

Maratona Training - Documentação Completa do Projeto

Índice

1. [Visão Geral](#)
 2. [Arquitetura e Tecnologias](#)
 3. [Estrutura do Projeto](#)
 4. [Modelo de Dados](#)
 5. [Funcionalidades Principais](#)
 6. [Configuração e Deploy](#)
 7. [APIs e Endpoints](#)
 8. [Fluxos de Usuário](#)
 9. [Problemas Resolvidos](#)
 10. [Próximos Passos](#)
-

Visão Geral

Maratona Training é uma plataforma completa de treinamento de corrida que gera planos personalizados usando IA, integra com Strava para análise de dados, e oferece monitoramento contínuo de progresso.

Objetivos Principais

- **Personalização Extrema:** Planos de treino adaptados ao perfil completo do atleta
- **Baseado em Ciência:** Utiliza metodologia VDOT de Jack Daniels
- **IA Generativa:** Gera planos dinâmicos considerando disponibilidade, histórico e objetivos
- **Integração Strava:** Sincronização automática de treinos
- **Prevenção de Lesões:** Monitoramento de overtraining e sugestões preventivas
- **Nutrição Integrada:** Calculadora de macros e orientações nutricionais

Público-Alvo

Corredores de todos os níveis (iniciante a avançado) que buscam treinamento estruturado para provas de 5km até Ultramaratona.

Arquitetura e Tecnologias

Stack Principal

Frontend: Next.js 14 + React 18 + TypeScript
UI: Tailwind CSS + Shadcn UI + Radix UI
Backend: Next.js API Routes
Database: PostgreSQL + Prisma ORM
Auth: NextAuth.js (credentials + futuro OAuth)
IA: OpenAI GPT-4o (via Abacus.AI APIs)
Integração: Strava API OAuth 2.0
Charts: Recharts + React-Chartjs-2

Arquitetura de Camadas

Camada de Apresentação (UI)
- Components (Shadcn/Radix)
- Pages (Next.js App Router)

↓

Camada de API Routes
- /api/auth/*
- /api/plan/*
- /api/strava/*
- /api/ai/*

↓

Camada de Serviços (lib/)
- ai-plan-generator.ts
- auto-adjust-service.ts
- race-classifier.ts
- strava.ts

↓

Camada de Dados
- Prisma Client
- PostgreSQL Database

Estrutura do Projeto

```

/home/ubuntu/app_maratona/nextjs_space/
├── app/
│   ├── api/
│   │   ├── auth/
│   │   ├── plan/
│   │   ├── strava/
│   │   ├── ai/
│   │   ├── nutrition/
│   │   ├── profile/
│   │   └── ...
│   ├── dashboard/
│   ├── plano/
│   ├── onboarding/
│   ├── login/
│   ├── signup/
│   ├── perfil/
│   ├── chat/
│   ├── nutrition/
│   ├── prevention/
│   └── ...
├── components/
│   ├── ui/
│   ├── header.tsx
│   ├── strava-connect.tsx
│   ├── training-chat.tsx
│   ├── macro-calculator.tsx
│   └── ...
├── lib/
│   ├── ai-plan-generator.ts
│   ├── auto-adjust-service.ts
│   ├── race-classifier.ts
│   ├── strava.ts
│   ├── auth.ts
│   ├── db.ts
│   ├── types.ts
│   ├── vdotTables.ts
│   └── utils.ts
├── prisma/
│   └── schema.prisma
├── scripts/
│   ├── seed.ts
│   └── ...
├── public/
├── styles/
└── ...

