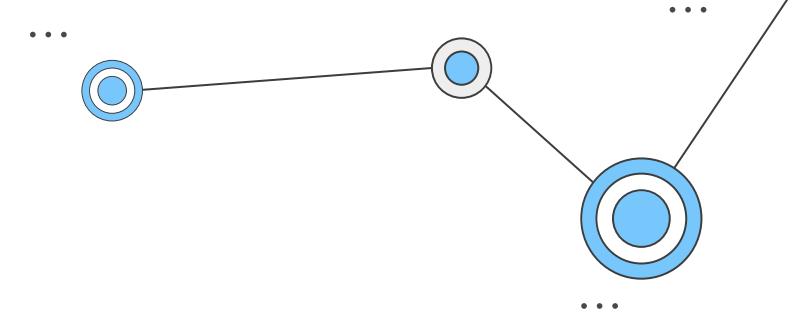




PUC Minas



GitHub API

Prof. Dr. João Paulo Aramuni



“Elaboração de roteiro de aula prática, com demonstração, de experimento que coleta e analisa as características dos repositórios mais populares do GitHub que utilizam arquitetura de Microsserviços”

Engenharia de Software

Apresentação

Aramuni

- 13 anos na área de desenvolvimento de sistemas
 - Java / Spring Boot
 - Python / Raspagem de Dados
 - Node.js / Back-end / Crawlers
 - VB6, VBA, ASP, C#, .Net
- 9 anos como professor em cursos de tecnologia
 - 4 anos e meio como professor dos cursos de Ciência da Computação, Sistemas de Informação e Redes de Computadores da Universidade Fumec.
 - 3 anos e meio em Startup EdTech (Trybe).
 - 1 ano nos cursos de Ciência da Computação, Análise e Desenvolvimento de Sistemas e Sistemas de Informação da Newton Paiva.
 - Lecionando no curso de Engenharia de Software e Ciência da Computação há 1 ano na PUC Minas.
- Doutor e mestre em Sistemas de Informação e Gestão do Conhecimento.
- Bacharel em Ciência da Computação e cursando Licenciatura em Andragogia.
- 15 artigos publicados em revistas científicas de nível Qualis B1 e B2.
- 31 TCC's orientados no curso de Ciência da Computação.





...

Experiência Profissional: 13 anos

Experiência Docente: 9 anos

Contato:

- Agenda: <https://calendly.com/aramuni/30min>
- LinkedIn: <https://www.linkedin.com/in/joaopauloaramuni/>
- GitHub: <https://github.com/joaopauloaramuni>
- Email: joaopauloaramuni@gmail.com

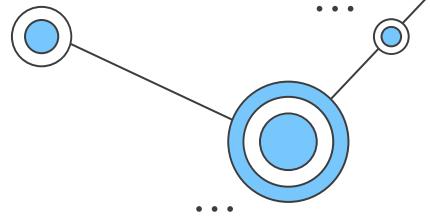
Assuntos de interesse:

- Inteligência Artificial
- Andragogia
- Tecnologias para ensino remoto

...



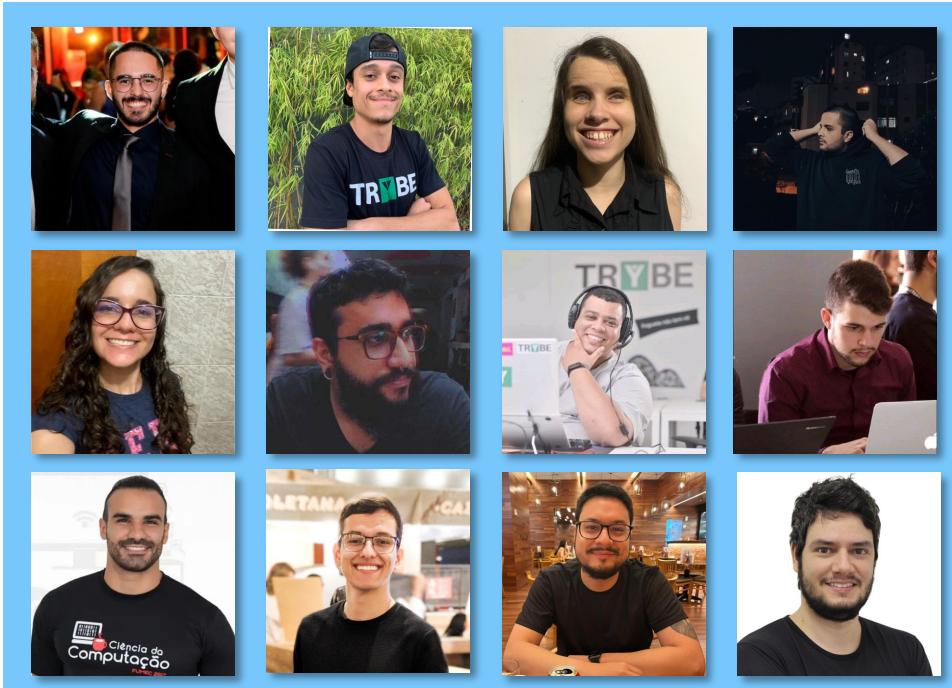
Troféus



Prof. Horácio Slughorn

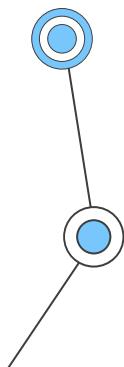
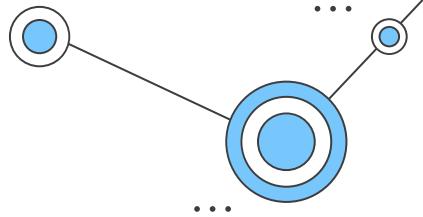


Mantinha fotos dos estudantes como troféus





Passado como Dev



Passado como Dev



Sistema: Automatização do cadastro de gasoduto – (SIMP)
Analista Elaborador (Ramal): Simone Teixeira – Ramal 8825
Emissão: 16/04/2014
Cenário: T2 42381

Automatização do cadastro de gasoduto no sistema SIMP

Data	Versão	Autor	Descrição
11/04/2014	1.0.0	Simone Teixeira	Documentação original
16/04/2014	1.0.1	Simone Teixeira	Alterações após validação da qualidade
25/04/2014	1.0.2	Simone Teixeira	Alterações após validação da usuária

Sumário

<u>1</u>	<u>OBJETIVO:</u>	2
<u>2</u>	<u>VALOR AGREGADO:</u>	2
<u>3</u>	<u>USUÁRIO RESPONSÁVEL PELA ATIVIDADE:</u>	2
<u>4</u>	<u>MINIMUNDO</u>	2
<u>4.1</u>	<u>RESUMO DA REGRA DE NEGÓCIO</u>	2
<u>4.2</u>	<u>VISÃO ATUAL DO NEGÓCIO</u>	3
<u>4.3</u>	<u>ALTERAÇÕES NO FLUXO DE NEGÓCIO</u>	3



Microsoft
Visual Basic 6.0

Passado como Dev

Gestión de Rutas Maestras

Filtro

Delegación: Tres Arroyos Día Semana: Dom, Lun, Mar, Mier, Jue, Vie,... Hábil? Clase Fija? Especial? Eventual?

Rutas: Inicio: Fin: Activo? Feriado? Conjunto Diario? Semanal?

Rutas

Delegación	Ruta	Día Semana	Inicio	Fin	Tipo Ruta	Descripción
Tres Arroyos	241	Dom	11:45	01:30	Trans...	DOMINGO
Tres Arroyos	242	Dom	11:00	22:00	Trans...	DOMINGO
Tres Arroyos	243	Dom	06:30	15:00	Trans...	DOMINGO
Tres Arroyos	971	Dom	00:31	11:00	GUAR...	DOMINGO
Tres Arroyos	100	Dom, Lun, Mar,...	07:00	17:01	Trans...	X4 ##
Tres Arroyos	20	Lun	01:00	12:34	ATM	ATM LU
Tres Arroyos	908	Lun, Mar	05:30	16:00	GUAR...	GUARDI...
Tres Arroyos	42	Lun, Mar, Mier	01:30	11:41	ATM	ATM X4

Cantidad de líneas 168

Recursos

Posee Tm: Nro Serie Tm:

Posee vehículo: Tipo vehículo: Selecione

Tamaño vehículo: Selecione

Nro vehículo:

Dispositivos: 0

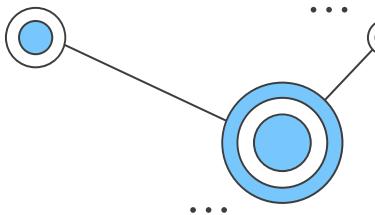
Permisos: 0

Personas:

Hr. Citação	Función	Recursos Programados	Habilidades Requeridas	Nivel Conocimiento	Habilidades

