# Manual do Projeto "DSMeventos": Plataforma de Microserviços para Eventos

**Visão Geral do Projeto**

O objetivo é construir uma plataforma web completa para criação e inscrição em eventos. A arquitetura será baseada em microserviços, onde cada parte fundamental do sistema é um serviço independente, com seu próprio banco de dados, que se comunica com os outros via API REST/HTTP.

Todos os serviços serão documentados com Swagger e implantados de forma independente na nuvem (Render).

**Fase 0: Preparação e Planejamento**

**Etapa 1: Divisão dos Grupos e Papéis**

Divisão em **4 equipes**, cada uma com uma missão clara:

1. **Equipe 1 (Serviço de Autenticação):** A base de tudo. Responsável por gerenciar usuários e garantir a segurança do sistema.
2. **Equipe 2 (Serviço de Eventos):** O coração do produto. Responsável pelo conteúdo principal da plataforma.
3. **Equipe 3 (Serviço de Inscrições):** A engrenagem de transações. Responsável pela interação entre usuários e eventos.
4. **Equipe 4 (Frontend & API Gateway):** A face do projeto. Responsável pela experiência do usuário e pelo ponto de entrada único do backend.

**Etapa 2: Configuração do Ambiente e Ferramentas**

- **Git & GitHub:** obs - cada equipe terá seu próprio repositório.
- **Node.js (v18+):** backend.
- **VS Code**
- **Postman ou Insomnia**
- **Conta na Render:** Todos devem criar uma conta gratuita para o deploy.
- **Conta na MongoDB Atlas:** Para o Banco

**Etapa 3: Definição dos Contratos da API com Swagger (Tarefa Coletiva)**

Todas as equipes de backend (1,2,3) devem se reunir para definir e documentar os endpoints no Swagger.

- **Ferramentas:** instalem swagger-ui-express e swagger-jsdoc (para descrever cada endpoint)

**Fase 1: Desenvolvimento dos Microserviços**

**Guia para Todas as Equipes de Backend:**

- **Estrutura:** Organizem o projeto em pastas (ex: routes, controllers, models, services).
- **Variáveis de Ambiente:** Usem um arquivo .env para todas as chaves secretas e configurações (conexão com o banco, segredo do JWT, URLs de outros serviços). **Nunca subam o arquivo .env para o GitHub (adicione-o no .gitignore).**
- **Banco de Dados:** Cada serviço **DEVE** ter seu próprio banco de dados no MongoDB Atlas.

**Equipe 1 (Serviço de Autenticação)**

- **Repositório:** DSMeventos-auth-service
- **Responsabilidades:** Cadastro, Login (com geração de JWT), busca e atualização de perfil de usuário.
- **Tecnologias:** Node.js, Express, MongoDB, Prisma, jsonwebtoken, bcryptjs, swagger-ui-express, swagger-jsdoc.
- **Endpoints Essenciais (Contrato):**
    - POST /auth/register: Cadastro.
    - POST /auth/login: Login, retorna o token JWT.
    - GET /users/me: Retorna dados do usuário logado (rota protegida).
    - PUT /users/me: Atualiza dados do usuário logado (rota protegida).

**Equipe 2 (Serviço de Eventos)**

- **Repositório:** DSMeventos-events-service
- **Responsabilidades:** CRUD (Criar, Ler, Atualizar, Deletar) de eventos.
- **Tecnologias:** Node.js, Express, MongoDB, Prisma, swagger-ui-express, swagger-jsdoc.
- **Endpoints Essenciais (Contrato):**
    - POST /events: Cria um novo evento (rota protegida).
    - GET /events: Lista todos os eventos (rota pública).
    - GET /events/:id: Detalhes de um evento (rota pública).
    - PUT /events/:id: Atualiza um evento (rota protegida, apenas o dono pode editar).
    - DELETE /events/:id: Deleta um evento (rota protegida, apenas o dono).

**Equipe 3 (Serviço de Inscrições)**

- **Repositório:** DSMeventos-orders-service

- **Responsabilidades:** Gerenciar inscrições de usuários em eventos.
- **Tecnologias:** Node.js, Express, MongoDB, Prisma, swagger-ui-express, swagger-jsdoc.
- **Comunicação Externa:** Este serviço **precisa** se comunicar com o events-service via REST/HTTP para verificar informações (ex: se o evento existe).
- **Endpoints Essenciais (Contrato):**
    - POST /orders/subscribe: Inscreve o usuário logado em um evento (rota protegida).
    - GET /orders/my-subscriptions: Lista os eventos em que o usuário logado está inscrito (rota protegida).
    - DELETE /orders/:subscriptionId: Cancela uma inscrição (rota protegida).

**Dossiê da Equipe 4 (Frontend & API Gateway)**

- **Repositório:** DSMeventos-web-app
- **Responsabilidades:**
    1. **API Gateway:** Criar um servidor Node.js/Express simples que atua como ponto de entrada único. Ele irá validar o JWT e redirecionar as requisições para os serviços corretos.
    2. **Frontend:** Construir a interface do usuário com Next.js, consumindo **apenas o API Gateway.**
- **Tecnologias:** Next.js, React, CSS (ex: Tailwind CSS ou outra que prefiram), Node.js/Express (para o Gateway).
- **Fluxo de Comunicação:**

    Usuário no navegador -> Frontend (Next.js) -> API Gateway -> Microserviço Específico.
- **Páginas Essenciais:** Home (listagem de eventos), Login, Cadastro, Detalhes do Evento, Criar Evento, Meu Perfil (com minhas inscrições).

**Fase 2: Deploy na Render e Integração**

Cada serviço (incluindo o API Gateway) será implantado como um "Web Service" separado e gratuito na Render.

**Passo a Passo do Deploy na Render para cada Serviço:**

1. **Crie um "New Web Service"** na Render.
2. **Conecte o Repositório do GitHub** correspondente ao serviço.
3. **Configure as Variáveis de Ambiente:** Na aba "Environment" do seu serviço na Render, adicione **todas** as variáveis do seu arquivo .env.

DATABASE_URL: A string de conexão do seu banco MongoDB Atlas.

JWT_SECRET: O segredo para os tokens.

**IMPORTANTE:** Para a comunicação entre serviços, adicione a URL pública do outro serviço (que a Render gera, ex: https://DSMeventos-events-service.onrender.com) nas variáveis de ambiente. O orders-service, **por exemplo**, terá uma variável EVENTS_SERVICE_URL=https://DSMeventos-events-service.onrender.com.

4. **Configure os Comandos:**
    - **Build Command:** npm install
    - **Start Command:** npm start
5. **Clique em "Create Web Service".** A Render fará o build e o deploy. Monitore os logs em caso de erros.

**Objetivo da Fase:**, todos os serviços devem estar rodando na Render e o Frontend deve conseguir se comunicar com o API Gateway para realizar um fluxo completo (cadastro, login, criação de evento, inscrição).

**Fase 3: Testes Finais e Apresentação**

- **Testes de Ponta a Ponta:** Realizem testes completos do sistema
- **Refinamento:** Corrijam bugs e melhorem a experiência do usuário no frontend.
- **Preparação da Apresentação:** Cada equipe deve preparar uma pequena apresentação sobre seu microserviço, mostrando:
    1. As responsabilidades do serviço.
    2. A documentação no Swagger.
    3. Os principais desafios técnicos enfrentados.
    4. Uma demonstração do serviço funcionando

A apresentação final será uma demonstração completa do **"DSMeventos"**, mostrando o sistema integrado e funcional.