Documentación Técnica - Kaede v3.0

Visión General

Kaede es una Progressive Web App (PWA) construida con tecnologías modernas para ofrecer una experiencia de asistente conversacional con memoria persistente.

Stack Tecnológico

Frontend

- **Next.js 14**: Framework React con App Router
- **TypeScript**: Tipado estático
- **Tailwind CSS**: Estilos utilitarios
- **Lucide React**: Iconografía

Backend

- **Next.js API Routes**: Endpoints serverless
- **Supabase**: Base de datos PostgreSQL + Auth

IA

- **Modo Cloud**: AbacusAI/OpenAI (GPT-5.2, GPT-4o, Claude)
- **Modo PC**: Ollama (modelos locales)

PWA

- **Service Worker**: Caché offline
- **Web App Manifest**: Instalación nativa
- **Web Speech API**: TTS/STT nativo

Estructura del Proyecto

```
kaede_pwa/
└── nextjs_space/
    ├── app/                    # App Router de Next.js
    │   ├── api/
    │   │   └── chat/
    │   │       └── route.ts    # API de chat (Cloud + Ollama)
    │   ├── globals.css         # Estilos globales + temas
    │   ├── layout.tsx          # Layout raíz + metadata PWA
    │   ├── page.tsx            # Página principal
    │   └── sw-register.tsx     # Registro de Service Worker
    │
    ├── components/
    │   ├── chat-input.tsx      # Input con archivos + voz
    │   ├── confirm-dialog.tsx  # Diálogo de confirmación
    │   ├── message-bubble.tsx  # Burbujas de chat + acciones
    │   ├── modal.tsx           # Modal reutilizable
    │   ├── sidebar.tsx         # Navegación lateral
    │   ├── typing-indicator.tsx # Indicador "escribiendo..."
    │   └── views/
    │       ├── ajustes-view.tsx    # Configuración
    │       ├── buffer-view.tsx     # Memoria corto plazo
    │       ├── chat-view.tsx       # Vista principal
    │       ├── historial-view.tsx  # Notas guardadas
    │       └── telarana-view.tsx   # Memorias largo plazo
    │
    ├── hooks/
    │   ├── use-settings.ts     # Estado de configuración
    │   ├── use-supabase.ts     # Hooks para DB
    │   └── use-voice.ts        # STT/TTS
    │
    ├── lib/
    │   ├── backup.ts           # Exportar/importar datos
    │   ├── constants.ts        # System prompt + defaults
    │   ├── context-builder.ts  # Construcción de contexto IA
    │   ├── database.types.ts   # Tipos de Supabase
    │   ├── supabase.ts         # Cliente Supabase
    │   └── utils.ts            # Utilidades
    │
    ├── public/
    │   ├── avatar.jpg          # Avatar de Kaede
    │   ├── icon.png            # Ícono PWA 512x512
    │   ├── favicon.svg         # Favicon
    │   ├── manifest.json       # Web App Manifest
    │   ├── offline.html        # Página offline
    │   └── sw.js               # Service Worker
    │
    └── prisma/
        └── schema.prisma       # (No usado - se usa Supabase directo)
```

Base de Datos (Supabase)

Tabla: `messages`

Almacena el historial de conversación.

```sql
CREATE TABLE messages (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id TEXT NOT NULL,
  role TEXT NOT NULL CHECK (role IN ('user', 'assistant')),
  content TEXT NOT NULL,
  is_in_buffer BOOLEAN DEFAULT true,
  created_at TIMESTAMPTZ DEFAULT now()
);
```

Tabla: `memories`

Memoria a largo plazo estructurada.

```sql
CREATE TABLE memories (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id TEXT NOT NULL,
  type TEXT NOT NULL CHECK (type IN ('core', 'identity', 'experience')),
  content TEXT NOT NULL,
  importance INTEGER DEFAULT 5 CHECK (importance >= 1 AND importance <= 10),
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);
```

Tabla: `saved_notes`

Notas guardadas del historial.

```sql
CREATE TABLE saved_notes (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id TEXT NOT NULL,
  message_id UUID REFERENCES messages(id) ON DELETE SET NULL,
  content TEXT NOT NULL,
  created_at TIMESTAMPTZ DEFAULT now()
);
```

API de Chat

Endpoint: `POST /api/chat`

Request:

```typescript
interface ChatRequest {
  messages: Array<{
    role: 'user' | 'assistant';
    content: string;
  }>;
  model?: string;       // Modelo específico
  mode?: 'cloud' | 'pc'; // Modo de conexión
  ollamaUrl?: string;    // URL de Ollama (solo PC)
}
```

Response: Server-Sent Events (streaming)

