AI Bookmark Manager - Documentació Tècnica

Data: 4 de Desembre de 2025 **Versió:** 1.0 **Autor:** Roger Masellas

---

📋 Índex

Descripció del Projecte

AI Bookmark Manager és una aplicació web desenvolupada inicialment amb Google AI Studio que permet gestionar i organitzar els bookmarks/preferits exportats des de Twitter (X). L'aplicació utilitza Intel·ligència Artificial (Google Gemini) per analitzar, categoritzar i titular automàticament els tweets relacionats amb IA.

Objectiu Principal

Facilitar l'organització de bookmarks de Twitter relacionats amb Intel·ligència Artificial mitjançant: - Categorització automàtica per temàtiques - Generació de títols descriptius en català - Filtrat de contingut no relacionat amb IA - Eliminació de duplicats - Sincronització multi-dispositiu

Funcionalitats Clau

1. **Import de JSON:** Lectura de fitxers JSON exportats des de Twitter
2. **Processament amb IA:** Anàlisi automàtic amb Google Gemini per:
    - Determinar si el tweet és relacionat amb IA
    - Generar un títol descriptiu en català
    - Assignar una categoria apropiada
    - Extreure enllaços externs rellevants
3. **Filtrat Intel·ligent:**
    - Descarta tweets no relacionats amb IA
    - Crea un fitxer descarregable amb tweets rebutjats
4. **Gestió de Duplicats:**
    - Detecta i evita la importació de tweets duplicats
    - Manté un registre de tweets eliminats per evitar reimportacions
5. **Sincronització VPS:**
    - Emmagatzema dades en un servidor VPS
    - Permet accés des de múltiples dispositius
    - Sincronització automàtica mitjançant API REST
6. **Sistema de Backup:**
    - Exportació de totes les dades
    - Importació per restaurar o fusionar dades

Arquitectura General

Stack Tecnològic

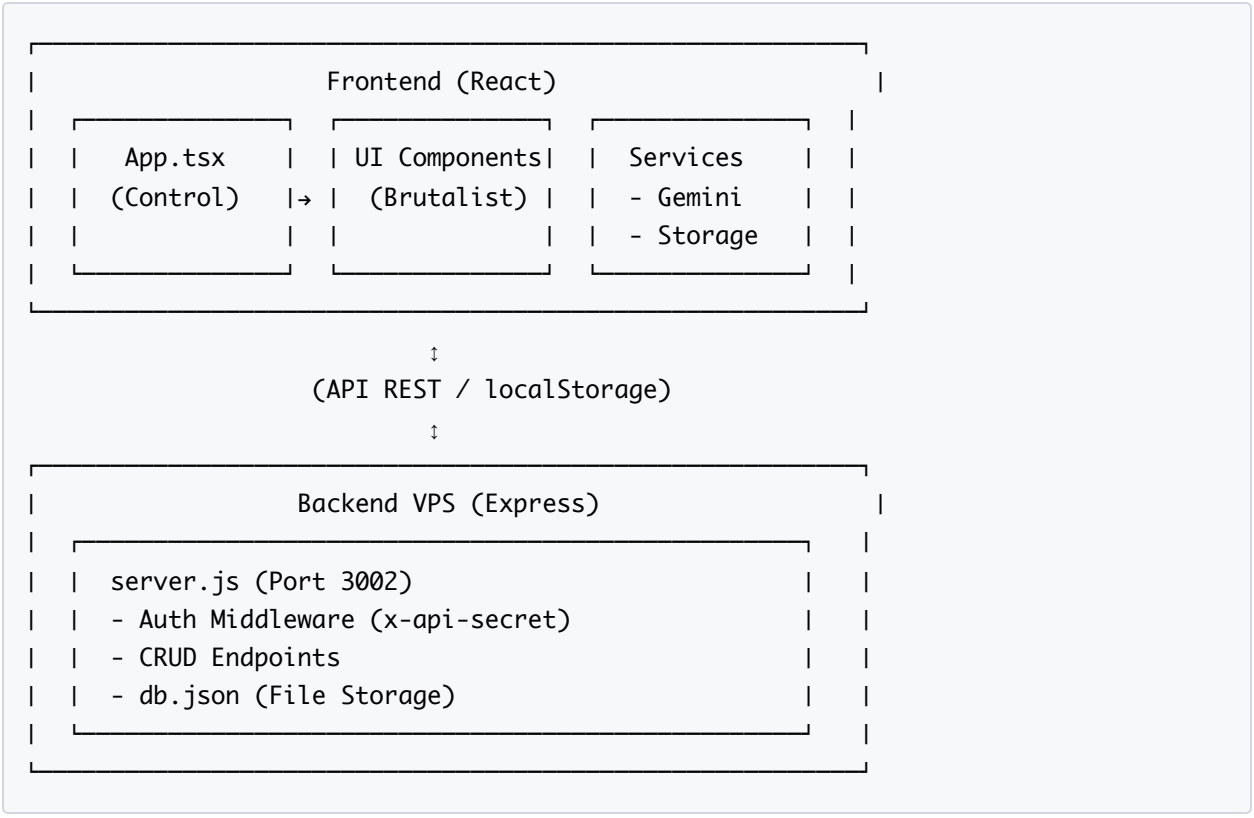
Frontend

- **Framework:** React 19.2.0
- **Llenguatge:** TypeScript (~5.9.3)
- **Build Tool:** Vite 7.2.4
- **Estils:** Tailwind CSS
- **Icones:** Lucide React
- **IA:** Google Generative AI SDK (@google/genai 1.30.0)

Backend (VPS)

- **Runtime:** Node.js
- **Framework:** Express 4.18.2
- **CORS:** cors 2.8.5
- **Storage:** Sistema de fitxers (db.json)

Patró d'Arquitectura



Principis de Disseny

1. **Component-Based:** Arquitectura modular amb components reutilitzables
2. **Separation of Concerns:** Separació clara entre UI, lògica de negoci i serveis
3. **Functional Programming:** Ús extensiu de React Hooks i programació funcional
4. **Type Safety:** TypeScript per prevenir errors en temps de desenvolupament
5. **Storage Agnostic:** Abstracció del sistema d'emmagatzematge (local/remot)

Estructura del Projecte

Arbre de Directoris

```
ai-bookmarks/
├── public/              # Assets públics
├── src/
|   ├── components/
|   |   └── UI.tsx      # Components UI reutilitzables (brutalista)
|   ├── services/
|   |   ├── geminiService.ts   # Integració amb Gemini AI
|   |   └── storage.ts         # Capa d'abstracció d'emmagatzematge
|   ├── App.tsx         # Component principal (890 línies)
|   ├── main.tsx        # Punt d'entrada React
|   ├── types.ts        # Definicions TypeScript
|   ├── translations.ts # Strings UI i prompts IA (català)
|   ├── index.css       # Tailwind + CSS custom
|   └── vite-env.d.ts   # Types variables d'entorn
├── .env                # Configuració (API keys, VPS URL)
├── index.html          # HTML d'entrada
├── package.json        # Dependencies i scripts
├── vite.config.ts      # Configuració Vite
├── tsconfig.json       # Configuració TypeScript
└── eslint.config.js    # Configuració ESLint
```

Fitxers de Configuració

.env

```
VITE_API_KEY=AIzaSyCtn6WyoYxBYAOPoijSWnPZ8v0FyBWx5fU
VITE_STORAGE_API_URL=http://62.169.25.188:3002
VITE_STORAGE_SECRET=aAgYYud97Kp29Lif9u0i
```

vite.config.ts

- Servidor dev: port 3000, host 0.0.0.0
- Plugin React amb Fast Refresh
- Càrrega de variables d'entorn

package.json - Scripts

```
(#cb4-1){
(#cb4-2)  "dev": "vite",          // Inicia servidor desenvolupament
(#cb4-3)  "build": "tsc && vite build",  // Compila TypeScript + build
(#cb4-4)  "lint": "eslint .",     // Verifica codi
(#cb4-5)  "preview": "vite preview"  // Preview build producció
(#cb4-6)}
```

Components Principals

1. App.tsx - Component Principal (890 línies)

Responsabilitats: - Gestió centralitzada d'estat - Orquestració de la UI (layout, modals, navegació) - Lògica d'import/export de dades - Processament de tweets amb Gemini - Sistema de logging i progrés

Estat Principal:

```
(#cb5-1)// Dades
(#cb5-2)bookmarks: Bookmark[]           // Tots els bookmarks processats
(#cb5-3)categories: Category[]          // Llista de categories
(#cb5-4)deletedIds: string[]           // Blacklist d'IDs eliminats
(#cb5-5)
(#cb5-6)// UI State
(#cb5-7)isLoading: boolean             // Indica processament actiu
(#cb5-8)selectedCategory: string       // Categoria filtrada visualment
(#cb5-9)showMobileMenu: boolean        // Control menú mòbil
(#cb5-10)showLogs: boolean             // Mostra consola de logs
(#cb5-11)
(#cb5-12)// Processament
(#cb5-13)progress: { current: number, total: number }  // Progrés IA
(#cb5-14)logs: LogEntry[]              // Registre del procés d'import
(#cb5-15)rejectedTweets: TweetRaw[]    // Tweets no-IA per exportar
(#cb5-16)
(#cb5-17)// Modals
(#cb5-18)editModalState: { bookmark, show }   // Edició bookmark
(#cb5-19)deleteModalState: { id, originalId, show }  // Confirmació eliminació
(#cb5-20)resultsModalState: { added, skipped, rejected, show }  // Resum import
```

Funcions Clau:

1. **handleFileUpload()**
    - Llegeix fitxer JSON pujat per l'usuari
    - Detecta tipus: backup propi o export de Twitter
    - Crida processTweetsData() o mergeix backup
2. **processTweetsData()**
    - Extreu tweets del JSON de Twitter
    - Deduplica contra bookmarks existents i deletedIds
    - Crida processBookmarksWithGemini()
3. **processBookmarksWithGemini()**
    - Processa tweets amb Gemini (1 a 1)
    - Gestiona rate limiting (4s entre requests)
    - Retry amb exponential backoff en errors 429
    - Separa tweets IA vs no-IA
    - Actualitza progress i logs en temps real
4. **confirmDelete()**
    - Elimina bookmark de l'estat
    - Afegeix ID a deletedIds (blacklist)
    - Persisteix canvis a storage
5. **exportBackupJSON()**
    - Genera JSON amb bookmarks, categories, deletedIds
    - Inclou metadata (versió, timestamp)
    - Descarrega com a fitxer

2. geminiService.ts - Integració IA

Configuració:

```
(#cb6-1)const model = genAI.getGenerativeModel({
(#cb6-2)  model: "gemini-2.0-flash-exp",
(#cb6-3)  generationConfig: {
(#cb6-4)    responseMimeType: "application/json",
(#cb6-5)    responseSchema: ProcessedTweetResultSchema,
(#cb6-6)    maxOutputTokens: 500  // Prevenir loops infinits
(#cb6-7)  }
(#cb6-8)})
```

Funcions:

1. **processBookmarksWithGemini(tweets, onProgress, onLog)**
   - Batch processing: itera tweets 1 a 1
   - Trunca text a 1000 chars per request
   - Crida generateContent() amb system instruction
   - Retry logic amb exponential backoff
   - Retorna arrays separats: AI tweets i rejected tweets
2. **handleRateLimitWithRetry()**
   - Detecta errors 429 (rate limit)
   - Exponential backoff: 10s → 15s → 22.5s → ... → 60s max
   - Màxim 10 intents
   - Logs detallats de cada retry

Schema de Resposta:

```
(#cb7-1){
(#cb7-2)  type: "ARRAY",
(#cb7-3)  items: {
(#cb7-4)    type: "OBJECT",
(#cb7-5)    properties: {
(#cb7-6)      originalId: STRING,
(#cb7-7)      isAI: BOOLEAN,
(#cb7-8)      title: STRING,        // Català
(#cb7-9)      description: STRING,  // No utilitzat
(#cb7-10)      category: STRING,     // De llista predefinida
(#cb7-11)      externalLinks: ARRAY<STRING>
(#cb7-12)    }
(#cb7-13)  }
(#cb7-14)}
```

3. storage.ts - Abstracció d'Emmagatzematge

Estratègia Dual:

```
(#cb8-1)const USE_API = !!import.meta.env.VITE_STORAGE_SECRET
(#cb8-2)
(#cb8-3)// Si existeix VITE_STORAGE_SECRET → API
(#cb8-4)// Altrament → localStorage
```

API Wrapper:

```
(#cb9-1)async function apiRequest<T>(endpoint, method, data?) {
(#cb9-2)  const response = await fetch(`${API_URL}/${endpoint}`, {
(#cb9-3)    method,
(#cb9-4)    headers: {
(#cb9-5)      'Content-Type': 'application/json',
(#cb9-6)      'x-api-secret': API_SECRET  // Autenticació
(#cb9-7)    },
(#cb9-8)    body: data ? JSON.stringify(data) : undefined
(#cb9-9)  })
(#cb9-10)
(#cb9-11)  if (!response.ok) throw new Error(`API error: ${response.status}`)
(#cb9-12)  const result = await response.json()
(#cb9-13)  return result.data  // Desembolicar { data: [...] }
(#cb9-14)}
```

Funcions CRUD: - `loadBookmarks()` / `saveBookmarks(data)` - `loadCategories()` / `saveCategories(data)` - `loadDeletedIds()` / `saveDeletedIds(data)` - `resetStorage()` - Esborra tot

LocalStorage Keys: - `ai-bookmarks-data` - `ai-bookmarks-categories` - `ai-bookmarks-deleted-ids`

4. UI.tsx - Sistema de Disseny Brutalista

Components Disponibles:

1. **Button**
   - Variants: primary, secondary, danger, ghost
   - Estats: hover, active (translate + shadow)
   - Suport per icons (Lucide)
2. **Card**

- Border gruixut negre (border-2)
- Shadow hard: shadow-[4px_4px_0px_0px_#000]
- Hover effect: translate + augment shadow

3. **Input / TextArea**
   - Border gruixut amb focus state
   - Padding generós
   - Tipografia bold
4. **Select**
   - Estilitzat custom (brutalist)
   - Dropdown consistent amb disseny
5. **Badge**
   - Etiquetes de categoria
   - Colors d'accent (groc #ffc107)
   - Text bold
6. **Modal**
   - Overlay amb backdrop blur
   - Animacions slide-in
   - Sistema de close button
   - Scroll independent del body

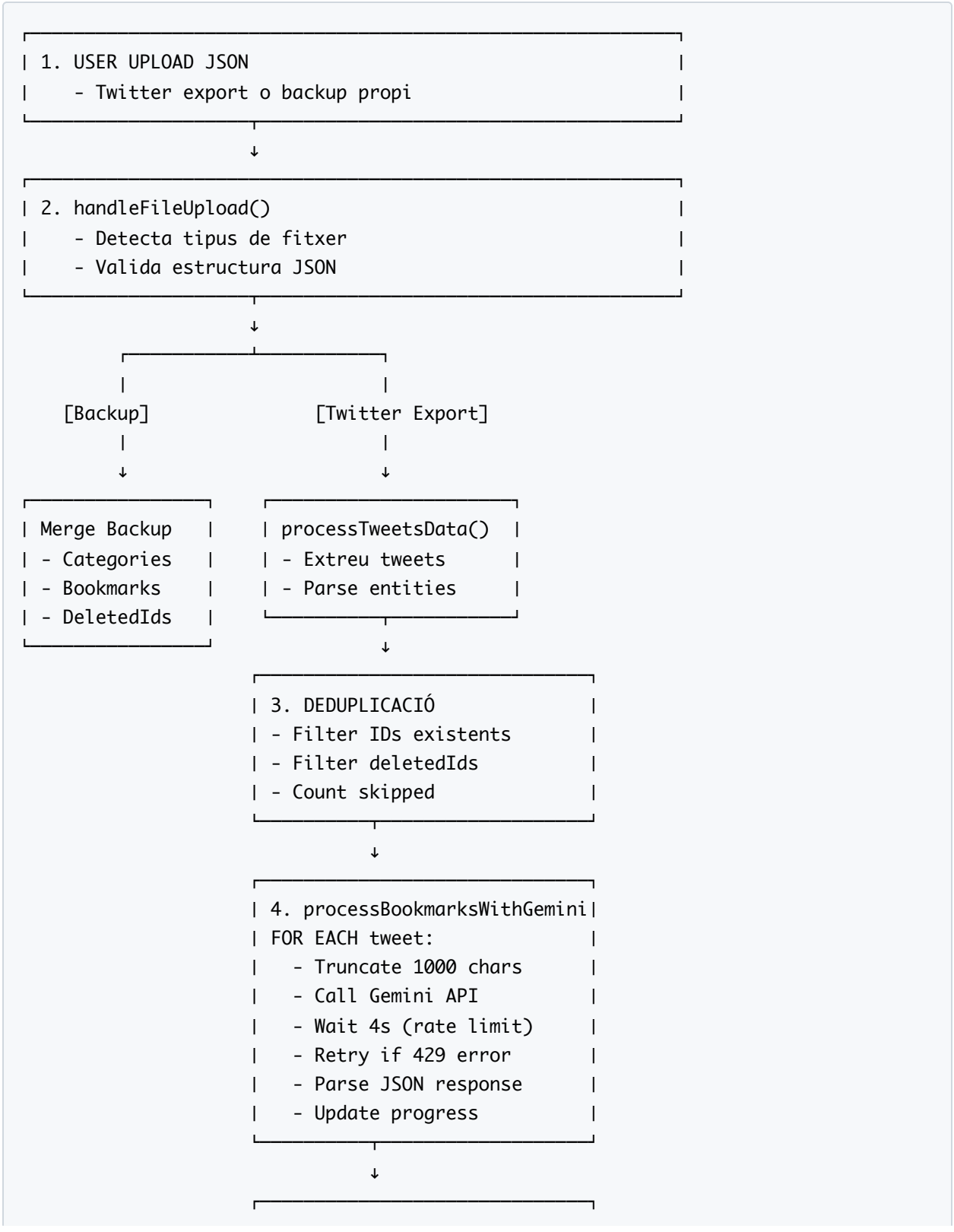
Estètica Brutalista:

```
/* Característiques clau */
border: 2px solid black
box-shadow: 4px 4px 0px 0px #000
font-weight: 800 (font-black)
colors: #000 (black), #fff (white), #ffc107 (yellow)

/* Efectes interactius */
hover: translate(-2px, -2px) + shadow-[6px_6px_0px_0px_#000]
active: translate(0, 0) + shadow-[0px_0px_0px_0px_#000]
```

Flux de Dades

1. Flux d'Importació Complert



```
                    | 5. SEPARACIÓ AI vs NO-AI     |
                    | IF isAI = true:              |
                    |   → Create Bookmark          |
                    | IF isAI = false:             |
                    |   → Add to rejectedTweets    |
                    └──────────┬──────────────────┘
                               ↓
                    ┌──────────────────────────────┐
                    | 6. CREATE BOOKMARK OBJECTS   |
                    | - Generate unique ID         |
                    | - Truncate desc (280 chars)  |
                    | - Extract metadata           |
                    | - Assign category            |
                    └──────────┬──────────────────┘
                               ↓
                    ┌──────────────────────────────┐
                    | 7. UPDATE STATE & STORAGE    |
                    | - setBookmarks([...new])     |
                    | - setCategories([...union])  |
                    | - saveBookmarks()            |
                    | - saveCategories()           |
                    └──────────┬──────────────────┘
                               ↓
                    ┌──────────────────────────────┐
                    | 8. SHOW RESULTS              |
                    | - Modal amb estadístiques    |
                    | - Opció download rejected    |
                    └──────────────────────────────┘
```