No hay líneas

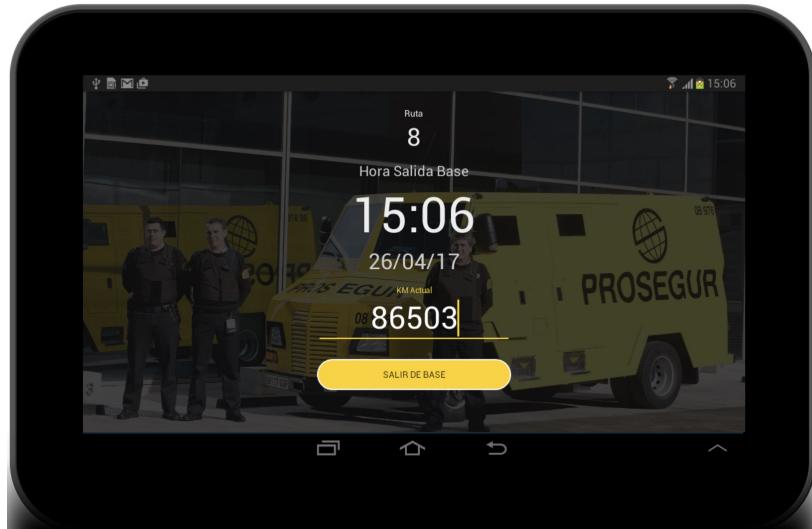
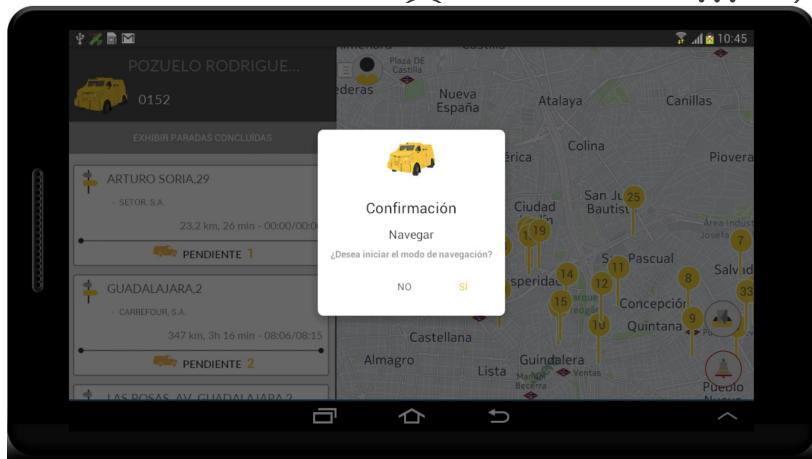
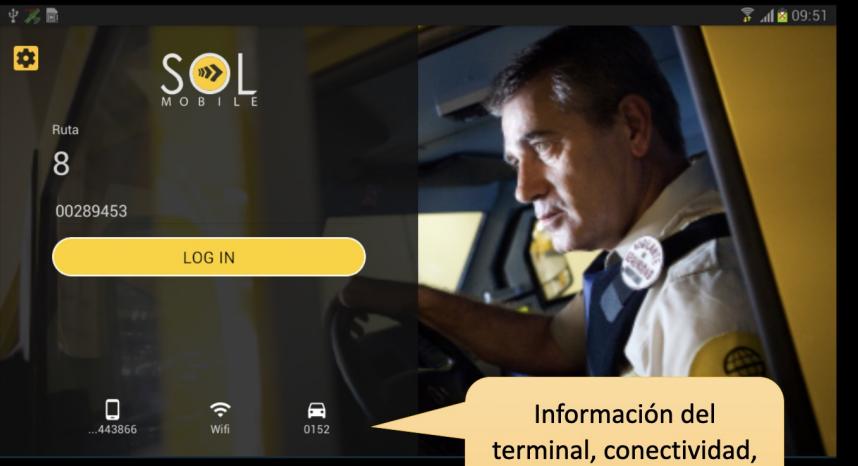
Grabar (F12) Cerrar (ESC)



Passado como Dev



Passado como Dev



Passado como Dev

[SOL] (Internal) / SOLPLATINT-1044

Programación de Recursos Directos: nueva columna día anterior

Editar Asignar Asignarme a mí Comentar Más Acciones moveToBeRelease Flujo de Trabajo

Criterio 1: Mostrar la columna día anterior sin tildar

Dado:	El programador de recursos genera las rutas del día
Y:	la fecha de citación de los recursos coincide con la fecha de inicio de la ruta
Cuando	el programador de recursos selecciona una ruta
Entonces:	El checkbox día anterior NO se muestra desmarcado (Ver imagen PRD - Día Anterior 1)

Criterio 2: Mostrar la columna día anterior tildada

Dado:	El programador de recursos genera las rutas del día
Y:	para al menos una ruta, la fecha de citación de los recursos es del día anterior a la fecha de inicio de la ruta
Cuando	el programador de recursos seleccionada una de estas rutas
Entonces:	El checkbox día anterior se muestra tildado (Ver imagen PRD - Día Anterior 3)

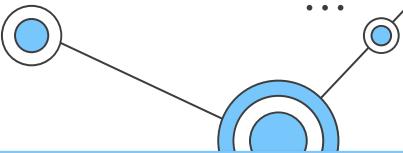
Criterio 3: Bloqueo al informar un horario de citación mayor al horario de salida de la ruta

Dado:	hay una ruta seleccionada en la pantalla Prog. Recursos Directos
Y:	el horario de citación de al menos un recurso es mayor al horario de inicio de la ruta
Y:	el checkbox Día Anterior NO está tildado
Cuando	el programador de recursos presiona Grabar
Entonces:	Sol mostrará un bloqueo (Msj009) solicitando al programador de recursos que ingrese un nuevo horario de citación (Ver imagen PRD - Día Anterior 2)

Criterio 4: Informar un horario de citación mayor al horario de salida de la ruta y tildar checkbox Día Anterior

Dado:	hay una ruta seleccionada en la pantalla Prog. Recursos Directos
Y:	el horario de citación de al menos un recurso es mayor al horario de inicio de la ruta
Y:	el checkbox Día Anterior está tildado
Cuando	el programador de recursos presiona Grabar

Passado como Dev



PROSEGUR

Cuadros de mandos | Proyectos | Incidencias | Agile | Simple Calendar | + Nueva Incidencia | Búsqueda Rápida

Filtros

Filtro nuevo

Buscar filtros

Mis Incidencias Abiertas

Reportados por Mí

Recientemente Vistas

Todas las Incidencias

Filtros Favoritos

Control_de_Rutas...

Bugs (Desenvolvimento) 3.2.0 — Editada — Guardar como

Poseído por: Daniel Alves Cavalcanti

Proyecto: Todos | Tipo Incidencia: Todos | Estado: Abierta, En progreso, ... | Responsable: Usuario Actual | Contiene el tex | + Más Criterios | Cambiar a Avanzada

1–5 de 5

T	Clave	Sumario	Responsable	Informador	Pr	Estado	Resolución	Created	Actualizada
	SOLPLATINT-1240	Ícone de seta usado para expandir e recolher dados do grid foi removido	João Paulo Carneiro Aramuni - Capgemini	Priscila Kelly dos Santos Chagas Costa - Squadra		Reviewed PO	Sin resolver	27/10/2015	28/10/2015
	SOLPLATINT-1239	Controle de Rotas: O sistema está apresentando uma validação errada ao alterar a hora inicio de uma OT de descarga que está colada na rota	João Paulo Carneiro Aramuni - Capgemini	Cesar Junior Guerra - MTP		Reviewed PO	Sin resolver	27/10/2015	27/10/2015
	SOLPLATINT-1227	Controle de Rotas: Ao reprogramar uma OT de forma manual (sem marcar o checkbox "Aplicar Automaticamente"), o sistema está bloqueando a linha da ot reprogramada.	João Paulo Carneiro Aramuni - Capgemini	Cesar Junior Guerra - MTP		Reviewed PO	Sin resolver	26/10/2015	26/10/2015
	SOLPLATINT-1219	Control de rutas: verificar alterar horas prevista e realizada que não é gravada no banco de dados	João Paulo Carneiro Aramuni - Capgemini	Mariana Mirale de Sena Pinto		En progreso	Sin resolver	23/10/2015	26/10/2015
	SOLPLATINT-1207	Prog. recursos directo: é possível programar a mesma pessoa duas vezes no mesmo horário ao marcar a mesma como dia anterior	João Paulo Carneiro Aramuni - Capgemini	Mariana Mirale de Sena Pinto		Reviewed PO	Sin resolver	22/10/2015	28/10/2015

Passado como Dev



The image shows a screenshot of the HotMilhas website. At the top left is the logo "HotMilhas". At the top right is a navigation bar with links: Início, Como vender suas milhas, Quem somos, Dúvidas, Blog, and Minha conta. Below the navigation is a large blue banner. On the left side of the banner, there's a form asking for name and email to get a quote. On the right side, there's a call-to-action for scheduled delivery with the text "GANHE MAIS" and "Suas milhas valem mais com o recebimento agendado." Two men wearing "HotMilhas" t-shirts are standing in front of the banner. One man is smiling, and the other has his hands on his hips. At the bottom left of the banner is a red button labeled "COTAR AGORA". At the bottom center is a small "SITE BLINDADO" badge.

HotMilhas

Início Como vender suas milhas Quem somos Dúvidas Blog | Minha conta

Cotação em 1 minuto
em seu e-mail

Nome:

E-mail:

Quero vender minhas milhas:

Qtd. milhas LATAM >

Qtd. milhas GOL >

Qtd. milhas AZUL >

Qtd. LATAM Platinum/Black >

COTAR AGORA

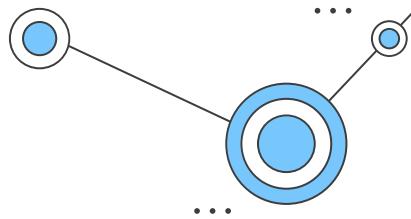
SITE BLINDADO

Veja nossa Política de Privacidade



Passado como Dev

The screenshot shows the homepage of the 123milhas website. At the top, there's a navigation bar with links for "Passagens aéreas", "Melhores destinos", "Pacotes Promocionais", "Carros", "Seguro Viagem", and a green button labeled "Venda suas milhas!" with a dollar sign icon. Below the navigation is a large orange banner with the text "Compre passagens aéreas com até 50% de desconto". It features three men wearing 123milhas t-shirts. On the left side of the banner is a search form for flights, including fields for departure city, arrival city, travel dates, and passengers, along with a "Buscar" button. Below the banner, there's a section for payment options with a "12x com juros" logo and a note stating "Você não precisa ter milhas para comprar suas passagens com desconto!". At the bottom, there's a dark overlay with the text "Viaje com os nossos melhores pacotes" and a dropdown menu showing "Saindo de: Florianópolis".



Passado como Dev

The screenshot shows the homepage of the Busca Milhas website. At the top, there's a navigation bar with links for HOME, SOBRE, SISTEMA BUSCADOR, BLOG, CONTATO, and START. There's also a language switcher for PT. The main content area features a large blue background with white text. It says "Buscador de passagens com milhas" and "Todo que sua empresa de milhas precisa em um só lugar". Below this, it lists "As melhores ferramentas para evoluir no mercado de milhas:" followed by a bulleted list of four items. To the right of the text is a graphic of three people (two men and one woman) looking at a globe with a flight path and a plane.

Buscador de passagens com milhas

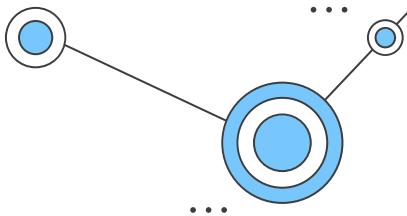
Todo que sua empresa de milhas precisa em um só lugar

As melhores ferramentas para evoluir no mercado de milhas:

- ✓ Desde 2010 no mercado, entendemos o mercado de milhas e levamos as melhores práticas para dentro da sua empresa.
- ✓ Alugamos o melhor sistema de gestão para que você possa controlar seu estoque e vender suas milhas.
- ✓ Mostramos sua marca para o mundo, sabemos como atrair os fornecedores e alcançar seus clientes através do marketing digital.



