BambiGO 技術架構規格書

版本：v4.0 (Tokyo Commercial MVP)

適用範圍：日本公共交通開放數據挑戰賽 (驗證場域：東京都心三區)

核心戰略：PWA 體驗優先、商業導流變現、情感化節點繼承

---

1. 專案願景與核心指令 (Project Vision & Prime Directive)

1.1 產品定義

BambiGO 是一個基於 **PWA (Progressive Web App)** 的城市感性導航服務。它利用 AI 將開放數據轉譯為具備「人格」的建議，並透過商業服務（計程車、共享運具、置物櫃）的導流來解決交通空白與移動焦慮。我們不只提供導航，而是將冷冰冰的開放數據（**ODPT, GTFS, OSM**）轉譯為具備同理心的行動建議 (Nudge)，以解決交通空白與過度旅遊等社會課題。

1.2 最高指導原則 (Prime Directive for AI Agents)

1. **Guest-First (訪客優先)**：90% 的功能（地圖、查詢、AI 對話、導流）必須在免登入、免註冊的情況下可用。註冊僅用於「Trip Guard 行程守護」。
2. **Commercial Reality (商業現實)**：在東京 MVP 中，**L4** 行動建議必須優先考量「可執行的替代方案」。若公車擁擠，直接提供 Uber/GO 或 LUUP 的 Deep Link，創造導流價值。
3. **Inheritance Efficiency (繼承效率)**：嚴守 **10-15** 個 **Hub** 母節點 限制。所有子節點必須透過演算法繼承母節點人格，確保 MVP 開發資源聚焦。
4. 動態擴充原則：文檔中提及的地點（如上野、淺草）僅為 MVP 驗證範例。系統架構必須支援動態新增任何符合地理圍欄內的節點與設施。
5. 一個建議原則：AI 輸出的決策必須收斂為「單一最佳建議」，消除使用者的決策癱瘓。
6. 多語言與國際化策略:繁體中文為預設系統語系，必須支援 英文 **(en)** 與 日文 **(ja)** 的 UI 切換，其他語系可透過AI自然對話輸入獲得其他語系的回應。

1.3 我們要解決的社會課題

1.過度旅遊 **(Overtourism):**數據驅動的識別。

2.資訊斷層導致的決策癱瘓**:**決策癱瘓、等待焦慮、異常焦慮、資源焦慮、抵達焦慮。

3.多模式整合的缺失**:**使用者需要切換多個 **App**。

---

2. 系統架構 (System Architecture)

採用 **Modern AI Stack**，強調 Serverless 與自動化運維。

2.1 智能核心 (Intelligence Core)

- **Reasoning Engine: Google Gemini 3 Pro** (透過 Dify 調用)。
    - 理由：負責理解複雜語意、檢索非結構化觀光資訊，並生成 L4 行動建議。利用更強大的多模態理解與長文本推理能力，處理複雜的東京交通路網。
- **Orchestration:** Dify (RAG Engine) - 管理 Prompt 與觀光知識庫。
- 開發核心**:** Trae (SOLO Mode) - 採用多智能體 (Multi-Agent) 模式進行全自動代碼生成與架構維護。

2.2 自動化與數據層 **(Automation & Data)**

- **ETL Pipeline:** n8n (Self-Hosted on Zeabur) - 自動化數據抓取中樞。
    - *API Key* 管理：ODPT API Key 儲存於 n8n Credentials，系統定時自動抓取，無須人工下載。
- **Database:** Supabase (PostgreSQL) - 儲存節點與用戶資料。

2.3 前端互動層 **(Interface)**

- **Platform:** Next.js PWA (可安裝至手機主畫面，離線支援)。
- **User State:**
    - **Guest (訪客):** 不強制授權 GPS 定位，可手動選擇節點查詢。
    - **Member (會員):** 授權 GPS定位追蹤 + LINE Login，啟用 Trip Guard 推播。

---

3. 核心邏輯：四層標籤化數據模型

L1：地點基因層 **(Location DNA) -** 骨架

定義：節點的靜態屬性與基礎設施能力。
數據來源：ODPT (車站/巴士站), OSM (景點/設施), MLIT (觀光資源), GBFS (共享運具站)。
通用邏輯：

