risk_monte_carlo[1:], name = 'Média', marker = dict(color = 'green'))
# Var metric
fig.add_hline(y = Var, line_dash = 'dot', annotation_text = f'<b>VaR: {round(Var,2)}</b>', annotation_font_color
| annotation_position = 'bottom left', line_color = 'red')
# initial value investment
fig.add_hline(y = initial_inv, line_dash = 'dash', annotation_text = f'<b>{initial_inv}</b>',
| annotation_font_color = 'black', annotation_position = 'top right', line_color = 'black')
```

- Evidência dos resultados:

Nesta Sprint foi criada um portfólio para servir de referência e a interface da ferramenta para inserção dos dados pelo usuário, processamento e visualização das métricas e gráficos de cada otimização.

Figura 36 - parte superior da tela com inserção dos dados

Otimização da carteira

Digite os código dos ativos*:

vale3.sa petr4.sa itub4.sa aapl
Pressione enter para adicionar mais

*Código dos ativos de acordo com o site: <https://finance.yahoo.com/>

Data inicial:

2020/06/29

Digite o valor

15000,00

Mostrar otimizações

Figura 37 - tabelas com informações da otimização por índice sharpe

Otimização por Índice Sharpe

Métricas:

	Valor
Índice Sharpe	1.21
Volatilidade anual	26.55%
Retorno esperado anual	34.04%
Beta do portfólio	0.88
CAPM	1.02%

Pesos de cada ativo:

Ativos	Pesos	Valor à aplicar	Beta
vale3.sa	0.0387	581.1	0.84
petr4.sa	0.4857	7,286.1	1.29
itub4.sa	0	0	1.05
aapl	0.4755	7,132.8	0.47

Figura 38 - gráfico da Fronteira Eficiente

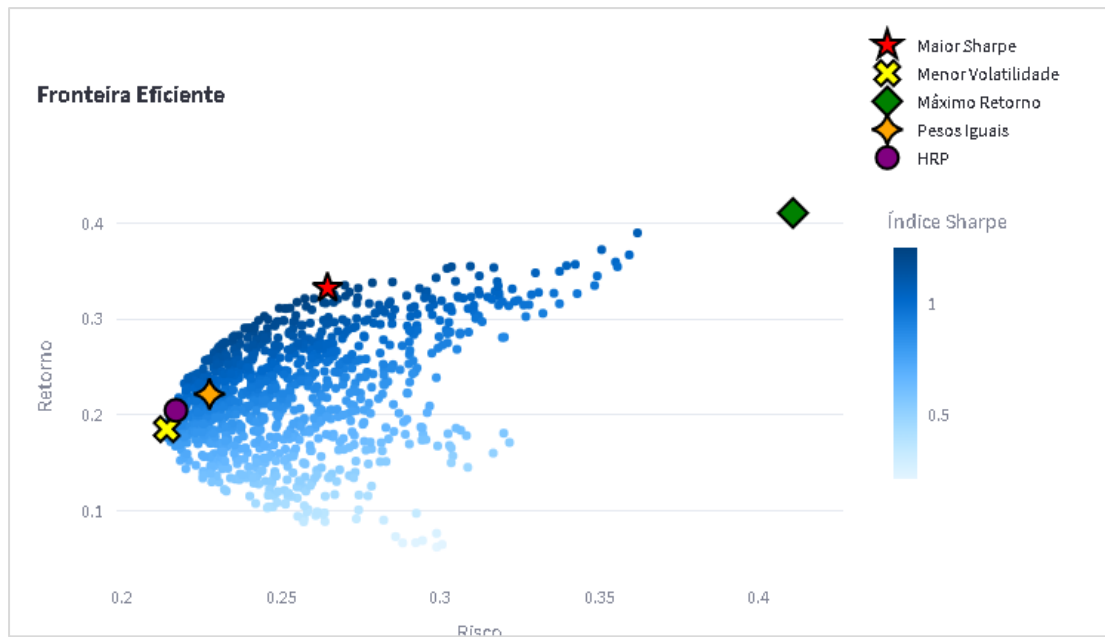


Figura 39 - informações dos rendimentos de cada otimização

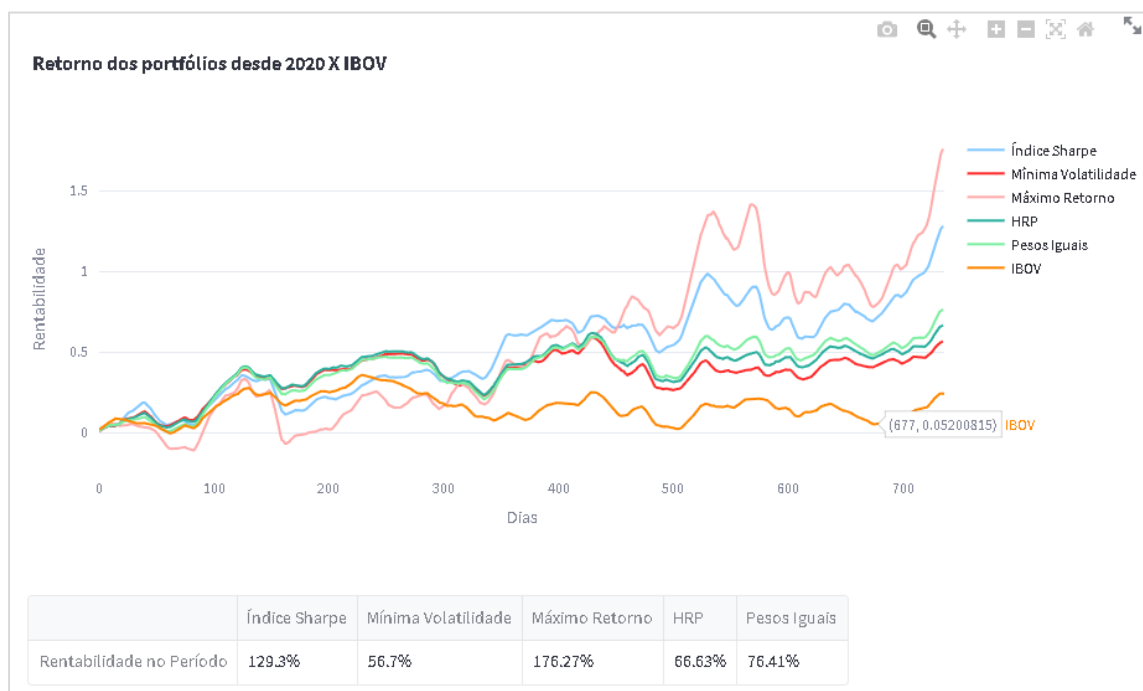
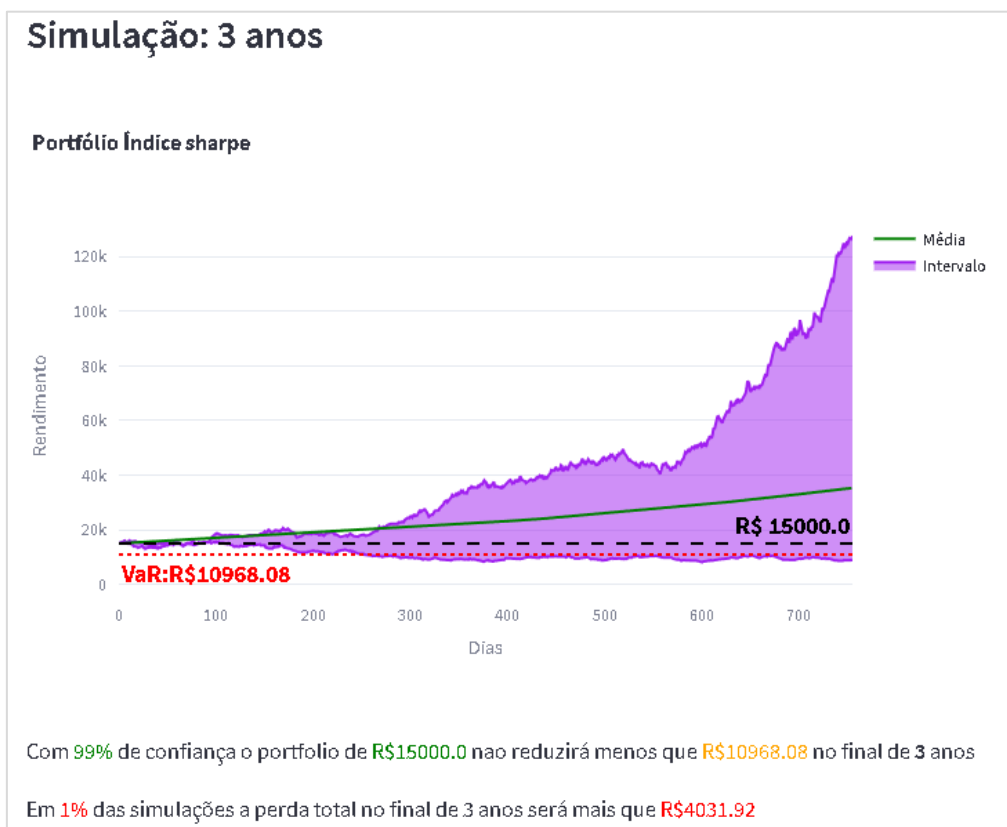


Figura 40 - gráfico com informações da simulação da otimização por índice sharpe



2.2.2 Lições Aprendidas

Inicialmente, foi desenvolvido os gráficos utilizando a biblioteca matplotlib, mas, a visualização no Streamlit não ficou com boa qualidade e não era chamativa, a biblioteca plotly, além de ser interativa, apresentou melhores tons e se tornou melhor para se adaptar com o Streamlit com relação a redimensionamento, tratamento de cores e tipo de gráfico.