2. Flux de Visualització

```
STATE (bookmarks, categories, selectedCategory)
              ↓
        useMemo() Filtra per categoria
              ↓
        Grid Layout (responsive)
              ↓
    BookmarkCard per cada bookmark
              ↓
    Render: Badge + Title + Description + Links + Actions
```

3. Flux d'Eliminació

```
Click Delete Button
        ↓
Show Delete Modal (warning)
        ↓
User Confirms
        ↓
confirmDelete()
  - Remove from bookmarks array
  - Add originalId to deletedIds
  - saveBookmarks()
  - saveDeletedIds()
        ↓
Re-render UI (card desapareix)
```

Sistema de Categorització amb IA

Prompt del Sistema (translations.ts:89-104)

```
Ets un assistent que analitza tweets i determina si estan
relacionats amb Intel·ligència Artificial.

REGLES ESTRICTES:
1. Si el tweet NO tracta sobre IA/ML/LLM/Data Science:
   - Posa isAI: false
   - Deixa title, description, category, externalLinks buits

2. Si el tweet SÍ tracta sobre IA:
   - Posa isAI: true
   - Genera un TÍTOL CURT I DESCRIPTIU en CATALÀ
   - NO generar description (s'usarà el text original)
   - Assigna UNA categoria de la llista
   - Extreu enllaços externs (NO twitter.com/x.com)

CATEGORIES DISPONIBLES:
- Divulgació: Contingut educatiu, explicacions, guies
- Agents: Sistemes d'agents autònoms, multi-agent
- Skills: Capacitats, funcions, eines específiques
- RAG: Retrieval Augmented Generation, embeddings
- Cursos: Formació, tutorials estructurats
- Notícies: Actualitat, anuncis, releases
- Eines: Software, APIs, frameworks, aplicacions
- Altres: Altres temes relacionats amb IA

MOLT IMPORTANT:
- El títol ha de ser FINAL, sense explicacions ni raonament
- Sigues estricte: només marca isAI=true si és clarament IA
- NO inventar informació que no estigui al tweet
```

Model i Configuració

Model: Gemini 2.0 Flash Experimental - Ràpid i econòmic (ideal per batch processing) - Suporta JSON schema enforced - 15 RPM en tier gratuït

GenerationConfig:

```
(#cb15-1){
(#cb15-2)  responseMimeType: "application/json",
(#cb15-3)  responseSchema: ProcessedTweetResultSchema,
(#cb15-4)  maxOutputTokens: 500  // Prevenir loops infinits
(#cb15-5)}
```

Esquema de Validació

```
(#cb16-1)const ProcessedTweetResultSchema = {
(#cb16-2)  type: SchemaType.ARRAY,
(#cb16-3)  items: {
(#cb16-4)    type: SchemaType.OBJECT,
(#cb16-5)    properties: {
(#cb16-6)      originalId: {
(#cb16-7)        type: SchemaType.STRING,
(#cb16-8)        description: "ID original del tweet"
(#cb16-9)      },
(#cb16-10)      isAI: {
(#cb16-11)        type: SchemaType.BOOLEAN,
(#cb16-12)        description: "true si relacionat amb IA, false si no"
(#cb16-13)      },
(#cb16-14)      title: {
(#cb16-15)        type: SchemaType.STRING,
(#cb16-16)        description: "Títol curt en CATALÀ (buit si isAI=false)"
(#cb16-17)      },
(#cb16-18)      description: {
(#cb16-19)        type: SchemaType.STRING,
(#cb16-20)        description: "NO GENERAR, deixar buit sempre"
(#cb16-21)      },
(#cb16-22)      category: {
(#cb16-23)        type: SchemaType.STRING,
(#cb16-24)        description: "Categoria assignada (buit si isAI=false)",
(#cb16-25)        enum: [
(#cb16-26)          "Divulgació", "Agents", "Skills", "RAG",
(#cb16-27)          "Cursos", "Notícies", "Eines", "Altres"
(#cb16-28)        ]
(#cb16-29)      },
(#cb16-30)      externalLinks: {
(#cb16-31)        type: SchemaType.ARRAY,
(#cb16-32)        items: { type: SchemaType.STRING },
(#cb16-33)        description: "URLs externs mencionats"
(#cb16-34)      }
(#cb16-35)    },
(#cb16-36)    required: ["originalId", "isAI", "title", "category", "externalLinks"]
(#cb16-37)  }
(#cb16-38)}
```

Categories Predefinides

Categoria	Descripció	Exemples
Divulgació	Contingut educatiu i explicatiu	Articles, threads explicatius
Agents	Sistemes d'agents autònoms	AutoGPT, BabyAGI, multi-agents
Skills	Capacitats i funcions específiques	Function calling, tool use
RAG	Retrieval Augmented Generation	Vector DBs, embeddings, context
Cursos	Formació estructurada	MOOCs, tutorials, workshops
Notícies	Actualitat i anuncis	Product launches, research papers
Eines	Software i frameworks	LangChain, LlamaIndex, APIs
Altres	Altres temes d'IA	Fallback per contingut ambigu

Exemples de Categorització

Exemple 1 - Tweet sobre IA (isAI: true):

```
Input: "New paper from OpenAI on improving RAG with
        hierarchical indexing https://arxiv.org/..."

Output:
{
  originalId: "1234567890",
  isAI: true,
  title: "Nou paper d'OpenAI sobre millora de RAG amb indexació jeràrquica",
  description: "",
  category: "RAG",
  externalLinks: ["https://arxiv.org/..."]
}
```

Exemple 2 - Tweet NO IA (isAI: false):

```
Input: "Just had the best pizza in Naples! 🍕"

Output:
{
  originalId: "0987654321",
  isAI: false,
  title: "",
  description: "",
  category: "",
  externalLinks: []
}
```

Gestió de Duplicats

Estratègia de Deduplicació

Objectiu: Evitar processar tweets ja existents o eliminats prèviament

Implementació (App.tsx:246-259):

```
(#cb19-1)function processTweetsData(tweetsData: TweetRaw[]) {
(#cb19-2)  // 1. Extreure IDs de bookmarks existents
(#cb19-3)  const existingIds = new Set(
(#cb19-4)    bookmarks.map(b => {
(#cb19-5)      const parts = b.originalLink.split('/')
(#cb19-6)      return parts[parts.length - 1]  // Últim segment = tweet ID
(#cb19-7)    })
(#cb19-8)  )
(#cb19-9)
(#cb19-10)  // 2. Crear Set amb IDs eliminats (blacklist)
(#cb19-11)  const blacklistIds = new Set(deletedIds)
(#cb19-12)
(#cb19-13)  // 3. Filtrar tweets duplicats o eliminats
(#cb19-14)  const uniqueTweets = tweetsData.filter(t => {
(#cb19-15)    const tweetId = t.id_str || t.id
(#cb19-16)    return !existingIds.has(tweetId) &&
(#cb19-17)           !blacklistIds.has(tweetId)
(#cb19-18)  })
(#cb19-19)
(#cb19-20)  // 4. Comptar skipped per logging
(#cb19-21)  const skippedCount = tweetsData.length - uniqueTweets.length
(#cb19-22)
(#cb19-23)  addLog('info', `${uniqueTweets.length} tweets nous (${skippedCount} dupli
(#cb19-24)
(#cb19-25)  // 5. Processar només tweets únics
(#cb19-26)  await processBookmarksWithGemini(uniqueTweets, ...)
(#cb19-27)}
```

Flux de Deduplicació

```
IMPORT JSON (1000 tweets)
        ↓
Extract tweet IDs → [id1, id2, id3, ...]
        ↓
Check against existing bookmarks (50 matches)
        ↓
Check against deletedIds blacklist (10 matches)
        ↓
RESULT: 940 tweets únics a processar
        ↓
Log: "940 tweets nous (60 duplicats ignorats)"
```

Format d'IDs

Twitter/X Tweet ID: - Format: String numèric de 19 dígits - Exemple: "1862103456789012345" - Origen: `tweet.id_str` o `tweet.id` - URL: `https://twitter.com/i/web/status/1862103456789012345`

Bookmark ID (intern): - Format: `{tweetId}-{randomString}` - Exemple: "1862103456789012345-a7b3c9" - Generació: `${originalId}-${Math.random().toString(36).substr(2, 6)}` - Propòsit: Clau única per React keys i edició

Casos Especials

1. **Mateix tweet en múltiples imports:**
    - Primera vegada: Processat i guardat
    - Següents vegades: Detectat com duplicat, ignorat
2. **Tweet eliminat i tornat a importar:**
    - Estat: `deletedIds.includes(tweetId) === true`
    - Acció: Ignorat automàticament
    - Logging: Comptat com "duplicat ignorat"
3. **Backup merge:**
    - Els IDs del backup també es comproven
    - Només s'afegeixen bookmarks nous
    - Categories es fusionen (unió de sets)

Sistema de Tracking d'Eliminacions

Objectiu

Crear una "blacklist" persistent d'IDs de tweets eliminats per l'usuari, evitant que es tornin a importar en futurs uploads de JSON que continguin aquests tweets.