- 交通節點：自動標記為 Hub (樞紐) 或 Spoke (末端)，依據路線數量權重計算。
- 觀光節點：依據 OSM tag (`tourism=*`) 自動分類（歷史、商業、自然）。
- 範例對應：系統不寫死 "上野站"，而是識別 "具備新幹線與地鐵交會屬性的 Hub"。

母子繼承架構

- **Hub (母節點):** 東京關鍵樞紐（如：上野站、東京站、淺草站、秋葉原站）。
    - 數量限制：MVP 鎖定 10-15 個。
    - 內容：手工撰寫的精細 Persona Prompt（人格、語氣、在地知識）。
- **Spoke (子節點):** 數百個普通站點。
    - 內容：自動繼承最近 Hub 的人格，僅覆蓋 `name` 和 `location`。

L2：即時狀態層 **(Live Status) -** 感知

定義：影響決策的動態變數。
數據來源：ODPT API (`TrainInformation`, `BusLocation`, `Train`), OpenWeather。

通用邏輯：

- 異常偵測：監控 `delay > 15min` 或 `status != Normal`。
- 擁擠度推估：整合巴士即時位置與歷史權重，標記「擁擠/舒適」。

ODPT 直連

- 數據源**:** ODPT API (TrainInformation, BusLocation)。
- 自動化**:** n8n 每 15 分鐘輪詢一次，異常狀態寫入 Redis。

L3：環境機能層 (Micro-Facilities) - 細節

定義：解決旅途中「微需求」的服務設施。
數據來源：OSM (`amenity=toilets`, `amenity=bench`), 爬蟲 (置物櫃狀態), 官方無障礙地圖連結。
通用邏輯：

- 關聯綁定：將設施以地理距離 (Radius 200m) 自動綁定至最近的 L1 母節點。
- 數據結構：`facilities_metadata` JSONB 欄位，支援動態新增類型（如：未來加入 AED 或 哺乳室）。

雙層標籤結構

為了讓 AI 建議更精準，將 L3 拆解為：

1. 供給標籤 **(Supply Tags):** 客觀事實。
    - has_locker (有置物櫃), has_bench (有椅子), has_wifi, has_elevator.
2. 適用標籤 **(Suitability Tags):** 主觀情境（化解焦慮的關鍵）。
    - good_for_waiting (適合久候), work_friendly (適合臨時工作), quiet_zone (安靜避難), luggage_friendly (適合大行李).

L4：行動策略層 (Mobility Strategy) - 決策

定義：AI 綜合 L1-L3 生成的最終建議。
運算核心：Dify RAG + LLM。
通用邏輯：

- 分流策略：當 L2 擁擠度過高 -> 檢索 L1 替代景點 -> 生成導引建議。
- 交通縫合：當 ODPT 顯示無車 -> 檢索 GBFS 共享單車或線上計程車資訊 -> 生成替代路徑。

商業導流

- 東京 **MVP** 特化：MVP不使用 DRT，改為「商業替代運具」。
- **Action Cards:** AI 輸出必須收斂為 **3** 張卡片，一句話行動建議：
    1. 最佳大眾運輸**:** (ODPT 數據) "搭銀座線，3 分鐘後發車"。
    2. 舒適/快速替代**:** (Taxi 導流) "搭 Uber/GO，約 ¥1200，省 10 分鐘"。
    3. 微型移動/體驗**:** (LUUP 導流) "騎共享滑板車，沿途風景好，約 15 分鐘"。

---

4. 關鍵演算法規格 (Executable Specifications)

雖然架構通用，但 Phase 1 驗證鎖定特定區域以集中資源。

4.1 驗證場域定義 (The Sandbox)

- 地理圍欄 **(Geo-fence):**
  - 核心區：台東區 (上野/淺草)、千代田區 (東京車站/皇居)、中央區 (銀座)。
  - **Bounding Box** 設定：`[139.73, 35.65]` 至 `[139.82, 35.74]`。
  - 註：系統應設計為更改 *Bounding Box* 參數即可切換至其他區域。

4.2 數據獲取策略 (Smart Ingestion)