Na visualização por colunas no Streamlit, não foi encontrado uma forma de se colocar uma faixa visual separando-as, foi necessário alterar a visualização de cada portfólios para linhas. No final, esta visualização se adaptou ainda melhor a tela para se compreender bem as informações de cada portfólio.

2.3 Sprint 3

2.3.1 Solução

- **Evidência do planejamento:**

Foi definida na Sprint 3 a criação da página inicial com texto explicando as técnicas e fórmulas utilizadas na ferramenta e a implementação, tornando-a disponível na web:

Figura 41 - Planejamento Sprint 3

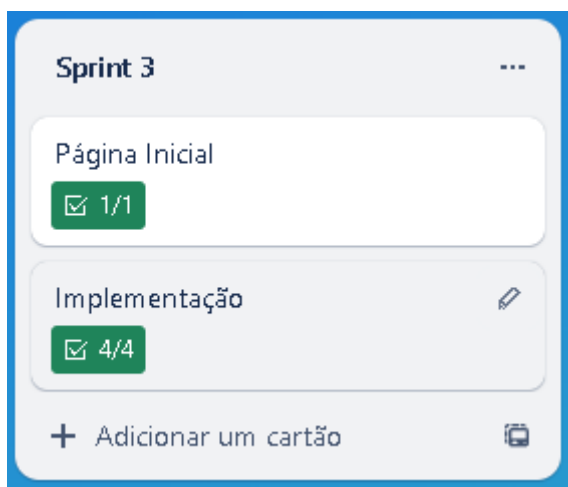


Figura 42 - Sprint 3: Criar página inicial

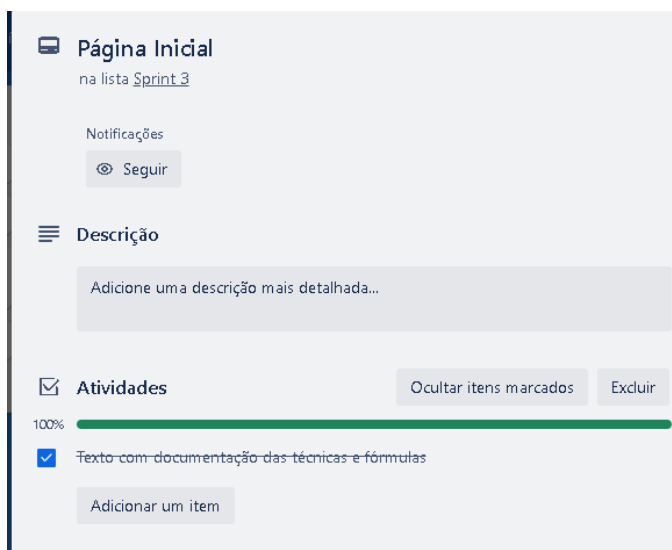
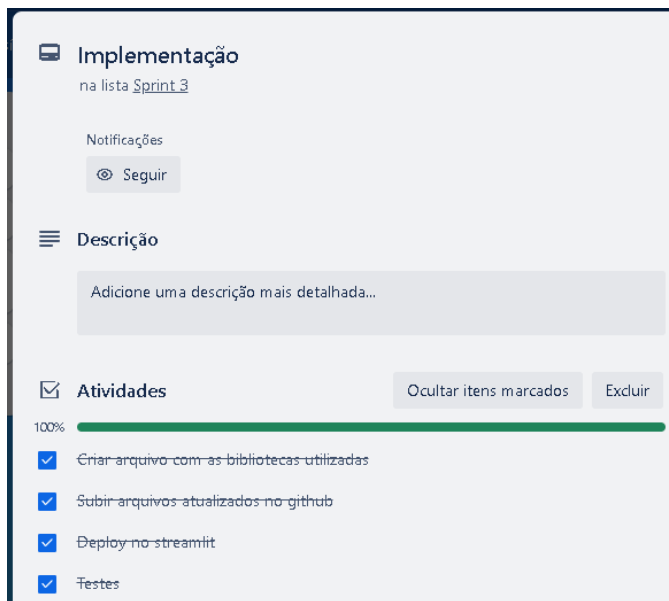


Figura 43 - Sprint 3: Implementação

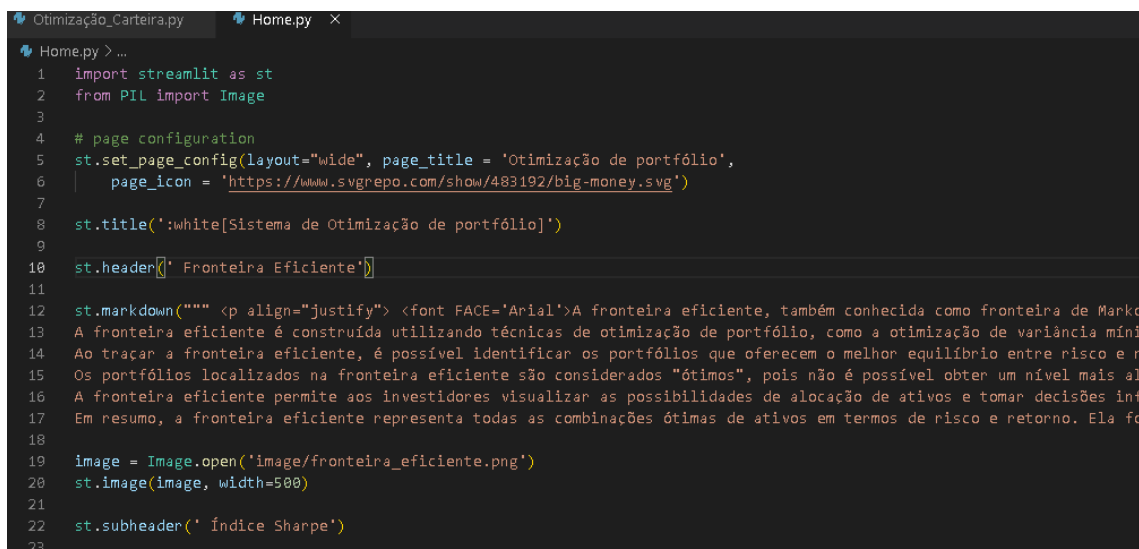


- Evidência da execução de cada requisito:

Para realização destas atividades foram utilizadas a biblioteca streamlit e a linguagem HTML para a criação da página inicial, e a ferramenta da web Streamlit Cloud para a implementação.

