**Documentação do Sistema: Tracking Military Physical Training**

**1. Visão Geral**

O "Tracking Military Physical Training" é uma aplicação web desenvolvida para permitir que militares da Força Aérea Brasileira (FAB), inicialmente focada na GUARNAE-RJ, registrem e acompanhem seus resultados no Teste de Avaliação do Condicionamento Físico (TACF) e suas atividades diárias de Treinamento Físico Militar (TFM). O sistema também oferece uma visão gerencial para acompanhamento agregado das atividades.

**2. Tecnologias Utilizadas**

**2.1. Backend (API)**

* **Linguagem:** JavaScript (Node.js)
* **Framework:** Express.js
* **Banco de Dados:** PostgreSQL
* **Bibliotecas Principais:**
  + pg: Driver de conexão com o PostgreSQL.
  + bcryptjs: Para hashing seguro de senhas.
  + jsonwebtoken (JWT): Para gerenciamento de sessões e autenticação baseada em token.
  + cors: Para permitir a comunicação entre frontend e backend.
  + dotenv: Para gerenciamento de variáveis de ambiente (credenciais de banco, segredos JWT).
* **Ferramentas de Desenvolvimento:** Nodemon (para reiniciar o servidor automaticamente durante o desenvolvimento).

**2.2. Frontend (WebApp)**

* **Framework/Biblioteca:** React.js (inicializado com Vite)
* **Linguagem:** JavaScript (com JSX)
* **Estilização:** Material-UI (MUI) - Biblioteca de componentes React baseada no Material Design.
* **Bibliotecas Principais:**
  + react-router-dom: Para gerenciamento de rotas (navegação entre páginas/abas).
  + axios: Para realizar chamadas HTTP para a API backend.
  + recharts: Para a criação dos gráficos de linha (Dashboard Pessoal) e barras/pizza (Dashboard Gerencial).
* **Outros:** HTML5, CSS3 (gerenciado principalmente pelo MUI e Emotion).

**2.3. Ferramentas Gerais**

* **Editor de Código:** Visual Studio Code (VS Code)
* **Controle de Versão:** Git / GitHub
* **Gerenciamento de Banco de Dados:** pgAdmin4
* **Testes de API:** Thunder Client (extensão do VS Code) ou Postman

**3. Arquitetura**

O sistema segue uma arquitetura cliente-servidor:

* **Frontend (Cliente):** Uma Single Page Application (SPA) construída em React que roda no navegador do usuário. É responsável pela interface do usuário e pela comunicação com o backend.
* **Backend (Servidor):** Uma API RESTful construída em Node.js/Express. É responsável pela lógica de negócios, autenticação, autorização e interação com o banco de dados PostgreSQL.

**4. Banco de Dados**

O banco de dados PostgreSQL armazena as informações em tabelas relacionais, incluindo:

* OrganizacaoMilitar: Lista das OMs (inicialmente da GUARNAE-RJ).
* Usuario: Dados cadastrais dos militares, incluindo credenciais de login e nivel\_acesso ('usuario' ou 'gerencial').
* LogTACF: Registros dos resultados do TAF de cada militar, incluindo campos antropométricos (peso, altura, cintura) e as menções calculadas (baseadas na ICA 54-1).
* Exercicio: Lista dos tipos de TFM disponíveis (Corrida, Natação, HIT, Outra Atividade, etc.) e os campos necessários para cada um (armazenados em JSONB).
* LogTFM: Registros dos treinos diários (TFM), incluindo data, exercício, percepção de intensidade e detalhes específicos do treino (armazenados em JSONB).

**5. Perfis de Usuário**

* **Militar (Padrão):** Pode se registrar, fazer login, editar seu perfil, registrar/editar/excluir seus próprios TAFs e TFMs, e visualizar seu dashboard pessoal ("Minha Evolução").
* **Gerencial:** Possui todas as permissões do militar padrão e, adicionalmente, pode acessar o "Dashboard Gerencial" para visualizar estatísticas agregadas e filtradas do efetivo.

**6. Descrição das Telas e Funcionalidades**

**6.1. Telas Públicas**

* **Login (/login):**
  + Permite ao usuário entrar no sistema usando SARAM e Senha.
  + Exibe o logo da UNIFA.
  + Utiliza componentes MUI (TextField, Button, Avatar).
  + Mostra mensagens de erro (ex: "Credenciais inválidas.").
  + Possui link para a tela de Registro.
  + Redireciona para o dashboard principal (/cadastro) após login bem-sucedido.
* **Registro (/registro):**
  + Formulário para cadastro de novos militares.
  + Campos: SARAM, Email, Senha (com confirmação), Nome Completo, Posto/Graduação (Dropdown MUI Select), Data de Nascimento, Sexo (Dropdown), Organização Militar (Dropdown preenchido via API com OMs da GUARNAE-RJ).
  + Exibe o logo da UNIFA.
  + Utiliza componentes MUI (TextField, Select, Button, Grid).
  + Validação básica (senhas conferem).
  + Exibe mensagens de sucesso ou erro (ex: "SARAM já cadastrado.").
  + Redireciona para a tela de Login após cadastro bem-sucedido.

