Kill Bill Project

สมาชิกกลุ่มที่ 6

62010029	นางสาวกฤติกามาส	สุโนภักดิ์
62010090	นางสาวคัมภีรดา	ภู่ทอง
62010175	นายชวกร	เหลาแก้ว
62010215	นางสาวโชติกา	ตระกูลนุช
62010285	นางสาวณัฐมน	บุญประเสริฐ
62010292	นายณัฐวัฒน์	จันทร์วิสิทธิ์
62011019	นายอภิรักษ์	อุลิศ

เสนอ

ดร.ปริญญา เอกปริญญา

รายงานนี้เป็นส่วนหนึ่งของวิชา สถาปัตยกรรมและการออกแบบซอฟต์แวร์ (SOFTWARE ARCHITECTURE AND DESIGN) รหัสวิชา 01076024

ภาคเรียนที่ 2 ปีการศึกษา 2564

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Kill Bill

Kill Bill เป็นแพลตฟอร์มการเรียกเก็บเงินและการชำระเงินแบบสมัครสมาชิกโอเพ่นซอร์สชั้นนำในช่วง 10 ปีที่ผ่านมา ซึ่งแพลตฟอร์มนี้มีไว้เพื่อช่วยคุณปรับขนาดโครงสร้างพื้นฐานการเรียกเก็บเงิน และการชำระเงิน และทำให้ธุรกิจของคุณเติบโตและยังสามารถเข้าถึงการวิเคราะห์แบบเรียลไทม์และรายงานทางการเงิน

GitHub LINK : KillBill GitHub

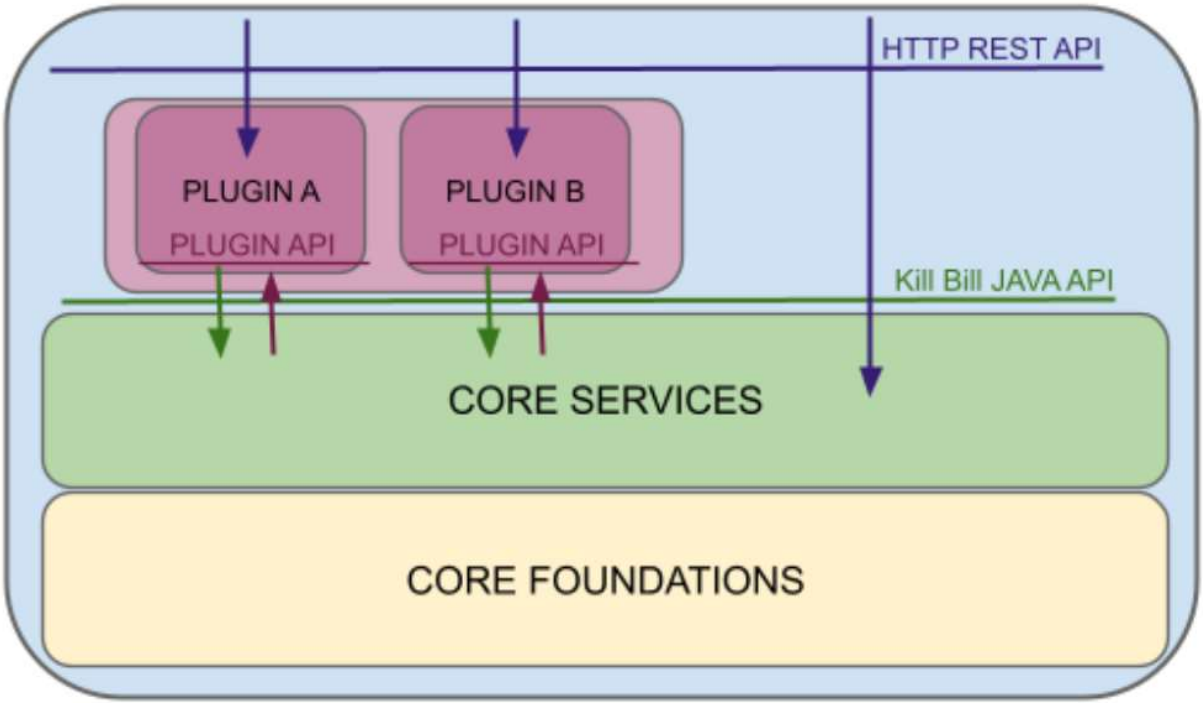
Document : What Is Kill Bill?

