TECHNICAL DOCUMENT — PRIOR ART / DOCUMENTO TECNICO — PRIOR ART

This document is published in both English and Italian. The English version is provided first, followed by the Italian version.

Questo documento è pubblicato sia in inglese che in italiano. La versione inglese è riportata per prima, seguita dalla versione italiana.

🇬🇧 ENGLISH VERSION

TECHNICAL DOCUMENT — PRIOR ART

System and Method for Pre-Association between Digital Licenses and Physical Media with Intermediary-Mediated Activation

Public Prior Art Disclosure

Purpose of this publication

This document is intentionally published to place the described technical solution in the public domain and to constitute prior art opposable to any future patent application claiming identical or substantially equivalent methods.

Authors: Connecting Beyond – Cioni Gianluca & team
Publication date: December 02, 2025
Official publication URL: https://github.com/G-Cioni/sys-arch-doc/

---

1. Introduction

This document describes an innovative system for the generation, management, and activation of digital licenses pre-associated with physical media, such as stainless steel plaques, keychains, NFC cards, bracelets, necklaces, frames, or any other Memory Link containing a QR Code or NFC tag.

The system is designed to enable license distribution through intermediaries (e.g., funeral agencies, referred to as “Collaborators” in the system), allowing the end user to autonomously activate the linked digital content through a simple QR Code scan or NFC tag reading.

This technology introduces an operational flow that is not present in currently available digital commemorative services and represents an original technical solution for managing offline-to-online memorial digital content.

Note on terminology: In this document, the terms “QR Code” and “NFC tag” are to be considered interchangeable. Both technologies allow access to the same system through scanning or reading a unique identifier (`link_key`).

---

2. Objective of the Invention

The objective of the system is to:

- Generate Memory Links (digital connections) that can be associated with physical media containing QR Codes or NFC tags
- Associate each Memory Link with a Memory License (digital license) that authorizes the creation of memorial content

- Allow delivery of the physical medium “already active” through an authorized intermediary (Collaborator)
- Enable license activation by the recipient through QR scan or NFC reading
- Eliminate the need to create an account or purchase online before owning the physical medium
- Manage the license lifecycle (claim, linking to memory, ownership management) in a centralized manner
- Ensure security, authenticity, and non-duplicability of Memory Links

In summary, the invention consists of a computer-implemented method that enables pre-association between a digital license and a physical medium identified by a `link_key`, distributed through an intermediary, with subsequent activation and claim by the end user through authentication, including state management, auto-assignment based on email, and unique linking to a digital memory.

---

3. System Components

3.1. License Management Backend

The system uses a PostgreSQL relational database (via Supabase) with the following main entities:

Table `memory_license`

Field	Type	Description
`id`	UUID	Unique license identifier (high entropy)
`created_at`	timestamp	Creation date
`updated_at`	timestamp	Last update date
`claimed_at`	timestamp	Date when the license was claimed by the user
`linked_to_memory_at`	timestamp	Date when the license was linked to a memory
`id_memory`	UUID	Reference to the created memory (nullable)
`id_user`	UUID	Reference to the owner user (nullable)
`intended_owner_email`	string	Email of the intended recipient user

`is_active`	boolean	License activation status
`source`	enum	License origin: `"collaborator"` or `"admin"`
`tribute_first_name`	string	First name of the commemorated person (optional)
`tribute_last_name`	string	Last name of the commemorated person (optional)

License States (derived from data)

The system determines the license state based on the fields present:

State	Condition
`unclaimed`	`id_user` is NULL
`claimed`	`id_user` is set, `id_memory` is NULL
`linked`	`id_user` and `id_memory` are both set

Table `memory_link`

Field	Type	Description
`id`	UUID	Unique link identifier
`link_key`	string	Unique key printed in the QR Code / NFC tag
`id_license`	UUID	Reference to the associated license
`id_memory`	UUID	Direct reference to the memory (synchronized with the license)
`id_owner`	UUID	Link owner (legacy, now managed via license)
`is_active`	boolean	Indicates if the link is active
`activated_at`	timestamp	Activation date
`link_key_is_immutable`	boolean	If true, the key cannot be modified

Table `memory_license_collaborator`

Associates licenses with Collaborators (intermediaries):

Field	Type	Description
`id`	UUID	Association identifier
`id_memory_license`	UUID	Reference to the license
`id_collaborator`	UUID	Reference to the Collaborator
`is_paid`	boolean	Indicates if the Collaborator has received payment
`paid_at`	timestamp	Payment date

Table `memory_link_collaborator`

Associates Memory Links with Collaborators for traceability:

Field	Type	Description
`id`	UUID	Association identifier
`id_memory_link`	UUID	Reference to the Memory Link
`id_collaborator`	UUID	Reference to the Collaborator
`is_paid`	boolean	Payment status

3.2. Memory Link Generator

The system generates Memory Links through:

- QR Code / NFC tag scanning in the administrative area
- API endpoint that automatically creates the `memory_link` record if it doesn't exist
- Each `link_key` is unique and validated through regex patterns to prevent injection attacks
- The system supports idempotent creation: if a link already exists, it is confirmed without duplicates

Creation process (actual code):

```
// From ensureMemoryLinkForKey in memoryLinks/index.ts
const ensureMemoryLinkForKey = async ({
  linkKey,
  memoryId,
  allowCreate = true,
}) => {
  const normalizedKey = linkKey.trim().replace(/\/+$/, "");

  // Search for existing link
  let memoryLink = await getMemoryLinkByKey({ linkKey: normalizedKey });

  // If it doesn't exist, create it
  if (!memoryLink && allowCreate) {
    const { data } = await supabaseServiceRoleClient
      .from("memory_link")
      .insert({
        id_memory: null,
        link_key: normalizedKey,
        link_key_is_immutable: true,
      })
      .select()
      .single();

    memoryLink = data;
  }

  return { record: memoryLink, created: wasCreated };
};
```

3.3. Physical Medium (Hardware)

The system supports any physical device defined as a Memory Link:

- Stainless steel plaques with engraved QR Code
- NFC cards
- Keychains
- Bracelets and necklaces
- Frames and decorative supports
- Any other object containing a readable QR Code or NFC tag

