

DOCUMENTACIÓN TÉCNICA - SPEAKLYPLAN

Arquitectura Completa del Sistema

Fecha de última actualización: 09 de Octubre de 2025

Versión: 1.0

Estado: Producción



ÍNDICE

1. [Resumen Ejecutivo](#)
 2. [Stack Tecnológico](#)
 3. [Arquitectura de Base de Datos](#)
 4. [Estructura del Proyecto](#)
 5. [Sistema de Autenticación](#)
 6. [Módulos Principales](#)
 7. [Sistema de Inteligencia Artificial](#)
 8. [API Endpoints](#)
 9. [Sistema de Gamificación](#)
 10. [Componentes UI Reutilizables](#)
 11. [Variables de Entorno](#)
 12. [Flujos de Usuario](#)
 13. [Consideraciones Importantes](#)
 14. [Roadmap y TODOs](#)
-

1. RESUMEN EJECUTIVO

SpeaklyPlan es una aplicación web educativa diseñada para ayudar a profesionales de tecnología (especialmente CTOs) a aprender inglés en un período de 6 meses. La aplicación combina un plan estructurado de aprendizaje con un tutor de IA conversacional, sistema de gamificación, y seguimiento de progreso.

Características Principales:

- ☒ Plan estructurado de 26 semanas de aprendizaje
 - ☒ Tutor de IA conversacional con corrección gramatical en tiempo real
 - ☒ Sistema de gamificación (puntos, niveles, logros, rachas)
 - ☒ Seguimiento de progreso personalizado
 - ☒ Sistema de vocabulario con repetición espaciada (SM-2)
 - ☒ Recursos educativos curados
 - ☒ Guía de uso interactiva
-

2. STACK TECNOLÓGICO

Frontend

```
{  
  "framework": "Next.js 14.2.28",  
  "ui_library": "React 18.2.0",  
  "styling": "Tailwind CSS 3.3.3",  
  "ui_components": "Radix UI + Shadcn/ui",  
  "animations": "Framer Motion 10.18.0",  
  "state_management": "Zustand 5.0.3 + React Query 5.0.0",  
  "forms": "React Hook Form 7.53.0 + Yup 1.3.0",  
  "charts": "Recharts 2.15.3",  
  "icons": "Lucide React 0.446.0"  
}
```

Backend

```
{  
  "runtime": "Node.js",  
  "framework": "Next.js API Routes",  
  "orm": "Prisma 6.7.0",  
  "database": "PostgreSQL",  
  "auth": "NextAuth.js 4.24.11",  
  "ai_provider": "Abacus AI (OpenAI compatible)",  
  "password_hashing": "bcryptjs 2.4.3"  
}
```

Infraestructura

```
{  
  "database_host": "Abacus AI Hosted DB",  
  "deployment": "Abacus AI Platform",  
  "package_manager": "Yarn"  
}
```

3. ARQUITECTURA DE BASE DE DATOS

Diagrama de Relaciones

```

User (Usuario Principal)
├── Account (NextAuth)
├── Session (NextAuth)
├── UserProgress (Progreso de actividades)
├── UserNote (Notas y reflexiones)
├── UserStreak (Rachas de estudio)
├── UserVocabularyProgress (Progreso de vocabulario)
├── UserAchievement (Logros desbloqueados)
├── ChatConversation (Conversaciones con tutor IA)
├── LearningContext (Contexto de aprendizaje)
├── CommonMistake (Errores frecuentes)
├── PracticeSession (Sesiones de práctica)
├── VocabularyCard (Tarjetas de vocabulario SRS)
└── SessionAnalytics (Analíticas detalladas)

PlanPhase (Fase del plan)
├── PlanWeek (Semana del plan)
│   └── PlanActivity (Actividad diaria)
│       └── UserProgress
VocabularyCategory (Categoría de vocabulario)
├── VocabularyTerm (Término de vocabulario)
│   └── UserVocabularyProgress
ResourceCategory (Categoría de recursos)
├── Resource (Recurso educativo)
Achievement (Definición de logros)
├── UserAchievement
ChatConversation (Conversación)
├── ChatMessage (Mensaje)

```

Modelos Clave

User (Usuario)

```

model User {
  id          String    @id @default(cuid())
  name        String?
  email       String    @unique
  password    String?
  role        String    @default("user")

  // Gamificación
  points      Int        @default(0)
  level       Int        @default(1)
  currentStreak Int      @default(0)
  bestStreak  Int        @default(0)
  lastActiveDate DateTime?

  createdAt   DateTime   @default(now())
  updatedAt   DateTime   @updatedAt
}

```

ChatConversation (Conversaciones con tutor)

```
model ChatConversation {
  id          String          @id @default(cuid())
  userId      String
  title       String?
  context     String?         // "casual", "meeting", "interview", "email", "grammar"
  startedAt   DateTime        @default(now())
  lastMessageAt DateTime      @default(now())
  isActive    Boolean         @default(true)
  messages    ChatMessage[]
}
```

