Documentación del Webchat - Plataforma Lumen

Fecha de actualización: 20 de octubre, 2025
URL de producción: https://lumen.abacusai.app

📋 Resumen General

El webchat de Lumen es un asistente inteligente que permite a los ciudadanos consultar información sobre presupuestos, gastos, proyectos y contratos municipales de forma conversacional. Utiliza tecnología RAG (Retrieval Augmented Generation) para anclar todas sus respuestas en datos reales del portal de administración.

🎯 Características Implementadas

1. Fuente de Datos (RAG/Grounding)

✅ Exclusivamente datos del portal de administración con `isPublic = true`
✅ Invalidación automática al cambiar datos en el admin
✅ Trazabilidad completa: logging interno de consultas
✅ Respuesta clara cuando no hay datos públicos disponibles

2. Generación de Respuestas

✅ RAG: Respuestas ancladas en registros filtrados
✅ Formato Chile: CLP con separador de miles (ej: $1.500.000)
✅ Fechas en formato DD-MM-AAAA
✅ Citación de fuentes (colección + ID + fecha)

3. UX/UI del Widget

✅ Avatar visible con estilo municipal/neutral
✅ Teasers animados con rotación de 2-3 mensajes
✅ Mensaje de bienvenida con chips de sugerencias
✅ Acciones rápidas: “Ver panel de proyectos”, “Consultar mapas”
✅ Componentes ricos: Cards para proyectos/contratos
✅ Estados del bot: “Consultando datos…”, “Escribiendo…”
✅ Accesibilidad: navegación teclado, ARIA labels, contraste AA
✅ Persistencia: última conversación guardada en localStorage
✅ Acción “Borrar conversación”
✅ Idioma: Español (Chile) por defecto, adapta al idioma del usuario

4. Seguridad & Privacidad

✅ No responde contenido con `isPublic = false`
✅ Sincronización inmediata con cambios en admin
✅ Fallback cuando el servicio no está disponible

🗄️ Base de Datos - Colecciones Utilizadas

El webchat consulta las siguientes tablas de PostgreSQL a través de Prisma:

1. budgets (Presupuestos)

```
Tabla: budgets
Campos consultados:
  - id: String (PK)
  - fiscalYearId: String
  - department: String
  - category: String
  - subcategory: String
  - amountPlanned: Float
  - currency: String
  - notes: String (nullable)
  - isPublic: Boolean (FILTRO OBLIGATORIO)
  - createdAt: DateTime
```

Ejemplo de consulta:
- “¿Cuál es el presupuesto 2025?”
- “Presupuesto total de educación”

2. expenditures (Gastos)

```
Tabla: expenditures
Campos consultados:
  - id: String (PK)
  - fiscalYearId: String
  - date: DateTime
  - department: String
  - category: String
  - subcategory: String
  - concept: String
  - amountActual: Float
  - currency: String
  - location: String (nullable)
  - isPublic: Boolean (FILTRO OBLIGATORIO)
  - createdAt: DateTime
```

Ejemplo de consulta:
- “¿Cuánto se ha gastado en salud?”
- “Gastos del último mes”

3. projects (Proyectos)

```
Tabla: projects
Campos consultados:
  - id: String (PK)
  - title: String
  - description: String
  - status: String
  - startDate: DateTime (nullable)
  - endDate: DateTime (nullable)
  - department: String
  - category: String
  - requestedBudget: Float (nullable)
  - approvedBudget: Float (nullable)
  - location: String (nullable)
  - isPublic: Boolean (FILTRO OBLIGATORIO)
  - createdAt: DateTime
```

Ejemplo de consulta:
- “¿Qué proyectos hay en Renca?”
- “Proyectos de infraestructura activos”

4. contracts (Contratos)

```
Tabla: contracts
Campos consultados:
  - id: String (PK)
  - title: String
  - description: String
  - supplierId: String
  - amount: Float
  - currency: String
  - startDate: DateTime
  - endDate: DateTime (nullable)
  - status: String
  - contractNumber: String (nullable)
  - isPublic: Boolean (FILTRO OBLIGATORIO)
  - createdAt: DateTime
```

Ejemplo de consulta:
- “Contratos vigentes”
- “¿Qué contratos hay con el proveedor X?”

🔌 Endpoints Utilizados

POST /api/ai-query

Ubicación: `/home/ubuntu/municipal_transparency_platform/frontend/nextjs_space/app/api/ai-query/route.ts`

Descripción: Endpoint principal del webchat que procesa consultas en lenguaje natural.