Each medium contains a unique `link_key` that points to the endpoint:

```
https://[domain]/memories/{link_key}
```

3.4. Authorized Intermediary (Collaborator)

In the system, intermediaries are called Collaborators and have the following characteristics:

Table `collaborator`

Field	Type	Description
`id`	UUID	Unique identifier
`id_user`	UUID	Reference to user account
`id_type`	number	Collaborator type (e.g., funeral agency)
`name`	string	Name/Company name
`phone_number`	string	Phone contact
`website_url`	string	Website
`address`	string	Address
`referral_code`	string	Unique referral code
`referred_by_collaborator_id`	UUID	Referring Collaborator

Collaborators can:

1. Receive pre-associated Memory Links through QR scanner in the dedicated area
2. Activate links on behalf of clients, entering:
    - Recipient's email
    - First and last name of the commemorated person
3. Monitor the status of all associated links through dashboard
4. Cannot see the contents of memories created by end users

3.5. End User

The end user interacts with the system through:

1. QR Code scan / NFC tag reading on the physical medium
2. Dynamic landing page showing the appropriate state

3. Authentication process (login or registration)
4. Automatic license claim on first authenticated access
5. Memory creation through guided wizard

---

4. Technical Process Description (Innovative Core)

Step 1: Memory Link Registration in the System

An administrator or the system itself registers new Memory Links through scanning:

```
// Endpoint: POST /api/admin/memory-links/ensure
export const ensureMemoryLinkRecord = async ({ linkKey }) => {
  const { record, created } = await ensureMemoryLinkForKey({
    linkKey: normalizedKey,
    memoryId: null,
  });

  return {
    id: record.id,
    linkKey: record.link_key,
    wasCreated: created,
  };
};
```

Resulting state:

- `memory_link.is_active = false`
- `memory_link.id_license = NULL`
- No license associated yet

Step 2: Memory Link Association with Collaborator

The Collaborator (funeral agency) scans the received QR Codes / NFC tags:

```
// Endpoint: POST /api/collaborators/{id}/link-memories
export const linkCollaboratorToMemoryLink = async ({
  linkKey,
  collaboratorId,
}) => {
```

```
  // Verify that the link exists
  const memoryLink = await getMemoryLinkByKey({ linkKey });

  // Verify it's not already associated with another Collaborator
  const existingLinks = await supabaseServiceRoleClient
    .from("memory_link_collaborator")
    .select("id_collaborator")
    .eq("id_memory_link", memoryLink.id);

  if (existingLinks.length > 0) {
    throw new ApiError(409, "Memory link already linked to a collaborator");
  }

  // Create the association
  await supabaseServiceRoleClient
    .from("memory_link_collaborator")
    .insert({
      id_memory_link: memoryLink.id,
      id_collaborator: collaboratorId,
    });
};
```

Resulting state:

- Memory Link associated with the Collaborator
- Still not active (no license)

Step 3: Memory Link Activation by the Collaborator

When the Collaborator delivers the physical medium to the client, they activate the link:

```
// Endpoint: POST /api/collaborators/{id}/activate-memory-link
export const activateMemoryLinkForCollaborator = async ({
  linkKey,
  collaboratorId,
  ownerEmail,
  tributeFirstName,
  tributeLastName,
}) => {
  // Verify that the link is associated with this Collaborator
  // Verify it's not already activated

  // Activate the link and create the license
  return await activateLinkAndCreateLicense({
    collaboratorId,
    memoryLink,
    ownerEmail,
    tributeFirstName,
    tributeLastName,
```

```
  });
};
```

The activation process (`activateLinkAndCreateLicense`):

```
const activateLinkAndCreateLicense = async ({
  collaboratorId,
  memoryLink,
  ownerEmail,
  tributeFirstName,
  tributeLastName,
}) => {
  const now = new Date().toISOString();

  // 1. Check if the user already exists
  let userId = null;
  const { data: authUsers } = await supabaseServiceRoleClient.auth.admin.listUsers({
    filters: { email: ownerEmail },
  });
  if (authUsers?.users?.length > 0) {
    userId = authUsers.users[0].id;
  }

  // 2. Reserve the link (atomic update with conditions)
  const { data: reservedLink } = await supabaseServiceRoleClient
    .from("memory_link")
    .update({
      is_active: true,
      activated_at: now,
    })
    .eq("id", memoryLink.id)
    .eq("is_active", false)  // Only if not already active
    .is("id_license", null)  // Only if it doesn't already have a license
    .select()
    .maybeSingle();

  // 3. Create the license
  const licenseRecord = await insertMemoryLicense({
    license: {
      created_at: now,
      id_user: userId,  // Auto-claim if user exists
      intended_owner_email: ownerEmail,
      tribute_first_name: tributeFirstName,
      tribute_last_name: tributeLastName,
      claimed_at: userId ? now : null,
      is_active: true,
      source: "collaborator",
    },
  });

  // 4. Link the license to the link
  await supabaseServiceRoleClient
    .from("memory_link")
```

```
    .update({ id_license: licenseRecord.id })
    .eq("id", memoryLink.id);

  // 5. Associate the license with the Collaborator
  await insertMemoryLicenseCollaborator({
    collaboratorId,
    licenseId: licenseRecord.id,
  });

  return { memoryLink, licenseRecord };
};
```

Resulting state:

- `memory_link.is_active = true`
- `memory_link.id_license = [license UUID]`
- `memory_license` created with:
    - `source = "collaborator"`
    - `intended_owner_email = [client's email]`
    - `id_user = [userId if exists, NULL otherwise]`
    - `claimed_at = [timestamp if auto-claimed, NULL otherwise]`
- `memory_license_collaborator` association created

Key technical point: The license is created at the moment of activation by the Collaborator, but the actual owner is determined only when the end user accesses it. If the user already exists in the system, the license is automatically assigned (auto-claim).

