



Documento Completo do Projeto tipo "Ninja" (Prestadores & Usuários)

Abaixo está um documento **extremamente detalhado**, contendo: - Arquitetura geral - Telas do app (Flutter) - Estrutura do backend (Spring Boot) - Todas as tabelas do banco com colunas e tipos - Fluxos completos - Nomes de classes, controllers, serviços, repositórios - Rotas da API - Dados enviados por cada requisição - Regras de negócio - Estrutura mínima do chat interno - Sistema de avaliações, fotos, pagamento e mediação - Controle de login obrigatório apenas quando necessário



1. Arquitetura Geral do Sistema



Visão Geral

O app terá dois tipos de usuários: 1. **Cliente** (contrata serviços) 2. **Prestador de Serviço** (aceita trabalhos)

Serviços são publicados pelos clientes, e prestadores podem aceitar. Pagamento é retido até o cliente confirmar com foto. Avaliação aparece no perfil e influencia prioridade de trabalhos.

O sistema terá: - App Flutter para clientes e prestadores - Backend em Spring Boot 3 + PostgreSQL - WebSocket para chat e notificações - Sistema de upload de imagens - Sistema de avaliação e pontuação



2. Telas do Aplicativo (Flutter)

SplashPage

- Nome: `SplashPage`
- Função: carregar dados, verificar login, navegar para Home ou Login.

HomePage

- Nome: `HomePage`
- Exibe lista de serviços publicados
- Filtros (por valor, categoria, distância)
- Acesso ao perfil e notificações

LoginPage

- Nome: `LoginPage`
- Campos: email, senha

CadastroPage

- Nome: **CadastroPage**
- Campos: nome, email, senha, tipo (cliente/prestador)

PerfilPage

- Dados pessoais
- Avaliações
- Fotos de trabalhos realizados (somente prestadores)

PublicarServicoPage (cliente)

- Nome: **PublicarServicoPage**
- Campos:
 - título
 - descrição
 - preço
 - categoria
 - fotos opcionais

 Requer login somente ao publicar.

DetalheServicoPage

- Info do serviço
- Botão "Aceitar Serviço" (somente prestador)

ChatPage

- Conversa em tempo real
- Notificações de novas mensagens

MinhasSolicitacoesPage (cliente)

- Lista de serviços publicados
- Status do serviço (aguardando, aceito, concluído)

MeusServicosPrestadorPage (prestador)

- Lista de serviços aceitos
- Chat com cliente

ConfirmarEntregaPage (cliente)

- Enviar fotos comprovando conclusão

PagamentoPage

- Exibe status do pagamento

- O dinheiro é liberado ao prestador quando o cliente confirma



3. Estrutura de Pastas do Flutter

```
lib/  
src/  
  pages/  
    splash/  
    home/  
    login/  
    cadastro/  
    publicar_servico/  
    detalhe_servico/  
    chat/  
    perfil/  
    prestador/  
    cliente/  
  models/  
  controllers/ (ou bloc, ou provider)  
  services/  
    api/  
    websocket/  
  utils/  
  widgets/
```



4. Tabelas do Banco de Dados (PostgreSQL)

1. users

| coluna | tipo | descrição |
|------------------|-----------------------------|---------------|
| id | UUID PK | id do usuário |
| nome | varchar(150) | |
| email | varchar(150) UNIQUE | |
| senha_hash | varchar | |
| tipo | enum('CLIENTE','PRESTADOR') | |
| foto_perfil | varchar | URL |
| avaliacao_media | decimal(2,1) default 0 | |
| total_avaliacoes | int default 0 | |
| data_criacao | timestamp | |

2. servicios

| coluna | tipo |
|-----------------|---|
| id | UUID PK |
| cliente_id | UUID FK users(id) |
| titulo | varchar |
| descricao | text |
| preco | decimal(10,2) |
| categoria | varchar |
| status | enum('ABERTO','ACEITO','CONCLUIDO','CANCELADO') |
| prestashop_id | UUID FK users(id) NULL |
| data_publicacao | timestamp |
| data_aceito | timestamp NULL |
| data_concluido | timestamp NULL |

