

# LiftHub - Sistema de Gerenciamento de Alunos

## Descrição

O LiftHub é uma plataforma digital que conecta pessoais trainers e nutricionistas aos seus alunos. Este projeto implementa a funcionalidade inicial de cadastro e gerenciamento de alunos, permitindo que personal trainers registrem e gerenciem seus alunos de forma segura e funcional.

## Tecnologias Utilizadas

### Backend

- **Node.js** - Runtime JavaScript
- **Express.js** - Framework web
- **MongoDB** - Banco de dados NoSQL
- **Mongoose** - ODM para MongoDB
- **Jest** - Framework de testes
- **dotenv** - Gerenciamento de variáveis de ambiente
- **cors** - Middleware para CORS

### Frontend

- **React** - Biblioteca para interfaces de usuário
- **HTML5 & CSS3** - Estrutura e estilização
- **JavaScript ES6+** - Linguagem de programação

## Arquitetura do Projeto





O projeto segue uma arquitetura modular e princípios de Clean Code:

```
lifthub/  
├── backend/  
│   ├── controllers/    # Controladores da aplicação  
│   ├── services/      # Lógica de negócio  
│   └── models/         # Modelos do banco de dados
```

├── routes/	# Definição das rotas
├── utils/	# Utilitários (validação de CPF)
├── config/	# Configurações (banco de dados)
├── tests/	# Testes automatizados
├── scripts/	# Scripts auxiliares
└── frontend/	
└── lifthub-frontend/	
├── src/	
├── components/	# Componentes React
├── assets/	# Recursos estáticos
└── App.jsx	# Componente principal
└── public/	# Arquivos públicos

## Funcionalidades

### CRUD Completo de Alunos

-  **CREATE:** Cadastro de novos alunos com CPF único
-  **READ:** Listagem e visualização de alunos cadastrados
-  **UPDATE:** Edição do CPF de alunos existentes
-  **DELETE:** Remoção de alunos da base de dados

### Validação de CPF

- Validação completa de CPF brasileiro
- Verificação de dígitos verificadores
- Rejeição de CPFs com todos os dígitos iguais
- Formatação automática para exibição

### Interface de Usuário

- Design responsivo para desktop e mobile
- Feedback visual para todas as operações
- Mensagens de sucesso e erro
- Interface limpa e profissional

## Instalação e Execução

### Pré-requisitos

- Node.js (versão 14 ou superior)
- MongoDB (local ou Atlas)

- npm ou yarn

## 1. Clone o repositório

```
git clone <url-do-repositorio>  
cd lifthub
```

## 2. Configuração do Backend

```
cd backend  
npm install
```

## 3. Configuração do MongoDB

Certifique-se de que o MongoDB está rodando localmente ou configure a string de conexão no arquivo `.env` :

```
PORT=3001  
MONGODB_URI=mongodb://localhost:27017/lifthub
```

## 4. Inicialização do Banco de Dados

```
node scripts/initDB.js
```

## 5. Execução do Backend

```
npm start
```

O backend estará disponível em `http://localhost:3001`

## 6. Configuração do Frontend

```
cd ../frontend/lifthub-frontend  
npm install
```

## 7. Execução do Frontend

```
npm run dev
```






O frontend estará disponível em `http://localhost:5173`

### Testes

#### Executar Testes do Backend

```
cd backend  
npm test
```

#### Cobertura de Testes

-  Validação de CPF
-  Services de alunos
-  Controllers
-  Testes de integração
-  Simulação de erros

### API Endpoints

Base URL: `http://localhost:3001`

#### Alunos

Método	Endpoint	Descrição	Body
POST	<code>/alunos</code>	Cadastra um aluno	<code>{ "cpf": "11144477735" }</code>
GET	<code>/alunos</code>	Lista todos os alunos	-
GET	<code>/alunos/:cpf</code>	Busca aluno por CPF	-
PUT	<code>/alunos/:cpf</code>	Atualiza CPF do aluno	<code>{ "novoCpf": "22255588846" }</code>

Método	Endpoint	Descrição	Body
DELETE	/alunos/:cpf	Remove um aluno	-

## Exemplos de Resposta

### Sucesso (201/200)

```
{
  "success": true,
  "message": "Aluno cadastrado com sucesso",
  "data": {
    "_id": "...",
    "cpf": "11144477735",
    "createdAt": "2025-06-15T20:45:20.945Z",
    "updatedAt": "2025-06-15T20:45:20.945Z"
  }
}
```

### Erro (400/404)

```
{
  "success": false,
  "message": "CPF inválido"
}
```

## Validações Implementadas

### CPF

- Formato: 11 dígitos numéricos
- Validação de dígitos verificadores
- Unicidade no banco de dados
- Rejeição de sequências repetidas (111.111.111-11)

### Entrada de Dados

- CPF obrigatório para cadastro
- Sanitização de entrada (remoção de caracteres especiais)
- Validação no frontend e backend

# Interface do Usuário

## Características

- **Design Responsivo:** Funciona em desktop e mobile
- **Feedback Visual:** Mensagens de sucesso e erro
- **Formatação Automática:** CPF exibido no formato xxx.xxx.xxx-xx
- **Operações Intuitivas:** Botões claros para cada ação
- **Loading States:** Indicadores de carregamento durante operações

## Cores e Estilo

- Paleta de cores profissional (azul/roxo)
- Tipografia legível
- Espaçamento adequado
- Sombras e bordas suaves

## Estrutura de Dados

### Modelo do Aluno (MongoDB)

```
{
  _id: ObjectId,
  cpf: {
    type: String,
    required: true,
    unique: true,
    validate: [validadorCPF, 'CPF deve conter 11 dígitos']
  },
  createdAt: Date,
  updatedAt: Date
}
```

## Índices

- `cpf`: Índice único para garantir unicidade



# Status do Projeto



## Funcionalidades Implementadas

- [x] Arquitetura modular do backend
- [x] Modelo de dados do aluno
- [x] Validação completa de CPF
- [x] CRUD completo de alunos
- [x] API RESTful
- [x] Testes automatizados (Jest)
- [x] Interface React responsiva
- [x] Integração frontend-backend
- [x] Tratamento de erros
- [x] Feedback visual para usuário



## Próximas Funcionalidades (Roadmap)

- [ ] Autenticação de personal trainers
- [ ] Campos adicionais do aluno (nome, foto, etc.)
- [ ] Sistema de anamnese
- [ ] Módulo de treinos
- [ ] Acompanhamento de evolução
- [ ] Dashboard analítico
- [ ] Notificações
- [ ] Backup automático



## Contribuição

### Padrões de Código

- **Clean Code:** Nomes descritivos, funções pequenas
- **Modularização:** Separação clara de responsabilidades
- **Tratamento de Erros:** Sempre capturar e tratar erros
- **Testes:** Cobertura mínima de 80%
- **Documentação:** Comentários em funções complexas

### Estrutura de Commits

tipo(escopo): descrição

feat(alunos): adiciona validação de CPF

```
fix(api): corrige erro de duplicação  
docs(readme): atualiza instruções de instalação  
test(cpf): adiciona testes de validação
```

## Suporte

Para dúvidas ou problemas: 1. Verifique a documentação 2. Execute os testes para identificar problemas 3. Consulte os logs do servidor 4. Verifique a conexão com o MongoDB

## Licença

Este projeto está sob a licença MIT. Veja o arquivo LICENSE para mais detalhes.

---

**LiftHub** - Conectando personal trainers e alunos através da tecnologia 💪