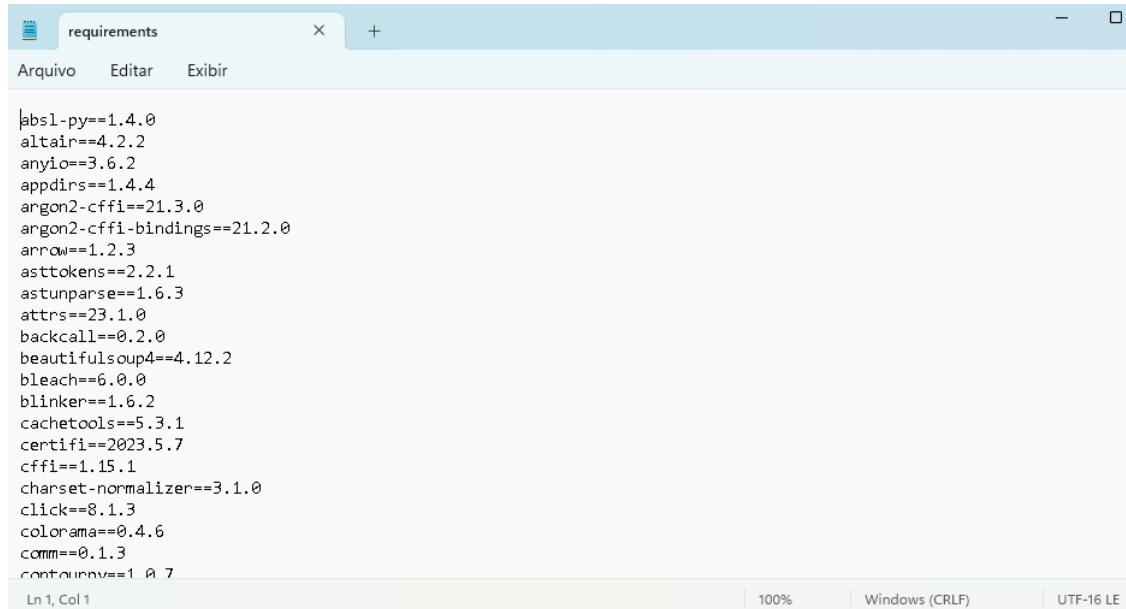
ITEM 1: Texto com documentação das técnicas e fórmulas

Figura 43 - arquivo py da página inicial



ITEM 2: Criar arquivo com as bibliotecas utilizadas

Figura 43 - arquivo requirements



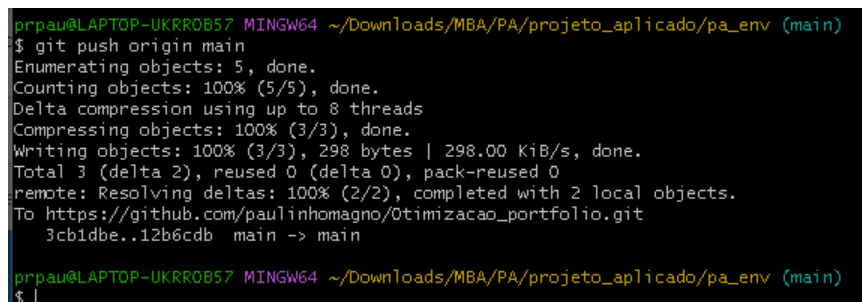
```

Arquivo  Editar  Exibir

|absl-py==1.4.0
|altair==4.2.2
|anyio==3.6.2
|appdirs==1.4.4
|argon2-cffi==21.3.0
|argon2-cffi-bindings==21.2.0
|arrow==1.2.3
|asttokens==2.2.1
|astunparse==1.6.3
|attrs==23.1.0
|backcall==0.2.0
|beautifulsoup4==4.12.2
|bleach==6.0.0
|blinker==1.6.2
|cachetools==5.3.1
|certifi==2023.5.7
|cffi==1.15.1
|charset-normalizer==3.1.0
|click==8.1.3
|colorama==0.4.6
|comm==0.1.3
|contourpy==1.0.7
Ln 1, Col 1  100%  Windows (CRLF)  UTF-16 LE
  
```

ITEM 3: Subir arquivos atualizados no GitHub

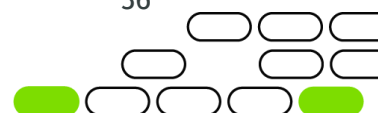
Figura 44 - prompt do Git Bash



```

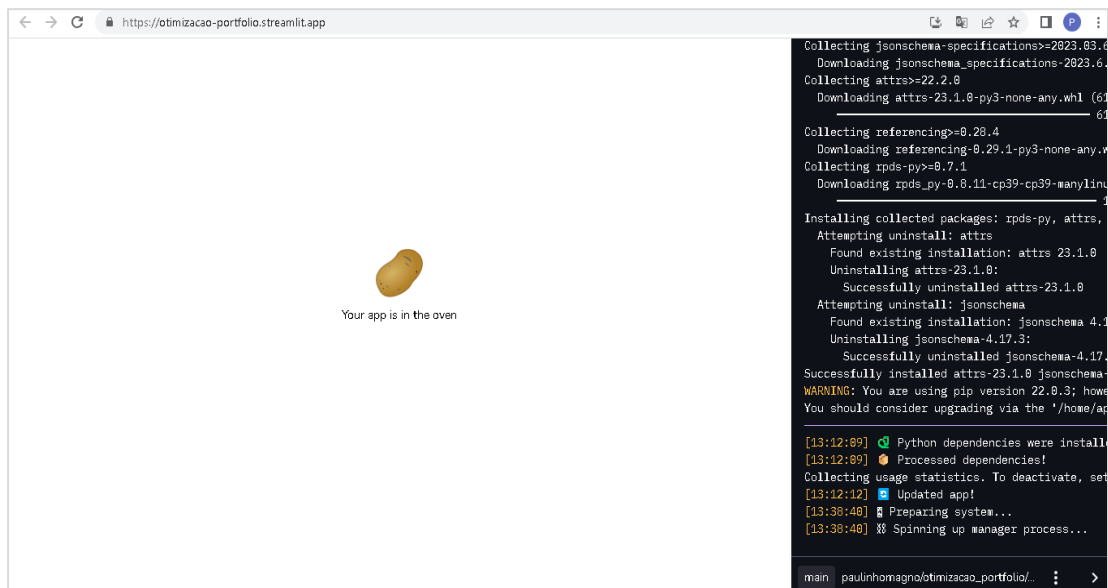