Cursos que ajudei a criar



Como Tech Lead na Trybe:

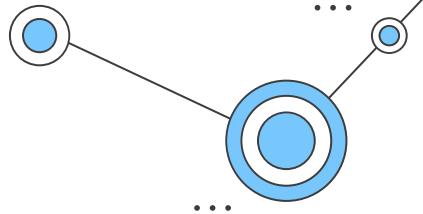
- Desenvolvimento Full Stack
 - Ciência da Computação
 - Estrutura de Dados I e II com Python
 - Análise de Algoritmos com Python
- Certificação de Python
 - Raspagem de Dados
 - POO com Python
 - Flask e Django
- Certificação de Java
 - POO com Java
 - Spring
 - Quarkus

Como professor na Fumec:

- Curso Superior de Tecnologia em Redes de Computadores
- Certificação de WordPress



Alguns artigos que publiquei



Gestão & Tecnologia de Projetos

10560an
2020-2021

Atual Arquivos Notícias Sobre ▾

Início / Arquivos / v. 13 n. 1 (2018) / Artigos

ANÁLISE DA ADOÇÃO DO LEAN MANUFACTURING NA GESTÃO DE PROJETOS DE TECNOLOGIA DA INFORMAÇÃO: ESTUDO DE CASO EM UMA MULTINACIONAL DESSE SEGMENTO

João Paulo C. Aramuni

Fundação Mineira de Educação e Cultura

Luiz Cláudio Gomes Maia

Fundação Mineira de Educação e Cultura

DOI: <https://doi.org/10.11606/gtp.v13i1.105650>

Palavras-chave: Lean Manufacturing, Filosofia Lean, Cultura Ágil, Gestão Ágil de Projetos

Resumo

Este artigo apresenta um estudo sobre gestão ágil de projetos por meio do



Ciência da Informação

Atual Edições anteriores Anúncios Sobre ▾

Início / Anteriores / v. 48 n. 1 (2019) / Revisão de Literatura

Filosofia ágil aplicada à gestão do conhecimento: um mapeamento sistemático da literatura

João Paulo Carneiro Aramuni

Universidade Fumec (Fumec)

Luiz Cláudio Gomes Maia

Universidade Fumec (Fumec)

Cristiana Fernandes de Muylder

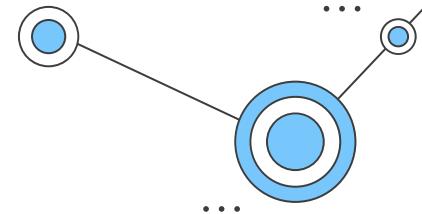
Universidade Fumec (Fumec)

Palavras-chave: Filosofia ágil, Gestão ágil do conhecimento, Métodos ágeis, Gestão do conhecimento, Mapeamento sistemático de literatura

Resumo



Alguns artigos que publiquei



educação & tecnologia

Capa Sobre Acesso Cadastro Pesquisa Atual

Capa > v. 22, n. 3 (2017) > Aramuni

**O IMPACTO DA TECNOLOGIA DA INFORMAÇÃO NO ENSINO SUPERIOR:
DESAFIOS DA UBIQUIDADE NA APRENDIZAGEM ESTUDANTIL**

João Paulo Aramuni

Resumo

Este artigo apresenta uma abordagem sobre a computação ubíqua a serviço do ensino superior do ponto de vista da contribuição das novas tecnologias digitais na aprendizagem. O período de convergência tecnológica da atual sociedade da informação tem迫使 alunos e professores a adaptarem-se a novos métodos de exposição de conteúdo e transferência de conhecimento. A análise de exemplos de interações homem-máquina entre o meio digital e o ambiente acadêmico ilustra as considerações teóricas sobre novas formas de absorção de conteúdo e descobertas do saber, mais precisamente a da ubiquidade, na construção de novas experiências cognitivas para o aluno e na modernização do ensino por parte dos professores.

Texto completo:

PDF

INÍCIO / ARQUIVOS / V. 23 N. 2 (2018) / ARTIGOS

A INFLUÊNCIA DA ENGENHARIA SEMIÓTICA NA EXPERIÊNCIA DO USUÁRIO DE APLICATIVOS MOBILE: Uma reflexão sobre a relação entre semiose e o desenvolvimento de APPs / THE INFLUENCE OF SEMIOTIC ENGINEERING IN THE USER EXPERIENCE OF MOBILE APPLICATIONS: A reflection on the relationship between semiosis and the development of APPs

João Paulo Carneiro ARAMUNI

Universidade FUMEC - MG

Luís Cláudio Gomes MAIA

Universidade FUMEC - MG

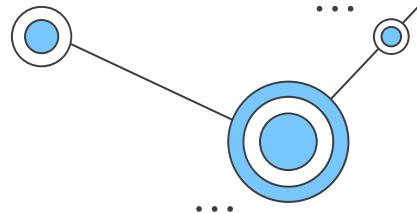
DOI: <https://doi.org/10.22478/ufpb.2446-7006.2018v23n2.43701>

RESUMO

RESUMO. Este artigo apresenta uma abordagem sobre a influência da



Meu GitHub



Todos os meus códigos e aulas

arquitetura-de-aplicacoes-web Public

Repo Arquitetura de Aplicações Web

● HTML ⭐ 3 Updated yesterday

linguagens-de-programacao Public

Repo Linguagens de Programação

● Java ⭐ 18 ⚡ 1 Updated 4 days ago

banco-de-dados Public

Repo Banco de Dados

⭐ 15 Updated 4 days ago

algoritmos-e-estruturas-de-dados-i Public

Repo Algoritmos e Estruturas de Dados I

⭐ 1 Updated 4 days ago

fundamentos-teoricos-da-computacao Public

Repo Fundamentos Teóricos da Computação

⭐ 13 Updated on Nov 9, 2023

<https://github.com/joaopauloaramuni>



João Paulo Aramuni
joaopauloaramuni · he/him

Tech Manager | Tech Lead | Professor

[Edit profile](#)

1.1k followers · 135 following

- Newton Paiva
Belo Horizonte/MG
11:58 (UTC -03:00)
joaopauloaramuni@gmail.com
<https://joaopauloaramuni.github.io/>
<https://orcid.org/0000-0001-7538-5927>
in/joaopauloaramuni
[@joaopauloaramuni](https://twitter.com/@joaopauloaramuni)
[@joaoaramuni](https://x.com/@joaoaramuni)

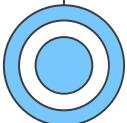
joaopauloaramuni / README.md



Olá 👋, sou o Aramuni, tenho 33 anos, moro em BH e sou programador desde os 15 anos de idade. Atualmente sou professor nos cursos de Ciência da Computação, Sistemas de Informação e Análise e Desenvolvimento de Sistemas do Centro Universitário Newton Paiva e também nos cursos de Engenharia de Software e Ciência da Computação da PUC Minas. 🎓

Sobre mim:

Doutor (2017-2020) e mestre (2014-2015) em Sistemas de Informação e Gestão do Conhecimento pela Universidade FUMEC, onde também obteve graduação em Ciência da Computação (2010-2013). Cursa licenciatura em Andragogia pela Univasellvi (2023-2027). Profissional com 15 anos de experiência no mercado de desenvolvimento de sistemas e há 9 anos na área de educação com ensino de tecnologia. Atualmente, é professor das disciplinas de Linguagens de Programação (Java), Arquitetura de Aplicações Web e Banco de Dados dos cursos de Ciência da Computação, Sistemas de Informação e Análise e Desenvolvimento de Sistemas do Centro Universitário Newton Paiva. Também é professor das disciplinas de Algoritmos e Estruturas de Dados I (Linguagem C) e Trabalho Interdisciplinar: Aplicações Web do curso de Engenharia de Software e da disciplina de Laboratório de Iniciação à Programação do curso de Ciência da Computação da PUC Minas. Atuou como Tech Manager na holding IN8, responsável pela liderança de squads de desenvolvimento de múltiplos projetos para o mercado de milhas aéreas. Foi Tech Lead



...

Prova didática



Duas etapas:

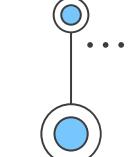
...

- Roteiro de aula prática
- Demonstração do experimento

...

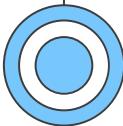


...



...





Roteiro de aula prática

1 – Introdução ao tema

- Introdução ao conceito de microsserviços e a sua importância.
- Apresentação do objetivo da aula: coletar e analisar características de repositórios populares no GitHub que utilizam a arquitetura de microsserviços.

...

2 – Preparação do ambiente

- Instalação do Python 3
- (Virtual env) Criação do ambiente virtual usando `python3 -m venv .venv`
- (Virtual env) Ativação do ambiente virtual usando `source .venv/bin/activate`
- (Dependências) Instalação da biblioteca `requests` usando `pip3 install requests`

...

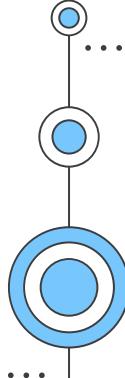
3 – Geração de token de acesso

- Criação de um token de acesso pessoal no GitHub nas configurações de desenvolvedor: <https://github.com/settings/tokens>

...

4 – Apresentação da documentação da API REST do GitHub

- <https://docs.github.com/pt/rest?apiVersion=2022-11-28>





Roteiro de aula prática

5 - Criação do código em Python

- Mão na massa: Coletar dados e métricas dos Top 10 repositórios mais populares que utilizam microsserviços.

6 - Explicação e execução do código

- Função por função, mostrando como fazer as requisições à API do GitHub para coletar dados dos repositórios.

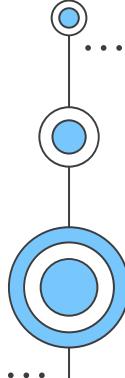
7 - Análise dos resultados

- Discutir e comparar as principais características de cada repositório.

8 - Criação de um relatório final sobre o experimento

- Conclusão sobre o que foi aprendido
- Explicar como os dados coletados podem ser usados para tomar decisões sobre a escolha de tecnologias e práticas de desenvolvimento em um projeto

...



Introdução ao tema

...