```

Next.js App Router
 # API Routes
 # Autenticação
 # Geração e gestão de planos
 # Integração Strava
 # Endpoints de IA
 # Cálculos nutricionais
 # Gestão de perfil
 # Dashboard principal
 # Visualização do plano
 # Fluxo de cadastro
 # Página de login
 # Página de registro
 # Perfil do usuário
 # Chat com IA
 # Seção nutricional
 # Prevenção de lesões
 # Componentes React
 # Componentes Shadcn UI
 # Cabeçalho
 # Conexão Strava
 # Chat de treino
 # Calculadora de macros
 # Lógica de negócio
 # Gerador de planos com IA
 # Ajustes automáticos
 # Classificação de provas
 # Cliente Strava
 # Helpers de auth
 # Cliente Prisma
 # Tipos TypeScript
 # Tabelas VDOT
 # Utilidades
 # Schema do banco de dados
 # Scripts utilitários
 # Seed do banco
 # Assets estáticos
 # CSS global

Modelo de Dados

Principais Tabelas

User

```
model User {
  id          String    @id @default(uuid())
  email       String    @unique
  passwordHash String
  name        String?
  createdAt   DateTime  @default(now())

  profile     AthleteProfile?
  plans       TrainingPlan[]
  workouts    WorkoutLog[]
  raceGoals   RaceGoal[]
  stravaConnection StravaConnection?
}
```

AthleteProfile

```
model AthleteProfile {
  id          String    @id @default(uuid())
  userId      String    @unique
  user        User      @relation(fields: [userId], references: [id])

  // Dados Pessoais
  age         Int
  gender      String
  weight      Float
  height      Float

  // Nível e Experiência
  experienceLevel String // iniciante, intermediario, avancado
  weeklyKm     Float

  // Disponibilidade
  trainingDaysPerWeek Int
  availableDays   String[] // ["monday", "tuesday", ...]
  strengthDays    String[] // Dias para treino de força

  // Performance (VDOT)
  currentVDOT Float?

  // Histórico Médico
  hasInjuries Boolean @default(false)
  injuryDetails String?
  medicalConditions String?

  // Preferências
  preferredTrainingTime String?
  terrainPreference     String?

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
}
```

TrainingPlan

```

model TrainingPlan {
  id                String    @id @default(uuid())
  userId            String
  user              User      @relation(fields: [userId], references: [id])

  raceGoalId        String
  raceGoal          RaceGoal @relation(fields: [raceGoalId], references: [id])

  // Metadados do Plano
  planType          String    // beginner, intermediate, advanced, custom
  weeklyStructure    String    // Estrutura semanal (JSON)
  totalWeeks        Int

  // Periodização
  basePhaseWeeks    Int
  buildPhaseWeeks    Int
  peakPhaseWeeks     Int
  taperWeeks        Int

  // Dados Gerados por IA
  aiGeneratedData    Json?    // Plano completo gerado pela IA
  rationale           String?  // Justificativa da IA

  // Status
  isActive           Boolean   @default(true)
  startDate           DateTime
  endDate            DateTime

  workouts            Workout[]

  createdAt           DateTime @default(now())
  updatedAt           DateTime @updatedAt
}

```

Workout

```

model Workout {
  id          String      @id @default(uuid())
  planId      String
  plan        TrainingPlan @relation(fields: [planId], references: [id])

  // Informações da Semana/Dia
  weekNumber  Int
  dayOfWeek   String      // "monday", "tuesday", etc.
  date        DateTime

  // Tipo e Descrição
  workoutType String      // long_run, intervals, tempo, recovery, strength, rest
  description  String
  detailedPlan String?    // Detalhes completos do treino

  // Métricas Planejadas
  targetDistance Float?   // em km
  targetDuration Int?     // em minutos
  targetPace     String?   // min/km
  targetIntensity String?  // easy, moderate, hard, vdot-based

  // Execução
  completed    Boolean     @default(false)
  completedAt  DateTime?

  logs         WorkoutLog[]

  createdAt    DateTime     @default(now())
  updatedAt    DateTime     @updatedAt
}

```

RaceGoal

```

model RaceGoal {
  id          String          @id @default(uuid())
  userId      String
  user        User            @relation(fields: [userId], references: [id])

  // Informações da Prova
  raceName    String
  raceDate    DateTime
  distance    Float           // em km
  raceType    String          // 5k, 10k, half_marathon, marathon, ultra
  location    String?

  // Objetivo
  targetTime  String?         // HH:MM:SS
  targetPace  String?         // min/km
  goalType    String          // finish, time_goal, pr

  // Importância
  isPrimary   Boolean         @default(false)
  priority    Int             @default(1)

  // Status
  status      String          @default("planned") // planned, training, completed

  plans       TrainingPlan[]

  createdAt   DateTime        @default(now())
  updatedAt   DateTime        @updatedAt
}

