

Documentação Técnica

Sistema Tecendo Saúde

Aplicação de Telemedicina para Regiões Remotas da Amazônia

Projeto: Baixo Amazonas e Tapajós

10 de dezembro de 2025

Sumário

1	Introdução	3
1.1	Problema que o Sistema Resolve	3
1.2	Quem Usa o Sistema	3
2	Arquitetura do Sistema	4
2.1	Camada de Interface (Frontend)	4
2.2	Camada Offline (IndexedDB + Dexie.js)	4
2.2.1	Stores Locais	4
2.3	Camada de Sincronização (Motor Automático)	5
2.4	Camada Backend (Supabase)	5
3	Fluxos de Uso Detalhados	6
3.1	Paciente (Offline)	6
3.1.1	Login e Cadastro Básico	6
3.1.2	Novo Registro	6
3.1.3	Histórico e Chat	7
3.2	Profissional (Online)	7
3.2.1	Autenticação	7
3.2.2	Dashboard Profissional	7
3.2.3	Chat Profissional	7
4	Gestão de Medicações	9
4.1	Lista	9
4.2	Editor	9
4.3	Tipos de Medicamentos	9
5	Conteúdo Educativo (Saiba Mais)	10
6	Componentes React Principais	10
6.1	Componentes Reutilizáveis	10
7	Funções Utilitárias	10
8	Dados Estáticos	12
8.1	Regiões (13)	12
8.2	UBS (9)	12
8.3	ACS (92)	12
9	Sistema de Badges	12
10	Limitações e Considerações	12
10.1	Plano Gratuito Supabase	12
10.2	Segurança	12
10.3	Escalabilidade	13
11	Conclusão	14
11.1	Trabalhos Futuros	14

1 Introdução

O **Tecendo Saúde** é um sistema de saúde digital criado para conectar pacientes de regiões remotas da Amazônia (especificamente o Baixo Amazonas e região do Tapajós) com profissionais de saúde. O principal diferencial deste sistema é que ele **funciona completamente sem internet** para os pacientes, salvando todos os dados no próprio dispositivo (celular ou computador) e sincronizando automaticamente quando houver conexão disponível.

1.1 Problema que o Sistema Resolve

Muitas comunidades amazônicas não têm acesso regular à internet ou ficam a longas distâncias das unidades de saúde. Quando um paciente sente um sintoma (febre, dor, ferida), ele precisa:

- Viajar horas de barco até a cidade mais próxima.
- Esperar dias para conseguir atendimento.
- Em alguns casos, o problema se agrava pela demora.

Com o Tecendo Saúde, o paciente pode:

- **Registrar sintomas** (texto, fotos, vídeos, áudios) diretamente no celular.
- **Salvar tudo sem internet** no próprio aparelho.
- Quando chegar em uma área com WiFi ou dados móveis, o sistema **envia automaticamente** tudo para os profissionais de saúde.
- Receber respostas com orientações e visualizá-las no celular.

1.2 Quem Usa o Sistema

Usuários do Sistema

Pacientes: Moradores das 13 regiões atendidas (Santarém, Belterra, Mojuí dos Campos, Alenquer, Curuá, Óbidos, Oriximiná, Terra Santa, Faro, Juruti, Monte Alegre, Almeirim, Prainha).

Profissionais: Agentes Comunitários de Saúde (ACS), enfermeiros e médicos das 9 UBS cadastradas.

2 Arquitetura do Sistema

A arquitetura do sistema é dividida em três camadas principais que trabalham juntas.

2.1 Camada de Interface (Frontend)

- **React 18 UMD:** Componentes declarados dentro de `index.html`.
- **Babel Standalone:** Compila JSX diretamente no navegador.
- **TailwindCSS via CDN:** Estilos utilitários sem build.
- **Arquivo Único:** `index.html` com 1.800+ linhas. Basta abrir no navegador ou servir via HTTP estático.

Vantagem Técnica

Nenhuma dependência de Node.js, npm, Webpack ou bundlers. Para publicar, copie `index.html` para um servidor ou abra localmente (com `python -m http.server` por exemplo).

2.2 Camada Offline (IndexedDB + Dexie.js)

- Banco local `TecendoSaudeDB_V22_Fixed`.
- Biblioteca Dexie v3 gerencia stores e migrações.