Conceito de microsserviços e a sua importância:

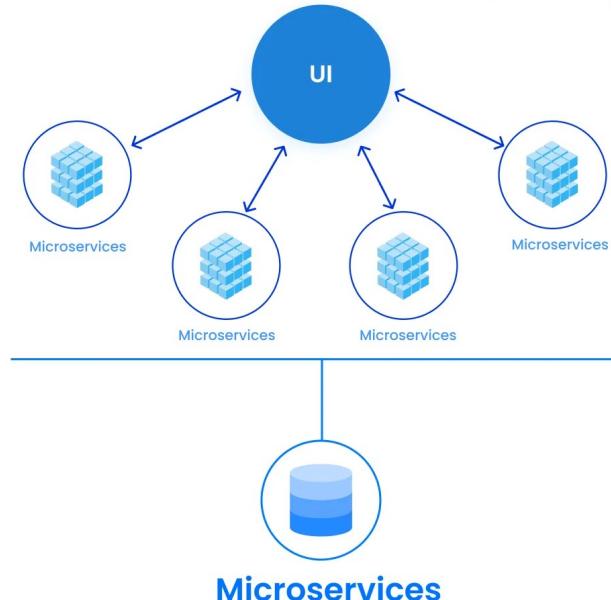
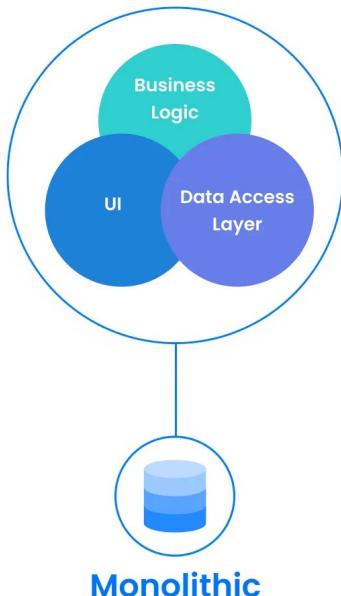
- Microsserviços são uma abordagem arquitetural para desenvolver aplicações de software como um conjunto de pequenos serviços independentes, cada um executando um processo de negócio específico.
- Cada serviço é construído em torno de uma regra de negócio única e pode ser desenvolvido, implantado, escalado e gerenciado de forma **independente**.

Demonstração



Introdução ao tema

- **Conceito de microsserviços** e a sua importância:



Demonstração

Introdução ao tema

...

Conceito de microsserviços e a sua importância:

- Nos últimos anos, o setor de tecnologia tem testemunhado uma rápida mudança na arquitetura aplicada em sistemas distribuídos, o que tem levado os gigantes da indústria como Netflix, Twitter, Amazon, eBay e Uber a abandonar a construção de aplicações monolíticas e adotar a arquitetura de microsserviços.



Demonstração



Introdução ao tema

...

Objetivo da aula: coletar e analisar características de repositórios populares no GitHub que utilizam a arquitetura de microsserviços.

- Ranquear os Top 10 repositórios mais populares que utilizam microsserviços.
- Coletar dados: Repository name, Owner, Stars, Forks, Commits, Watchers, Pull requests, Last commit date, Main language, License, contributors, Size, Main branch, Releases, Closed issues e Topics.

Demonstração



Preparação do ambiente

...

Instalação do Python 3

- Windows: <https://www.python.org/>
- Linux: sudo apt install python3 python3-venv python3-pip
- macOS: brew install python

Criação do ambiente virtual

- Um ambiente virtual em Python é um ambiente isolado onde você pode instalar pacotes Python e suas dependências sem interferir no sistema Python global ou em outros ambientes virtuais que você tenha configurado.
- (*Criação*) `python3 -m venv .venv`
- (*Ativação*) `source .venv/bin/activate`

Dependências

- `pip3 install requests`
- `pip3 install numpy`

Demonstração



03

Geração de token de acesso

Demonstração

...

Criação de um token de acesso pessoal no GitHub nas configurações de desenvolvedor:

<https://github.com/settings/tokens>

The screenshot shows the GitHub settings interface for managing personal access tokens. On the left, there's a sidebar with options like GitHub Apps, OAuth Apps, Personal access tokens (selected), Fine-grained tokens (Beta), and Tokens (classic). The main area is titled "Edit personal access token (classic)" and contains a note: "Make sure to copy your token now as you will not be able to see it again." Below this is a text input field containing a token value: "ghp_ClaF0lx3Twpr99VPVCfy7drxld2fw1fqG5w". To the right of the input is a blue "Copy" button with a clipboard icon. Below the input field is a "Note" section with the text "meutoken" in a box. Under "Expiration", it says "This token has no expiration date. To set a new expiration date, you must [regenerate the token](#)".

Geração de token de acesso

...

Utilização do token no código python e no cabeçalho (headers) da requisição http:

```
main.py > ...
1 import requests
2
3 # Adicione seu token de acesso pessoal aqui
4 token = "ghp_ClaF0Ix3Twpr99VPVCfy7drxld2fw1fqG5w"
```

```
headers = {"Authorization": f"token {token}"}
response = requests.get(url, headers=headers)
```

Demonstração



Documentação da API REST do GitHub

<https://docs.github.com/pt/rest?apiVersion=2022-11-28>

Demonstração

The screenshot shows the GitHub REST API documentation page. At the top left is the GitHub logo and 'GitHub Docs'. To its right is a dropdown menu showing 'Version: Free, Pro, & Team' and a search bar with a magnifying glass icon. On the far right is a 'Pesquisar no GitHub' input field.

The main header is 'Documentação da API REST do GitHub' with a subtitle: 'Crie integrações, recupere dados e automatize seus fluxos de trabalho com a API REST do GitHub.' Below the header are two tabs: 'Visão geral' (selected) and 'Guia de Início Rápido'.

The left sidebar contains a navigation tree under 'REST API' with sections like 'Início Rápido', 'Sobre a API REST', 'Usando a API REST', 'Autenticação', 'Guides', 'Ações', 'Atividade', 'Aplicativos', 'Cobrança', 'Branches', 'Verificações', 'Sala de aula', 'Varredura de código', 'Code security settings', 'Códigos de conduta', 'Espaços de código', and 'Colaboradores'. Each section has a dropdown arrow indicating more content.

The main content area features several cards:

- Sobre a API REST**: 'Obtenha orientação sobre a documentação da API REST.'
- Introdução à API REST**: 'Saiba como usar a API REST do GitHub.'
- Autenticação na API REST**: 'Você pode se autenticar na API REST para acessar mais pontos de extremidade e ter um limite de taxa mais alto.'
- Práticas recomendadas para usar a API REST**: 'Siga estas melhores práticas quando usar a API do GitHub.'
- Popular**:
 - Limites de taxa para a API REST**: 'Saiba mais sobre os limites de taxa da API REST, como não ultrapassá-los e o que fazer se você excedê-los.'
 - Solucionar problemas do API REST**: 'Saiba como diagnosticar e resolver problemas comuns de API REST.'
 - Scripts com a API REST e o JavaScript**: 'Escreva um script usando o SDK do Octokit.js para interagir com a API REST.'
 - Manter suas credenciais de API seguras**: 'Siga estas práticas recomendadas para manter as suas credenciais de API e tokens seguros.'
- Novidades**:
 - GitHub Apps can now use the client ID to fetch installation tokens**: 'May 01'
 - Check if private vulnerability reporting is enabled via REST API**: 'March 08'
 - New limits on scoped token creation for GitHub Apps**: 'February 22'

Documentação da API REST do GitHub

<https://docs.github.com/en/rest/repos?apiVersion=2022-11-28>

← Home

REST API

API Version: 2022-11-28 (latest)

- Quickstart
- About the REST API
- Using the REST API
- Authentication
- Guides
- Actions
- Activity
- Apps
- Billing
- Branches
- Checks
- Classroom
- Code scanning
- Code security settings
- Codes of conduct
- Codespaces
- Collaborators
- Commits

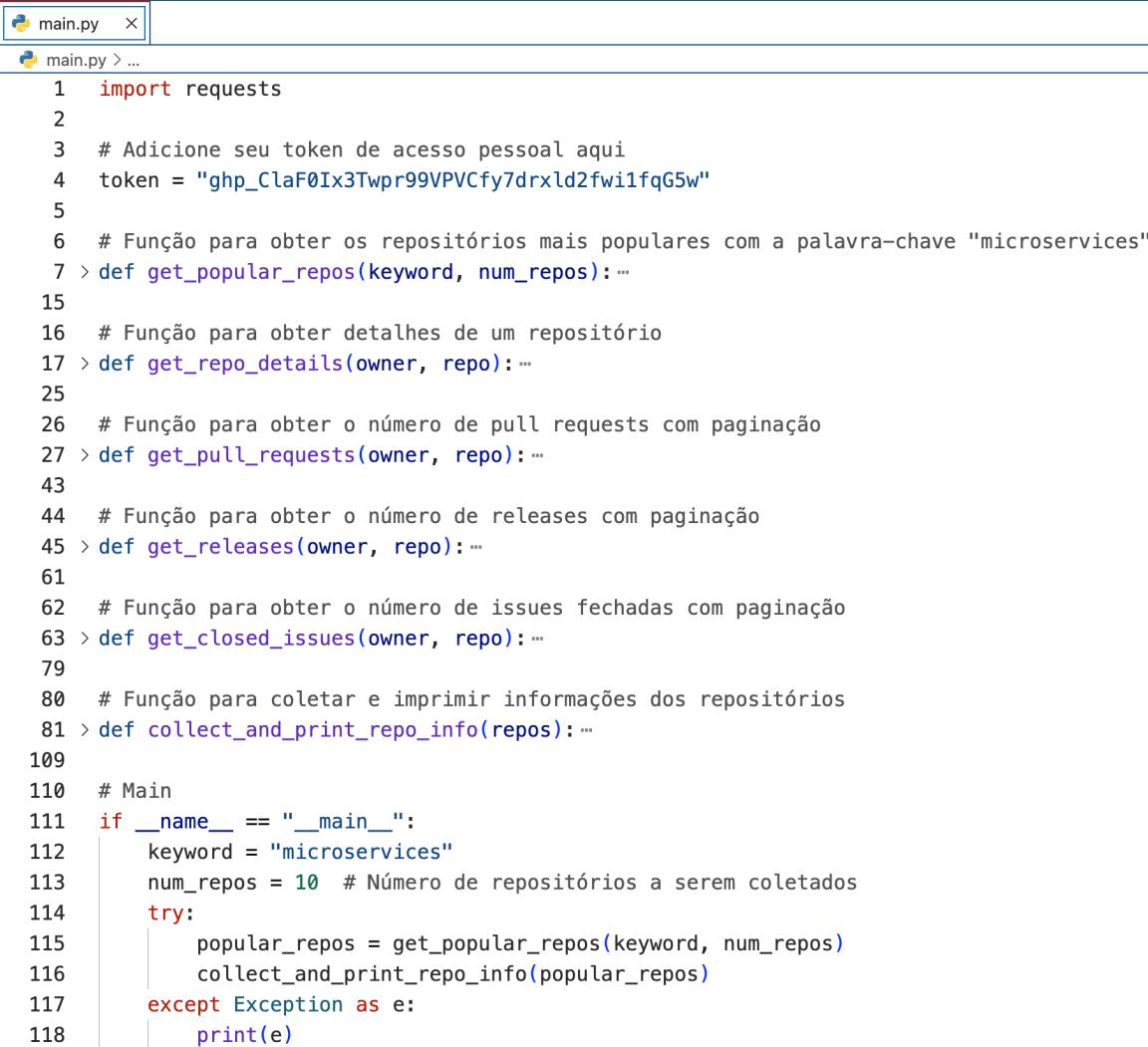
REST API endpoints for repositories

Use the REST API to create, manage and control the workflow of public and private GitHub repositories.