Step 4: Scanning by the End User

When the user scans the QR Code or reads the NFC tag:

```
// Endpoint: GET /api/memory-links/{key}
export const resolveMemoryLinkLanding = async ({
  linkKey,
  currentUserId,
}) => {
  // 1. Retrieve the link with the associated license
  const memoryLink = await getMemoryLinkWithLicenseByKey({ linkKey });

  // 2. Check link status
  if (!memoryLink) {
    return { status: "link_not_found" };
  }

  if (!memoryLink.is_active) {
```

```
    return { status: "link_inactive" };
  }

  const license = memoryLink.memory_license;

  // 3. If the license has no owner and the user is authenticated, attempt claim
  if (!license.id_user && currentUserId) {
    const claimResult = await claimMemoryLicenseOwnership({
      licenseId: license.id,
      userId: currentUserId,
    });

    if (claimResult.success) {
      return {
        status: "license_claimed_now",
        canCreateMemory: true,
        newlyClaimed: true,
      };
    }
  }

  // 4. Determine the appropriate state
  if (!license.id_user) {
    return {
      status: "license_available_auth_required",
      requiresAuth: true,
    };
  }

  if (license.id_user === currentUserId) {
    if (license.id_memory) {
      return {
        status: "license_owned_by_me_with_memory",
        redirectMemoryId: license.id_memory,
      };
    }
    return {
      status: "license_owned_by_me_no_memory",
      canCreateMemory: true,
    };
  }

  // License owned by another user
  if (license.id_memory) {
    return {
      status: "license_owned_by_other_with_memory",
      redirectMemoryId: license.id_memory,
    };
  }

  return { status: "license_owned_by_other_no_memory" };
};
```

Possible Landing Page States:

State	Description	Action
`link_not_found`	Non-existent link	Error message
`link_inactive`	Link not yet activated	Informational message
`license_available_auth_required`	License available, user not authenticated	Login/registration request
`license_claimed_now`	License just claimed by the user	Memory creation wizard
`license_owned_by_me_no_memory`	User is owner, memory not created	Memory creation wizard
`license_owned_by_me_with_memory`	User is owner, memory exists	Redirect to memory
`license_owned_by_other_no_memory`	License owned by others, memory not created	“In preparation” message
`license_owned_by_other_with_memory`	License owned by others, memory exists	Redirect to public memory

Step 5: License Claim

The claim process is atomic and race-condition safe:

```
export async function claimMemoryLicenseOwnership({
  licenseId,
  userId,
}) {
  const now = new Date().toISOString();

  // Conditional update: only if id_user is NULL
  const { data } = await supabaseClient
    .from("memory_license")
    .update({
      id_user: userId,
      claimed_at: now,
    })
    .eq("id", licenseId)
    .is("id_user", null)  // Atomic condition
    .select()
    .maybeSingle();
```

```
  if (!data) {
    // No rows updated = already claimed by others
    const existing = await getMemoryLicenseById({ licenseId });
    return existing
      ? { success: false, reason: "already_claimed" }
      : { success: false, reason: "not_found" };
  }

  return { success: true, license: data };
}
```

Step 6: Memory Creation

Once the user has claimed the license, they can create the memory:

```
// The frontend passes the memoryLicenseId to the creation wizard
// Endpoint: POST /api/memories

// License validation before creation
export const validateMemoryLicenseForCreation = async ({
  licenseId,
  userId,
}) => {
  const license = await getMemoryLicenseById({ licenseId });

  if (!license) throw new ApiError(404, "License not found");
  if (!license.id_user) throw new ApiError(403, "License not claimed");
  if (license.id_user !== userId) throw new ApiError(403, "Not your license");
  if (license.id_memory) throw new ApiError(409, "License already linked");

  return { valid: true };
};

// After memory creation, it is linked to the license
export async function linkMemoryLicenseToMemory({
  licenseId,
  memoryId,
}) {
  const now = new Date().toISOString();

  await supabaseClient
    .from("memory_license")
    .update({
      id_memory: memoryId,
      linked_to_memory_at: now,
    })
    .eq("id", licenseId);

  // Also synchronize the associated memory_links
```

```
  const { data: links } = await supabaseClient
    .from("memory_link")
    .select("id")
    .eq("id_license", licenseId);

  await Promise.all(
    links.map((link) =>
      updateMemoryLinkMemoryId({ linkId: link.id, memoryId })
    )
  );
}
```

---

5. Alternative Flow: License Creation by Admin

Administrators can create licenses directly without going through a Collaborator:

```
// Endpoint: POST /api/admin/licenses
export const createAdminLicense = async ({
  ownerEmail,
  tributeFirstName,
  tributeLastName,
}) => {
  // Auto-claim if user exists
  let userId = null;
  const { data: authUsers } = await supabaseServiceRoleClient.auth.admin.listUsers({
    filters: { email: ownerEmail },
  });
  if (authUsers?.users?.length > 0) {
    userId = authUsers.users[0].id;
  }

  const license = await insertMemoryLicense({
    license: {
      id_user: userId,
      intended_owner_email: ownerEmail,
      tribute_first_name: tributeFirstName,
      tribute_last_name: tributeLastName,
      claimed_at: userId ? now : null,
      is_active: true,
      source: "admin",  // Distinguishes from Collaborator activation
    },
  });

  return license;
};
```

---

6. Security and Anti-Cloning Protection

6.1. Input Validation

```
// From validation.ts
export function validateAndSanitizeLinkKey(key: string): string {
  const trimmed = key.trim();

  // Regex to prevent injection
  if (!/^[a-zA-Z0-9_-]+$/.test(trimmed)) {
    throw new ApiError(400, "Invalid link key format");
  }

  if (trimmed.length > 100) {
    throw new ApiError(400, "Link key too long");
  }

  return trimmed;
}
```

6.2. Race Condition Protection

- Use of conditional updates with `.is()` and `.eq()` clauses
- Atomic operations at database level
- Logging of race conditions for audit

6.3. Authentication

- JWT tokens with automatic refresh
- Row Level Security (RLS) on Supabase
- Endpoints protected with `requireSuperAdmin()` for admin operations

---

7. Technical Advantages of the Method

1. Eliminates the need for manual activation by the intermediary for the content
2. Allows scalable production and distribution of physical media
3. Extremely smooth offline → online flow through QR/NFC scanning
4. Ensures that only the final recipient can activate and modify
5. Complete traceability of Collaborator-License associations
6. Intelligent auto-claim for already registered users
7. Granular states for optimal UX in every scenario
8. Duplication protection through database constraints and validation