LearningContext (Perfil de aprendizaje)

```
model LearningContext {
  id          String          @id @default(cuid())
  userId      String          @unique
  currentLevel String         @default("A1") // CEFR level
  weakAreas   Json            @default("[]")
  strongAreas Json            @default("[]")
  preferredTopics Json        @default("[]")
  learningGoals Json          @default("[]")
  totalConversations Int       @default(0)
  totalMessages Int           @default(0)
}
```

VocabularyCard (Sistema de repetición espaciada)

```
model VocabularyCard {
  id          String          @id @default(cuid())
  userId      String
  word        String
  translation  String
  context     String?         @db.Text
  difficulty   Int             @default(0) // SM-2 0 = very hard, 5 = very easy

  // Algoritmo SM-2
  easeFactor   Float           @default(2.5)
  interval     Int             @default(0) // días hasta próxima revisión
  repetitions   Int            @default(0)
  nextReviewDate DateTime      @default(now())

  lastReviewedAt DateTime?
  createdAt     DateTime        @default(now())
}
```

4. ESTRUCTURA DEL PROYECTO

```
/home/ubuntu/speaklyplan/nextjs_space/
```

```

├── app/                                     # App Router de Next.js
│   ├── api/                               # API Routes
│   │   ├── auth/
│   │   │   ├── [...nextauth]/route.ts    # NextAuth configuración
│   │   │   ├── progress/route.ts         # Progreso de actividades
│   │   │   ├── signup/route.ts          # Registro de usuarios
│   │   │   └── tutor/
│   │   │       ├── analytics/route.ts    # Analíticas del tutor
│   │   │       ├── chat/route.ts        # Chat con tutor IA
│   │   │       ├── context/route.ts     # Contexto de aprendizaje
│   │   │       ├── gamification/route.ts # Sistema de gamificación
│   │   │       ├── history/route.ts     # Historial de conversaciones
│   │   │       └── vocabulary-cards/route.ts # Tarjetas de vocabulario
│   │   └── vocabulary/
│   │       └── progress/route.ts         # Progreso de vocabulario
│   ├── auth/                             # Páginas de autenticación
│   │   ├── login/page.tsx
│   │   └── register/page.tsx
│   ├── dashboard/                        # Panel principal
│   │   ├── components/
│   │   │   ├── dashboard-client.tsx
│   │   │   ├── progress-overview.tsx
│   │   │   ├── week-view.tsx
│   │   └── page.tsx
│   ├── tutor/                           # Tutor de IA
│   │   ├── components/
│   │   │   ├── tutor-client.tsx
│   │   │   └── vocabulary-review/
│   │   │       ├── components/
│   │   │       │   ├── vocabulary-review-client.tsx
│   │   │       └── page.tsx
│   │   └── page.tsx
│   ├── vocabulario/                     # Sistema de vocabulario
│   │   ├── components/
│   │   │   ├── vocabulario-client.tsx
│   │   └── page.tsx
│   ├── guia/                           # Guía de uso
│   │   ├── components/
│   │   │   ├── guia-client.tsx
│   │   └── page.tsx
│   ├── recursos/                       # Recursos educativos
│   │   ├── components/
│   │   │   ├── recursos-client.tsx
│   │   └── page.tsx
│   ├── layout.tsx                      # Layout principal
│   ├── page.tsx                        # Landing page
│   ├── providers.tsx                   # Providers (SessionProvider)
│   └── globals.css                     # Estilos globales
├── components/                          # Componentes reutilizables
│   ├── gamification/                   # Componentes de gamificación
│   │   ├── achievement-card.tsx
│   │   └── activity-completion-celebration.tsx

```

```

├── daily-missions.tsx
├── gamification-panel.tsx
├── level-badge.tsx
├── level-up-modal.tsx
├── progress-ring.tsx
├── streak-display.tsx
├── xp-toast.tsx
├── ui/
│   └── # Shadcn/ui components
│       ├── button.tsx
│       ├── card.tsx
│       ├── dialog.tsx
│       ├── input.tsx
│       ├── select.tsx
│       ├── toast.tsx
│       └── ... (50+ componentes UI)
├── theme-provider.tsx
│   └── # Provider de temas
├── lib/
│   └── # Utilidades y servicios
│       ├── ai/
│       │   └── # Servicios de IA
│       │       ├── analytics-service.ts
│       │       │   └── # Analíticas del tutor
│       │       ├── gamification-service.ts
│       │       │   └── # Lógica de gamificación
│       │       ├── prompts.ts
│       │       │   └── # Prompts del tutor IA
│       │       ├── spaced-repetition.ts
│       │       │   └── # Algoritmo SM-2
│       │       └── tutor-service.ts
│       │           └── # Servicio principal del tutor
│       ├── auth.ts
│       │   └── # Configuración NextAuth
│       ├── db.ts
│       │   └── # Cliente Prisma
│       ├── types.ts
│       │   └── # Tipos TypeScript
│       └── utils.ts
│           └── # Utilidades generales
├── prisma/
│   └── schema.prisma
│       └── # Schema de base de datos
├── .env
│   └── # Variables de entorno
├── next.config.js
│   └── # Configuración de Next.js
├── tailwind.config.ts
│   └── # Configuración de Tailwind
├── tsconfig.json
│   └── # Configuración de TypeScript
└── package.json
    └── # Dependencias