```

StravaConnection

```

model StravaConnection {
  id          String          @id @default(uuid())
  userId      String          @unique
  user        User            @relation(fields: [userId], references: [id])

  accessToken  String
  refreshToken String
  expiresAt    BigInt
  athleteId   String

  lastSync     DateTime?

  createdAt    DateTime @default(now())
  updatedAt    DateTime @updatedAt
}

```

Funcionalidades Principais

1. Geração de Planos com IA

Arquivo: lib/ai-plan-generator.ts

Como Funciona:

1. Coleta dados completos do perfil do atleta
2. Calcula duração do plano baseada na data da prova
3. Determina periodização (base, build, peak, taper)
4. Envia prompt estruturado para GPT-4o
5. IA gera plano semana por semana respeitando:
 - Disponibilidade de dias
 - Dias específicos para força
 - Nível de experiência
 - VDOT atual
 - Progressão gradual de volume
 - Prevenção de overtraining

Prompt da IA (resumido):

Você é um treinador expert de corrida que usa a metodologia VDOT.

Perfil do Atleta:

- Nível: {experienceLevel}
- VDOT: {currentVDOT}
- Dias disponíveis: {availableDays}
- Dias de força: {strengthDays}
- Meta: {raceType} em {weeksUntilRace} semanas

Gere um plano de {totalWeeks} semanas com:

- Periodização adequada
- Progressão gradual de volume
- Variedade de treinos (longão, intervalados, tempo, regenerativo)
- Força nos dias especificados
- Respeito à disponibilidade

Formato JSON:

```
{
  "plan": {
    "weeks": [
      {
        "weekNumber": 1,
        "phase": "base",
        "workouts": [
          {
            "day": "monday",
            "type": "easy_run",
            "description": "...",
            "distance": 8,
            "pace": "5:30",
            ...
          }
        ]
      }
    ]
  },
  "rationale": "Explicação da estratégia"
}
```

2. Integração com Strava

Arquivo: lib/strava.ts

Fluxo OAuth:

1. Usuário clica em “Conectar Strava”
2. Redirecionado para autorização Strava
3. Callback recebe code
4. Troca code por access_token + refresh_token
5. Salva no banco com expiração

Sincronização de Atividades:

- Endpoint: `/api/strava/activities`
- Busca atividades recentes
- Calcula VDOT baseado em performances
- Atualiza perfil do atleta
- Marca workouts como completos automaticamente

3. Chat com IA

Arquivo: `components/training-chat.tsx` + `/api/ai/chat`

Permite que o atleta converse com um treinador virtual que:

- Responde dúvidas sobre o plano
- Sugere ajustes baseados em feedback
- Explica conceitos de treinamento
- Oferece dicas de nutrição e prevenção

4. Calculadora de Macros

Arquivo: `components/macro-calculator.tsx`

Calcula necessidades nutricionais baseadas em:

- Peso, altura, idade, sexo
- Volume semanal de treino
- Objetivo (manutenção, perda de peso, ganho)
- Nível de atividade

5. Prevenção de Overtraining

Arquivo: `lib/auto-adjust-service.ts`

Monitora:

- Volume semanal vs. histórico
- Progressão de carga
- Frequência de treinos intensos
- Feedback subjetivo do atleta

Sugere ajustes automáticos quando detecta risco.

