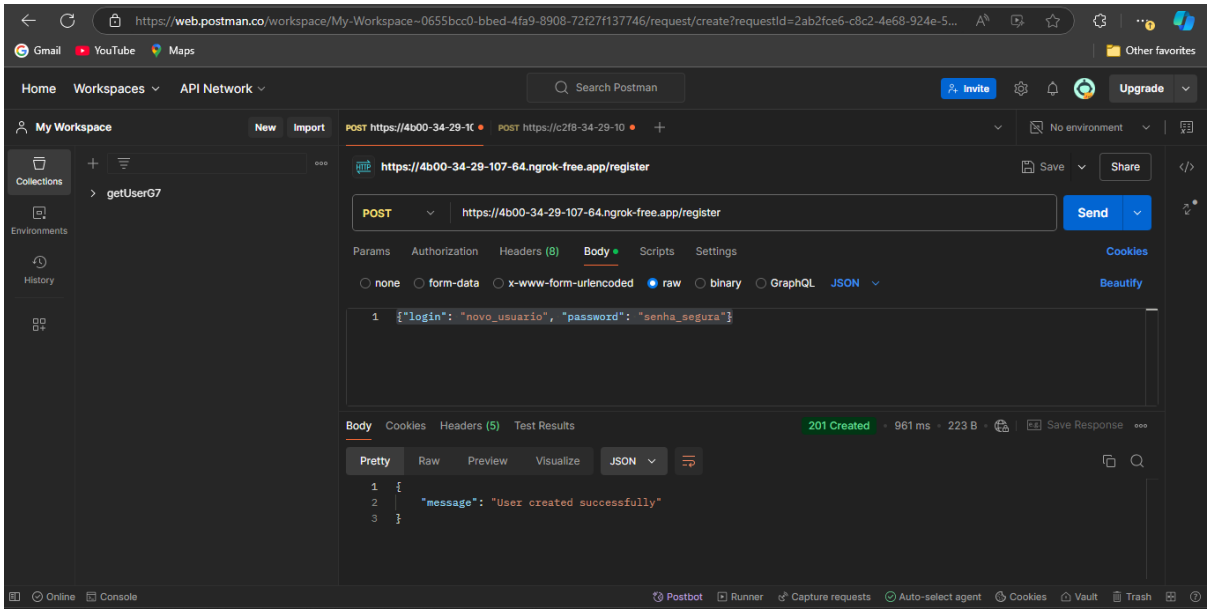
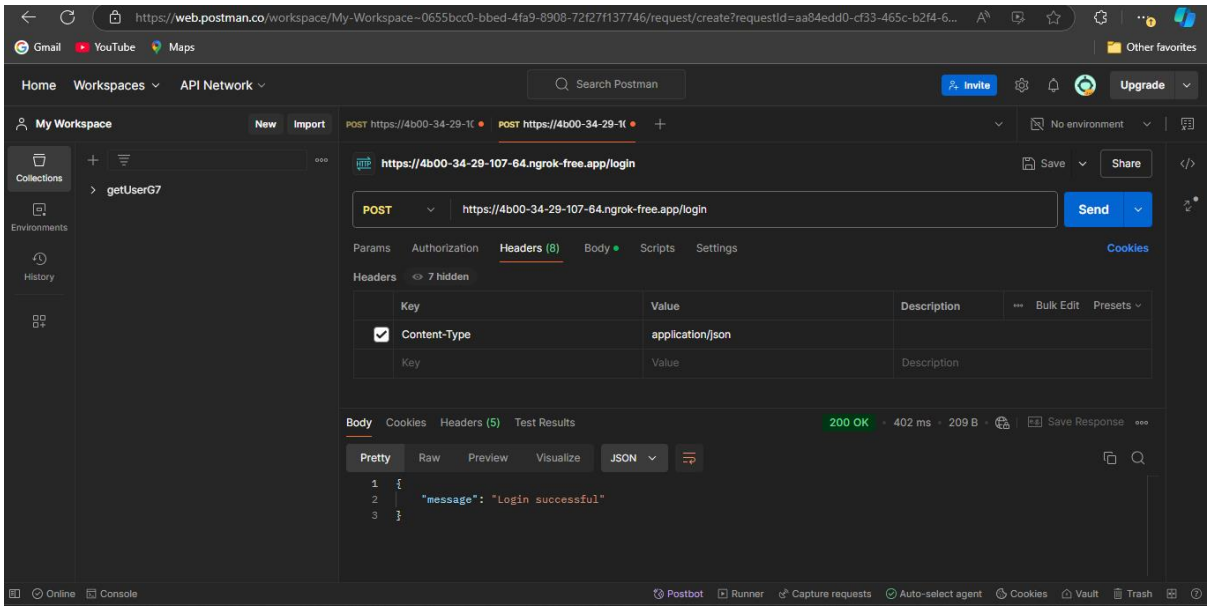
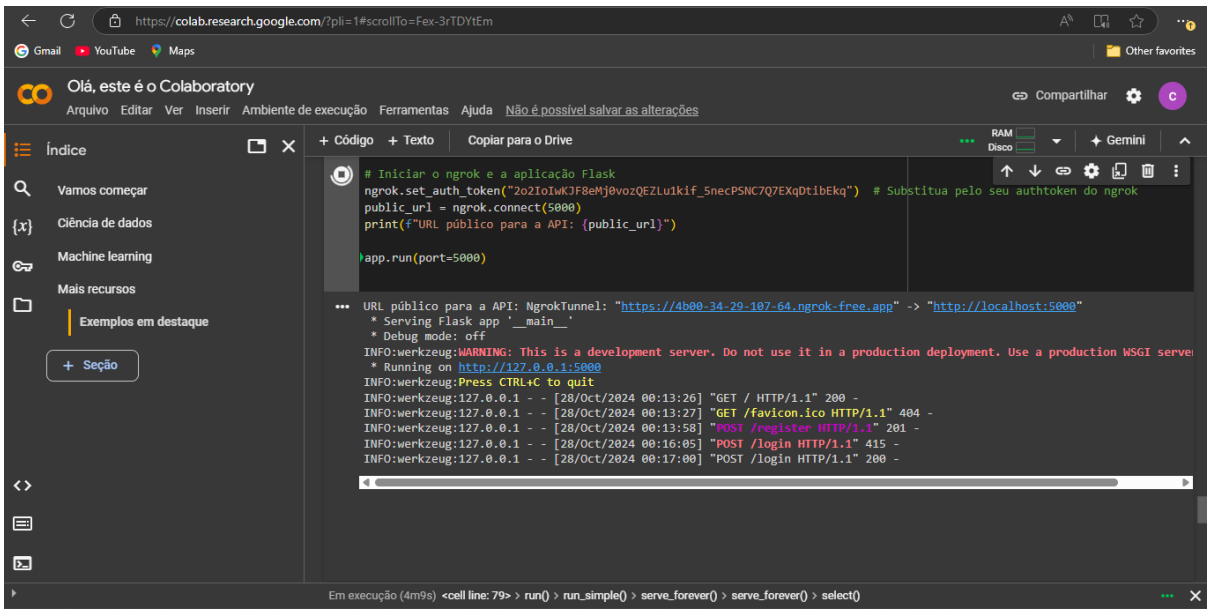
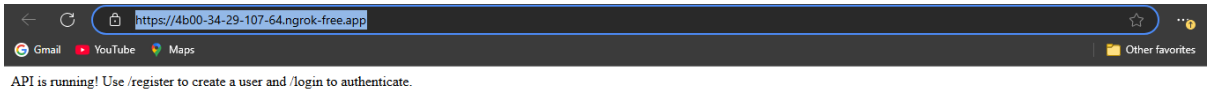
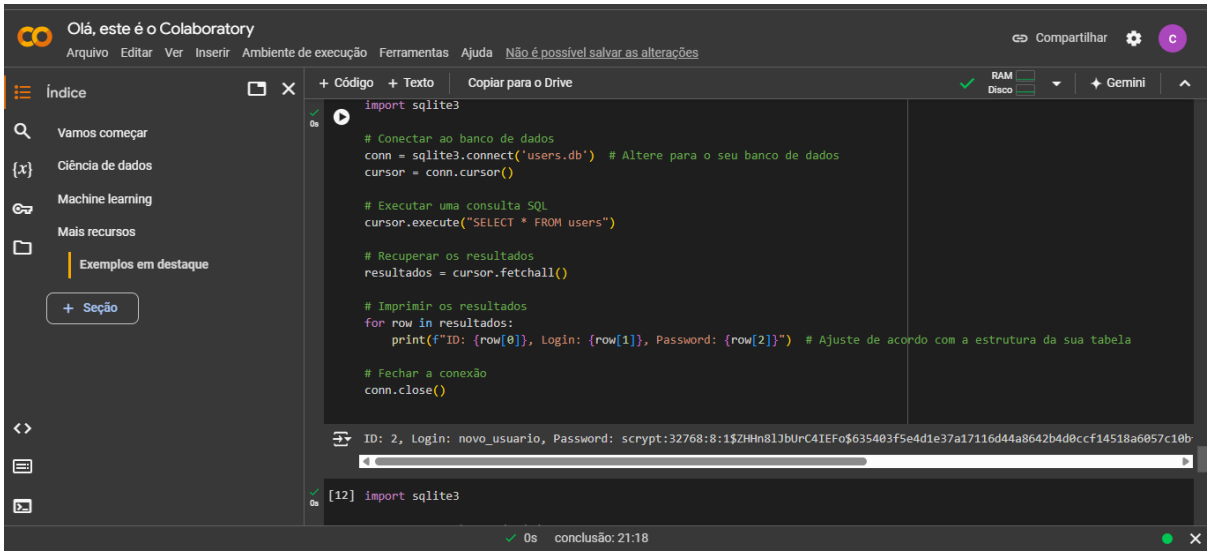

https://web.postman.co/workspace/My-Workspace~0655bcc0-bbed-4fa9-8908-72f27f137746/request/create?requestId=2ab2fce6-c8c2-4e68-924e-5...
Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
No environment
Collections
Environments
History
getUserG7
https://4b00-34-29-107-64.ngrok-free.app/register
Save
Share
POST
https://4b00-34-29-107-64.ngrok-free.app/register
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{"login": "novo_usuario", "password": "senha_segura"}
Body
Cookies
Headers (5)
Test Results
201 Created
961 ms
223 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"message": "User created successfully"
}
Online
Console
Postbot
Runner
Capture requests
Auto-select agent
Cookies
Vault
Trash


https://web.postman.co/workspace/My-Workspace~0655bcc0-bbed-4fa9-8908-72f27f137746/request/create?requestId=aa84edd0-cf33-465c-b2f4-6...
Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
No environment
Collections
Environments
History
getUserG7
https://4b00-34-29-107-64.ngrok-free.app/login
Save
Share
POST
https://4b00-34-29-107-64.ngrok-free.app/login
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
Headers
7 hidden
Key
Value
Description
Bulk Edit
Presets
Content-Type
application/json
Body
Cookies
Headers (5)
Test Results
200 OK
402 ms
209 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"message": "Login successful"
}
Online
Console
Postbot
Runner
Capture requests
Auto-select agent
Cookies
Vault
Trash


https://colab.research.google.com/?pli=1#scrollTo=Fex-3rTDYtEm
Olá, este é o Colaboratory
Arquivo
Editar
Ver
Inserir
Ambiente de execução
Ferramentas
Ajuda
Não é possível salvar as alterações
Compartilhar
Índice
+ Código
+ Texto
Copiar para o Drive
RAM
Disco
Gemini
Vamos começar
Ciência de dados
Machine learning
Mais recursos
Exemplos em destaque
+ Seção
# Iniciar o ngrok e a aplicação Flask
ngrok.set_auth_token("2o2IoIwKJF8eMj0vozQEZLu1kif_5necPSNC7Q7EXqDtibEkq") # Substitua pelo seu authtoken do ngrok
public_url = ngrok.connect(5000)
print(f"URL público para a API: {public_url}")
app.run(port=5000)
URL público para a API: NgrokTunnel: "https://4b00-34-29-107-64.ngrok-free.app" -> "http://localhost:5000"
* Serving Flask app '__main__'
* Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug:127.0.0.1 - - [28/Oct/2024 00:13:26] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [28/Oct/2024 00:13:27] "GET /favicon.ico HTTP/1.1" 404 -
INFO:werkzeug:127.0.0.1 - - [28/Oct/2024 00:13:58] "POST /register HTTP/1.1" 201 -
INFO:werkzeug:127.0.0.1 - - [28/Oct/2024 00:16:05] "POST /login HTTP/1.1" 415 -
INFO:werkzeug:127.0.0.1 - - [28/Oct/2024 00:17:00] "POST /login HTTP/1.1" 200 -
Em execução (4m9s)
<cell line: 79> > run() > run_simple() > serve_forever() > serve_forever() > select()





ID: 2, Login: novo_usuario, Password: scrypt:32768:8:1$ZHHn8lJbUrC4IEFo$635403f5e4d1e37a17116d44a8642b4d0ccf14518a6057c10bf721722662d959597c3cd2802a6aa7b55a5028967af968f34186fd826958f1b656ecf6b64130c2

```
from flask import Flask, request, jsonify
from werkzeug.security import generate_password_hash, check_password_hash
from pyngrok import ngrok
import sqlite3

app = Flask(__name__)

# Conectando ao banco de dados SQLite
def init_db():
    conn = sqlite3.connect('users.db')
```

```
    c = conn.cursor()
    c.execute('''
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            login TEXT NOT NULL UNIQUE,
            password TEXT NOT NULL
        )
    ''')
    conn.commit()
    conn.close()

# Rota para a homepage
@app.route('/')
def home():
    return "API is running! Use /register to create a user and /login to
authenticate."

# Rota para criar usuários
@app.route('/register', methods=['POST'])
def register():
    data = request.get_json()
    login = data.get('login')
    password = data.get('password')

    if not login or not password:
        return jsonify({"message": "Login and password are required"}), 400

    password_hash = generate_password_hash(password)

    conn = sqlite3.connect('users.db')
    c = conn.cursor()
    try:
        c.execute("INSERT INTO users (login, password) VALUES (?, ?)", (login,
password_hash))
        conn.commit()
        return jsonify({"message": "User created successfully"}), 201
    except sqlite3.IntegrityError:
        return jsonify({"message": "User already exists"}), 400
    finally:
        conn.close()

# Rota para login
@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    login = data.get('login')
```

```
    password = data.get('password')

    if not login or not password:
        return jsonify({"message": "Login and password are required"}), 400

    conn = sqlite3.connect('users.db')
    c = conn.cursor()
    c.execute("SELECT password FROM users WHERE login = ?", (login,))
    row = c.fetchone()
    conn.close()

    if row and check_password_hash(row[0], password):
        return jsonify({"message": "Login successful"}), 200
    else:
        return jsonify({"message": "Invalid login or password"}), 401

# Inicializar o banco de dados
init_db()

# Iniciar o ngrok e a aplicação Flask
ngrok.set_auth_token("2o2IoIwKJF8eMj0vozQEZLu1kif_5necPSNC7Q7EXqDtibEkq")  #
Substitua pelo seu authtoken do ngrok
public_url = ngrok.connect(5000)
print(f"URL público para a API: {public_url}")

app.run(port=5000)
```