```

5. SISTEMA DE AUTENTICACIÓN

Configuración (lib/auth.ts)

```
// Usa NextAuth.js con estrategia JWT
// Proveedor: CredentialsProvider (email/password)
// Adapter: PrismaAdapter
// Hashing: bcrypt con 10 salt rounds

export const authOptions: AuthOptions = {
  adapter: PrismaAdapter(prisma),
  providers: [
    CredentialsProvider({
      credentials: {
        email: { label: 'Email', type: 'email' },
        password: { label: 'Password', type: 'password' }
      },
      async authorize(credentials) {
        // Validación de credenciales
        // Retorna usuario o null
      }
    })
  ],
  session: { strategy: 'jwt' },
  callbacks: {
    jwt: // Añade role al token
    session: // Añade id y role a la sesión
  },
  pages: {
    signIn: '/auth/login'
  }
}
```

Protección de Rutas

```
// En componentes de servidor
const session = await getSession(authOptions);
if (!session) redirect('/auth/login');

// En API routes
const session = await getSession(authOptions);
if (!session?.user?.id) {
  return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
}

// En componentes de cliente
'use client'
import { useSession } from 'next-auth/react'

const { data: session, status } = useSession() || {};
if (status === 'loading') return <Loading />;
if (!session) redirect('/auth/login');
```


6. MÓDULOS PRINCIPALES

6.1 Dashboard (Panel Principal)

Ubicación: `app/dashboard/`

Funcionalidad:

- Vista general del progreso del usuario
- Calendario semanal con actividades
- Resumen de rachas y logros
- Acceso rápido a todas las secciones

Componentes principales:

- `dashboard-client.tsx` - Componente principal del dashboard
- `progress-overview.tsx` - Resumen de progreso
- `week-view.tsx` - Vista de semana actual

Datos mostrados:

- Progreso total (% completado)
- Semana actual
- Racha actual y mejor racha
- Actividades de la semana
- Logros recientes

6.2 Tutor de IA

Ubicación: `app/tutor/`

Funcionalidad:

- Chat conversacional con IA
- Corrección gramatical en tiempo real
- Traducción para principiantes (A1-A2)
- Múltiples contextos: casual, meeting, interview, email, grammar
- Historial de conversaciones
- Sistema de gamificación integrado

Características del Tutor:

- Modelo: GPT-4o-mini (via Abacus AI)
- Adaptación al nivel CEFR del usuario (A1-C2)
- Detección de errores gramaticales
- Sugerencias de vocabulario
- Feedback constructivo

Prompts del Sistema:

```
// lib/ai/prompts.ts

getTutorSystemPrompt(context)
// Define personalidad y comportamiento del tutor

getContextualPrompt(context, vocabulary)
// Prompt específico para cada tipo de conversación

getGrammarAnalysisPrompt(text, level)
// Análisis gramatical con feedback

getTranslationPrompt(text)
// Traducción al español
```

Flujo de Conversación:

1. Usuario envía mensaje
2. Se guarda en DB (ChatMessage)
3. Se obtiene contexto de aprendizaje (LearningContext)
4. Se genera respuesta con IA
5. Se analiza gramática (si nivel A1-B1)
6. Se traduce (si nivel A1-A2)
7. Se guarda respuesta en DB
8. Se actualizan métricas
9. Se actualizan errores comunes

6.3 Sistema de Vocabulario

Ubicación: app/vocabulario/

Funcionalidad:

- Exploración de vocabulario por categorías
- Sistema de repetición espaciada (SM-2)
- Progreso de palabras aprendidas
- Tarjetas de vocabulario personalizadas

Categorías de Vocabulario:

1. Comunicación Profesional
2. Tecnología y Software
3. Reuniones y Presentaciones
4. Gestión de Proyectos
5. Liderazgo y Gestión
6. Finanzas y Negocios
7. Networking y Eventos
8. Email y Comunicación Escrita

Algoritmo SM-2:

```
// lib/ai/spaced-repetition.ts

calculateNextReview(quality, easeFactor, interval, repetitions)
// Calcula próxima revisión basada en:
// - quality: 0-5 (calidad de respuesta)
// - easeFactor: dificultad de la tarjeta (>=1.3)
// - interval: días hasta próxima revisión
// - repetitions: número de repeticiones

// Fórmula:
// newEF = oldEF + (0.1 - (5-q) * (0.08 + (5-q)*0.02))
// newInterval = oldInterval * newEF
```

6.4 Recursos Educativos

Ubicación: app/recursos/

Funcionalidad:

- Curación de recursos externos
- Categorización por tipo
- Filtros por nivel y categoría
- Ratings y recomendaciones

Categorías de Recursos:

1. Cursos Online
2. Podcasts
3. Canales de YouTube
4. Apps Móviles
5. Libros y eBooks
6. Sitios Web
7. Comunidades
8. Herramientas

