Projeto: GymLog AI

Versão: 1.0 **Data de Início:** 19 de Setembro de 2025 **Autor:** Pablo Carvalho do Nascimento dos Santos

1. Visão Geral e Contexto

Este documento detalha o planejamento, a arquitetura e o desenvolvimento de um assistente pessoal para registro e análise de treinos de academia, apelidado de "GymLog AI".

Atualmente, o processo de registro de treinos é feito manualmente em um grupo de WhatsApp compartilhado, onde eu e minha namorada anotamos informações como local, exercícios, máquinas, pesos e repetições. Embora funcional, este método apresenta desafios significativos para a consulta de histórico, monitoramento de progresso e análise de performance de forma estruturada.

2. O Problema a ser Resolvido

O método atual (via WhatsApp) carece de estrutura, tornando tarefas como as seguintes ineficientes e manuais:

- **Análise de Progressão de Carga:** Verificar a evolução de peso em um exercício específico ao longo do tempo.
- **Monitoramento de Frequência:** Contabilizar o número de treinos em um determinado período (mês, semana).
- **Consulta Rápida:** Encontrar rapidamente qual foi a carga utilizada em um exercício na semana anterior.
- **Geração de Insights:** Extrair dados consolidados para tomar decisões sobre futuros treinos.

O objetivo do GymLog AI é automatizar e estruturar a coleta desses dados, e fornecer uma interface simples e poderosa para análise, resolvendo os problemas acima.

3. A Solução Proposta

A solução consiste em um **chatbot no Telegram** que utiliza uma **API de Modelo de Linguagem Grande (LLM)**, como a do ChatGPT, para interpretar as mensagens de texto livre dos usuários.

O fluxo de trabalho será o seguinte:

1. O usuário envia uma mensagem em linguagem natural para o bot no Telegram, descrevendo seu treino.
2. O bot encaminha essa mensagem para a API da OpenAI.
3. A API da OpenAI, instruída por um prompt de engenharia, extrai as informações relevantes (local, exercícios, séries, pesos, repetições) e as retorna em um formato de dados estruturado (JSON).
4. O bot recebe este JSON e o envia para um servidor de back-end local.
5. O servidor back-end valida os dados e os armazena em um banco de dados local.
6. Posteriormente, o usuário pode solicitar relatórios e visualizações de dados ao bot (ex: `/relatorio evolução supino`), que consulta o back-end para gerar e retornar as informações solicitadas.

4. Arquitetura e Stack Tecnológica

- **Interface do Usuário:** Bot no Telegram
- **Biblioteca do Bot:** `python-telegram-bot` (Python)
- **Processamento de Linguagem Natural:** API da OpenAI (GPT-3.5/4)

- **Servidor Back-end:** API REST com **FastAPI** (Python)
- **Banco de Dados: SQLite** (para simplicidade inicial)
- **ORM (Mapeamento Objeto-Relacional): SQLAlchemy** (Python)
- **Análise e Relatórios: Pandas** e **Matplotlib** (Python)
- **Túnel de Desenvolvimento: ngrok** (para expor o servidor local à internet)

5. Roadmap de Desenvolvimento (MVP - Produto Mínimo Viável)

Este roadmap divide o projeto em etapas gerenciáveis, com objetivos claros e tarefas chave.

Etapa 1: O Coração - Back-end e Banco de Dados

- **Objetivo Principal:** Criar a fundação onde os dados de treino serão armazenados e gerenciados.
- **Tarefas Chave:**
  - [ ] Instalar Python, FastAPI, SQLAlchemy e Uvicorn.
  - [ ] Definir os modelos de dados (tabelas) com SQLAlchemy: `User`, `WorkoutSession`, `Exercise`, `Set`.
  - [ ] Criar o arquivo do banco de dados SQLite.
  - [ ] Implementar o primeiro endpoint da API: `POST /log_workout` para receber e salvar os dados de um treino.
  - [ ] Implementar um endpoint de teste: `GET /workouts/{user_id}` para verificar se os dados foram salvos.
  - [ ] Testar os endpoints exaustivamente usando a interface de documentação automática do FastAPI (`/docs`).
- **Conceitos a Estudar/Reforçar:** Princípios de APIs REST, modelagem de dados SQL (chaves primárias/estrangeiras), sintaxe básica do SQLAlchemy.

Etapa 2: O Cérebro e o Mensageiro - Bot do Telegram com IA

- **Objetivo Principal:** Configurar o bot para receber mensagens e traduzi-las em dados estruturados usando a IA.
- **Tarefas Chave:**
  - [ ] Criar o bot no Telegram via BotFather e salvar o token de API de forma segura (variáveis de ambiente).
  - [ ] Instalar `python-telegram-bot` e a biblioteca `openai`.
  - [ ] Escrever a lógica básica do bot para responder a comandos simples (`/start`).
  - [ ] Desenvolver a função de "prompt engineering" que monta a instrução para a API da OpenAI.
  - [ ] Integrar a chamada à API da OpenAI: a função deve receber texto e retornar um objeto JSON.
  - [ ] Testar a robustez da extração de dados com diferentes estilos de mensagens.
- **Conceitos a Estudar/Reforçar:** Engenharia de Prompt para LLMs, manipulação de JSON em Python, programação assíncrona (se a biblioteca do bot utilizar).