prpau@LAPTOP-UKRROB57 MINGW64 ~/Downloads/MBA/PA/projeto_aplicado/pa_env (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 298 bytes | 298.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/paulinhomagno/Otimizacao_portfolio.git
  3cb1dbe..12b6cdb  main -> main

prpau@LAPTOP-UKRROB57 MINGW64 ~/Downloads/MBA/PA/projeto_aplicado/pa_env (main)
$ |
  
```



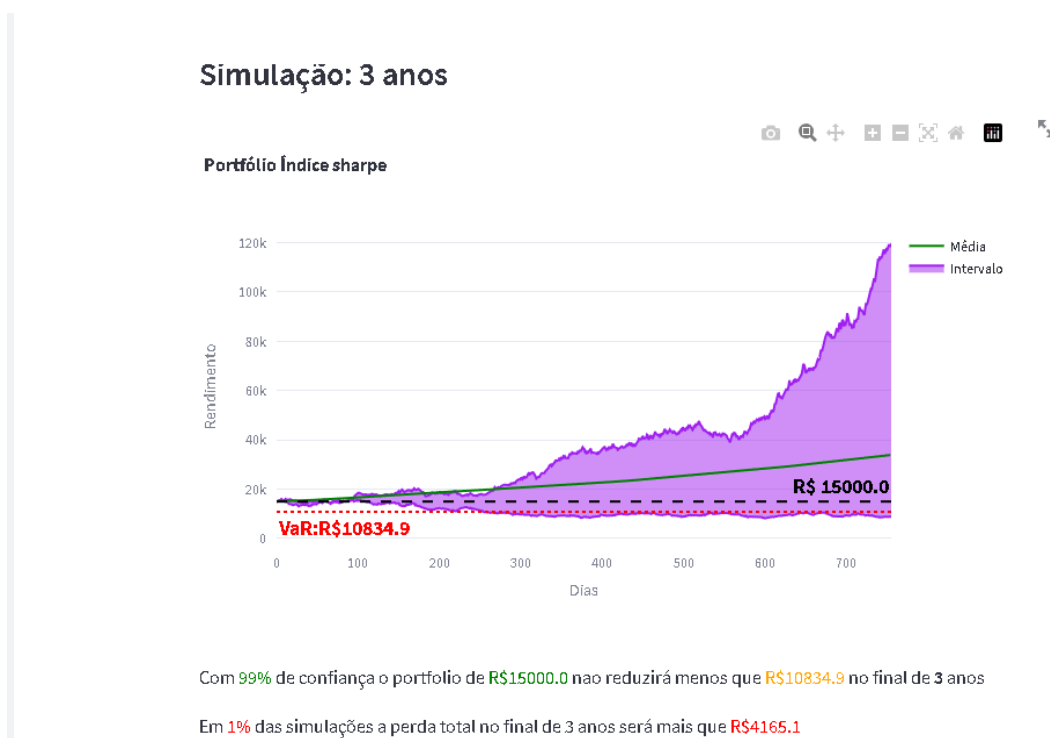
ITEM 4: Deploy no streamlit cloud

Figura 45 - logs do deploy



ITEM 5: Testes

Figura 46 - teste - simulação do portfólio índice sharpe



- Evidência dos resultados:

Nesta Sprint foi criada a página inicial da ferramenta utilizando o Streamlit, onde contém informações da ferramenta, a teoria das métricas, otimizações e fórmulas utilizadas. Após, a solução foi implementada através da nuvem do Streamlit, e assim ficando em produção através do link: <https://otimizacao-portfolio.streamlit.app/>.

