JITSU

Plataforma de Gestão Pedagógica e Comunidade para Jiu-Jitsu

Versão:	1.2.0 (MVP RN)
Data:	21 de Janeiro de 2026
Status:	Em planejamento
Produto:	Jitsu

Sumário

1. Visão Geral do Produto

Jitsu é uma plataforma SaaS/Mobile voltada para a gestão pedagógica e comunitária de turmas de Jiu-Jitsu. O produto não se propõe a ser um ERP financeiro de academias, mas sim um ecossistema digital do tatame, focado em:

- Jornada do aluno
- Autonomia do professor
- Organização pedagógica
- Comunidade privada e segura

O Jitsu centraliza presença, graduação, agenda, eventos e comunicação, respeitando a realidade do Jiu-Jitsu moderno, onde alunos e professores transitam entre diferentes times, unidades e contextos.

---

2. Objetivos do MVP

1. Eliminar o uso de fichas de papel para presença e graduação
2. Criar um histórico vitalício e portável do praticante
3. Facilitar a gestão do professor sem aumentar carga operacional
4. Organizar comunicação do time fora de redes sociais abertas
5. Estimular engajamento orgânico através de fotos, eventos e agenda

---

3. Princípios do Produto

- **Privado por padrão:** não existe busca aberta de times ou pessoas
- **Aluno no centro:** conta e histórico pertencem ao aluno
- **Professor no controle:** times, graduações, agenda e aprovações
- **MVP pragmático:** resolver dores reais antes de complexidade técnica
- **Cultura do Jiu-Jitsu:** foco no tatame, eventos, presença e respeito

---

4. Atores, Papéis e Permissões

4.1 Atores do Sistema

- **Aluno** - Praticante de Jiu-Jitsu
- **Professor** - Responsável por um ou mais times
- **Assistente** - Auxilia na gestão do time
- **Sistema** - Processos automatizados

4.2 Papéis no Time (RBAC)

- **STUDENT** - Acesso ao próprio perfil, check-in, feed e agenda do time
- **ASSISTANT** - Apoio operacional (ex.: ajustes simples e moderação conforme política do time)
- **HEAD_COACH** - Controle total do time: aprovações, agenda, graduação, moderação e exportação

---

5. Regras de Negócio

5.1 Conta e Histórico

- O perfil global pertence ao aluno
- Um aluno pode participar de múltiplos times
- A faixa atual é derivada do histórico de graduação
- O histórico nunca é apagado, apenas atualizado por novos registros

5.2 Entrada em Times

- Entrada apenas via convite
- Solicitação gera status **PENDING**
- Professor aprova (**ACTIVE**) ou rejeita (**REJECTED**)
- Professor pode corrigir faixa na aprovação

5.3 Graduação

- Graduação sempre manual
- Gera log imutável com professor responsável
- Pode estar associada a evento (graduação coletiva, seminário)

5.4 Presença

- Aluno pode marcar presença antes do treino
- Após o horário limite, apenas professor pode editar
- Professor pode ajustar presenças no fim do treino

5.5 Agenda

- Grade semanal cadastrada uma única vez
- Exceções sobrescrevem ocorrências recorrentes
- Aulas extras podem ser adicionadas manualmente

5.6 Comunidade

- Mural restrito ao time
- Eventos com link externo
- Fotos do treino publicadas pelo professor
- Exportação com marca d’água do app

5.7 Amizades

- Solicitações entre membros do time
- Adição fora do time via QR Code ou link
- Sem busca global de usuários

---

6. Requisitos Funcionais

6.1 Autenticação e Perfil

- Cadastro e login por e-mail
- Carteirinha digital por faixa
- Timeline de graduações
- Tempo em cada faixa calculado automaticamente

6.2 Times

- CRUD de times
- Convites com aprovação
- Lista de membros filtrável

6.3 Agenda

- Grade recorrente semanal
- Exceções por data
- Visualização semanal/mensal

6.4 Presença

- Self check-in do aluno
- Ajustes pelo professor
- Histórico por aluno e por sessão