Implementació

Estructura de Dades:

```
(#cb21-1)deletedIds: string[]  // Array d'IDs de tweets eliminats
(#cb21-2)
(#cb21-3)// Exemple:
(#cb21-4)[
(#cb21-5)  "1862103456789012345",
(#cb21-6)  "1862104567890123456",
(#cb21-7)  "1862105678901234567"
(#cb21-8)]
```

Flux d'Eliminació (App.tsx:447-453):

```
(#cb22-1)const confirmDelete = async () => {
(#cb22-2)  const { id, originalId } = deleteModalState
(#cb22-3)
(#cb22-4)  // 1. Eliminar bookmark de l'array
(#cb22-5)  setBookmarks(prev => prev.filter(b => b.id !== id))
(#cb22-6)
(#cb22-7)  // 2. Afegir originalId a blacklist
(#cb22-8)  if (originalId) {
(#cb22-9)    setDeletedIds(prev => [...prev, originalId])
(#cb22-10)  }
(#cb22-11)
(#cb22-12)  // 3. Tancar modal
(#cb22-13)  setDeleteModalState({ id: '', originalId: '', show: false })
(#cb22-14)}
```

Efecte de Persistència:

```
(#cb23-1)useEffect(() => {
(#cb23-2)  if (deletedIds.length > 0) {
(#cb23-3)    saveDeletedIds(deletedIds)  // Guarda a storage (API o localStorage)
(#cb23-4)  }
(#cb23-5)}, [deletedIds])
```

Modal d'Advertència

Quan l'usuari clica el botó de delete, apareix un modal amb:

```
ATENCIÓ: Aquesta acció és permanent

Si elimines aquest bookmark:
- Desapareixerà de la teva llista
- NO es tornarà a importar en futurs uploads
- L'ID quedarà a la blacklist per sempre

Estàs segur que vols eliminar-lo?

[Cancel·lar]  [Eliminar]
```

Sincronització Multi-Dispositiu

Escenari: Usuari amb 2 dispositius (ordinador + mòbil)

1. **Dispositiu A:** Elimina tweet X
   - `deletedIds` s'actualitza localment
   - POST `/deleted` → VPS guarda blacklist actualitzada
2. **Dispositiu B:** Carrega app
   - GET `/deleted` → Descarrega blacklist del VPS
   - Té la blacklist sincronitzada amb dispositiu A
3. **Dispositiu B:** Import JSON amb tweet X
   - Deduplicació detecta X a `deletedIds`
   - Tweet X no es processa ni es mostra

Persistència

LocalStorage:

```
(#cb25-1)Key: "ai-bookmarks-deleted-ids"
(#cb25-2)Value: ["1862103456789012345", "1862104567890123456", ...]
```

VPS (db.json):

```
(#cb26-1){
(#cb26-2)  "bookmarks": [...],
(#cb26-3)  "categories": [...],
(#cb26-4)  "deletedIds": [
(#cb26-5)    "1862103456789012345",
(#cb26-6)    "186210456789012345"
(#cb26-7)  ]
(#cb26-8)}
```

Backup Export:

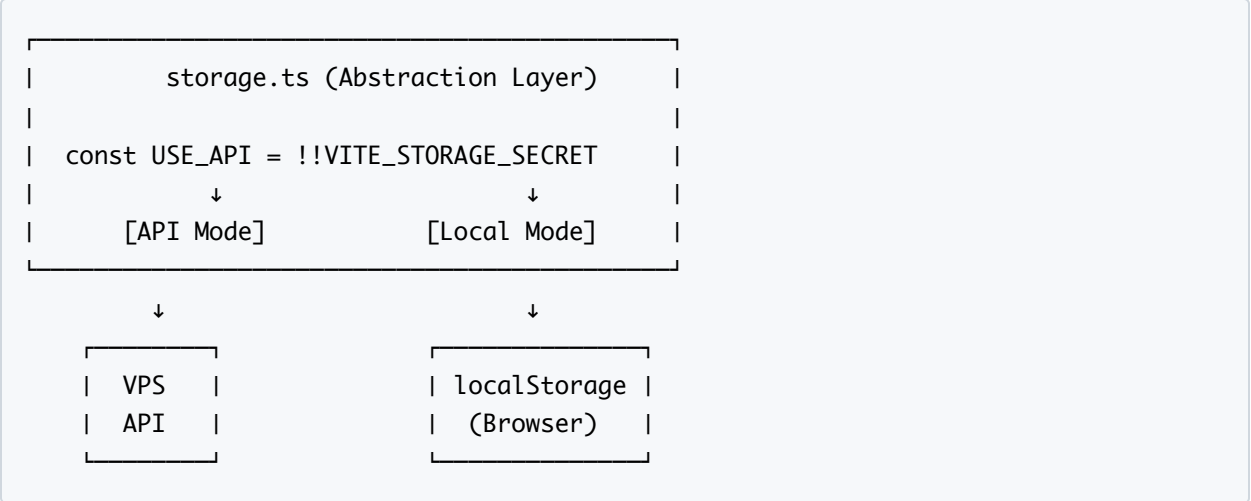
```
(#cb27-1){
(#cb27-2)  "backupVersion": 1,
(#cb27-3)  "timestamp": 1733328000000,
(#cb27-4)  "bookmarks": [...],
(#cb27-5)  "categories": [...],
(#cb27-6)  "deletedIds": [
(#cb27-7)    "1862103456789012345",
(#cb27-8)    "186210456789012345"
(#cb27-9)  ]
(#cb27-10)}
```

Limitacions i Consideracions

1. **No hi ha “undo”:**
    - Un cop eliminat, no es pot recuperar des de la UI
    - Solució: Restaurar des d’un backup anterior
2. **Creixement de blacklist:**
    - La llista pot créixer indefinidament
    - Possible millora futura: System de garbage collection
3. **Reset manual:**
    - Endpoint `/reset` esborra tot (incloent deletedIds)
    - Usar amb precaució

Emmagatzematge i Sincronització

Arquitectura d'Emmagatzematge



Mode Local (localStorage)

Quan s'activa: - `VITE_STORAGE_SECRET` no està definit al .env - Navegador suporta localStorage

Implementació:

```
function saveBookmarks(data: Bookmark[]) {
  localStorage.setItem(
    'ai-bookmarks-data',
    JSON.stringify(data)
  )
}

function loadBookmarks(): Bookmark[] {
  const stored = localStorage.getItem('ai-bookmarks-data')
  return stored ? JSON.parse(stored) : []
}
```

Avantatges: - Funciona offline - Sense latència - Sense cost servidor

Desavantatges: - Només accessible des del mateix navegador - Límit de ~10MB - Pot ser esborrat per l'usuari

Mode API (VPS)

Quan s'activa: - `VITE_STORAGE_SECRET` està definit al .env

Configuració:

```
(#cb30-1)const API_URL = import.meta.env.VITE_STORAGE_API_URL
(#cb30-2)const API_SECRET = import.meta.env.VITE_STORAGE_SECRET
(#cb30-3)
(#cb30-4)// Exemple:
(#cb30-5)// API_URL = "http://62.169.25.188:3002"
(#cb30-6)// API_SECRET = "aAgYYud97Kp29Lif9u0i"
```

Endpoints:

Endpoint	Mètode	Descripció	Auth
`/bookmarks`	GET	Obté tots els bookmarks	✓
`/bookmarks`	POST	Guarda array de bookmarks	✓
`/categories`	GET	Obté llista de categories	✓
`/categories`	POST	Guarda array de categories	✓
`/deleted`	GET	Obté IDs eliminats	✓
`/deleted`	POST	Guarda array d'IDs eliminats	✓
`/reset`	POST	Esborra totes les dades	✓

Format Request:

```
POST /bookmarks HTTP/1.1
Host: 62.169.25.188:3002
Content-Type: application/json
x-api-secret: aAgYYud97Kp29Lif9u0i

{
  "data": [
    { "id": "...", "title": "...", ... },
    { "id": "...", "title": "...", ... }
  ]
}
```

Format Response:

```
{
  "data": [
    { "id": "...", "title": "...", ... },
    { "id": "...", "title": "...", ... }
  ]
}
```

Gestió d'Errors:

```
async function apiRequest<T>(endpoint, method, data?) {
  try {
    const response = await fetch(...)

    if (!response.ok) {
      throw new Error(`API error: ${response.status}`)
    }

    const result = await response.json()
     return result.data

  } catch (error) {
    console.error(`Failed to ${method} ${endpoint}:`, error)
    throw error
  }
}
```

Avantatges: - Sincronització multi-dispositiu - Backup automàtic al servidor - No depèn del navegador

Desavantatges: - Requereix connexió a internet - Latència de xarxa - Cost de servidor VPS

Sistema d'Autenticació

Mètode: Custom secret header

```
x-api-secret: aAgYYud97Kp29Lif9u0i
```

Validació (server.js):

```
(#cb35-1)const checkAuth = (req, res, next) => {
(#cb35-2)  const secret = req.headers['x-api-secret']
(#cb35-3)  if (secret !== API_SECRET) {
(#cb35-4)    return res.status(403).json({ error: 'Unauthorized' })
(#cb35-5)  }
(#cb35-6)  next()
(#cb35-7)}
(#cb35-8)
(#cb35-9)app.use(checkAuth)  // Aplicat a tots els endpoints
```

Seguretat: - ⚠️ Secret compartit simple (no JWT) - ⚠️ HTTP sense HTTPS (dades en clar) - ✅ Suficient per ús personal - ❌ NO adequat per producció pública

Millores Recomanades (futur): - Migrar a HTTPS - Implementar JWT amb refresh tokens - Rate limiting per IP - Hashing de secrets

Race Conditions

Problema Potencial: Dos dispositius actualitzen simultàniament:

```
Temps    | Dispositiu A          | Dispositiu B
---------|-----------------------|----------------------
T0       | GET /bookmarks        | GET /bookmarks
         | → [b1, b2, b3]        | → [b1, b2, b3]
T1       | Afegir b4             | Afegir b5
T2       | POST [b1,b2,b3,b4]    | POST [b1,b2,b3,b5]
T3       | ✅ Guardat            | ✅ Guardat (SOBREESCRIU)
Result   | b4 ES PERD!           | Només b5 queda
```

Solució Actual: - Última escriptura guanya (last-write-wins) - Acceptable per ús personal amb 1 usuari

Millores Futures: - Timestamps per detectar conflictes - Merge automàtic basat en IDs únics - Versionat optimista (etags) - WebSockets per sync en temps real