6.5 Guía de Uso

Ubicación: app/guia/

Funcionalidad:

- Tutorial interactivo
- Checklist de primeros pasos
- Quiz de objetivos personalizados
- Planificador semanal interactivo
- Gráficos de progreso
- Videos demostrativos
- Tips contextuales animados
- Tarjetas interactivas con flip
- Tour guiado de la plataforma

Características especiales:

- Animaciones con Framer Motion
 - Integración con canvas-confetti para celebraciones
 - Diseño responsive
 - Interactividad completa
-

7. SISTEMA DE INTELIGENCIA ARTIFICIAL

7.1 Tutor Service

Archivo: lib/ai/tutor-service.ts

Funciones principales:

```
// Generar respuesta del tutor
async function generateTutorResponse({
  userMessage: string,
  conversationHistory: Message[],
  learningContext: LearningContextType,
  context: string,
  vocabulary: VocabularyTerm[]
}): Promise<{
  content: string,
  vocabularyUsed: string[]
}>

// Analizar gramática
async function analyzeGrammar(
  text: string,
  level: string
): Promise<{
  errors: Array<{
    type: string,
    original: string,
    correction: string,
    explanation: string
  }>,
  feedback: {
    hasErrors: boolean,
    suggestion: string
  }
}>

// Traducir al español
async function translateToSpanish(
  text: string
): Promise<string>
```

Configuración API:

```
const API_ENDPOINT = 'https://apps.abacus.ai/v1/chat/completions'
const API_KEY = process.env.ABACUSAI_API_KEY
const MODEL = 'gpt-4o-mini'
const TEMPERATURE = 0.7
const MAX_TOKENS = 300
```

7.2 Prompts Inteligentes

Archivo: lib/ai/prompts.ts

Los prompts están diseñados para:

- Adaptar el nivel de complejidad al usuario
- Proporcionar feedback constructivo
- Mantener conversaciones naturales

- Enfocarse en contextos profesionales
- Introducir vocabulario relevante

Niveles CEFR soportados:

- A1 (Principiante)
- A2 (Elemental)
- B1 (Intermedio)
- B2 (Intermedio-Alto)
- C1 (Avanzado)
- C2 (Maestría)

7.3 Analytics Service

Archivo: `lib/ai/analytics-service.ts`

Funciones para análisis de sesiones:

- Cálculo de scores de fluidez
- Análisis de diversidad de vocabulario
- Métricas de precisión gramatical
- Identificación de áreas de mejora
- Generación de feedback personalizado

8. API ENDPOINTS

8.1 Autenticación

```
// POST /api/auth/[...nextauth]
// GET  /api/auth/[...nextauth]
// Maneja autenticación con NextAuth

// POST /api/signup
// Body: { email, password, name }
// Response: { success: boolean, message: string }
```

8.2 Tutor de IA

```

// POST /api/tutor/chat
// Body: {
//   message: string,
//   conversationId?: string,
//   context: string,
//   userId: string
// }
// Response: {
//   messageId: string,
//   conversationId: string,
//   content: string,
//   translation?: string,
//   grammarFeedback: Object,
//   vocabularyUsed: string[]
// }

// GET /api/tutor/chat?conversationId={id}
// Response: { conversation: Conversation }

// GET /api/tutor/chat
// Response: { conversations: Conversation[] }

// GET /api/tutor/history
// Response: { conversations: Conversation[] }

// GET /api/tutor/context
// Response: { context: LearningContext }

// POST /api/tutor/context
// Body: { updates: Partial<LearningContext> }
// Response: { success: boolean, context: LearningContext }

// GET /api/tutor/analytics
// Response: {
//   sessions: SessionAnalytics[],
//   summary: {
//     totalSessions: number,
//     averageDuration: number,
//     overallScore: number,
//     strengths: string[],
//     areasToImprove: string[]
//   }
// }

// POST /api/tutor/gamification
// Body: { action: string, metadata?: any }
// Response: {
//   points: number,
//   level: number,
//   streak: number,
//   achievements: Achievement[]
// }

// GET /api/tutor/vocabulary-cards
// Response: { cards: VocabularyCard[] }

// POST /api/tutor/vocabulary-cards
// Body: { word: string, translation: string, context?: string }
// Response: { card: VocabularyCard }

// PUT /api/tutor/vocabulary-cards/{id}

```

```
// Body: { quality: number } // 0-5 para SM-2
// Response: { card: VocabularyCard }
```

8.3 Progreso

```
// GET /api/progress
// Response: {
//   totalActivities: number,
//   completedActivities: number,
//   currentWeek: number,
//   currentStreak: number,
//   bestStreak: number,
//   percentageCompleted: number
// }

// POST /api/progress
// Body: { activityId: string }
// Response: { success: boolean, progress: UserProgress }
```

8.4 Vocabulario

```
// GET /api/vocabulary/progress
// Response: {
//   total: number,
//   mastered: number,
//   learning: number,
//   percentage: number
// }

// POST /api/vocabulary/progress
// Body: { wordId: string, mastered: boolean }
// Response: { success: boolean, progress: UserVocabularyProgress }
```