2.2.1 Stores Locais

1. **perfil:** Dados do paciente (30+ campos, foto base64, metas, histórico).
2. **registros:** Consultas com texto, status, replies, timestamps, flags de notificação.
3. **midias:** Blobs de fotos/vídeos/áudios associados a registros.
4. **medicamentos:** Prescrições com tipo, dosagem, horários, datas e status.

```
const db = new Dexie('TecendoSaudeDB_V22_Fixed');

db.version(1).stores({
  midias: '++id, registroId, name, type, synced'
});

db.version(3).stores({
  medicamentos: '++id, medicationId, patientId, synced, tipo_medicamento,
    nome_medicamento, dosagem, horarios, data_prescricao, data_dispensacao,
    data_inicio, data_termino, ativo'
}).upgrade(tx => {
  return tx.table('perfil')
    .toCollection()
    .modify(p => { if (!p.foto_url) p.foto_url = ''; });
});
```

2.3 Camada de Sincronização (Motor Automático)

- Loop de 5s em syncManager().
- Apenas executa se navigator.onLine = true.
- Entidades sincronizadas em ordem: perfis → registros → mídias → medicações.

```
async function syncManager() {
  if (!navigator.onLine) return;

  const pendingProfiles = await db.perfil.where('synced').equals(0).toArray();
  for (const perfil of pendingProfiles) {
    await supabase.from('perfis').upsert(perfil);
    await db.perfil.update(perfil.id, { synced: 1 });
  }

  const pendingRecords = await db.registros.where('synced').equals(0).toArray();
  for (const registro of pendingRecords) {
    await supabase.from('registros').upsert(registro);
    await db.registros.update(registro.id, { synced: 1 });
  }

  const pendingMedias = await db.midias.where('synced').equals(0).toArray();
  for (const media of pendingMedias) {
    const path = `${media.registroId}/${media.name}`;
    await supabase.storage.from('midias')
      .upload(path, media.blob, { upsert: true, contentType: media.type });
    await db.midias.update(media.id, { synced: 1 });
  }

  const pendingMeds = await db.medicamentos.where('synced').equals(0).toArray();
  for (const med of pendingMeds) {
    await supabase.from('medicamentos').upsert(med);
    await db.medicamentos.update(med.id, { synced: 1 });
  }
}
```

2.4 Camada Backend (Supabase)

- PostgreSQL: Tabelas perfis, registros, medicamentos, profissionais.
- Storage: Bucket midias. Estrutura midias/<registroId>/<arquivo>.
- API REST auto-gerada (supabase-js@2).

3 Fluxos de Uso Detalhados

3.1 Paciente (Offline)

3.1.1 Login e Cadastro Básico

1. Paciente entra em “SOU PACIENTE” e informa CPF.
2. Sistema busca localmente, depois Supabase (se online).
3. Se não encontrar, abre cadastro com 6 campos obrigatórios.

```
const basicProfile = {
  nome: 'Joao da Silva',
  cpf: '12345678900',
  nascimento: '15/03/1985',
  regiao: 'Santarem',
  telefone: '(91) 98765-4321',
  email: 'joao@exemplo.com',
  patientId: `cpf-${'12345678900'}`,
  synced: 0
};
await db.perfil.add(basicProfile);
localStorage.setItem('currentProfileId', basicProfile.patientId);
```

3.1.2 Novo Registro

- Componente NovaMensagem: texto + anexos.
- Limites: foto até 1MB, áudio/vídeo 60s, combinação exclusiva (1 vídeo ou 1 foto + 1 áudio).
- Salva registros e midias com synced=0.

```
const registroId = uuidv4();

await db.registros.add({
  registroId,
  patientId,
  texto,
  tipo: files[0]?.type || 'texto',
  status: 'pendente',
  replies: [],
  createdAt: new Date().toISOString(),
  synced: 0
});

for (const file of files) {
  await db.midias.add({
    registroId,
    name: file.name,
```

```

    type: file.type,
    blob: file,
    synced: 0
  });
}

```

3.1.3 Histórico e Chat

- Dashboard lista cards com data, resumo, thumbnail (`RegistroThumb`).
- Modal `PatientHistoryModal` mostra conversa completa e anexos.
- Atualização automática a cada 15s quando online.