3. servicio_fotos

| coluna | tipo |
|-------------|---------|
| id | UUID |
| servicio_id | UUID |
| url | varchar |

4. chat_mensagens

| coluna | tipo |
|-----------------|-----------|
| id | UUID |
| servicio_id | UUID |
| remetente_id | UUID |
| destinatario_id | UUID |
| mensagem | text |
| data_envio | timestamp |
| visto | boolean |

5. avaliações

| coluna | tipo |
|--------------|-----------|
| id | UUID |
| avaliador_id | UUID |
| avaliado_id | UUID |
| nota | int (1-5) |
| comentario | text |
| data | timestamp |

6. pagamentos

| coluna | tipo |
|----------------|--|
| id | UUID |
| servico_id | UUID |
| cliente_id | UUID |
| prestador_id | UUID |
| valor_total | decimal(10,2) |
| status | enum('PENDENTE','RECEBIDO','LIBERADO') |
| data_pagamento | timestamp |
| data_liberacao | timestamp |

5. Backend - Spring Boot

Estrutura de pacotes

```
com.app.ninja/
  controller/
  model/
  dto/
  repository/
  service/
  config/
```

security/
websocket/



6. Controllers e Rotas da API (com DTOs)

AuthController

POST /auth/login

Envia

```
{ "email": "", "senha": "" }
```

Retorna

```
{ "token": "jwt", "usuario": {...} }
```

POST /auth/cadastro

```
{
  "nome": "",
  "email": "",
  "senha": "",
  "tipo": "CLIENTE"
}
```

ServicoController

POST /servicos

```
{
  "titulo": "",
  "descricao": "",
  "preco": 100.0,
  "categoria": "",
  "fotos": ["url1", "url2"]
}
```

GET /servicos/abertos

POST /servicos/{id}/aceitar

(prestashop)

POST /servicos/{id}/confirmar-entrega

(cliente)

```
{  
  "fotos": ["ur1"]  
}
```

ChatController (WebSocket)

ws://api.com/ws/chat

Envia:

```
{  
  "servicoId": "",  
  "mensagem": "",  
  "destinatarioId": ""  
}
```

Recebe:

```
{  
  "remetente": "",  
  "mensagem": "",  
  "timestamp": ""  
}
```

AvaliacoesController

POST /avaliacoes

```
{  
  "avaliadoId": "",  
  "nota": 5,  
}
```

```
        "comentario": "Muito bom!"  
    }  
  
-----
```



7. Regras de Negócio



Prestadores só podem aceitar serviço se:

- Têm boa nota (regra configurável)
- Não estão bloqueados



Cliente só libera pagamento após enviar foto confirmado.



Prestadores com mais pontos visualizam serviços melhores.



Chat interno obrigatório para qualquer serviço aceito.



Login só exigido para publicar serviço ou aceitar serviço.



Fotos são armazenadas em S3 (AWS) ou Cloud Storage.



8. Notificações

Usar Firebase Cloud Messaging (FCM): - nova mensagem - serviço aceito - serviço concluído



9. Upload de Arquivos

Endpoint: **POST /upload** Retorna:

```
{ "url": "https://bucket/...jpg" }
```



10. Autenticação

- JWT
- Refresh Token opcional
- UserDetailsService custom

11. O que você pode adicionar depois

- Geolocalização para serviços locais
 - Pagamento via PIX automático
 - Plano VIP para prestadores
 - Ranking semanal
-

Se quiser, posso gerar também: UML completa (caso de uso + classes)

- Scripts SQL completos
- Estrutura do backend com todas as classes vazias
- Estrutura completa do Flutter com arquivos prontos

É só pedir!