9. SISTEMA DE GAMIFICACIÓN

9.1 Componentes

Archivo: lib/ai/gamification-service.ts

Sistema de Puntos:

```
const POINTS = {
  MESSAGE_SENT: 5,
  SESSION_COMPLETED: 20,
  PERFECT_GRAMMAR: 50,
  NEW_WORD_LEARNED: 10,
  VOCABULARY_REVIEWED: 5,
  DAILY_GOAL_ACHIEVED: 30,
}
```

Sistema de Niveles:


```
// Fórmula: level = floor(sqrt(points / 100)) + 1
function calculateLevel(points: number): number {
    return Math.floor(Math.sqrt(points / 100)) + 1;
}

// Puntos necesarios para el siguiente nivel
function pointsForNextLevel(currentLevel: number): number {
    return Math.pow(currentLevel, 2) * 100;
}
```

Progresión de Niveles:

- Nivel 1: 0 puntos
- Nivel 2: 100 puntos
- Nivel 3: 400 puntos
- Nivel 4: 900 puntos
- Nivel 5: 1,600 puntos
- Nivel 10: 10,000 puntos
- Nivel 20: 40,000 puntos

9.2 Logros (Achievements)

Categorías:

1. Rachas (Streak)

- En Racha (3 días) - 50 pts
- Una Semana Completa (7 días) - 150 pts
- Dedicación Total (30 días) - 500 pts

1. Mensajes (Messages)

- Conversador (10 mensajes) - 30 pts
- Charlatán (100 mensajes) - 200 pts
- Maestro de la Conversación (500 mensajes) - 1000 pts

2. Sesiones (Sessions)

- Principiante Dedicado (5 sesiones) - 50 pts
- Estudiante Consistente (25 sesiones) - 250 pts
- Experto en Práctica (100 sesiones) - 1000 pts

3. Gramática (Grammar)

- Gramática Perfecta (1 sesión sin errores) - 100 pts
- Maestro de la Gramática (10 sesiones sin errores) - 500 pts

4. Vocabulario (Vocabulary)

- Constructor de Vocabulario (50 palabras) - 200 pts
- Políglota en Desarrollo (200 palabras) - 800 pts

9.3 Sistema de Rachas

```

async function updateStreak(userId: string): Promise<{
  currentStreak: number,
  bestStreak: number,
  streakMaintained: boolean
}>

// Lógica:
// - Si es el mismo día: mantiene racha
// - Si es el día siguiente: incrementa racha
// - Si pasó más de 1 día: reinicia racha a 1
// - Actualiza bestStreak si currentStreak > bestStreak

```

9.4 Componentes UI de Gamificación

Ubicación: components/gamification/

- **achievement-card.tsx** - Tarjeta de logro desbloqueado
- **activity-completion-celebration.tsx** - Animación de completar actividad
- **daily-missions.tsx** - Misiones diarias
- **gamification-panel.tsx** - Panel completo de gamificación
- **level-badge.tsx** - Badge de nivel del usuario
- **level-up-modal.tsx** - Modal de subida de nivel
- **progress-ring.tsx** - Anillo de progreso circular
- **streak-display.tsx** - Visualización de racha
- **xp-toast.tsx** - Notificación de XP ganados

10. COMPONENTES UI REUTILIZABLES

10.1 Shadcn/ui Components

Ubicación: components/ui/

La aplicación utiliza **Shadcn/ui**, una colección de componentes construidos con:

- Radix UI (primitivas accesibles)
- Tailwind CSS (estilos)
- TypeScript (tipado fuerte)

Componentes disponibles (50+):

- Accordion, Alert, Avatar, Badge, Button
- Calendar, Card, Carousel, Checkbox
- Command, Context Menu, Dialog, Drawer
- Dropdown Menu, Form, Hover Card, Input
- Label, Menubar, Navigation Menu
- Pagination, Popover, Progress, Radio Group
- Resizable, Scroll Area, Select, Separator
- Sheet, Skeleton, Slider, Switch
- Table, Tabs, Textarea, Toast, Toggle
- Tooltip

10.2 Uso de Select (IMPORTANTE)

REGLA CRÍTICA: NUNCA usar valores vacíos o inválidos en `<Select>`

```
// ✗ INCORRECTO - Causará errores
<SelectedItem value="">Seleccionar</SelectedItem>
<SelectedItem value={undefined}>Opción</SelectedItem>

// ✓ CORRECTO - Siempre usar strings válidos
<SelectedItem value="all">Todos</SelectedItem>
<SelectedItem value="no-filter">Sin filtro</SelectedItem>

// Para valores dinámicos, usar fallback
<Select value={category || "all"}>
  {categories?.map(cat => (
    <SelectedItem key={cat.id} value={cat.id || `fallback-${cat.name}`}>
      {cat.name}
    </SelectedItem>
  ))}
</Select>
```

10.3 Prevención de Errores de Hidratación

Reglas importantes:

1. No usar `Math.random()`, `Date.now()` en el render inicial
2. No usar `new Date().toLocaleString()` directamente
3. Envolver lógica del navegador en `useEffect`
4. Usar `getServerSideProps` para datos dinámicos

```
// ✗ INCORRECTO - Causará error de hidratación
function Component() {
  const [date] = useState(new Date().toLocaleDateString());
  return <div>{date}</div>;
}

// ✓ CORRECTO - Usar props desde servidor o useEffect
function Component({ serverDate }: { serverDate: string }) {
  return <div>{serverDate}</div>;
}

// O usar useEffect para cliente
function Component() {
  const [date, setDate] = useState('');
  useEffect(() => {
    setDate(new Date().toLocaleDateString());
  }, []);
  return <div>{date || 'Cargando...'}</div>;
}
```

10.4 Manejo de Imágenes con Next.js

Siempre usar el componente `Image` de Next.js:

```
import Image from 'next/image'

// Con aspecto ratio fijo
<div className="relative aspect-video bg-muted rounded-lg overflow-hidden">
  <Image
    src="/images/photo.jpg"
    alt="Descripción específica y significativa"
    fill
    className="object-cover"
    onError={(e) => {
      e.currentTarget.src = '/images/fallback.jpg';
    }}
  />
</div>

// IMPORTANTE:
// - Siempre usar contenedor con aspect ratio
// - Siempre incluir clase 'relative' en el contenedor al usar fill
// - Siempre incluir alt descriptivo (no genérico)
// - Implementar manejo de errores
```

11. VARIABLES DE ENTORNO

Archivo: `.env`

```
# Base de Datos PostgreSQL (Abacus AI Hosted)
DATABASE_URL="postgresql://role_f7dfd0c44:bmSuaocSJKxYSqWigBD5ZpRJfieebf7@db-f7df-d0c44.db002.hosteddb.reai.io:5432/f7dfd0c44?connect_timeout=15"



# NextAuth.js
NEXTAUTH_SECRET="uZ0sEmc0W56hBzgqzJ3Ut0MSaXG6JILQ"
NEXTAUTH_URL="http://localhost:3000" # En producción: URL del dominio

# Abacus AI API (para tutor de IA)
ABACUSAI_API_KEY="6d20e7d7b5b14d3a80cac4a202928078"
```

CRÍTICO: Estas variables NO deben modificarse al continuar el desarrollo. Son esenciales para el funcionamiento de la aplicación.

12. FLUJOS DE USUARIO

12.1 Flujo de Registro y Login

1. Usuario visita landing page (/)
2. Click en "Comenzar Ahora"  /auth/register
3. Completa formulario (email, password, name)
4. POST /api/signup
5. Si éxito  Redirect a /auth/login
6. Login con credenciales
7. NextAuth valida y crea sesión JWT
8. Redirect a /dashboard

12.2 Flujo del Tutor de IA

1. Usuario navega a /tutor
2. Selecciona **contexto** (casual, meeting, **interview**, etc.)
3. Escribe mensaje en inglés
4. Click "Enviar"
5. **POST** /api/tutor/chat con `{ message, conversationId, context, userId }`
6. Backend:
 - a. Obtiene historial de conversación
 - b. Obtiene LearningContext del usuario
 - c. Obtiene vocabulario relevante
 - d. Construye prompts (**system** + **contextual**)
 - e. Llama a Abacus AI API
 - f. Analiza gramática (si nivel A1-B1)
 - g. Traduce respuesta (si nivel A1-A2)
 - h. Guarda mensaje del usuario y respuesta del asistente
 - i. Actualiza errores comunes
 - j. Actualiza métricas
7. Frontend recibe y muestra:
 - Respuesta del tutor
 - Traducción (si aplica)
 - Feedback gramatical (si hay errores)
 - Vocabulario usado
8. Se actualizan puntos y rachas en tiempo real
9. Si desbloquea **logro** `{ }` Modal de celebración

12.3 Flujo de Completar Actividad

1. Usuario en /dashboard ve actividades de la semana
2. Click en checkbox de actividad
3. **POST** /api/progress con `{ activityId }`
4. Backend:
 - a. Verifica autenticación
 - b. Crea/actualiza UserProgress
 - c. Marca como completado
 - d. Actualiza racha
 - e. Otorga puntos
 - f. Verifica **logros**
5. Frontend:
 - a. Actualiza UI (checkbox marcado)
 - b. Muestra animación de celebración
 - c. **Toast** con XP ganados
 - d. Si subió de nivel `{ }` Modal de Level Up
 - e. Si desbloqueó **logro** `{ }` Modal de Achievement

12.4 Flujo de Revisión de Vocabulario (SM-2)

1. Usuario navega a /tutor/vocabulary-review
2. **GET** /api/tutor/vocabulary-cards
3. Backend filtra tarjetas con **nextReviewDate** <= now
4. Frontend muestra tarjeta actual
5. Usuario revela traducción
6. Usuario evalúa dificultad (0-5):
 - 0: No recordé nada
 - 1: Muy difícil
 - 2: Difícil
 - 3: Bien
 - 4: Fácil
 - 5: Muy fácil
7. **PUT** /api/tutor/vocabulary-cards/{id} con { quality }
8. Backend aplica algoritmo SM-2:
 - Calcula nuevo easeFactor
 - Calcula nuevo **interval**
 - Calcula **nextReviewDate**
 - Actualiza tarjeta en DB
9. Frontend muestra siguiente tarjeta
10. Al finalizar sesión:
 - Muestra resumen
 - Otorga puntos
 - Actualiza progreso