採用 **"ODPT First, OSM Second"** 策略：

1. **Phase 1 (骨幹):** 透過 n8n 呼叫 ODPT API，抓取圍欄內所有 JR東日本、東京Metro、都營地鐵/巴士站點。
2. **Phase 2 (肌肉):** 透過 n8n 呼叫 Overpass API (OSM)，抓取圍欄內的 `amenity=toilets`, `tourism=attraction`, `amenity=locker`。
3. **Phase 3 (神經):** 接入 GBFS API (Docomo Cycle / LUUP)，將微型移動站點疊加至地圖。

4.3 母子節點繼承演算法 (Persona Inheritance)

TypeScript

```
// 偽代碼邏輯
function resolveNodePersona(targetNode: Node): string {
  // 1. 如果自己就是 Hub，直接回傳專屬 Prompt
  if (targetNode.is_hub) {
    return targetNode.persona_prompt;
  }

  // 2. 如果是子節點，尋找「路網距離」最近的 Hub
  // (MVP 簡化：使用地理距離，「同路線優先」邏輯。)
  const nearestHub = findNearestHub(targetNode.location, allHubs);

  // 3. 動態合成 Prompt
  return `
    ${nearestHub.persona_prompt}
    (Context Override: 你現在位於其子站點「${targetNode.name}」，
     請保持原本的人格，但針對此地點的具體設施進行回應。)
  `;
}
```

4.4 ODPT API 自動化抓取流程 (n8n Workflow)

1. 設定：在 n8n 的 Env Vars 中設定 ODPT_API_KEY = "您的長字串Key"。
2. 節點 **1 (HTTP Request):**
    - Method: GET
    - URL: https://api.odpt.org/api/v4/odpt:Station
    - Query Parameter: acl:consumerKey = {{$env.ODPT_API_KEY}}
3. 節點 **2 (Filter):** 篩選 lat/lon 是否在東京 MVP 圍欄內。
4. 節點 **3 (Upsert):** 寫入 Supabase。

- 結果：全自動運行，無需人工下載。

---

5. 核心功能模組與使用者體驗流程 (PWA UX Flow)

核心:所有功能皆為通用設計，不僅限於特定路線或景點。

5.1 City Adapter (城市適配器)

- 目的：實現標準化擴張 (Standardization)。
- 設計：透過 `lib/adapters/{city_id}.ts` 定義該城市的數據源開關 (Feature Flags)。
    - `hasSubway`: boolean
    - `hasSharedMobility`: boolean
    - `odptOperators`: string[] (白名單)
- 應用：東京版啟用 Subway & GBFS；未來切換至飯能版則關閉 Subway 並開啟 DRT。

5.2 Trip Guard (行程守護者)

- 觸發機制：使用者訂閱任意 ODPT 路線（不限於山手線）。
- 運作邏輯：n8n 背景輪詢該路線 L2 狀態 -> 若異常 -> Dify 判讀嚴重性 -> 推播替代方案。
- 替代方案邏輯：動態計算。若鐵路停駛，自動搜尋周邊巴士或共享運具。

5.3 Overtourism Diversion (過度旅遊分流)

- 觸發機制：區域 L2 擁擠度 > 閾值 (e.g. 80%)。
- 運作邏輯：AI 檢索 L1 資料庫中 `vibe=quiet` 且距離 < 1km 的同類型景點。
- 範例（非寫死）：
    - 若淺草寺擁擠 -> 推薦待乳山聖天。
    - 若上野公園擁擠 -> 推薦舊岩崎邸庭園。
    - *(系統依據數據動態配對，而非人工寫死)*

5.4 訪客模式 (Guest Mode) - 預設狀態

- 進入點: 進入PWA網頁，手動選擇節點，或者GPS定位最近的節點。

- 權限: 無需授權 GPS (可手動選點), 無需登入。
- 功能:
    - 查看地圖與 L1/L2 資訊。
    - 與 AI 對話詢問路線。
    - 查看 L4 建議卡片 (含商業導流連結)。
    - 核心價值: 快速解決當下焦慮, 導流至付費服務 (Taxi/Locker/租借行動電源)。

5.5 會員模式 (Member Mode) - 行程守護

- 觸發點: 使用者點擊「🔔 開啟行程守護 (Trip Guard)」。
- 授權:
    - **GPS:** "為了準確監控您的行程..."
    - **LINE Login:** "為了發送緊急通知給您..."
- 功能:
    - 背景監控 ODPT 運行狀態。
    - 主動推播延誤警報與替代路徑。