- [REST API endpoints for repositories](#)
 - [List organization repositories](#)
 - [Create an organization repository](#)
 - [Get a repository](#)
 - [Update a repository](#)
 - [Delete a repository](#)
 - [List repository activities](#)
 - [Check if automated security fixes are enabled for a repository](#)
 - [Enable automated security fixes](#)
 - [Disable automated security fixes](#)
 - [List CODEOWNERS errors](#)
 - [List repository contributors](#)
 - [Create a repository dispatch event](#)
 - [List repository languages](#)
 - [Check if private vulnerability reporting is enabled for a repository](#)
 - [Enable private vulnerability reporting for a repository](#)
 - [Disable private vulnerability reporting for a repository](#)
 - [List repository tags](#)
 - [List repository teams](#)

Demonstração





```
main.py > ...
  1 import requests
  2
  3 # Adicione seu token de acesso pessoal aqui
  4 token = "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"
  5
  6 # Função para obter os repositórios mais populares com a palavra-chave "microservices"
  7 > def get_popular_repos(keyword, num_repos):...
 15
 16 # Função para obter detalhes de um repositório
 17 > def get_repo_details(owner, repo):...
 25
 26 # Função para obter o número de pull requests com paginação
 27 > def get_pull_requests(owner, repo):...
 43
 44 # Função para obter o número de releases com paginação
 45 > def get_releases(owner, repo):...
 61
 62 # Função para obter o número de issues fechadas com paginação
 63 > def get_closed_issues(owner, repo):...
 79
 80 # Função para coletar e imprimir informações dos repositórios
 81 > def collect_and_print_repo_info(repos):...
109
110 # Main
111 if __name__ == "__main__":
112     keyword = "microservices"
113     num_repos = 10 # Número de repositórios a serem coletados
114     try:
115         popular_repos = get_popular_repos(keyword, num_repos)
116         collect_and_print_repo_info(popular_repos)
117     except Exception as e:
118         print(e)
```

Demonstração

Criação do código
em Python

Algoritmo de
rankeamento

...

main.py X

main.py > ...

```
1 import requests
2
3 # Adicione seu token de acesso pessoal aqui
4 token = "ghp_ClaF0Ix3Twpr99VPVCfy7drxld2fw1fqG5w"
5
6 # Função para obter os repositórios mais populares com a palavra-chave "microservices"
7 def get_popular_repos(keyword, num_repos):
8     url = f"https://api.github.com/search/repositories?q={keyword}&sort=stars&order=desc&per_page={num_repos}"
9     headers = {"Authorization": f"token {token}"}
10    response = requests.get(url, headers=headers)
11    if response.status_code == 200:
12        return response.json()["items"]
13    else:
14        raise Exception(f"Failed to fetch repositories: {response.status_code}")
15
16 # Função para obter detalhes de um repositório
17 def get_repo_details(owner, repo):
18     url = f"https://api.github.com/repos/{owner}/{repo}"
19     headers = {"Authorization": f"token {token}"}
20     response = requests.get(url, headers=headers)
21     if response.status_code == 200:
22         return response.json()
23     else:
24         raise Exception(f"Failed to fetch repository details: {response.status_code}")
```

...

```
main.py X
main.py > ...

26 # Função para obter o número de pull requests com paginação
27 def get_pull_requests(owner, repo):
28     url = f"https://api.github.com/repos/{owner}/{repo}/pulls?state=all"
29     headers = {"Authorization": f"token {token}"}
30     page = 1
31     pull_requests = []
32     while True:
33         response = requests.get(f"{url}&page={page}&per_page=100", headers=headers)
34         if response.status_code == 200:
35             page_pull_requests = response.json()
36             if not page_pull_requests:
37                 break
38             pull_requests.extend(page_pull_requests)
39             page += 1
40         else:
41             raise Exception(f"Failed to fetch pull requests: {response.status_code}")
42     return len(pull_requests)
```

...

```
main.py X
main.py > ...

44 # Função para obter o número de releases com paginação
45 def get_releases(owner, repo):
46     url = f"https://api.github.com/repos/{owner}/{repo}/releases"
47     headers = {"Authorization": f"token {token}"}
48     page = 1
49     releases = []
50     while True:
51         response = requests.get(f"{url}?page={page}&per_page=100", headers=headers)
52         if response.status_code == 200:
53             page_releases = response.json()
54             if not page_releases:
55                 break
56             releases.extend(page_releases)
57             page += 1
58         else:
59             raise Exception(f"Failed to fetch releases: {response.status_code}")
60     return len(releases)
```

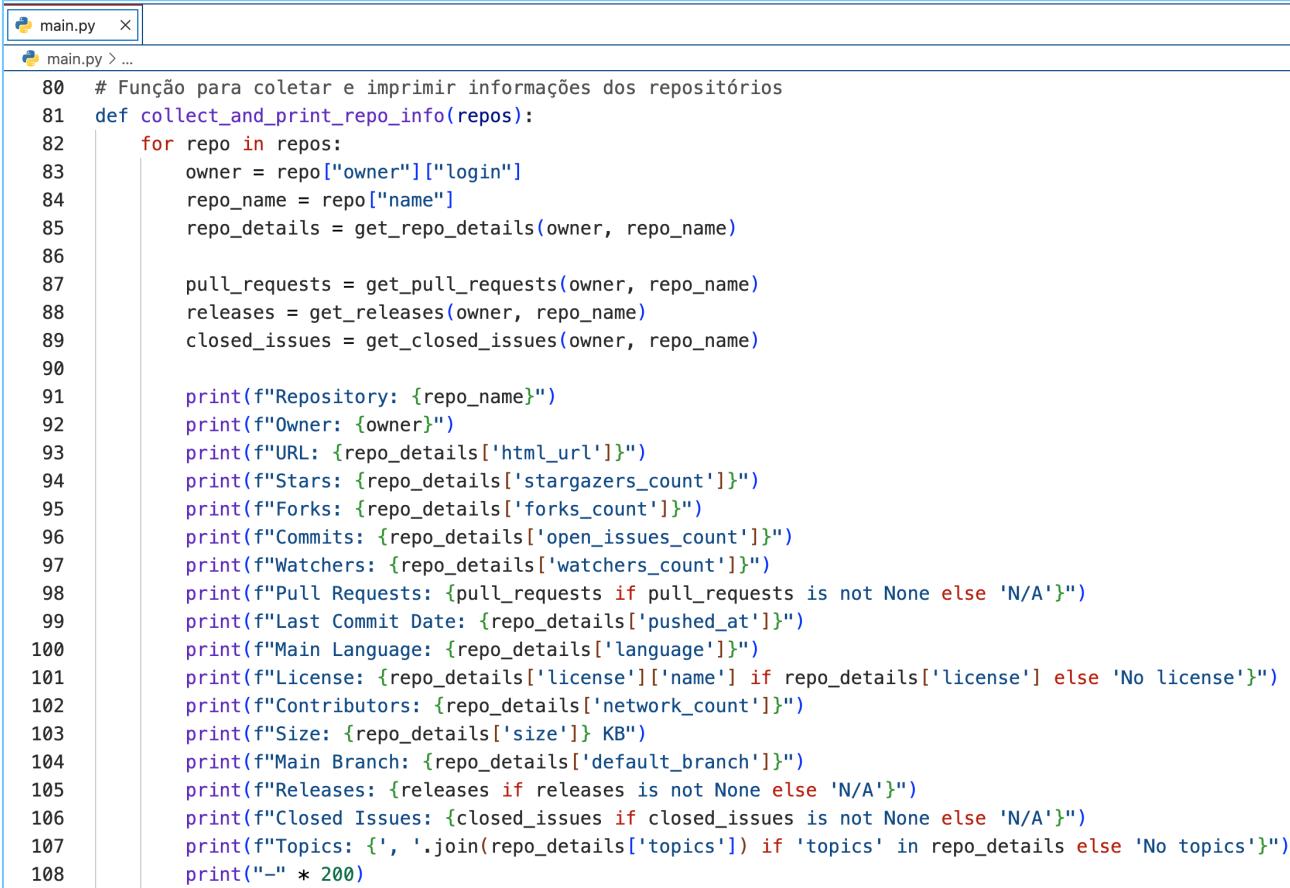
...

```
main.py X
main.py > ...

62 # Função para obter o número de issues fechadas com paginação
63 def get_closed_issues(owner, repo):
64     url = f"https://api.github.com/repos/{owner}/{repo}/issues?state=closed"
65     headers = {"Authorization": f"token {token}"}
66     page = 1
67     closed_issues = []
68     while True:
69         response = requests.get(f"{url}&page={page}&per_page=100", headers=headers)
70         if response.status_code == 200:
71             page_closed_issues = response.json()
72             if not page_closed_issues:
73                 break
74             closed_issues.extend(page_closed_issues)
75             page += 1
76         else:
77             raise Exception(f"Failed to fetch closed issues: {response.status_code}")
78     return len(closed_issues)
```

