云碩科技股份有限公司

用 AI 協助學校
做校務系統轉型

不是把舊系統重寫一遍，而是用 AI 幫學校把「架構」與「知識」一起重建。

重建　留存　有感

AGENDA　本場地圖

今天談三件事

01

學校共同的痛

廠商能量有限、架構老舊、人才難找、知識鎖在人身上

02

我們的解法

不是重寫，而是「重建架構 + 留存知識」，而且要讓使用者有感

03

兩個已在跑的 POC

考勤與學費計算，兩種舊語言，證明做法可複製

SECTION 1

為什麼是現在？

有些事，不是我們變聰明了，是工具終於成熟了

1

為什麼是現在 · 1 / 2

過去做不到的事，這一兩年突然可行

重構老系統

把二十年的老程式一塊一塊搬新家，過去成本高到不敢想，現在做得到了

留存開發知識

讓 AI 把散落的開發紀錄整理成「懂系統的助手」，知識不再只鎖在某個人腦中

對話式服務

使用者用「講的」就能辦事，不必再背選單、記流程

我們的起點不是技術，是「聽到的痛」

> 長期跟很多學校交流，我們反覆聽到主任、同仁抱怨同一批問題。

久了，就動了一個念頭

與其每次都聽別人喊痛，
不如做一套「可複製」的解法。

一次一次幫忙止痛，不如把方法整理成一套流程，讓它可以在不同學校重複使用。這也是今天兩個 POC 想證明的事。

SECTION 2

學校共同的痛點

不是某一間學校的問題，是幾乎每一間的問題

2

學校共同的痛點

反覆聽到的四個問題

回應很慢，只能等

系統長年委由同一家廠商，廠商能量有限，需求排不進去，學校只能排隊等。

只能一直往上疊

架構是二十年前的設計，整套重寫費用、時間、風險都太高，只好在舊系統上不斷疊加。

老技術，人愈來愈少

PowerBuilder、Visual Basic 這類老技術，會的人愈來愈少，維護愈來愈貴。

知識鎖在人身上

關鍵幾個人一離開，系統為什麼這樣設計就沒人知道了。

最致命的痛

知識，鎖在人身上

關鍵的那幾個人一離開，「系統為什麼這樣設計」就再也沒人說得清楚。

> 今天還在會議上討論的問題，二十年後很可能還在討論。

SECTION 3

我們的解法：不是重寫，是「重建 + 留存」

全場核心——重建架構、留存知識，讓系統真的有感

3

破除迷思：轉型 ≠ 打掉重練

常見的誤會

「轉型」= 把整套系統打掉、重寫一遍

聽起來乾脆，但費用、時間、風險都太高——這正是學校二十年不敢動的原因。

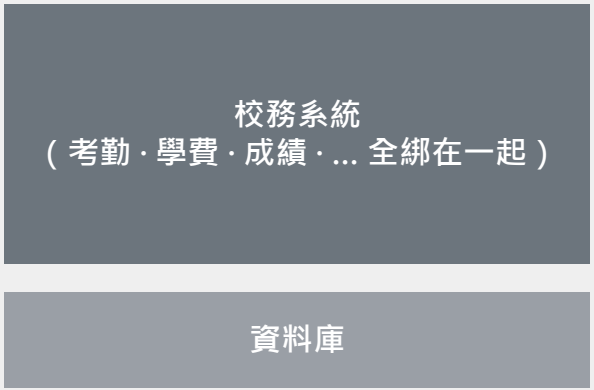
我們的做法

漸進重構：舊系統照跑，一塊一塊搬

學校的服務一天都不能停。我們不關掉舊系統，而是像換房子一樣，一個房間一個房間慢慢搬，風險最小。

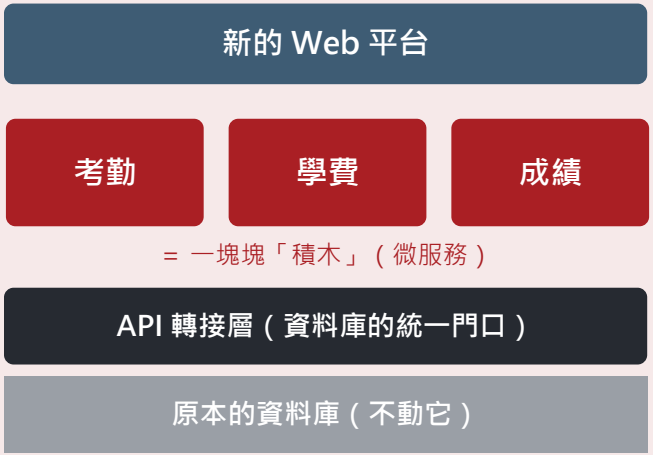
舊架構 vs 新架構（一張圖看懂）

舊架構：一塊大鐵板



動一個地方，整塊都要跟著動

新架構：一塊塊積木 + 一層 API



要換哪塊、就換哪塊，其他照跑

我們的解法 · 三根支柱

重建、留存、有感——缺一不可

重建

架構重建

資料庫外包一層 API，功能拆成積木，再組成新的 Web 平台。

