微服務建構方法 – 以線上版 Kanban 桌遊為例

專題編號：111-CSIE-S023
執行期限：110 年第 1 學期至 111 年第 1 學期
指導教授：鄭有進
專題參與人員： 108820001 羅羽軒
108820015 周雨柔
108820016 郭梓琳

一、摘要

本專題以現有 ezKanban 系統為基礎，開發進階功能 Kanban 桌遊線上版，探討微服務的建構方法。運用 **Domain-Driven Design (DDD)** 作為核心方法論、**Clean Architecture** 使軟體易於維護與擴增，透過搭配 **Event Storming** 釐清產品需求，並使用 **Test-Driven Development (TDD)** 作為開發模式，以及 **Jenkins**、**Docker** 作為持續整合工具，提升開發品質與效率。

二、緣由與目的

為降低軟體開發之修改成本，本專題以微服務設計為基礎，使系統專注於責任與功能的區分及有快速應變需求的能力。

開發此線上版桌遊的緣由為能使修習北科「敏捷與精實軟體開發」課程的學生能打破距離與病毒的隔閡，可不需透過 The getKanban Board Game 實體桌遊也能學習 Kanban 此一視覺化敏捷開發流程的工具。

三、使用技術及工具

(一) 整潔架構 Clean Architecture
整潔架構是一種階層式軟體架構。有使問題領域核心及應用業務邏輯獨立於框架、UI、DB、可測試等之特性。

(二) 領域驅動設計 Domain-Driven Design (DDD)
領域驅動設計是一種軟體的設計方法。透過開發人員與領域專家合作建立出該問題領域的領域模型來釐清系統目標功能，並依此模型開發。

(三) 事件風暴 Event Storming
事件風暴是一種工作坊形式的活動，透過跨部門、跨領域的成員協作探索複雜的問題領域。

(四) 測試驅動開發 Test-Driven Development (TDD)
測試驅動開發，是一種軟體開發方式。此方法的基本理念是先寫測試再撰寫程式以實踐功能。

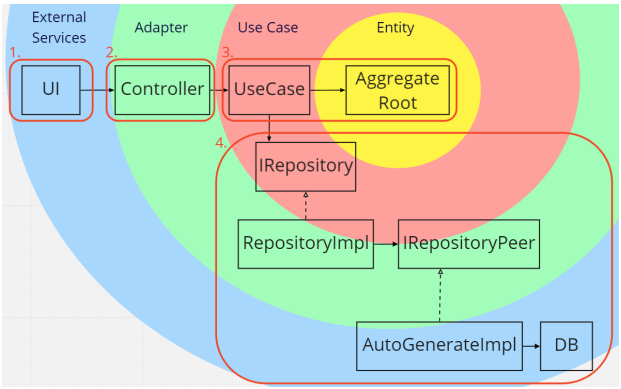
(五) Docker
Docker 是一個建立並運行虛擬環境於容器中的開源工具，能夠用來建立及部署應用程式。

(六) Jenkins
Jenkins 是一種持續整合工具，可以幫助使用者達成專案建置、測試及部署等階段自動化的目標。

四、架構流程

我們使用 DDD 中的 Bounded Context 將後端分成使用者帳戶管理、團隊管 理與看板管理三個微服務系統，並利用 Postgres Message Store 中事件儲存與取回的機制，使三個微服務系統進行狀態同步。

使用者透過網頁送出需求至後端改變狀態。流程可分為將事件存入資料庫與從資料庫取出事件兩個部分。第一部份以 Clean Architecture 呈現使用者從送出需求至將新事件存入資料庫的流程:



圖一、專案架構圖(步驟一~四)

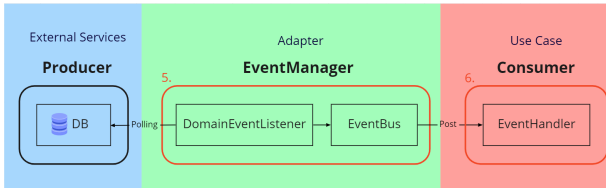
(一) 使用者透過瀏覽器送出需求。

(二) 後端 Controller 接收需求並轉呼叫 Use Case。

(三) Use Case 改變物件狀態。

(四) 物件狀態及事件儲存至資料庫。

第二部份以 Clean Architecture 呈現事件從資料庫拿出後之事件驅動流程：



圖二、事件驅動流程架構(步驟五~六)

(五) Event Listener 輪詢資料庫並取回事件

(六) 有訂閱此事件之 Event Handler 再處理後續操作。

藉由上述流程，不同系統之 Event Handle 可各自訂閱需要相應變化之事件，以此達到狀態同步。

五、實驗結果

本專案目前已部署上線並提供北科「敏捷與精實軟體開發」課程使用。

(一) 微服務架構

在實作微服務架構開發的過程中，我們以 DDD 作為切割方法，並以事件驅動的方式使各微服務維持狀態同步，以實現各微服務間的獨立。

(二) Kanban 敏捷開發視覺化工具

最終實作之遊戲功能包含：卡片操作、骰子操作、多人協作。並可於遊戲結束後，產生遊戲期間的相應報表：Lead Time Distribution Chart、Control Chart、Cumulative Flow Diagram、Financial Report。

六、結論

藉由 Kanban 桌遊線上版的開發，本團隊已建立一套標準化的開發方式，利用上述所列方法與工具使系統保持彈性，從設計、開發、自動化建置等層面降低修改成本，即使尚未將產品拆分部署，也已體會使用微服務架構所帶來的靈活性與可維護性。

本專題尚未將系統分開部署，未來隨著專案架構的擴增，可將子系統獨立建置，並使用 kubernetes 自動化部署及管理所有子系統，實現微服務之完整架構。

七、參考文獻

[1] getKanban, “The getKanban Board Game,” 2022. [Online].
Available: https://getkanban.com/.
[2] E. Evans, “Domain-driven design: tackling complexity in the heart of software,” Addison-Wesley Professional, 2004.
[3] R. C. Martin, J. Grenning, and S. Brown, “Clean architecture: a craftsman's guide to software structure and design,” Prentice Hall, 2018.
[4] K. Beck, “Test-driven development: by example,” Addison-Wesley Professional, 2003.
[5] B. PostgreSQL, “PostgreSQL,” 1996. [Online].
Available: http://www.PostgreSQL.org/about .