

# 專案建置0~1經驗

從零到一的建置經驗分享 - 技術選型、挑戰與解決方案的討論

by Mike Lin

建置經驗分享



# 專案需求-BC

## 01. 平台整合

打造支援多種金流與遊戲商串接的統一平台系統

## 02. 彈性活動

押注返水、首儲回饋、儲值回饋、福利金等多元化活動機制

## 03. 後台對帳

完整的帳務對帳與財務管理功能確保資金安全

# 核心功能-BC

## 用戶管理系統

註冊登入、身份驗證、權限管理

## 活動

活動設置、返水機制、福利處理

## 錢包

資金操作、充值提現、餘額管理



## 遊戲串接

多家遊戲商API串接、遊戲數據同步

## 金流串接

多種支付方式、充值提現、第三方金流串接

## 抄單對帳

跟單功能、策略複製、帳務對帳

# 使用的技術棧

## 後端技術

- Go語言
- 微服務架構

## 資料存儲

- MySQL
- Redis

## 基礎設施

- AWS雲端服務
- Docker容器化
- GitLab CI/CD

# 核心微服務架構

## 會員服務

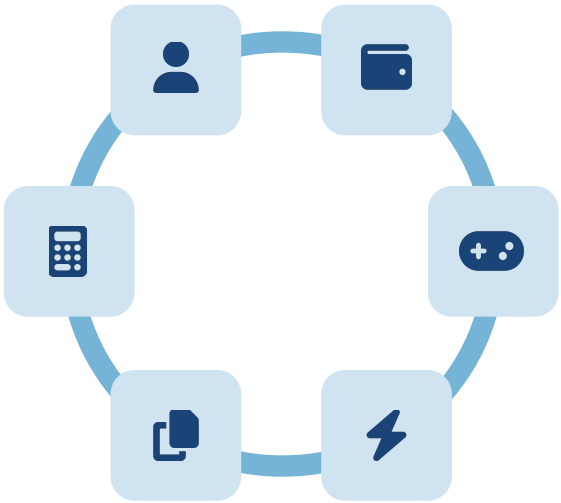