DEMO LINK : Kaui (killbill.io)

Architecture

ในส่วนของตัวระบบ Kill Bill นั้นมี Software Architecture Styles คือ Microkernel Architecture หรือที่เรียกอีกชื่อว่า Plugin Architecture เป็น Architecture ที่แบ่งโครงสร้างเป็น 2 ส่วนหลักที่สำคัญ คือ Core System และ Plug-in Modules หรือ Extensions

ซึ่งใน logic ของตัวระบบ Kill Bill ถูกแบ่งระหว่าง plug-in components ที่เป็นอิสระต่อกันและ core system ที่ให้ความสามารถในการปรับขยาย การปรับปรุงแก้ไข เพิ่มเติมและการแยก logic การประมวลผลที่สามารถกำหนดเองได้ดังรูปที่แสดง Topology พื้นฐาน ของ Microkernel Architecture

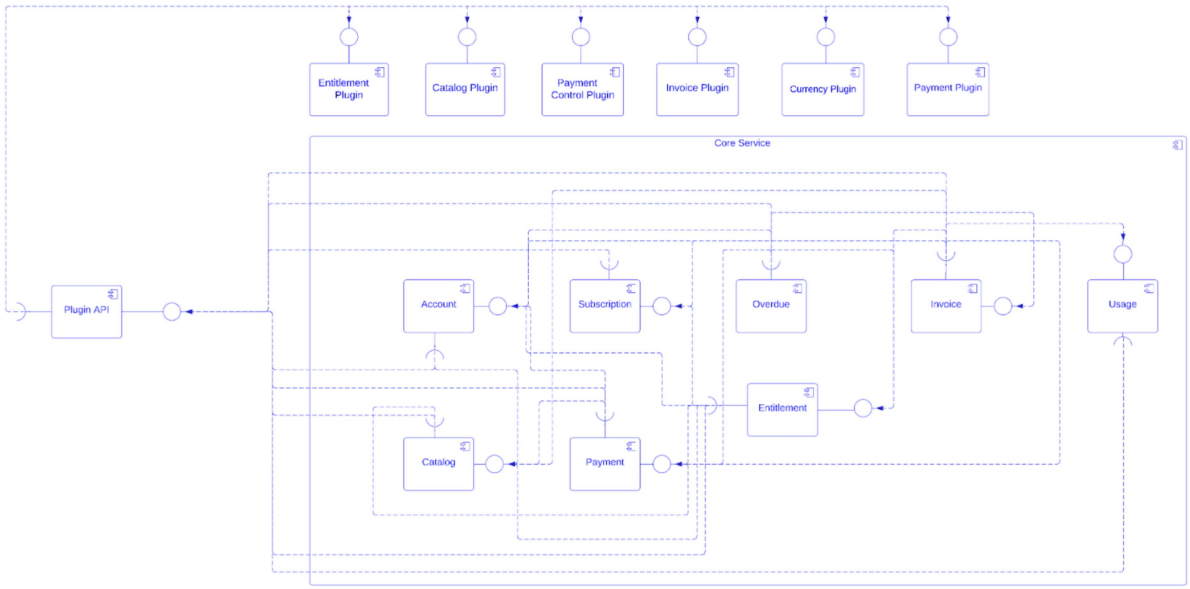


Plugin ของ Kill Bill นั้นเป็นไปตาม OSGI Standard โดยตัวระบบจะใช้วิธีการเขียนโค้ดแยกบนแพลตฟอร์ม Kill Bill และโต้ตอบกับระบบด้วยวิธีต่าง ๆ ได้แก่

- สามารถเรียกได้จากแพลตฟอร์ม Kill Bill ผ่าน Plugin API
- สามารถรับ bus events จากแพลตฟอร์ม Kill Bill
- สามารถเรียก API ไปที่ Kill Bill ได้โดยตรง

อ้างอิง : Microkernel Architecture
อ้างอิง : Overview Kill Bill

UML Diagram ของ Architecture (Component Diagram)

