📚 Biblioteca Familiar API

Sistema de gerenciamento de biblioteca familiar com controle de empréstimos, avaliações e gamificação.

📋 Descrição

API REST desenvolvida em Flask para gerenciar uma biblioteca familiar, permitindo controle de empréstimos internos (família) e externos (amigos), sistema de avaliações, lista de desejos e gamificação com pontos de leitura.

🚀 Funcionalidades

Gestão de Membros

- Cadastro de membros da família com perfis personalizados
- Sistema de pontos e níveis (gamificação)
- Avatares coloridos e preferências de leitura
- Soft delete para desativação segura

Gestão de Livros

- Cadastro completo com ISBN, localização física e estado de conservação
- Controle de origem (comprado, presente, herdado)
- Classificação por idade recomendada
- Filtros por gênero, disponibilidade e idade

Sistema de Empréstimos

- Empréstimos internos (família) e externos (amigos)
- Controle de prazos diferenciados
- Registro de contatos para empréstimos externos
- Histórico completo de empréstimos

Avaliações e Recomendações

- Sistema de avaliação com 1-5 estrelas
- Comentários e tags personalizadas
- Recomendações por faixa etária

- Pontos extras por avaliações

Lista de Desejos

- Wishlist com prioridades (baixa, média, alta)
- Suporte para livros não cadastrados
- Notas e observações

Estatísticas e Rankings

- Dashboard com métricas gerais
- Leitor do mês
- Livros mais populares
- Valor total da biblioteca
- Taxa de atrasos

💻 Instalação

Pré-requisitos

- Python 3.8 ou superior
- pip (gerenciador de pacotes Python)
- SQLite (já incluído no Python)

Passos de Instalação

1. Clone o repositório:

```bash
git clone https://github.com/seu-usuario/biblioteca-familiar-backend.git
cd biblioteca-familiar-backend
```

2. Crie um ambiente virtual:

```bash
python -m venv venv
```

3. Ative o ambiente virtual:

Windows:

```bash
venv\Scripts\activate
```

Mac/Linux:

```bash
source venv/bin/activate
```

4. Instale as dependências:

```bash
pip install -r requirements.txt
```

5. Execute as migrações do banco de dados:

```bash
flask db init
flask db migrate -m "Initial migration"
flask db upgrade
```

6. Inicie o servidor:

```bash
python app.py
```

O servidor estará disponível em `http://localhost:5000`

📚 Documentação da API

Swagger UI

Acesse a documentação interativa em: `http://localhost:5000/apidocs`

Principais Endpoints

Membros da Família

- `GET /api/membros` - Lista todos os membros
- `POST /api/membros` - Cadastra novo membro

- `GET /api/membros/{id}` - Busca membro específico
- `DELETE /api/membros/{id}` - Desativa membro
- `PUT /api/membros/{id}/pontos` - Adiciona pontos de leitura

Livros

- `GET /api/livros` - Lista livros (com filtros opcionais)
- `POST /api/livros` - Cadastra novo livro
- `GET /api/livros/{id}` - Busca livro específico
- `PUT /api/livros/{id}` - Atualiza informações do livro

Empréstimos

- `POST /api/emprestimos` - Realiza empréstimo
- `PUT /api/emprestimos/{id}/devolver` - Devolve livro
- `GET /api/emprestimos` - Lista empréstimos com filtros
- `GET /api/emprestimos/membro/{id}` - Empréstimos por membro

Estatísticas

- `GET /api/estatisticas` - Dashboard completo com métricas

Avaliações

- `POST /api/avaliacoes` - Cria avaliação
- `GET /api/avaliacoes/livro/{id}` - Lista avaliações de um livro

Lista de Desejos

- `POST /api/wishlist` - Adiciona item
- `GET /api/wishlist` - Lista itens (com filtros)

🎮 Sistema de Gamificação

Níveis de Leitura

- **Iniciante**: 0-99 pontos
- **Leitor**: 100-499 pontos
- **Bookworm**: 500-999 pontos
- **Mestre dos Livros**: 1000+ pontos

Como Ganhar Pontos

- Devolver livro: 10-100 pontos (baseado no número de páginas)
- Bônus por devolução no prazo: +20 pontos
- Fazer avaliação: +15 pontos