---

6. 東京 MVP 開發清單 (Implementation Checklist)

Phase 1: 骨幹與自動化 (Trae + n8n)

- [ ] 設定 Zeabur 環境變數 `ODPT_API_KEY`。
- [ ] 使用 Trae 建立 **Hub/Spoke** 資料庫結構 (含 `parent_hub_id` 欄位)。
- [ ] 使用 n8n 建立 **ODPT** 自動抓取 **Workflow** (東京核心三區)。

Phase 2: 感性與 L3 標籤 (Dify + OSM)

- [ ] 定義 10 個核心 Hub (上野, 東京, 銀座, 秋葉原, 淺草...) 並撰寫 Prompt。
- [ ] 實作 **L3 供給/適用 雙欄位** 結構。
- [ ] 抓取 OSM 數據, 自動填入 `Supply Tags` (如 `has_locker`)。

Phase 3: 商業導流與 PWA (Next.js)

- [ ] 實作 PWA Manifest (讓網頁可安裝)。
- [ ] 開發 **L4 Action Cards UI** (三張卡片版面)。
- [ ] 整合 **Deep Links**:
    - Taxi: `https://go.mo-t.com/...` (GO Taxi)
    - Luup: `https://luup.sc/...`
    - Locker: `https://cloak.ecbo.io/...`

---

7. 商業變現邏輯 (Monetization Logic - for MVP)

BambiGO MVP 的商業價值不在於「賣 App」, 而在於「焦慮解法的中介 (Broker of Anxiety Relief)」。

1. 移動導流**:** 情境：電車延誤 (焦慮) -> BambiGO 建議 -> 點擊叫車 (GO Taxi)。
    - 價值：潛在的 *CPA (Cost Per Action)* 分潤。
2. 空手觀光導流**:**
    - 情境：找不到置物櫃 (焦慮) -> BambiGO 建議 -> 預約 Ecbo Cloak。
    - 價值：服務手續費分潤。

8.資料庫設計摘要 (Supabase Schema Snapshot)

請另外參考**BambiGO** 資料庫設計規格文件。
設計原則：正規化核心實體、JSONB 保留擴充彈性、索引優化查詢效能

BambiGO 採用「**ODPT First, Long-term Sustainability First**」的資料使用策略。
即時交通資料必須符合以下原則：

1. 長期可用性優先於資料完整度

2. 即使即時資料不可用，系統仍需提供可行的行動建議

3. 所有降級行為必須對使用者誠實說明

核心原則：
BambiGO 不隱藏資料缺口，而是把「不知道」轉譯為可理解的行動建議。

為降低 **ODPT API** 呼叫頻率並提升系統穩定性，**L2** 即時狀態採用快取策略。

MVP 預設方案（低維運成本）

- 快取儲存位置：**Supabase Table**（或 **KV / Edge Cache**）
- 快取單位：`operator_id + line_id + station_id`
- **TTL**（**Time To Live**）：**20** 分鐘

若未來需要高頻即時反應，可再切換為 **Redis**（不影響上層邏輯）。

BambiGO 遵循「**One Recommendation Principle**」：

- 系統 必須選出 **1** 張 **Primary Action Card**（系統推薦）
- 其餘最多 **2** 張為 **Secondary Cards**（替代或體驗選項）
- **UI** 預設僅強調 **Primary Card**，其餘為可展開選項

9.開發里程碑 (Development Roadmap)

Phase 1: 骨幹建置 (Current Focus)

- 完成 City Adapter 介面實作。
- 透過 ODPT API 建立東京核心三區的交通節點地圖。
- 地圖分層渲染 (Layering) 實作。

Phase 2: 感知與細節

- 接入 OSM 數據，補足公廁、置物櫃、觀光景點資訊。
- 實作 L2 即時狀態顯示 (點擊車站顯示運行情報)。

Phase 3: 決策與神經

- 接入 GBFS 共享運具數據。
- 完成 Dify 知識庫對接，實作 L4 AI 對話建議功能。
- Trip Guard 自動化流程上線。

本文件 *v4.0* 為最終開發規格。請使用此文件作為 *Trae SOLO* 的 *#project_rules.md*，它包含了自動化 *API* 串接與商業導流的明確指令。