---

8. Applications

- Permanent digital memorials
- Offline-to-online onboarding for digital services
- Physical products linked to digital content
- Digital gifting systems with physical support
- Guaranteed authenticity of digital content through physical objects
- Commemorative services for funeral agencies

---

9. Originality Requirements (for legal use)

The described system presents:

Novelty: Currently there are no commemorative platforms with:

- License–Memory Link (QR/NFC) pre-association
- Deferred activation by the intermediary
- Intelligent email-based auto-claim
- Granular state system for the landing page

Inventive step: The method eliminates typical logistical steps in the sector:

- No configuration required by the intermediary for the content

- No online purchase necessary before owning the medium
- Automatic claim for existing users

Industrial application: Usable in a reproducible and scalable manner for:

- Any volume of physical media
- Any number of Collaborators
- Any geographic market

---

10. Conclusion

This publication constitutes prior art for any future attempt to patent:

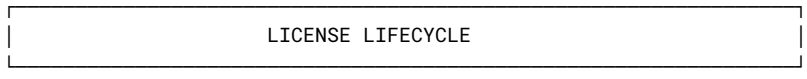
- Pre-association between digital licenses and physical media (QR Code / NFC tag)
- Offline-to-online provisioning through intermediary (Collaborator)
- The deferred activation method linked to memorial content
- The email-based auto-claim system
- The landing page state management for Memory Links
- The entire operational flow described

This disclosure makes unpatentable by third parties any invention that includes methods substantially equivalent or traceable to the steps described.

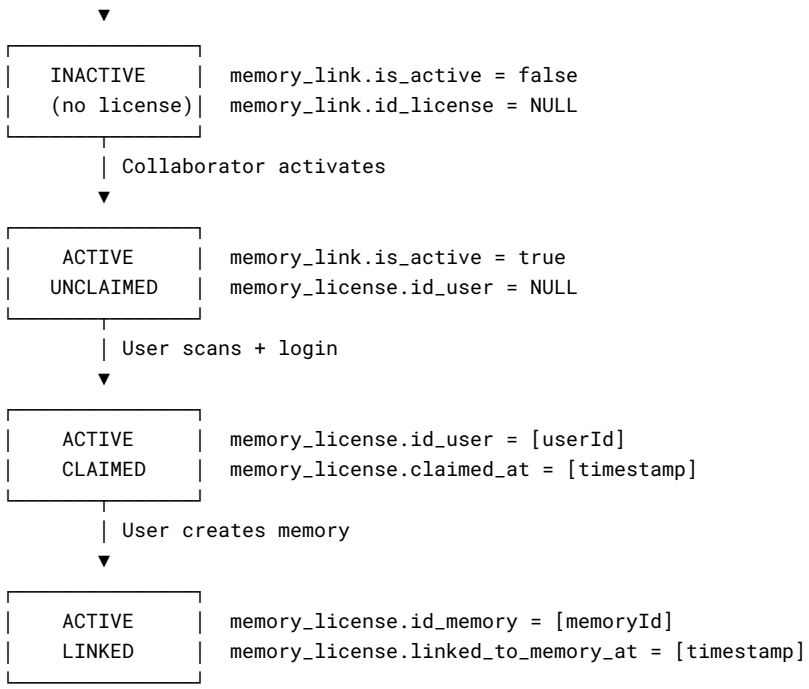
With the publication of this document on _____ at _____, Connecting Beyond makes the entire technical solution described publicly accessible and opposable.

Any future patent claiming identical or substantially equivalent processes must consider this disclosure as state of the art (prior art).

Appendix A: License State Diagram




```
[Memory Link Created]
        |
```



Appendix B: Main API Endpoints

Method	Endpoint	Description
GET	`/api/memory-links/{key}`	Resolves landing state for QR/NFC scan
PATCH	`/api/memory-links/{key}`	Links license to memory
POST	`/api/admin/memory-links/ensure`	Creates/confirms Memory Link
POST	`/api/admin/licenses`	Creates admin license
POST	`/api/collaborators/{id}/activate-memory-link`	Activates link for Collaborator
POST	`/api/collaborators/{id}/link-memories`	Associates link to Collaborator
GET	`/api/memories/pending-licenses`	Lists licenses available for memory creation

🇮🇹 VERSIONE ITALIANA

DOCUMENTO TECNICO — PRIOR ART

Sistema e Metodo di Pre-Associazione tra Licenze Digitali e Supporti Fisici con Attivazione Mediata da Intermediario

Public Prior Art Disclosure

Scopo di questa pubblicazione

Il presente documento è pubblicato intenzionalmente per rendere di pubblico dominio la soluzione tecnica descritta e costituire prior art opponibile a qualsiasi futura domanda di brevetto che rivendichi metodi uguali o sostanzialmente equivalenti.

Autori: Connecting Beyond – Cioni Gianluca & team
Data di pubblicazione: 02 dicembre 2025
URL ufficiale della pubblicazione: https://github.com/G-Cioni/sys-arch-doc

1. Introduzione

Questo documento descrive un sistema innovativo per la generazione, gestione e attivazione di licenze digitali pre-associate a supporti fisici, come targhe in acciaio inox, portachiavi, card NFC, bracciali, collane, quadri o qualsiasi altro Memory Link contenente QR Code o tag NFC.

Il sistema è progettato per consentire la distribuzione delle licenze tramite intermediari (es. agenzie funebri, denominate “Collaboratori” nel sistema), permettendo all’utente finale di attivare autonomamente il contenuto digitale collegato tramite una semplice scansione del QR Code o lettura del tag NFC.

Questa tecnologia introduce un flusso operativo che non risulta presente nei servizi commemorativi digitali attualmente disponibili e rappresenta una soluzione tecnica originale alla gestione di contenuti memoriali digitali offline-to-online.

Nota sulla terminologia: Nel presente documento, i termini “QR Code” e “tag NFC” sono da intendersi intercambiabili. Entrambe le tecnologie permettono l’accesso allo stesso sistema tramite la scansione o lettura di un identificatore univoco (`link_key`).

