**Trabalho Prático: Construção de uma API REST Utilizando a Biblioteca Automata**

**Disciplina:** Teoria da Computação
**Valor do Trabalho:** 30 pontos
**Objetivo:** Aplicar conceitos teóricos de autômatos em uma implementação prática, criando uma API REST que manipula autômatos utilizando a biblioteca Automata e o framework FastAPI em Python.

---

**Introdução:** O estudo da teoria dos autômatos envolve a compreensão de máquinas abstratas e dos problemas computacionais que elas podem resolver. Neste trabalho, vocês irão desenvolver uma API REST para manipular e analisar autômatos (autômatos finitos, autômatos com pilha ou máquinas de Turing) utilizando a biblioteca **Automata (https://github.com/caleb531/automata)** em Python. Este trabalho foi concebido para conectar os conceitos teóricos às aplicações práticas em ciência da computação.

A biblioteca **Automata** oferece algoritmos otimizados para processar grandes entradas, além de recursos de visualização. Combinando essa biblioteca com o **FastAPI (https://fastapi.tiangolo.com/)**, um framework moderno e de alta performance para aplicações web, vocês criarão um sistema funcional e interativo para explorar autômatos.

---

**Requisitos do Projeto:**

1. **Desenvolvimento da API:**
    - Desenvolver uma API RESTful utilizando Python e FastAPI.
    - A API deve expor endpoints para:
        - Criar autômatos (autômatos finitos, autômatos com pilha ou máquinas de Turing).
        - Recuperar informações sobre os autômatos (ex.: estados, transições, estados de aceitação).
        - Testar a aceitação de strings pelos autômatos.
        - Visualizar os autômatos em um formato gráfico (ex.: PNG ou SVG).
2. **Funcionalidades dos Autômatos:**
    - O sistema deve implementar três endpoints separados, um para cada tipo de autômato:
        - **Autômatos Finitos Determinísticos (AFD):** Permitir a criação, manipulação e visualização de AFDs, com suporte para grandes entradas.
        - **Autômatos com Pilha:** Implementar funcionalidades para criar e visualizar autômatos que utilizam pilha, possibilitando operações mais complexas.
        - **Máquinas de Turing:** Fornecer suporte para criar e representar graficamente máquinas de Turing, explorando sua capacidade de resolver problemas computacionais gerais.
    - Cada endpoint deve:

        - Receber dados para criar ou configurar o autômato correspondente.
        - Retornar informações detalhadas sobre o autômato criado.
        - Permitir a visualização gráfica do autômato.
3. **Frontend:** (Opcional, para pontos extras)
    - Os estudantes podem criar uma interface simples utilizando qualquer tecnologia para interagir com a API.
4. **Documentação:**
    - Fornecer documentação clara da API utilizando os recursos automáticos do FastAPI (Swagger UI ou Redoc).
    - Incluir um arquivo README.md descrevendo:
        - Como configurar e executar o projeto.
        - Exemplos de uso da API.
        - Limitações e pressupostos.

---

**Etapas de Implementação:**

1. **Configuração Inicial:**
    - Instalar o Python e configurar um ambiente virtual.
    - Instalar o FastAPI, Uvicorn (para executar a API) e a biblioteca Automata.
    - Preparar a estrutura do projeto.
2. **Noções Básicas sobre Autômatos:**
    - Durante uma aula presencial, o professor apresentará:
        - As principais funcionalidades e métodos da biblioteca Automata.
        - Conceitos teóricos de autômatos relevantes para o projeto.
    - Serão fornecidos exemplos de códigos para utilização da biblioteca.
3. **Design da API:**
    - Definir os endpoints e os requisitos de entrada/saída correspondentes.
    - Planejar como os dados relacionados aos autômatos serão armazenados e manipulados (ex.: em memória, arquivos JSON ou um banco de dados).
4. **Desenvolvimento:**
    - Implementar a funcionalidade central da API, garantindo modularidade e qualidade do código.
    - Utilizar tratamento de erros e logs apropriados para garantir a robustez.
5. **Testes:**
    - Testar os endpoints da API com entradas de exemplo.
    - Validar a funcionalidade das operações com autômatos utilizando casos de teste teóricos.
6. **Requisitos para Submissão:**
    - Hospedar o projeto no GitHub ou plataforma semelhante.
    - Enviar o link do repositório pelo sistema de submissão da disciplina.
    - O repositório deve incluir:
        - O código completo da implementação.
        - O arquivo README.md com instruções de configuração.
        - Exemplos de uso da API.

---

https://github.com/caleb531/automata

**Critérios de Avaliação:**

- **Funcionalidade da API (15 pontos):**
    - Implementação correta dos endpoints principais e funcionalidades dos autômatos.
    - Adesão aos princípios RESTful.
- **Qualidade e Modularidade do Código (5 pontos):**
    - Código legível e manutenível.
    - Uso adequado do FastAPI e da biblioteca Automata.
- **Documentação (5 pontos):**
    - Documentação clara e completa da API.
    - Qualidade do arquivo README.md.
- **Testes e Validação (5 pontos):**
    - Testes abrangentes dos endpoints da API e das funcionalidades dos autômatos.
    - Resultados válidos para os casos de teste fornecidos.

---

**Pontos Extras (Opcional):**

- **Integração com Frontend (5 pontos):**
    - Uma interface de usuário interativa para interagir com a API.
- **Funcionalidades Avançadas de Autômatos (3 pontos):**
    - Implementação de algoritmos ou visualizações adicionais relacionadas a autômatos.

---

**Prazo Final:**
O projeto deve ser entregue até ***18/02/2024***. Submissões fora do prazo serão penalizadas conforme a política da disciplina.

**Nota Final:**
Este trabalho é uma oportunidade para combinar seu entendimento de conceitos teóricos de autômatos com habilidades práticas de programação. Participem da aula presencial para orientação sobre a configuração inicial e as expectativas do projeto. Em caso de dúvidas, não hesitem em entrar em contato.

Boa sorte e aproveitem o desafio!