13. CONSIDERACIONES IMPORTANTES

13.1 Rendimiento

1. Lazy Loading de Componentes

```
typescript
const HeavyComponent = dynamic(() => import('./HeavyComponent'), {
  loading: () => <Skeleton />,
  ssr: false
});
```

2. Optimización de Imágenes

- Usar Next.js Image component
- Implementar placeholders
- Lazy loading automático

3. Caching de API

- Usar React Query para caching
- Implementar revalidación inteligente
- Cache de conversaciones del tutor

13.2 Seguridad

1. Validación de Input

- Validar todos los inputs del usuario
- Sanitizar mensajes del chat
- Prevenir SQL injection (Prisma lo hace automáticamente)

2. Autenticación

- Verificar sesión en cada API route
- Usar JWT con expiración
- Hash de passwords con bcrypt (10 rounds)

3. Rate Limiting

- Implementar límite de mensajes por minuto
- Límite de creación de tarjetas de vocabulario
- Protección contra spam

13.3 Accesibilidad

1. Navegación por Teclado

- Todos los componentes son accesibles por teclado
- Orden de tabulación lógico
- Focus visible

2. ARIA Labels

- Todos los botones tienen labels descriptivos
- Roles ARIA apropiados
- Live regions para notificaciones

3. Contraste de Colores

- Cumple WCAG AA en modo claro y oscuro
- Texto legible en todos los fondos

13.4 Errores Comunes a Evitar

1. Select con valores vacíos

```
```typescript
// ❌ NUNCA hacer esto
Opción
Opción

// ✅ SIEMPRE hacer esto
Opción
```
```

1. Errores de hidratación

```
```typescript
// ❌ NUNCA hacer esto
const [date] = useState(new Date().toLocaleDateString());

// ✅ SIEMPRE hacer esto
const [date, setDate] = useState("");
useEffect(() => {
 setDate(new Date().toLocaleDateString());
}, []);
```
```

1. useSession sin optional chaining

```
```typescript
// ❌ NUNCA hacer esto
const { data: session } = useSession();
```

```
// ✅ SIEMPRE hacer esto
const { data: session, status } = useSession() || {};
...

```

#### 1. No envolver con SessionProvider

```
```typescript
// ❌ NUNCA hacer esto
export default function RootLayout({ children }) {
  return <>{children};
}

// ✅ SIEMPRE hacer esto
export default function RootLayout({ children }) {
  return {children};
}
...

```

13.5 Estándares de Código

1. TypeScript Estricto

- Siempre definir tipos para props
- No usar `any` (usar `unknown` si es necesario)
- Interfaces para objetos complejos

2. Convenciones de Nombres

- Componentes: PascalCase
- Funciones: camelCase
- Constantes: UPPER_SNAKE_CASE
- Archivos: kebab-case para páginas, PascalCase para componentes

3. Organización de Imports

```
```typescript
// 1. React y Next.js
import { useState } from 'react'
import { useRouter } from 'next/navigation'

// 2. Librerías de terceros
import { useSession } from 'next-auth/react'
import { toast } from 'sonner'

// 3. Componentes UI
import { Button } from '@components/ui/button'
import { Card } from '@components/ui/card'

// 4. Componentes locales
import { Header } from '../_components/header'