---

2. Obiettivo dell’Invenzione

L’obiettivo del sistema è:

- Generare Memory Link (collegamenti digitali) associabili a supporti fisici contenenti QR Code o tag NFC
- Associare ogni Memory Link a una Memory License (licenza digitale) che autorizza la creazione di contenuti memoriali
- Permettere la consegna del supporto fisico “già attivo” tramite un intermediario autorizzato (Collaboratore)
- Rendere possibile l’attivazione della licenza da parte del destinatario tramite scansione QR o lettura NFC
- Eliminare la necessità di creare un account o acquistare online prima di possedere il supporto fisico
- Gestire il lifecycle della licenza (claim, linking alla memoria, gestione proprietà) in modo centralizzato
- Garantire sicurezza, autenticità e non duplicabilità dei Memory Link

In sintesi, l’invenzione consiste in un metodo informatico che permette la pre-associazione tra una licenza digitale e un supporto fisico identificato da un `link_key`, distribuito tramite intermediario, con successiva attivazione e claim da parte dell’utente finale mediante autenticazione, includendo gestione degli stati, auto-assegnazione in base all’email, e collegamento univoco a una memoria digitale.

---

3. Componenti del Sistema

3.1. Backend di Gestione Licenze

Il sistema utilizza un database relazionale PostgreSQL (via Supabase) con le seguenti entità principali:

Tabella `memory_license`

Campo	Tipo	Descrizione
`id`	UUID	Identificatore univoco della licenza (ad alta entropia)
`created_at`	timestamp	Data di creazione
`updated_at`	timestamp	Data ultimo aggiornamento
`claimed_at`	timestamp	Data in cui la licenza è stata rivendicata dall'utente
`linked_to_memory_at`	timestamp	Data in cui la licenza è stata collegata a una memoria
`id_memory`	UUID	Riferimento alla memoria creata (nullable)
`id_user`	UUID	Riferimento all'utente proprietario (nullable)
`intended_owner_email`	string	Email dell'utente destinatario previsto
`is_active`	boolean	Stato di attivazione della licenza
`source`	enum	Origine della licenza: `"collaborator"` o `"admin"`
`tribute_first_name`	string	Nome della persona commemorata (opzionale)
`tribute_last_name`	string	Cognome della persona commemorata (opzionale)

Stati della Licenza (derivati dai dati)

Il sistema determina lo stato della licenza in base ai campi presenti:

Stato	Condizione

`unclaimed`	`id_user` è NULL
`claimed`	`id_user` è valorizzato, `id_memory` è NULL
`linked`	`id_user` e `id_memory` sono valorizzati

Tabella `memory_link`

Campo	Tipo	Descrizione
`id`	UUID	Identificatore univoco del link
`link_key`	string	Chiave univoca stampata nel QR Code / tag NFC
`id_license`	UUID	Riferimento alla licenza associata
`id_memory`	UUID	Riferimento diretto alla memoria (sincronizzato con la licenza)
`id_owner`	UUID	Proprietario del link (legacy, ora gestito via licenza)
`is_active`	boolean	Indica se il link è attivo
`activated_at`	timestamp	Data di attivazione
`link_key_is_immutable`	boolean	Se true, la chiave non può essere modificata

Tabella `memory_license_collaborator`

Associa le licenze ai Collaboratori (intermediari):

Campo	Tipo	Descrizione
`id`	UUID	Identificatore dell'associazione
`id_memory_license`	UUID	Riferimento alla licenza
`id_collaborator`	UUID	Riferimento al Collaboratore
`is_paid`	boolean	Indica se il Collaboratore ha ricevuto il pagamento
`paid_at`	timestamp	Data del pagamento

Tabella `memory_link_collaborator`

Associa i Memory Link ai Collaboratori per tracciabilità:

Campo	Tipo	Descrizione
`id`	UUID	Identificatore dell'associazione
`id_memory_link`	UUID	Riferimento al Memory Link
`id_collaborator`	UUID	Riferimento al Collaboratore
`is_paid`	boolean	Stato del pagamento

3.2. Generatore dei Memory Link

Il sistema genera Memory Link tramite:

- Scansione QR Code / tag NFC nell'area amministrativa
- Endpoint API che crea automaticamente il record `memory_link` se non esiste
- Ogni `link_key` è univoco e validato tramite pattern regex per prevenire injection attacks
- Il sistema supporta la creazione idempotente: se un link esiste già, viene confermato senza duplicati

Processo di creazione (codice reale):

```
// Da ensureMemoryLinkForKey in memoryLinks/index.ts
const ensureMemoryLinkForKey = async ({
  linkKey,
  memoryId,
  allowCreate = true,
}) => {
  const normalizedKey = linkKey.trim().replace(/\/+$/, "");

  // Cerca link esistente
  let memoryLink = await getMemoryLinkByKey({ linkKey: normalizedKey });

  // Se non esiste, lo crea
  if (!memoryLink && allowCreate) {
    const { data } = await supabaseServiceRoleClient
      .from("memory_link")
      .insert({
        id_memory: null,
        link_key: normalizedKey,
        link_key_is_immutable: true,
```

```
      })
      .select()
      .single();

    memoryLink = data;
  }

  return { record: memoryLink, created: wasCreated };
};
```

3.3. Supporto Fisico (Hardware)

Il sistema supporta qualsiasi dispositivo fisico definito come Memory Link:

- Targhe in acciaio inox con QR Code inciso
- Card NFC
- Portachiavi
- Bracciali e collane
- Quadri e supporti decorativi
- Qualsiasi altro oggetto contenente un QR Code leggibile o tag NFC

Ogni supporto contiene un `link_key` univoco che punta all'endpoint:

```
https://[dominio]/memories/{link_key}
```

3.4. Intermediario Autorizzato (Collaboratore)

Nel sistema, gli intermediari sono denominati Collaboratori e hanno le seguenti caratteristiche:

Tabella `collaborator`

Campo	Tipo	Descrizione
`id`	UUID	Identificatore univoco
`id_user`	UUID	Riferimento all'account utente
`id_type`	number	Tipo di Collaboratore (es. agenzia funebre)