---

Configuració del Servidor VPS

Fitxers del Servidor

Ubicació: `/root/ai-bookmarks-backend/` (o similar al VPS)

1. server.js

```
(#cb37-1)const express = require('express')
(#cb37-2)const cors = require('cors')
(#cb37-3)const fs = require('fs')
(#cb37-4)const path = require('path')
(#cb37-5)
(#cb37-6)const app = express()
(#cb37-7)const PORT = 3002
(#cb37-8)const DB_FILE = path.join(__dirname, 'db.json')
(#cb37-9)const API_SECRET = 'aAgYYud97Kp29Lif9u0i'
(#cb37-10)
(#cb37-11)// Middleware
(#cb37-12)app.use(cors())  // Permet requests des de qualsevol origen
(#cb37-13)app.use(express.json({ limit: '50mb' }))  // Parse JSON + límit gran
(#cb37-14)
(#cb37-15)// Auth Middleware
(#cb37-16)const checkAuth = (req, res, next) => {
(#cb37-17)  const secret = req.headers['x-api-secret']
(#cb37-18)  if (secret !== API_SECRET) {
(#cb37-19)    return res.status(403).json({ error: 'Unauthorized' })
(#cb37-20)  }
(#cb37-21)  next()
(#cb37-22)}
(#cb37-23)
(#cb37-24)app.use(checkAuth)
(#cb37-25)
(#cb37-26)// Helper functions
(#cb37-27)const readDB = () => {
(#cb37-28)  if (!fs.existsSync(DB_FILE)) {
(#cb37-29)    return { bookmarks: [], categories: [], deletedIds: [] }
(#cb37-30)  }
(#cb37-31)  return JSON.parse(fs.readFileSync(DB_FILE, 'utf8'))
(#cb37-32)}
(#cb37-33)
(#cb37-34)const writeDB = (data) => {
(#cb37-35)  const current = readDB()
(#cb37-36)  const newData = { ...current, ...data }
(#cb37-37)  fs.writeFileSync(DB_FILE, JSON.stringify(newData, null, 2))
(#cb37-38)}
(#cb37-39)
(#cb37-40)// Endpoints
(#cb37-41)app.get('/bookmarks', (req, res) => {
(#cb37-42)  const db = readDB()
(#cb37-43)  res.json({ data: db.bookmarks || [] })
(#cb37-44)})
(#cb37-45)
(#cb37-46)app.post('/bookmarks', (req, res) => {
```

```
(#cb37-47)  const { data } = req.body
(#cb37-48)  writeDB({ bookmarks: data })
(#cb37-49)  res.json({ success: true })
(#cb37-50)})
(#cb37-51)
(#cb37-52)app.get('/categories', (req, res) => {
(#cb37-53)  const db = readDB()
(#cb37-54)  res.json({ data: db.categories || [] })
(#cb37-55)})
(#cb37-56)
(#cb37-57)app.post('/categories', (req, res) => {
(#cb37-58)  const { data } = req.body
(#cb37-59)  writeDB({ categories: data })
(#cb37-60)  res.json({ success: true })
(#cb37-61)})
(#cb37-62)
(#cb37-63)app.get('/deleted', (req, res) => {
(#cb37-64)  const db = readDB()
(#cb37-65)  res.json({ data: db.deletedIds || [] })
(#cb37-66)})
(#cb37-67)
(#cb37-68)app.post('/deleted', (req, res) => {
(#cb37-69)  const { data } = req.body
(#cb37-70)  writeDB({ deletedIds: data })
(#cb37-71)  res.json({ success: true })
(#cb37-72)})
(#cb37-73)
(#cb37-74)app.post('/reset', (req, res) => {
(#cb37-75)  fs.writeFileSync(
(#cb37-76)    DB_FILE,
(#cb37-77)    JSON.stringify({
(#cb37-78)      bookmarks: [],
(#cb37-79)      categories: [],
(#cb37-80)      deletedIds: []
(#cb37-81)    })
(#cb37-82)  )
(#cb37-83)  res.json({ success: true })
(#cb37-84)})
(#cb37-85)
(#cb37-86)app.listen(PORT, () => {
(#cb37-87)  console.log(`Server running on port ${PORT}`)
(#cb37-88)})
```

2. package.json

```
{
  "name": "ai-bookmarks-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2"
  }
}
```

3. db.json (generat automàticament)

```
{
  "bookmarks": [
    {
      "id": "1862103456789012345-a7b3c9",
      "title": "Guia completa sobre RAG amb LlamaIndex",
      "description": "Aprèn a implementar...",
      "author": "@username",
      "originalLink": "https://twitter.com/i/web/status/1862103456789012345"
      "externalLinks": ["https://llamaindex.ai/..."],
      "category": "RAG",
      "createdAt": 1733328000000
    }
  ],
  "categories": [
    "Divulgació",
    "Agents",
    "Skills",
    "RAG",
    "Cursos",
    "Notícies",
    "Eines",
    "Altres"
  ],
  "deletedIds": [
    "1862104567890123456"
  ]
}
```

Instal·lació i Deploy

1. Instal·lar Node.js al VPS:

```
(#cb40-1)# Ubuntu/Debian
(#cb40-2)curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
(#cb40-3)sudo apt-get install -y nodejs
(#cb40-4)
(#cb40-5)# Verificar
(#cb40-6)node --version  # v20.x.x
(#cb40-7)npm --version   # 10.x.x
```

2. Crear directori i pujar fitxers:

```
(#cb41-1)mkdir -p /root/ai-bookmarks-backend
(#cb41-2)cd /root/ai-bookmarks-backend
(#cb41-3)
(#cb41-4)# Pujar server.js i package.json via SCP/FTP
(#cb41-5)# O crear manualment amb nano/vim
```

3. Instal·lar dependències:

```
(#cb42-1)npm install
```

4. Executar servidor:

```
(#cb43-1)# Directe (per testing)
(#cb43-2)npm start
(#cb43-3)
(#cb43-4)# Amb PM2 (recomanat per producció)
(#cb43-5)npm install -g pm2
(#cb43-6)pm2 start server.js --name ai-bookmarks
(#cb43-7)pm2 save
(#cb43-8)pm2 startup  # Configura autostart
```

5. Configurar firewall:

```
(#cb44-1)# UFW (Ubuntu)
(#cb44-2)sudo ufw allow 3002/tcp
(#cb44-3)sudo ufw reload
(#cb44-4)
(#cb44-5)# iptables
(#cb44-6)sudo iptables -A INPUT -p tcp --dport 3002 -j ACCEPT
```

6. (Opcional) Nginx reverse proxy:

```
# /etc/nginx/sites-available/ai-bookmarks
server {
    listen 80;
    server_name bookmarks.example.com;

    location / {
        proxy_pass http://localhost:3002;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Gestió del Servidor

Comandes PM2:

```
(#cb46-1)pm2 status              # Veure estat
(#cb46-2)pm2 logs ai-bookmarks   # Veure logs
(#cb46-3)pm2 restart ai-bookmarks  # Reiniciar
(#cb46-4)pm2 stop ai-bookmarks   # Aturar
(#cb46-5)pm2 delete ai-bookmarks # Eliminar
```

Backup de db.json:

```
(#cb47-1)# Backup manual
(#cb47-2)cp db.json db.json.backup
(#cb47-3)
(#cb47-4)# Backup automàtic diari
(#cb47-5)crontab -e
(#cb47-6)# Afegir: 0 2 * * * cp /root/ai-bookmarks-backend/db.json /root/backups/db-$(
```

Monitorització:

```
(#cb48-1)# Logs en temps real
(#cb48-2)pm2 logs ai-bookmarks --lines 100
(#cb48-3)
(#cb48-4)# Mètrics
(#cb48-5)pm2 monit
```

Detalls de Xarxa

IP Servidor: 62.169.25.188 **Port:** 3002 **Protocol:** HTTP (no HTTPS) **CORS:** Obert a tots els orígens (*)

URL Completa: `http://62.169.25.188:3002`

Testing endpoint:

```
(#cb49-1)# GET bookmarks
(#cb49-2)curl -H "x-api-secret: aAgYYud97Kp29Lif9u0i" \
(#cb49-3)     http://62.169.25.188:3002/bookmarks
(#cb49-4)
(#cb49-5)# POST bookmark
(#cb49-6)curl -X POST \
(#cb49-7)     -H "Content-Type: application/json" \
(#cb49-8)     -H "x-api-secret: aAgYYud97Kp29Lif9u0i" \
(#cb49-9)     -d '{"data":[{"id":"test","title":"Test"}]}' \
(#cb49-10)     http://62.169.25.188:3002/bookmarks
```

Estructures de Dades

TypeScript Types (types.ts)

1. TweetRaw - Format Twitter Export

```
(#cb50-1)interface TweetRaw {
(#cb50-2)  full_text?: string        // Text complet del tweet
(#cb50-3)  text?: string             // Camp alternatiu per text
(#cb50-4)  id_str?: string           // ID del tweet (string)
(#cb50-5)  id?: string               // ID alternatiu
(#cb50-6)  created_at?: string       // Data creació format Twitter
(#cb50-7)  user?: {
(#cb50-8)    name?: string           // Nom display de l'usuari
(#cb50-9)    screen_name?: string    // @handle
(#cb50-10)  }
(#cb50-11)  entities?: {
(#cb50-12)    urls?: Array<{
(#cb50-13)      expanded_url: string  // URLs expandides
(#cb50-14)    }>
(#cb50-15)  }
(#cb50-16)}
```

Exemple Real:

```
(#cb51-1){
(#cb51-2)  "full_text": "Just released a new tutorial on RAG...",
(#cb51-3)  "id_str": "1862103456789012345",
(#cb51-4)  "created_at": "Wed Dec 04 10:30:00 +0000 2024",
(#cb51-5)  "user": {
(#cb51-6)    "name": "AI Researcher",
(#cb51-7)    "screen_name": "ai_researcher"
(#cb51-8)  },
(#cb51-9)  "entities": {
(#cb51-10)    "urls": [
(#cb51-11)      { "expanded_url": "https://example.com/tutorial" }
(#cb51-12)    ]
(#cb51-13)  }
(#cb51-14)}
```

2. Bookmark - Format Intern Aplicació