用戶註冊、登入驗  
證、權限管理

## 流水計算

帳務計算、報表生  
成、對帳功能

## 抄單系統

跟單功能、策略複製



## 錢包服務

資金操作、充值提  
現、餘額管理

## 遊戲服務

遊戲邏輯、狀態管  
理、結果計算

## 事件服務

活動設置、返水機  
制、福利處理



# 過程中遇到的挑戰

1

## 遊戲商整合問題

遊戲商可能突然改變API結構導致漏單，需要建立穩定的監控與容錯機制。

2

## 版本管理與部署

頻繁更版需求，自架GitLab Server與CI/CD Pipeline建置挑戰。

3

## 開發順序問題

先做前台再做後台的開發順序造成後期調整困難，應該同步開發。

# 重新開始的優化策略



## 改用Kafka+多Worker架構

流水計算不使用MySQL SP，改用程式搭配Kafka多Worker消耗，提升處理效率。



## 抄單系統優化

一次性寫入原始資料，再用多Worker消耗解析，避免解析失敗導致掉單問題。



## 優先開發後台系統

先建置後台管理功能，減少後期調整範圍，提升開發效率。



## 採用單一錢包模式

使用單一錢包而非多錢包，配合遊戲商轉帳錢包趨勢，簡化架構複雜度。



## 實行MVP開發模式

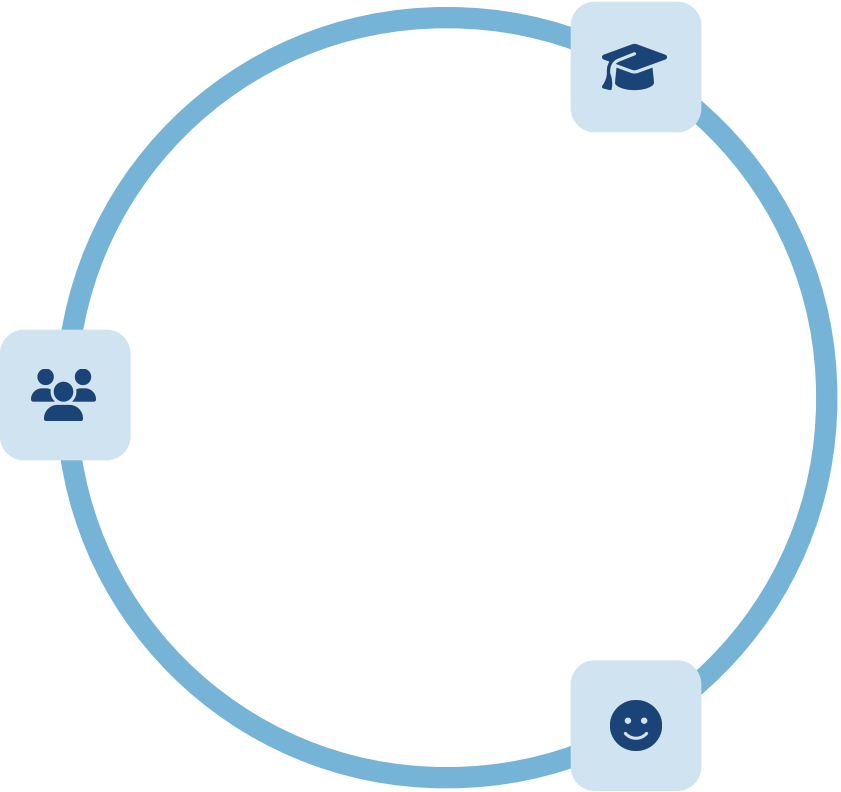
避免過度設計，專注核心功能，減少不必要的複雜度和開發成本。



# 專案成果與經驗

## 團隊協作

建立有效的開發流程、程式碼規範和技術文檔，提升整體開發效率和品質。



## 技術成長

深入掌握分散式系統設計、微服務架構模式、高併發優化和金融級安全實作。

## 穩定運營

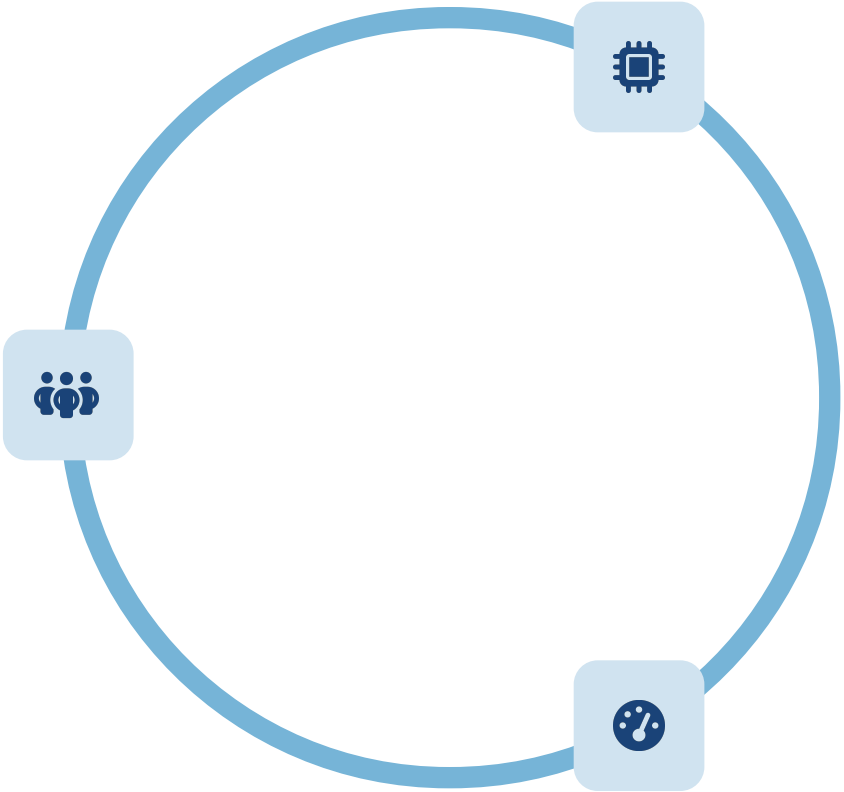
成功建構穩定的微服務架構，實現高併發處理能力，建立完整的資金安全保障機制。



# 專案成果與經驗

## 團隊協作

需求變換快速，溝通成本龐大。



## 技術成長

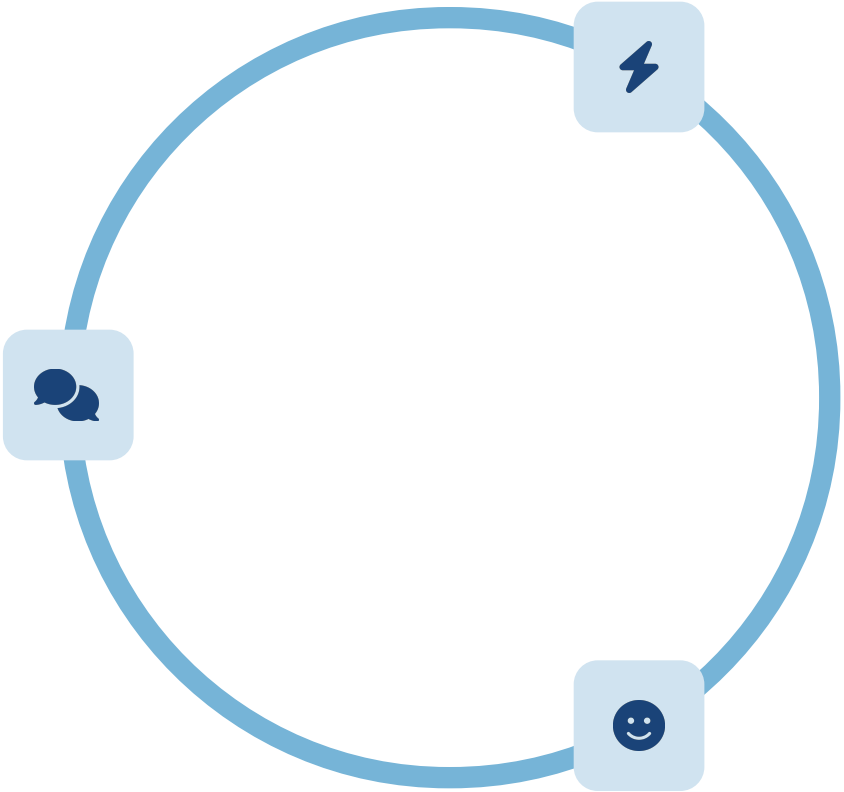
過度設計，多數高效能的設計根本無發會空間。

## 穩定運營

產品穩定前即終止，成本沒有回收。

# 專案成果與經驗

**最小**  
專案最小可行性。