Figura 47 - site disponível na web



Figura 48 - página inicial no ar

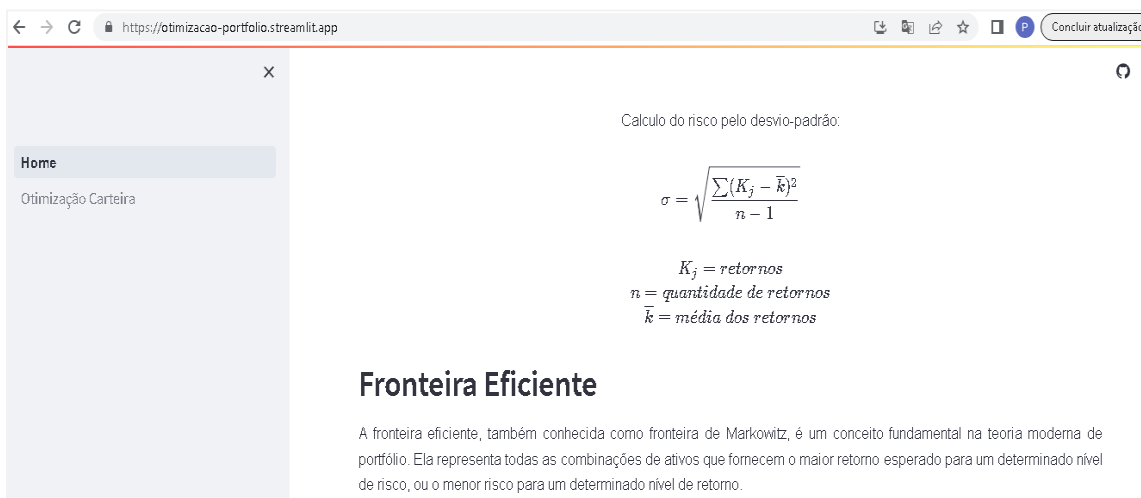


Figura 49 - exemplo de utilização da ferramenta na web

eamlit.app/Otimização_Carteira

Otimização da carteira

Digite os código dos ativos*:

vale3.sa × itub4.sa × petr4.sa × aapl × Pressione enter para adicionar mais

*Código dos ativos de acordo com o site: <https://finance.yahoo.com/>

Data inicial:

2020/07/24

Digite o valor

15000,00 - +

Mostrar otimizações

Figura 50 - parte do processamento do exemplo

Otimização pelo algoritmo Hierarchical Risk Parity (HRP)

Métricas

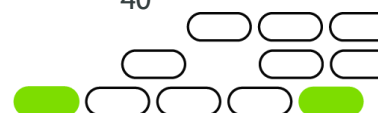
Métricas	Valor
Índice Sharpe	0.97
Volatilidade anual	21.49%
Retorno Esperado	20.94%
Beta do portfólio	0.86
CAPM	1.2%

Pesos de cada ativo:

Ativos	Pesos	Valor à aplicar	Beta
vale3.sa	0.2568	3,852.7237	0.84
itub4.sa	0.149	2,235.0718	1.04
petr4.sa	0.2635	3,951.9729	1.3
aapl	0.3307	4,960.2316	0.54

2.3.2 Lições Aprendidas

Desde o início do projeto de pensou em tornar a ferramenta disponível através de uma plataforma gratuita, o Heroku, mas, durante a idealização do projeto a versão gratuita não ficou mais disponível, assim, se pesquisou alternativas. O Render foi utilizado, porém, começou a se ter problemas com algumas versões de bibliotecas na plataforma, foi testado a nuvem do Google utilizando o serviço Google App Engine, mas, o projeto ultrapassou o limite de tamanho para o serviço gratuito. No final a ferramenta de deploy do Streamlit Share atendeu a necessidade de disponibilidade, apesar de ser demorado não se teve complicações no processo. A limitação ocorre ao não haver acessos durante algum tempo, o link pode ficar inativo e é necessário reativá-lo.



3. Considerações Finais

3.1 Resultados

Este projeto teve como objetivo desenvolver uma ferramenta que auxilia investidores a montar um portfólio de ativos baseados em técnicas e conceitos apresentados de forma automatizada, eliminando a necessidade de desenvolver cálculos e códigos pelo usuário.

A ferramenta processa os dados de ativos escolhidos e apresenta métricas, projeções e opções de otimização do portfólio em questão de segundos, de maneira visualmente simples e objetiva.