Etapa 3: A Ponte - Conectando Bot e Back-end

- **Objetivo Principal:** Fazer com que os dados estruturados pelo bot sejam enviados e persistidos pelo back-end.
- **Tarefas Chave:**

- [ ] Instalar e configurar o `ngrok` para criar um túnel para o seu servidor FastAPI local.
- [ ] Instalar a biblioteca `requests` no ambiente do bot.
- [ ] Na lógica do bot, após receber o JSON da OpenAI, implementar a chamada `requests.post()` para o endpoint `/log_workout` da sua API via URL do ngrok.
- [ ] Implementar o tratamento de respostas (sucesso/erro) e enviar uma mensagem de feedback para o usuário no Telegram.
- [ ] Realizar o primeiro teste ponta a ponta: Enviar uma mensagem de treino no Telegram e verificar se os dados aparecem corretamente no banco de dados SQLite.

- **Conceitos a Estudar/Reforçar:** Requisições HTTP (POST, GET), status codes (200, 404, 500), uso de variáveis de ambiente para guardar chaves de API e URLs.

Etapa 4: A Recompensa - Relatórios e Visualizações

- **Objetivo Principal:** Permitir que o usuário extraia valor dos dados armazenados.
- **Tarefas Chave:**
  - [ ] Adicionar um novo handler de comando no bot (ex: `/relatorio <tipo> <exercicio>`).
  - [ ] Criar um novo endpoint no FastAPI (ex: `GET /report/exercise_progress`).
  - [ ] Implementar a lógica no back-end para consultar o banco de dados, filtrando pelo usuário e exercício.
  - [ ] Usar Pandas para estruturar os dados para análise.
  - [ ] Usar Matplotlib para gerar um gráfico (ex: evolução de carga vs. data).
  - [ ] Salvar o gráfico como um arquivo de imagem temporário.
  - [ ] Fazer o endpoint retornar o arquivo de imagem.
  - [ ] Fazer o bot chamar este endpoint, receber a imagem e enviá-la ao usuário no chat.
- **Conceitos a Estudar/Reforçar:** Manipulação básica de DataFrames com Pandas, geração de gráficos simples com Matplotlib.

---

6. Plano de Estudos Fundamentais (Track Paralelo)

Este plano visa fortalecer os conceitos base da Web, que são cruciais para o desenvolvimento de software moderno, incluindo o projeto da faculdade.

- **Foco 1: HTML Semântico**
  - **Recurso:** MDN Web Docs.
  - **Meta:** Entender a estrutura e o significado por trás das tags.
- **Foco 2: CSS Layouts Modernos**
  - **Recurso:** Flexbox Froggy, Grid Garden.
  - **Meta:** Dominar Flexbox e Grid para entender como o Tailwind funciona por baixo dos panos.
- **Foco 3: JavaScript Moderno (ES6+)**
  - **Recurso:** MDN Web Docs, tutoriais sobre `async/await`.
  - **Meta:** Ficar confortável com manipulação de arrays/objetos e programação assíncrona (`fetch`, `Promises`, `async/await`).

7. Cronograma Sugerido

Período	Foco Principal	Milestone
Set 22 - Out 05	Plano de Estudos Fundamentais (HTML, CSS, JS)	Concluir mini-projetos práticos de cada tecnologia.
Semana de Out 06	Etapa 1: Back-end e Banco de Dados	API local funcional, capaz de salvar dados via `/docs`.
Semana de Out 13	Etapa 2: Bot do Telegram com IA	Bot converte texto em JSON estruturado com sucesso.
Semana de Out 20	Etapa 3: A Ponte e Teste End-to-End	Primeiro treino registrado com sucesso do Telegram ao BD.
Semana de Out 27	Etapa 4: Relatórios Básicos	Bot gera e envia um gráfico de evolução de um exercício.
Novembro 2025	**MVP Concluído.** Refatoração e Planejamento v2.0	Sistema estável e funcional para uso pessoal.

8. Próximos Passos e Features Futuras (Pós-MVP)

- [ ] **Geração de Treinos:** Usar a IA para sugerir o próximo treino com base no histórico e em metas.
- [ ] **Dashboard Web:** Criar uma interface web com React (aproveitando os estudos da faculdade) para visualizações mais ricas e interativas.
- [ ] **Autenticação de Múltiplos Usuários:** Permitir que amigos e familiares usem o bot de forma segura.
- [ ] **Deploy na Nuvem:** Mover o back-end do servidor local para um serviço de nuvem (ex: Heroku, Render, AWS) para que ele funcione 24/7.
- [ ] **Melhorar a Extração de Dados:** Lidar com casos mais complexos, como superséries, drop sets, etc.