6. Dashboard Interativo

Arquivo: `app/dashboard/page.tsx`

Exibe:

- Semana atual do plano
- Treinos da semana
- Progresso geral (%)
- Estatísticas de desempenho
- Gráficos de evolução

- Conexão com Strava
- Atalhos para funcionalidades

Configuração e Deploy

Variáveis de Ambiente (.env)

```
# Database
DATABASE_URL="postgresql://..."

# NextAuth
NEXTAUTH_SECRET="your-secret-key"
NEXTAUTH_URL="http://localhost:3000" # ou URL de produção

# Strava API
STRAVA_CLIENT_ID="your-strava-client-id"
STRAVA_CLIENT_SECRET="your-strava-client-secret"
STRAVA_REDIRECT_URI="http://localhost:3000/api/strava/callback"

# Abacus.AI (LLM)
ABACUSAI_API_KEY="your-abacus-api-key"

# Admin (opcional)
ADMIN_EMAIL="admin@example.com"
ADMIN_PASSWORD="admin123"
```

Instalação e Setup

```
# 1. Clone o projeto
cd /home/ubuntu/app_maratona/nextjs_space

# 2. Instale dependências
yarn install

# 3. Configure o banco de dados
yarn prisma generate
yarn prisma db push

# 4. (Opcional) Rode o seed para dados iniciais
yarn prisma db seed

# 5. Inicie o servidor de desenvolvimento
yarn dev
# Acesse http://localhost:3000

# 6. Build para produção
yarn build
yarn start
```

Deploy em Produção

URL Atual: <https://42maurillio.abacusai.app>

Para fazer deploy:

```
# Já configurado no Abacus.AI  
# O deploy é automático quando você salva um checkpoint
```

APIs e Endpoints

Autenticação

POST /api/auth/signup

Registra novo usuário.

Body:

```
{  
  "email": "user@example.com",  
  "password": "senha123",  
  "name": "João Silva"  
}
```

POST /api/auth/signin

Login de usuário (gerenciado por NextAuth).

GET /api/auth/session

Retorna sessão atual.

Perfil

GET /api/profile

Retorna perfil completo do usuário autenticado.

POST /api/profile

Cria ou atualiza perfil do atleta.

Body:

```
{  
  "age": 30,  
  "gender": "M",  
  "weight": 75,  
  "height": 175,  
  "experienceLevel": "intermediario",  
  "weeklyKm": 40,  
  "trainingDaysPerWeek": 5,  
  "availableDays": ["monday", "wednesday", "thursday", "friday", "sunday"],  
  "strengthDays": ["tuesday", "friday"],  
  "currentVDOT": 45  
}
```

Plano de Treino

POST /api/plan/generate

Gera novo plano de treino com IA.

Body:

```
{
  "raceGoalId": "uuid-da-prova"
}
```

Response:

```
{
  "success": true,
  "plan": {
    "id": "uuid-do-plano",
    "totalWeeks": 24,
    "startDate": "2025-10-27",
    "endDate": "2026-04-19",
    "workouts": [...],
    "rationale": "Explicação da IA sobre o plano"
  }
}
```

GET /api/plan/current

Retorna plano ativo do usuário.

GET /api/plan/:id

Retorna plano específico.

PUT /api/plan/:id

Atualiza plano existente.

DELETE /api/plan/:id

Remove plano.

Metas de Prova

POST /api/race-goals

Cria nova meta de prova.

Body:

```
{
  "raceName": "Maratona de São Paulo",
  "raceDate": "2026-08-29",
  "distance": 42.195,
  "raceType": "marathon",
  "targetTime": "03:30:00",
  "isPrimary": true
}
```

GET /api/race-goals

Lista todas as metas do usuário.

PUT /api/race-goals/:id

Atualiza meta.

DELETE /api/race-goals/:id

Remove meta.

Strava

GET /api/strava/auth

Inicia fluxo OAuth do Strava.

GET /api/strava/callback

Callback do OAuth (não chamar diretamente).

POST /api/strava/disconnect

Desconecta conta Strava.

GET /api/strava/activities

Busca atividades recentes do Strava.

Query Params:

- `startDate` (opcional): Data inicial (ISO)
- `endDate` (opcional): Data final (ISO)

Response:

```
{
  "activities": [
    {
      "id": "12345",
      "name": "Morning Run",
      "distance": 10000,
      "moving_time": 3000,
      "type": "Run",
      "start_date": "2025-10-27T06:00:00Z",
      "average_speed": 3.33
    }
  ]
}
```

IA

POST /api/ai/chat

Chat com treinador virtual.