**6.2. Telas Privadas (Acessíveis após Login)**

* **Layout Principal (DashboardLayout.jsx):**
  + Estrutura comum a todas as telas logadas.
  + AppBar (MUI) no topo com: Logo da UNIFA, Título da Aplicação, Saudação ao usuário, Botão "Sair" (Logout).
  + Barra de Tabs (MUI) para navegação entre as abas: "Cadastro Pessoal", "TACF", "TFM", "Minha Evolução".
  + A aba "Dashboard Gerencial" só aparece para usuários com nivel\_acesso 'gerencial'.
  + Container (MUI) onde o conteúdo da aba ativa é renderizado (<Outlet>).
  + Rodapé com informações institucionais.
* **Cadastro Pessoal (/ ou /cadastro):**
  + Formulário (MUI Paper, Grid, TextField, Select) preenchido com os dados atuais do usuário (buscados da API).
  + Permite editar: Nome, Posto (Dropdown), Data Nasc., Sexo (Dropdown), OM (Dropdown). SARAM e Email não são editáveis aqui.
  + Botão "Salvar Alterações".
  + Exibe mensagens de sucesso ou erro.
* **TACF (/tacf):**
  + Formulário (MUI Paper, Grid, TextField) para inserir ou editar resultados do TAF.
  + Campos: Data, Peso (kg), Altura (cm), Cintura (cm), Cooper (m), Abdominal (reps), Flexão (reps), Barra (reps).
  + Tabela (MUI TableContainer, Table, etc.) exibindo o histórico de TAFs do usuário, ordenado por data decrescente.
  + Colunas da Tabela: Data, Peso, Altura (m), **IMC (calculado no frontend)**, Cintura, Cooper, **Menção Cooper**, Abdominal, **Menção Abdominal**, Flexão, **Menção Flexão**, Barra, Ações.
  + As menções são calculadas pelo backend com base na ICA 54-1 (idade, sexo, resultado).
  + Botões de Ícone (MUI IconButton) para Editar e Excluir registros.
* **TFM (/tfm):**
  + Formulário (MUI Paper, Grid, TextField, Select) para inserir ou editar treinos diários.
  + Campos Principais: Data/Hora, Exercício (Dropdown preenchido via API), Percepção de Intensidade (0-10).
  + **Seção Dinâmica "Detalhes do Treino":** Campos TextField aparecem/desaparecem (usando renderCamposDinamicos) com base no exercício selecionado, conforme definido no backend (JSONB campos\_necessarios). Inclui a opção "Outra Atividade" que pede o nome da atividade.
  + Tabela (MUI) exibindo o histórico de TFMs, com Data, Exercício, Detalhes (formatados do JSONB), Intensidade e Ações (Editar/Excluir).
* **Minha Evolução (/evolucao):**
  + Dashboard pessoal do militar.
  + Exibe 4 Gráficos de Linha (Recharts) mostrando a evolução temporal (eixo X = Data do TAF) das métricas: Peso, Cooper, Flexão e Barra.
  + Busca dados de uma rota específica da API (/usuarios/me/historico).
  + Requer pelo menos 2 registros de TAF para exibir os gráficos.
* **Dashboard Gerencial (/admin - Acessível apenas por Gerente):**
  + Integrado ao DashboardLayout (mesmo cabeçalho/rodapé).
  + **Filtros:** Dropdowns (MUI Select) para filtrar os dados por Organização Militar (OM) e por Sexo. A opção "Todos" está disponível para ambos.
  + **Cards de Resumo:** Exibe métricas agregadas (MUI Paper, Grid) que respondem aos filtros (exceto onde indicado): Total de Usuários, Média Cooper, Média Abdominal, Média Flexão, Média Barra, Total Treinos TFM, Média Intensidade TFM, Média Peso, Média IMC, Média Cintura.
  + **Gráfico de Barras (Recharts):** Mostra o total de treinos TFM por OM (este gráfico *não* é afetado pelos filtros, servindo para comparação geral).
  + **Gráfico de Pizza (Recharts):** Mostra a distribuição percentual de treinos TFM por tipo de exercício, *refletindo os filtros* de OM e Sexo selecionados. Exibe mensagem se não houver dados para os filtros.

**7. Possíveis Próximas Etapas (Melhorias Futuras)**

* Adicionar mais filtros ao Dashboard Gerencial (ex: por Posto/Graduação, por Faixa Etária, por Período de Data).
* Melhorar a validação de dados nos formulários (frontend e backend).
* Criar um painel de administração mais completo (gerenciamento de usuários, adição/edição de exercícios e OMs pela interface).
* Implementar testes automatizados (unitários, integração, end-to-end).
* Realizar o "Deploy" da aplicação (backend e frontend) para um ambiente de produção na nuvem.
* Refinar a interface do usuário (UI) e a experiência do usuário (UX).