`name`	string	Nome/Ragione sociale
`phone_number`	string	Contatto telefonico
`website_url`	string	Sito web
`address`	string	Indirizzo
`referral_code`	string	Codice referral univoco
`referred_by_collaborator_id`	UUID	Collaboratore referente

I Collaboratori possono:

1. Ricevere Memory Link pre-associati tramite scanner QR nell'area dedicata
2. Attivare i link per conto dei clienti, inserendo:
    - Email del destinatario
    - Nome e cognome della persona commemorata
3. Monitorare lo stato di tutti i link associati tramite dashboard
4. Non possono vedere i contenuti delle memorie create dagli utenti finali

3.5. Utente Finale

L'utente finale interagisce con il sistema attraverso:

1. Scansione del QR Code / lettura tag NFC sul supporto fisico
2. Landing page dinamica che mostra lo stato appropriato
3. Processo di autenticazione (login o registrazione)
4. Claim automatico della licenza al primo accesso autenticato
5. Creazione della memoria tramite wizard guidato

---

4. Descrizione del Processo Tecnico (Core Innovativo)

Step 1: Registrazione del Memory Link nel Sistema

Un amministratore o il sistema stesso registra nuovi Memory Link tramite scansione:

```
// Endpoint: POST /api/admin/memory-links/ensure
export const ensureMemoryLinkRecord = async ({ linkKey }) => {
  const { record, created } = await ensureMemoryLinkForKey({
    linkKey: normalizedKey,
    memoryId: null,
  });

  return {
    id: record.id,
    linkKey: record.link_key,
    wasCreated: created,
  };
};
```

Stato risultante:

- `memory_link.is_active = false`
- `memory_link.id_license = NULL`
- Nessuna licenza ancora associata

Step 2: Associazione del Memory Link al Collaboratore

Il Collaboratore (agenzia funebre) scansiona i QR Code / tag NFC ricevuti:

```
// Endpoint: POST /api/collaborators/{id}/link-memories
export const linkCollaboratorToMemoryLink = async ({
  linkKey,
  collaboratorId,
}) => {
  // Verifica che il link esista
  const memoryLink = await getMemoryLinkByKey({ linkKey });

  // Verifica che non sia già associato ad altro Collaboratore
  const existingLinks = await supabaseServiceRoleClient
    .from("memory_link_collaborator")
    .select("id_collaborator")
    .eq("id_memory_link", memoryLink.id);

  if (existingLinks.length > 0) {
    throw new ApiError(409, "Memory link already linked to a collaborator");
  }

  // Crea l'associazione
  await supabaseServiceRoleClient
    .from("memory_link_collaborator")
    .insert({
      id_memory_link: memoryLink.id,
      id_collaborator: collaboratorId,
    });
```

```
};
```

Stato risultante:

- Memory Link associato al Collaboratore
- Ancora non attivo (nessuna licenza)

Step 3: Attivazione del Memory Link da parte del Collaboratore

Quando il Collaboratore consegna il supporto fisico al cliente, attiva il link:

```
// Endpoint: POST /api/collaborators/{id}/activate-memory-link
export const activateMemoryLinkForCollaborator = async ({
  linkKey,
  collaboratorId,
  ownerEmail,
  tributeFirstName,
  tributeLastName,
}) => {
  // Verifica che il link sia associato a questo Collaboratore
  // Verifica che non sia già attivato

  // Attiva il link e crea la licenza
  return await activateLinkAndCreateLicense({
    collaboratorId,
    memoryLink,
    ownerEmail,
    tributeFirstName,
    tributeLastName,
  });
};
```

Il processo di attivazione (`activateLinkAndCreateLicense`):

```
const activateLinkAndCreateLicense = async ({
  collaboratorId,
  memoryLink,
  ownerEmail,
  tributeFirstName,
  tributeLastName,
}) => {
  const now = new Date().toISOString();

  // 1. Verifica se l'utente esiste già
  let userId = null;
  const { data: authUsers } = await supabaseServiceRoleClient.auth.admin.listUsers({
```

```
    filters: { email: ownerEmail },
  });
  if (authUsers?.users?.length > 0) {
    userId = authUsers.users[0].id;
  }

  // 2. Riserva il link (aggiornamento atomico con condizioni)
  const { data: reservedLink } = await supabaseServiceRoleClient
    .from("memory_link")
    .update({
      is_active: true,
      activated_at: now,
    })
    .eq("id", memoryLink.id)
    .eq("is_active", false)  // Solo se non già attivo
    .is("id_license", null)  // Solo se non ha già una licenza
    .select()
    .maybeSingle();

  // 3. Crea la licenza
  const licenseRecord = await insertMemoryLicense({
    license: {
      created_at: now,
      id_user: userId,  // Auto-claim se l'utente esiste
      intended_owner_email: ownerEmail,
      tribute_first_name: tributeFirstName,
      tribute_last_name: tributeLastName,
      claimed_at: userId ? now : null,
      is_active: true,
      source: "collaborator",
    },
  });

  // 4. Collega la licenza al link
  await supabaseServiceRoleClient
    .from("memory_link")
    .update({ id_license: licenseRecord.id })
    .eq("id", memoryLink.id);

  // 5. Associa la licenza al Collaboratore
  await insertMemoryLicenseCollaborator({
    collaboratorId,
    licenseId: licenseRecord.id,
  });

  return { memoryLink, licenseRecord };
};
```

Stato risultante:

- `memory_link.is_active = true`
- `memory_link.id_license = [UUID della licenza]`
- `memory_license` creata con:

    - `source = "collaborator"`
    - `intended_owner_email = [email del cliente]`
    - `id_user = [userId se esiste, NULL altrimenti]`
    - `claimed_at = [timestamp se auto-claimed, NULL altrimenti]`
- Associazione `memory_license_collaborator` creata

Punto tecnico chiave: La licenza viene creata nel momento dell'attivazione da parte del Collaboratore, ma il proprietario effettivo viene determinato solo quando l'utente finale accede. Se l'utente esiste già nel sistema, la licenza viene automaticamente assegnata (auto-claim).