**最快**  
最快速的功能demo產出。

**最簡單**  
使用最簡單的技術棧，降低開發複雜度。

# 專案需求-健身App

做一個可以「訓練前設定計劃、訓練時紀錄、訓練後查看趨勢」的專業健身app。



## 使用的技術棧-健身App



開發語言

Swift



資料存儲

Core Data, Firebase



AI輔助開發

cursor

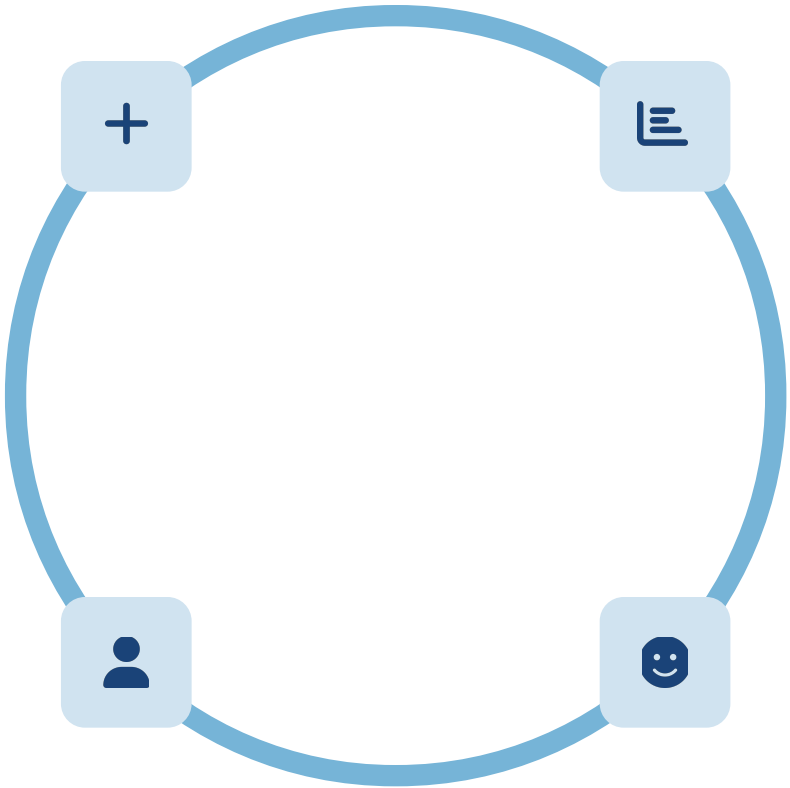
# 核心功能-健身App

## 訓練記錄

新增訓練項目、記錄組數重量、計時功能

## 個人檔案

數據記錄、訓練偏好設定、模板/動作系統



## 數據分析

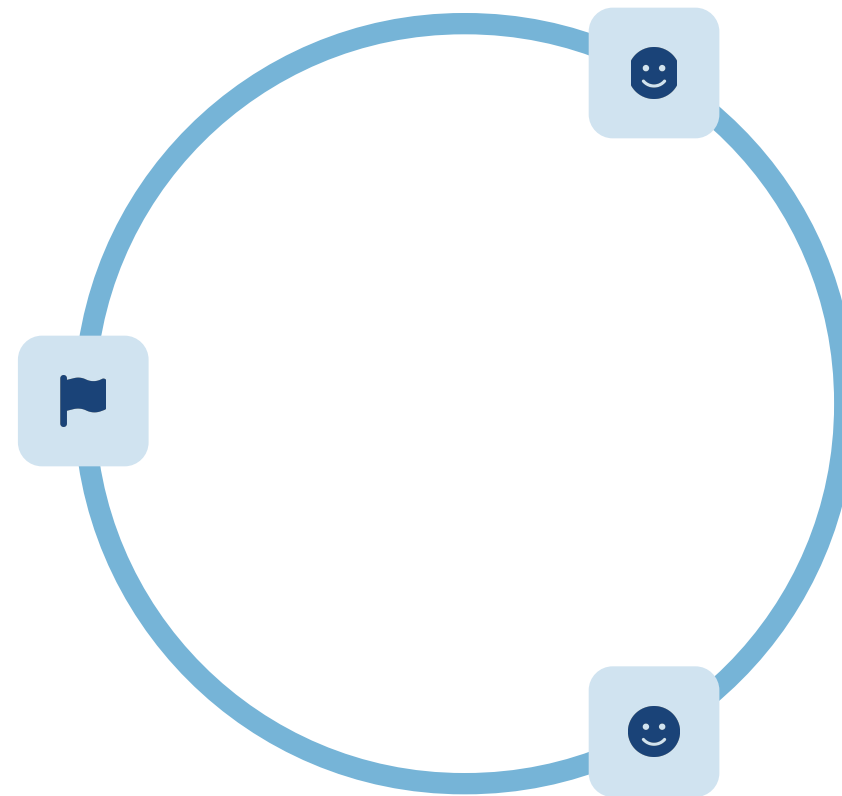
訓練容量圖表、個人記錄統計、成長趨勢

## 計劃排程

週期化訓練計畫、提醒通知、目標設定

# 專案成果與經驗-健身App

**最小**  
只專注在核心需求。



**最快**

總開發用時兩天，並在一週內送審通過並上線。

**最簡單**

使用少的技術棧。

# 感謝聆聽

期待與您進一步交流技術架構設計與團隊管理的寶貴經驗