生物樣品管理系統優化文件

一、文件目的

本文件旨在說明現有「生物樣品管理系統」之整體優化與升級規劃，涵蓋 資料庫結構升級、系統效能與安全強化 以及 前後端網頁架構升級，以提升系統穩定性、擴充性與長期維運能力。

二、現況概述

2.1 系統功能現況

- 樣品建檔與基本屬性管理
- 樣品位置（箱 / 格 / 冷凍櫃）管理
- 樣品進出紀錄（IN / OUT）
- 簡易查詢與列表顯示

2.2 系統架構優化更為嚴謹

- 前後端責任邊界不夠清晰：資料驗證、狀態計算與流程控制容易散落在多處，導致維護成本上升
- 後端分層不足：缺少明確的 服務層（Service Layer，負責封裝商業邏輯）/ Domain 邏輯封裝，使 控制器（Controller，負責請求接收與回應）可能逐步膨脹
- 交易紀錄與主資料耦合度高：樣品主檔與異動紀錄未充分解耦，追溯與統計需求擴大時容易受限
- 資料庫擴充與查詢效能風險：欄位逐年增加、常用查詢缺少一致索引策略，資料量成長時效能下降
- 版本化與擴充機制不足：應用程式介面版本控管（API Versioning）、DTO 契約與錯誤回應格式未統一，影響前端迭代與外部整合

三、資料庫升級與修改規劃

3.1 資料庫設計原則

- 主資料 / 交易資料分離
- 可追溯性（**Traceability**）優先
- 避免重複欄位，採用正規化設計
- 為未來 **AI /** 統計分析預留結構

3.2 核心資料表調整

(1) 樣品主檔（**Samples**）

- SampleId（PK）
- SampleCode（唯一編碼）
- SampleType
- Status（Active / Archived / Disposed）
- CreatedAt / UpdatedAt

(2) 樣品屬性表（**Sample_**中繼資料（**Metadata**））

- 中繼資料（Metadata）Id（PK）
- SampleId（FK）
- Key
- Value
- ValueType

優點：可因應不同研究計畫、物種、實驗需求彈性擴充

(3) 位置結構表（**Storage_Location**）

- LocationId（PK）
- ParentLocationId（階層式結構）
- LocationType（Freezer / Box / Slot）
- Code

(4) 樣品交易紀錄（**Sample_Transactions**）

- TransactionId（PK）
- SampleId（FK）
- ActionType（IN / OUT / MOVE / SPLIT）
- FromLocationId
- ToLocationId
- Operator
- ActionTime

3.3 索引與效能優化

- SampleCode、ActionTime 建立索引
- 常用查詢使用 複合式索引（Composite Index）
- 歷史交易資料依年度分區（資料分割（Partition））

四、網頁與系統架構升級

4.1 整體架構調整

```
None
[ Frontend (Vue / Angular) ]
          ↓ RESTful 應用程式介面（Representational State
Transfer Application Programming Interface）
[ Backend API (.NET / Flask) ]
          ↓
[ Database (MariaDB / SQL Server) ]
```

4.2 前端升級方向

- 採用 **Component-based UI**
- 表單與列表模組化（Sample Form / Location Picker）
- 支援大量資料的 虛擬捲動技術（Virtual Scrolling）/ 分頁機制（Pagination）
- UI 狀態統一管理（Pinia / NgRx）

4.3 後端升級方向

- 採用 服務層（**Service Layer**，負責封裝商業邏輯）分離商業邏輯
- Transaction 與 Sample 分開 控制器（Controller，負責請求接收與回應）
- 明確定義 資料傳輸物件（**Data Transfer Object, DTO**）與 畫面回傳模型（**View Model**），區分 API 傳輸結構與前端顯示需求
- 加入 應用程式介面版本控管（API Versioning）（v1 / v2）

五、安全與權限設計

5.1 權限角色

- Admin：系統設定、刪除
- Researcher：樣品操作
- Viewer：唯讀查詢

5.2 記錄與稽核

- 所有異動寫入 稽核紀錄（Audit Log）
- 關鍵操作不可 實體刪除（Hard Delete）

六、CSV 匯入功能規劃（新增）

本系統將提供「批次匯入」能力，以利研究計畫初始建檔、歷史資料移轉、或跨系統整併。匯入功能需兼顧 資料正確性、可追溯性、可回滾 與 使用者體驗。

6.1 匯入範圍與資料型態

建議先提供以下 CSV 類型（由簡到難逐步上線）：

1. 樣品主檔匯入（**Samples**）
    - 必填：SampleCode（若採系統產生則可空）、SampleType、Status（預設 Active）
    - 選填：描述、來源、批次資訊、採集日期等（可進入 中繼資料（Metadata））
2. 樣品屬性匯入（**Sample_**中繼資料（**Metadata**））
    - 欄位：SampleCode 或 SampleId、Key、Value、ValueType
3. 位置結構匯入（**Storage_Location**）
    - 欄位：Code、LocationType、ParentCode（或 ParentLocationId）
4. 交易匯入（**Sample_Transactions**）（建議第二階段上線）
    - 欄位：SampleCode、ActionType、FromLocationCode、ToLocationCode、Operator、ActionTime