3.2 Profissional (Online)

3.2.1 Autenticação

- Tela “ÁREA PROFISSIONAL” solicita CPF.
- Busca `supabase.from('profissionais')` por CPF.
- Se não existir, cadastro com nome (92 ACS), UBS (9), telefone.

3.2.2 Dashboard Profissional

- Contadores automáticos (30s) para atendimentos pendentes e cadastros incompletos.
- Lista de registros com badges (atendimento, medicação, cadastro).
- Acesso ao prontuário completo (`FichaMedica`).
- Chat assíncrono via `MonitorConversationModal`.

```

SELECT registro_id, patient_id, status, updated_at
FROM registros
ORDER BY updated_at DESC;

SELECT patient_id, nome, foto_url, genero
FROM perfis;

```

3.2.3 Chat Profissional

```

// Buscar midias localmente primeiro
const localMedias = await db.midias
  .where('registroId').equals(registroId)
  .toArray();

// Se não houver localmente, buscar no Supabase
if (localMedias.length === 0 && navigator.onLine) {
  const { data: files } = await supabase.storage

```

```
.from('midias')  
.list(registroId);  
// Baixar URLs assinadas conforme necessidade  
}
```

```
UPDATE registros  
SET status = 'respondido',  
    updated_at = NOW(),  
    replies_json = replies_json || :nova_resposta,  
    resposta = :texto_resumo  
WHERE registro_id = :registroId;
```

4 Gestão de Medicações

4.1 Lista

- `MedicationManagementList` combina dados locais/offline com Supabase.
- Filtros por nome/CPF.
- Badges: cadastro pendente, medicação pendente, atendimento pendente.

4.2 Editor

- Campos: nome, tipo (17 categorias), dosagem, horários (12 padrões), datas de prescrição/início/término, status ativo.
- Validação: nome, tipo, ao menos um horário.
- Salva local e, quando online, envia ao Supabase.

4.3 Tipos de Medicamentos

1. Analgésicos
2. Anti-inflamatórios
3. Corticoides
4. Antibióticos
5. Antifúngicos
6. Antivirais
7. Anti-hipertensivos
8. Antidiabéticos
9. Anticoagulantes
10. Antidepressivos
11. Ansiolíticos
12. Antialérgicos
13. Broncodilatadores
14. Antiácidos
15. Hipolipidemiantes
16. Relaxantes musculares
17. Não faz uso

5 Conteúdo Educativo (Saiba Mais)

- Temas: Hipertensão, Diabetes, Cuidados Infantis, Gestação.
- Textos embarcados no HTML (offline).
- Áudios locais (audios/*.mp3) com duração calculada via `loadedmetadata`.
- Vídeos: links YouTube (necessita internet).

6 Componentes React Principais

App

```
+-- PatientLogin
+-- CadastroRapido
+-- PatientDash
|   +-- NovaMensagem
|   \-- PatientHistoryModal
+-- ProAuth
\-- ProDash
    +-- FichaMedica
    +-- MedicationManagementList
    |   \-- MedicationPatientDetail
    +-- MedicationEditorModal
    +-- MonitorConversationModal
    \-- MediaModal
```

6.1 Componentes Reutilizáveis

- **SyncIndicator**: Badge flutuante “Sincronizando X...”.
- **OfflineBanner**: Aviso persistente quando sem conexão.
- **MediaModal**: Expande imagem/vídeo/áudio em tela cheia.
- **MediasDoRegistro**: Carrega midias (offline primeiro).
- **RegistroThumb**: Thumbnail do primeiro anexo.
- **Loading**: Tela de carregamento com emoji .