Criação do código em Python

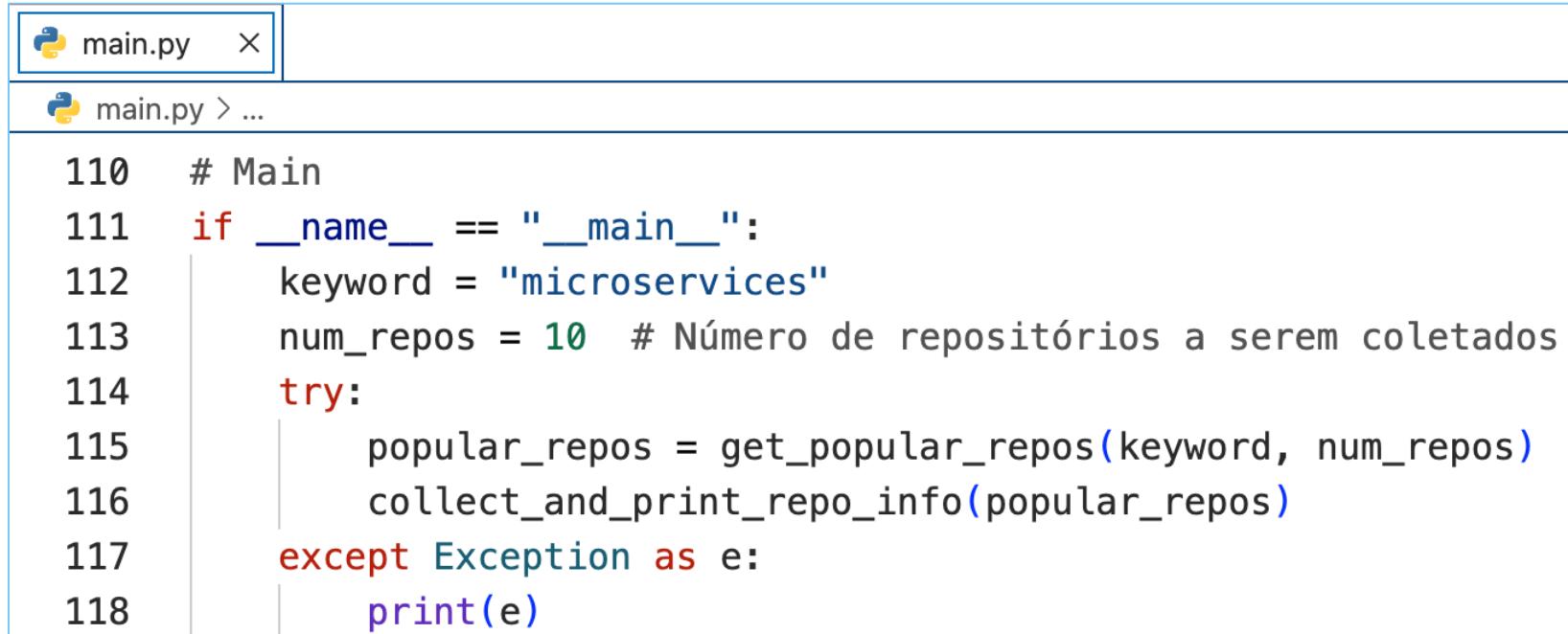
Demonstração



The screenshot shows a code editor window titled "main.py". The code is a function named "collect_and_print_repo_info" that iterates through a list of repositories. For each repository, it prints various details such as owner, URL, stargazers count, forks count, commits count, watchers count, pull requests count, last commit date, main language, license name, contributors count, size in KB, main branch, releases count, closed issues count, and topics. The code uses f-strings for printing and conditional statements to handle cases where certain information might be None.

```
80 # Função para coletar e imprimir informações dos repositórios
81 def collect_and_print_repo_info(repos):
82     for repo in repos:
83         owner = repo["owner"]["login"]
84         repo_name = repo["name"]
85         repo_details = get_repo_details(owner, repo_name)
86
87         pull_requests = get_pull_requests(owner, repo_name)
88         releases = get_releases(owner, repo_name)
89         closed_issues = get_closed_issues(owner, repo_name)
90
91         print(f"Repository: {repo_name}")
92         print(f"Owner: {owner}")
93         print(f"URL: {repo_details['html_url']}")
94         print(f"Stars: {repo_details['stargazers_count']}")
95         print(f"Forks: {repo_details['forks_count']}")
96         print(f"Commits: {repo_details['open_issues_count']}")
97         print(f"Watchers: {repo_details['watchers_count']}")
98         print(f"Pull Requests: {pull_requests if pull_requests is not None else 'N/A'}")
99         print(f"Last Commit Date: {repo_details['pushed_at']}")
100        print(f"Main Language: {repo_details['language']}")
101        print(f"License: {repo_details['license']['name'] if repo_details['license'] else 'No license'}")
102        print(f"Contributors: {repo_details['network_count']}")
103        print(f"Size: {repo_details['size']} KB")
104        print(f"Main Branch: {repo_details['default_branch']}")
105        print(f"Releases: {releases if releases is not None else 'N/A'}")
106        print(f"Closed Issues: {closed_issues if closed_issues is not None else 'N/A'}")
107        print(f"Topics: {', '.join(repo_details['topics'])} if 'topics' in repo_details else 'No topics'")
108        print("-" * 200)
```

Demonstração



The image shows a screenshot of a code editor with a light blue background. At the top, there's a header bar with a Python icon, the file name "main.py", and a close button ("X"). Below the header, the code is displayed in a text area:

```
110 # Main
111 if __name__ == "__main__":
112     keyword = "microservices"
113     num_repos = 10 # Número de repositórios a serem coletados
114     try:
115         popular_repos = get_popular_repos(keyword, num_repos)
116         collect_and_print_repo_info(popular_repos)
117     except Exception as e:
118         print(e)
```

The code is a Python script named "main.py". It contains logic to handle the main module execution, set a keyword to "microservices", define a variable for the number of repositories to collect (set to 10), and attempt to call functions to get popular repositories and collect/print their information. It includes exception handling to print any errors.

Explicação e execução do código

...

keyword = "microservices"

https://api.github.com/search/repositories?q={keyword}&sort=stars&order=desc&per_page={num_repos}

Nome do repositório: Re却tórios cujo nome contêm a palavra-chave.

Descrição do repositório: Re却tórios cuja descrição contêm a palavra-chave.

README: Re却tórios cujo README contêm a palavra-chave.

Tópicos do repositório: Re却tórios que têm tópicos (tags) associadas que contêm a palavra-chave.

Demonstração



• • •

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO

TERMINAL

PORTAS

Python

```
(.venv) joaoauloaramuni@MacBook-Pro-de-Joao Projeto GitHubAPI Python % python3 main.py
Repository: nodebestpractices
Owner: goldbergonyi
URL: https://github.com/goldbergonyi/nodebestpractices
Stars: 97090
Forks: 9864
Commits: 72
Watchers: 97090
Pull Requests: 955
Last Commit Date: 2024-06-20T13:00:09Z
Main Language: Dockerfile
License: Creative Commons Attribution Share Alike 4.0 International
Contributors: 9864
Size: 69818 KB
Main Branch: master
Releases: 0
Closed Issues: 1202
Topics: best-practices, es6, eslint, express, expressjs, javascript, jest, microservices, mocha, node-js, nodejs, nodejs-development, npm, rest, style-guide, styleguide, testing, types
```

```
Repository: nest
Owner: nestjs
URL: https://github.com/nestjs/nest
Stars: 65393
Forks: 7447
Commits: 117
Watchers: 65393
Pull Requests: 8085
Last Commit Date: 2024-06-24T01:01:09Z
Main Language: TypeScript
License: MIT License
Contributors: 7447
Size: 312829 KB
Main Branch: master
Releases: 96
Closed Issues: 13262
Topics: framework, hacktoberfest, javascript, javascript-framework, microservices, nest, nestjs, node, nodejs, nodejs-framework, typescript, typescript-framework, websockets
```

```
Repository: dubbo
Owner: apache
URL: https://github.com/apache/dubbo
Stars: 40191
Forks: 26331
Commits: 819
Watchers: 40191
Pull Requests: 7220
Last Commit Date: 2024-06-24T09:54:52Z
Main Language: Java
License: Apache License 2.0
Contributors: 26331
Size: 55366 KB
Main Branch: 3.2
Releases: 100
Closed Issues: 13306
Topics: distributed-systems, dubbo, framework, grpc, http, java, microservices, restful, rpc, service-mesh, web
```

Repository: kong
Owner: Kong
URL: <https://github.com/Kong/kong>
Stars: 38213
Forks: 4747
Commits: 138
Watchers: 38213
Pull Requests: 7828
Last Commit Date: 2024-06-24T17:22:18Z
Main Language: Lua
License: Apache License 2.0
Contributors: 4747
Size: 97515 KB
Main Branch: master
Releases: 141
Closed Issues: 12038
Topics: ai, ai-gateway, api-gateway, api-management, apis, artificial-intelligence, cloud-native, consul, devops, docker, kong, kubernetes, kubernetes-ingress, kubernetes-ingress-controller, luajit, microservice, microservices, nginx, reverse-proxy, serverless

Repository: awesome-design-patterns
Owner: DovAmir
URL: <https://github.com/DovAmir/awesome-design-patterns>
Stars: 37357
Forks: 2752
Commits: 17
Watchers: 37357
Pull Requests: 51
Last Commit Date: 2024-06-11T05:34:10Z
Main Language: None
License: No license
Contributors: 2752
Size: 90 KB
Main Branch: master
Releases: 0
Closed Issues: 47
Topics: architecture, awesome, awesome-list, cloud-computing, design-patterns, gof-patterns, lists, microservices, resources

Repository: istio
Owner: istio
URL: <https://github.com/istio/istio>
Stars: 35312
Forks: 7618
Commits: 613
Watchers: 35312
Pull Requests: 31709
Last Commit Date: 2024-06-24T20:49:29Z
Main Language: Go
License: Apache License 2.0
Contributors: 7618
Size: 266190 KB
Main Branch: master
Releases: 346
Closed Issues: 50574
Topics: api-management, circuit-breaker, consul, enforce-policies, envoy, fault-injection, kubernetes, lyft-envoy, microservice, microservices, nomad, polyglot-microservices, proxies, request-routing, resiliency, service-mesh