Flujo:
1. Recibe mensajes + configuración
2. Construye contexto con memorias (context-builder.ts)
3. Si modo=cloud → llama a AbacusAI/OpenAI
4. Si modo=pc → llama a Ollama local
5. Retorna respuesta en streaming

Sistema de Memoria

Contexto Builder (`lib/context-builder.ts`)

```
function buildContext(
  memories: Memory[],
  bufferMessages: Message[],
  maxTokens: number
): string
```

Prioridad de inclusión:
1. System prompt (KAEDE_SYSTEM_PROMPT)
2. Memorias tipo “core” (siempre)
3. Memorias tipo “identity”
4. Memorias tipo “experience” (más recientes)
5. Buffer de conversación (recientes primero)

Sistema de Voz

Hook: `useVoice()`

```
interface UseVoiceReturn {
  // STT (Speech-to-Text)
  isListening: boolean;
  transcript: string;
  startListening: () => void;
  stopListening: () => void;
  sttSupported: boolean;

  // TTS (Text-to-Speech)
  isSpeaking: boolean;
  speak: (text: string) => void;
  stopSpeaking: () => void;
  ttsSupported: boolean;
  voices: SpeechSynthesisVoice[];
}
```

Configuración de voz estilo Ukyo:
- Idioma: `es-MX` (español mexicano)
- Rate: `1.1` (ligeramente rápido)
- Pitch: `1.15` (tono juvenil)
- Prioridad de voces: Mexicana femenina > Mexicana > Española

Sistema de Temas

Variables CSS (`globals.css`)

```
:root {
  --color-primary: #0B1F3B;     /* Navy */
  --color-background: #FFF6E9;  /* Ivory */
  --color-accent: #C4473D;      /* Brick Red */
  --color-text: #1C1C1C;
  --color-text-muted: #6B6B6B;
  --color-border: #9FB3C8;
  --color-success: #22C55E;
}

.dark {
  --color-primary: #1E3A5F;
  --color-background: #0F172A;
  --color-accent: #EF4444;
  --color-text: #F1F5F9;
  --color-text-muted: #94A3B8;
  --color-border: #334155;
  --color-success: #22C55E;
}
```

Hook: `useSettings()`

```
type ThemeMode = 'light' | 'dark' | 'system';

function getEffectiveTheme(): 'light' | 'dark';
```

Sistema de Respaldo

Formato de Backup (`lib/backup.ts`)

```
interface BackupData {
  version: string;              // "1.0"
  exportedAt: string;           // ISO date
  memories: Array<{
    type: 'core' | 'identity' | 'experience';
    content: string;
    importance: number;
  }>;
  savedNotes: Array<{
    content: string;
    createdAt: string;
  }>;
}
```

Funciones:

- `createBackup()` : Genera objeto BackupData
- `downloadBackup()` : Descarga archivo JSON

- `validateBackup()` : Valida formato
- `readBackupFile()` : Lee archivo subido

PWA Configuration

Manifest (`public/manifest.json`)

```
{
  "name": "Kaede - Tu Asistente Personal",
  "short_name": "Kaede",
  "display": "standalone",
  "background_color": "#FFF6E9",
  "theme_color": "#0B1F3B",
  "icons": [...]
}
```

Service Worker (`public/sw.js`)

- Estrategia: Cache-first para assets estáticos
- Fallback: offline.html cuando no hay red

Variables de Entorno

```
# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://xxx.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJ...
SUPABASE_SERVICE_ROLE_KEY=eyJ...

# IA Cloud (AbacusAI)
ABACUSAI_API_KEY=xxx
```

Despliegue

Abacus.AI (Actual)

La app está desplegada en: `https://kaede.abacusai.app`

Vercel (Alternativa)

Requisitos:
1. Cuenta de Vercel
2. Repositorio Git con el código
3. Variables de entorno configuradas

Pasos:
1. Fork/Push del código a GitHub
2. Importar proyecto en Vercel

3. Configurar variables de entorno
4. Deploy

Nota: El endpoint de IA (`/api/chat`) usa AbacusAI por defecto. Para Vercel, necesitarías configurar tu propia API key de OpenAI o modificar el endpoint.

Extender la App

Agregar nuevo tipo de memoria

1. Modificar `lib/database.types.ts`
2. Actualizar `telarana-view.tsx`
3. Ajustar `context-builder.ts`

Agregar nuevo modelo de IA

1. Agregar a `CLOUD_MODELS` o `OLLAMA_MODELS` en `constants.ts`
2. El API route lo manejará automáticamente

Personalizar la voz

1. Modificar `use-voice.ts`
2. Ajustar `voiceRate` y `voicePitch` en `use-settings.ts`