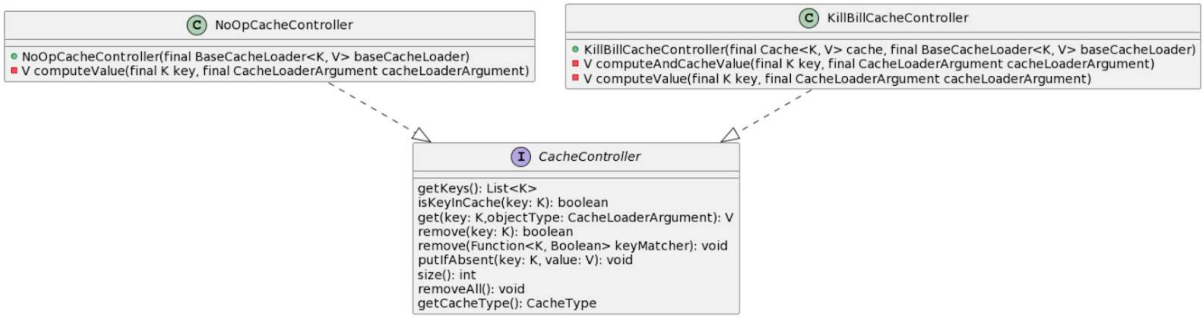


อ้างอิง : UML diagram

Design Pattern

1. Null Object

ใน KillBill จะมีการทำงานของโค้ดในส่วนของ CacheController ซึ่งจะมีหรือไม่มีก็ได้ แต่การจะทำแบบนั้นจำเป็นต้องใช้ Null Object ซึ่งกรณีที่ไม่มีจะสร้าง NoOpCacheController ส่วนกรณีที่มีการเรียกใช้งานก็จะสร้าง OpCacheController ซึ่งการที่จะสร้างคลาสแยกกันออกมาว่าคลาสนึงคือมีและอีกคลาสไม่มี คือต้องการให้สามารถตรวจสอบหรือเรียก method ที่สามารถทำงานได้ แม้จะไม่มีตัว Object นั้นอยู่ก็ตาม ตัวอย่างใน KillBill ที่ยังต้องการให้ Null Object มี Behavior ที่ยังทำงานได้อย่างเช่น computeValue method ซึ่งทั้ง NoOpCacheController และ OpChacheController ยังต้องคำนวณ value จาก key และ cacheLoaderArgument ได้เหมือนกัน

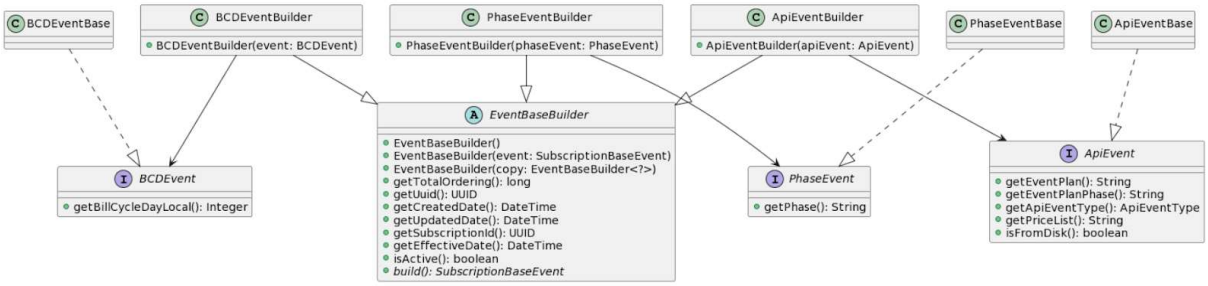


Null Object ที่สามารถพบได้ในโปรเจคนี้อยู่ในไฟล์ Source Code ต่อไปนี้

- CacheController.java
- NoOpCacheController.java
- KillBillCacheController.java

2. Builder

เนื่องจาก Killbill เป็นแอพลิเคชั่นที่ใหญ่และมีระบบที่ซับซ้อน จึงจะเห็นได้ว่ามีการใช้ Builder ภายในโปรเจคค่อนข้างเยอะ เพราะในการสร้าง object ที่มีความซับซ้อนหรือมีหลายขั้นตอน Builder จะทำการแยก object ออกจาก class ไปสร้าง object แยกที่อาจจะเปลี่ยนคุณลักษณะบางอย่าง ช่วยให้ไม่ต้องใส่ parameter เยอะๆ จากตัวอย่างที่ยกมา จะเห็นว่า BCDEventBuilder จะเป็น object หลัก ในการสร้าง BCDEventBuilder, PhaseEventBuilder.java และ ApiEventBuilder