Explicação e execução do código

Demonstração

• • •

```
Repository: system-design
Owner: karanpratapsingh
URL: https://github.com/karanpratapsingh/system-design
Stars: 30105
Forks: 3095
Commits: 2
Watchers: 30105
Pull Requests: 22
Last Commit Date: 2024-04-10T15:00:47Z
Main Language: None
License: Other
Contributors: 3095
Size: 5672 KB
Main Branch: main
Releases: 0
Closed Issues: 40
Topics: architecture, distributed-systems, engineering, interview, interview-preparation, microservices, scalability, system-design, system-design-interview, tech
```

```
Repository: nacos
Owner: alibaba
URL: https://github.com/alibaba/nacos
Stars: 29396
Forks: 12650
Commits: 244
Watchers: 29396
Pull Requests: 3842
Last Commit Date: 2024-06-24T11:17:21Z
Main Language: Java
License: Apache License 2.0
Contributors: 12650
Size: 56612 KB
Main Branch: develop
Releases: 64
Closed Issues: 11885
Topics: alibaba, config, configuration-management, distributed-configuration, dns, dubbo, istio, kubernetes, microservices, nacos, service-discovery, service-mesh, spring-cloud
```

```
Repository: apollo
Owner: apolloconfig
URL: https://github.com/apolloconfig/apollo
Stars: 28908
Forks: 10186
Commits: 139
Watchers: 28908
Pull Requests: 1680
Last Commit Date: 2024-06-16T06:14:59Z
Main Language: Java
License: Apache License 2.0
Contributors: 10186
Size: 53731 KB
Main Branch: master
Releases: 39
Closed Issues: 4737
Topics: config-management, configuration-management, distributed-configuration, microservices, spring-boot, spring-cloud
```

Repository: go-zero
Owner: zeromicro
URL: <https://github.com/zeromicro/go-zero>
Stars: 28063
Forks: 3826
Commits: 353
Watchers: 28063
Pull Requests: 2558
Last Commit Date: 2024-06-23T03:47:17Z
Main Language: Go
License: MIT License
Contributors: 3826
Size: 20796 KB
Main Branch: master
Releases: 109
Closed Issues: 3777
Topics: cloud-native, code-generation, framework, gateway, go, go-zero, goctl, golang, gozero, grpc-gateway, microservice, microservice-framework, microservices, microservices-architecture, rest-api, restful, restful-api, rpc, rpc-framework, web-framework

○ (.venv) joaopauloaramunha@MacBook-Pro-de-Joao: Projeto GitHubAPI Python % █

Tempo médio de execução do script:

30 minutos devido a paginação.

Demonstração

Análise dos resultados

...

Top 10 repositórios com microsserviços ordenados por Stars (resultado bruto):

- 1 - <https://github.com/goldbergonyi/nodebestpractices>
- 2 - <https://github.com/nestjs/nest>
- 3 - <https://github.com/apache/dubbo>
- 4 - <https://github.com/Kong/kong>
- 5 - <https://github.com/DovAmir/awesome-design-patterns>
- 6 - <https://github.com/istio/istio>
- 7 - <https://github.com/karanpratapsingh/system-design>
- 8 - <https://github.com/alibaba/nacos>
- 9 - <https://github.com/apolloconfig/apollo>
- 10 - <https://github.com/zeromicro/go-zero>

Demonstração



Análise dos resultados

...

Vamos considerar repositórios mais populares como:

Stars + Forks

Stars (Estrelas): As estrelas em um repositório do GitHub são uma forma de os usuários marcarem um projeto como interessante ou útil. Quando alguém clica na estrela de um repositório, está expressando que gosta do projeto ou que deseja acompanhar suas atualizações. É um indicativo de popularidade e interesse da comunidade no projeto.

Forks (Bifurcações) : Os forks são cópias de um repositório que um usuário cria em sua própria conta no GitHub. Fazer um fork permite ao usuário experimentar mudanças sem afetar o repositório original.

Demonstração



Análise dos resultados

...

Vamos considerar repositórios mais populares como:

Stars + Forks

Para realizar uma análise mais completa que considere tanto o número de estrelas (stars) quanto o número de forks, podemos aplicar um teste estatístico que combine ambas as variáveis.

Podemos usar uma pontuação **composta** que pondera ambos os fatores.

Demonstração



Análise dos resultados

...

Vamos considerar repositórios mais populares como:

Stars + Forks

- 1) Normalização dos dados: Primeiro, normalizamos os dados para que as duas variáveis estejam na mesma escala.
- 2) Cálculo da pontuação composta: Em seguida, calculamos uma pontuação composta para cada repositório.
- 3) Ordenação dos repositórios: Finalmente, ordenamos os repositórios com base na pontuação composta.

Demonstração



Análise dos resultados

...

Vamos considerar repositórios mais populares como:

Stars + Forks

- 1) Normalização dos dados: A normalização pode ser feita utilizando a técnica de min-max scaling:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- onde X' é o valor normalizado, X é o valor original, X_{min} é o valor mínimo do conjunto de dados e X_{max} é o valor máximo do conjunto de dados.

Demonstração



Análise dos resultados

...

Vamos considerar repositórios mais populares como:

Stars + Forks

- 1) Normalização dos dados: A normalização pode ser feita utilizando a técnica de min-max scaling:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

```
# Normalização
stars_normalized = (stars - stars.min()) / (stars.max() - stars.min())
forks_normalized = (forks - forks.min()) / (forks.max() - forks.min())
```

Demonstração



Análise dos resultados

...

Vamos considerar repositórios mais populares como:

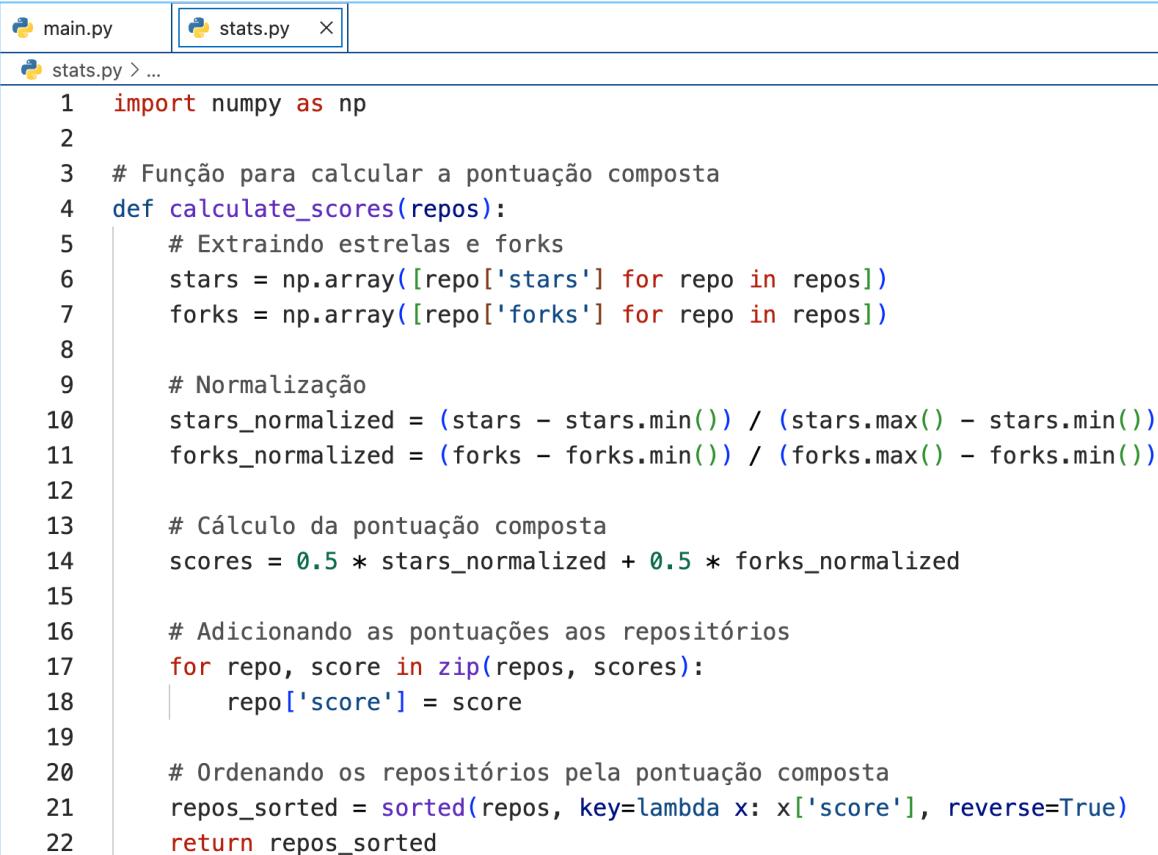
Stars + Forks

- 2) Cálculo da pontuação composta:
- Vamos atribuir pesos iguais para ambas as variáveis (estrelas e forks) para simplificar:
- Combinação linear ponderada:
$$\text{scores} = 0.5 \times \text{normalized_stars} + 0.5 \times \text{normalized_forks}$$
- 3) Ordenação dos repositórios: Os repositórios serão ordenados de acordo com a pontuação composta.

Demonstração



stats.py



```
main.py stats.py X
stats.py > ...

1 import numpy as np
2
3 # Função para calcular a pontuação composta
4 def calculate_scores(repos):
5     # Extrairando estrelas e forks
6     stars = np.array([repo['stars'] for repo in repos])
7     forks = np.array([repo['forks'] for repo in repos])
8
9     # Normalização
10    stars_normalized = (stars - stars.min()) / (stars.max() - stars.min())
11    forks_normalized = (forks - forks.min()) / (forks.max() - forks.min())
12
13    # Cálculo da pontuação composta
14    scores = 0.5 * stars_normalized + 0.5 * forks_normalized
15
16    # Adicionando as pontuações aos repositórios
17    for repo, score in zip(repos, scores):
18        repo['score'] = score
19
20    # Ordenando os repositórios pela pontuação composta
21    repos_sorted = sorted(repos, key=lambda x: x['score'], reverse=True)
22
23    return repos_sorted
```

```
...  
main.py stats.py ...  
stats.py > ...  
24 # Main  
25 if __name__ == "__main__":  
26     # Dados dos repositórios  
27     repos = [  
28         {"name": "nodebestpractices", "stars": 97090, "forks": 9864},  
29         {"name": "nest", "stars": 65393, "forks": 7447},  
30         {"name": "dubbo", "stars": 40191, "forks": 26331},  
31         {"name": "kong", "stars": 38213, "forks": 4747},  
32         {"name": "awesome-design-patterns", "stars": 37357, "forks": 2752},  
33         {"name": "istio", "stars": 35312, "forks": 7618},  
34         {"name": "system-design", "stars": 30105, "forks": 3095},  
35         {"name": "nacos", "stars": 29396, "forks": 12650},  
36         {"name": "apollo", "stars": 28908, "forks": 10186},  
37         {"name": "go-zero", "stars": 28063, "forks": 3826},  
38     ]  
39  
40     # Calculando as pontuações e ordenando os repositórios  
41     repos_sorted = calculate_scores(repos)  
42  
43     # Exibindo os resultados  
44     for repo in repos_sorted:  
45         print(f"Repository: {repo['name']}, Stars: {repo['stars']}, Forks: {repo['forks']}, Score: {repo['score']:.4f}")
```

stats.py

...