6.5 Graduação

- Promoção individual e em lote
- Registro histórico imutável

6.6 Mural

- Posts e avisos
- Eventos com inscrição externa
- Reações customizadas (OSS, FOGO, JIU)

6.7 Foto do Treino

- Upload pelo professor
- Postagem no mural
- Exportação com marca d’água

6.8 Amizades

- Solicitações internas
- QR Code e link externo
- Lista de amigos

6.9 Campeonatos

- Registro de campeonatos
- Categoria, faixa, peso, resultado
- Link externo e verificação opcional

---

7. Requisitos Não Funcionais

- Mobile-first e baixa fricção de uso
- Segurança (JWT, hash forte, permissões por time)
- LGPD e consentimento explícito para mídia
- Logs, auditoria e trilha de ações (immutável)
- Escalabilidade moderada (crescer sem reescrever o produto)
- Observabilidade (erros, métricas, rastreabilidade)

8. Arquitetura Técnica

8.1 Decisões de Stack (recomendação)

- **Mobile App:** React Native (Expo) + TypeScript
- **Backend/API:** Python 3.11 + FastAPI
- **Banco de dados:** PostgreSQL 16
- **ORM:** SQLAlchemy (async) + Alembic (migrations)
- **Container:** Docker
- **Storage de mídia:** S3 compatível (ex.: AWS S3 ou equivalente)
- **Notificações:** Firebase Cloud Messaging (FCM)
- **Jobs (quando necessário):** Redis + Celery

A escolha prioriza: velocidade de MVP, boa experiência mobile, segurança e capacidade de crescer com baixo custo de manutenção.

8.2 Visão de Arquitetura (alto nível)

Fluxo principal:

- **App React Native:** consome a API FastAPI (HTTPS + JWT)
- **API:** persiste dados no PostgreSQL
- **Upload de fotos:** URL pré-assinada -> S3 (API só salva metadados)
- **Notificações:** disparadas via FCM (ex.: lembrete de treino, aprovação no time)
- **Processamentos pesados:** exportação/relatórios -> fila opcional Redis/Celery

8.3 Aplicativo Mobile

- **Organização:** estrutura por feature (auth, teams, schedule, attendance, feed, profile, media)
- **Roteamento:** React Navigation
- **HTTP:** Axios + interceptors (JWT/refresh, retry controlado)
- **Estado:** Zustand (ou Redux Toolkit, se preferir)
- **Offline leve:** cache local para agenda e carteirinha (SQLite + MMKV/AsyncStorage)
- **Segurança:** tokens em storage seguro (Keychain/Keystore)
- **Build/Distribuição:** EAS Build (Expo) + TestFlight/Play Console
- **Atualizações OTA (opcional):** Expo Updates (quando fizer sentido)

8.4 Backend/API

- **Arquitetura:** routers (controllers) -> services (regras) -> repositories (persistência)
- **Documentação:** OpenAPI/Swagger automático (FastAPI)
- **Autenticação:** JWT + refresh token; senha com Argon2 (ou bcrypt)
- **Autorização:** RBAC por time (STUDENT/ASSISTANT/HEAD_COACH) em todos endpoints sensíveis
- **Versionamento:** endpoints em /v1 (desde o início)
- **Qualidade:** validação forte com Pydantic, paginação e ordenação em listagens

8.5 Banco de Dados (PostgreSQL)

- **Chaves:** UUID em entidades (bom para mobile e crescimento)
- **Integridade:** constraints e índices (ex.: unique (session_id, user_id) para presença)
- **Histórico imutável:** graduation_history e audit_log só inserem (sem update/delete)
- **Migrations:** Alembic com versionamento no repositório
- **Performance:** índices em team_id, created_at e colunas de busca (ex.: feed)

8.6 Mídia (Fotos) e Exportação

- **Storage:** S3 compatível; banco guarda apenas metadados (team_id, owner_id, chave, tamanho, mime)
- **Upload:** pré-assinado (o app envia direto para o storage)
- **Retenção:** política por time (ex.: 30/90/365 dias) e consentimento explícito
- **Exportação:** geração com marca d'água; no início pode ser tarefa simples, depois mover para Celery

