Apresentação - Trabalho Bimestral 3

Grupo: Artistick
Integrantes: Ayanna, Ana Clara, Camila
Disciplina: PJI 2 / LTP 3

1. **Tema**
   Aplicativo Artistick — realiza a sugestão de atividades e avaliação de inteligências para crianças saírem das telas. O aplicativo é direcionado para sugerir as atividades aos pais, a criança não interage diretamente com o app.

2. **Entidades escolhidas**
   - Usuário (Responsável): cadastra conta e gerencia os seus filhos.
   - Filho: perfil da criança (relacionado ao usuário).
   - Atividade: atividades recomendadas/registradas para filhos.

3. **Relacionamentos**
   - Usuario (1) — (N) Filho
   - Filho (1) — (N) Atividade

4. **Estrutura do Banco**
   Tabelas: usuarios, filhos, atividades (ver schema em init_db.py)

5. **Endpoints principais**
   - POST url/usuarios → cria usuário
   - GET url/usuarios → lista usuários
   - GET url/usuarios/{id} → busca usuário por id
   - PUT url/usuarios/{id} → atualiza usuário por id
   - DELETE url/usuarios/{id} → apaga usuário por id
   - POST url/filhos → cria filho (usuario_id obrigatório)
   - GET url/filhos/{id} → busca filho por id
   - GET url/filhos?usuario={id} → lista filhos do usuário
   - PUT url/filhos/{id} → atualiza filho
   - DELETE url/filhos/{id} → remove filho por id
   - POST url/atividades → cria atividade (filho_id obrigatório)
   - GET url/atividades → lista atividades do filho
   - GET url/atividades/{id} → busca atividades por id
   - PUT url/atividades/{id} → atualiza atividade por id
   - DELETE url/atividade/{id} → deleta atividade por id

6. **Exemplos de testes (Insomnia)**

   USUÁRIOS

   1. Criar Usuário
   POST /usuarios
   Body:
   ```
   {
     "nome": "Maria Oliveira",
     "email": "maria.oliveira@example.com",
     "senha": "123456"
   }
   ```
   Resposta esperada:
   ```
   {
     "id": 1,
   ```

```
  "nome": "Maria Oliveira",
  "email": "maria.oliveira@example.com"
}
```

2. Listar Usuários
GET /usuarios
Resposta esperada:
```
[
  {
    "id": 1,
    "nome": "Maria Oliveira",
    "email": "maria.oliveira@example.com"
  }
]
```

3. Buscar Usuário por ID
GET /usuarios/1
Resposta esperada:
```
{
  "id": 1,
  "nome": "Maria Oliveira",
  "email": "maria.oliveira@example.com"
}
```

4. Atualizar Usuário
PUT /usuarios/1
Body:
```
{
  "nome": "Maria Silva",
  "email": "maria.silva@example.com"
}
```
Resposta esperada:
```
{
  "id": 1,
  "nome": "Maria Silva",
  "email": "maria.silva@example.com"
}
```

5. Deletar Usuário
DELETE /usuarios/1
Resposta esperada:
```
{
  "mensagem": "Usuário removido com sucesso."
}
```

FILHOS

6. Criar Filho
POST /filhos
Body:
```
{
  "nome": "João Oliveira",
  "idade": 7,
  "usuario_id": 1
}
```
Resposta esperada:
```
{
  "id": 1,
```

```
  "nome": "João Oliveira",
  "idade": 7,
  "usuario_id": 1
}
```

7. Criar Filho com usuario_id inválido (erro FIL001)
POST /filhos
Body:
```
{
  "nome": "Pedro",
  "idade": 5,
  "usuario_id": 999
}
```
Resposta esperada:
```
{
  "erro": "FIL001",
  "mensagem": "Usuário não encontrado."
}
```

8. Listar Filhos de um Usuário
GET /filhos?usuario=1
Resposta esperada:
```
[
  {
    "id": 1,
    "nome": "João Oliveira",
    "idade": 7,
    "usuario_id": 1
  }
]
```

9. Buscar Filho por ID
GET /filhos/1
Resposta esperada:
```
{
  "id": 1,
  "nome": "João Oliveira",
  "idade": 7,
  "usuario_id": 1
}
```

10. Atualizar Filho
PUT /filhos/1
Body:
```
{
  "nome": "João Pedro Oliveira",
  "idade": 8
}
```
Resposta esperada:
```
{
  "id": 1,
  "nome": "João Pedro Oliveira",
  "idade": 8,
  "usuario_id": 1
}
```

11. Deletar Filho
DELETE /filhos/1

Resposta esperada:
```
{
  "mensagem": "Filho removido com sucesso."
}
```

ATIVIDADES

12. Criar Atividade
POST /atividades
Body:
```
{
  "descricao": "Ler um livro",
  "data": "2025-09-24",
  "filho_id": 1
}
```
Resposta esperada:
```
{
  "id": 1,
  "descricao": "Ler um livro",
  "data": "2025-09-24",
  "filho_id": 1
}
```

13. Criar Atividade com filho_id inválido (erro ATI001)
POST /atividades
Body:
```
{
  "descricao": "Fazer tarefa de matemática",
  "data": "2025-09-25",
  "filho_id": 888
}
```
Resposta esperada:
```
{
  "erro": "ATI001",
  "mensagem": "Filho não encontrado."
}
```

15. Buscar Atividade por ID
GET /atividades/1
Resposta:
```
{
  "id": 1,
  "descricao": "Ler um livro",
  "data": "2025-09-24",
  "filho_id": 1
}
```

16. Atualizar Atividade
PUT /atividades/1
Body:
```
{
  "descricao": "Ler um capítulo do livro",
  "data": "2025-09-26"
}
```
Resposta:
```
{
  "id": 1,
  "descricao": "Ler um capítulo do livro",
```

```
  "data": "2025-09-26",
  "filho_id": 1
}
```


17. Deletar Atividade
DELETE /atividades/1
Resposta:


```
{
  "mensagem": "Atividade removida com sucesso."
}
```


7. **Conclusão:**
   Com o desenvolvimento da API, conseguimos aplicar na prática o que aprendemos em aula, usando banco de dados (SQLite3), organização em camadas e tratamento de erros. O aplicativo Artistick mostrou como a tecnologia pode ajudar os pais a oferecer atividades educativas para as crianças fora das telas. O trabalho também foi importante para treinar nossos conhecimentos de programação, modelagem de dados e testes de endpoints.