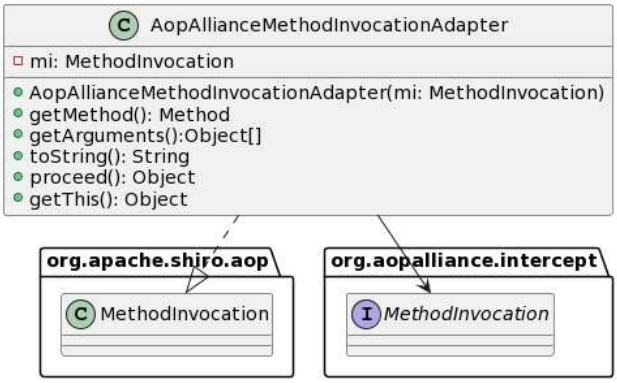


Builder ที่สามารถพบได้ในโปรเจคนี้อยู่ในไฟล์ Source Code ต่อไปนี้

- BCDEventBuilder.java
- BCDEvent.java
- PhaseEventBuilder.java
- EventBaseBuilder.java
- ApiEventBuilder.java
- PhaseEvent.java
- ApiEventBase.java
- ApiEvent.java

3. Adapter

ในส่วนของโค้ดการทำงานของ KillBill ได้มีการใช้ Library ที่ import จากข้างนอก ซึ่งในการที่จะทำให้ โค้ดทำงานกับเซอร์วิสข้างนอกที่มีเงื่อนไขหรือฐานที่ต่างกันได้ ควรใช้ Adapter Design Pattern ที่มี Adapter ในการเชื่อมต่อระหว่างสองส่วนนี้ให้ทำงานเข้าด้วยกัน



Adapter ที่สามารถพบได้ในโปรเจคนี้อยู่ในไฟล์ Source Code ต่อไปนี้

- AopAllianceMethodinvocationAdapter.java

4. Singleton

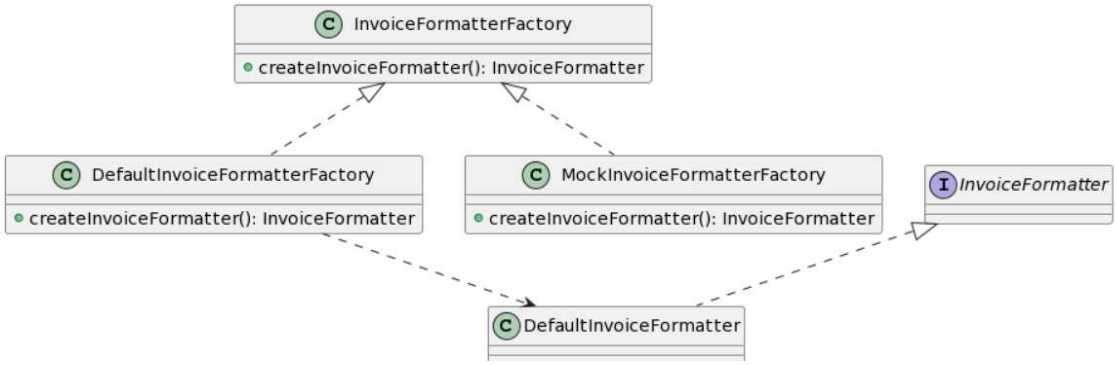
ในส่วนของโค้ดนี้เป็นลดการใช้โค้ดซ้ำซ้อนและสามารถแก้ไขปรับปรุงได้ง่าย โดยข้างต้น class SubscriptionEventOrdering ได้เรียกใช้ method เดิม