O primeiro desafio é a escolha de API's íntegras para a extração dos dados tanto de ativos quanto da taxa de juros utilizada (Selic), bem como o tratamento desses dados, uma vez que dados faltantes ou zerados poderiam comprometer os cálculos das bibliotecas a serem utilizadas.

Para agilizar os cálculos e a apresentação de resultados, a biblioteca PyPortfolioOpt foi empregada, mas, também apresentou desafios para tratar os dados e evitar possíveis erros. O algoritmo Hierarchical Risk Parity (HRP) trouxe dificuldades de compreensão e implementação. Na exibição dos resultados no gráfico da Fronteira Eficiente percebeu-se que esta otimização diferia significativamente das outras, levando à revisão de e do código e raciocínio. Posteriormente, com a realização de testes, os resultados foram ajustados, e descobriu-se que a indentação poderia ter causado as discrepâncias anteriores.

A escolha da projeção de Monte Carlo baseou-se na experiência de um projeto anterior, contribuindo para uma rápida implementação. Como esta técnica envolve a criação de centenas ou até milhares de projeções, a apresentação gerou extensa pesquisa e estudo para garantir que as previsões fossem exibidas de forma concisa e compreensível aos usuários.

A biblioteca Streamlit apresentou desafios para adequação das exibições de gráficos visualmente agradáveis e interativos. A escolha da biblioteca Plotly Express foi fundamental para alcançar esse resultado, mas a harmonia entre as duas exigiu muita pesquisa e testes.



A Sprint 1 foi a mais exigente em termos de tempo e desenvolvimento de códigos e cálculos, o que possibilitou que as seguintes não ficassem sobrecarregadas, permitindo considerar melhorias e maior eficiência no planejamento.

No final, os resultados obtidos por meio deste projeto demonstram que conseguimos oferecer uma resposta eficaz ao problema identificado e proposto, disponibilizando uma ferramenta capaz de auxiliar na tomada de decisão sobre investimentos de forma simples e automatizada.

3.2 Contribuições

Este projeto demonstra alta relevância ao apresentar uma ferramenta que automatiza a obtenção de resultados através de programação, cálculos e modelo preditivo proporcionando aos investidores ou profissionais de investimentos as métricas e visualização de previsões sobre os investimentos em questão.

O usuário ao inserir os ativos e a quantia, a ferramenta retorna em questão de segundos as opções de otimizações de portfólio, juntamente com seus indicadores e projeções. Essa abordagem eficiente oferece alternativas, conceitos e informações robustas que facilitam a tomada de decisão em um curto período de tempo.

Essa automatização e agilidade são fundamentais para proporcionar uma visão abrangente das possíveis oportunidades e riscos relacionados ao investimento estudado. Além disso, o fato de a ferramenta eliminar a necessidade de cálculos manuais e análises demoradas torna o processo mais acessível e prático a usuários de qualquer nível de experiência, salientando que esta ainda contém na página inicial uma documentação das bases teóricas e cálculos utilizados.

3.3 Próximos passos

Os próximos passos planejados para o projeto são visam aprimorar ainda mais a ferramenta, tornando-a ainda mais robusta e abrangente para os usuários.

- Aperfeiçoar o modelo preditivo: Testar diferentes abordagens e técnicas para desenvolver a projeção de Monte Carlo. Além disso, incluir outra forma de



predição com séries temporais pode agregar mais diversidade e opções aos investidores, ampliando o escopo de análise de dados.

- Incluir mais técnicas de otimização: Ao oferecer diversas técnicas de otimização de portfólio, a ferramenta poderá apresentar mais alternativas, permitindo adaptação às suas preferências e objetivos financeiros. Isso oferecerá mais informação e conhecimento, ajudando-os a tomar decisões mais informadas e personalizadas.
- Melhorar a infraestrutura e o design: Com a introdução de novas funcionalidades e técnicas mais avançadas, aprimorar a infraestrutura é essencial para garantir que a ferramenta mantenha alta disponibilidade e desempenho satisfatório. Utilizar serviços em nuvem pagos pode ser uma solução para garantir o processamento rápido e eficiente dos dados, mesmo em momentos de alto tráfego.

Estes passos visam colocar o projeto em constante evolução, oferecendo uma ferramenta cada vez mais robusta e completa para auxiliar nas decisões construção de portfólios sólidos e rentáveis.

