Dump.do API Documentation

Edge Function: Chat

Endpoint

```
POST /functions/v1/chat
```

Authentication

Requires a valid Supabase Auth JWT token in the Authorization header.

```
Authorization: Bearer <user_jwt_token>
```

Request Body

```typescript
interface ChatRequest {
  // Required: The user's message
  message: string;  // max 10000 characters

  // Optional: Session ID for continuing a conversation
  sessionId?: string;

  // Optional: Chat mode
  mode?: 'dump' | 'processar';  // default: 'dump'

  // Optional: Flag to switch mode mid-session
  switchMode?: boolean;
}
```

Response

Success (200 OK)

```typescript
interface ChatResponse {
  // The AI's response message
  message: string;

  // Session ID (use this for follow-up messages)
  sessionId: string;

  // Current mode
  mode: 'dump' | 'processar';

  // Risk level detected in the user's message
  riskLevel: 'none' | 'low' | 'medium' | 'high' | 'critical';

  // Whether this is an emergency response (bypassed LLM)
  isEmergencyResponse: boolean;

  // Token usage (only for non-emergency responses)
  tokensUsed?: {
    input: number;
    output: number;
  };
}
```

Error Responses

Status	Description
400	Bad Request - Invalid or missing message
401	Unauthorized - Invalid or missing token
405	Method Not Allowed - Only POST is accepted
429	Too Many Requests - Rate limit exceeded
500	Internal Server Error

```typescript
interface ErrorResponse {
  error: string;
  details?: string;  // Only in development environment
}
```

Usage Examples

Starting a New Session (Dump Mode)

```
curl -X POST "https://your-project.supabase.co/functions/v1/chat" \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "message": "Cara, to muito estressado com o trabalho",
    "mode": "dump"
  }'
```

Response:

```
{
  "message": "Pesado. Conta mais - o que tá rolando no trabalho?",
  "sessionId": "550e8400-e29b-41d4-a716-446655440000",
  "mode": "dump",
  "riskLevel": "low",
  "isEmergencyResponse": false,
  "tokensUsed": {
    "input": 156,
    "output": 23
  }
}
```

Continuing a Session

```
curl -X POST "https://your-project.supabase.co/functions/v1/chat" \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "message": "Meu chefe me cobrou na frente de todo mundo",
    "sessionId": "550e8400-e29b-41d4-a716-446655440000"
  }'
```

Switching to Processar Mode

```
curl -X POST "https://your-project.supabase.co/functions/v1/chat" \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "message": "Ok, agora quero resolver isso",
    "sessionId": "550e8400-e29b-41d4-a716-446655440000",
    "mode": "processar",
    "switchMode": true
  }'
```

Emergency Response Example

When a critical risk is detected:

```
curl -X POST "https://your-project.supabase.co/functions/v1/chat" \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "message": "Não aguento mais, quero acabar com tudo"
  }'
```

Response:

```
{
  "message": "🚨 **Estou aqui com você.**\n\nO que você está sentindo é real...",
  "sessionId": "550e8400-e29b-41d4-a716-446655440001",
  "mode": "dump",
  "riskLevel": "critical",
  "isEmergencyResponse": true
}
```

JavaScript/TypeScript Client Example

```typescript
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(SUPABASE_URL, SUPABASE_ANON_KEY);

interface ChatOptions {
  message: string;
  sessionId?: string;
  mode?: 'dump' | 'processar';
  switchMode?: boolean;
}

async function chat(options: ChatOptions) {
  const { data: { session } } = await supabase.auth.getSession();

  if (!session) {
    throw new Error('User not authenticated');
  }

  const response = await fetch(
    `${SUPABASE_URL}/functions/v1/chat`,
    {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${session.access_token}`,
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(options),
    }
  );

  if (!response.ok) {
    const error = await response.json();
    throw new Error(error.error);
  }

  return response.json();
}

// Usage
const response = await chat({
  message: 'Preciso desabafar sobre algo',
  mode: 'dump'
});

console.log(response.message);
// Continue with: chat({ message: '...', sessionId: response.sessionId })
```

Rate Limiting

- **20 requests per minute** per user
- Returns 429 when exceeded
- Resets after 60 seconds

Risk Levels

Level	Description	Behavior
`none`	No risk indicators	Normal LLM processing
`low`	Mild stress/anxiety	Normal processing
`medium`	Significant distress	Logged, normal processing
`high`	Crisis indicators	Emergency response
`critical`	Imminent risk	Emergency response + CVV referral

Best Practices

1. **Always store the sessionId** returned and use it for follow-up messages
2. **Handle emergency responses** appropriately in your UI
3. **Don’t retry immediately** on 429 errors - wait at least 60 seconds
4. **Validate messages client-side** before sending (max 10000 chars)
5. **Respect user privacy** - don’t log message content client-side