Request:

```
{
  "query": "¿Qué proyectos hay en Renca?",
  "conversationHistory": [] // opcional
}
```

Response: Streaming (text/event-stream)

```
data: {"choices":[{"delta":{"content":"Según los registros..."}}]}
data: {"choices":[{"delta":{"content":" de proyectos públicos..."}}]}
data: [DONE]
```

Flujo interno:
1. **Análisis de consulta** (LLM): Determina tipo de datos y filtros necesarios
2. **Consulta a DB**: Obtiene datos públicos (`isPublic = true`)
3. **Logging**: Registra timestamp, query, dataTypes, filters, resultCount
4. **Generación de respuesta** (LLM + RAG): Respuesta anclada en datos reales
5. **Streaming**: Transmite respuesta en tiempo real

Modelo LLM: `gpt-4.1-mini` (Abacus.AI API)

📊 Logging de Consultas

Ubicación: `/home/ubuntu/municipal_transparency_platform/frontend/nextjs_space/logs/chat-queries-YYYY-MM-DD.json`

Formato:

```
{
  "timestamp": "2025-10-20T15:30:45.123Z",
  "query": "¿Qué proyectos hay en Renca?",
  "dataTypes": ["projects"],
  "filters": {
    "municipality": "Renca"
  },
  "resultCount": 5
}
```

Campos registrados:
- `timestamp` : Fecha y hora ISO 8601
- `query` : Texto de la consulta del usuario
- `dataTypes` : Colecciones consultadas (budgets, expenditures, projects, contracts, general_stats)
- `filters` : Filtros aplicados (year, department, category, municipality, status)
- `resultCount` : Número de registros devueltos

🎨 Componentes de UI

AIChatWidget

Ubicación: `/home/ubuntu/municipal_transparency_platform/frontend/nextjs_space/components/ai-chat-widget.tsx`

Características:
- Botón flotante con avatar animado
- Teasers con rotación automática cada 4 segundos
- Widget maximizable/minimizable
- Indicador de estado “conectado”
- Botón para borrar conversación

AIChat

Ubicación: `/home/ubuntu/municipal_transparency_platform/frontend/nextjs_space/components/ai-chat.tsx`

Características:
- Mensaje de bienvenida personalizado
- 3 chips de sugerencias con íconos
- 2 acciones rápidas con enlaces
- Estados del bot: “Consultando datos...”, “Escribiendo...”
- Formato de mensajes con timestamp
- Persistencia en localStorage
- Accesibilidad: ARIA labels, navegación por teclado

🔐 Seguridad

Filtro de Datos Públicos

Todas las consultas incluyen el filtro obligatorio:

```
const publicFilter = { isPublic: true }
const combinedFilters = { ...filters, ...publicFilter }
```

Mensajes de Error

- **Sin datos públicos:**

  > “No encuentro información pública para esa consulta. Puedes revisar los módulos en el sitio o realizar una solicitud formal de información.”

- **Servicio no disponible:**

  > “El servicio de datos está temporalmente no disponible. Intenta nuevamente o revisa los módulos del sitio.”

🧪 Casos de Prueba

✅ Test 1: Proyectos en Renca

Input: "¿Qué proyectos hay en Renca?"
Expected: Cards con: título, estado, monto, ubicación + referencia de fuente

✅ Test 2: Presupuesto 2025

Input: "¿Cuál es el presupuesto 2025?"
Expected: Totales en CLP, % ejecución, fuente citada

✅ Test 3: Sin datos públicos

Input: Consulta sobre datos no públicos
Expected: Mensaje claro + enlaces a módulos

✅ Test 4: UX/UI

Verificar: Teasers, avatar, chips, accesibilidad activos

🌐 URLs de Acceso

- **Producción:** https://lumen.abacusai.app
- **Portal Ciudadano:** https://lumen.abacusai.app/ciudadano
- **Portal Admin:** https://lumen.abacusai.app/admin
- **Login Admin:**
- Email: admin@muni.cl
- Password: admin123

📦 Dependencias Principales

```
{
  "@prisma/client": "6.7.0",
  "framer-motion": "10.18.0",
  "next": "14.2.28",
  "react": "18.2.0",
  "lucide-react": "0.446.0"
}
```

API LLM: Abacus.AI (gpt-4.1-mini)
Base de datos: PostgreSQL (via Prisma ORM)

🚀 Comandos de Desarrollo

```
# Generar cliente Prisma
cd frontend/nextjs_space
yarn prisma generate

# Aplicar cambios de schema
yarn prisma db push

# Desarrollo local
yarn dev

# Build de producción
yarn build
```

📞 Soporte

Para cualquier consulta técnica o reporte de errores, contactar al equipo de desarrollo de la Plataforma de Transparencia Municipal.

Última actualización: 20 de octubre, 2025