留存

知識留存

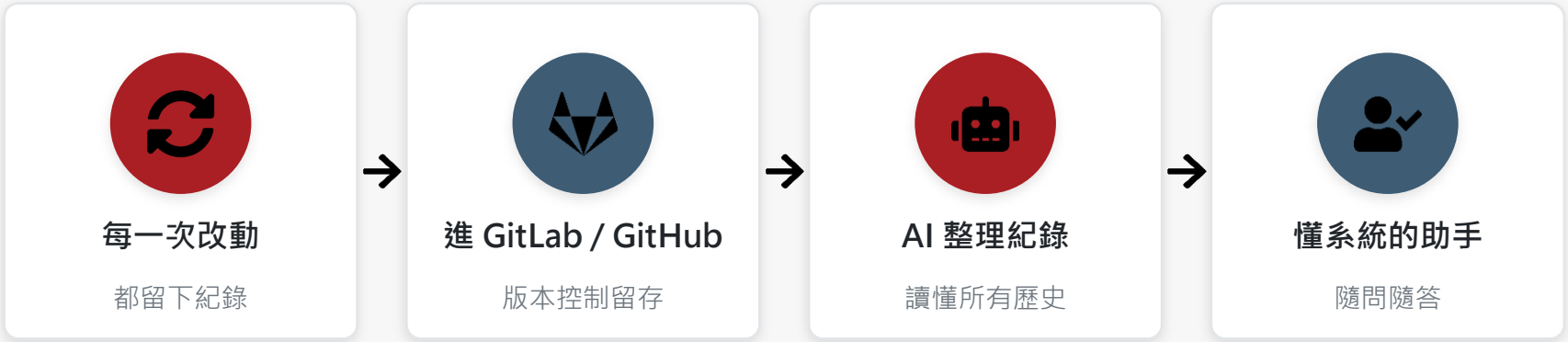
開發全程進 GitLab / GitHub，再讓 AI 整理成「懂系統的助手」。

有感

使用者有感

導入 AI Agent，讓系統不再只是系統，而是能對話、會替你辦事的助手。

讓知識由大家一步步累積，不再流失



資料要留在校內？ 可用自架 GitLab；需要雲端協作時用 GitHub。版本控制與知識留存，兩種都支援。

讓系統不再只是系統，而是會辦事的助手

過去：你去適應系統

- 登入、找選單、一格一格填表
- 要記得流程、記得規則
- 你不問，它不動——系統是被動的

現在：系統來適應你

- 用「講的」就能辦事
- AI Agent 替你把事情辦掉
- 隨問隨答，像個懂系統的同事

系統變成會辦事的助手，使用者才真的有感——這才是學校願意轉型的理由。

SECTION 4

POC 案例一：從考勤切入

4

匿名案例 · 呼應關鍵字「留存」

痛點與做法

痛點

- 校務系統用 PowerBuilder 寫成
- 廠商能量有限、回應很慢
- 想動又不敢動——怕一改就壞

做法

- 把「考勤」當作第一塊積木
- PowerBuilder → 逐步重構
- 全程用 GitLab / GitHub 留紀錄
- 舊系統照跑，風險最小

亮點：知識留下來了，還能對話

每一步都被記錄

- 未來交接、擴充都有跡可循
- 不再依賴「某個人的記憶」
- 這正是對「知識鎖在人身上」的解答

再往前一步：對話式請假

使用者用「講的」就能請假。

跟過去只用向量做的 RAG 不一樣：
它不只是「查得到」，而是能對話、
能真的幫你把事情辦掉。

SECTION 5

POC 案例二：從學費計算切入

匿名案例 · 呼應關鍵字「有感」

痛點、做法、亮點

痛點

- 用 Visual Basic 老程式碼
- 改不動，只能在舊架構上一直疊

做法

- 讓 AI 先跑完原本 VB 的計算邏輯
- 把規則吃透，再復刻成新網站
- = 用 AI 把舊資產「翻譯」成新架構

亮點：有感

- 新網站不只是換皮
- 裡面放了 AI Agent
- 讓使用者真的覺得比以前方便

兩種舊語言，同一套做法

案例一

PowerBuilder

考勤 → 對話式請假

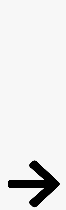
呼應「留存」

案例二

Visual Basic

學費計算 → 新網站

呼應「有感」



同一套做法，可複製

兩種不同的舊語言都成立，證明這不是單一個案的巧合。

SECTION 6

收尾：留給二十年後的禮物

回扣開場的那句話

6

> 今天還在討論的問題，之所以拖了二十年，是因為知識沒有被留下來。

當架構被重建、知識被 GitLab / GitHub + AI 一步步記錄下來，未來接手的人，不必再從零猜起。

我們做的不只是重寫系統，

是替學校蓋一座「會自己長知識的房子」。

Q & A

與下一步

從一個小範圍 POC 開始

像考勤、或單一計算模組——花小成本先看到具體成果，再決定要不要往下走。歡迎會後找我們聊聊。