```
(#cb52-1)interface Bookmark {
(#cb52-2)  id: string                // ID únic: {tweetId}-{random}
(#cb52-3)  title: string             // Títol generat per Gemini (català)
(#cb52-4)  description: string       // Text original (max 280 chars)
(#cb52-5)  author: string            // @username o nom display
(#cb52-6)  originalLink: string      // URL tweet: twitter.com/i/web/status/{id}
(#cb52-7)  externalLinks: string[]   // URLs extrets per Gemini
(#cb52-8)  category: string          // Categoria assignada
(#cb52-9)  createdAt: number         // Timestamp Unix (ms)
(#cb52-10)}
```

Exemple:

```
(#cb53-1){
(#cb53-2)  "id": "1862103456789012345-a7b3c9",
(#cb53-3)  "title": "Tutorial complet sobre implementació de RAG",
(#cb53-4)  "description": "Just released a new tutorial on RAG...",
(#cb53-5)  "author": "@ai_researcher",
(#cb53-6)  "originalLink": "https://twitter.com/i/web/status/1862103456789012345",
(#cb53-7)  "externalLinks": ["https://example.com/tutorial"],
(#cb53-8)  "category": "RAG",
(#cb53-9)  "createdAt": 1733313000000
(#cb53-10)}
```

3. ProcessedTweetResult - Resposta Gemini

```
interface ProcessedTweetResult {
  originalId: string        // ID del tweet processat
  isAI: boolean             // true si relacionat amb IA
  title: string             // Títol generat (buit si isAI=false)
  description: string       // NO USAT (sempre buit)
  category: string          // Categoria (buit si isAI=false)
  externalLinks: string[]   // URLs extrets ([] si isAI=false)
}
```

4. Category - Alias

```
type Category = string

// Valors permesos:
const DEFAULT_CATEGORIES = [
  "Divulgació",
  "Agents",
  "Skills",
  "RAG",
  "Cursos",
  "Notícies",
  "Eines",
  "Altres"
]
```

5. LogEntry - Sistema de Logs

```
interface LogEntry {
  timestamp: Date           // Moment del log
  type: 'info' | 'success' | 'warning' | 'error'
  message: string           // Text del missatge
}
```

Exemple:

```
{
  "timestamp": "2024-12-04T10:30:00.000Z",
  "type": "success",
  "message": "Processat tweet 15/20"
}
```

6. BackupFormat - Fitxer d'Exportació

```
interface BackupFormat {
  backupVersion: number     // Versió del format (actualment 1)
  timestamp: number         // Unix timestamp de l'export
  categories: Category[]    // Llista de categories
  bookmarks: Bookmark[]     // Array de bookmarks
  deletedIds: string[]      // IDs eliminats (blacklist)
}
```

Exemple:

```
{
  "backupVersion": 1,
  "timestamp": 1733313000000,
  "categories": ["Divulgació", "Agents", "RAG"],
  "bookmarks": [
    { "id": "...", "title": "...", ... }
  ],
  "deletedIds": ["1862104567890123456"]
}
```

Transformacions de Dades

TweetRaw → Bookmark:

```
(#cb60-1)function createBookmark(tweet: TweetRaw, aiResult: ProcessedTweetResult): Bo
(#cb60-2)  // 1. Extreure text
(#cb60-3)  const fullText = tweet.full_text || tweet.text || ''
(#cb60-4)  const truncated = fullText.length > 280
(#cb60-5)    ? fullText.substring(0, 280) + '[...]'
(#cb60-6)    : fullText
(#cb60-7)
(#cb60-8)  // 2. Extreure autor
(#cb60-9)  const author = tweet.user?.screen_name
(#cb60-10)    ? `@${tweet.user.screen_name}`
(#cb60-11)    : (tweet.user?.name || 'Unknown')
(#cb60-12)
(#cb60-13)  // 3. Crear link original
(#cb60-14)  const tweetId = tweet.id_str || tweet.id
(#cb60-15)  const originalLink = `https://twitter.com/i/web/status/${tweetId}`
(#cb60-16)
(#cb60-17)  // 4. Generar ID únic
(#cb60-18)  const id = `${tweetId}-${Math.random().toString(36).substr(2, 6)}`
(#cb60-19)
(#cb60-20)  // 5. Timestamp
(#cb60-21)  const createdAt = tweet.created_at
(#cb60-22)    ? new Date(tweet.created_at).getTime()
(#cb60-23)    : Date.now()
(#cb60-24)
(#cb60-25)  return {
(#cb60-26)    id,
(#cb60-27)    title: aiResult.title,
(#cb60-28)    description: truncated,
(#cb60-29)    author,
(#cb60-30)    originalLink,
(#cb60-31)    externalLinks: aiResult.externalLinks,
(#cb60-32)    category: aiResult.category,
(#cb60-33)    createdAt
(#cb60-34)  }
(#cb60-35)}
```

Característiques Especials

1. Disseny Brutalista

Filosofia: - Minimalista i funcional - Tipografia bold i gran - Borders gruixuts negres - Shadows dures (no difuminades) - Colors limitats: blanc, negre, groc - Sense gradients ni

transparències

Implementació CSS:

```
/* Base components */
.button {
  border: 2px solid #000;
  box-shadow: 4px 4px 0px 0px #000;
  font-weight: 800;
  transition: all 0.15s ease;
}

.button:hover {
  transform: translate(-2px, -2px);
  box-shadow: 6px 6px 0px 0px #000;
}

.button:active {
  transform: translate(0, 0);
  box-shadow: 0px 0px 0px 0px #000;
}

/* Card */
.card {
  border: 2px solid #000;
  box-shadow: 4px 4px 0px 0px #000;
  background: white;
}

/* Badge */
.badge {
  border: 2px solid #000;
  background: #ffc107;
  font-weight: 700;
  padding: 4px 12px;
}
```

Exemples Visuals:

```
┌──────────────────────────────────┐
|  [Divulgació]                    | ← Badge groc
|                                  |
|  Guia completa sobre RAG         | ← Title (bold)
|                                  |
|  Aprèn a implementar...          | ← Description
|                                  |
|  @username • 04/12/2024          | ← Metadata
|                                  |
|  🔗 example.com                  | ← External links
|                                  |
|  [✏️ Editar] [🗑️ Eliminar]       | ← Actions
└──────────────────────────────────┘
    └──────┘ ← Hard shadow
```

2. Sistema de Logs en Temps Real

Interfície de Consola:

```
═══════════════════════════════════════════
  Registre d'Importació
═══════════════════════════════════════════

[10:30:15] ℹ️  Iniciant procés d'importació...
[10:30:16] ℹ️  Detectat arxiu: twitter-bookmarks.json
[10:30:16] ℹ️  150 tweets trobats al fitxer
[10:30:17] ℹ️  120 tweets nous (30 duplicats ignorats)
[10:30:18] ⏳ Processant tweet 1/120...
[10:30:22] ✅ Tweet 1/120: Guia sobre RAG
[10:30:26] ✅ Tweet 2/120: Nou model d'OpenAI
...
[10:35:40] ⚠️  Rate limit detectat, esperant 10s...
[10:35:50] ⏳ Reintentant tweet 45/120...
[10:35:54] ✅ Tweet 45/120: Tutorial LangChain
...
[10:45:20] ✅ Procés finalitzat!
[10:45:20] ℹ️  95 tweets d'IA importats
[10:45:20] ℹ️  25 tweets rebutjats (no IA)
[10:45:20] ℹ️  30 duplicats ignorats

═══════════════════════════════════════════
```

Colors per Tipus: - 🔵 **Info:** Text blanc/gris - 🟢 **Success:** Text verd - 🟡 **Warning:** Text groc/taronja - 🔴 **Error:** Text vermell

Features: - Auto-scroll a últim log - Timestamp per cada entrada - Comptador de progrés (X/Y) - Botó “Aturar” per cancel·lar procés

3. Responsive Design Mòbil

Breakpoints:

```
(#cb64-1)/* Mobile */
(#cb64-2)@media (max-width: 768px) {
(#cb64-3)  /* 1 columna */
(#cb64-4)  .grid { grid-template-columns: 1fr; }
(#cb64-5)
(#cb64-6)  /* Menu hamburguesa */
(#cb64-7)  .desktop-sidebar { display: none; }
(#cb64-8)  .mobile-menu-button { display: block; }
(#cb64-9)}
(#cb64-10)
(#cb64-11)/* Tablet */
(#cb64-12)@media (min-width: 769px) and (max-width: 1024px) {
(#cb64-13)  /* 2 columnes */
(#cb64-14)  .grid { grid-template-columns: repeat(2, 1fr); }
(#cb64-15)}
(#cb64-16)
(#cb64-17)/* Desktop */
(#cb64-18)@media (min-width: 1025px) {
(#cb64-19)  /* 3-4 columnes */
(#cb64-20)  .grid { grid-template-columns: repeat(auto-fill, minmax(350px, 1fr)); }
(#cb64-21)}
```

Adaptacions Mòbil: 1. **Header:** - Logo + Hamburger button - Category button fix al centre top

2. **Sidebar:**
    - Modal overlay en pantalla completa
    - Llista vertical de categories
    - Botó tancar (X) dalt a la dreta
3. **Cards:**
    - Full width (1 columna)
    - Touch targets més grans (48x48px min)
    - Padding augmentat

4. **Modals:**
    - Full screen en mòbil
    - Padding reduït
    - Font size ajustat

4. Rate Limiting Intel·ligent

Problema: Gemini Free Tier = 15 RPM (requests per minute)

Solució Implementada:

```
(#cb65-1)// 1. Delay base entre requests
(#cb65-2)const BASE_DELAY = 4000  // 4 segons = ~15 RPM
(#cb65-3)
(#cb65-4)async function processBookmarksWithGemini(tweets) {
(#cb65-5)  for (let i = 0; i < tweets.length; i++) {
(#cb65-6)    try {
(#cb65-7)      // Process tweet
(#cb65-8)      const result = await generateContent(tweet)
(#cb65-9)
(#cb65-10)      // Wait abans del següent
(#cb65-11)      if (i < tweets.length - 1) {
(#cb65-12)        await sleep(BASE_DELAY)
(#cb65-13)      }
(#cb65-14)
(#cb65-15)    } catch (error) {
(#cb65-16)      // Handle 429 error
(#cb65-17)      if (is429Error(error)) {
(#cb65-18)        await handleRateLimitWithRetry(tweet, i)
(#cb65-19)      }
(#cb65-20)    }
(#cb65-21)  }
(#cb65-22)}
(#cb65-23)
(#cb65-24)// 2. Exponential backoff
(#cb65-25)async function handleRateLimitWithRetry(tweet, attempt) {
(#cb65-26)  const delays = [10, 15, 22.5, 33.75, 50.625, 60, 60, 60, 60, 60]
(#cb65-27)  const maxRetries = 10
(#cb65-28)
(#cb65-29)  if (attempt >= maxRetries) {
(#cb65-30)    throw new Error('Max retries exceeded')
(#cb65-31)  }
(#cb65-32)
(#cb65-33)  const waitTime = delays[attempt] * 1000
(#cb65-34)  addLog('warning', `Rate limit! Esperant ${delays[attempt]}s...`)
(#cb65-35)
(#cb65-36)  await sleep(waitTime)
(#cb65-37)
(#cb65-38)  // Retry
(#cb65-39)  return await generateContent(tweet)
(#cb65-40)}
```

Timeline Exemple:

```
T+0s:   Request 1 → Success
T+4s:   Request 2 → Success
T+8s:   Request 3 → Success
...
T+56s:  Request 15 → Success
T+60s:  Request 16 → 429 ERROR!
T+60s:  Wait 10s...
T+70s:  Retry request 16 → Success
T+74s:  Request 17 → Success
```

Detecció de 429:

```
(#cb67-1)function is429Error(error: any): boolean {
(#cb67-2)  // 1. Check error message
(#cb67-3)  if (error.message?.includes('429')) return true
(#cb67-4)
(#cb67-5)  // 2. Check status property
(#cb67-6)  if (error.status === 429) return true
(#cb67-7)
(#cb67-8)  // 3. Check response text
(#cb67-9)  if (error.toString().includes('RESOURCE_EXHAUSTED')) return true
(#cb67-10)
(#cb67-11)  return false
(#cb67-12)}
```

5. Sistema de Backup/Restore

Format de Backup:

```
(#cb68-1){
(#cb68-2)  "backupVersion": 1,
(#cb68-3)  "timestamp": 1733313000000,
(#cb68-4)  "categories": [...],
(#cb68-5)  "bookmarks": [...],
(#cb68-6)  "deletedIds": [...]
(#cb68-7)}
```

Export:

```
(#cb69-1)function exportBackupJSON() {
(#cb69-2)  const backup = {
(#cb69-3)    backupVersion: 1,
(#cb69-4)    timestamp: Date.now(),
(#cb69-5)    categories,
(#cb69-6)    bookmarks,
(#cb69-7)    deletedIds
(#cb69-8)  }
(#cb69-9)
(#cb69-10)  const blob = new Blob(
(#cb69-11)    [JSON.stringify(backup, null, 2)],
(#cb69-12)    { type: 'application/json' }
(#cb69-13)  )
(#cb69-14)
(#cb69-15)  const url = URL.createObjectURL(blob)
(#cb69-16)  const a = document.createElement('a')
(#cb69-17)  a.href = url
(#cb69-18)  a.download = `ai-bookmarks-backup-${Date.now()}.json`
(#cb69-19)  a.click()
(#cb69-20)}
```

Import/Merge:

```
(#cb70-1)function mergeBackup(backup: BackupFormat) {
(#cb70-2)  // 1. Merge categories (unió de sets)
(#cb70-3)  const newCategories = [
(#cb70-4)    ...new Set([...categories, ...backup.categories])
(#cb70-5)  ]
(#cb70-6)
(#cb70-7)  // 2. Merge bookmarks (deduplica per ID)
(#cb70-8)  const existingIds = new Set(bookmarks.map(b => b.id))
(#cb70-9)  const newBookmarks = backup.bookmarks.filter(
(#cb70-10)    b => !existingIds.has(b.id)
(#cb70-11)  )
(#cb70-12)
(#cb70-13)  // 3. Merge deletedIds (unió de sets)
(#cb70-14)  const newDeletedIds = [
(#cb70-15)    ...new Set([...deletedIds, ...backup.deletedIds])
(#cb70-16)  ]
(#cb70-17)
(#cb70-18)  // 4. Update state
(#cb70-19)  setCategories(newCategories)
(#cb70-20)  setBookmarks([...bookmarks, ...newBookmarks])
(#cb70-21)  setDeletedIds(newDeletedIds)
(#cb70-22)}
```

6. Sistema de Cerca i Filtratge

Filtratge per Categoria:

```
(#cb71-1)const filteredBookmarks = useMemo(() => {
(#cb71-2)  if (selectedCategory === 'Tots') {
(#cb71-3)    return bookmarks
(#cb71-4)  }
(#cb71-5)  return bookmarks.filter(b => b.category === selectedCategory)
(#cb71-6)}, [bookmarks, selectedCategory])
```

UI de Categories:

```
┌──────────────────────────────┐
|  [Tots (150)]                | ← Totes les categories
|  [Divulgació (45)]           |
|  [Agents (23)]               |
|  [Skills (18)]               |
|  [RAG (32)]                  |
|  [Cursos (12)]               |
|  [Notícies (8)]              |
|  [Eines (10)]                |
|  [Altres (2)]                |
└──────────────────────────────┘
```

Comptadors Dinàmics:

```
(#cb73-1)const categoryCounts = useMemo(() => {
(#cb73-2)  const counts: Record<string, number> = {}
(#cb73-3)
(#cb73-4)  bookmarks.forEach(b => {
(#cb73-5)    counts[b.category] = (counts[b.category] || 0) + 1
(#cb73-6)  })
(#cb73-7)
(#cb73-8)  return counts
(#cb73-9)}, [bookmarks])
(#cb73-10)
(#cb73-11)// Render: {category} ({categoryCounts[category] || 0})
```

Build i Deployment

Desenvolupament Local

1. Instal·lació:

```
(#cb74-1)cd ai-bookmarks
(#cb74-2)npm install
```

2. Configuració .env:

```
VITE_API_KEY=<gemini-api-key>
VITE_STORAGE_API_URL=http://62.169.25.188:3002
VITE_STORAGE_SECRET=aAgYYud97Kp29Lif9u0i
```

3. Executar dev server:

```
(#cb76-1)npm run dev
```

→ Servidor disponible a: `http://localhost:3000`

4. Verificar: - Frontend carrega correctament - Pot importar JSON de Twitter - Gemini processa tweets (API key vàlida) - Storage funciona (local o API segons .env)

Build de Producció

1. Compilar TypeScript + Build Vite:

```
(#cb77-1)npm run build
```

Sortida:

```
dist/
├── index.html
├── assets/
|   ├── index-[hash].js
|   ├── index-[hash].css
|   └── [altres assets]
└── vite.svg
```

2. Preview build local:

```
(#cb79-1)npm run preview
```

→ Servidor disponible a: `http://localhost:4173`

3. Optimitzacions aplicades: - Code splitting automàtic - Minificació JS/CSS - Tree shaking (elimina codi no usat) - Asset hashing per cache busting - Compression (gzip)

Deployment Frontend

Opció 1: Vercel (Recomanat)

```
(#cb80-1)# Instal·lar Vercel CLI
(#cb80-2)npm i -g vercel
(#cb80-3)
(#cb80-4)# Deploy
(#cb80-5)cd ai-bookmarks
(#cb80-6)vercel
(#cb80-7)
(#cb80-8)# Configurar variables d'entorn a Vercel dashboard:
(#cb80-9)# VITE_API_KEY
(#cb80-10)# VITE_STORAGE_API_URL
(#cb80-11)# VITE_STORAGE_SECRET
```

Opció 2: Netlify

```
(#cb81-1)# netlify.toml
(#cb81-2)[build]
(#cb81-3)  command = "npm run build"
(#cb81-4)  publish = "dist"
(#cb81-5)
(#cb81-6)# Deploy
(#cb81-7)netlify deploy --prod
```

Opció 3: VPS (mateix servidor que API)

```
# 1. Build local
npm run build

# 2. Pujar dist/ al VPS
scp -r dist/* root@62.169.25.188:/var/www/ai-bookmarks/

# 3. Configurar Nginx
# /etc/nginx/sites-available/ai-bookmarks-frontend
server {
    listen 80;
    server_name ai-bookmarks.example.com;
    root /var/www/ai-bookmarks;
    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }
}

# 4. Activar site
sudo ln -s /etc/nginx/sites-available/ai-bookmarks-frontend \
           /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

Opció 4: GitHub Pages

```
# package.json
{
  "scripts": {
    "deploy": "vite build && gh-pages -d dist"
  },
  "devDependencies": {
    "gh-pages": "^6.0.0"
  }
}

# vite.config.ts
export default {
  base: '/ai-bookmarks/',  // Nom del repo
  ...
}

# Deploy
npm run deploy
```

Deployment Backend

Ja cobert a secció "Configuració del Servidor VPS"

Resum: 1. VPS amb Node.js instal·lat 2. Pujar server.js, package.json 3. `npm install` 4. `pm2 start server.js --name ai-bookmarks` 5. Configurar firewall (port 3002) 6. (Opcional) Nginx reverse proxy