// 5. Utilidades y tipos
import { cn } from '@lib/Utils'
import type { User } from '@lib/types'
...

```

---



## 14. ROADMAP Y TODOS

### 14.1 Features Implementados

- [x] Sistema de autenticación completo
- [x] Plan de 26 semanas estructurado
- [x] Dashboard con seguimiento de progreso
- [x] Tutor de IA conversacional
- [x] Corrección gramatical en tiempo real
- [x] Traducción para principiantes
- [x] Sistema de gamificación completo
- [x] Sistema de vocabulario con categorías
- [x] Repetición espaciada (SM-2)
- [x] Recursos educativos curados
- [x] Guía de uso interactiva
- [x] Analíticas de sesiones
- [x] Historial de conversaciones
- [x] Sistema de logros y rachas

### 14.2 Mejoras Pendientes

#### Alta Prioridad:

1. ☐ Implementar tests unitarios y de integración
2. ☐ Agregar modo oscuro completo
3. ☐ Optimizar carga de imágenes y assets
4. ☐ Implementar PWA (Progressive Web App)
5. ☐ Agregar notificaciones push
6. ☐ Implementar export de progreso (PDF/Excel)
7. ☐ Agregar búsqueda global

#### Media Prioridad:

8. ☐ Sistema de amigos y competencia
9. ☐ Agregar más contextos de conversación
10. ☐ Implementar transcripción de voz (Speech-to-Text)
11. ☐ Agregar pronunciación con síntesis de voz (TTS)
12. ☐ Implementar flashcards colaborativas
13. ☐ Agregar estadísticas avanzadas con charts
14. ☐ Implementar sistema de recompensas

#### Baja Prioridad:

15. ☐ Integración con calendarios externos
16. ☐ Modo offline con Service Workers
17. ☐ Compartir logros en redes sociales
18. ☐ Sistema de badges personalizados
19. ☐ Temas de personalización
20. ☐ Multi-idioma (i18n)

### 14.3 Bugs Conocidos

#### 1. Dynamic Server Usage Warning

- Ruta: `/api/tutor/context`

- Causa: Uso de headers/cookies en route handler
- Impacto: Warning en build, no afecta funcionalidad
- Solución pendiente: Refactorizar para usar solo body params

## 2. Scroll en móvil en conversaciones largas

- Área: Tutor chat
- Impacto: Menor, UX en móvil
- Solución: Implementar auto-scroll mejorado

## 3. Carga lenta inicial de vocabulario

- Área: Sistema de vocabulario
- Impacto: Menor, primera carga
- Solución: Implementar caching + lazy loading

# 14.4 Deuda Técnica

## 1. Refactorizar componentes grandes

- `tutor-client.tsx` (>800 líneas)
- `dashboard-client.tsx` (>500 líneas)
- Dividir en sub-componentes más pequeños

## 2. Mejorar tipado de JSON fields

- `LearningContext.weakAreas` (`Json` → `string[]`)
- `ChatMessage.grammarErrors` (`Json` → `GrammarError[]`)
- Usar tipos explícitos en Prisma

## 3. Abstractar lógica de gamificación

- Crear hook `useGamification`
- Centralizar actualizaciones de puntos
- Simplificar componentes que usan gamificación

## 4. Implementar error boundaries

- Componente `ErrorBoundary` global
- Error boundaries específicos por sección
- Mejores mensajes de error para el usuario

## 5. Optimizar queries de Prisma

- Agregar índices adicionales
  - Usar `select` para reducir payload
  - Implementar `cursor pagination`
-

## 15. COMANDOS ÚTILES

---

### Desarrollo

```
Instalar dependencias
cd /home/ubuntu/speaklyplan/nextjs_space && yarn install

Modo desarrollo
cd /home/ubuntu/speaklyplan/nextjs_space && yarn dev

Build para producción
cd /home/ubuntu/speaklyplan/nextjs_space && yarn build

Iniciar producción
cd /home/ubuntu/speaklyplan/nextjs_space && yarn start

Linting
cd /home/ubuntu/speaklyplan/nextjs_space && yarn lint
```

### Base de Datos (Prisma)

```
Generar cliente Prisma
cd /home/ubuntu/speaklyplan/nextjs_space && yarn prisma generate

Crear migración
cd /home/ubuntu/speaklyplan/nextjs_space && yarn prisma migrate dev --name migration_name

Aplicar migraciones
cd /home/ubuntu/speaklyplan/nextjs_space && yarn prisma migrate deploy

Abrir Prisma Studio (GUI)
cd /home/ubuntu/speaklyplan/nextjs_space && yarn prisma studio

Reset database (¡CUIDADO!)
cd /home/ubuntu/speaklyplan/nextjs_space && yarn prisma migrate reset

Seed database
cd /home/ubuntu/speaklyplan/nextjs_space && yarn prisma db seed
```

### Testing

```
Run tests (cuando estén implementados)
cd /home/ubuntu/speaklyplan/nextjs_space && yarn test

Run tests en modo watch
cd /home/ubuntu/speaklyplan/nextjs_space && yarn test:watch

Coverage
cd /home/ubuntu/speaklyplan/nextjs_space && yarn test:coverage
```

---

## 16. CONTACTO Y SOPORTE

---

**Proyecto:** SpeaklyPlan

**Versión:** 1.0

**Última actualización:** 09 de Octubre de 2025

**Ubicación:** /home/ubuntu/speaklyplan/

**Framework:** Next.js 14.2.28

**Base de datos:** PostgreSQL (Abacus AI Hosted)

**Deploy:** Abacus AI Platform

---

## NOTAS FINALES

---

Este documento contiene toda la información necesaria para continuar el desarrollo de SpeaklyPlan en una nueva conversación. Incluye:

- ✓ Arquitectura completa del sistema
- ✓ Stack tecnológico detallado
- ✓ Estructura de base de datos
- ✓ Documentación de todos los módulos
- ✓ Sistema de IA y prompts
- ✓ API endpoints completos
- ✓ Sistema de gamificación
- ✓ Componentes UI
- ✓ Variables de entorno
- ✓ Flujos de usuario
- ✓ Mejores prácticas
- ✓ Errores comunes a evitar
- ✓ Roadmap de desarrollo
- ✓ Comandos útiles

**IMPORTANTE:** Este documento debe actualizarse cada vez que se realicen cambios significativos en la arquitectura o se agreguen nuevas funcionalidades.

---

**FIN DEL DOCUMENTO**