Step 4: Scansione da parte dell'Utente Finale

Quando l'utente scansiona il QR Code o legge il tag NFC:

```
// Endpoint: GET /api/memory-links/{key}
export const resolveMemoryLinkLanding = async ({
  linkKey,
  currentUserId,
}) => {
  // 1. Recupera il link con la licenza associata
  const memoryLink = await getMemoryLinkWithLicenseByKey({ linkKey });

  // 2. Verifica stato del link
  if (!memoryLink) {
    return { status: "link_not_found" };
  }

  if (!memoryLink.is_active) {
    return { status: "link_inactive" };
  }

  const license = memoryLink.memory_license;

  // 3. Se la licenza non ha proprietario e l'utente è autenticato, tenta il claim
  if (!license.id_user && currentUserId) {
    const claimResult = await claimMemoryLicenseOwnership({
      licenseId: license.id,
      userId: currentUserId,
    });

    if (claimResult.success) {
      return {
        status: "license_claimed_now",
        canCreateMemory: true,
        newlyClaimed: true,
      };
    }
  }
```

```
  // 4. Determina lo stato appropriato
  if (!license.id_user) {
    return {
      status: "license_available_auth_required",
      requiresAuth: true,
    };
  }

  if (license.id_user === currentUserId) {
    if (license.id_memory) {
      return {
        status: "license_owned_by_me_with_memory",
        redirectMemoryId: license.id_memory,
      };
    }
    return {
      status: "license_owned_by_me_no_memory",
      canCreateMemory: true,
    };
  }

  // Licenza di proprietà di un altro utente
  if (license.id_memory) {
    return {
      status: "license_owned_by_other_with_memory",
      redirectMemoryId: license.id_memory,
    };
  }

  return { status: "license_owned_by_other_no_memory" };
};
```

Stati possibili della Landing Page:

Stato	Descrizione	Azione
`link_not_found`	Link inesistente	Messaggio di errore
`link_inactive`	Link non ancora attivato	Messaggio informativo
`license_available_auth_required`	Licenza disponibile, utente non autenticato	Richiesta login/registrazione
`license_claimed_now`	Licenza appena rivendicata dall'utente	Wizard creazione memoria
`license_owned_by_me_no_memory`	Utente proprietario, memoria non creata	Wizard creazione memoria

`license_owned_by_me_with_memory`	Utente proprietario, memoria esistente	Redirect alla memoria
`license_owned_by_other_no_memory`	Licenza di altri, memoria non creata	Messaggio “in preparazione”
`license_owned_by_other_with_memory`	Licenza di altri, memoria esistente	Redirect alla memoria pubblica

Step 5: Claim della Licenza

Il processo di claim è atomico e race-condition safe:

```
export async function claimMemoryLicenseOwnership({
  licenseId,
  userId,
}) {
  const now = new Date().toISOString();

  // Aggiornamento condizionale: solo se id_user è NULL
  const { data } = await supabaseClient
    .from("memory_license")
    .update({
      id_user: userId,
      claimed_at: now,
    })
    .eq("id", licenseId)
    .is("id_user", null)  // Condizione atomica
    .select()
    .maybeSingle();

  if (!data) {
    // Nessuna riga aggiornata = già claimed da altri
    const existing = await getMemoryLicenseById({ licenseId });
    return existing
      ? { success: false, reason: "already_claimed" }
      : { success: false, reason: "not_found" };
  }

  return { success: true, license: data };
}
```

Step 6: Creazione della Memoria

Una volta che l’utente ha il claim della licenza, può creare la memoria:

```
// Il frontend passa il memoryLicenseId al wizard di creazione
// Endpoint: POST /api/memories

// Validazione della licenza prima della creazione
export const validateMemoryLicenseForCreation = async ({
  licenseId,
  userId,
}) => {
  const license = await getMemoryLicenseById({ licenseId });

  if (!license) throw new ApiError(404, "License not found");
  if (!license.id_user) throw new ApiError(403, "License not claimed");
  if (license.id_user !== userId) throw new ApiError(403, "Not your license");
  if (license.id_memory) throw new ApiError(409, "License already linked");

  return { valid: true };
};

// Dopo la creazione della memoria, viene collegata alla licenza
export async function linkMemoryLicenseToMemory({
  licenseId,
  memoryId,
}) {
  const now = new Date().toISOString();

  await supabaseClient
    .from("memory_license")
    .update({
      id_memory: memoryId,
      linked_to_memory_at: now,
    })
    .eq("id", licenseId);

  // Sincronizza anche i memory_link associati
  const { data: links } = await supabaseClient
    .from("memory_link")
    .select("id")
    .eq("id_license", licenseId);

  await Promise.all(
    links.map((link) =>
      updateMemoryLinkMemoryId({ linkId: link.id, memoryId })
    )
  );
}
```

---

5. Flusso Alternativo: Creazione Licenza da Admin

Gli amministratori possono creare licenze direttamente senza passare per un Collaboratore:

```
// Endpoint: POST /api/admin/licenses
export const createAdminLicense = async ({
  ownerEmail,
  tributeFirstName,
  tributeLastName,
}) => {
  // Auto-claim se l'utente esiste
  let userId = null;
  const { data: authUsers } = await supabaseServiceRoleClient.auth.admin.listUsers({
    filters: { email: ownerEmail },
  });
  if (authUsers?.users?.length > 0) {
    userId = authUsers.users[0].id;
  }

  const license = await insertMemoryLicense({
    license: {
      id_user: userId,
      intended_owner_email: ownerEmail,
      tribute_first_name: tributeFirstName,
      tribute_last_name: tributeLastName,
      claimed_at: userId ? now : null,
      is_active: true,
      source: "admin",  // Distingue dall'attivazione via Collaboratore
    },
  });

  return license;
};
```

---

6. Sicurezza e Protezione Anti-Cloning

6.1. Validazione degli Input

```
// Da validation.ts
export function validateAndSanitizeLinkKey(key: string): string {
  const trimmed = key.trim();

  // Regex per prevenire injection
  if (!/^[a-zA-Z0-9_-]+$/.test(trimmed)) {
    throw new ApiError(400, "Invalid link key format");
  }
```

```
  if (trimmed.length > 100) {
    throw new ApiError(400, "Link key too long");
  }

  return trimmed;
}
```