Executando `python3 stats.py`:

Tempo de execução: 1s

PROBLEMAS	SAÍDA	CONSOLE DE DEPURAÇÃO	TERMINAL	PORAS

```
● (.venv) joaopauloaramuni@MacBook-Pro-de-Joao Projeto GitHubAPI Python % python3 stats.py
Repository: nodebestpractices, Stars: 97090, Forks: 9864, Score: 0.6508
Repository: dubbo, Stars: 40191, Forks: 26331, Score: 0.5878
Repository: nest, Stars: 65393, Forks: 7447, Score: 0.3700
Repository: nacos, Stars: 29396, Forks: 12650, Score: 0.2195
Repository: apollo, Stars: 28908, Forks: 10186, Score: 0.1638
Repository: istio, Stars: 35312, Forks: 7618, Score: 0.1557
Repository: kong, Stars: 38213, Forks: 4747, Score: 0.1158
Repository: awesome-design-patterns, Stars: 37357, Forks: 2752, Score: 0.0673
Repository: go-zero, Stars: 28063, Forks: 3826, Score: 0.0228
Repository: system-design, Stars: 30105, Forks: 3095, Score: 0.0221
○ (.venv) joaopauloaramuni@MacBook-Pro-de-Joao Projeto GitHubAPI Python % █
```



Análise dos resultados

...

Novo Top 10 repositórios com microsserviços considerando stars e forks (**scores**):

Posição	Repository	Stars	Forks	Score	Novo ranking	
1	nodebestpractices	97090	9864	0.6508	1	
2	dubbo	40191	26331	0.5878	3->2	↑
3	nest	65393	7447	0.3700	2->3	↓
4	nacos	29396	12650	0.2195	8->4	↑
5	apollo	28908	10186	0.1638	9->5	↑
6	istio	35312	7618	0.1557	6	
7	kong	38213	4747	0.1158	4->7	↓
8	awesome-design-patterns	37357	2752	0.0673	5->8	↓
9	go-zero	28063	3826	0.0228	10->9	↑
10	system-design	30105	3095	0.0221	7->10	↓

Criação de um relatório final sobre o experimento

...

Relatório técnico: Análise de repositórios populares que utilizam microsserviços no GitHub.

Objetivo: O objetivo deste experimento é coletar e analisar características de repositórios populares no GitHub que implementam a arquitetura de microsserviços. Os principais parâmetros analisados incluem estrelas, forks, commits, watchers, pull requests, data do último commit, linguagem principal, licença, contribuidores, tamanho, ramo principal, releases, issues fechadas e tópicos.

Linguagem de programação: Python 3 foi a linguagem escolhida para desenvolver o experimento.

Dependências: Para executar este experimento, as seguintes dependências foram utilizadas:

- `requests` (para fazer requisições HTTP à API do GitHub)
- `numpy` (para realizar operações matemáticas, como normalização)

API utilizada: Foi utilizada a GitHub API para coletar os dados necessários dos repositórios. Mais detalhes sobre a API podem ser encontrados na documentação do GitHub REST API.

- <https://docs.github.com/pt/rest?apiVersion=2022-11-28>

Criação de um relatório final sobre o experimento

...

Metodologia:

- Coleta de dados: Foi utilizada a GitHub API para obter informações detalhadas dos repositórios que incluem a palavra-chave "microservices" em seus tópicos ou descrições. Como restrição do experimento, a coleta não inclui repositórios que implementam microserviços, mas que não mencionam explicitamente a palavra-chave microservices em nenhum lugar.
- Filtragem e paginação: A coleta de dados envolveu a utilização da paginação da API do GitHub devido ao grande número de informações nos repositórios mais relevantes. Esse processo pode levar em média 30 minutos para terminar devido à necessidade de lidar com múltiplas requisições e alto volume de dados.
- Normalização de dados: Os dados foram normalizados usando a técnica de min-max scaling, garantindo que todos os valores estivessem na mesma escala para o cálculo da pontuação composta.
- Cálculo da pontuação composta: Foi utilizado um método de combinação linear ponderada (scores) para calcular uma pontuação composta para cada repositório, levando em consideração tanto as estrelas quanto os forks.
- Ordenação dos repositórios: Os repositórios foram ordenados em ordem decrescente com base na pontuação composta calculada.

Criação de um relatório final sobre o experimento

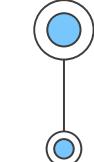
...

Resultados:

Considerando estrelas e forks, os seguintes repositórios foram ranqueados como os mais populares que utilizam a arquitetura de microserviços: 1) nodebestpractices; 2) dubbo; 3) nest; 4) nacos; 5) apollo; 6) istio; 7) kong; 8) awesome-design-patterns; 9) go-zero; 10) system-design.

Conclusão:

Este experimento demonstrou a viabilidade de coletar e analisar dados de repositórios populares no GitHub que implementam microserviços. A metodologia utilizada permitiu identificar e ranquear os principais repositórios com base em critérios objetivos como estrelas e forks, apesar da limitação de apenas considerar repositórios que explicitamente mencionam "microservices" em seus metadados. O algoritmo de ranqueamento leva em média 30 minutos para rodar devido à necessidade de lidar com a paginação da API do GitHub, enquanto o algoritmo de cálculos estatísticos é instantâneo, pois os repositórios são fixos no código.

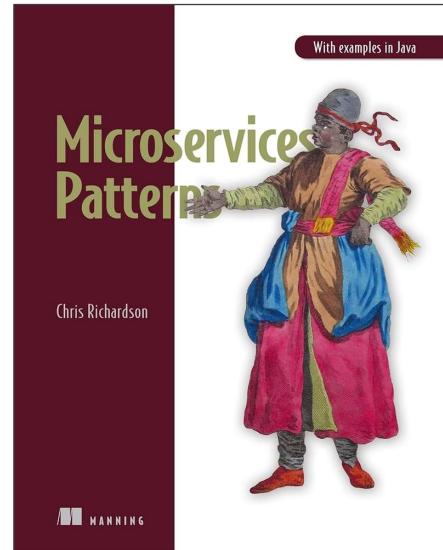
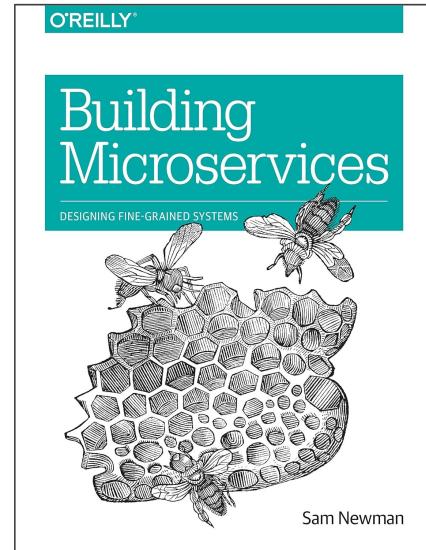


...

Prova didática



Referências básicas:



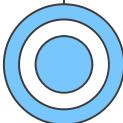
...



...

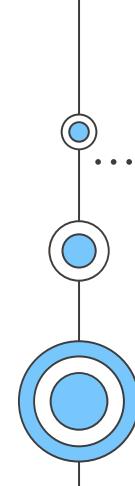
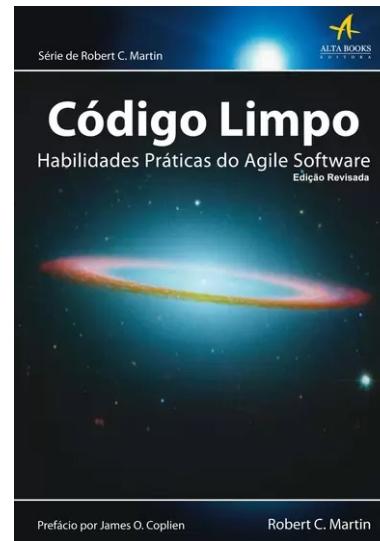
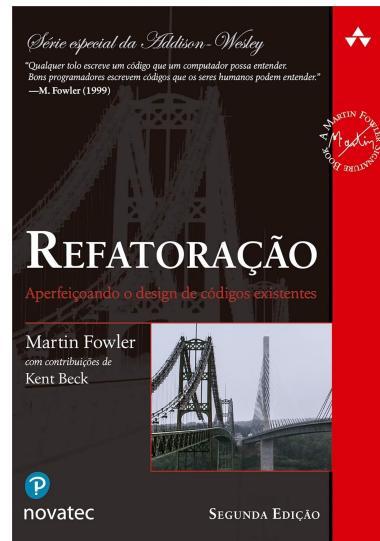
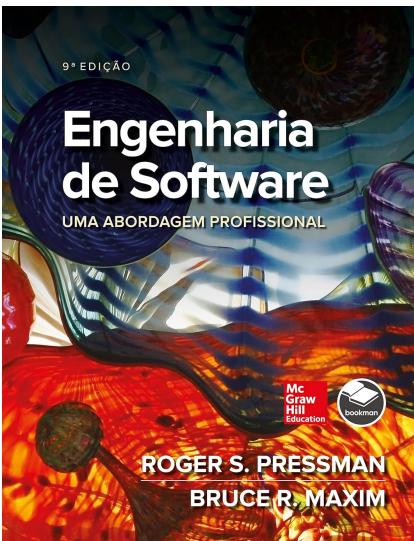
...





Prova didática

Outras referências:



Obrigado!

Dúvidas?

joaopauloaramuni@gmail.com



[GitHub](#)



[LinkedIn](#)



[Lattes](#)

...



...