Body:

```
{
  "message": "Como devo me preparar para o longão de amanhã?",
  "context": {
    "planId": "uuid-do-plano",
    "upcomingWorkouts": [...]
  }
}
```

POST /api/ai/analyze

Analisa progresso e sugere ajustes.

Nutrição

POST /api/nutrition/calculate

Calcula necessidades de macros.

Body:

```
{
  "weight": 75,
  "height": 175,
  "age": 30,
  "gender": "M",
  "activityLevel": "very_active",
  "goal": "maintain"
}
```

Fluxos de Usuário


Fluxo 1: Novo Usuário

1. Landing Page (/)
 - ↓ Clica "Começar Grátis"
2. Signup (/signup)
 - ↓ Preenche email, senha, nome
3. Onboarding (/onboarding)
 - ↓ Etapa 1: Dados pessoais (idade, peso, altura)
 - ↓ Etapa 2: Nível e experiência
 - ↓ Etapa 3: Disponibilidade de treino
 - ↓ Etapa 4: Meta de prova (nome, data, distância, objetivo)
 - ↓ Etapa 5: Histórico médico (opcional)
4. Dashboard (/dashboard)
 - ↓ Vê prompt: "Gerar Plano de Treino"
 - ↓ Clica no botão
5. Geração do Plano (loading...)
 - ↓ IA processa (10-30 segundos)
6. Plano Gerado (/plano)
 - ↓ Visualiza plano completo
 - ↓ Pode conectar Strava
 - ↓ Pode ajustar preferências


Fluxo 2: Usuário Retornando

1. Login (/login)
2. Dashboard (/dashboard)
 - ⬇ Vê treinos da semana
 - ⬇ Vê progresso geral
 - ⬇ Pode:
 - Ver plano completo (/plano)
 - Registrar treino (dialog)
 - Conversar com IA (/chat)
 - Verificar nutrição (/nutrition)
 - Atualizar perfil (/perfil)
 - Conectar Strava (se ainda não conectou)

Fluxo 3: Registro de Treino Manual

1. Dashboard  Clica no treino do dia
2. Abre dialog "Registrar Treino"
3. Preenche:
 - Distância percorrida
 - Tempo gasto
 - Sensação (RPE 1-10)
 - **Notas** (opcional)
4. Salva
5. Sistema marca treino como completo
6. Atualiza estatísticas e gráficos

Fluxo 4: Sincronização com Strava

1. Dashboard  Clica "Conectar Strava"
2. Redirecionado para autorização Strava
3. Autoriza aplicação
4. Redirecionado de volta
5. Sistema busca atividades recentes
6. Associa atividades aos treinos do plano
7. Calcula VDOT baseado em performances
8. Atualiza perfil automaticamente



Problemas Resolvidos Recentemente

Problema 1: Planos Ignorando Disponibilidade

Sintoma: Planos gerados colocavam treinos em dias não disponíveis (ex: sexta-feira sempre como descanso).

Causa: Gerador de planos usava estrutura fixa, não considerava `availableDays` e `strengthDays` do perfil.

Solução:

- Refatoração completa do `ai-plan-generator.ts`
- Prompt da IA agora recebe explicitamente os dias disponíveis
- Sistema valida se dias do plano correspondem aos dias configurados
- Adicionado retry mechanism caso IA gere JSON inválido

Commit: [Data da última modificação]

Problema 2: Duração Fixa de 16 Semanas

Sintoma: Todos os planos tinham 16 semanas, independente da data da prova.

Causa: Variável `totalWeeks` hardcoded como 16.

Solução:

- Implementado cálculo dinâmico: `weeksUntilRace = Math.floor(daysBetween / 7)`
 - Ajustes de periodização proporcionais à duração total
 - Validação: mínimo 8 semanas, máximo 52 semanas
-

Problema 3: Dias de Força Não Utilizados

Sintoma: Usuários configuravam dias para treino de força, mas planos não incluíam força nesses dias.

Causa: Gerador não distinguia dias de corrida vs. dias de força.