6.2. Protezione Race Condition

- Utilizzo di update condizionali con clausole `.is()` e `.eq()`
- Operazioni atomiche a livello database
- Logging delle race condition per audit

6.3. Autenticazione

- Token JWT con refresh automatico
- Row Level Security (RLS) su Supabase
- Endpoint protetti con `requireSuperAdmin()` per operazioni admin

---

7. Vantaggi Tecnici del Metodo

1. Elimina la necessità di attivazione manuale da parte dell'intermediario per il contenuto
2. Permette produzione e distribuzione scalabile dei supporti fisici
3. Flusso offline → online estremamente fluido tramite scansione QR/NFC
4. Garantisce che solo il destinatario finale possa attivare e modificare
5. Tracciabilità completa delle associazioni Collaboratore-Licenza
6. Auto-claim intelligente per utenti già registrati
7. Stati granulari per UX ottimale in ogni scenario
8. Protezione da duplicazione tramite vincoli database e validazione

---

8. Applicazioni

- Memoriali digitali permanenti
- Onboarding offline-to-online per servizi digitali
- Prodotti fisici collegati a contenuti digitali
- Sistemi di gifting digitali con supporto fisico
- Autenticità garantita di contenuti digitali tramite oggetti fisici
- Servizi commemorativi per agenzie funebri

---

9. Requisiti di Originalità (per uso legale)

Il sistema descritto presenta:

Novità: Attualmente non esistono piattaforme commemorative con:

- Pre-associazione licenze–Memory Link (QR/NFC)
- Attivazione differita da parte dell'intermediario
- Auto-claim intelligente basato su email
- Sistema di stati granulare per la landing page

Inventive step: Il metodo elimina passaggi logistici tipici del settore:

- Nessuna configurazione richiesta all'intermediario per i contenuti
- Nessun acquisto online necessario prima del possesso del supporto
- Claim automatico per utenti esistenti

Applicazione industriale: Utilizzabile in modo riproducibile e scalabile per:

- Qualsiasi volume di supporti fisici
- Qualsiasi numero di Collaboratori
- Qualsiasi mercato geografico

---

10. Conclusione

La presente pubblicazione costituisce prior art per qualsiasi tentativo futuro di brevettare:

- La pre-associazione tra licenze digitali e supporti fisici (QR Code / tag NFC)
- Il provisioning offline-to-online tramite intermediario (Collaboratore)

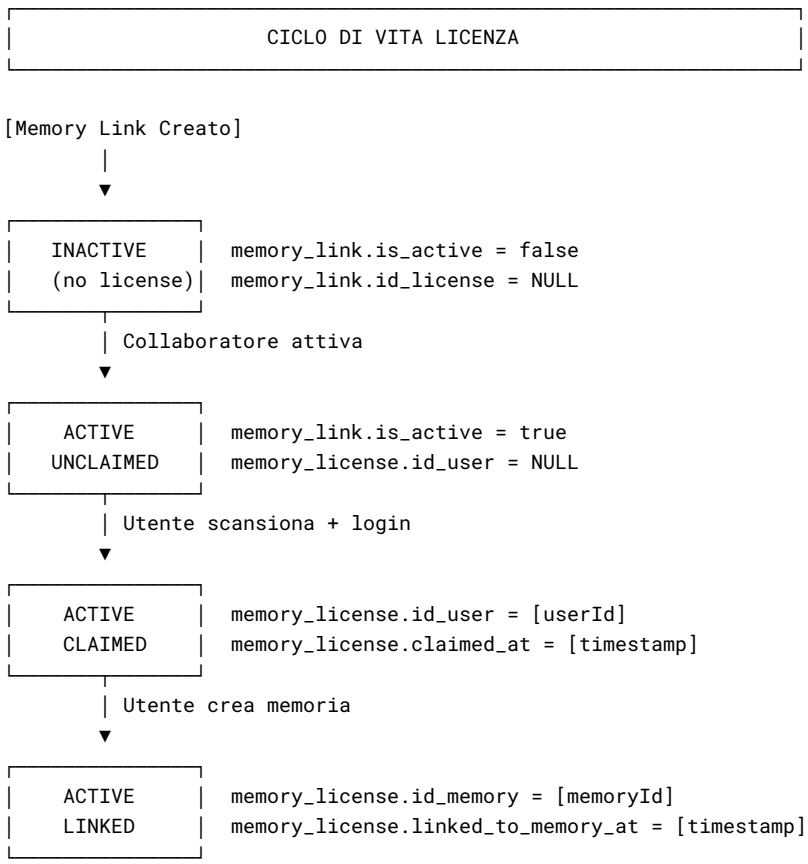
- Il metodo di attivazione differita collegato a contenuti memoriali
- Il sistema di auto-claim basato su email
- La gestione degli stati della landing page per Memory Link
- L'intero flusso operativo descritto

Questa divulgazione rende non brevettabile da parte di terzi qualsiasi invenzione che includa metodi sostanzialmente equivalenti o riconducibili agli step descritti.

Con la pubblicazione di questo documento in data _____ su _____, Connecting Beyond rende pubblicamente accessibile e opponibile l'intera soluzione tecnica descritta.

Qualsiasi brevetto futuro che rivendichi processi uguali o sostanzialmente equivalenti dovrà considerare questa disclosure come stato dell'arte (prior art).

Appendice A: Diagramma degli Stati della Licenza



Appendice B: Endpoint API Principali

Metodo	Endpoint	Descrizione
GET	`/api/memory-links/{key}`	Risolve stato landing per scansione QR/NFC
PATCH	`/api/memory-links/{key}`	Collega licenza a memoria
POST	`/api/admin/memory-links/ensure`	Crea/conferma Memory Link
POST	`/api/admin/licenses`	Crea licenza admin
POST	`/api/collaborators/{id}/activate-memory-link`	Attiva link per Collaboratore
POST	`/api/collaborators/{id}/link-memories`	Associa link a Collaboratore
GET	`/api/memories/pending-licenses`	Lista licenze disponibili per creazione memoria