6.2 匯入流程（**UI / API**）

UI 流程（建議 **4** 步驟 精靈式操作流程（精靈式操作流程（**Wizard**））

1. 上傳 CSV
2. 欄位對應（Mapping）：CSV 欄位 ↔ 系統欄位
3. 預檢（Preview & Validate）：顯示錯誤行、可下載錯誤報表
4. 正式匯入（Import）：顯示進度、結果摘要、可回滾

API 流程（非同步 **Job**）

- POST /api/imports （上傳並建立 匯入作業任務（Import Job））
- GET /api/imports/{jobId}（查詢進度與結果）
- POST /api/imports/{jobId}/commit（提交寫入）
- POST /api/imports/{jobId}/rollback（回滾/取消）

建議以「先落地暫存表 → 驗證 → Commit」模式，避免半套資料寫入主表。

6.3 匯入驗證規則（必做）

- 必填欄位檢查（空值、格式）
- 重複檢查（SampleCode、LocationCode）
- 參照完整性（Location 需存在、ParentCode 需存在）
- 列舉型別（Enumeration, Enum）檢查（ActionType、Status、LocationType）
- 時間格式（ActionTime）與時區統一（建議 UTC 儲存、前端顯示轉換）

6.4 匯入記錄與稽核

- 匯入作業任務（Import Job）：建立者、時間、原始檔名、檔案雜湊（Hash）、匯入摘要
- ImportRowError：行號、錯誤欄位、錯誤原因
- 重要：匯入造成的資料異動亦需寫入 稽核紀錄（Audit Log）（可標記來源為 匯入作業任務（Import Job）Id）

七、唯一碼（**Unique ID / SampleCode**）建置策略

系統需同時具備：

- 內部主鍵（**SampleId**）：資料庫用、不可變
- 外部可見唯一碼（**SampleCode**）：人員掃碼、貼標、跨系統交換用

7.1 主鍵策略（**SampleId**）

- 建議採用 通用唯一識別碼（**Universally Unique Identifier, UUID**）/ 全域唯一識別碼（**Globally Unique Identifier, GUID**）（或 可排序唯一識別碼（**Universally Unique Lexicographically Sortable Identifier, ULID**））作為 SampleId
    - 優點：分散式產生、跨系統整併不易衝突
    - 若使用 SQL Server：uniqueidentifier
    - 若使用 MariaDB：CHAR(36) 或 BINARY(16)

  若你希望排序友好（依時間遞增），可考慮 可排序唯一識別碼（Universally Unique Lexicographically Sortable Identifier, ULID）（字串 26 長度）或 全域唯一識別碼（Globally Unique Identifier, GUID）v7。

7.2 外部唯一碼（**SampleCode**）規格

設計原則

- 具可讀性、可貼標、可掃碼
- 不暴露敏感資訊
- 長度適中（建議 12～20 字元）
- 系統保證唯一（Unique Index）

推薦格式（範例）

- 方案 A（簡潔）：SM-{YYYYMMDD}-{SEQ4}
    - 例：SM-20260114-0042
- 方案 B（含計畫代碼）：{Project}-{YYMM}-{SEQ5}
    - 例：P01-2601-00042
- 方案 C（無序列、靠隨機）：SM-{可排序唯一識別碼（Universally Unique Lexicographically Sortable Identifier, ULID）}
    - 例：SM-01J1QZ3M8H2Z0K9QW3D6T7Y8X9

  若需要「同日連號」需求，方案 A/B 會最符合現場操作直覺。

7.3 SampleCode 產生機制（避免衝突）

資料庫層保證

- Samples.SampleCode 建立 UNIQUE INDEX

序列產生（方案 **A/B**）建議做法

- 建立 Sequence/Counter 表：Code_Counters
    1. Key（例如：20260114 或 Project+YYMM）
    2. CurrentValue
- 產生流程：
    1. 開啟交易（Transaction）
    2. SELECT ... FOR UPDATE（鎖定該 Key）
    3. CurrentValue + 1
    4. 組合 SampleCode
    5. 寫入 Samples
    6. Commit

這可避免多使用者同時建檔造成重複碼。

7.4 CSV 匯入時的 SampleCode 規則

- 若 CSV 提供 SampleCode：
    - 系統檢查是否重複；重複則該行視為錯誤或改為更新模式（依設定）
- 若 CSV 未提供 SampleCode：
    - 系統自動依規格產生 SampleCode

7.5 唯一碼與 快速回應碼（Quick Response Code, QR Code）/條碼（Barcode）貼標

- 建議 快速回應碼（Quick Response Code, QR Code） 內容：SampleCode（或 SampleId + 校驗碼）
- 貼標建議欄位：SampleCode、SampleType、日期、LocationCode（可選）
- 之後若要做「離線掃碼」：SampleCode 為最佳鍵

八、資料遷移與升級流程

1. 舊系統資料備份
2. 新資料表建立
3. 撰寫 Migration Script
4. CSV 匯入功能（含暫存表與預檢）驗證

九、後續擴充方向

- 快速回應碼（Quick Response Code, QR Code） Code / 條碼（Barcode） 掃描整合
- 樣品生命週期視覺化