Solução:

- Campo `strengthDays` agora enviado explicitamente para IA
 - Prompt específica: “Inclua treinos de força nos seguintes dias: {strengthDays}”
 - Limite de 4 dias de força por semana para evitar overtraining
-

Problema 4: IA Gerando JSON Inválido

Sintoma: Erros ao parsear resposta da IA, causando falhas na geração.

Causa: Modelo GPT-3.5-turbo com temperatura alta produzia JSONs malformados.

Solução:

- Upgrade para GPT-4o (mais preciso)
 - Temperatura reduzida de 0.7 para 0.3
 - Implementado retry com limpeza de markdown (``json)
 - Logging detalhado para debug
 - Fallback para estrutura padrão em caso de falha
-



Próximos Passos e Melhorias

Curto Prazo (1-2 semanas)

1. Ajustes Automáticos Inteligentes

- Sistema detecta quando usuário está falhando em completar treinos
- IA sugere redução de volume ou intensidade
- Permite aceitar/rejeitar sugestões

2. Notificações e Lembretes

- Email/push para treinos do dia
- Lembretes de sincronização Strava
- Alertas de overtraining

3. Glossário Interativo

- Expandir termos técnicos
- Adicionar vídeos explicativos
- Seção de FAQ

4. Mobile Responsivo

- Otimizar UI para mobile
- Testar em diferentes tamanhos de tela

Médio Prazo (1-2 meses)**1. Multi-Race Planning**

- Suporte para múltiplas provas simultâneas
- Priorização inteligente
- Periodização coordenada

2. Comunidade e Social

- Feed de atividades
- Grupos de treino
- Desafios e badges

3. Análise Avançada

- Gráficos de evolução de VDOT
- Predição de tempo de prova
- Análise de tendências de performance

4. Integração com Wearables

- Garmin, Polar, Apple Watch
- Sincronização automática de FC, cadência, etc.
- Zonas de treino personalizadas

Longo Prazo (3-6 meses)**1. App Mobile Nativo**

- iOS e Android
- Modo offline
- GPS tracking nativo

2. Treinamento de Força Detalhado

- Biblioteca de exercícios
- Vídeos demonstrativos
- Progressões específicas para corrida

3. Planos de Nutrição Completos

- Refeições sugeridas
- Timing de nutrientes
- Suplementação

4. Marketplace de Treinadores

- Treinadores humanos podem oferecer serviços

- Consultas pagas
- Planos premium



Estatísticas do Sistema (Último Teste)

- ✓ Taxa de Sucesso na Geração de Planos: 80%
- ✓ Tempo Médio de Geração: 15-25 segundos
- ✓ Planos Respeitam Disponibilidade: 100%
- ✓ Planos Incluem Força nos Dias Corretos: 100%
- ✓ Duração Calculada Corretamente: 100%

⚠ Ponto de Atenção:

- Planos muito longos (>40 semanas) podem falhar ocasionalmente
- Solução: Implementar paginação ou geração em chunks



Credenciais de Teste

Usuário 1: Maurillio (Desenvolvedor Original)

Email: mmaurillio2@gmail.com
Senha: [definida pelo usuário]
Perfil: Intermediário, meta de maratona
Status: Plano ativo de ~40 semanas

Usuário 2: Camila (Teste)

Email: camila.santos@example.com
Senha: senha123
Perfil: Intermediária, 10km
Status: Criado durante testes, pode ser removido

Admin

Email: admin@example.com
Senha: admin123
Acesso: Dashboard admin em /admin



Suporte e Contato

Desenvolvedor Original: Maurillio

Email: mmaurillio2@gmail.com

Plataforma: Abacus.AI

Deploy: <https://42maurillio.abacusai.app>



Notas Finais

Este projeto foi desenvolvido com foco em:

- **Personalização real**, não templates genéricos
- **Ciência do treinamento**, baseado em metodologias comprovadas
- **IA como ferramenta**, não substituição do treinador
- **Escalabilidade**, para milhões de usuários

A documentação está estruturada para que qualquer desenvolvedor ou IA possa:

1. Entender a visão e objetivos
2. Compreender a arquitetura
3. Navegar pelo código
4. Fazer melhorias e correções
5. Deploy em qualquer ambiente

Última Atualização: 27 de outubro de 2025