7 Funções Utilitárias

```
const mascaraCPF = v =>
  v.replace(/\D/g, '')
  .replace(/(\d{3})(\d)/, '$1.$2')
  .replace(/(\d{3})(\d)/, '$1.$2')
  .replace(/(\d{3})(\d{1,2})/, '$1-$2')
  .replace(/(-\d{2})\d+?$/, '$1');
```

```

const mascaraTelefone = v => {
  const nums = v.replace(/\D/g, '');
  if (nums.length <= 10) {
    return nums.replace(/(\d{2})(\d)/, '($1) $2')
      .replace(/(\d{4})(\d)/, '$1-$2');
  }
  return nums.replace(/(\d{2})(\d)/, '($1) $2')
    .replace(/(\d{5})(\d)/, '$1-$2');
};

const mascaraData = v =>
  v.replace(/\D/g, '')
    .replace(/(\d{2})(\d)/, '$1/$2')
    .replace(/(\d{2})(\d)/, '$1/$2')
    .replace(/(\d{4})\d+?$/, '$1');

const calcAge = data => {
  if (!data) return '';
  const [d, m, y] = data.split('/');
  const birth = new Date(`${y}-${m}-${d}`);
  const diff = Date.now() - birth.getTime();
  return Math.floor(diff / 31557600000) + ' anos';
};

const normalizeDateInput = raw => {
  if (!raw) return '';
  const str = String(raw).trim();
  if (str.includes('T')) {
    return normalizeDateInput(str.split('T')[0]);
  }
  if (/^\d{4}-\d{2}-\d{2}$/.test(str)) {
    const [y, m, d] = str.split('-');
    return `${d}/${m}/${y}`;
  }
  return str;
};

const patientUsesMedication = raw => {
  if (!raw) return false;
  const normalized = String(raw).trim().toLowerCase();
  if (/^(nao|não|nenhum|sem)/.test(normalized)) return false;
  return normalized.includes('sim') || normalized.length > 0;
};

function formatDuration(seconds) {
  const total = Math.max(0, Math.floor(seconds));
  const minutes = Math.floor(total / 60);
  const secs = total % 60;
  return `${String(minutes).padStart(2, '0')}:${String(secs).padStart(2, '0')}`;
}

```

8 Dados Estáticos

8.1 Regiões (13)

Santarém, Belterra, Mojuí dos Campos, Alenquer, Curuá, Óbidos, Oriximiná, Terra Santa, Faro, Juruti, Monte Alegre, Almeirim, Prainha.

8.2 UBS (9)

Antônio Evangelista, Boa Esperança, Divinópolis, Márcio Marinho, Haroldo Martins, Maria Bibiana da Silva, Nadime Miranda, Neli Loeblein, Vicente Alves da Silva.

8.3 ACS (92)

Lista completa embutida em `LISTA_ACS` no código (ver `index.html`).

9 Sistema de Badges

- **CADASTRO PENDENTE:** faltam foto, gênero ou hipertensão.
- **CADASTRO OK:** prontuário completo.
- **MEDICAÇÃO PENDENTE:** paciente declara uso mas não há prescrições.
- **MEDICAÇÃO REGISTRADA:** prescrições ativas.
- **NAO DECLAROU USO:** campo vazio/negativo.
- **ATENDIMENTO PENDENTE:** status pendente/em análise.
- **ATENDIMENTO OK:** status respondido.

10 Limitações e Considerações

10.1 Plano Gratuito Supabase

Recurso	Gratuito	Pro (US\$ 25/mês)
Storage	1 GB	100 GB
Bandwidth	2 GB/mês	200 GB/mês
PostgreSQL	500 MB	8 GB

Tabela 1: Limites do Supabase

10.2 Segurança

Login via CPF sem senha, RLS desabilitado e anon key exposta são decisões conscientes para o contexto remoto. Mitigações: CPF é dado público, mídias organizadas por UUID não adivinhável e auditoria via logs Supabase.

10.3 Escalabilidade

Arquivo único HTML e dependência de IndexedDB exigem monitoramento em produção. Falhas de sincronização podem ocorrer se a conexão oscila.

11 Conclusão

O sistema **Tecendo Saúde** demonstra ser viável oferecer telemedicina para regiões com infraestrutura limitada, mantendo:

1. **Offline-first real.**
2. **Arquitetura simples** sem build.
3. **Suporte a múltiplas mídias.**
4. **Prontuário eletrônico completo.**
5. **Chat assíncrono funcional.**

11.1 Trabalhos Futuros

1. Notificações push.
2. Exportação PDF.
3. Integração e-SUS.
4. Gráficos de evolução.
5. Videochamada WebRTC.
6. Modo escuro.