8.7 Jobs, Notificações e Agendamentos

- **Notificações:** FCM (push) com gatilhos de negócio (ex.: evento criado, treino cancelado, aprovação)
- **Jobs (opcional):** Redis + Celery para exportações, relatórios, limpeza por retenção e tarefas em lote
- **Agendamentos:** cron interno (Celery Beat) ou serviço de scheduler (quando em cloud)

8.8 Deploy e Operação

- **Container:** Docker para dev/staging/prod
- **Produção recomendada:** API em ECS Fargate (ou alternativa simples como Render/Fly.io no início)
- **Banco:** RDS PostgreSQL (backups automáticos)
- **Arquivos:** S3
- **Ambientes:** dev, staging e prod com variáveis separadas (secrets fora do repo)

8.9 Logs, Auditoria e Observabilidade

- **Logs estruturados:** JSON com correlation_id/request_id
- **Audit log:** trilha imutável de ações (promoções, aprovações, remoções, moderação)
- **Erros:** Sentry (mobile + backend)
    - **Métricas/tracing:** OpenTelemetry (quando necessário)

8.10 Alternativa de Stack (opcional)

Caso você prefira unificar todo o ecossistema em JavaScript/TypeScript, uma alternativa sólida é: React Native (TS) + NestJS (Node) + PostgreSQL. A recomendação principal deste documento passa a ser React Native (Expo) para o app; o backend pode permanecer em FastAPI ou migrar para NestJS conforme o objetivo do time.

9. Modelo de Dados (Resumo)

Principais entidades:

- users
- teams
- team_memberships
- invite_links
- class_schedules
- schedule_exceptions
- training_sessions
- attendance
- graduation_history
- posts
- reactions
- events
- session_media
- friend_requests
- competitions
- audit_log

Regras e constraints recomendadas:

- attendance: unique (session_id, user_id) para impedir duplicidade
- team_memberships: unique (team_id, user_id) + status (PENDING/ACTIVE/REJECTED)
- graduation_history e audit_log: tabelas imutáveis (insert-only)
- posts/events/session_media: sempre ligados a team_id (conteúdo restrito ao time)

---

10. API (Preview)

- POST /v1/auth/login
- POST /v1/auth/refresh
- GET /v1/me
- POST /v1/teams
- POST /v1/teams/{id}/join
- PATCH /v1/teams/{id}/members/{user}/approve
- POST /v1/schedules
- POST /v1/sessions/{id}/check-in
- POST /v1/graduations/promote
- POST /v1/posts
- GET /v1/feed
- POST /v1/media/upload-url

- POST /v1/media/export
- POST /v1/friends/request

---

11. Privacidade e LGPD

- Consentimento explícito para fotos (e possibilidade de revogação)
- Conteúdo restrito ao time (sem indexação pública)
- Bloqueio, denúncia e moderação
- Retenção configurável de mídia por time (política clara)
- Registro de auditoria para ações sensíveis

---

12. Roadmap de Desenvolvimento

- **Fase 1:** Auth, Times, Convites
- **Fase 2:** Agenda e Presença
- **Fase 3:** Graduação e Perfil
- **Fase 4:** Mural e Eventos
- **Fase 5:** Foto do treino e exportação
- **Fase 6:** Amizades
- **Fase 7:** Campeonatos

---

13. Métricas de Sucesso

- Ativação do professor (primeiro time configurado e primeiro treino lançado)
- Frequência semanal do aluno (presenças por semana)
- Uso da agenda (visualizações e check-ins originados por agenda)
- Exportações para redes sociais (uso do recurso de exportar foto)
- Retenção mensal (alunos ativos por time)

---

14. Monetização

- **Free:** 1 time, funções básicas
- **Pro:** múltiplos times, relatórios, mídia estendida
- **Team:** branding, permissões avançadas