🗂️ Estrutura do Projeto

```
biblioteca-familiar-backend/
│
├── app.py              # Aplicação principal
├── requirements.txt    # Dependências
├── README.md           # Este arquivo
│
├── models/             # Modelos de dados
│   ├── __init__.py
│   ├── membro.py       # Modelo de membros da família
│   ├── livro.py        # Modelo de livros
│   ├── emprestimo.py   # Modelo de empréstimos
│   ├── avaliacao.py    # Modelo de avaliações
│   └── wishlist.py     # Modelo de lista de desejos
│
├── routes/             # Rotas da API
│   ├── __init__.py
│   ├── membros.py      # Endpoints de membros
│   ├── livros.py       # Endpoints de livros
│   ├── emprestimos.py  # Endpoints de empréstimos
│   ├── estatisticas.py # Endpoints de estatísticas
│   ├── avaliacoes.py   # Endpoints de avaliações
│   └── wishlist.py     # Endpoints de lista de desejos
│
└── migrations/         # Migrações do banco de dados
    └── versions/       # Versões das migrações

## 🔧 Configuração Adicional

### Variáveis de Ambiente

Crie um arquivo `.env` na raiz do projeto (opcional):
```env
SECRET_KEY=sua-chave-secreta-aqui
DATABASE_URL=sqlite:///biblioteca_familiar.db
```
```

```
FLASK_ENV=development
FLASK_DEBUG=True
```

CORS

A API está configurada para aceitar requisições de:

- `http://localhost:*`
- `http://127.0.0.1:*`

Para produção, ajuste as origens permitidas no arquivo `app.py`.

🛠️ Tecnologias Utilizadas

- **Flask** - Framework web
- **Flask-SQLAlchemy** - ORM para banco de dados
- **Flask-Migrate** - Migrações de banco de dados
- **Flask-CORS** - Suporte a CORS
- **Flasgger** - Documentação Swagger/OpenAPI
- **SQLite** - Banco de dados

📝 Exemplos de Uso

Cadastrar Membro da Família

json

```
POST /api/membros
{
  "nome": "João Silva",
  "email": "joao@familia.com",
  "apelido": "Jô",
  "idade": 12,
  "tipo": "membro",
  "avatar_cor": "#FF6B6B",
  "generos_favoritos": "Aventura,Fantasia"
}
```

Cadastrar Livro

json

```
POST /api/livros
{
  "titulo": "Harry Potter e a Pedra Filosofal",
  "autor": "J.K. Rowling",
  "genero": "Fantasia",
  "subgenero": "Magia",
  "ano_publicacao": 1997,
  "num_paginas": 264,
  "idade_recomendada": "10+",
  "localizacao": "Estante A, Prateleira 2",
  "estado_conservacao": "Bom",
  "origem": "Presente"
}
```

Realizar Empréstimo para Membro

```
json

POST /api/emprestimos
{
  "id_membro": 1,
  "id_livro": 1,
  "tipo_emprestimo": "interno",
  "observacoes": "Leitura para escola"
}
```

Emprestar para Amigo

```
json

POST /api/emprestimos
{
  "id_livro": 2,
  "tipo_emprestimo": "externo",
  "nome_amigo": "Carlos Mendes",
  "contato_amigo": "(21) 98765-4321",
  "observacoes": "Prometeu devolver em 2 semanas"
}
```

🐛 Resolução de Problemas

Erro de Importação Circular

Se encontrar erros de importação circular, certifique-se de que está usando a função `create_app()` e importando os modelos dentro do contexto da aplicação.

Banco de Dados não Criado

Execute os comandos de migração:

```bash
flask db init
flask db migrate
flask db upgrade
```

Porta 5000 já em Uso

Modifique a porta no arquivo `app.py`:

```python
app.run(debug=True, port=5001)
```

🤝 Contribuindo

1. Faça um fork do projeto
2. Crie uma branch para sua feature (`git checkout -b feature/NovaFuncionalidade`)
3. Commit suas mudanças (`git commit -m 'Adiciona nova funcionalidade'`)
4. Push para a branch (`git push origin feature/NovaFuncionalidade`)
5. Abra um Pull Request

📄 Licença

👥 Autores

- **Seu Nome** - Desenvolvimento inicial

📞 Suporte

Para suporte, envie um email para suporte@bibliotecafamiliar.com ou abra uma issue no GitHub.

Desenvolvido com ❤️ para o curso de Desenvolvimento Full Stack Básico - PUC-Rio