Singleton ที่สามารถพบได้ในโปรเจคนี้อยู่ในไฟล์ Source Code ต่อไปนี้

- SubscriptionEventOrdering.java

5. Abstract Factory

เป็นการใช้ method ที่มีชื่อเดียวกัน แต่มีการทำงานฟังก์ชันในการส่งค่าต่างกัน ตามวัตถุประสงค์ของผู้สร้าง



Abstract Factory ที่สามารถพบได้ในโปรเจคนี้อยู่ใน Source Code ต่อไปนี้

- InvoiceFormatterFactory.java
- MockInvoiceFormatterFactory.java
- DefaultInvoiceFormatterFactory.java
- DefaultInvoiceFormatter.java

Quality Attributes

1. จุดเด่น

1.1 Portability

เนื่องจาก KillBill เป็นรูปแบบที่เป็น Microkernel Architecture ทำให้ระบบสามารถทำงานได้ในสภาพแวดล้อมที่หลากหลายโดยทำการเพิ่ม plug-in เข้าไปที่ระบบ เช่น ระบบ Payment ที่มีสามารถเพิ่ม plug-in ในส่วนของ tripe Braintree PayPal หรือ Adyen ทำการเก็บค่าบริการเกิดความหลากหลายทั้งจากบัตรเครดิต เดบิต ACH การโอนเงินผ่านธนาคาร หรือแม้แต่ Bitcoin

อ้างอิง : https://docs.killbill.io/latest/features.html#_payment_plugins

1.2 Security

KillBill เป็นระบบที่มีการจัดการสิทธิ์การเข้าใช้งาน ซึ่งออกแบบตาม Role-based access control (RBAC) ซึ่งเป็นการควบคุมการเข้าใช้งานในส่วนต่าง ๆ ของ application ที่เราได้สร้างขึ้น โดยเมื่อมีการเรียกใช้งานในแต่ละส่วน RBAC จะคอยเช็คผู้ใช้งานแต่ละคนว่า มีสิทธิ์ที่จะเข้าใช้งานในส่วนนี้หรือไม่ หากมีก็จะยอมให้เข้าใช้งาน แต่ถ้าหากไม่มีก็จะทำการแสดง error forbidden ออกไป เพื่อให้ผู้ใช้งานทราบว่าไม่สามารถเข้าใช้งานในส่วนนี้ได้

อ้างอิง : https://docs.killbill.io/latest/internal_design.html#_rbac
อ้างอิง : https://Standardized_levels

1.3 Testability

KillBill ทำการแบ่งระบบออกเป็นส่วนย่อย หลาย ๆ ส่วนทำงานแยกจากกัน ทำให้เราสามารถทดสอบการทำงานได้อย่างง่ายได้ด้วยการเรียกใช้งานแต่ละส่วนที่ต้องการเพื่อการทำtesting หรืออาจทำการ testing e2e

อ้างอิง :
https://github.com/Apizz789/Kill_Bill_Project_SoftArch/blob/main/killbill-master/account/src/test/java/org/killbill/billing/account/AccountTestSuiteNoDB.java

1.4 Scalability

โดยทั่วไปแล้วการใช้รูปแบบที่เป็น Microkernel Architecture จะมีความโดดเด่นในการเพิ่ม plugin ได้ โดยไม่กระทบหรือแก้ไขต่อระบบการทำงานของแต่ละส่วน

อ้างอิง : https://www.developer.com/design/microservices-scalability-challenges/

2. จุดด้อย

2.1 Performance

การใช้รูปแบบที่เป็น Microkernel Architecture นั้น clients ไม่สามารถติดต่อ services ได้โดยตรง ทำให้เกิด overhead ในการติดต่อส่วนกลาง (core services) ก่อน ซึ่งจะส่งผลกระทบให้เวลาในการทำงานเพิ่มขึ้น

โดยวิธีแก้ไขนั้น อาจจะเพิ่มขนาด core services ให้มากขึ้น เพื่อให้ มีการประมวลผลได้เร็วขึ้น

อ้างอิง :
https://www.techstarthailand.com/blog/detail/Software-Architecture-Patterns-5-minutes-read/1825

https://th.gadget-info.com/difference-between-microkernel