Variables d'Entorn per Producció

Frontend (.env.production):

```
VITE_API_KEY=<production-gemini-key>
VITE_STORAGE_API_URL=https://api.bookmarks.example.com
VITE_STORAGE_SECRET=<strong-random-secret>
```

Backend (server.js):

```
(#cb85-1)const API_SECRET = process.env.API_SECRET || 'default-secret'
(#cb85-2)const PORT = process.env.PORT || 3002
```

Recomanacions: - ✅ Usar HTTPS en producció - ✅ Secrets forts (min 32 chars random) - ✅ API keys diferents per dev/prod - ✅ Rate limiting al servidor - ✅ Backups automàtics de db.json

CI/CD (Opcional)

GitHub Actions (.github/workflows/deploy.yml):

```yaml
(#cb86-1)name: Deploy Frontend
(#cb86-2)
(#cb86-3)on:
(#cb86-4)  push:
(#cb86-5)    branches: [main]
(#cb86-6)
(#cb86-7)jobs:
(#cb86-8)  deploy:
(#cb86-9)    runs-on: ubuntu-latest
(#cb86-10)    steps:
(#cb86-11)      - uses: actions/checkout@v3
(#cb86-12)
(#cb86-13)      - name: Setup Node.js
(#cb86-14)        uses: actions/setup-node@v3
(#cb86-15)        with:
(#cb86-16)          node-version: '20'
(#cb86-17)
(#cb86-18)      - name: Install dependencies
(#cb86-19)        run: npm ci
(#cb86-20)
(#cb86-21)      - name: Build
(#cb86-22)        env:
(#cb86-23)          VITE_API_KEY: ${{ secrets.VITE_API_KEY }}
(#cb86-24)          VITE_STORAGE_API_URL: ${{ secrets.VITE_STORAGE_API_URL }}
(#cb86-25)          VITE_STORAGE_SECRET: ${{ secrets.VITE_STORAGE_SECRET }}
(#cb86-26)        run: npm run build
(#cb86-27)
(#cb86-28)      - name: Deploy to Vercel
(#cb86-29)        uses: amondnet/vercel-action@v25
(#cb86-30)        with:
(#cb86-31)          vercel-token: ${{ secrets.VERCEL_TOKEN }}
(#cb86-32)          vercel-org-id: ${{ secrets.ORG_ID }}
(#cb86-33)          vercel-project-id: ${{ secrets.PROJECT_ID }}
```

Conclusions i Recomanacions

Punts Forts del Projecte

1. **Arquitectura Modular:**
    - Separació clara entre UI, serveis i lògica de negoci
    - Components reutilitzables ben dissenyats

- Fàcil d'estendre i mantenir

2. **Integració IA Robusta:**
   - Rate limiting intel·ligent preveu errors
   - Retry amb exponential backoff
   - Logging detallat per debugging
3. **Storage Flexible:**
   - Funciona offline (localStorage)
   - Sincronització multi-dispositiu (API)
   - Canvi transparent segons configuració
4. **Experiència d'Usuari:**
   - Disseny brutalista distintiu i modern
   - Progress tracking en temps real
   - Responsive design mòbil-friendly
   - Sistema de backup/restore complet
5. **Integritat de Dades:**
   - Deduplicació automàtica
   - Blacklist d'eliminats prevé reimportacions
   - Validació TypeScript prevé errors

Possibles Millores Futures

Curts Termini

1. **Seguretat:**
   - Migrar a HTTPS
   - Implementar autenticació JWT
   - Rate limiting per IP al backend
   - Sanitització d'inputs
2. **UX:**
   - Cerca per text (títol/descripció)
   - Ordenació (data, autor, categoria)

- Vista compacta/expandida
- Dark mode

3. **Rendiment:**
   - Lazy loading d'imatges
   - Virtualització per llistes llargues (>1000 items)
   - Service Worker per cache offline
   - Compressió d'imatges
4. **Funcionalitats:**
   - Edició inline de títols
   - Assignació manual de categoria
   - Tags personalitzats
   - Notes privades per bookmark

Llarg Termini

1. **Escalabilitat:**
   - Migrar a base de dades real (PostgreSQL, MongoDB)
   - Pagination/infinite scroll
   - Full-text search (Elasticsearch)
   - CDN per assets
2. **Col·laboració:**
   - Compartició de col·leccions
   - Bookmarks públics/privats
   - Sistema de "m'agrada" o favorits
3. **Analytics:**
   - Dashboard d'estadístiques
   - Tendències de categories
   - Timeline d'imports
4. **Integracions:**
   - Sync automàtic amb Twitter API
   - Export a Notion, Obsidian, etc.

- Chrome extension per afegir bookmarks
- Mobile app (React Native)

5. **IA Avançada:**
   - Resums automàtics de threads
   - Recomanacions personalitzades
   - Clustering automàtic de temes
   - Extracció d'entitats (persones, empreses, conceptes)

Limitacions Actuals

1. **Seguretat Bàsica:**
   - HTTP en lloc de HTTPS
   - Secret compartit simple
   - Sense gestió d'usuaris
2. **Race Conditions:**
   - Last-write-wins en updates simultanis
   - No hi ha versionat o merge automàtic
3. **Limits de Gemini:**
   - 15 RPM en free tier (lent per imports grans)
   - Pot requerir API key de pagament per ús intens
4. **Storage Limitat:**
   - LocalStorage: ~10MB max
   - db.json pot créixer indefinidament
   - No hi ha garbage collection de deletedIds
5. **Offline Support:**
   - Mode API requereix internet sempre
   - No hi ha sync automàtic quan torna connexió

Recomanacions d'Ús

1. **Imports Grans:**
   - Dividir fitxers JSON en batches de <100 tweets

- Executar imports fora d'hores punta (Gemini rate limits)

2. **Backups:**
   - Exportar backup setmanalment
   - Guardar backups en múltiples llocs (Drive, Dropbox)
   - Verificar integritat després d'imports grans
3. **Manteniment VPS:**
   - Backups diaris de db.json (cron job)
   - Monitoritzar ús de disc
   - Revisar logs de PM2 regularment
4. **Optimització:**
   - Eliminar tweets obsolets ocasionalment
   - Consolidar categories similars
   - Revisar i actualitzar títols generats per IA

Contacte i Suport

Autor: Roger Masellas **Projecte:** AI Bookmark Manager **Repositori:** https://github.com/rogeraclaro/AI-Bookmark-Manager **Data Documentació:** 4 de Desembre de 2025

Llicència

Aquest projecte és d'ús personal. Tots els drets reservats.

Apèndix

A. Glossari de Termes

- **Bookmark:** Preferit o marcador de Twitter guardat per l'usuari

- **RAG:** Retrieval Augmented Generation, tècnica d'IA per millorar respostes amb context extern
- **Brutalisme:** Estil de disseny minimalista amb elements visuals crus i funcionals
- **Rate Limiting:** Limitació de peticions per unitat de temps
- **Exponential Backoff:** Estratègia de retry amb esperes progressivament més llargues
- **Blacklist:** Llista d'elements prohibits o exclosos
- **VPS:** Virtual Private Server, servidor virtual privat
- **CORS:** Cross-Origin Resource Sharing, mecanisme de seguretat per requests entre dominis
- **JWT:** JSON Web Token, estàndard d'autenticació
- **ETL:** Extract, Transform, Load, procés de transformació de dades

B. Comandes Ràpides

```
# Frontend
npm install          # Instal·lar dependències
npm run dev          # Dev server (localhost:3000)
npm run build        # Build producció → dist/
npm run preview      # Preview build local
npm run lint         # Verificar codi

# Backend (VPS)
pm2 start server.js --name ai-bookmarks  # Iniciar servidor
pm2 logs ai-bookmarks                    # Veure logs
pm2 restart ai-bookmarks                 # Reiniciar
pm2 stop ai-bookmarks                    # Aturar

# Backup (VPS)
cp db.json db.json.backup-$(date +%Y%m%d)  # Backup manual
```

C. Endpoints API Reference

Endpoint	Mètode	Body	Response	Auth
`/bookmarks`	GET	-	`{data: Bookmark[]}`	✓
`/bookmarks`	POST	`{data: Bookmark[]}`	`{success: true}`	✓
`/categories`	GET	-	`{data: string[]}`	✓
`/categories`	POST	`{data: string[]}`	`{success: true}`	✓
`/deleted`	GET	-	`{data: string[]}`	✓
`/deleted`	POST	`{data: string[]}`	`{success: true}`	✓
`/reset`	POST	-	`{success: true}`	✓

Auth Header Required:

```
x-api-secret: aAgYYud97Kp29Lif9u0i
```

D. Troubleshooting Comú

1. Error "Failed to fetch" al carregar: - Verificar que el servidor VPS està actiu: `pm2 status` - Verificar firewall permet port 3002: `sudo ufw status` - Comprovar VITE_STORAGE_API_URL al .env

2. Gemini API error 429: - Esperar uns minuts (rate limit temporal) - Reduir velocitat d'imports (augmentar BASE_DELAY) - Considerar upgrade a tier de pagament

3. LocalStorage quota exceeded: - Exportar backup i eliminar bookmarks antics - Activar mode API per storage il·limitat - Netejar dades del navegador i reimportar backup

4. Tweets duplicats després d'eliminar: - Verificar que deletedIds es guarda correctament - Comprovar que l'ID del tweet és correcte - Revisar logs del procés d'import

5. Servidor VPS no respon: - SSH al VPS i verificar `pm2 status` - Revisar logs: `pm2 logs ai-bookmarks --lines 50` - Reiniciar si cal: `pm2 restart ai-bookmarks` - Verificar espai en disc: `df -h`

Fi de la Documentació

Aquesta documentació cobreix tots els aspectes tècnics i funcionals de l'AI Bookmark Manager. Per qualsevol dubte o millora, consultar el repositori de GitHub o contactar amb l'autor.