**Atividade Desenvolvimento Orientado a Reuso de Software**

***Vinicius G. P. Drewnowski***

**Pesquisa e Seleção de Bibliotecas**

1. **Pesquisa e Seleção de Bibliotecas**
   1. **Linguagem Utilizada**

Linguagem escolhida: **Python**

**1.2 Bibliotecas escolhidas**

**Bibliotecas**:

Starlette x Flask

Pydantics x Marshmallow

* 1. **Markdown**

**Biblioteca**: Starlette  
**Última** **Atualização**: Versão 0.48.0 – 13/09/2025  
**Downloads** / **Semana**: 37.563.093 downloads durante a última semana  
**Licença**: Licença BSD 3-Clause  
**Complexidade**: Moderado  
**Observações**: Permissão livre de uso comercial e pessoal. Permissão de modificação e distribuição necessitam de regras e condições a serem seguidas.

**Biblioteca**: Pydantics   
**Última** **Atualização**: Versão 2.12.3 – 15/10/2025  
**Downloads** / **Semana**: 108.526.766 downloads durante a última semana  
**Licença**: Licença MIT (Open Source)  
**Complexidade**: Baixa e Moderada  
**Observações**: Termos de Uso – Exigência de caso uso da licença original e avisos de copyright, sejam incluídos no Projeto.

**Biblioteca**: Flask   
**Última** **Atualização**: Versão 3.1.2 – Agosto de 2025  
**Downloads** / **Semana**:  33.821.228 downloads durante a última semana  
**Licença**: Licença BSD 3-Clause  
**Complexidade**: Baixa   
**Observações**: Requisições síncronas (WSGI), Não possui suporte nativo a async/await no nível do Starlette

**Biblioteca:** Marshmallow   
**Última Atualização:** Versão 3.21.2 - Maio de 2024  
**Downloads / Semana:** 84 Milhões durante o mês  
**Licença:** MIT (Open Souce)  
**Complexidade:** Média  
**Observações:** Uma biblioteca poderosa de serialização/validação, mas que não utiliza *type hints* modernos. Sua verbosidade é maior e, crucialmente, ela não se integra nativamente ao FastAPI

* 1. **Decisão Técnica**

**Biblioteca escolhida**: Starlette   
**Motivo**:  
- Biblioteca leve e moderna para criação de aplicações e APIs assíncronas baseadas em ASGI.  
- É a base técnica do FastAPI, mas mantém o controle total nas mãos do desenvolvedor.  
- Oferece excelente performance, tipagem completa e compatibilidade com httpx e pydantic.  
**Alternativas descartadas**:  
**- Flask:** síncrono e baseado em WSGI, sem suporte nativo a *async*.

**Biblioteca escolhida**: Pydantics  
**Motivo**:  
- Fornece validação e serialização de dados baseada em *type hints*, garantindo segurança e integridade das entradas em APIs e sistemas complexos.  
- É moderna, de alto desempenho (reescrita em Rust a partir da versão 2) e amplamente usada em projetos Python atuais.  
- Integra-se perfeitamente com bibliotecas e frameworks assíncronos (como Starlette e FastAPI ).  
- A API é clara e consistente, facilitando a manutenção e o uso em sistemas de médio e grande porte.

**Alternativas descartadas:**  
**- Marshmallow:** mais verbosa e menos integrada com *type hints*.

1. **Implementação no Código**

Criação de API’s com operações CRUD (Create, Read, Update & Delete) usando a framework FastAPI com suas bibliotecas internas “Starlette” e “Pydantics”.

API’s criadas:

**/musicas/**

Texto

O conteúdo gerado por IA pode estar incorreto.

**/tasks/**

Texto

O conteúdo gerado por IA pode estar incorreto.

Link Github: <https://github.com/Vinicius-Drewnowski/Atv-Reuso---Bibliotecas>

1. **Arquivo READ.ME**

**# Módulo HTTP Reutilizável**

**## 📦 Biblioteca utilizada**

fastapi v0.120.0

pydantic v2.12.3

starlette v0.48.0

uvicorn v0.38.0

**## 🧰 Instalação**

npm install "fastapi[all]"

**## ⚙️ Execução**

uvicorn app.main:app --reload

**## 🧩 Motivação**

Atividade de criação de rotas com Python utilizando bibliotecas que se complementam através da framework FastAPI

**## 🔍 Saída esperada**

[INFO] GET http://127.0.0.1:8000/musicas/

[INFO] GET http://127.0.0.1:8000/musicas/1

Link Github: <https://github.com/Vinicius-Drewnowski/Atv-Reuso---Bibliotecas/blob/main/README.md>